International Journal of Forecasting 40 (2024) 1507–1520

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Instance-based meta-learning for conditionally dependent univariate multi-step forecasting[☆]

Vitor Cerqueira [a,*], Luis Torgo [a], Gianluca Bontempi [b]

[a] *Faculty of Computer Science, Dalhousie University, 6050 University Ave, Halifax, NS B3H 1W5, Canada*
[b] *Machine Learning Group, Département d'Informatique, Université Libre de Bruxelles, Belgium*

ARTICLE INFO

Keywords:
Time series
Multi-step forecasting
k-nearest neighbors
Meta-learning
Gradient Boosting

ABSTRACT

Multi-step prediction is a key challenge in univariate forecasting. However, forecasting accuracy decreases as predictions are made further into the future. This is caused by the decreasing predictability and the error propagation along the horizon. In this paper, we propose a novel method called `Forecasted Trajectory Neighbors` (`FTN`) for multi-step forecasting with univariate time series. `FTN` is a meta-learning strategy that can be integrated with any state-of-the-art multi-step forecasting approach. It works by using training observations to correct the errors made during multiple predictions. This is accomplished by retrieving the nearest neighbors of the multi-step forecasts and averaging these for prediction. The motivation is to introduce, in a lightweight manner, a conditional dependent constraint across the forecasting horizons. Such a constraint, not always taken into account by most strategies, can be considered as a sort of regularization element. We carried out extensive experiments using 7795 time series from different application domains. We found that our method improves the performance of several state-of-the-art multi-step forecasting methods. An implementation of the proposed method is publicly available online, and the experiments are reproducible.

Crown Copyright © 2024 Published by Elsevier B.V. on behalf of International Institute of Forecasters. All rights reserved.

1. Introduction

Time series forecasting is a relevant problem across many application domains. Usually, forecasting is defined as a one-step-ahead prediction problem, i.e. predicting the value of the next observation. However, predicting multiple instances ahead yields important practical benefits. It reduces the long-term uncertainty of time series, thereby enabling better operations planning. This task is known as multi-step-ahead forecasting, and several works have been devoted to this problem (Chen, Yang, & Hafner, 2004; Guo, Bai, & An, 1999; Taieb, Bontempi, Atiya, & Sorjamaa, 2012; Venkatraman, Hebert, & Bagnell, 2015).

Multi-step forecasting is a challenging task for two main reasons: the uncertainty associated with prediction targets that are far into the future, and the possibility of error propagation along the horizons. Consequently, forecasting accuracy decreases over the forecasting horizon. This issue is illustrated in Fig. 1, showing the performance of a forecasting model across the horizon (up to 18 observations). The values represent the percentage difference in average error relative to the error incurred in the first prediction step ($t+1$). Overall, there is a clear tendency for the error to increase along the horizon. We provide further details on this figure in Section 4.

While several methods exist, multi-step forecasting is still an open challenge (Bontempi, Ben Taieb, & Le Borgne, 2013). The most intuitive method to tackle multi-step-ahead forecasting is called `recursive`. This approach involves fitting a model for one-step-ahead prediction and

☆ The preparation of this paper has been overshadowed by the passing of Luis in April 2023. Several ideas were worked out together and the co-authors have done their own best to complete them. In sorrow, we dedicate this work to his memory.

* Corresponding author.
E-mail addresses: vcerqueira@fe.up.pt (V. Cerqueira), luis.torgo@dal.ca (L. Torgo), gianluca.bontempi@ulb.be (G. Bontempi).

https://doi.org/10.1016/j.ijforecast.2023.12.010
0169-2070/Crown Copyright © 2024 Published by Elsevier B.V. on behalf of International Institute of Forecasters. All rights reserved.

iterating it with its previous outputs to obtain predictions for multiple steps. The main criticism of this approach is the propagation of errors along the horizon. A common alternative to the recursive strategy is a `direct` approach, where an individual model is created for each horizon. Yet, these methods violate the conditional dependence assumption among different horizons. In other words, each step ahead is forecast in isolation, and the relatedness among different horizons (e.g. due to the autocorrelation between consecutive steps) is not considered. As discussed in previous works (Bontempi & Taieb, 2011; Taieb & Atiya, 2015), this introduces a bias in the predictor and may lead to a sub-optimal forecasting accuracy.

This paper proposes a novel method for multi-step univariate time series forecasting called `Forecasted Trajectory Neighbors` (`FTN`) that regularizes a forecasting model by enforcing a conditional dependency constraint. An observed trajectory refers to the embedding of a time series in a state space (Bontempi & Birattari, 2000), while a forecasted trajectory refers to the multi-step forecasts at a given instant (McNames, 1998). `FTN` is a meta-learning algorithm that can be integrated with any state-of-the-art multi-step-ahead forecasting approach. It works by finding the nearest training trajectories of the predicted trajectory. Then, these training trajectories are averaged for the final predictions. `FTN` addresses the two problems outlined above:

- Conditional dependency: Many state-of-the-art strategies for multi-step-ahead forecasting do not take into account the relatedness or dependency among different forecasting horizons. The proposed method aims at enforcing the constraint of conditional dependency in a lightweight manner.
- Compounding errors: Multi-step predictions may be erroneous due to error propagation. The multi-step predictions resulting from `FTN` rely on past trajectories (i.e. in the training data), reducing the issue of error propagation.

We carried out extensive experiments using a set of 7795 time series. The results suggest that `FTN` leads to systematic improvements in forecasting accuracy. Moreover, `FTN` is particularly adequate for large forecasting horizons. In summary, the contributions of this work are the following:

- A novel instance-based meta-learning method for multi-step forecasting with univariate time series is proposed that accounts for conditional dependency using a different strategy from the one proposed in articles like (Bontempi & Taieb, 2011; Taieb et al., 2012; Taieb, Sorjamaa, & Bontempi, 2010);
- A set of extensive experiments compare `FTN` with state-of-the-art approaches for multi-step forecasting.

`FTN` is publicly available in an online repository.[1] This repository also contains the code for reproducing the experiments.

[1] https://github.com/vcerqueira/ftn

This paper is organized as follows. In Section 2, we describe the background of this paper. We define the problem of multi-step forecasting and we overview state-of-the-art approaches. In Section 3, we formalize the proposed method, highlighting our contributions. Then, we validate the proposed method in Section 4 with extensive experiments. We discuss the results of the paper in Section 5. Conclusions and future directions are discussed in Section 6.

2. Background

In this section, we first formalize the problem of time series forecasting, focusing on an auto-regressive formulation (Section 2.1). Then, we overview the state-of-the-art methods for multi-step-ahead forecasting (Section 2.2), and we highlight our contributions relative to the current literature. Finally, we describe the nearest-neighbors algorithm (Section 2.3). This method is a common approach for multi-step forecasting and a central method to our work.

2.1. Time series forecasting

A univariate time series is a temporal sequence of values $Y = \{y_1, y_2, \dots, y_t\}$, where $y_i \in \mathcal{Y} \subset \mathbb{R}$ is the numeric value of Y at time i, and t is the length of Y. The goal of multi-step-ahead forecasting[2] is to predict the value of the H upcoming observations of the time series, $y_{t+1}, \dots, y_{t+H}$, where H denotes the forecasting horizon.

We formalize the predictive task by representing the time series using an embedded time delay according to the Takens theorem (Takens, 1981). Then, we apply an auto-regressive approach where each observation of the time series is modeled as a function of its past lags (Bontempi et al., 2013).

The outcome is a data set $\mathcal{D} = \{(X, y)\}$, where $y_i \in \mathcal{Y} \subset \mathbb{R}$ represents the ith observation and $X_i \in \mathcal{X} \subset \mathbb{R}^q$ is the ith associated embedding vector of size q. In other terms, the value of the target y_i is associated with the input X_i containing the previous q values $X_i = \{y_{i-1}, y_{i-2}, \dots, y_{i-q}\}$. Given such a representation, a multiple regression approach may be used to learn the dependency $y_i = f(X_i)$ on the basis of the data set $\mathcal{D}$.

2.2. Multi-step-ahead forecasting

The above formalization relates to the prediction of a single value of a time series, i.e., one-step-ahead forecasting. In this subsection, we review several state-of-the-art methods that extend this formalization to a multi-step forecasting setting. These can be split into two types: single-output methods (Section 2.2.1), and multi-output methods (Section 2.2.2). We follow Taieb et al. (2012) to describe some of these methods.

[2] Multi-step-ahead forecasting is also referred to as long-term forecasting (Taieb et al., 2012).

2.2.1. Single-output methods

The simplest approach for multi-step-ahead forecasting is the recursive method, also known as the iterative strategy. It works by fitting a model f to predict the value of the following observation:

$$y_{i+1} = f(\{y_i, y_{i-1}, \dots, y_{i-q+1}\})$$

The model f is iterated H times to produce forecasts for the complete forecasting horizon. For example, $\{y_i, y_{i-1}, \dots, y_{i-q+1}\}$ is the input vector used to forecast the value of y_{i+1} in the ith time step. Let $\hat{y}_{i+1}$ denote this forecast. Then, $\{\hat{y}_{i+1}, y_i, \dots, y_{i-q}\}$ is the input vector for forecasting $\hat{y}_{i+2}$ in the same time step.

The main limitation of the recursive strategy is the propagation of errors along the forecasting horizon. Effectively, an accurate model specification is critical for the application of this method. Nevertheless, some extensions to the recursive approach have been developed to mitigate error accumulation. Taieb and Bontempi (2011) proposed a method called RECNOISY. The gist of the approach is to add noise to the data set at each forecasting horizon. The goal is to make the model robust to the propagation of errors.

Another approach that attempts to mitigate the error propagation problem is the data as demonstrator (DaD) approach by Venkatraman et al. (2016, 2015). The authors argue that the compounding of errors leads to a change in the input distribution. Therefore, they attempt to use the training set to correct errors that occur during multi-step prediction. It accomplishes this by iteratively enriching the training set with these corrections. After this, a recursive approach is carried out using the enriched training data.

The approach behind DaD is similar to the one presented here. Both leverage the training data to correct errors made during multi-step predictions. However, DaD works as a preprocessing step for the recursive strategy. It augments the training data by imitation learning. By contrast, FTN does not alter the training data. It takes multi-step predictions as the main input and corrects (or regularizes) them using an instance-based approach.

The direct strategy is a common alternative to the recursive approach. It works by building a forecasting model for each horizon:

$$y_{i+h} = f_h(\{y_i, y_{i-1}, \dots, y_{i-q+1}\}),$$

where $h \in \{1, \dots, H\}$. By building an individual model for each horizon, the direct approach avoids the problem of error propagation. There are two main disadvantages of this approach. The first is that it incurs greater computational cost than a recursive method. The second is that it assumes that each horizon is independent, which can lead to poor results.

DirRec is an approach that attempts to combine the main ideas of the recursive and direct strategies. It builds one model for each forecasting horizon, similar to the direct approach. For each successive horizon, the set of inputs is augmented with the predictions of previous steps. This method resembles classifier chains from the machine learning literature (Read, Pfahringer, Holmes, & Frank, 2021).

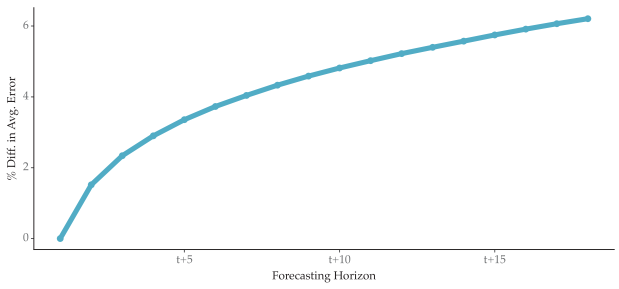

Fig. 1. Performance of a forecasting model following a `recursive` strategy for multi-step-ahead forecasting. The values represent the percentage difference between the error in the respective horizon and the error incurred at $t + 1$. The error increases along the horizon.

2.2.2. Multiple output methods

The main criticism of single-output models is that, by modeling one horizon at a time, they do not account for the conditional stochastic dependency along the forecasting horizon (Bontempi & Taieb, 2011). This problem is solved by multi-input multi-output (MIMO) models (Taieb et al., 2012). All methods described so far are multi-input insofar as the explanatory variables are multivariate.

A multi-output model learns a single model for the complete forecasting horizon:

$$[y_{i+1}, y_{i+2}, \dots, y_{i+H}] = F(\{y_i, y_{i-1}, \dots, y_{i-q+1}\}),$$

where F denotes the multi-output model. DIRMO extends MIMO by combining it with the direct approach. Essentially, it applies a MIMO approach for different blocks of the forecasting horizon.

Taieb et al. (2012) compared different approaches for multi-step ahead forecasting. They resorted to 111 time series from the NN5 forecasting competition and concluded that multi-output strategies provide better forecasting accuracy. Their experiments also suggest that deseasonalizing the series improves forecasting accuracy.

The method we propose is complementary to all multi-step forecasting strategies listed above. FTN is an instance-based meta-learning algorithm that takes multi-step predictions (a forecasted trajectory) as input. Therefore, FTN can be regarded as a postprocessing procedure designed to correct errors made during multi-step predictions by enforcing a conditionally dependent criterion.

2.3. k-Nearest neighbors

The proposed method leverages k-nearest neighbors (KNN) as a learning algorithm. Therefore, in this subsection, we describe the KNN method (Altman, 1992; Fix & Hodges, 1989). KNN is referred to as a *lazy* algorithm because it defers the computational work to the inference stage when a new query arrives. As the name implies, the idea behind KNN is to find instances close to a query instance (the neighbors). Then, the corresponding target variable of these neighbors is aggregated for prediction. Usually, distance-based approaches are used to determine how close two instances are to each other, for example, the Euclidean distance.

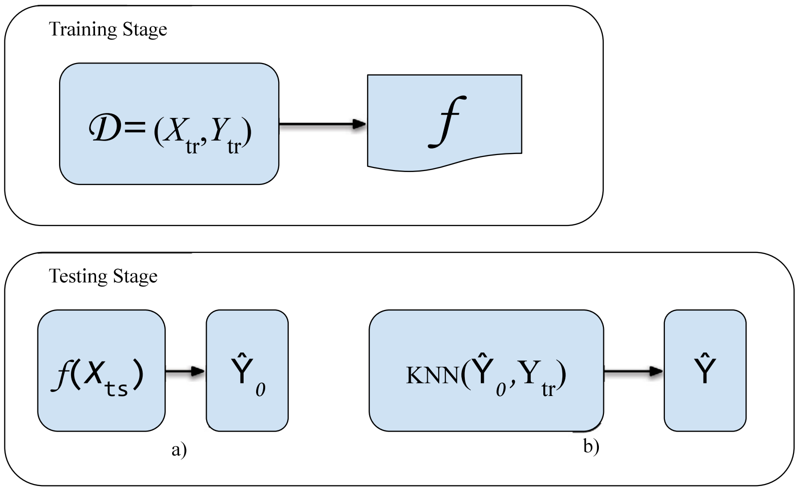

Fig. 2. Workflow of FTN. In the training stage, a model f is created for multi-step-ahead forecasting. Then, the testing stage is split into two parts. First, we obtain the tentative forecasts $\hat{Y}_0$ from the model. Then, these forecasts are corrected using a KNN algorithm to obtain the final predictions $\hat{Y}$.

Traditionally, KNN keeps all training data in memory for the inference stage. Nonetheless, fast indexing structures can be used to speed up the search, for example, the ball-tree or KD-tree data structures (Pedregosa et al., 2011). Here, we use the implementation from the *scikit-learn* library (Pedregosa et al., 2011), which automatically selects the best indexing approach for the input data.[3]

KNN has often been used in time series forecasting (see Bontempi, Birattari, & Bersini, 1999 and the references therein) and is commonly used for multi-step forecasting as a MIMO approach (Taieb et al., 2012). Accordingly, it works by finding input vectors (X in the formalization in Section 2.1) and then averaging the respective target variables. Besides multi-step forecasting, KNN is also popular for time series classification (Bagnall, Lines, Bostrom, Large, & Keogh, 2017).

3. Methodology

In this section, we first formalize FTN (Section 3.1) and then we highlight the contributions relative to the current literature (Section 3.2).

3.1. Forecasted trajectory neighbors

Forecasting accuracy decreases significantly as predictions are made further into the future (cf. Fig. 1). This is caused by two main issues: the decreased predictability of time series in the long term, and the propagation of errors along the horizons. Besides these problems, the conditional dependency among different forecasting horizons is not always taken into account by forecasting models (Bontempi & Taieb, 2011).

We developed FTN to tackle these limitations. The general workflow for this method is summarized in Fig. 2. Let $\mathcal{D}(X_{tr}, Y_{tr})$ denote the training data set, where X_{tr} and X_{ts} represent the training and testing explanatory variables, respectively. Similarly, Y_{tr} and Y_{ts} represent the respective target variables, i.e., the observed trajectories. Note that Y_{tr} and Y_{ts} are matrices with a number of columns equal to the forecasting horizon H. The term trajectory refers to the embedding of a time series in a state space (Bontempi et al., 2013). We distinguish between observed trajectories and forecasted trajectories. The former refer to the trajectories obtained after reconstructing the time series using time-delay embedding. The latter represent the trajectories predicted by a forecasting model.

In the offline stage, we train a model f to forecast the upcoming observations using (X_{tr}, Y_{tr}). The online (inference) stage works in two steps. First, we retrieve $\hat{Y}_0$ (considered as a tentative forecast) which contains the predictions for the testing instances. We assume that such a forecast may be improved since it may contain errors due to propagation or the lack of conditional dependency. Our rationale is to use the training data to correct these predictions. This is accomplished by retrieving the k nearest neighbors of $\hat{Y}_0$ in the target values of the training set (Y_{tr}). The rationale behind our approach can be encapsulated in the following question: What are the closest trajectories of this prediction that already happened in the past? Since these trajectories have been observed, by definition they comply with the conditional dependency of the time series (Bontempi & Taieb, 2011).

Fig. 3 illustrates how FTN works in a given time step. The model receives the latest observations as input and produces its forecasts. The forecasts by the model are then tailored according to their nearest neighbors, leading to the final forecasts by FTN.

For the ith time step, the process behind FTN can be formalized as follows. The KNN algorithm is based on a distance metric dist(a, b). Let dist(a, b) be defined as the

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

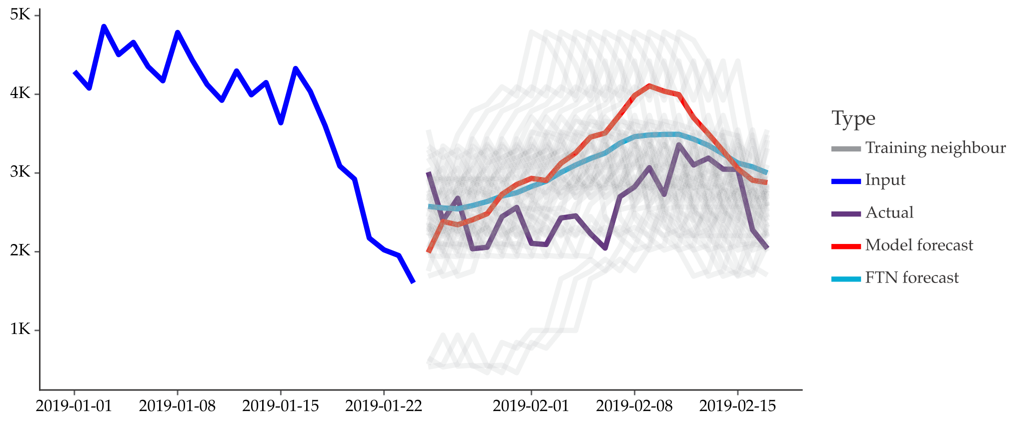

Fig. 3. Example of the application of FTN in a given time step.

Euclidean distance between two vectors a and b:

$$dist(a, b) = \sqrt{\sum_{j=1}^{q}(a_j - b_j)^2} \tag{1}$$

where q is the dimension of a and b. Let $\mathcal{N}_{\hat{Y}_0^i}$ denote the k nearest neighbors of $\hat{Y}_0^i$, the ith tentative forecasted trajectory. $\mathcal{N}_{\hat{Y}_0^i}$ is a subset of Y_{tr}, the target variables of the training set:

$$\begin{aligned} \mathcal{N}_{\hat{Y}_0^i} &\subseteq Y_{tr} \\ \text{s.t.} |\mathcal{N}_{\hat{Y}_0^i}| &= k, \text{ and} \\ \forall b \in Y_{tr} \setminus \mathcal{N}_{\hat{Y}_0^i}, \quad \text{dist}(b, \hat{Y}_0^i) &\geq \max_{a \in \mathcal{N}_{\hat{Y}_0^i}} \text{dist}(a, \hat{Y}_0^i) \end{aligned} \tag{2}$$

The k instances in $\mathcal{N}_{\hat{Y}_0^i}$ are averaged to obtain the final prediction $\hat{Y}^i$:

$$\hat{Y}^i = \overline{\mathcal{N}_{\hat{Y}_0^i}} \tag{3}$$

The proposed method works as a postprocessing procedure. The main input is the forecasts $\hat{Y}_0^i$ for the h upcoming observations in the ith time step. FTN is agnostic to the underlying approach used to obtain multi-step-ahead forecasts. Therefore, it can be integrated with any state-of-the-art approach, such as the ones described in Section 2.2.

3.2. Relation to previous work

The KNN algorithm is central to the proposed method. As we mentioned in Section 2.3, KNN is commonly applied for multi-step forecasting tasks. However, in our work, we apply this method in a different manner.

The standard KNN approach compares the explanatory variables (X) of a new observation with X_{tr} (the training explanatory variables). Then, it retrieves and averages the target variables (Y_{tr}) of the closest instances (nearest neighbors). Previous works have followed this type of approach for multi-step-ahead forecasting (Bontempi & Birattari, 2000; Bontempi et al., 1999). By contrast, our method requires a vector of forecasts $\hat{Y}_0$ that are compared directly with the set of rows of Y_{tr}, bypassing the explanatory variables. In both cases, the final prediction is obtained by averaging observations from Y_{tr}. The key difference is in how these observations are retrieved. Essentially, state-of-the-art approaches retrieve the neighbors of the current lagged state. The proposed approach retrieves the neighbors of the forecasted trajectory.

FTN is an example of a meta-learning algorithm for forecasting. It leverages the predictions of a baseline forecasting model to return the final forecasts in an instance-based fashion. In some respects, FTN resembles stacked generalization by Wolpert (Wolpert, 1992), commonly referred to as stacking. However, there are two key differences between FTN and stacking. First, stacking uses a meta-model to predict the target on the basis of the predictions of several base models. Instead, FTN uses the predictions of a single baseline model.

The second difference may be made explicit by supposing that stacking, like FTN, adopts KNN as a meta-model. In front of a prediction query, the stacking KNN searches in the set of predicted continuations and averages the associated targets to produce the final forecasting. This is not the case with FTN, which directly searches for the nearest continuations in the training set.

4. Experiments

In this section, we describe the experiments carried out to validate the proposed method. We present the data sets (Section 4.1), and we discuss the experimental design (Section 4.2) and the considered methods (Section 4.3). Then, we outline the research questions (Section 4.4) and we present the results (Section 4.5).

4.1. Data

The experiments encompass 7795 time series from eight popular databases. A summary is presented in Table 1. The names of the data sets refer to their identifiers in the Python library *gluonts*, where they were retrieved.

Table 1
Summary of the data sets.

Name	Frequency	# Time series	Avg. length
electricity_nips	Hourly	370	5833
nn5_daily_without_missing	Daily	111	735
solar-energy	Hourly	137	7009
traffic_nips	Hourly	963	4001
taxi_30min	Half-hourly	1214	1488
m4_hourly	Hourly	414	960
m4_daily	Daily	4227	2940
m4_weekly	Weekly	359	934

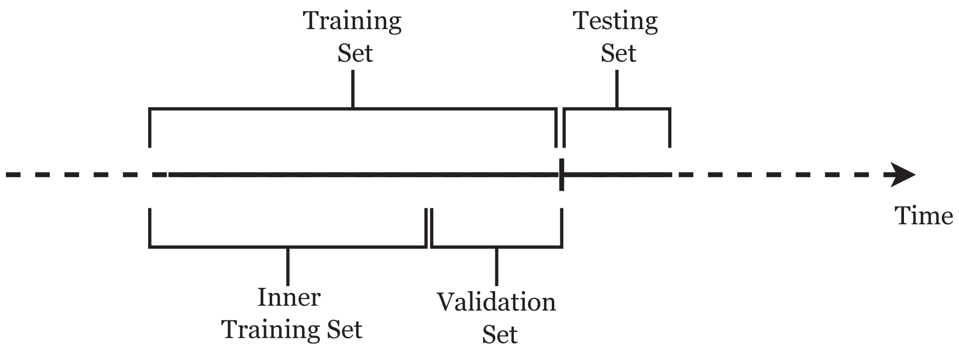

Fig. 4. Data partitioning process in one run of the cross-validation procedure.

4.2. Experimental design

We applied time delay embedding according to the formalization presented in Section 2.1. The maximum forecasting horizon is set to 18 ($H = 18$). The number of lags (q) is chosen in {2, 4, 6, 8, 10, 20, 30, 50} using a validation strategy (explained below). Preprocessing consists of trend removal by first differences and seasonality modeling by assessing the inclusion of two Fourier terms (Young, Pedregal, & Tych, 1999). The two terms are two sine series plus two cosine series that are added to the lagged variables. The Fourier terms are included only if they improve forecasting accuracy on the validation set.

We used a Monte Carlo cross-validation procedure to evaluate the accuracy of models (Picard & Cook, 1984). This estimation method, also referred to as repeated holdout, has been shown to provide competitive forecasting accuracy estimates with univariate time series (Cerqueira, Torgo, & Mozetič, 2020). Monte Carlo cross-validation is applied with 10 folds. The training and test sizes in each fold are 60% and 10% of the size of the available data, respectively. In each of the 10 runs, a point is randomly selected from the available data (constrained by the size of training and testing data). This point then represents the end of the training set and the start of the testing set (Cerqueira et al., 2020).

Fig. 4 illustrates the data partitioning process within a given run of the Monte Carlo cross-validation procedure. Each run also includes a nested validation split. This validation set is used for two purposes: to optimize the parameters of the learning algorithm (cf. Section 4.3.1), and to optimize the number of models in an ensemble (cf. Section 4.3.2). The validation process is carried out as follows. The training set is split into two parts: an inner training set and a validation set (see Fig. 4). The inner training set contains 70% of the observations of the complete training set. The validation set contains the subsequent and final 30% of the complete training set. For the optimization and training process, the models are fit using the inner training set and evaluated in the validation set. The methods are re-fit using the complete training set after the optimization.

We evaluate forecasting accuracy according to the mean absolute error (MAE). While this metric is scale-dependent, we resort to ranks and percentage differences to compare the results across different data sets.

4.3. Methods

In this subsection, we detail the benchmarking methods that were used in the experiments.

4.3.1. Single output

We used the following single-output approaches:

- `Direct`: building a forecasting model for each horizon.
- `Recursive`: building a forecasting model for one-step-ahead forecasting, and iterating it to obtain forecasts for multiple horizons.
- `DirRec`: building a forecasting model for each horizon and adding the previous forecasts as input to the subsequent ones.

These methods are described in more detail in Section 2.2.2.

All forecasting models that follow the strategies above were built using the `lightgbm` learning algorithm. This algorithm was the backbone of the winning solution of the M5 forecasting competition (Makridakis, Spiliotis, &

Table 2
Pool of parameters for the `lightgbm`.

ID	Parameter	Value
`num_leaves`	Max # of leaves per tree	{3, 5, 10, 15}
`max_depth`	Max depth per tree	{-1, 3, 5, 10, 15}
`lambda_l1`	L1 regularization	{0.1, 1, 10, 100}
`lambda_l2`	L2 regularization	{0.1, 1, 10, 100}
`learning_rate`	Learning rate	{0.05, 0.1, 0.2}
`min_child_samples`	Min # of points per leaf	{7, 15, 30}
`boosting_type`	Base algorithm	gbdt
`num_boost_round`	Boosting iterations	200
`early_stopping_rounds`	Early stopping	30

Table 3
Learning algorithms used in the multi-output ensemble.

ID	Algorithm	Parameter	Value
`Ridge`	Ridge regression	L2 alpha	{0.25, 0.5, 0.75, 1}
`LASSO`	LASSO regression	L1 alpha	{0.25, 0.5, 0.75, 1}
`ElasticNet`	ElasticNet regression	*Default*	–
`KNN`	*k*-nearest neighbors	No. of neighbors	{5, 10, 20}
		Weights	{Uniform, Distance}
`ExtraTree`	Extra trees regression	No. trees	{50, 100}
		Max depth	{*default*, 3}
`RF`	Random forests	No. trees	{50, 100}
		Max depth	{*default*, 3}
`BAG`	Bagging	No. of trees	{50, 100}
`PLS`	Partial least regr.	No. of components	{2, 5}
`PCR`	Principal components regr.	No. of components	{2, 5}

Assimakopoulos, 2022). This method is sensitive to different parameter configurations. Therefore, for each time series, we carried out 100 iterations of random search to optimize it using a validation set. The pool of parameters is detailed in Table 2.

4.3.2. *Multi-output*

We also tested three multi-output multi-step forecasting methods:

- `KNN`: this is the method in Taieb et al. (2010), where the nearest neighbors are selected in the input space and the corresponding target continuations are simply averaged.
- `Ensemble`: it returns the average forecasting of a heterogeneous ensemble of several multi-output models.
- `FTN Ensemble` (`FTNE`): a variant of the ensemble method where `FTN` is applied to each ensemble member. In other words, `FTNE` is an ensemble of `FTN`-augmented forecasting models, also weighted uniformly.

The list of models for the ensemble is shown in Table 3. All these algorithms were implemented in Python via the *scikit-learn* library. The final ensemble contains 10 learning algorithms with different parameter configurations. Note that the starting number of models is 30, and that we pruned the ensemble by removing the ones with the worst validation accuracies (Cerqueira, Torgo, Pinto, & Soares, 2019; Jose & Winkler, 2008). We remark that a dynamic combination rule (e.g. exponentially weighted average) typically improves the performance of forecasting ensembles relative to a static one. Nevertheless, the simple average is a solid benchmark according to previous works on this topic (Cerqueira et al., 2019).

We also include the following three classical forecasting methods as baselines:

- `ARIMA`: the ARIMA method with order (1,1,1) (Box, Jenkins, Reinsel, & Ljung, 2015). We fixed the order of the model rather than using automatic optimization procedures in the interest of computational efficiency.
- `Theta`: the theta method (Assimakopoulos & Nikolopoulos, 2000), which is an exponential smoothing-based approach.
- `Naive`: the seasonal naive method that uses the last known value of the same seasonal period as forecast.

4.3.3. *FTN*

We consider three variants of `FTN`:

- Standard `FTN`: basic version described in the previous section, where the k nearest trajectories to $\hat{Y}_0$ are averaged.
- `FTN Alpha`: it linearly combines the original forecast ($\hat{Y}_0$) and the `FTN` forecast ($\hat{Y}$) as follows: $\alpha \times \hat{Y} + (1 - \alpha) \times \hat{Y}_0$, where $\alpha = 0.5$ in the experiments.
- `FTN Smooth`: it returns the `FTN` forecast using a smoothed version of the time series according to the principle discussed in Helmi, Fakhr, and Atiya (2018). In the experiments, smoothing consists of an exponentially weighted moving average with a 0.6 smoothing factor. Note that the smoothing operation

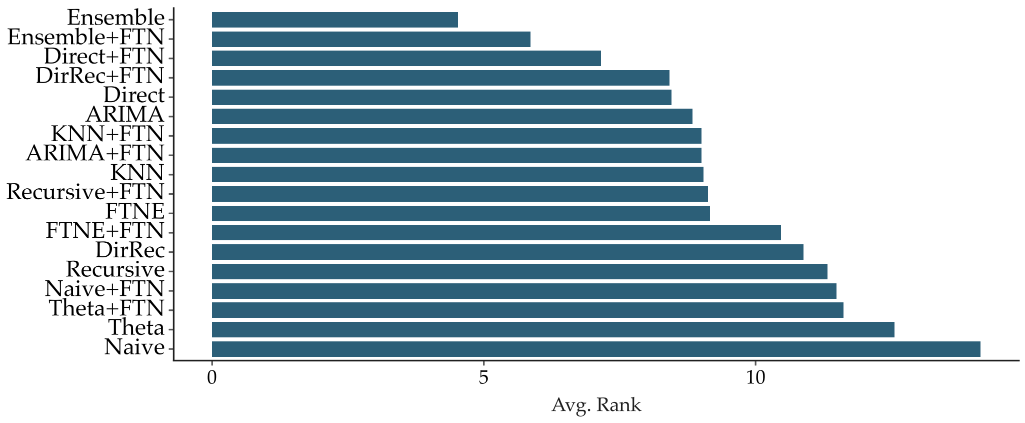

Fig. 5. Average rank across the 7795 time series.

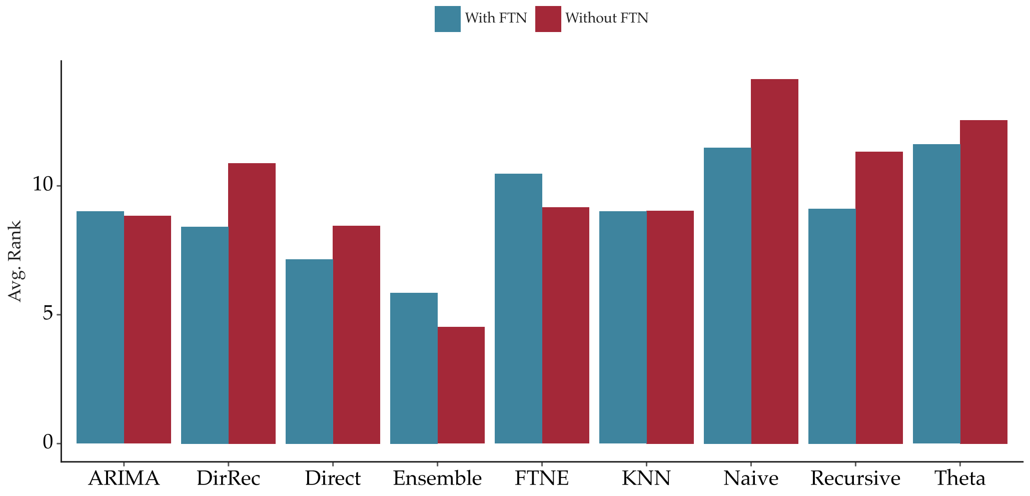

Fig. 6. Average paired ranks: Most approaches improve their average rank when combined with FTN.

is only applied during FTN postprocessing, while the original forecast is obtained using the original time series without smoothing. Moreover, this smoothing is applied to each neighbor and not after their average.

Note that for all variants, the most important FTN hyperparameter is the number of nearest neighbors. In the main experiments, this value was set to $k = 150$. A justification for these choices is presented in 4.5.2, below.

4.4. Research questions

The experiments were designed to address the following research questions:

1. **RQ1**: What is the impact of FTN on the performance of each multi-step-ahead forecasting approach?
2. **RQ2**: Which is the overall best approach for multi-step-ahead forecasting?
3. **RQ3**: Does the impact of FTN depend on the forecasting horizon?
4. **RQ4**: What is the sensitivity of FTN to the number of nearest neighbors of KNN?
5. **RQ5**: How do the different variants of FTN compare with each other?
6. **RQ6**: Does the size of the time series have an impact on the performance of FTN?

4.5. Results

In this subsection, we present the results of the experiments. Fig. 1 (cf. Section 1) shows the application of the recursive approach (Recursive) to the 7795 time series. Each value represents the average percentage difference in error (MAE) of Recursive in the respective horizon relative to the error of Recursive for one-step-ahead forecasting (at $t + 1$). This analysis quantifies the impact of predicting multiple steps in forecasting accuracy.

4.5.1. Added value of FTN

Fig. 5 shows the average rank plot, representing the average relative position of a method. Note that for each

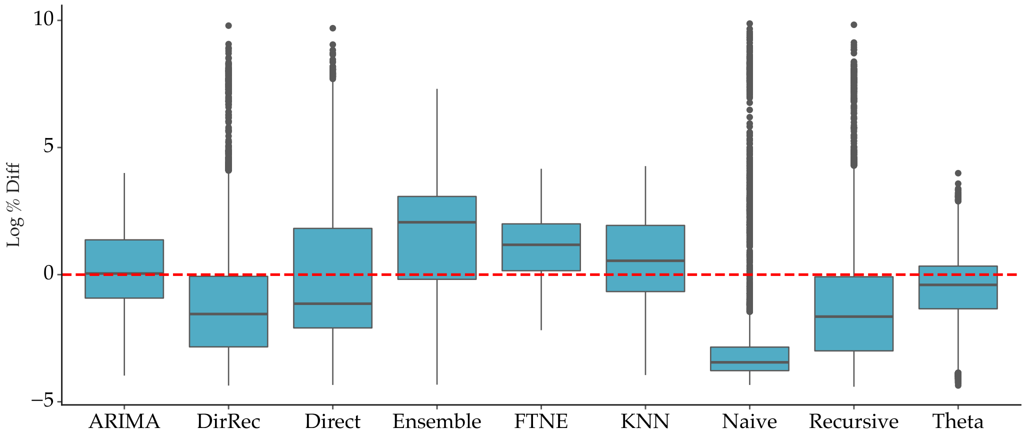

Fig. 7. Percentage difference in accuracy with and without FTN postprocessing. Negative values denote a gain in accuracy due to FTN.

task, a method is given the rank 1 if it has the lowest mean absolute error. It appears that Ensemble has the best performance, followed by Ensemble+FTN and Direct+FTN.

Fig. 6 details the impact of FTN on the average rank of each method. Most methods improve their average rank score when applied with FTN, except for Ensemble, ARIMA, and FTNE. Since the average rank score does not consider the magnitude of differences in forecasting accuracy, we complement this metric with the percentage difference in accuracy between each method and a reference. For a given method m, the percentage difference is

$$100 \times \frac{\mathrm{MAE}_{\mathtt{m}} - \mathrm{MAE}_{\mathrm{reference}}}{\mathrm{MAE}_{\mathrm{reference}}}$$

where $\mathrm{MAE}_{\mathtt{m}}$ and $\mathrm{MAE}_{\mathrm{reference}}$ represent the MAE of method m and the reference method, respectively. The lower the score, the better the accuracy. Negative values denote that m outperforms the reference. This quantity is used to assess the accuracy gain related to the FTN adoption over the 7795 time series (Fig. 7). FTN has the highest impact when applied to the Naive approach. Also, the median percentage difference is below or close to zero for most methods.

Fig. 8 repeats the analysis by using Recursive, the popular approach for multi-step forecasting. Methods are ranked in terms of the median percentage difference. Interestingly, this order is similar to the one in the average rank plot.

While the average rank indicates that FTN leads to better forecasting accuracy in most methods, Figs. 7 and 8 suggest that the percentage difference is often negligible (close to zero). For this reason, we carry out additional analysis by neglecting small differences in accuracy, according to the principle of practical equivalence (Benavoli, Corani, Demšar, & Zaffalon, 2017). The region of practical equivalence is set to [−1%, 1%] and two methods are considered equivalent if their percentage difference is within this interval. Fig. 9 illustrates the probability of a winning FTN in blue (percentage difference below −1%), of drawing in green (results within [−1%, 1%]), and losing in red (percentage difference above 1%). For instance, for DirRec, FTN has a winning/losing/drawing probability around 70%/25%/5%, respectively. For Direct, DirRec, Recursive, Theta, and Naive, the probability of FTN winning is higher than that of losing. In the cases of Ensemble, FTNE, KNN, and ARIMA, the results are reversed: the probability of FTN losing is higher than that of winning.

The previous analyses measured the average accuracy over the entire forecasting horizon, ranging from $t + 1$ to $t + 18$. Now, we assess how the impact of FTN varies over the forecasting horizon. Fig. 10 reports the difference in average rank with and without FTN. Negative values denote improved accuracy due to the adoption of FTN postprocessing. FTN appears to have the highest impact for large horizons. For one-step-ahead prediction, all methods perform better without FTN (except for Naive). However, as the forecasting horizon increases, FTN improves the accuracy for Direct, Recursive, DirRec, and Theta.

4.5.2. Sensitivity analyses

The FTN strategy, like all the data-driven strategies, requires the definition of some hyperparameters. This section aims to assess experimentally the impact of those choices on the final accuracy.

The first sensitivity study concerns the choice of the number of neighbors k, which, in all the experiments discussed so far, was set to 150. For the sake of conciseness, we report only the results related to the direct strategy (Fig. 11). The best value of k was chosen in the set: {1, 3, 5, 10, 20, 50, 100, 150, 200, 300, 500, 750, 1000, 2000}. Direct+FTN(150) shows the best average rank score, followed by Direct+FTN(200) and Direct+FTN(100). It is interesting to note a bias–variance tradeoff related to the setting of k. Too small values (1,3,5) increase the variance, while too large values of k induce a bias in the prediction.

The second sensitivity study concerns the FTN variants presented in Section 4.3.3. Fig. 12 shows the average rank of each variant across all time series. For simplicity, we considered only three forecasting methods:

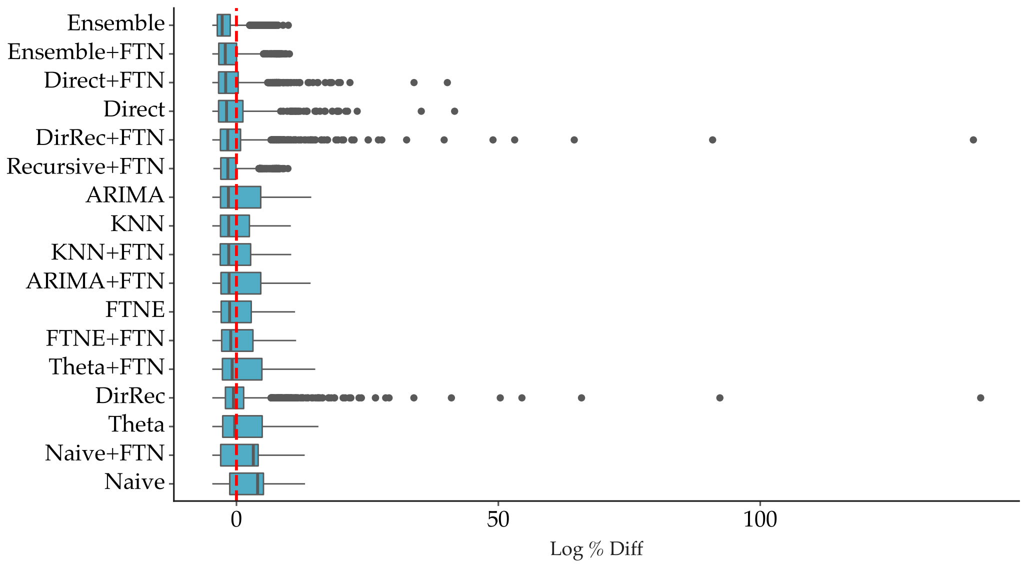

Fig. 8. Percentage difference in performance between the respective method and the recursive approach. Negative values denote better accuracy than the recursive approach.

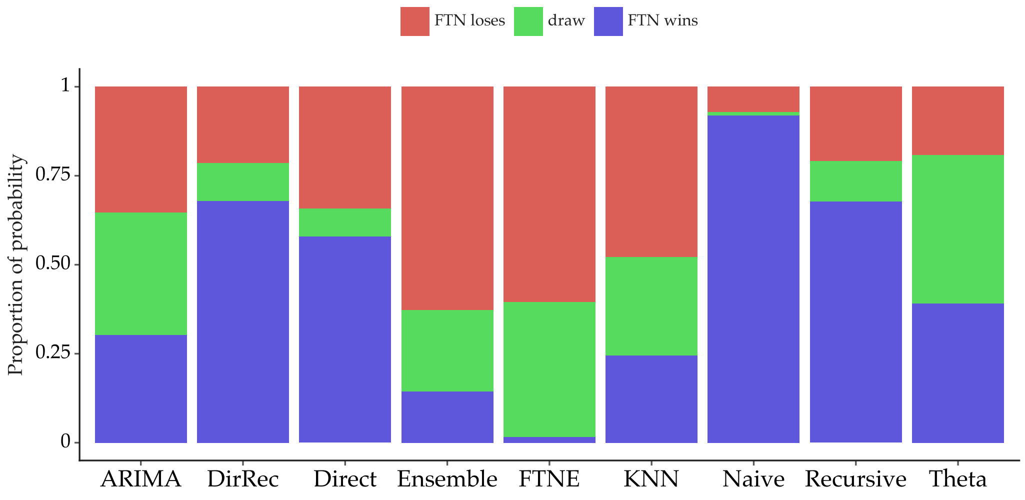

Fig. 9. Paired comparisons for each multi-step forecasting strategy. Each stacked barplot shows the probability of FTN winning in blue (result below −1%), drawing in green (result within [−1%, 1%]), or losing in red (result above 1%). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(Recursive, Direct, and DirRec). The alpha version of FTN outperformed the other variants. Overall, all FTN versions improved the original forecasts (except for FTN and Direct).

The third study concerns the sample size (cf. Table 1) and its impact on the final accuracy (Cerqueira, Torgo, & Soares, 2022). Similarly to Fig. 7, we consider the percentage gain in accuracy with and without FTN and we assess whether this difference depends on the sample size (Fig. 13). For simplicity, we focus on three methods: Recursive, Direct, and DirRec. The figure is a scatterplot that illustrates the percentage difference (log-scaled) on the *y*-axis and the sample size on the *x*-axis. Negative values denote better performance when the model is FTN postprocessed. Although most of the time series contain fewer than 2000 data points, the results indicate that the performance difference is reasonably stable concerning the sample size.

5. Discussion

The experiments provided empirical evidence about the beneficial impact of FTN in terms of forecasting accuracy for several multi-step forecasting strategies (**RQ1**).

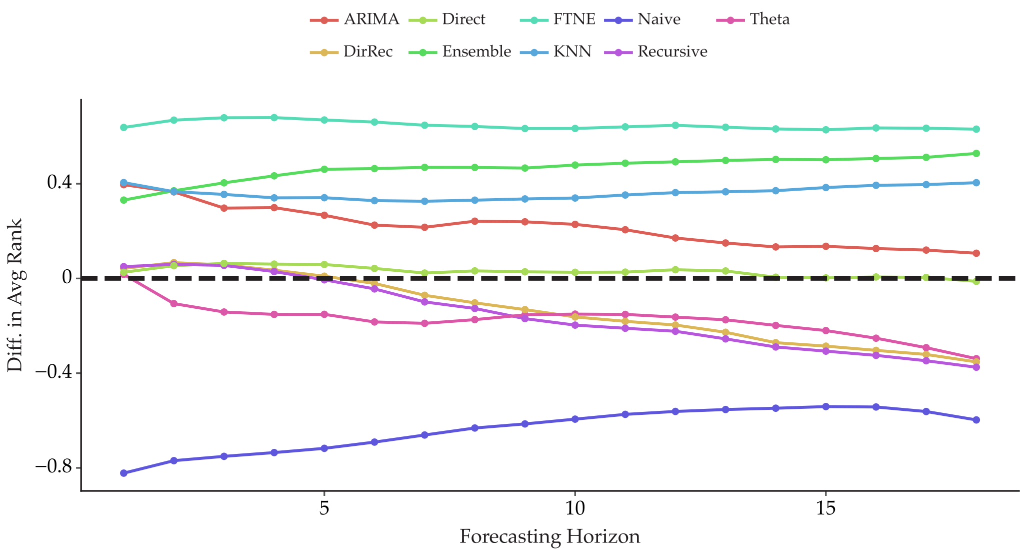

Fig. 10. Average percentage difference with and without FTN preprocessing for different forecasting horizons. A negative value means that accuracy improves by using FTN postprocessing.

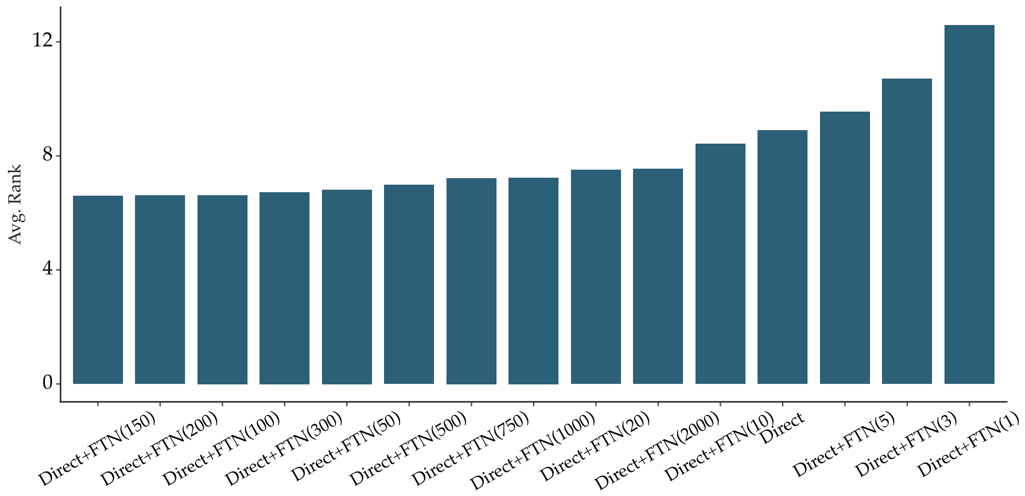

Fig. 11. Average rank of different FTN configurations when applied with the direct approach across the 7795 time series. The number of nearest neighbors used in FTN is denoted between parentheses.

The first objective of FTN is to address the problem of error propagation. Indeed, FTN improves the performance of two methods highly exposed to error propagation: recursive and DirRec. The second rationale of FTN is the enforcement of a conditional dependency constraint, namely to regularize conditionally independent strategies like the direct approach. This reasoning seems to be supported by (i) the accuracy improvements due to the adoption of the FTN postprocessing with the direct approach, and (ii) the slight improvement of the MIMO KNN strategy, which already satisfies the conditionally dependent constraint. Also, FTN fails to improve the performance of a classic approach, namely ARIMA. Nevertheless, it improves the performance of Naive and Theta.

Overall, the best-performing approach is a heterogeneous ensemble of multi-output models. An ensemble is said to be heterogeneous if it combines different types of learning algorithms. The ensemble performs well even without FTN (**RQ2**). A possible justification for this result is related to the bias–variance tradeoff. While both FTN and Ensemble rely on averaging several trajectories to reduce variance, FTN is more exposed to bias in cases where the nearest neighbors are too far from the original forecasts.

In the third research question (**RQ3**), we take into consideration the impact of FTN postprocessing for different forecasting horizons. It appears that our method is most beneficial for long-term prediction. The two methods that suffer from error propagation (namely Recursive and DirRec) do not take advantage of FTN for short horizons. Nevertheless, Direct has better accuracy for large forecasting horizons when coupled with FTN.

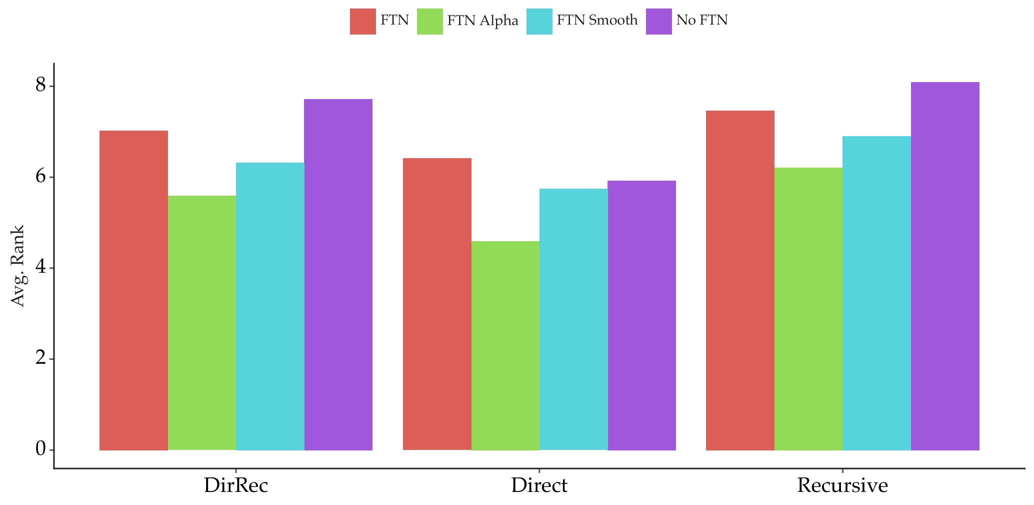

Fig. 12. Average rank (over the 7795 series) of FTN and No FTN postprocessing for three forecasting strategies. The lower the rank, the better.

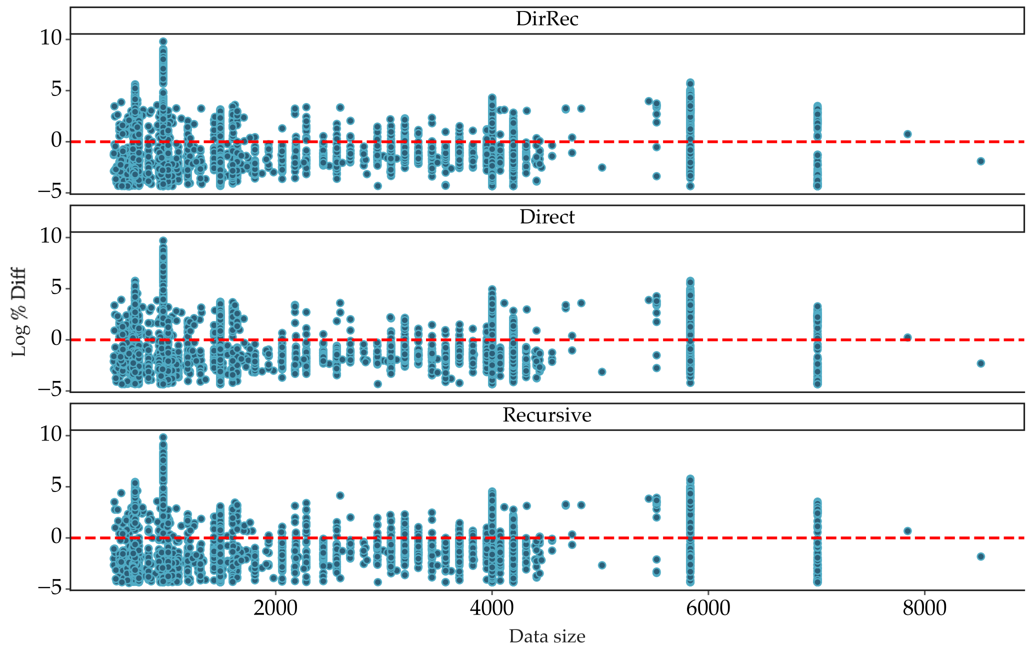

Fig. 13. Percentage accuracy difference between FTN and No FTN postprocessing vs. the sample size of the series. Negative values denote an accuracy improvement due to FTN postprocessing.

Finally, we studied the sensitivity of FTN to different parameter configurations.

Given the nearest-neighbor nature of FTN, we considered the impact of the number of neighbors. The results confirm the usual bias–variance tradeoff of local learning algorithms. Too few neighbors (10 or fewer) as well as too many neighbors may be detrimental to the accuracy. The best tradeoff is around 150 neighbors (**RQ4**), though this parameter might strongly depend on the stochastic nature of the series.

The second sensitivity study concerned the possible variants of FTN. The experiments indicate that FTN Alpha is the best FTN variant (**RQ5**). For the sake of simplicity, we set $\alpha = 0.5$ in the experiments to ensure the same weight for the original and the postprocessed forecast. However, this parameter could be tuned (like k) by using a validation strategy.

Regarding the impact of the sample size (**RQ6**), the experimental results suggest that such a property does not impact the performance of FTN. However, our analysis is limited by the size of the time series in our case study.

We used a large set of time series from several application domains. We biased our selection toward time series with medium to high sampling frequency and a reasonable sample size (at least 500 observations). Nonetheless, the methodology applies to generic univariate time series given the agnostic nature of the approach concerning the underlying time series. This includes time series with exogenous input variables. FTN is agnostic to the input explanatory variables, as it works directly with the multi-step predictions and output variables (observed trajectories).

The method is also agnostic to the underlying learning algorithm or multi-step forecasting strategy. In our case, single-output multi-step forecasting methods were trained using lightgbm. This algorithm is a state-of-the-art method for different predictive tasks, including forecasting. It was the learning algorithm used in the winning solution of the M5 forecasting competition (Makridakis et al., 2022). We used several multi-output methods in ensembles, including random forests and nearest neighbors, among others (cf. Table 3).

In Section 3, we described how FTN relates to other meta-learning approaches. To our knowledge, the most similar approach is stacking (Wolpert, 1992). In our case, there is only one base predictive model, and the generalizer (meta-model) is a KNN. Both approaches use base predictions as input. Stacking compares these predictions with past predictions and then uses the corresponding targets. By contrast, FTN works directly with the target variables of the training data.

6. Conclusions

This paper dealt with univariate multi-step-ahead forecasting. This task is challenging because of the increasing uncertainty related to long-term prediction. Further, two issues amplify the difficulty of this problem: the propagation of errors along the horizon, and the lack of conditional dependency between different horizons.

We proposed FTN, an instance-based meta-learning algorithm that regularizes a forecasting model by enforcing conditional dependency. The algorithm is made of three main steps: (i) computing an initial multi-step forecast, e.g. using a state-of-the-art predictive model, (ii) retrieving the nearest neighboring trajectories from the training set, and (iii) returning the average of these neighbors as the final prediction. The main rationale of the approach is: (i) to use a multi-output strategy to reduce the error propagation, and (ii) to average actually observed trajectories and then preserve the conditional dependency property in the forecast.

We carried out experiments using 7795 time series from several domains of application, and we assessed the beneficial impact of FTN postprocessing once combined with conventional multi-step-ahead forecasting strategies. We found that the proposed method improves the forecasting accuracy of several forecasting approaches, especially when applied to single-output strategies. Future work will focus on multivariate versions of the approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work of L. Torgo was undertaken, in part, thanks to funding from the Canada Research Chairs program. Gianluca Bontempi was supported by Service Public de Wallonie Recherche under grant number 2010235–ARIAC by DIGITALWALLONIA4.AI.

References

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.

Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, *16*(4), 521–530.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research*, *18*(1), 2653–2688.

Bontempi, G., Ben Taieb, S., & Le Borgne, Y.-A. (2013). Machine learning strategies for time series forecasting. In *Business intelligence: second European summer school, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures 2* (pp. 62–77). Springer.

Bontempi, G., & Birattari, M. (2000). A multi-steap ahead prediction method based on local dynamic properties. In *Proceedings of ESANN 2000* (pp. 311–316).

Bontempi, G., Birattari, M., & Bersini, H. (1999). Local learning for iterated time series prediction. In *ICML* (pp. 32–38).

Bontempi, G., & Taieb, S. B. (2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting*, *27*(3), 689–699.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, *109*(11), 1997–2028.

Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2019). Arbitrage of forecasting experts. *Machine Learning*, *108*(6), 913–944.

Cerqueira, V., Torgo, L., & Soares, C. (2022). A case study comparing machine learning with statistical methods for time series forecasting: Size matters. *Journal of Intelligent Information Systems*, *59*(2), 415–433.

Chen, R., Yang, L., & Hafner, C. (2004). Nonparametric multistep-ahead prediction in time series analysis. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *66*(3), 669–686.

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*(3), 238–247.

Guo, M., Bai, Z., & An, H. Z. (1999). Multi-step prediction for nonlinear autoregressive models based on empirical distributions. *Statistica Sinica*, 559–570.

Helmi, A., Fakhr, M. W., & Atiya, A. F. (2018). Multi-step ahead time series forecasting via sparse coding and dictionary based techniques. *Applied Soft Computing*, *69*, 464–474.

Jose, V. R. R., & Winkler, R. L. (2008). Simple robust averages of forecasts: Some empirical results. *International Journal of Forecasting*, *24*(1), 163–169.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, *38*(4), 1346–1364.

McNames, J. (1998). A nearest trajectory strategy for time series prediction. In *Proceedings of the international workshop on advanced black-box techniques for nonlinear modeling* (pp. 112–128). KU Leuven Belgium.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Picard, R. R., & Cook, R. D. (1984). Cross-validation of regression models. *Journal of the American Statistical Association*, *79*(387), 575–583.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2021). Classifier chains: A review and perspectives. *Journal of Artificial Intelligence Research*, *70*, 683–718.

Taieb, S. B., & Atiya, A. F. (2015). A bias and variance analysis for multistep-ahead time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(1), 62–76.

Taieb, S. B., & Bontempi, G. (2011). Recursive multi-step time series forecasting by perturbing data. In *2011 IEEE 11th international conference on data mining* (pp. 695–704). IEEE.

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, *39*(8), 7067–7083.

Taieb, S. B., Sorjamaa, A., & Bontempi, G. (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, *73*(10–12), 1950–1957.

Takens, F. (1981). *Dynamical systems and turbulence, warwick 1980: Proceedings of a symposium held at the university of warwick 1979/80* (pp. 366–381). Berlin, Heidelberg: Springer Berlin Heidelberg, ISBN: 978-3-540-38945-3.

Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., & Bagnell, J. A. (2016). Improved learning of dynamics models for control. In *International symposium on experimental robotics* (pp. 703–713). Springer.

Venkatraman, A., Hebert, M., & Bagnell, J. A. (2015). Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI conference on artificial intelligence*.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Young, P. C., Pedregal, D. J., & Tych, W. (1999). Dynamic harmonic regression. *Journal of Forecasting*, *18*(6), 369–394.