National Technical
University of Athens

FORECASTING &
STRATEGY UNIT

Making your model generalize better

Cross-validation and data augmentation

Evangelos Spiliotis
Forecasting & Strategy Unit
National Technical University of Athens

CMAF Friday Forecasting Talks
8 March, 2022

The rise of ML

Machine Learning has gained considerable prominence fueled by a number of high profile applications in:

- Image and speech recognition
- Sentiment analysis
- Online customer support
- Medical and law usage
- Product recommendations
- Autonomous vehicles
- Fraud detection

among others…

Over the last decade, ML methods, and especially Neural Networks (NNs) and Regression Trees (RTs), have also found several applications in the field of **forecasting**, particularly in areas where data are highly available (e.g. energy, financial, and product sales forecasting)

State-of-the-Art ML forecasting methods

Neural Networks

DeepAR: Auto-regressive recurrent NN that can be optionally associated with categorical or dynamic features (*Salinas et al., 2020*)

N-BEATS: Deep auto-regressive feed-forward NN based on backward and forward residual links (*Oreshkin et al., 2020*)

Regression Trees

LightGBM: An efficient variant of gradient boosted trees that works accurately and fast, while requiring less data and handling more features (*Ke et al., 2017*)

- ✓ Excludes a significant proportion of data instances with small gradients and uses the rest to estimate the information gain
- ✓ Bundles mutually exclusive features

*These models have achieved great performance in the **M4** and **M5** competitions, several Kaggle competitions, and many forecasting studies, outperforming other forecasting approaches, both standard and advanced ones*

Advantages and limitations of data-driven learning

- In contrast to statistical models, that prescribe the underlying data generating process (e.g. in terms of data patterns and distribution), ML models allow for data relationships to be **identified** and **estimated automatically**

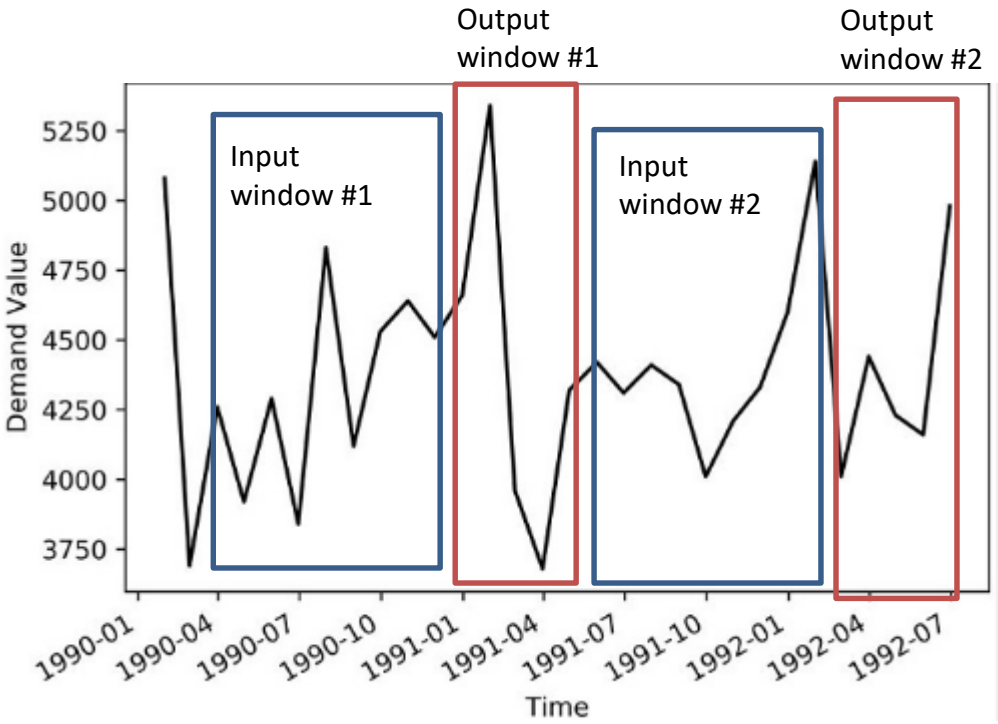
In practice, ML models learn through examples and trial-and-error, thus:

- Providing data-driven instead of model-driven forecasts
- Being more generic
- Making less assumptions

In addition, ML models usually have greater learning capacity (e.g. more trainable parameters or rules)

- Naturally, more flexibility and automation come at a **cost**: **We need significantly more data to effectively train our models!**

Auto-regressive ML forecasting models



- More often than not, time series ML models generate forecasts by applying a form of auto-regression (although external variables can also be considered)
- We have to construct multiple sets of input/output windows to let the models learn how to accurately predict the target observations based on the information provided as input
- Since the future is rarely the same as the past, involving uncertainty, the learning process becomes challenging and requires separating "**signal**" from "**noise**" (**model generalization**)

Data requirements

- This time series can be easily forecast using statistical models, such as exponential smoothing, that explicitly assume e.g. additive seasonality and zero trend

$$\hat{y}_{t+h|t} = \ell_t + s_{t+h-m(k+1)}$$
$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)\ell_{t-1}$$
$$s_t = \gamma(y_t - \ell_{t-1}) + (1-\gamma)s_{t-m}$$

The same is not always true for ML models:

- Even if the trainable parameters are few, it is very difficult in practice to estimate them accurately using a limited number of observations as a train set
- If we force the model to fit the observations available for training, it will most likely over-fit the data and result in poor post-sample performance
- If we reduce the width of the input window and/or consider overlapped windows, we'll increase the size of the train set but **limit the information available** in the input, while **also reducing the diversity** of the samples

Data requirements: *More or longer series*

Forecast?

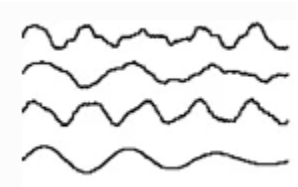
Data limitations may not be an issue when:

A. Working with long series

Such series can **naturally** contribute multiple windows of

- High representativeness
- Moderate diversification

B. Working with sets that consist of multiple series



Such sets can **artificially** contribute multiple windows of

- Moderate representativeness
- High diversification

Local vs. Global models

- Till 2017, forecasting using local models, trained in a *series-by-series fashion*, was considered standard practice in the field
  - ✓ In their paper, Makridakis et al. (2018) prove empirically that such ML models are outperformed by statistical models, even by the simplest ones
  - ✓ Many other similar studies followed, reconfirming the above finding

- In the years that followed, global models, that learn from *multiple series how to predict each series separately*, were gradually established as the new best practice
  - ✓ The **M4** introduced some preliminary cross-learning methods
  - ✓ The **M5** was dominated by global models
  - ✓ Global NN-based models become more popular among academics and practitioners

Creating a train set

So, we know that global models are

- Much faster (a single model is used for predicting N series)
- More accurate (better generalization)

Time to create an **appropriate** train set!

Option 1: Collect more, real series

- What kind of series should we collect in terms of domain and time series features?
- Time consuming – Difficult to identify publicly available series of similar patterns

Option 2: Generate new, artificial series

- How should the series be constructed in terms of time series features and distribution?
- Computationally expensive – Additional analysis required to make sure the patterns of the generated series match those of the real ones

Option 3: Augment the existing series

- How should the series be augmented?
- Computationally cheap and straightforward to apply (in most of the cases)

Data augmentation

Even if we consider multiple series and overlapping windows, the resulting train set may still be insufficiently large, limiting the potential benefits of global models

Augmentation techniques can be used to **artificially** increase the size of the train set and allow common time series patterns to be effectively learned

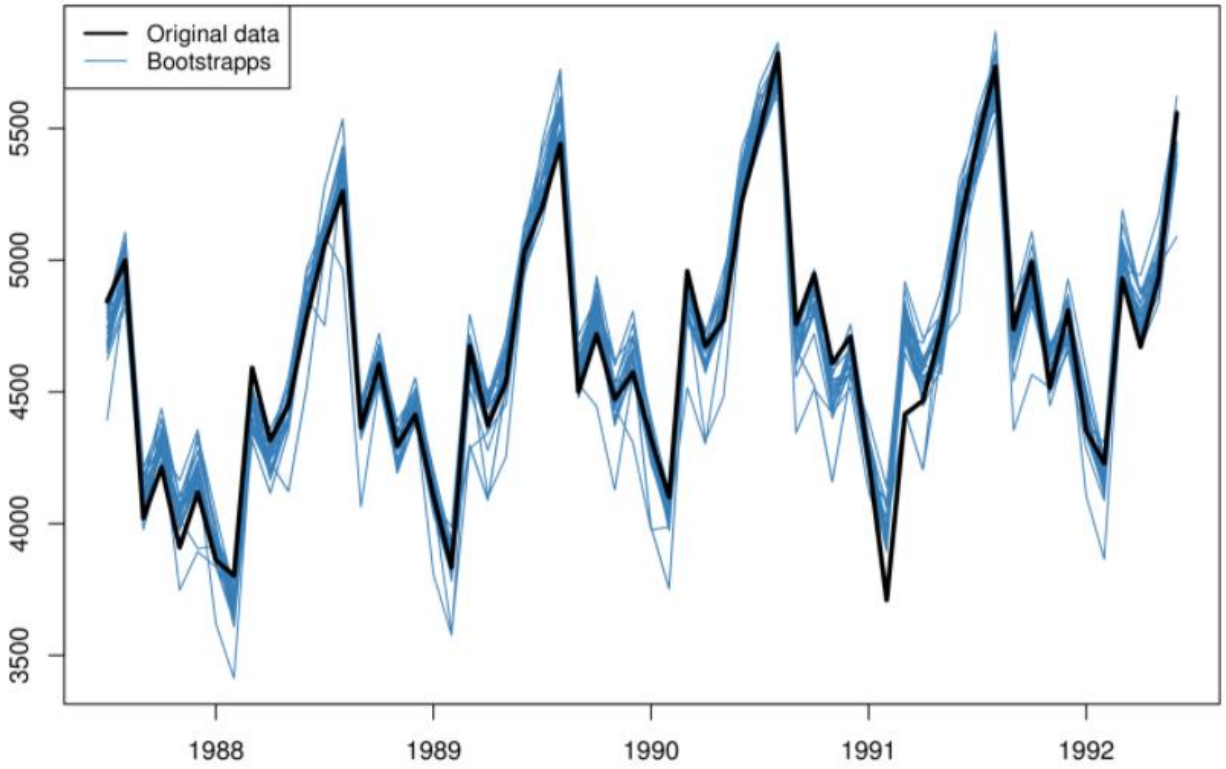
Although these techniques are widely used in many fields of ML, such as image classification and speech recognition, they remain highly unexplored in the area of forecasting and particularly in the context of global forecasting methods (Bandara et al., 2020)

Bootstrapping*

- A **Box-Cox transformation** is applied on the original time series – *stabilization of variance & conversion of multiplicative patterns into additive ones*
- The **STL decomposition** is used to separate the transformed series into the **trend**, **seasonal**, and **remainder components** – *Loess decomposition is used for non-seasonal data*
- The remainder component is **bootstrapped** using moving blocks (**MBB**) towards the creation of a new vector that follows the empirical distribution of the original one
- The bootstrapped remainder vector is added to the components of trend and seasonality (if any)
- An **inverse Box-Cox transformation** is applied to bring the bootstrapped series back to the original scale

**As proposed by Bergmeir et al. (2016)*

Bootstrapping



- First, the *original* series of the data set are bootstrapped
- Then, new look-back windows are extracted from the bootstrapped series and added to the initial train set

Bootstrapping

Advantages:

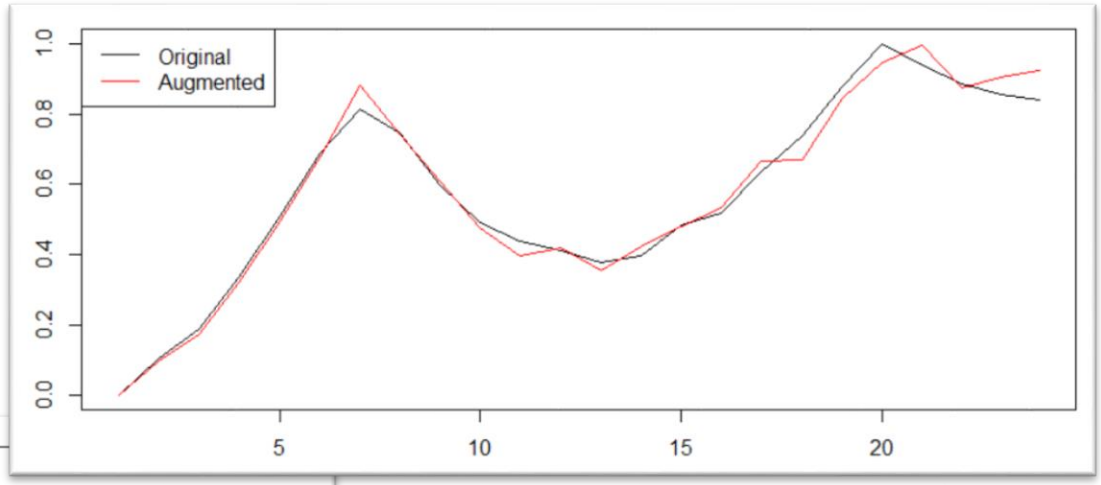
- Extreme values and structural changes are filtered out

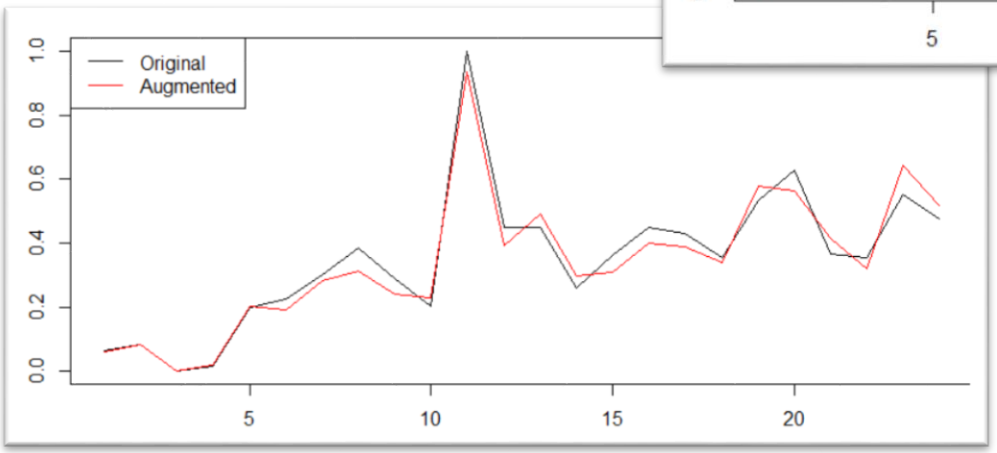
Drawbacks:

- Computationally expensive
- Requires “off-line” computations
- The patterns of the new samples are not significantly different than the existing ones in terms of trend and seasonality

Adding noise

The *n* observations of each look-back window are multiplied by *n* random values which, in this example, are uniformly distributed in [0.9,1.1]





Adding noise

Advantages:

- Computationally cheap
- Easy to implement “on-line”

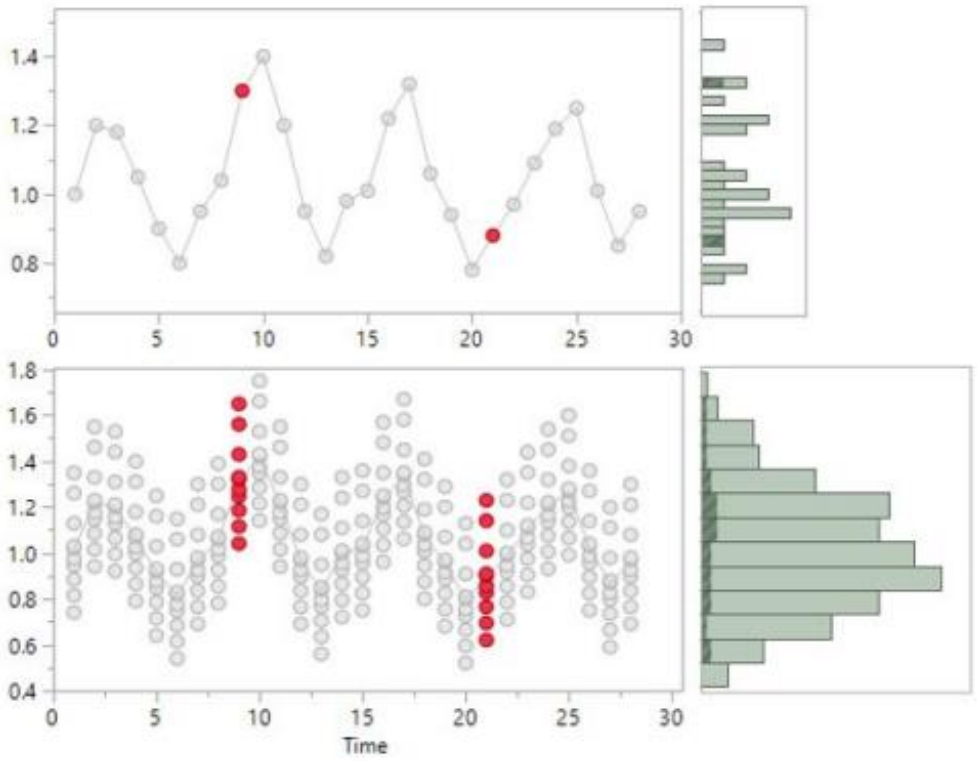
Drawbacks:

The new samples display similar patterns with the existing ones

- Trend and seasonality are practically the same
- Extreme values and structural changes are not filtered out

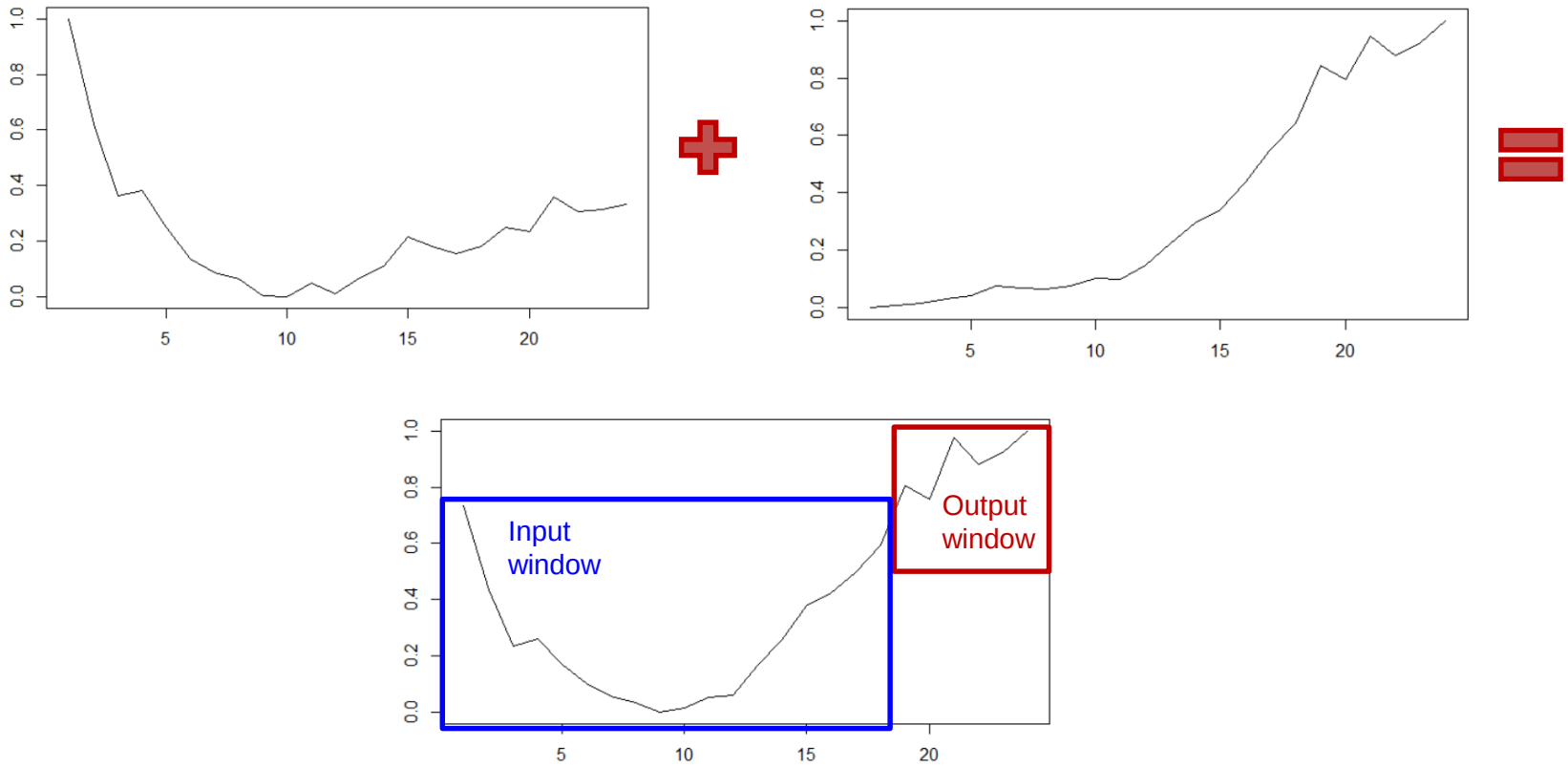
Level shifting

- In a similar fashion, we may simply shift the windows using Gaussian noise
- This approach was used by the winners of the **M5 Uncertainty** competition, Russ Wolfinger and David Lander
- It is computationally cheap and can be easily applied in an "on-line" fashion but the new samples will still display similar patterns with the existing ones
- Becomes **inapplicable when data are scaled before training**



Combining samples

K look-back windows are **randomly selected** and aggregated using an operator of choice. In this example, *K*=2 samples are combined, using the mean operator



Combining samples

Advantages:

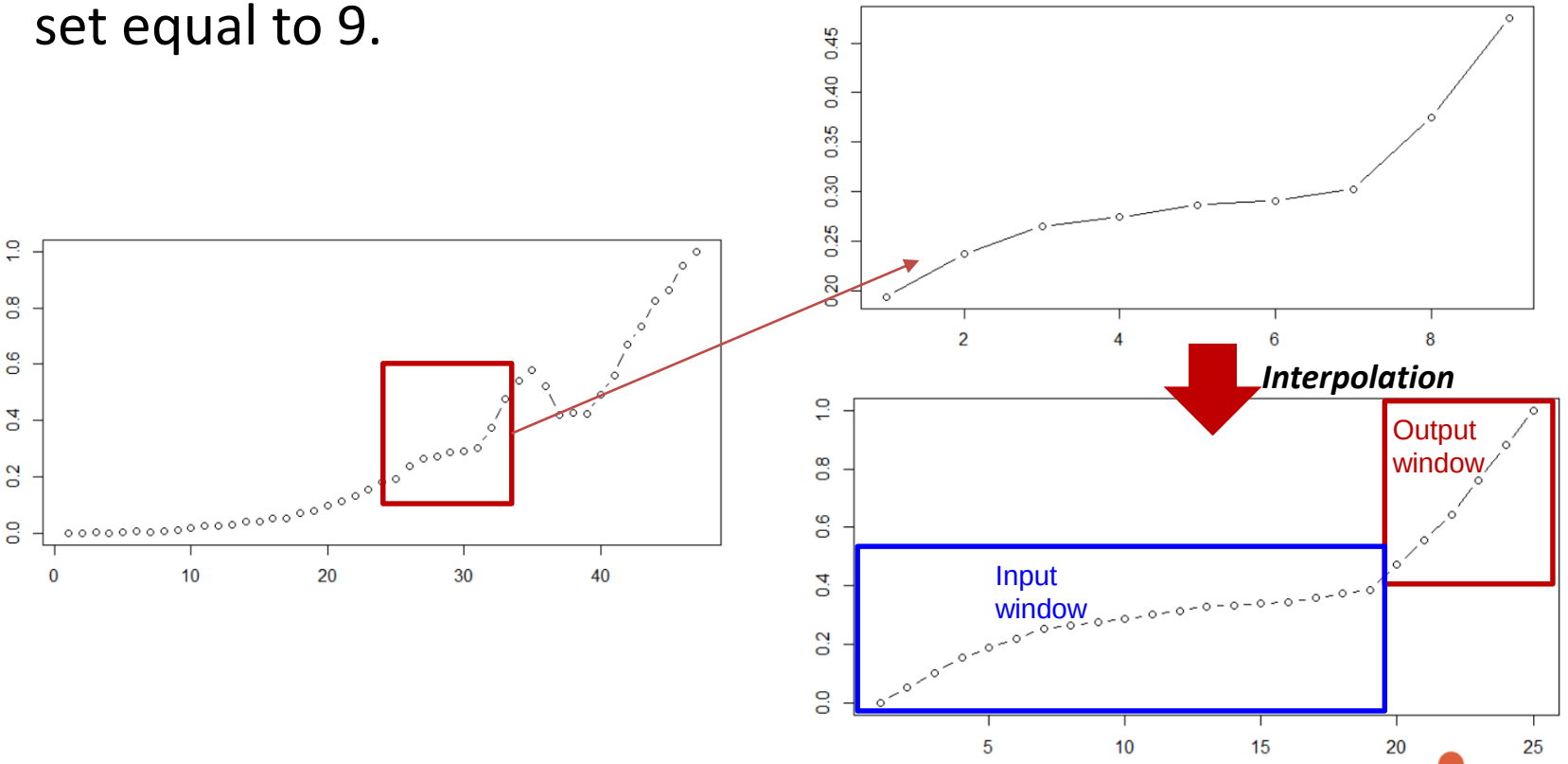
- Computationally cheap
- Easy to implement “on-line”
- The patterns of the new samples are different than the existing ones in terms of trend and seasonality

Drawbacks:

- The number of samples that can be possibly augmented is limited
- As the number of the windows being combined is increased, the diversity of the samples created is significantly decreased

Interpolating samples

From the initial train set, a look-back window is **randomly selected** and a **subsequence** of it, of *m* observations, is extracted. Then, *n-m* observations are added between the *m* observations of the sequence using linear interpolation. In this example, *m* is set equal to 9.



FORECASTING & STRATEGY UNIT

Interpolating samples

Advantages:

- Computationally cheap
- Easy to implement “on-line”
- The patterns of the new samples are different than the existing ones in terms of trend and seasonality

Drawbacks:

- The number of the samples that can be possibly augmented is limited and depends on the initial size of the train set

Time series generation

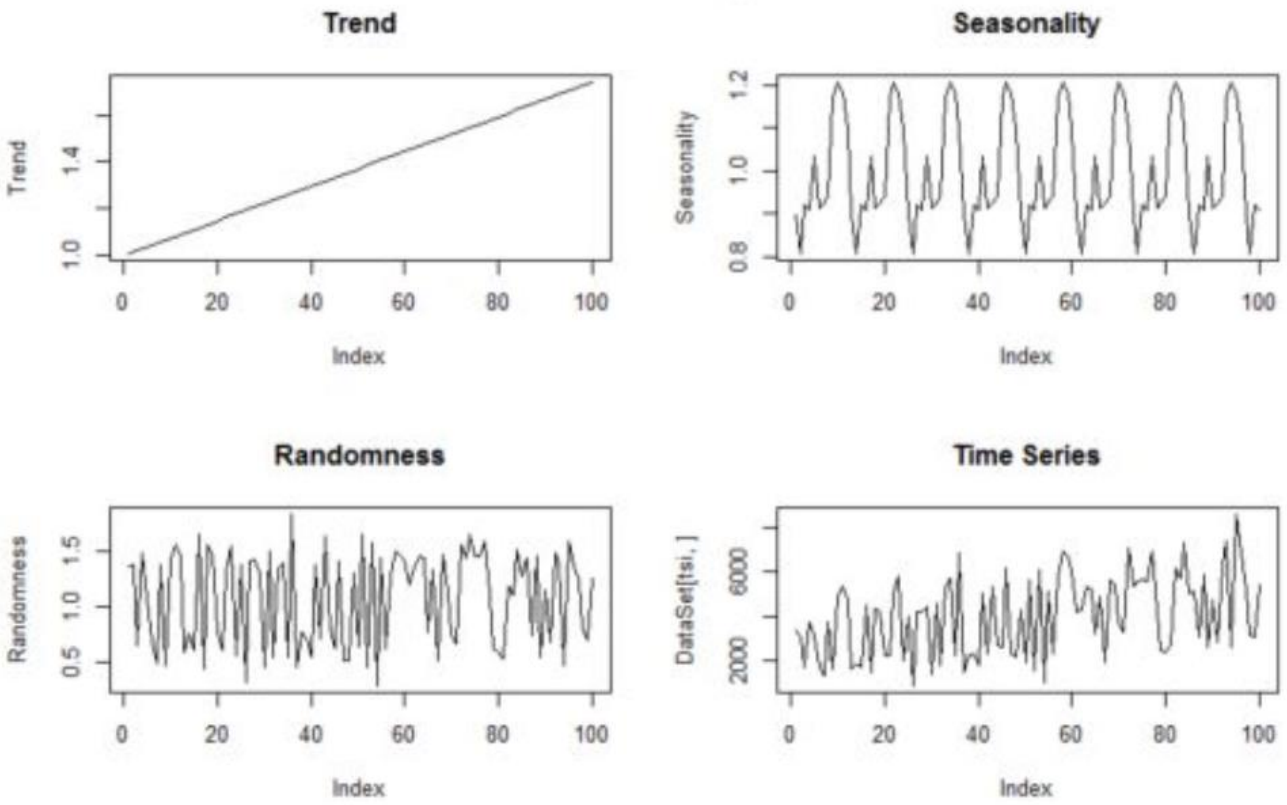
Assume that the series of the train set can be effectively decomposed using the *classical multiplicative method* into three components, as follows:

$$Data = Trend * Seasonality * Randomness$$

Also, assume that the **strength** of the components can be precisely measured using appropriate *statistics*

- Then, we can create a new series whose strength of trend, seasonality and randomness are similar to those of the reference series included in the original train set, as proposed by Petropoulos et al. (2014).

Time series generation



- We can either create a new sample directly or generate a series and then extract multiple look-back windows from it

Time series generation

Advantages:

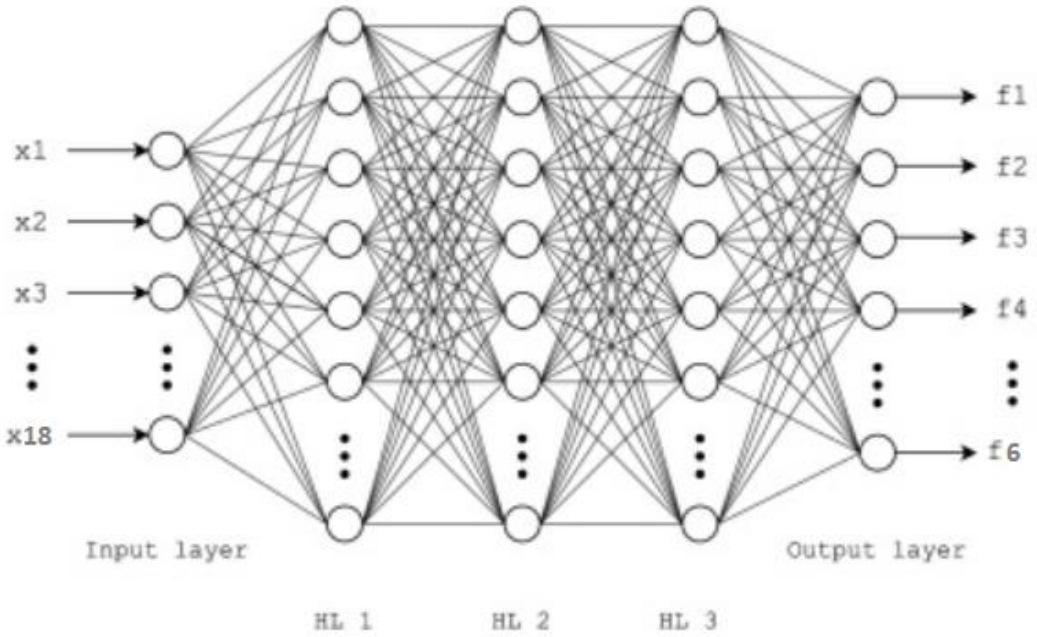
- The patterns of the new samples are significantly different than the existing ones in terms of trend and seasonality

Drawbacks:

- Computationally expensive
- Requires “off-line” computations

Experimental Setup

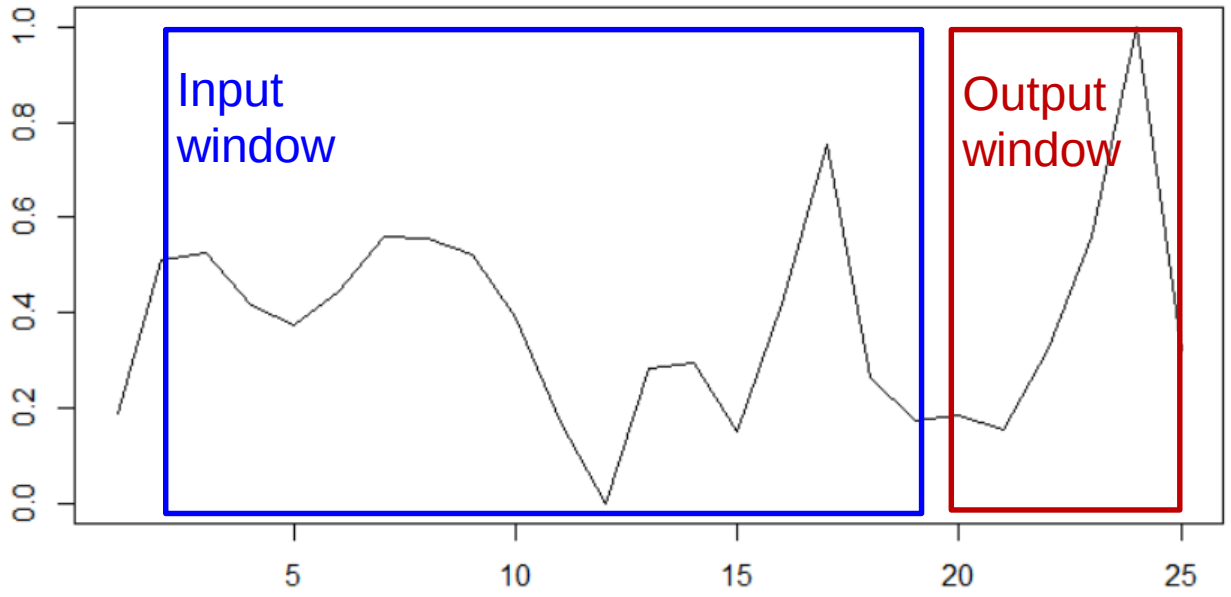
I consider the **23,000 yearly** time series of the **M4** competition and develop a *simple*, yet relatively *accurate* **global** forecasting model, namely a Multilayer Perceptron (**MLP**)



Hyper-parameters *Size of hidden layers:* 27, *Activation function*: RelU, *Optimizer*: Adam, *Learning rate*: 0.001, *Max training epochs*: 500, *Early stopping patience*: 10, *Batch size*: 64, *Loss function*: MAE

Experimental Setup

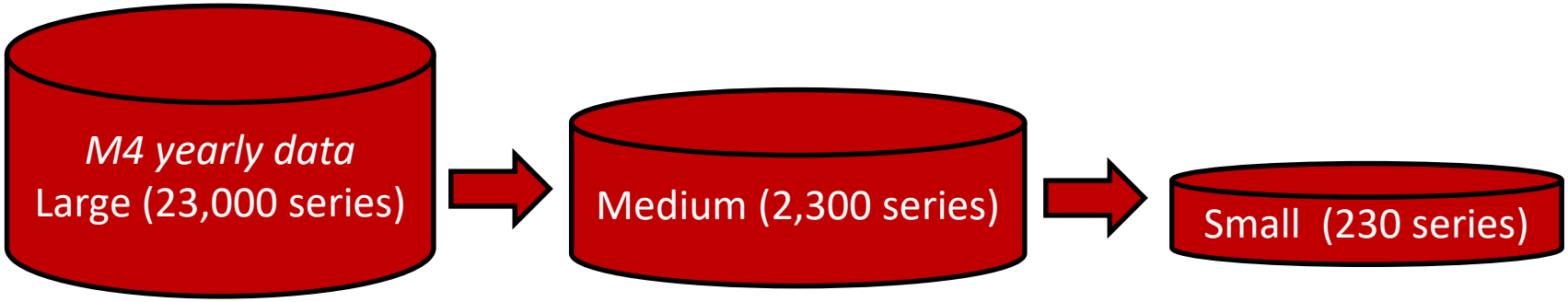
The NN is trained using multiple look-back windows, each one scaled in [0,1] and consisting of 24 observations (*18 input and 6 output points*)



Each series may contribute a **single** or **several windows**. The first approach, using the last window available per series, results in 14,242 samples, while the second, using all possible windows, results in 235,460 samples.

Experimental Setup

In order to evaluate the effect of time series augmentation techniques on **various sizes of data sets**, we consider the following three subsets of series:



Note that the "*Medium*" data set contains all the series included in the "*Small*" one, while the "*Large*" data set all the series included in the two other data sets

- Forecasting performance is evaluated using the Mean Absolute Scaled Error (**MASE**)

Results

We consider two baseline global methods

- **MLP-LW**: *No augmentation – The last look-back window of each series is used for creating the train set*
- **MLP-AW**: *No augmentation – All possible look-back windows of the series are used for creating the train set*

Method	SMALL	MEDIUM	LARGE
Theta	3.58	3.50	3.38
MLP-LW	3.46	3.31	3.18
MLP-AW	3.10	3.08	2.95
Bootstrap	3.11	3.11	2.96
Noise	3.14	3.09	2.97
Combination	**3.03**	3.07	2.95
Interpolation	3.06	**3.06**	**2.93**
Generator	3.14	3.09	2.97

Samples included in the train set

Approach	SMALL	MEDIUM	LARGE
Last windows	134	1,439	14,242
All windows	2,544	24,275	235,460
Augmentation	5,088	48,550	470,920

Making modeling choices

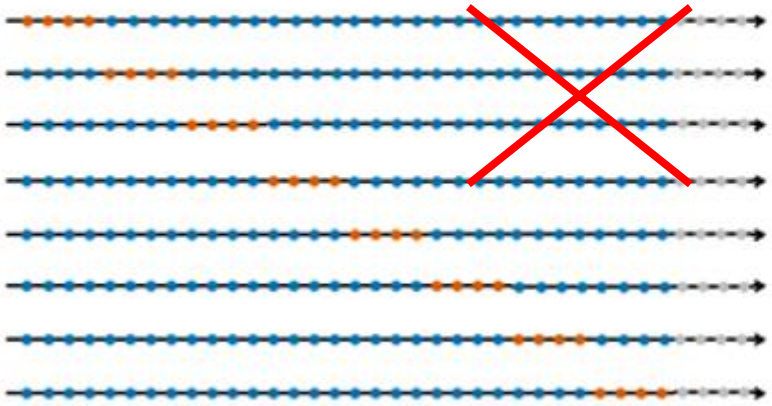
Suppose that, through augmentation, the train set is now sufficiently large to train a ML model. How can we safely:

- Determine the optimal values of model's hyperparameters?
- Define the optimal size of the input window?
- Select the features that should be used as input (lags or external variables)?
- Choose whether multi-step-ahead forecasts should be produced directly or iteratively?
- Decide how often the model should be re-trained (or re-optimized)?
- Conclude whether ensembling should take place (and how exactly) or not?

➢ We need a robust **validation scheme** that closely mimics the forecasting process and provides results that are indicative of the model's post-sample performance

Standard time-series cross-validation

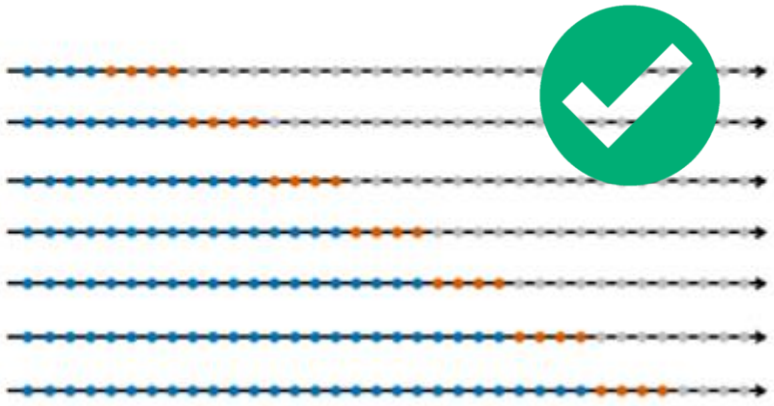
K-fold validation



Standard *k*-fold validation is inappropriate since:

- The train and the test sets are not chronologically aligned
- Data leakage is evident and can negatively affect the representativeness of the results

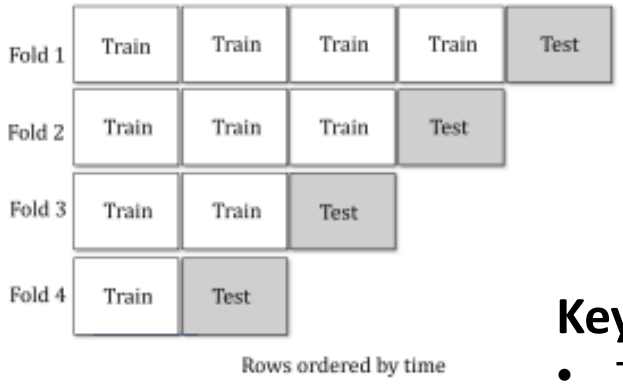
Rolling-origin evaluations



Closely mimics the forecasting process
We have to define:

- Whether the size of the train set is fixed or continuously expanding
- How often the forecast origin is updated
- Whether the folds are overlapping or not

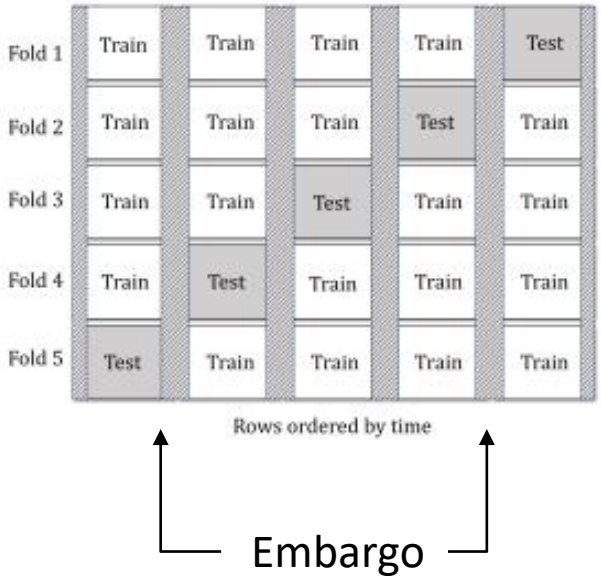
Standard time-series cross-validation



Key issues:

- The earliest folds are completely unrepresented in any performance measurements AND receive the highest weight in training
- Other early folds are fit on very limited training data
- More valuable data goes unused or underweighted

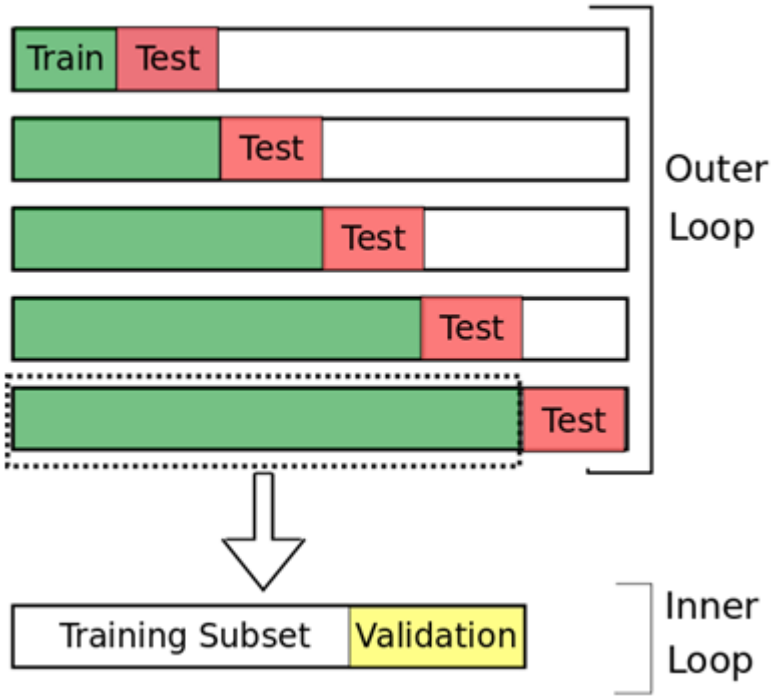
Purged (embargo) k-fold validation



- If the data are "reasonably" limited, future data can be safely used to forecast the past
- Some of the data points near the boundaries of the folds are removed
- Full, balanced, and robust training sets
- Each fold is isolated and data leakage is mitigated or completely prevented

Nested (double) cross-validation

Treat “model selection and optimization” as part of the modelling approach itself and evaluate the alternatives within the broader cross-validation procedure.



- Instead of using cross-validation to select the best setup across the *k* folds, the best setup is selected for each of the *k* folds separately and evaluated accordingly
- Parameter search cannot overfit the data as it is only exposed to a subset of the data originally available, provided by the outer cross-validation procedure
  - ✓ Reduced risk of overfitting
  - ✓ Less biased estimates on model’s performance
  - ✓ Increased computational time

Takeaways

- Fine-tune hyper-parameters using proper CV strategies (e.g. purged and nested k-fold validation)
- When large data sets are already available for training, augmentation may insignificantly affect forecasting performance – Yet, it is highly recommended for small data sets
- Augmentation must be carefully performed so that the samples created are
  - Diverse in terms of patterns
  - Representative of the series to be forecast
- Traditional, time-intensive augmentation techniques, such as bootstrapping and time series generation, should be replaced by innovative, computational cheaper ones that can be applied in an on-line fashion

References

- Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y. & Bergmeir, C. (2021). Improving the Accuracy of Global Forecasting Models using Time Series Data Augmentation. *Pattern Recognition*, 120, 108148
- Bergmeir, C., Hyndman, R.-J. &Benítez, J.-M. (2016) Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation. *International Journal of Forecasting,* 32(2), 303-312
- Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye Q. & Liu T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA
- Makridakis S., Spiliotis E., Assimakopoulos V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13, 1-26
- Oreshkin B. N., Carpov D., Chapados N. & Bengio Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *1905.10437*
- Petropoulos, F., Makridakis, S., Assimakopoulos, V. & Nikolopoulos, K. (2014). 'Horses for Courses' in demand forecasting. *European Journal of Operational Research*, 237(1), 152-163
- Salinas D., Flunkert V., Gasthaus J. & Januschowski T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36, 3, 1181-1191

Questions?

 spiliotis@fsu.gr

 www.fsu.gr