Probabilistic and Summary Forecasting

... and some Pitfalls in Forecasting Practice

Christoph Bergmeir[1], Slawek Smyl[2]

[1]University of Granada, Spain
https://www.cbergmeir.com
[2]Walmart Labs. My presentation, comments and opinions are provided in my personal capacity and not as a representative of Walmart. They do not reflect the views of Walmart and are not endorsed by Walmart.

Outline

- Pitfalls in forecasting practice
- Probabilistic forecasting
- Summary forecasting

Motivation

- With some colleagues I had a recent paper about Forecast Evaluation for Data Scientists:

H Hewamalage, K Ackermann, and C Bergmeir. "Forecast evaluation for data scientists: common pitfalls and best practices." Data Mining and Knowledge Discovery 37.2 (2023): 788-832.

- Also a paper in the Foresight practitioner journal forthcoming

Motivation (2)

- Our main claim in that paper is that many machine learning academics don't know how to properly evaluate forecasts
- I'm seeing this continuously when reviewing papers, and also reading published papers
- This becomes even more relevant in practice as nowadays often forecasting is done by "Data Scientists" that may not have any specialized training in forecasting, but in ML and Stats

We identified the following problems:

1. Datasets too small / irrelevant
2. Data leakage
3. Not using adequate benchmarks
4. Wrongly used or ad-hoc evaluation measures
5. Reliance on forecast plots for other things than sanity-checking
6. Assumption that a forecast needs to be a realistic scenario

Problem 1: Datasets too small / irrelevant

- "If a method has more words in its name than it has time series it is tested on, be sceptical!'' (adapted from Goodwin (2011))
- "We have tested our method on 3 stock market time series''
- Clearly more series could be easily available
- The authors clearly don't care about their particular application

Problem 2: Data leakage

- Not trivial to avoid in forecasting
- Rolling origin: data travels from test to training set
- Hard to completely separate training from evaluation code base

Problem 2: Data leakage (2)

- Usually in ML: Don't normalize (calculate mean, variance) before splitting into training and test set
- In forecasting also problematic:
  - Normalization
  - Any form of smoothing or decomposition
  - Seasonal decomposition: STL, etc.
  - Feature extraction
  - Empirical mode decomposition

- Unaligned datasets (e.g., M3, M4 datasets): Some series may contain information about the future of other series

Problem 3: Not using adequate benchmarks

- Always use simple benchmarks!
- Especially with high-noise series such as asset/stock prices
- There is a pocket of research papers in ML proposing new transformer architectures (Informer, Autoformer, Robformer, ETSformer, etc.)
- Many of them use an exchange rate dataset where their task is to predict 720 days out, without any covariates.
- We show in the paper they all loose against naive, on that dataset

Problem 4: Wrongly used or ad-hoc evaluation measures

- Measures on the original scale: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)
- They work well, but if series are on very different scales, some series can dominate the evaluation
- We need a scale-free measure.
- We need to divide by "something' '
- After 30 years of research in forecasting, we have still not found this "something' ' in a way that it works under any possible non-stationarity and series characteristics.
- There are over 40 error measures proposed in the literature that we are aware of (sMAPE, MASE, etc.)

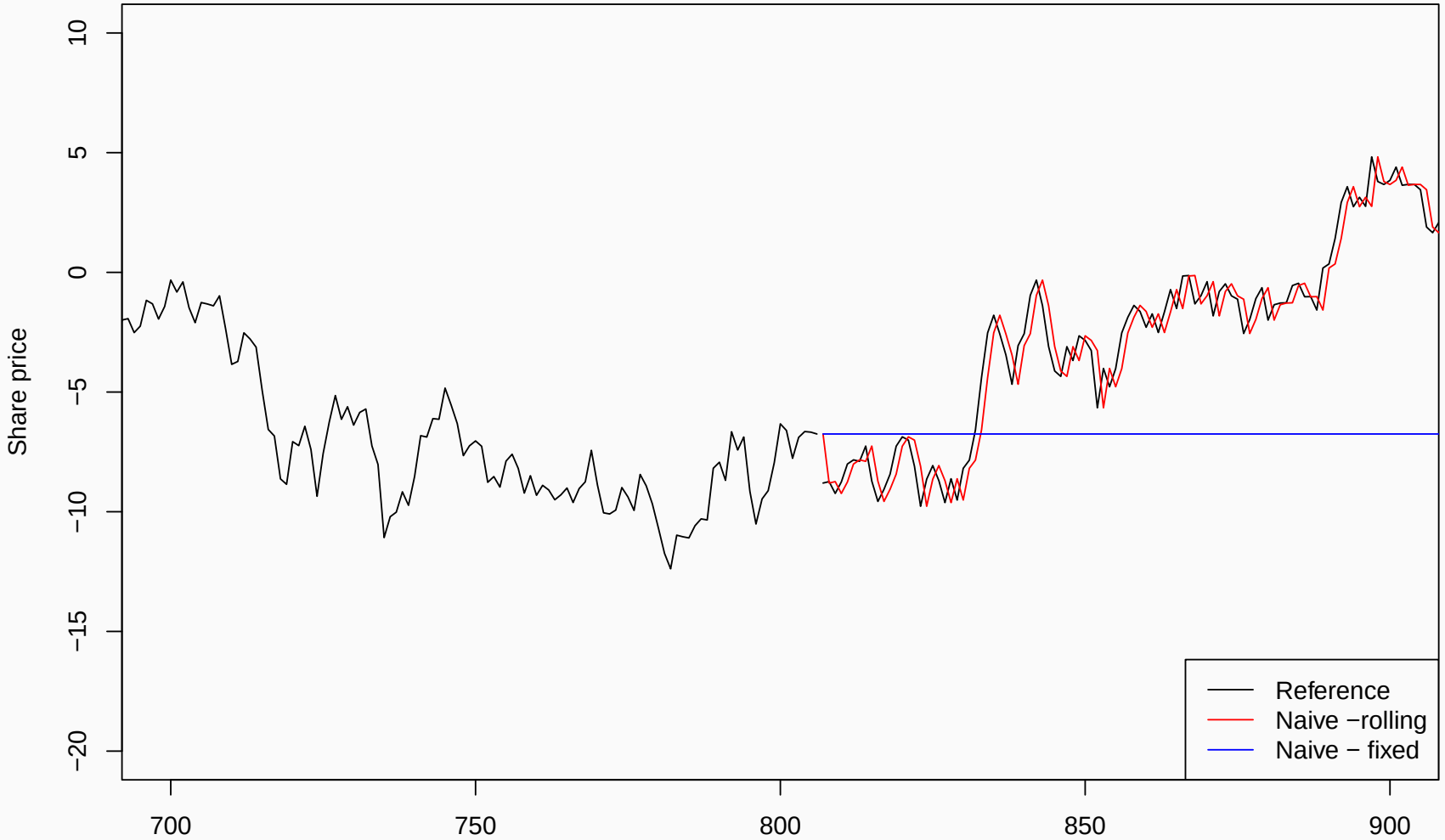
Problem 4: Wrongly used or ad-hoc evaluation measures (2)

–> Do not invent your own measure, it will be more difficult than you think

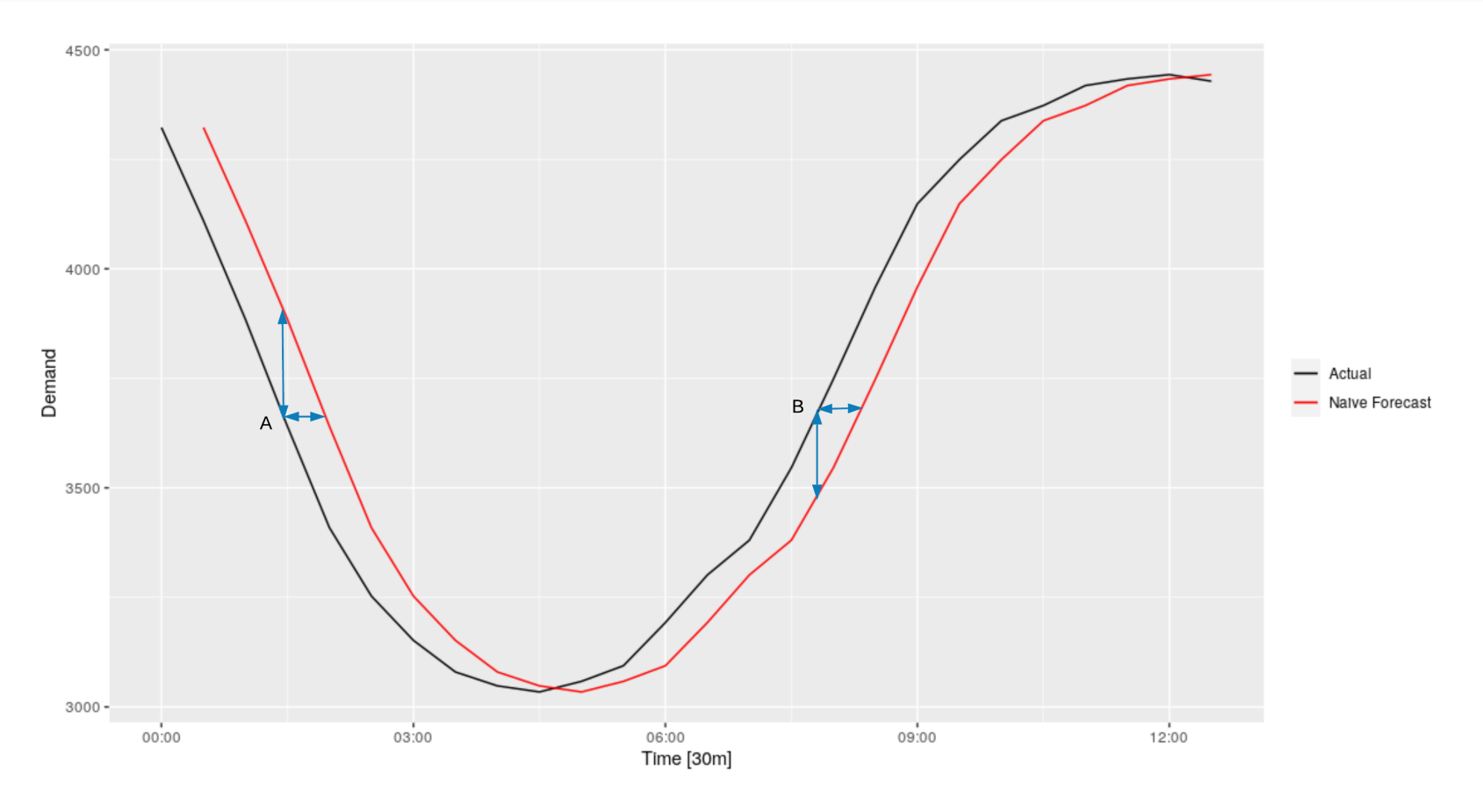
–> It depends on the characteristics of your data which measure will be adequate

Problem 5: Reliance on forecast plots for other things than sanity-checking

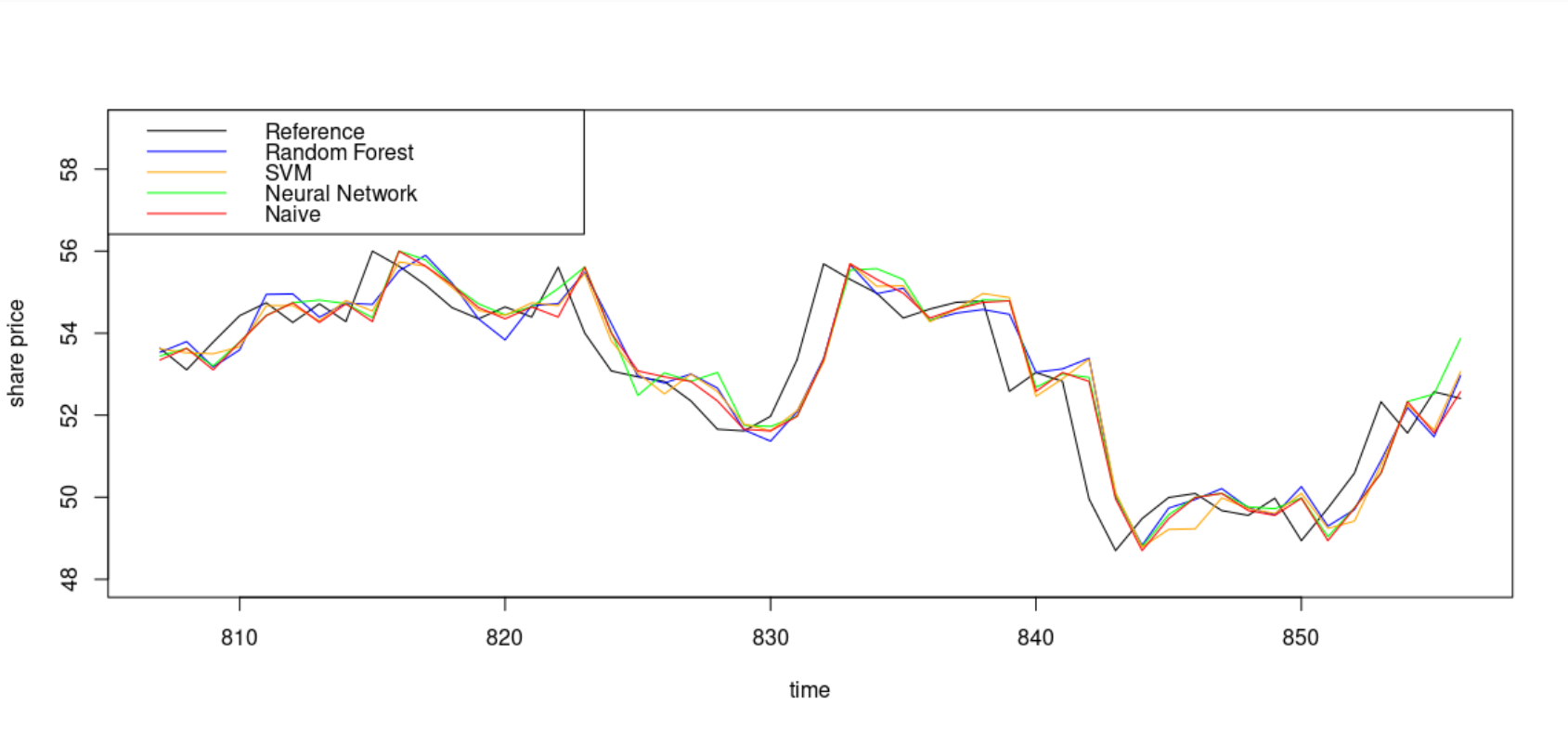
Fixed and rolling origin evaluation are very different



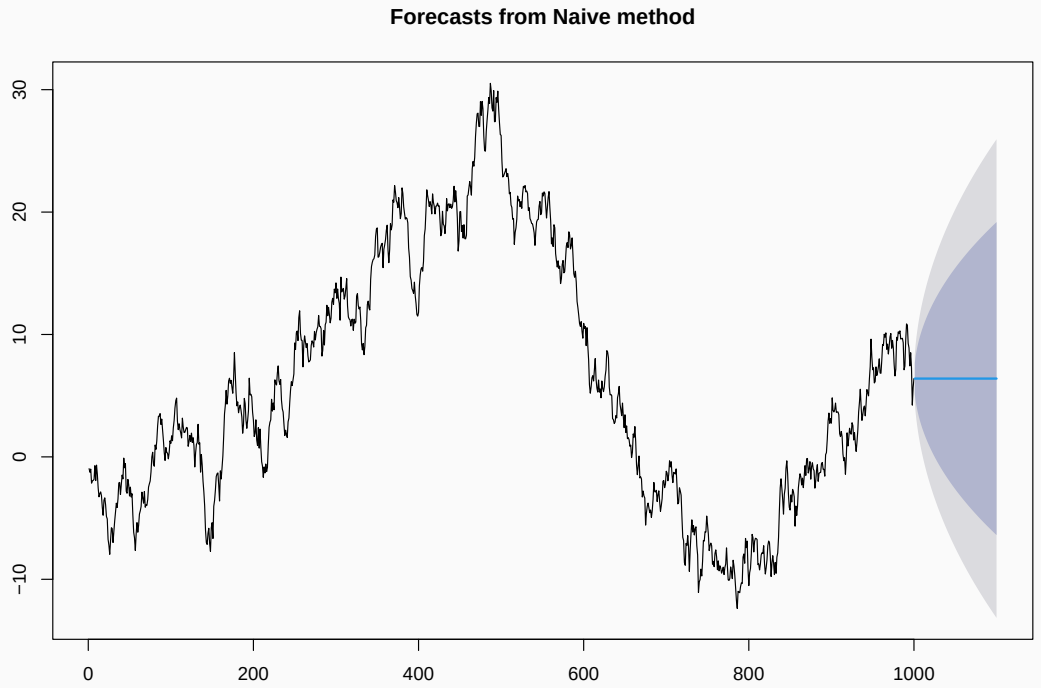
Plots for rolling origin are deceiving for small horizons



Which one is best?



Problem 6: Assumption that a forecast needs to be realistic

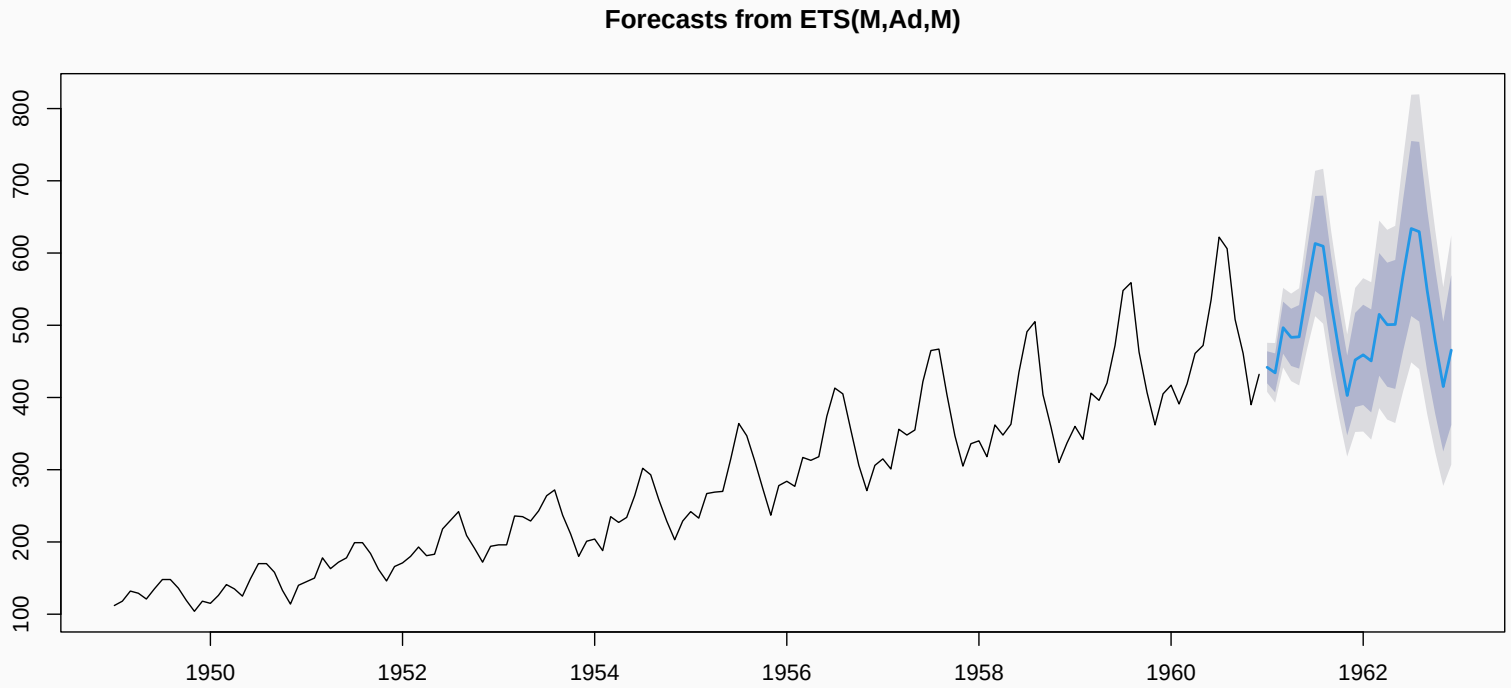


Will the share price just remain constant in the future? –> No!

Is a constant value the best forecast (that minimizes the error) –> Yes!

Let's take a step back...what is a point forecast?

- All forecasts are wrong, but some are useful. (to quote Tim and Jan, in a modified quote of George E.P. Box)
- The future always holds a lot of uncertainty/noise
- We reflect this with probabilistic forecasting, where we forecast a distribution



What is a point forecast? (2)

- A point forecast is a summary statistic of this distribution.
- For example, point forecast can be mean, median, mode of the distribution
- When we say we want the point forecast to be the "most likely future value'', it is the mode
- The mean is in general not the most likely future value, but the expectation.
- We get it from multiplying each possible value by its corresponding probability and summing them up
- It is therefore a blend of different scenarios!
- Highly intermittent series: "most likely future value'' will always be a zero

What is a point forecast? (3)

- To truly understand what we do when we do point forecasting, we need to understand probabilistic forecasting
- Some authors (Stephan Kolassa, Tim Januschowski) have argued that we should always do probabilistic forecasting
- So let's look at probabilistic forecasting next

Main ways for probabilistic forecasting:

- use analytical prediction intervals
- bootstrapping
- using a Bayesian model (MCMC sampling)
- forecast the parameters of a distribution
- use quantile regression (pinball loss)
- determine uncertainty empirically through backtesting, conformal prediction
- Levelset approach (Hasson et al. 2021)

Quantile regression (pinball loss)

- implemented in, e.g., Wen et al. (2017)
- no distribution assumptions need to be made; therewith better if a lot of data are available
- fast to compute and easy to implement
- only certain quantiles can be obtained, not the full distribution
- in practice, often 5 or 7 quantiles are enough anyway
- can interpolate between quantiles to get full distribution (Gasthaus et al. 2019)
- we can fit different quantiles at the same time, e.g. with a neural network, with losses for each quantile in the loss function

Forecasting any quantile

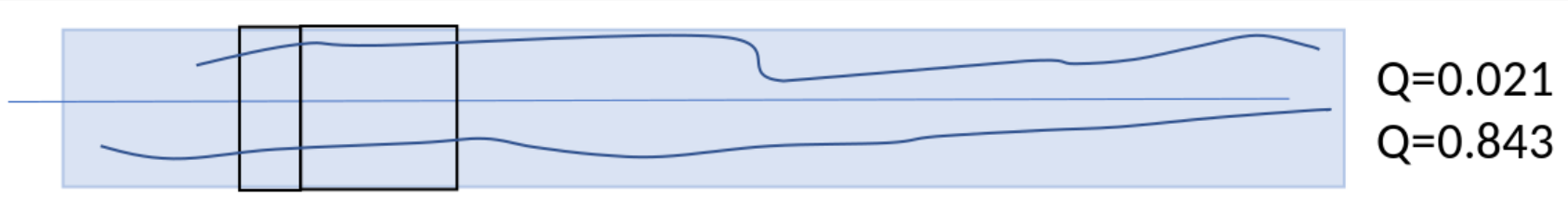
- Often the same forecast is used by several recipients with different needs. Some care only about high quantiles, others pay attention mainly to P50 or avg value, etc. So, the list of needed quantiles tends to grow.
- High quantiles (e.g. between .97 and 1) may be not easy to interpolate.
- Additionally, in some business contexts, e.g. to allow interactive exploration of scenarios, it would be great to be able to generate quickly any quantile on demand.
- So, ideally, we would like to train a system to predict (quickly) any quantile, but a particular value requested during serving (inference)
- The idea came from Gouttes, A et al., "Probabilistic Time Series Forecasting with Implicit Quantile Networks", https://arxiv.org/abs/2107.03743, although what follows is quite a bit modified.

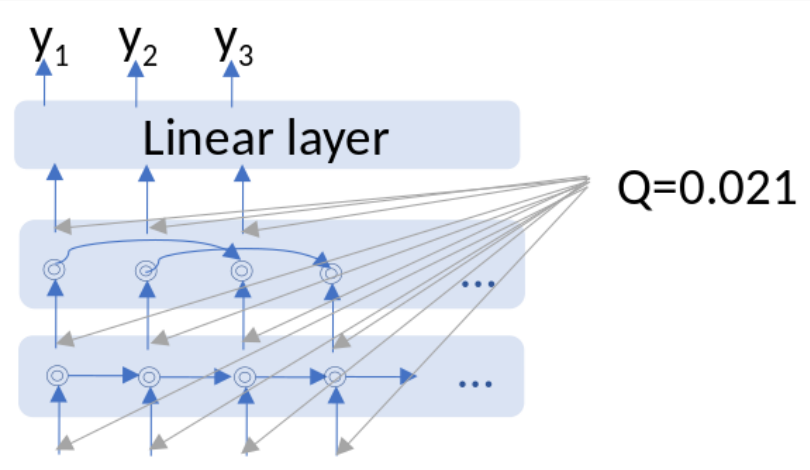
Training and serving

- In ML systems, there are two distinct phases: training and serving. Training takes usually between several hours to several days, but serving can be very fast, real time.
- Training adjusts NN weights or, more generally, the ML system parameters. At the end of training we save them.
- A separate serving program loads them into memory and waits for a new input: list of series and quantiles.
- Then, forecasting is very quick, as no optimization takes place, a matter of milliseconds per series/quantile, especially when doing it for a large batches of series*quantiles.

Training

Allocate to each series in a batch a randomly chosen quantile. Append it to the preprocessed input

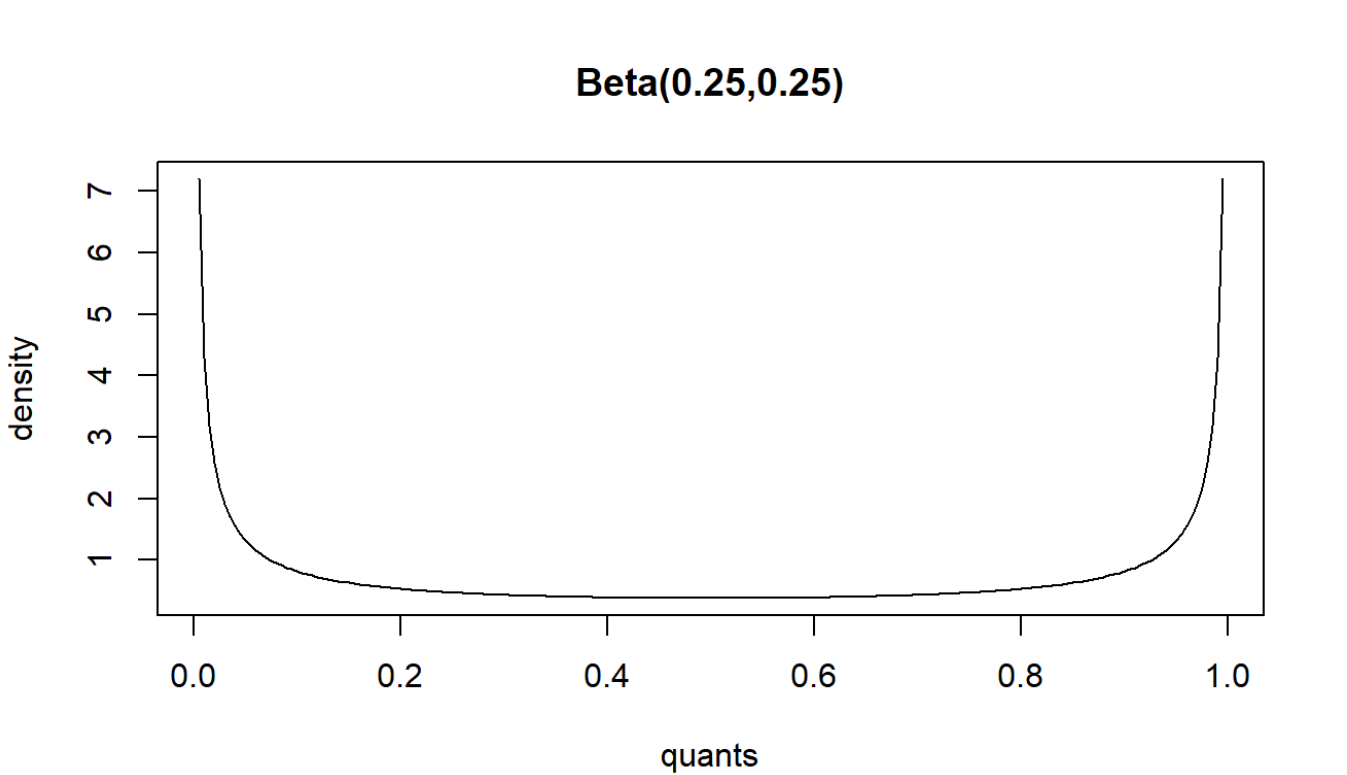




- (An RNN, single series view) Add also the same quantile to the intermediary outputs of the RNN's layers
- Loss function= quantile loss for the chosen quantile

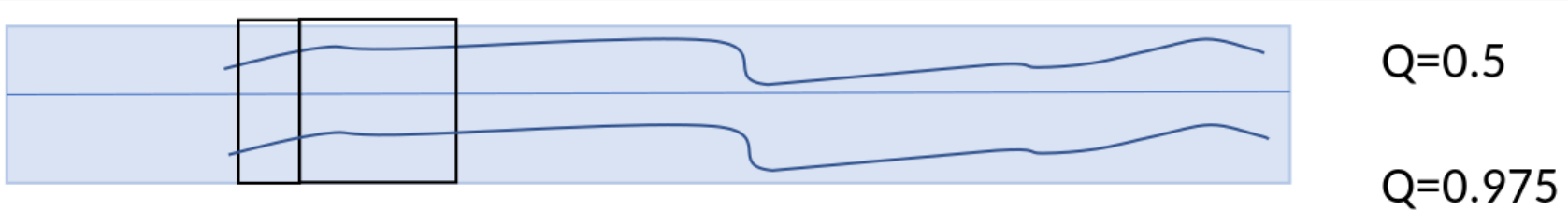
Training: non-uniform sampling

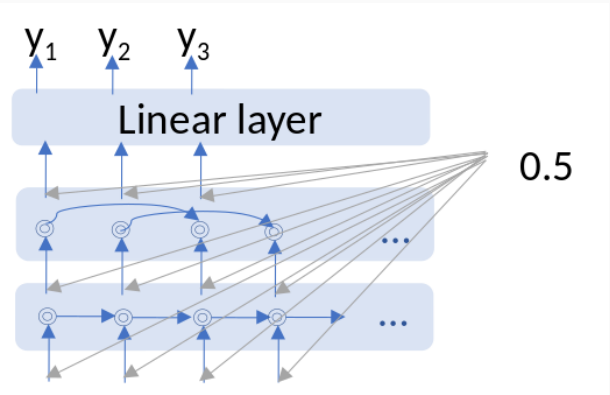
Because quantile function is likely to change quickly as we approach 0 and 1, and we often like to be precise there, the "edges" should be sampled more frequently, e.g., using symmetrical Beta distribution with parameters smaller than 1, e.g., Beta(0.3,0.3)



Serving

During inference, create batch e.g. composed of the same series, but with different quantiles





- Append the same quantile to the intermediary outputs of the RNN's layers
- **Interestingly, the center (q=0.5) forecast can be as good as in a system dedicated only to this point forecast. Occasionally multi-task learning works.**

Forecasting Any Quantile - Summary

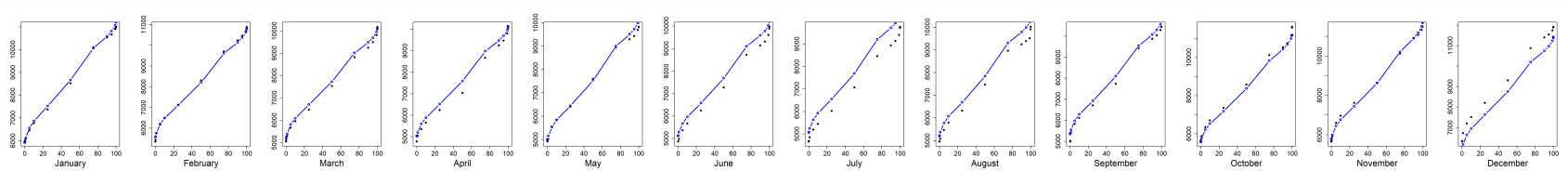
- It is a meta-algorithm, possible to be implemented in several NNs architectures, perhaps some other ML models.
- During training generate quantiles randomly, and apply the loss calculated for this quantile.
- During serving, provide the serving program with a list of desired quantiles. This part is fast, milliseconds, so it can be used in interactive scenarios.

Summary forecasts

- The forecasted value can be a statistical **summary** of the values in a period, e.g. average daily temperature based on minute-level data. And then, we may be interested in not just an average, but also min, max, some quantiles of it, and perhaps some other derived variables, based on a more frequent underlying data.
- Summary forecasts are unusual, seldom, except averages, discussed in the forecasting literature but they are right choice when high-frequency data is available, but decisions are made in low-frequency domain, and this is a common situation in business.
- The most flexible is to forecast a number of quantiles

Summary forecasts (cont'd)

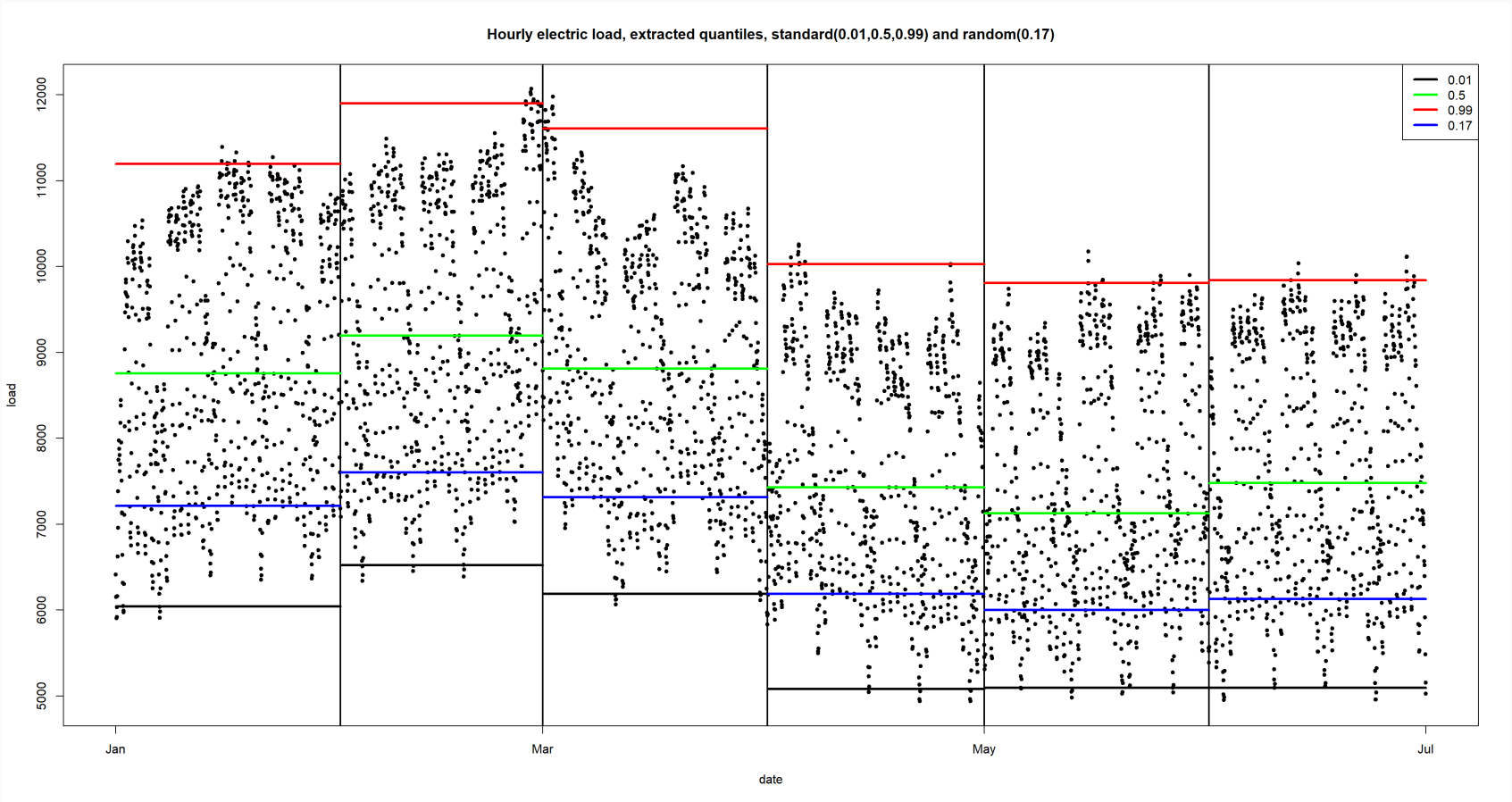
- Consider 12 months-ahead forecast at hourly grain (365*24). It would be very difficult to do it, and even if we did, the **users would aggregate it anyway**. E.g. a capacity planner will care about monthly or quarterly aggregates, like max, avg, and P99 of the forecast, and ignore everything else.
- Instead, we can predict a number of quantiles per month. Quantiles have good business meaning – they measure risk (what percentage of time a variable will exceed a particular threshold). Also with enough quantiles, one can calculate any other derived variable.



Input preprocessing is robust

- Hi-freq data is not a good input for the quantile predictions, obviously is too long.
- Also, as a rule, part of input should be related to the output. So, if we are predicting quantiles in, say, monthly periods, input should also contain them.
- Missing values in time series is a frequent phenomenon, especially for high-freq data. For tree-based algorithms they are not a (big) problem, as one can replace them with an impossible value (say -1 for counts), but in NN systems, the missing values in inputs should be imputed, which is not always straightforward, and always distorts reality.
- Quantile calculations are robust, even if a large portion of data is missing

Input preprocessing is robust (cont'd)

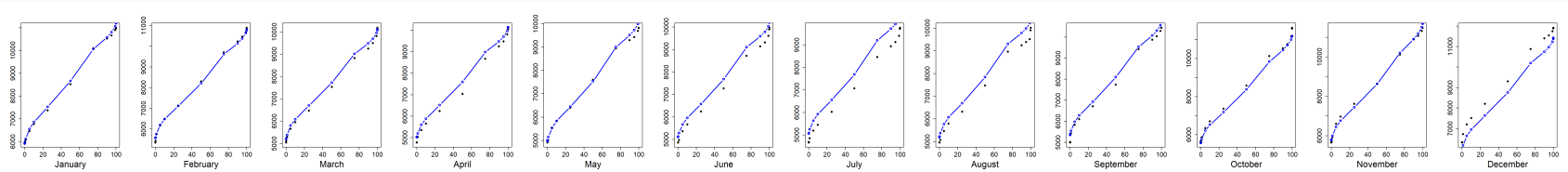


Hierarchical Reconciliation of Summary Forecasts

- Can be also done, it was a subject of last year ISF talk by Alaleh Razmjoo and I.
- We presented two methods
  1. Based on extending the summation matrix, into one with sums !=1, and then following various approaches of Optimal reconciliation family
  2. Minimizing adjustment needed to the forecasts so historical relationship between the total and components would be maintained.

Summary Forecasting - Summary

- When the underlying data is of high-frequency and the horizon is long but the decision making happens in low-frequency, it makes sense to forecast summaries in the low-frequency periods.
- The learning is much faster, a lot of technical problems disappear, and we provide to users what they really need.
- A NN system can be trained to predict any requested quantile, requested at serving (inference) time.



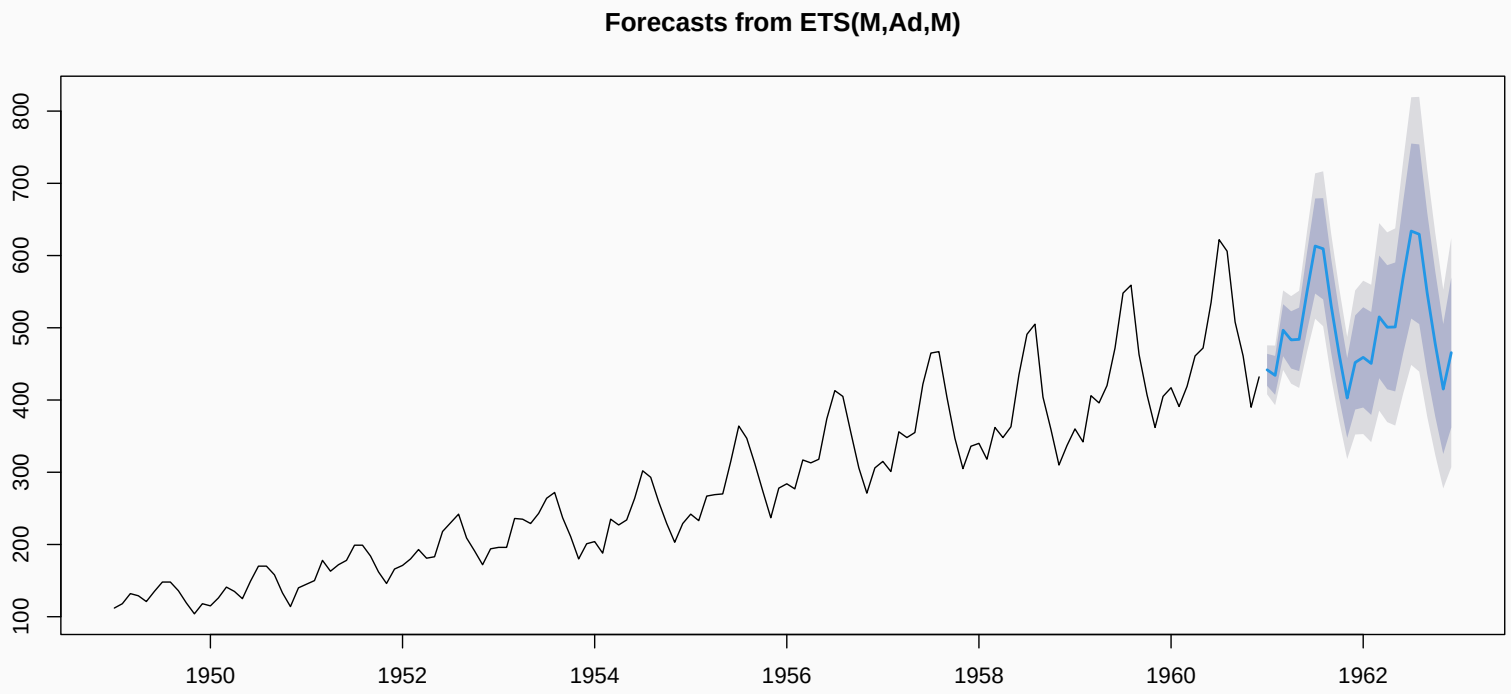
Thank You

https://www.cbergmeir.com

bergmeir@ugr.es, slawek.smyl@walmart.com

Use analytical prediction intervals

- Possible for some (well-understood) models
- Usually assume normally-distributed errors
- ETS, ARIMA do this (see Hyndman and Athanasopoulos (2018))
- intervals tend to be too narrow (Bermudez, Segura, and Vercher 2010)



Simulation and bootstrapping

- slow
- intervals can also be too narrow if, e.g., they only consider parameter uncertainty
- either need to bootstrap residuals (from an additional validation set) or assume a distribution
- then simulate forecasting paths by feeding the generated/bootstrapped values back into the model
- see Hyndman and Athanasopoulos (2018), Section "Neural Networks''

MCMC sampling

- needs to be a Bayesian model
- Examples: LGT (Smyl et al. 2019), Orbit (Ng et al. 2020), Bayesian ETS (Bermudez, Segura, and Vercher 2010)
- slow

Determine uncertainty empirically through backtesting

- often used by companies in practice
- leads to more realistic prediction intervals
- needs a lot of past data and rolling origin forecasts (large validation sets)
- Theoretical underpinning in conformal prediction

Forecast the parameters of a distribution

- e.g., assuming a normal distribution: μ, σ
- DeepAR (Salinas et al. 2019): Normal distribution and negative binomial distribution
- NGBoost (Duan et al. 2020)
- Have to assume a certain distribution
- Good if we have limited amounts of data, or knowledge of the distribution

References

Bermudez, J. D., J. V. Segura, and E. Vercher. 2010. "Bayesian Forecasting with the Holt-Winters Model." *Journal of the Operational Research Society* 61 (1): 164–71.

Duan, Tony, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. 2020. "Ngboost: Natural Gradient Boosting for Probabilistic Prediction." In *International Conference on Machine Learning*, 2690–700. PMLR.

Gasthaus, Jan, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. 2019. "Probabilistic Forecasting with Spline Quantile Function Rnns." In *The 22nd International Conference on Artificial Intelligence and Statistics*,