ARTICLE IN PRESS

International Journal of Forecasting xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Forecasting

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

Forecasting with trees

Tim Januschowski [a,*], Yuyang Wang [a], Kari Torkkola [a], Timo Erkkilä [b], Hilaf Hasson [a], Jan Gasthaus [a]

[a] Amazon Research, Charlottenstr. 4 10969, Berlin, Germany
[b] Veracell Oy, Polttimonkatu 4, Tampere, Finland

ARTICLE INFO

Keywords:
Random forests
Probabilistic forecasting
Gradient Boosted Trees
Global forecasting models
Deep Learning

ABSTRACT

The prevalence of approaches based on gradient boosted trees among the top contestants in the M5 competition is potentially the most eye-catching result. Tree-based methods out-shone other solutions, in particular deep learning-based solutions. The winners in both tracks of the M5 competition heavily relied on them. This prevalence is even more remarkable given the dominance of other methods in the literature and the M4 competition. This article tries to explain why tree-based methods were so widely used in the M5 competition. We see possibilities for future improvements of tree-based models and then distill some learnings for other approaches, including but not limited to neural networks.

1. Introduction

The top ranks in the M5 competition—both in the accuracy and the uncertainty track—were dominated by entries that used tree-based machine learning methods (*trees* for short).[1] In particular, gradient-boosted trees as implemented in the LightGBM framework (Ke et al., 2017) were featured prominently as a core building block in many entries. This dominance of tree-based methods—rather than classical forecasting techniques as in the M3 competition, or deep learning-based approaches (and hybrids) as in the M4 competition—is one of the most eye-catching outcomes of the M5 competition. It certainly surprised us based on our own experience from forecasting practice. We saw deep learning-based approaches as the best-in-class models on data sets similar to those used in the M5 competition. Additionally, the field has witnessed increasing research on deep learning-based forecasting models over the last five years. The excellent performance of these approaches has been demonstrated on a diverse set of tasks, with the M4 competition where the winners in both tracks heavily relied on deep learning being the most prominent example.

In this article, we dissect our initial feeling of surprise regarding the success of tree-based methods in the competition—in particular in comparison to neural networks. We first provide reasons why we believe tree-based methods were so widely and effectively used in the competition, based on observations from the top entries and our own experience with similar techniques (Section 2). Our central observation is that tree-based methods can effectively and robustly be used as black-box learners. However, current deep learning-based techniques and their implementations for forecasting have not yet reached the same level of robustness and require skillful model adjustments to achieve competitive performance. We noted that contestants successfully used tree-based models as black-boxes. This has been used in

* Corresponding author.
E-mail addresses: tjnsch@amazon.com (T. Januschowski), yuyawang@amazon.com (Y. Wang), karito@amazon.com (K. Torkkola), timo.erkkila@veracell.com (T. Erkkilä), hilaf@amazon.com (H. Hasson), gasthaus@amazon.com (J. Gasthaus).

[1] Decision trees, random forests, and gradient boosted trees are in this class of models. Throughout our article, we assume familiarity with these models and refer to standard Machine Learning textbooks such as Hastie, Tibshirani, and Friedman (2009) for background.

https://doi.org/10.1016/j.ijforecast.2021.10.004

Please cite this article as: T. Januschowski, Y. Wang, K. Torkkola et al., Forecasting with trees. International Journal of Forecasting (2021), https://doi.org/10.1016/j.ijforecast.2021.10.004.

Section 3 to outline specific areas in which tree-based models can be improved by tailoring them to the forecasting task if one chooses to open up the black box. For this, we highlight some model extensions for tree-based models that we have found to be effective in our forecasting practice. In particular, we show in preliminary experiments how a general technique for obtaining probabilistic forecasts from tree-based models can turn the winning solution from the accuracy competition into a highly competitive contestant for the uncertainty competition. We hope that based on the M5 competition, we will see an increase in methodological work around tree-based methods and practical work, making deep learning methods more robust so they can be used in black-box settings. The success of both of these lines of work will be judged compared to the skillfully crafted entries in the M5 competition.

We close our discussion in Section 4 by deriving some lessons from the success of tree-based models in the M5 competition.

2. The success and ubiquity of tree-based methods in practice

The recent literature on forecasting methods—in particular in machine learning venues—is dominated by deep-learning-based forecasting models (e.g., de Bézenac et al., 2020; Li et al., 2019; Lim, Arik, Loeff, & Pfister, 2019; Rangapuram, Seeger, Gasthaus, Stella, Wang, & Januschowski, 2018; Rasul, Seward, Schuster, & Vollgraf, 2021; Salinas, Flunkert, Gasthaus, & Januschowski, 2019; Smyl, 2020; Wen, Torkkola, Narayanaswamy, & Madeka, 2017; Zhou et al., 2021). Dedicated open-source packages implementing these techniques are readily available (e.g., GluonTS (Alexandrov et al., 2020), PytorchTS (Rasul, 2021), and PyTorch Forecasting) and they are widely used in practice (e.g., Laptev, Yosinsk, Li Erran, & Smyl, 2017; Liberty et al., 2020; Wen et al., 2017; see also Benidis et al., 2020 and references therein). Tree-based forecasting methods, on the other hand, have not received nearly as much academic attention.[2] This is illustrated, for example, by a recent encyclopedic overview on forecasting (Petropoulos et al., 2020), which extensively covers deep learning but does not mention tree-based models in nearly as much detail.

However, the leader boards of forecasting competitions held in recent years (including of course the M4 and M5 competitions, but also the Global Energy Forecasting (GEFCom) competitions, as well as various Kaggle forecasting competitions summarized in Bojer and Meldgaard (2020)) paint a somewhat different picture. Here, tree-based approaches—in particular in the form of Gradient Boosted Decision Trees (GBDTs)—feature heavily in the top ranks (e.g., top two places in the Kaggle Rossmann store sales competition, runner-up in M4 (Montero-Manso, Athanasopoulos, Hyndman, & Talagala, 2020), as well as various top-10 rankings the GEFCom and other Kaggle competitions (Bojer & Meldgaard, 2020)).

In light of this, it is perhaps less surprising that the M5 competition featured tree-based methods heavily among the submissions and within the top ranks (with the winning solutions in both tracks being based on GBDTs). Neural networks *do* appear as second and third places in the M5 accuracy competition (based on DeepAR and NBEATS (Alexandrov et al., 2020; Oreshkin, Carpov, Chapados, & Bengio, 2020; Salinas et al., 2019)) and as members in ensembles. However, they are outnumbered vastly by approaches based on gradient boosted trees.

In the following sections, we will try to shed light on why GBDTs were particularly attractive and ultimately effective for the M5 competition, based on observations and learnings from previous competitions and our own experience. We cover insights from other Kaggle competitions in Section 2.1, highlight what makes GBDTs particularly effective in Section 2.2, take a closer look at how trees were leveraged in the M5 competition in Section 2.3, and discuss how this aligns with our own experience with tree-based methods in Section 2.4.

2.1. Tree-based methods at Kaggle competitions

While deep learning dominates the headlines with spectacular breakthroughs in ML (e.g., AlphaZero (Silver et al., 2018), AlphaFold (Senior et al., 2020), or GPT-3 (Brown et al., 2020)), the success and prevalence of tree-based methods in the M5 competition become less surprising in the context of Kaggle competitions. In this case, GBDTs have consistently performed exceedingly well, in particular on data that is not images/videos or text. In a recent interview, Anthony Goldbloom, the CEO of Kaggle, called the prominence of GBDTs on Kaggle the "most glaring difference" between what is used on Kaggle and what is "fashionable in academia".[3] This is also reflected in the Kaggle Data Science and Machine Learning surveys from 2019 and 2020,[4] where tree-based methods appear in places two (random forests) and three (gradient-boosted trees) in the list of most popular techniques (behind linear/logistic regression in first place, but before any deep learning technique).

One high-level explanation is that tree-based models can very successfully be used as black-box learners. This allows contestants to focus on improving the model input (through feature engineering) and optimizing their ranking through well-understood recipes, such as careful hyperparameter tuning and ensembling. Feature engineering is arguably both more accessible and more composable than the building blocks of deep learning-based forecasting techniques. For example, contestants can easily discuss and share features that have proven effective, while integrating a sub-component of another deep architecture is often highly non-trivial. Further, software packages implementing gradient boosting, such as LightGBM (Ke et al., 2017) and XGBoost (Chen & Guestrin, 2016), are extremely robust and performant. In contrast,

[2] We speculate that this is because they lack a perceived component of novelty required for publication acceptance.

[3] https://www.youtube.com/watch?v=0ZJQ2Vsgwf0.

[4] https://www.kaggle.com/kaggle-survey-2019 and https://www.kaggle.com/kaggle-survey-2020.

publicly available deep learning-based techniques are often significantly less robust, requiring careful feature scaling and tuning of hyperparameters and potentially structural changes to obtain good performance on a novel task. This overall explanation aligns with the observation that the top contestants *did* use tree-based models as black-boxes (without modifications to the software and core model). The ability to rely on a mature model implementation as a black-box is a time-effective and often superior strategy in a time-constrained environment such as a Kaggle competition. With the learning algorithm fixed and robust, contestants can invest their time into data analysis, understanding the domain, feature engineering, hyperparameter tuning, and ensembling, all of which are accessible to a broad audience. On the other hand, adapting and tuning deep learning architectures for a given task is a riskier strategy. It requires investing time in deeply understanding a given architecture to make the necessary modifications. While neural forecasting techniques and their implementations have rapidly advanced, they still lack the required robustness to be used as black boxes. However, they can be very competitive when expertly applied.

2.2. Ingredients that make trees so successful in competitions

Modern tree-based implementations have several attractive features built-in that make them particularly suited for the M5 competition. We go over some subjectively chosen examples in the following without claiming comprehensiveness.

Loss functions. LightGBM allows users to choose from a large collection of loss functions. Contestants found the Tweedie (1947) loss to be particularly effective, and it was successfully used across many solutions in the M5 competition. The Tweedie loss is specifically geared towards sparse (zero-inflated) target data. While deep learning forecasting frameworks are available with many different loss functions (Alexandrov et al., 2020), this particularly successful loss function is not currently implemented in any of them. Additional implementation effort was needed for this, and not every team wanted to invest this time. Jeon and Seong (2021) implemented the Tweedie loss for DeepAR in PyTorchTS (Rasul, 2021) and placed in the top 3.

Tree-based methods also handle **sparse targets** easily when used as inputs (e.g., lagged/aggregated values). Inputs are implicitly discretized as needed, and scaling plays no role (multiplying a feature column by some constant will leave the solution unchanged). In contrast, training techniques for deep learning are often highly sensitive to scaling, and handling scaling of lagged target features correctly, particularly when combined with probabilistic output, causes substantial difficulty. As a case in point, Anderer and Li (2021) use NBEATS for the top 5 (non-sparse) levels in the hierarchy and LightGBM at the bottom levels, where they found it to outperform DeepAR consistently. This advantage regarding scaling also applies to other input features, though the challenges are somewhat easier to deal with.

Real-world data complications. LightGBM and XGBoost are battle-hardened implementations that have built-in support for many real-world data attributes, such as missing values or categorical feature support. LightGBM offers many missing value handling methods, while users of other frameworks may need to implement this functionality manually.

Other factors. In a competition, speedy, iterative improvements are often key. For this, tree-based methods provide fast training and a form of interpretability that can be useful for identifying directions for improvement. In contrast, neural networks are more time-consuming to train, and diagnosing model shortcomings is challenging. It depends highly on the model architecture chosen and, ultimately, is still more of an art than a science.

Furthermore, the sophistication of the default parameterization of LightGBM and XGBoost is such that the first solution often is competitive. The 59th place in the accuracy track obtained with LightGBM and minimal feature engineering is a case in point.[5] Again, this is in contrast to the amount of work needed when working with deep learning. There is a whole zoo of methods from which to choose, and the default parametrization may lead to disappointing results. To showcase this point, consider Table 1. If we use DeepAR in its default settings (column DeepAR (student-t)), its accuracy is far from satisfactory. At the same time, another method available in GluonTS, MQCNN (Wen et al., 2017) fares well in its default settings. A change to one parameter (exchanging the default student-t likelihood with the negative binomial likelihood, column DeepAR (neg bin) in Table 1) leads to a much-improved result. A hypothetical M5 competitor considering these results would now have to choose into which model to invest time and energy as learnings are not easily transferable from MQCNN to DeepAR. This is more complex and confusing than tuning a LightGBM model where learnings from previous competitions can be readily carried over.

2.3. Tree-based methods at the M5 competition

When examining the winning solutions in the M5 competition more closely, we note that all teams treated the core learning algorithm as a black box. While GBDT packages such as LightGBM and XGBoost are open source, none of the top entries report modifying the code. Contrast this, for example, with the winning deep learning hybrid solution from the M4 competition, which required highly custom code. With the algorithm fixed, masterful feature engineering, hyperparameter tuning, and ensembling became the deciding factors.

Mature software toolkits for GBDTs come with a large number of configuration options but also appropriate default parameter settings that work well across a variety of tasks. Consequently, most teams invested their time tuning these settings and optimizing everything "around"

[5] See https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/163460.

Table 1
The table shows accuracy at the lowest level of granularity in the hierarchy for different quantiles and a summary metric (mean_wQL) that approximates CRPS.

Quantile	M5 winner	DeepAR (student-t)	DeepAR (neg bin)	MQCNN
0.005	0.010	0.023	0.016	0.010
0.025	0.050	0.075	0.061	0.051
0.165	0.312	0.319	0.319	0.320
0.25	0.444	0.458	0.450	0.451
0.50	0.698	0.766	0.705	0.712
0.75	0.687	0.821	0.699	0.709
0.835	0.585	0.754	0.600	0.609
0.975	0.182	0.430	0.202	0.213
0.995	0.055	0.310	0.074	0.084
Mean_wQL	0.336	0.440	0.347	0.351

the model, for example, via work on evaluation schemes, feature processing, ensembling, taking care of the hierarchical information, and, interestingly, cross-validation schemes. We comment briefly on these aspects in the following.

Feature processing used by the top entrants is mostly standard and unsurprising: Lagged and aggregated (across time or the product hierarchy) target values, seasonal dummy variables, calendar features, and transformations of the provided covariate information (e.g., converting prices from absolute into percentages). Normalization of features across time series is crucial in global models to enable effective learning across time series (Montero-Manso & Hyndman, 2020; Salinas et al., 2019). However, one advantage of GBDTs is that—unlike for neural networks—feature scaling does not noticeably influence their learning dynamics. Otherwise, care must be taken so that normalization schemes do not use future values or otherwise leak information from the future. Notably, most successful teams did not use any additional external data.

Cross-validation was stressed by many participants as particularly important. Ensuring that information does not leak during cross-validation is crucial (e.g., in the normalization of features) and to make sure we choose enough the appropriate cross-validation splits. Again, this is a topic well-discussed in the forecasting literature (e.g., by Hyndman and Athanasopoulos (2021)). However, cross-validation in a time series setting does not seem to be as well understood in the general data science community.

Post-processing was used by some (but not all) teams to correct systematic biases. The need for such corrections is generally a sign of shortcomings of the model that should be fixable in a principled way. However, we note that such pragmatic post-processing correction techniques are surprisingly hard to beat in our experience from practice (as much as it pains the modeling purist).

GBDTs do not readily produce **probabilistic forecasts**. Entries in the uncertainty track either obtained forecast distributions using past residuals or directly forecasting quantiles (LightGBM supports quantile regression by optimizing the pinball loss). We will discuss schemes for obtaining probabilistic forecasts in the next section in more detail.

Finally, **ensembling** featured heavily—as in most Kaggle competitions. For **addressing hierarchies**, for one area where we were expecting methodological innovation, we observed that the rich body of literature on this topic was mostly not used by the winning contestants.[6] One discussant noted that "too many other factors in this competition such as out-of-stock and survivorship-bias [...] needed to [be] addressed first before [trying] these approaches.[7]"

2.4. Tree-based forecasting methods at Amazon

Before deep learning became the dominant force in operational forecasting applications[8] at Amazon,[9] random forest-based methods were the tool of choice for the most difficult forecasting problems on the retail side: forecasting the demand of products with little to no sales history. We speculate that the reason for this choice is similar to why the participants in the M5 competition resorted to tree-based algorithms. In our case, in the early 2010s for this challenging problem, random forests were able to explore rich covariate information without having to excessively pre-process them (in contrast to more traditional models like Generalized Linear Models or ARIMAX, which we experimented with heavily and adopted to this case (Böse et al., 2017; Seeger et al., 2017; Seeger, Salinas, & Flunkert, 2016)).

Random forests have the advantage of being able to train end-to-end (as opposed to complicated pipelines consisting of clustering similar products and then forecasting new products as weighted combinations of similar products). They are also *global* models (Januschowski et al., 2019), allowing for borrowing statistical strength from other, related time series via parameter sharing. Tree-based models are interpretable, which allows iteration over them meaningfully, obtaining continuous

6 See the 68th place for an interesting discussion https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/163578.

7 https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/166814.

8 A terminology coined by Januschowski and Kolassa (2019) to distinguish between the modern forecasting problem of a large collection of time series, each having a limited signal. Contrast this with strategic forecasting problems, which have traditionally been the focus of the disciple. The problems exhibit a small number of time series with a sufficient amount of history.

9 For other types of forecasting problems that are less related to the M5 competition, other methods are more suitable. For the demand forecasting problem in Amazon retail, https://www.amazon.science/latest-news/the-history-of-amazons-forecasting-algorithm provides a historical summary.

progress. Furthermore, they handle sparsity for the target values much better natively than other methods.

Random forests also support feature importance calculation, which can guide the experimenter to iterate over large feature sets, re-engineer them, and identify the ones that work the best. There is prior work that proposes injecting artificial contrast variables (features) into the trees with probability p, resulting in so-called Artificial Contrasts with Ensembles (ACE) (Tuv, Borisov, Runger, & Torkkola, 2009). ACE is useful for estimating the redundant and compact feature sets, shrinking the feature set even more. However, feature selection does not necessarily improve model accuracy due to random forests' inherent capability to ignore noisy features (the splitting criterion favors informative features over noisy ones, and can completely disregard irrelevant ones). From a computational performance point of view, working on a smaller feature set improves the execution speed of the machine learning pipeline involving random forests and may therefore be favored.

Several model innovations were needed that ensured random forests were the model of choice to make random forests work well in the retail domain for Amazon. We will discuss them in more detail in the next section. For this section, the most important point is that tree-based methods are a strong global model family and that it took intense research to replace them with some eventually better, deep-learning-based models.

3. Possibilities for model extensions

In the previous section, we noted that the most successful teams in the M5 competition could employ tree-based models as black boxes. This is a perfectly reasonable strategy in a competition where time is short. However, we believe that step-function changes in accuracy improvement typically come from profound changes to the core forecasting model. At the same time, an outlier (for a Kaggle competition) and a case in point (for a model innovation) is the M4 competition: the winning solution in the M4 competition (Smyl, 2020) won by a large margin and was an original model idea that required considerable risk-taking on behalf of the competitor. Outside the context of a competition, such risk-taking is easier, and we commonly witness it in scientific publications (often novel model work is a prerequisite for publishing) and in industrial practice for companies with enough critical mass in the area (e.g., Arik et al., 2020; Bandara, Shi, Bergmeir, Hewamalage, Tran, & Seaman, 2019; Laptev et al., 2017; Salinas et al., 2019). We will provide two previously unpublished examples of model improvements for tree-based models next. The purpose is to illustrate directions that we would find interesting for further investigation in future work for tree-based models in forecasting.

3.1. Sparse quantile random forests at Amazon

In Section 2.4, we had noted that we employed random forests for the problem of new product forecasting. Without frameworks such as LightGBM and XGBoost around in the early 2010s, we adapted the core random forests algorithms via the following model innovations: (i) treat missing values natively, (ii) handle sparsity, and (iii) effective splitting. In practice, *missing values* often occur in particular for high-dimensional categorical covariates.[10] In our applications at Amazon, we decided to rather not impute these values or prune data, but instead, make sure that we can handle these cases natively and gracefully by introducing an extra branch in the branching rule explicitly for "missing" values. Similarly, sparse data frequently occurs in practice at Amazon and similarly in the M5 competition. However, sparsity in the M5 competition was mainly at the target data, which tree-based methods handle well (in particular compared to other methods, see our earlier discussion). In our practice, we also faced sparsity in the covariates in two cases, (i) using a single model to forecast the demand of a diverse set of products, and (ii) considering textual data as input. Diverse products from different categories may have different non-overlapping feature sets in addition to common features. For example, books may have a feature of "book binding type", whereas video games may have "platform". This feature occurs only within a category but may be important in making distinctions within that category. Textual data can lead to high-dimensional numerical representations in which meaningful values are sparse and do not lend themselves well to uniform sampling. Instead, a weighted sampling scheme based on the entropy is more effective. Finally, splitting optimally is prohibitively costly for high-cardinality variables, and we have used splitting schemes based on stochastic greedy variants to speed this up.

Another splitting optimization we did was sub-sampling. Sub-sampling permits to use random forests with even larger data sets. As the search of the optimal split function involves sorting of arrays, which has a computational complexity of $O(n*\log(n))$, it can therefore grow prohibitively large with sample sizes of, say, over 10 million. Experiments showed that with aggressive sub-sampling close to the root of the tree where the splits are the largest, the execution speed of the model improves, and in most cases, the accuracy. This finding is in line with the earlier work on Extremely Randomized Trees (Geurts, Ernst, & Wehenkel, 2006).

To obtain probabilistic forecasts, we resorted to the approach based on using all training examples in the leaf nodes of trees (Meinshausen, 2006). The main idea behind this approach is as follows. Instead of generating the prediction based on the average values of the examples in the leaf nodes, the training examples in the leaves are drawn to form the probabilistic prediction. Furthermore, Meinshausen (2006) has shown that the estimator is consistent for any fixed quantile level. However, Meinhausen's approach implicitly assumes that all training examples are contained in the tree's leaf nodes and that each prediction reaches a leaf node in the tree. In the presence of missing values, as we often see in practice, this assumption is violated.

[10] As mentioned before, LightGBM contains sophisticated handling of missing values.

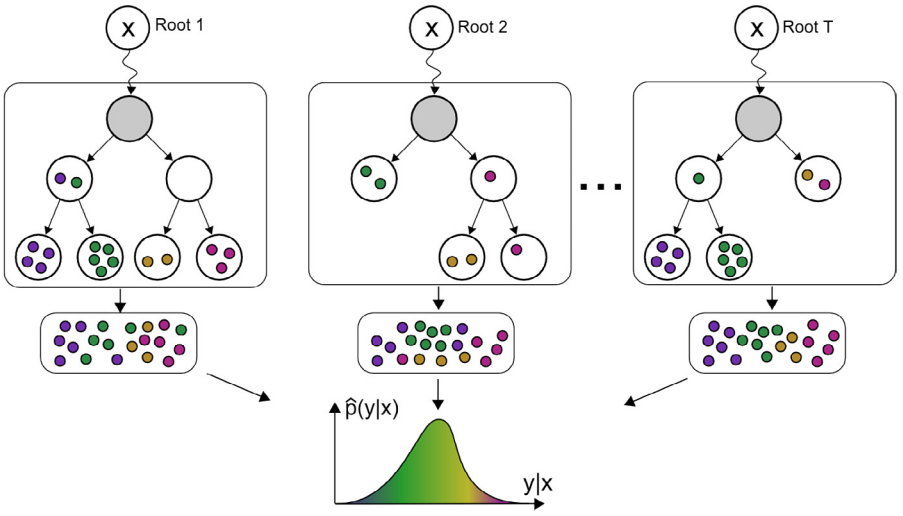

Fig. 1. Pooling training samples from the T subtrees relative to the prediction nodes (gray), which may or may not correspond to the leaf nodes. One tree at a time, qSamples training instances are drawn with replacement. finally, all the qSamples · nTrees samples are pooled together to form the basis set for estimating the target at different quantiles, ${}^{y}(q) = F^{-1}_{y|x}(q)\ q \in [0, 1]$.

A straightforward workaround is to collect all train samples contained in the subtree with the prediction nodes as the relative roots and pool them for computing quantiles. However, this would emphasize trees for which the prediction node is close to the root (i.e., prediction terminates very early) since more samples would be collected. This could lead to unfavorable consequences since early termination indicates poor prediction, i.e., only the first few decision rules were applied. This can be circumvented by an additional bootstrapping layer. That is, if we draw qSamples from the pool of training samples from the subtrees, and pool afterwards to yield qSamples · nTrees training samples to predict target variable y at different quantiles q, or $y(q) = F^{-1}_{y|x}(q)\ q \in [0, 1]$. Fig. 1 illustrates how the sampling and pooling of samples looks.

3.2. Level set forecaster

Obtaining probabilistic forecasts from tree-based models presents an opportunity for model innovation in general and in the M5 competition in particular, as there is comparatively little work available. However, contestants were successful in the M5 without having to invent new approaches. We briefly review standard approaches from the literature to obtain probabilistic forecasts (some of which were used in the competition) and then provide a new way to present encouraging preliminary empirical results.

A generic technique, which works for any point forecaster, is to bootstrap prediction intervals from empirical errors obtained in the past. This is called conformal predictions (see, e.g., Shafer & Vovk, 2008) in the literature, and contestants in the M5 competition resorted to this technique with success. Kolassa[11] discusses this in detail for tree-based forecasting methods. A second approach is to assume a parametric form of the distribution and estimate the free parameters via the algorithm. Again, this is a generic, well-known technique (e.g., Bishop, 2006) also commonly applied in neural networks (Salinas et al., 2019). In random forests and gradient-boosted trees, this has been successfully implemented in major frameworks (Duan et al., 2020; März, 2019, 2020). The concurrent work by Sprangers, Schelter, and de Rijke (2021) offers a version targeted at tree-based methods. But, this wasn't employed by the top competitors in the competition. Instead, they either resorted to conformal prediction or quantile regression. We note that since GBDTs can work with any loss function, quantile loss can be used. This is a good approach for applications that require one or a few quantiles of the forecast distribution rather than the whole distribution. In the following, we describe in rough terms a novel approach that we refer to as *Level Set Forecaster (LSF)* (Hasson, Wang, Januschowski, & Gasthaus, 2021).

In LSF during the training phase, we identify the set of training examples x_i that are mapped to the same point forecast value $f(x_i)$ by the tree-based algorithm and collect their corresponding true target values in a bin. We then perform a simple procedure to ensure that each bin includes a certain minimum of training examples by grouping bins with similar model values. We pick the corresponding bin based on the predicted value at prediction or forecast time for a given input covariate vector. Then for the quantile q of choice at which we want to forecast, we pick the qth quantile of the true values in the associated bin.

Note that the above procedure can be applied to any point forecaster, and it is not specific to trees. Still, it is readily implementable for XGBoost or LightGBM.[12] Hasson et al. (2021) prove the consistency of LSF under mild

[11] https://forecasters.org/wp-content/uploads/gravity_forms/7-c6dd08fee7f0065037affb5b74fec20a/2017/07/Random_Forest_Density_Forecasts_in_Retail.pdf.

[12] For example, https://github.com/awslabs/gluon-ts/blob/master/src/gluonts/model/rotbaum/_model.py.

Table 2
LSF on the top point forecast submission compared with top results in the uncertainty competition.

quantile	LSF	Winner	Runner-up	Third	Fourth	Fifth
0.005	0.01227	0.010	0.04201	0.01891	0.01225	0.01133
0.025	0.05484	0.05004	0.08649	0.06446	0.05483	0.05739
0.165	0.31677	0.31276	0.33782	0.33371	0.31566	0.35329
0.25	0.45108	0.44436	0.46158	0.46699	0.44937	0.49415
0.5	0.72434	0.69886	0.69044	0.74712	0.71279	0.7661
0.75	0.72848	0.68747	0.72231	0.69522	0.71071	0.70321
0.835	0.60748	0.58519	0.59897	0.59059	0.61292	0.59058
0.975	0.21801	0.18236	0.19664	0.19268	0.19348	0.20365
0.995	0.0863	0.0551	0.07689	0.07433	0.06225	0.06811
Mean_wQL	0.35551	0.33624	0.35702	0.35378	0.34714	0.36087

Table 3
Selected differences between tree-based and neural network based forecasting methods.

	Trees	Neural networks
Training complexity	In parallel over trees	Sequential stochastic gradient descent
Data size	Large: In parallel over trees	Depending on architecture, the sequence nature can be handled natively
Training data	May need all data in memory	Can read data sequentially
Missing values	Handles gracefully	Depending on architecture may need imputation
Hyperparameters	Simple	Selection of network structure and parameters is an art

assumptions, and show that it compares favorably with vanilla conformal predictions. To showcase the empirical effectiveness of LSF, we implemented it on top of the winning solution of the accuracy competition to obtain probabilistic forecasts. Table 2 contains preliminary results for the bottom level of the hierarchy for specific quantiles, including a weighted average over all quantile losses. While the results are only for the bottom level of the hierarchy,[13] they show that already without further tuning, this approach is immediately competitive and would comfortably fall into the top 5 entries in the uncertainty competition for the most challenging level in the hierarchy. More generally speaking, the results show the potential for research into generic techniques for turning point forecasts into probabilistic ones. We hope that more research in this direction will be spawned from the competition, similarly to the research efforts for obtaining realistic probabilistic forecasts in neural networks (e.g., de Bézenac et al., 2020; Rasul, Sheikh, Schuster, Bergmann, & Vollgraf, 2020). Finally, we remark that the results in Table 2 are a testimonial to the quality and robustness of the first place solution in the accuracy competition.

4. Conclusion

The M5 competition has re-affirmed that tree-based methods and gradient-boosted trees belong to the toolbox of forecasters working on big data, operational forecasting problems. We believe that the sophistication of the existing implementations ensures a strong performance. These include feature processing, appropriate loss functions, execution speed, and robust default parametrization. Table 3 provides a high-level comparison between tree-based methods and neural networks. Similar care and sophistication in details will help any other forecasting approach, particularly deep learning-based tools that are still relatively immature compared to XGBoost and LightGBM. Also, in contrast to tree-based models, no dominant neural architecture has emerged yet, adding a further layer of complexity in a competition. It is an exciting research challenge for the deep learning community to develop methods that allow to use, improve, diagnose and interpret deep learning models in a similarly user-friendly way as tree-based models.

When we considered the winning entries of the M5 competition in more detail, we found that teams did not have to change the core algorithms much. We hope that the M5 competition will spark some research into model improvements for tree-based forecasting methods. We believe that meaningful changes in accuracy will come mainly from more sweeping changes to the core models. In this sense, we reject the notion that expert forecasters will become less needed (Makridakis & Spiliotis, 2021). On the contrary, we believe that a lot of model innovation is still needed in at least the following areas. First, enabling gradient-boosted trees to provide probabilistic forecasts is an area that deserves more attention, similar to the amount of research in neural networks (de Bézenac et al., 2020; Rasul et al., 2021, 2020). Second, expert forecasters are needed for continuing to innovate on handling intermittent data better (in particular for neural networks to handle intermittent data similarly well as tree-based methods do, see, for example, Kourentzes (2013), Turkmen, Januschowski, Wang, and Cemgil (2021)), for better taking advantage of the hierarchical structure (potentially as part of the model like Rangapuram, Werner, Mercado Lopez, Benidis, and Januschowski (2021)) and for

[13] The mean weighted quantile loss for the bottom level of the hierarchy does not preserve the overall order of the winners. However, the results are indicative. Full details of our approach are available in Hasson et al. (2021).

handling multi-variate forecasting problems better. We are sure that the M5 data set will trigger the development of these improvements in the academic literature, and with model innovations surely pending, we expect the results on the M5 data set to improve significantly over time.

While success in competitions may favor black-box approaches, especially in time-constrained situations, there are large-scale real-world forecasting deployments, such as Amazon demand forecasting, that benefit from tailored neural network-based approaches (Eisenach, Patel, & Madeka, 2020; Wen et al., 2017). Of course, for smaller-scale applications with a trade-off involving time-to-deploy, the situation may be the same as in competitions: trees offer a particularly attractive solution.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., et al. (2020). GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, *21*(116), 1–6, URL http://jmlr.org/papers/v21/19-820.html.

Anderer, M., & Li, F. (2021). Forecasting reconciliation with a top-down alignment of independent level forecasts. http://arxiv.org/abs/2103.08250.

Arik, S., Li, C.-L., Yoon, J., Sinha, R., Epshteyn, A., Le, L., et al. (2020). Interpretable Sequence Learning for Covid-19 Forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), *Vol. 33, Advances in neural information processing systems* (pp. 18807–18818). Curran Associates, Inc..

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., & Seaman, B. (2019). Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology. http://arxiv.org/abs/1901.04028.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., et al. (2020). Neural forecasting: Introduction and literature overview. arXiv preprint arXiv:2004.10240.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Bojer, C. S., & Meldgaard, J. P. (2020). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*.

Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., et al. (2017). Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, *10*(12), 1694–1705.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). Language Models are Few-Shot Learners. arXiv E-prints arXiv:2005.14165.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *KDD '16, Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). ACM.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., et al. (2020). Normalizing Kalman Filters for Multivariate Time Series Analysis. *Advances in Neural Information Processing Systems*, *33*.

Duan, T., Avati, A., Ding, D. Y., Thai, K. K., Basu, S., Ng, A. Y., et al. (2020). NGBoost: Natural Gradient Boosting for Probabilistic Prediction. http://arxiv.org/abs/1910.03225.

Eisenach, C., Patel, Y., & Madeka, D. (2020). MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. http://arxiv.org/abs/2009.14799.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Hasson, H., Wang, Y., Januschowski, T., & Gasthaus, J. (2021). Probabilistic Forecasting: A Level-Set Approach. In *Advances in neural information processing systems*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction* (2nd ed.). Springer.

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice. https://www.otexts.org/fpp3.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., et al. (2019). Criteria for classifying forecasting methods. *International Journal of Forecasting*.

Januschowski, T., & Kolassa, S. (2019). A Classification of Business Forecasting Problems. *Foresight: The International Journal of Applied Forecasting*, *52*, 36–43.

Jeon, Y., & Seong, S. (2021). Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*, http://dx.doi.org/10.1016/j.ijforecast.2021.07.004, URL https://www.sciencedirect.com/science/article/pii/S0169207021001151.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), *Vol. 30, Advances in neural information processing systems*. Curran Associates, Inc..

Kourentzes, N. (2013). Intermittent Demand Forecasts with Neural Networks. *International Journal of Production Economics*, *143*(1), 198–206. http://dx.doi.org/10.1016/j.ijpe.2013.01.009.

Laptev, N., Yosinsk, J., Li Erran, L., & Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. In *ICML time series workshop*.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., et al. (2019). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, R. Garnett (Eds.), *Vol. 32, Advances in neural information processing systems*. Curran Associates, Inc..

Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., et al. (2020). Elastic Machine Learning Algorithms in Amazon SageMaker. In *SIGMOD '20, Proceedings of the 2020 International Conference on Management of Data*. New York, NY, USA: ACM.

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. arXiv preprint arXiv:1912.09363.

Makridakis, S., & Spiliotis, E. (2021). The M5 Competition and the Future of Human Expertise in Forecasting. *Foresight: The International Journal of Applied Forecasting*, (60), 33–37.

März, A. (2019). XGBoostLSS – An extension of XGBoost to probabilistic forecasting. arXiv:1907.03178.

März, A. (2020). CatBoostLSS – An extension of CatBoost to probabilistic forecasting. http://arxiv.org/abs/2001.02121.

Meinshausen, N. (2006). Quantile Regression Forests. *Journal of Machine Learning Research*, *7*, 983–999.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, *36*(1), 86–92, M4 Competition.

Montero-Manso, P., & Hyndman, R. J. (2020). Principles and algorithms for forecasting groups of time series: Locality and globality. arXiv preprint arXiv:2008.00444.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR*.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Bergmeir, C., et al. (2020). Forecasting: theory and practice. http://arxiv.org/abs/2012.03854.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in neural information processing systems* (pp. 7785–7794).

Rangapuram, S., Werner, L., Mercado Lopez, P., Benidis, K., & Januschowski, T. (2021). End-to-End Learning of Coherent Probabilistic Forecasts for Hierarchical Time Series.

Rasul, K. (2021). PyTorchTS. URL https://github.com/zalandoresearch/pytorch-ts.

Rasul, K., Seward, C., Schuster, I., & Vollgraf, R. (2021). Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. http://arxiv.org/abs/2101.12072.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., & Vollgraf, R. (2020). Multi-variate probabilistic time series forecasting via conditioned normalizing flows. arXiv preprint arXiv:2002.06103.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2019). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*.

Seeger, M., Rangapuram, S., Wang, Y., Salinas, D., Gasthaus, J., Januschowski, T., et al. (2017). Approximate Bayesian Inference in Linear State Space Models for Intermittent Demand Forecasting at Scale. http://arxiv.org/abs/1709.07638.

Seeger, M. W., Salinas, D., & Flunkert, V. (2016). Bayesian intermittent demand forecasting for large inventories. In *Advances in neural information processing systems* (pp. 4646–4654).

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., et al. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, *577*(7792), 706–710.

Shafer, G., & Vovk, V. (2008). A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, *9*(3).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144. http://dx.doi.org/10.1126/science.aar6404, URL https://science.sciencemag.org/content/362/6419/1140.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85. http://dx.doi.org/10.1016/j.ijforecast.2019.03.017, URL https://www.sciencedirect.com/science/article/pii/S0169207019301153, M4 Competition.

Sprangers, O., Schelter, S., & de Rijke, M. (2021). Probabilistic Gradient Boosting Machines for Large-Scale Probabilistic Regression. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. ACM, http://dx.doi.org/10.1145/3447548.3467278, URL http://dx.doi.org/10.1145/3447548.3467278.

Turkmen, A. C., Januschowski, T., Wang, Y., & Cemgil, A. T. (2021). Forecasting intermittent and sparse time series: a unified probabilistic framework via deep renewal processes. *PlosOne*.

Tuv, E., Borisov, A., Runger, G., & Torkkola, K. (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, *10*, 1341–1366.

Tweedie, M. (1947). Functions of a statistical variate with given means, with special reference to Laplacian distributions. *Vol. 43*, In *Mathematical proceedings of the cambridge philosophical society* (1), (pp. 41–49). Cambridge University Press.

Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2017). A Multi-Horizon Quantile Recurrent Forecaster. http://arxiv.org/abs/1711.11053.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The thirty-fifth AAAI conference on artificial intelligence, AAAI 2021* (p. online). AAAI Press.