International Journal of Forecasting 38 (2022) 1426–1433

Contents lists available at ScienceDirect

International Journal of Forecasting

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

Forecasting with gradient boosted trees: augmentation, tuning, and cross-validation strategies

Winning solution to the M5 Uncertainty competition

A. David Lainder [a], Russell D. Wolfinger [b,*]

[a] TrueFii.AI, United States
[b] SAS Institute Inc. Cary, NC, United States

ARTICLE INFO

Keywords:
Gradient boosted trees
Neural networks
Purged k-fold cross-validation
Feature engineering
Forecasting competitions
M competitions
Uncertainty
Probabilistic forecasts
Time series
Machine learning
Retail sales forecasting
Time-series forecasting

ABSTRACT

Deep neural networks and gradient boosted tree models have swept across the field of machine learning over the past decade, producing across-the-board advances in performance. The ability of these methods to capture feature interactions and nonlinearities makes them exceptionally powerful and, at the same time, prone to overfitting, leakage, and a lack of generalization in domains with target non-stationarity and collinearity, such as time-series forecasting. We offer guidance to address these difficulties and provide a framework that maximizes the chances of predictions that generalize well and deliver state-of-the-art performance. The techniques we offer for cross-validation, augmentation, and parameter tuning have been used to win several major time-series forecasting competitions—including the M5 Forecasting Uncertainty competition and the Kaggle COVID19 Forecasting series—and, with the proper theoretical grounding, constitute the current best practices in time-series forecasting.

1. Introduction

Recent years have witnessed explosive growth in the use of two major classes of predictive modeling methods: gradient boosted trees (GBTs) and neural networks (NNs). Progress has been accelerated by data science competition platforms like Kaggle (Kaggle, 2010) as well as the availability of high-quality open-source packages like XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and Catboost (Dorogush, Ershov, & Gulin, 2018; Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin, 2017) for GBTs, and PyTorch (Paszke et al., 2019) and Tensorflow/Keras (Abadi et al., 2016; Chollet et al., 2015) for NNs. The winners of every machine learning competition in recent years featuring quantitative datasets have leveraged some combination of these in their solutions.

In the fields of computer vision and natural language processing, fairly standard techniques for network design, data augmentation, and training have emerged and provide a nearly automatic strong baseline for many tasks. These fields usually involve enormous datasets, such as ImageNet (Deng et al., 2009), that allow for a straightforward comparison of methods. Most areas of time-series forecasting, however, are just the opposite. In key use cases, such as forecasting product sales based on early traction or predicting the course of a novel epidemic, the number of truly independent observations is quite low and it is easy to fit a perfect model with the benefit of hindsight.

In such cases, where the number of independent observations is in the thousands rather than millions, the only way to measure actual generalizable performance is to forecast based on a known set of data, and be judged by on its predictive ability on a completely hidden, or often yet-to-happen, set of outcomes.

* Corresponding author.
E-mail address: russ.wolfinger@sas.com (R.D. Wolfinger).

https://doi.org/10.1016/j.ijforecast.2021.12.003
0169-2070/

Table 1
Popular Kaggle forecasting competitions.

Competition	Teams	Target
Walmart Store Sales (Kaggle, 2014)	688	Department Sales
Walmart Stormy Weather (Kaggle, 2015b)	484	Product Sales
Rossmann (Kaggle, 2015a)	3298	Store Sales
Wikipedia Web Traffic (Kaggle, 2017)	1095	Page Views
Corporación Favorita (Kaggle, 2018a)	1671	Product Sales
Recruit Restaurant (Kaggle, 2018b)	2148	Visits
Zillow Prize Phase 1 (Kaggle, 2018c)	3770	Zestimate Error
Zillow Prize Final (Kaggle, 2019)	70	Home Prices
COVID19 Global Weeks 1–4 (Kaggle, 2020a, 2020b, 2020c, 2020d)	371	Cases, Fatalities
COVID19 Global Week 5 (Kaggle, 2020e)	173	Case and Fatality Quantiles
M5 Accuracy (Kaggle, 2020f)	5558	Product Sales
M5 Uncertainty (Kaggle, 2020g)	909	Product Sale Quantiles

The best practices and theory discussed in this article have largely been forged in the fires of data science competitions over the past several years and draw upon lessons learned amidst thousands of hours of crowd-sourced experimentation. Richly detailed solution descriptions, code, and forum discussions have greatly facilitated the sharing of ideas and provide the primary references for our work. The techniques we describe are certainly not limited in efficacy to M5 Uncertainty; they are the culmination and refinement of successful methods employed in Kaggle time-series competitions over the past few years, as listed in Table 1.

For example, over the course of 2020, variations on these ideas were used to win multiple rounds of the COVID19 Global Forecasting series (Kaggle, 2020a, 2020b, 2020c, 2020d, 2020e), including consecutive wins in Rounds 3 and 4 and consistent top performance out of hundreds of teams amidst continuing national and international attention on this topic (Cramer, Ray, et al, & Reich, 2021).

This sequence of time-series competitions, a welcome addition to the machine learning landscape, provided the training ground to hone and optimize these techniques to the point where they are a framework easily applicable to any form of time-series forecasting, and capable of producing world-class performance, or at least very strong performance, if applied correctly.

2. M5 competition

The M competitions, organized by Professor Spyros Makridakis and colleagues (Makridakis, Spiliotis, & Assimakopoulos, 2021a; Makridakis et al., 2021b), provide a well-regarded proving ground for state-of-the-art time series methods. The most recent M5 competition (Kaggle, 2020f, 2020g), hosted on Google's Kaggle machine learning competition platform, enjoyed a particularly large field of over 5000 teams.

The M5 competition featured five years of daily sales data on over 30,000 Walmart store-item combinations, with a focus on the particularly difficult challenge of forecasting items with intermittent sales patterns. Two simultaneous competitions had distinct aims: one involved point forecasts for all items in order to be accessible to the largest possible field of participants, while the Uncertainty competition required participants to forecast sales at 12 different levels of aggregation (from store-item all the way up to overall sales) for nine different quantiles, to anticipate the full range of future outcomes for every possible grouping.

Table 2
M5 Uncertainty top ten leaderboard. Differences are computed sequentially and represent scoring gaps.

Rank	Team	WSPL	Percent Difference
1	Everyday Low SPLices	0.15420	.
2	[GoodsForecast] Nick Mamonov	0.15890	2.96
3	Ouranos Point to uncertainty v2	0.15892	0.01
4	Marisaka Mozz	0.15958	0.41
5	IHiroaki	0.16147	1.17
6	WalSmart	0.16156	0.06
7	Ka Ho_STU	0.16160	0.02
8	JQuant	0.16199	0.24
9	Praveen Adepu	0.16334	0.83
10	golubyatniks	0.16341	0.04

Forecasts for the M5 Uncertainty competition were scored based on the weighted scaled pinball loss (WSPL), a carefully designed metric that evaluated the ability of the quantile forecasts to precisely capture the extreme outcomes and range of uncertainty inherent in forecasting. The loss for each of the nine quantiles at each of the 12 levels of aggregation was averaged to produce an overall ranking of forecast accuracy.

Table 2 displays the top ten of the final leaderboard for M5 Uncertainty. Our team (Everyday Low SPLices) achieved a noticeably lower WSPL than all other competitors, using the methods described in the following sections.

3. Cross-validation

Machine learning is the art and science of training a model on one dataset in a way that maximizes its performance measured on unseen data. Time-series forecasting takes this to a whole new level, as while it may be possible to achieve perfect accuracy in classifying images, the future is inherently unknowable and most time series worth forecasting have inherently high levels of volatility.

Cross-validation (CV) is typically the most straightforward part of machine learning: divide the set into five *folds*, or partitions, train on four of the five, and measure performance on the remaining fold, ideally rotating so each of the five folds is out-of-sample once. This is the

classic k-fold cross-validation strategy. In cases where some data points are linked (e.g. multiple images of the same object, or correlated time series), related items are typically assigned to the same fold, to prevent leakage across folds.

When it comes to cross-validation for time-series forecasting, we face the following challenges:

(a) The series are almost all correlated with each other.
(b) Forecasting intervals usually overlap with others (e.g. 15 days forward versus 30 days forward).
(c) Sales can trend upward or downward over time, and exhibit other forms of non-stationarity.
(d) Some points come later in time than others; is it valid to use future results to predict past performance?

These challenges mean we are often working with a very limited number of truly independent observations (e.g. the number of 30-day periods in a five-year sample), and at no point can we make the assumption, inherent in many statistical and machine learning domains, that the data are independent and identically distributed.

3.1. Time-series split

The earliest and simplest mechanism designed to tackle the challenges inherent in time-series CV is the time-series split, as shown in Fig. 1(a). While this solves some of the aforementioned challenges, it faces major issues with regard to data utilization, model diversity, hyperparameter selection, and overall robustness. First, the earliest fold of the dataset is completely unrepresented in any performance measurements, and the other early folds are fit on very limited training data, using hyperparameters that are unlikely to work well on the full dataset. Second, given the expanding size of each fold, the earliest time periods—the ones least relevant to current performance—receive the highest weight in training.

Even with clever tricks to avoid some of these pitfalls, the result of any time-series split is that valuable data go unused or underweighted, leaks between folds are common, and the architectures and hyperparameters selected will rarely be optimal for full-set training.

3.2. Purged k-fold

The solution to this challenge is to realize that the future can, in many cases, actually be used to forecast the past, if we limit the data on each series appropriately. The model and selected hyperparameters gain the benefit of a full and robust training set, while each data point remains aware of only its own past results. The key insight is to *purge* or *embargo* some data between each fold, sufficient to cover the longest forecasting interval (here, 28 days) and any excess amount needed to prevent any short-term trends appearing across multiple folds (see De Prado (2018)), as illustrated by the gaps between the blocks in Fig. 1(b).

The five years of training data were a natural choice for folds in the M5 competition. Given that this competition focused on intermittently sold items in the food

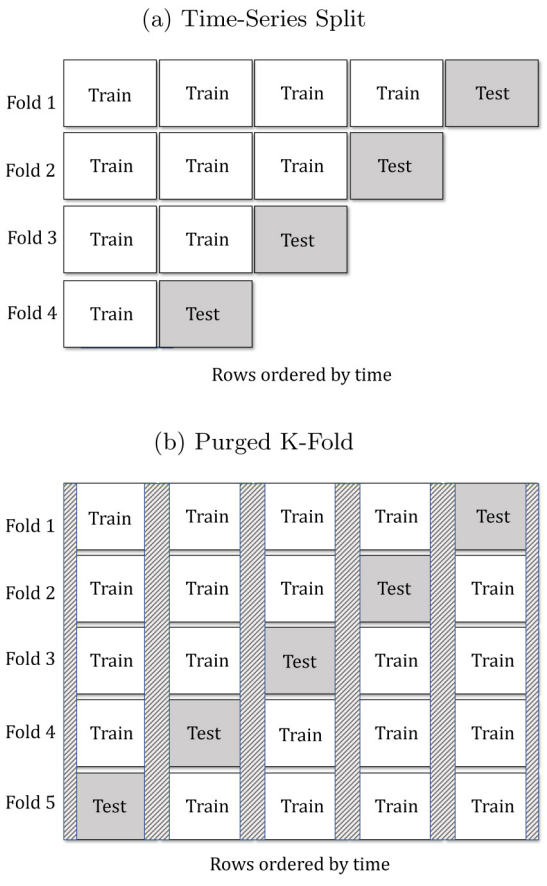

Fig. 1. Cross-validation fold schemes for time series.

and hobbies categories and that our target forecasting window was outside the holiday season (November and December), we chose that period as the best to leave out, to avoid any overlap between folds. This is illustrated by the gaps between blocks in Fig. 1(b). Having a stable, extremely robust CV was key to everything else we did, as it enabled us to quantify the impact of every possible change to our setup, and to determine when it produced a true improvement in forecasting accuracy.

3.3. Nested cross-validation

One common approach to CV in working with large datasets is to find the parameters that work best across all folds, and then go back and measure their performance on each holdout fold. While this poses few issues with millions of truly independent data points, it leads to an inflated view of performance on smaller, highly correlated, or non-stationary datasets. The reason is obvious: the more parameter combinations you try while looking across the entire training data, the more your performance improves—or rather, seems to improve.

When tuning models in the time-series context, or any context with any correlation or potential non-stationarity between items, it is essential to use nested cross-validation.

Table 3
Engineered features for M5 Uncertainty. Detrending is performed by dividing a specific series by the series of overall sales for its store. Rolling windows are multiples of seven to be able to detect uptrends or downtrends independent of the weekly seasonality.

Feature set	Number
Mean and median of each series over the past 7, 14, 21, 28, 56, 112 days	12
Standard deviation of each series over the past 7, 14, 28, 84, 168 days,	5
Skewness and kurtosis of each series over the past 7, 14, 28, 84, 168 days,	10
High and low quantiles (10th and 90th) of each series over the past 14, 28, 56 days	6
Exponentially weighted moving averages over the past 3, 7, 15, 30, 100 days	5
Each of the above features, performed on detrended sales	38
Percent nonzero sales days over the past 7, 14, 28, 56, 112 days	5
Pairs and differences of moving averages	20
Prior sales over each of the preceding ten days	10
Department ID, Category ID, Store ID, State ID	4
Holidays (sporting, cultural, national, religious)	8
Day of the week, day of the month, season, SNAP-related features, days forward	11
Total	130

This entails taking each of the training folds, and searching for the best hyperparameters through an internal cross-validation process, so that the remaining holdout data are truly a validation fold. While this typically produces fewer and more modest gains in validation performance, it ensures that any that occur are truly gains in *generalization* performance, and not overfitted to this specific training set.

4. Modeling

4.1. Model selection

A common rule of thumb is that NNs are all but mandatory for spatially oriented data, e.g. images or audio, while GBTs outperform on quantitative or tabular datasets. While both classes of methods are capable of modeling feature interactions and non-linearities, GBTs are typically both faster and easier to train, and much more straightforward to regularize. Both of these are enormous benefits when forecasting 108 separate targets (12 levels $\times$ 9 quantiles) across five very different folds, as in M5 Uncertainty. In this section we focus on the main aspects of effectively using GBT models for time-series forecasting.

4.2. Feature engineering

Many fields of modern machine learning increasingly involve minimal if any feature engineering, with image classification and text modeling increasingly being reduced to merely putting the data into a deep learning model and (optionally) tuning the pipeline, parameters, and architecture. While this oversimplifies the depth of those fields, the distance between an acceptable starter model and a great solution in many areas of machine learning is converging rapidly. However, with time-series forecasting, we have several complexities:

(a) A series of observations across time has in inherent order and structure, and is often reasonably well characterized by various rolling measures of variance and central tendency.

(b) Time-series data often span several orders of magnitude, requiring scaling and normalization to generalize across related series.

(c) The large pretrained models that work so well for images and audio do not exist for time-series forecasting: each problem has a unique set of inputs and structure (see, e.g., the last three sets of features in Table 3).

These considerations motivate engineering a wide collection of features to use as inputs to GBTs. Table 3 summarizes the 130 features we crafted for M5 Uncertainty.

4.3. Interpretability and feature importance

A key goal of any forecasting exercise is typically not just to make great black-box predictions, but to understand what is driving them and deciphering the key causal features in the system being studied.

Fig. 2 illustrates some patterns of feature importance for our GBT models in M5 Uncertainty, revealing some interesting patterns when summarizing by store or product level. At Level 3, or overall store sales, for the 75th quantile, the day of the week is the most important feature, followed closely by the mean and detrended mean sales over the last four weeks, with additional contributions from the store and the local SNAP patterns of the matching state. At Level 10, or individual product sales across all stores, the median or 50th percentile sales are best predicted by the 15-day and 30-day moving averages of sales, along with other means and averages. Once we reach the lower quantile Level 12, or individual store-product sales, feature importance is dominated by series intermittency, as represented by the number of nonzero-sales days over recent weeks.

4.4. Augmentation

When working with image data, there is a fairly standard set of augmentations, such as those implemented in the popular packages Albumentations (Buslaev et al., 2020) and RandAugment (Cubuk, Zoph, Shlens, & Le, 2020). Each operation is designed to take an image and produce an image that may look vastly different (stretched,

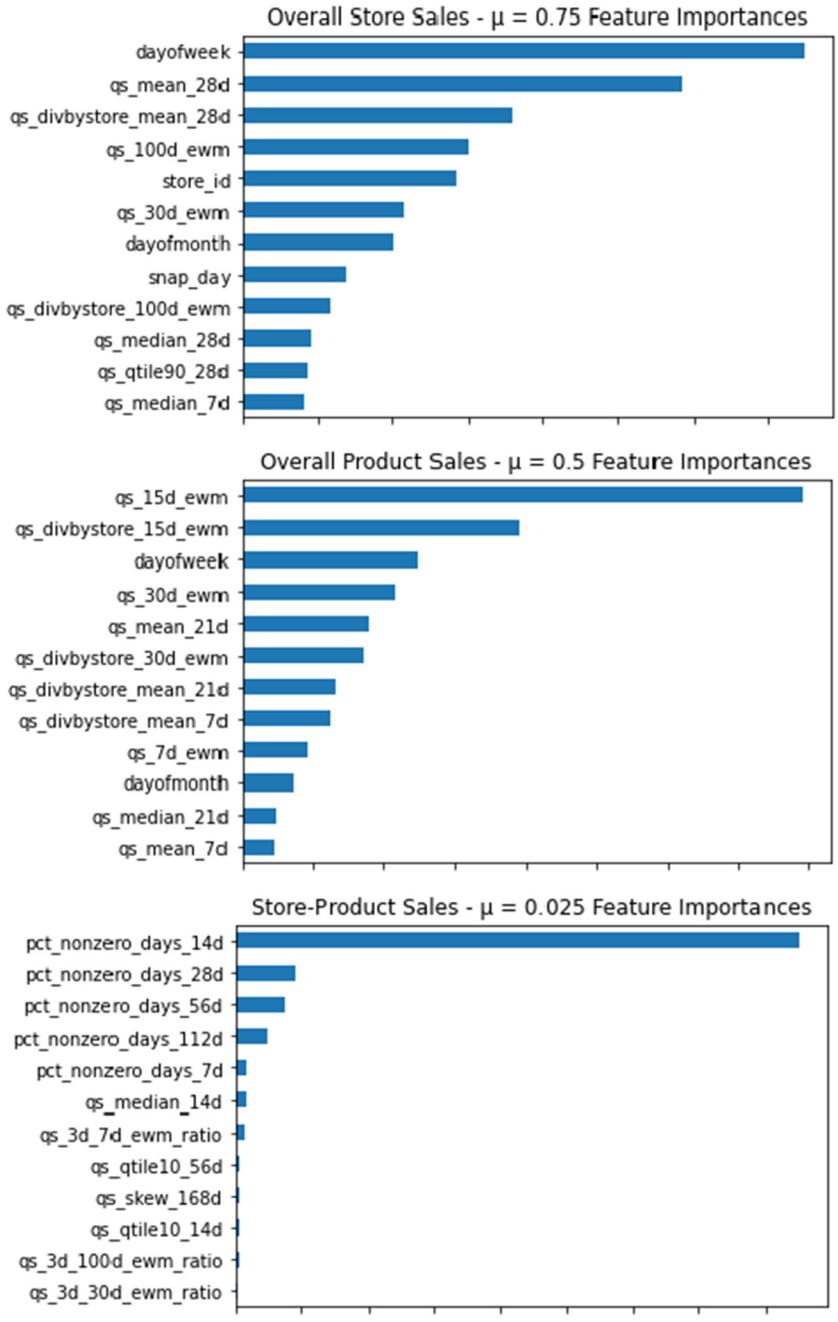

Fig. 2. Feature importance (gain) for LightGBM models in M5 Uncertainty. Top: overall store sales (level 3, quantile 0.75). Middle: overall product sales (level 10, quantile 0.5). Bottom: store-product sales (level 12, quantile 0.025).

cropped, flipped, rotated, or even with random cutouts) but should still belong to the same class of image.

Quantitative augmentation, in contrast, is usually quite limited. An image can have its brightness turned up 200% and still be recognizable; a trailing sales average series cannot quite be increased 200% and still validly predict its target—without also scaling the target. This key insight is what drives a new form of augmentation unique to quantitative forecasting.

The basic idea behind what we refer to as *range blending*, or *coupled Gaussian noise*, is that any perturbation to the training features should be both

(a) uniformly applied to all scalable features, and
(b) applied to the target value being forecast as well.

Concretely, we can take any of the thousand days of overall sales we are forecasting at Level 1, and shift all sales-scaled features (i.e. anything but the calendar or

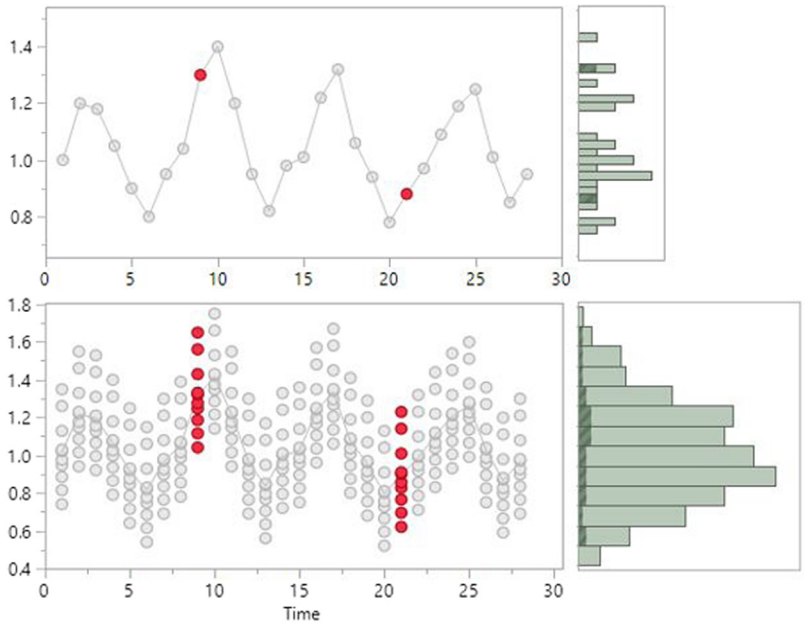

Fig. 3. Range-blending augmentation. Top: single training series. The two points highlighted in red fall into single bins in the histogram on the right. Bottom: across augmentations, the same points fall into multiple bins, smoothing out noisy splits and making the model more robust to overfitting. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

categorical features) by a common amount that is also applied to the target. This transformation is at once obvious and yet completely novel as far as we know.

This method takes a very limited set of data points, sometimes numbering in the thousands, and creates hundreds of thousands of similar ones, while still precisely preserving all feature–target and feature–feature interactions. Fig. 3 provides a graphical illustration. A basic Python implementation is as follows:

```
# randomize scaling
scaler = np.exp(SCALE_RANGE * random.normal(0, 1, len(x))

# now rescale y and 'scalable variables' in X by this amount
for col in [c for c in X.columns if 'quantity_' in c and 'ratio' not in c]:
X[col] *= scaler
y *= scaler
```

Gradient boosted trees are known as a less-than-smooth modeling method, given that each data point is placed in precisely one of a finite number of bins (e.g. 255) for each feature. With range blending, these independent observations are instead allowed to smooth and spread across dozens of bins, resulting in a much more accurate model. Even more importantly, this augmentation allows for a tuned amount of cross-learning between series at different levels of scale.

Range blending proved to be quite effective in M5 Uncertainty, especially for the series with a high amount of aggregation. It improved our WSPL CV by 5% overall, due primarily to a 5%–15% improvement on Levels 1–9.

4.5. Ensembling

A common criticism of machine learning competitions is that winning solutions typically consist of monstrous ensembles, with hundreds of models stacked through two, three, or even four or five layers of successive modeling. In our view, this is a valid concern, as real-world models require regular monitoring, retraining, and reasonable run-times.

The M5 competition provides a welcome test of real-world deployability, as forecasting each of the 108 levels of quantiles/granularity requires a more systematic and constrained solution. As with our automatic hyperparameter tuning and fully robust cross-validation strategy, our level of ensembling was also kept to a level of commercial practicability.

In short, each of our models was itself an ensemble due in large part to several architectural choices:

(a) Our use of range-blending augmentation ensured that each data point at the higher levels of aggregation could be oversampled hundreds of times, without ever appearing within the same combination of histogram bins.
(b) Our automated hyperparameter tuning setup had the additional benefit that each of the five folds were modeled using a distinct set of leaves, trees, and learning rates.
(c) The fact that hyperparameters were selected through fully internal nested cross-validation means that the final deployed models were very well tuned and not in need of a stacker layer to sub-select or average noisy or untuned parameter settings.

The final result was that our entire model stack—trained over a few hundred hours—could also be trained in a few hours and produce a result that would *still* be the winning solution to this competition.

Over the course of 2020, variations on these ideas were used to win multiple rounds of COVID19 Global Forecasting (Kaggle, 2020a, 2020b, 2020c, 2020d, 2020e), including consecutive wins in Rounds 3 and 4 and consistent top performance out of hundreds of teams amidst continuing national and international attention on this topic (Cramer et al., 2021).

This sequence of time-series competitions, a welcome addition to the machine learning landscape, provided the training ground to hone and optimize these techniques to the point where they are a framework easily applicable to any form of time-series forecasting, and capable of producing world-class performance, or at least very strong performance, if applied correctly.

5. Conclusions

The M5 competitions, this Special Issue of the *IJF*, and this article provide a crucial step in moving time-series forecasting forward as a distinct field of machine learning, one with best practices capable of producing strong models quickly and reliably. The challenge of predicting the future is that it does not yet exist, and a strong theoretical grounding is necessary to build models that fully generalize while avoiding overfitting to the past and present.

Machine learning models are capable of exceptional predictive power when used properly. Their application in the time-series domain—with only a limited number of truly independent observations—is fraught with the risk of leakage and overfitting. With rigorous cross-validation, feature engineering, data augmentation, and parameter tuning best practices, it is possible to more than overcome these and produce models that are simple to train and that predict the future reliably.

The techniques we discussed have produced, by large margins, top solutions in four consecutive time-series forecasting competitions. With the proper understanding, they provide the base knowledge to tackle any time series dataset using machine learning.

Python code

The M5 Uncertainty winning solution code and further details are available at http://github.com/david-1013/m5.

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

We thank Professor Makridakis and the M Competition Team, Walmart, and Kaggle for organizing and sponsoring the M5 competition. We also sincerely appreciate top Kaggle competitors who have freely shared their ingenious solutions and wisdom over the past several years.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on operating systems design and implementation (OSDI 16) (pp. 265–283).

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*(2).

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). New York, NY, USA: ACM.

Chollet, F., et al. (2015). Keras.

Cramer, E. Y., Ray, E. L., et al, & Reich, N. G. (2021). Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. MedRxiv preprint.

Cubuk, E., Zoph, B., Shlens, J., & Le, Q. (2020). Randaugment: Practical automated data augmentation with a reduced search space (pp. 3008–3017).

De Prado, M. L. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support.

Kaggle (2010). Kaggle competitions. [Online] https://www.kaggle.com/competitions. (Accessed 06 June 2021).

Kaggle (2014). Walmart store sales. [Online] https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting. (Accessed 06 June 2021).

Kaggle (2015a). Rossmann store sales. [Online] https://www.kaggle.com/c/rossmann-store-sales/. (Accessed 06 June 2021).

Kaggle (2015b). Walmart stormy weather. [Online] https://www.kaggle.com/c/walmart-recruiting-sales-in-stormy-weather. (Accessed 06 June 2021).

Kaggle (2017). Wikipedia web traffic time series forecasting. [Online] https://www.kaggle.com/c/web-traffic-time-series-forecasting. (Accessed 06 June 2021).

Kaggle (2018a). Corporacion favorita grocery sales forecasting. [Online] https://www.kaggle.com/c/favorita-grocery-sales-forecasting. (Accessed 06 June 2021).

Kaggle (2018b). Recruit restaurant visitor forecasting. [Online] https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting. (Accessed 06 June 2021).

Kaggle (2018c). Zillow prize (phase 1). [Online] https://www.kaggle.com/c/zillow-prize-1. (Accessed 06 June 2021).

Kaggle (2019). Zillow prize (phase 2, final). [Online] https://www.kaggle.com/c/zillow-prize-2. (Accessed 06 June 2021).

Kaggle (2020a). COVID19 global forecasting (week 1). [Online] https://www.kaggle.com/c/covid19-global-forecasting-week-1/. (Accessed 06 June 2021).

Kaggle (2020b). COVID19 global forecasting (week 2). [Online] https://www.kaggle.com/c/covid19-global-forecasting-week-2/. (Accessed 06 June 2021).

Kaggle (2020c). COVID19 global forecasting (week 3). [Online] https://www.kaggle.com/c/covid19-global-forecasting-week-3/. (Accessed 06 June 2021).

Kaggle (2020d). COVID19 global forecasting (week 4). [Online] https://www.kaggle.com/c/covid19-global-forecasting-week-4/. (Accessed 06 June 2021).

Kaggle (2020e). COVID19 global forecasting (week 5). [Online] https://www.kaggle.com/c/covid19-global-forecasting-week-5/. (Accessed 06 June 2021).

Kaggle (2020f). M5 forecasting - accuracy. [Online] https://www.kaggle.com/c/m5-forecasting-accuracy/. (Accessed 06 June 2021).

Kaggle (2020g). M5 forecasting - uncertainty. [Online] https://www.kaggle.com/c/m5-forecasting-uncertainty/. (Accessed 06 June 2021).

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3146–3154.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021a). The M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, submitted for publication.

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Chen, Z., Gaba, A., Tsetlin, I., et al. (2021b). The M5 uncertainty competition: Results, findings, and conclusions. *International Journal of Forecasting*, submitted for publication.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems, Vol. 32* (pp. 8024–8035). Curran Associates, Inc.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). CatBoost: unbiased boosting with categorical features.