Machine Learning for Time-Series Forecasting: Theory and Practice

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

[a]*Northquay Capital, TrueFit.AI*
[b]*SAS Campus Drive, SAS Institute, Inc., Cary, NC 27513 USA*

Abstract

Deep neural networks and gradient boosted tree models have swept over the field of machine learning over the past decade, producing across-the-board advances in performance. The ability of these methods to capture features interactions and non-linearities makes them exceptionally powerful and, at the same time, prone to overfitting, leakage, or 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, and with the proper theoretical grounding, constitute the current best practices in time-series forecasting.

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

1. Introduction

Recent years have witnessed explosive growth in 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 availability of high-quality open source packages like XGBoost (Chen & Guestrin, 2016), LightGBM (Ke, Meng, Finley, Wang, Chen, Ma, Ye & Liu, 2017), and Catboost (Prokhorenkova, Gusev, Vorobev, Dorogush & Gulin, 2017; Dorogush, Ershov & Gulin, 2018) for GBTs and PyTorch (Paszke, Gross, Massa, Lerer,

*Corresponding author
Email addresses: david@truefit.ai (A. David Lander), russ.wolfinger@sas.com (Russell D. Wolfinger)

Bradbury, Chanan, Killeen, Lin, Gimelshein, Antiga, Desmaison, Kopf, Yang, DeVito, Raison, Tejani, Chilamkurthy, Steiner, Fang, Bai & Chintala, 2019) and Tensorflow/Keras (Abadi, Barham, Chen, Chen, Davis, Dean, Devin, Ghemawat, Irving, Isard et al., 2016; Chollet et al., 2015) for NNs. 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, Dong, Socher, Li, Li & Fei-Fei, 2009), that allow 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's easy to fit the perfect model in hindsight.

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

2. M5 Competitions

The M Competitions, organized by Professor Spyros Makridakis and colleagues (Makridakis, Spiliotis & Assimakopoulos, 2021a; Makridakis, Spiliotis, Assimakopoulos, Chen, Gaba, Tsetlin & Winkler, 2021b), provide a well-regarded proving grounds for state-of-the-art time series methods. The most recent M5 Competitions (Kaggle, 2020f,g), hosted on Google's Kaggle machine learning competition platform, enjoyed a particularly large field of over five thousand teams. They were also the first to set up a head-to-head comparison of classic time series methods versus the more modern approaches utilizing NNs and GBTs.

The M5 Competitions 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 different aims: the Accuracy competition featured basic point forecasts for all items and was the most accessible to the widest number of participants, while the Uncertainty competition required participants to forecast sales at twelve different levels of aggregation (from Store-Item all the way up to Overall Sales, Table 1) and extend their forecasts out to nine different levels of uncertainty around their forecasts.

Level	Aggregation	Number of Series
1	Total	1
2	State	3
3	Store	10
4	Category	3
5	Department	7
6	State-Category	9
7	State-Department	21
8	Store-Category	30
9	Store-Department	70
10	Product	3,049
11	Product-State	9,147
12	Product-Store	30,490
Total		42,840

Table 1: M5 Data Aggregation Levels

Forecasts for the Uncertainty competition were scored based on Weighted Scaled Pinball Loss (WSPL), a carefully-designed metric that evaluated the ability of the quantile forecasts to precisely capture quantify the extreme outcomes and range of uncertainty inherent in forecasting. Through weighting to reflect the relative importance of each series and scaling each series by its prior volatility, this metric provided an extremely robust measure of each participant's performance across the full range of quantiles and levels of aggregation. WSPL is computed as follows:

$$SPL(u) = \frac{1}{h} \frac{\sum_{t=n+1}^{n+h} [(y_t - f_t(u))u\mathbf{1}(f_t(u) \leq y_t) + (f_t(u) - y_t)(1-u)\mathbf{1}(f_t(u) > y_t)]}{\frac{1}{n-1}\sum_{t=2}^{n} |y_t - y_{t-1}|}$$

$$WSPL = \sum_{i=1}^{42{,}840} \frac{w_i}{9} \sum_{j=1}^{9} SPL(u_j)$$

where y_t is the observed value of the time series at time t, $f_t(u)$ the generated forecast for quantile u at time t, h the forecasting horizon, n the length of the training sample, and $\mathbf{1}$ the indicator function. The quantile u ranges over nine values $[0.005, 0.025, 0.165, 0.25, 0.5, 0.75, 0.835, 0.975, 0.995]$, offering a wide and sensible range of worst to best case scenarios around the median forecast. Similar to the M5 Accuracy competition, the weights w_i are computed based on the last 28 days of the final training set and are based on the cumulative actual dollar sales for the ith series (sum of units sold multiplied by their respective price).

As shown in Table 2, our team (Everyday Low SPLices) was able to achieve a noticeably lower WSPL than all other competitors using the methods described in the following sections. This 3% gap is especially large given we observed 0.5% to be threshold for statistical significance for a single quantile of a single aggregation level during our feature selection and parameter tuning process, and that participants were forecasting an inherently uncertain future sales for products with substantial levels of intermittency.

			Percent
Rank	Team	WSPL	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

Table 2: M5 Uncertainty Top Ten Leaderboard

Winning solutions had to not only predict the overall level of sales and any trends, but also specify the full range of uncertainty around each series, very closely matching real-world forecasting demands, where the ability to predict edge cases is just as, or more important, than having a single point forecast of trends. Figure 1 displays a couple of example forecasts from our GBT models, including the quantile uncertainty predictions.

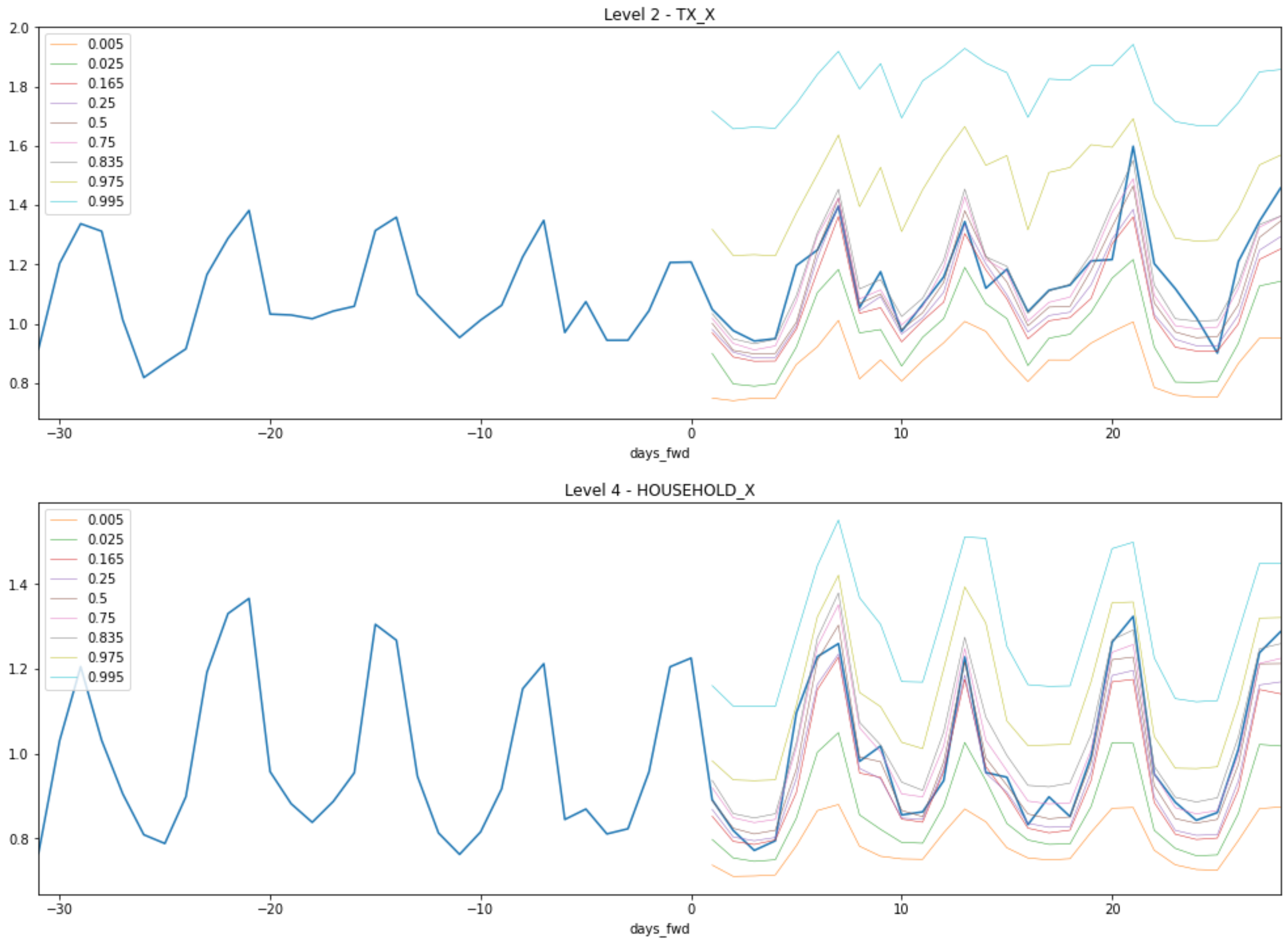


Figure 1: Sample forecasts for next 28 days sales for all nine quantiless

3. Cross Validation

Machine learning is the process 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's more than possible to achieve perfect accuracy in classifying images, the future is inherently unknowable and most time-series like item-level sales 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 of the dataset, train on four of the five and measure performance on the remaining fifth, 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 related (e.g. multiple images of the same object), these items are typically assigned to the same fold, to provide a reliable estimate of model accuracy.

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. fifteen days forward versus thirty 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're often working with a very limited number of truly independent observations (e.g. the number of thirty day periods in a five-year sample), and at no point can we make the typical assumption inherent in almost all large datasets that our data are independent and identically distributed.

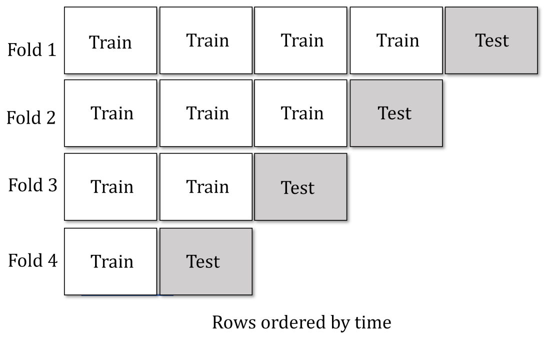
3.1. Time-Series Split

The earliest and simplest mechanism designed to tackle the challenges inherent in time-series CV is the Time Series Split, as shown in Figure 2a. While this solves some of the aforementioned challenges, it faces major issues with regard to data utilization, model diversity, hyper-parameter 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.

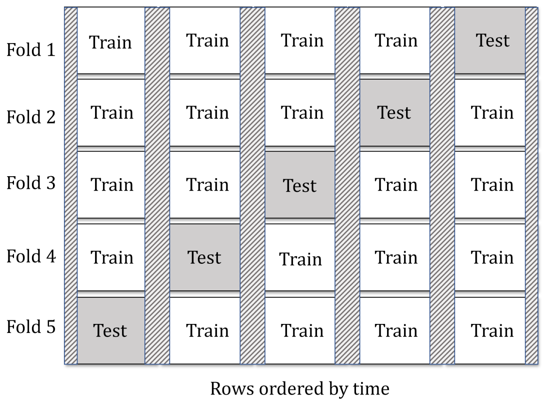
Even with clever tricks to avoid some of these pitfalls, the result of any Time-Series Split is that valuable data goes unused or underweighted, leaks between folds are common, and the architectures and hyperparameters selected will rarely be optimal for full-set training.

Figure 2: Cross Validation Fold Schemes for Time Series

(a) Time-Series Split



(b) Purged K-Fold



3.2. Purged K-Fold

The solution to this CV 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. For a fuller discussion of this technique, see De Prado (2018).

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 and hobbies categories and 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 Figure 2b. 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, knowing for sure that it represented a true improvement in forecasting performance.

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, 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's essential to use nested cross-validation. This entails taking each of the training folds, and searching for the best hyperparameters through internal cross-validation process, so that the remaining holdout data is truly a validation set. While this produces fewer and more modest gains in validation performance, it ensures that any that occur are also very likely gains in true *generalization* performance, and not overfit to this specific set of folds.

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 across five very different folds, as in M5

Uncertainty. In this section we focus on the main aspects for effectively using GBT models for time series.

4.2. Model Tuning

The benefit of a robust cross-validation strategy is that it becomes safe to perform a very precise fit for each model, and it is possible to do so nearly automatically. We conducted a full randomized search across nearly all material hyperparameters for LightGBM (Ke et al., 2017), performed independently across the nested sub-folds of each fold, for each model. Our specification in Python for RandomizedSearchCV (Scikit-learn, 2018) is as follows:

```
lgb_quantile_params = {
    'max_depth': [10, 20],
    'n_estimators': [200, 300, 350, 400],
    'min_split_gain': [0, 0, 0, 0, 1e-4, 1e-3, 1e-2, 0.1],
    'min_child_samples': [2, 4, 7, 10, 20, 40, 60, 80, 100, 150, 200, 300, 500, 1000],
    'min_child_weight': [0, 0, 0, 0, 1e-4, 1e-3, 1e-3, 1e-3, 5e-3, 2e-2, 0.1],
    'num_leaves': [20, 30, 30, 30, 50, 70, 90],
    'learning_rate': [0.02, 0.03, 0.04, 0.04, 0.05, 0.05, 0.07],
    'colsample_bytree': [0.3, 0.5, 0.7, 0.8, 0.9, 0.9, 0.9, 1, 1, 1, 1, 1, 1, 1, 1],
    'colsample_bynode':[0.1, 0.15, 0.2, 0.2, 0.2, 0.25, 0.3, 0.5, 0.65, 0.8, 0.9, 1],
    'reg_lambda': [0, 0, 0, 0, 1e-5, 2e-5, 3e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10, 100],
    'reg_alpha': [0, 1e-5, 3e-5, 1e-4, 1e-3, 3e-3, 1e-2, 0.1, 1, 1, 10, 100, 1000],
    'subsample': [0.9, 1],
    'subsample_freq': [1],
    'cat_smooth': [0.1, 0.2, 0.5, 1, 2, 5, 7, 10],
}
```

Best models from each subfold search are then saved and used to predict both the holdout fold and the test set, and these predictions are subsquently averaged to form the final forecasts.

4.3. Feature Engineering

Many fields of modern machine learning increasingly involve minimal if any feature engineering, with image classification and text processing increasingly being reduced to (a) put the data into a deep learning model, (b) optional: tune 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)

(d) The standard neural network technique known as 'early stopping' once performance is optimized for the validation set produces the exact issue we worked so hard to avoid in designing our CV: that of avoiding illusory gains overfit to any single year or season.

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.

4.4. Feature Importance

A consistent set of features was used for all targets forecast in this competition, but as Figure 3 reveals, each level of aggregation and quantile range shows distinct patterns of importance of each feature to our GBTs. At the level of overall store sales, in forecasting 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 store identified and the local snap patterns of the matching state. Moving down to individual product sales, 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 individual store-item sales, the feature importance for the lower quantiles is dominated by the number of nonzero-sales days over recent weeks.

4.5. Augmentation

When working with image data, there is a fairly standard set of augmentations, such as those implemented in the popular packages Albumentations (Buslaev, Iglovikov, Khvedchenya, Parinov,

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

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.

Druzhinin & Kalinin, 2020) and RandAugment (Cubuk, Zoph, Shlens & Le, 2020). Each operation is designed to take an image and produce a variant that may look vastly different (stretched, cropped, flipped, rotated, or even with random 'cut-outs') but should still belong to the same class of image.

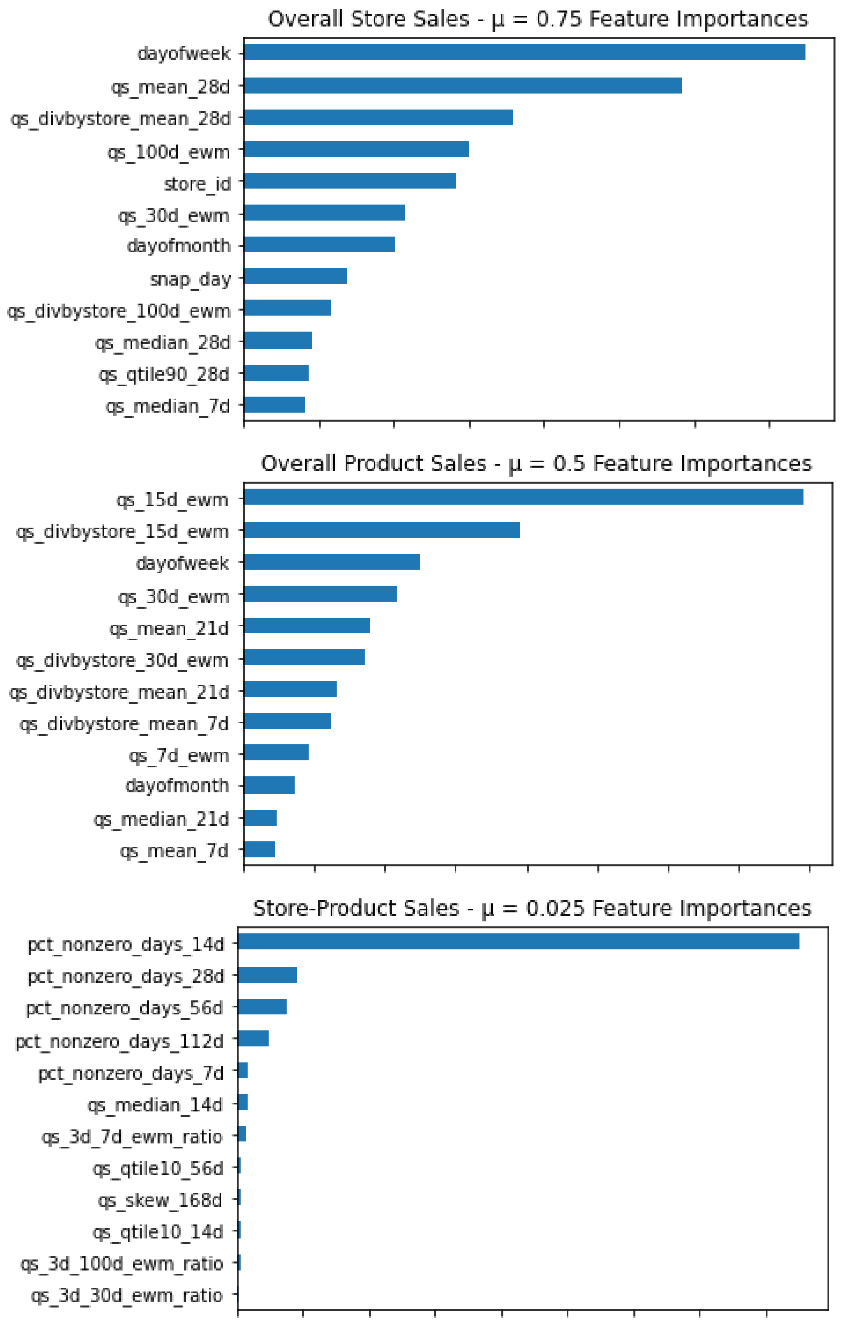
Quantitative augmentation, in contrast, is usually quite limited. An image can have its brightness turned up 200% and still be recognizable; a trailing sales average series cannot quite be increased 200% and still validly predict its target...without also scaling the target. This key insight is what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

The basic idea behind what we will 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.

Figure 3: Feature Importances (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)



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 categorical features) by a common amount that is also applied to the target. This transformation is at once both 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.

Figure 4 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
```

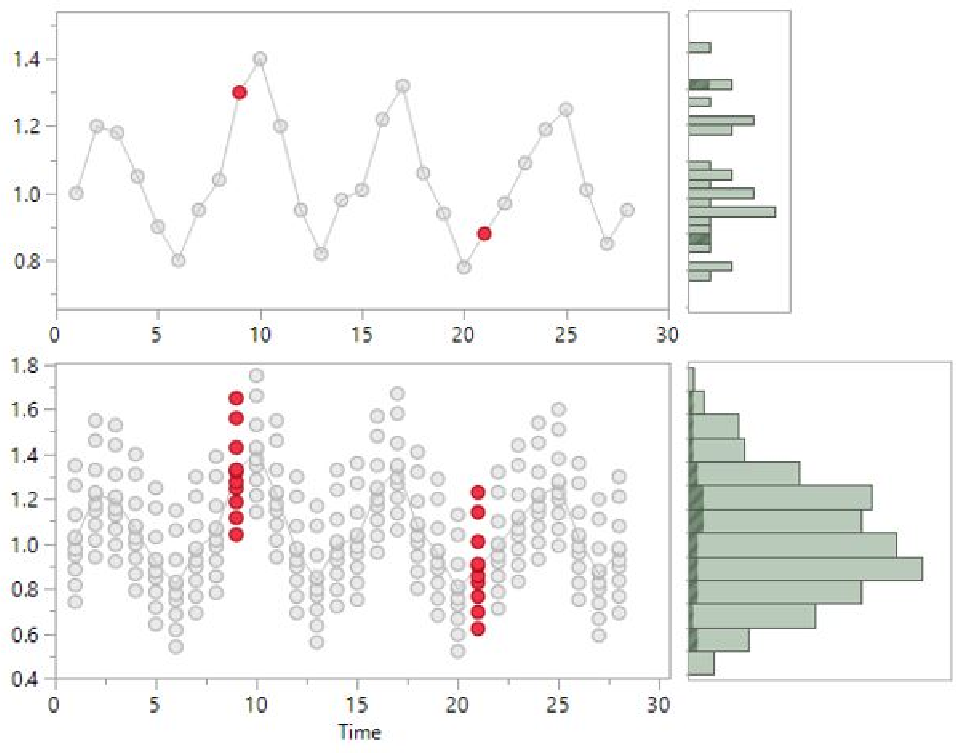
Gradient-boosted trees are known as a less-than-smooth modeling method, given each data point is typically placed in only one of 255 bins for each feature. With range-blending, these independent observations are instead allowed to smooth and spread across dozens of bins, resulting in a much more accurate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

4.6. 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 very 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.

Figure 4: 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 in multiple bins, smoothing out noisy splits and making the model more robust to overfitting



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—while trained over a few hundred hours—could also be trained in a single digit number of hours with performance that would *still* produce the winning solution to this competition.

5. Additional Competitions

The techniques discussed in this article are not limited in efficacy to this particular competition; they are the culmination and refinement of successful methods employed in Kaggle time-series competitions over the past few years, as listed in Table 4.

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,b,c,d)	371	Cases, Fatalities
COVID19 Global Week 5 (Kaggle, 2020e)	173	Cases, Fatalities

Table 4: Kaggle Forecasting Competitions

Over the course of 2020, variations on these ideas were used to win multiple rounds of COVID19 Global Forecasting (Kaggle, 2020a,b,c,d,e), 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.

6. Conclusions

The M5 Competitions, this Special Issue of the *IJF*, and our article here 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 doesn't exist yet, and a strong theoretical grounding is necessary to build models that fully generalize while avoiding overfitting to 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's possible to more than overcome these and produce models that are simple to train and predict the future reliably.

The techniques we've discussed have produced, often by large margins, the 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 detail are available at `http://github.com/david-1013/m5`

Acknowledgements

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., Devin, M., Ghemawat, S., Irving, G., Isard, M. 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*. URL: `https://www.mdpi.com/2078-2489/11/2/125`. doi:`10.3390/info11020125`.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '16 (pp. 785–794). New York, NY, USA: ACM. URL: `http://doi.acm.org/10.1145/2939672.2939785`. doi:`10.1145/2939672.2939785`.

Chollet, F. et al. (2015). Keras. URL: `https://github.com/fchollet/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*, . URL: `https://covid19forecasthub.org/`. doi:`https://doi.org/10.1101/2021.02.03.21250974`.

Cubuk, E., Zoph, B., Shlens, J., & Le, Q. (2020). Randaugment: Practical automated data augmentation with a reduced search space. (pp. 3008–3017). doi:`10.1109/CVPRW50498.2020.00359`.

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. URL: `https://arxiv.org/abs/1810.11363`.

Kaggle (2010). Kaggle Competitions. `https://www.kaggle.com/competitions`. [Online; accessed 06-June-2021].

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

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

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

Kaggle (2017). Wikipedia Web Traffic Time Series Forecasting. `https://www.kaggle.com/c/web-traffic-time-series-forecasting`. [Online; accessed 06-June-2021].

Kaggle (2018a). Corporacion Favorita Grocery Sales Forecasting. `https://www.kaggle.com/c/favorita-grocery-sales-forecasting`. [Online; accessed 06-June-2021].

Kaggle (2018b). Recruit Restaurant Visitor Forecasting. `https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting`. [Online; accessed 06-June-2021].

Kaggle (2018c). Zillow Prize (Phase 1). `https://www.kaggle.com/c/zillow-prize-1`. [Online; accessed 06-June-2021].

Kaggle (2019). Zillow Prize (Phase 2, Final). `https://www.kaggle.com/c/zillow-prize-2`. [Online; accessed 06-June-2021].

Kaggle (2020a). COVID19 Global Forecasting (Week 1). `https://www.kaggle.com/c/covid19-global-forecasting-week-1/`. [Online; accessed 06-June-2021].

Kaggle (2020b). COVID19 Global Forecasting (Week 2). `https://www.kaggle.com/c/covid19-global-forecasting-week-2/`. [Online; accessed 06-June-2021].

Kaggle (2020c). COVID19 Global Forecasting (Week 3). `https://www.kaggle.com/c/covid19-global-forecasting-week-3/`. [Online; accessed 06-June-2021].

Kaggle (2020d). COVID19 Global Forecasting (Week 4). `https://www.kaggle.com/c/`

covid19-global-forecasting-week-4/. [Online; accessed 06-June-2021].

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

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

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

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*, 3146–3154.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021a). The m5 accuracy competition: Results, findings, and conclusions. *Submitted to International Journal of Forecasting*, *0*, 0 – 0.

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Chen, Z., Gaba, A., Tsetlin, I., & Winkler, R. (2021b). The m5 uncertainty competition: Results, findings, and conclusions. *Submitted to International Journal of Forecasting*, *0*, 0 – 0.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). CatBoost: unbiased boosting with categorical features. URL: https://arxiv.org/abs/1706.09516.

Scikit-learn (2018). RandomizedSearchCV. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

Scikit-learn (2020). permutation_importance. URL: https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html.