Demand Planning with Machine Learning: The Tradeoffs Faced by Business Users Today

by

Jyoti Kumari, Kevin Lu, Kanika Mahajan, Jesse Miller, and Estella Ndiranasy

Faculty Advisors: Alberto Todeschini, PhD and Fred Nugen, PhD

Industry Advisors: Eric Wilson and The Institute Of Business Forecasting & Planning

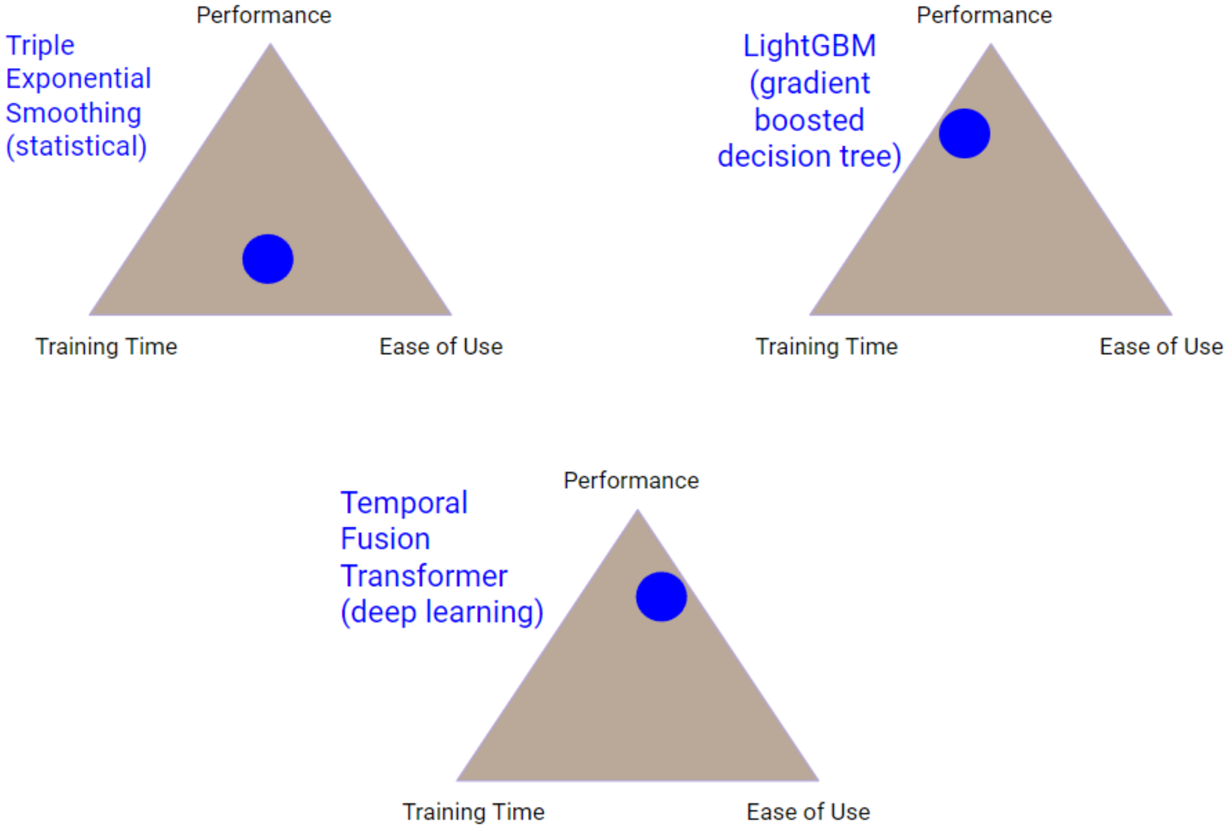
MASTER OF INFORMATION & DATA SCIENCE AT UC BERKELEY
Dec 2022

OVERVIEW

The application of new technology for demand planning continues to generate excitement among business leaders. In fact, an August 2022 report by McKinsey & Company listed demand planning as the #1 priority among global supply chain leaders for new technology initiatives.[1] However, many business users continue to be unfamiliar with the technology available today, and the same McKinsey report cited an "acute shortage of talent" as one of the top obstacles hindering the application of new technology for demand planning.

To address this obstacle, our team gained first-hand experience applying leading tools & techniques for demand planning in the business context. We found that statistical (structured) models, gradient boosted decision tree models, and deep learning models each bring specific pros and cons that business users should be aware of.[2] The trade-offs center around performance, training time, and ease of use (see figure 1). To effectively apply these models, business users should understand the nuances of these trade-offs, which we discuss in this paper.

Figure 1. Overview of model tradeoffs


Triple Exponential Smoothing (statistical)
Performance
Training Time
Ease of Use
LightGBM (gradient boosted decision tree)
Performance
Training Time
Ease of Use
Temporal Fusion Transformer (deep learning)
Performance
Training Time
Ease of Use

[1] Supply chain disruption and resilience | McKinsey
[2] Machine learning in M4: What makes a good unstructured model? - ScienceDirect

DATASET

To perform our analysis, we selected the Walmart Sales dataset from the M5 Forecasting Competition.[3] We selected this dataset for 3 reasons: 1) it is a well-known dataset among demand planning practitioners; 2) it is a large & diverse dataset (covering 3049 products in 10 stores over 5 years), which supports our objective of evaluating application in a "typical" business context; 3) it includes important real world elements faced by business users, specifically external features (price/promo) and scoring on multiple aggregation levels..

MODELS

We explored a number of models throughout our project, but focus on 3 in this paper. The 3 models are Triple Exponential Smoothing, LightGBM, and Temporal Fusion Transformer. They respectively cover the categories of statistical (structured) models, decision tree models, and deep learning models.

These categories (statistical, decision tree, and deep learning) were identified and selected based on our review of literature and discussions with professors, classmates, and business experts. We found the categories to be well-known as the most suitable for demand planning.

Within these three categories, we selected models that, based on our research, we believed would deliver the best performance on the dataset. However, we only selected models that can readily be implemented by a business user with basic knowledge of Python. For deep learning models in particular, we therefore eliminated models that are potentially more "state-of-the-art", but are not available in commonly used python packages and would therefore be highly difficult for a typical business user to employ.

The code for our models can be found at the Github page linked in the footnotes.[4]

Triple Exponential Smoothing (statistical)

We selected this model because it is a well known statistical model that achieved the top results among the 16 statistical benchmarks in the M5 competition.

LightGBM (gradient boosted decision tree)

We selected this model because it was the star of the M5 competition, dominating the top ranks.[5] It is important to note that decision tree models can exhibit substantially different performance depending on the features and hyperparameters selected. This is different from triple exponential smoothing, which only takes one feature (the historical sales), and has just a few hyperparameters which can be selected by autofit. The LightGBM model that we built

[3] M5 Forecasting - Accuracy | Kaggle
[4] jmiller558/Capstone (github.com)
[5] Forecasting with trees - ScienceDirect

leveraged code and blog posts from the top M5 competitors, and is therefore representative of a top-tier implementation with optimal features and hyperparameters..

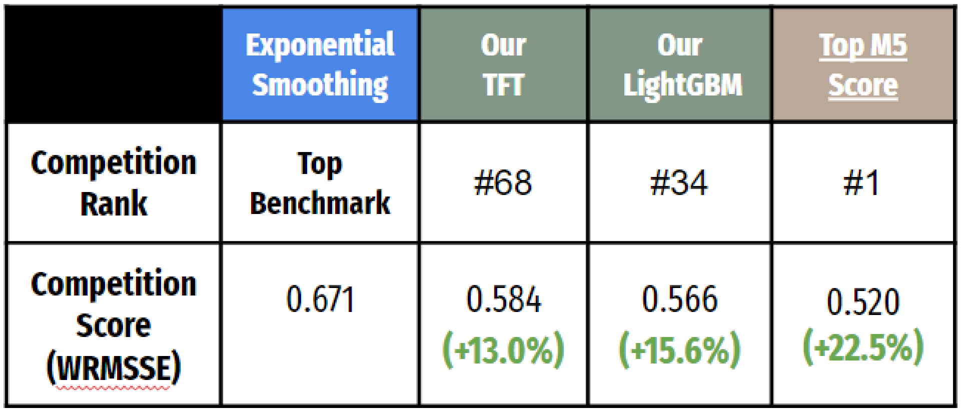
Temporal Fusion Transformer (deep learning)

We selected this model through an exploration of deep learning models available in commonly used python packages.[6][7][8][9] It is important to note that similar to gradient boosted decision tree models, deep learning models can exhibit substantially different performance depending on the features and hyperparameters selected. Temporal Fusion Transformer was relatively new at the time of the M5, and we were unable to find any code or descriptions of an implementation on the M5 dataset. Our implementation was therefore essentially “from scratch”, and thus at a disadvantage compared to our LightGBM implementation.

PERFORMANCE RESULTS

In our results (using the same test data and scoring metric employed by the M5), we found LightGBM to deliver the best results, followed by Temporal Fusion Transformer as a close 2nd, and Triple Exponential Smoothing further behind. The M5 competition had over 5K entries, putting our results in the top 1%. In the table below we have also included the score of the first place entry in the M5 for reference.

Figure 2. Performance Results (Results for LightGBM and Temporal Fusions Transformer include the % improvement over Triple Exponential Smoothing)

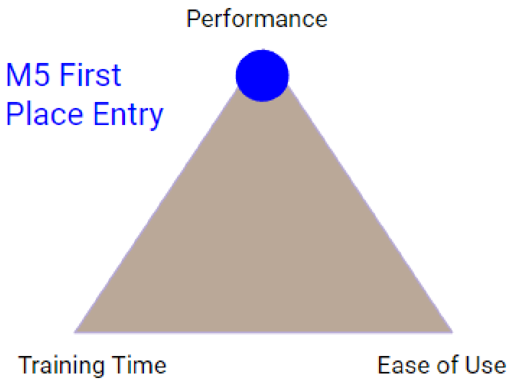


	Exponential Smoothing	Our TFT	Our LightGBM	Top M5 Score
Competition Rank	Top Benchmark	#68	#34	#1
Competition Score (WRMSSE)	0.671	0.584 (+13.0%)	0.566 (+15.6%)	0.520 (+22.5%)

The first place entry in the M5 used a weighted combination of 220 LightGBM models. It represents an implementation that is optimized for performance (vs. ease of use or training time). As we limited our study to models that can reasonably be implemented by a typical business user, we did not attempt to recreate this ensemble entry.

[6] GluonTS documentation
[7] PyTorch Forecasting Documentation — pytorch-forecasting documentation
[8] Nixtla/neuralforecast: Scalable and user friendly neural forecasting algorithms. (github.com)
[9] Time Series Made Easy in Python — darts documentation (unit8co.github.io)

Figure 3. Overview of the first place M5 model

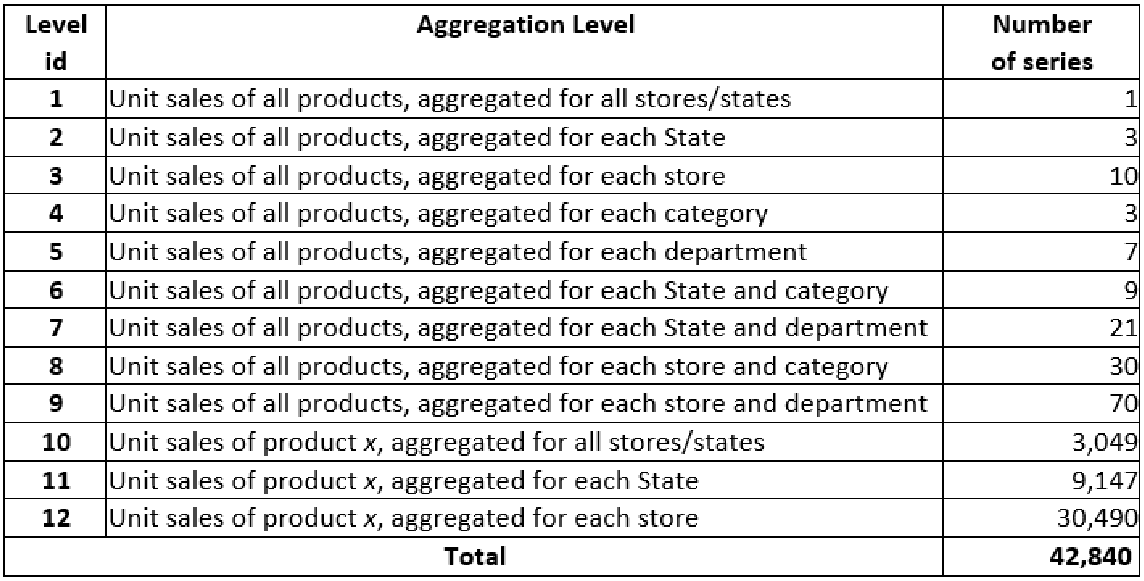


Performance at Different Aggregation Levels

The results of the M5 competition were widely publicized in the demand planning community with the tagline that machine learning models (gradient boosted decision tree & deep learning) had outperformed statistical models by 20+%. However, in our conversations with business users they immediately questioned whether these results are consistent across all of the M5 aggregation levels.

In the M5 competition, forecasts are submitted at the bottom level (level 12) and then aggregated up and scored at 11 additional aggregation levels. Competitor's overall M5 score is based on an equally weighted average of performance over all 12 aggregation levels.

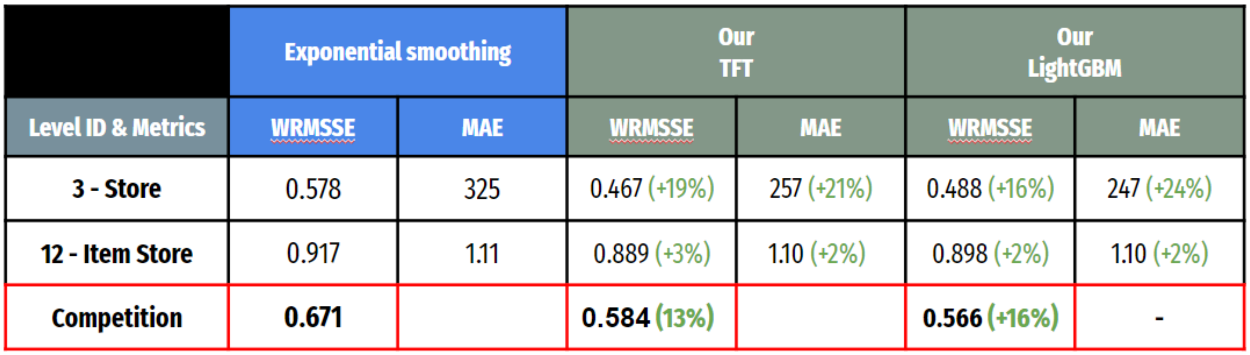
Figure 4. Different aggregation levels in the M5 competition



Level id	Aggregation Level	Number of series
1	Unit sales of all products, aggregated for all stores/states	1
2	Unit sales of all products, aggregated for each State	3
3	Unit sales of all products, aggregated for each store	10
4	Unit sales of all products, aggregated for each category	3
5	Unit sales of all products, aggregated for each department	7
6	Unit sales of all products, aggregated for each State and category	9
7	Unit sales of all products, aggregated for each State and department	21
8	Unit sales of all products, aggregated for each store and category	30
9	Unit sales of all products, aggregated for each store and department	70
10	Unit sales of product *x*, aggregated for all stores/states	3,049
11	Unit sales of product *x*, aggregated for each State	9,147
12	Unit sales of product *x*, aggregated for each store	30,490
Total		**42,840**

When we examined performance at each of levels, we found that it was **not** consistent. Our LightGBM and Temporal Fusion Transformer models performed only slightly better than Triple Exponential Smoothing at the bottom levels. Their strong performance in the overall score was driven primarily by their ability to aggregate up well.

Figure 5. Model performance at different levels



	Exponential smoothing		Our TFT		Our LightGBM	
Level ID & Metrics	WRMSSE	MAE	WRMSSE	MAE	WRMSSE	MAE
3 - Store	0.578	325	0.467 (+19%)	257 (+21%)	0.488 (+16%)	247 (+24%)
12 - Item Store	0.917	1.11	0.889 (+3%)	1.10 (+2%)	0.898 (+2%)	1.10 (+2%)
Competition	**0.671**		**0.584 (13%)**		**0.566 (+16%)**	-

While there is important business value in having forecasts that aggregate-up well, a business user seeking to only forecast the bottom level of this dataset might be better served by Triple Exponential Smoothing given its faster training time and ease of use.

In addition, we would caution against relying on Gradient Boosted Decision Tree and Deep Learning models to consistently aggregate-up well. Research done after the M5 found that top performing entries in the competition did not consistently aggregate-up well across different time periods in the dataset.[10] We observed similar results when we altered the M5 dataset to forecast in weekly buckets (discussed further in the subsequent section below). Simply put, chance seems to play an important role in the models' ability to aggregate-up well, resulting in different performance on different test periods and/or datasets.

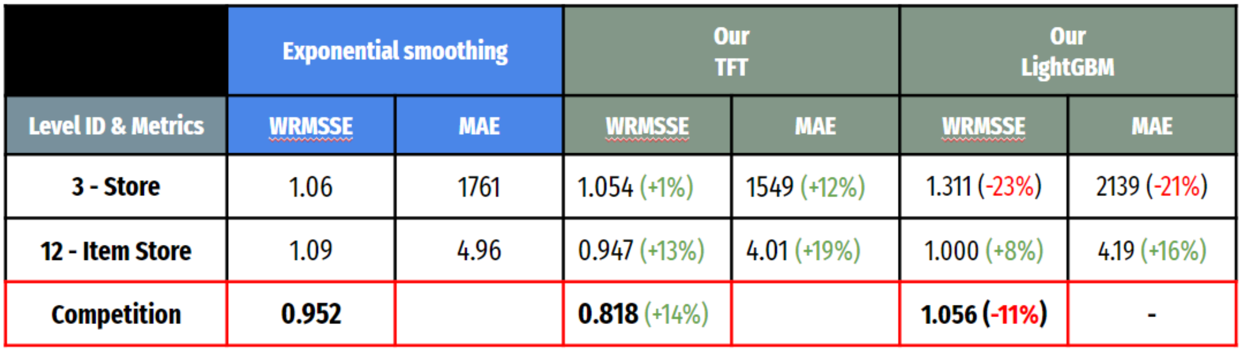
For business users seeking to forecast multiple aggregation levels, we would recommend using both a bottoms-up and a top-down approach. This methodology was used by the second place entry in the M5, and is further described in a paper written by that competitor (see footnotes).[11] Essentially one would use a Gradient Boosted Decision Tree or Deep Learning model at the bottom level, and also create other forecasts generated at the top aggregation levels to check for alignment. One can generate multiple permutations of the bottom-level forecast by varying the hyperparameters, and then select the one that aligns best with the top level forecasts.

Performance in Different Business Scenarios (M5 Converted to Weekly)

In our conversations with business users, one widely raised question was whether Gradient Boosted Decision Tree and Deep Learning models would consistently outperform Triple Exponential Smoothing in other business scenarios not covered by the M5. In the M5, competitors submitted daily forecasts for 28 days. However, business users that we spoke with felt that weekly forecasts for approximately a ~3 month horizon are a more common need.

Based on this, we converted the M5 dataset from a daily to a weekly dataset, and ran forecasts for a 12 week horizon.

Figure 6. Performance results on the weekly M5 dataset over a 12 week horizon



	Exponential smoothing		Our TFT		Our LightGBM	
Level ID & Metrics	**WRMSSE**	**MAE**	**WRMSSE**	**MAE**	**WRMSSE**	**MAE**
3 - Store	1.06	1761	1.054 (+1%)	1549 (+12%)	1.311 (-23%)	2139 (-21%)
12 - Item Store	1.09	4.96	0.947 (+13%)	4.01 (+19%)	1.000 (+8%)	4.19 (+16%)
Competition	**0.952**		**0.818** (+14%)		**1.056 (-11%)**	-

We found that both models further outperformed Triple Exponential Smoothing at the bottom-level, but no longer aggregated-up as well. This bolsters the finding by other researchers that the ability of these models to aggregate up well is inconsistent.

[10] The performance of the global bottom-up approach in the M5 accuracy competition: A robustness check - ScienceDirect
[11] Hierarchical forecasting with a top-down alignment of independent-level forecasts - ScienceDirect

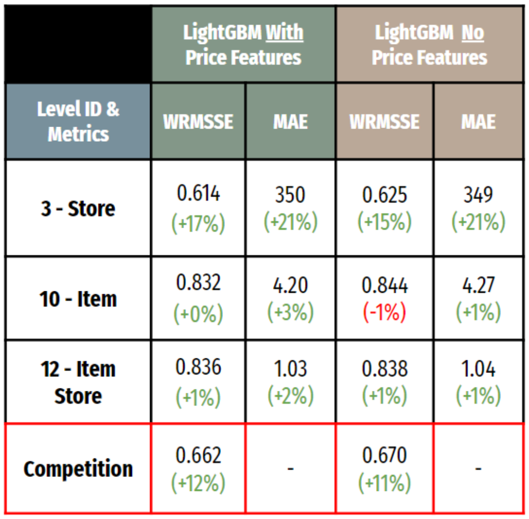
LightGBM in particular exhibited poor performance aggregating-up on the weekly dataset. One important contributing factor to this result may be that our LightGBM implementation was built with features and hyperparameters highly optimized for performance on the original daily M5 dataset. It was a complex and cumbersome model (further described in the below section on ease of use), and therefore we may not have adequately adapted its features and hyperparameters to the weekly dataset.

The Role of External Features (Price/Promo) in Performance

The final element of performance that we explored centered on the models' ability to derive insights from external features, specifically the price/promo data in the M5 dataset. Most statistical models like Triple Exponential Smoothing can only take 1 input feature - the historical sales. This potentially puts these models at a performance disadvantage on datasets with many features available. Price/promo features are particularly important to demand planners as they are critical to sales outcomes in many companies.

We found, however, that in the M5 dataset, the pricing/promo features actually played a limited role. When we removed these features, our LightGBM model's performance was reduced by only ~1%.

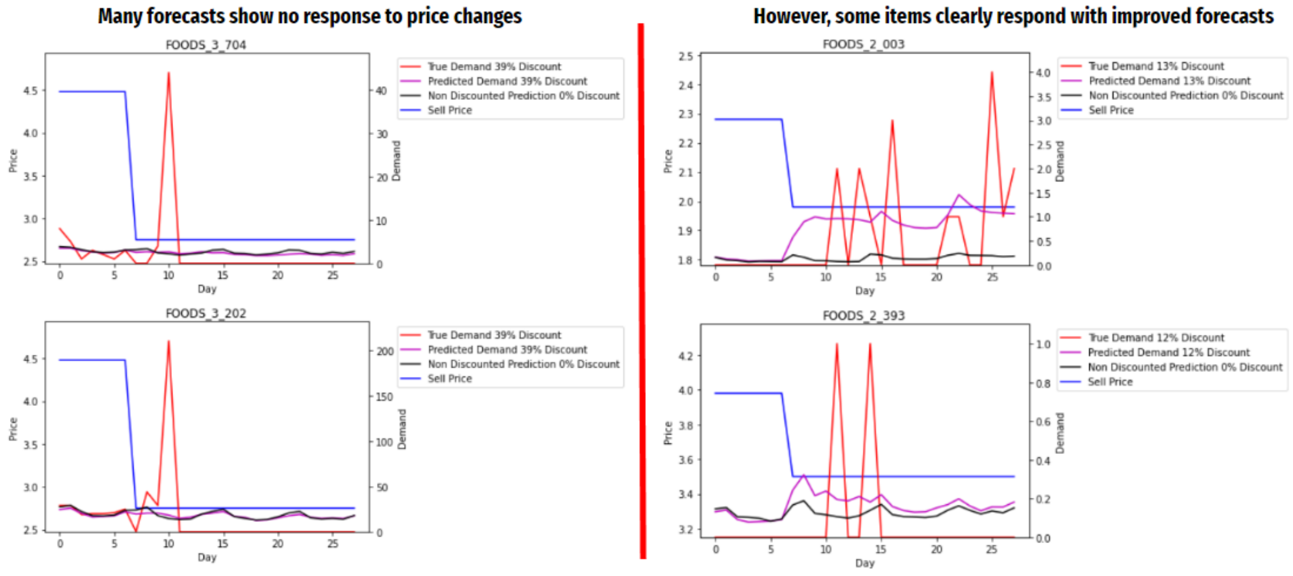
Figure 7. LightGBM results with and without price features



	LightGBM With Price Features		LightGBM No Price Features	
Level ID & Metrics	WRMSSE	MAE	WRMSSE	MAE
3 - Store	0.614 (+17%)	350 (+21%)	0.625 (+15%)	349 (+21%)
10 - Item	0.832 (+0%)	4.20 (+3%)	0.844 (-1%)	4.27 (+1%)
12 - Item Store	0.836 (+1%)	1.03 (+2%)	0.838 (+1%)	1.04 (+1%)
Competition	0.662 (+12%)	-	0.670 (+11%)	-

The limited impact of the price features is likely due to the fact that price changes were relatively rare in the M5 dataset. When we further examined the items with price changes we found evidence that at least for some of the items, the model was taking the price changes into account and improving its forecasts. This indicates that for a dataset with more price changes, the Gradient Boosted Decision Tree and Deep Learning Models might further outperform Triple Exponential Smoothing.

Figure 8. LightGBM response to pricing changes at the item level



TRAINING TIME RESULTS

With our focus on the tradeoffs faced by typical business users, we did not leverage any complex computational infrastructure. All training was done on Google Colab Pro+, which is a cloud computing platform with a simple user interface and simple billing structure (flat rate of ~$50 per month). A typical business user can set up Google Colab Pro+ in ~5-10 minutes. The training time reported below is based on a single Colab instance, but in practice, a Pro+ subscription allows the user to run multiple instances, and we typically would split our dataset and run two instances, thus cutting the training time in half.

Triple Exponential Smoothing (statistical)

We ran this model with little effort to optimize the code, and found the run time to take several hours. However, others have reported being able to run the model on the M5 dataset in as little as ~15 minutes.[12]

LightGBM (gradient boosted decision tree)

Our run time for LightGBM was ~20 hours. As mentioned above, in practice this was cut in half to ~10 hours by using multiple instances.

Temporal Fusion Transformer (deep learning)

Our run time for Temporal Fusion Transformer was ~35 hours. Similarly to LightGBM, in practice this was cut in half to ~17.5 hours by using multiple instances.

[12] https://github.com/Nixtla/statsforecast/tree/main/experiments/amazon_forecast

EASE OF USE RESULTS

Ease of use is difficult to define and quantify as it is both subjective and highly dependent on the rapidly evolving set of tools & packages available on the internet. Nonetheless, when operating these models from the perspective of a typical business user, we found ease of use to be a critical element and felt it was essential to discuss in our analysis.

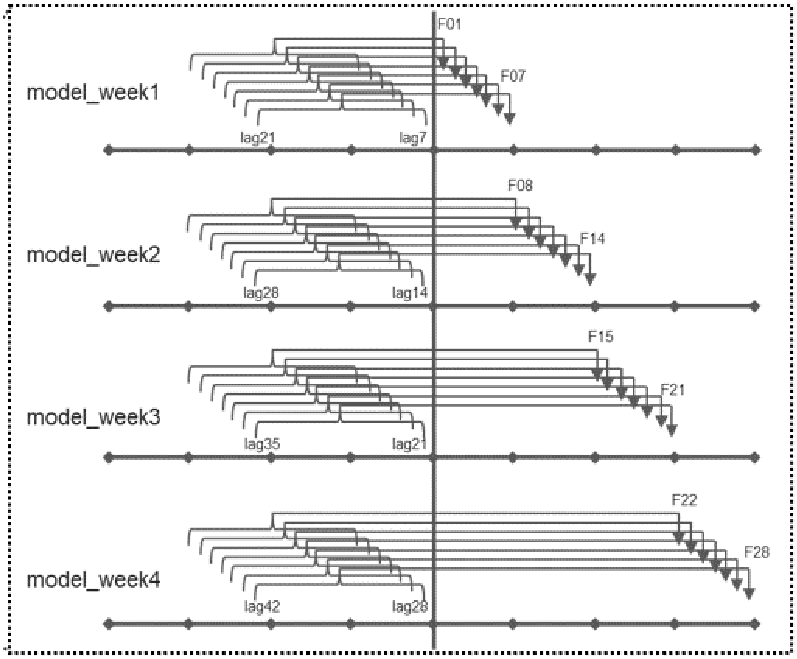
Triple Exponential Smoothing (statistical)

Triple Exponential Smoothing is very easy to use; it requires no feature engineering, and autofit functions available in commonly used python packages select the optimal hyperparameters. As of the writing of this paper, Triple Exponential Smoothing is simply in a different league for ease of use compared to decision tree and deep learning models, and this is unlikely to change in the near future.

LightGBM (gradient boosted decision tree)

Leveraging blog posts and code from top M5 competitors, we developed a LightGBM model that actually ran 4 separate forecasts using different lag features (see figure below).

Figure 9. Structure of our LightGBM model[13]



This structure helped to achieve top level performance, but also made it complicated and cumbersome to operate and interpret. Overall, we found that working with LightGBM was difficult as the package was built for general use, as opposed to specific use on time-series

[13] M5 Forecasting - Accuracy | Kaggle

forecasting. That said, a LightGBM package built specifically for time-series forecasting could potentially be developed in the near-future, if it doesn't already exist.

Temporal Fusion Transformer (deep learning)

While Temporal Fusion Transformer is complex compared to triple exponential smoothing, the python package we used was built specifically for time-series forecasting, and therefore came with a lot of elements that helped with ease of use for demand planning.

In contrast to LightGBM, which required a complex setup with 4 separate forecasts using different lags created by feature engineering, Temporal Fusion Transformer handles lag features automatically. The user simply specifies the overall size of the lag window for the model to look at (for example the past 56 days), and the model does all the rest.

Overall we found Temporal Fusion Transformer to be easy to use. However, in contrast to LightGBM, there were fewer implementations & blog posts publicly available on the internet to use as reference. As that body of knowledge grows, ease of use will continue to be enhanced.

CONCLUSIONS & RECOMMENDATIONS FOR BUSINESS USERS

Given the model tradeoffs described above, we would recommend different approaches for different business use cases.

Forecasting a single aggregation level

If the user is only forecasting a single aggregation level, we would recommend exploring statistical models first, followed by deep learning models. This stands in contrast to the perspective of some thought leaders who believe that "Deep learning is what you do not need unless you have tons of clean data and tens of top PhDs working on forecasting".[14]

It is true that with limited data, deep learning models may show limited improvement over statistical models. However, tens of top PhDs are by no means required, and even with limited data it may be worth at least trying deep learning models. Our team of first-time users with limited reference code found that Temporal Fusion Transformer outperformed the leading statistical benchmark on the original M5 dataset, and that the gap widened on the weekly version. For datasets with even more external features or price/promo variation, we have reason to believe the performance gap would widen even further.

While the training time for deep learning was much longer, computational power is increasingly cheap & easy to access with Google Colab Pro+ costing just $50 per month. Finally, with the python packages available today, we found Temporal Fusion Transformer to be much easier to use than LightGBM.

[14] **Deep Learning Is What You Do Not Need** | by Valery Manokhin, PhD, MBA, CQF | Medium

Forecasting multiple aggregation levels

If the user is forecasting multiple aggregation levels, we would recommend deep learning for the bottom aggregation level. On both the original and the weekly M5 dataset, we found that Temporal Fusion Transformer aggregated-up better than Triple Exponential Smoothing.

However, we would caution users not to rely on these results, and to take additional steps to ensure that forecasts aggregate-up well. Specifically, we would recommend users also generate forecasts at the top levels, and check the bottoms-up aggregation for alignment. To further optimize, users can follow similar steps to the 2nd place M5 competitor, generating multiple bottom level forecasts with variations in hyperparameters, and selecting the one that aggregates up with the best alignment to the top level forecasts.