M5 Forecasting – Accuracy competition
2nd place solution summary

A1. Background

Competition Name: M5 Forecasting – Accuracy competition

Team Name: „Matthias“

Private Leaderboard Score: 0.52816

Private Leaderboard Place: #2

My academic background is aerospace engineering with focus on flight system dynamics and flight control. My professional background is mix of strategy consulting (Bain&Company), building up the tech side of a startup accelerator and working now as an independent technology consultant.

Prior experience regarding time series prediction was limited to standard stuff like Holt-Winters, (S)ARIMA(X), Prophet etc.

Time spent on the competition is hard to estimate as the models run for several hours and I just had to return to the computer every 5-7 hours to start a new batch of Kaggle kernels. I would say in total working days (=8hours) maybe 5 days…

A2. Summary

The overall approach consists of two separate modelling streams on top and bottom of the grouping hierarchy that are aligned to get a result.
It should be reproducible for any given time-frame as well as automatable and therefore valuable for any complex grouped/hierarchical time series problems.

For the bottom level one lgb model is trained for each store. This split was chosen based on the hypothesis that local buying patterns will be best learned by this. (Local trends, weather, sports events, religous communities etc.)

The top level model is a NBEATS ensemble trained for all level 1-5 aggregated series.

All models were trained either on Kaggle or on Colab.

For a very short overview please also refer to the public writeup:

https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/164599

A3. Feature selection

For feature engineering/selection we only used features that were available from public notebooks. During initial tests it became obvious that historic sales features (lags of X days, rolling means/stds for last X days) lead to very unstable results.

Our explanation for this is that for intermittent sales data there is mostly noise in those signals. Intuitively it is clear that the information that a peppermint candy was sold once last Monday and twice yesterday does not really provide a lot of information for the model. The information „every Thursday on average over the last four years people tend to buy more peppermint candy than on Monday" is important and learned from datetime features.

Maybe we could have optimized the bottom level predictions by further investigating the features provided by public kernels but due to time restrictions the features were taken as is from this kernel:

https://www.kaggle.com/kyakovlev/m5-simple-fe

The feature transformations for price (min/max/std/mean/norm etc.) were not further investigated. Also the release feature was included based on the fact that it does not need any further calculations during prediction.

We did not use any feature importance information: The experience with lgb feature importance when using categorical features so far was rather .... confusing.

External data was not used.

A4. Training Methods

The bottom level models were trained with a custom asymmetric loss function that allowed to let the model predictions over- or undershoot the ground truth. This is important to later on align the bottom level with the top-level predictions.

For final submission an ensemble of 5 bottom level predictions with different parameters for the asymmetric loss was built. All 5 predictions were equally weighted.

For the top level N-BEATS EnsembleEstimator we changed the standard GluonTS Trainer to incorporate lookahead. We actually never had the time to test if this helped a lot or not but it did not harm so we kept the modified version for all computations.

In early tests we saw that (for this dataset) the results are best when only training the EnsembleEstimator with the sMAPE loss and chose that configuration for all further computations.

To balance compute time with performance we chose to set the bagging_size to 3 and the meta_context_length to [3,5,7] – which leads to a ensemble size of 9 models (taking around 4,5 h to train on Colab)

A batch size of 16 seemed to produce a nice gradient for the top levels (1-5) (For levels further down the grouping this configuration will not work)

A4. Interesting findings

I think the most important aspect was not to try to train a model that incorporated historic sales features. Many participants struggled to find a robust cross-validation for lgb bottom level models that were trained with lag and rolling features – an endeavour that I thought was not possible.

So the alignment with top level N-BEATS predictions is most likely the „most important trick" used.

Overall it will be interesting to see how all submitted solutions perform for timeframes different than the evaluation period.

A5. Model Execution Time

The feature pre-processing notebook took ~9 minutes on Kaggle to complete

One run of the bottom level lgbm notebook to train and predict for a given loss multiplier took around 4.7 hours on Kaggle

One run of the top level N-BEATS notebook to train and predict the levels 1-5 took around 4.5 hours on Colab GPU instance.