**M5 Uncertainty** **Competition**

**Everyday Low SPLices**

1st Place – Private LB: 0.15420

**Background**

David Lander, [landerad@gmail.com](mailto:landerad@gmail.com)

* + Quantitative Hedge Fund Founder
  + Kaggle #6; Solo 1st Place in Covid-19 Global Forecasting Competitions
  + Focused on gradient boosting and M5 Uncertainty

Russ Wolfinger, [russ.wolfinger@jmp.com](mailto:russ.wolfinger@jmp.com)

* + Director of Scientific Discovery and Genomics at SAS
  + Kaggle #10; #1 in Zillow Competition; #4 Rossman Same-Store Sales
  + Focused on neural networks and M5 Accuracy

**Summary**

LightGBM models for all quantiles and levels.

Cross-Validation was GroupKFold by year, excluding the holiday months.

Features included exponentially weighted moving averages, rolling means, medians, quantiles, skewness and kurtosis, and more; along with categorical features, and day of week, day of month, etc.

**Features**

**I. Feature Engineering**

Features were added incrementally, while testing their impact on CV, and later culled and tune by looking at CV across multiple levels and targets.

One notable feature was de-trended sales for each series, i.e. quantity sold divided by average quantity sold for that store, to capture item-level trends.

Gaussian noise was added to the month to derive the ‘season’ feature, which outperformed leave-out or raw month as a feature.

No external data was used. Neither price data nor item\_id were used in forecasting.

**II.** **Feature Importances**

Overall Sales – Mid-Quantile:

A close up of a logo

Description automatically generated

Store Sales – Low Quantile:

A close up of a logo

Description automatically generated

Store Sales – High Quantile:

A close up of a logo

Description automatically generated

Item Sales – Mid Quantile:

A picture containing screenshot

Description automatically generated

Store-Item Sales – Low Quantile:

A picture containing drawing

Description automatically generated

Store-Item Sales – Mid-Low Quantile:

A close up of a logo

Description automatically generated

Store-Item Sales – Mid-High Quantile:

A screenshot of a cell phone

Description automatically generated

Store-Item Sales – High Quantile:

A picture containing drawing

Description automatically generated

**Training Methods**

LightGBM with RandomizedSearchCV

The validation submission used only one model per target—and included no 2016 data. The evaluation submission was a simple linear blend of the five training folds for each target, with a higher weight for the most recent folds.

The final submission reconciled Levels 1-9 mean forecasts and applied an adjustment based on this to all quantiles for these levels.

**Key Techniques**

Range-Blended Gradient Boosting was crucial on the low-n series.

This improved CV by 5% overall, due to a 5-15% improvement on Levels 1-9.

**Observations**

**Metrics**

A simple baseline model without range-blending, just solid feature engineering, hyperparameter searches, and decent command of LGB would have scored ~0.162.

A simple ten-feature model that trains in under a half hour would score ~0.170.

Teams ahead of the 0.162 baseline, clustered from 0.159 to 0.162, have strong models that managed to achieve some form of smoothing.

Teams just above this baseline score have somewhat ‘unsmooth models’ or missing features, or occasional issues with CV or leakage.

Anything much beyond 0.170 seems to show minimal command of the task.

**Hierarchical Forecasting**

Levels 10-12 always had the same CV. Neural networks came in in-line with LGB, and no adjustments made any material difference. Even the 10-feature model only shows 3% higher error after stripping out basically \*all\* relevant information.

The separator in this competition was Levels 1-9—how to do quantile forecasting, or even forecasting generally, on very sparse low-n series: ~1000 days of data, for a one or two digit number of series.

Working with large datasets isn’t actually challenging—our subsampling rarely even included more than 1 million rows in training—the true skill is forecasting when data is sparse.

**Model Types**

**Design**

Full Model: Very high tree and leaf counts, and huge number of parameter searches.

Elegant architecture, but turned all the way up to take a few days.

Most of that compute is for a couple tenths of a percent gain.

Fast Model: One fold, less data, lighter parameter sampling, lower tree and leaf counts.

Achieves +0.5% cross-validation performance and +1.0% leaderboard performance.

Doesn’t actually have \*any\* 2016 data included. May match with tuning.

Sparse Model: 1/10th the data, all prior optimizations, and only 10 features!

Achieves 17th place despite 30 min. runtime and basically no information

This is barely a ‘model’ so clearly the architecture is doing a lot of work here.

**Run-Time**

Full Model: 200 hours training; 10 hours prediction

Fast Model: 5 hours training; 30 min. prediction

Sparse Model: 30 mins total

**Sample Forecasts**

A close up of a logo

Description automatically generated

A close up of a mans face

Description automatically generated

**Conclusions**

After four Covid forecasting competitions and M5—and having a first place solo solution in every time-series competition except my very first one—I would offer the following:

* Good cross-validation strategies are rarer than you think. Many people ‘know’ how to do cross-validation when there are static datasets and you can split it into folds, but for time-series people tend to spazz out and optimize one window, with disastrous results. The only way to work on a problem is to have an intelligent CV.
* Leakage is very common. Some target encode information that isn’t held out, others scale or impute back out-of-sample information, ruining their cross-validation entirely. Time series forecasting requires a certain level of judgment to avoid these errors, that isn’t present in nor developed by simpler modeling schemes where every data point is independent from every other one.
* Often the key technique is squeezing more from less. “Big data” is easy. “Small data” is tricky—you have to truly solve the variance/bias problem and extract maximal signal while treading closely to ‘overfitting’. If you want complex feature interactions, and a rich feature base, you have to design your structure carefully.
* Often the data needs to be brought to the same scale to work for gradient boosting or neural networks, and then weighting is necessary to restore proper weights or even match custom scoring. These are key areas where strong competitors pick better approaches—usually rescaling based on backward-looking information only, and then subsampling to achieving proper weighting and speed are key.
* Time series forecasting is like poker; everyone is ‘sure’ they’re in the top decile, they just got ‘unlucky’. They didn’t. Even in the accuracy competitions, a reasonable CV was 0.55-0.57 for a high-end team; some obviously hit the better or worse side of chance, given a +-0.02/0.03 sd., but most teams far outside that window just didn’t have strong submissions.
* The noise range was around 0.0015 or 1% for this competition, and around 0.025 or 5% for the Accuracy competition—tight enough to see three clusters of gold medal teams behind us, but containing a few dozen teams within error range in Accuracy. A “Kernels” competition that must run on unseen data automatically, or a much longer forecasting window and diversity of forecasting of tasks, could be an interesting approach to a future competition that could truly find a winner for accuracy.

Once again, thank you to all hosts, organizers, and sponsors.