# A1. Background on you/your team

Competition Name: *M5 Forecasting - Uncertainty*

Team Name: [GoodsForecast] *Nick Mamonov*

Private Leaderboard Score: *0.15890*

Private Leaderboard Place: *2*

Name: *Mamonov Nikolay*

Location: *Moscow*

Email: [mnv0895@gmail.com](mailto:mnv0895@gmail.com)

# A2. Background on you/your team

* What is your academic/professional background?

*Master’s degree at Lomonosov Moscow State University in Computational Mathematics and Cybernetics. PhD student at MSU now.*

*Working as Senior Data Scientist at GoodsForecast.*

* Did you have any prior experience that helped you succeed in this competition?

*I worked on the similar tasks at GoodsForecast*

* What made you decide to enter this competition?

*One of directors of my company is active member of forecasting community leaded by Spyros Makridakis. He suggested me to take a part at this competition.*

* How much time did you spend on the competition?

*About two months.*

# A3. Summary

I used hybrid approach with gradient boosting models, simple forecasting time series models and statistics methods.

The final method is based on probability distribution estimation and correction of it with external forecasts.

The method consists of the following steps.

**Step 1. Building base level for forecasting at low level (Python scripts)**

I used Light GMB model for building low-level forecast. I made several experiments on cross-validation periods (with five cross-validation intervals).

At these periods several models were compared. The main models are:

* Simple time series exponential models like Theil-Wage
* Time series models provided by GoodsForecast
* LGMB with simple features (category features + lags)
* Hierarchical approach with building forecast at Store-Department level and distribution it to low level (with LGMB at both levels)
* Recursive LGBM model provided by Konstantin Yakovlev

I’ve found out that Recursive LGBM shows the best cross validation results but only with correction of the total-level of forecast (meaning BIAS).

Sum of all forecasts on private period was about 1.3m. I’ve decided to look at forecast of other models on test period.

**Step 2. Forecasts correction (Python scripts)**

Total level with 1.3m looked a bit overpriced. So, I’ve tested SSA method for Total-level series and simple season coefficients model. SSA showed ~1.23m sum and season coefficients model showed ~1.26m sum. These methods were better in terms of total level series forecasting for cross-validation, so I’ve took average total level from these models. This led to the adjustment of forecasts for the coefficient 0.96.

Also, I’ve noticed that 2011 calendar for May and June looks the same as 2016 calendar. Which is important for predicting SNAP and holiday effects. So, I’ve made additional correction for F7 (coefficient 0.9) - this is weekend before Memorial Day.

**Step 3. Building probability distribution estimation – histogram (exe)**

At the next step, I’ve built histograms for each forecasting day for each time series (meaning at all twelve forecasting levels). This means that we aggregate data for each level (1-11) from low-level series.

“Hist” algorithm (development of GoodsForecast) was used for histogram calculation.

The main idea of the algorithm is weighting time series history points according to forecasting point.

Assume that time series is , where T is time series length (maximum 731 points because of only two years of sales history was used for calculating).

The weight for every point is calculated by multiplying three components:

– is as “forgetting” component decreasing to the beginning of the time series and equal to one at the forecast point.

- is a year season component. As history time series point is close to forecasting point in terms of year seasonality (365 days), the weight has value close to 1.

- is a week season component. As history time series point is close to forecasting point in terms of week seasonality (7 days), the weight has value close to 1.

The weights are calculated with the core functions of “Hist” algorithm. We remove points from with equal to zero.

After that we build histogram. We split interval into subintervals with equal length – histogram bars. Then considering weights as probabilities we calculate bars heights.

The Hist model hyperparameters were selected with grid search.

**Step 4. Shifting histogram to fit forecasts from LGMB (exe)**

It is obvious that histogram does not the shift is made in the following way.

We calculate histogram median (0.5-quantile) for certain forecast of a certain time series. Then we find the difference between LGBM forecast for the same point and median. For upper levels (1-11) forecasts are aggregated from low level. Then we just shift histogram by difference, e. g. the new median of histogram is LGBM forecast.

**Step 5. Histogram post-processing (exe)**

As we build uncertainty estimation for different aggregation levels separately, we may not care about BIAS at top level when we change low-level forecasts. So, I find out that rounding for forecasts is a good strategy for metric decrease. I used Floor rounding for quantiles 0.005-0.5, Round for 0.75 and 0.835 quantiles and Ceil for the rest quantiles.

After that I multiplied some quantiles (0.165, 0.25, 0.75) by correction coefficients, which showed stable Cross-validation results.

# A4. Features Selection / Engineering

For forecasting-part the features are:

- item release date

- price features (item, store mean, max, mean, std; price divided by month or year mean price; price lag 1 week)

- date + calendar features (snap + events)

- timeseries features (rolling mean, lags)

- target encodings of categorical features

# A5. Training Method(s)

I used LGBM training for forecasting and grid search for distribution.

# A6. Interesting findings

Finding 1 – the forecasting of total level-series is really important for metric WSPL.

Finding 2 – the day of week and day of month calendar of 2011 year is the same with 2016 year. This allowed made heuristic corrections.

Finding 3 – rounding of quantiles made decrease for WSPL metric, especially for low-level series.

# A7. Simple Features and Methods

This paragraph is not applicable for my solution.

# A8. Model Execution Time

Training forecasting model with feature generation takes about 6 hours (Windows 10, Intel i7 3.4 Ghz, 16 Gb RAM)

Prediction for forecasting models takes about 2 hours

Building the distribution takes about 30 minutes (with exe file)

# A9. References

The forecasting part of the solution based on public notebooks for M5 forecasting accuracy by kyakovlev.

# B. SUBMISSION MODEL

Attached at model.zip