M5 Forecasting with LightGBM

Link of Presentation Video: https://www.bilibili.com/video/bv1ka4y1e7Ev

Abstract

The objective of the M5 forecasting competition is to advance the theory and practice of forecasting by identifying the method(s) that provide the **most accurate point forecasts** for each of the **42,840** time series of the competition. To that end, the participants of M5 are asked to provide **28 days ahead point forecasts (PFs)** for all the series of the competition. This article will discuss the data preprocessing, and go through some machine learning methods that we employ when trying to forecast. Generally speaking, we extract the feature, randomized the training set,use *boost* (based on *lightgbm* developed by Microsoft), and finally applied the model to the test set to get the corresponding sales and submit the final result. As of writing this report, we currently rank **9 percent** out of all teams.

The Dataset

The M5 dataset, generously made available by **Walmart**, involves the unit sales of various products sold in the USA, organized in the form of **grouped time series**. More specifically, the dataset involves the unit sales of **3,075 products**, classified in **3 product categories** (Hobbies, Foods, and Household) and **7 product departments**, in which the above-mentioned categories are disaggregated. The products are sold across **10 stores**, located in **3 States** (CA, TX, and WI). In this respect, the bottom-level of the hierarchy, i.e., product-store unit sales, can be mapped either across product categories or geographical regions, as follows:

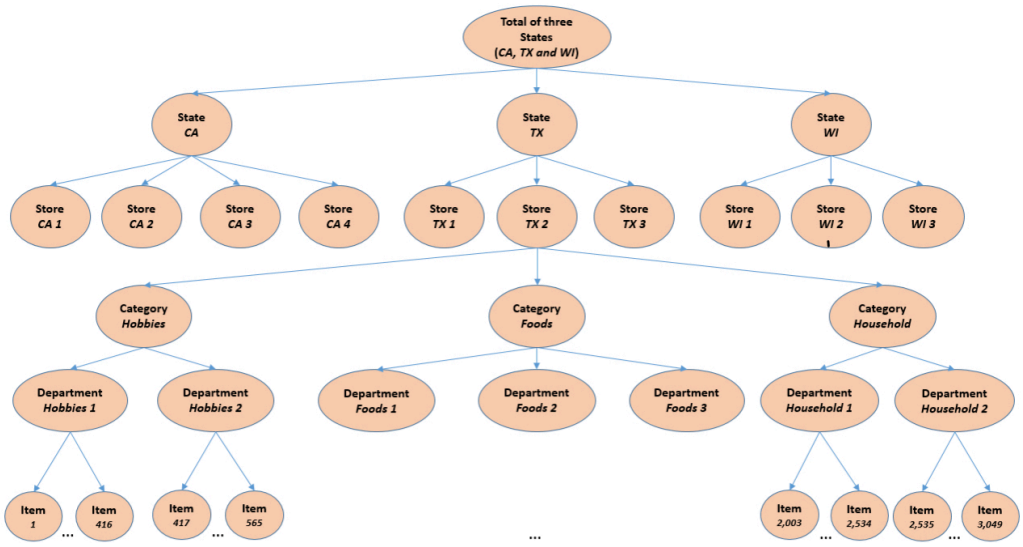

Figure 1: An overview of how the M5 series are organized

Evaluation

The accuracy of the point forecasts will be evaluated using the **Root Mean Squared Scaled Error (RMSSE)**, which is a variant of the well-known Mean Absolute Scaled Error (MASE) proposed by Hyndman and Koehler (2006). The measure is calculated for each series as follows:

$$\textbf{RMSSE} = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}(Y_t - \hat{Y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(Y_t - Y_{t-1})^2}},$$

where Y_t is the actual future value of the examined time series at point t, $\hat{Y}_t$ the generated forecast, n the length of the training sample (number of historical observations), and h the forecasting horizon.

Exploratory Data Analysis (EDA)

In the project, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

Time Series Views

First of all, we may consider the problem to be a time series issue, with 42,840 time series from 2011-01-29 to 2016-06-19 to deal with. A given time series is thought to consist of three systematic components including level, trend, seasonality, and one non-systematic component called noise. For the purpose, we plot the daily overall sales time series.

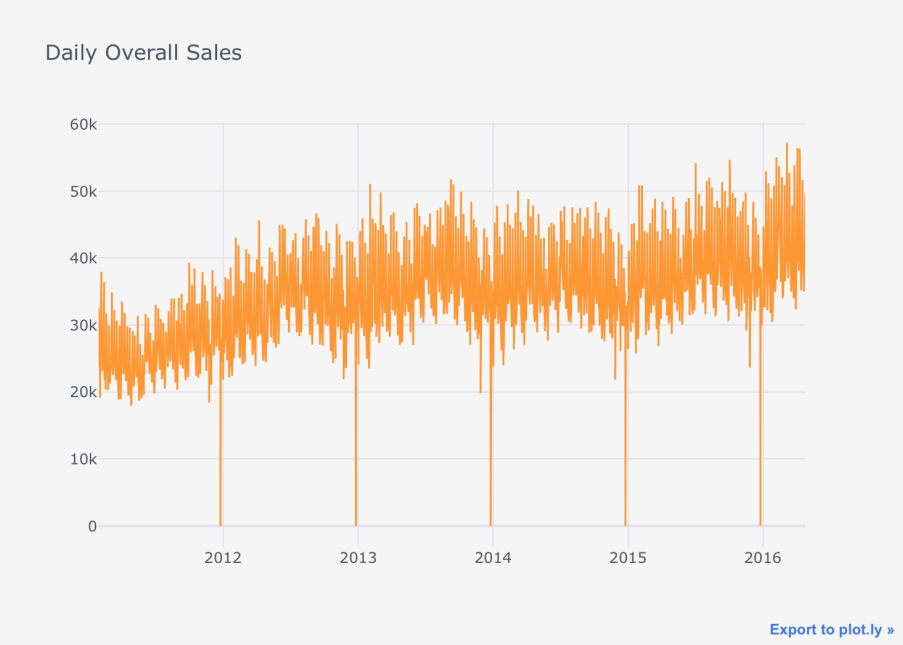

Figure 2: Daily overall sales

From the above figure, we can see a overall upward trend. What's more, we also see a monthly seasonality peaking at August.

Next, we plot monthly sales time series across the 3 states.

Still we can see that CA sales has always been the highest. The peaks in August are the most evident at CA in comparison to other states. Seasonality impacts CA sales the most. Another interesting point we made out is that the sales of WI were lower than TX before 2013. TX and WI sales were similar in 2013 to 2015 August. WI has shown higher sales after 2015 August than TX. We were very eager to find out what has caused WI to increase significantly over the years.

Then we turn to categories, plotting sales time series across categories.

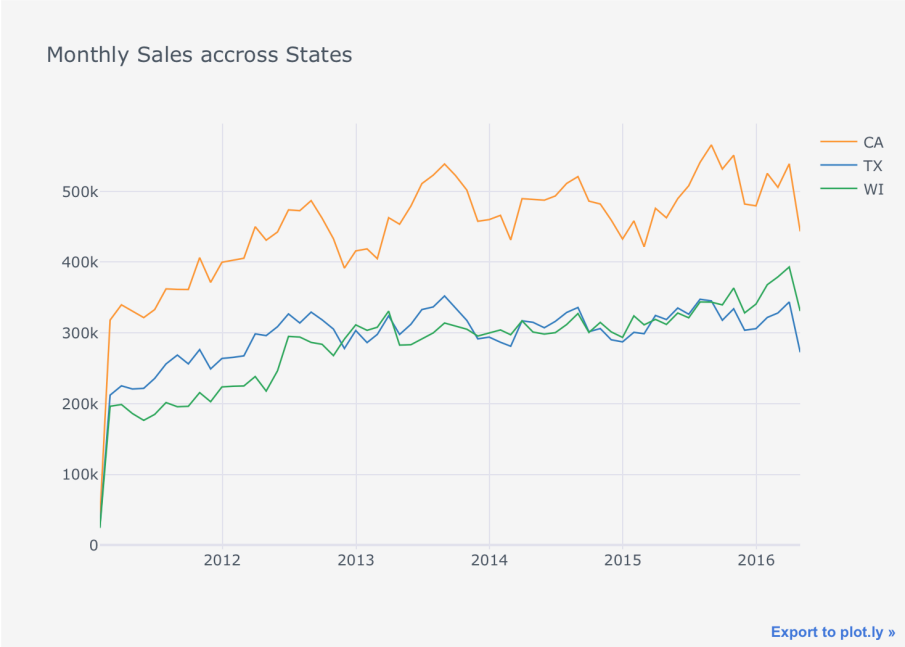

Figure 3: Monthly sales across states

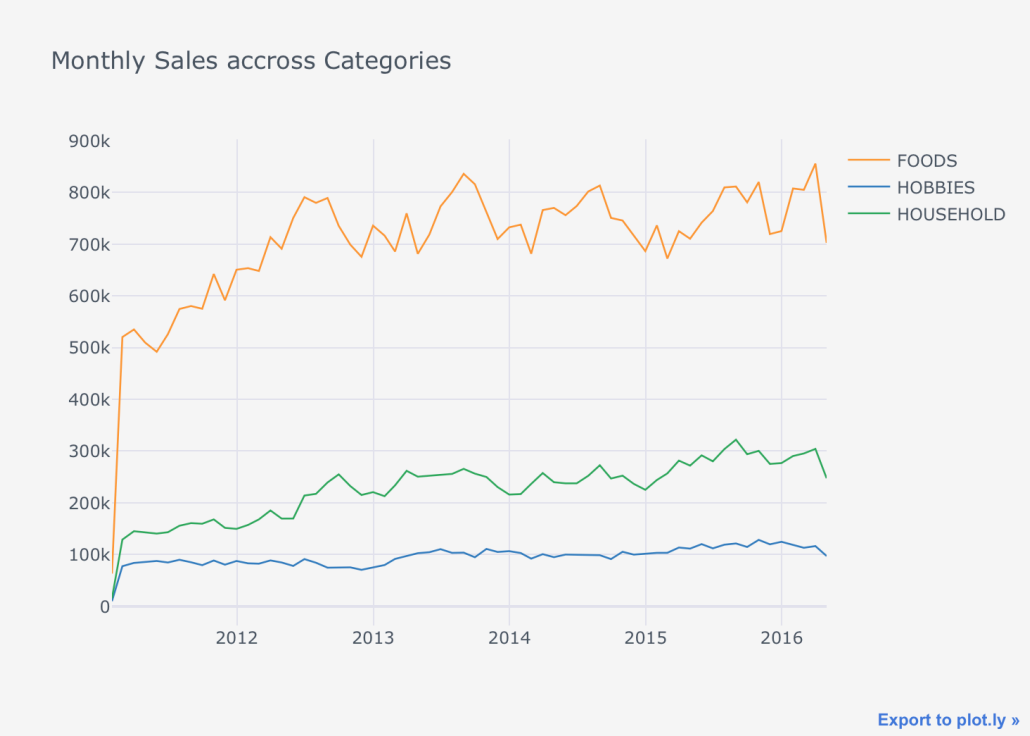

Figure 4: Monthly sales across states

We can see that food sales have always remained much higher than household and hobbies. Hobbies show a pretty much flat trend and less seasonality, while food sales show the highest seasonality and household sales tend to show the highest increase in sales overall.

To be more specific about the impact of date on different categories, the following figure tells us something:

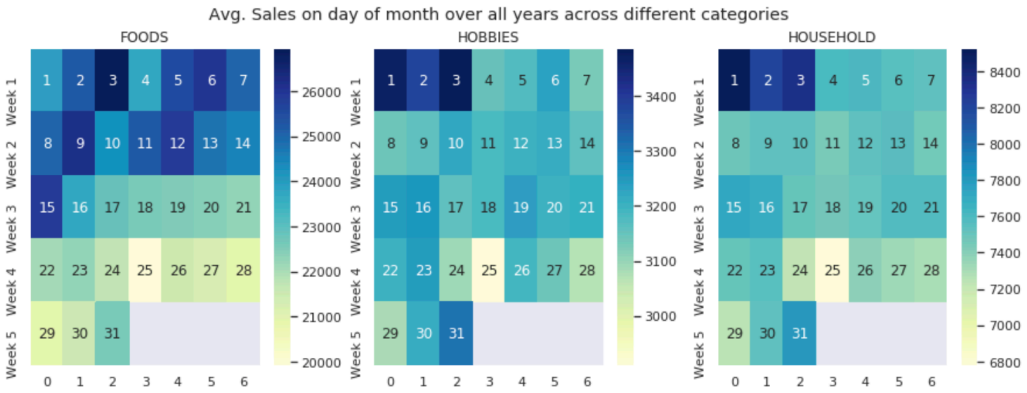

Figure 5: Average sales on day of month over all years across different categories

It's quite obvious that food items differ from the other two in that food items are bought in both the first and second week of a month while hobby and household items are only primarily bought in the first 3 days of the months.

Impact of Events and SNAP Days on Sales

Take 2012 as an example. We carefully analyse the impact of the events, which will be very important for Feature Engineering in the next chapter.

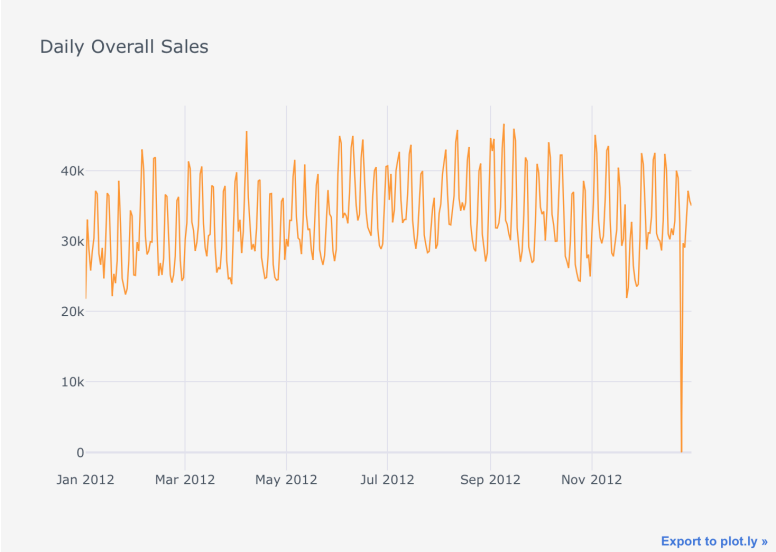

Figure 6: Daily overall sales in 2012

If we closely look at the sales for 2012 we see that people prefer buying on the weekends(Saturdays preferably) before the events rather on the days of the events (it's pretty easy to understand, as we always buy ahead of the time, and weekend is a good choice for most of people). So we don't see a increase in the

sales on the days of the events. But the increase in sales on the weekends before that can be attributed to that event. We see some exception though like Labour Day which is a Monday, we still see a peak on that day. Another one is Thanksgiving which is on a Thursday but we see a peak on the day before which is a Wednesday.

To make the point clearer and get the sense how much the impact is, we plot the following figure:

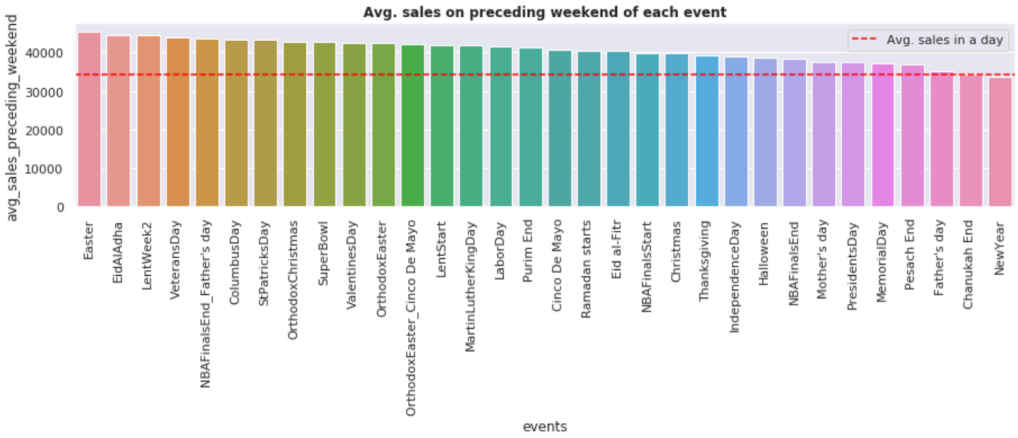

Figure 7: Sales of weekends preceding each event

Almost all weekends have higher sales than the average sales so events do impact sales. The highest sales is in the weekend before Easter as expected about 44k each day followed by EidAlAdha which are both religious events.

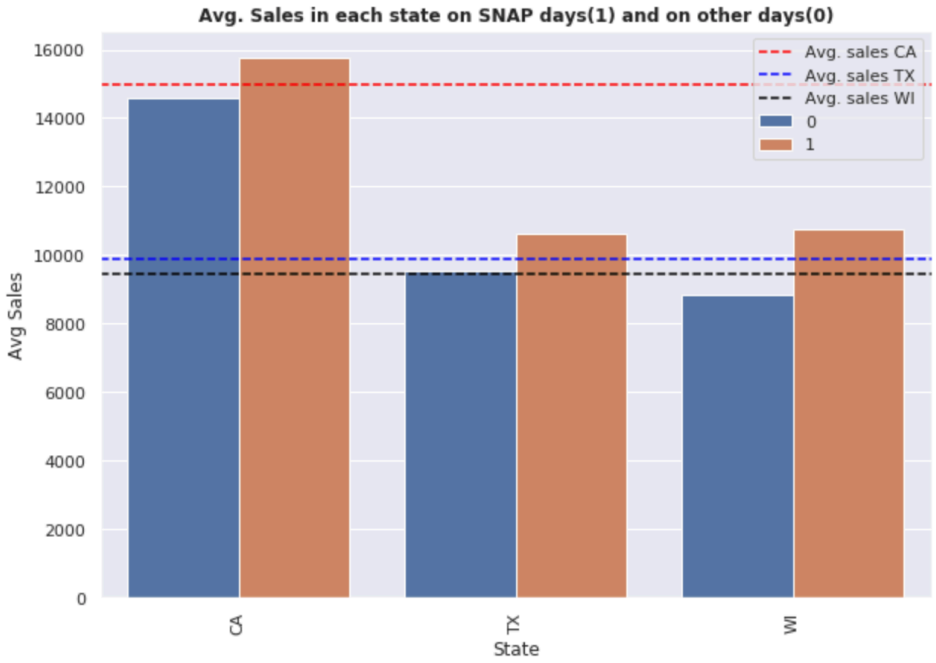

Figure 8: Average sales in each state in SNAP days or not

Another impact on sales are the SNAP days. SNAP provides low income families and individuals with an Electronic Benefits Transfer debit card to purchase food products. In many states, the monetary benefits are dispersed to people across 10 days of the month and on each of these days 1/10 of the people will receive the benefit on their card. Regarding to the information, we guess that SNAP days mainly infect food sales.

All the states have higher sales on SNAP days. We can see the max increase in WI (2k more sales on each day) while CA and TX have 1k more sales on each day of SNAP We have seen higher sales during the first 10-15 days of a month. We have also seen that SNAP days fall in the first 15 days. So, there might be a combined effect.

Analysis on Prices Changes

Let's look at the changes in prices over the weeks for each Category and Price Bucket level.

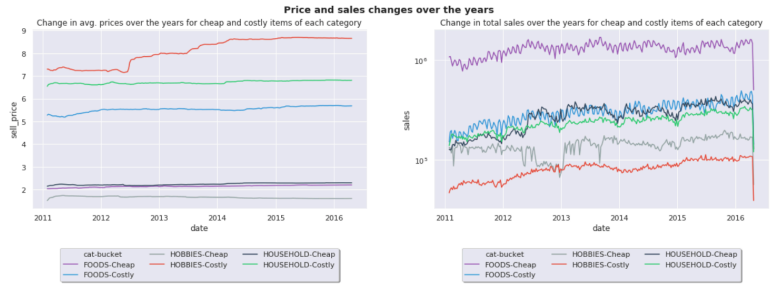

Figure 9: Price and sales changes over years

The prices for products in the cheaper products have increased very less over the years for all the 3 Categories. The increase in prices in Costly Hobbies product didn't impact the sales in 2012 and 2013, it actually grew quite well in those years.What we actually see is a dip in sales of Cheaper Hobbies products between 2012 and 2013.

Feature Engineering (FE)

Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself.

From the above analysis, we may have the sketchy idea about the rather important features. Besides the features in the original data, we can also produce some features ourselves.

When we view the data as many many time series, each time series has a weekly as well as monthly seasonality, and thus we know that the sale today is very likely related to the sale a week or a month before. So the sale of lag-7 or lag-28 are very important features. What's more, very similar to stock price, the average sale during the last week or the last month can reflect the short-term and long-term recent sale condition very well, so we put them into the feature pool too.

Special Features

- lag_7: sales shifted 7 steps downwards for each group. The example above focuses on one group only as an example. That is why the first value appears on the 7th index.
- lag_28: sales shifted 28 steps downwads. That is why the first value appears on the 28th index.
- rmean_7_7: rolling mean sales of a window size of 7 over column lag_7. First value (0.2857) appears on the 13th index because means including nan are nan.
- rmean_7_28: rolling mean sales of a window size of 7 over column lag_28. First value (0.357) appears on the 34th index because that is the first time the mean formula gets all 7 non-nan values.
- rmean_28_7: rolling mean sales of a window size of 28 over column lag_7. First value (0.2857) appears on the 3th index because it is the first time the mean formula gets 28 non-nan values.

- rmean_28_28: rolling mean sales of a window size of 28 over column lag_28. First value appears on 55th index because that is the first time the formula here all non-nan values.

Intuition behind the Features

- lag_7: Captures the week-on-week similarity and that too of just the past week. In other words, people are likely to shop this Monday similar to the last Monday (except it is some special occasion).
- lag_28: Captures the weekly similarity from a month-to-month perspective. Example: people in the 1st weekend of a month shop more so that weekend looks more similar to first weeks of other months than the previous weekend. (Though 28 is arguable here. A month is generally 30. Interesting would be a variable window depending on when the comparative week starts. Dealing with edge cases like week divided into 2 months will be tricky).

Since individual data points are prone to erratic spikes or troughs, mean provides a more ”representative” picture.

- rmean_7_7: Captures the information regarding the sales of the whole previous week ending 7 days in the past i.e. if we are at day 14, then the average is of sales from days 1-7 NOT days 7-14. This provides the information about the whole week and not just a single day sale comparison like lag_7 to bring the lag_7 value into ”better weekly context”.
- rmean_7_28: Captures the information regarding the sales of the entire previous 4 weeks ending 7 days in the past i.e. if we are at day 35, then the average is sales from days 1-28.
- rmean_28_7: Captures the information regarding the sales of the whole week ending 4 weeks ago i.e. if we are on day 35, then the average is of sales from day 1-7. (Assuming for simplicity the month is 28 days), this provides the information of not just a month-to-month comparison of the same day (day 7 of month one vs day 7 of month two), but the entire week leading up to day 7. Again the idea I believe is to capture the whole week and not just a single day sale comparison like lag_28 to bring the lag_28 value into ”better weekly context”.
- rmean_28_28: Captures the information regarding the sales of the entire previous 4 weeks ending 4 weeks in the past i.e. if we are at day 56, then the average is of days 1-28. (Assuming for simplicity the month is 28 days), the idea again is to bring the point value of lag_28 into a better context (i.e. of day 28 when being compared to day 56) into a ”better monthly context”.

LightGBM

In fact we have noticed that many of the kernels written for Kaggle have used LightGBM and have produced really good results. These kernels have produced really useful insights into how to use LightGBM.

LightGBM is a boosting method, which combines a set of weak learners to form a strong rule. It is an iterative process. The weak rules are generated by a base learning algorithm. These rules are generated iteratively and combined at the end to form a single strong rule. Boosting gives more importance to observations which are wrongly classified.

We omit the really fabulous details of LightGBM, which would be much too complicated and prolix for our report. Here we just show the parameters in our training.

```
params = {"objective": "poisson", "metric" :"rmse", "force_row_wise" : True,
"learning_rate": 0.075, "sub_row": 0.75, "bagging_freq": 1, "lambda_l2": 0.1, "metric":
["rmse"], 'verbosity': 1, 'num_iterations': 1200, 'num_leaves': 128, "min_data_in_leaf":
100}
```

What' s more, we use some method like: for feature engineering, the lag feature, rolling feature, and the mean and variance of some statistical features are used, and for saving memory we converting strings to categories, and speed up loading by saving to pickles.

Forecast

Here, we implement a recursive way of forecasting, that is, we have only one model to forecast but we use it day by day and so on we obtain all of the 28 days.

There's a little trap here. When a recursive way is applied on the forecasting, the result from t_n will be used to predict the next result t_{n+1}, so it's very easy to understand if you change t_1 by a tiny fraction the data on t_{28} might be changed a lot.

As for the trap, we use a magic here. We change the weight of the predicting for a little bit and then average them, so we can get different predictions, and thus to some extent, we can eliminate the chance we fall into the trap.

Performance

By following the method we mentioned before, we backtest it in the kaggle and ranked 9 % in the end, and here is the score performance:

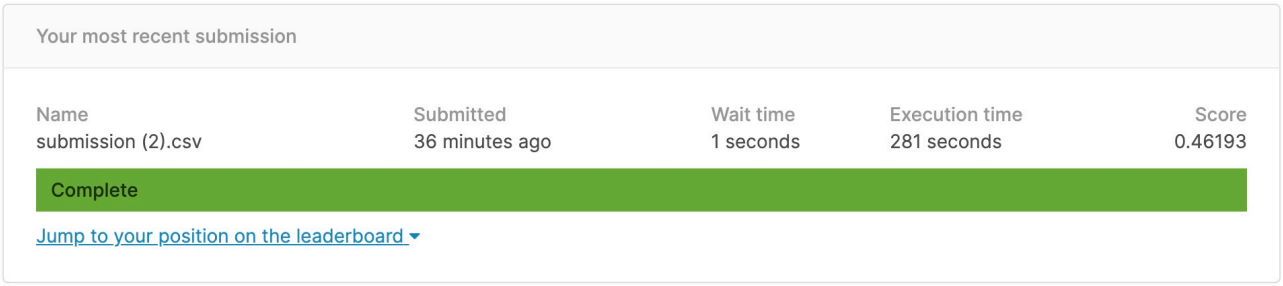

Figure 10: Performace of LightGBM

Improvements

- When we did EDA, we discovered the point that the impact of SNAP days on sales is not very simple, but linked to the previous weekend, but during FE we failed to produce such a feature to embody the feature.
- The loss function in LightGBM is **RMSE**, which is essentially the same as our target **RMSSE**. As suggested by some Kagglers, however, if we appropriately choose some other metrics, we might have better performance.

Work Distribution

- ZHANG Zhiyi: Code Part1 + Write Report
- HU Zhenzhuo: Code Part2 + Write Report
- WU Siliang: Presentation Representative and make the slides.