M5 Forecasting Competition Report

Shihao Yang
Financial Mathmetics
HKUST
20660753
syangbw@connect.ust.hk

Fangrui Zhang
Financial Mathmetics
HKUST
20641161
fzhangap@connect.ust.hk

Haolan Fan
Financial Mathmetics
HKUST
20657536
hfanai@connect.ust.hk

Yijia Song
Financial Mathmetics
HKUST
20665947
ysongba@connect.ust.hk

Tingting Jiang
Financial Mathmetics
HKUST
20643913
tjiangam@connect.ust.hk

Yinuo Wang
Financial Mathmetics
HKUST
20661719
ywangkz@connect.ust.hk

Wenjun Li
Financial Mathmetics
HKUST
20654780
wlice@connect.ust.hk

Abstract

In this paper, we extracted many meaningful features in data analysis, feature engineering and feature importance analysis processes. We built the models at the store level, the model is lightGBM. It is also used many times in the feature selection process to obtain the most effective features.
Code link: $https://www.dropbox.com/sh/1xjxj5izzkftc8z/AACLN2QGkdl8spZnDb-i0_OUa?dl=0$
Video link: $https://drive.google.com/file/d/1aIBpNUghY6UIKbm37P_etnOPYlwwU9p8/view?usp=sharing$

1 Team division

Table 1: Team division

Name	Contribution
Yinuo Wang	Part1 Data Analysis
Tingting Jiang	Part2 Basic Feature Engineering
Wenjun Li	Part3 Calendar Features
Haolan Fan	Part4 Lag Features
Fangrui Zhang	Part5 Feature Importance Analysis
Yijia Song	Part6 PCA analysis
Shihao Yang	Part7 Integrated Model training using multiple feature

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Part1 Data Analysis

2.1 Visualizing the Time Series Data

Figure 1 shows the first time series in the data. Going through the different time series data we can see that a lot of the items have intermittent demand. Some series have many zeros with bursts of demand in between, and that's one of the challenges we're going to deal with.

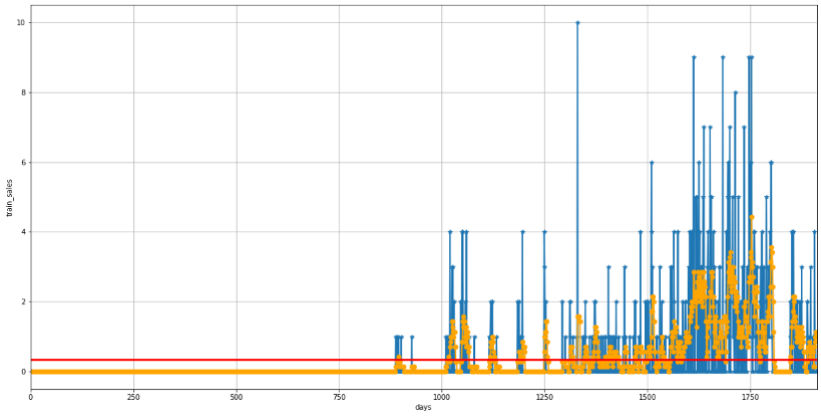

Figure 1: Time series

From the analysis above, there are lots of the time series data start with leading zeros. We can describe these leading zeros as items that were not selling or available to sell during those time periods. Then we investigate the distribution of leading zeros (figure 2), which can help us reduce the large data size. After that, we plot the distribution of zeros per series.

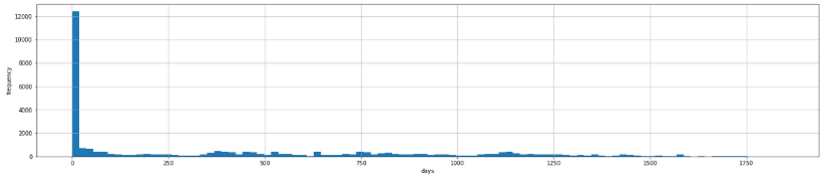

Figure 2: Distribution of zeros per series

Figure 3 indicates that the distribution of zeros for each of the series has a mean around 0.8 which means there is a lot of intermittent data. Further, we find the distribution of max number of sales for each of the series. Figure 4 shows that most of the items have a max number of sales between 2 and

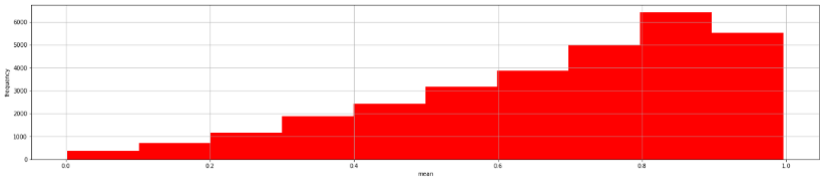

Figure 3: The distribution of max number of sales

12. For some items, the number of sales for a particular item can be large, and it might be fruitful to investigate these items.

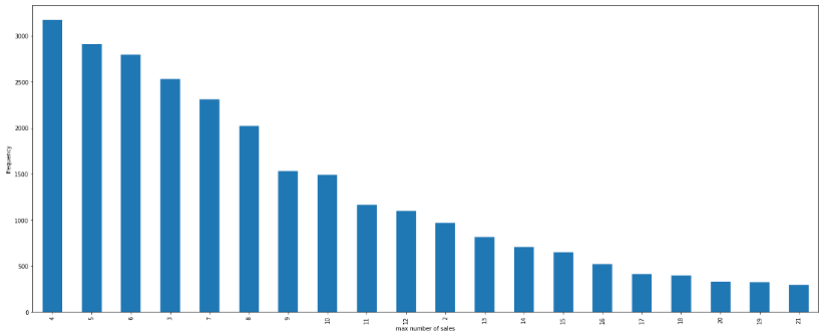

Figure 4: Items with a max number of sales

3 Part2 Feature Engineering

The data we use contains 3 files, calendar.csv, sales_tarin_validation.csv, sell_prices.csv respectively. For the first file, it involves 1914 days for training. Since the dataset is really large, to make it more efficient, we first reduce the memory use by simplifying the data type. Then we transform horizontal representation to vertical "view", after the transformation, the length change from 30490 to 58327370.

- **Date feature** We construct a new feature product release date then merge into our train_df. Since prices are set by week, so it will not have very accurate release week, however we can still try to see whether it will be helpful in our model.
- **Price Feature** We generate some basic price feature like "price_max", "price_min", "price_std", "price_mean", which are calculated by grouping the store."Price_norm" is a normalization of sell prices and "price_nunique","item_nunique" are also calculated by grouping the store.Then we add some price "momentum" feature. The features are price momentum, price momentum_m, price momentum_y which are calculated by dividing the current sell price by the previous average sell price in week, month and year. This also conduct by grouping different stores.

4 Part3 Calendar Features

At this part, we need to process the calendar dataset, and the data processing details are shown as following:

- From the dataset, select the variables named date, d, event_name_1, event_type_1, event_name_2, event_type_2, snap_CA, snap_TX, and snap_W1. Next, merge these variables into the previous grid_df according to the variable d.
- Transform the types of variables event_name_1, event_type_1, event_name_2, event_type_2, snap_CA, snap_TX, and snap_W1 into category type.

Transform the type of variable date into datetime type, and make some features from variable date. There are 7 features we can extract from variable date:

- variable tm_d, extract day from the date.
- variable tm_w, extract week from the date.
- variable tm_m, extract month from the date.
- variable tm_y, extract year and find the minimum of the year from the date, and use the year minus minimum year to get final tm_y.
- variable tm_wm, extract week of the month from variable tm_d.
- variable tm_dw, extract day of the week from the date.

- variable tm_w_end, label the days in a week which are Friday, Saturday and Sunday from variable tm_dw.

Therefore, after data cleaning for calendar dataset, we can obtain two types of data, category and int8, including event_name_1, event_type_1, event_name_2, event_type_2, snap_CA, snap_TX, snap_W1, and tm_d, tm_w, tm_m, tm_y, tm_wm, tm_dw, tm_w_end, respectively.

5 Part4 Lag Features

Since many variables provided in the datasets, it is important to conduct feature selection. We only concern about the id, sales and selling price of each commodity recorded. So we choose 8 variables, including id, item_id, dept_id, cat_id, store_id, state_id, d and sales, to be our final choice. However, if we feed this data to lag model directly using all d_1 to d_1912 columns, we only have 12,196 rows to train, which is not sufficient. In other hand, we can get 23,330,948 rows for our model to train if we use selling information of commodities as rows in time order. Then we can generate different lag columns with different rolling windows to be our final features.

The details of feature selection process is as below:

- Load the pickle data saved before and get the data with key variables mentioned before.
- Generate new columns with lags from 28 to 42 in data frame got in step1.
- Generate new rolling window columns of mean and standard deviation in 7,14,30,60,180 periods corresponded to each lag columns in step 2.
- Use the new datasets columns as final features to train the model.

Another thing we should mention is that we only train dataset limited to CA state in this part, in order to save memory usage when running the codes.

6 Part5 Feature Importance Analysis

Actually, there are many methods to measure feature importance by studying the changes of the score like accuracy, MSE or RMSE when a feature is not available. Traditionally, we will remove feature from the dataset, re-train the estimator and check the score. However, the re-training process makes it computationally intensive, especially when dataset is large. To avoid such problem, we replace one feature with random noise by shuffling values for the feature instead of removing the feature from the dataset. This method is called permutation importance and is used in this project. Basically, the process is as follows:

- Derive a trained model.
- Choose one feature, shuffle the values of this feature and make predictions using the validation dataset. Compute the RMSE with regard to the shuffled feature, which is used to measure the importance of the feature.
- Repeat step 2 with another feature in the dataset, until the importance of all features has been measured.

The result is as follows:

The values of each features denote the RMSE calculated based on dataset with the shuffled feature minus the RMSE calculated based on original dataset. So the higher the value is, the more important the feature is. From the above figure, we can find that sales with 1 day lag are most important, next comes tm_dw, and price_nunique with strong negative values(-0.0018) is least important and probably just noise.

7 Part6 PCA analysis

Principal Component Analysis (Principal Component Analysis) is a technique for exploring high-dimensional data. PCA is usually used for the exploration and visualization of high-dimensional data

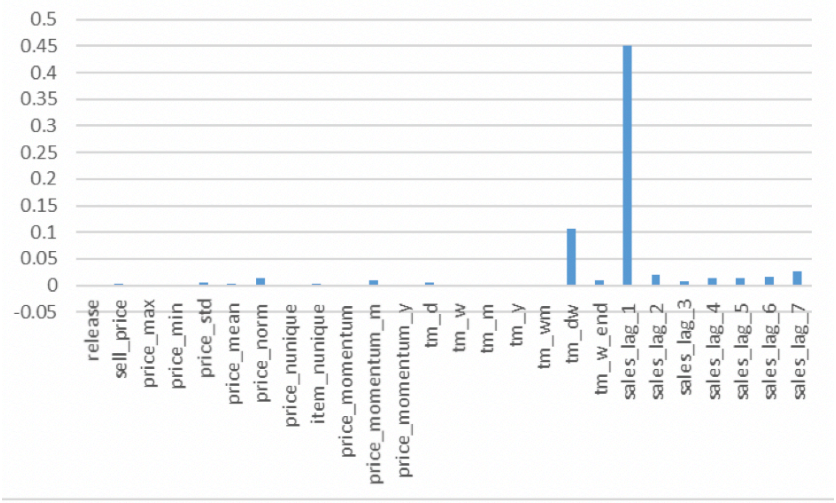

Figure 5: Feature Importance

sets. It can also be used for data compression, data preprocessing, etc. PCA can synthesize high-dimensional variables that may have linear correlation into linearly independent low-dimensional variables, called principal components. The new low-dimensional data set will retain the variables of the original data as much as possible. The set is mapped to a low-dimensional space while retaining as many variables as possible. More variables mean more complete information. It should be noted that dimensionality reduction means the loss of information. If the original data has no effect on the model, it is unrealistic to expect dimensionality reduction to improve it. However, considering the correlation that often exists in the actual data itself, we can think of ways to reduce the loss of information as much as possible while reducing dimensionality. To put it simply, PCA can only be used to reduce dimensionality in order to simplify the model and save time only if a good effect can be obtained on the original data.

Table 2: pca outcome

	1	2	3	4	5	6	7
variance contribution rate	72.24	6.62	5.93	4.20	3.89	3.61	3.50
Sum contribution	72.24	78.87	84.80				

The optimization goal of the dimensionality reduction problem: reduce a set of N-dimensional vectors to R dimensions, that is, select R unit orthogonal bases, so that after the original data is transformed onto this set of bases, the covariance between each dimension is 0 The variance of each dimension is as large as possible (under the constraint of orthogonality, take the largest R variance). When the covariance is 0, it means that the two variables have no correlation. The large variance means that the data points are scattered enough to retain as much original information as possible. The main steps are: firstly, centralize all samples; secondly, calculate the covariance matrix, eigenvalues and eigenvectors; finally, sort the eigenvalues, and select the eigenvector corresponding to the largest R eigenvalues. Through the calculation of the eigenvalues, we can get the percentage of the principal component, which is used to measure the quality of the model. In addition, it can also be evaluated by cross-validation.

Table 3: rmse

	training rmse	valid_1 rmse
7 dimensions	2.66922	2.27719
3 dimensions	2.77878	2.37865

8 Part7 Integrated Model training using multiple feature

This part is mainly based on the results of various feature engineering previously performed model training. In order to save space and time, the model chose to use lightGBM. The specific work is divided into the following steps:

8.1 Data modules combination

So far, we've done a lot of feature engineering works, and then we just need to merge them. Some of these features are not required to participate in training, and these characteristics are

Table 4: Feature table

Feature class	Feature name
remove_features	$['id','state\_id','store\_id','date','wm\_yr\_wk','d',TARGET]$
mean_features	$['cat\_id_mean','cat\_id\_std','dept\_id\_mean','dept\_id\_std','item\_id\_mean','item\_id\_std']$

It is convenient to combine all the modules in pandas to form the final training set and validation set.

8.2 Lag features adjustment

Its a good way to use lag features, but there are some problems. Rollings are calculated this way because we need to update our Target before rolling calculations. Because we want to avoid Nans and feed our model with somehow valuable information. For example, if We need to predict day 1920, and We have rolling mean feature with 7 days window. So To calculate such feature we need Target for days 1913, 1914, 1915, 1916, 1917, 1918, 1919, but With training set in our hands we have only Target for day 1913 (Test set Target is Nan). So we can only choose to do recursive predictions and rolling calculations for 1914, 1915, 1916, 1917, 1918, 1919 days.

Regarding the "rollingmeantmp_" features, we recalculate them only at prediction times. Here is the programming stucture:

```
# Removing features that we need to calculate recursively
griddf = griddf[predsmask].resetindex(drop=True)
keepcols = [col for col in list(griddf) if 'tmp' not in col]
griddf = griddf[keep_cols]

# Make temporary grid to calculate rolling lags
griddf = basetest.copy()
griddf = pd.concat([griddf, dfparallelizerun(makelagroll, ROLS_SPLIT)]
                                    , axis=1)
```

8.3 Store-level lightGBM training

In terms of model training, training a model may overlook the uniqueness of different regions and stores. So for each store we train a different lightGBM model. There is no need to recalculate the lag feature during the training and validation phase, only training and calculating losses using the data from each store. And at the time of forecasting, because of the factors mentioned above, it is necessary to first forecast all store sales at one time and at the next point in time. The code structure is as follows:

```
for PREDICT_DAY in range(1,29):

    # Compute new lag_roll feature

    for store_id in STORES_IDS:

         #do prediction
         #updata the test dataset
```

8.4 Generating submission

Finally we generated the outcome:

```
submission = pd.read_csv(ORIGINAL+'sample_submission.csv')[['id']]
submission = submission.merge(all_preds, on=['id'], how='left').fillna
                                                                (0)
submission.to_csv('submission_v'+str(VER)+'.csv', index=False)
```

9 Summary

This project mainly focuses on feature engineering, the data is more comprehensive, and the model training at the store level, the final score is 0.47506

References

[1] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu., *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.*, Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.