Midterm

M5 Forecasting

Shen Yiqin, Liu Jianyi, Sha Haochen

Abstract

The aim of this question is to predict the sales of Walmart supermarkets in the next 28 days through the existing data, which is a regression forecast problem. First, we made EDA on the dataset, which help us to grasped the changing of the data through the statistics, integration and visualization of the data set. Then we carried out Feature Engineering, and created a certain number of new features through Lag, Mean Encoding, Sliding windows, etc to improve the prediction accuracy in the modeling stage. In the modeling stage, we selected three models of ARIMA, LSTM, and LGB. Due to the huge amount of data, the result of LSTM is not ideal. The ARIMA model reached the baseline, and the LGB model achieved a good result, with an RMSSE of 0.474 on the public dataset, so we finally chose the LGB model to participate in the competition. At the same time, we believe that LSTM and LGB models still have some space for improvement.

Workload: Sha Haochen did the EDA, Feature Engineering and the first part Preprocess of the report. Liu Jianyi did the Time Series Forecasting and built the Arima and Sarima model,wrote the LSTM and Arima parts of this report. Shen Yiqin did the LGBM and LSTM coding and model turing, wrote LGBM and competition plan and model analysis parts in report.

1 Data Preprocess

1.1 Exploratory Data Analysis

There are three main files in this dataset: ‘sales’ contains the daily sales quantity of products, ‘prices’ contains the price changes of different types of products, and ‘calender’ contains information on the sales date of the products. EDA can help us to grasp the data from an overall perspective and discover potential connections or trends in the data, laying the foundation for feature engineering.

The ‘sales’ file is the most important as we need to predict the items sold amount at the end. Through the statistics of category,store,state and product category, we found that the data contains sales information of 10 stores in 3 states. There are a total of 3,049 products in 3 categories. Then we randomly selected 10 rows of sales information for visualization, and the results are as follows in Figure3.

From the results, we can see that there is a clear gap between the highest value and the lowest value of sales of most items, the fluctuation range is large, but the average sales volume seems stable. Many products have long-term sales of 0 from the first day, which may be caused by the fact that the products are not on the shelves. These 0 value data also cause a lot of memory waste and may affect the results of data processing.

Then we want to observe these sales data from other different dimensions. To do this, we need to integrate the three tables. After showing the information of the three files, we found that the memory cost of the files is very large, and it is necessary to perform memory-saving operations. The principle is to save memory by changing the column data types automatically set by pandas, such as Int64,Float32 Object, to Int16,Category that occupy less memory without losing data. After actual use, about three-quarters of the memory resources can be saved. In addition, in order to merge the three tables, we modified the structure of the sales table, changing the horizontal expression to the vertical expression, that is, converting the daily sales of products from 1914 columns to rows. Then we can merge the three tables together.

So the three tables become one, after that we select a product from each category and get the sales of different time dimensions:

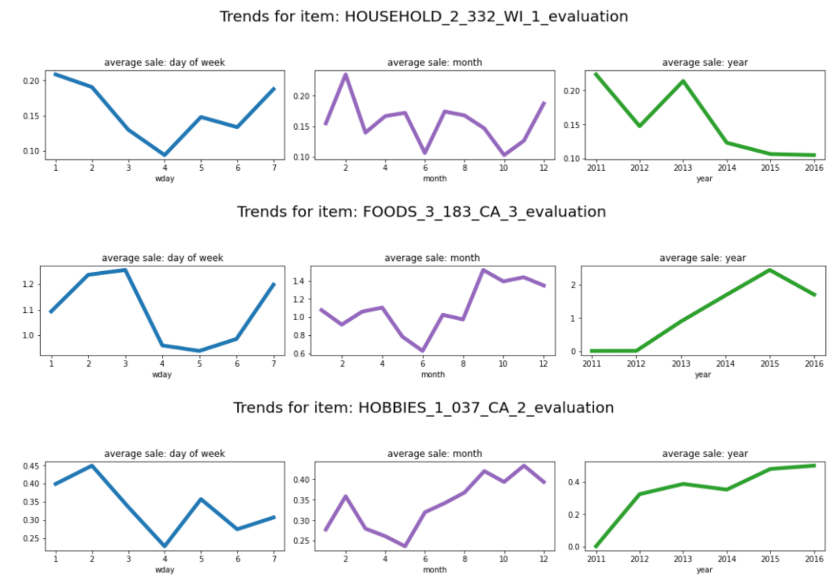


Figure 1: Sales trend in different dimension

We can find that the sales of products in the dimensions of weeks, months and years all has a certain trend and pattern, and there is no obvious irregular fluctuation. It is to say that the previous sales of products have an influence on the current sales. We can use the discipline during the feature engineering.

We then plotted the overall trend of merchandise sales based on three categories:

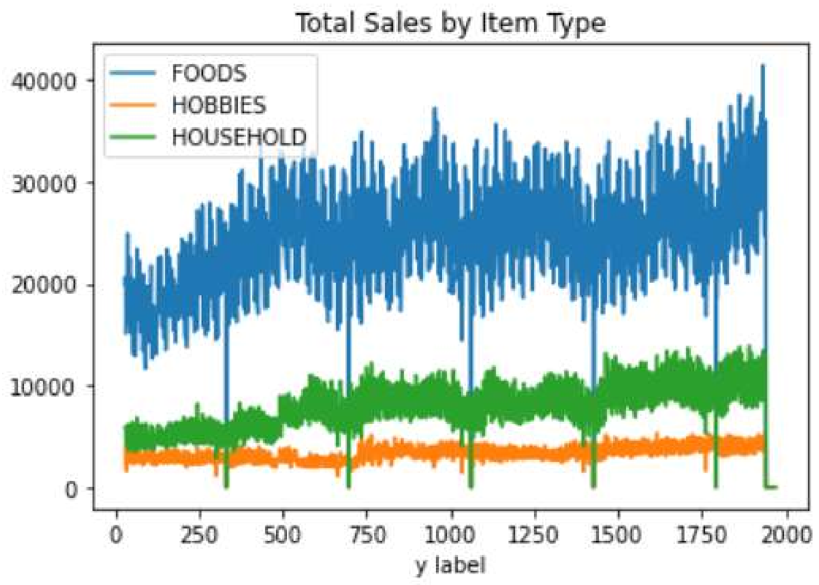


Figure 2: Sales for different categories

It can be found that the sales of Food products are the highest, and the upward trend is obvious. Hobbies and Household products are also on the rise year by year, but the growth rate is slow. For the huge one-day fluctuations in the data, we found that this day appeared on Christmas, when Walmart was closed.

1.2 Feature Engineering

In this topic, the purpose of feature engineering is mainly to process the existing features to support the model better predict the sales data. The first feature added is Lag. For example, if one want to predict the sales of goods at time t, he can also use the sales of goods at time t-1 as a feature. This is appropriate in this problem, since sales have a momentum relationship, which we proved with the previous EDA. But we couldn't make sure what time dimension lag was most appropriate to choose, so we added Lags of 1, 2, 3, 7, 14, 28 days.

Second, we added sliding time windows with the mean, of time periods 7, 30 and 60 days. The purpose is to smooth the sales data within a certain period of time, to better reflect the trend of the sales data, and to avoid the impact of short-term sales fluctuations. Similarly, we also added an extended window, which works like a sliding window. The difference is that the starting position of the expansion window does not change, but rather by adding subsequent sales data and then averaging. The selected expansion window size is 7 days, that is, 7 days of data are added each time and get averaged.

Then we added Mean Encoding. In category type features, if the category of the feature is too much, such as address, etc. Using Label Encoder or One-Hot Encoder often does not work well. At this time we can use the Mean Encoder. It will weight each category as a variable according to its proportion in the overall data, solving the problem of having a wide variety in a feature. The 'item_id' in the sales data conforms to this situation, and it has a total of 3049 categories. By combining 'item_id' with different kinds of features such as 'cat_id', 'store_id' and then doing Mean Encoding, we get four new features.

Since our merged table also includes price information, we also process the price information. The method is to treat price as price momentum, which is the same in essence as the previous sliding window, which can smooth data, ignore short-term fluctuations, and reflect longer-term trends. In the dimension of time, we chose week, month and year.

After completing all the feature engineering above, the feature and format information of the data is as follow:

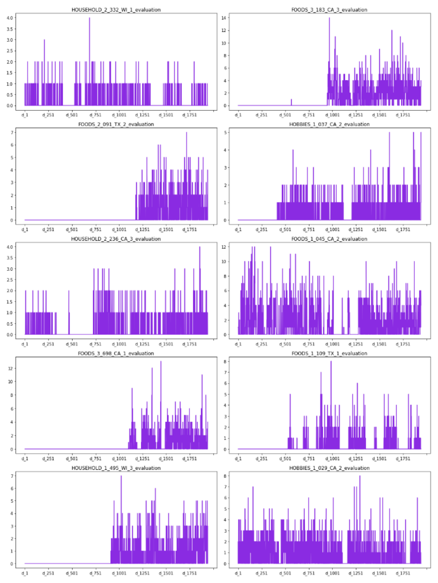
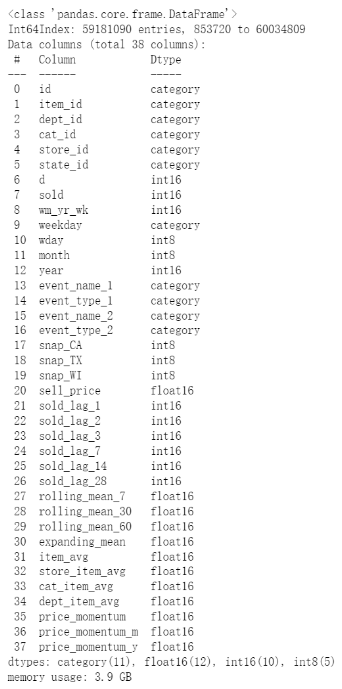


Figure 3: Sample of item



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 59181090 entries, 853720 to 60034809
Data columns (total 38 columns):
 #   Column             Dtype
---  ------             -----
 0   id                 category
 1   item_id            category
 2   dept_id            category
 3   cat_id             category
 4   store_id           category
 5   state_id           category
 6   d                  int16
 7   sold               int16
 8   wm_yr_wk           int16
 9   weekday            category
 10  wday               int8
 11  month              int8
 12  year               int16
 13  event_name_1       category
 14  event_type_1       category
 15  event_name_2       category
 16  event_type_2       category
 17  snap_CA            int8
 18  snap_TX            int8
 19  snap_WI            int8
 20  sell_price         float16
 21  sold_lag_1         int16
 22  sold_lag_2         int16
 23  sold_lag_3         int16
 24  sold_lag_7         int16
 25  sold_lag_14        int16
 26  sold_lag_28        int16
 27  rolling_mean_7     float16
 28  rolling_mean_30    float16
 29  rolling_mean_60    float16
 30  expanding_mean     float16
 31  item_avg           float16
 32  store_item_avg     float16
 33  cat_item_avg       float16
 34  dept_item_avg      float16
 35  price_momentum     float16
 36  price_momentum_m   float16
 37  price_momentum_y   float16
dtypes: category(11), float16(12), int16(10), int8(5)
memory usage: 3.9 GB
```

Figure 4: Feature after processing

Finally, considering that we have different kinds of models, which may depend on different features, we don't filter or drop features. But we represent the importance of each feature according to the 'feature_importance' function of the LGB model. It will be easy for us to change the features during the training and evaluation process of the model after this procedure. The different feature weights given by LGB function are as follows:

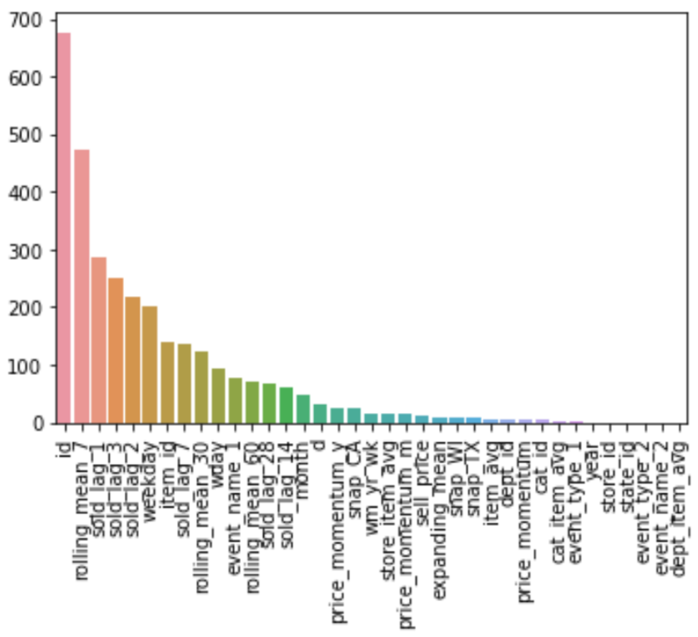


Figure 5: Feature importance

2 Modeling

2.1 LGBM

GBDT (Gradient Boosting Decision Tree) is an enduring model in machine learning. Its main idea is to use weak classifiers (decision trees) to iteratively train, obtaining the optimal model. The model has good training effect and is not easy to overfit. Etc. GBDT is not only widely used in the industry, but is usually used for tasks such as multi-classification, click-through rate prediction, search ranking, etc. It is also a deadly weapon in various data mining competitions. According to statistics, more than half of the championship schemes in Kaggle competitions are based on GBDT. LightGBM (Light Gradient Boosting Machine) is a framework that implements the GBDT algorithm, supports efficient parallel training, and has faster training speed, lower memory consumption, better accuracy, support for distributed and can quickly process massive data, etc.

LightGBM is the decision tree algorithm based on histogram. The basic idea of the histogram algorithm is to first discretize the continuous floating-point eigenvalues into k integers, and at the same time construct a histogram with a width of k. When traversing the data, the statistics are accumulated in the histogram according to the discretized value as an index. After traversing the data once, the histogram accumulates the required statistics, and then according to the discrete value of the histogram, it is traversed to find the optimal one. split point.

Beside histogram algorithm, LightGBM is further optimized. First, it abandons the level-wise decision tree growth strategy used by most GBDT tools and uses a leaf-wise algorithm with depth constraints. This strategy finds the leaf with the largest split gain from all the current leaves each time, and then splits, and so on. Therefore, compared with Level-wise, the advantages of Leaf-wise are: in the case of the same number of splits, Leaf-wise can reduce more errors and obtain better accuracy; the disadvantage of Leaf-wise is that it may grow Deeper decision trees to produce overfitting. Therefore, LightGBM will add a maximum depth limit on Leaf-wise to prevent overfitting while ensuring high efficiency.

2.2 LSTM

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory.

Common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

2.3 Arima

An autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

An ARIMA model can be understood by outlining each of its components as follows:

- Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- Integrated (I): represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
- Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

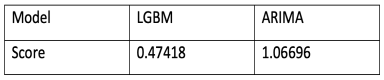
In an autoregressive integrated moving average model, the data are differenced in order to make it stationary. A model that shows stationarity is one that shows there is constancy to the data over time. Most economic and market data show trends, so the purpose of differencing is to remove any trends or seasonal structures.

Seasonality, or when data show regular and predictable patterns that repeat over a calendar year, could negatively affect the regression model. If a trend appears and stationarity is not evident, many of the computations throughout the process cannot be made with great efficacy.

3 Final Plan and Analysis

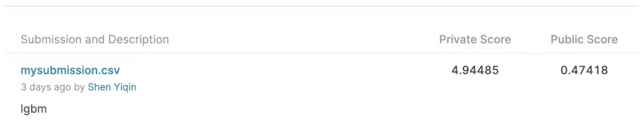
3.1 Plan for Competition

Due to the limit of computational resources, we found that LSTM is hard to train the total data (too much memory need to use). We finally plan to train the whole data with LGBM and ARIMA. This competition uses a Weighted Root Mean Squared Scaled Error (RMSSE). The scoring results give the following. We choose LGBM as our final model since it gives better performance on forecasting feature demand.



Model	LGBM	ARIMA
Score	0.47418	1.06696

Figure 6: Model performance



Submission and Description	Private Score	Public Score
mysubmission.csv 3 days ago by Shen Yiqin lgbm	4.94485	0.47418

Figure 7: Final model

3.2 Analysis

When training with LightGBM, we find it't important to choose proper hyperparameters, for instance, if we select too large depth of each tree, it may cause overfitting problem because the gap between training accuracy and validation accuracy become large. LightGBM can quickly fit the data and gives the prediction of test set, following figure shows part of validation data vs. prediction results by LightGBM with small number of estimators. We can see it's obvious that the model can learn the trend very well, but sometimes the prediction always be same. If we change the larger number of estimators, the problem fixed.

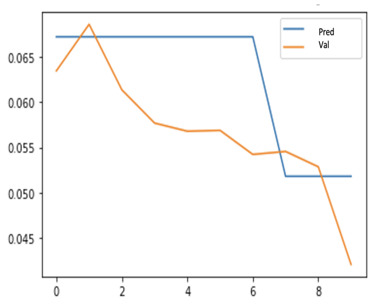


Figure 8: Sample Prediction by LightGBM

For ARIMA, from the following figure, we can clearly see it can fit seasonal term very well, but since it is an auto-regression model, and don't input any other information besides the demand itself, it may less consider the correlation of different features with predicted values, so the overall score of ARIMA is not good.

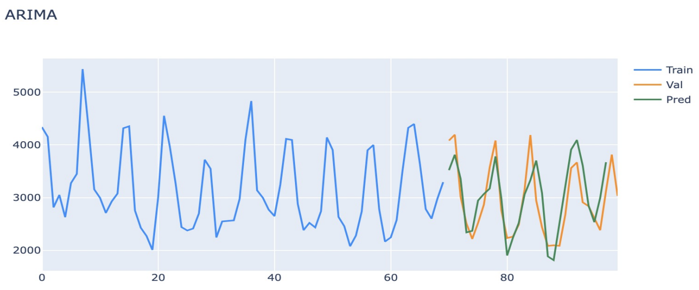


Figure 9: Sample Prediction by ARIMA

References

[1] https://www.kaggle.com/code/kyakovlev/m5-three-shades-of-dark-darker-magic

[2] https://www.kaggle.com/code/anshuls235/time-series-forecasting-eda-fe-modelling

[3] Guolin Ke; Qi Meng; Thomas Finely; Taifeng Wang; Wei Chen; Weidong Ma; Qiwei Ye; Tie-Yan Liu (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree"