Time Series Forecasting - Walmart Retail

Xinzhen Liao
Yuan Xu
Rui MA
Yiyuan Shi
Nov. 14, 2021

CONTENTS

LIST OF FIGURES

Time Series Forecasting - Walmart Retail

***Abstract*—In this report, we were working with 42,840 hierarchical time series data provided from Walmart to forecast the future daily sales for the next 28 days, containing 3,049 individual products from 3 categories and 7 departments respectively, sold in 10 different stores among 3 states. More specifically, the data were obtained in the 3 US states of California (CA), Texas (TX), and Wisconsin (WI), and were aggregated on multiple levels like item, department, product category and state. In addition, information about prices, promotions, and holidays from January 2011 to June 2016 were also provided. Here is approximate the underlying logic of the given data.**

This report will contain 4 parts respectively, which are Project Introduction, Data, Modeling and Result.

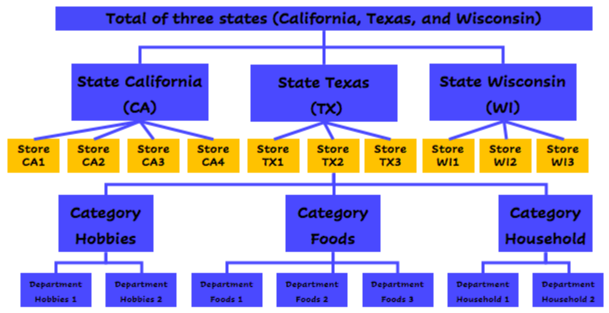


Fig. 1. Underlying Logic of the Given Data

I. Introduction

Even though Walmart recently divested from a number of markets, like Brazil, United Kingdom and Japan, it still remains the No.1 in the top 50 global retailers. Walmart keep continuing to adjust business model to align with a broader omnichannel approach to its markets as it continued expanding into marketplace platforms and services. Historically, Walmart successfully went to public in 1972 and further entered into China in 1996. In 2018, Walmart topped in the Fortune 500 list with a revenue of $485.8 billion, which is the fifth consecutive year that Walmart has been ranked as the top 500 company in the world. In recent year, Walmart is devoting itself to digital transformation, through data_based methods and integrated shopping scenarios to allow consumers shop in their unique style anytime and anywhere.

Making an accurate point forecast of the unit sales could indeed add values across the entire business, as a result, accurate prediction is extreme vital for big retailer like Walmart. One challenge of modeling is the correct usage for limited provided historical data. This research was based on the competition held by the organization Makridakis Open Forecasting Center(MOFC), who use cutting_edge forecasting research to provide business forecast training. It aims to help companies to achieve accurate predictions, estimate the levels of uncertainty, avoiding costly mistakes, and further in order to apply best forecasting practices. The Makridakis competitions target to provide a better understanding and advancement of forecasting methodology by comparing the performance of different methods in solving a well_defined, real_world problem.

[1]

II. Data

A. Data Background and Processing

Firstly, in order to improve the computing speed, we downcast the data form. For example, downcast 32 to 16 and 64 to 32. Below it the result after the downcast, which is quite good, the sales fine was sharply compressed from 454.5 MB to 97.1MB after the downcast.

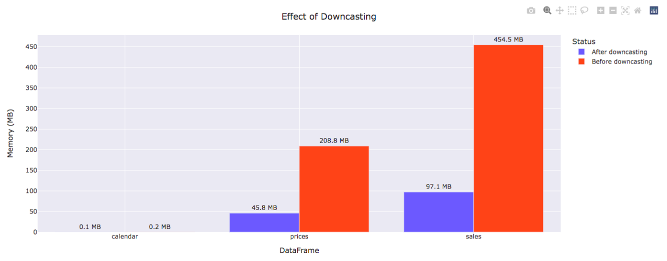


Fig. 2. Effect of Downcasting

The entire data_set mainly includes 3 separate files for us to predict unit of sales, which are “calendar.csv”, “sell_price.csv”, and “sales_train.csv” respectively, each of them containing different information:

1) Calendar.csv: contains information about the dates and types on which the products are sold.

Here are some variables and its explanations for this data file:

- **Date:** the date in the format of “year_month_day”.
- **Event_name & Event_type:** If there exists an event in that day, then the name and type of the event.
- **W_day:** The id of the weekday starting from Saturday.
- **Snap_CA, Snap_TX, and Snap_WI:** These 3 binary variables are binary variable (0 or 1) that indicate whether the stores of California, Texas, and Wisconsin allow for the snap purchase on the examined date. More specifically, “1” indicates allowed and “0” indicates not allowed.

As for this file, we accurately combined the data giving variable date as the index. If there are repeated character, it will be double merged. The data after merge will be showed in below part – data view.

2) Sell_price.csv: contains information about the price of the products sold per store and date.

Here are some variables and its explanations for this data file:

- **Store_id:** The id of the corresponding store.
- **Sell_price:** The price of the product for the given week and store. The price is provided per week, which means, the average across seven days.

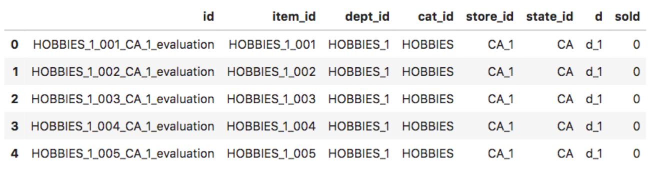
As for this file, we accurately merged the data using store_id, item_id, and wm_(yr_wk) into one table. The data after merge will be showed in below part – data view.

3) sales_train.csv: contains the historical daily unit sales data per store and product.

Here are some variables and its explanations for this data file:

- **d_1, d_2, d_3, … ,d_1941:** The number of units sold at day i.
- **Item_id:** The id of the product.
- **Dept_id:** The id of the department where the product belongs to.

As for this file, we turned the variable item_id into a category, which a unique data format for pandas. Also, we transferred the variable d_1 to d_249 into two integrated columns, which are days and sells. Here is part of the data after our adjustment, where columns are sharply reduced but row added at the same time.



	id	item_id	dept_id	cat_id	store_id	state_id	d	sold
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0

Fig. 3. Feature Engineering

In addition, in order to compare the model performance more easily, we add some average variables like item_sold_average, state_sold_average, store_sold_average, and cat_sold_average etc. in the model part. Take the item_sold_average as an example, it is the average of sold item based on the index item_id.

We divided the original data_set into 3 categories based on the returns. Where 0 stands for non_positive returns (i.e., the down trend), 1 stands for positive returns (i.e., the uptrend), and 2 stands for the situation contains a NaN result.

B. Data View

1) Distribution of items prices among stores: As we could easily observe from the picture above, we found out that the distribution of item prices is almost uniform for all the stores across California, Texas, and Wisconsin. As for California, the item Hobbies_1_361 is the most expensive item sold in Walmart, with a price among $ 0.5. Also, as for stores in Texas, different from California, the item Household_1_060

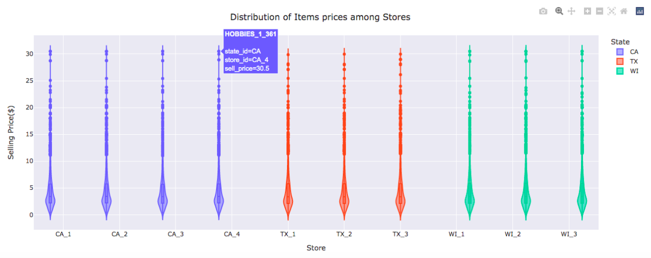


Fig. 4. Distribution of items prices among stores

is the costliest item with a price around $ 29.88. As for the 3 stores in Wisconsin, the purchase is not that unified. Like consumes in California, Hobbies_1_361 is the most expensive item for Store_1 and Store_3. However, Hobbies_1_225 is the costliest item for Store_2.

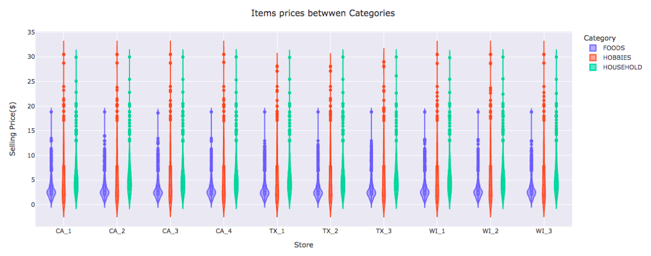


Fig. 5. Item price between categories

2) Item price between categories: From the above data, we could easily find out that the price for different types of goods basically have the same trends among stores in each state. Generally, the average price for category Foods is the lowest, fluctuating around $9.6, and the costliest food is the item Food_3_298, with a price at $18.83. The price for both category Hobbies and Household are much higher than category Foods, while category Hobbies is slightly more expensive than category Household. In addition, for each store in each state, item with the costliest price is not the same, which indicates that each state's consumers have their own consumption habits.

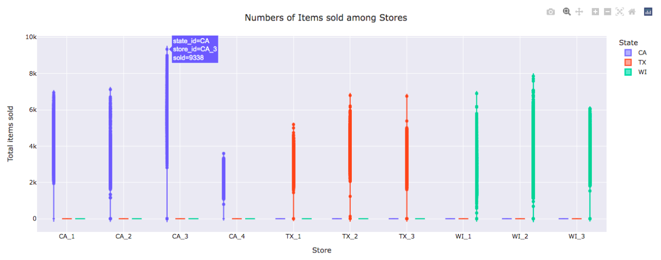


Fig. 6. Number of items sold among stores

3) Number of items sold among stores: According to the picture above, we know that the total items sold in California is larger than both Texas and Wisconsin, while the average total items sold for each state is 6,750, 6,244, and 6,940 respectively.

Among the 10 stores, both the stores with the highest and the lowest sales are all in California, which indicates that the provided data is tough and strong since it truly indicates the different consumption levels even in one state.

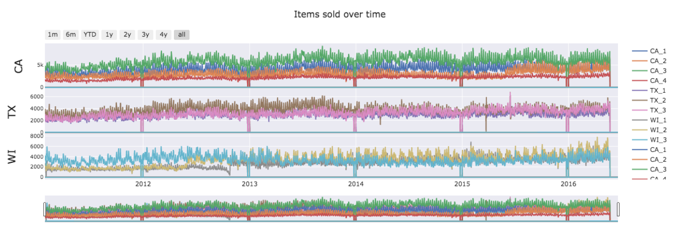


Fig. 7. Items sold over time(all)

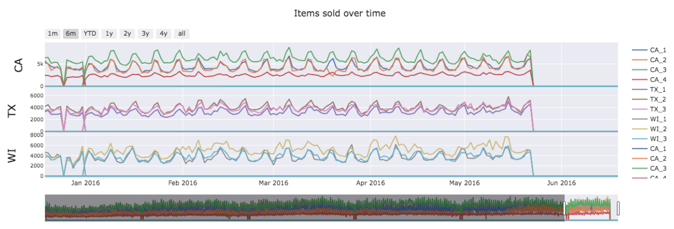


Fig. 8. Items sold over time(6 months)

4) Items sold over time: Above is the overall picture for items sold from year 2012 to 2016 and partial picture only for the half_year in 2016, which specifically indicates the sales performance for each store in each state. It is obvious that Store_3 in California has the best performance all the time.

5) Data Split: Our group divides the samples into training, validation and testing samples. The samples of days before 1914 are used to train the model, and the samples of days from 1914 to 1942 are used for validating. The remaining samples of days greater than 1942 includes out_of_sample test data sets.

III. LightGBM Model

GBDT (Gradient Boosting Decision Tree) is a long_lasting model in machine learning. Its main idea is to use weak classifiers (decision trees) to iteratively train to obtain the optimal model. This model has good training effects and is not easy to overfit Etc. According to statistics, more than half of the championship programs on Kaggle are based on GBDT. And LightGBM (Light Gradient Boosting Machine) is a framework that implements the GBDT algorithm, supports high_efficiency parallel training, and has faster training speed, lower memory consumption, better accuracy, support for distributed, and can quickly process massive amounts Data and other advantages.

A. Different from other tree_based algorithms

The growth method of LightGBM tree is vertical, and other algorithms are horizontal, which means that Light GBM grows the leaves of the tree, and other algorithms grow the level of the tree. LightGBM selects the leaves with the largest error for growth. When the same leaves are grown, the leaf_growing algorithm can reduce more loss than the layer_based algorithm. The figure below explains how LightGBM works:

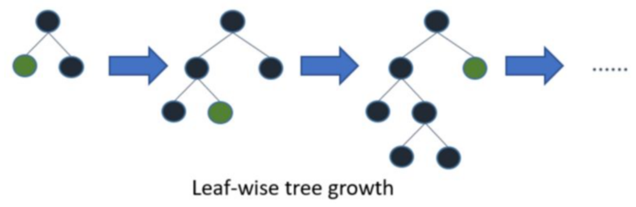


Fig. 9. LightGBM

The figure below explains how other boosting algorithm works:

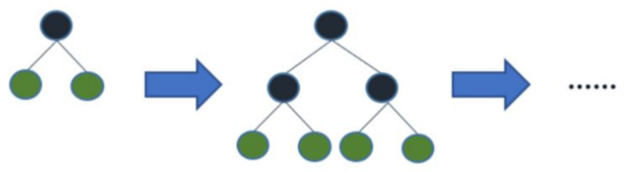


Fig. 10. Boosting Algorithm

B. Suitable data set and advantages

LightGBM can handle a large amount of data and occupies very little memory when running. LightGBM focuses on the accuracy of the results and also supports GPU learning.

C. Core parameter

1) n_estimators:: the number of iterations of boosting. The default setting is 100.Generally choose between 100 1000 according to the data set and characteristic data. A more conservative approach is to set a larger value with early_stopping_round to let the model automatically select the best number of iterations based on performance. Choosing a larger number of iterations will achieve better performance in the training set, but it is easy to overfit and cause the performance of the test set to decrease.

2) learning_rate:: learning rate. The default setting is 0.1, and the general setting is between 0.05_0.1. Choosing a relatively small learning rate can obtain stable and better model performance.

3) subsample:: If this parameter is less than 1.0, LightGBM will randomly select part of the data (row) in each iteration without resampling, which can be used to accelerate training and deal with overfitting. The default setting is 1, and it is generally set between 0.8 and 1.0 to prevent overfitting.

4) colsample_bytree:: If this parameter is less than 1.0, LightGBM will randomly select some features (col) in each iteration, which can be used to accelerate training and deal with overfitting. The default setting is 1, and the general setting is between 0.8 and 1.0 to prevent overfitting.

5) *max_depth::* The maximum depth of the tree model. The most important parameter to prevent overfitting is generally limited to between 3 and 5. It is the core parameter that needs to be adjusted and has a decisive effect on the model performance and generalization ability.

6) *num_leaves::* The number of leaf nodes on a tree. The default setting is 31, and the shape of the empty tree with max_depth is generally set to a value of (0, 2m̂ax_depth_1]. It is a parameter that needs to be adjusted and has a great impact on the performance of the model.

7) *min_child_weight::* The smallest hessian sum on a leaf. The default setting is 0.001, and the general setting is 1. It is not recommended to adjust, increasing the value will result in a shallower tree depth.

D. Parameter Adjustment

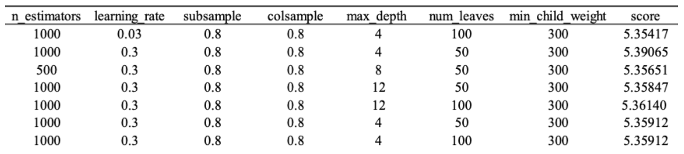
Our group adjust the following four parameters, and the tuning results are shown in the table below:

1) *learning_rate::* Generally speaking, the lower the learning rate, the better the final performance of the model performance, but a too small learning rate will often lead to overfitting of the model and affect the training time of the model. We search for a good value between 0.03 and 0.3 as the final model parameter. And when we keep the other parameters the same, the forecast score of 0.03 learning_rate is less than that of 0.3 learning_rate, which may also relates to the fact that too small learning_rate usually leads to overfitting of the model. Usually when the learning rate is small, the value of n_estimators will be large, and when the learning rate is large, the value of n_estimators will be relatively small. They are a pair of parameters that trade each other.

2) *n_estimators::* The greater the number of iterations, the better the model performance, but too large iterations will often lead to over_fitting of the model and affect the training time of the model. Generally, we choose a value between 500 and 1000. When we compare and adjust the parameters between 500 and 1000, we find that the fit score levels of 500 and 1000 are not much different, so we choose 1000 as our n_estimators. When training, we need to pay attention to the situation of overfitting in order to adjust the number of iterations in time. If the model has not stopped training when the performance of the test set tends to decline, it means that overfitting has occurred. . Usually in order to prevent overfitting, we choose a larger n_estimators, and then set early_stop_round=20 to stop the model where the test set works well.

3) *max_depth::* Select in 4, 8 and 12 to prevent setting a too large value from causing serious over_fitting. As shown in the figure below, with the other parameters unchanged, max_depth is set to 4, which has relatively good performance.

4) *num_leaves::* In LightGBM, the number of leaf nodes should be set in accordance with max_depth. While keeping other parameters as ideal, setting num_leaves to 50 and 100 has little effect on the result, so choose a smaller num_leaves to match the smaller max_depth.



n_estimators	learning_rate	subsample	colsample	max_depth	num_leaves	min_child_weight	score
1000	0.03	0.8	0.8	4	100	300	5.35417
1000	0.3	0.8	0.8	4	50	300	5.39065
500	0.3	0.8	0.8	8	50	300	5.35651
1000	0.3	0.8	0.8	12	50	300	5.35847
1000	0.3	0.8	0.8	12	100	300	5.36140
1000	0.3	0.8	0.8	4	50	300	5.35912
1000	0.3	0.8	0.8	4	100	300	5.35912

Fig. 11. Parameter Tuning Result

IV. Result

A. Feature Importance

Using the LightGBM.feature_importance() function to rank the importance of each feature of the trained LightGBM model. The following figure shows the analysis result of feature importance.

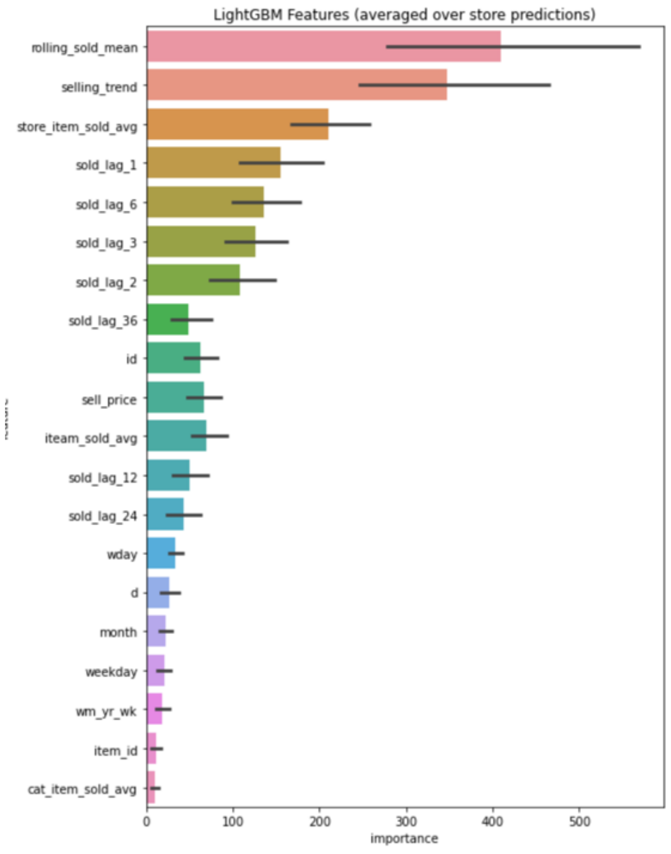
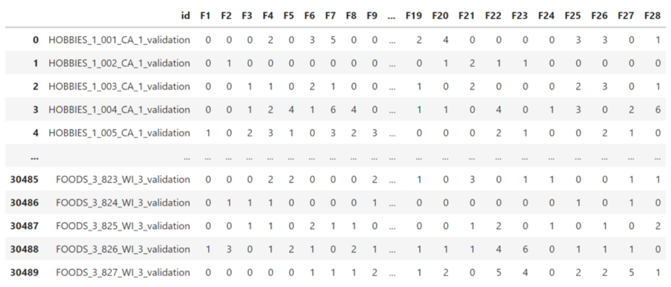


Fig. 12. Analysis Result of Feature Importance

From the evaluated feature importance results, rolling_sold_mean is the weekly average, selling_trend is the difference between the daily average and the corresponding average of the total number of days of sales, and store_item_sold_avg is the average sales of different products in different stores. These three featuers are very important for the prediction effect of the LGBM model. Besides, sold_lag_1, sold_lag_2, sold_lag_3, and sold_lag_6 have significant importance for model prediction, which is also in line with our common sense that the closer to the sales on the forecast day, the better the forecast performance of the sales on the forecast day.

B. *Model performance*

After the above tuning process, the parameters we selected for the LGBM model are n_estimators=1000, learning_rate=0.3, subsample=0.8, colsample_bytree=0.8, max_depth=4, num_leaves=50, min_child_weight=300, the final prediction score for the next 28 days is 5.39065. The following picture shows the form required for the Kaggle competition:



	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	...	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28
0	HOBBIES_1_001_CA_1_validation	0	0	0	2	0	3	5	0	0	...	2	4	0	0	0	0	3	3	0	1
1	HOBBIES_1_002_CA_1_validation	0	1	0	0	0	0	0	0	0	...	0	1	2	1	1	0	0	0	0	0
2	HOBBIES_1_003_CA_1_validation	0	0	1	1	0	2	1	0	0	...	1	0	2	0	0	0	2	3	0	1
3	HOBBIES_1_004_CA_1_validation	0	0	1	2	4	1	6	4	0	...	1	1	0	4	0	1	3	0	2	6
4	HOBBIES_1_005_CA_1_validation	1	0	2	3	1	0	3	2	3	...	0	0	0	2	1	0	0	2	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30485	FOODS_3_823_WI_3_validation	0	0	0	2	2	0	0	0	2	...	1	0	3	0	1	1	0	0	1	1
30486	FOODS_3_824_WI_3_validation	0	1	1	1	0	0	0	0	1	...	0	0	0	0	0	0	1	0	1	0
30487	FOODS_3_825_WI_3_validation	0	0	1	1	0	2	1	1	0	...	0	0	1	2	0	1	0	1	0	2
30488	FOODS_3_826_WI_3_validation	1	3	0	1	2	1	0	2	1	...	1	1	1	4	6	0	1	1	1	0
30489	FOODS_3_827_WI_3_validation	0	0	0	0	0	1	1	1	2	...	1	2	0	5	4	0	2	2	5	1

Fig. 13. Submission File Format

V. Team Work Illustration

A. *Members*

- Xinzhen Liao (20813829)
- Yuan Xu (20801498)
- Rui Ma (20736954)
- Yiyuan Shi (20745230)

B. *Contribution*

The whole team actively discuss the entire structure and idea for this project. Xinzhen Liao and Yuan Xu mainly in charge of the coding part, while Rui Ma and Yiyuan Shi are responsible for report writing. Although each of our teammate has different obligations, all of us help each other and study together through the whole project.

References

[1] [Online]. Available: https://www.cnblogs.com/wqbin/p/12780549.html. [Accessed: Sep. 14, 2020].

[2] David Marcotte Senior Vice President, Global Insights & Technology, Kantar, “2021 Top 50 Global Retailers Walmart continues its reign, closely trailed by Amazon, Alibaba and JD.com” Mar. 24, 2021. [Online]. Available: https://nrf.com/blog/2021-top-50-global-retailers. [Accessed: Mar. 24, 2021].