

Report Project Work in Machine Learning

Michele Vece

June 8, 2022

1 Introduction

The topic of this project work is sales forecasting. In details, it addresses the M5 Competition.

In the following, first, there is a brief review of the competition whose dataset is used. Then, the main issues encountered are listed and the top five solutions are briefly summarized. Finally, the proposed solution is illustrated.

2 Overview of the competition

2.1 Objective

Given a list of products and a list of stores where these items are sold, the objective of the competition consists in predicting the unit sales of each product in each store for the next 28 days.

Public leaderboard is based on predictions whose day $d \in [1914; 1941]$; private leaderboard, instead, takes into account days $d \in [1942; 1969]$.

2.2 Dataset

The M5 dataset is made available by Walmart and involves the unit sales of various products sold in the USA, organized in the form of grouped time series.

Products organization

More specifically, 3,049 *products* are sold across ten *stores*, located in three *states* (CA, WI and TX). The items are classified in three *categories* (hobbies, household and foods), further split in *departments*.

Figure 1 shows the organization.

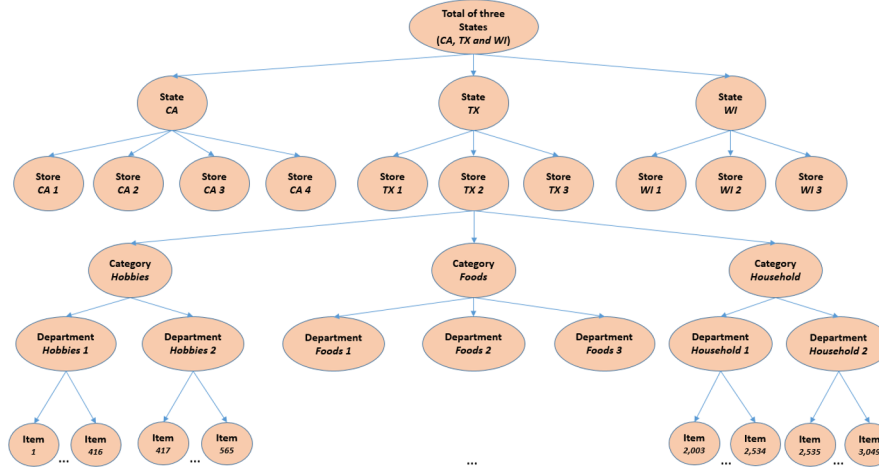


Figure 1: Overview of products organization. Image from [1]

Files

The M5 dataset consists of the following four files:

- `calendar.csv`: information about the dates the products are sold;
- `sell_price.csv`: information about the price of the products sold per store and date;
- `sales_train_validation.csv`: historical daily unit sales data per product and store (until day 1914), it does *not* contain the labels of the days in the public leaderboard;
- `sales_train_evaluation.csv`: historical daily unit sales data per product and store (until day 1941), it contains the labels of the days in the public leaderboard.

Further information about the fields of each file can be found in [1].

2.3 Metric

As indicated in [1], the metric used in the leaderboards is the Weighted Root Mean Squared Scaled Error (WRMSSE), which is obtained by weighting the RMSSE of each timeseries.

First, the accuracy of the forecasts is evaluated using the RMSSE:

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

where:

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

Second, the result is ranked using the Weighted RMSSE (WRMSSE):

$$WRMSSE = \sum_{i=1}^{42,840} w_i * RMSSE$$

where w_i is the weight of the i_{th} series. A lower WRMSSE score is better.

The weight of each series is computed based on the last 28 observations of the training sample of the dataset, i.e., the *cumulative actual dollar sales*, the sum of the units sold multiplied by their respective price.

Table 1 shows the aggregation levels involved in the M5 competition. They are equally weighted [1].

3 Main challenges

Amount of data

Joining and melting the provided dataframes ends up with a huge amount of data. As a consequence, it is required to find a way to partition data.

In feature engineering, one strategy may be first computing high-level features on aggregated data, then dividing data in subsets and computing low-level features on each subset. In model implementation, instead, the main strategy consists in dividing data in subsets and training a different model on each subset.

Here, the adopted solution consists in splitting data according to the states before the feature engineering phase. After that, dataframes are further split based on the store. Training is done on a per-store basis.

Cross correlation

Cross correlation refers to the presence of similar behaviours among timeseries, even if they differ in product, department, category, store or state.

Undoubtedly, this is an opportunity because rules that work for an item may work as well for others. However, it may become a *missed* opportunity, because it makes further difficult establishing a criterion for dividing data, since each kind of solution may separate subsets that share some common behaviour.

id	Aggregation Level	Number of series
1	Unit sales of all products, aggregated for all stores/states	1
2	Unit sales of all products, aggregated for each state	3
3	Unit sales of all products, aggregated for each store	10
4	Unit sales of all products, aggregated for each category	3
5	Unit sales of all products, aggregated for each department	7
6	Unit sales of all products, aggregated for each state and category	9
7	Unit sales of all products, aggregated for each state and department	21
8	Unit sales of all products, aggregated for each store and category	30
9	Unit sales of all products, aggregated for each store and department	70
10	Unit sales of product x , aggregated for all stores/states	3,049
11	Unit sales of product x , aggregated for each State	9,147
12	Unit sales of product x , aggregated for each store	30,490
Total		42,840

Table 1: Aggregation levels

Sales intermittency

Sales intermittency is one of the reason why this competition differ from the previous ones [1]. Sporadic sales are a distinctive feature for those categories of products that tend to be requested on special occasions, such as **HOBBIES**.

However, consecutive days of zero sales may simply indicate that a product is not currently available in the store. This issue is referred as out-of-stock problem. This might involve those items that are usually sold with high frequency, such as **FOODS**. In this case, we should distinguish between the demand and the offer of a product: are we forecasting the demand or the actual sales?

Two possible solutions are: (1) making a double prediction, by predicting, on one hand, the demand and, on the other hand, the probability that an item is out-of-stock; (2) using an objective function that works better with non-negative right-skewed distributions, such as Tweedie loss [2].

Atomicity of the prediction

It is required to make forecasts at product-level and on a daily basis. Due to this high level of specificity, they are more error-prone than those on aggregated data. Fortunately, this issue is mitigated by the final score, which takes into account also intermediate results at different levels of aggregation (such as predictions at state, store or category level).

Forecasting horizon and recursive features

The objective of the competition is forecasting unit sales for the next 28 days. As a general rule, latest days have higher predictive power w.r.t. non recent information. As a consequence, it is highly desirable to use them, but at the same time, data leakage should be avoided.

Figure 2 shows three main alternatives discussed in [3]:

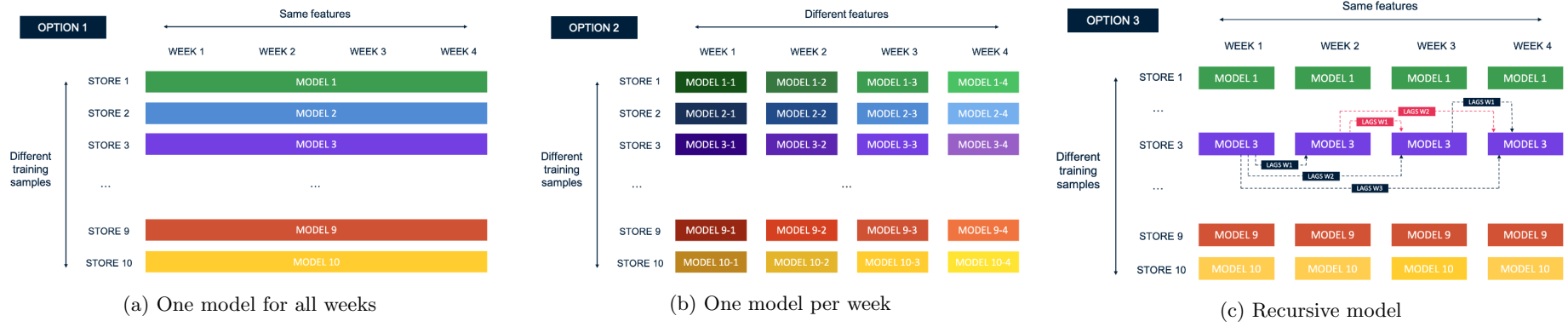


Figure 2: Alternative forecasting models. Images from [3]

- **one model for *all* weeks:** use the *same* model to make predictions for *all* weeks in the forecasting horizon (fig. 2a). Unfortunately, its simplicity comes with a huge constraint: only features available for predicting at $d + 28$ can be used. Therefore, information given by the 27 most recent lags should be discarded;
- **one model *per* week:** every model is learning from the most recent possible lags with respect to the constraint imposed by its prediction horizon (fig. 2b). It allows to better capitalize on lag information for the first weeks and thus improve forecast accuracy. The drawback is the higher complexity.
- **recursive model:** predictions generated for a given week will be used as lag features for the following weeks (fig. 2c). It uses as recent information as available. However, it may be unstable since eventually incorrect predictions are used as input to next weeks forecasts.

Here, recursive model is adopted. For the sake of simplicity, in the training phase ground truth from previous weeks is used (leading to data leakage). As a consequence, the error during the training is slightly smaller than the error obtained in the submitted predictions.

4 Brief review of other solutions

The following resumes the five winning submissions, further explained in [4]:

1. 220 models; combination of various LightGBM models trained per store (10 models), store-category (30 models) and store-department (70 models), and with two approaches (recursive and non-recursive); tweedie distribution, no early-stopping;
2. 50 models; LightGBM models are first trained per store (10 models) and then five different multipliers are used to adjust their forecasts and to correctly capture the trend;
3. 43 models; neural networks trained with an optimized tweedie distribution;

4. 40 models; non-recursive LightGBM models trained per store (10 models) and per week, namely, each week in the forecasting horizon is forecast separately using a different model;
5. 7 models; recursive LightGBM models trained per department (7 models). Then, forecasts are externally adjusted by using multipliers.

From the previous solutions and as stated in [4], it emerges that, in this competition:

- machine learning methods are preferred to deep learning models. In particular, most of the solutions are based on LightGBM models;
- solutions use cross-learning, by exploiting all the information in the dataset. As a consequence, a high amount of small models are proposed: each of them focuses on a different aspect/relationship;
- importance of cross-validation: last 28-day-long window(s) of available data were used to assess the forecasting performance, providing a reasonable approximation of the post-sample accuracy;
- tweedie loss is preferred to RMSE as objective function. It suits better sales intermittency since it assumes a non-negative, right-skewed distribution with most of the points concentrated around 0 [2];

5 Proposed solution

The work can be divided into six main parts:

- Preprocessing;
- Data Exploration (EDA);
- Feature Engineering (FE);
- Model implementation.

After that, predictions are computed and submitted.

5.1 Preprocessing

The preprocessing stage aims essentially at eliminating some redundant features and introducing a few new ones. However, the main contribution is probably the casting of the existing columns in order to reduce the amount of memory required.

Tables 2a, 2b, 2c resume the fields of the input dataframes at the end of this stage.

5.2 Data exploration

This section aims at discovering useful patterns or relationships inside data.

Column id	Meaning
store_id	id of the store
item_id	id of the item
wm_yr_wk	year and week
sell_price	price of a product

(a) Columns in prices

Column id	Meaning
wm_yr_wk	year and week
weekofmonth	week of the month
weekofyear	week of the year
dayofweek	day of the week
dayofmonth	day of the month
dayofyear	day of the year
month	month of the year
year	year
event_name_1	Name of the event (if any)
event_type_1	Type of the event (if any)
snap_XX	SNAP day in the state XX

(b) Columns in calendar

Column id	Meaning
id	id of the item in a store
item_id	id of the item
dept_id	id of the department
cat_id	id of the category
store_id	id of the store
state_id	id of the state
1	unit sales on day 1
...	...
1941	unit sales on day 1941

(c) Columns in sales

Table 2: Input dataframes after preprocessing

5.2.1 Cross correlation

Figure 3 shows that unit sales are not uniformly distributed.

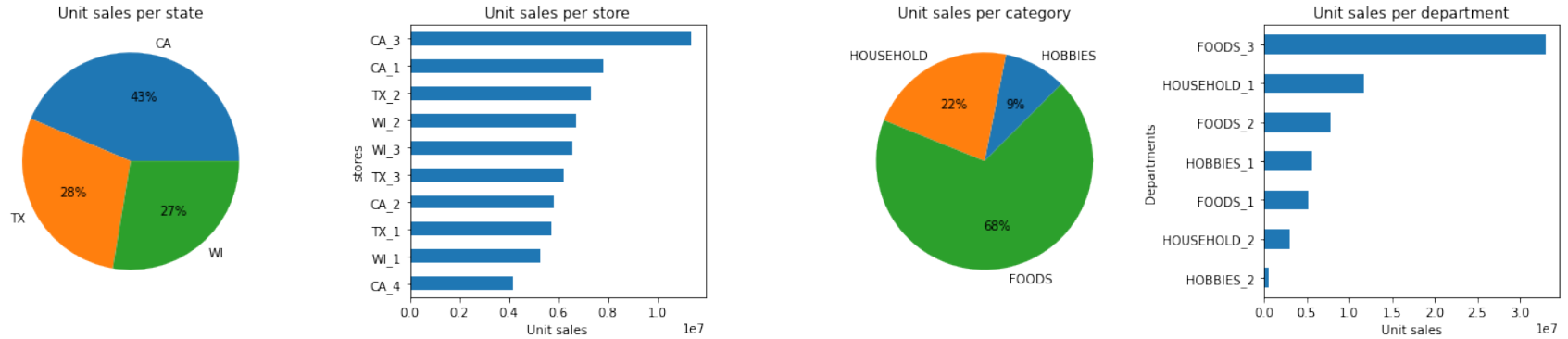


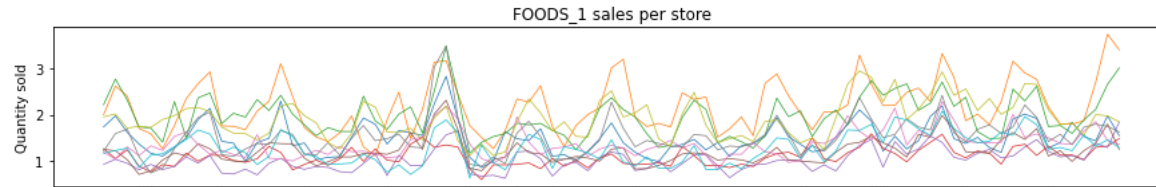
Figure 3: Total unit sales per state, store, category and department

Most of sales come from California, which has one additional store if compared to the other two states. However, units sales are not equal also among the stores, even in the same state. Still, in California there are both the store with the highest amount of sales (CA_3) and the one with the lowest (CA_1).

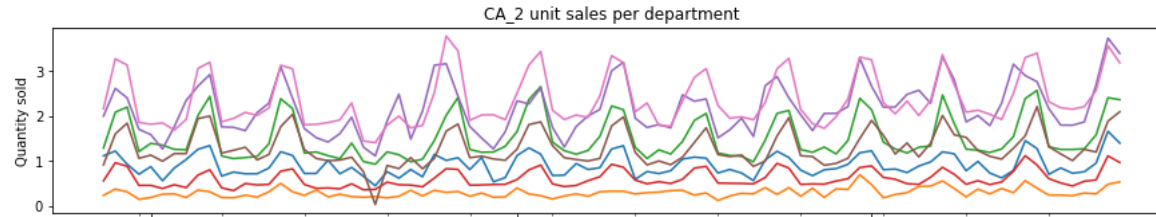
Regarding the categories and the departments, about 70% of the total sales comes from **FOODS** products, and **FOODS_3** represents about a half of the total unit sales.

Due to the high amount of data, finding a criterion to split data is highly recommended. Common strategies are creating store-level or department-level solutions. Figure 4a shows an example of the presence of similarities among unit sales of the same department but in different stores; while figure 4b shows how similar are the unit sales of different departments in a selected store. In both cases, there is some sort of correlation between the timeseries.

Splitting would produce datasets with a different number of records, which is not desirable. However, store-level datasets are less unbalanced than department-level datasets. This is one of the reasons why store-level models have been preferred and implemented.



(a) Example of unit sales of the same department: each line represents a different store.



(b) Example of unit sales in the same store: each line represents a different department

Figure 4: Examples of cross correlations across same store or department.

5.2.2 Seasonality

It is reasonable that in similar periods a similar amount of the same product is sold. As a consequence, average sales on different days, months, years etc. are computed.

It emerges that unit sales are higher:

- during the weekends w.r.t. the rest of the week;
- in the first half of the month (especially for **FOODS**);
- on snap days (especially for **FOODS_2** and **FOODS_3**);

Moreover, even if the impact of events is slightly different across the states, it is possible to outline that:

- unit sales are higher on some days, i.e., `SuperBowl`, `OrthodoxEaster`, `LaborDay`;
- on other days, such as `Thanksgiving` or `NewYear`, they are lower, instead;
- on `Christmas` no item is sold, probably because the stores are closed.

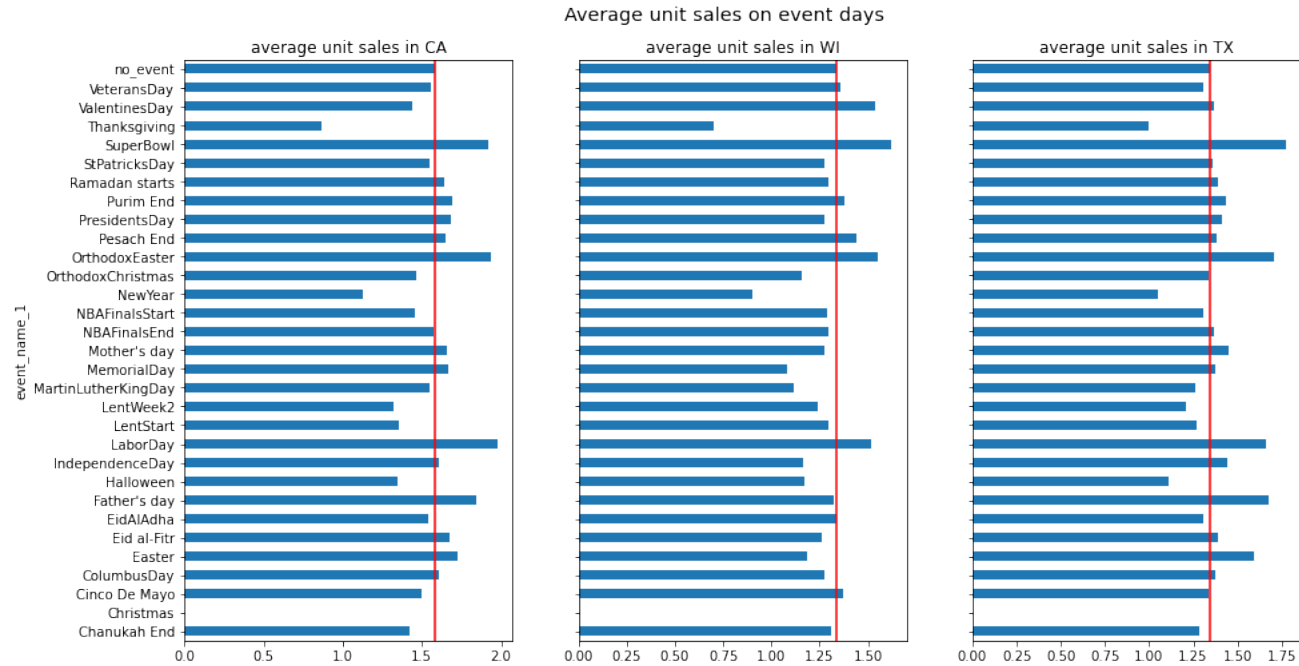


Figure 5: Average unit sales per each event.

5.2.3 Autocorrelation

Another characteristic to consider in timeseries is the lag, namely the unit sales n days before. To this purpose, autocorrelation is used.

Results change according to stores and departments, but in general *7-days* lags and its multiples are the best lag values. Also *monthly* lags are quite good, especially for `FOODS`. Obviously, further we go in the past, worse the lag value. `HOBBIES`, instead, do not exhibit relevant lag values.

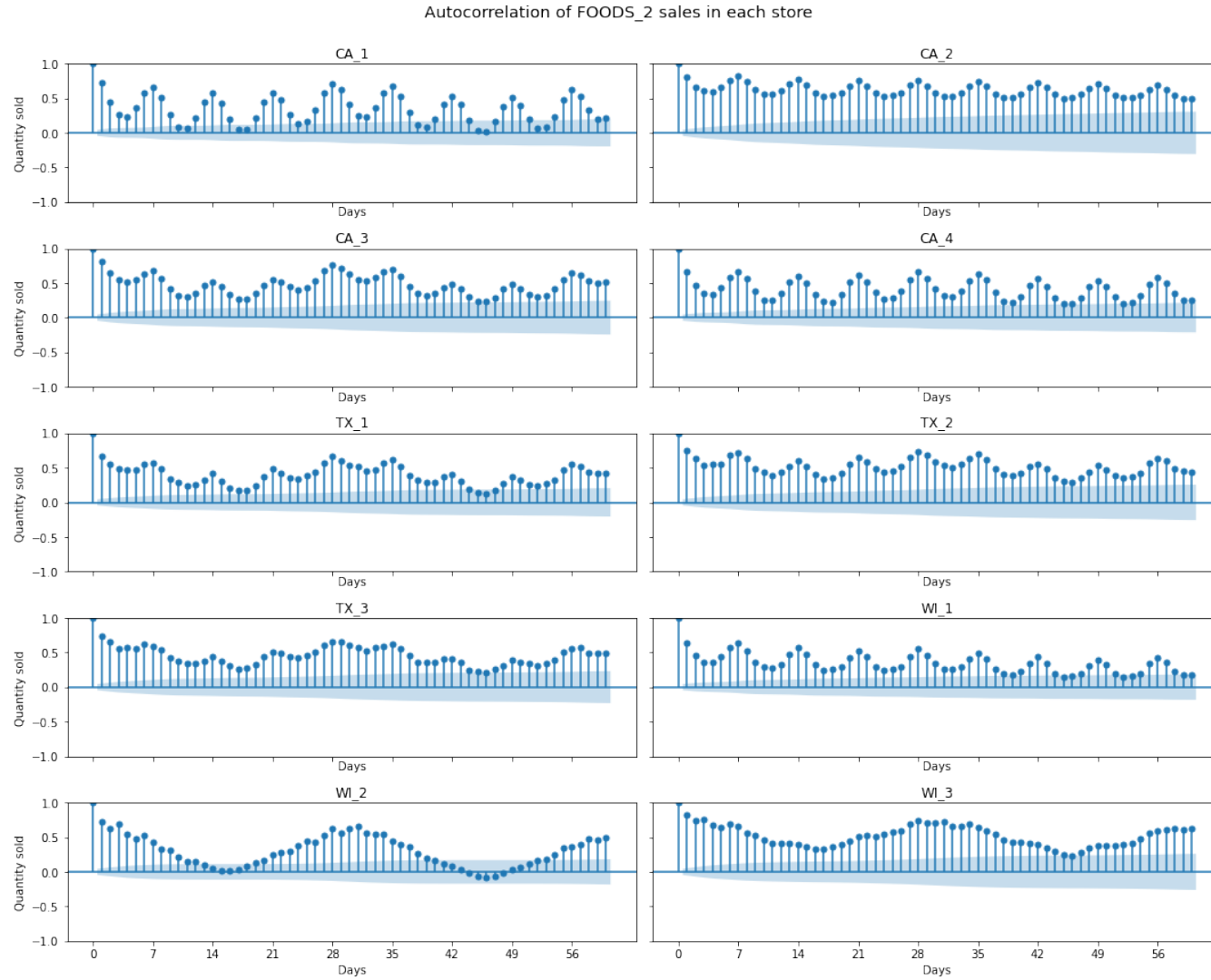


Figure 6: Example of autocorrelation for a department (FOODS_2) in different stores.

5.3 Feature engineering

This section aims at inserting new features obtained by combining information from already existing columns, such that they could be good predictors for the label. They can be derived from prices or sales information.

Column id	Meaning
<code>price_last_week</code>	difference (in %) wrt the price of the previous week
<code>price_next_week</code>	difference (in %) wrt the price of the next week
<code>price_same_dept</code>	ratio wrt the average price of all the items of the same department
<code>price_mean</code>	difference (in %) wrt the average price
<code>price_std</code>	standard deviation of the price

(a) Price related columns

Column id	Meaning
<code>rolling_mean_dayofweek</code>	rolling mean on the same day of the week, over the last 6 weeks
<code>rolling_mean_7_28</code>	rolling mean over 28 consecutive days, shifted by one week
<code>rolling_mean_28_r</code>	rolling mean over 28, 90, 180 and 365 consecutive days, shifted by 28 days
<code>avg_item_store</code>	average sales per item, store
<code>std_item_store</code>	standard deviation sales per item, store
<code>avg_item_state</code>	average sales per item, state
<code>avg_dept_store</code>	average sales per department, store
<code>avg_snap</code>	average sales on snap/non-snap days
<code>avg_event</code>	average sales on that event day
<code>avg_weekofmonth</code>	average sales on that week of month
<code>n_items</code>	number of items of the same department in the same store on a that day

(b) Sales related columns

Table 3: List of columns added during featuring engineering.

Price columns

Table 3a lists *price*-related features. They are obtained by comparing the current price of an item to the price of the previous or next¹ week, its average and standard deviation, and the average price of similar products (namely, items belonging to the same department).

Since, in general, there is not a large variation in prices between consecutive weeks, missing values are filled with a 0 (namely, no change in price).

¹Future prices are considered available at the moment of the prediction

Sales columns

Table 3b contains, instead, *sales*-related features. They consist in rolling means (over the same day of week or different windows of consecutive days), averages (on specific days or grouped per item, department, ...) and amount of similar products available in the store.

In case of averages, they are computed by using days until 1914, excluding dates in public leaderboard (used in the validation). This choice is motivated by the intention of avoiding possible leakages, such that the obtained solution works well also on unseen data.

In this case, missing values are filled by mean imputation. The average takes into account the attributes used (in the `groupby`) to compute the correspondent column.

These features include both recursive and non-recursive features, such that the first ones mitigate the drawbacks of the others, and viceversa: *recursive* ones use information from one to six weeks before, they are more informative since they are up-to-date, but may include prediction errors; *non-recursive* ones cannot contain errors since they are not predicted, but may be less informative since they are less recent.

5.4 Model implementation

The chosen model is a LightGBM, since it has been the most used and successful in this competition, as stated in [4]. The following choices and hyperparameters were adopted:

Number of leaves: it indicates the maximum number of leaves in a tree. Actually, the default value (31) is already the best for almost all stores;

Objective function: as already stated, Tweedie is adopted as cost function since better than RMSE in this competition. Its variance power is fine-tuned for each store in $[1.1; 1.5]$;

Metric: RMSE is used to measure the performance of the models since it is a regression task;

Cross validation: CV is adopted since it allows to reuse part of the validation as training data. Since the objective consists in predicting the nearby future and recent data is the most informative, using fresh data both as validation and training may improve the performance. In details, the last 3 periods of 28 days are used as CV;

Early stopping: used with the objective of preventing overfitting, it stops the training after 100 consecutive epochs in which no further improvement on cross validation is obtained;

Adaptive learning rate: the training starts with an higher learning rate ($lr_{start} = 0.1$) which allows the model to learn a lot in the first steps and which is decreased at each step, according to the following formula:

$$lr_{iter} = \max(lr_{min}, lr_{start} \cdot 0.99^{iter})$$

where $iter$, lr_{iter} are, respectively, the number and the learning rate of the current iteration and $lr_{min} = 0.05$ is the minimum learning rate value;

Bagging: used to speed up training and deal with over-fitting, as suggested in [5]. In details, every 10 steps 25% data is subsampled.

For each store, a different model is trained for a maximum of 2000 epochs. At the end, three **CVBoosters** are returned. The one with the lowest rmse on the public data is selected as the final model for that store.

5.5 Predictions

Predictions are computed on a weekly basis:

- first, sales for week n. 1 are forecast by using only historical data;
- second, week n. 1 predictions are added to the dataframe and recursive features are recomputed;
- third, sales for week n. 2 are forecast by using updated dataframe,
- finally, also week n. 2 predictions are added to the dataframe, and so on.

At the end, predictions for each store are grouped in a single file and submitted.

5.6 Final results

Figure 7 shows the overall WRMSSE used in the leaderboards and also its decomposition at different levels of aggregation.

As expected, higher the atomicity of the prediction, higher the error. In fact, the error dramatically increases starting from level 10, namely product-level predictions. Instead, it is (almost) under the average when forecasts concern aggregated results (levels 1 - 9).

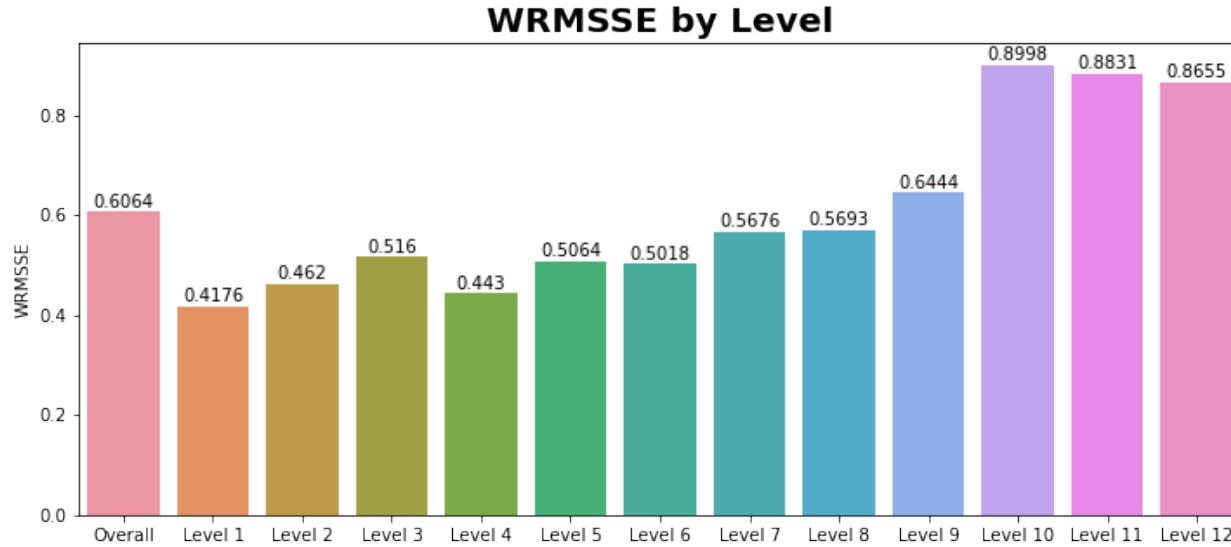



Figure 7: WRMSSE at different levels of aggregation

The proposed solution beats the top performing benchmark [4].
 Figure 8a shows public and private scores of the final submission. The proposed solution obtained 0.57415 as private score, which corresponds to the 46th position in the private leaderboard (fig. 8b).

YOUR RECENT SUBMISSION



m5_final_submission.csv

Submitted by Michele V · Submitted 19 hours ago





Score: 0.57415

Public score: 0.60636

↓

Jump to your leaderboard position

(a) Final submission scores

45	▲ 1665	Hiromitsu Kigure			0.57378	22	2Y
46	▲ 360	YK			0.57457	27	2Y

(b) Private leaderboard

Figure 8: Submission scores and private leaderboard

References

- [1] “THE M5 COMPETITION - Competitors’ Guide.” [Online]. Available: <https://mofc.unic.ac.cy/wp-content/uploads/2020/03/M5-Competitors-Guide-Final-10-March-2020.docx>
- [2] “Why tweedie works?” [Online]. Available: <https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/150614>
- [3] “Sales forecasting in retail: what we learned from the M5 competition.” [Online]. Available: <https://medium.com/artefact-engineering-and-data-science/sales-forecasting-in-retail-what-we-learned-from-the-m5-competition-445c5911e2f6>
- [4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The M5 Accuracy competition: Results, findings and conclusions,” *International Journal of Forecasting*, 2022. [Online]. Available: <https://doi.org/10.1016/j.ijforecast.2021.11.013>
- [5] “Lgbm parameters.” [Online]. Available: <https://lightgbm.readthedocs.io/en/latest/Parameters.html>