# 16 Days Sales Prediction for a Large Grocery Chain


50015627 JIANG Zhuoyang
33.33%
Responsibilities include:**Project framework and technical path; Main content, framework, and graphics of the PPT; Presentation demonstration; Report writing; EDA and feature engineering, model selection**
-
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

50016061 GAN Wenxi
33.33%
Responsibilities include:**Project framework and technical path; Supplementary content for the PPT; Presentation demonstration; EDA and feature engineering, model selection**
-
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

50015627 WANG Kaibo
33.33%
Responsibilities include:**Technical path; Supplementary content for the PPT; Presentation demonstration; Feature engineering, model selection, and model training**
-
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China


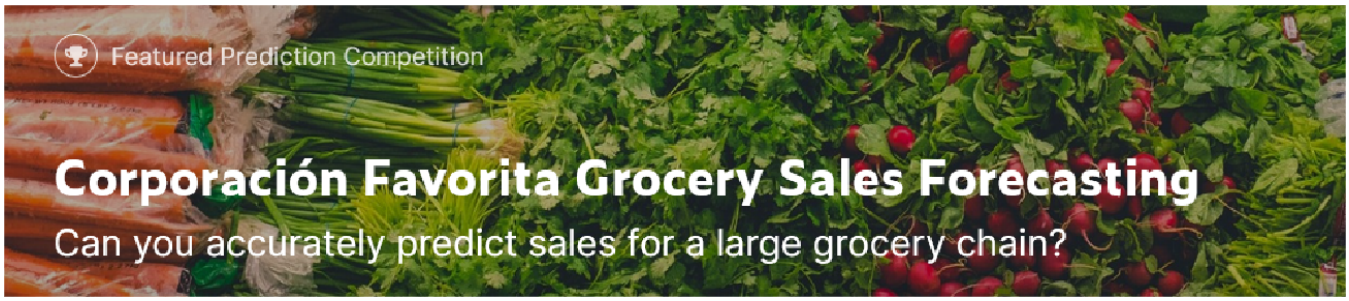


**Figure 1.** Kaggle Competition - Corporación Favorita Grocery Sales Forecasting


## Abstract

This project aims to address the time series unit sales prediction challenge within the Corporación Favorita Grocery Sales Forecasting competition. Through exploration and reconstruction of historical sales records, sales planning, and auxiliary information, we successfully constructed feature vectors for each individual product, organized into a comprehensive Feature-Batch encompassing all items. Leveraging the XGBoost machine learning model, which takes the complete Feature-Batch as input and predicts the unit sales for all items over the upcoming 16 days, we trained the model using the provided data to forecast the unit sales of thousands of products across various Favorita stores in Ecuador over the next 16 days.



***CCS Concepts:*** • **Computing methodologies** → *Supervised learning by regression.*

***Keywords:*** XGBOOST, Feature Engineering, Prediction, Machine Learning


## 1 Project Introduction

### 1.1 Initial Motivation and "Role Model"

In the given topic provided by our teacher, we chose this task mainly because we encountered relevant knowledge in the field of "time series prediction" during stock forecasting assignments, and we felt the learning value and application prospects of this machine learning domain. Therefore, our initial motivation is to fully engage in learning the practical application of machine learning time series prediction tasks.

To achieve this, we primarily need to clarify several important subtasks aimed at learning: First, we should define the pipeline for time series prediction tasks. Second, we need to understand how to model based on time series prediction tasks, defining how the objective should be modeled as output and how the data should be modeled as input. Third, we need to familiarize ourselves with technical details at various stages of time series tasks, such as initial data exploration, mid-term model selection, and late-stage model evaluation.

Based on these motivations and tasks, we have outlined the fundamental approach to completing this project: within a more refined application-oriented competition, using an existed excellent solution as a "Role Model", we aim to accomplish the several learning-oriented subtasks we've proposed.

We selected and extensively studied the solutions of the top competitors in the competition [3–5], and ultimately found that they employed relatively similar input-output modeling approaches. Therefore, we began our own practical implementation based on this insight.

## 1.2 Project Background

This project originates from a Kaggle competition — "Corporación Favorita Grocery Sales Forecasting." The competition requires us to utilize historical sales records and supplementary data to forecast the unit sales for 16 days in the future for numerous items sold across various Favorita stores in Ecuador.

Through this competition, we aim to familiarize ourselves with the process of time series prediction using extensive datasets, gained through participation in a well-established competition.

## 1.3 Task Analysis

### 1.3.1 Requirements:

The requirements of this competition unveil two critical pieces of information: firstly, it provides historical transaction records as the foundational dataset, and secondly, it necessitates predicting unit sales for the upcoming sixteen days.

From these two pieces of information, it's evident that this is a time-series forecasting problem, ideally suited for solving using machine learning models. Therefore, our analysis of this task's requirements from a machine learning perspective involves concretizing the abstract task into three dimensions: "input," "model," and "output." As depicted in the diagram below, historical data needs to be transformed into the input for machine learning models through the determination of features, while the prediction target required by the task needs to be modeled as the output of the machine learning model. Once both input and output are explicitly designed, we can establish the data format suitable for the model, enabling us to define the input and labels for training samples to construct a training dataset.

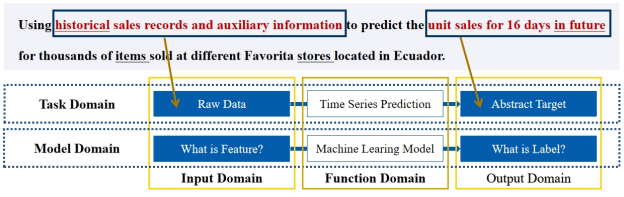


**Figure 2.** Requirements of the task

### 1.3.2 Data:

Having concretized the abstract task requirements into stages for machine learning, we conduct a preliminary analysis of the data provided to clarify the specific operational elements.

As illustrated in the above diagram, we primarily specify two crucial operational elements:

Firstly, the data input in the form of features comprises "One Feature-Batch of 16 Days." Within one feature-batch, it encompasses all transactional information for "a 16-day cycle" related to each city's every store and every item sold. These transactional details can be constructed into three types of features:

- The first type encompasses historical sales time-series features preceding "the 16-day cycle."
- The second type involves promotional sales time-series features during "the 16-day cycle."
- The third type pertains to specific identification features for "every city's every store and every item."

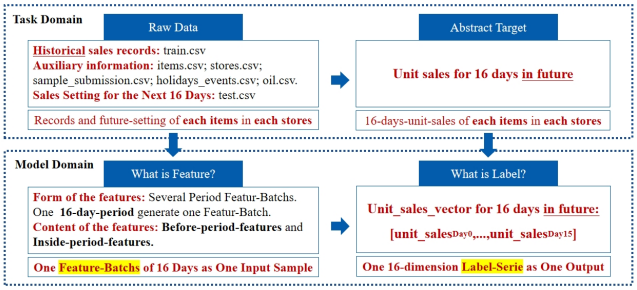


**Figure 3.** Data of the task

Secondly, the data output in the form of labels comprises "One 16-dimension Label-Series of 16 Days." Within one label-series, it includes a 16-day unit sale vector for "a 16-day cycle" encompassing all city-store-item combinations.

### 1.3.3 Technical Path:

Based on the concrete stages of machine learning tasks and specific operational elements in the data, we can construct the following technical path as depicted in the diagram below:

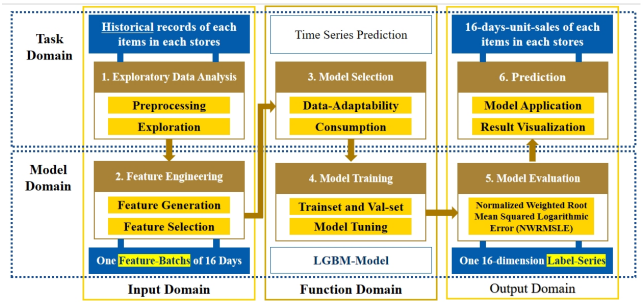


**Figure 4.** Pipeline of the task

Initially, we conduct fundamental Exploratory Data Analysis (EDA) and preprocessing on the imported data to explore potential information that could serve as possible feature elements. Subsequently, following the established approach, we perform Feature Engineering on the preprocessed data to build an efficient input system and the foundational elements for model training.

Next, we select a model that aligns with the input system and is efficient, leading to the construction of a training

dataset and subsequent model training. Finally, we evaluate the model's performance on the test set and proceed with the deployment of predictions in subsequent stages.

## 2 INPUT DOMAIN: From Data to Feature

### 2.1 Data Import

After sequentially importing the given datasets, we outputted the initial rows of each table to observe their content and format. Through preliminary assessment, I categorized these datasets into the following types:

1. **train.csv**: Firstly, the data in the 'train.csv' table is the most comprehensive, mainly displaying historical unit sale information and historical sales planning time-series features, which constitute our primary historical sales records.
2. **test.csv**: Secondly, the 'test.csv' table provides sales planning time-series features for "the 16-day cycle." These features, categorized as the second type, are also present in 'train.csv.' However, what's unique about 'test.csv' is that it solely provides sales planning time-series features for the specified period without offering unit sale information for that 16-day cycle, making it future-oriented information.
3. **stores.csv** and **items.csv**: Thirdly, among the remaining tables, 'stores.csv' and 'items.csv' contain crucial location and product identification information, corresponding to the third type of features—specific identification features for "every city's every store and every item."
4. Fourthly, several other tables contain small amounts of data with unique characteristics, suitable as supplementary features for unconventional insights. We need to conduct Exploratory Data Analysis (EDA) on them to confirm their value.

### 2.2 Exploratory Data Analysis (EDA)

Based on initial observations, the information provided by 'train.csv' and 'test.csv' is crucial and serves as the backbone for data exploration. 'stores.csv' and 'items.csv,' due to their relevance to the primary data, are also explored alongside the main data. However, the remaining tables are only suitable for displaying distinctive data characteristics through limited exploration to determine their usability.

Therefore, we initially conducted simple data exploration on three tables with small data volumes and strong specific characteristics. As depicted in the diagram below (Figure 5), we explored 'No. of transactions per Month/Week/Store for Every Year,' 'No. of Events per state and Sales on These Days,' and 'No. of Events per state and Sales on These Days.' We found that their patterns were not clear, and the data quality varied significantly. Hence, for now, we refrain from introducing these features to reduce data processing and enhance efficiency.

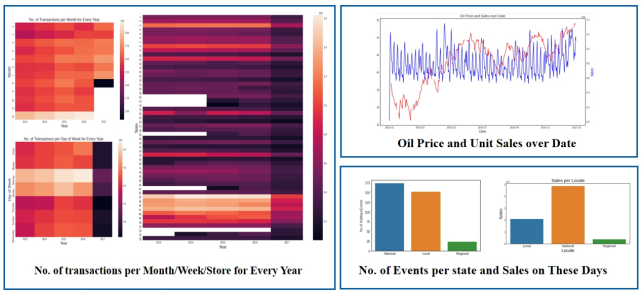


**Figure 5.** EDA on relevant but not primary tables

Subsequently, we explored the remaining four highly important tables and found that 'Date/Day/Locate/Type/Cluster' all exhibited a certain degree of variation in the average sale numbers. Consequently, they represent crucial pieces of information that need consideration.

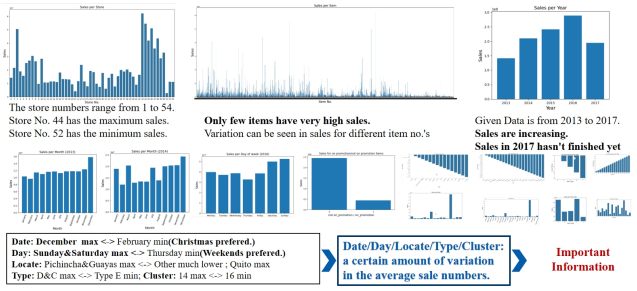


**Figure 6.** EDA on important tables

### 2.3 Data Preprocessing

Following the data exploration, we haven't yet performed a sequential observation for a time-series prediction task. This is mainly because the original data organizes information using time as a key but doesn't explicitly utilize time to construct time-series features. Therefore, in data preprocessing, we focus on leveraging temporal information. Utilizing the identification information provided by store and item columns as unique identifiers for each item, we transform the tables from "each row representing the sale of items at each time point" to "each row denoting the sales of each item at every time point in a time series."

### 2.4 Features Engineering

After conducting EDA and data preprocessing, we identified valuable information within the data and explicitly constructed time-series features for unit sales, providing an input framework for the task of predicting unit sales over time.

Next, we further refine this input framework into concrete input features. Firstly, it's essential to clarify the target of feature generation. Since the prediction task requires forecasting unit sales for all items across all stores for a future

16-day period, we need to create features based on a 16-day reference frame for all items. This means our model needs to take as input a Features Vector encompassing all items simultaneously for prediction, where the information within each item's Features Vector is extracted with reference to a 16-day period.

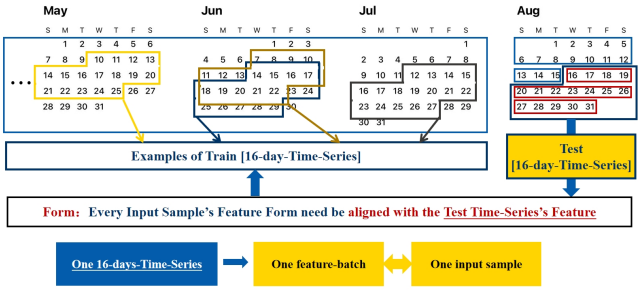


**Figure 7.** Feature generation

The Features Vector for all items constitutes what we previously referred to as a "Feature-Batch" during our data observations. Ultimately, each input into the model comprises a complete Feature-Batch. In other words, a Feature-Batch serves as an input for the model.

Moving forward, we specify the generation approach for various categories of features within the Feature-Batch. Based on previous data understanding, three crucial types of features were identified: historical sales time-series features before "a 16-day cycle," promotional sales time-series features during "a 16-day cycle," and specific identification features for "every city's every store and every item." Thus, based on the data reorganized according to time-series, we selected specific "16-day cycles" and extracted the following four types of features for each cycle: Promotional Features, Sales Features Depending on Promotion, Sales Features, and Count Features. Within each feature category, two distinctions based on time were made: 1. If features were derived from time-series information before "a 16-day cycle," they were termed "past features." 2. If features were derived from time-series information during "a 16-day cycle," they were termed "cycle features."

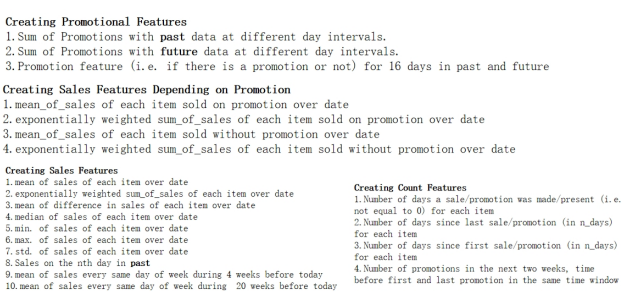


**Figure 8.** Feature types

It's important to note that while generating "cycle features" in 'train.csv,' although we have historical sales information for the 16-day cycle, we only use the information available in 'test.csv' to generate these features. This is because during prediction, we won't have access to the sales information for that specific "16-day cycle" used for prediction and can only rely on the promotional sales features explicitly provided in 'test.csv.'

Apart from these two types of time-series-related features, we also extracted features for product identification, where each row of the Feature-Batch corresponds to a unique set of product identification features. This resulted in the following organization of features:

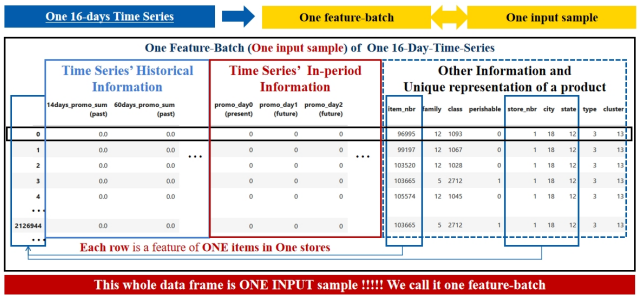


**Figure 9.** Feature organization

These steps collectively generated a total of 664 features. Subsequently, we utilized the interpretability of Random Forest to filter out the top 200 features as the final feature vector's dimensions. This is done to mitigate the curse of dimensionality and make the model training more efficient.

After Features Engineering, we utilized this feature framework to construct input feature vectors, facilitating subsequent model training and deployment.

# 3 MODEL DOMAIN: Model Construction

## 3.1 Training Dataset

We've designed the input feature vectors and explicitly defined the output format in the initial task analysis as follows: "One 16-dimension Label-Series of 16 Days" (comprising a 16-day unit sale vector for all city-store-item combinations within "a 16-day cycle").

Accordingly, the format of our constructed training dataset is illustrated in the diagram below. The task requires us to predict unit sales from 8.16 to 8.31. Therefore, we need to create a Features-Batch for the 16-day period from 8.16 to 8.31 as the final test data, forming the Testset. To train and evaluate the model, we utilize information solely from 'train.csv.' We selected five date ranges starting from 6.28/7.05/7.12/7.19/7.26, each representing a 16-day cycle, constructing five combinations of "Features-Batch and corresponding Label-Series." The first four are used for training the model, constituting

the Trainset, while the last one is used to assess the model's training performance, forming the Valset.

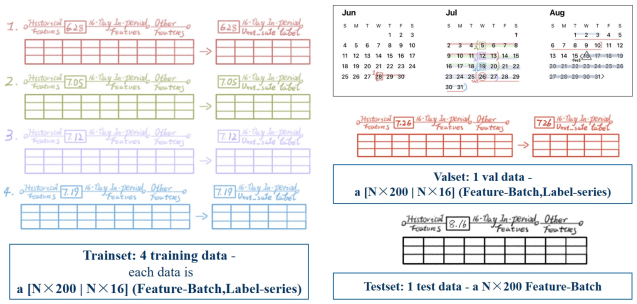


**Figure 10.** Training Dataset Organization

### 3.2 Model Selection

After exploring and comparing various models, we initially opted for XGBoost [1] to train the model. This choice was made due to certain limitations found in other models. Linear Regression showed poor performance in multi-feature training, encountering underfitting issues. On the other hand, LSTM [2] is time-consuming and doesn't easily accommodate the addition of other features. Meanwhile, XGBoost consistently demonstrates lower training loss for multi-feature scenarios compared to Linear Regression, offering better performance. Moreover, XGBoost is notably faster than LSTM, making it a preferable choice.

### 3.3 Model Training

During model training, we encountered a new challenge: the impact and manner in which historical sales information and promotional sales information influence unit sales for each day within the "16-day cycle" vary. Hence, we need to train 16 different sub-models to predict the unit sales for all items on each day within the "16-day cycle" and obtain a one-dimensional Label-Series for each day. Finally, we'll combine these 16 single-dimensional Label-Series into the ultimate 16-dimensional Label-Series as the prediction result.

The training process involves repeating the following steps for each day within the 16-day period:

1. Select the single-dimensional Label-Series for that day from the four 16-dimensional Label-Series used for training.
2. Sequentially input the four training Feature-Batches into the model, iterating through each row of data.
3. Input each row's Feature Vector into XGBoost for computation, obtaining a label.
4. Utilize the loss function with "the calculated label for each row" and "the label for that row from the single-dimensional Label-Series for that day" to optimize parameters.
5. Iterate through all rows of the four Feature-Batches in the Trainset, completing steps 3 and 4 for each row, thus finishing all training for the model corresponding to that day. The model's training effectiveness can be assessed on the Valset.

The model's configured hyperparameters are as follows:

```
%%time

# defining space for searching optimal parameters
space = [
        Real(0.4, 0.8, name="colsample_bytree"),
        Real(0.01, 0.5, name="gamma"),
        Real(0.001, 0.1,"log-uniform", name="eta"),
        Integer(3, 10, name="max_depth"),
        Integer(1, 5, name="min_child_weight"),
        Real(0.4, 0.8, name="subsample"),
        ]
# Using high no. of trees (i.e. 2000) to get better model performance with early stopping.
boost_rounds = 2000

count=0
model_hyper_params = [ 'colsample_bytree', 'gamma', 'eta', 'max_depth', 'min_child_weight', 'subsample']

# Objective function which will return nwrmsle .
objective_function = partial(return_model_assessment,
                                                    X_train=X_train, y_train=y_train, X_val = X_val,y_val= y_val ,
                                                    model='xgb', n_days=4, items=items ,
                                                    features= filtered_features,num_boost_rounds= boost_rounds)

# Running the algorithm
n_calls = 10 # number of times to train model
results = gp_minimize(objective_function, space, base_estimator=None, n_calls=n_calls, n_random_starts=n_calls-1, random_state=42)
```

**Figure 11.** Model's hyperparameters configuration

The training process proceeds as follows (using the model for the last day as an example):

```
================================================
Step 16
================================================
[0]     train-rmse:0.99575      val-rmse:0.98797
[50]    train-rmse:0.66454      val-rmse:0.67646
[100]   train-rmse:0.59041      val-rmse:0.61506
[150]   train-rmse:0.57012      val-rmse:0.60350
[200]   train-rmse:0.56010      val-rmse:0.60059
[250]   train-rmse:0.55295      val-rmse:0.59927
[300]   train-rmse:0.54718      val-rmse:0.59861
[350]   train-rmse:0.54203      val-rmse:0.59804
[400]   train-rmse:0.53752      val-rmse:0.59773
[450]   train-rmse:0.53347      val-rmse:0.59748
[500]   train-rmse:0.52993      val-rmse:0.59721
[550]   train-rmse:0.52651      val-rmse:0.59710
[600]   train-rmse:0.52317      val-rmse:0.59699
[650]   train-rmse:0.52002      val-rmse:0.59692
[700]   train-rmse:0.51691      val-rmse:0.59687
[750]   train-rmse:0.51391      val-rmse:0.59674
[800]   train-rmse:0.51097      val-rmse:0.59668
[850]   train-rmse:0.50795      val-rmse:0.59666
[900]   train-rmse:0.50521      val-rmse:0.59661
[950]   train-rmse:0.50245      val-rmse:0.59666
[1000]  train-rmse:0.49961      val-rmse:0.59667
[1039]  train-rmse:0.49742      val-rmse:0.59671
CPU times: user 5h 19min 3s, sys: 4min 3s, total: 5h 23min 6s
Wall time: 4h 59min 55s
```

**Figure 12.** Model's training process

Through this process, we can train all 16 XGBoost models.

## 4 OUTPUT DOMAIN: Evaluation and Prediction

### 4.1 Model Evaluation

The competition requires us to use nwrmsle as the final model evaluation metric, which stands for Normalized Weighted Root Mean Squared Logarithmic Error. This metric

is suitable when predicting values across a large range of orders of magnitudes. It avoids penalizing large differences in prediction when both the predicted and the true number are large: predicting 5 when the true value is 50 is penalized more than predicting 500 when the true value is 545. The evaluation metric is defined as follows:

$$NWRMSLE = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(\ln(\hat{y}i + 1) - \ln(y_i + 1)\right)^2}{\sum i = 1^n w_i}}$$

We utilized the "perishable" information provided in items.csv to define weights according to the project's requirements. Perishable items were assigned a weight of 1.25, while all other items were assigned a weight of 1.00. This enabled the creation of an NWRMSLE calculation function:

```
weight = items["perishable"] * 0.25 + 1
def calculate_nwrmsle(true, pred, weight):
    temp = (true - np.array(pred).transpose())**2
    temp = temp.sum(axis=1) * weight
    nwrmsle = np.sqrt(temp.sum() / weight.sum() / 16)
    return nwrmsle
```

**Figure 13.** NWRMSLE calculation function

On the Valset, applying the 16 models resulted in prediction results (Result-Series), which, when combined with the Label-Series, were inputted into this function to calculate the model's evaluation effect. Here, we showcase the NWRMSLE prediction effect for the best-performing model:

- Model –> **Xgboost**
- Best Parameters :
  - *colsample_bytree* = **0.436243**
  - *gamma* = **0.313009**
  - *eta* = **0.005820**
  - *max_depth* = **10**
  - *min_child_weight* = **3**
  - *subsample* = **0.743976**
- Best Score (NWRMSLE) –> **0.5941**

**Figure 14.** Best-performing model

### 4.2 Model Deployment

After obtaining the models, we sequentially applied the 16 models to the Testset, resulting in 16 single-dimensional Result-Series, each representing the unit sale of all items for each day between 8.16 and 8.31. By combining these, we obtained the final prediction result—an ultimate 16-dimensional Result-Series.

## 5 Conclusion

This project has achieved commendable results within the Corporación Favorita Grocery Sales Forecasting competition.

Through a refined understanding, analysis, and reorganization of the task, we delineated the nature of the task, established the correspondence between the data and task, and outlined the technical roadmap.

Thorough exploratory data analysis (EDA), preprocessing, and feature engineering enabled us to construct a Feature-Batch with strong interpretability and excellent feature quality as the model input. This integration and extraction of sales information from different dates provided robust support for model training and prediction.

Following the comparison of various models for their adaptability and performance in our task, we opted for the XGBoost approach to build multiple sub-models. This strategic choice facilitated highly accurate predictions of future 16-day product sales, offering an effective method for sales forecasting in the retail industry.

## References

[1] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[2] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[3] oberoiheman. 2020. *Corporaci-n-Favorita-Grocery-Sales-Forecasting*. https://github.com/oberoiheman/Corporaci-n-Favorita-Grocery-Sales-Forecasting

[4] sjvasquez. 2018. *web-traffic-forecasting*. https://github.com/sjvasquez/web-traffic-forecasting

[5] weiwei. 2018. *1st Place LGB Model(public:0.506, private:0.511)*. https://www.kaggle.com/code/shixw125/1st-place-lgb-model-public-0-506-private-0-511/script