Forecasting Unit Sales of Retail Goods

Module: 2022 - CMT400

MSc Advanced Computer Science

Supervisor: Yuhua Li

Author: Yuxi Wu

School of Computer Science and Informatics, Cardiff University

2022 – 12 - 05

Forecasting Unit Sales of Retail Goods

ABSTRACT 5

INTRODUCTION 6

1.1 Background and Objectives 6

1.2 Problem Description and Task Analysis 6

1.3 Research Significance 7

1.4 Advantages Compared with Other Methods 7

LITERATURE REVIEW 9

2.1 Background of the Literature 9

2.2 Field and Scope of Investigation 10

2.3 Analytical Methods Mentioned in the Literature 10

2.4 Ways to Improve Robustness and Improve System Performance 13

2.5 Limitations of the Studies Covered in the Literature 13

2.6 Future Directions Mentioned in the Literature 15

THEORETICAL BASES AND HYPOTHESIS 17

3.1 Principles of the LightGBM Algorithm 17

3.2 Hypothesis 20

METHODOLOGY 22

4.1 Data Sources 22

4.2 Data Description 22

4.3 Sample Selection 23

4.4 Way of Solving the Problem 23

4.5 How to Generate Output 24

4.6 How to Prove the Correctness 24

IMPLEMENTATION 26

5.1 Introduce Associated Libraries 26

5.2 Load Data Sets and Reduce Memory Usage 26

5.3 Encode the Category 28

5.4 Melt and Merge 28

5.5 Feature Engineering 30

5.6 Split the Dataset 32

5.7 Evaluation Method 32

5.8 Model Building 34

5.8.1 'Regression' as the Objective Loss Function 34

5.8.2 'Poisson' as the Objective Loss Function 35

5.9 Design Document and Flowchart 35

DATA ANALYSIS AND DISCUSSION 37

6.1 Generation of Outputs 37

6.2 Features' Importance Analysis 38

6.2.1 For 'Regression' Objective Loss Function 38

6.2.2 For 'Poisson' Objective Loss Function 39

6.3 Time-consuming Analysis 40

6.4 Accuracy Analysis 40

6.4.1 Metrics Related to the Output 40

6.4.2 Public Score 43

6.5 Comparison of Outputs and Assumptions 44

6.6 Discussion 45

CONCLUSION AND RECOMMENDATION 47

7.1 Summary and Conclusions 47

7.2 Limitations and Recommended Future Studies 47

REFERENCES 50

Abstract

In some large chain retail stores, such as Walmart, there are a large number and variety of goods sold every day. To improve the service quality and revenue of the store, and at the same time to ensure a balance between supply and demand, the key point is how to do it in the rapid circulation of goods, which means how to generate accurate forecasts for them. It is very challenging to build a regression model based on limited historical data. Still, with the continuous development and increasing usage of machine learning, to a certain extent, it could solve problems in an efficient and accurate way. At the same time, traditional forecasting methods need references with a lot of additional information to achieve the same goal. Based on historical data and extracted eigenvalues, this project can also be seen as a kind of data mining problem, where regression methods are used to predict the sales of a single item for Walmart stores in 45 regions of 3 states in the United States. Also, as a time series forecasting problem, it is based on the data set provided by Kaggle, and Light Gradient Boosting Machine (LightGBM) is applied to predict the demands of Walmart products in the next 28 days. In this process, two sets of different parameters are used to test the impact on the model, and then the model could be optimized according to the accuracy of the prediction results.

Keywords: machine learning, time series problem, retail forecasting, decision tree algorithm.

Introduction

1.1 Background and Objectives

Walmart was established in the United States in 1962 and has now developed into the world's largest retail chain. The success of such a business giant lies not only in the continuous provision of excellent products and services but also in its excellent data analysis teams, especially those who provide cutting-edge predictive research for the company. Since accurate commodity sales forecasts play a vital role in the retail industry, missing opportunities due to inaccurate forecast result is less common than ever. Furthermore, the purpose of this study is to finely predict the sales of each item in Walmart stores in the next 28 days from 24th Apr 2016 and to improve the accuracy of the prediction as much as possible. For the M5 series of competitions, it is even more essential to enable the results to be successfully utilized in the development of real enterprises.

1.2 Problem Description and Task Analysis

Nowadays, predictive analytics is playing a more and more critical role in the development of businesses. But the considerable retail system relies on accurate sales forecasts since accurate forecasts provide a high reference value for warehousing, supply chain, cash flow, etc., once the forecast data has a significant deviation, there will be a problem that can greatly interfere with the company's decision-making, resulting in the loss of opportunities and business and other issues. For example, on a specific festival, the sales of products related to cultural customs will increase. Suppose the store does not conduct proper investigations, resulting in deviations in the results of data analysis, then in the inventory allocation stage. In that case, the inventory of unneeded commodities will increase, and the commodities sold in accordance with the season will be in short supply. Such mistakes are detrimental to the retail industry and

can lead to problems such as high transportation costs, chaotic inventory management, and missed opportunities that can significantly reduce store sales.

Therefore, in order to increase Walmart's share in the retail market and improve the brand's business insight, it is not only necessary to mine data value based on appropriate means to predict the future sales volume of a unit of goods but also to improve the accuracy and reduce the error of the forecast as much as possible.

1.3 Research Significance

Based on knowledge reserves such as principles and application of machine learning, it is possible to organize reasonable data in the pre-processing stage and complete appropriate feature extraction. The above knowledge also plays a key role in model selection. And from another perspective, this topic has a practical application. For example, the forecast of commodity sales greatly facilitates the management of supply chain processes because it can help companies better understand customer needs, and then ensure a balance between supply and demand and help companies maximize profits. This advantage can be embodied in all aspects of supply chain delivery, such as forecast quantity, order delivery, return and exchange, and inventory sorting. As a time series problem, this topic can also be viewed as a predictive analysis of certain commodities within a specific period, and the main idea is to plan a more efficient inventory space, reduce operational risks and explore customer buying habits. And now, commodity sales forecast analysis is a critical tool in the business process of many large retail companies such as Walmart.

1.4 Advantages Compared with Other Methods

The decision tree-based machine learning algorithm Light Gradient Decision Machine (LightGBM) is applied in this model, which can directly support categorical features while optimizing the memory footprint, it also helps to omit some steps, such as feature

conversion. Since most machine learning tools are sensitive to numerical values and cannot directly support categorical features, a scaling method needs to be added. Generally speaking, this requires converting categorical features into multi-dimensional 0/1 features. LightGBM optimizes the support for category features, can directly input category features without additional 0/1 expansion and adds decision rules for category features to the decision tree algorithm, which improves the efficiency of time and space utilization.

On the other hand, LightGBM also has a faster training speed, higher efficiency, and a lower memory occupation. The histogram-based algorithm allows continuous feature values to be packed into discrete buckets, which makes the whole training process faster, and the use of discrete bins to store and replace constant values takes up less memory.

Compared to other boosting algorithms, LightGBM achieves higher accuracy by generating more complex trees through the leaf-wise splitting method than the level-wise splitting method. Moreover, its extensive data processing capability and support for parallel learning can be shown to be vastly superior in solving the problem presented in this topic.

Finally, the accuracy of the results predicted using most traditional time series forecasting methods is not up to the desired level. For each such model applied, the model must be manually adjusted to fit the dataset, whereas machine learning methods allow the model to be actively iterated multiple times. Using the LightGBM algorithm in machine learning is a relatively suitable solution here.

Literature review

2.1 Background of the Literature

With the development of new science and technology, accurate forecasts could provide more reference information for people's lives: as consumers, we will make rough predictions based on previous years' promotion dates, specific discount information, and commodity price trends. Then getting an idea about when to spend money can be cost-effective for us because it is easy for us to forecast when they will drop in price with the help of prediction. On the other hand, as a merchant, comparing the historical sales volume can not only provide a reference for their supply chain, but also be of great benefit to their future strategic development. Therefore, an accurate sales forecast plays a vital role in the retail industry.

Looking back at previous research, such as the research of (Vakharia 2002), it can be found that the rapid development of artificial intelligence and various data analysis methods has brought new inspiration to the information field in recent years. It is obvious that it has played a very significant role in data mining and assisting business decision-making. For instance, when it comes to practical problems, much key information will be analysed in different schemes, such as historical sales per unit of time, the reputation and promotion of a particular brand, also some changes in the external environment, etc. According to (Wang et al. [no date]), when solving such time series prediction problems, the above factors can be extracted as features to participate in the model's training.

In the era of big data, whether it is offline retail or e-commerce, the amount of data involved is enormous, and it is still showing an explosive growth trend with the development of the times, then it is reflected in accurate prediction. As a common strategy in the sales supply chain, it solves the operator's problem of balancing supply

and demand. In recent years, many experts and scholars have studied sales forecasting and provided much literature for reference.

2.2 Field and Scope of Investigation

For some time series analysis cases, as far as the field of investigation is concerned, (Maçaira et al. 2018) mentioned in terms of the number of relevant pieces of literature, environmental science, economics, and health are the top three. While for applied models, the top three most popular are regression models, artificial neural networks, and support vector machines. (Chen et al. 2020) indicated that in recent years, since time series forecasting is essentially doing some research and analysis of past data, it is easier to find some trends or regularities on this basis so that it can also be used in other fields such as traffic modelling, solar intensity predictions, and demonstration findings. While most of the surveys on the traditional retail industry are about clothing, general merchandise, electronic products, and automobiles, some studies, such as the research of (Chen and Lu 2017) is based on sales forecasts for information technology or computer products while (Athanasopoulos et al. 2009) focus on locally domestic tourism industry.

2.3 Analytical Methods Mentioned in the Literature

In the collected literature, there are various solutions to different problems, ranging from traditional statistical analysis models to machine learning and artificial intelligence models. Each analysis method has its characteristics, and many scholars begin to explore the comprehensive framework.

(Chen and Lu 2017) not only applied three clustering techniques such as self-organizing map (SOM), growing hierarchical self-organizing map (GHSOM), and K-means, but also combined two machine learning techniques, including support vector regression (SVR) and extreme learning machine (ELM) methods, and then evaluated the effect of

various combination methods. Finally, through testing this method, it could be found that the most suitable model for this retail situation is actually the GHSOM-ELM model.

(Chen et al. 2020) proposed a new framework TADA for the first time, which is used for time series prediction together with dual-attention and multi-task RNNs. They first modelled internal and external features with a multi-task RNN encode and found that the TADA framework has good performance and high accuracy in real-world predictions.

(Di Pillo et al. 2016) emphasized that the premise of prediction is the selection of suitable input attributes for the machine. At the same time, they discussed different situations in the sales prediction for a specific item in a retail store. Since the considered case includes promotional activities, it is found that some methods in machine learning algorithms, which are based on support vector machine (SVM), are more inclined to obtain efficient results in this nonlinear prediction.

(Dong et al. 2019a) pointed out that a supervised regression model is adopted, which mainly tends to combine three machine learning models at the same time, consisting of Linear Regression (LR), Extreme Gradient Boosting (XGBoost) and Light Gradient Boosting Machine (LightGBM). The results show that the comprehensive model performs very well on the premise of extracting both statistical and discrete features.

(Na et al. 2019) used traditional statistical methods which are based on the ARIMA model to calculate the sales volume and geographic location of retail stores, also combined with customer data (such as passenger flow and customer unit price) for application calculations.

(Dong et al. 2019b) utilized Incentive Autoregressive Integrated Moving Average (I-ARIMA), Long Short-Term Memory (LSTM), and Artificial Neural Network (ANN), respectively, according to the characteristics of different datasets in the e-commerce

field. After testing, they pointed out that LSTM can play a significant role in solving the problem of gradient dispersion and gradient explosion in the Recurrent Neural Network (RNN).

(Lingxian et al. 2019) focused on a comprehensive framework combining K-Means and Long Short-Term Memory (LSTM) models. They indicated that since LSTM analyses the information of the previous period to carry out the information of the next period, it can be seen as a very suitable method for the prediction of time series problems.

(Makridakis et al. 2022) analysed the M5 competition itself, and it is found that most of the winning works are based on LightGBM. Moreover, in the comparison of results, the superiority of the machine learning method is directly highlighted.

(Nasios and Vogklis 2022) revealed that in order to solve the problem of point and probability, a comprehensive framework is proposed, and the core idea of which is to mix gradient boosting trees and neural network algorithms, specifically Keras_nas - Keras MLP model and fastai_cos - FastAI MLP model. At the same time, they proposed an opinion that choosing a representative validation set plays a vital role in the prediction of the entire dataset.

According to (Nenova and May 2016), for the sake of saving time overhead, a new two-level skewed linear discriminant tree model was first proposed, which determines the best hierarchical prediction technique for a given hierarchical database without costing a long time.

(Carbonneau et al. 2008) argued that recurrent neural networks (RNNs), support vector machines (SVMs), and neural networks (NNs) generally perform better in applications to reduce distorted demand at the end of the supply chain, compared to simpler algorithms such as moving average, naive, and trend methods, which represents the

critical position of advanced artificial intelligence technology in the application of factory data.

(Ismail Fawaz et al. 2019) found the lack of previous research in this regard and presented the most detailed study of DNNs for the first time, they divided DNNs into generative models and discriminative models. Finally, the results they got are achieved in the field of end-to-end time series forecasting through fully dialogue neural networks and deep residual networks.

2.4 Ways to Improve Robustness and Improve System Performance

(Bandara et al. 2020) provides an improvement scheme for the basic LSTM model, which introduces a clustering variant. This prediction framework first determines a series of subgroups by temporal clustering techniques, builds models for them separately on this basis, and uses LSTM Network and various clustering algorithms such as K-Means, DBScan, Partition Around Medoids (PAM) and Snob to evaluate the stability of the framework. As a result, the translation brought by this time series clustering method can significantly reduce the computation time of the base model. (Ma and Fildes 2022) pointed out that in the M5 time series prediction competition, the top teams basically adopt the global top-down analysis direction. They believe that when forecasting hierarchical time series, if these three techniques, optimal reconciliation (ORC), global bottom-up approach (GBU), and global middle-out (GMO) are used to generate forecast results, it is possible to combine the two advantages such as overall high performance of GMO and local high performance of GBU method. (Nenova and May 2016) combined the characteristics of bottom-up (BU), top-down using average historical proportions (TDGSA) and top-down using the proportions of historical averages (TDGSF), top-down using forecasted proportions (TDFP), which provide support for the robustness of the model.

2.5 Limitations of the Studies Covered in the Literature

The above is an analysis of some data forecasting methods used in the works of literature. Although they deal with the time series problem from various perspectives, there are still some shortcomings that still need to be overcome through now. The following will discuss the limitations of the existing literature from several aspects.

Basically, the data used for training in the literatures discussed above are usually based on a relatively sufficient time to accumulate. According to (Dong et al. 2019b), if there is only a dataset with a scarce timeline, which means that less historical time series data could be relied on, then how should these data be put into the training process to establish a suitable analysis model?

(Bandara et al. 2020) mentioned that, when using the clustering algorithm, since some data may be ignored after grouping the time series, it might cause some information loss. In this case, the final performance of the model will be affected by the number of time series groups. If we want to achieve this optimal effect, the critical point is to make full use of these groups and achieve an average impact by measuring the value of each time series grouping scheme. In this process, a challenging problem is how to determine the best number of time series groups, what needs to be done to avoid any information loss on the basis of maximizing the utilization of cross-information?

As in the study by (Ismail Fawaz et al. 2019), on a specific dataset, different models and methods can work and be tested for their prediction accuracy, while multiple evaluation methods and indicators can be applied to evaluate the overall performance. Nonetheless, time series problems are not the same as computer vision and natural language processing tasks in the application of deep neural networks. For example, in the data pre-processing stage, it lacks in-depth research on data augmentation and transfer learning. So, two questions are raised here, that is, how to overcome the limitations of deep learning in time series forecasting? And what impact can normalization methods have on the performance of deep neural networks?

In other studies on sales forecasting like one by and (Nenova and May 2016), the researchers only considered part of the factors to build the model, which causes the data containing other information was not fully utilized, these variables also implied some potential information of consumers' preferences for specific products. Therefore, another question is that if it is possible to efficiently train a model based on all possible features to achieve accurate results. Furthermore, considering the breadth of datasets, most studies still lack extensive surveys of real-world datasets. Suppose there is a lack of explanations for the variance predictors of positive and negative correlation groups, then in practical applications. In that case, the versatility of some models will be significantly reduced, such as 2S-LDA.

Finally, (Mancuso et al. 2021) pointed out that possibly due to the size of the dataset, the complexity of the underlying hierarchy, and other constraints such as non-negative prediction and additional computational cost, depending on the performance of different methods, they have the potential to be used to make predictions separately at different aggregation levels. But how to solve this problem remains to be explored.

2.6 Future Directions Mentioned in the Literature

The limitations mentioned above that have not been realized actually point to the research direction that can be referred to in the future at a certain level. On the one hand, the following analysis explores possible solutions based on these limitations. On the other hand, they can express the prospects for future technological development on this topic.

For traditional statistical models, there are two directions for future research: as mentioned by (Maçaira et al. 2018), researchers have successfully used the ARIMA model in time series prediction problems, and the next step can be some research to de-integrate a variety of business indicators, such as profit, stores, commodity attributes,

etc., which can be extracted into features. Another feasible research direction, which is also the current trend of time series forecasting, is to integrate statistical models and machine learning techniques to analyse data, including the comprehensive application of models such as clustering, classification, and neural networks.

For machine learning and neural network technology, in the process of feature selection, some methods can be developed in the future to accurately find more features containing adequate information, especially based on consumers' behaviour preferences and personal information to build efficient models. In addition, refer to the work of (Ferreira et al. 2018) , in terms of parameter selection, how to choose parameters and what kind of parameter adjustment strategies can help researchers make more accurate predictions. On the other hand, as mentioned by (Lingxian et al. 2019), in the time series forecasting problem involving geographical factors, a more considerable amount of data can be used to consider the regional location in the future, so that the trained model can be widely promoted. While (Ma and Fildes 2022) believed that when testing the model's performance, the selection of test sets can be improved in the future, such as by using multiple rolling test sets for model evaluation.

In addition, future research directions can also be based on finding suitable aggregation methods, the models used by (Nenova and May 2016) are trained only from data that exhibit consistent aggregation preferences, and hierarchical datasets that show inconsistent aggregation preferences can be studied in the future. Also, when the data set used contains data with more levels, and more complex structure, a dataset with more than two levels can be considered, so that the finally returned data also has multiple levels. Finally, when calculating the actual misclassification cost of the chosen aggregation method, an entire aggregation misclassification analysis can be performed on it during the model-building process, which is also one of the research directions that can be considered in the future.

Theoretical Bases and Hypothesis

3.1 Principles of the LightGBM Algorithm

As a widely used machine learning algorithm, gradient boosting decision tree (GBDT) is superior in efficiency, accuracy, and interpretability. LightGBM is a framework for implementing this algorithm. In addition to having the characteristics of GBDT's good training effect, less overfitting, high accuracy, and interpretability, it can also solve the problem of GBDT on large data sets, which is about its performance is not satisfactory due to the large amount of data and features. In this paper, the principles of LightGBM are mainly explained in four aspects: decision tree algorithm based on histogram, leaf-wise algorithm with depth restriction, Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB).

- The essence of the histogram-based algorithm is histogram statistics. By assigning bins to features, large-scale data is placed in a histogram. Based on the histogram algorithm, LightGBM is further optimized. Here is the process (Ke et al. 2017) described for this algorithm.

Algorithm 1: Histogram-based Algorithm

```
Input: I: training data, d: max depth
Input: m: feature dimension
nodeSet ← {0} ▷ tree nodes in current level
rowSet ← {{0, 1, 2, ...}} ▷ data indices in tree nodes
for i = 1 to d do
    for node in nodeSet do
        usedRows ← rowSet[node]
        for k = 1 to m do
            H ← new Histogram()
            ▷ Build histogram
            for j in usedRows do
                bin ← I.f[k][j].bin
                H[bin].y ← H[bin].y + I.y[j]
                H[bin].n ← H[bin].n + 1
            Find the best split on histogram H.
            ...
    Update rowSet and nodeSet according to the best
    split points.
    ...
```

Figure 3-1. Histogram-based Algorithm

- In addition, most GBDT tools will use a level-wise decision tree growth strategy, but in the LightGBM framework, a leaf-wise algorithm with a depth limit is used, which will find the one with the most significant splitting gain among all the leaves, then split, and so on. What is more, leaf-wise decision tree can reduce more errors and get better accuracy with the same number of splits. However, it may grow a deeper decision tree, there might be an overfitting. Therefore, LightGBM adds a maximum depth limit on top of the leaf-wise decision tree to prevent overfitting while maintaining high efficiency.

- For the principles of Gradient-based One-Side Sampling (GOSS), (Ke et al. 2017) pointed out that, "GOSS keeps all the instances with large gradients and performs random sampling on the instances with small gradients. In order to compensate for the influence on the data distribution, when computing the information gain, GOSS introduces a constant multiplier for the data instances with small gradients. Specifically, GOSS first sorts the data instances according to the absolute value of their gradients and selects the top a × 100% instances. Then it randomly samples b × 100% instances from the rest of the data. After that, GOSS amplifies the sampled data with small gradients by a constant 1−a when calculating the information gain. By doing so, we put more focus on the under-trained b instances without changing the original data distribution by much." Here is the relevant algorithm they proposed.

Algorithm 2: Gradient-based One-Side Sampling

```
Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: loss: loss function, L: weak learner
models ← {}, fact ← (1−a)/b
topN ← a × len(I) , randN ← b × len(I)
for i = 1 to d do
    preds ← models.predict(I)
    g ← loss(I, preds), w ← {1,1,...}
    sorted ← GetSortedIndices(abs(g))
    topSet ← sorted[1:topN]
    randSet ← RandomPick(sorted[topN:len(I)],
    randN)
    usedSet ← topSet + randSet
    w[randSet] × = fact ▷ Assign weight fact to the
    small gradient data.
    newModel ← L(I[usedSet], − g[usedSet],
    w[usedSet])
    models.append(newModel)
```

Figure 3-2. Gradient-based One-Side Sampling

- For the principles of Exclusive Feature Bundling (EFB), (Ke et al. 2017) also explained that, "High-dimensional data are usually very sparse. The sparsity of the feature space provides us with a possibility of designing a nearly lossless approach to reduce the number of features. Specifically, in a sparse feature space, many features are mutually exclusive, i.e., they never take nonzero values simultaneously. We can safely bundle exclusive features into a single feature (which we call an exclusive feature bundle). By a carefully designed feature scanning algorithm, we can build the same feature histograms from the feature bundles as those from individual features. In this way, the complexity of histogram building changes from O(#data × #feature) to O(#data × #bundle), while #bundle << #feature. Then we can significantly speed up the training of GBDT without hurting the accuracy." The following two algorithms are used to implement EFB specifically：

Algorithm 3: Greedy Bundling

```
Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← {}, bundlesConflict ← {}
for i in searchOrder do
    needNew ← True
    for j = 1 to len(bundles) do
        cnt ← ConflictCnt(bundles[j],F[i])
        if cnt + bundlesConflict[i] ≤ K then
            bundles[j].add(F[i]), needNew ← False
            break
    if needNew then
        Add F[i] as a new bundle to bundles
Output: bundles
```

Algorithm 4: Merge Exclusive Features

```
Input: numData: number of data
Input: F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
for f in F do
    totalBin += f.numBin
    binRanges.append(totalBin)
newBin ← new  Bin(numData)
for i = 1 to numData do
    newBin[i] ← 0
    for j = 1 to len(F) do
        if F[j].bin[i] ≠ 0 then
            newBin[i] ← F[j].bin[i] + binRanges[j]
Output: newBin, binRanges
```

Figure 3-3. Methods for Implementing Exclusive Feature Bundling

3.2 Hypothesis

The aim of this project is to provide a solution that can be used to generate an accurate forecasting model for Wal-Mart supermarkets. In this case, the prediction is for the sales of 30,490 items in a particular time series, and the prediction is mainly based on historical sales data. And the forecasts are implemented based on machine learning algorithms, while testing how different parameter choices will affect the output. Although there are difficulties in ensuring the accuracy of the point predictions, it is necessary to both ensure that the RMSE of the prediction results is within an acceptable range and to reduce the error values by adjusting the model and continuously testing its performance in real-world evaluations, and finally, to obtain the RMSSE generated for it by the Kaggle platform.

It is assumed that the sales volume of each product in a Wal-Mart shop will show a certain pattern as time progresses and that the change in sales volume of a single item can be summarised as a combination of the above factors due to the inclusion of external variables such as special events as well as price factors. This pattern can be discovered by training the model and can work well in the application. In addition to this, as the model is built, a validity conjecture can be confirmed, i.e., the model performs well in

predicting the accuracy of the model's predictions when forecasting the sales volume of an item for each day in the next 28 days.

Methodology

4.1 Data Sources

The datasets used are all from the relevant data records of Walmart stores published by Kaggle in the m5-accuracy competition. It contains four CSV files related to historical data and one file for submitting results.

4.2 Data Description

The dataset includes sales statistics for Walmart stores in three US states.

- The file calendar.csv contains some information related to the date from day 1 to day 1969. And specific variables are included in this file to indicate some helpful information: year, month, weekdays or weekends, festival information which includes sports events, major national or ethnic events, and festivals (two festival description variables are involved, event_1 and event_2), and the distribution of whether snaps are supported on the day in three states.

- The file sales_train_evaluation.csv contains the name of the 30490 products whose last segment of the product name is marked as evaluation, the store, and the region it belongs to, the category and department of the product, and all sales records from day 1 to day 1941, which mainly to provide labels for the predictions from day 1913 to day 1941. According to the official documentation, most of the sales data in the more than 42,840 time series are zero values, which can be regarded as sporadic sales. At the same time, the file sales_train_validation.csv contains all sales records from day 1 to day 1913 marked as validation in the last segment of the product name. Except for the different time lengths of the data, the rest are consistent with the evaluation file. For example, the data set divides commodity categories into three categories: 'HOBBIES', 'FOODS', and 'HOUSEHOLD'.

However, because of this commodity data without specific external parameters, information about consumer preferences, such as the brand and word-of-mouth of the commodity cannot be obtained.

- The file 'sell_prices.csv' mainly has four aspects of information: the specific information of the store (such as 'CA_1'), the id of the product, the date information, and the price corresponding to the date. The selling price is marked according to the week, that is, the selling price of the commodity recorded within a week is unchanged.

4.3 Sample Selection

The parts of the dataset that are used for training: datasets in 'sell_prices.csv', 'sales_train_validation.csv', and 'calendar.csv'. At the time of submission, LightGBM makes predictions based on the sales volume data in the 'sales_train_validation.csv' dataset. Here the subject of the problem is the forecast based on sales volume. Also, considering the impact of festival factors and price fluctuations on sales, historical sales volume, date, and selling price are used as the primary reference. In addition, there are explanatory variables that can be trained to greatly improve the prediction accuracy, such as promotions, day of the week, and a specific holiday or event. After the model is built, it is possible to intuitively understand which factor can play a key enlightening role in prediction by outputting the training importance ranking of features.

4.4 Way of Solving the Problem

In order to predict the sales volume of each product and reduce the RMSSE value as much as possible under the premise of ensuring low RMSE, two sets of parameter values applicable to the model LightGBM are tested here, and the difference in final accuracy between the two is compared.

4.5 How to Generate Output

After the necessary pre-processing, the data can then be used as input to train the appropriate model, after which the final model is applied to the data in the test set to check the accuracy of the method. The RMSE, the more commonly used error measure, will also be used in this study to evaluate the performance of the overall model.

The form of the output data has a fixed standard based on the dataset provided by Kaggle.

Submission.csv provides a standard format for submission, including predictions for 30,490 evaluations and 30,490 validation data, the main difference being the different timelines. The forecasts generated are for daily sales of each item over a 28-day period, with the sample format shown below.

id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HOBBIES_1_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-1. Sample Format of Submission File

4.6 How to Prove the Correctness

For each product sales volume, there is a final predicted value, which could be compared with the actual value and evaluated specifically through the metric. Here the Root Mean Square Error (RMSE) of the test is used as a criterion to evaluate the

performance of the model. For each testing period, an RMSE is generated and referred by the next training round to adjust the model and achieve a smaller error continually. In general, an RMSE of around two could be excellent, and an RMSE of less than three is considered to be of good performance. While comparing the impact of two different sets of parameters on the accuracy of the same framework, this report proposes that a model with an RMSE of 2.5 or less is considered as a usable model, i.e., one that satisfies the above hypothesis that customer preferences and product sales are regular over the timeline and that future sales can be predicted by the model with a high degree of accuracy.

Implementation

5.1 Introduce Associated Libraries

First, introduce the library to be used:

```
In [1]:
import pandas as pd
import numpy as np
import dask.dataframe as dd
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
import lightgbm as lgb
from sklearn import preprocessing, metrics
import gc
import joblib
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

Where numpy is used for the linear algebra, pandas is used for data pre-processing as well as loading and outputting data, and saving it as a CSV file, while sklearn is used for processing the data and matplotlib.pyplot is related to plotting.

The whole program runs on a Kaggle notebook, as the platform Kaggle not only makes it easy to start up an accelerator such as a GPU, but also makes the composition of the data on the input and output side very clear. Moreover, by submitting the predictions directly on the Kaggle website, the prediction scores can be obtained quickly, which not only facilitates the timely adjustment of the model, but also gives an accurate official assessment of the final model.

5.2 Load Data Sets and Reduce Memory Usage

In [4]:

```
# The files used

def read_data():
    sell_prices_df = pd.read_csv(INPUT_DIR_PATH + 'sell_prices.csv')
    sell_prices_df = reduce_mem_usage(sell_prices_df)
    print('Sell prices has {} rows and {} columns'.format(sell_prices_df.shape[0], sell_prices_df.shape[1]))

    calendar_df = pd.read_csv(INPUT_DIR_PATH + 'calendar.csv')
    calendar_df = reduce_mem_usage(calendar_df)
    print('Calendar has {} rows and {} columns'.format(calendar_df.shape[0], calendar_df.shape[1]))

    sales_train_validation_df = pd.read_csv(INPUT_DIR_PATH + 'sales_train_validation.csv')
    print('Sales train validation has {} rows and {} columns'.format(sales_train_validation_df.shape[0], sales_train_va
lidation_df.shape[1]))

    submission_df = pd.read_csv(INPUT_DIR_PATH + 'sample_submission.csv')
    return sell_prices_df, calendar_df, sales_train_validation_df, submission_df
```

To start with, the path to the folder where each dataset is located is read in so that it can be modified.

In order to avoid this phenomenon, it is necessary to process the data: one is to discard some data that has relatively little value, for example, according to the observation of the dataset, there are actually a lot of dates with zero sales. Moreover, for a total of 1913 days of training time series, the first 200 days are of relatively low reference value, so after importing the data set, the variable ‘startDay’ is used to determine from which day the data is actually applied.

The second is to design functions to reduce memory usage, combined with the rubbish collection and byte change methods provided by Python to minimize the risk of crashes. For instance, memory space usage can be saved by converting 64-bit floating point numbers to 32-bit. By calling the ‘downcast_dtypes()’ function, the memory footprint of the sell prices file, which has an enormous amount of data, has been reduced by 37.5%, successfully to 130.48Mb. The calendar file has reduced memory usage by 41.9% and now accounts for approximately 0.12Mb.

```
Mem. usage decreased to 130.48 Mb (37.5% reduction)
Sell prices has 6841121 rows and 4 columns
Mem. usage decreased to  0.12 Mb (41.9% reduction)
Calendar has 1969 rows and 14 columns
Sales train validation has 30490 rows and 1919 columns
```

5.3 Encode the Category

In order to avoid errors in the data encoding conversion, the 'NaN' data needs to be processed first, preserved, and supplemented with default values. The original data here includes 'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2' in the calendar dataset, 'item_id', 'dept_id', 'cat_id', ' warehouse_id', 'state_id' in the sales dataset and 'item_id', 'warehouse_id' in the price dataset.

In [6]:

```
# Convert features into continuous codes

def encode_categorical(df, cols):
    for col in cols:
        # Leave NaN as it is.
        le = preprocessing.LabelEncoder()
        not_null = df[col][df[col].notnull()]
        df[col] = pd.Series(le.fit_transform(not_null), index=not_null.index)

    return df
```

5.4 Melt and Merge

In [7]:

```
# Function to read the data and merge it
# Reshape data, transform product id data into product id-date data

def melt_and_merge(calendar, sell_prices, sales_train_validation, submission, nrows = 55000000, merge = False):

    # melt sales data, get it ready for training

    sales_train_validation = pd.melt(sales_train_validation, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id'
, 'state_id'], var_name = 'day', value_name = 'demand')
    print('Melted sales train validation has {} rows and {} columns'.format(sales_train_validation.shape[0], sales_trai
n_validation.shape[1]))
    sales_train_validation = reduce_mem_usage(sales_train_validation)

    sales_train_validation = sales_train_validation.iloc[-nrows:,:]

    # seperate test dataframes

    test1_rows = [row for row in submission['id'] if 'validation' in row]
    test2_rows = [row for row in submission['id'] if 'evaluation' in row]
    test1 = submission[submission['id'].isin(test1_rows)]
    test2 = submission[submission['id'].isin(test2_rows)]

    # change column names
```

```
    test1.columns = ['id', 'd_1914', 'd_1915', 'd_1916', 'd_1917', 'd_1918', 'd_1919', 'd_1920', 'd_1921', 'd_1922', 'd_
1923', 'd_1924', 'd_1925', 'd_1926', 'd_1927', 'd_1928', 'd_1929', 'd_1930', 'd_1931',
                       'd_1932', 'd_1933', 'd_1934', 'd_1935', 'd_1936', 'd_1937', 'd_1938', 'd_1939', 'd_1940', 'd_1941'
]
    test2.columns = ['id', 'd_1942', 'd_1943', 'd_1944', 'd_1945', 'd_1946', 'd_1947', 'd_1948', 'd_1949', 'd_1950', 'd_
1951', 'd_1952', 'd_1953', 'd_1954', 'd_1955', 'd_1956', 'd_1957', 'd_1958', 'd_1959',
                       'd_1960', 'd_1961', 'd_1962', 'd_1963', 'd_1964', 'd_1965', 'd_1966', 'd_1967', 'd_1968', 'd_1969'
]

    # get product table

    product = sales_train_validation[['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id']].drop_duplicates()

    # merge with product table

    product['id'] = product['id'].str.replace('_evaluation','_validation')
    test1 = test1.merge(product, how = 'left', on = 'id')
    product['id'] = product['id'].str.replace('_validation','_evaluation')
    test2 = test2.merge(product, how = 'left', on = 'id')

    test1 = pd.melt(test1, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id'], var_name = 'day', v
alue_name = 'demand')
    test2 = pd.melt(test2, id_vars = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id'], var_name = 'day', v
alue_name = 'demand')
```

```
    sales_train_validation['part'] = 'train'
    test1['part'] = 'test1'
    test2['part'] = 'test2'

    data = pd.concat([sales_train_validation, test1, test2], axis = 0)

    del sales_train_validation, test1, test2

    print(data.shape)

    calendar.drop(['weekday', 'wday', 'month', 'year'], inplace = True, axis = 1)

    # delete test2 for now

    data = data[data['part'] != 'test2']

    if merge:
        data = pd.merge(data, calendar, how = 'left', left_on = ['day'], right_on = ['d'])
        data.drop(['d', 'day'], inplace = True, axis = 1)
        # get the sell price data (this feature should be very important)
        data = data.merge(sell_prices, on = ['store_id', 'item_id', 'wm_yr_wk'], how = 'left')
        print('Our final dataset to train has {} rows and {} columns'.format(data.shape[0], data.shape[1]))
    else:
        pass

    gc.collect()

    return data
```

Then go to merge the multiple data sets into one data frame. After importing the data, it can be seen that the formal composition of the data regarding sales is such that the first six columns of the table are respectively: the identifier of the data 'id', the identifier of the item “item_id”, the department to which the item belongs “dept_id”, the broad category to which the item belongs “cat_id”, the identifier of the shop “store_id” and the identifier of the state in which it is located “state_id”. And a separate column for each day of sales corresponds. In this step, the data is reshaped to turn the wide data into long data and to reduce the memory footprint by discarding some rows with less reference value. The data grid for days 1914-1941, which will be used for forecasting, is merged with the data grid for the first 1913 days contained in the sales_train_validation file, and the calendar as well as price information is also reshaped into the data frame, with the data table header for the last 28 days used for forecasting coming from the sample_submission. CSV file. After this step, the final dataset to train has 28353720 rows and 18 columns.

5.5 Feature Engineering

This section is responsible for the extraction of features and is the part that plays a major role in the overall forecast. The second part is price information, which is also of considerable reference value for forecasting purposes. The third part is time-related features, such as determining whether the day is a weekend, which is chosen mainly because of the assumption that weekends have a relatively high reference value for people's shopping trends. In the end, 25 features were selected for each of the data:

For original features, there are ‘item_id’, ‘dept_id’, ‘cat_id’, ‘store_id’, ‘state_id’, ‘event_name_1’, ‘event_type_1’, ‘snap_CA’, ‘snap_TX’, ‘snap_WI’, ‘sell_price’. The event-related features represent the labels of the holidays. The main reason for omitting the extraction of ‘event_2’-related features and focusing only on ‘event_1’-related content is that ‘event_2’ occurs very infrequently in the data, i.e., there are rarely two

holidays at the same time on a given day, so discarding the extraction of this feature can improve memory utilisation and thus achieve high efficiency.

And for demand features, there are ‘shift_t28’, ‘rolling_std_t7’, ‘rolling_std_t30’, ‘rolling_std_t90’, ‘rolling_std_t180’, ‘rolling_mean_t7’, ‘rolling_mean_t30’, ‘rolling_mean_t60’. The demand data involved here, i.e., historical sales data, is an important reference for training. In this case, some lagged features are extracted, e.g., sliding with a window of 28 and rolling features with periods of 7, 30, 90, and 180 days respectively, including standard deviation, mean, skew, and Kurt calculation.

Also, for price features like ‘price_change_t1’, ‘price_change_t365’, ‘rolling_price_std_t7’, the single-day price change, annual price change and 7 day standard deviation are exploited in the training.

Finally, a series of date features are generated, consisting of ‘year’, ‘month’, ‘dayofweek’.

```
# A simple feature engineering

def simple_fe(data):

    # demand features

    for val in [28, 29, 30]:
        data[f"shift_t{val}"] = data.groupby(["id"])["demand"].transform(lambda x: x.shift(val))
    for val in [7, 30, 60, 90, 180]:
        data[f"rolling_std_t{val}"] = data.groupby(["id"])["demand"].transform(lambda x: x.shift(28).rolling(val).std())
    for val in [7, 30, 60, 90, 180]:
        data[f"rolling_mean_t{val}"] = data.groupby(["id"])["demand"].transform(lambda x: x.shift(28).rolling(val).mean(
))

    data["rolling_skew_t30"] = data.groupby(["id"])["demand"].transform( lambda x: x.shift(28).rolling(30).skew())
    data["rolling_kurt_t30"] = data.groupby(["id"])["demand"].transform(lambda x: x.shift(28).rolling(30).kurt())

    # price features
    data['lag_price_t1'] = data.groupby(['id'])['sell_price'].transform(lambda x: x.shift(1))
    data['price_change_t1'] = (data['lag_price_t1'] - data['sell_price']) / (data['lag_price_t1'])
    data['rolling_price_max_t365'] = data.groupby(['id'])['sell_price'].transform(lambda x: x.shift(1).rolling(365).max(
))
    data['price_change_t365'] = (data['rolling_price_max_t365'] - data['sell_price']) / (data['rolling_price_max_t365'])
    data['rolling_price_std_t7'] = data.groupby(['id'])['sell_price'].transform(lambda x: x.rolling(7).std())
    data['rolling_price_std_t30'] = data.groupby(['id'])['sell_price'].transform(lambda x: x.rolling(30).std())
    data.drop(['rolling_price_max_t365', 'lag_price_t1'], inplace = True, axis = 1)
```

```
# time features
data['date'] = pd.to_datetime(data['date'])
attrs = ["year", "quarter", "month", "week", "day", "dayofweek", "is_year_end", "is_year_start", "is_quarter_end", \
    "is_quarter_start", "is_month_end","is_month_start",
]

for attr in attrs:
    dtype = np.int16 if attr == "year" else np.int8
    data[attr] = getattr(data['date'].dt, attr).astype(dtype)
data["is_weekend"] = data["dayofweek"].isin([5, 6]).astype(np.int8)

return data
```

5.6 Split the Dataset

Generally tabular data is divided into training set, validation set and test set by a random division approach. But for time-series data. This can lead to "time traversal", which is essentially a data leakage and can lead to serious overfitting problems. The training/validation/testing set must therefore be divided according to chronological order.

```
In [11]:
def run_lgb(data):

    # going to evaluate with the last 28 days

    # day 1 - day 1885 as training set
    x_train = data[data['date'] <= '2016-03-27']
    y_train = x_train['demand']
    # day 1886 - day 1913 as validation set
    x_val = data[(data['date'] > '2016-03-27') & (data['date'] <= '2016-04-24')]
    y_val = x_val['demand']
    # day 1914 - day 1941 as test set
    test = data[(data['date'] > '2016-04-24')]
    del data
    gc.collect()
```

The training set is the data from 2011-01-29 to 2016-03-27 (1885 days), the validation set is the data from 2016-03-28 to 2016-04-24 (28 days), and the test set is the time series from 2016-04-24 to 2016-05-22.

5.7 Evaluation Method

In the process of evaluating the results using the point forecast method, the forecast period h is 28 days, that is, a period of 4 weeks. Since the sales volume of the products

displayed in the dataset is intermittent and involves many 0 values, the point prediction here is based on the squared error to accurately predict the average demand. However, before uploading to Kaggle to get the final score, to obtain a preliminary judgment based on the prediction results, the Root Mean Square Error (RMSE) is introduced as a raw score which is related to the accuracy, that is, the sum of squares of the errors divided by the square root of the number of samples minus one. In broad terms, RMSE is defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_t - \widehat{Y}_t\right)^2}{n}},$$

where Y_t is the actual future value of the examined time series at point t, $\widehat{Y}_t$ the generated forecast, n the length of the training sample (number of historical observations).

When the model is adjusted so that its generated RMSE is within an ideal range, the result can be uploaded to the platform provided by Kaggle for accuracy checking. The official evaluation method is Root Mean Squared Scaled Error (RMSSE), a variant based on Mean Absolute Scaled Error (MASE). The measure is calculated for each series as follows:

$$\mathrm{RMSSE} = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}\left(Y_t - \widehat{Y}_t\right)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(Y_t - Y_{t-1})^2}},$$

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

Since the most crucial purpose of the entire process is high accuracy, the target of this forecast is the sales volume of 30,490 items from the 1913th day to the 1941st day, and the public score provided by Kaggle evaluates the final model. It is essential that the value of this score is required to be as low as possible.

5.8 Model Building

As mentioned above, due to the superior performance of LightGBM, it is primarily considered as a basic framework to implement iterative training using a weak classifier (decision tree) to obtain the optimal model. In order to compare the impact of different parameter choices, different objective loss functions are mainly set here. By observing and comparing their performance differences, a final judgement is made to obtain a model with higher prediction accuracy.

5.8.1 'Regression' as the Objective Loss Function

Here using 'boosting_type': 'gbdt', specifying that the type of the weak learner is a gradient boosting tree 'gbdt'. And according to the description, this time series prediction can be classified as a regression problem. LightGBM offers regression as an objective function, and other parameters can be adjusted as follows. In particular, during the training process the message "[LightGBM] [Warning] No further splits with positive gain, best gain: -inf" might appear, which means that because the tree in the current iteration cannot be split any further, the learning should be stopped, and this kind of problem like improper parameters can be solved by changing the value of 'min_data_in_leaf'.

```
params = {
    'boosting_type': 'gbdt',
    'metric': 'rmse',
    'objective': 'regression',
    'seed': 341,
    'learning_rate': 0.07,
    'lambda': 0.1,
    'num_leaves': 64,
    'min_data_in_leaf': 50,
    'max_depth': 5,
    'bagging_fraction': 0.75,
    'bagging_freq' : 1,
    'colsample_bytree': 0.75,
    'verbose': -1,
    'force_col_wise' : 'true'
}
```

5.8.2 'Poisson' as the Objective Loss Function

To be more specific about the type of regression prediction, the Poisson distribution is guessed to be more in line with the distribution of the original data, because the dependent variable in the problem is a large amount of zero-inflated data, which is essentially a regression scheme, so the target loss function is set to Poisson loss. The following is a calculation of the Poisson distribution using a tree-based model and, correspondingly, modifying some of the corresponding parameters to help the model achieve a better training result. To avoid the warning that memory is not enough, here set 'force_col_wise = true'.

```
params = {
    'boosting_type': 'gbdt',
    'metric': 'rmse',
    'objective': 'poisson',
    'n_jobs': -1,
    'seed': 20,
    'learning_rate': 0.1,
    'alpha': 0.1,
    'lambda': 0.1,
    'bagging_fraction': 0.66,
    'bagging_freq': 2,
    'colsample_bytree': 0.77,
    'force_col_wise':'true'}
```

5.9 Design Document and Flowchart

The project is completed in the python3-based Jupyter notebook editor provided by Kaggle. The Sklearn library is deployed for data processing and the boosting integrated model LightGBM, developed by Microsoft, was introduced for training the model.

The flow chart for this project is as follows.

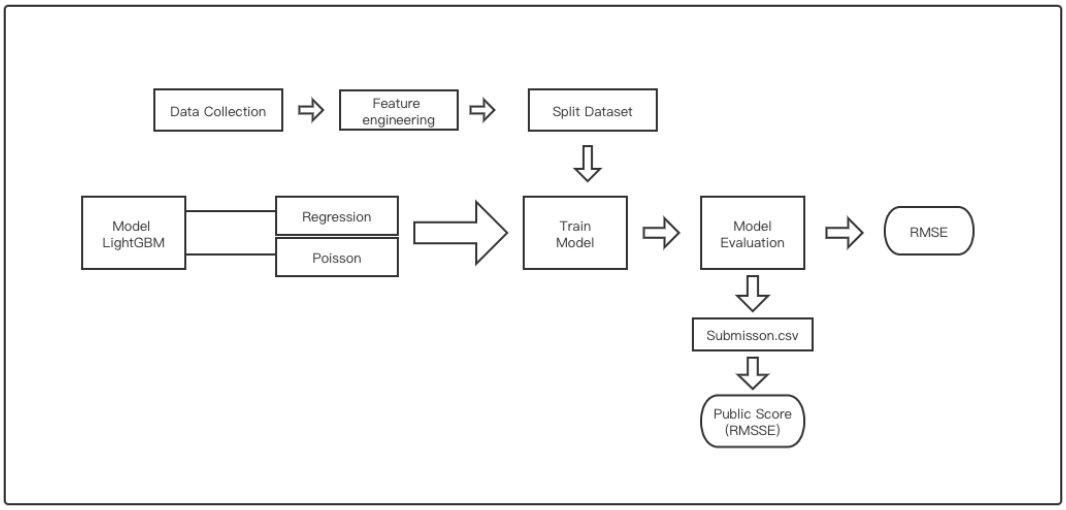


Figure 5-1. Flowchart

Data Analysis and Discussion

6.1 Generation of Outputs

To relieve CPU pressure, it is necessary to temporarily delete unneeded data that has already been used in a timely manner. Otherwise, it will easily cause memory overflow.

The performance of the model is first evaluated on the validation set, and it shows that after 2000 rounds of boosting iterations, good RMSE error values can be obtained. After applying the model to the test set for prediction, the predicted sales for each product for the days 1914-1941 are obtained. At this point, the data is populated according to the empty submission table created earlier, and a separate submission file is generated in the format required by the competition question.

The part of the submission file generated by the model with 'Regression' Objective Loss Function is as follows:

id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
HOBBIES_1_	0.86172947	0.7124552	0.686137	0.66494074	0.7038802	0.8356083	0.89407107	0.58891156	0.80033324	0.83775977	0.80961846	0.93521184	1.26031695	1.23651048	0.9670036	0.80463442	0.73476421	0.73706765
HOBBIES_1_	0.38439607	0.32092846	0.32092846	0.30706768	0.32475903	0.4158385	0.40729005	0.20814708	0.17517199	0.17976794	0.09484031	0.21756809	0.32245201	0.2698465	0.18378026	0.15979539	0.13939494	0.13939494
HOBBIES_1_	0.45140235	0.41467086	0.41467086	0.41467086	0.46535065	0.54123543	0.64666862	0.37018845	0.32966772	0.31628695	0.25337398	0.37365695	0.495482	0.41020406	0.32757658	0.30009029	0.31649655	0.40635737
HOBBIES_1_	1.92059159	1.66481421	1.47356648	1.43087107	1.7241997	1.95308064	1.92791307	1.49323568	1.55647568	1.45361641	1.29130402	1.60923315	2.36716201	2.32610984	1.95654242	1.5758532	1.75611643	1.67857531
HOBBIES_1_	0.82949184	0.72458296	0.89712492	1.05140294	1.01650366	1.30900376	1.76584493	1.11480333	1.14303505	0.97681076	0.80413313	0.97938539	1.250768	1.14689802	1.00661325	0.88905872	0.8406763	0.8406763
HOBBIES_1_	1.26434084	1.19461534	1.4083346	1.17529677	1.21939276	1.37119428	1.28844082	1.01212847	0.8815727	1.04528267	0.8238681	1.13879748	1.31196864	1.21357122	1.01242315	1.34618029	1.07692981	1.07692981
HOBBIES_1_	0.3231595	0.40436051	0.32378974	0.28357857	0.42931574	0.50972471	0.52370591	0.32196512	0.24497406	0.25421927	0.15914886	0.26445608	0.48379449	0.44145743	0.30045183	0.31794039	0.28159791	0.28159791
HOBBIES_1_	5.75189511	6.53114056	5.99400948	5.80704119	7.27172598	11.7071511	11.5340146	8.82715904	8.13558279	8.41132357	8.97758474	10.3862258	11.6950055	8.99871559	8.9849201	7.71652268	8.13075602	7.18843678
HOBBIES_1_	0.32272884	0.34722677	0.29069429	0.35440166	0.49777908	0.54814055	0.98079872	0.64681476	0.4537785	0.44584499	0.28110721	0.38347202	0.61462507	0.49038212	0.3711287	0.99922346	0.626633	0.626633
HOBBIES_1_	0.31644437	0.34516439	0.36179225	0.41700494	0.52559575	0.62640479	0.93560371	0.80411383	0.78556243	0.74393292	0.56072909	0.79558461	0.94985448	0.69137756	0.57741723	0.46700448	0.41811471	0.41811471
HOBBIES_1_	0.20473758	0.15469708	0.15469708	0.15469708	0.1514111	0.21978368	0.23164646	0.15407114	0.12109605	0.13034126	0.04874759	0.14793681	0.23125174	0.18375682	0.15432612	0.13034126	0.1099408	0.1099408
HOBBIES_1_	0.24419357	0.16533383	0.16533383	0.16533383	0.20303175	0.26331259	0.27517537	0.13477102	0.10179592	0.17792164	0.06812476	0.18292959	0.27740987	0.19765338	0.23282989	0.17648908	0.13946076	0.13946076
HOBBIES_1_	0.32804561	0.24262187	0.22599401	0.24809866	0.27588843	0.44072413	0.60401836	0.34915049	0.31479238	0.32191919	0.18875601	0.31909131	0.51913435	0.36548815	0.32977735	0.27343654	0.25515449	0.30224775
HOBBIES_1_	1.64879031	1.50529025	1.40420769	1.43292051	1.645661	1.93583364	1.97471177	1.63083231	1.48133258	1.4837623	1.60463192	1.78575253	2.10206115	1.9046607	1.55110462	1.38869437	1.52586491	1.41821543
HOBBIES_1_	3.17663191	2.83862598	2.90255372	2.64641383	3.844925	3.9104551	3.98481709	3.00893125	3.12493734	2.77954709	2.52598691	3.16864512	4.42741754	3.60641224	3.11547773	2.7438941	3.37108526	3.20948324
HOBBIES_1_	4.78699713	4.65384149	4.98927766	4.64048708	6.40577442	5.94264214	6.70374717	5.20644041	5.39935316	4.61465144	4.59156628	5.84393296	7.06939825	6.05038097	5.25347168	4.67629379	4.94871883	5.16927651
HOBBIES_1_	0.75517248	0.75766345	0.67765799	0.7340993	0.86947759	1.01840418	1.14129292	0.75935737	0.96548122	0.97872661	0.69217516	0.88133688	1.51376794	1.39960239	1.00754729	1.10712139	1.2825565	1.31239224
HOBBIES_1_	0.13140098	0.09817091	0.09817091	0.09817091	0.130396	0.19464098	0.2020052	0.13305604	0.09558238	0.10932615	0.02323392	0.12242314	0.20161048	0.15861413	0.12881245	0.10932615	0.08442714	0.08442714
HOBBIES_1_	4.44198951	4.62356427	4.38042005	4.68952933	5.61020001	6.81413228	8.7973874	7.62109436	7.95473617	7.22338166	6.64648168	9.77544839	9.48612905	8.33613549	9.04480912	7.05770345	7.07326904	6.97020593
HOBBIES_1_	0.26369847	0.19301838	0.19301838	0.21572807	0.23958466	0.41253088	0.47312695	0.36832158	0.29161215	0.29469474	0.16153156	0.28388947	0.384265	0.30409765	0.18891809	0.16493322	0.14453277	0.14453277
HOBBIES_1_	0.76492717	0.68972822	0.59826219	0.54833448	0.60942467	0.76625808	0.74641313	0.55203273	0.48741968	0.47221886	0.38360234	0.52757824	0.84757511	0.7428462	0.55553609	0.5842627	0.54411598	0.58498758
HOBBIES_1_	0.34417715	0.28070955	0.49047577	0.40501078	0.40434908	0.51999051	0.50227703	0.31116164	0.27680354	0.26834639	0.15521554	0.2814638	0.50908615	0.59781219	0.40805027	0.35226572	0.37822397	0.39723089
HOBBIES_1_	0.74070009	0.91433977	0.83589784	0.78718097	1.01152676	1.36055382	1.26931831	0.96462087	0.77966669	0.78679349	0.72366692	0.81872748	1.46009752	1.44333411	1.23733333	1.0549354	1.02295891	1.03309455
HOBBIES_1_	0.16151857	0.1282885	0.1282885	0.1282885	0.16765847	0.24397197	0.25583475	0.16317363	0.13019854	0.13944374	0.05451611	0.16418418	0.25544003	0.20501935	0.15998422	0.13599936	0.1155989	0.17361568
HOBBIES_1_	0.54398663	0.50725514	0.47822232	0.4293843	0.44983852	0.65686791	0.7700084	0.55486027	0.49376604	0.54170475	0.39538648	0.60423994	0.78756584	0.68131656	0.5040824	0.61050889	0.51295269	0.51295269
HOBBIES_1_	0.19986214	0.16313064	0.16313064	0.16313064	0.25365103	0.31780642	0.29771057	0.20151719	0.16504068	0.15889376	0.11935614	0.25229515	0.33402211	0.27426601	0.29248404	0.29441484	0.25807236	0.21786119
HOBBIES_1_	0.51882121	0.39069641	0.39069641	0.37128824	0.37294218	0.4860977	0.50445243	0.35964412	0.33323819	0.39384384	0.42833425	0.56128297	0.68777409	0.59846136	0.43501402	0.38640241	0.32790919	0.28527624

The part of the submission file generated by the model with 'Poisson' Objective Loss Function is as follows:

id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
HOBBIES_1_	0.90955465	0.76751038	0.73918728	0.73440907	0.78061955	0.92136226	0.9872031	0.69211847	0.85741194	0.84140194	0.88289325	0.93637814	1.26389692	1.20756634	0.9931755	0.81602808	0.77789007	0.79002617
HOBBIES_1_	0.42174526	0.36213409	0.36118428	0.33621259	0.36354328	0.44066449	0.44509492	0.29710515	0.26270993	0.25633385	0.25083748	0.28992455	0.35818247	0.31174011	0.24421137	0.22486845	0.22010521	0.22526347
HOBBIES_1_	0.48514649	0.43803502	0.43490493	0.44036492	0.50326181	0.56672855	0.7211492	0.45522124	0.39734807	0.38521713	0.37028842	0.42874555	0.54306913	0.45276001	0.38140553	0.34015179	0.36828919	0.42953116
HOBBIES_1_	1.84866232	1.52497891	1.45535705	1.36299053	1.80791436	2.07853027	2.19064384	1.63705624	1.63311573	1.49571163	1.33083433	1.75739149	2.47851548	2.22447354	2.01130199	1.52899049	1.66713551	1.56744081
HOBBIES_1_	0.96133687	0.728683	0.91520005	0.96761055	1.01721991	1.39090758	1.68911123	1.13154368	1.1023755	0.95238223	0.91910227	1.05067912	1.4290951	1.17029755	1.09971339	0.91980146	0.90026155	0.91673678
HOBBIES_1_	1.1470353	1.05095692	1.19219269	1.04354481	1.1776064	1.37016541	1.29832669	0.98949267	0.90104616	1.07247698	0.86894326	1.13499842	1.27562147	1.04980074	1.00254883	1.233937	0.96198613	0.98090559
HOBBIES_1_	0.35552094	0.38345508	0.35979642	0.35255989	0.42956965	0.51329095	0.52899551	0.39581736	0.33062853	0.32332473	0.3115012	0.35125153	0.48617912	0.43997737	0.3618686	0.35027255	0.33308254	0.34022999
HOBBIES_1_	5.63483687	5.92158526	6.02354997	5.81332271	6.92851236	9.92446366	9.90304048	8.93079153	7.94877162	7.9148471	6.58897166	9.93547992	10.1775413	7.56338502	8.41991093	7.70459388	7.72511075	7.07723295
HOBBIES_1_	0.3089503	0.37975273	0.35141506	0.40236825	0.49479388	0.57490243	0.97377748	0.66947143	0.56868913	0.53715583	0.48237329	0.54295699	0.74088339	0.65928673	0.52759634	0.78176886	0.420313	0.42708486
HOBBIES_1_	0.37491411	0.4108474	0.42145755	0.43836724	0.5022035	0.62731163	0.73835315	0.64350006	0.63741115	0.64834197	0.55145951	0.71868917	0.87168783	0.68304481	0.62570511	0.51331197	0.47141601	0.48153189
HOBBIES_1_	0.1600785	0.13641034	0.13605256	0.13802725	0.13902391	0.1727029	0.17435271	0.13000353	0.12031854	0.12121165	0.11895414	0.13495204	0.16494719	0.14685496	0.12822777	0.12086463	0.1167706	0.11950716
HOBBIES_1_	0.21353827	0.17265849	0.16885648	0.17097638	0.19577315	0.22028554	0.22547503	0.15519094	0.14479782	0.17044575	0.16052762	0.18697027	0.22315607	0.18244536	0.19508715	0.17375302	0.15533561	0.15897596
HOBBIES_1_	0.34298847	0.28289499	0.27319968	0.29962512	0.31395599	0.42857952	0.49198265	0.358353	0.3292991	0.31873263	0.29218243	0.34912519	0.46573743	0.39550579	0.37555785	0.34205687	0.33012669	0.35213025
HOBBIES_1_	1.78717498	1.44527209	1.36692485	1.42287584	1.71422918	2.04387766	2.14397183	1.70524782	1.42681058	1.45707687	1.56458026	1.89170281	2.08417434	1.85420049	1.55234923	1.37821947	1.52894742	1.48162783
HOBBIES_1_	3.29633044	2.85158433	2.83321576	2.74656662	3.75035086	3.90329804	3.83240063	2.95789036	2.84180425	2.69401314	2.47711914	3.04348293	3.98443135	3.36557084	3.07213238	2.63279662	2.98595941	2.98700237
HOBBIES_1_	4.94198023	4.35322866	4.87777834	4.6106011	5.85280363	6.32362283	6.58041834	5.13103503	5.28740336	4.77587009	4.58551706	5.67276056	7.04103298	5.55941493	5.073066	4.38723502	4.67988496	5.01988221
HOBBIES_1_	0.81792572	0.83223877	0.75840428	0.81475379	0.97449429	1.1146586	1.31799064	0.92954672	0.97551409	0.95380276	0.82984927	1.00016108	1.41989044	1.3118184	1.06167057	1.07903006	1.15199634	1.31714177
HOBBIES_1_	0.07803796	0.0722268	0.07203736	0.07130326	0.0830835	0.10652521	0.09880192	0.07492809	0.0665294	0.06957616	0.0656761	0.07562026	0.09472278	0.09062663	0.07158954	0.06981913	0.0628055	0.06427738
HOBBIES_1_	4.70302298	4.82580556	4.57552059	5.43801965	5.71947311	5.97885801	7.2173602	6.92184415	7.07060001	7.19672408	5.79121526	9.14396101	8.0097072	7.97880335	8.54523486	7.47559633	7.16638147	7.20440831
HOBBIES_1_	0.25356453	0.22439712	0.22281038	0.24499856	0.2631757	0.3699895	0.38606486	0.33572612	0.28938957	0.27544976	0.25221816	0.29040843	0.35920301	0.30151728	0.22734474	0.21198709	0.20618826	0.20657239
HOBBIES_1_	0.74861439	0.64074997	0.55352495	0.56671659	0.59674922	0.73698583	0.73500493	0.58502927	0.50130702	0.48044976	0.34254168	0.40694196	0.81945643	0.70264021	0.59222473	0.58828923	0.53412522	0.59120378
HOBBIES_1_	0.33970563	0.3001114	0.38569905	0.38062738	0.41862713	0.49272236	0.46843992	0.34663638	0.30959164	0.31883235	0.30214969	0.35828937	0.48215691	0.47685934	0.39277442	0.35269092	0.35637344	0.39678225
HOBBIES_1_	0.78189765	0.86616856	0.78849317	0.77123329	0.98458402	1.36896015	1.32053867	1.03596138	0.7993861	0.80010282	0.8162665	0.89690661	1.37770855	1.32008326	1.28643141	1.10153275	0.98145314	1.02366861
HOBBIES_1_	0.16856946	0.15680497	0.15639371	0.15866363	0.17163298	0.21143895	0.20821889	0.1649221	0.14762951	0.15272056	0.14422045	0.1539368	0.18953982	0.1743561	0.14730418	0.14402425	0.13376308	0.1672398
HOBBIES_1_	0.61469477	0.53114117	0.51817795	0.49132939	0.50790244	0.71833939	0.79487266	0.62138961	0.52994871	0.5568757	0.48555858	0.64580096	0.80799407	0.69324189	0.57566337	0.5985973	0.53789465	0.54943706
HOBBIES_1_	0.1790699	0.16509008	0.16465709	0.16704694	0.22553615	0.27418334	0.24149517	0.18210689	0.16837567	0.15616722	0.18001741	0.22770773	0.27743562	0.24484248	0.24625589	0.2707576	0.25782126	0.22285936
HOBBIES_1_	0.60204217	0.48559611	0.46863696	0.47246009	0.51343713	0.6381639	0.65975347	0.50794743	0.47161177	0.50156519	0.51179529	0.62942714	0.74534929	0.6260082	0.53381681	0.47206567	0.44164635	0.42711727

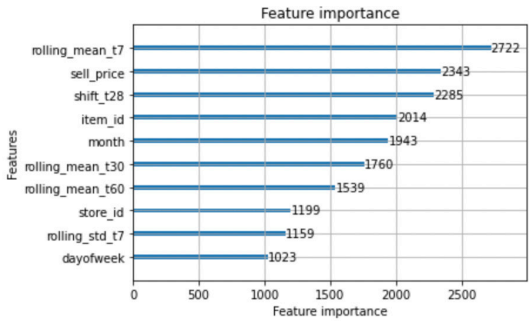
After generating the corresponding result, it can be submitted on the Kaggle platform to receive the measurement accuracy provided by the system. This evaluation method, RMSSE, is highly reliable and can be used as an additional measure for this solution.

6.2 Features' Importance Analysis

The tree-based model can be used to evaluate the importance of features, and in LightGBM, the feature importance ranking can be done by the 'lgb.plot_importance()' method where the ranking is based on the number of times a feature is used as a segmentation attribute in all trees and the total gain from using the feature as a segmentation. Of the 25 features used for training, the top 10 important features are selected as having the most excellent value for training. In general, both sets of parameters show that the price and the product itself play a crucial role in sales forecasting, followed by historical sales data and date information such as month and week. Next, a specific analysis of each model constructed with two sets of parameters follows.

6.2.1 For 'Regression' Objective Loss Function

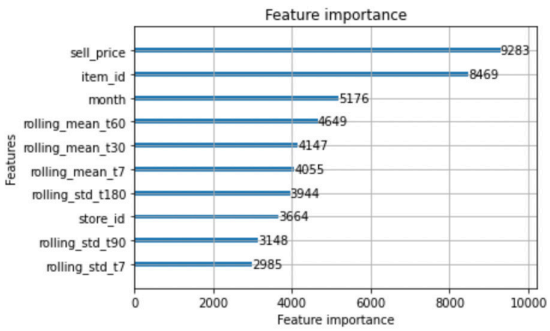
```
Here is the most imortant 10 features:
```



The image shows that the most important features in the training process are the features obtained by rolling on sales, followed by the raw unprocessed selling price, the ID of the item itself and the month, the shop ID, and the day of the week. The above result shows that the model where the target loss function is specified as regression places more importance on the two pieces of information, sales, and price. And the pattern demonstrates that these two pieces of data, ‘rolling_mean_t7’ and ‘sell_price’, can be more indicative in the model’s training process.

6.2.2 For ‘Poisson’ Objective Loss Function

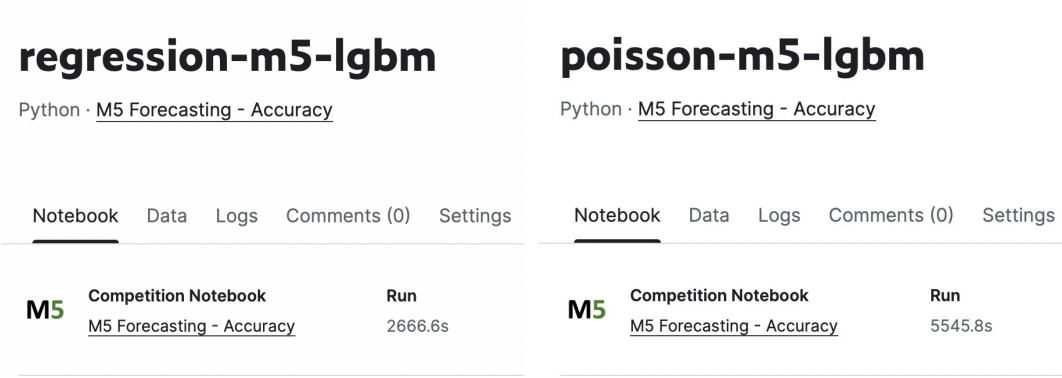
```
Here is the most imortant 10 features:
```



As can be seen from the visual graphs, in the model with Poisson as the target loss function, the two features that play the most crucial role when the training process is underway are ‘sell_price’ and ‘item_id’, which proves that, in this paper's assumption

of a Poisson distribution for the pattern of item sales, sell price and the item itself are the most valuable information for sales volume prediction. This is followed by the month, the demand characteristics after rolling, and the shop ID.

6.3 Time-consuming Analysis



Due to memory limitations, evaluating the model's performance can also be reflected in the efficiency of the run. In the same configuration environment, the model with regression as the target loss function has a run time of 2666.6 s, while another set of parameters has a run time of 5545.8 s. The results show that between the two, even in the fastest running model, every 100 rounds run, the time consumed is close to 222 seconds. Furthermore, the comparison shows that the former runs only 48% of the latter's time.

6.4 Accuracy Analysis

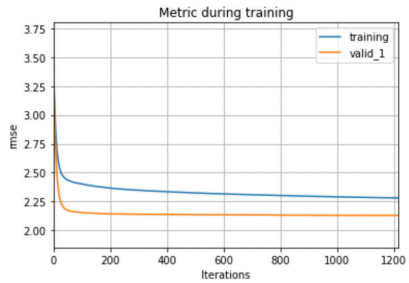
6.4.1 Metrics Related to the Output

The results for the ‘regression’ objective loss function:

```
Training until validation scores don't improve for 200 rounds
[100]	training's rmse: 2.40159	valid_1's rmse: 2.15446
[200]	training's rmse: 2.36701	valid_1's rmse: 2.1423
[300]	training's rmse: 2.34757	valid_1's rmse: 2.13901
[400]	training's rmse: 2.33499	valid_1's rmse: 2.13767
[500]	training's rmse: 2.32441	valid_1's rmse: 2.13541
[600]	training's rmse: 2.31592	valid_1's rmse: 2.13451
[700]	training's rmse: 2.30845	valid_1's rmse: 2.13405
[800]	training's rmse: 2.30177	valid_1's rmse: 2.13253
[900]	training's rmse: 2.29582	valid_1's rmse: 2.13097
[1000]	training's rmse: 2.29014	valid_1's rmse: 2.13004
[1100]	training's rmse: 2.28566	valid_1's rmse: 2.13029
[1200]	training's rmse: 2.28125	valid_1's rmse: 2.13028
Early stopping, best iteration is:
[1017]	training's rmse: 2.28947	valid_1's rmse: 2.12971
```

```
Our val rmse score is 2.129705727009993
```

```
Metric during training:
```

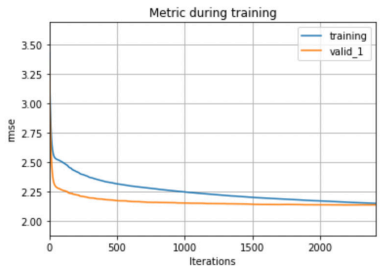


The results for ‘Poisson’ objective loss function:

```
Training until validation scores don't improve for 200 rounds
[100]   training's rmse: 2.49648        valid_1's rmse: 2.25882
[200]   training's rmse: 2.41661        valid_1's rmse: 2.21925
[300]   training's rmse: 2.36747        valid_1's rmse: 2.19277
[400]   training's rmse: 2.33562        valid_1's rmse: 2.18054
[500]   training's rmse: 2.31259        valid_1's rmse: 2.17044
[600]   training's rmse: 2.29464        valid_1's rmse: 2.16295
[700]   training's rmse: 2.28169        valid_1's rmse: 2.15888
[800]   training's rmse: 2.26759        valid_1's rmse: 2.15486
[900]   training's rmse: 2.25578        valid_1's rmse: 2.15283
[1000]  training's rmse: 2.24391        valid_1's rmse: 2.14943
[1100]  training's rmse: 2.2332 valid_1's rmse: 2.14732
[1200]  training's rmse: 2.22349        valid_1's rmse: 2.14481
[1300]  training's rmse: 2.21412        valid_1's rmse: 2.14432
[1400]  training's rmse: 2.20602        valid_1's rmse: 2.14243
[1500]  training's rmse: 2.19819        valid_1's rmse: 2.13901
[1600]  training's rmse: 2.19097        valid_1's rmse: 2.13714
[1700]  training's rmse: 2.18483        valid_1's rmse: 2.13653
[1800]  training's rmse: 2.17946        valid_1's rmse: 2.13677
[1900]  training's rmse: 2.17252        valid_1's rmse: 2.13464
[2000]  training's rmse: 2.16724        valid_1's rmse: 2.13481
[2100]  training's rmse: 2.16237        valid_1's rmse: 2.1341
[2200]  training's rmse: 2.15625        valid_1's rmse: 2.13325
[2300]  training's rmse: 2.15153        valid_1's rmse: 2.13401
[2400]  training's rmse: 2.1473 valid_1's rmse: 2.13383
Early stopping, best iteration is:
[2215]  training's rmse: 2.15559        valid_1's rmse: 2.1329
```

```
Our val rmse score is 2.132899742625695
```

```
Metric during training:
```



The performance of each set of parameters on the training and validation sets is first observed, and their prediction accuracy is initially measured by the value of the RMSE.

Plotting the change in RMSE of the model during training as the number of iterations increases reveals some insights, as shown in the figure below:

Both models show predictive power for time series questions, where the model using regression as the objective function has a decrease in RMSE from 2.40159 to 2.28947 over the training set and from 2.15446 to 2.12971 on the validation set.

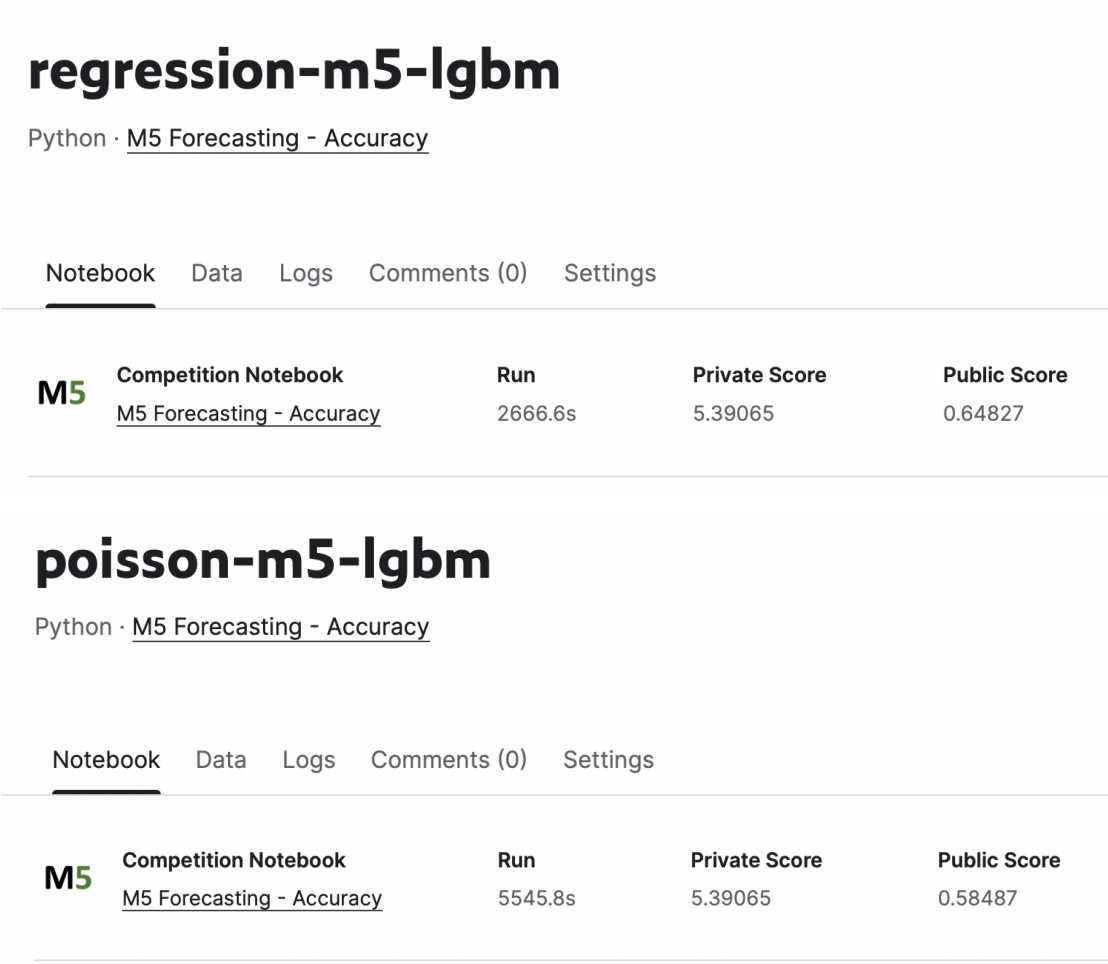
Similarly, for the regression model specified as Poisson distribution, it can be observed that the general trend of the RMSE, both on the training and validation sets, gradually decreases and approaches approximately 2.1 with the number of iteration rounds. Still, after 2215 rounds, it shows a rebounding trend, and the results of the program run also indicate that at round 2215, the lowest RMSE on the validation set is at 2.1329, showing that the model is best trained at this time.

In order to obtain the best training model, the model constructed with regression as the target loss function takes 1017 iterations to be trained, while the model constructed with Poisson as the objective loss function only reaches the best performance at the 2215th iteration. So far, in terms of the part of the procedure itself that can be demonstrated, the former can be a significant time saver, although the training error on the training set is larger. However, between the two, its RMSE on the validation set is the lowest. While for the latter, although time-consuming, after multiple iterations, the difference between the final results of the model on the training and validation sets is minimal, and this gap is only about 0.003, which seems to produce the most stable results with the expectation of better scores when there is an unknown RMSSE.

The predictions generated by both can next be uploaded to the official channel for evaluation to get the final RMSSE value.

6.4.2 Public Score

The fitted model is then applied to the test set to get specific predictions. After generating the submission file, it should be uploaded to the Kaggle website to get the official RMSSE-based scores. In addition, since this project only predicts sales from day 1914 to day 1941, the evaluation of the private score is ignored here, and only the public score is focused on. The values of the specific evaluation metrics are as follows.



It could be learnt that the public score of the two sets of parameters are 0.64827 and 0.58487 respectively.

6.5 Comparison of Outputs and Assumptions

In the previous hypothesis, it is assumed that each product should have a high prediction accuracy for the 28-day prediction values, which is reflected in the experiments as a measure of the magnitude of the error values. It is also assumed that when the model is available, then its loss value should be within the interval of the excellent work score submitted in the public kernel. This is demonstrated by the fact that the RMSE on the validation set should be between 1.0 and 3.0 and the final RMSSE on the test set should be between 0.3 and 0.8. Where RMSE is an important secondary measure, the final accuracy is measured primarily based on the value of RMSSSE.

The results of the evaluation of the LightGBM model constructed from these two sets of parameters indicate that this goal has been achieved. In this project, the RMSE on the validation set for both is between 2.1 and 2.2. Also, two RMSSSEs on the test set are between 0.5 and 0.7. Therefore, the results are as expected. Of these, the LightGBM model using the Poisson distribution is the best model to predict Wal-Mart merchandising, as its final best RMSSSE is 0.58487. The results also show that LightGBM has a way of bringing out the importance of features in the training process and can be used for further prediction and model optimisation. Therefore, LightGBM is a well-suited model for this project.

6.6 Discussion

As the goal of this project is to find the best method for daily sales forecasting, the LightGBM model is trained here with a limited number of variables, and the performance of models constructed with different parameters is compared, with each set of parameters showing their strengths and weaknesses.

Again, both are trained using LightGBM as the model framework, the difference being the target loss function, one using regression and the other using Poisson, and the other parameters are adjusted as the function varies.

According to the final results, the model using regression as the target loss function produces the lowest RMSE on the validation set as output, which is the best-performing model in terms of the initial judging criteria, and the run time is halved compared to the other loss function, which is much more efficient in comparison. However, according to the final criterion, although the RMSE is somewhat higher, the RMSSE with Poisson as the target loss function produces the lowest loss value for the output. This indicates that the distribution of the target in this regression task is closer to the Poisson distribution. It has the best prediction accuracy but is also the most time-consuming.

However, given the highest accuracy and the depth of the model in the data set, it is still recommended as the best model for daily sales and even time series forecasting. Of course, the model still needs to be modified, and a more time-saving method needs to be found to make the forecasts more accurate.

It is also found that future sales trends are well predicted for most items with stable sales data and some seasonal characteristics. Overall, the RMSSEs of the best solutions that won the comparison competition were generally between 0.3 and 0.8, and the two methods used in this project, with different sets of parameters, are in this range, which means they have good performance.

Conclusion and Recommendation

7.1 Summary and Conclusions

For the Walmart sales forecasting problem, the LightGBM algorithm is used here. The project mainly refers to historical sales data, and at the same time combines external variables during model training, such as the item itself and the effect of the date on buying habits. In addition, it is easy to implement and it overcomes the overfitting effect. Another benefit of using LightGBM as a model for training is that as a decision tree-based algorithm and it is a good way to visualise the importance of features, which can help to improve the model and to analyse the benefits of the features. In the implementation, the features are first rolled and extracted, then the importance of them is selected in different sets of parameters, and the output about RMSE is performed for each product in the dataset for both the training and validation sets, also the sales forecast for the next 28 days.

As a result, comparing the two sets of parameters, it is found that the LightGBM model using a Poisson distribution has a higher accuracy. Therefore, the final output is mainly based on the model constructed with this set of parameters.

The output shows that the model design is as expected, and it is determined that using the LightGBM model constructed with Poisson as the objective loss function, which produces an RMSE of 2.1329 on the validation set and a final RMSSE of 0.58487 on the test set from the Kaggle submission page.

7.2 Limitations and Recommended Future Studies

The difficulty of the problem is that the data available for reference is limited, so in order to improve the performance of the model, future research could be based on the goal of providing the model with more data of reference value to extract into features.

For example, some product information and corresponding data on customer behaviour and preferences. More useful features can help to build accurate models. In addition, this report only compares the predictive performance of individual algorithms for different sets of parameters. And it does not make an integrated relationship between them, resulting in training that may need to be more comprehensive. Moreover, just like the study of (Lai et al. 2018) and (Xie et al. 2021), there are still many excellent algorithms that are not covered, such as the recently popular neural network algorithm Long Short-Term Memory (LSTM), also some simple regression algorithms that are useful in time-series prediction problems but are not mentioned in this paper. Then a future research direction could be to combine the new model with LightGBM to build a comprehensive model to achieve better prediction accuracy performance.

This paper uses sliding average features to predict sales, however, there are many extremes in the actual data, e.g., a large number of 0 and above ten values for some items, but most predictions are essentially between 0 and 10, which can lead to some bias in the forecasts. Subsequent work could focus on grouping sales in different cases, especially for extreme values since this could be important. By looking at other good works, i.e. programs with higher final accuracy, it can be seen that training in terms of shops or types of goods allows, on the one hand, the training to be carried out in parallel after the division, thus increasing the efficiency of the model run; on the other hand, as the concept of classification is strengthened, there is a certain correlation between the various shops or goods under the category, which makes it easier to obtain patterns in the training process, thus reducing the prediction error.

The Tweedie distribution might be more suitable for this problem, as can be seen from other works in the public kernel. It is a composite distribution of the Poisson and Gamma distributions and is related to the Poisson distribution. Both are more applicable to counting variables with many zero values (zero-inflated data), like the dependent variable being sales. However, the Tweedie objective function runs very slowly in the calculation, so Poisson is still used here as the loss function.

In terms of result generation, a low RMSE does not necessarily guarantee a low RMSSE (interpreted RMSSE), and future research directions could be based on this. In addition to this, cross-validation or k-fold validation methods could be used to increase the robustness of the program. And using multi-threading can also be considered to increase the speed of running.

References

Athanasopoulos, G., Ahmed, R.A. and Hyndman, R.J. 2009. Hierarchical forecasts for Australian domestic tourism. International Journal of Forecasting 25(1), pp. 146–166.

Bandara, K., Bergmeir, C. and Smyl, S. 2020. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. Expert systems with applications 140, p. 112896.

Carbonneau, R., Laframboise, K. and Vahidov, R. 2008. Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research 184(3), pp. 1140–1154.

Chen, I.-F. and Lu, C.-J. 2017. Sales forecasting by combining clustering and machine-learning techniques for computer retailing. Neural Computing and Applications 28(9), pp. 2633–2647.

Chen, T., Yin, H., Chen, H., Wang, H., Zhou, X. and Li, X. 2020. Online sales prediction via trend alignment-based multitask recurrent neural networks. Knowledge and Information Systems 62(6), pp. 2139–2167.

Di Pillo, G., Latorre, V., Lucidi, S. and Procacci, E. 2016. An application of support vector machines to sales forecasting under promotions. 4OR 14(3), pp. 309–325.

Dong, D., He, R. and Xiong, G. 2019a. Foreign Commodity Sales Forecast Based on Model Fusion. In: Proceedings of the 2019 4th International Conference on Big Data and Computing., pp. 185–188.

Dong, W., Li, Q. and Zhao, H.V. 2019b. Statistical and machine learning-based e-commerce sales forecasting. In: Proceedings of the 4th International Conference on Crowd Science and Engineering., pp. 110–117.

Ferreira, M.D., Corrêa, D.C., Nonato, L.G. and de Mello, R.F. 2018. Designing architectures of convolutional neural networks to solve practical problems. Expert Systems with Applications 94, pp. 205–217.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.-A. 2019. Deep learning for time series classification: a review. Data mining and knowledge discovery 33(4), pp. 917–963.

Ke, G. et al. 2017. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems 30

Lai, G., Chang, W.-C., Yang, Y. and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research & development in information retrieval., pp. 95–104.

Lingxian, Y., Jiaqing, K. and Shihuai, W. 2019. Online retail sales prediction with integrated framework of K-mean and neural network. In: Proceedings of the 2019 10th International Conference on E-business, Management and Economics., pp. 115–118.

Ma, S. and Fildes, R. 2022. The performance of the global bottom-up approach in the M5 accuracy competition: A robustness check. International Journal of Forecasting 38(4), pp. 1492–1499.

Maçaira, P.M., Thomé, A.M.T., Oliveira, F.L.C. and Ferrer, A.L.C. 2018. Time series analysis with explanatory variables: A systematic literature review. Environmental Modelling & Software 107, pp. 199–209.

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. 2022. M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting

Mancuso, P., Piccialli, V. and Sudoso, A.M. 2021. A machine learning approach for forecasting hierarchical time series. Expert Systems with Applications 182, p. 115102.

Na, L., Ying, C. and De, B. 2019. The empirical analysis of convenience store sales forecast based on multidimensional data in emerging markets. In: Proceedings of the 3rd International Conference on Business and Information Management., pp. 32–36.

Nasios, I. and Vogklis, K. 2022. Blending gradient boosted trees and neural networks for point and probabilistic forecasting of hierarchical time series. International Journal of Forecasting

Nenova, Z.D. and May, J.H. 2016. Determining an optimal hierarchical forecasting model based on the characteristics of the data set. Journal of Operations Management 44, pp. 62–68.

Vakharia, A.J. 2002. E-business and supply chain management. Decision Sciences 33(4), pp. 495–504.

Wang, M.-H., Gao, T., Li, H. and Xie, W. [no date]. M5 Forecasting-Accuracy Estimate the unit sales of Walmart retail goods.

Xie, H.H., Li, C., Ding, N. and Gong, C. 2021. Walmart Sale Forecasting Model Based On LSTM And LightGBM. In: 2021 2nd International Conference on Education, Knowledge and Information Management (ICEKIM). IEEE, pp. 366–369.