CSCI567 Project - Store Sales Forecasting

Ruoxi Li
ruoxil@usc.edu

Pan Hu
panhu@usc.edu

Dingzhou Cui
dingzhoc@usc.edu

Abstract

"Store Sales - Time Series Forecasting" is a Kaggle competition that using machine learning to predict grocery sales. In this competition, contestants will predict sales for the thousands of product families sold at Favorita stores located in Ecuador. Here, we will show how to preprocess data and build useful features and introduce the our algorithm to make precise sales preductions for grocery stores.

1 Data Preparation

The training data includes dates, store and product information, whether that item was being promoted, as well as the sales numbers. Additional files include supplementary information that is useful in building models. "store.csv" includes store metadata, city, state, type, and cluster. "oil.csv" includes oil price. "holidays_events.csv" includes holidays and events.

1.1 Dataset Selection

First, we should decide how much time series data to use. Since the sales data that are far away from 2017 has different distribution, using too much data can be harmful for the performance of the machine learning model. In our work, we use only data after 2016. We tried to use data in longer time span but did not get any improvement on the model performance.

1.2 Dataset Split

Splitting training and testing data randomly is not a good idea for time series data, as there's dependence from one observation to the other. The splitting should be based on time to avoid the look-ahead bias. To minimize the gap between the validation set and the testing set, the validation set should start at the same day of week and have the same time span as the testing set. Thus, we used data from 2017-07-26 to 2017-08-10 as the validation set and the others as the training set.

1.3 Data Preprocessing

The evaluation metric for this competition is Root Mean Squared Logarithmic Error (RMSLE). The RMSLE is calculated as: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(1+\hat{y}_i)-\log(1+y_i)\right)^2}$. It is important to note that, unlike the RMSE, the RMSLE is asymmetric; penalizing much more the underestimated predictions than the overestimated predictions. To align with the evaluation metric, we take the $\ln(1+y)$ of the target, in this case the target is the sales. Then we can use Mean Square Error (MSE) as the target function during model training.

2 Feature Engineering

Feature engineering is a crucial step for training machine learning model and it can determine the upper limit of the model. Since 2017-8-16, the first day in the testing data, is Wednesday, we extracted

Preprint. Under review.

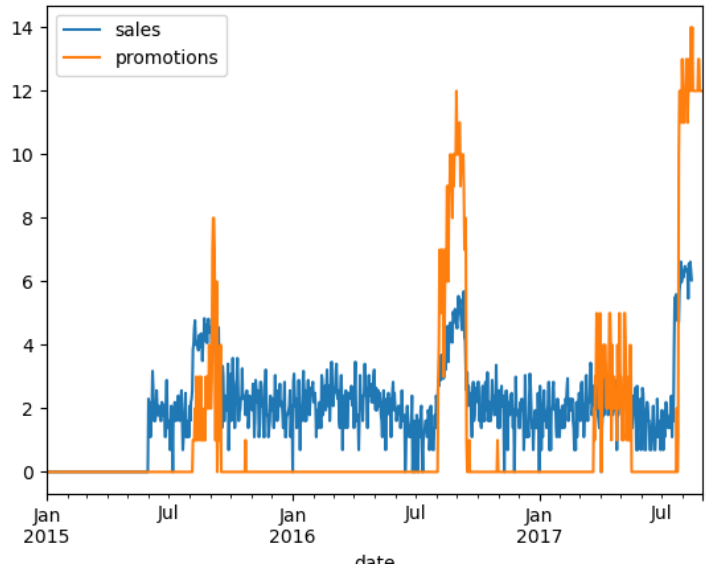

Figure 1: The relationship between promotion and sales for SCHOOL AND OFFICE SUPPLIES

Figure 2: Dataset Splitting and Feature Engineering

Table 1: Features based on past sales and promotions

Feature Description	Total Number
Means of the ratio of sales to promotion	5
Means and weighted sum of sales in promotion day	5×2
Means and weighted sum of sales in non-promotion day	5×2
Statistic of sales (diff., weighted sum, mean, median, min., max., std.)	$5 \times 7 \times 2$
Total, first and last days that have sale larger than 0	5×3
Total, first and last days that have promotion larger than 0	5×3
Last 15 days' sale	15
Past 16 days' and future 15 days' promotions	32
Means of the sales on the same day of the last 4 weeks	7
Sales in the same period of time in last year	42
Mean of the sales in the same period of time in last year	3

features on each Wednesday instead of randomly selecting days in training set. We extracted features mainly based on promotions and sales. We can expect that the future sales is related to the past sales. And Figure 1 shows that promotions is another important factors for the sales of many items. The sales were be boosted in the promotion day.

Base on a selected day, we define two type of features, lag features and future features. Lag features are values at prior timesteps that are considered useful because they are created on the assumption that what happened in the past can influence or contain a sort of intrinsic information about the future. We built different lag features by choosing different window size and calculate several statistics. The windows are 3 days, 7 days, 14 days, 30 days, 60 days. These windows can capture weekly, biweekly, monthly and bimonthly trends and they are determined based on the validation error. The sales that we are going to predict should also be correlated to the promotion on that day and this information is given for the testing data, thus we can also define features based on the future 15 days' promotion. Our goal is to predict the 16 days' sales, thus we use the 16 days' sale (from t to t + 16) as the labels.

In our work, we did not use the oil price and the holiday information to build the features, because we believed that the past sales already include the impact of oil price and holidays. Actually, we tried to add features extracted from oil price and holidays, but did not get any improvement. Figure 2 shows how we split the dataset and define features.

We use 21 days (each Wednesday from 2017-3-1 to 2017-7-19) to build the training data, this number is determined by the validation error. We started from only 6 days and found the validation error decreases as we add more data to the training set. And when the number of days is larger than 21, the validation performance starts to get worse.

The next two section summarize all the features that we used to train the model. Actually, we tried many other features but they did not improve the accuracy on the testing set, here we just introduce the features that we used in the final model.

2.1 Features based on past sales and promotions

As we have discussed, we extracted lag features based on each Wednesday. First, we used different window sizes to capture weekly, biweekly, monthly and seasonal trends and calculate different statistic for each window. We tried different window size and determine the number based on the validation error and feature importance. The window sizes that we finally chose are 3 days, 7 days, 14 days, 30 days and 60 days. Second, the recent sales and promotions can reflect future sales, thus we included recent 15 days' sales and 32 days' promotions as features. Third, the same day of week can have similar sales, thus we included means of the sales of the same day in the last 4 weeks as features. Finally, this year's sale should be correlated to last year's sale, thus we included the sales in the same period of time as features. The number of days and weeks are also determined based on validation error and feature importance. Table 1 summarizes all the features based on past sales and promotions.

Table 2: Features based on future promotions

Feature Description	Total Number
Sum of future 3/7/14 days' promotion	3
The number of promotion day in the future 15 days	1
The first promotion day in the future 15 days	1
The last promotion day in the future 15 days	1

Table 3: Hyperparameters of GBDT

Hyperparameter Name	Daily GBDT	Global GBDT
`num_leaves`	31	120
`min_data_in_leaf`	10	40
`learning_rate`	0.02	0.02
`feature_fraction`	0.8	0.8
`bagging_fraction`	0.7	0.7
`early_stopping_rounds`	100	100

2.2 Features based on future sales

As we have discussed, we can also define features based on the promotions after the day t. We calculated the total number of promotion and promotion days in the future 3/7/14 days, the first and last promotion day in the future 15 days and the promotion on each days, Table 2 summarized the features.

3 Model Training

The model that we selected are Gradient Boosting Decision Tree (GBDT) and Neural Network (NN). Both of them have shown their power in many Kaggle competition. We tried two popular GBDT algorithms, XGBoost and LightGBM. Based on our experiment, LightGBM can achieve lower error on validation and testing set and run much faster than XGBoost. We used Keras for build NN. We tuned the hyperparameters of GBDT and NN based on the validation error. We used grid search to find the best combination of the hyperparameters.

Ensemble is a very important step to get a good score in Kaggle competition. Ensemble the prediction from different models can further improve the accuracy. Here, we used the simplest method to ensemble different models, calculating the weighted average of multiple predictions. The weight is determined based on the validation error. Actually, ensemble is a very tricky step. Which models should we use? What features should we use to train each models? The final score is highly depend on your choice.

We used three different methods. First, we trained 16 GBDT (refered as daily GBDT) to predict the sales for each day. Second, we trained 16 NN to predict the sales for each day. Last, we trained 1 GBDT (refered as global GBDT) to predict all 16 days' sale. We add two more features for the last method, the day of week and the lag of days to the first Wednesday, otherwise all days in the same 16 timespan will have the same features.

3.1 GBDT

Table 3 summarizes the grid search result of the hyperparameters for the daily GBDT and global GBDT. A relative low learning rate can make the model find a better local minimum. Training the model with 80% features and 70% samples guarantee a good generalization. And early stopping can avoid the model overfitting on the training set. Since training global GBDT use all 21 days data, the model should be more complicated compared to the daily GBDT. And we can see that the decition trees in the global GBDT have more leaves. Also, the `min_data_in_leaf` is larger in global GBDT to avoid overfitting.

Table 4: Hyperparameters of NN

Hyperparameter Name	Values
`structure`	1 LSTM + 7 DNN
`optimizer`	Adams
`learing rate`	0.001
`early stopping patience`	10
`learing rate reducing patience`	7

Table 5: Ensemble different models

Models	Score
Daily GBDT + Global GBDT	0.38324
Daily GBDT + Global GBDT + NN	0.38279
NN + Dart + Upgini	0.38453
Global GBDT + Dart + Upgini	0.38015
Daily GBDT + Dart + Upgini	0.37857

3.2 NN

Table 4 summarizes the hyperparameters of the NN. It has one LSTM layer followed by 7 fully connected layers. The batch normalization is used to prevent overfitting and accelerate training. We used Adams as the optimizer with a 0.001 learning rate. The learning rate decreases during the training. And we also use early stopping to prevent overfitting.

3.3 Zero forecasting

Some stores do not sell specific products at all or probably stop selling products after some time. So we made it as a hard coded prediction with following logic: 15 days of zero sales from 2017-08-01 till 2017-08-15 will lead to zero forecast. This process can lower the testing error abound 0.002.

3.4 Ensemble

The daily GBDT has get the lowest error on the testing data (0.38613) and the global GBDT and NN reach the error around 0.393. And we also followed these two notebooks, the Dart and Upgini, to get some prediction results. These two notebooks build different features based on the data and used different method to make prediction. Due to the 6 pages limitation we will not talk too much details about these two notebooks. We tried different ways to ensemble the prediction results. Table 5 summarizes top five of the results. We found that ensemble the result from daily GBDT, Dart and Upgini can give the best testing score, 0.37857.

4 Conclusion

In this report, we have shown our method to deal with the time series data. We found that feature engineering is the most crucial step for reaching a low testing error. At the beginning, we can only reach about 0.389 for the daily GBDT. After we added more useful features, especially the features about the sales in the last year, we finally reach 0.386. For Kaggle competition, GBDT is always a good choice, it run pretty fast and can reach a pretty good performance. Ensemble is an important step to further improve the score but it is very tricky and requires a lot of experience. Due to the limitation of time, we only tried the simplest ensemble method, advanced method such as stacking may further improve the score.

Please see the README to reproduce the result.

References

[1] Corporación Favorita Grocery Sales Forecasting, 1st Solution, *https://www.kaggle.com/c/favorita-grocery-sales-forecasting/discussion/47582*

[2] Dart Forecasting. *https://www.kaggle.com/code/ferdinandberr/darts-forecasting-deep-learning-global-models*

[3] Upgini. *https://www.kaggle.com/code/romaupgini/guide-external-data-features-for-multivariatets*

[4] Store Sales: Using the average of the last 16 days *https://www.kaggle.com/code/carlmcbrideellis/store-sales-using-the-average-of-the-last-16-days*