Project 2: Walmart Store Sales Forcasting

sh53, 655723273, SICHENG HE

1 Introduction

Historical sales data for 45 Walmart stores located in different regions is provided. Each store contains a number of departments, and our goal is to predict the future weekly sales for each department in each store based on the historical data.
In addition, Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labor Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks.

2 Data processing

2.1 Trian data

In each fold, train data has 5 columns,

- Store: Store number
- Dept: Department number
- Date: The date for weeks
- Weekly_Sales: Weekly sales for a certain department in a certain store
- IsHoliday: Whether the week is a special holiday week

Then we need to predict the weekly sales for the next two months.

2.2 Get the week number

I create a new variable called **'weeknum'** from **'Date'**, which numbers weeks in each year starting from 0 to 51. I use the function **isocalendar()** provided by **datetime** in Python. Because of the different week definition, I do not meet the trouble that week numbers of 2010 does not match with 2011. So I do not subtract 1 from weeks in 2010.

- **'weeknum'** = 6, Super Bowl
- **'weeknum'** = 36, Labor Day
- **'weeknum'** = 47, Thanksgiving
- **'weeknum'** = 52, Christmas

2.3 Get the year

I create a new variable called **'year'**, which simply extract the year from **'Date'**. There are three possible values 2010, 2011 and 2012.

2.4 Holiday weights

In the data, the feature **'IsHoliday'** is coded as True/False. Since our accuracy refers to the average of the weighted mean absolute error and the weeks including holidays are weighted five times higher in the evaluation than non-holiday weeks. I transform **'IsHoliday'** to be 5/1. That is, 5 for weeks including holidays and 1 for non-holiday weeks. I am going to train a random forest so it doesn't really matter how I code it. By doing this, I can easily get the weights for every data point and train a weighted model.

3 Model fitting and prediction

3.1 Random forest

For each fold, I train a random forest using 'sklearn.ensemble.RandomForestRegressor()' with **'Store', 'Dept', 'IsHoliday', 'weeknum', 'year'** as features and the training parameters are as follows.

- $n\_estimators = 150$. The number of trees in the forest.
- $max\_depth = 40$. The maximum depth of the tree.
- $max\_features =' auto'$. The number of features to consider when looking for the best split. Here I use all features because there are only five features.
- $criterion =' squared\_error'$. Mean squared error is used in training. I do not use the mean absolute error because training using "absolute_error" is significantly slower than when using "squared_error".
- $n\_jobs = -1$. The number of jobs to run in parallel. -1 means using all processors, which can significantly accelerate the model fitting.
- $bootstrap = True$. Bootstrap samples are used when building trees
- $random\_state = 542$. Control both the randomness of the bootstrapping of the samples used when building trees.

3.2 Sample weight

Our accuracy refers to the average of the weighted mean absolute error and the weeks including holidays are weighted five times higher. So it maybe helpful to include 'sample_weight' in training to fit a weighted model. I just set 'sample_weight' to be the 'IsHoliday' as I mentioned previously.

3.3 Prediction

After fitting the model, I first set the date range which needs prediction. For example, in fold 1, we need to predict weekly sales from 2010-02 (February 2010) to 2011-02 (February 2011). Then I extract the corresponding rows in 'test.csv', process it as how I transform the train data and get the prediction. Finally, a new column 'Weekly_Pred' is added to the test data.

3.4 Running time and the accuracy

Runing time and the accuracy for 10 folds are computed. Runing time includes the time for data processing and model fitting.
My computer system: RedmiBook Pro 14, 1.80GHz, 8Cores/16Threads, 16GB memory, Windows10.

Folds	running time(s)	WMAE
1	8.11	1741.38
2	5.5	1493.79
3	6.7	1363.15
4	7.85	1462.81
5	8.92	2615.57
6	10.31	1643.71
7	10.91	1683.64
8	12.21	1364.33
9	12.89	1294.79
10	13.62	1330.36
	All: 97.02	Avg: 1599.34

Table 1: Running time and WAME for 10 splits

4 Discussion

By doing this project, I realize that time series predicting problems do not always require complex models. Even basic models can give relatively ideal performance. Of course, we need to think carefully about the seasonality and holiday effects when implementing models.

Taking seasonality into account is important in time series problems. In this project, I look carefully about the date and the corresponding week numbers to make it match with each other. Then my model is able to deal with seasonality. For holiday effects, I set it as a feature and also include it in the fitting procedure to fit a weighted model. And the MWAE is reduced compared to unweighted model.