Capstone Project
Machine Learning Engineer Nanodegree

Zijun Hu
Oct 20th, 2020

Rossmann Store Sales

Definition

Project overview

Data mining is a process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems. The difference between data analysis and data mining is that data analysis is used to test models and hypotheses on the dataset, e.g., analyzing the effectiveness of a marketing campaign, regardless of the amount of data; in contrast, data mining uses machine learning and statistical models to uncover clandestine or hidden patterns in a large volume of data. [1]

This Project is on this domain. It will be built to conduct the data from Featured Prediction Competition „Rossmann Store Sales“ hosted at Kaggle. Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality.
With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

Project Statement

The sales of a store are impacted by many features, such as store type, competitions, promotions, holidays, day of week ans so on. This project is a forecasting problem using time series. The goal is to predict the next 6 weeks of sales for 1,115 stores located across Germany using models trained by a time series from the span 2013-1-1 to 2015-7-31. The accuracy and quality of the modeling will be evaluated through the Root Mean Squre Percentage Error.

Strategy

For this project XGBoost and LightGBM are applied to do the modeling. A linear regression was also be conducted just for a purpose of camparison. Then for the sake of efficiency I'll choose LightGBM to do the parameter tuning and weight correction further more to improve the results.

Metrics

As Kaggle required the updated forecast are valuated on the Root Mean Square Percentage Error (RMSPE). The RMSPE is calculated as

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

where y_i denotes the sales of a single store on a single day and yhat_i denotes the corresponding prediction. Any day and store with 0 sales is ignored in scoring.

RMSPE is a measurement of the error between the predicted value $\hat{y}$ and the true value y. This eroor should be scaled by the level of the true value and not be biased by the sign of the error. This means the smaller a RMSPE is, the better modeling has been done because the predicting is more accurate.

The goal of the this problem is to pursue a small RMSPE on the test dataset. The deduced result should be camparable with the outcomes in Leaderboard.

Analysis

Data Exploration

for modeling the datasets ‚train‘ and ‚store‘ are used.

Train dataset

By reading in the dataset ‚train‘ in a dataframe from a .csv file the feature ‚Date‘ was applied as the index, so that it was assigned as the datetype ‚Datetime‘ to have a benifit for the further processing.

Feature description:

- Store - a unique Id for each store
- Sales - the turnover for any given day (this is what you are predicting)
- Customers - the number of customers on a given day
- Open - an indicator for whether the store was open: 0 = closed, 1 = open
- Promo - indicates whether a store is running a promo on that day
- StateHoliday - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- SchoolHoliday - indicates if the (Store, Date) was affected by the closure of public schools

General information:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1017209 entries, 2015-07-31 to 2013-01-01
Data columns (total 8 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   Store          1017209 non-null  int64
 1   DayOfWeek      1017209 non-null  int64
 2   Sales          1017209 non-null  int64
 3   Customers      1017209 non-null  int64
 4   Open           1017209 non-null  int64
 5   Promo          1017209 non-null  int64
 6   StateHoliday   1017209 non-null  object
 7   SchoolHoliday  1017209 non-null  int64
dtypes: int64(7), object(1)
```

Store dataset

Feature description:

- Store: a unique Id for each store
- StoreType: differentiates between 4 different store models: a, b, c, d
- Assortment: describes an assortment level: a = basic, b = extra, c = extended
- CompetitionDistance: distance in meters to the nearest competitor store
- CompetitionOpenSince[Month/Year]: gives the approximate year and month of the time the nearest competitor was opened
- Promo2: Promo2 is a continuing a promotion for some stores: 0 = store is not participating, 1 = store is participating
- Promo2Since[Year/Week]: describes the year and calendar week when the store started participating in Promo2
- PromoInterval: describes the consecutive intervals Promo2 is started, naming the months the promotion is started. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

General information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Store                      1115 non-null   int64  
 1   StoreType                  1115 non-null   object 
 2   Assortment                 1115 non-null   object 
 3   CompetitionDistance        1112 non-null   float64
 4   CompetitionOpenSinceMonth  761 non-null    float64
 5   CompetitionOpenSinceYear   761 non-null    float64
 6   Promo2                     1115 non-null   int64  
 7   Promo2SinceWeek            571 non-null    float64
 8   Promo2SinceYear            571 non-null    float64
 9   PromoInterval              571 non-null    object 
dtypes: float64(5), int64(2), object(3)
```

There are missing values in ‚store‘. Then we‘ll handle those Nan‘s in ‚Data preprocessing‘.
As we see in the feature description the object column would be also converted to numeric values.

Exploratory Visualization

for dataset ‚train‘ we could have a glance at the distribution of Sales and Customers by plotting ECDF (empirical cumulative distribution function).

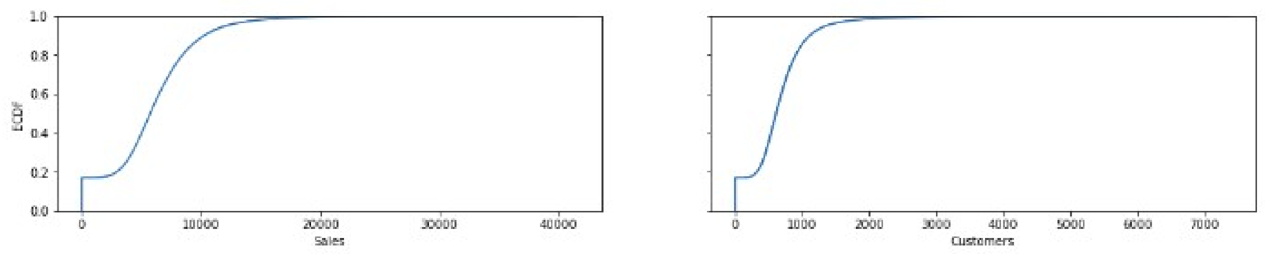


Fig. 1 ECDF of Sales and Customers

There are about 20% values in ‚Sales‘/‚Customers‘ equals zero. These values should be excluded from model training. We only take the records with no zero `Sales` into our modeling.

How is the impact of ‚Promo‘ on ‚Sales‘ ?

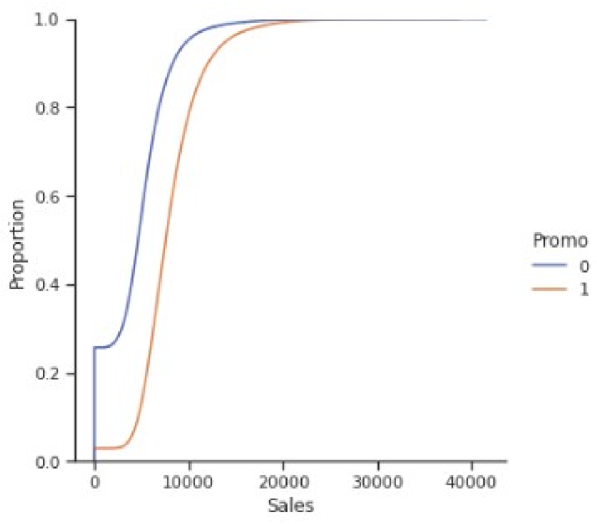


Fig. 2 ECDF of Sales for Promo and No-Promo

Obviously on Promo day they have two different trend and more daily Sales. Thus the ‚Promo‘ is an important predictive feature to build the model.

After merging the dataset ‚store‘ to ‚train‘ on the feature ‚Store‘ Id. We could get an overall visualization.

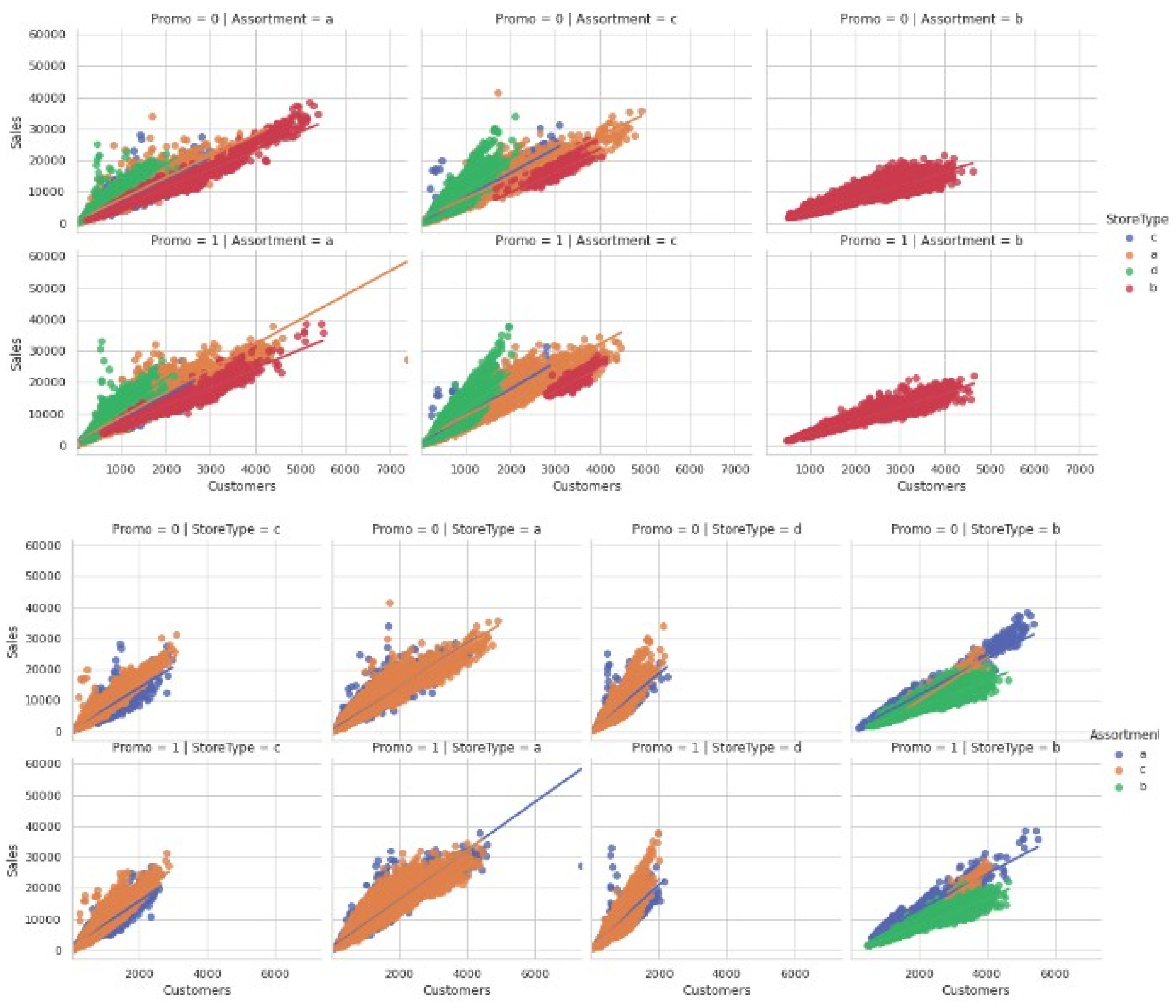


Fig. 3 ‚Sales‘ per Customer for different Assortment and StoreType

From Fig 3. we could have a general inference that for Assortment a and c different StoreType displays a different average Sale per Customer. Besides on Promo day people intend to pay more.

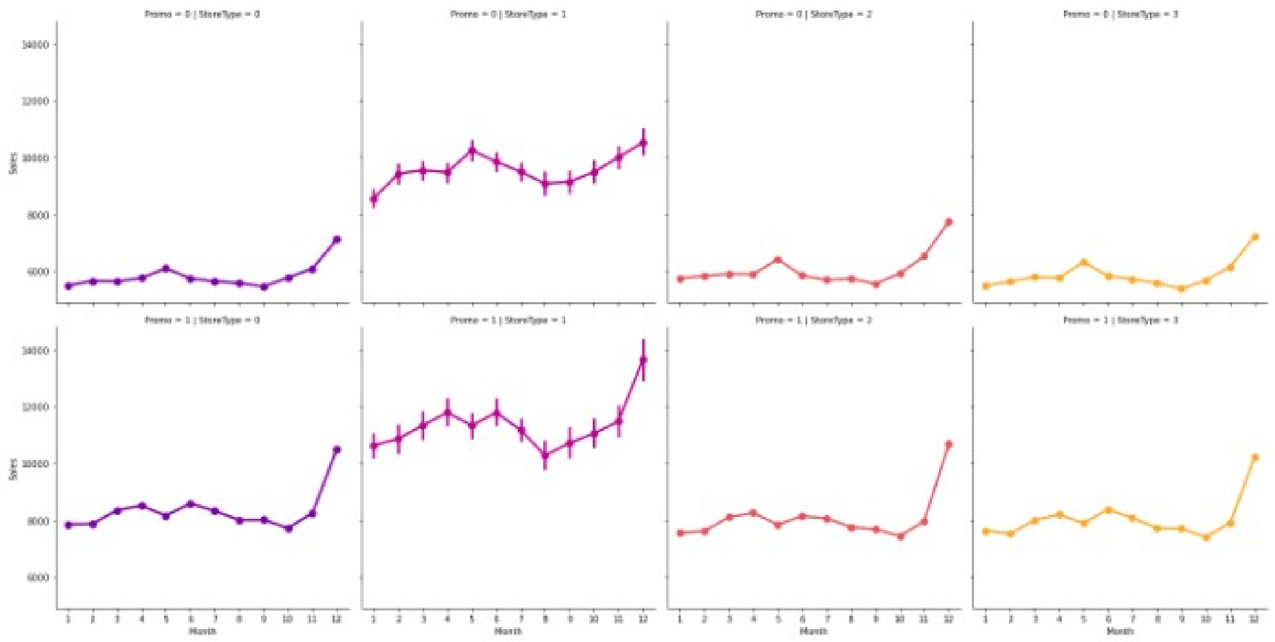


Fig. 4 Monthly trend of ‚Sales' for different StoreType on Promo and No-Promo day

All store types despite of the runing promotion follow the same trend but at different scales depending on the presence of ‚Promo' and `StoreType` itself. Already at this point, we can see that Sales escalate towards Christmas holidays and reach a peak at the year end.

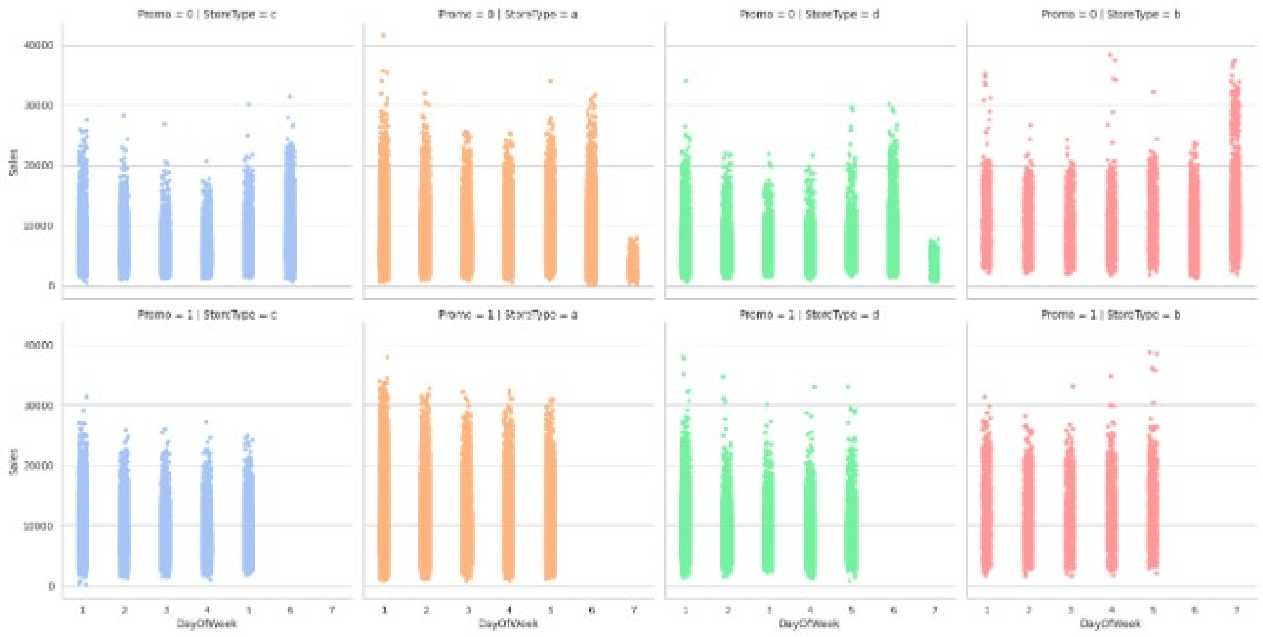


Fig. 5 Daily trend in a week of ‚Sales'

All Store type don't open on weekend in case of a ‚Promo' (equals 1). For StoreType d ‚Sales' tends to peak on Sunday under no ‚Promo'.

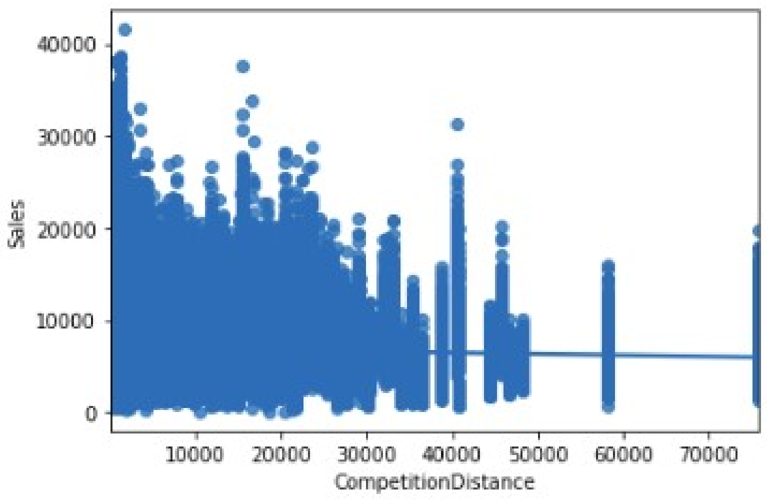


Fig. 6 Relationship between ‚Sales‘ and ‚CompetitionDistance‘

It can be seen that ‚CompetitionDistance‘ has a slight negative impact on the ‚Sales‘, but the trend is not obvious.

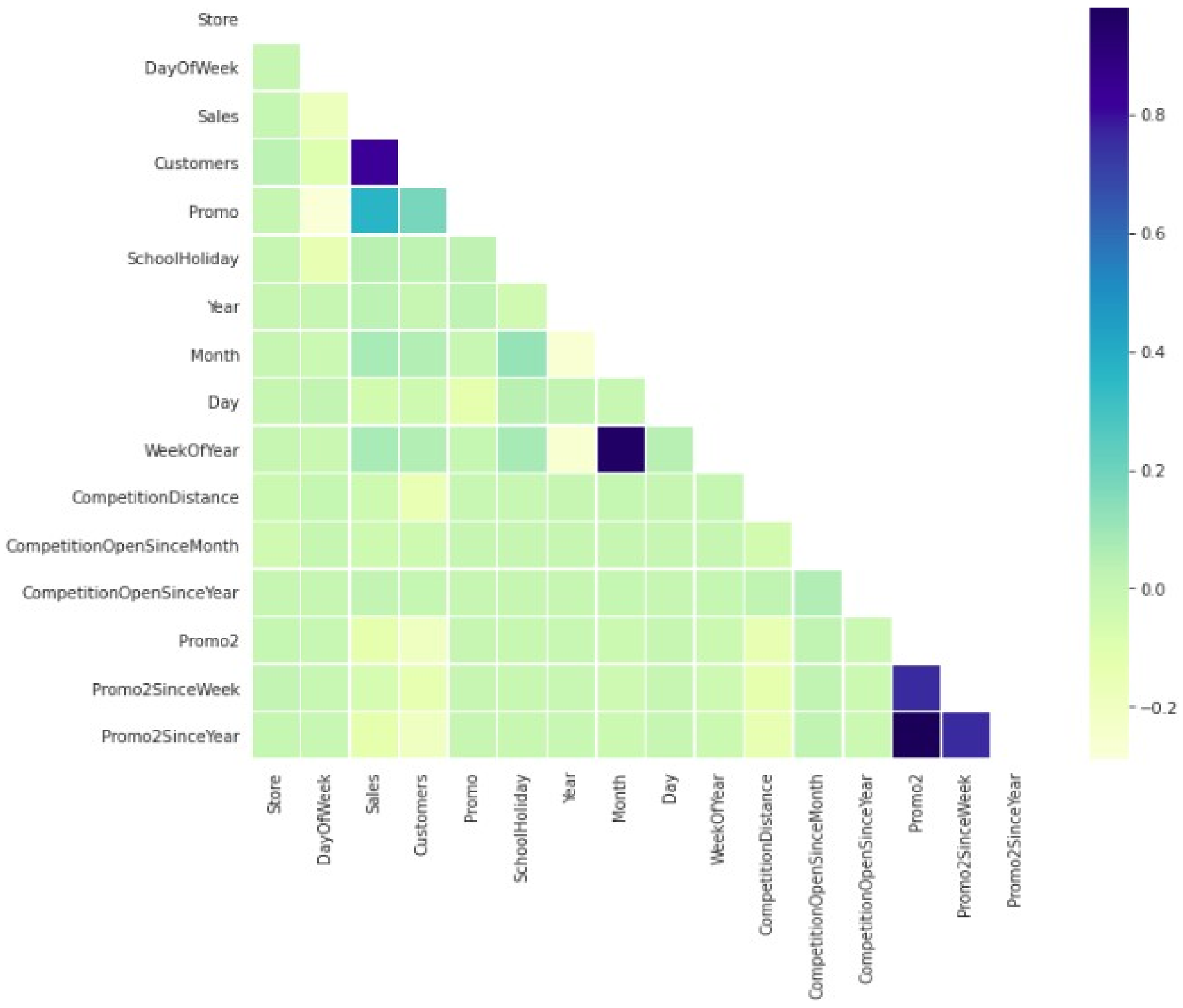


Fig. 7 Correlation heatmap

As mentioned before, we have a strong positive correlation between the amount of Sales and Customers of a store. We can also observe a positive correlation between the fact that the store had a running promotion (‚Promo‘ equal to 1) and amount of ‚Customers‘.

However, as soon as the store continues a consecutive promotion (‚Promo2‘ equal to 1) the number of ‚Customers‘ and ‚Sales‘ seems to stay the same or even decrease, which is described by the pale negative correlation on the heatmap. The same negative correlation is observed between the presence of the promotion in the store and the day of a week.

Besides we could infer a fact of marketing campaign that in case of a short distance of Competitions the store tend to run a consecutive promotion.

Benchmark

As we can see from the Notebooks many solutions has been shared. Competiters has applied XGBoost, Random Forest, time series model even linear Regression. Generally, the most accurate predictions ware made by XGBoost. Due to the test dateset the private leaderboard is stricter. There are totally 3298 results on private board.

The best score on private board is 0.10021. My goal for this project is to obtain a results that at least achieves 0.12 ranked at about top 20%. Of course the smaller score RMSPE was achieved the better modeling I‘ve made.

Methodology

Data Preprocessing

As we‘ve done the process data exploration and Visulization, some preprocessiong were already done:

1. Remove the records in ‚train‘ whose ‚Sales‘ equal to 0.
2. Replace the NaN‘s in columns ‚CompetitionDistance‘, ‚CompetitionOpenSinceMonth‘ and ‚CompetitionOpenSinceYear‘ in dataset ‚store‘ with their own mode(), the most commen values.
3. Merge dataset ‚store‘ to ‚train‘ on the store Id.

Then we do some work on the features:

1. Engineer two features: ‚CompetitionOpen‘ (How long the competition has already been open) and ‚PromoOpen‘ (how long the consecutive promo has been on). This two features should be identified and calculated from other features.
2. Convert the column ‚PromoInterval‘ to ‚IsPromoMonthh‘(if the current month launches a consecutive promotion.
3. Convert the object columns to numeric through mapping.

Implementation

Split the dataset into train and validation set. 30% records are reserved to do the validation during training.

This notebook focuses on Xgboost and LightGBM these two models. In oder to verify the quality of the modeling a linear Regression was conducted to the purpose of comparison.

Applying the **Linear Regression** the model has just a RMSPE of 0,439822 on the reserved validation dataset. This value is too bad to be considered in this forecast problem. Hence, in the next chapter I‘ll only discuss XGBoost and LightGBM.

XGBoost is a scalable machine learning system for tree boosting. From the aspect of regularization, Decision tree ensembles, Tree boosting, Model Complexity, Structure Score and so on these principles in machine learning XGBoost has been developed with deep consideration in terms of systems optimization. [2]Thus with a base paramter set after a 3212 rounds we get a trained model whose RMSPE on validation dataset is 0.098462.

A benefit of using gradient boosting is that after the boosted trees are constructed, it is relatively straightforward to retrieve importance scores for each attribute. Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

XGBoost provide a built-in function to plot features ordered by their importance. The function is called plot_importance()

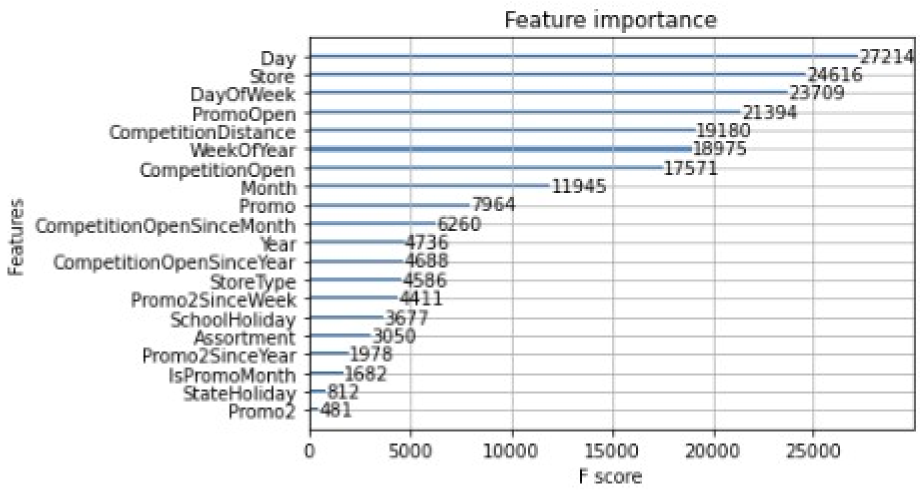


Fig. 8 Feature importance of the trained XGBoost model

LightGBM has a benifit on efficiency and accuracy compared with XGBoost. Many boosting tools use pre-sort-based algorithms(e.g. default algorithm in xgboost) for decision tree learning. It is a simple solution, but not easy to optimize [4].

LightGBM uses histogram-based algorithms, which bucket continuous feature (attribute) values into discrete bins. This speeds up training and reduces memory usage. Advantages of histogram-based algorithms include the following [5]:

- Reduced cost of calculating the gain for each split
- Use histogram subtraction for further speedup
- Reduce memory usage
- Reduce communication cost for parallel learning

By providing a base parameter set I‘ve trained a LightGBM model I‘ve trained a model whose RMSPE on validation set reaches 0.098993.

From the side of efficiency training a XGBoost model with 3212 rounds takes 2517.76 s. On contrary, a LightGBM model trained with 11780 rounds just spends 565.19 s. Due to such a disparity on training efficiency I‘ll apply the LightGBM to do the further exploration - Refinement.

Refinement

The Refinement process can be split into two main stages:

1. Parameter tuning: applying the Scikit learn wrapper GridSearchCV or RandomizedSearchCV to search for a more appropriate Parameter Set for modeling.
2. Weight Correction by error

After the parameter tuning although the RMSPE score on training and validation set is slightliy lower than base LightGBM model, but it performes better on kaggle competition board. I‘ll list the result on private and publich board in the next chapter: Model Evaluation and Validation.

Based on stage 1 we could then plot the correction weight and the corresponding RMSPE score on validation dataset. In this way a valley of the error is easily identified. This weight could be used in improve the results.

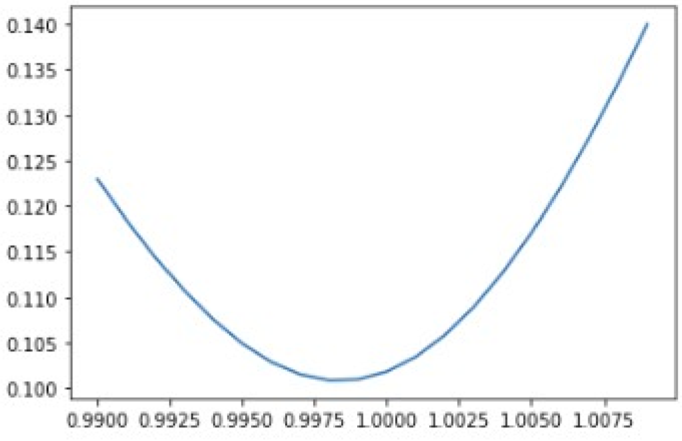


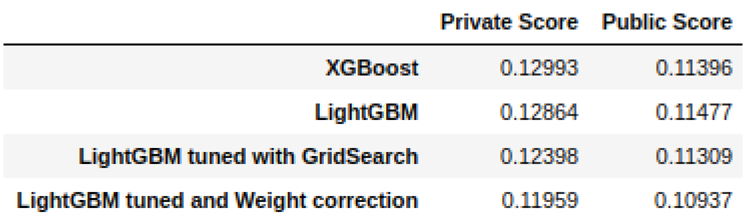
Fig. 9 Correlation between correction weight and RMSPE

Results

Model Evaluation and Validation

Kaggle provieds Leaderboard to do the validation of modeling. There are two leaderboards: private and public, distinuished from different dataset. The private leaderboard is calculated with approximately 67% of the test data while the public with approximately 33% of the test data.

After some precessing of the test dataset provided by the competition a result was generated to be uploaded on kaggle. The results are displayed in fig. 10:



	Private Score	Public Score
XGBoost	0.12993	0.11396
LightGBM	0.12864	0.11477
LightGBM tuned with GridSearch	0.12398	0.11309
LightGBM tuned and Weight correction	0.11959	0.10937

fig. 10 Comparion of the results

Firstly two models were trained with XGBoost and LightGBM with their own base parameter. The RMSPE on the hold-out validation dataset are 0.098462 (XGBoost) and 0.098993 (LightGBM). LightGBM is even slightly worse. However, The LightGBM performs better on the unseen dataset on private leaderboard. Then we take LightGBM as base approach model to do the improvement. After paramater tuning the private score has reduced to 0.12398. Then with the weight correction I‘ve succeeded reducing the score under 0.12.

Justification

As we see the previous results from the leaderboard. My score 0.11959 is ranked at 494 out of 3298 competitors, about top 15% on private board.

As the introduction in chapter Implement XGBoost has applied a series of methods on gradient boosted trees to improve the performance in terms of systems optimization. And LightGBM continues to do the improvement on the efficiency, accuracy and memory usage.

From the training time and the accuracy of the two models applied to solve the kaggle competition we could get a first impression of the difference on performance between XGBoost and LightGBM.

The score on private leaderboard was calculated with approximately 67% of the test data which is unseen by the training. The score is not far from the score on validation dataset. Thus we could conclude that the obtained solution is robust.

Conclusion

Reflection

At the beginning I thought a time series were provided. Beside XGBoost the Times Series Analysis using ARIMA or Prophet maybe a good approach as a solution. However, even in the split step by splitting the dataset into train and validation set not by random I‘ve used this thought and reserve the last 6 weeks as hold-out set to do the validation. The RMSPE on the hold-out validation set could only be reduced to 0.132, even worse then the randomly splitting. So I‘ve given up the solution from time series.

Also I‘ve encounted the problem of overfitting. By importing the feature ‚PromoDaysperweek‘ although the score on training dataset could reduce to 0.089, but on validation set it could only reach 0.244. This model is bad on generalization on the unknown dataset.

Then as the score is getting smaller, the more difficult to improve the results further. Some competiters even introduced other infomation to improve the results e.g. the weather Germany.

Improvement

Though, there are still potential to get a better forecasting.

1. The weight could be further more calculated through stores.

2. Instead of RadomsizedSearchCV for searching the approapriate parameter set the Grid SearchCV could be used. But it takes more resource, time and computaion and so.

3. The ensembling model is also worth a trial.

References:

1. https://en.wikipedia.org/wiki/Data_mining
2. Tianqi C. and Carlos G: XGBoost: A Scalable Tree Boosting System
3. Guolin K., Qi M. Thomas F. And Taifeng W. : LightGBM: A Highly Efficient Gradient Boosting Decision Tree
4. J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189–1232, 2001.
5. https://lightgbm.readthedocs.io/en/latest/index.html
6. https://xgboost.readthedocs.io/en/latest/index.html