Project Report: Forecast Rossmann Store Sales

Project Overview

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.

In their first Kaggle competition, Rossmann is challenging you to predict 6 weeks of daily sales for 1,115 stores located across Germany. Reliable sales forecasts enable store managers to create effective staff schedules that increase productivity and motivation. By helping Rossmann create a robust prediction model, you will help store managers stay focused on what’s most important to them: their customers and their teams!

Project Statement

Sales Prediction is essentially a data regression problem. The dataset has two files:

- Train.csv -- the historical data including Sales. More concretely, it’s a Sale Record per store per day within a period of time, with 8 input features.
- Store.csv -- supplementary information about the stores. It provides 9 features to describe a store

Our target is to predict the Sales by store by day in future 6 weeks, based on the information above.

I will play with the dataset, trying to identify those important features, and choose a appropriate model to fit and make prediction. Since this is a competition on Kaggle (two years ago), I’ll submit my prediction result to Kaggle to do the evaluation. My plan is to reach the top 10% of the rank list, which means the result score will reach 0.11773 or less.

Metrics

Submissions are evaluated on the Root Mean Square Percentage Error (RMSPE). The RMSPE is calculated as:

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

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.

Submission File

The file should contain a header and have the following format:

```
Id,Sales
1,0
2,0
3,0
etc.
```

Analysis

Data Exploration

The dataset is consist of 1,017,209 sales records from 1115 stores, from 2013-01-01 to 2015-07-31. Original 8 features and Sales are:

Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1	5	7/31/2015	5263	555	1	1	0	1
2	5	7/31/2015	6064	625	1	1	0	1
3	5	7/31/2015	8314	821	1	1	0	1

There are also additional 9 features in store.csv:

Store	StoreType	Assortment	Competition-Distance	Competition-OpenSinceMonth	Competition-OpenSinceYear	Promo2	Promo2 SinceWeek	Promo2 SinceYear	PromoInterval
1	c	a	1270	9	2008	0			
2	a	a	570	11	2007	1	13	2010	Jan,Apr,Jul,Oct
3	a	a	14130	12	2006	1	14	2011	Jan,Apr,Jul,Oct

Feel free to check etailed description of each column located at:
https://www.kaggle.com/c/rossmann-store-sales/data

Test data has 41,088 records of the same set of stores. Date range from 2015-08-01 to 2015-09-17:

Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
1	1	4	9/17/2015	1	1	0	0
2	3	4	9/17/2015	1	1	0	0
3	7	4	9/17/2015	1	1	0	0

So this project is a problem of “from sales data of 2.5 years to predict sales of next 1.5 months, by store by day”.

First impression:

1. DayOfWeek is part of Date, so it’s a redundant feature.
2. Customers feature is not in test.csv, so it needs to be transferred in some way to have that information in test data.
3. Features have Null values:
   a) Test.Open
   b) Store.CompetitionDistance
   c) Store.CompetitionOpenSinceYear
   d) Store.CompetitionOpenSinceMonth
   e) Store.Promo2SinceYear
   f) Store.Promo2SinceWeek
   g) Store.PromoInterval

   These null values need to be filled.
4. Categorical data need to be encoded:
   a) Train.StateHoliday
   b) Store.StoreType
   c) Store.Assortment

   These features are all of values “0, a, b, c, d”, and need to be translated to numerical data and then One-Hotted.

Methodology

To me Machine Learning is more of a practical approach than a theoretical approach. Like Andrew Ng said: built a model quickly and make prediction, then see what’s missing and what can be improved. It’s hard to plan every step in advance.

So in this project, I’m going to do some basic analysis of the data, choose some features, do data preprocessing and make prediction, then start tuning from there. Thoughts are verified by uploading the result to kaggle and see the results.

Feature Analysis and Choose

In a sales prediction context, I consider the features to be in three parts:

- Daily Sales data
- Store inherent features
- Past recent sales trend for each store

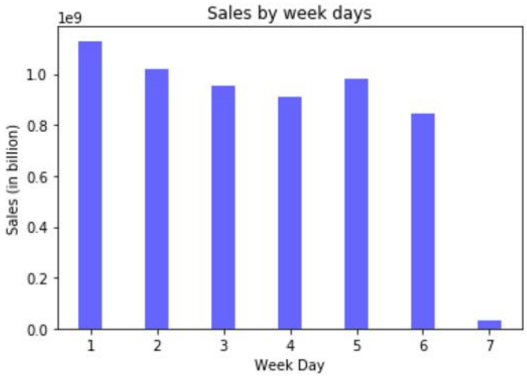
Let's discuss each type in detail:

1. **Daily Sales Data**

   Mostly came from train.csv.

   i. **Store**: just id of stores. Essentially this is a categorical data instead of a numerical data, because Store "889" doesn't mean anything bigger than Store "20". However we won't do one-hot to this field, instead, we use it just to link to Store data. We will more inherent store features later, after that we will remove the Store feature.

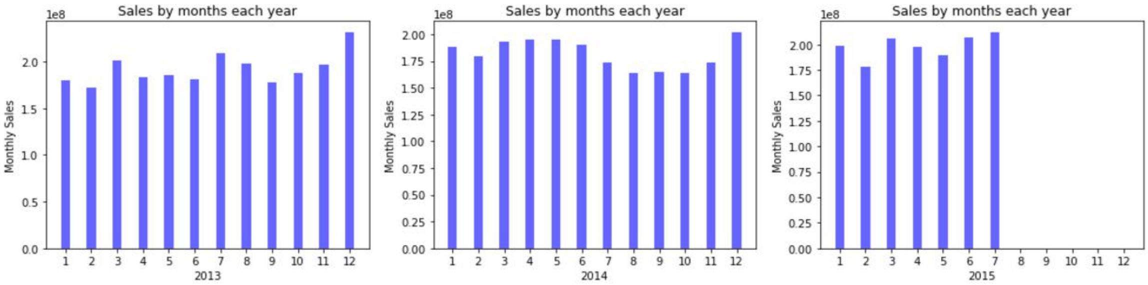
   ii. **DayOfWeek**: based on our experience DayOfWeek is a important factor to sales. Normally people bought things regularly each week:



We can see most stores closed on Sunday. This also implies when fill null values for Open in test data, we can set it to 0 when it's Sunday, otherwise set to 1.

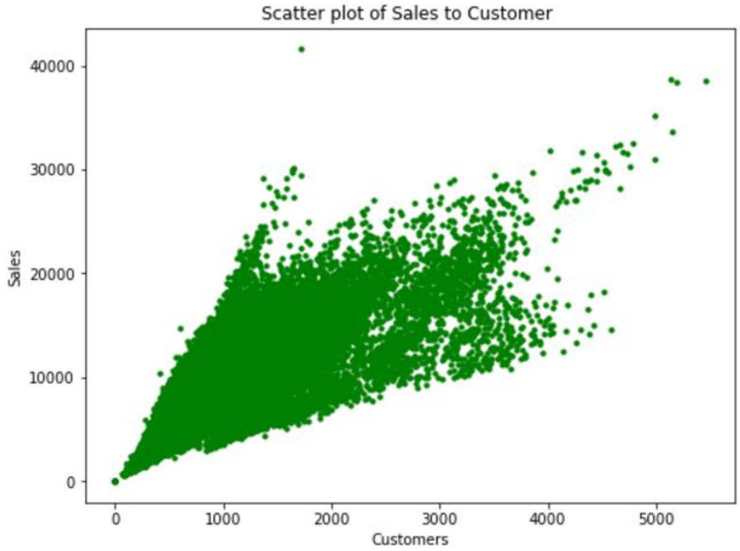
However, DayOfWeek can get from Date field thus is a redundant feature.

   iii. **Date**: Date is important for Sales in many ways. It can infer DayOfWeek, and it can also get many other numbers like Year, Month, Day, WeekOfYear etc. And Month/WeekOfYear to some extent includes some hidden factor like "season" and "weather", which should also be important for sales



We can see the monthly sales in each year is sort of repeatable. They look similar.

iv. **Customer**: let's see the relationship between Customers and Sales --

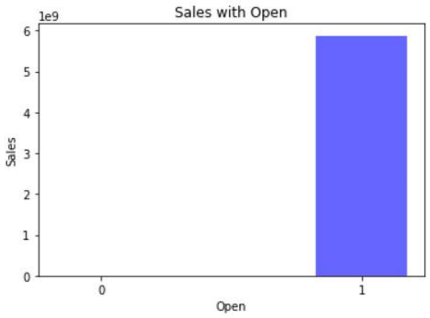


As expected, Customer is highly positively correlated with Sales. What I don't expect is the correlation is not very linear: Sales differs a lot even with same amount of Customer (like the Sales for 1000, 2000 and 3000 customers). This implies that each Customer spent different amount of money in different stores. This can be viewed as different regions of the Stores: some stores lives in rich zone and some lives in relatively poor zone.

However, there is no Customer feature in the test data, it's maybe another number we want to predict. Here I tried to transfer the Customer into two inherent features of Store:

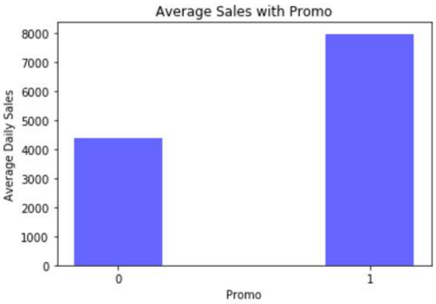
1. SpendPerCustomer -- the average spend of each customer for each store. Formula is: Sales/Customers, then calculate the mean SpendPerCustomer for each store.
2. AverageCustomers -- the average number of customer for each store. Formula is: train_data.groupby('Store')['Customers'].mean()

v. **Open**: whether a store is open on that day.



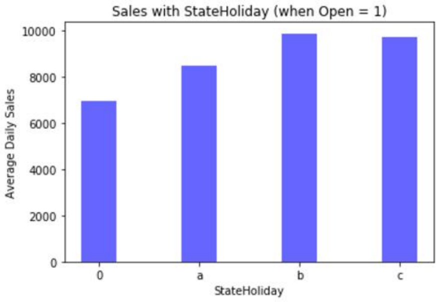
When a store is close there is zero sales. We don’t need to fit and predict for those days when stores are closed. I’ll just set Sales to be zero in test data when Open=0.

vi. **Promo**:



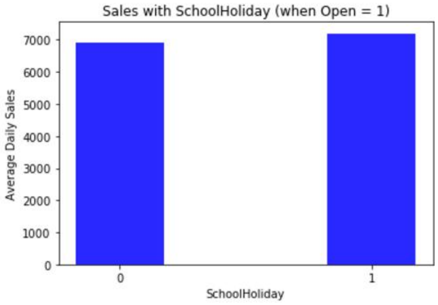
Promotion also has remarkable impact on sales.

vii. **StateHoliday**:



StateHoliday also impacts sales, not as big as promotion though.

viii. **SchoolHoliday**:



SchoolHoliday does not seem to have apparent impact on sales.

2. Store Inherent Features

Features in Store.csv:

i. **Store**: used to merge with trainning data. Skip it.

ii. **StoreType and Assortment:**

Two categorical features. I'll do encoding and one-hot on both but will skip the exploration here.

iii. **CompetitionDistance**:

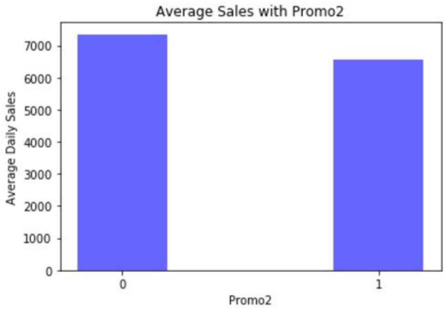
The distance in meters to the nearest competitor store. For this feature, 3 out of 1115 stores are null. There 3 ways to fill null:

1. 0 -- I don't think it's good, 0 means there is a competitor in next door
2. Average CompetitionDistance. It's nature since null may mean "no information"
3. A big value: this look at null as "no competitors nearby"

Option 2 and 3 need to test and see. Based on result option 3 is slightly better (increase kaggle score around 0.001), so I'll use option a bigger value (2 * max) as to fill null value.

iv. **CompetitionOpenSinceYear and CompetitionOpenSinceMonth**: it makes more sense to translate them into number of months of competition

v. **Promo2**: a continuing promotion for some stores it has less impact to Sales than Promo:



vi. **Promo2SinceYear**, **Promo2SinceWeek**: same as CompetitionOpenSinceYear and CompetitionOpenSinceMonth, caculate the number of months since Promo2, then remove these features

vii. **PromoInterval**: This feature is confusing. It describes from which months the Promo2 interval started. But it's not clear how long each Promo2 will last. After referring to some discussion, I just translate it to a boolean feature (0/1) to describe if current month is the Promo2Month or not, then fill null values as 0. However this feature is still useful. Based on my testing, having this feature will increase about 0.0015 on the final score on Kaggle than not having this feature.

Checkpoint:

At this point I finished the first batch of feature design. After merging store data into trainning set and test set, I got below feature list on both:

['DayOfWeek', 'Open', 'Promo', 'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance', 'Promo2', 'Month', 'Day', 'WeekOfYear', 'MonthsOfCompetition', 'MonthsOfPromo2', 'SalesPerCustomer', 'AverageCustomer', ‘IsPromoMonth’]

Now let’s see the algorithm and we can start make some prediction and refine from that.

Algorithm

I, as most of other competitors, quickly choose XGBoost. The reason is:

1. Everyone is using XGBoost in this kind of regression problem, there must be reasons. This is probably not a best reason but it is a important reason in fact.
2. The only other algorithm I tried is Random Forest with Grid Search. The fitting is slow and result is not good. With same data preprocessing:

Algorithm	Kaggle Public Score	Kaggle Private Score
Random Forest + Grid Search	0.30130	0.30404
XGBoost	0.12366	0.13949

I’m sure there are space to improve for Random Forest, however it’s enough for me to choose XGBoost.

I used Grid Search for XGBoost hyper-parameters, and also referred to other people’s suggestion, and come up with below parameters:

```
params = {
	'objective': "reg:linear",
	'booster': 'gbtree',
	'n_estimators': 20,
	'max_depth': 10,
	'subsample': 0.9,
	'colsample_bytree': 0.7,
	'slient' : 1.0,
	'n_jobs' : 16,
	'eta' : 0.3
}
```

Also I used early stop of 100 rounds to avoid overfitting.

Evaluation Metric

This competition use RMSPE as the final score instead of RMSE, the formula is:

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

Comparing with RMSE, the main difference is the difference between real value(Yi) and predicted value(Yi hat) is divided by Yi. This is understandable in this case, since different stores may have huge different sales volume, and only comparing the absolute value of difference may not make sense. For example, Store 1114 has average daily Sales of 17200, while Store 22 only has 3695. If we predict 1114 a sale of 12000, the difference is even bigger than entire daily sales of Store 22. So using a “percent” difference is better.

However, the problem is RMSPE is not supported out of box as objective function in most Machine Learning Algorithm, such as XGBoost. So I write a customized evaluation metric function to calculate the RMSPE.

The result of fitting is as below:

```
Will train until valid-rmspe hasn't improved in 100 rounds.
[25]    train-rmse:679.078      valid-rmse:735.867      train-rmspe:0.011326    valid-rmspe:0.010405
[50]    train-rmse:614.658      valid-rmse:687.256      train-rmspe:0.009266    valid-rmspe:0.009256
[75]    train-rmse:583.966      valid-rmse:673.007      train-rmspe:0.008625    valid-rmspe:0.008958
[100] train-rmse:560.385        valid-rmse:660.384      train-rmspe:0.008138    valid-rmspe:0.008655
Stopping. Best iteration:
[10]    train-rmse:820.006      valid-rmse:857.713      train-rmspe:0.000428    valid-rmspe:0.002013
```

We can see that by default xgboost will use rmse together with my rmspe function as evaluation function. Since rmse is absolute value and rmspe is a ratio value (0-1), they looks very different.

One way is to scale/normalize the sales. But there’s a better solution, based on the discussion on: https://www.kaggle.com/c/rossmann-store-sales/discussion/17026 , a log transformation on Y value will make the RMSE approximately equal to RMSPE by a Taylor series expansion.

In practise I use y = log(sales + 1), just to prevent sales to be zero or very small number and there will be undefined or accuracy issue. In my case the zeroes sales are all filtered out so it makes no difference whether to plus 1.

After applying the log transformation the fitting result looks much better:

```
with log: 181 seconds, they are much similar, and best score is 20 times better
[25]    train-rmse:0.098481     valid-rmse:0.099115     train-rmspe:0.004473    valid-rmspe:0.001467
[50]    train-rmse:0.090241     valid-rmse:0.093184     train-rmspe:0.004418    valid-rmspe:0.001844
[75]    train-rmse:0.086384     valid-rmse:0.09129      train-rmspe:0.003965    valid-rmspe:0.001824
[100]  train-rmse:0.082592      valid-rmse:0.089829     train-rmspe:0.003585    valid-rmspe:0.00168
Stopping. Best iteration:
```

```
[22]   train-rmse:0.101183      valid-rmse:0.101493      train-rmspe:0.003238    valid-rmspe:0.000124
```

The score is better, and fitting is little bit faster than before. The differences are:

Algorithm	Fitting time	Kaggle Private Score
Without log transformation	189 seconds	0.11829
With log transformation	170 seconds	0.11351

With all the information above, let's summarize the implementation and see the results.

Implementation

Data Preprocessing

1. Training data:
   a) Date: extract Year, Month, Day, DayOfWeek, WeekOfYear from Date, then remove Date feature
   b) Customer: calculate SpendPerCustomer and AverageCustomers for each store, and add the two features to store
   c) remove all training data where sales is zero
   d) merge store data into training and test data by store id
   e) Null values:
   f) CompetitionDistance: set to a big value (max(CompetitionDistance) * 2)
   g) CompetitionSinceYear, CompetitionSinceMonth: fill 0, then compute the MonthsOfCompetition then remove both
   h) Promo2SinceYear, Promo2SinceWeek: fill 0, caculate MonthsOfPromo2, then remove both
   i) PromoInterval: fill empty string, calculate if current day isPromo2Month then drop it

2. Test data:
   a) Open: fill null value to 1(open)
   b) remember the test data lines where Open=0, after prediction add them back as Sales=0

3. Categorical data:
   a) StateHoliday, StoreType, Assortment
   b) Do one-hotting to above features

4. Doing log transformation to Sales

Fit and prediction

Splitting training/validation sets, fitting the data with parameters listed above with XGBoost, and make prediction(don't forget to transform back the prediction by exp(y_pred) - 1 since we did log transform on y). I got the first result.csv. Uploading to Kaggle, score is: 0.12741

The score is not bad! However, our target is 0.11773, there are still way to go. Let's see how we can refine the model and improve the score.

Refinement and Result

1. In above feature I used **SpendPerCustomer** and **AverageCustomers**, but both feature are by Store, this might be too generic. We can imagine that even for each Store, the spend of each customer may differ among weekday or weekend, holiday or non-holiday, and people will spend more when there are good promotions. Same for number of customers, during weekend and holiday and promotion, there are more customers than normal. So the two featuers should be calculated in better granularity: besides by Stores, also by DayOfWeek, StateHoliday and Promo.

   After applying this change, my score increased from 0.12741 to 0.11817. We are getting closer!

2. We haven't take into account another factor: the recent sales trend for each store. In a time series data, the trend of data is usually a important factor. In our case, I looked at each store sales as constant over 3 years, but that's often not the case. Many external factors may impact the sales for a store in a period of time, and the sales in the recent past period of time may reveal that impact.

   There are different choices of past trend data: average sales of past one month, one quarter or half an year. I choosed the sales of last quarter as suggested by Gert (Champion of the competition).

   After adding the **SalesLastQuarter** feature to each store, my score increased to 0.11503! I hit the target! But let's see if I can improve it a little bit further.

   The last feature list I'm using is:

   ```
   ['DayOfWeek', 'Open', 'Promo', 'StateHoliday', 'SchoolHoliday',
        'StoreType', 'Assortment', 'CompetitionDistance', 'Promo2', 'Month',
        'Day', 'WeekOfYear', 'MonthsOfCompetition', 'MonthsOfPromo2',
        'IsPromoMonth', 'SalesPerCustomer', 'AverageCustomer',
        'SalesLastQuarter']
   ```

3. There is a issue in **SalesLastQuarter** above. Look at the test data, they are from 2015-08-01 to 2015-09-17. I don't have the sales for this period (of course), so what's the SalesLastQuarter of test data?

I just used the most recent quarter data I have: 2015-05 to 2-15-07. That lead to all the SalesLastQuarter in test data are the same (for each store). So I realized that we have the sales prediction from first round and we can just use these predictions to re-calculate the SalesLastQuarter of test data. This idea is implied by Collaborating Filter approach in Recommendation algorithm.

Now after using the first round prediction to calculate the SalesLastQuarter and redo the prediction, my score increase to 0.11264!

Conclusion

During the entire process of this project, I spent most of my time in feature engagement, trying to find out the best feature design, the relationship behind among them and Sales. I also referred to discussion on Kaggle competition board, including Girt and others, they gave me lots of hints and help.

Look back all the features, they are in three groups:
1. the Store inherent attributes: Store type, AverageCustomers, SpendPerCustomer, competitors information etc.
2. External factors: DayOfWeek, Promotion, StateHoliday, SchoolHoliday, Month(season --> Weather)
3. Past trend of data

These types of features is general among the sales prediction problems and may transferred to other projects.

As future improvement, normally Ensemble is a way to improve the score, which means to design/train 5-12 different models and average on their results. Girt(champion) also used this trick. However Ensemble normally consumes much larger computation capabilities which I don't have right now, and is rarely used in real project for the same reason.

For data that of time series, there is another entire different model to do the prediction -- Recurrent Neural Network/LSTM. I'll try to explore that direction later.

XGBoost does a execllent job in both fitting performance and prediction accuracy. It's the best framework as of now for this kind of regression problem.