Capstone Project 2

Final Report

Title: Rossmann Store Sales Prediction

Cuthbert Lo

Jun 21 , 2018

Table of Contents

1.Introduction

This is problem is one of the competition on Kaggle to use machine learning to forecast future 6 weeks sales of 1,115 drug stores of Rossmann across Germany based on their historical data. The practical usage of solution to this problem is to enable store managers to create effective staff schedule that increase productivity and motivation as planning for stock level as well.

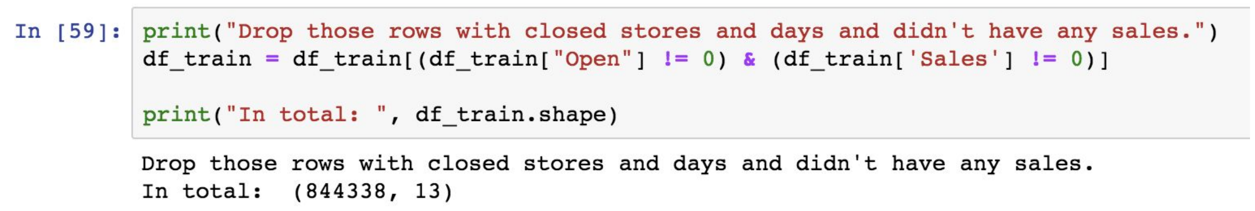
2. Dataset

The dataset is all provided by Kaggle. There are mainly two files train.csv and store.csv, which contain historical sales data and supplementary store information respectively. The dataset contains data from 1,115 stores from 2013-01-01 to 2015-07-31, a total of 31 months and about 1 million data points.

3. Data Cleaning and Wrangling

There are two files train.csv and store.csv we need to combine two files into one, however before we combine, we will do the following. In train.csv, Split the Date field into three separate fields namely year, month and day, add one more features WeekOfYear.

By exploring the data, there are some row that the store is open but sales is zero which doesn't make sense, so we need to drop those row.



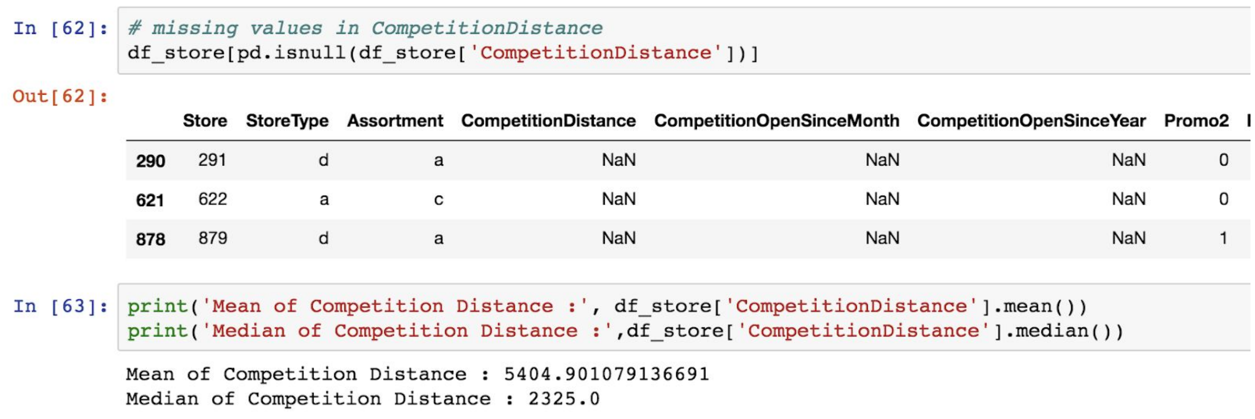
```
In [59]: print("Drop those rows with closed stores and days and didn't have any sales.")
         df_train = df_train[(df_train["Open"] != 0) & (df_train['Sales'] != 0)]

         print("In total: ", df_train.shape)

         Drop those rows with closed stores and days and didn't have any sales.
         In total:  (844338, 13)
```

Fig 1. Dropping rows with closed store and days didn't have sales

In store.csv, there are some missing values in CompetitionDistance. We don't want to drop those valid row as there is not obvious pattern of those missing data, those rows will be replace with the median.



```
In [62]: # missing values in CompetitionDistance
         df_store[pd.isnull(df_store['CompetitionDistance'])]

Out[62]:
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2
290	291	d	a	NaN	NaN	NaN	0
621	622	a	c	NaN	NaN	NaN	0
878	879	d	a	NaN	NaN	NaN	1

```
In [63]: print('Mean of Competition Distance :', df_store['CompetitionDistance'].mean())
         print('Median of Competition Distance :',df_store['CompetitionDistance'].median())

         Mean of Competition Distance : 5404.901079136691
         Median of Competition Distance : 2325.0
```

Fig 2. Filling median distance to missing competition distance record

Finally, we replace all NA by zero then join train dataframe and store dataframe together.

4. Data Exploration

There are more than 800k rows of data with 24 columns for 31 months of data.

4.1 Store sales

There are four types of store and three level of assortment of store across 1,115 stores. The Type b store has the highest average sales, and Assortment level b has the highest average sales.

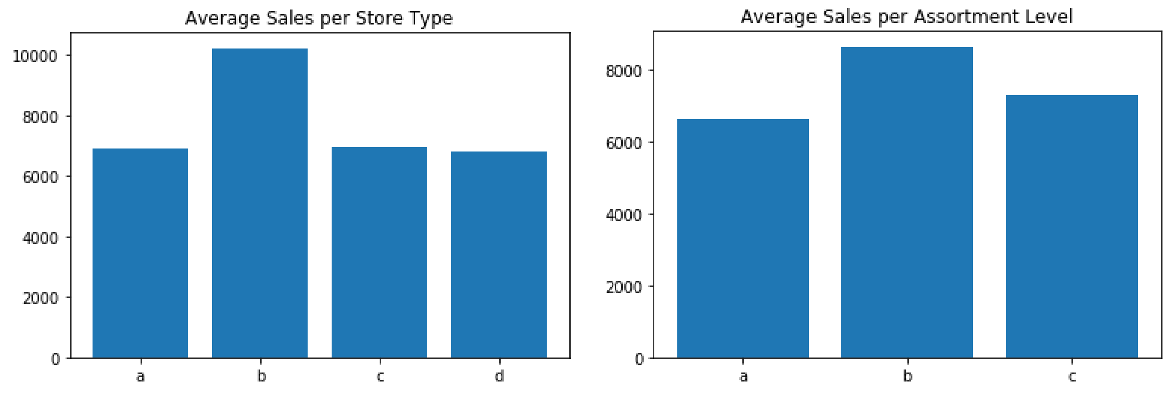


Fig 3. Average Sales for Store Type and Assortment Level

Holidays are one of the main aspects that affect the sales, the dataset categorized holidays into 4 groups a = public holiday, b = Easter holiday, c = Christmas, 0 = None. From Fig 4 below, state holidays are definitely increase sales, public holiday has a lower sales than Easter holiday and Christmas while Christmas has slightly higher than Easter.

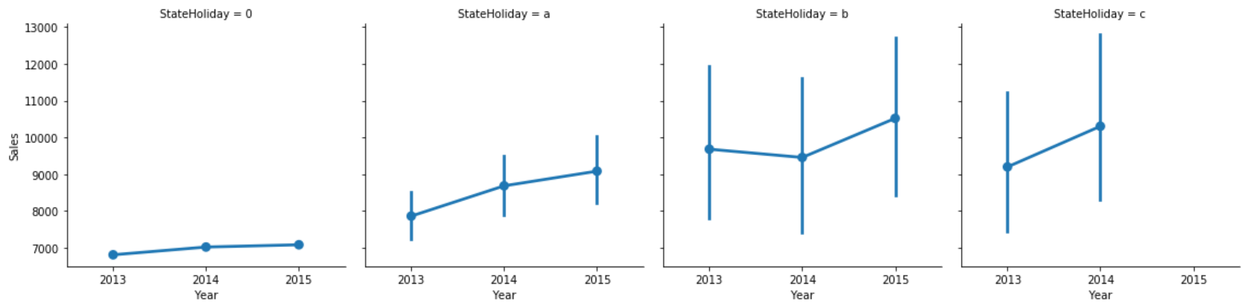


Fig 4. Average Sales by Holidays categories

Referring to Fig 5 figure, the upper row represent sales without promotion and lower row is sales with promotion, columns are different type of stores. We can see the average daily sales amount by month significantly increased. And we can see the sales of months toward to end of year ramp up across all store types. And this also confirm that store type b (red line) has higher average sales.

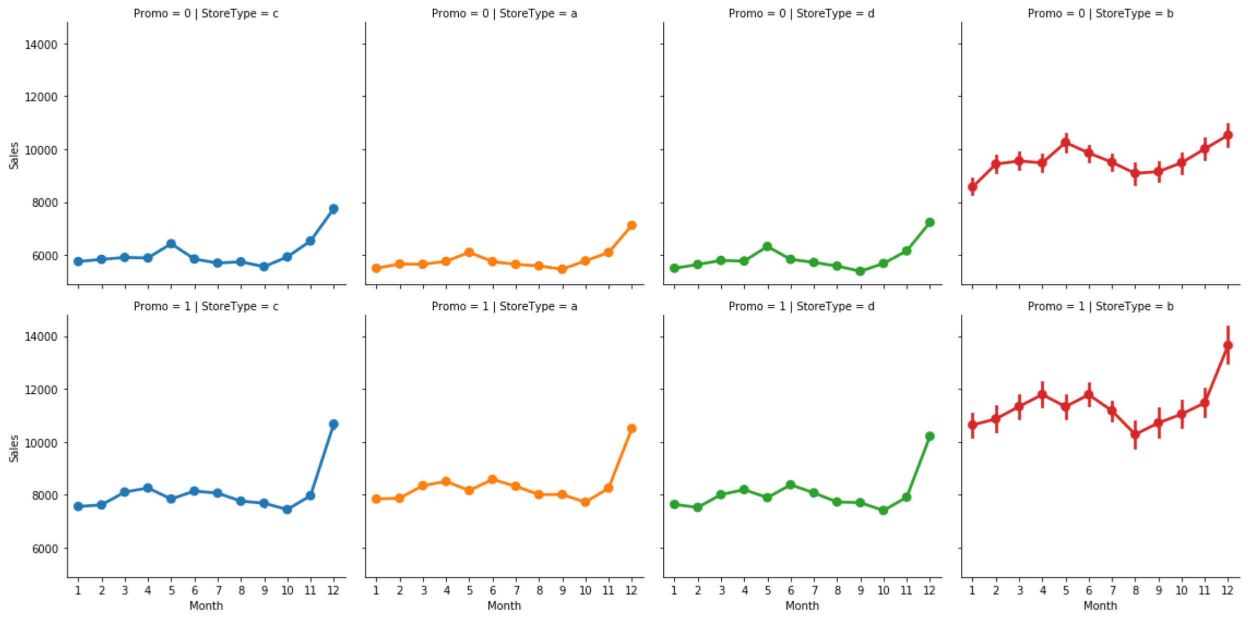


Fig. 5 Average sales amount by store types and with/without promotion

Although store type b has a higher average sales, the sales per customer is actually much lower than others (Fig. 6). This is because the number of customers is significantly higher than the rest of store types, see Fig. 7.

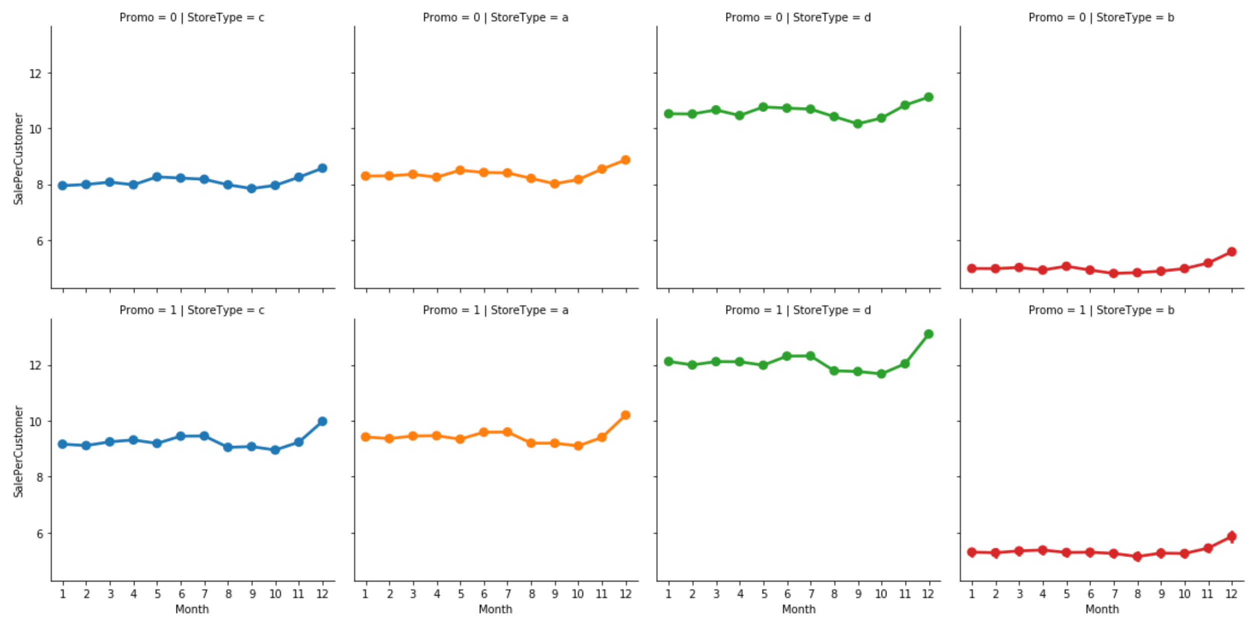


Fig 6. Sales per customer by Promotion (row) and Store Type (column)

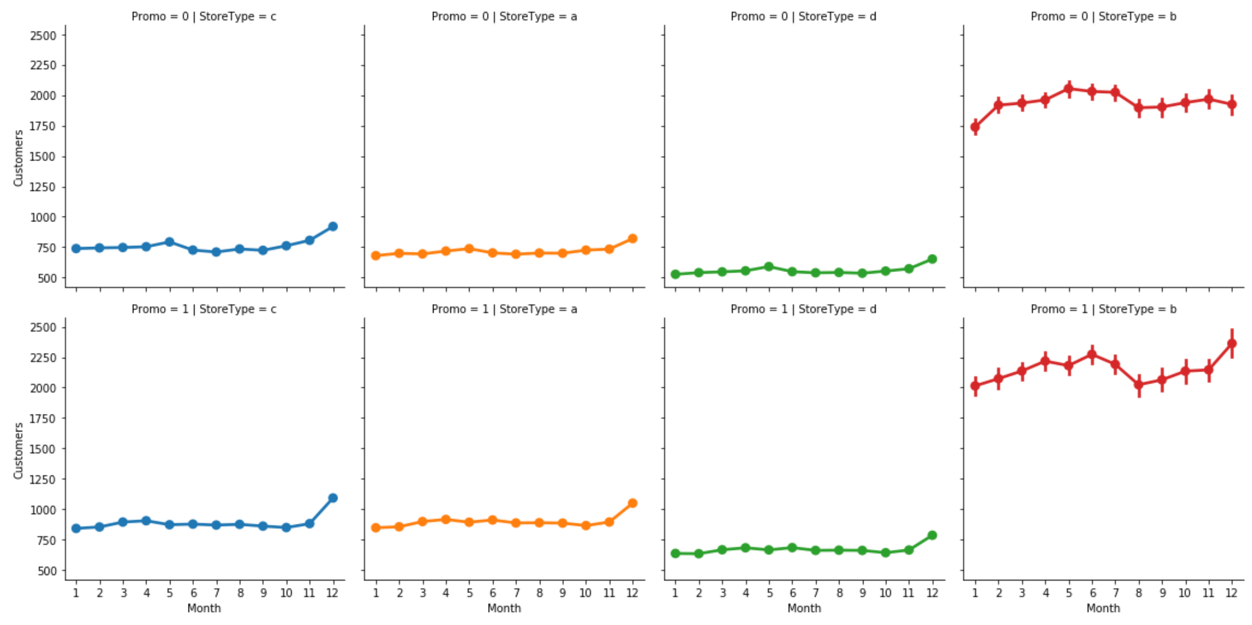


Fig 7. Customer number by Promotion (row) and Store Type (column)

Other than promotion and holiday effect, day of week has some effect on the sales too. Looking at Fig. 8 below, row is store type while column is day of week and first column is Monday and last column is Sunday. Throughout the week, Monday has slightly higher sales in type C, A and D, row 1, row 2 and row 3 respectively. Saturday has a drop in all types of store and on Sunday type A and D (row 2, row 3) have dropped even further but type B climbs back up to weekdays level. Type C has no data on Sunday since it's closed.

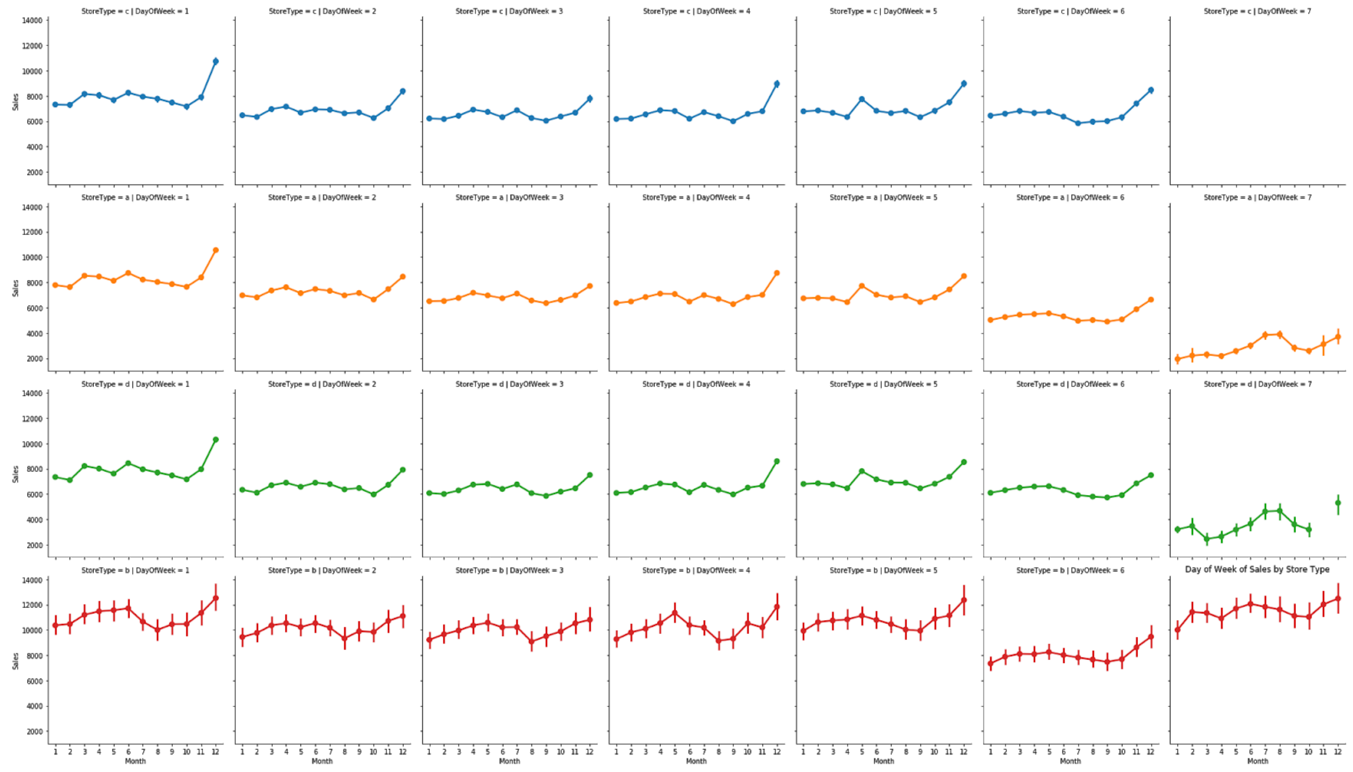


Fig 8. Sales by Store Type (row) and Day of Week (column)

With all the features, we produce a correlation heatmap between features (Fig 9.). Sales, at column 3 in heatmap, is more correlated to number of customers and promotion, however, in test set provided by Kaggle, there is no number of customers. Therefore, this and sales per customer will be removed before modeling. Other than customers and promotion, week of year and promotion open by month (how long the long term promotion has been running) have the higher correlation to sales.

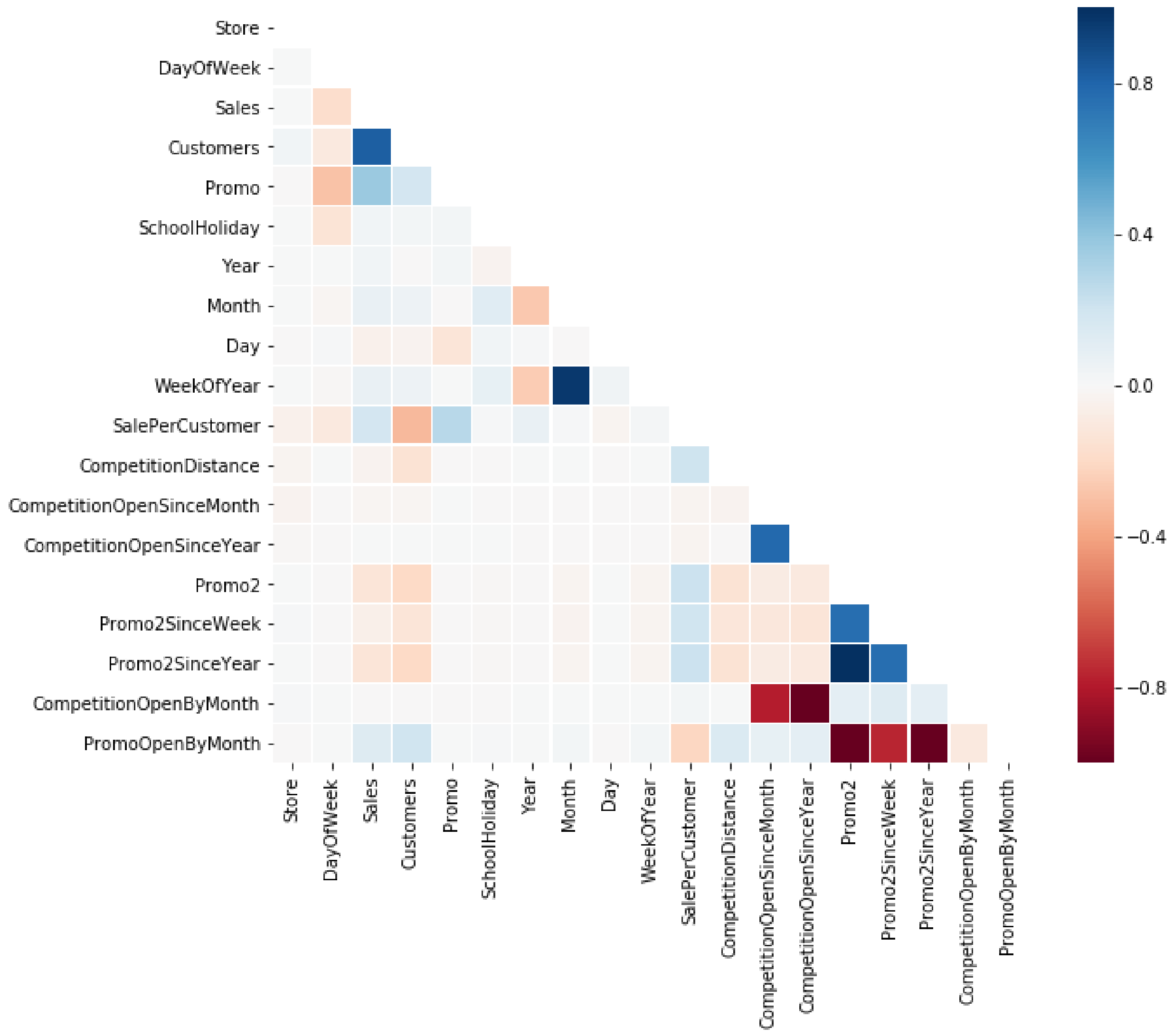


Fig 9. Heatmap of features correlation

Below figure are the time series by store type (row). Type A (row 1) and C (row 3) have an increase in sales at about mid-Dec and a drop in just before Jan. No data for Type D store between Jul 2014 and Jan 2015 which is closed.

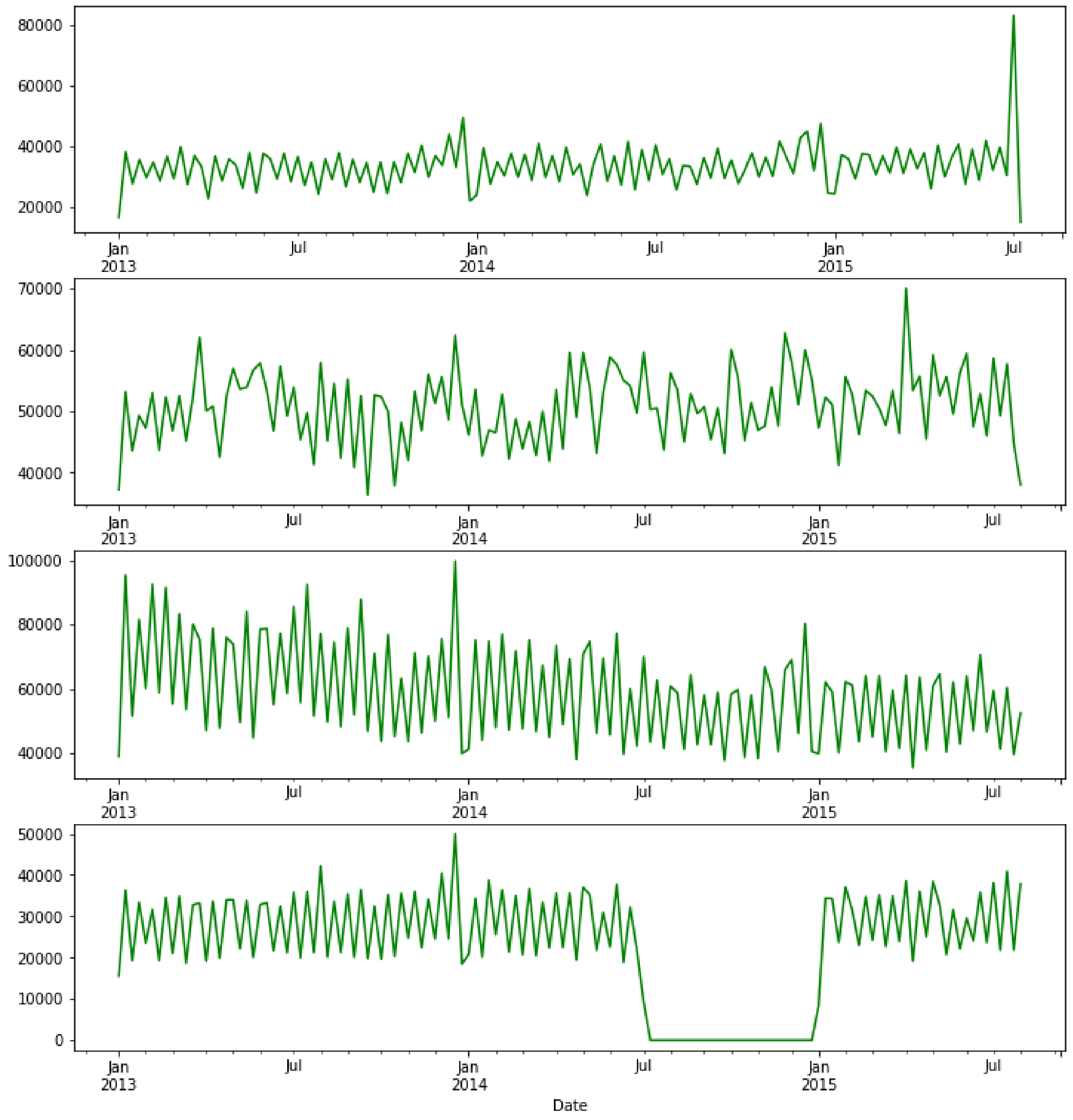


Fig 10. the rate of delay flight of day of week

5. Feature Engineering

The original train file has only 15 features. First we break up the date into year, month and day, then add week of year as new feature. Promo2SinceYear and Promo2SinceWeek describes the year and calendar week when the store started participating in Promo2, but we need to replace and convert this to number of months for each date sales records. And we do the same on CompetitionOpenSinceMonth, CompetitionOpenSinceYear, gives the approximate year and month of the time the nearest competitor was opened, to be converted and replaced by CompetitionOpenByMonth, number of months that the competition opened.

6. Modeling

We would like to predict sales amount of 6 weeks after the train set provided by Kaggle. However, we are not participating the competition and we would like to have a complete modeling workflow including the model performance evaluation. We will pick the last 6 weeks of data originally in the train set provided.

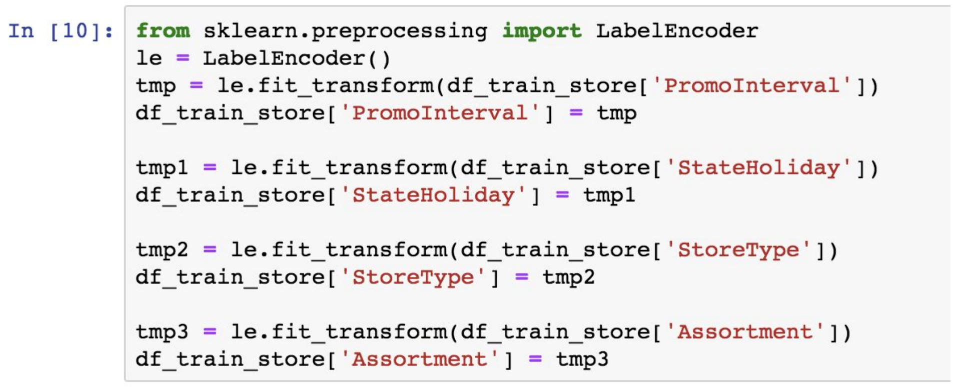
As sales amount is the prediction target, it is a regression problem. We use supervised learning regression algorithms to build the models.

6.1 Data Pre-processing

Before we can feed any data to train the model, there are some preprocessing steps which must be done.

A. Label encoding

Categorical features like like StateHoliday, StoreType, Assortment and PromoInterval which are in text. We need to first convert those into labels by sklearn LabelEncoder function.



```
In [10]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         tmp = le.fit_transform(df_train_store['PromoInterval'])
         df_train_store['PromoInterval'] = tmp

         tmp1 = le.fit_transform(df_train_store['StateHoliday'])
         df_train_store['StateHoliday'] = tmp1

         tmp2 = le.fit_transform(df_train_store['StoreType'])
         df_train_store['StoreType'] = tmp2

         tmp3 = le.fit_transform(df_train_store['Assortment'])
         df_train_store['Assortment'] = tmp3
```

Fig. 11 Promotion (row) and Store Type (column)

B. Data splitting

There are total of 31 months of data. The requirement of competition is to predict 6 weeks of sales. To replicate such requirement, we split the dataset into train set and test set while the test set has the last 6 weeks of dataset. The train set is from 2013-1-1 to 2015-6-15 and test set is from 2015-6-16 to 2015-7-31.

C. Hyperparameters tuning

Before we can identify the best suitable model for the problem, we have to tune the model by finding out the best hyperparameters for each model. We ran a grid search to find the best hyperparameters for each ML model, then we use the best hyperparameters found to initiate the classifiers, then plug in the training set to training the model and the test set for prediction.

6.2 Model Selection

After pre-processing the dataset and grid searching best hyperparameters, we can start training the model and use the test set to evaluate model performance. We compared 5 different regressors for this problem, the best performance model is Gradient Boosting while the second is XGBoost.

6.3 Metrics selection and Benchmark

In the following, we will evaluate each model and select the best suitable one. It’s good to have a benchmark for evaluating performance of models, we make use of the simple Linear Regression as baseline. In this competition problem, we are required to minimize the Root Mean Square Percentage Error (RMSPE). The RMSPE is calculated as

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

Fig. 12 RMSPE Formula

where y_i denotes the sales of a single store on a single day and yhat_i denotes the corresponding prediction. The RMSPE of each models will be compared to the baseline and the best one will be selected.

6.4 Models

After running grid search to find the best hyperparameters, we train all models with train set and evaluate the model by test set. Other than the Linear Regression as baseline, we have four models. Three out of four are ensemble type of model and the remaining one is neural network.

The result shows, see Table 1, all three ensemble type models are performing much better than the baseline, while neural network is worse. Gradient Boosting is the best performing model among all models reached RMSPE 17.37%, 0.3 percentage points better than XGBoost.

Model	RMSPE
Linear Regression	49.67%
Gradient Boosting	17.37%
XGBoost	17.64%
Random Forest	21.52%
Neural Network	51.92%

Table 1. Model comparison

7. Further improvement

Currently this model is built based on dataset provided by Kaggle which provide limited perspective. If adding some perspective like weather, seasonal flu epidemic distribution which would help the model to be more predictive in solving the sales prediction problem.

8. Conclusion

After exploring all datasets, we used five different supervised regression algorithms (Linear Regression, Random Forest, Gradient Boosting, XGBoost and Neural Network to train the predictive model by using 29.5 months of the whole data. The remaining 6 weeks data was used to evaluate the model.

1. Using Root Mean Square Percentage Error (RMSPE) as performance evaluation metric, we found that the Gradient Boosting regressor gives the the best model performance.
2. We achieved the RMSPE to be about 17.34% while the baseline Linear Regression model is 49.67%

This model is good enough to forecast the sales amount for a store with competitors and promotion data.

Sources:

Kaggle, from
https://www.kaggle.com/c/rossmann-store-sales