Rossmann Store Sales Prediction

Summery

Rossmann is a drug store chain in Europe. Building an accurate sales prediction model can help the chain to better control its inventory: a good sales prediction model can prevent products from going out of stock situation or overstock; moreover, it can help the chain to decide the best sales strategy, e.g., when to have promotion events on what products to achieve the best earning results.

We use about three years of historical sales data from over 1,000 Rossmann stores to build a sales prediction model with Decision Tree Algorithm as well as analyze the factors affecting the sales.

Problem Statement

We use the Rossmann store sales dataset from Kaggle [1] that include historical promotions, holidays, and sales data from 2013 to 2015. In this dataset, the useful information is separated in two files (train.csv and store.csv), so we merge the data fields from these two files before starting to process the data. After having the merged dataset, we split 80% of the data as the training dataset and 20% of the data as the testing dataset.

Metrics

The scoring metrics from the Kaggle competition is Root Mean Square Percentage Error (RMSPE) [1]. The RMSPE is defined as follows:

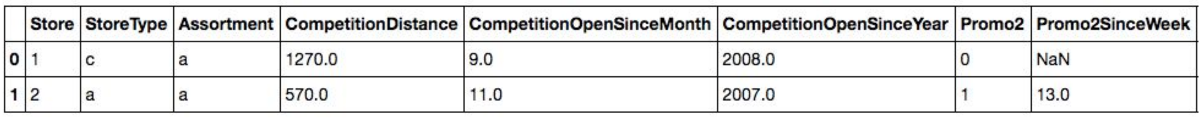
$$\mathrm{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2},$$

where y_i denotes the sales of a single store on a single day and $\hat{y}_i$ denotes the corresponding prediction.

Analysis

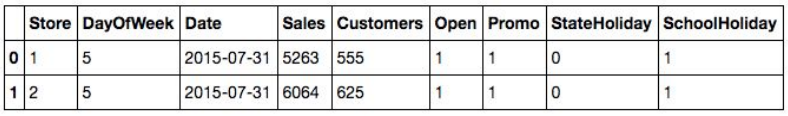
Exploratory Visualization

To better understand the dataset, we use Figure 1 to show a couple of rows from train.csv and store.csv, and use Figure 2 to categorize the dataset.



	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
0	1	c	a	1270.0	9.0	2008.0	0	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0

Figure 1. (a) Sample rows from store.csv.



	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1

Figure 1. (b) Sample rows from train.csv.

There are two features that can represent the store type: StoreType and Assortment. Figure 2 categorizes the dataset with StoreType and Assortment, and we can see that for both features, the number of rows categorized as type b are much lower than those categorized as other types.

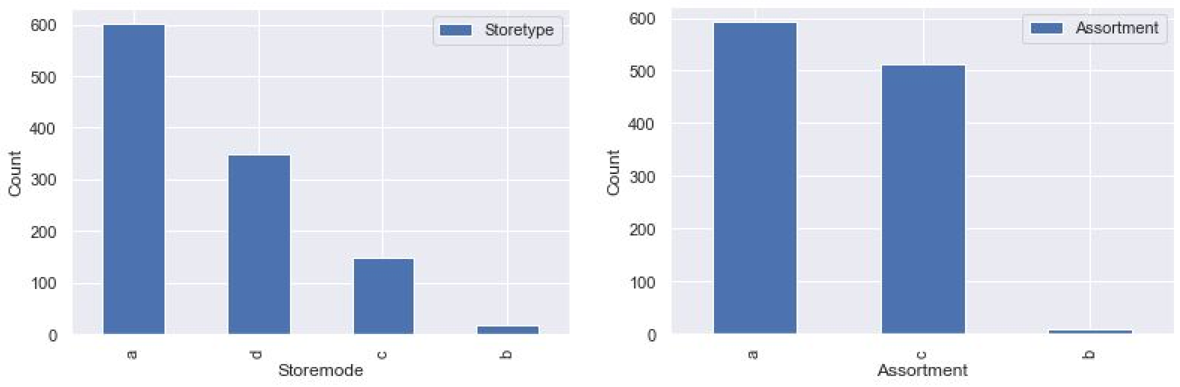


Figure 2. (a) Dataset categorized by StoreType. (b) Dataset categorized by Assortment.

Since StoreType and Assortment both represent the type of a store, we combine these two features into one feature. After merging StoreType and Assortment to a new feature, Storemode, we now have 9 different categories: aa, ac, dc, da, ca, cc, bb, ba, and bc as shown in Figure 3.

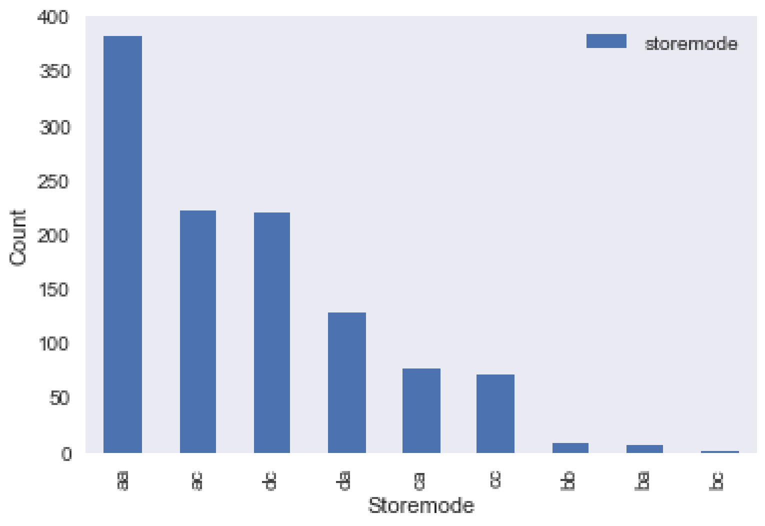


Figure 3. Dataset categorized by Storemode.

Since the StoreType data is unbalanced, we want to learn more about each type of the store as shown in Figure 4. We plot the mean number of sales and mean number of customers of each store and then we can see that although the number of stores with StoreType b is much lower than other types, the mean number of sales and mean number of customers is much higher than other type of store. Therefore, a reasonable hypothesis is that a store with StoreType b might be one located in a good location or has better sales strategy that can attract more customer than other stores. However, as shown in Figure 5, the average spending per customer in StoreType b is lowest, which means that the purchase power of the customers in store with StoreType b is lower than other type of stores. This shows that even the average spending per customer is low in a store, the total sales can still be good as long as the store has enough customers.

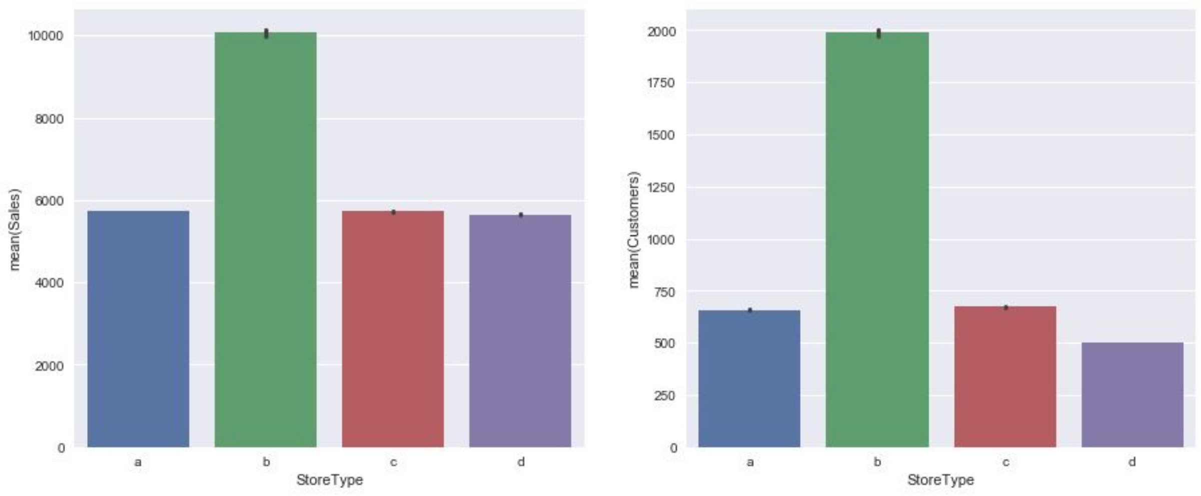


Figure 4. (a) Mean number of total sales by StoreType. (b) Mean of number of total customers by StoreType.

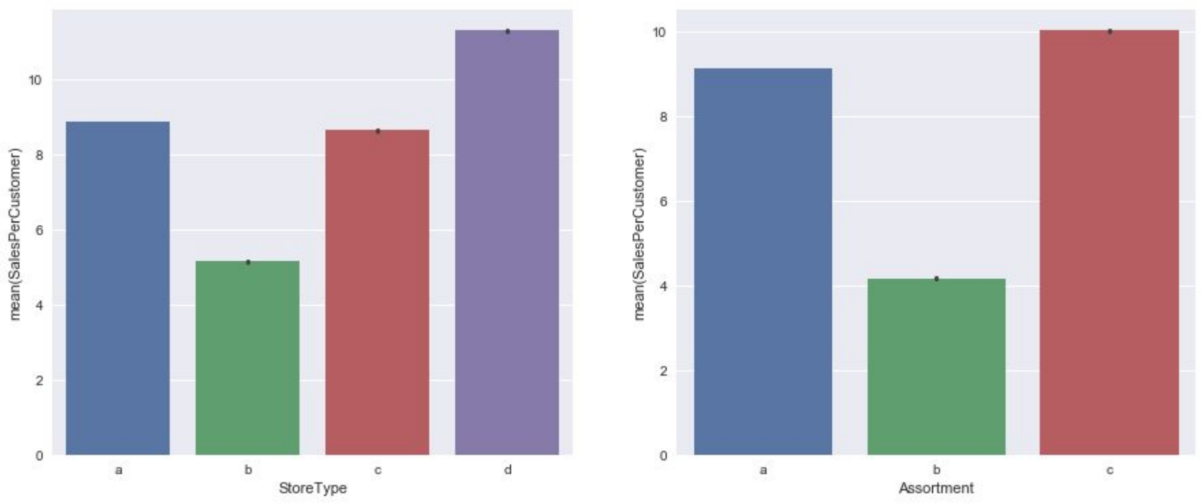


Figure 5. (a) Mean sales per customer by StoreType. (b) Mean sales per customer by StoreType.

An interesting observation is that if we consider both Assortment and StoreType at the same time as a combined Storemode feature, as shown in Figure 6, each Storemode would has similar number of customers. In addition, the sales of stores with Storemode bc is significantly lower than other types of stores.

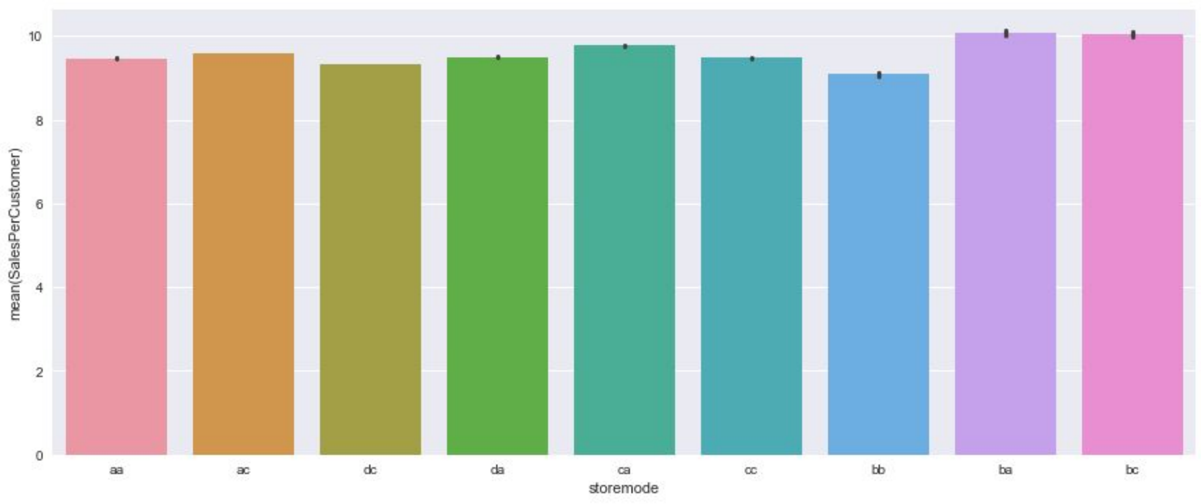


Figure 6. Mean sales per customer by Storemode.

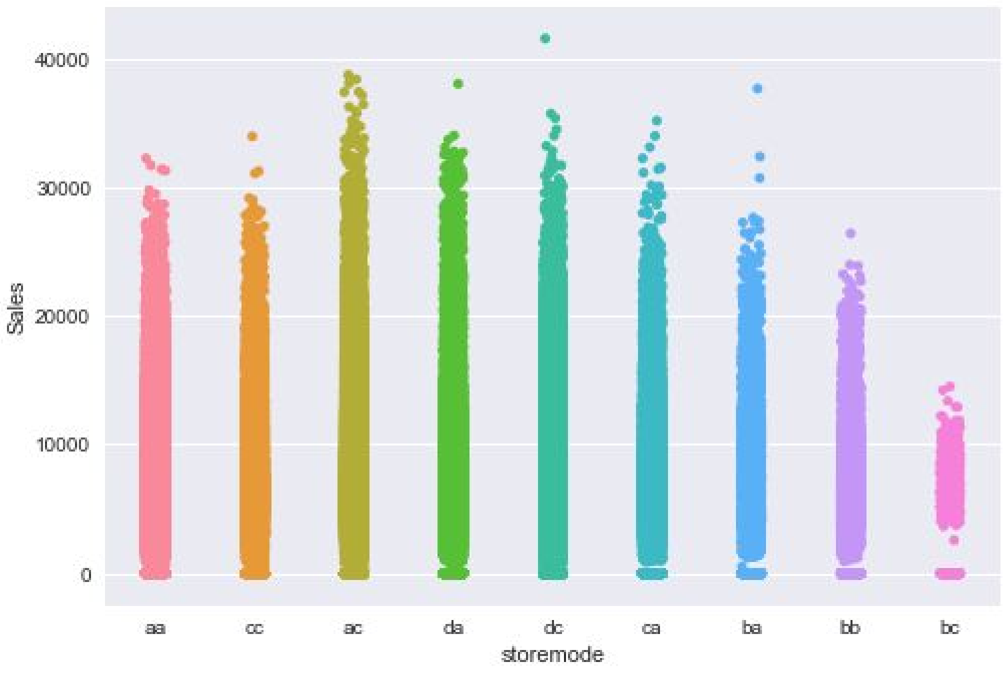


Figure 7. Sales by Storemode.

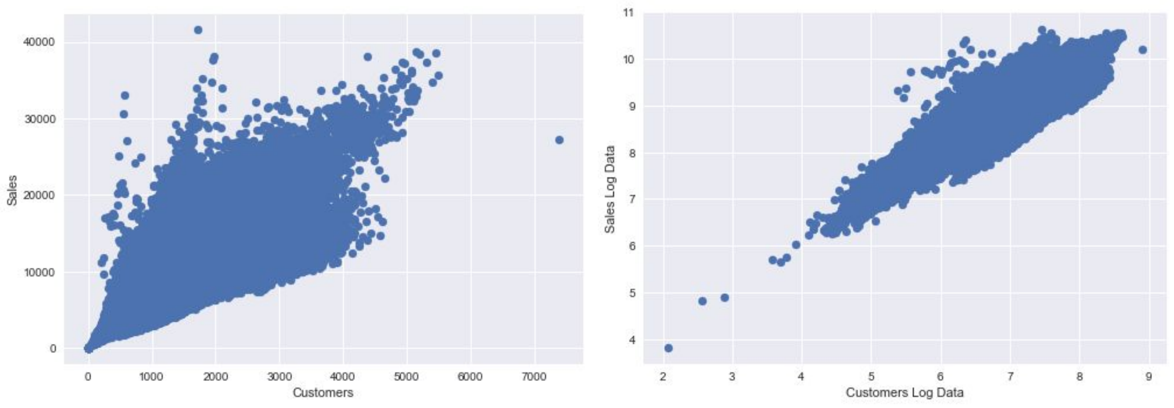


Figure 7. (a) Customers vs. sales. (b) Customers vs. sales in log scale.

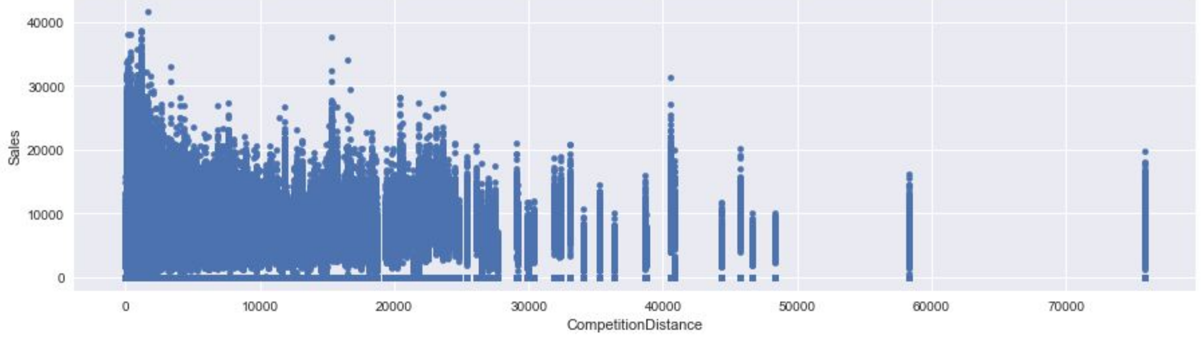


Figure 8. Competition distance vs. sales.

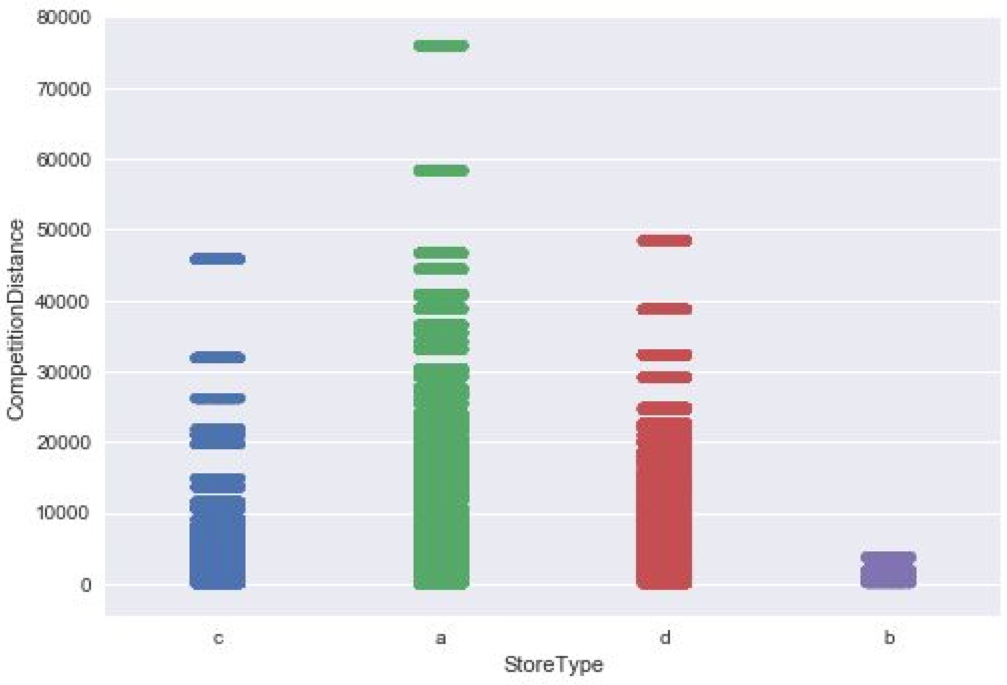


Figure 9. (a) Competition distance by StoreType.

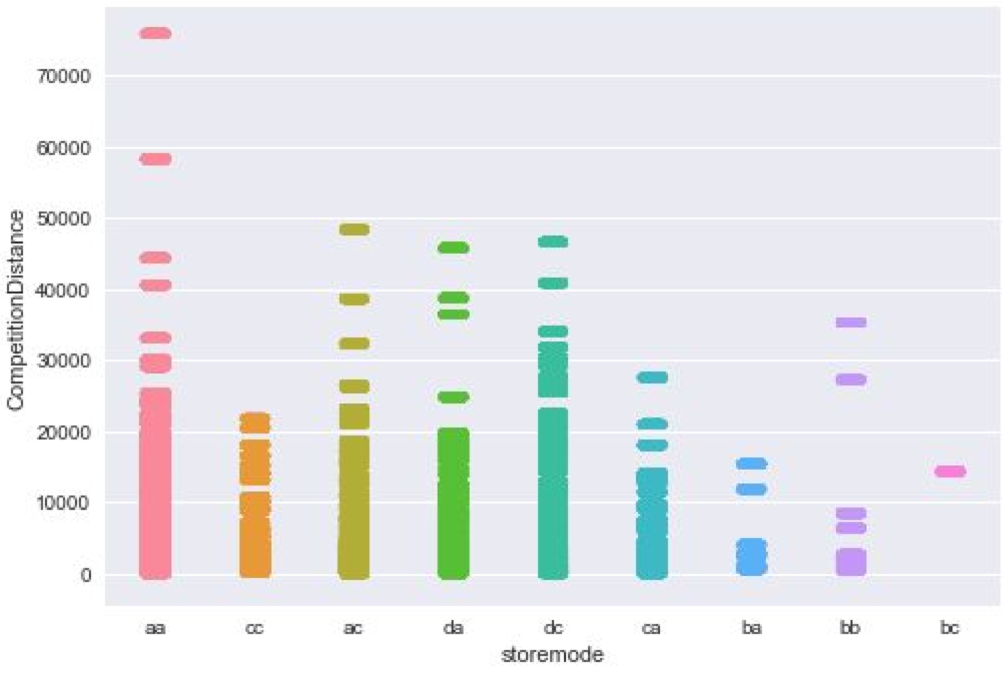


Figure 9. (b) Competition distance by Storemode.

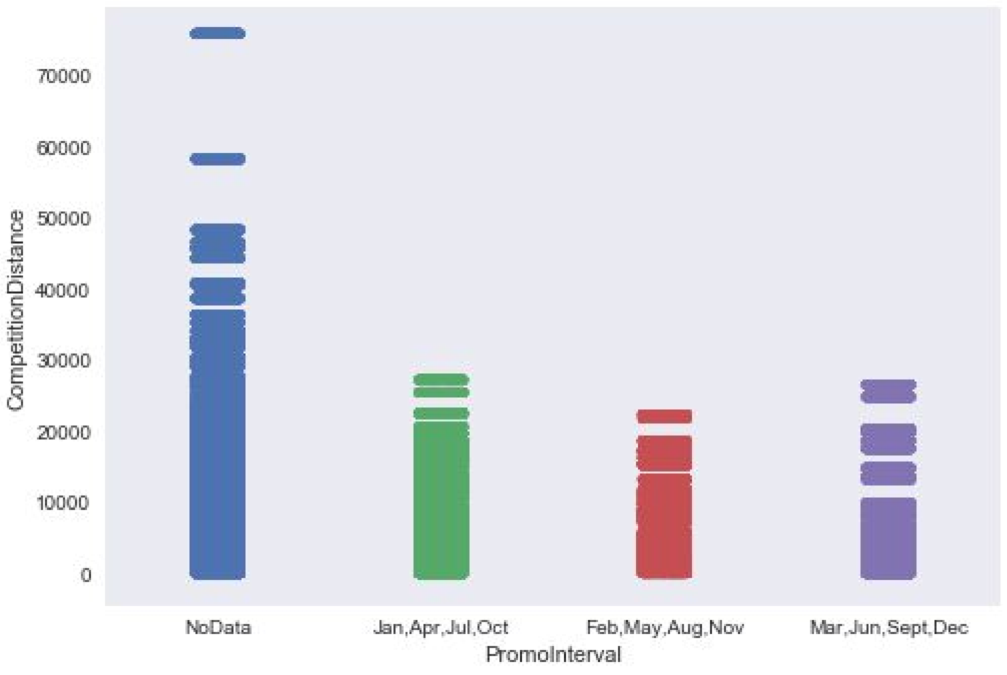


Figure 9. (c) Promotion interval vs. Competition distance.

Figure 10 shows the relationship between Promo2 and competition distance/sales. We can see that stores having the Promo2 feature is closer to a competitor.

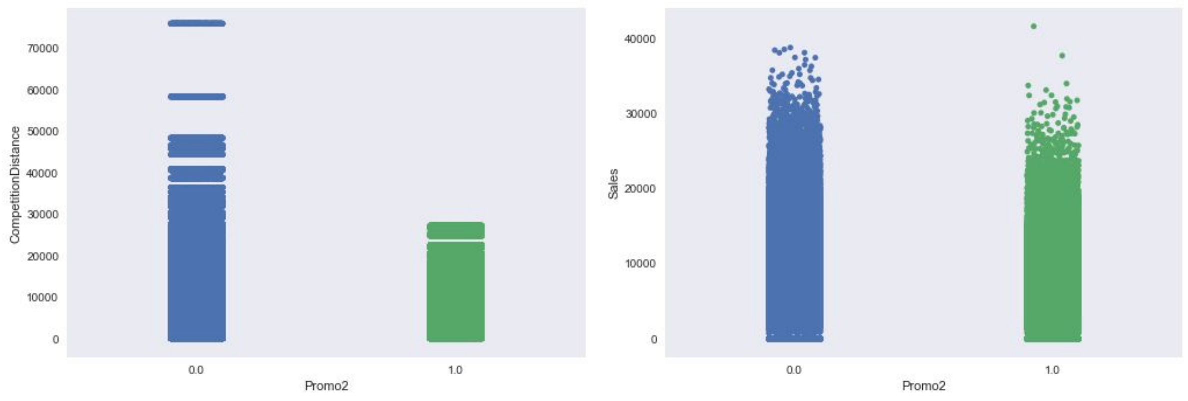


Figure 10. (a) Promo2 vs. competition distance. (b) Promo2 vs. sales.

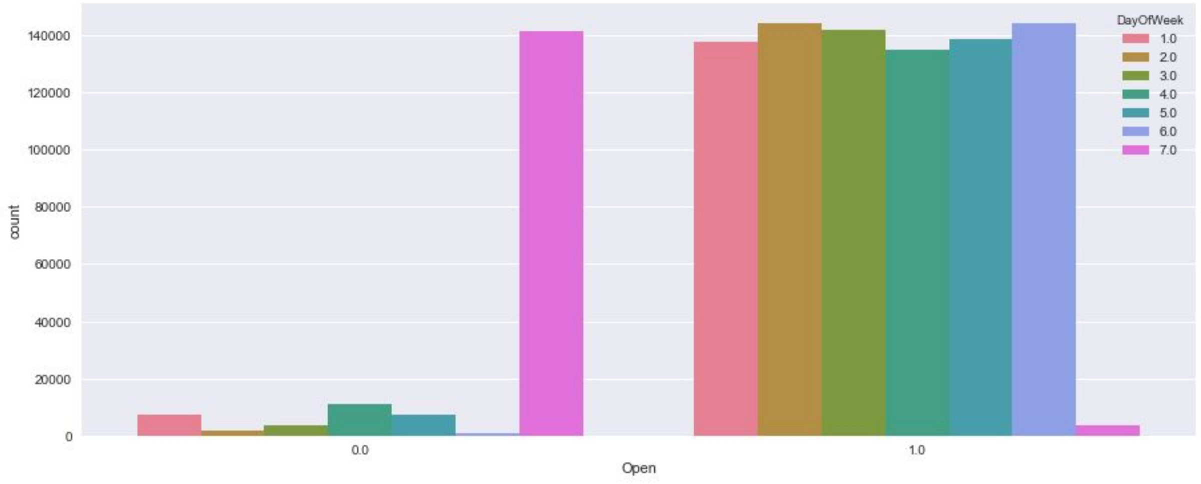


Figure 11. Dataset catagorized by opening on what day of a week.

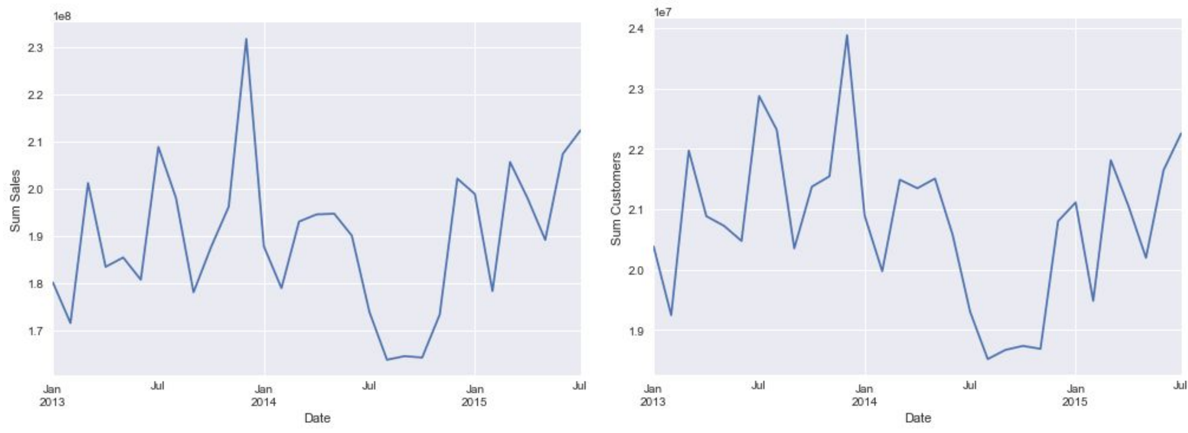


Figure 12. (a) Date vs. total number of sales. (b) Date vs. total number of customers.

Since there are many holidays in December, the average sales per customer is higher than that in other months as shown in Figure 13.

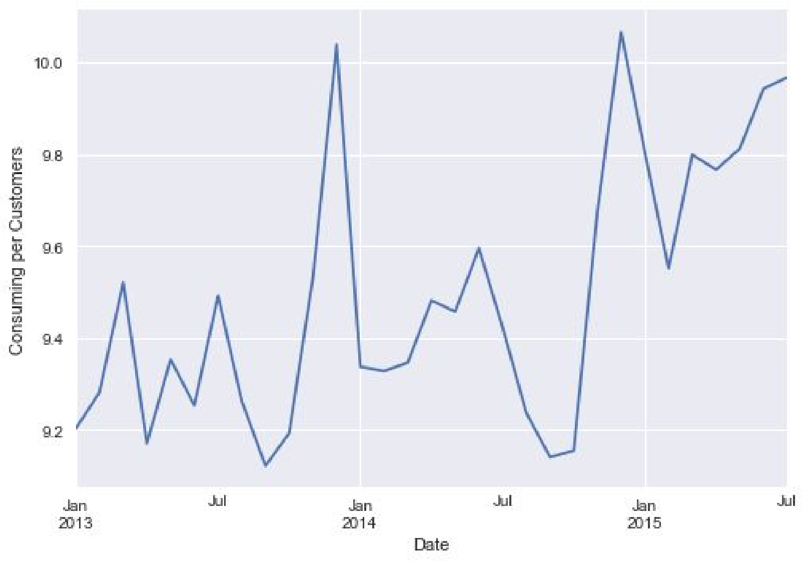


Figure 13 Date vs. consuming per customers.

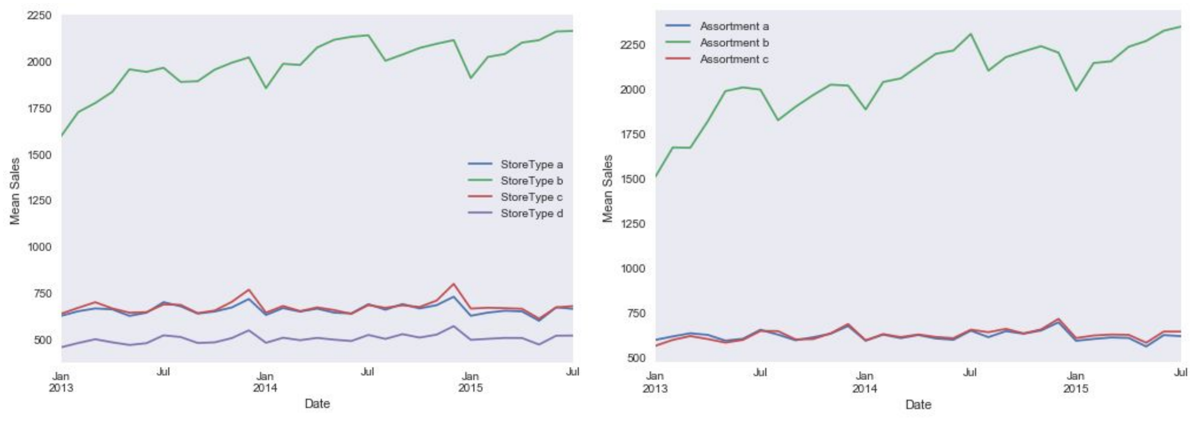


Figure 14. (a) Date vs. mean number of sales by StoreType. (b) Date vs. mean number of sales by Assortment.

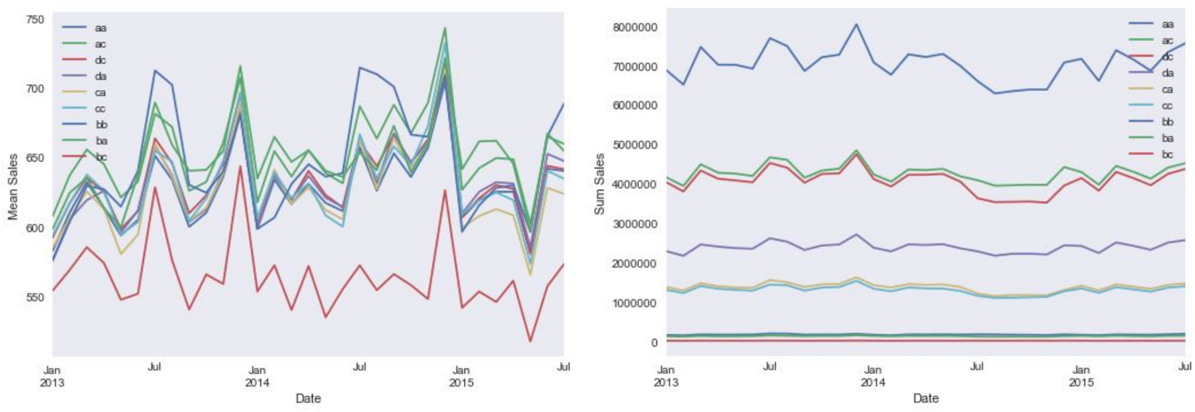


Figure 15. (a) Date vs. mean number of sales by Storemode. (b) Date vs. total number of sales by Storemode.

Benchmark

We use the average number of store sales as the benchmark to evaluate our model. The RMSPE score of the benchmark is 0.45.

Data Preprocessing and Cleaning

- The original data include year, month, and date. However, year and date is not important for predicting sales, so we only keep month for training and prediction.
- Following the instructions on Kaggle, we remove all rows with a 0 in the "Sales" column, because that means the particular store is not open on that day.
- Deal with missing data: we use "nan" to fill categorical missing data and medium to fill numerical missing data.
- Encode catagorical data.
- Normalized numerical data
- Use grid search to find the optimal hyperparameter.
- After analyzing feature importance with Decision Tree, we remove the least important features: "StateHoliday," "Store," and "Promo2SinceWeek." Then, train the model again to get the final predictions.

Experiments and Results

To avoid overfitting we analyze the model complexity and the learning performance of the model. Figure 16. shows the model complexity of Decision Tree. The error decreases as the max depth increases, but when max depth is over 25, the validation error starts increasing because the the model starts overfitting the data.Therefore, although the training error still decreases as the max depth increases over 25, we choose max depth to be 25 as the optimal hyperparameter.

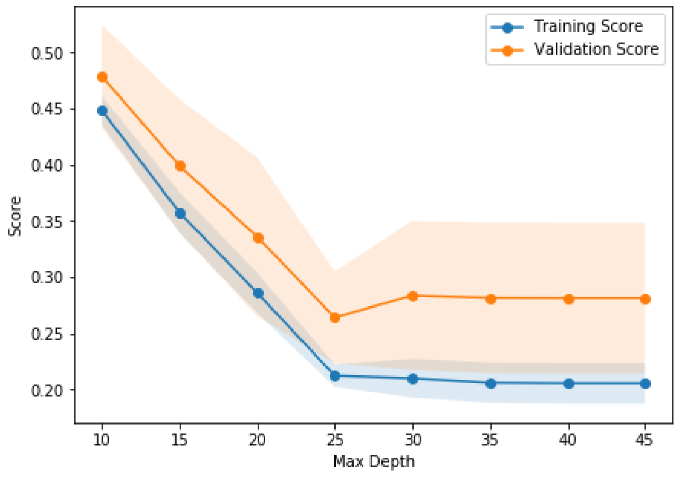


Figure 16. Model complexity of Decision Tree.

Figure 17 shows the learning performance. The gap between training score and testing score decreases as the number of the dataset increases. This means the more data we have, the more accurate we can train a model. As there is still a gap in between the training score and testing score, we might be able to get a even better training model if we have more than 500,000 data points.

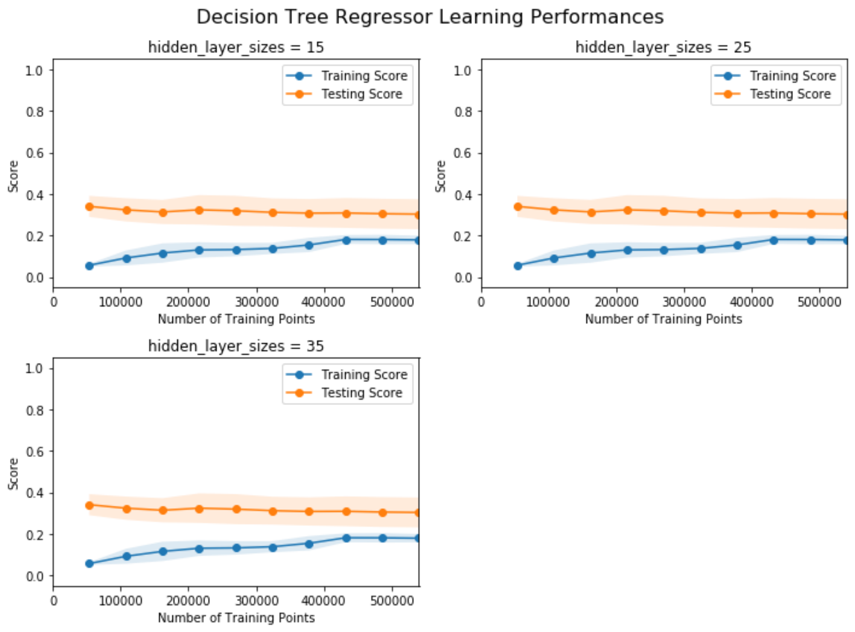


Figure 17. Learning performance of Decision Tree.

The RMSPE we get from decision tree is 0.24, which is almost twice as accurate as the benchmark.

Reference

[1] https://www.kaggle.com/c/rossmann-store-sales/data

[2]
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html