School of Electronic Engineering and Computer Science

MSc Big Data Science -Data Analytic
Project Report 2018

Rossmann Store Sales prediction

Ngoc Minh Nguyen (170771278)
Watchara Sirinaovakul (170779274)
Chun Kit Tsoi (140300468)
Suthiwat Suthapakti (170164540)

Index

1. Background

Rossmann Store Sales is a data science competition, which posted on Kaggle in 2015. The purpose of this campaign is to hire the skillful data scientist across the world. Rossmann is Germany's second-largest drugstore chain operating over 3,600 drugstores in 7 European countries, which was founded in 1972 by Dirk Rossmann. The goal is to predict 6 weeks of daily sale 1,115 stores located across Germany. A precise prediction model will help store managers stay focused on their customer service, to create effective staff schedules that increase productivity and motivation.

Sales were affected by a series of factors, such as the distance of competitors, promotions, school/state holidays, seasonality, and locality. We are provided with a historical sales data between 01/01/2013 to 31/07/2015 from Rossmann stores. In addition, some stores were open every day while some temporarily closed due to refurbishment.

We want to find the main factors that contribute to sales by using different of kind approaching, building up a good model, different kind of regression and time series to see which of this can give the best prediction and accuracy. In this report, we are not only using the Machine Learning methods, we also provide the Time-series analysis to see the difference.

1.1 Aims

We have also made some problem statement and hypothesis as listed below:

Problem statements:

a) How much sales on some stores in the next month?
b) Overall trend of sales in the future.

Hypothesis:

1) Which store type has the most sales?
2) Which assortment type has the most sales?
3) Is it better to open stores on Sunday?
4) Which months have the highest sales?
5) How about competition stores with sales?
6) Which features have the most impact for prediction?
7) Will customers relate to sales?

1.2 Dataset

The dataset is from Rossmann Store Competition in Kaggle. It consists of three files, **train.csv**, **test.csv**, and **store.csv**. Train data (Figure 1.2.1) contains historical data including Sales while test data (Figure 1.2.2) contains the same data as in the **train.csv** but excluding Sales. Store data contains the description about the stores (Figure 1.2.3).

	A	B	C	D	E	F	G	H	I
1	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
2	1	5	31/7/2015	5263	555	1	1	0	1
3	2	5	31/7/2015	6064	625	1	1	0	1
4	3	5	31/7/2015	8314	821	1	1	0	1
5	4	5	31/7/2015	13995	1498	1	1	0	1

Figure 1.2.1 an example data of test.csv

	A	B	C	D	E	F	G	H
1	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
2	1	1	4	17/9/2015	1	1	0	0
3	2	3	4	17/9/2015	1	1	0	0
4	3	7	4	17/9/2015	1	1	0	0
5	4	8	4	17/9/2015	1	1	0	0

Figure 1.2.2 an example data of train.csv

	A	B	C	D	E	F	G	H	I	J
1	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
2	1	c	a	1270	9	2008	0			
3	2	a	a	570	11	2007	1	13	2010	Jan,Apr,Jul,Oct
4	3	a	a	14130	12	2006	1	14	2011	Jan,Apr,Jul,Oct
5	4	c	c	620	9	2009	0			

Figure 1.2.3 an example data of store.csv

The list of features with description in the dataset will be displayed on Table 1.2.1. In addition, some features might contain null value. So, we have to determine the actions before training our model such as filtering out or cleaning the missing value.

Feature	Description
Id	Id that represents the store and date
Store	Unique Id for each store
Sales	The daily sales
Customers	The number of customers
Open	The status of the shops 0 = close, 1 = open
StateHoliday	a = public holiday, b = Easter holiday, c = Christmas, 0 = None
SchoolHoliday	School will close on Weekend and state holidays 0 = close, 1 = open
StoreType	Store models; a, b, c, d
Assortment	Assortment level a = basic, b = extra, c = extended
CompetitionDistance	The distance of the nearest competitor store in meters
CompetitionOpenSinceMonth	Month that the nearest competitor store was opened
CompetitionOpenSinceYear	Year that the nearest competitor store was opened
Promo	The store will run the promotion on that day 0 = no, 1 = yes
Promo2	The store will run the monthly promotion 0 = no, 1 = yes
Promo2SinceWeek	The week that store start the monthly promotion
Promo2SinceYear	The year that store start the monthly promotion
PromoInterval	The months that store will run the monthly promotion

Table 1.2.1 the description of features in the dataset

2. Technologies and External library

2.1 Technologies

Technologies and external libraries selection is an important part. **Python** will be used for this project, not only because it is required. In fact, I think no one will against that Python is the most popular (might more useful) language for data science/analysis

In term of technologies. We believe that **Anaconda** is a good platform for Windows user as well as **Linux Terminal**. Apart from traditional IDE, **Spyder**, **Jupyter Notebook**. We will be trying the newest technology – **Jupyter Lab**, which is basically the new version of Jupyter Notebook and will replace the Jupyter Notebook soon. The advantage of Jupyter Lab or Jupyter Notebook is that it is just like a physical notebook where you can code along with having comments and results which are kept consistently when being seen by other people.

2.2 External Libraries

Pandas is used for data manipulation and analysis, it offers data structure and operations for manipulating numerical tables and time series. It allows the user to represent and manipulate data as they would in a language like R or MATLAB. A DataFrame is a labelled 2D data matrix that supports many operations on it for convenience.

Numpy is a high-level mathematical functions' library, which deal with n-dimensional arrays and metrics. It contains implementations of computational algorithms in the form of functions and operators, optimized for working with multidimensional arrays.

Matplotlib is extension of Numpy, it provides different type of statistical(2D/3D) plot for Python. It helps to visualize data. It contains different kind of plot.

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

Sklearn is a machine learning library for Python, it contains several machine learning algorithms, for examples, **features regression**, **classification** and **clustering** algorithm. It is quite convenient to train the model such as k-means, random forest, gradient descent, support vector machines

Prophet is a time-series modeling library created by Facebook. It follows the sklearn model API, which we have Prophet class and then calling **fit** and **predict** methods. The input of Prophet is always a Pandas Dataframe with two columns: **ds** (timestamp) and **y** (numeric value).

3. Data Exploration

3.1 Statistics of the data

First of all, we loaded all data from files into **Pandas - DataFrames.** Then, we created new columns called **Year**, **Month**, **Day**, **WeekOfYear** from Date column and **SalePerCustomer** by Sales divided by Customer. The examples and information are in the following figures. (Figure 3.1.1 – 3.1.3)

```
train.head()
```

	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
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

```
test.head()
```

	Id	Store	DayOfWeek	Date	Open	Promo	StateHoliday	SchoolHoliday
0	1	1	4	2015-09-17	1	1	0	0
1	2	3	4	2015-09-17	1	1	0	0
2	3	7	4	2015-09-17	1	1	0	0
3	4	8	4	2015-09-17	1	1	0	0
4	5	9	4	2015-09-17	1	1	0	0

Figure 3.1.1 the original train/test data

```
store.head()
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear
0	1	c	a	1270.000000	9	2008	0	NaN	NaN
1	2	a	a	570.000000	11	2007	1	13	2010
2	3	a	a	14130.000000	12	2006	1	14	2011
3	4	c	c	620.000000	9	2009	0	NaN	NaN
4	5	a	a	29910.000000	4	2015	0	NaN	NaN

Figure 3.1.2 the original store data

Date	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	Year	Month	Day	WeekOfYear	SalePerCustomer
2015-07-31	1	5	5263	555	1	1	0	1	2015	7	31	31	9.482883
2015-07-31	2	5	6064	625	1	1	0	1	2015	7	31	31	9.702400
2015-07-31	3	5	8314	821	1	1	0	1	2015	7	31	31	10.126675
2015-07-31	4	5	13995	1498	1	1	0	1	2015	7	31	31	9.342457
2015-07-31	5	5	4822	559	1	1	0	1	2015	7	31	31	8.626118

Figure 3.1.3 reformatted train data

Now, moving to the statistics of the train data (Figure 3.1.4). The average spending for each customer per day is 9.49 (with SD = 2.19). Interestingly, the minimum sale is 0, meaning that there are some days where the stores could not sell anything or not operating. The maximum average spending per customer was surprisingly high at almost 65. We will need to look closer into this record.

	Store	DayOfWeek	Sales	Customers	Open	Promo	SchoolHoliday	Year	Month	Day	WeekOfYear	SalePerCustomer
count	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	844340.000000
mean	5.584297e+02	3.998341e+00	5.773819e+03	6.331459e+02	8.301067e-01	3.815145e-01	1.786467e-01	2.013832e+03	5.846762e+00	1.570279e+01	2.361551e+01	9.493619
std	3.219087e+02	1.997391e+00	3.849926e+03	4.644117e+02	3.755392e-01	4.857586e-01	3.830564e-01	7.773960e-01	3.326097e+00	8.787638e+00	1.443338e+01	2.197494
min	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.013000e+03	1.000000e+00	1.000000e+00	1.000000e+00	0.000000
25%	2.800000e+02	2.000000e+00	3.727000e+03	4.050000e+02	1.000000e+00	0.000000e+00	0.000000e+00	2.013000e+03	3.000000e+00	8.000000e+00	1.100000e+01	7.895563
50%	5.580000e+02	4.000000e+00	5.744000e+03	6.090000e+02	1.000000e+00	0.000000e+00	0.000000e+00	2.014000e+03	6.000000e+00	1.600000e+01	2.200000e+01	9.250000
75%	8.380000e+02	6.000000e+00	7.856000e+03	8.370000e+02	1.000000e+00	1.000000e+00	0.000000e+00	2.014000e+03	8.000000e+00	2.300000e+01	3.500000e+01	10.899729
max	1.115000e+03	7.000000e+00	4.155100e+04	7.388000e+03	1.000000e+00	1.000000e+00	1.000000e+00	2.015000e+03	1.200000e+01	3.100000e+01	5.200000e+01	64.957854

Figure 3.1.4 the statistics of train data

As in Figure 3.1.5, the minimum and maximum distance of competition stores are 20 and 75,860 meters respectively. In addition, the oldest competition store was open in 1900 while the newest was in 2015.

	Store	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear
count	1115.00000	1112.000000	761.000000	761.000000	1115.000000	571.000000	571.000000
mean	558.00000	5404.901079	7.224704	2008.668857	0.512108	23.595447	2011.763573
std	322.01708	7663.174720	3.212348	6.195983	0.500078	14.141984	1.674935
min	1.00000	20.000000	1.000000	1900.000000	0.000000	1.000000	2009.000000
25%	279.50000	717.500000	4.000000	2006.000000	0.000000	13.000000	2011.000000
50%	558.00000	2325.000000	8.000000	2010.000000	1.000000	22.000000	2012.000000
75%	836.50000	6882.500000	10.000000	2013.000000	1.000000	37.000000	2013.000000
max	1115.00000	75860.000000	12.000000	2015.000000	1.000000	50.000000	2015.000000

Figure 3.1.5 the statistics of store data

3.2 Missing Values

It is important to figure out which features have the missing value or unexpected value. Those values can come from computer or human error. As a result, it would decrease the performance of our model. The below table (Table 3.2.1) is the features list, which contains null value. For full detail, please refer to **Data Exploration.ipynb** file in the appendix.

Dataset	Features
Train	-
Test	Open
Store	CompetitionDistance
	CompetitionOpenSinceMonth
	CompetitionOpenSinceYear
	Promo2SinceWeek
	Promo2SinceYear
	PromoInterval

Table 3.2.1 the list of features which contain null value

According to the Table 3.2.1, train data does not have any column containing the null value. The **Open** column of test data contains some null values. We will assume that all row in test data should be opened.

For **CompetitionDistance**, **CompetitionOpenSinceMonth**, and **CompetitionOpenSinceYear** column, we will treat it as no competition store if some columns have null value. We will set it to be 0 instead of null value. Another group is **Promo2**, if **promo2** is 0 other columns including **Promo2SinceWeek**, **Promo2SinceYear**, and **PromoInterval** will be null value. Lastly, we changed null value in **PromoInterval** to be None.

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1017209 entries, 2015-07-31 to 2013-01-01
Data columns (total 13 columns):
Store              1017209 non-null int64
DayOfWeek          1017209 non-null int64
Sales              1017209 non-null int64
Customers          1017209 non-null int64
Open               1017209 non-null int64
Promo              1017209 non-null int64
StateHoliday       1017209 non-null object
SchoolHoliday      1017209 non-null int64
Year               1017209 non-null int64
Month              1017209 non-null int64
Day                1017209 non-null int64
WeekOfYear         1017209 non-null int64
SalePerCustomer    844340 non-null float64
dtypes: float64(1), int64(11), object(1)
memory usage: 128.6+ MB
```

Figure 3.2.1 train data Information

As shown in Figure 3.2.1, there are (1,017,209 - 844,340) 172,869 missing values from the **SalePerCustomer** columns. This is due to 0 values of customer columns which leading nan values after dividing by 0.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
Store                      1115 non-null int64
StoreType                  1115 non-null object
Assortment                 1115 non-null object
CompetitionDistance        1112 non-null float64
CompetitionOpenSinceMonth  761 non-null float64
CompetitionOpenSinceYear   761 non-null float64
Promo2                     1115 non-null int64
Promo2SinceWeek            571 non-null float64
Promo2SinceYear            571 non-null float64
PromoInterval              571 non-null object
dtypes: float64(5), int64(2), object(3)
memory usage: 87.2+ KB
```

```
Store                        0
StoreType                    0
Assortment                   0
CompetitionDistance          3
CompetitionOpenSinceMonth  354
CompetitionOpenSinceYear   354
Promo2                       0
Promo2SinceWeek            544
Promo2SinceYear            544
PromoInterval              544
dtype: int64
```

Figure 3.2.2 the data type and missing value on store data

We can see the number of each column in Figure 3.2.2. We know the columns containing the missing value. Therefore, we will handle it before building the model as shown below.

Here is the histogram of distance/duration of competition stores,

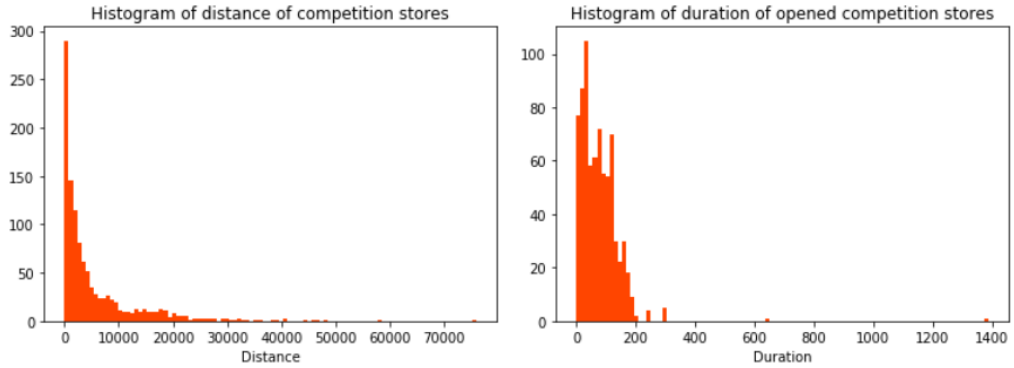


Figure 3.2.3 the histogram of distance and duration of competition stores

Refer to Figure 3.2.3, some competitors were further away. Generally, the distance of competitors were less than 5,000 meters. In addition, most competitors had opened more than 200 months or about 16 years. The longest running competitor had been opened for 100 years.

First, we will fill up the missing values for **CompetitionDistance** with its median since there are some outliers which make the distribution become positive skew. It makes sense to use median for the missing values.

Secondly, checking whether the missing values of **Promo2SinceWeek**, **Year**, and **Interval** are coming from not having **Promo2** by querying when **Promo2** is not 0. The result is 0 records meaning that not having **Promo2** leads to those missing values. In this case, we will fill the missing values with 0.

The result below (Figure 3.2.4) is when checking **sales** = **0** and **Customer > 0**, meaning there were no sales but having customers. Surprisingly, there are 2 records meeting this criterion.

Date	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	Year	Month	Day	WeekOfYear	SalePerCustomer
2014-04-29	1100	2	0	3	1	1	0	0	2014	4	29	18	0.0
2013-04-25	948	4	0	5	1	1	0	0	2013	4	25	17	0.0

Figure 3.2.4 the zero sale rows when having the customers

We are checking the stores when they were closed and had no sales. As you can see from the Figure 3.2.5, there are 172,817 rows meeting this criterion. We must remove these rows to prevent the model bias towards zero sales prediction.

(172817, 13)

Date	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	Year	Month	Day	WeekOfYear	SalePerCustomer
2015-07-31	292	5	0	0	0	1	0	1	2015	7	31	31	NaN
2015-07-31	876	5	0	0	0	1	0	1	2015	7	31	31	NaN
2015-07-30	292	4	0	0	0	1	0	1	2015	7	30	31	NaN
2015-07-30	876	4	0	0	0	1	0	1	2015	7	30	31	NaN
2015-07-29	292	3	0	0	0	1	0	1	2015	7	29	31	NaN

Figure 3.2.5 the closing day with no sales

Then we are checking the stores when the stores are open with zero sales (Figure 3.2.6). Surprisingly, 54 rows shown up. We will remove these rows as well.

Date	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	Year	Month	Day	WeekOfYear	SalePerCustomer
2015-05-15	971	5	0	0	1	0	0	1	2015	5	15	20	NaN
2015-03-26	674	4	0	0	1	0	0	0	2015	3	26	13	NaN
2015-02-05	699	4	0	0	1	1	0	0	2015	2	5	6	NaN
2014-10-01	708	3	0	0	1	1	0	0	2014	10	1	40	NaN
2014-09-22	357	1	0	0	1	0	0	0	2014	9	22	39	NaN

Figure 3.2.6 the zero sales when opening the stores

3.3 Visualization

The following figure, Figure 3.3.1, shows the proportion of the opened and closed day of the stores. Around 85 percent of data were opened while the rest of them were closed. Additionally, train data and test data contain the similar ratio.

Figure 3.3.1 the proportion of stores that are closed and opened

Here are the proportion of the state/school holiday,

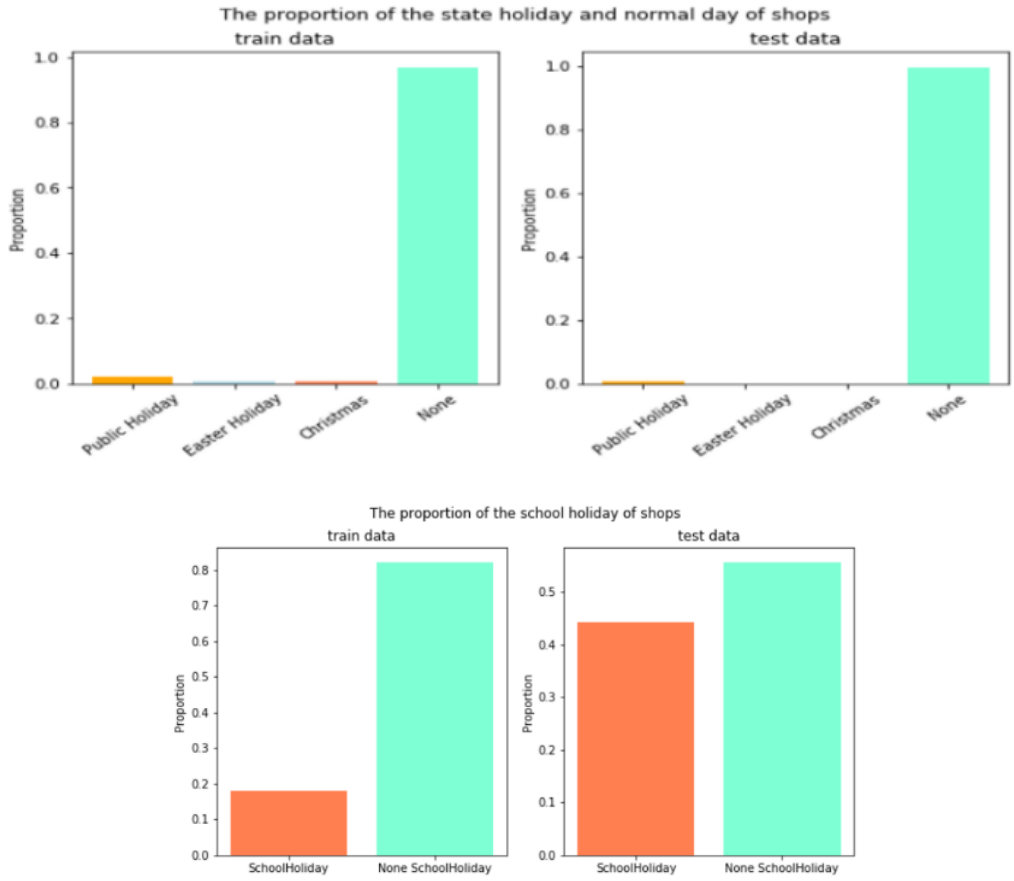


Figure 3.3.2 the proportion of school holiday and state holiday

According to Figure 3.3.2, there are less than 1% state holiday for both dataset. Compared to state holidays, school holidays had a bigger proportion since the Summer holiday and 'LONGER' states holiday are included, resulting in around 20 percent of train data is school holidays. In test data, the ratio of school holidays and none school holidays is about 45% and 55% respectively. Please note that the summer holidays are in test data.

Then we are looking at the proportion of the promotion day by day in the dataset. We can see that around 40% had promotion. The percentages were similar for train and test data (Figure 3.3.3).

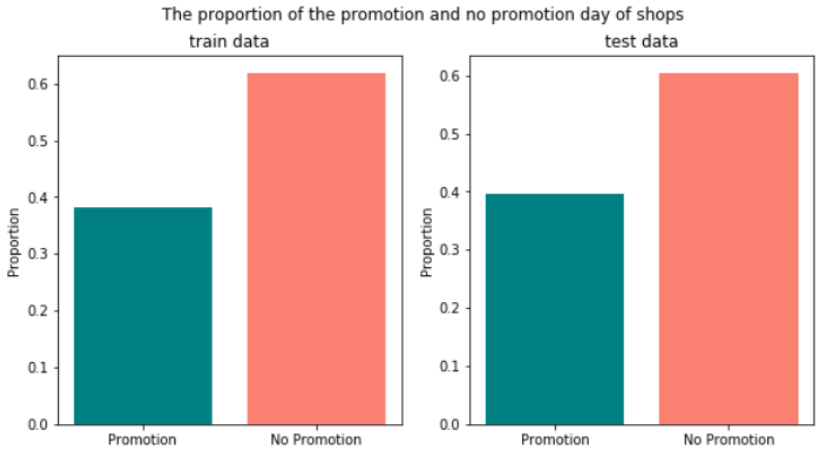


Figure 3.3.3 the proportion of the promotion

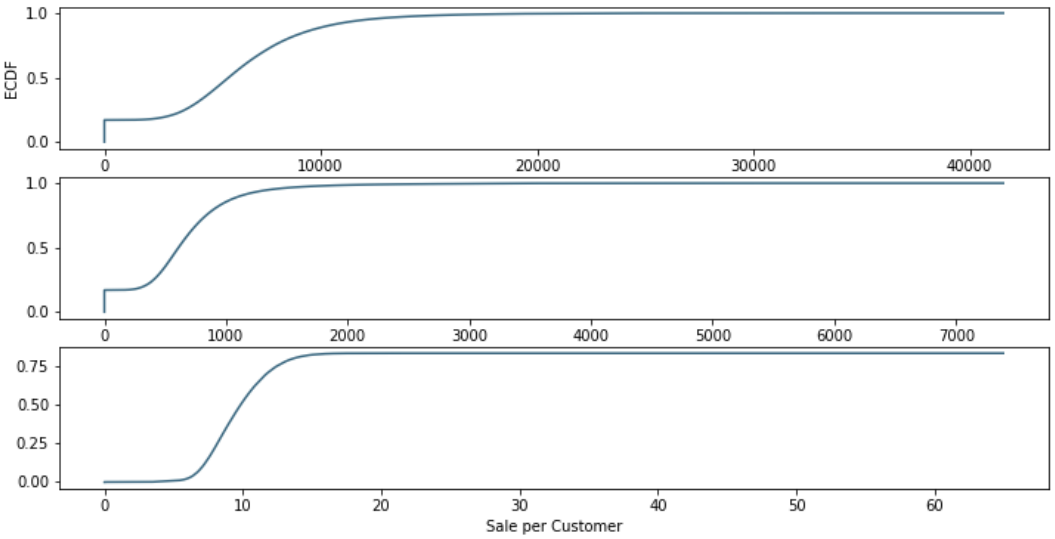


Figure 3.3.4 the Empirical Cumulative Distribution Function(ECDF) of Sales, Customer, SalePerCustomer

According to the fig 3.3.4, there are significant records having 0 values and there are also a lot of outliers (too high values).

Here is the histogram of average sales/customers of each store,

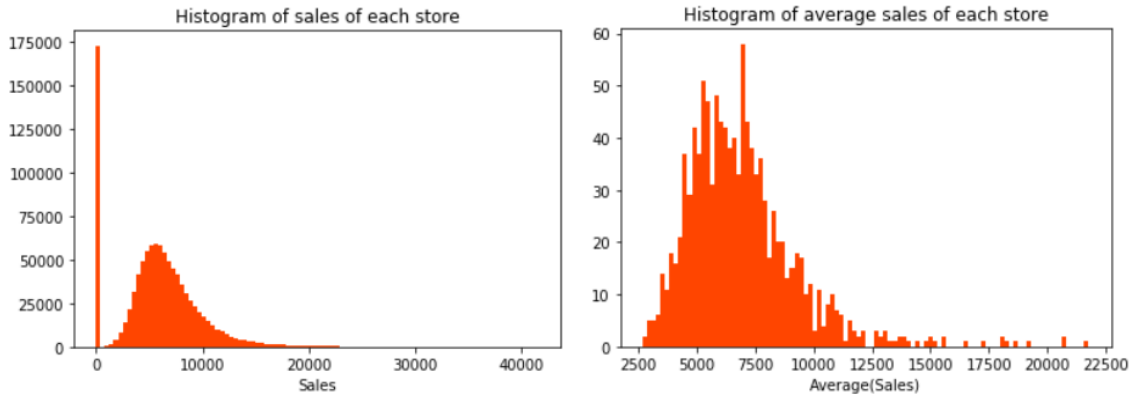


Figure 3.3.5 the histogram of sales by stores

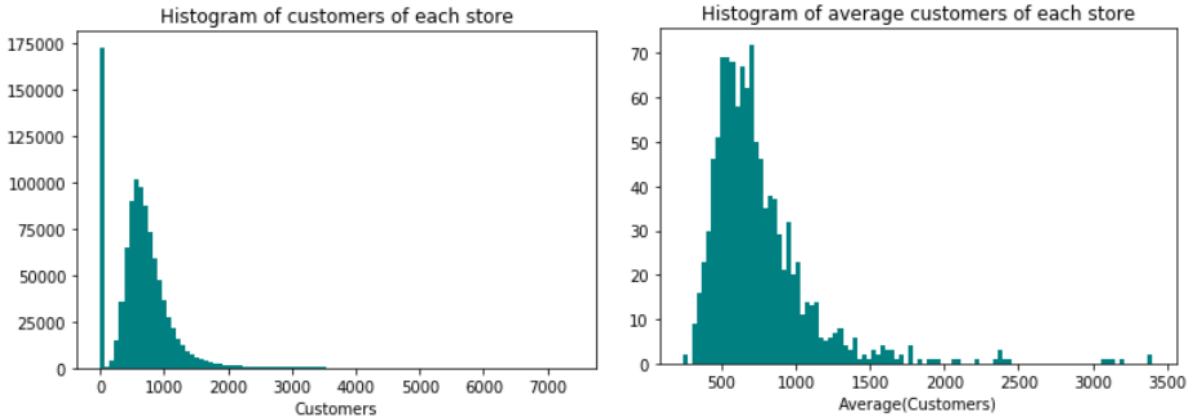


Figure 3.3.6 the histogram of customers by stores

According to Figure 3.3.5 and 3.3.6, the distribution of the sales is similar to that of customers, which are positive skewed. The graphs from left hand side shows that there is a high number of zero sales and customers.) They makes the models biased, we therefore removed the zeros (as shown right hand side).

Now, we will explore the number of sales and customer on each day,

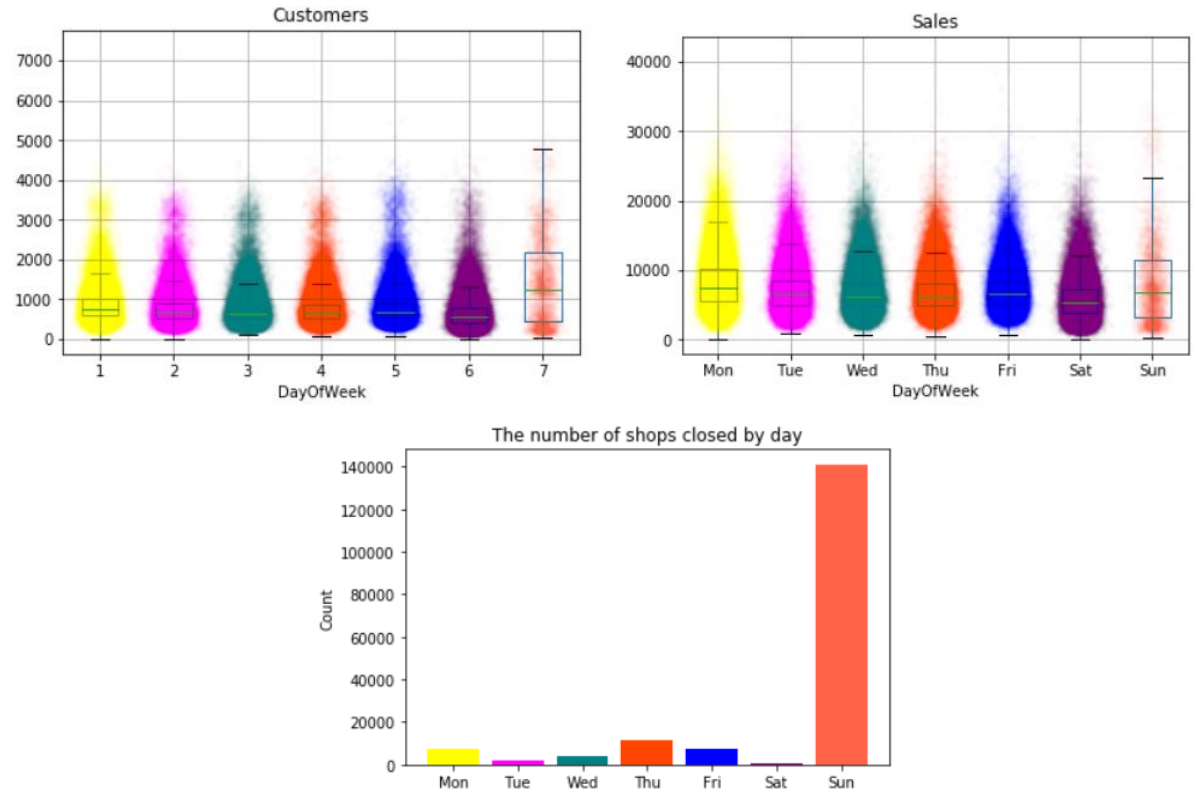


Figure 3.3.7 the sales and customers by days of week

We know from the Figure 3.3.7 that Monday had the most sales/customers. In addition, since most of the stores were closed on Sunday, we can expect the less sales is on Sunday. The variance is the biggest compare to the other days.

Here are the factors that would affect the sales,

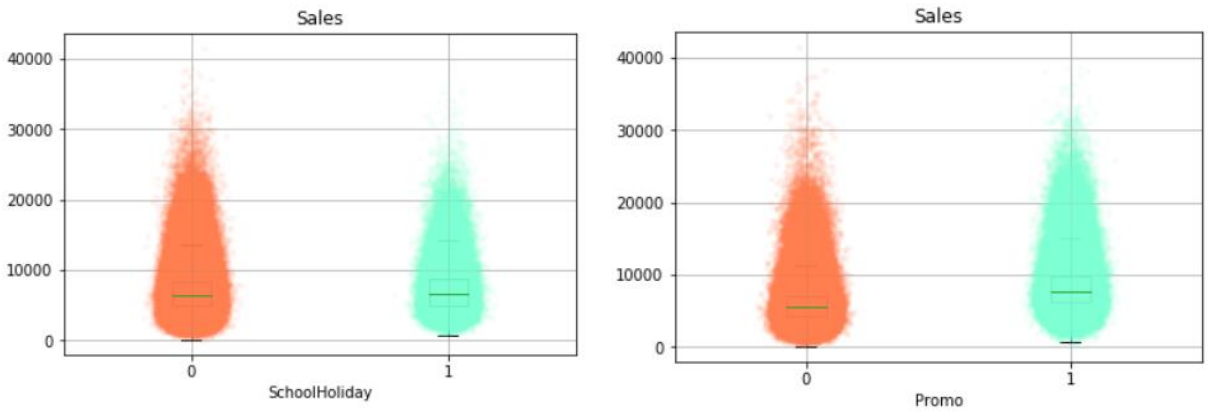


Figure 3.3.8 the sales by school holidays and daily promotions

From Figure 3.3.8, the overall sales seem to be higher when it is not on school holidays. We can assume that the parents chose to look after their children rather than spending money. When applying the promotion, it could increase the sales.

Here is the graph of relation between customers and sales,

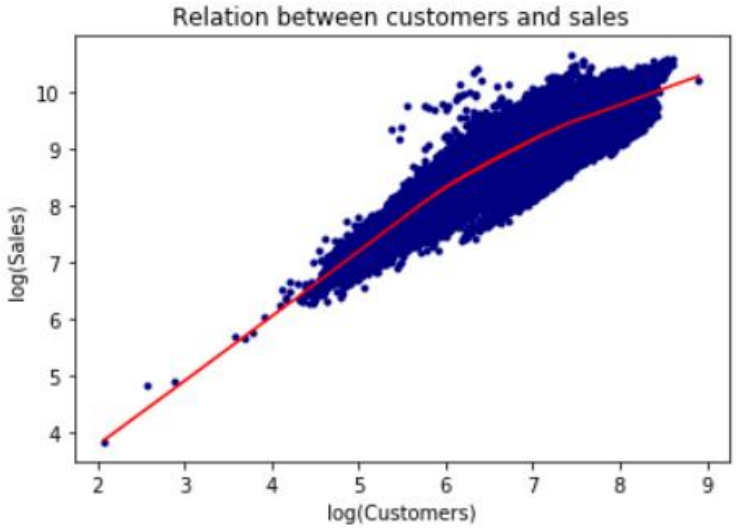


Figure 3.3.9 the graph between customers and sales

According to Figure 3.3.9, there is a positive correlation between customers and sales. The graph has confirmed our hypothesis is correct. When the number of customers increases, the sales also increase.

Some stores had zero sales because they were closed on Sunday or state holiday. However, the histogram of the zero sales is different (Figure 3.3.10). Now we have understood that some stores were closed for a while, because of "refreshment". We will pick some of them to check our assumption.

Store	Number of zero sales
103	311
708	255
349	242
972	240
674	197

Figure 3.3.10 the histogram of zero sales of each store

We found that store 103 had 311 zero sales which is the most. So, we will plot the sales day by day, as in Figure 3.3.11. Regarding the figure, the reason that this store has the highest number of zero sales is the store was opened on 07/2013 which was later than other stores, resulting in a high spike on sales on its first month.

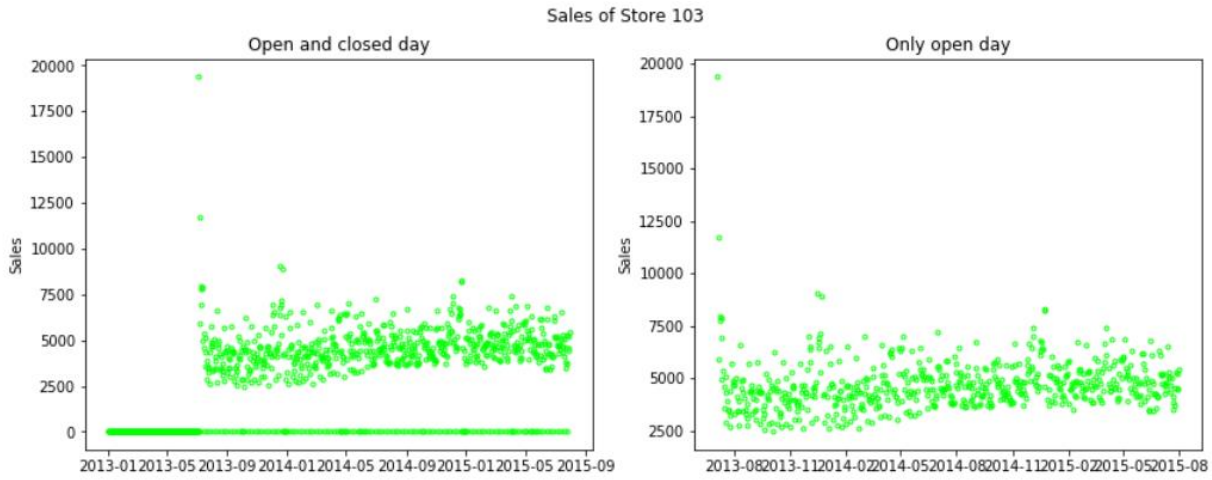


Figure 3.3.11 the daily sales of store 103

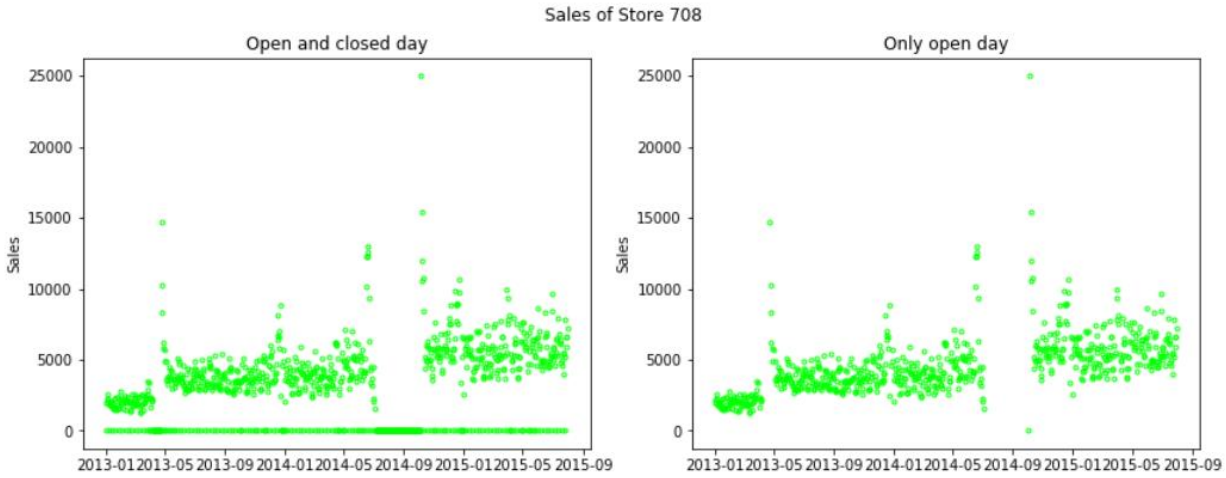


Figure 3.3.12 the daily sales of store 708

Regarding the store 708, Figure 3.3.12, although this store was opened since the first date of the record, it was closed for a about 3 months. We can see on the figure 3.3.12 that after reopened, the sales had a significant growth. Also, it also has a high spike on sales in the first few days.

We can conclude that there are some special events such as promotion which could affect the sales.

After that, we proved one of our hypothesis - if the stores opening on Sunday, would they have higher sales than other days of week?

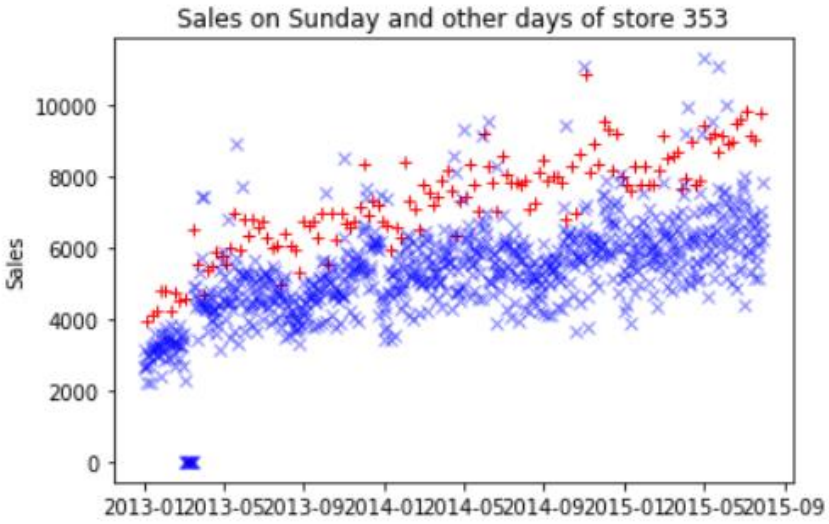
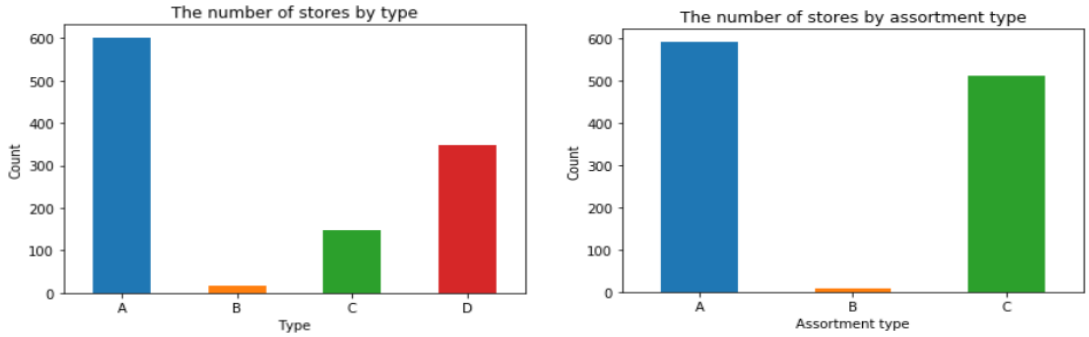


Figure 3.3.13 the sales on Sunday and other days of store 353

The reason why we picked the store 353 is that it never closed on Sunday and state holiday. However, it might have some zero sales on some period. According to Figure 3.3.13, the red plus symbol is the sales on Sunday while the blue cross symbol is of the other days. It could be summarized that there is a greater sale on Sunday and this is the reason why the store is still opened on Sunday.

Here is the number of stores by assortment type,



Assortment	a	b	c
StoreType			
a	381	0	221
b	7	9	1
c	77	0	71
d	128	0	220

Figure 3.3.14 the number of store type and assortment

According to Figure 3.3.14, about a half of the stores are type A and type B is having the smallest number of all stores. We also know that highest assortment are A and C, B is the smallest assortment. Store A with Assortment A is the highest number, 381 stores. The number of store A and D with Assortment C is quite similar, about 220 stores.

Here is the number of stores having promotion for each month,

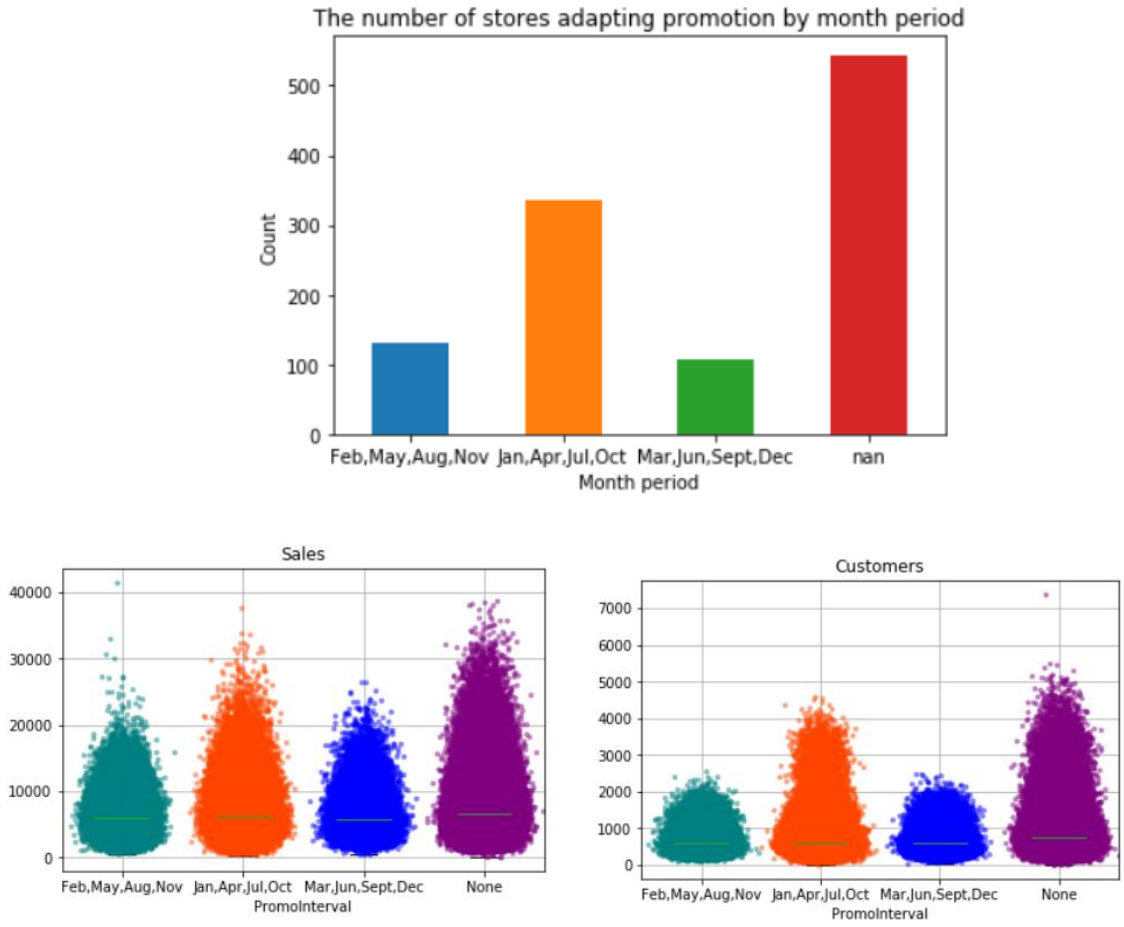


Figure 3.3.15 the graph sales/customers against promotion interval.

Due to Figure 3.3.15, about a half of all store does not have monthly promotions. The most number of monthly promotion were on Jan, Apr, Jul, Oct due to season change, which they want to clear their storage for the new season.

Here is the histogram of duration of promotion,

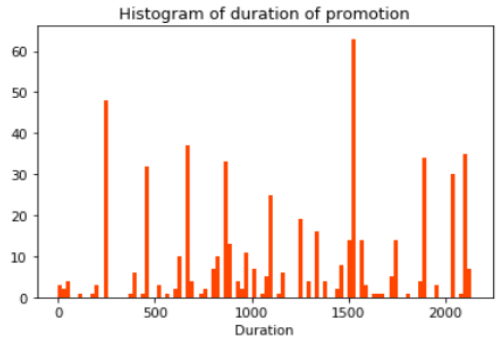


Figure 3.3.16 the histogram of duration of monthly promotion

Then we are focusing on the monthly promotion. The longest duration of promotion is 2,000 days ago.

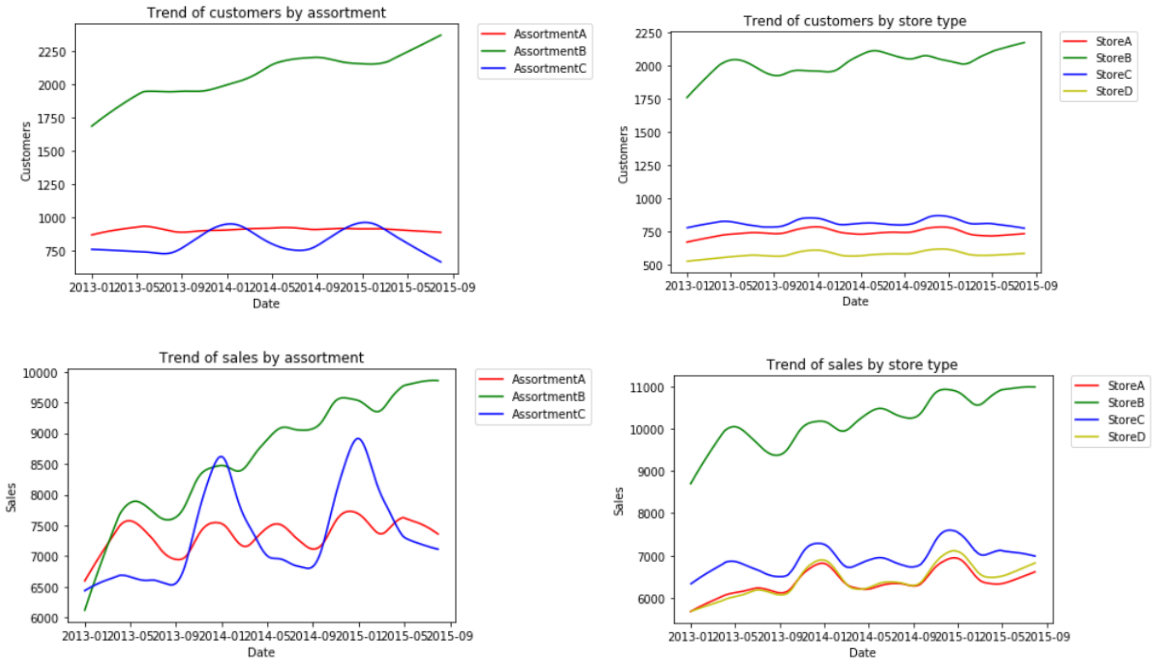


Figure 3.3.17 the graphs for the average sales or customers of different type of stores/assortments

According to Figure 3.3.17, assortment B had the highest number of customers and sales. Only assortment B had an upward trend in term of customers, while the others remain stable. The store type B had the highest average sale and customers. Although we have only the small number of store type B, it shows a high potential and the company can plan to expand type B stores.

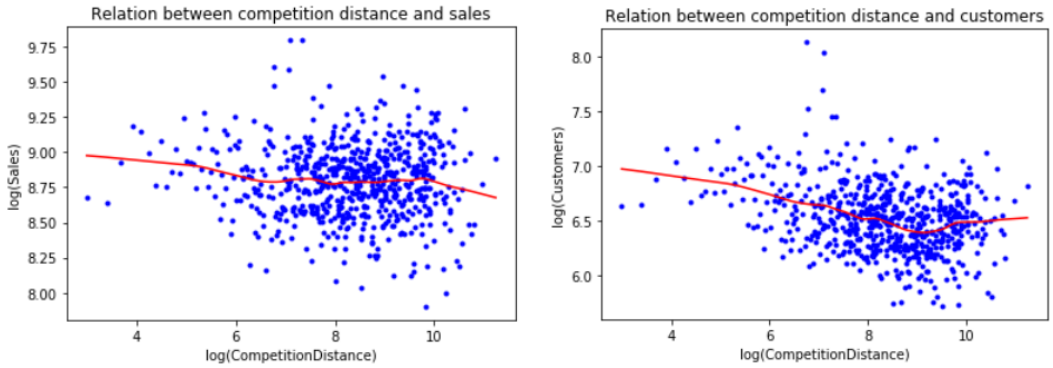


Figure 3.3.18 the graphs explaining the relationship between competition distance and sales/customers

The correlation shows a negative relationship between competition distance and sales/customers, meaning that when the distance to competitors is short, the sales and customers increased (Figure 3.3.18). This might be because of being in more crowded places and having higher sales in total.

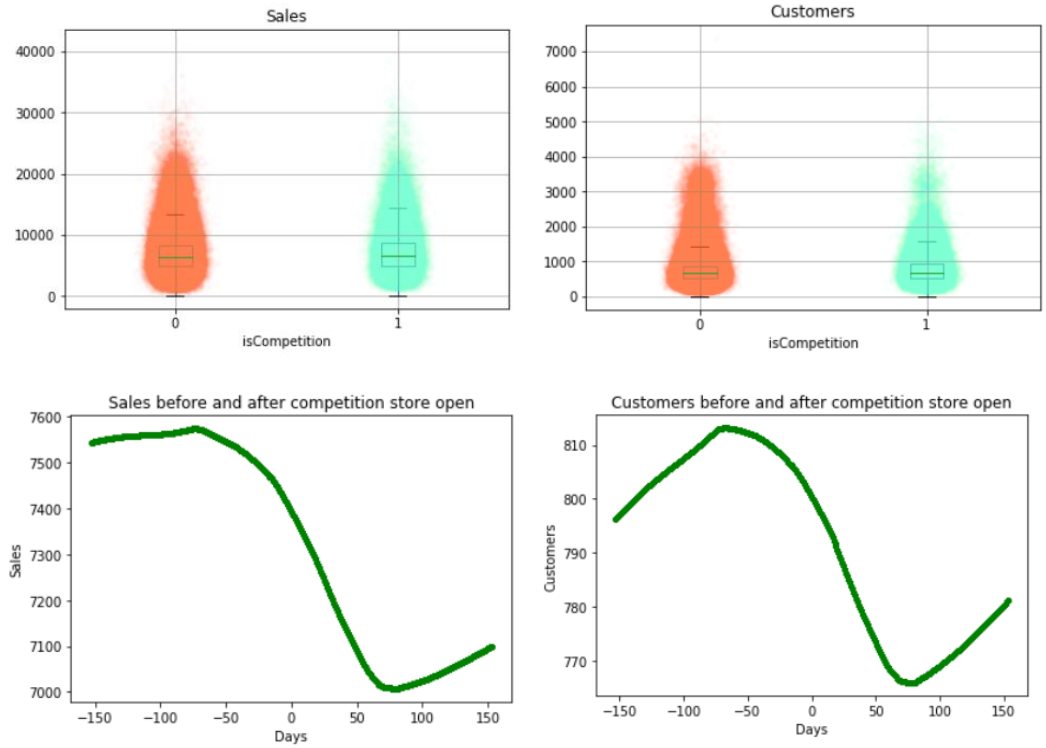


Figure 3.3.19 the sales and customers after competitor open

As we can see from Figure 3.3.19. If there is a competitor nearby, sales might slightly decrease as well as customers. If there was a competitor opened nearby, sales are significant dropped (Meaning that if there were only our stores in shopping center, sales or customer would be more). After competitors has opened nearby, the number of customers is significant dropped for a while and tend to increase.

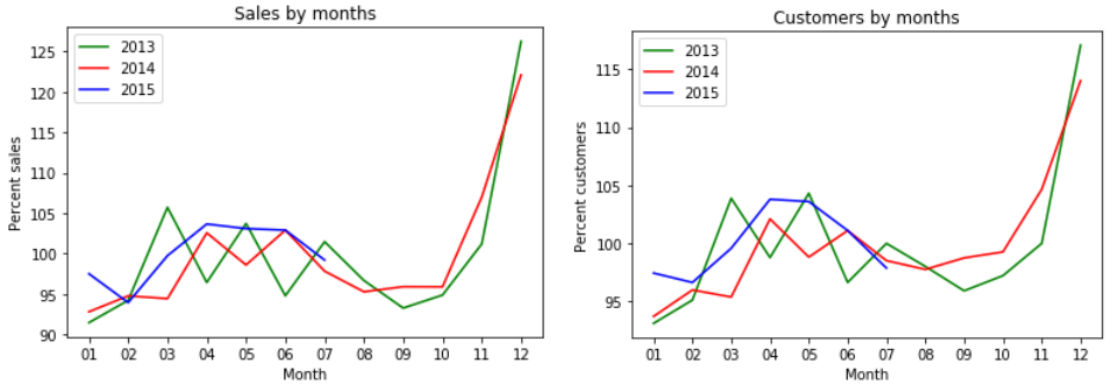


Figure 3.3.20 the graphs for sales/customers by months

We can say that 2014 were following the similar pattern that 2013 had. We can also assume that 2015 would be following this (since we are predicting 2015's sales). In addition, it is not hard to see the sales and customers increased hugely in Nov and Dec (Figure 3.3.20).

In next chapter, we will be exploring more about the imports features that we will use to train our model.

4. Features Ranking and selection

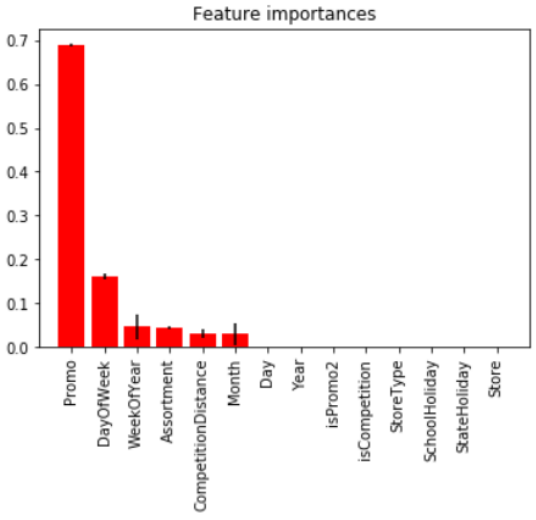
Firstly, we merged the train data and store data together. We added some features to our dataset. The first one is **isCompetition** column, if there exist competitors nearby. The second column is **Promo**, if there is any 'recent' promotion. Then we split the date data to be **year**, **month**, **day**, and **week of year**. The category columns are transformed to number. The result is in Figure 4.1. We will do the features importance on the result dataset.

*For the background of each Algorithm, please refer to section 7.

Store	DayOfWeek	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	isCompetition	isPromo2	Year	Month	Day	WeekOfYear
1	5	1	0	1	3	1	1270.000000	0	0	2015	7	31	31
1	4	1	0	1	3	1	1270.000000	0	0	2015	7	30	31
1	3	1	0	1	3	1	1270.000000	0	0	2015	7	29	31
1	2	1	0	1	3	1	1270.000000	0	0	2015	7	28	31
1	1	1	0	1	3	1	1270.000000	0	0	2015	7	27	31

Figure 4.1 the data after transformation

We first used the Random Forest algorithm with max depth equal 3 to find the features importance (Figure 4.2).




```
Feature ranking:
1. feature Promo (0.688098)
2. feature DayOfWeek (0.160879)
3. feature WeekOfYear (0.047234)
4. feature Assortment (0.043709)
5. feature CompetitionDistance (0.030922)
6. feature Month (0.029159)
7. feature Day (0.000000)
8. feature Year (0.000000)
9. feature isPromo2 (0.000000)
10. feature isCompetition (0.000000)
11. feature StoreType (0.000000)
12. feature SchoolHoliday (0.000000)
13. feature StateHoliday (0.000000)
14. feature Store (0.000000)
```

Figure 4.2 the features importance by Random Forest with 3 depths

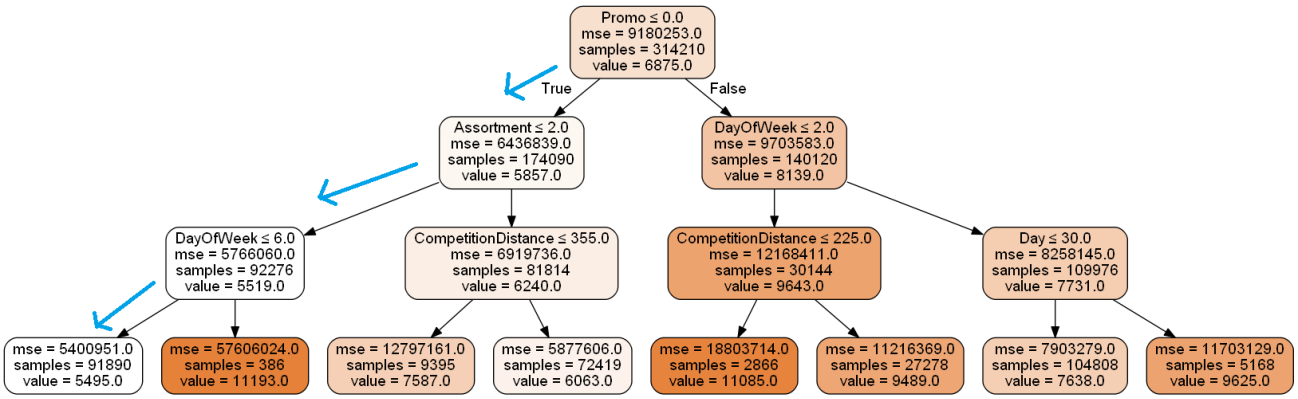


Figure 4.3 a single Decision Tree (with max-depth=3) from Random Forest

Figure 4.3 illustrates a prediction for a new data point of the store, which opens on Monday, April 2, 2018. We pick up one left-hand branch of tree (which is highlighted in the blue arrow) and its actual variables are **Promo = 0**, **Assortment = 'b'** (basic level), and **DayOfWeek= 1**.

We start at the root node and the first answer is true because the store does not run the promotion on that day.

We move to the left and have the answer is true for the second criteria as **Assortment <=2.**

It means that the Assortment level of store is basic (Assortment = 'b'). Move down to the left and on to the third question which is true as well because **DayOfWeek <=6**.

To sum up, our estimation for the store's sales is $5495 on that day as shown by the value of the leaf node.

From the root node, there are only 314210 samples despite there being 496976 training data points. This is because each tree in the forest is trained on a random number of the data points with replacement. The sample of data points and subset of the features at each node of the tree are random, that is why the model is called a 'random' forest. The interpretation of the tree diagram will be provided in Figure 4.4.

Root node

1) It is the topmost decision node in a tree which appropriates to the best predictor.

Internal nodes

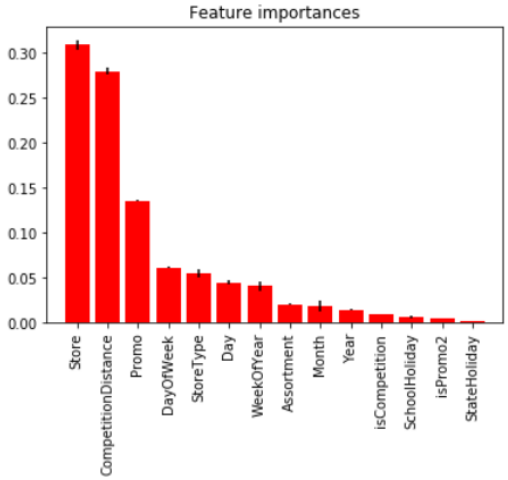
2) First line: The feature (variable) with its value is used as a condition to split node on (going to the left if true; otherwise, going to the right)
3) mse: the mean squared error of that node calculated by comparing the node's actual response values against its mean response value
4) samples: the number of data points in that node before splitting

Terminal nodes / Leaves

5) mse: the mean squared error calculated by comparing the actual response values in that node against "value"
6) samples: the number of data points (observations) in that node
7) value: predicted value for all data points in that node

Figure 4.4 the tree diagram interpretation

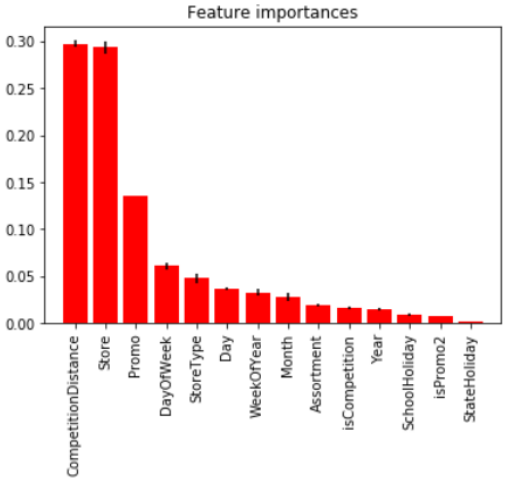
We will then use the Random Forest algorithm to find the important features. The result is in Figure 4.5.




```
Feature ranking:
1. feature Store (0.308725)
2. feature CompetitionDistance (0.279591)
3. feature Promo (0.135554)
4. feature DayOfWeek (0.061615)
5. feature StoreType (0.054834)
6. feature Day (0.044655)
7. feature WeekOfYear (0.040436)
8. feature Assortment (0.020117)
9. feature Month (0.017913)
10. feature Year (0.014513)
11. feature isCompetition (0.009501)
12. feature SchoolHoliday (0.006547)
13. feature isPromo2 (0.004393)
14. feature StateHoliday (0.001607)
```

Figure 4.5 the features importance by Random Forest

Lastly, the Extra Tree algorithm is used to find the feature importance (Figure 4.6).




```
Feature ranking:
1. feature CompetitionDistance (0.297520)
2. feature Store (0.293604)
3. feature Promo (0.135531)
4. feature DayOfWeek (0.060950)
5. feature StoreType (0.047653)
6. feature Day (0.036701)
7. feature WeekOfYear (0.032729)
8. feature Month (0.027524)
9. feature Assortment (0.018737)
10. feature isCompetition (0.016106)
11. feature Year (0.015271)
12. feature SchoolHoliday (0.008639)
13. feature isPromo2 (0.007490)
14. feature StateHoliday (0.001545)
```

Figure 4.6 the feature importance by Extra Tree

Although the result is slightly different than using Decision tree. That should not affect us much, since the top 5 features are the same. We can summarize that the top 5 features should be **Store**, **CompetitionDistance**, **Promo**, **DayOfWeek**, and **StoreType**. Of course, we will add/remove more features if anything necessary.

Before we get into machine learning models, we will first see a famous statistical model - Time Series.

5. Time-Series

Time-Series Analysis is the use of a statistical model to predict future values, based on past results. Any kinds of values that can be tracked and collected over time can be used time-series model e.g. annual population, company's daily stock price, quarterly sale figures. The time-series dataset is depended on time and order changing could change the meaning of the data. Each interval of the data has to be the same and have at most 1 data point.

There are 2 main methods of doing time-series analysis: ETS and ARIMA.

Error, Trend, Seasonality (ETS)

ETS uses weighted averages of past observations, giving more weight to the most recent observation with weight gradually getting smaller as the observation gets older. Three parts of ETS are Error, Trend, and Seasonality.

Seasonality shows the seasonal pattern ie. occur at a regular interval, usually the same period of the year. It can be constant or increasing magnitude.

Trend is the general course or tendency of the time series. It is a centered-moving-average of the time series and fits between the seasonal peaks and valleys.

Error or remainder is the piece that is not accounted for by combining the seasonal piece and the trend piece.

Autoregressive Integrated Moving Average (ARIMA)

ARIMA is the method of transforming time series data into more workable format to forecast. There are 2 types of ARIMA: Seasonal and Non-Seasonal. Three parts of ARIMA are Autoregressive, differencing, and moving average.

Autoregressive term (p) is a linear regression with predictive variables (certain number of previous periods).

Differencing or integrated (d) is the processing we use to transform a time series into a stationary model (series without trend or seasonality). D is the number of transformation used in the process.

Moving average term (q) is the lag of the error component or the previous q period of the error.

Autocorrelation is how correlated a time series is with its past values.

Autocorrelation function plot (Correlogram/ ACF) shows a series correlated with itself. Vertical axis is correlation coefficient. Horizontal axis is number of lag.

Partial Autocorrelation Function plot (PACF) is the correlation between two variables controlling for the values of another set. PACF shows the correlation of the previous points, controlling for the values of all previous lag variables.

5.1 Time-Series Analysis

For the ease of presentation, I will select 1 store per each store type, which are store 2, 85, 1, and 15 for A, B, C, and D respectively (there is no particular reason of selection), to do Time series Analysis.

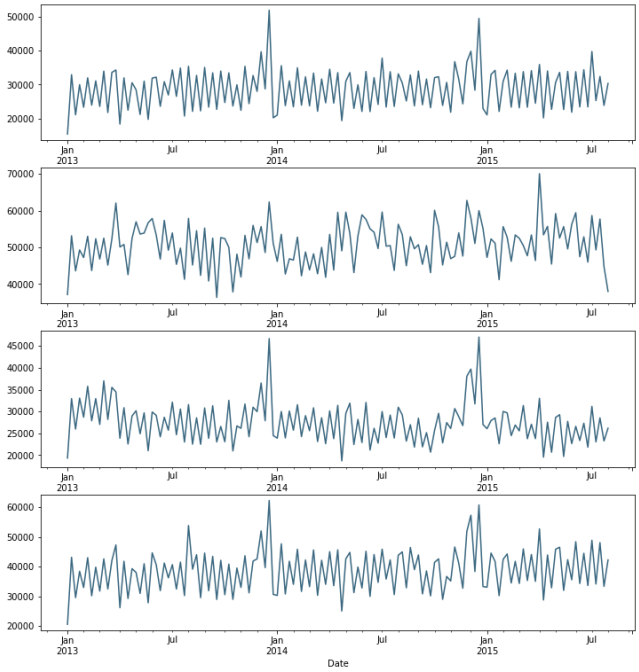


Figure 5.1.1 store 2, 85, 1, and 15 representing store type A, B, C, and D in order, respectively

Firstly, as you can see on the figures 5.1.1, store type A and C have sudden peaks around December every year. Store type B and D are also peak at around December, but they are as obvious as A and C.

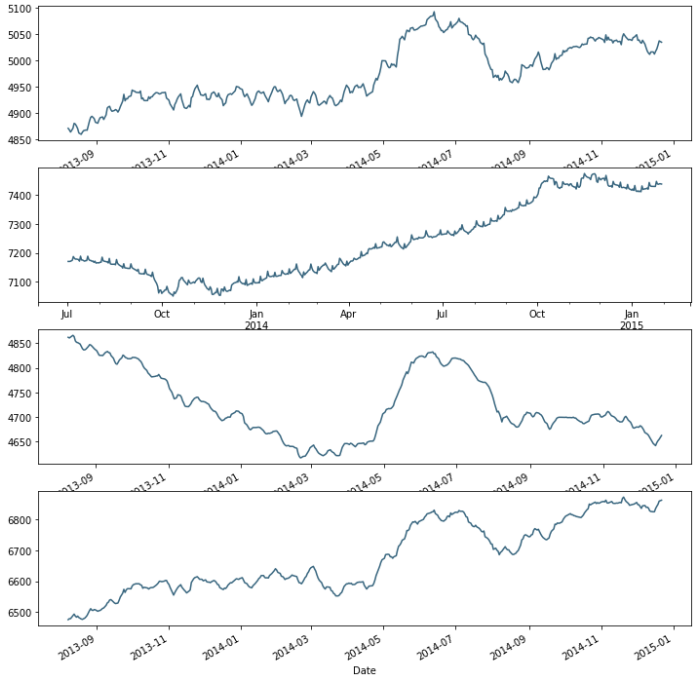


Figure 5.1.2 the trend for store type A, B, C, and D, in order

Next, one of the most important components of the time series, **trends**. According to the figures 5.1.2, the trends of store type A, B, D are upwards while that of store type C is downwards. Notably, the trends of store type A, C and D have a sudden jump on May 2014 to August 2014.

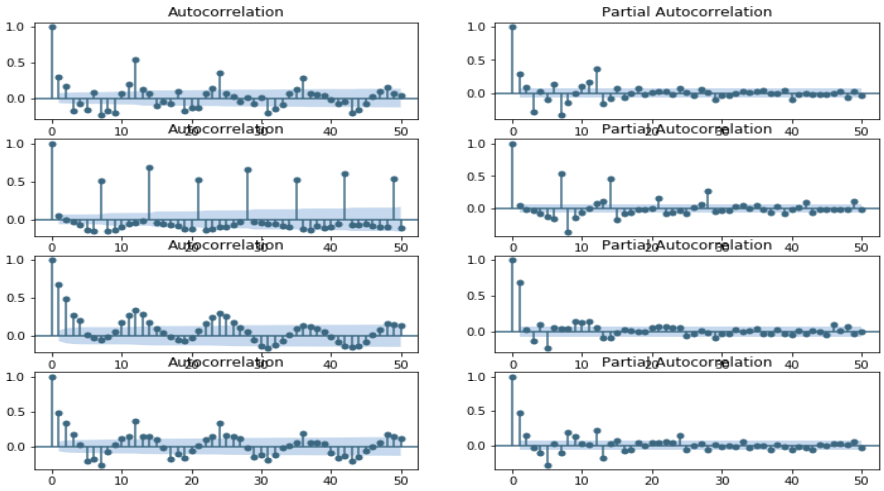


Figure 5.1.3 ACF and PACF for store type A, B, C, and D, in order

Next, we will look at the **autocorrelation functions** (**ACF**) and **Partial- autocorrelation functions** (**PACF**). Figures 5.1.3 shows that store type A has high correlation pattern at 12X. For examples, 12th , 24th , 36th and store type B has high correlation every 7 days or weekly seasonality. Store type C and D are more complex as they are correlated to their adjacent points.

5.2 Time-Series Modelling

We will use the Facebook's Prophet library to build the time-series models in this section. Just like the previous part, store 2, 85, 1, and 15 are selected to represent the store type A, B, C, and D respectively. In addition, we will use the last month in the dataset to be the test set, which is July 2015. Next, we will remove the records when the stores are closed and no sale from the train data to reduce bias toward 0. Below are the figures of train data (as we mention in chapter 3.2).

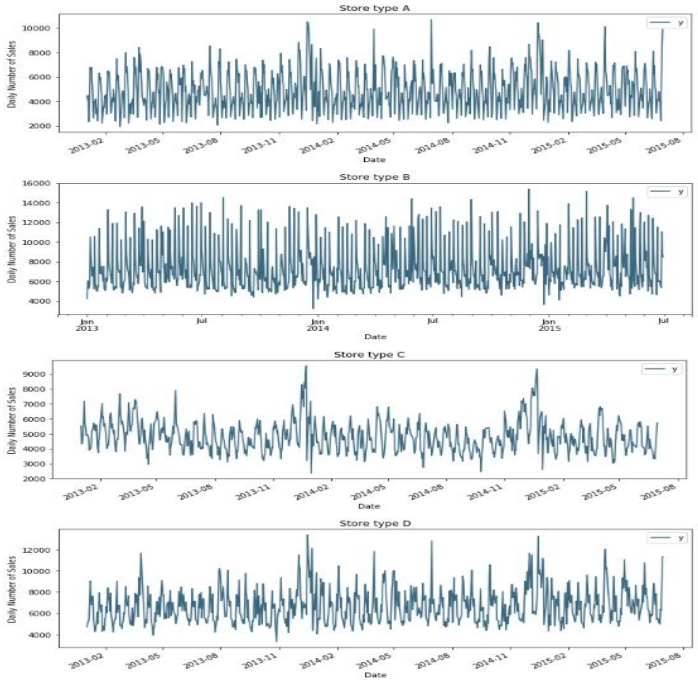
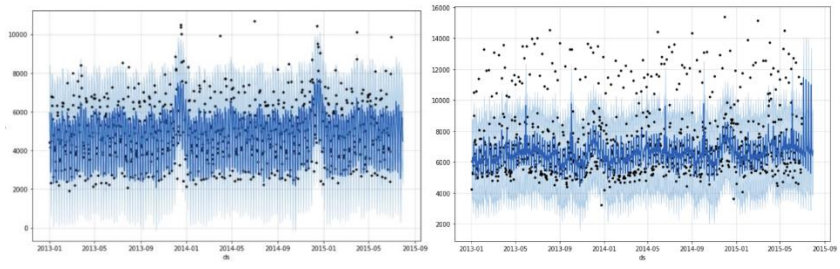


Figure 5.2.1 the time series of train data

Prophet also considers holidays into account, so we can include the holiday information to the models. Next, we will train the models separately one for each store type with 95% uncertainty interval (confident interval). The results of each model are as bellow.



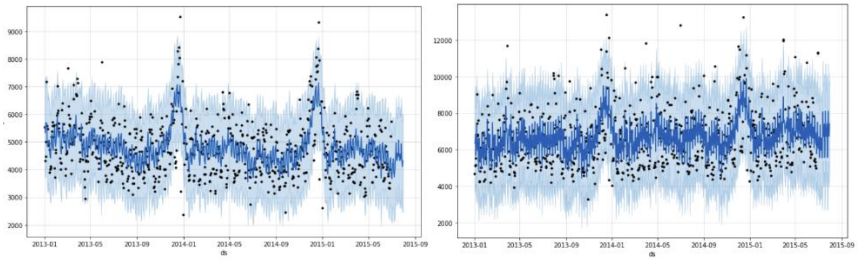


Figure 5.2.2 models - top left: type A, top right: type B, bottom left: type C, bottom right: type D

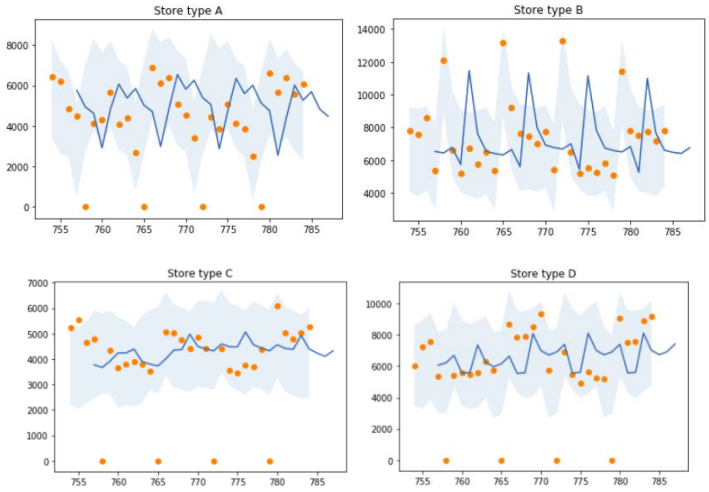


Figure 5.2.3 the results of predicted periods

The results of the predicted periods (last month) are quite great. As you can see from the figures 5.2.3, almost all of the true values are in the confident intervals of the predicted values (blue areas). The note is that there are some points of 0 sales (the orange points at the bottom part of each figure) which are out of the scope of the models.

We will move onto the most important part – Machine learning in the next chapter.

6. Model Prediction

In this section, we will discuss the Machine Learning theories and its algorithms that we have used. The results and evaluations will be covered on the next section.

6.1 Linear Model

Linear Regression:

Linear Regression is the oldest, simple and widely used supervised machine learning algorithm for predictive analysis. It is a method to predict a **target variable** by fitting the *best linear relationship* between the dependent and independent variable.

Multi-Linear Regression:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_n X_n + \varepsilon.$$

Multi-Non-Linear Regression:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2^2 + \beta_3 X_3^3 + \cdots + \beta_n X_n^n + \varepsilon.$$

where β_n are the weight and x_n are the data.

Ridge Regression/ Lasso Regression:

The difference between Ridge Regression and Lasso Regression is the penalties, as explained below,

$$p_\alpha = \sum_{i=1}^{p} \left[\frac{1}{2}(1-\alpha)_j^{b^2} + a|b_j|_i\right],$$

Where a and b are constants.
If i=1(L1 norm), then we have a Ridge Regression.
If i=2(L2 norm) then we have a Lasso Regression.

Elastic Net Regression:

$$p_\alpha = \sum_{i=1}^{p} \left[\frac{1}{2}(1-\alpha)_j^{b^2} + a|b_j|_1 + a|b_j|_1 + a|b_j|_2\right],$$

Where a and b are constants.
Elastic Net Regression is the regularization of L1 norm plus L2 norm.

Algorithm for regression:
The Algorithm that we will use is call Gradient Descent, which search the maximum/minimum value of gradient.

Step 1: Input data x, labels y, learning rate alpha.

Step 2: Picking initial w^0 randomly.
Step 3: Repeat: $w^{S+1} \coloneqq w^S + \alpha \Sigma x_i (y_i - w^T x_i)$ until it converges to $|w^{s+1} - w^s| < \varepsilon$.
Step 4: Output w^s.

6.2 Decision Tree Regression

A tree has many analogies in real life and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, it's also widely used in machine learning.(Prashant Gupta, Toward Data science)

Algorithm for Decision Tree:
Response function is continuous.

Goal: select a partition of regions (nodes): R_1, …, R_M, so that the response can be modeled as a constant c_m in each region.

Step 1: For a splitting variable X_j and a splitting point s, define

$$R_1(j,s) = \{X \mid X_j \le s\}$$
$$R_2(j,s) = \{X \mid X_j > s\},$$

Seek j and s, so that the sum of squared error is minimized.

$$\sum_i (y_i^{(1)} - \bar{y}_i^{(1)})^2 + \sum_i (y_i^{(2)} - \bar{y}_i^{(2)})^2,$$

Step 2: For each R_M, refine the partition by repeating step 1, stop when the number of nodes reaches a predefined cutoff.

6.3 Ensemble Method

Extra Tree Regression:

The Extra Tree (Extremely Randomized Trees) algorithm is one of the ensemble method, which splits nodes randomly from the training data and uses it to build the several trees. The brief explanation of splitting the node is provided on the following figure.

Split_a_node(S)
Input: the local learning subset S corresponding to the node we want to split
Output: a split $[a < a_c]$ or nothing
– If **Stop_split**(S) is TRUE then return nothing.
– Otherwise select K attributes $\{a_1, \ldots, a_K\}$ among all non constant (in S) candidate attributes;
– Draw K splits $\{s_1, \ldots, s_K\}$, where $s_i = \textbf{Pick\_a\_random\_split}(S, a_i), \forall i = 1, \ldots, K$;
– Return a split s_* such that $\text{Score}(s_*, S) = \max_{i=1,\ldots,K} \text{Score}(s_i, S)$.

Pick_a_random_split(S,a)
Inputs: a subset S and an attribute a
Output: a split
– Let $a^S_{\max}$ and $a^S_{\min}$ denote the maximal and minimal value of a in S;
– Draw a random cut-point a_c uniformly in $[a^S_{\min}, a^S_{\max}]$;
– Return the split $[a < a_c]$.

Stop_split(S)
Input: a subset S
Output: a boolean
– If $|S| < n_{\min}$, then return TRUE;
– If all attributes are constant in S, then return TRUE;
– If the output is constant in S, then return TRUE;
– Otherwise, return FALSE.

Figure 6.3.1 algorithm of Extra Tree

Random Forest Regression:

Random Forests are a combination (ensemble) of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Random Forest is capable of regression and classification. It can handle a large number of features, and it is helpful for measuring variables importance.

Algorithm for Random Forest

The pseudo code for random forest algorithm can split into two stages.

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Figure 6.3.2 algorithm of Random Forest

Ada Boost Regression:

AdaBoost can combine with other algorithms to improve the overall performance of the model. The weighted sum from the previous model will be used in the next model. AdaBoost will pick only those

features, which can improve the accuracy of the model in training process. At each iteration, it will fit a sequence of weak learners by updating the weights.

1: Init data weights $\{w_n\}$ to $1/N$
2: **for** $m = 1$ to M **do**
3: fit a classifier $y_m(x)$ by minimizing weighted error function J_m:
4: $J_m = \sum_{n=1}^{N} w_n^{(m)} 1[y_m(x_n) \neq t_n]$
5: compute $\epsilon_m = \sum_{n=1}^{N} w_n^{(m)} 1[y_m(x_n) \neq t_n] / \sum_{n=1}^{N} w_n^{(m)}$
6: evaluate $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
7: update the data weights: $w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m 1[y_m(x_n) \neq t_n]\}$
8: **end for**
9: Make predictions using the final model: $Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$

Figure 6.3.3 algorithm of Ada Boost

6.4 Model Evaluation

After we got feature importance ranking and understand the methods. We will use those methods to train our model. In addition, we will use that **Root Mean Square Percentage Error (RMSPE)** to evaluate the accuracy of the model.

Root Mean Square Percentage Error (RMSPE):

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

Where n is the sample size, y_i is actual value, $\hat{y}_i$ is the predicted value.

Type	Model	RMSPE	
		All feature	Top 5 features
Linear Model	Linear Regression (Degree = 1)	0.358688	0.395980
	Linear Regression (Degree = 2)	0.349503	0.392817
	Linear Regression* (Degree = 3)	0.725115	0.384141
	Ridge Regression (Alpha = 0.05)	0.358687	0.395980
	Ridge Regression (Alpha = 1)	0.358683	0.395980
	Ridge Regression (Alpha = 10)	0.411515	0.419454
	Lasso Regression (Alpha = 0.05)	0.357880	0.395979
	Lasso Regression (Alpha = 1)	0.357768	0.395951
	Lasso Regression (Alpha = 10)	0.358405	0.395709
	Elastic Net Regression (Alpha = 0.05)	0.358038	0.394815
	Elastic Net Regression (Alpha = 1)	0.376812	0.405028
	Elastic Net Regression (Alpha = 10)	0.400399	0.420736
Ensemble Method	Extra Tree Regression	0.249020	0.163912
	Random Forest Regression (Max Depth = 3)	0.382055	0.383146
	Random Forest Regression (Max Depth = Infinity)	0.197128	0.164289
	Ada Boost Regression	0.193825	0.164329

Table 6.4.1 The RMSPE of each model

*Note that there exists the case of overfitting when the degree of polynomial is greater than 3 for linear regression.

According to Table 6.4.1, we used several models to train the data, Linear models and Ensemble methods.

At first, we used all features to train the model. We realized that the result can be improved by using reducing the useless features.

In term of regression modelling, if we use all futures to train the model, the error is lower than using top 5 features. Apart from Random Forest (max depth= infinity), Ensemble method is the opposite. Ada Boost Regression, Extra tree regression and Random Forest (max depth=3) are having the best result, which is approximately 0.164. In addition, the regression model with top 5 features has the worst result which are around 0.40.

6.5 Sales Forecasting

Eventually, we will use the Random Forest with top 5 features to forecast the sales of store 3.

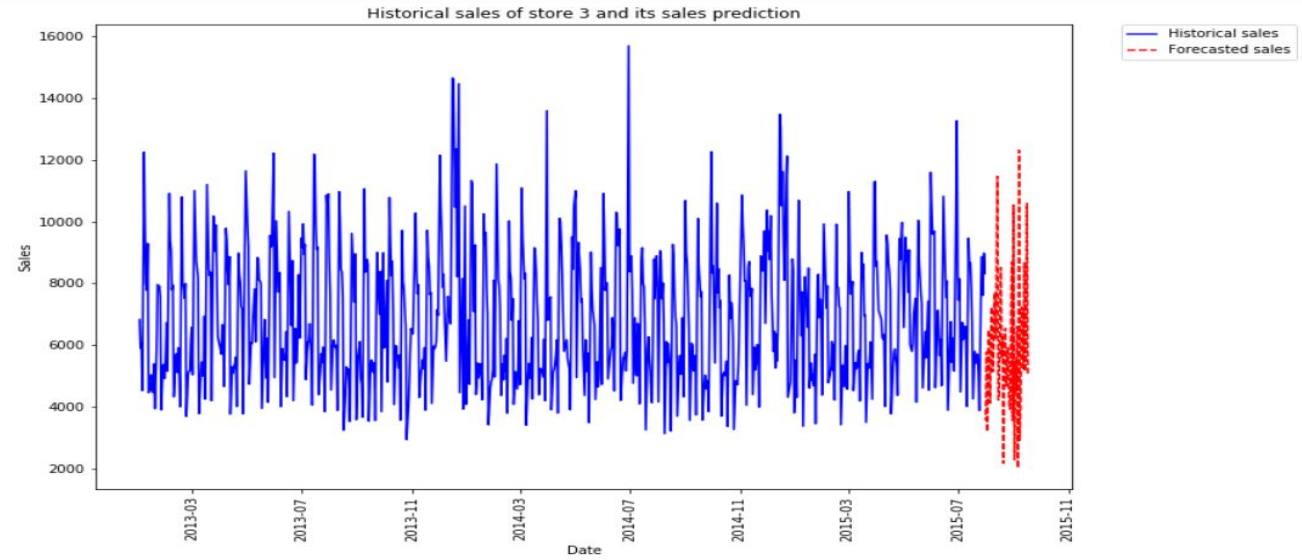


Figure 6.5.1 The historical sales and its prediction of store 3

In Figure 6.5.1, the blue line is the historical sales of store 3. The forecast of sales is the red line. Interestingly, some of the predicted value reached global minimum, which means the store will be facing the lowest sales, compared to the historical data.

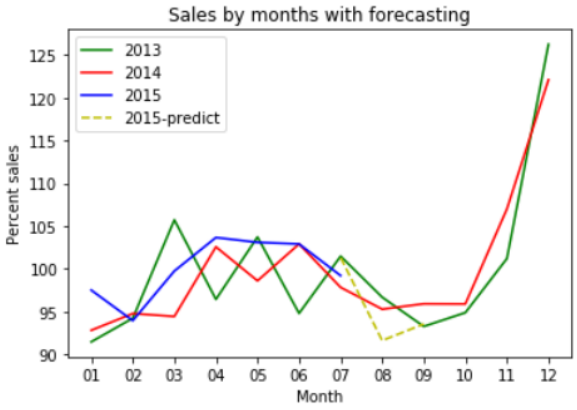


Figure 6.5.2 the overall historical sales by years and its prediction

Next, we will predict the overall trend of the stores (Figure 6.5.2). The yellow line indicates the future trend.

7. Conclusion

We picked the Rossmann sale dataset to do an analysis. We applied what we leant in the Data Analytics course to the project, namely data selection, data exploration, data cleaning, data preparation, data modeling, and model evaluation. Additionally, we applied several data analysis methods and models to the dataset, such as time-series analysis, linear regression, multiple linear regression, random forest etc.

As mentioned in the data exploration, some stores (eg. store type B) have a lot of potential to grow. Some store such as A and D are at the saturated stage where their sales were not increased and needed to be introduced something new to the stores.

The initial goal is that we would like to do a practical project where we can apply what we have learned in this course to a real life. We particularly selected time-series dataset since we believe that almost all companies have forecasting problems and it is also crucial for them to be able to utilize the insight from their data. We believe that Rossman Sale is one of the perfect match in this case.

The key challenges in this project is the nature of the dataset which is a time-series, where it is needed some specific ways to handle. However, it is not easy to handle a time-series dataset by only having one time-series lecture, meaning it required a very good preparation. Working as a group allowed us to help each other, which significantly improved our project management skill, communication skill and coding skill. These have given us a very good opportunity to be ready for the real-life's problems.

As we mentioned, time-series analysis takes a very important role in real life. One of model application weather prediction, using either regression modelling or time series.

The possible extension in the future is that we can add more features by integrating external API such as weather forecast to get the weather condition for each day, to check if the weather condition would affect our model.

We would like to say thank you to our instructors, Dr. Anthony Constantinou and Dr. Bhusan Chettri as well as the demonstrators for giving us this opportunity to do this amazing project and giving us the knowledge to carry on.

8. Bibliography

BREIMAN, L. (2001). Random Forests. *Machine Learning, 45*, 5–32.

Chakon, O. (2017, 8 3). *PRACTICAL MACHINE LEARNING: RIDGE REGRESSION VS. LASSP*. Retrieved from Coding Startups: https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/

Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly Media.

Geurts, E. &. (2006). Extremely randomized trees. *Machine Learning, 63*(1), 3-42.

Gupta, P. (2017, 05 17). *Decision Trees in Machine Learning*. Retrieved from Towards Data Science: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Himrod, D. (29, 12 2017). *Pandas*. Retrieved from Python Data Analysis Library: https://pandas.pydata.org/

McKinney, W. (2017). *Python for Data Analysis, 2nd Edition.* O'Reilly Media.

Ng, A. (2018). *Features and Polynomail Regression*. Retrieved from Coursera: https://www.coursera.org/learn/machine-learning/lecture/Rqgfz/features-and-polynomial-regression

Nigel Duffy, D. H. (2002). Boosting Methods for Regression∗. *Machine Learning, 47*, 153–200.

Petrova, E. (n.d.). *Time Series Analysis and Forecasts with Prophet*. Retrieved from Prophet: https://www.kaggle.com/elenapetrova/time-series-analysis-and-forecasts-with-prophet

Prophet. (2017). Retrieved from https://facebook.github.io/prophet/docs/quick_start.html

Sarah Guido, A. C. (2016). *Introduction to Machine Learning with Python.* O'Reilly Media.

Udacity. (2017). Retrieved from Time Seires Forcasting: https://auth.udacity.com/sign-in?next=https%3A%2F%2Fclassroom.udacity.com%2Fauthenticated

Waskom, M. (2017). *Seaborn*. Retrieved from https://seaborn.pydata.org/

9. Appendices

Our source codes are in ipynb format, which can be opened on the **Jupyter Notebook**. We also split it into several files according to the main sections.

Notebook for Data Exploration

It is located at the same folder with file named "**DataExploration.ipynb**"

Notebook for Time-Series Analysis

It is located at the same folder with file named "**TimeSeriesAnalysis.ipynb**"

Notebook for Features Importance

It is located at the same folder with file named "**FeaturesSelection.ipynb**"

Notebook for Prediction and Evaluation

It is located at the same folder with file named "**PredictionAndEvaluation.ipynb**"