



Rossmann Store Sales

Machine Learning Project

Team name: Dilbar

Team Members:

Apurva Bhatt IMT2016010

Swapnil Buchke IMT2016085

Kartik Gupta IMT2016128

Important links

[Link](#) to IIIT-B inclass competition

[Link](#) to Rossmann Sales competition

[Link](#) to leaderboard

[Link](#) to colab research(the jupyter notebook where we worked)

Problem Statement

Rossmann operates over 3,000 drug stores in 7 European countries. The sales are influenced by several factors like the holiday(school, state), the day of the week, type of store etc. We have to make a prediction model to predict one day sales of 1,115 drug stores owned by Rossmann.

About the Data

The dataset is distributed in three files :

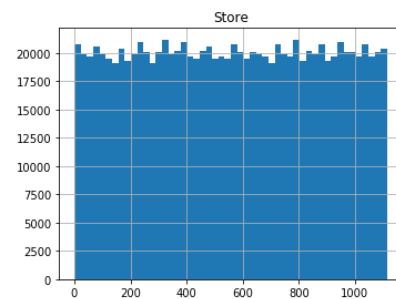
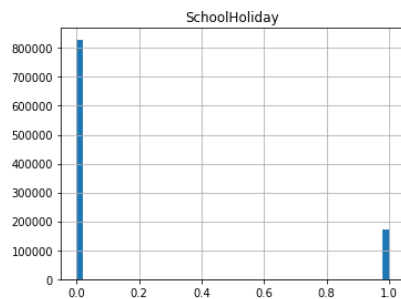
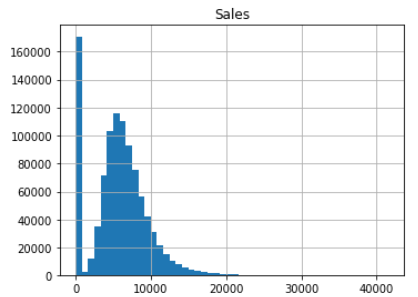
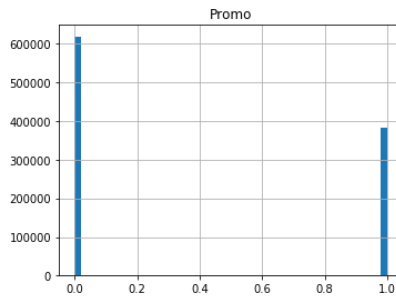
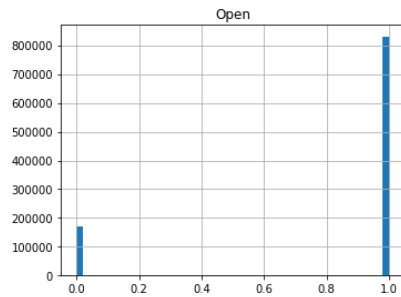
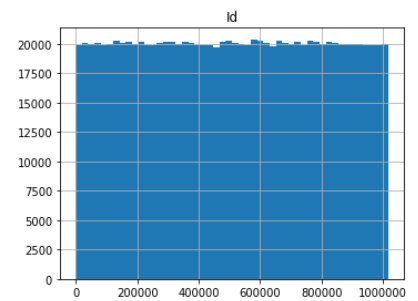
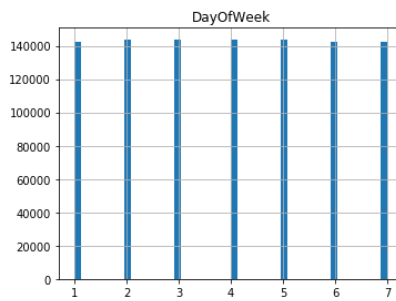
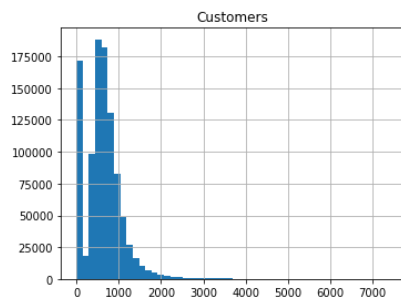
1. store.csv: It contains the data regarding each store like its type, assortment, competition distance etc
2. test.csv: It contained the data of the stores regarding which we have we have to predict the sales.
3. train.csv: It contains the data of the stores on which we have to train the model. The columns in test and train are the same.

Visualisation

A quick glimpse at the data on hand:

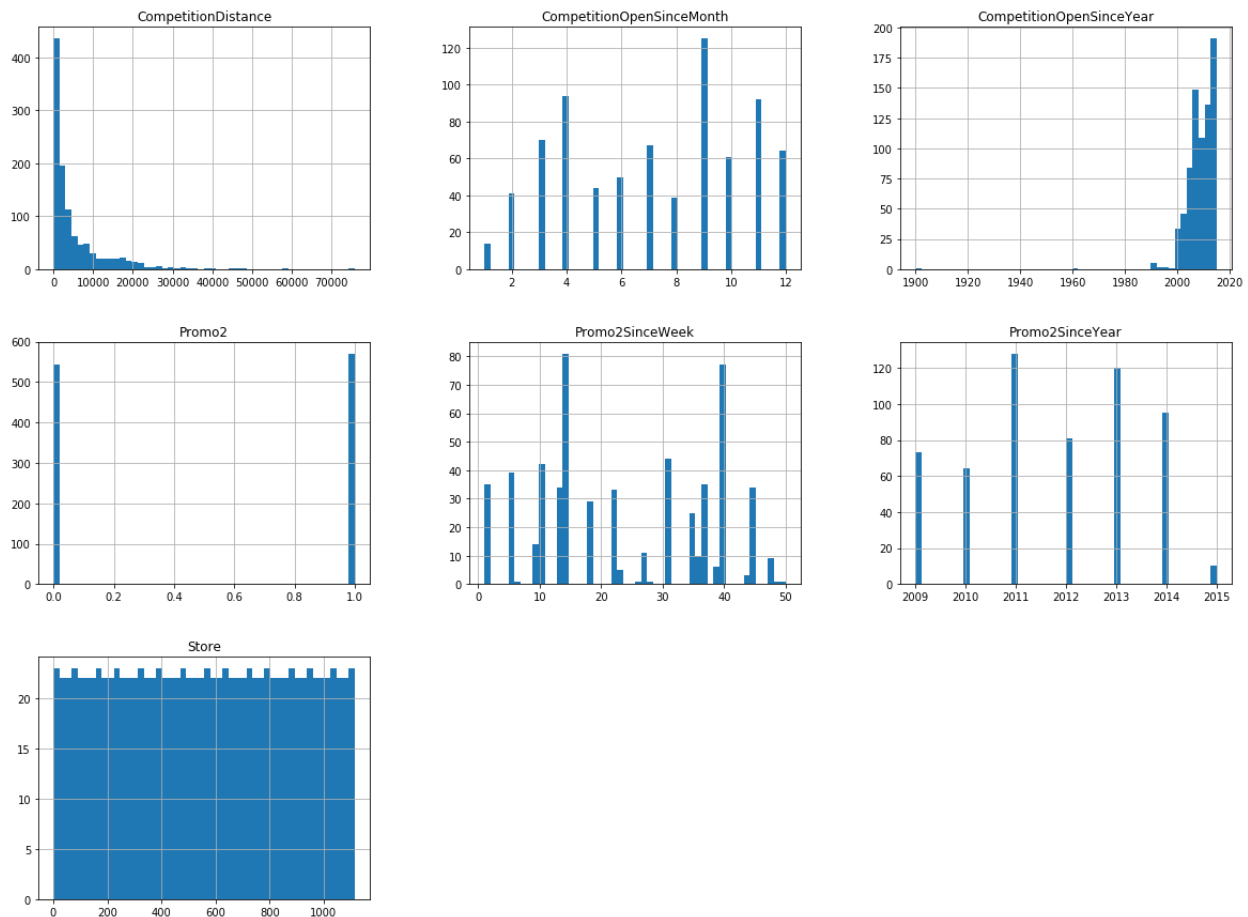
The Train dataset has 1001599 Rows and 10 Variables :

```
Store          int64
DayOfWeek      int64
Date           datetime64[ns]
Sales          int64
Customers      int64
Open           int64
Promo          int64
StateHoliday   object
SchoolHoliday  int64
Id             int64
dtype: object
```



The Store dataset has 1115 Rows (which means unique Shops) and 10 Variables :

Store	int64
StoreType	object
Assortment	object
CompetitionDistance	float64
CompetitionOpenSinceMonth	float64
CompetitionOpenSinceYear	float64
Promo2	int64
Promo2SinceWeek	float64
Promo2SinceYear	float64
PromoInterval	object
dtype: object	



Checking missing values :

Data columns (total 10 columns):

Store	1001599	non-null	int64
DayOfWeek	1001599	non-null	int64
Date	1001599	non-null	object
Sales	1001599	non-null	int64
Customers	1001599	non-null	int64
Open	1001599	non-null	int64
Promo	1001599	non-null	int64
StateHoliday	1001599	non-null	object
SchoolHoliday	1001599	non-null	int64
Id	1001599	non-null	int64

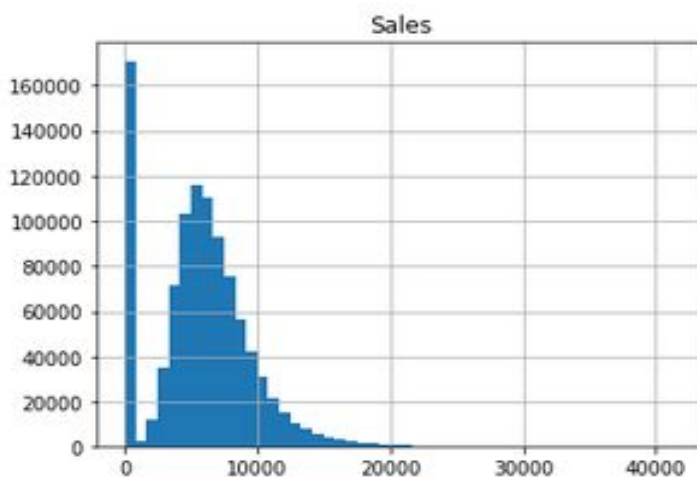
Here we can see that the data is complete and there are no missing values.

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

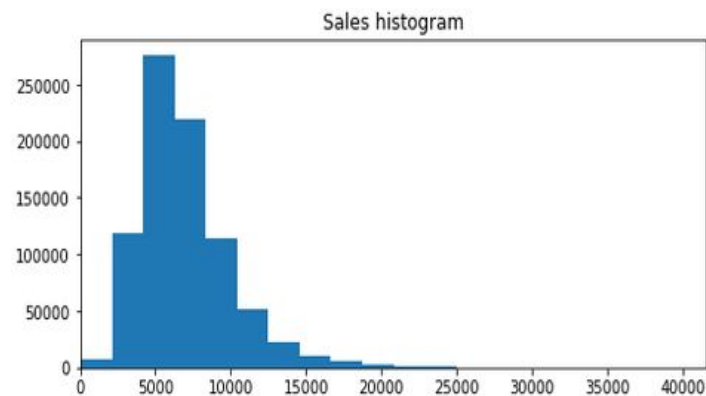
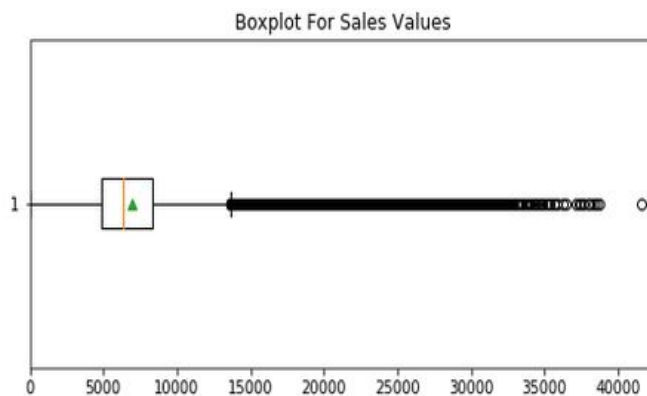
Whereas in store data we have some fields which have a large amount of missing data.

A Closer look at the Train set and store Set:

Sales :

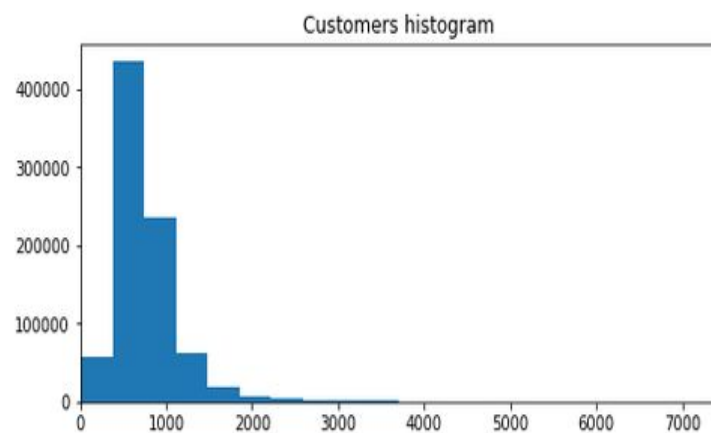
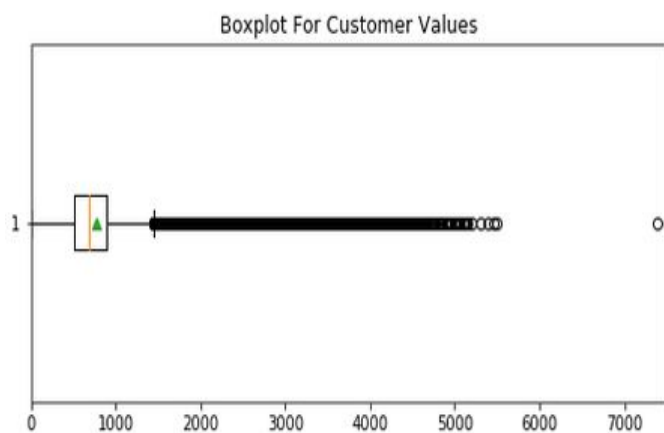


We can see that there is a large number of data points where sales are equal to zero. From the data we can also easily conclude that these are the same data points which were closed at that date, so we need to get rid of such data points.



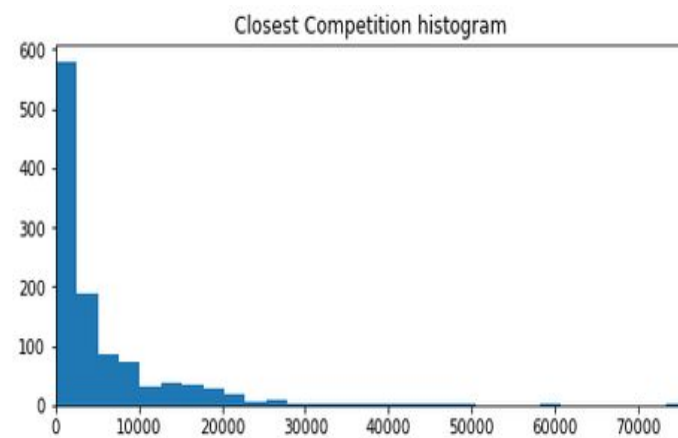
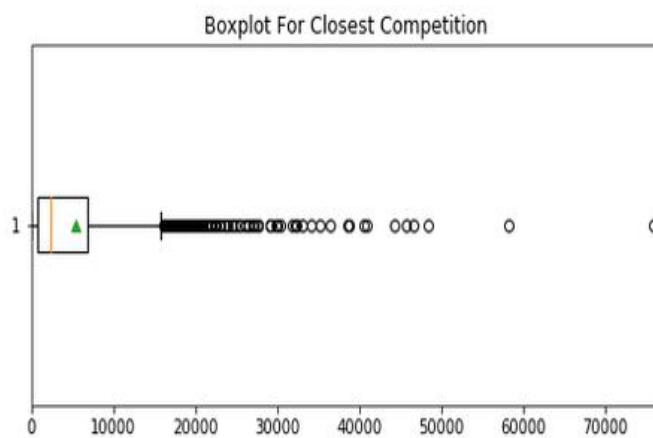
We can see that some exceptions (outliers) in the boxplot had to be checked to see if it's wrong inputted data but it turns out this big amount of sales on certain days is explained by either promotional purposes, the type of the store being big and popular or just not having near enough competition.

Customers :



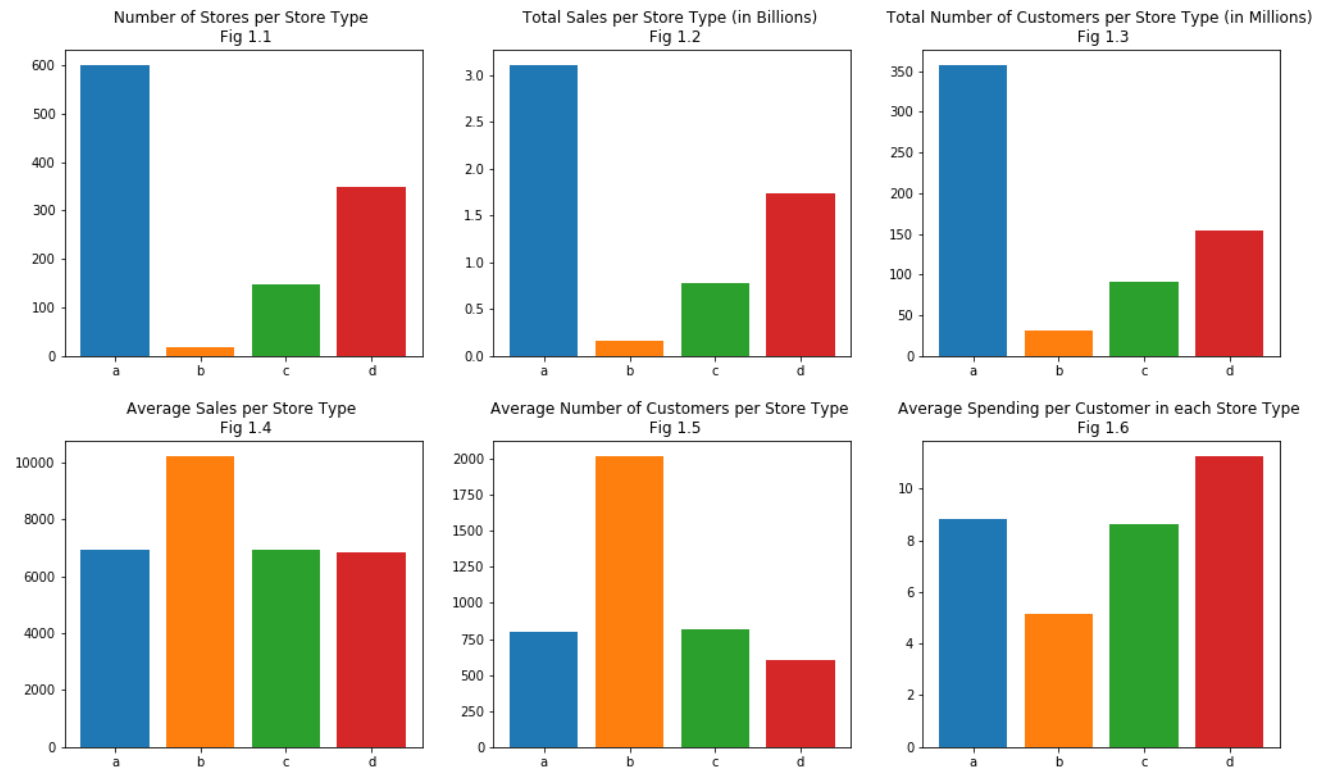
We can see similar patterns with the customers column and the Sales column, in fact our pearson correlation factor of 0.82 explains that there is a strong positive correlation between Sales and Customers. In general, the more customers you have in a store, the higher your sales for the day.

Competition Distance :



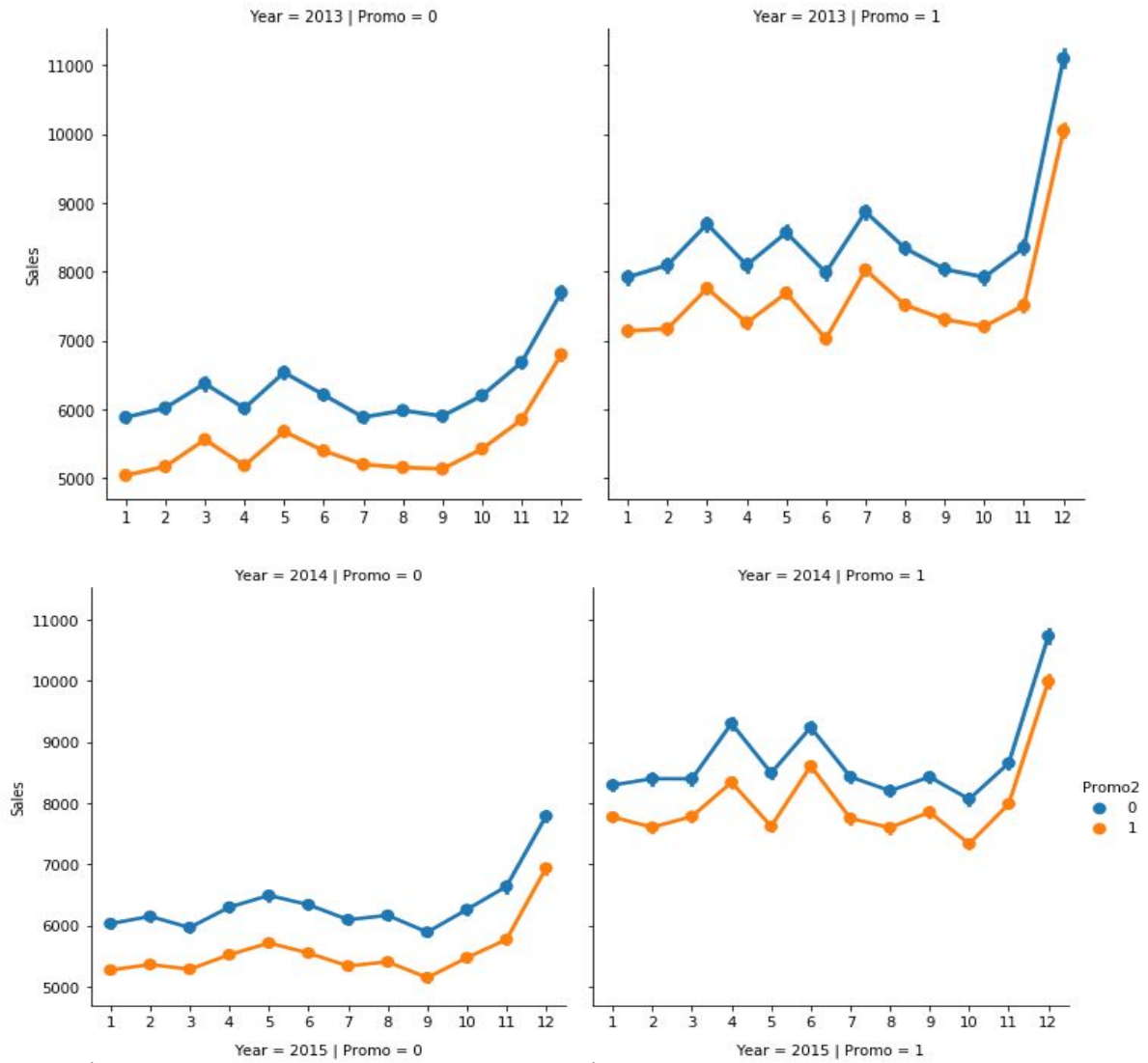
We see a highly right-skewed distribution for this variable with a significant difference between the mean and the median.

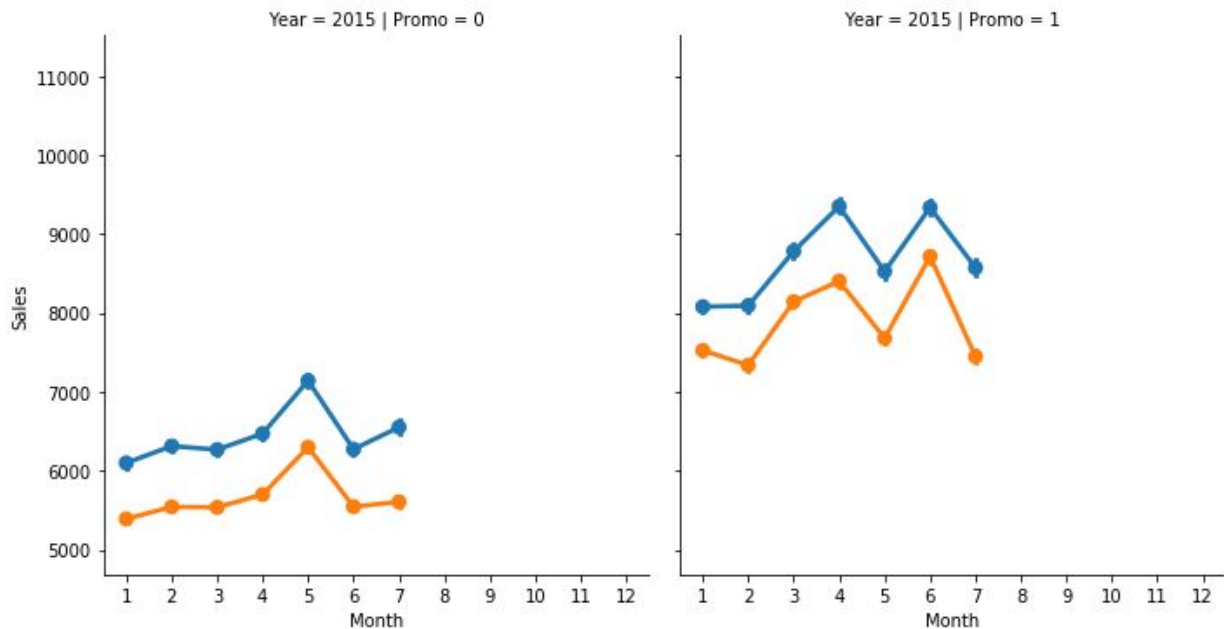
Store Type Analysis :



- From this training set, we can see that Storetype A has the highest number of branches, sales, and customers from the 4 different Storetypes.
- When looking at the average sales and number of customers, we see that actually, it is Storetype B who was the highest average Sales and highest average Number of Customers. One assumption could be that if B has only 17 stores but such a high amount of average sales and customers that it is likely hyper Rossman branches whereas A would be smaller in size but much more present.
- Surprisingly it is StoreType D who has the highest average spending per Customer, this is probably explained by an average competition distance higher than the rest which means each customer will buy more since he knows there isn't a lot of similar shops around.

Promotion :





- We see the dramatic change when we compare having promotion Promo=1 to not having promotion Promo=0 and can conclude that a store that has a promotion on a given day changes its amount of sales considerably.
- But Surprisingly, when we check more granularly at the Promo2 variable we see that in general when there is no consecutive promotion stores tend to sell more than with consecutive promotion. This is probably a solution they're putting in place to treat stores with very low sales in the first place. And indeed when checking the Sales per Customer over promotion we understand that initially those stores suffer from low sales and those continuous promotion shows a tremendous increase in the buying power of customers.
- If we look over the years, there is a slight increase Year over Year but we don't see any major change from 2013 to 2015 and we actually see a very similar pattern in the months over the years with major spikes first around Easter period in March and April than in Summer in May, June, and July and then finally around the Christmas period in November and December.

Pre-processing and feature engineering

- We have done all the pre-processing and feature engineering in the online jupyter notebook [Link](#)(google colab research) it helped the team of programmers to stay synced.
- We have used linear regression while performing pre-processing and featuring. As linear regression is an inexpensive operation, we can easily check whether the new change in pre-processing has made a positive or negative effect on our error.
- We have tried different approaches for pre-processing and feature engineering. Out of which the following two approaches gave the best results.

One-hot and removal of null value columns

We have first tried to use the domain knowledge and our intuition for pre-processing.

- a. We removed "Promo2SinceWeek", "Promo2SinceYear" and "PromoInterval" from store data as they contained nearly half of the values as NULL.
- b. We did one hot encoding on "StoreType", "Assortment", "PromoInterval" on store data and "DayofWeek" on train and test data.
- c. In train data, the maximum contribution is by the column "DayofWeek" to the regression. So, we chose to perform one-hot encoding in it.
- d. By using this method our error is 0.1996

Label encoding

By keeping our intuition on the side, we tried an alternative to our previous approach and deeply understood the data and came up with the following results.

- a. We performed one-hot encoding on the "StoreType" and "Assortment" in store.
- b. Filled all null values of store data with -1.
- c. In train data, we encoded "StateHoliday" by assigning 1 if there is a holiday and 0 otherwise. There are 3 types of holidays mentioned in the StateHoliday namely "a", "b" and "c" which are a public holiday, Easter holiday and Christmas respectively.
- d. We chose to remove "PromoInterval" from store as it contributed negligible to the result and had too many null values.
- e. By using this method our error is 0.1904

- Both the approaches give nearly equal results, but the second approach gives a slightly better result. So, we chose it for making our final model
- We are considering only the cases in the train data which are open and the sales are >0, which is 729319(1001599-830918). As keeping them only increases the error.
- We have separated day, month and year from the date column and made a separate column of each of them.
- The null value of store can be replaced with 0 or 1 as well but using -1 shows the least error. All three values show nearly the same error but with -1 it was minimum.
- We have performed left outer join between train and store and test and store to add store attributes to each store id.

Model Building

- We have used train_test_split() function for validation of our data. We have taken 80% data for training and 20% for testing.

Selection of Model

- We have tried different models like DecisionTreeRegressor, RandomForestRegressor, LinearRegressor, XGBRegressor etc. with their default parameters.
- RandomForestRegressor gave the minimal error with default parameters. So, we decided to go for parameter tuning with it.
- We also tried parameter tuning in XGBRegressor as xgboost is the most logical model for this data.
- On parameter tuning, XGBRegressor has much less error than RandomForestRegressor, so we did our final submission using XGBRegressor.
- RandomForest can perform better on small datasets whereas Xgboost generally works on large datasets and since our dataset was not too big not too small we thought of giving RandomForest a shot.
- Since Xgboost tries to add more trees that compliments the already built ones, this gives you better accuracy with less trees.


























Parameter tuning

- We have used GridSearchCV for parameter tuning.
- It was a challenging part, we several times ran into overfitting. Mostly in random forest model. So we chose to tune only the parameters which have maximum effect on the result.
- We trained max_depth, learning_rate, etc.
- We trained mostly one or two parameters at a time while using GridSearchCV. The order in which they are tuned affects the result a lot.
- We followed the procedure similar to the one described in the Analyticsvidhya [link](#). Note the gives the explanation about how to parameter tuning of xgboost for classification.
- It states first to tune max_depth, min_child_weight first than others as they matter the most.

PARAMETER	TUNE	RMSE
max_depth	14	0.0589303
learning_rate	0.1	0.0585522
min_child_weight	11	0.0613056
subsample	0.8	0.0608804

Kaggle Rank and Score Achieved:

We secured **1st rank** among 9 competing teams for “Rossman Sales Competition” with a Kaggle score of **0.05477**.

#	△pub	Team Name	Kernel	Team Members	Score ⓘ	Entries	Last
1	▲ 5	Dilbar		  	0.05477	23	8d
2	▲ 2	Trifecta		  	0.05487	28	10d
3	▼ 2	DP-TheSalesPredictor		 	0.05602	49	7d
4	▼ 2	The Brogrammers		  	0.06173	13	1mo
5	▼ 2	Here to compete		  	0.06187	24	7d
6	▲ 1	Bade Aaram Se		 	0.06356	22	7d
7	▲ 1	MichaelTrevorAndFranklin		  	0.07321	32	10d
8	▼ 3	Mavin		  	0.07911	20	7d
9	—	Code-Blooded		  	0.11128	16	9d