DATA71011

Report: Sales Forecasting

Words: 2300 (Without Titles, Figures and appendix)

11186818 Samit Srinivas Uttarkar

Contents

1. Introduction:

The Rossmann dataset is a historical dataset for over 1000 Rossmann stores located in different regions of Germany. The Rossmann dataset is a valuable resource for those interested in retail sales analysis. Before building forecasting models, it is necessary to understand the dataset first and pre-process it accordingly so that the predictions are accurate. The report aims to provide an overview of the data preprocessing and forecasting processes applied to the Rossmann dataset. This report will also outline the forecasting method' limits and emphasise the significance of taking measures to guarantee the accuracy and dependability of the outcomes.

1.1 The following will be the approach taken:

- Business Problem
- Data Collection and Preprocessing
- Exploratory Data Analysis
- Feature Selection and Outlier Detection
- Modelling
- Model Performance and Evaluation
- Conclusion

1.2 Understanding Datasets

The dataset offered contains precise historical sales information for 1,115 Rossmann locations. Predicting the "Sales" column for the test set is the key job. Since the coursework has already provided the dataset's specifications. The particulars of each parameter will not be covered in this report.
The sales column, which represents the daily turnover, should be the primary area of focus for this report (this is what we are predicting). Additionally, it has been noted that each business has a distinctive store ID. Most of the fields have clear instructions.
It's time to analyse the dataset now that we have a good grasp of the dataset itself.

2. Data Preprocessing:

Data wrangling and processing require cleaning the data and preparing it for further analysis. Our cleaning process involves the following steps:

2.1 Null Value Treatment:

The train and test dataset don't have any null values in particular so there is no need for null value treatment for these datasets

For the store dataset, there are five main columns that have null values. They are '**CompetitionOpenSinceMonth', 'CompetitionDistance', 'CompetitionOpenSinceYear', 'Promo2SinceWeek', 'Promo2SinceYear'** and **'PromoInterval'**

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

Fig 1: Null Values

The preceding figure shows that of these five, four have the majority of their values missing.

For imputation data for CompetitionDistance we should first check the **skewness** of the data

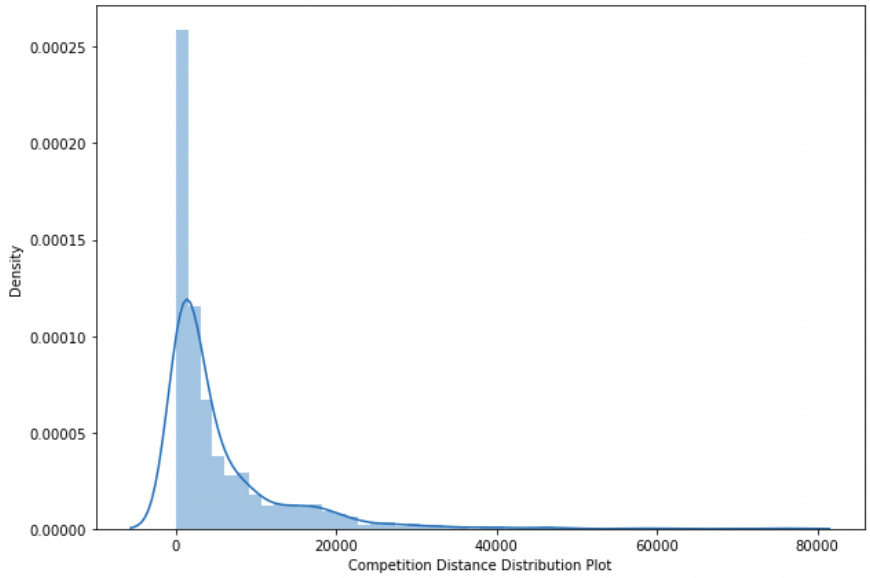

Fig 2: Skewness for imputation

It seems like the data for competition distance is right skewed. Therefore inputting median is the right choice

Next, we replace null values present in competition open since month and year with the most occurring values of the columns i.e modes of those columns and with zeros in **PromoOpen** and **PromoInterval** because if **Promo2** is zero it indicates that the rest of the Promo's are zero

2.2 Merging datasets:

We then combine the datasets train, test, and stores to produce the final datasets train_df and test_df. As we need to keep all the values from the train and test datasets, which will be required for machine learning, we use a left join to combine the datasets.

The resulting merged train data set is organised into 1017209 rows and 18 columns.

```
Int64Index: 1017209 entries, 0 to 1017208
Data columns (total 24 columns):
 #   Column                     Non-Null Count    Dtype
---  ------                     --------------    -----
 0   Store                      1017209 non-null  int64
 1   DayOfWeek                  1017209 non-null  int64
 2   Date                       1017209 non-null  datetime64[ns]
 3   Sales                      1017209 non-null  int64
 4   Customers                  1017209 non-null  int64
 5   Open                       1017209 non-null  int64
 6   Promo                      1017209 non-null  int64
 7   StateHoliday               1017209 non-null  object
 8   SchoolHoliday              1017209 non-null  int64
 9   StoreType                  1017209 non-null  object
 10  Assortment                 1017209 non-null  object
 11  CompetitionDistance        1017209 non-null  float64
 12  CompetitionOpenSinceMonth  1017209 non-null  float64
 13  CompetitionOpenSinceYear   1017209 non-null  float64
 14  Promo2                     1017209 non-null  int64
 15  Promo2SinceWeek            1017209 non-null  float64
 16  Promo2SinceYear            1017209 non-null  float64
 17  PromoInterval              1017209 non-null  object
 18  Year                       1017209 non-null  int64
 19  Month                      1017209 non-null  int64
 20  Day                        1017209 non-null  int64
 21  WeekOfYear                 1017209 non-null  int64
 22  CompetitionOpen            1017209 non-null  float64
 23  Promo2Open                 1017209 non-null  float64
dtypes: datetime64[ns](1), float64(7), int64(12), object(4)
memory usage: 194.0+ MB
```

Fig 3: The final merged dataset

2.3 Data Extraction:

We have extracted Date, Year, Month, Day, Week, Week Of year for the Date column for further analysis. We do this by extracting each attribute individually and then adding it as a new column to our original dataset.

2.4 Combining and Creating New Variables:

- We have created a new column called '**PromoOpen**' from '**Promo2SinceYear**' and '**Promo2SinceWeek**' to measure more accurate period in months from when the store is participating in Promo2
- We have created a new column called '**CompetitionOpen**' From '**CompetitionOpenSinceYear**' and 'CompetitionOpenSinceMonth' to measure more accurate period in months from when the nearest competition has opened

- The new replaced the negative values present in '**PromoOpen**' and '**CompetitionOpen**' with zero value, as there is no significance of them being negative.

2.5 Scaling of data:

For this data set, we have specifically used the standard scaling, or Z score method. In this method, each value is subtracted from the mean of that variable and then divided by the standard deviation.

$$z = (x - \mu)/\boldsymbol{\sigma}$$

We scale the data because standardising numerical input variables to a predefined range improves the performance of machine learning algorithms.
As our data is skewed but somewhat follows a normal distribution, it is therefore better to use StandardScaler as it is useful for the features that follow a normal distribution.

	Store	DayOfWeek	Date	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	-1.731640	5	2015-07-31	555	1	1.273237	0	2.144211
1	-1.728534	5	2015-07-31	625	1	1.273237	0	2.144211
2	-1.725427	5	2015-07-31	821	1	1.273237	0	2.144211
3	-1.722321	5	2015-07-31	1498	1	1.273237	0	2.144211
4	-1.719214	5	2015-07-31	559	1	1.273237	0	2.144211
...	...	...	...	...	...	...	...	...
1017204	1.716545	2	2013-01-01	0	0	-0.785400	a	2.144211
1017205	1.719651	2	2013-01-01	0	0	-0.785400	a	2.144211
1017206	1.722758	2	2013-01-01	0	0	-0.785400	a	2.144211
1017207	1.725864	2	2013-01-01	0	0	-0.785400	a	2.144211
1017208	1.728970	2	2013-01-01	0	0	-0.785400	a	2.144211

Fig 4: Data after scaling

This step comes under "**feature engineering**" but is fundamentally under "**data preprocessing.**" This step was done after exploratory data analysis.

2.6 Encoding Categorical variables:

Encoding categorical columns basically means converting categorical data into binary or integer format so that the dta with converted categorical values can provide a better prediction for our forecasting
Let’s first see how many categorical variables we have and what are they

```
Unique values for DayOfWeek: [5 4 3 2 1 7 6]
Unique values for Open: [1 0]
Unique values for Promo: [1 0]
Unique values for StateHoliday: ['0' 'a' 'b' 'c' 0]
Unique values for SchoolHoliday: [1 0]
Unique values for StoreType: ['c' 'a' 'd' 'b']
Unique values for Assortment: ['a' 'c' 'b']
Unique values for CompetitionOpenSinceMonth: [ 9. 11. 12.  4. 10.  8.  3.  6.  5.  1.  2.  7.]
Unique values for CompetitionOpenSinceYear: [2008. 2007. 2006. 2009. 2015. 2013. 2014. 2000. 2011. 2010. 2005. 1999.
 2003. 2012. 2004. 2002. 1961. 1995. 2001. 1990. 1994. 1900. 1998.]
Unique values for Promo2: [0 1]
Unique values for Promo2SinceWeek: [ 0. 13. 14.  1. 45. 40. 26. 22.  5.  6. 10. 31. 37.  9. 39. 27. 18. 35.
 23. 48. 36. 50. 44. 49. 28.]
Unique values for Promo2SinceYear: [   0. 2010. 2011. 2012. 2009. 2014. 2015. 2013.]
Unique values for PromoInterval: [0 'Jan,Apr,Jul,Oct' 'Feb,May,Aug,Nov' 'Mar,Jun,Sept,Dec']
```

Fig 5: Categorical variables

As we can see, we basically have four columns where we may further encode the categories. They are "**StateHoliday**", "**DayOfWeek**", "**StoreType**", and "**Assortment**".
For this step, we will be using **OneHotEncoder** from sklearn’s preprocessing library.
After encoding the encoded variable looks like this

DayOfWeek_1	DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	DayOfWeek_7	S
0.0	0.0	0.0	0.0	1.0	0.0	0.0	
0.0	0.0	0.0	0.0	1.0	0.0	0.0	
0.0	0.0	0.0	0.0	1.0	0.0	0.0	
0.0	0.0	0.0	0.0	1.0	0.0	0.0	
0.0	0.0	0.0	0.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	
0.0	1.0	0.0	0.0	0.0	0.0	0.0	
0.0	1.0	0.0	0.0	0.0	0.0	0.0	
0.0	1.0	0.0	0.0	0.0	0.0	0.0	
0.0	1.0	0.0	0.0	0.0	0.0	0.0	
0.0	1.0	0.0	0.0	0.0	0.0	0.0	

StateHoliday_0	StateHoliday_a	StateHoliday_b	StateHoliday_c
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
...	...	...	...
0.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0

Fig 6: Encoded variable

StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c
0.0	0.0	1.0	0.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...
1.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	1.0	0.0	0.0	1.0

Fig 7: Encoded variables

3. Exploratory Data Analysis

3.1 Univariate Analysis:

Let's check for outliers in sales and customers using box plots.

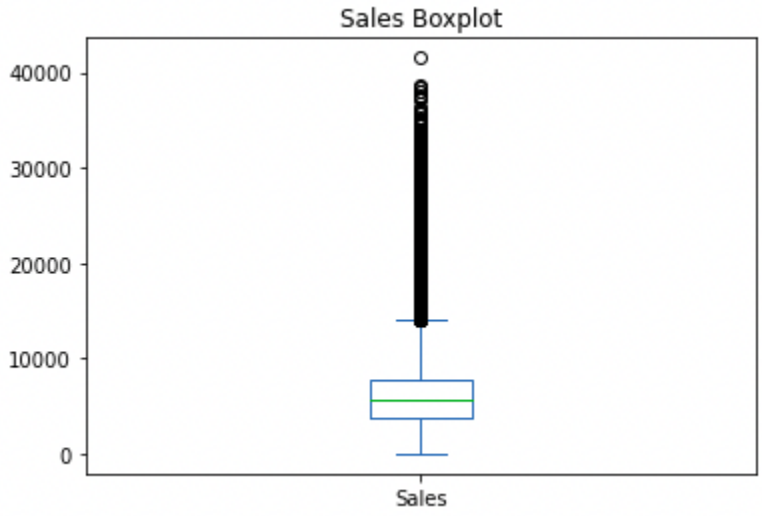

Fig 8: Boxplot for sales

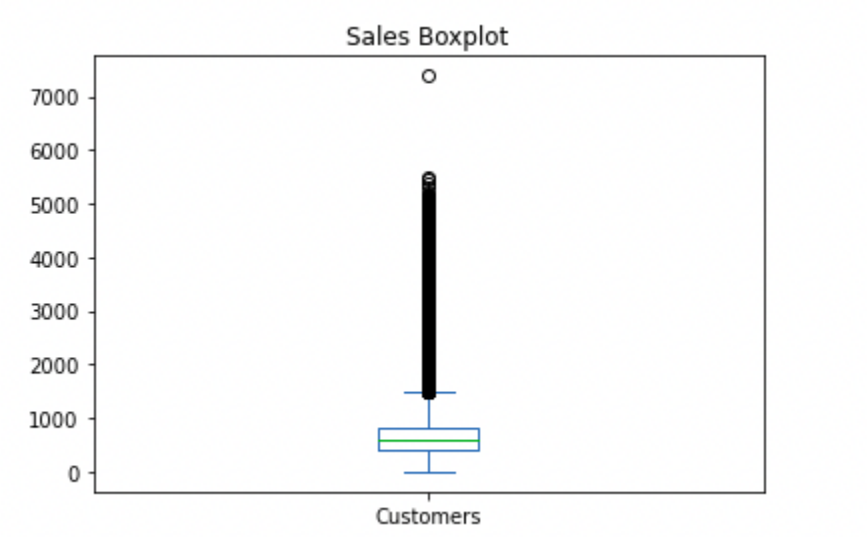

Fig 9: Boxplot for Customers

As we can see, the sales and customers columns include a lot of outliers.

Let's look at the dataframe to identify any outliers in our sales data and learn more about them.

Date	Store	DayOfWeek	Sales	Customers	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	Promo2	Year	Month	W
2013-01-07	817	1	10.381676	4065	1	0	0	a	a	140.0	0	2013	1	
2013-01-08	817	2	10.241744	3862	1	0	0	a	a	140.0	0	2013	1	
2013-01-21	817	1	10.330942	3900	1	0	0	a	a	140.0	0	2013	1	
2013-01-22	817	2	10.210605	7388	1	0	0	a	a	140.0	0	2013	1	
2013-02-03	262	7	10.272323	4144	0	0	0	b	a	1180.0	0	2013	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2015-07-13	1114	1	10.245516	3592	1	0	0	a	c	870.0	0	2015	7	
2015-07-19	262	7	10.362967	4691	0	0	0	b	a	1180.0	0	2015	7	
2015-07-26	262	7	10.390440	4783	0	0	0	b	a	1180.0	0	2015	7	
2015-07-27	842	1	10.235701	1493	1	0	0	d	c	1200.0	0	2015	7	
2015-07-31	1114	5	10.222232	3745	1	0	1	a	c	870.0	0	2015	7	

429 rows × 17 columns

Fig 10: Outlier dataframe

Observation:

Some interesting insights can be drawn from these outliers dataframe:

- First thing that comes to notice is the DayOfWeek for Store 262. It's sunday and it has high sales and it's the store type B.
- It is undeniable that the outliers for the stores with promotion = 1 and store type B exhibit this trend. The reasons for their behaviour seem reasonable, thus treating them would not be a good decision.

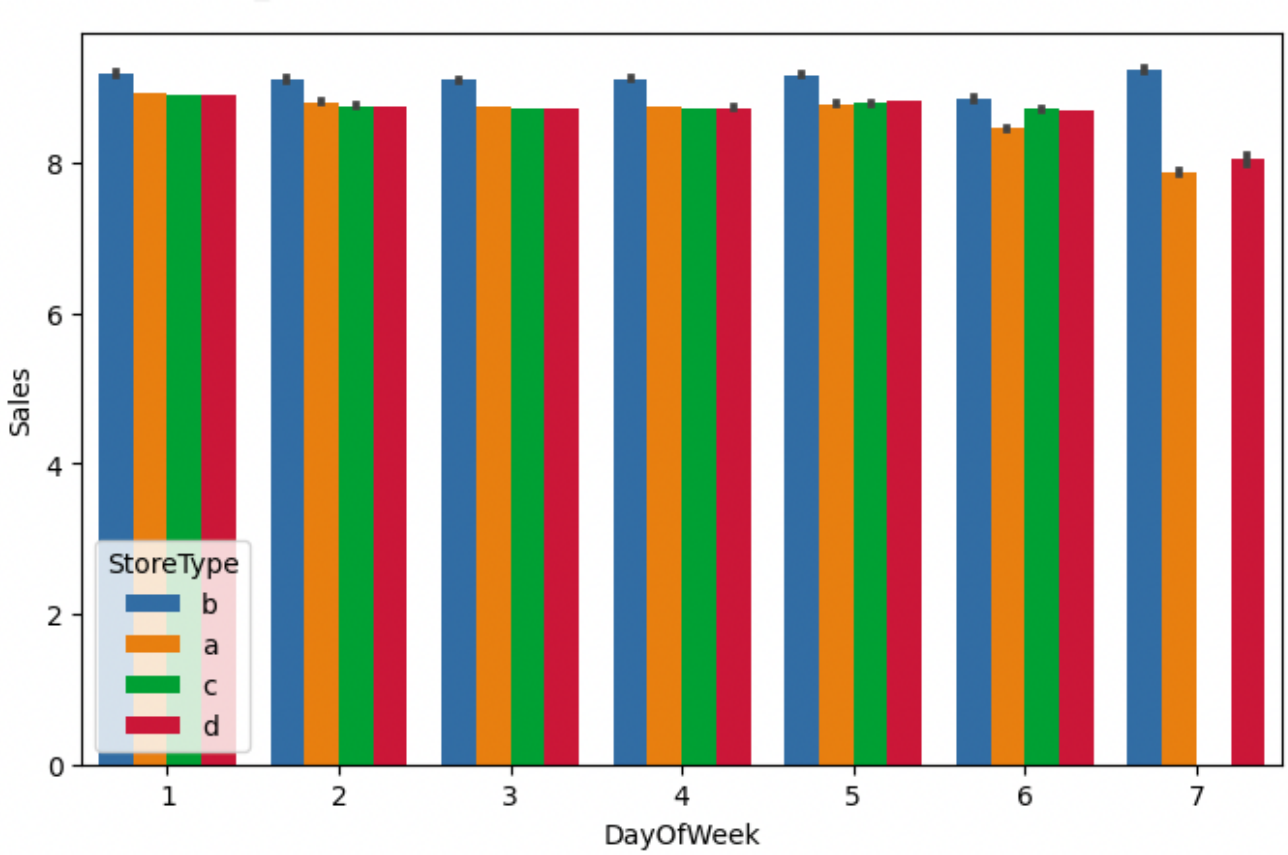

Fig 11: Sales for each outlier store in outliers

3.1.1 Outlier Treatment

When we have established the ups and downs of the target variable, which is '**Sales**' in relation to the other features, it would be prudent not to delete or otherwise alter the outliers if they are a real occurrence.

Next let's check the distribution for 'sales' using the distplot method of seaborn

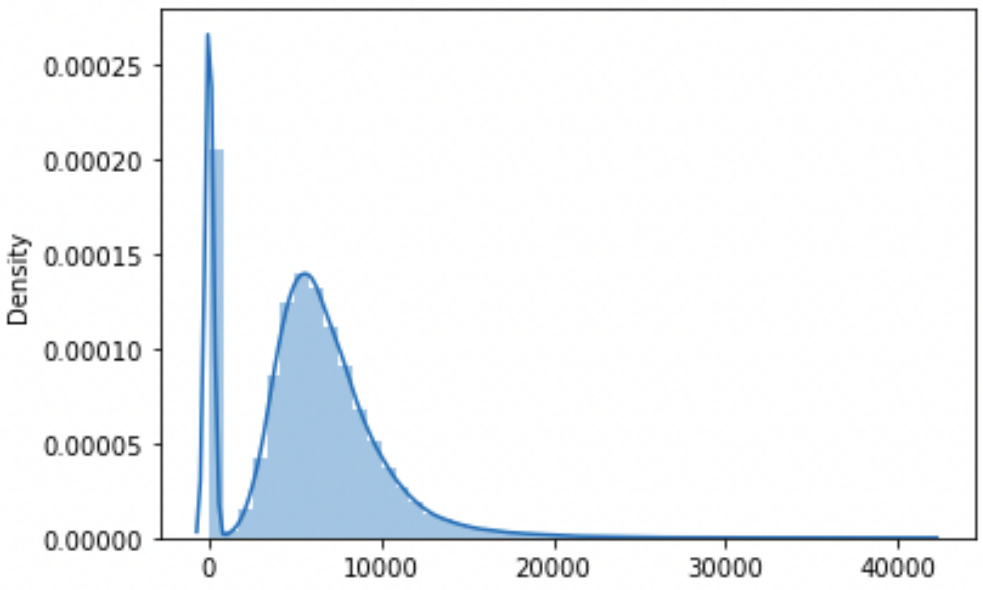

Fig 12: Distribution plot for Sales

Observation:

The drop in sales reflects the 0 sales attributable to the stores that were temporarily closed for refurbishment.

3.2 Multivariate Analysis:

Let's check how well each store in doing in terms of total sales

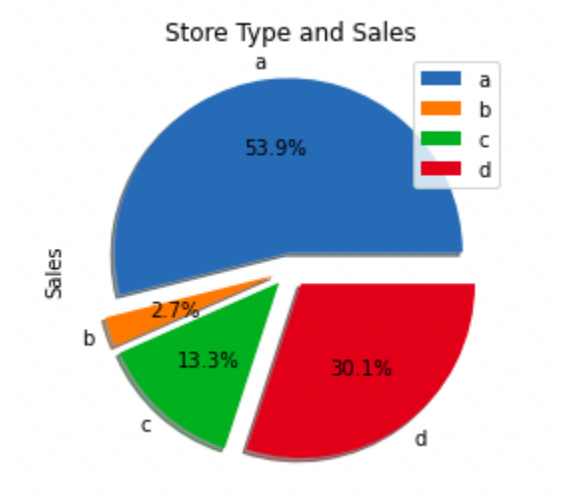

Fig 13: Pie chart for total sales per store

Observation:

Store type 'a' has the most amount with over 53% of sales followed by store type 'b' with 30%.

Now let's check for total number of customer for each store type

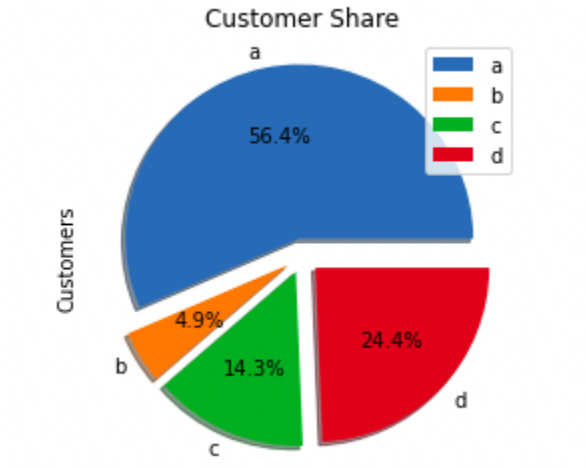

Fig 14: Share of customer per store type

The distribution is similar to what we saw in the sales pie chart.

Now that we know how the distribution looks for total sales, let's check the average sales for each type of store.

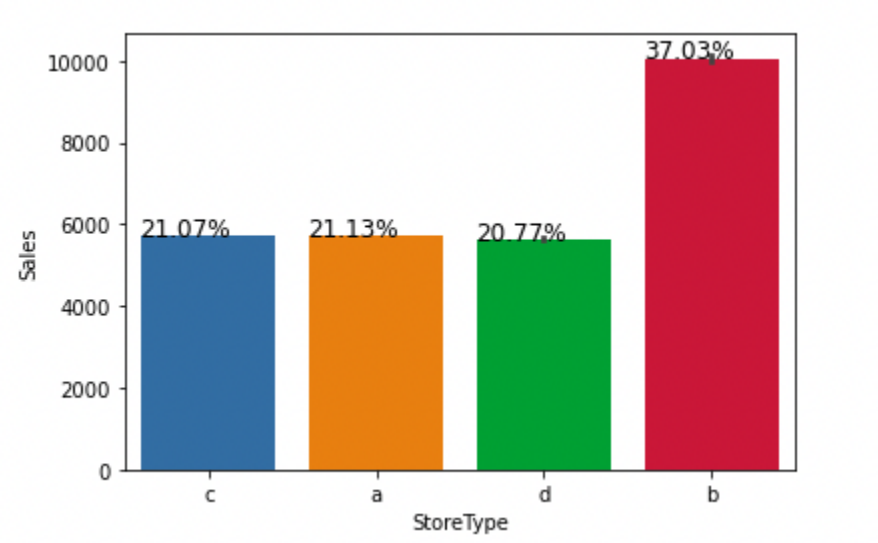

Fig 15: Average sales per store type

This is interesting as we saw earlier that store 'a' had the most sales in total but from the above bar plot we can observe that store 'b' has higher average number of sales

We can visualise it more clearly using a line plot

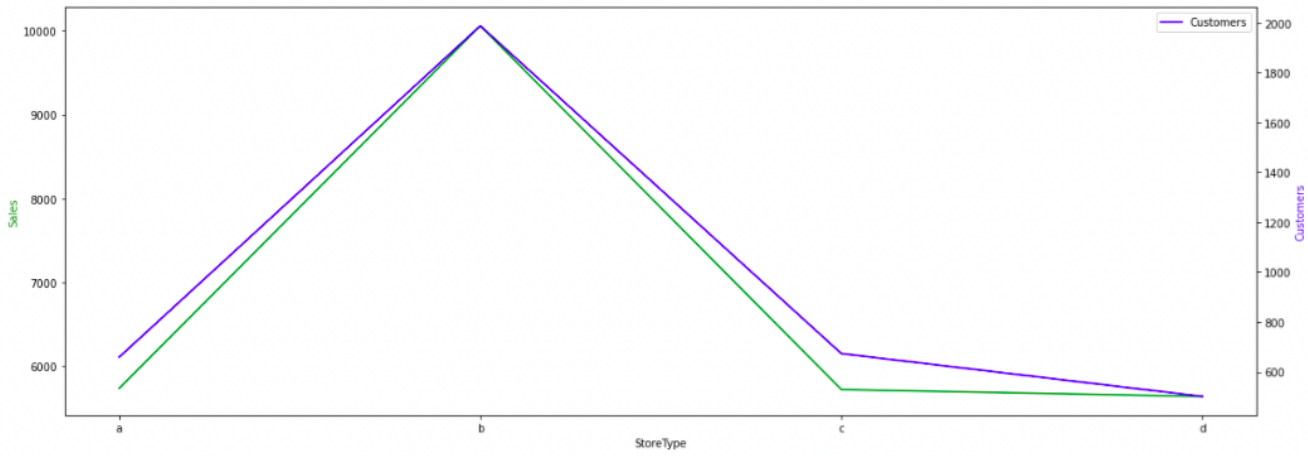

Fig 16: Line plot for average sales for each store

Let's look into this further and find out why this is taking place. To do this, we'll check the connection between store type, assortment levels, and sales.

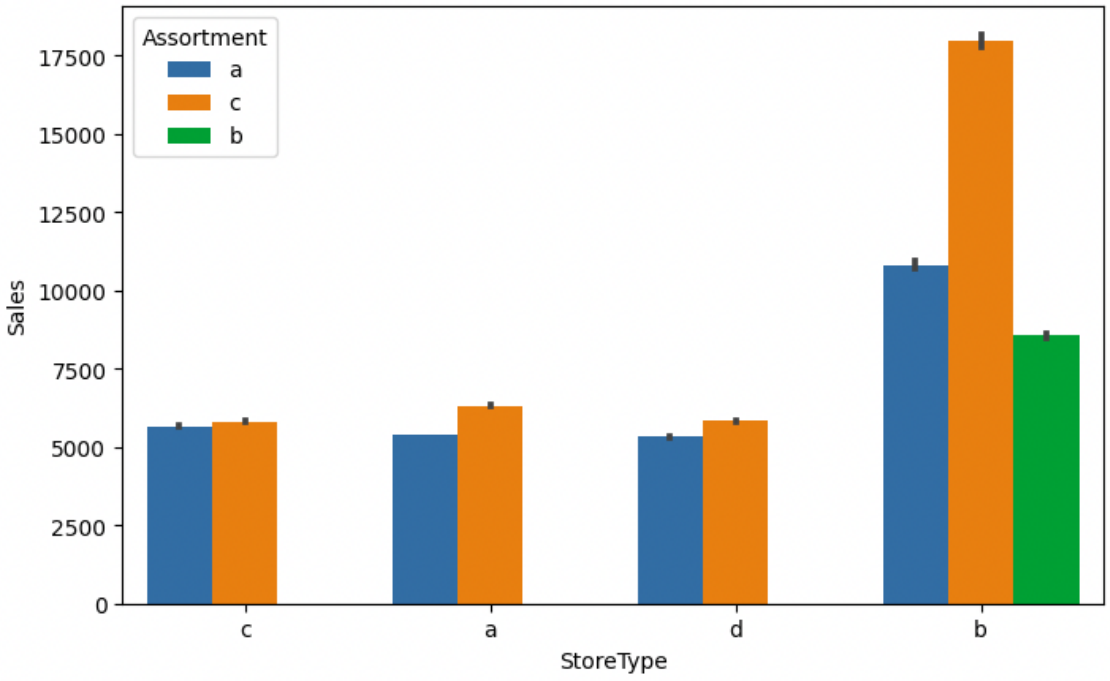

Fig 16: Average for each assortment level for each store

Observation:

- The above bar plot shows that the store types a, c and d have only assortment level a and c. On the other hand the store type b has all the three kinds of assortment strategies, a reason why average sales were high for store type b stores.
- Interesting insight to note is that store type b with highest average sales and per store revenue generation looks healthy and a reason for that would be all three kinds of assortment strategies involved .

Now let's check how Promo affects sales.

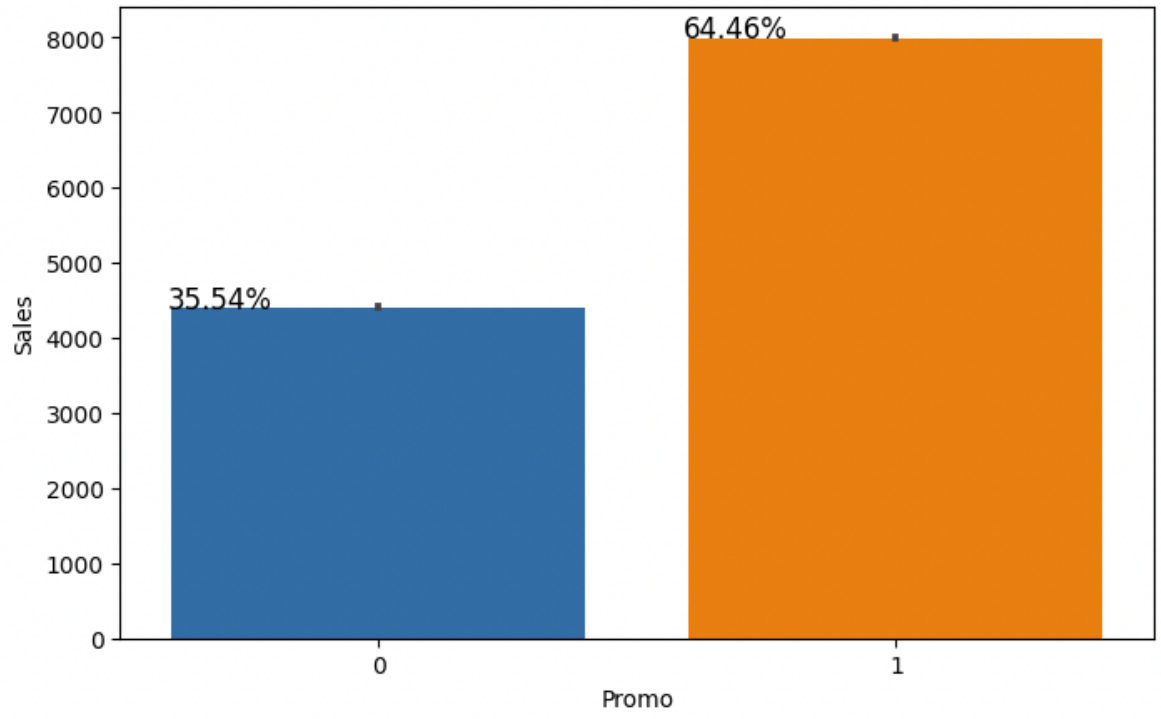

Fig 17: Bar plot for promo vs sales

Observation:

It can be concluded that promo leads to more sales

Now Let's check for average sales per assortment type

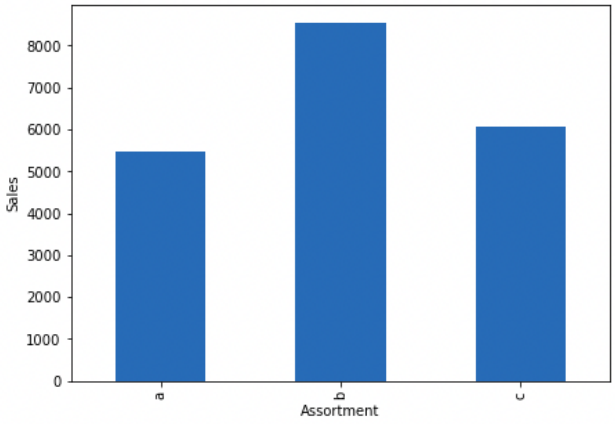

Fig 18: Average sales for each assortment type

Observation:

Assortment type ‘b’ which is extra has the highest sales on average

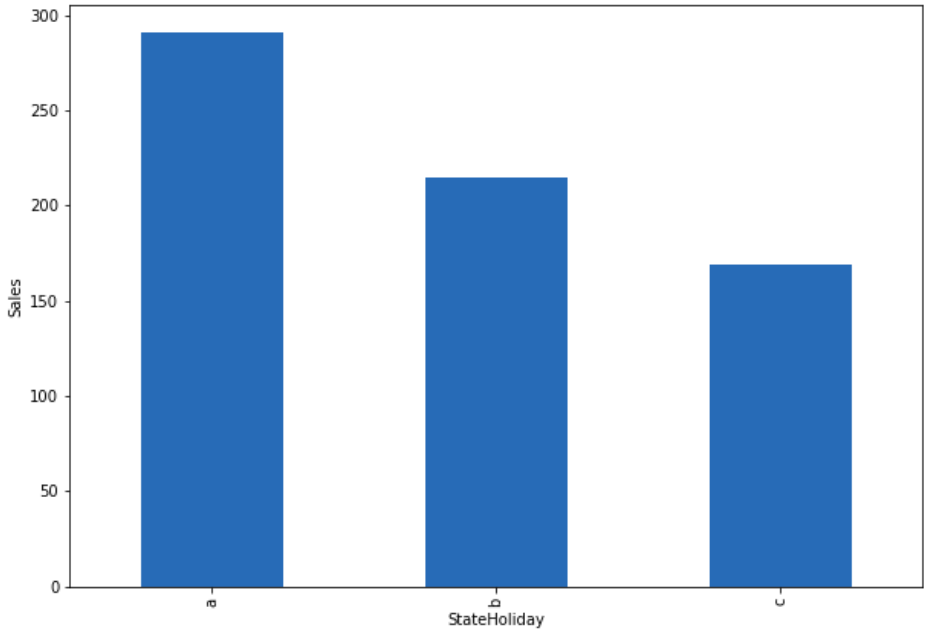

Fig 19: Average sales on State Holidays

Observation:

Stores have made more sales during Public holidays compared to Easter and Christmas holidays.

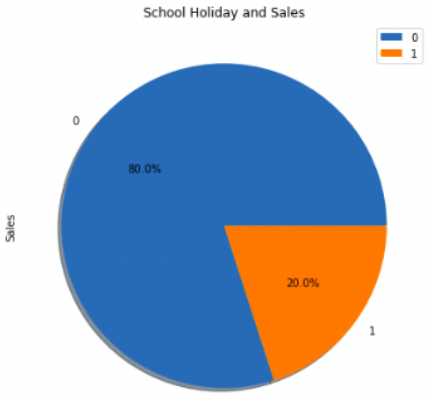

Fig 20: Pie Chart for effect of school holiday on sales

Observation:

20% of the total sales gets affected by the schools holidays which also means that around 20% of the sales are oriented from the school students

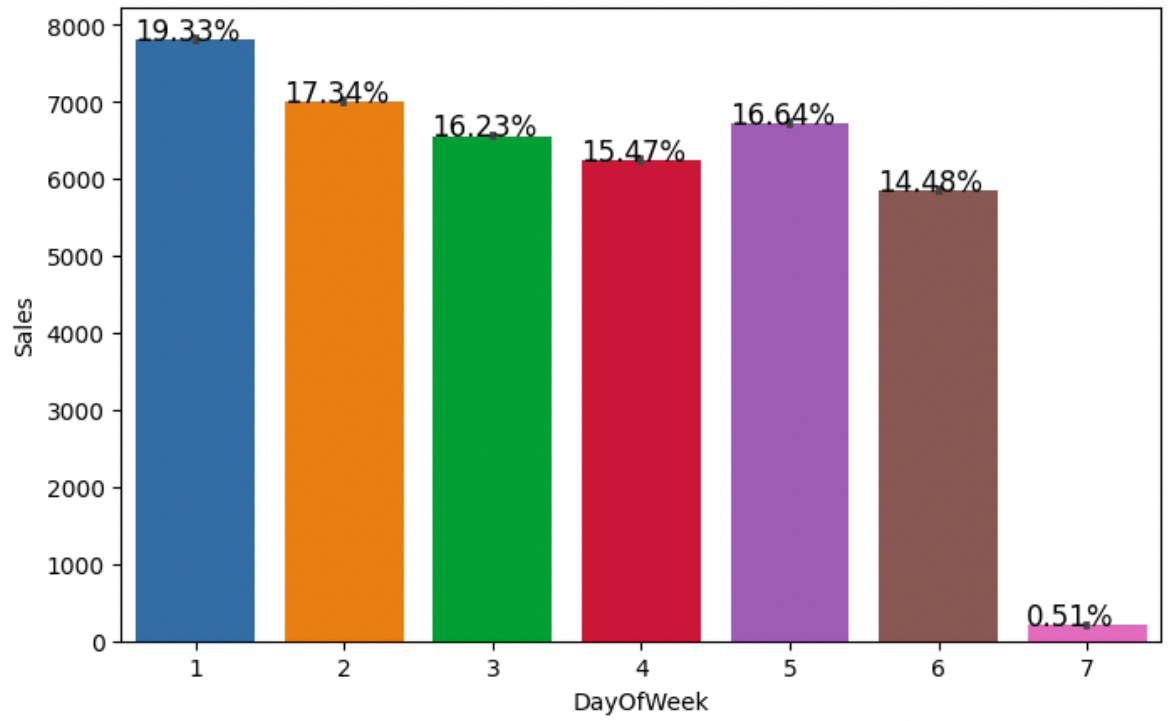

Fig 21: Average Sales per week

Observation:

There were more sales on Monday and the lowest on Sunday. This might be because shops generally remain closed on Sundays.

3.2.1 Continuous Features Analysis:

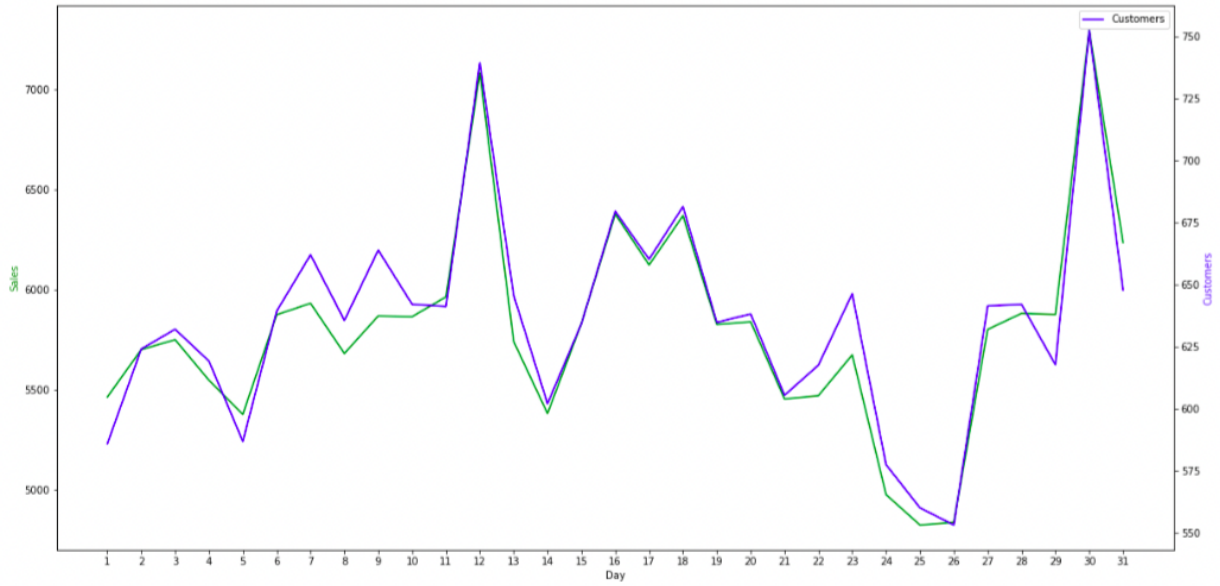

Fig 22: Sales over day of the month

Observation:

Every month, the 30th and 12th see the biggest sales and consumer traffic, while the 25th and 26th see the lowest traffic and sales.

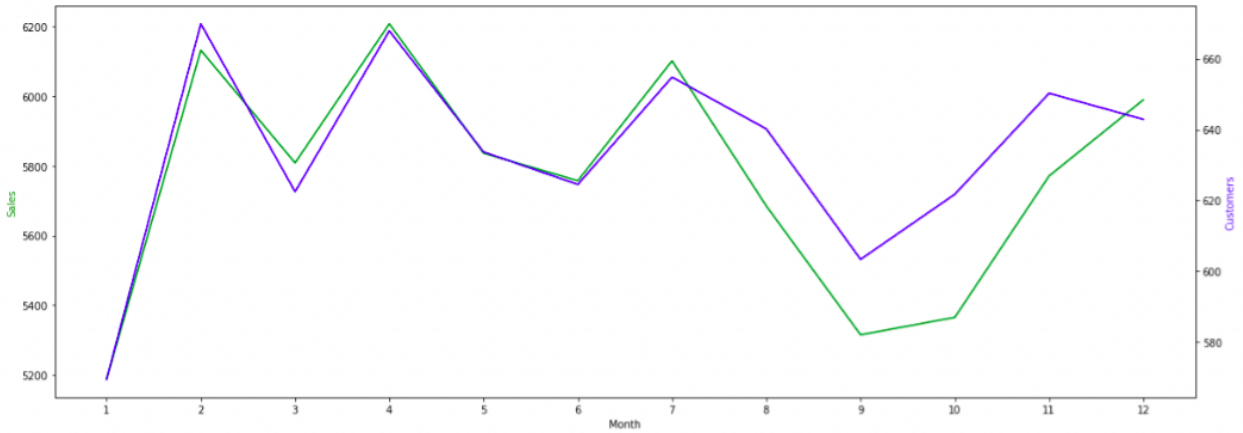

Fig 23: Sales over month of the year

Observation:

Due to festivals, sales are at their highest in February and March. January and September, or the off-seasons, are the months with the lowest sales. The sales begin to rise in December, maybe as a result of Christmas.

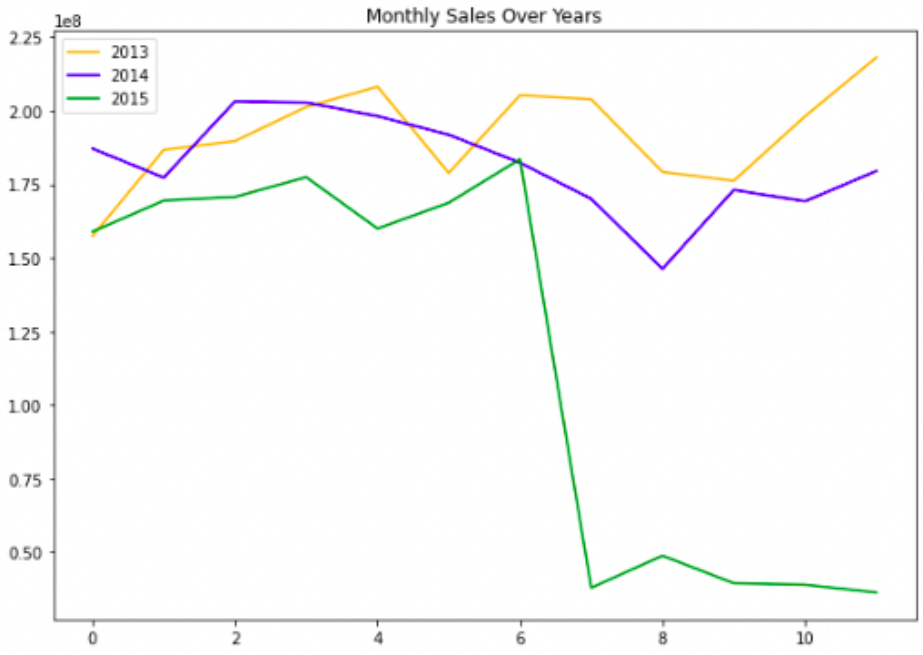

Fig 24: Monthly sales over the years

Observation:

By the end of the year, just before the holidays, sales increase. Sales decreased from July to September in 2014, indicating that some stores had to close for refurbishment. Sales in 2015 decreased dramatically from June, which may have been caused by shuttered stores.

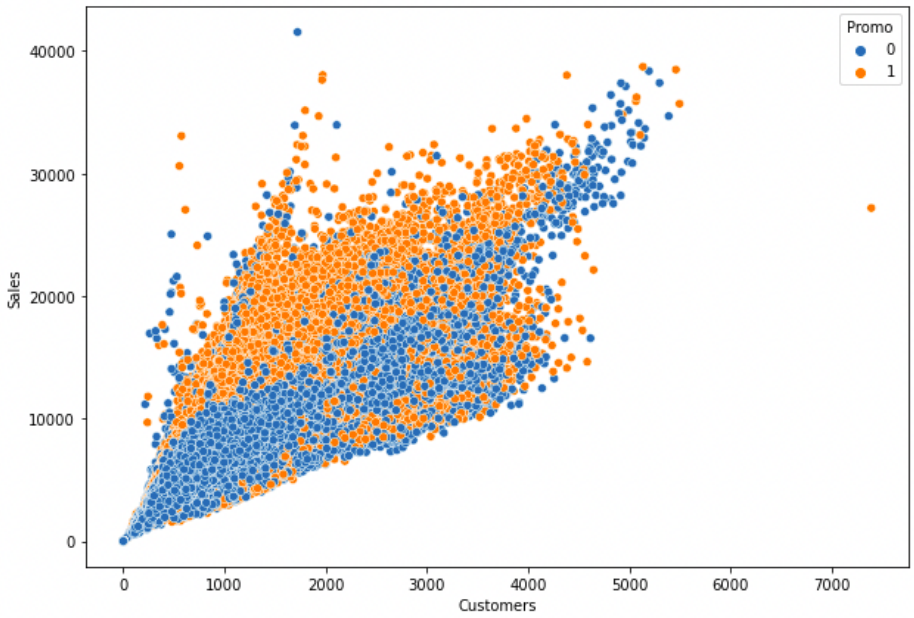

Fig 25: Scatter Plot for Sales vs Customers

Observation:

There is a linear relationship between customers and sales and it is also noticeable that whatever the promo was open, stores had higher sales and customers compared to the similar period when promo was closed, which means promo had a good impact on the business

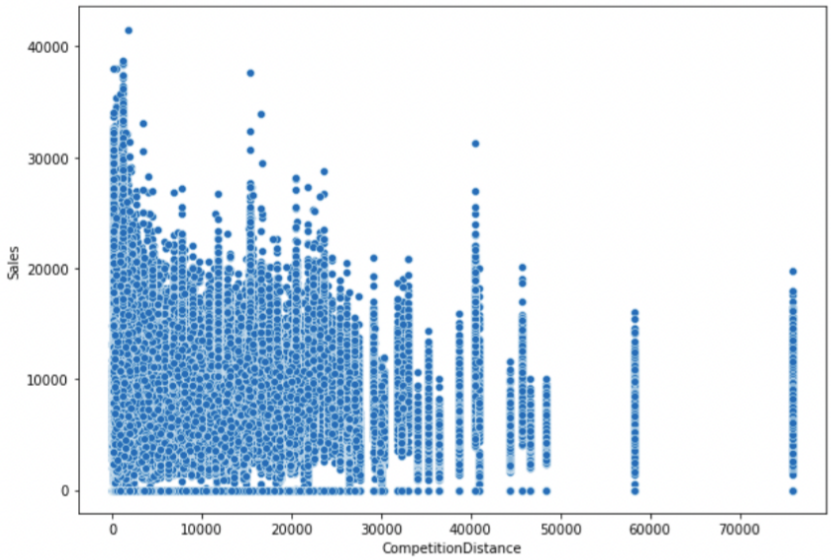

Fig 26: Scatter plot for competitor distance vs sales

Observation:

Surprisingly, sales increased when competitors were closer even though most stores were not that far from one another and were packed close together.

3.2.2 Correlation Matrix

By checking the correlation, the factors affecting sales can be figured out.

As we only need meaningful numeric columns here, we will drop the unnecessary to get a clear picture

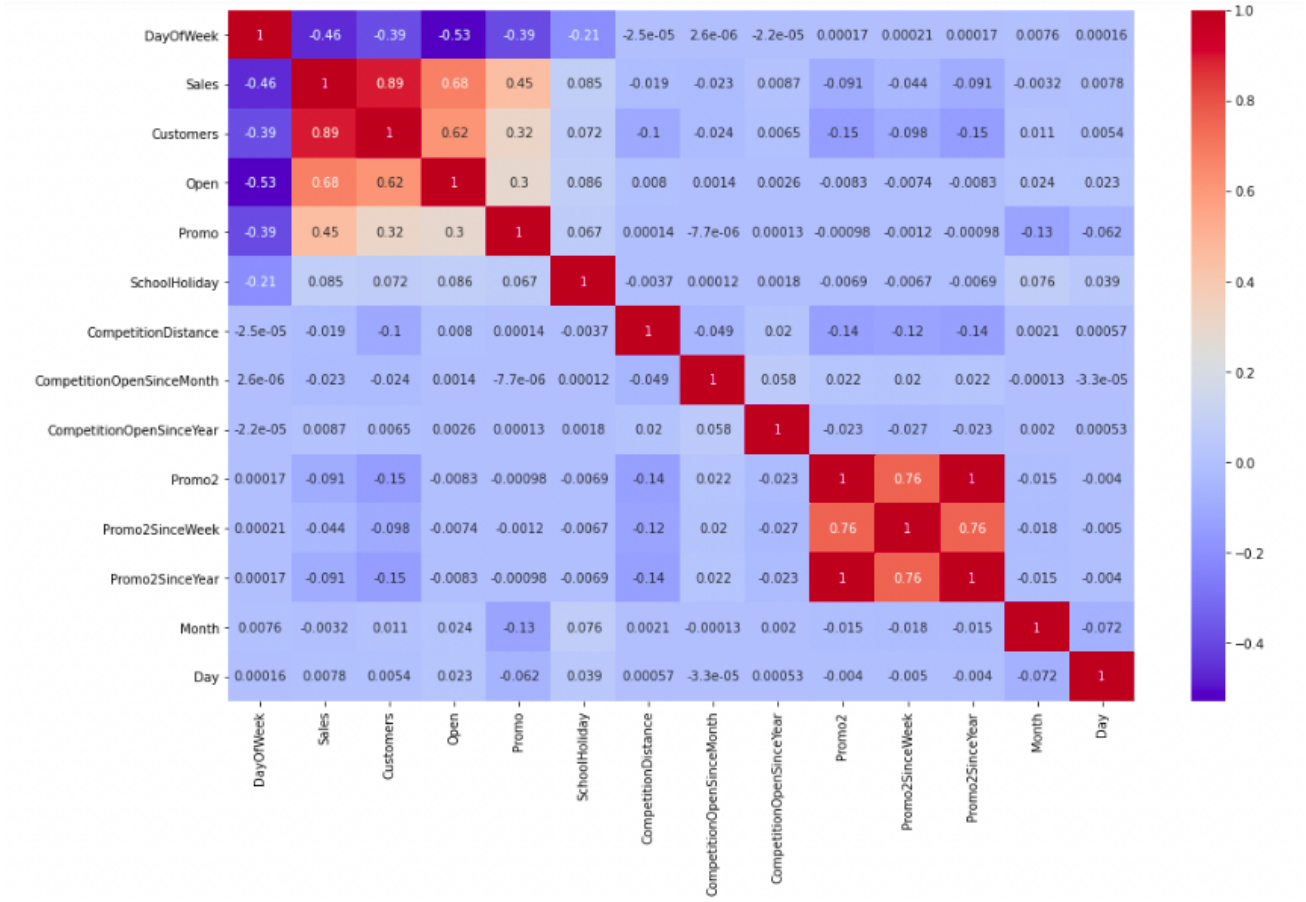

Fig 27: Correlation Matrix

Observation:

- Day of the week has a negative correlation indicating low sales as the weekends, and promo, customers and open has positive correlation.
- State Holiday has a negative correlation suggesting that stores are mostly closed on state holidays indicating low sales.
- CompetitionDistance showing negative correlation suggests that as the distance increases sales reduce, which was also observed through the scatterplot earlier.
- There's multicollinearity involved in the dataset as well. The features telling the same story like Promo2, Promo2 since week and year are showing multicollinearity.
- The correlation matrix is agreeing with all the observations done earlier while exploring through barplots and scatterplots.

4. Modelling

4.1 Selection of Model

In our modelling, we will use the XGBoost regression as our model for predicting sales. Because XGBoost uses parallel processing for quick performance, performs well on medium-sized datasets, and guards against overfitting, it is recommended to adopt this regression model. It is well known that there is seasonality at play, making it impossible to fit a linear relationship. Tree-based machine learning techniques that are resistant to the effects of outliers are utilised for these types of datasets. Extreme Gradient Boosting, or XGBoost, is a classification and regression approach based on the gradient boosting ensemble algorithm.

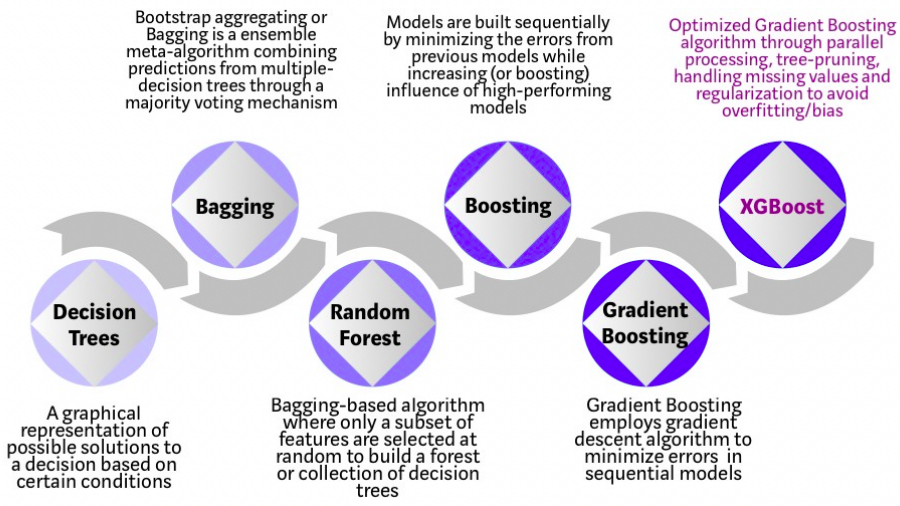

Fig 28: XGBoost compared to other tree based models

4.2 Dividing dataset into input and target

Before we begin modelling, it is crucial to scale our data and appropriately encode categorical values, both of which we have done in the preceding steps.

Our final train dataset will then be divided into input columns and target columns, with input columns being columns we want to train on and target columns being the "Sales" column.

```
input_cols = ['Store', 'DayOfWeek', 'Date', 'Customers', 'Open', 'Promo',
       'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment',
       'CompetitionDistance', 'CompetitionOpenSinceMonth',
       'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek',
       'Promo2SinceYear', 'PromoInterval', 'Year', 'Month', 'Day',
       'WeekOfYear', 'CompetitionOpen', 'Promo2Open']

target_col = 'Sales'
```

Fig 29: Separating input and target columns

4.3 Model fitting:

We will create our train dataset for validation, which will include numeric columns and the encoded columns we have created using OneHotEncoder in the preceding steps, and we will fit this along with our target dataset, which is the "Sales," in order to validate our model using only the train dataset.

```
%%time
xgb_model.fit(X, targets)
```

```
CPU times: user 35.5 s, sys: 2.52 s, total: 38 s
Wall time: 5.45 s
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=4, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=20, n_jobs=-1, num_parallel_tree=None, predictor=None,
```

Fig 30: Model fitting

We can assess our predicting precision using the Root Mean Square Error (RMSE).

The RMSE that we obtained is pretty high at **2248.719**. But we must keep in mind that no cross validation or hyper parameter adjustment was done to acquire this RMSE.

We can visualise individual trees using plot_tree method

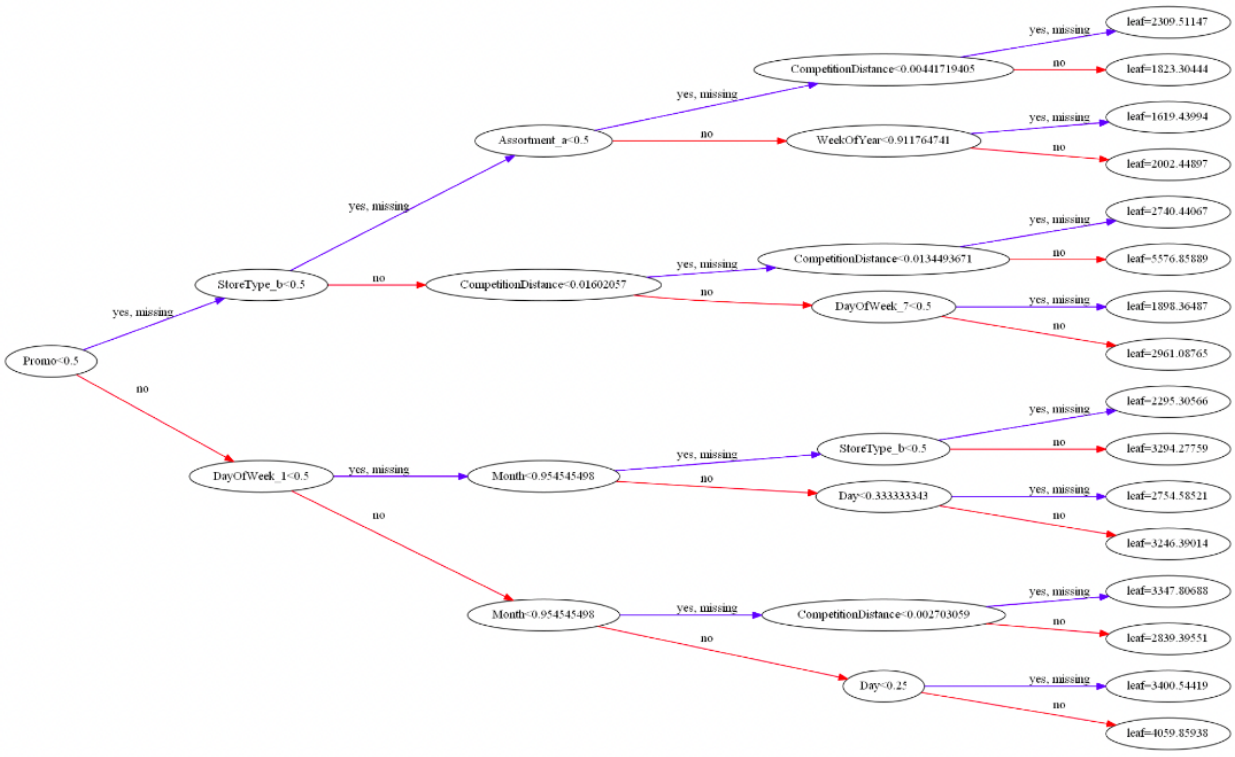

Fig 31: Tree visualisation for our XGBoost tree model

4.4 Feature Importance

Let's take a look at which features were most commonly used in sales prediction.

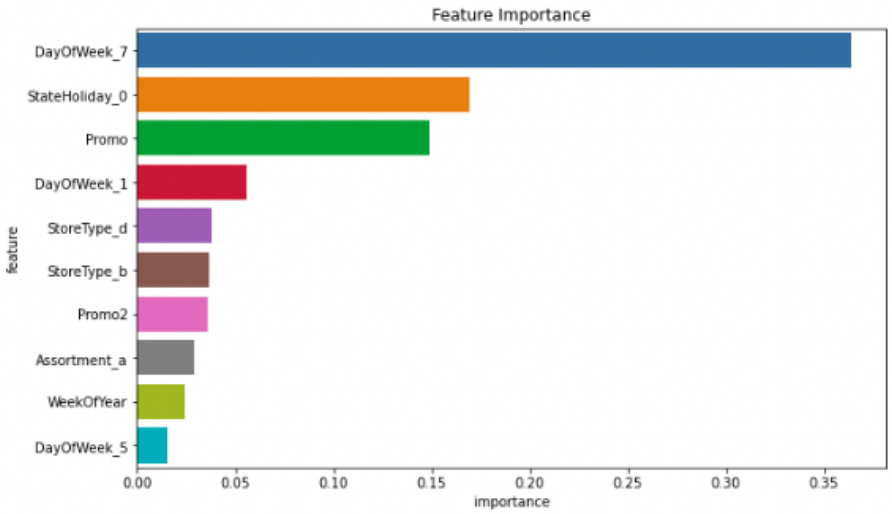

Fig 32: Feature Importance

From our observations **DayOfWeek_7** was the most important features along with **StateHoliday_0**

After Hyper Parameter tuning these were the values for parameters

```
(n_jobs=-1, random_state=42, n_estimators=1000,
 learning_rate=0.2, max_depth=10, subsample=0.9,
 colsample_bytree=0.7)
```

Fig 33: Hyper parameters

These parameters resulted in a very low RMSE of **411.319** on the validation set

Now that we have predicted the sales let's visualise what our predicted sales look like

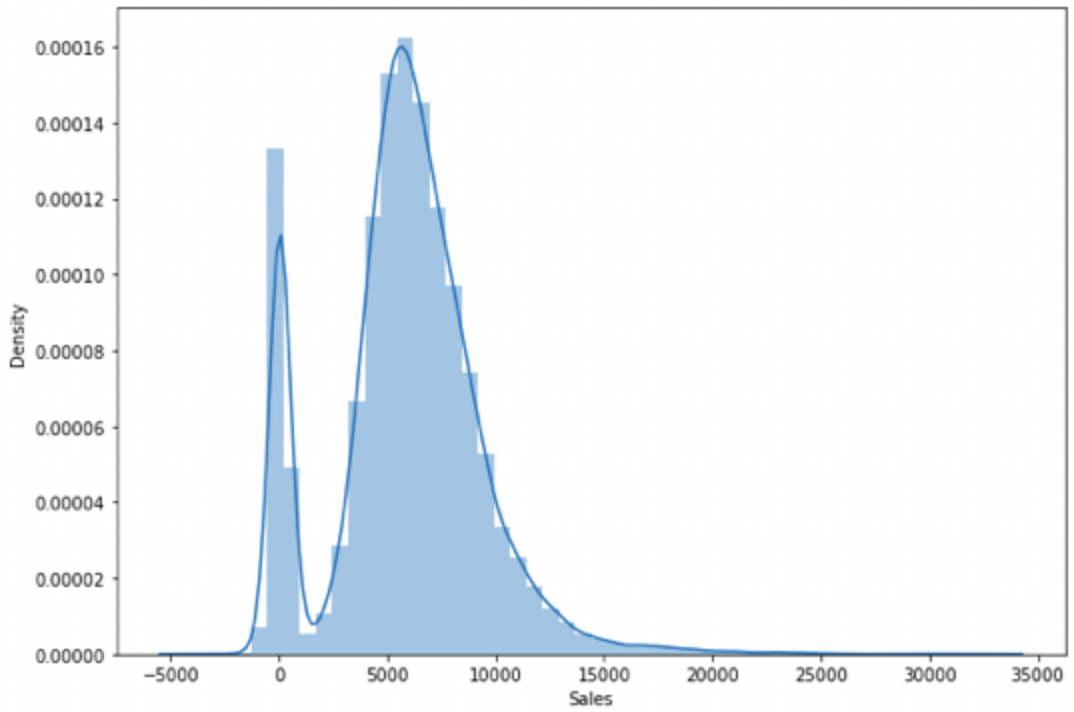

Fig 34: Predicted sales distribution plot

The predicted sales plot looks fairly similar to our train dataset, which means our prediction closely resembles the actual dataset which is good.

5. Conclusion:

5.1 EDA Conclusions:

- There's a positive correlation between customers and sales which is explanatory as more customers means more sales .
- Here it can be deduced that there were more sales on Monday, probably because shops generally remain closed on Sundays which had the lowest sales in a week. This validates the hypothesis about this feature.
- When comparing Sales and Day of the month it was observed that sales were particularly higher on 30th and 12th of every month.
- The positive effect of promotion on Customers and Sales is observable.
- It is clear that most of the stores remain closed during State and School Holidays.

- Based on the above findings it seems that there are quite a lot of opportunities in store type 'b' & 'd' as they had more number of customers per store and more sales per customer, respectively. Store type a & c are quite similar in terms of "per customer and per store" sales numbers and just because the majority of the stores were of these kinds, they had the best overall revenue numbers. On the other hand, store type b were very few in number and even then they had better average sales than others.
- Earlier, it was observed that only store type b had all three kinds of assortment levels and rest of the store types had two of them. It seems that in some b type stores the products were different as compared to others because the revenue per store is significantly more than the others.
- When comparing the sales for the 3 StateHoliday, it was observed that StateHoliday ‘a’ which is basic one had the most average sales as compared to other StateHoliday
- Sales were lower at the beginning of the month than they were at the end, leading one to believe that individuals used to shop for the following month towards the end of the previous one. These goods can mostly be considered to be a daily necessity.
- Most stores have competition distance within the range of 0 to 10 kms and had more sales than stores far away.

5.2 Model Conclusion:

Due to the structure of our data, it was determined that XGBoost was the optimal model for our dataset. We discovered that the RMSE prior to parameter adjustment was **2248.719**. This was a significantly high value. This showed that the forecasts were not reliable enough to use. Additionally, we discovered that DayoftheWeek_7 was the key component in our model. The distribution map closely reflects our training dataset, and after parameter adjustment, we obtained an RMSE of **411.319**, which is better than our previous findings.

5.3 Limitations of the dataset

- Temporal Data Cutoff: Only cover time period from January 2013 to July 2015
- Geographical Limitations: Only covers germany
- Limited information about customers: Not enough personal information to understand customer behaviour
- Incomplete data
- Lack of External Data
- Bias in the Data: The data may be biassed due to factors such as store locations and store size