DATA71011 | Individual Report

Coursework Report – ROSSMANN Sales Analysis

STUDENT ID: 11358614

WORD COUNT (excluding heading, sub-headings, text boxes, references): 1503.

Table of Contents

1. INTRODUCTION

Estimating customer demands and predicting future sales are crucial aspects of for enhancing a business financial performance. Sales forecasting involves the process of estimating the demand for products and services in stores over a defined period, allowing businesses to make informed decisions about their inventory, marketing strategies, and overall business planning.

. . . .

2. OBJECTIVE

The Rossmann dataset encompasses the historical records of more than 1000 Rossmann stores situated across diverse regions in Germany, making it asset for retail analysis. Before delving into the constructing forecasting models, a comprehensive understanding of dataset is essential, along with the need for appropriate pre-processing to ensure accurate predictions. This report primarily focuses on the initial steps of data pre-processing for a sales forecasting problem. Various factors, including competition, promotions, holidays, seasonality, and location influence sales at different stores. The current objective is to predict daily sales for the upcoming six weeks.

. . . .

3. VALIDATION METRIC: Root Mean Square Percentage Error (RMSPE)

Formula for the metric:

$$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \qquad where,\ e = \frac{y\_true - y\_pred}{y\_true}$$
$$n = total\ number\ of\ points.$$

. . . .

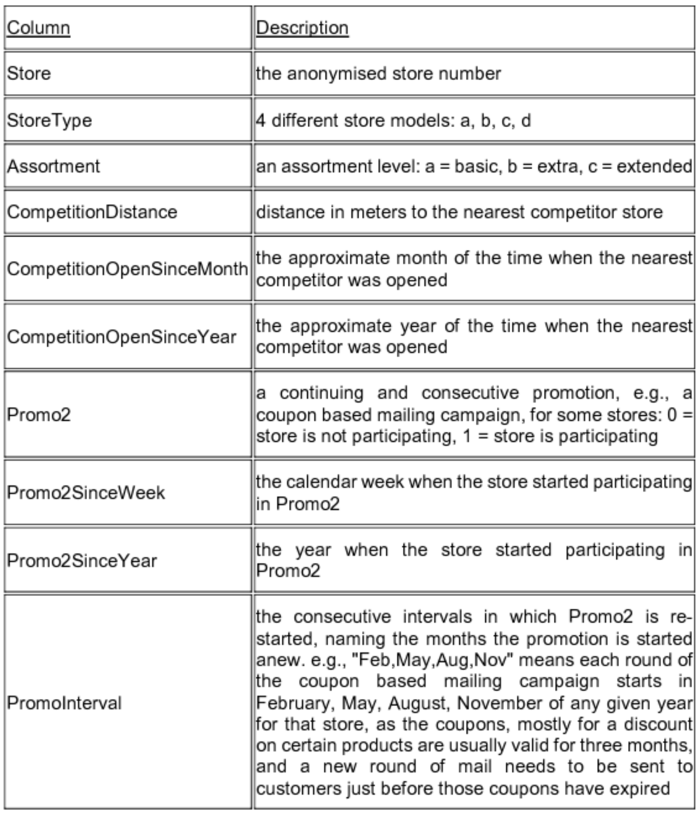
4. EXPLORING THE DATASETS

The provided dataset includes detailed historical sales data for 1115 Rossmann locations, and the primary goal is to predict the values of ‘Sales’ column.

4.1 Data

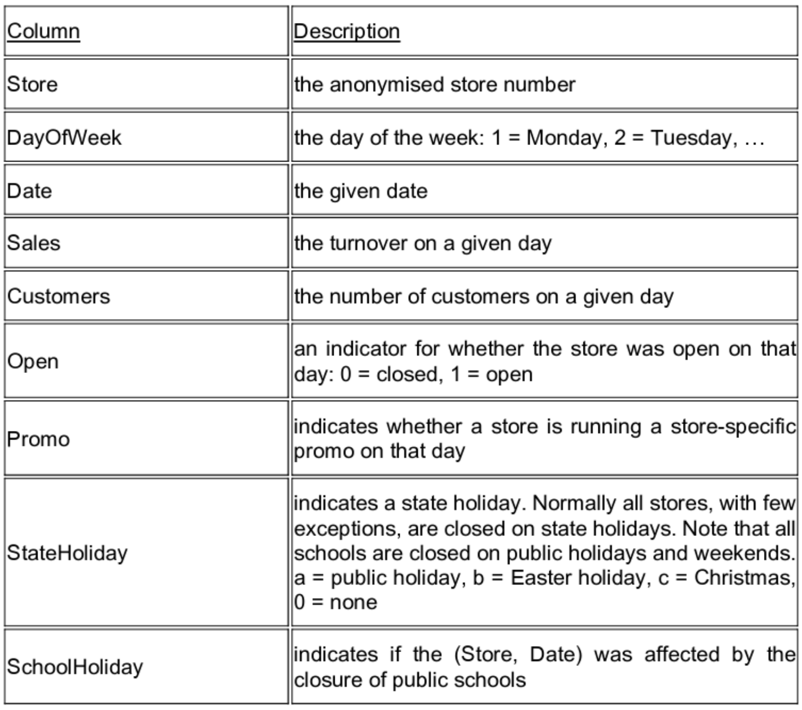
The files provided are:

a. **stores.csv** - This excel file contains the information for the 1,115 drug stores.



Column	Description
Store	the anonymised store number
StoreType	4 different store models: a, b, c, d
Assortment	an assortment level: a = basic, b = extra, c = extended
CompetitionDistance	distance in meters to the nearest competitor store
CompetitionOpenSinceMonth	the approximate month of the time when the nearest competitor was opened
CompetitionOpenSinceYear	the approximate year of the time when the nearest competitor was opened
Promo2	a continuing and consecutive promotion, e.g., a coupon based mailing campaign, for some stores: 0 = store is not participating, 1 = store is participating
Promo2SinceWeek	the calendar week when the store started participating in Promo2
Promo2SinceYear	the year when the store started participating in Promo2
PromoInterval	the consecutive intervals in which Promo2 is re-started, naming the months the promotion is started anew. e.g., "Feb,May,Aug,Nov" means each round of the coupon based mailing campaign starts in February, May, August, November of any given year for that store, as the coupons, mostly for a discount on certain products are usually valid for three months, and a new round of mail needs to be sent to customers just before those coupons have expired

b. **train.csv** - This csv file contains the historical sales data, which covers sales from 01/01/2013 to 31/07/2015.



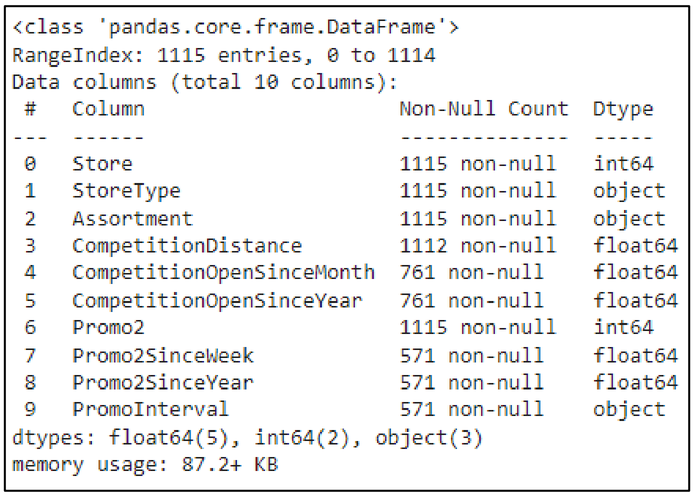
Column	Description
Store	the anonymised store number
DayOfWeek	the day of the week: 1 = Monday, 2 = Tuesday, …
Date	the given date
Sales	the turnover on a given day
Customers	the number of customers on a given day
Open	an indicator for whether the store was open on that day: 0 = closed, 1 = open
Promo	indicates whether a store is running a store-specific promo on that day
StateHoliday	indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = none
SchoolHoliday	indicates if the (Store, Date) was affected by the closure of public schools

c. **test.csv** - This csv contains the historical sales data, spanning from from 01/01/2013 to 31/07/2015.

4.2 Data Description

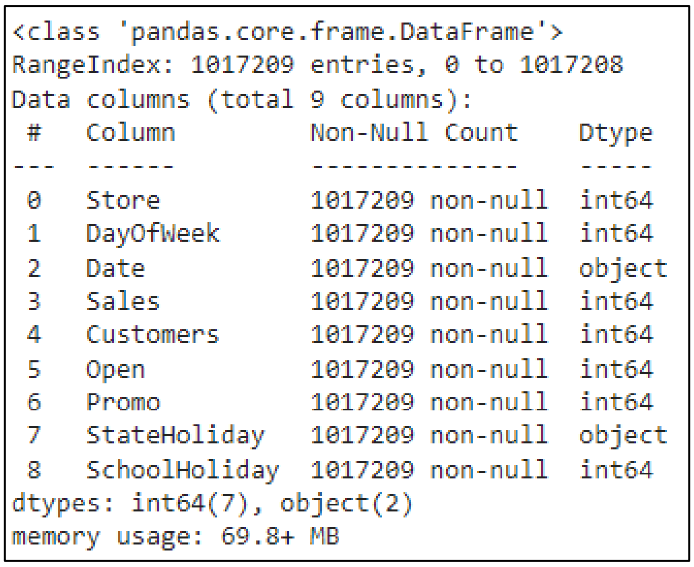
"store.csv" – Contains 1115 observations representing different stores, including features such as promotions, competition, and store type.
"train.csv" – Consists of 1017209 observations providing daily sales for various stores, spanning from 2013 to 2015.
"test.csv" – Comprises 1115 observations for stores, with the objective of predicting the sales column.

The store column is common in both "train.csv" and "store.csv" and can be joined.



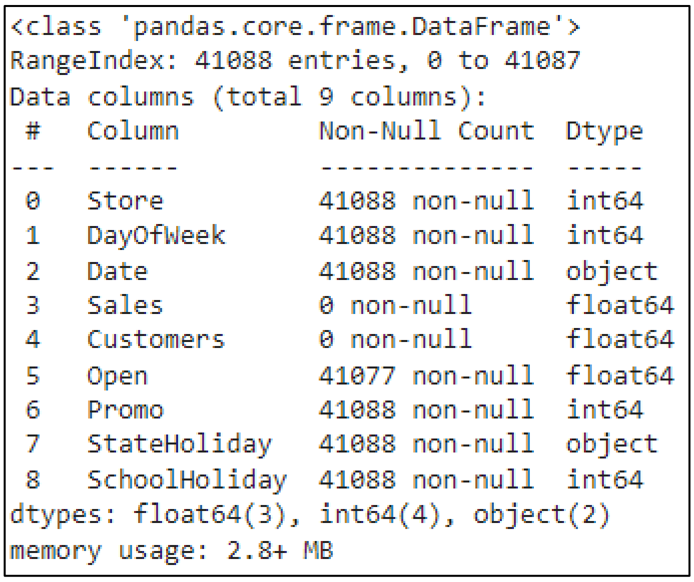
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Store                      1115 non-null   int64
 1   StoreType             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

Figure 1: store.csv columns datatypes and non-null count. Total we have 1115 observations and few columns have missing values which is handled in the later stages.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   Store          1017209 non-null  int64
 1   DayOfWeek      1017209 non-null  int64
 2   Date           1017209 non-null  object
 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
dtypes: int64(7), object(2)
memory usage: 69.8+ MB
```

Figure 2: train.csv columns and data types with 1017209 non-null values.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41088 entries, 0 to 41087
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Store          41088 non-null  int64  
 1   DayOfWeek      41088 non-null  int64  
 2   Date           41088 non-null  object 
 3   Sales          0 non-null      float64
 4   Customers      0 non-null      float64
 5   Open           41077 non-null  float64
 6   Promo          41088 non-null  int64  
 7   StateHoliday   41088 non-null  object 
 8   SchoolHoliday  41088 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 2.8+ MB
```

Figure 3: test.csv which has no data for Sales and Customers columns.

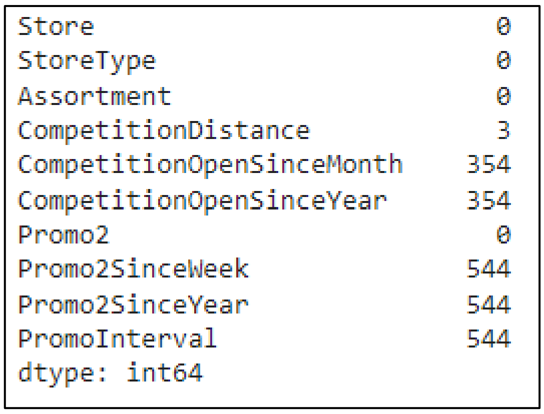
5. DATA PRE-PROCESSING

Data pre-processing involves pre-processing or manipulating raw data from one or more sources into a structured and clean dataset for analysis. The steps involved in our pre-processing are:

a. Collecting and Selecting data to use.
b. Integrating data sources together.
c. Conducting exploratory data analysis.
d. Cleaning and repairing data.
e. Data reduction, Feature selection and extraction.

5.1 Cleaning and repairing data.

Train and Test dataset don't have any null values.
For the store dataset, columns with null values are: '**CompetitionOpenSinceMonth**', '**CompetitionOpenSinceYear**', '**CompetitionDistance**', '**Promo2SinceWeek**', '**Promo2SinceYear**' and '**PromoInterval**'. Method used to replace null values is Mean/Median/Mode Imputation.



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

Figure 4: Null values

a. CompetitionDistance

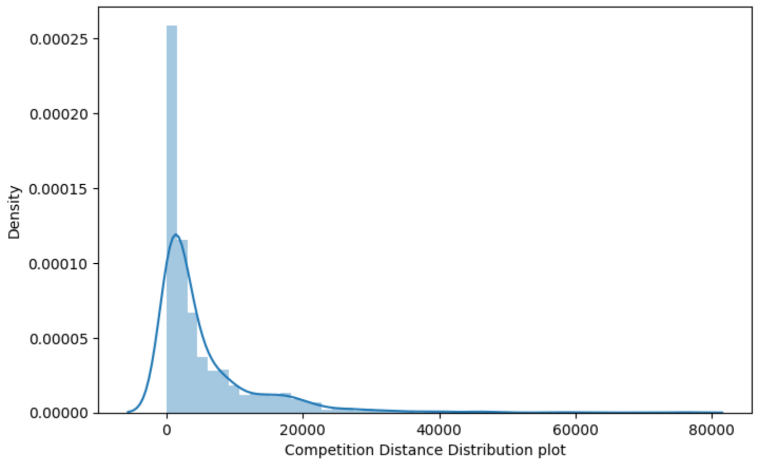


Figure 5: Skewness for CompetitionDistance

We first checked the skewness of the data. Since the data is right skewed, applying **median imputation** seems to be the right choice.

b. CompetitionOpenSinceMonth

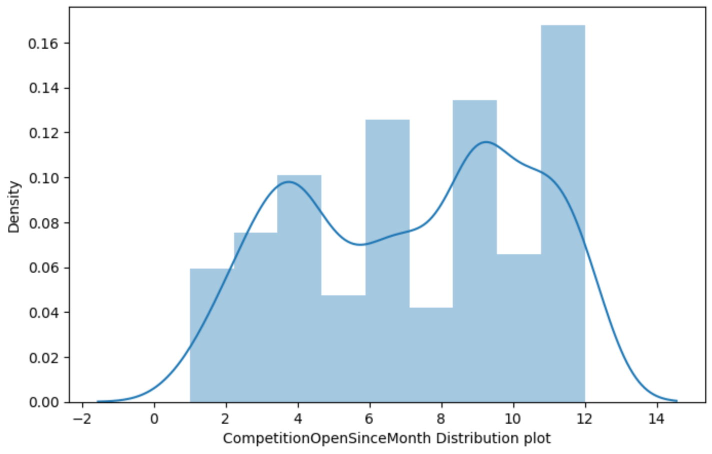


Figure 6: Skewness for CompetitonOpenSinceMonth which is Fairly symmetrical.

c. CompetitionOpenSinceYear

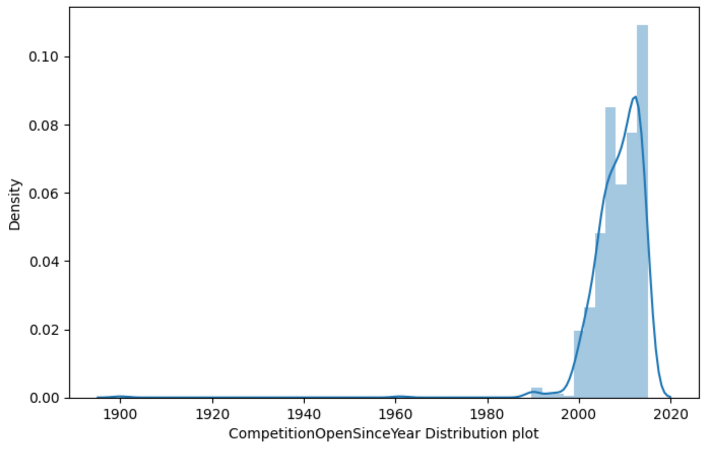


Figure 7: Skewness for CompetitonOpenSinceYear which is left-skewed.

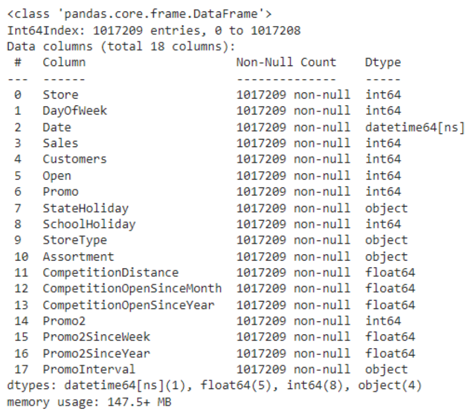
Next, from the figure (6) and (7) we replace null values present in the columns with most occurring values of columns i.e. modes.

d. Promo2SinceWeek, Promo2SinceYear and PromoInterval

The mentioned three columns are associated with Promo2, and the absence of these values corresponds to Promo2=0, indicating no promotion at stores. Imputing the missing values with zeros is suitable in this context, as a Promo2 value of zero implies that the remaining promotion-related values are also zero.

5.2 Datasets Linkage

We then combine the datasets train, test with stores to produce the final datasets **train_merged** and **test_merged** using left join as we need all columns from the train and test dataset for modelling.



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1017209 entries, 0 to 1017208
Data columns (total 18 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
dtypes: datetime64[ns](1), float64(5), int64(8), object(4)
memory usage: 147.5+ MB
```

Figure 8: Merged dataset with 18 columns and 1017209 rows.

5.3 Feature Extraction

We have successfully extracted insights from the date variable, breaking it down into distinct components to enhance our analysis. We have parsed the date information into separate features, including day, month, and year. Additionally, we have extracted day of the week (Day_Name) and the corresponding week of the year (WeekOfYear) and then adding them as a new column to our dataset.

5.4 Feature Creation

a. **PromoOpen:** Created a new feature from 'Promo2SinceYear' and 'Promo2SinceWeek'.
b. **CompetitionOpen:** Created a new feature from 'CompetitionOpenSinceYear' and 'CompetitionOpenSinceMonth'.

The main objective was to convert these features from 'year' unit to 'month' unit. Negative values are replaced with zeroes in new features.

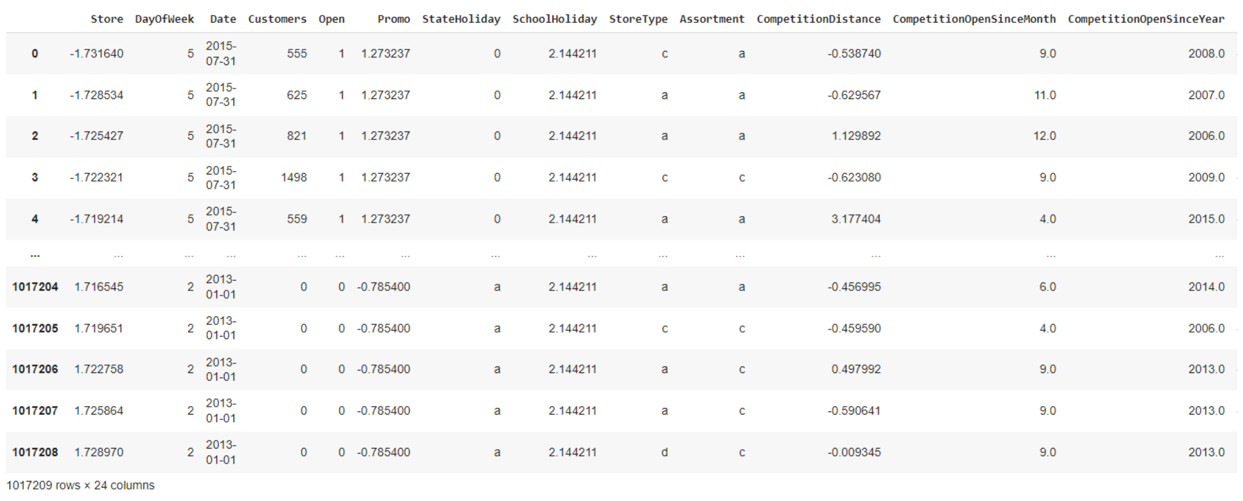
5.5 Scaling of Data

In our dataset, we employed the standard scaling method, also knows as z-score method.
Formula is represented by:

$$z = (x - \mu)/\sigma$$

The purpose of scaling the data is to standardise numerical input variables.

Since our dataset exhibits skewness but somewhat follows normal distribution, opting StandardScaler is preferable.



	Store	DayOfWeek	Date	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear
0	-1.731640	5	2015-07-31	555	1	1.273237	0	2.144211	c	a	-0.538740	9.0	2008.0
1	-1.728534	5	2015-07-31	625	1	1.273237	0	2.144211	a	a	-0.629567	11.0	2007.0
2	-1.725427	5	2015-07-31	821	1	1.273237	0	2.144211	a	a	1.129892	12.0	2006.0
3	-1.722321	5	2015-07-31	1498	1	1.273237	0	2.144211	c	c	-0.623080	9.0	2009.0
4	-1.719214	5	2015-07-31	559	1	1.273237	0	2.144211	a	a	3.177404	4.0	2015.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1017204	1.716545	2	2013-01-01	0	0	-0.785400	a	2.144211	a	a	-0.456995	6.0	2014.0
1017205	1.719651	2	2013-01-01	0	0	-0.785400	a	2.144211	c	c	-0.459590	4.0	2006.0
1017206	1.722758	2	2013-01-01	0	0	-0.785400	a	2.144211	a	c	0.497992	9.0	2013.0
1017207	1.725864	2	2013-01-01	0	0	-0.785400	a	2.144211	a	c	-0.590641	9.0	2013.0
1017208	1.728970	2	2013-01-01	0	0	-0.785400	a	2.144211	d	c	-0.009345	9.0	2013.0

1017209 rows × 24 columns

Figure 9: Scaled Dataset

This step falls under 'feature engineering but is essentially a part of 'data preprocessing'.
This step was done after exploratory data analysis.

5.6 Encoding Categorical Features

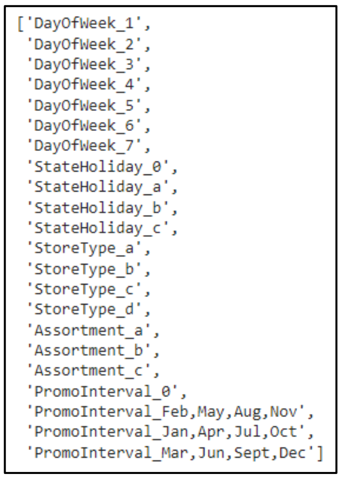
Encoding columns means converting categorical data into binary or integer format to provide a better prediction for our forecasting.

From the analysis, we have four columns where we may further encode the categories. They are:

a. State Holiday
b. DayOfWeek
c. StoreType
d. Assortment
e. PromoInterval

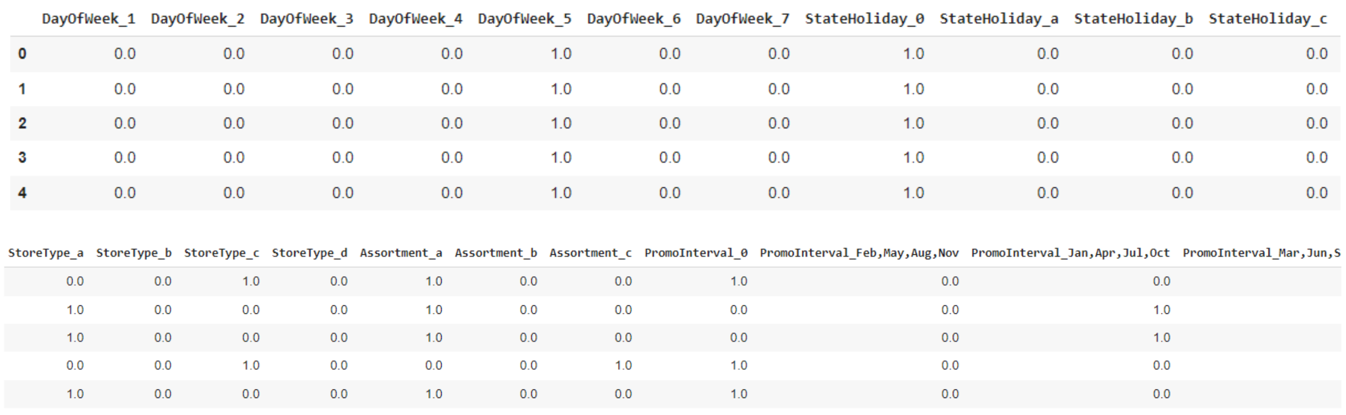
For this step, we'll be using OneHotEncoder from sklearn's preprocessing library.

After encoding our dataset looks like this given below.



```
['DayOfWeek_1',
 'DayOfWeek_2',
 'DayOfWeek_3',
 'DayOfWeek_4',
 'DayOfWeek_5',
 'DayOfWeek_6',
 'DayOfWeek_7',
 'StateHoliday_0',
 'StateHoliday_a',
 'StateHoliday_b',
 'StateHoliday_c',
 'StoreType_a',
 'StoreType_b',
 'StoreType_c',
 'StoreType_d',
 'Assortment_a',
 'Assortment_b',
 'Assortment_c',
 'PromoInterval_0',
 'PromoInterval_Feb,May,Aug,Nov',
 'PromoInterval_Jan,Apr,Jul,Oct',
 'PromoInterval_Mar,Jun,Sept,Dec']
```

Figure 10: Encoded columns after using OneHotEncoder.



	DayOfWeek_1	DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	DayOfWeek_7	StateHoliday_0	StateHoliday_a	StateHoliday_b	StateHoliday_c
0	0.0	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	1.0	0.0	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0

StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c	PromoInterval_0	PromoInterval_Feb,May,Aug,Nov	PromoInterval_Jan,Apr,Jul,Oct	PromoInterval_Mar,Jun,S
0.0	0.0	1.0	0.0	1.0	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	
1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
0.0	0.0	1.0	0.0	0.0	0.0	1.0	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	

Figure 11: Encoded columns with values.

6. EXPLORATORY DATA ANALYSIS

6.1 Univariate Analysis

Outliers in sales and customers using box plots.

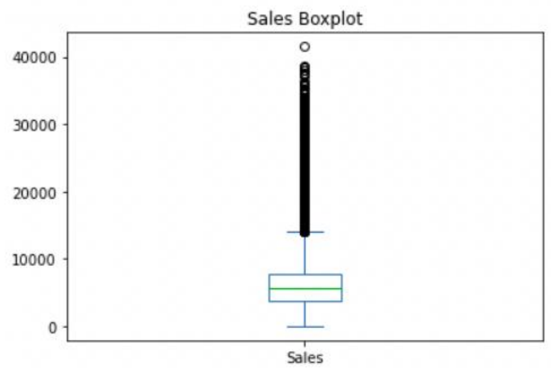


Figure 12: Boxplot for Sales.

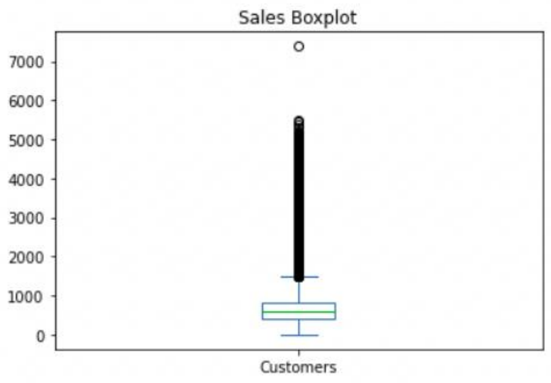


Figure 13: Boxplot for Customers.

Observation: The sales and customers columns include a lot of outliers.

6.1.1 Handling Outliers

Once we have analysed the fluctuations of the target variable, 'Sales' in relation to other features, it would be prudent not to delete or alter outliers if they are a real occurrences.

Let's check the distribution of 'sales' using distplot.

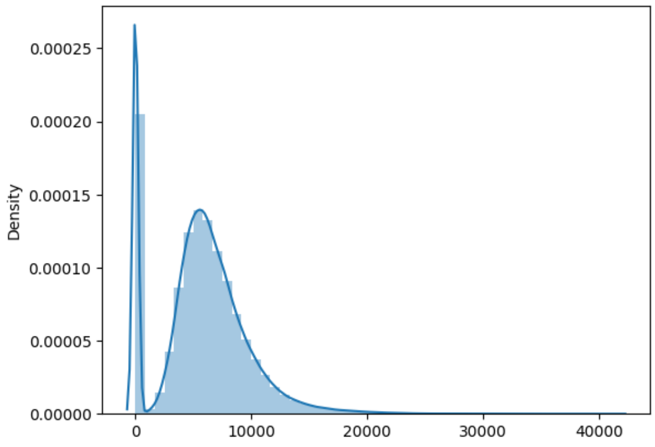


Figure 14: Distribution of Sales. The drop in sales reflects the 0 sales attributable to the stores that were temporarily closed for refurbishment.

6.2 Multivariate Analysis

a. Mean Sales (Daily, Weekly, Monthly, Yearly)

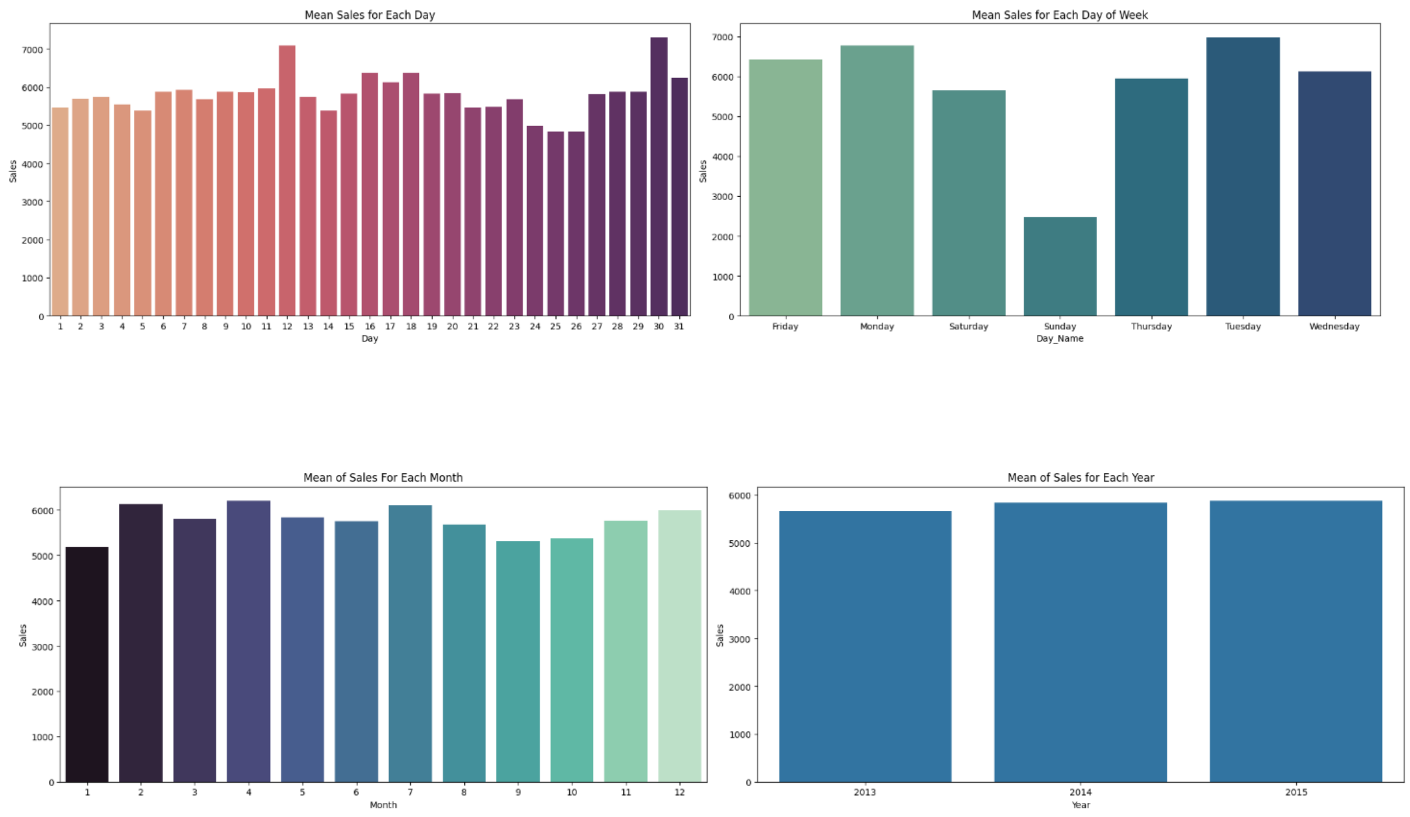


Figure 15: Daily, Weekly, Monthly and Yearly sales

Observation:

We observe that Sales is least on Sundays as most of the stores are closed on Sunday.

Also, Sales on Monday is highest for the whole week.

b. Sales across different Stores

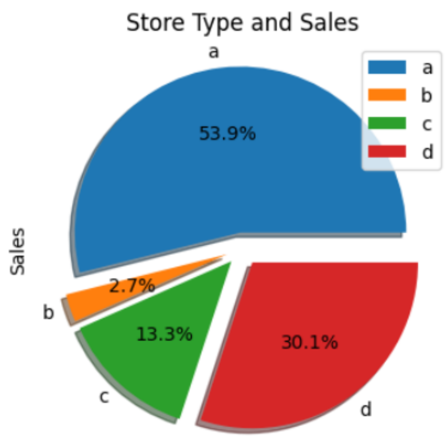


Figure 16: Pie chart for total sales per store. Store Type 'A' has the highest sales with over 53% of sales followed by 'D' with 30.1 %.

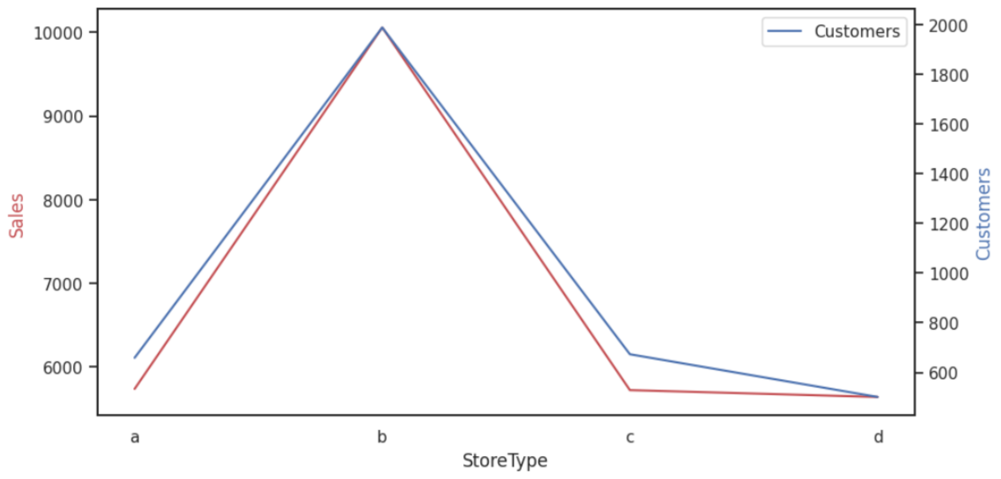


Figure 17: Line plot - Average Sales per store type and customers.

Observation:

Store 'A' had the highest number of total sales but line plot shows that store 'b' has the highest average sales.

c. Sales across Stores with Assortment Types

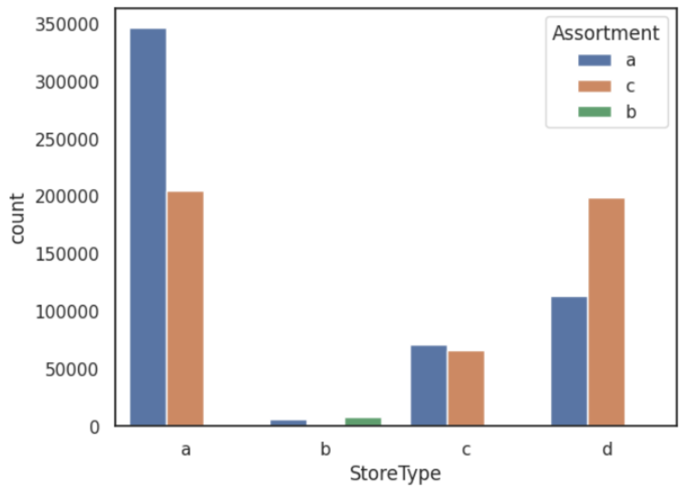


Figure 18: Sales for each assortment level for each store

Observation:

- Store 'b' has all three kinds of assortments while other stores(a,c,d) only have two kinds (a and c).
- Interestingly, store type 'b' with highest average sales looks healthy and a reason for that would be all three assortments involved.

d. Effect of Promos

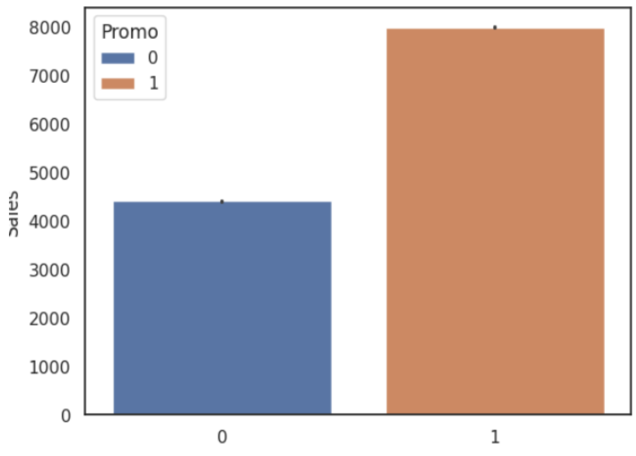


Figure 19: Promo vs Sales

Observation: Promos leads to more sales.

e. Effect of Holidays on Sales

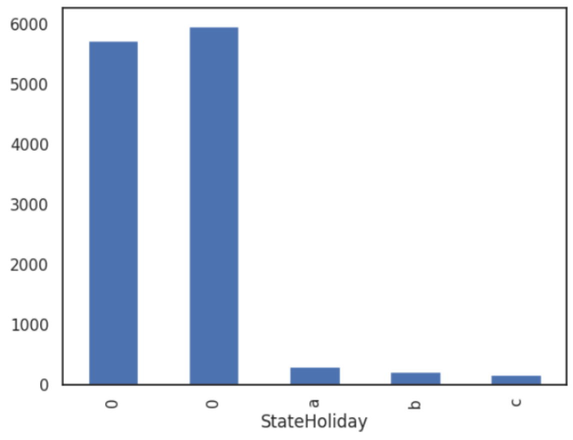


Figure 20: Sales on Holidays and No Holiday

Observation:

- More number of Sales on a 'No Holiday' day as stores are open.
- Among the holidays, Public holiday (a) had comparatively more number of sales then Easter (b) and Christmas (c).

f. Sales over the years (Yearly and monthly analysis)

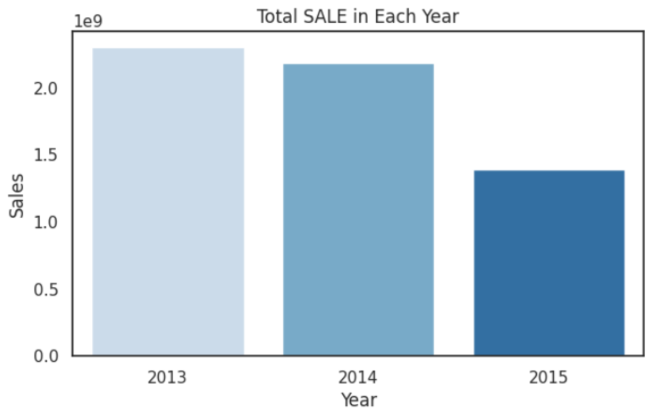


Figure 21: Yearly Sales analysis. 2013 had highest sales while 2015 had lowest sales.

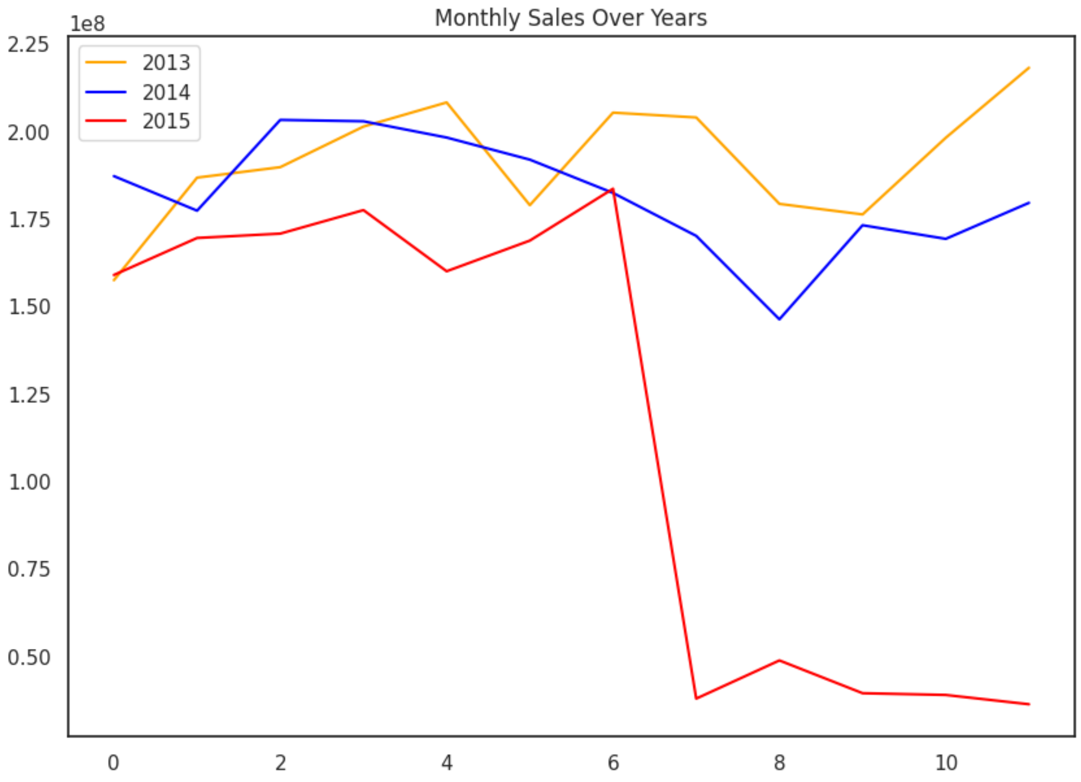


Figure 21: Monthly Sales analysis across years.

Observation:

- By the end of the year, sale increases.
- Sales in 2015 decreased dramatically from June onwards.

g. Sales vs Customer

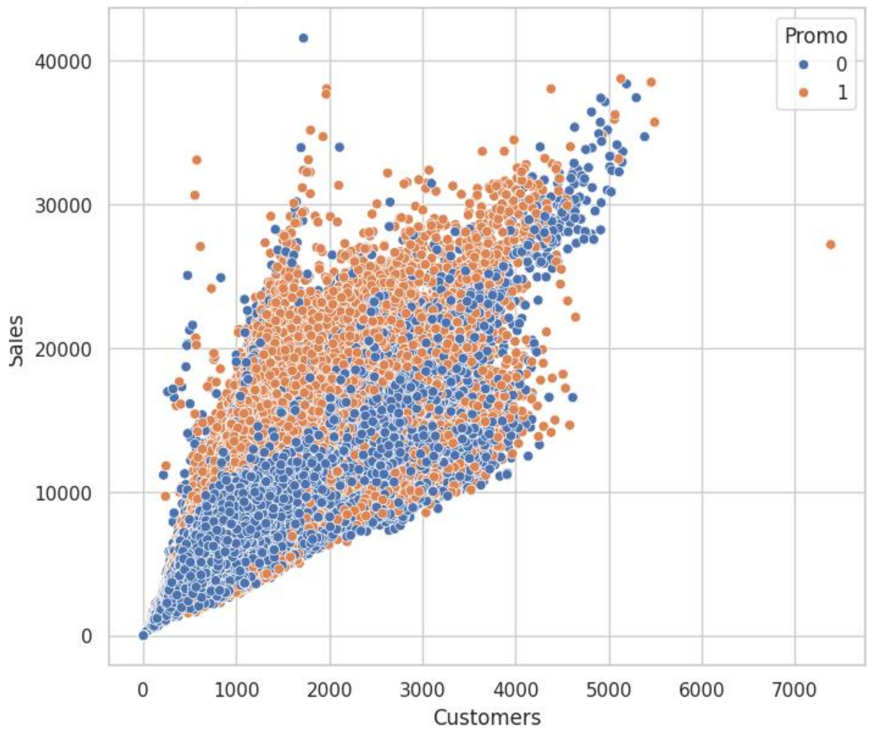


Figure 22: Scatter plot for Sales vs Customers.

Observation: Linear relationship between Sales and Customer. Whenever promo was open, sales and customer increased.

h. Effect of Competition distance on Sales

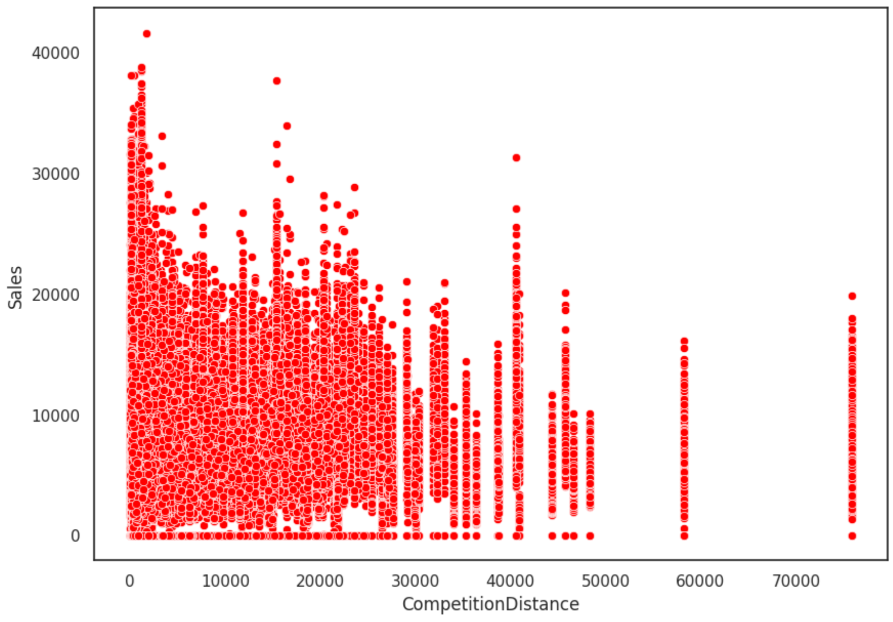


Figure 23: Scatter plot for competition distance and sales

Observation:
Sales increased when competitors were closer even though most stores were closely packed together.

i. Correlation Analysis

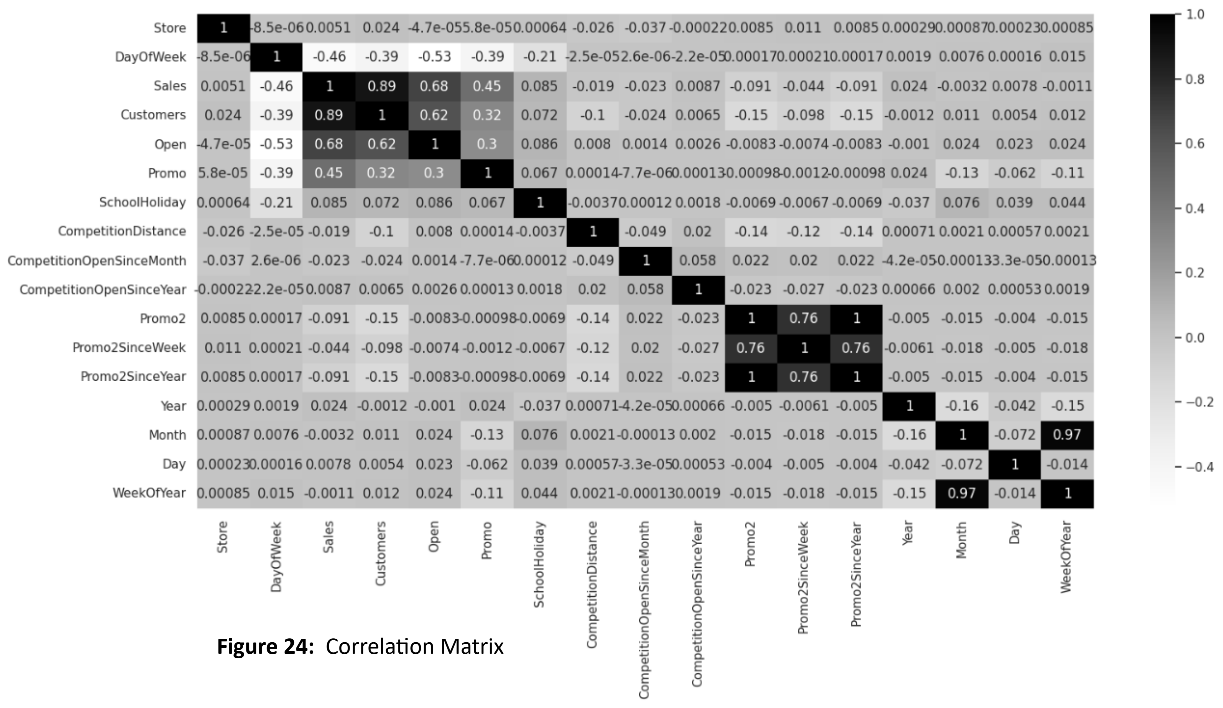


Figure 24: Correlation Matrix

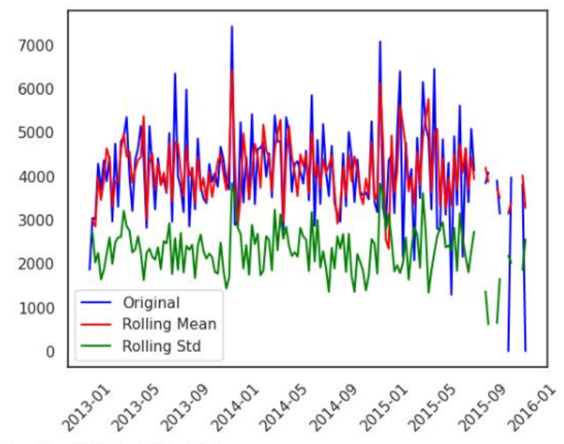
Observation:

- Day of the week has a negative correlation indicating low sales as the weekends, and promo, customers and open has positive correlation.
- CompetitionDistance showing negative correlation suggests that as the distance increases sales reduce, which was also observed through the scatterplot earlier.
- Multicollinearity is present in the dataset, with certain features conveying similar information. Notably, variables such as 'Promo2,' 'Promo2 since week,' and 'Promo2 since year' exhibit multicollinearity, indicating a high degree of correlation among them.

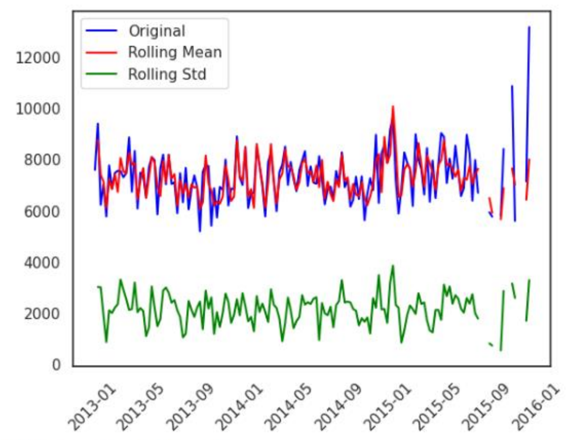
6.3 Time Series Analysis

We will consider one store from each store type that will represent their groups. The trends will be shown on weekly basis.

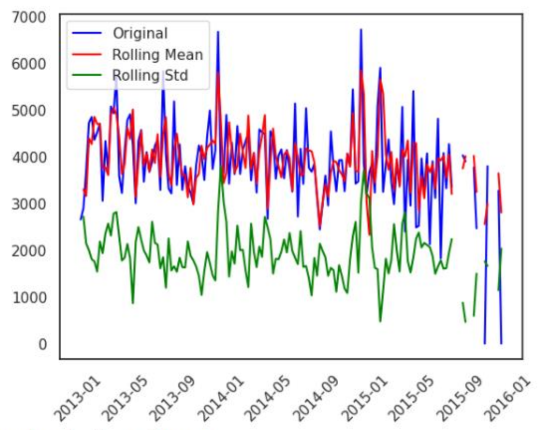
6.3.1 Stationarity Check Using Augmented Dickey Fuller Test



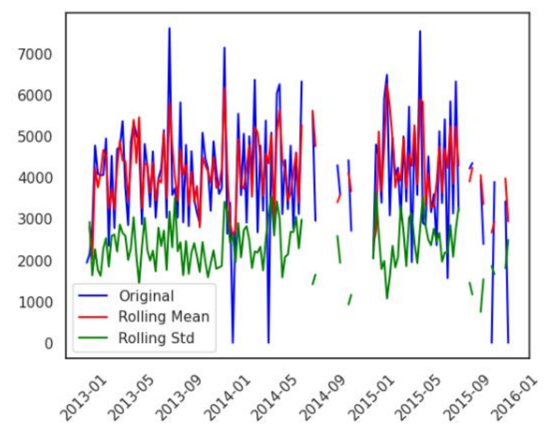
```
Results of Dickey-Fuller Test:
ADF Statistic: -11.085055
p-value: 0.000000
Critical Values:
1% -3.4374010121538072
5% -2.864652794604883
10% -2.5684274413226964
```



```
Results of Dickey-Fuller Test:
ADF Statistic: -7.841342
p-value: 0.000000
Critical Values:
1% -3.4374162506455126
5% -2.8646595140022946
10% -2.568431020409111
```



```
Results of Dickey-Fuller Test:
ADF Statistic: -10.702943
p-value: 0.000000
Critical Values:
1% -3.437363201927513
5% -2.864636122077874
10% -2.5684185607252137
```



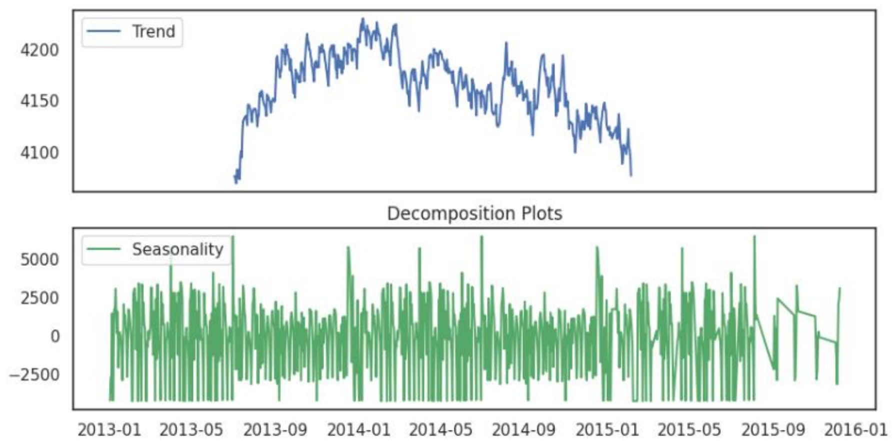
```
Results of Dickey-Fuller Test:
ADF Statistic: -7.078553
p-value: 0.000000
Critical Values:
1% -3.4392057325732104
5% -2.8654483492874236
10% -2.5688512291811225
```

Figure 25: Stationarity check across store types using ADF.

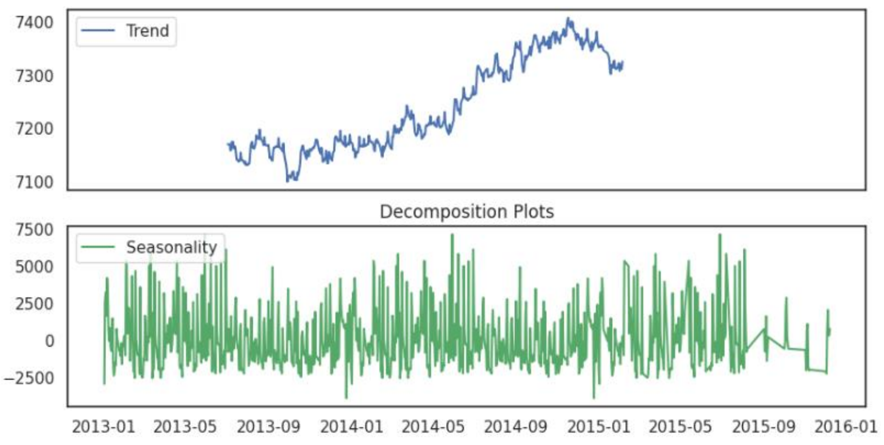
Observation:

- Statistical tests that mean and variation doesn't change much with time, i.e. they are constant. Time series is stationary.

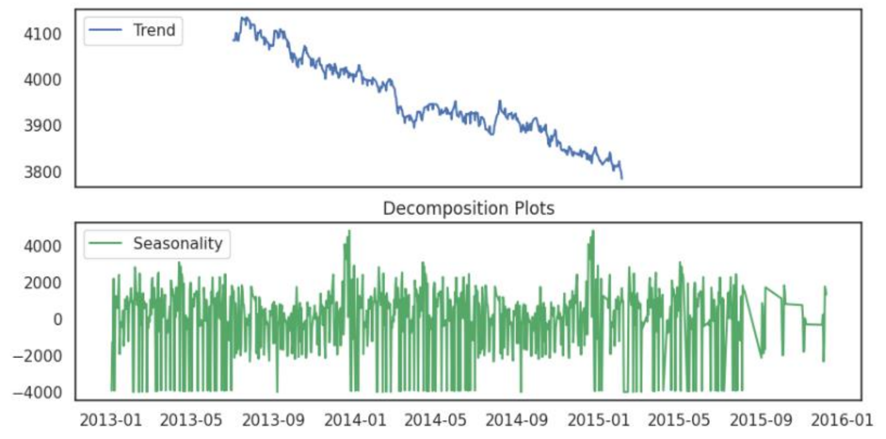
6.3.2 Trend and Seasonality Check



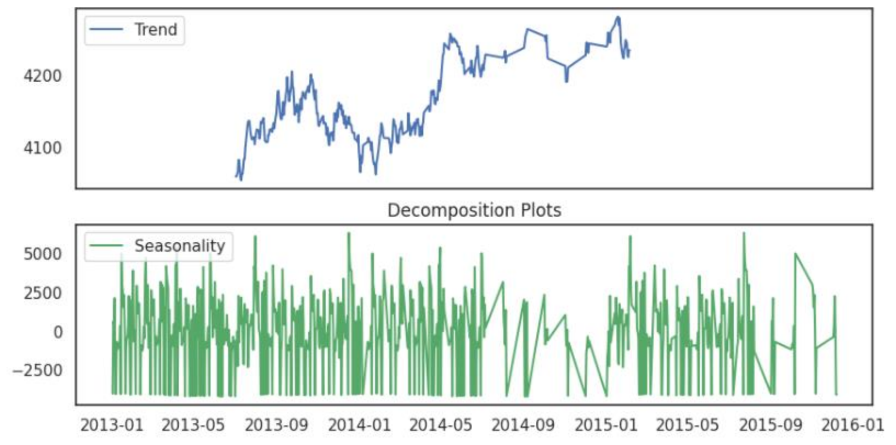
Store A



Store B



Store C



Store D

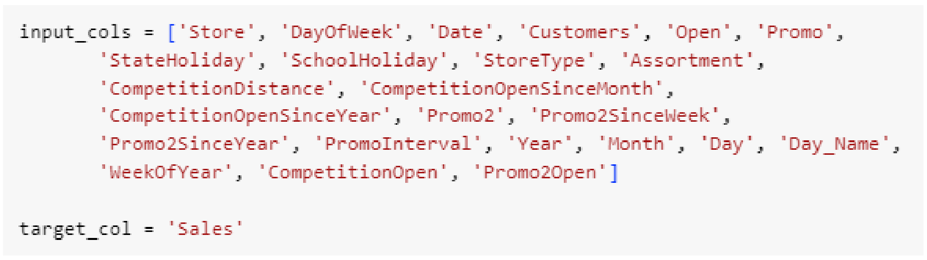
Figure 26: Trend and Seasonality for different store type. store types using ADF.

Observation:

- From the above plots, we can see that there is seasonality and trend present in our data.

7. MODELLING

7.1 Division of data into input and targets



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

target_col = 'Sales'
```

Figure 27: Dependent and Independent columns

After scaling and encoding our data, our final dataset will be divided into input columns and target columns, with input columns (independent features) on which model is being trained and target column being the Sales column.

7.2 Train-Test Split

The dataset is divided into an 80-20 ratio before building any model using ‘**train_test_split**’ from sklearn’s model_selection library.

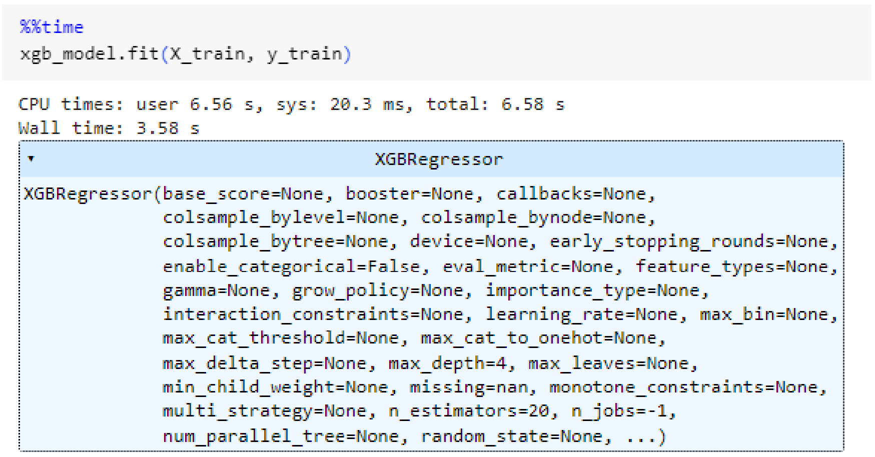
7.3 Model Selection

For modelling, we will use **XGBoost regression** for predicting sales. Reasons for choosing are:

a. XGBoost uses parallel processing for quick performance.
b. Best results on medium-sized datasets
c. Prevents overfitting.

Also, datasets with seasonality make it challenging to establish a linear relationship. In such cases, tree-based machine learning models, known for their resilience to outliers, are employed. An example of this is XGBoost, which is a classification and regression technique built upon gradient boosting ensemble algorithm.

7.4 Model Fitting



```
%%time
xgb_model.fit(X_train, y_train)
```

```
CPU times: user 6.56 s, sys: 20.3 ms, total: 6.58 s
Wall time: 3.58 s
```

```
▾                               XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=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,
             multi_strategy=None, n_estimators=20, n_jobs=-1,
             num_parallel_tree=None, random_state=None, ...)
```

Figure 28: Model fitting using XGBRegressor.

For the initial phase, we employed XGBoost without any set of parameters.

- Upon fitting the model, the initial RMSE was observed to be **1271.0006** which suggested that the model, in its basic configuration, had room for improvement.

We can visualise individual trees using plot_tree() method given below:

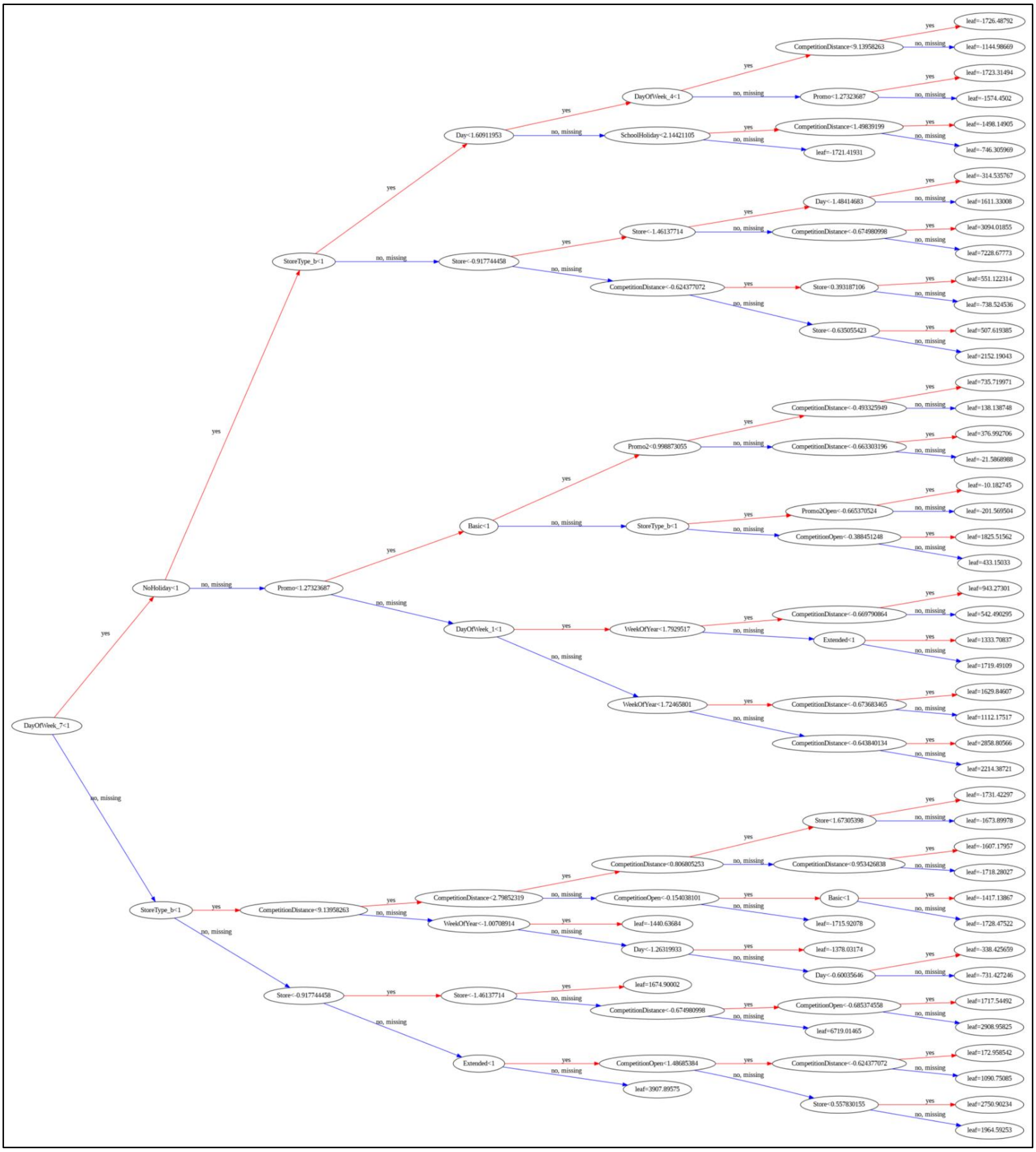
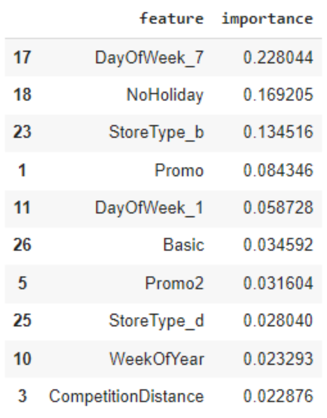


Figure 29: Tree visualization of XGB model

7.5 Feature Importance

Let’s look at the important features used in sales predictions.



	feature	importance
17	DayOfWeek_7	0.228044
18	NoHoliday	0.169205
23	StoreType_b	0.134516
1	Promo	0.084346
11	DayOfWeek_1	0.058728
26	Basic	0.034592
5	Promo2	0.031604
25	StoreType_d	0.028040
10	WeekOfYear	0.023293
3	CompetitionDistance	0.022876

Figure 30: Feature Importance

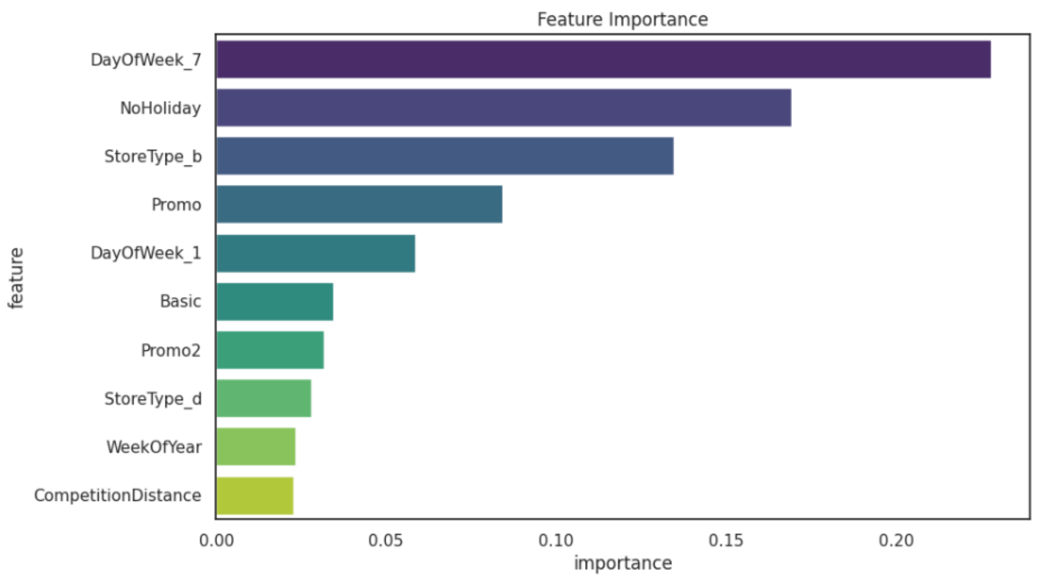


Figure 31: Feature Importance using barplot.

DayOfWeek_7 was the most important feature along with **NoHoliday.**

7.6 Hyperparameter Tuning

Following feature selection, we proceeded with hyperparameter tuning, where we systematically explored different configurations that led to improved performance.

After tuning, values required for parameters are:

```
model = XGBRegressor(n_jobs=-1,n_estimators=1000,learning_rate=0.2,
                     max_depth=10,colsample_bytree=0.7,booster="gbtree")
```

Figure 32: Hyperparameters

7.7 Improved Model Performance

After completing feature selection and hyperparameter tuning, the refined XGBoost model showed substantial improvement over the initial RMSE of 1271.006 to **final RMSE of 691.433** with **RMPSE of 11.982 %**.

8. KEY RESULTS

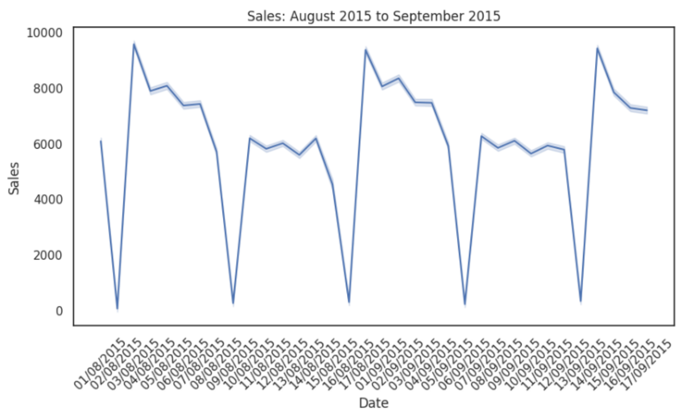


Figure 33: Predicted Sales from August 2015 to September 2015

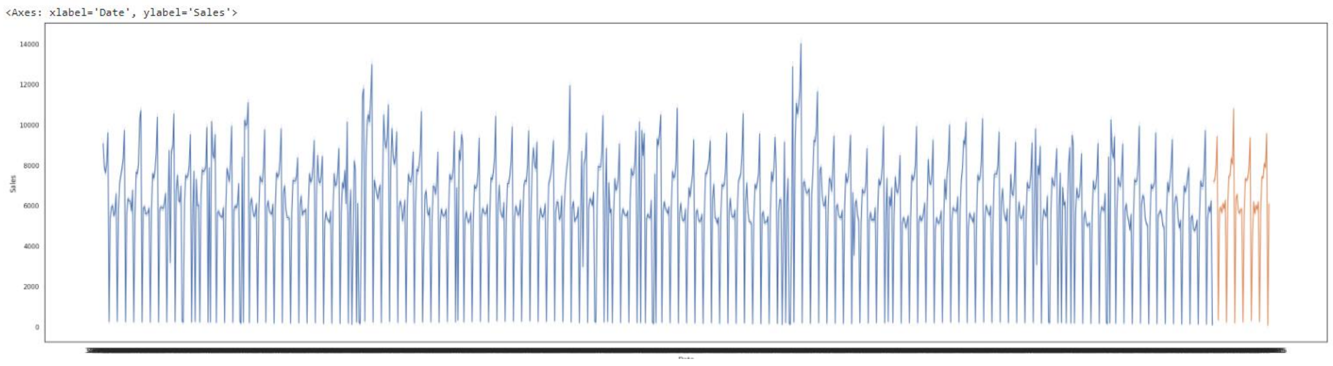


Figure 33: Predicted Sales Plot

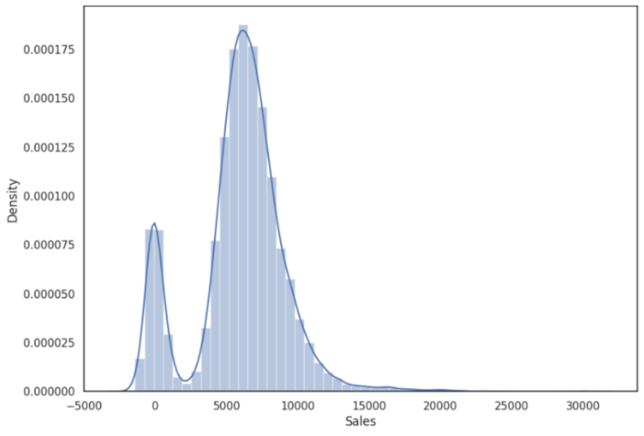


Figure 34: KDE plot for forecasted sales

The predicted sales KDE plot is similar to train dataset indicating close predictions with the actual dataset which is good.

9. CONCLUSION

- **Data Handling**
  - Applied mean, median, and mode imputation for missing values.
  - Utilized Standard Scaler and One-hot Encoder for numerical and categorical features.
- **Feature Engineering**
  - Extracted temporal information from the date feature.
- **Conducted EDA and Time Series Analysis** to understand the patterns.
- **Modelling**:
  - Selected XGBoost Regressor for its robust performance.
  - Initial RMSE: 1271 and Final Tuned RMSE: 691 with RMPSE: 11.98%

Key takeaways:

a. Effective preprocessing and feature engineering improved model interpretability.
b. XGBoost Regressor, coupled with hyperparameter tuning, resulted in a significant MSE reduction.
c. Store of type 'A' are more popular than 'B' and 'C' in terms of weekly sales.
d. Sales are significantly affected by the wee of the year. Weeks with holidays have higher sales as compared to non-holiday weeks.
e. Customer and Sales increases at the time of Promos in stores.

10. LIMITATIONS

- Temporal Scope: limited period from January 2013 to July 2015.
- Geographical Specificity: Limited to Germany
- Data Size limitation
- Missing Data
- Limited information about customers
- Scope of predictive tasks

REFERENCES

- J. Fan, Q. Yao, Nonlinear Time Series. Nonparametric and Parametric Methods, Springer Series in Statistics, Springer, New York (2003).
- Chen, T. & Guestrin, C. (2016) „XGBoost: A scalable tree boosting system", in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining- KDD '16. New York, USA: ACM Press, p. 785–794
- Ghosh, R. & Purkayastha, P. (2017) „Forecasting pro tability in equity trades using random forest, support vector machine, and xgboost", in 10th International Conference on Recent Trades in Engineering Science and Management, p. 473–486.
- Gurnani, M. et al. (2017) „Forecasting of sales by using the fusion of machine learning techniques", in 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI). IEEE, p. 93–101
- Gumus, M. & Kiran, M. S. (2017) „Crude oil price forecasting using XGBoost", in 2017 International Conference on Computer Science and Engineering (UBMK). IEEE, p. 1100–1103.
- https://medium.com/@amdeamd7/rossmann-pharmaceutical-sales-prediction-a-deep-learning-approach-9c4434fc809c
- https://medium.com/analytics-vidhya/rossmann-store-sales-prediction-998161027abf
- Andrabi, Syed & Alice, Dr & Srivastava, Siddharth. (2022). Sales Forecasting using XGBoost. 10.36227/techrxiv.21444129.v1.

- Grid Search Optimization (GSO) Based Future Sales Prediction for Big Mart | IEEE Conference Publication |IEEE Xplore.
- For XGBoost parameter tuning [Online]. Available: http://www.analyticsvidhya.com
- Forecasting of sales by using fusion of machine learning techniques | IEEE Conference Publication | IEEE Xplore