Professor Eghbal Rahimikia

DATA71011 | Report

Report: <u>Rossmann Sales Analysis & Prediction</u>

Student ID: 11356488

Word Count: 1650

(Excluding Headings,FigureCaption,Tables,References)

MSc Data Science

The University of Manchester

DATA71011 | Understanding Data and their Environment

Contents

1. Introduction..2

1.1 Dataset Descriptions..3

2. Data Preprocessing..4

2.1 Missing Data Imputation..4

● Missing Completely at Random (MCAR)..5

● Missing not at Random (MNAR)..7

2.2 Merging Dataset..7

2.2 Feature Extraction..8

2.2 Feature Creation..8

2.2 Scaling Numerical Features..9

2.2 Encoding Categorical Features..10

3.Exploratory Data Analysis..12

3.1 Univariate EDA..12

3.1.1 Outlier Handling..13

3.2 Multivariate EDA..14

3.2.2 Correlation Heatmap..20

4. Modelling..21

4.1 Key Evaluation Metric..21

4.2 Model Selection..21

4.3 Model Fitting..22

4.4 Feature Importance..23

4.5 Model Tuning..23

5. Key Results..24

6. Conclusion..26

7. Limitations..26

8. Recommendation..27

References..27

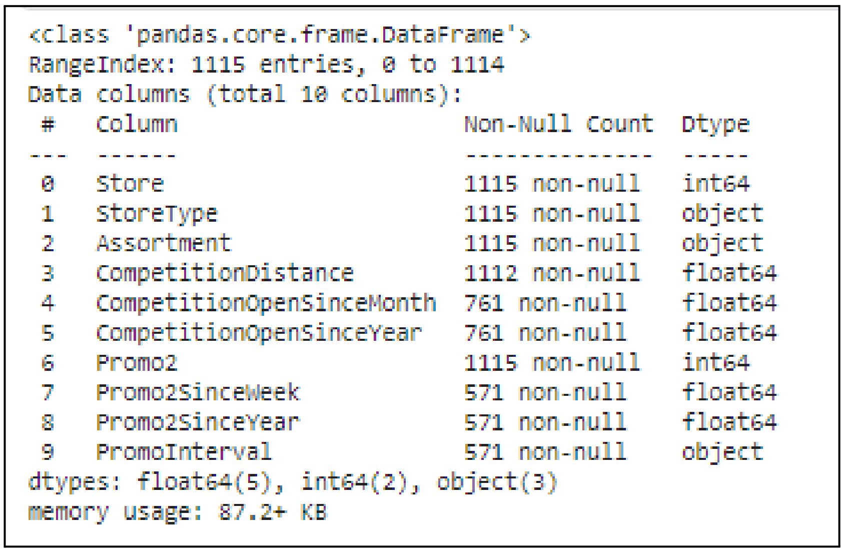
1. Introduction

The Rossmann Dataset containing historical records for 1115 drug stores situated across diverse regions in Germany, is valuable for retail sales analysis and predicting sales for each store up to six weeks in advance. Sales at different stores are influenced by distinct factors such as competition, holidays, seasonality, promotions and location. Prior to forecasting, it is essential to comprehend and preprocess the dataset thoroughly to ensure the accuracy of predictions. This report aims to provide a clear understanding of the procedures involved in data preprocessing and forecasting applied to the Rossmann dataset analysis.

1.1 Dataset Descriptions

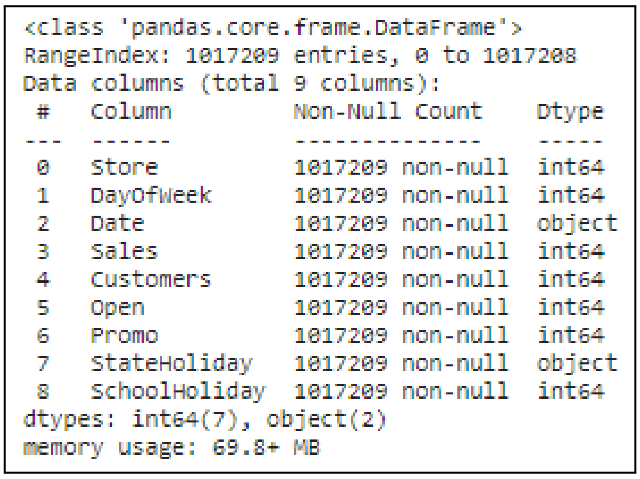
The three datasets provided are : “store.csv”, “train.csv”, “test.csv”.
“store.csv” contains 1115 records of stores and features like promotion, competition and StoreType.
“train.csv” contains 1017209 records of daily sales of different stores from 2013 - 2015.
“test.csv” contains 1115 stores data with sales and customers columns to be predicted for 6 weeks (Aug,2015 to Sept,2015).

“**Store**” and “**Train**” datasets can be joined together based on a common column - ‘store’ ID. Similarly **Store** and **Test** Datasets will 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
```

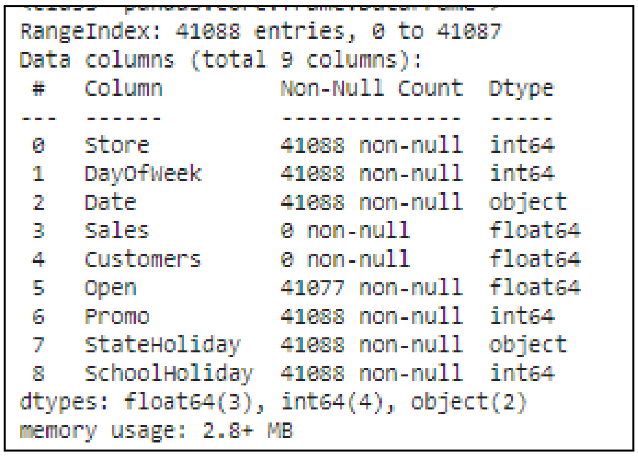
Figure.1 store.csv with non-null value counts
(handled during data cleaning stage)



```
<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 strain.csv features and their

datatypes with 1017209 records.



```
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
```

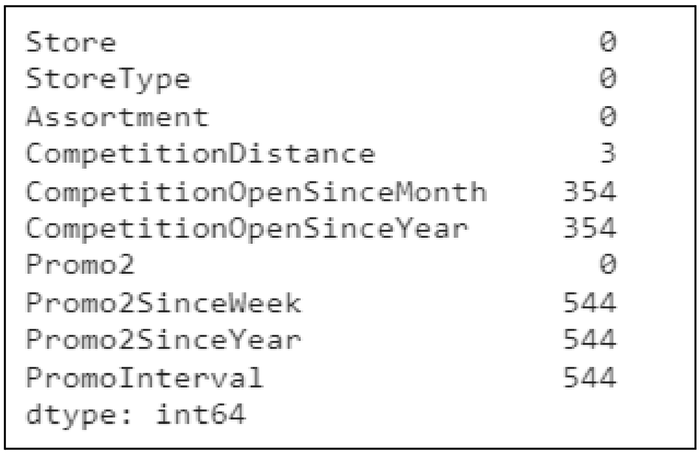
Figure.3 test.csv with sales and customers

columns having no values.

2. Data Preprocessing

Data cleaning and preparation for further analysis involves the following steps:

2.1 Missing Data 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
For store dataset , Null value counts are represented and columns that require missing data imputations are -

- CompetitionDistance
- CompetitionOpenSinceMonth
- CompetitionOpenSinceYear
- Promo2SinceWeek
- Promo2SinceYear
- PromoInterval

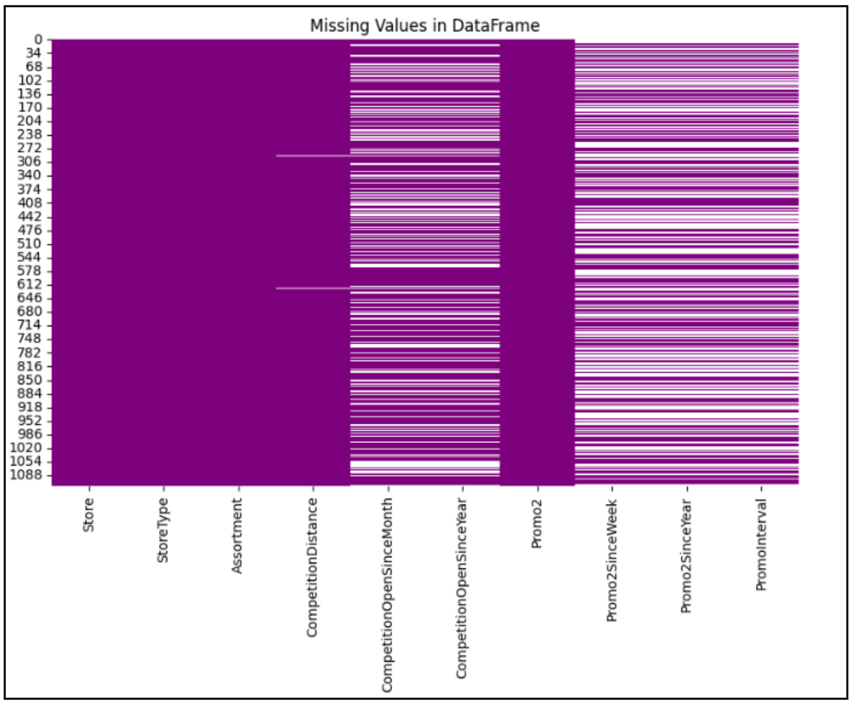


Figure .5 HeatMap representing Null Values

- ## Missing Completely at Random (MCAR)

We should first check the skewness of the data to decide the imputation type.

1) CompetitionDistance

It seems like the data is right skewed and positive. Clearly since the mean is biased by outliers, we will apply Median Imputation.

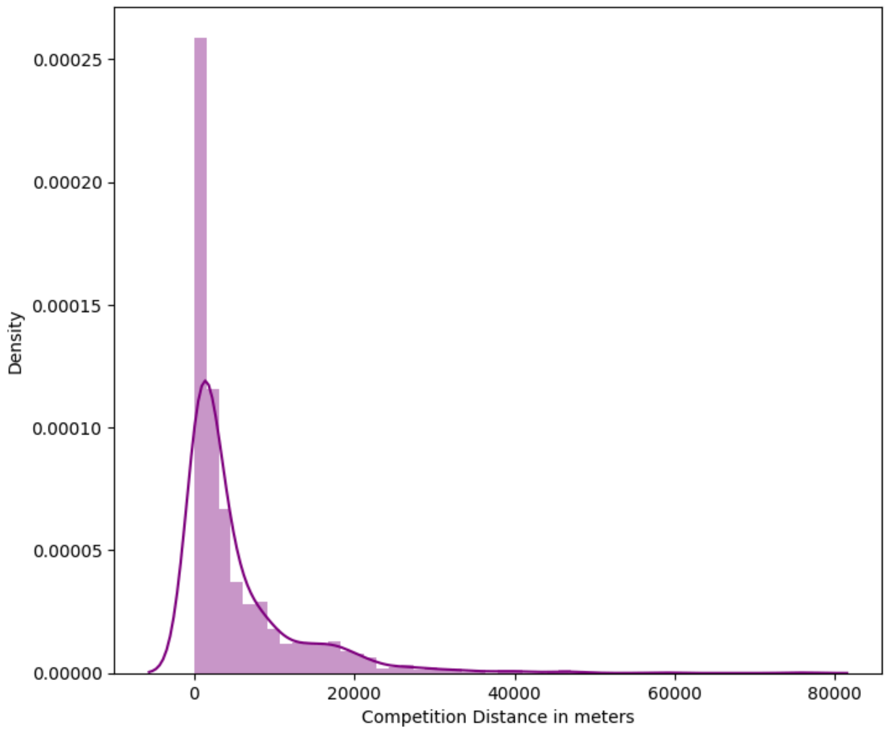


Figure 3. CompetitionDistance KDE plot.
Skewness:**2.929**

2) CompetitionOpenSinceMonth

The data appears symmetrical as explained by negative skewness and KDE curve. Therefore, Mean Imputation will be the best choice for replacing null values.

3) CompetitionOpenSinceYear

The data is not symmetrical or skewed and hence there is not right way to replace null values. Applying Mean Imputation

{'Mean': 2008.6688567674114}

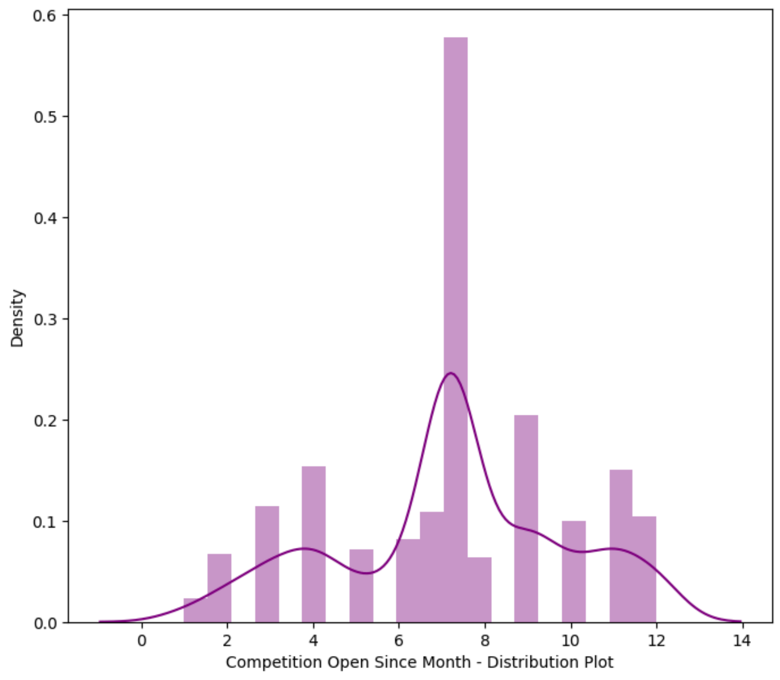


Figure 3. CompetitionOpenSinceMonth KDE distribution Skewness**: -0.206.**
{'Mean': 7.2247043363994745}

- **Missing not at Random (MNAR)**

These features are not missing at random since they all depend on Promo2.

1) **Promo2SinceWeek, Promo2SinceYear, PromoInterval**

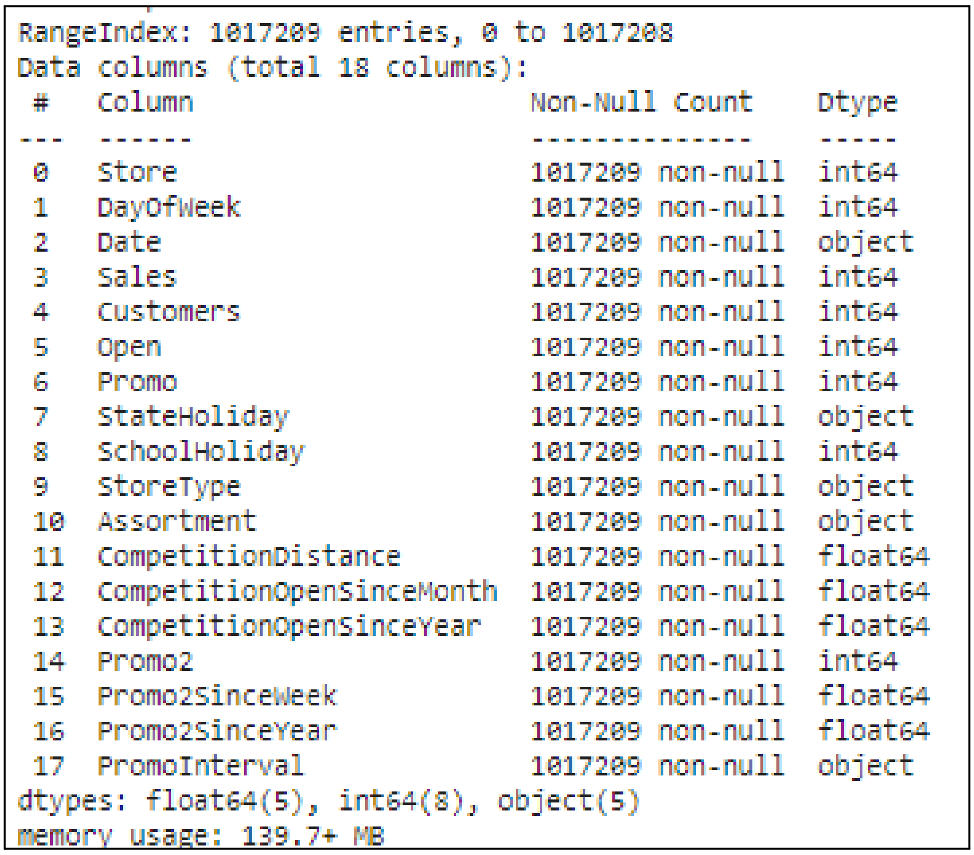
The 3 features above depend on ‘Promo2’ values. Wherever Promo2=0 (no continuous promotional activities for those stores) all linked Promo2 related features will also be 0. So, we replace the missing values with **‘0**’.

2.2 Merging Dataset

We now merge the Store dataset with train and test using **left join** which ensures that all the values are retained.

The merged train dataset comprises 1,017,209 rows and 18 columns.

The merged test dataset comprises 41,088 rows and 18 columns.



```
RangeIndex: 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  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
 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: float64(5), int64(8), object(5)
memory usage: 139.7+ MB
```

Figure.4 Final Merged train dataset

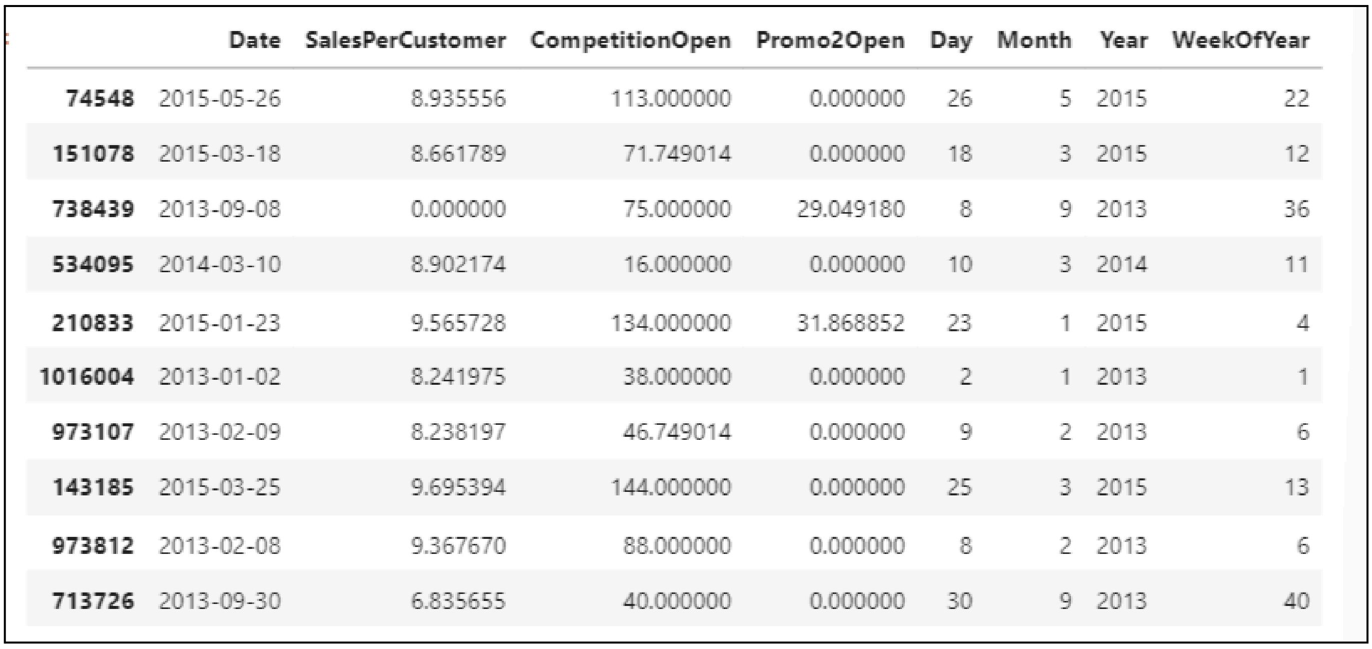
2.2 Feature Extraction

We have extracted date features from the ‘Date’ column for time-required data explorations. The extracted features are **Day, Month, Year, WeekOfYear.** These features are appended to the dataset as new features.

2.2 Feature Creation

- We introduced a new feature **‘CompetitionOpen’** from ‘CompetitionOpenSinceMonths' and ‘CompetitionOpenSinceYear’ to evaluate the duration in months since the nearest competitor store opened.
- Similarly we created a feature **‘Promo2Open’** from Promo2SinceWeek and Promo2SinceYear to estimate the duration in months since the store started applying Promo2.
  (The negative Values in Promo2 and CompetitionOpen have been replaced with ‘0’ value, as there will be no significance of having negative values.)

- Another feature was created called **‘SalesPerCustomer’** from Sales and Customers, to get a better understanding of the underlying patterns.

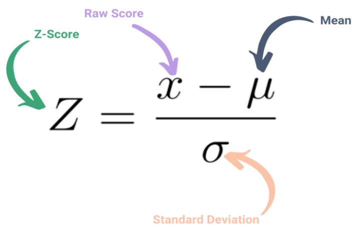


	Date	SalesPerCustomer	CompetitionOpen	Promo2Open	Day	Month	Year	WeekOfYear
74548	2015-05-26	8.935556	113.000000	0.000000	26	5	2015	22
151078	2015-03-18	8.661789	71.749014	0.000000	18	3	2015	12
738439	2013-09-08	0.000000	75.000000	29.049180	8	9	2013	36
534095	2014-03-10	8.902174	16.000000	0.000000	10	3	2014	11
210833	2015-01-23	9.565728	134.000000	31.868852	23	1	2015	4
1016004	2013-01-02	8.241975	38.000000	0.000000	2	1	2013	1
973107	2013-02-09	8.238197	46.749014	0.000000	9	2	2013	6
143185	2015-03-25	9.695394	144.000000	0.000000	25	3	2015	13
973812	2013-02-08	9.367670	88.000000	0.000000	8	2	2013	6
713726	2013-09-30	6.835655	40.000000	0.000000	30	9	2013	40

Figure.5 Newly created and extracted features.

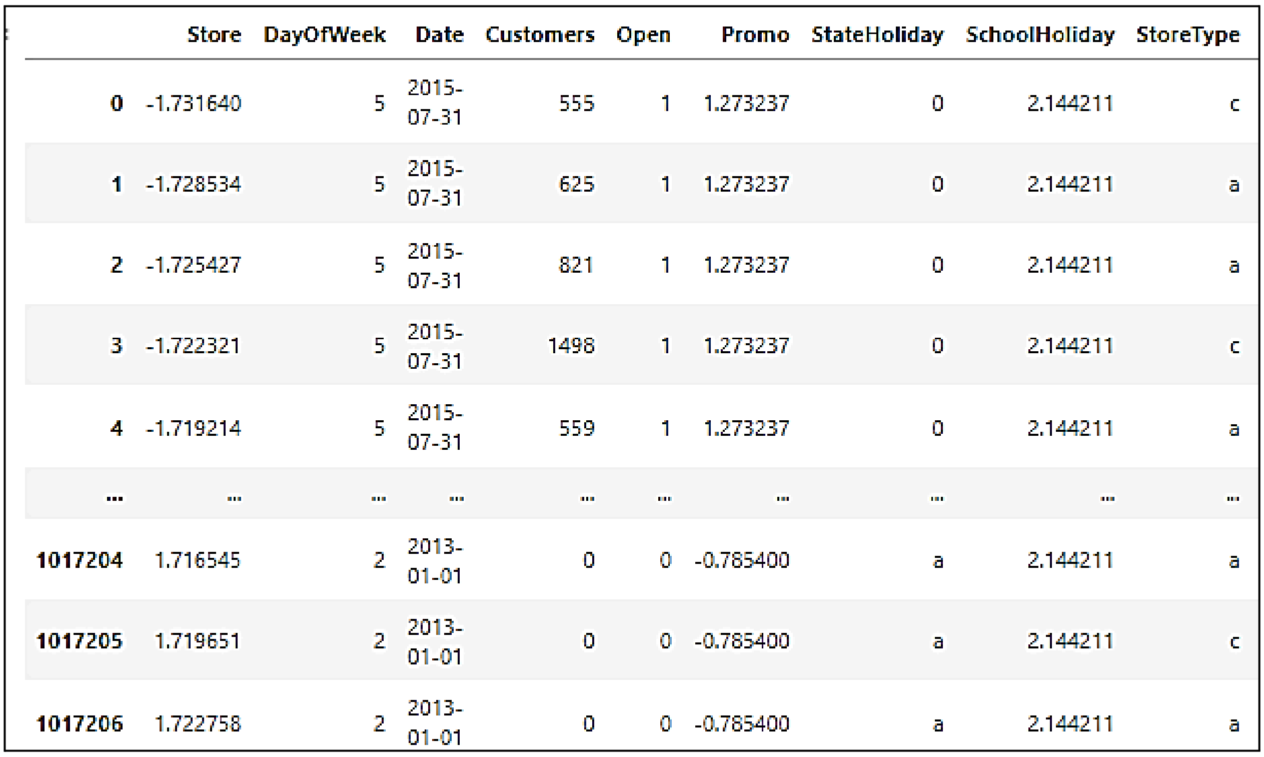
2.2 Scaling Numerical Features

Standard Scaling was applied on this dataset. This process enhances ML model performance by standardising numerical variables to a predefined common scale.



$$Z = \frac{x - \mu}{\sigma}$$

We decided to apply Standard scaling because our data exhibits skewness but more generally follows normal distribution. This method is particularly suitable for numeric features with normal distribution. This step was **implemented post Exploratory Data Analysis.**



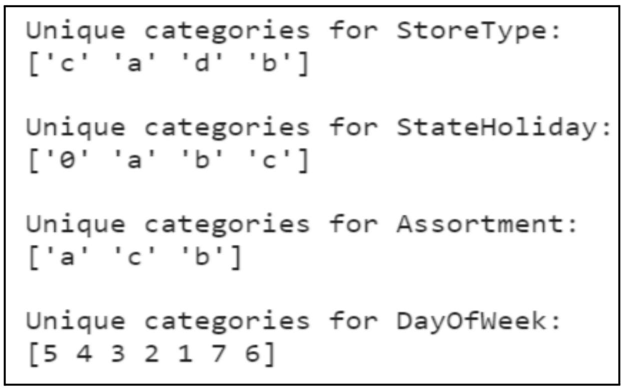
	Store	DayOfWeek	Date	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType
0	-1.731640	5	2015-07-31	555	1	1.273237	0	2.144211	c
1	-1.728534	5	2015-07-31	625	1	1.273237	0	2.144211	a
2	-1.725427	5	2015-07-31	821	1	1.273237	0	2.144211	a
3	-1.722321	5	2015-07-31	1498	1	1.273237	0	2.144211	c
4	-1.719214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

Figure.6 Scaled train dataset.

2.2 Encoding Categorical Features

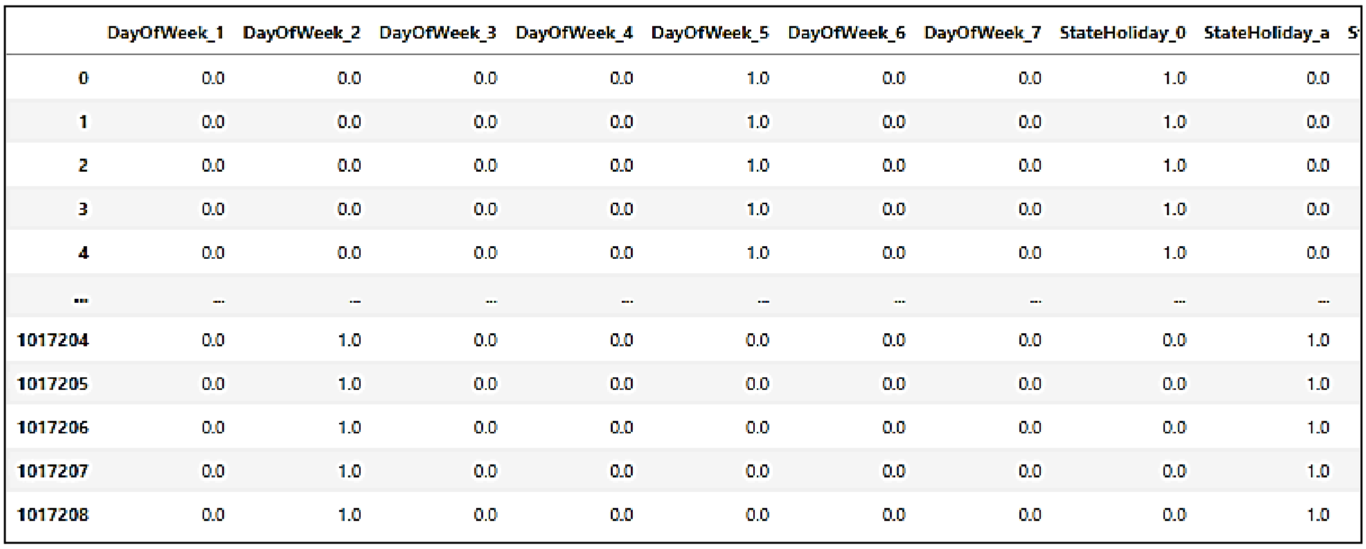
Feature Encoding involves converting categorical variables into binary or integer format to enhance predictive accuracy for models. In this dataset, we have four categorical variables: **'StoreType', 'StateHoliday','Assortment**' & '**DayofWeek.**

For our dataset, we will encode the values using **OneHotEncoder** from sklearn's preprocessing library which will exhibit a binary column for each category.



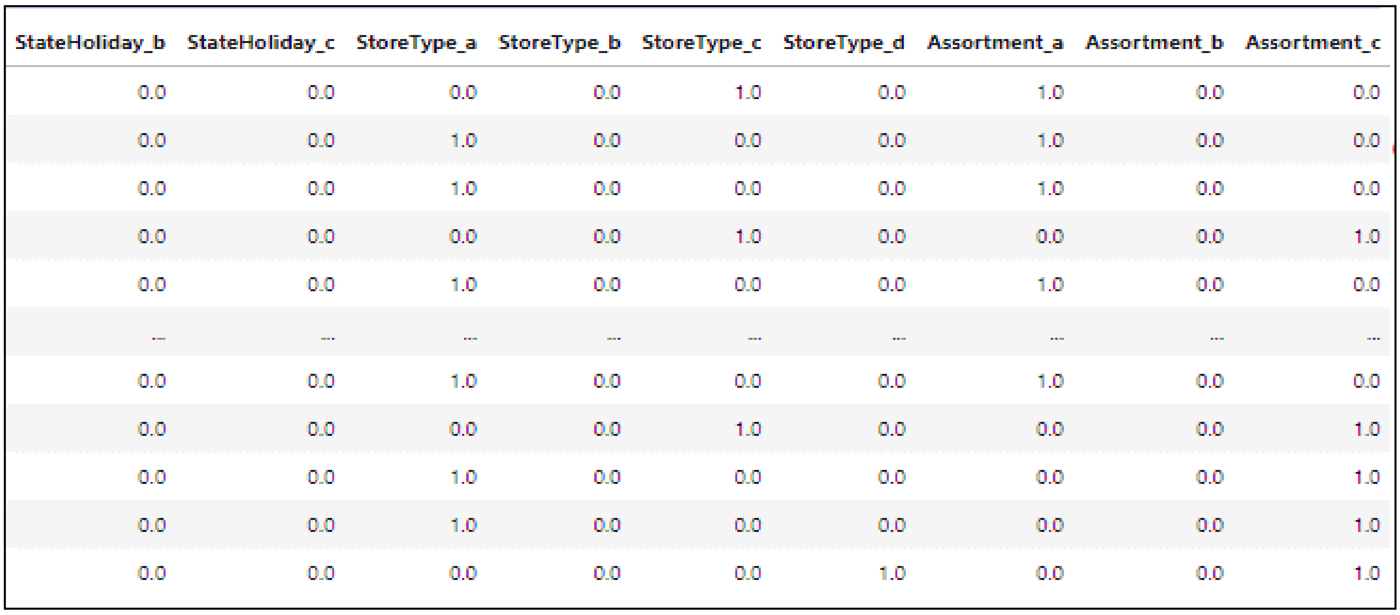
```
Unique categories for StoreType:
['c' 'a' 'd' 'b']

Unique categories for StateHoliday:
['0' 'a' 'b' 'c']

Unique categories for Assortment:
['a' 'c' 'b']

Unique categories for DayOfWeek:
[5 4 3 2 1 7 6]
```

Figure.7 Categories of each categorical Feature



	DayOfWeek_1	DayOfWeek_2	DayOfWeek_3	DayOfWeek_4	DayOfWeek_5	DayOfWeek_6	DayOfWeek_7	StateHoliday_0	StateHoliday_a	S
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	
1017204	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
1017205	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
1017206	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
1017207	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
1017208	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

Figure.8 Encoded Features/Columns



StateHoliday_b	StateHoliday_c	StoreType_a	StoreType_b	StoreType_c	StoreType_d	Assortment_a	Assortment_b	Assortment_c
0.0	0.0	0.0	0.0	1.0	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	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	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	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	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0

Figure.9 Encoded Features/Columns

3.Exploratory Data Analysis

3.1 Univariate EDA

The distributions of Sales and Customers were analysed for any outliers/anomalies using KDE and Box plots.

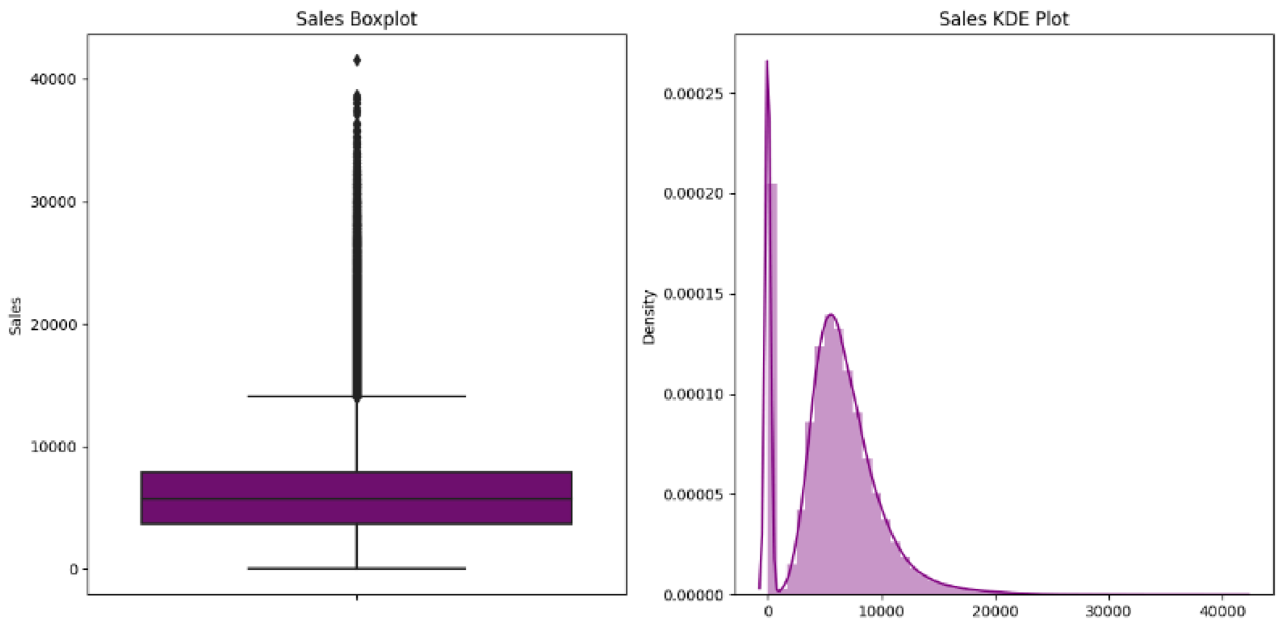


Figure.10 Boxplot(left) showing outliers & KDE(Right) showing distribution for **Sales.**

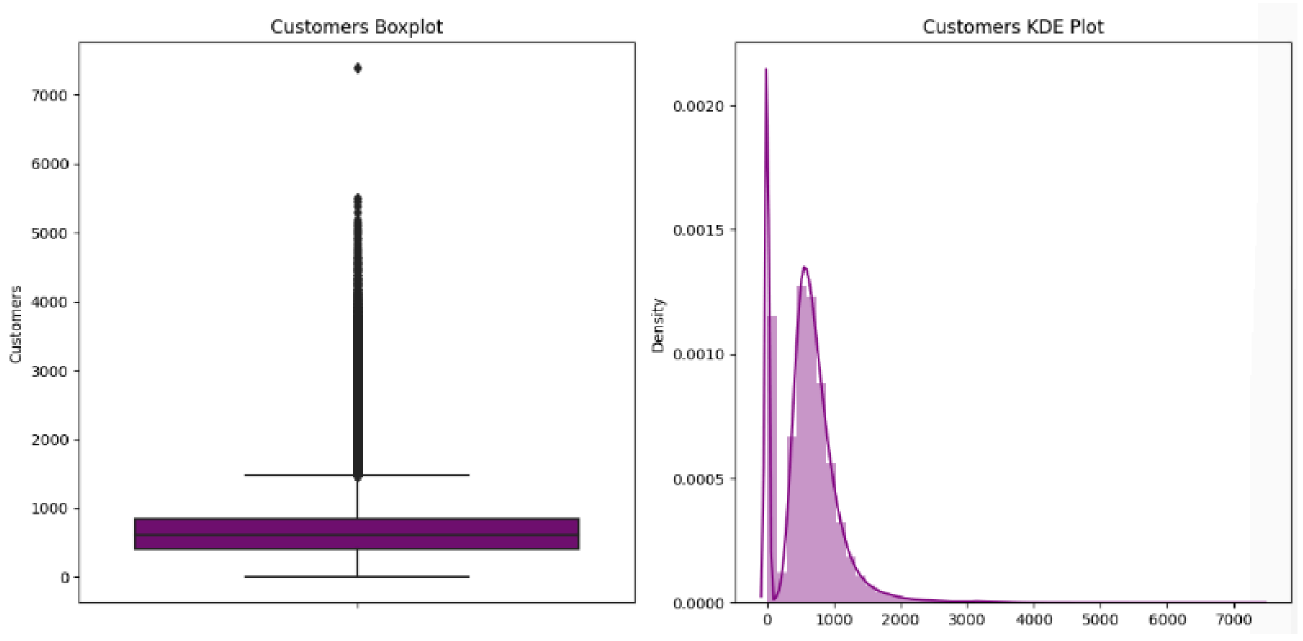


Figure.11 Boxplot(left) showing outliers & KDE(Right) showing distribution for **Customers.**

Observation:

- From figure.14 we can observe that sales>=15,000 (3.3% data) can be considered as outliers. Potential factors contributing to increased sales include popularity and lack of competition.
- Some stores have higher customer count probably due to heavy promotions.
- Due to these outliers both sales and customers have right-skewed distributions with strong positive correlation of 0.82 so, higher the customers, higher the sales.

3.1.1 Outlier Handling

When we have analysed relationships between 'Sales' and other features, genuine outliers should not be deleted or altered as they represent real occurrences. Doing so could distort the underlying patterns leading to biased insights.

- The drop in sales (outliers) to 0 indicate the stores were temporarily closed for renovation.

3.2 Multivariate EDA

Let us analyse how each **storetype** is performing in terms of **sales** and **customers.**

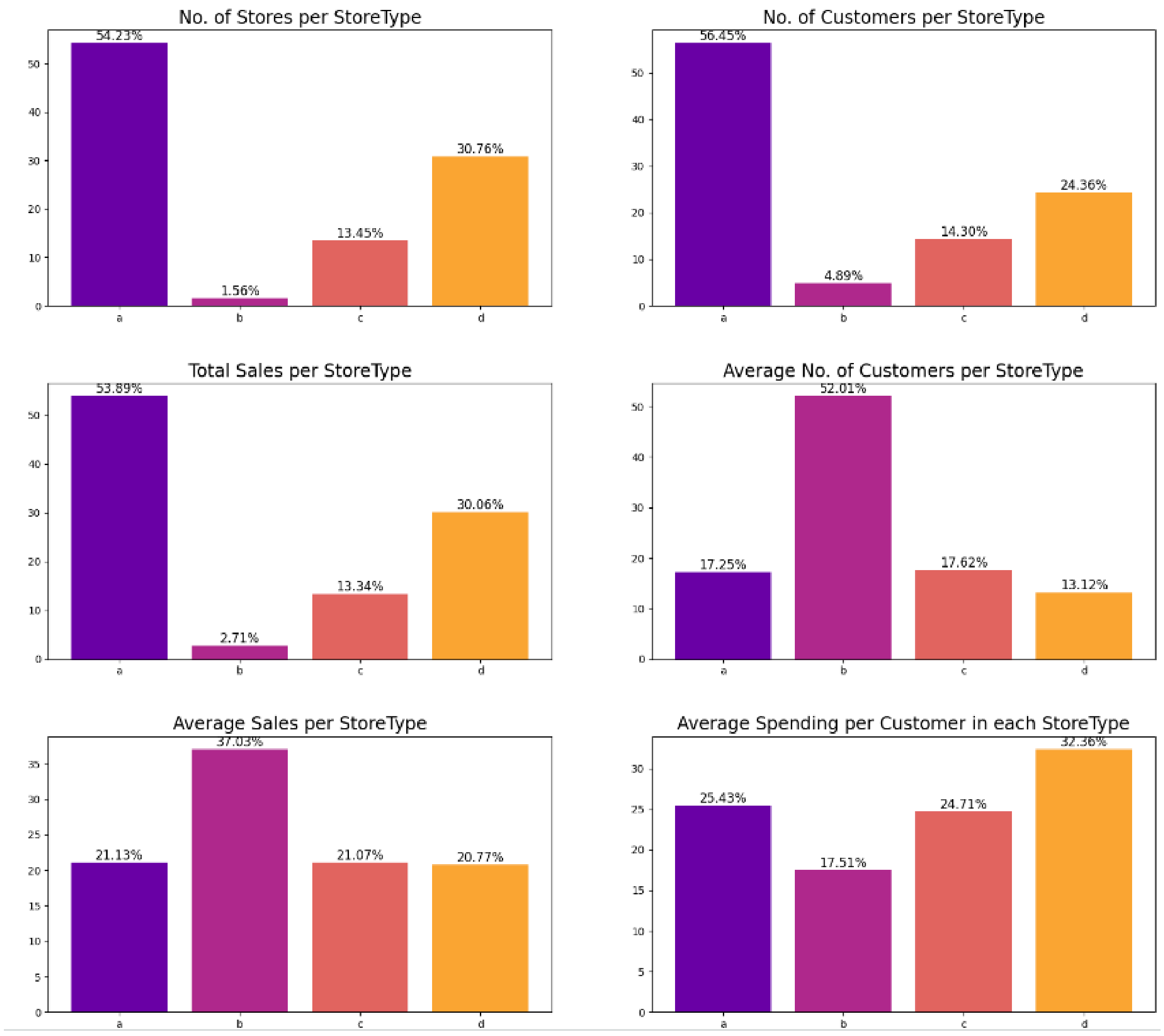


Figure.12 Different storetype performance with total/average sales and customers using Bar plots.

Observation:

- StoreType **'A'** is the most common with maximum no. of stores and has the greatest number of sales over **53%** followed by 'D'.
- Surprisingly, StoreType **'D'** exhibits the highest average spending per customer**.** Reason being the higher competition distance which makes it a key player within its locality.

Effect of Assortments on Sales

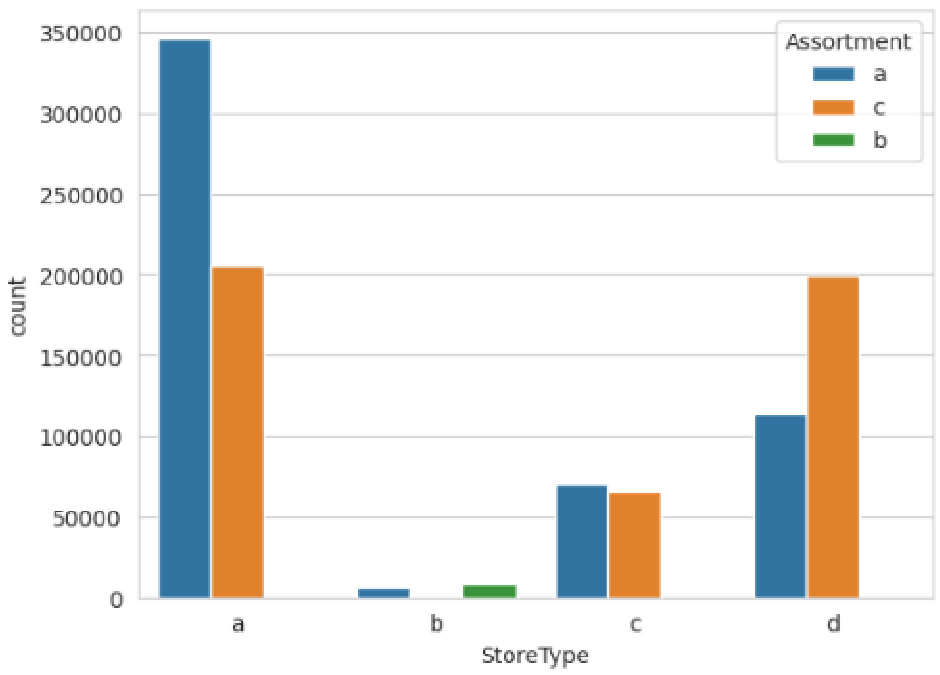


Figure.13 Assortment Types (a=basic, b=extra, c=extended) available on different StoreTypes

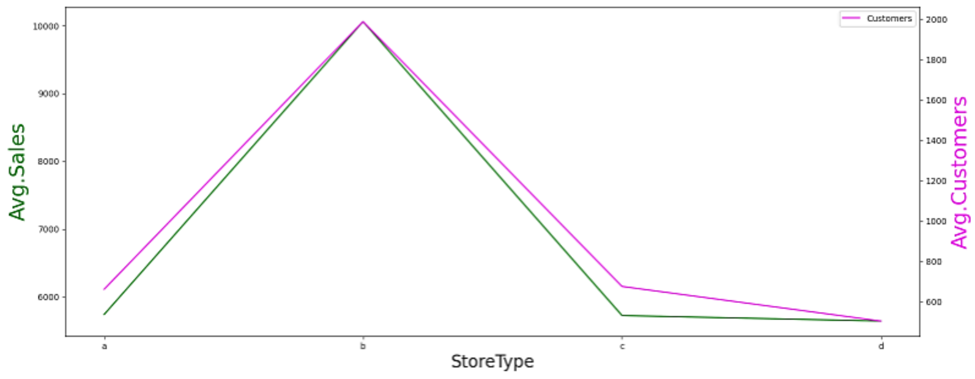


Figure.14 Sales and Customer distribution for StoreTypes

Observation:

- A-type and C-type assortments/products are most common as seen in Fig.18.
- StoreType D has highest average SalesPerCustomer due to assortment C, since having variety of products tends to improve customer spending patterns.
- StoreType ‘B’ is the only one having all 3 assortments attracting more customers, which is why the average sales are highest for ‘B’ (Fig 19).

Let us now understand sales pattern by week-days

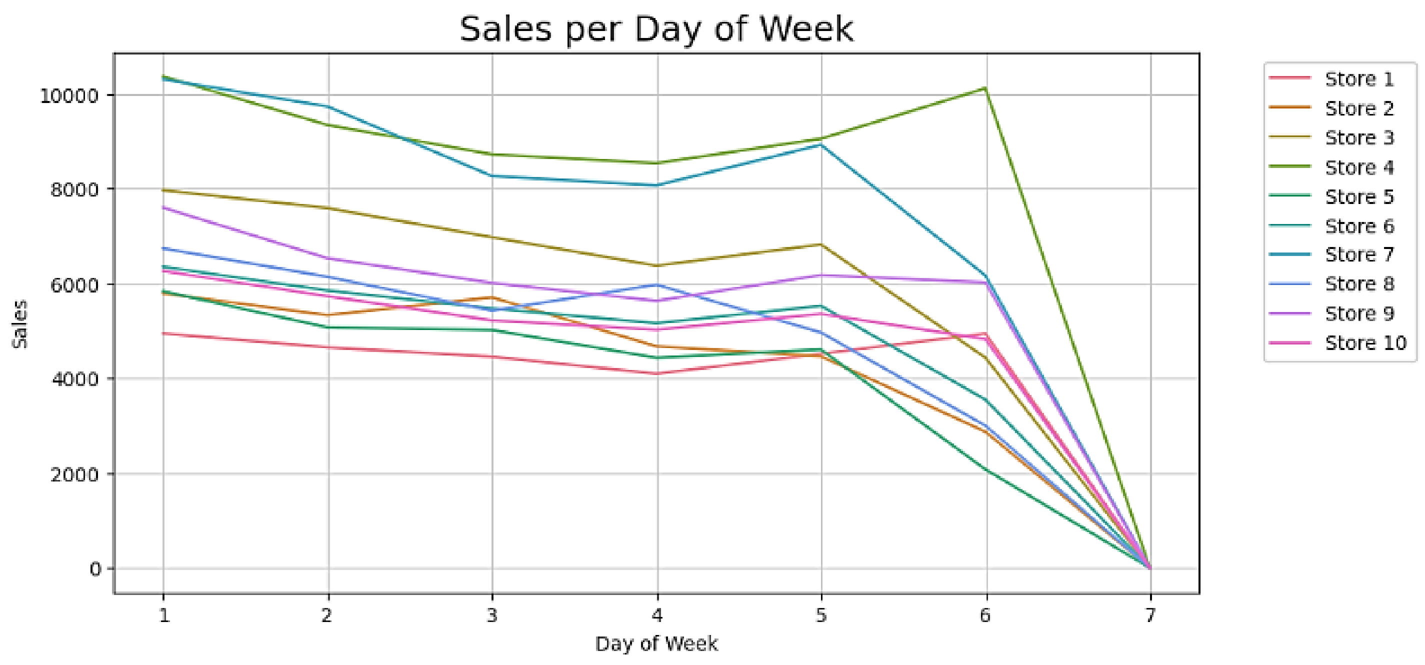


Figure.15 Sales per day of the week

Observation:

The sales were almost negligible on Sunday and maximum on Mondays, this might be because most shops remain closed on Sundays.

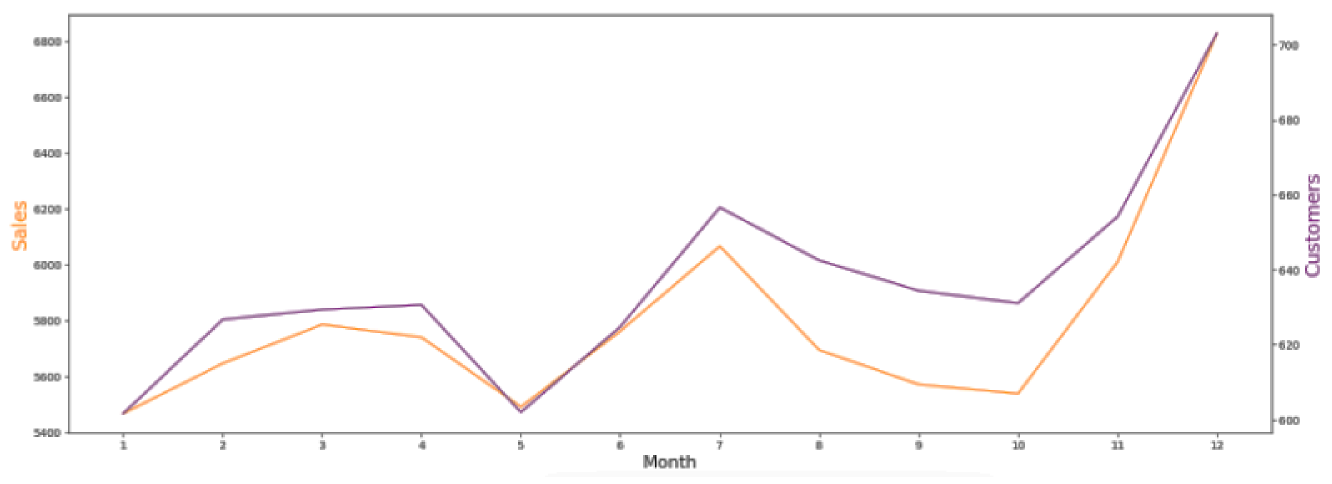


Figure.16 Sales vs Customers distribution per month (Line Graph)

Observation:

The peak festival seasons observe the highest sales in February, March, July, and December. January and May being off seasons have minimum sales. The Sales trend is rising in December due to Christmas.

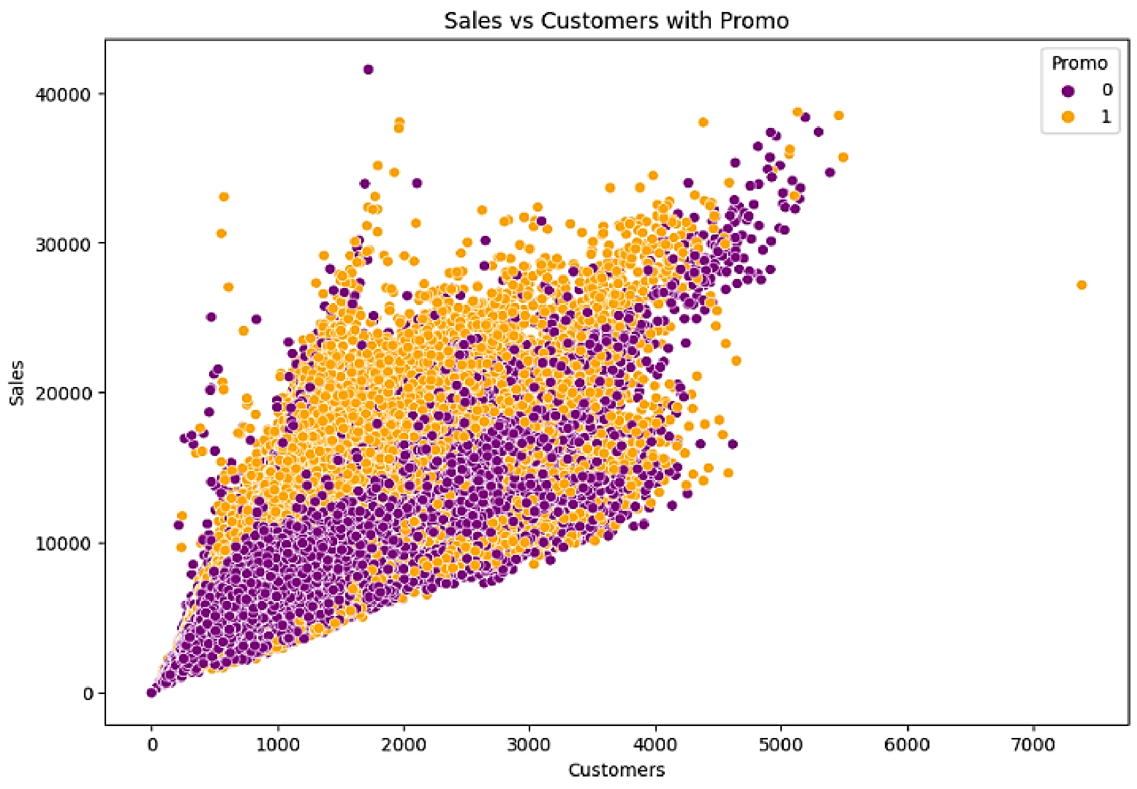


Figure.17 Scatter plot for Sales vs Customers with Promo influence

Observation:

A clear linear correlation exists between customer footfall and Sales particularly during **promo=1** is applied on stores. This indicates a positive impact of promotions on business performance as compared to non-promo periods.

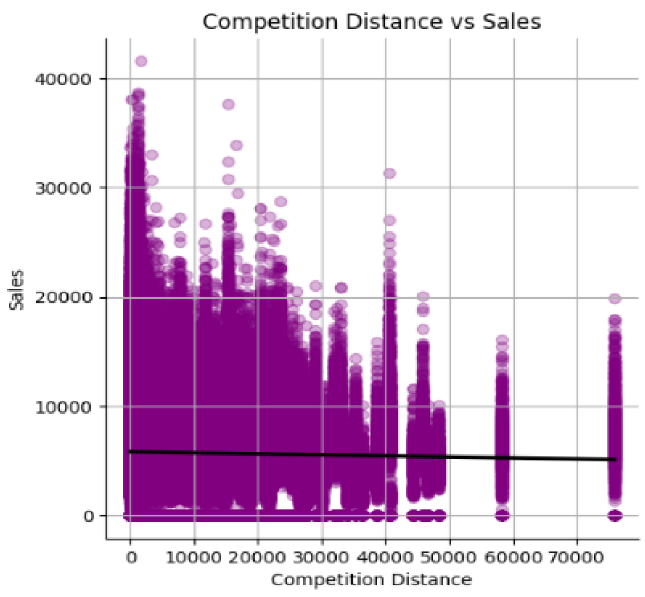


Figure.18 Scatter plot for Sales affected by Competition distance.

Surprisingly the Sales are increasing when the competitor is closer, and most stores are placed very closely.

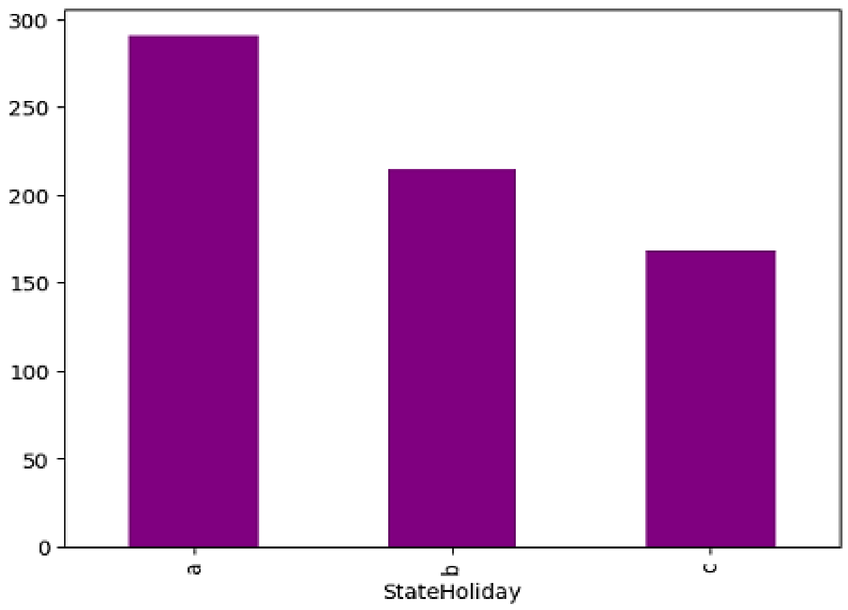


Figure.19 Few sales on State Holidays (Public, Easter,Christmas)

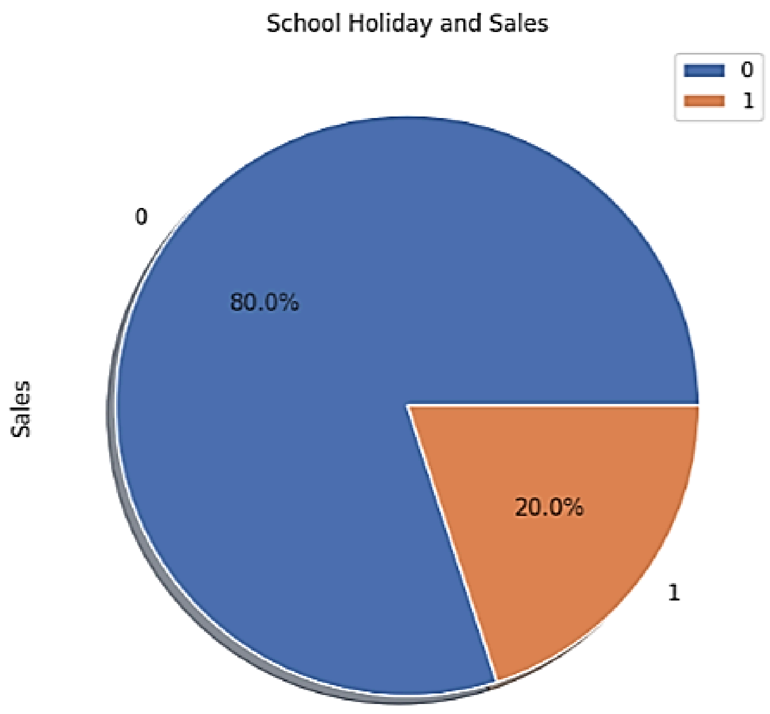


Figure.20 Pie-chart depicting sales during school holidays.

Observation:

- During public holidays stores have made the maximum sales compared to Easter and Christmas.
- School holidays contribute to 20% of overall sales as stores are open during school holidays.

3.2.2 Correlation Heatmap

We figure out the factors affecting **'Sales'** by checking correlation by dropping irrelevant numeric features.

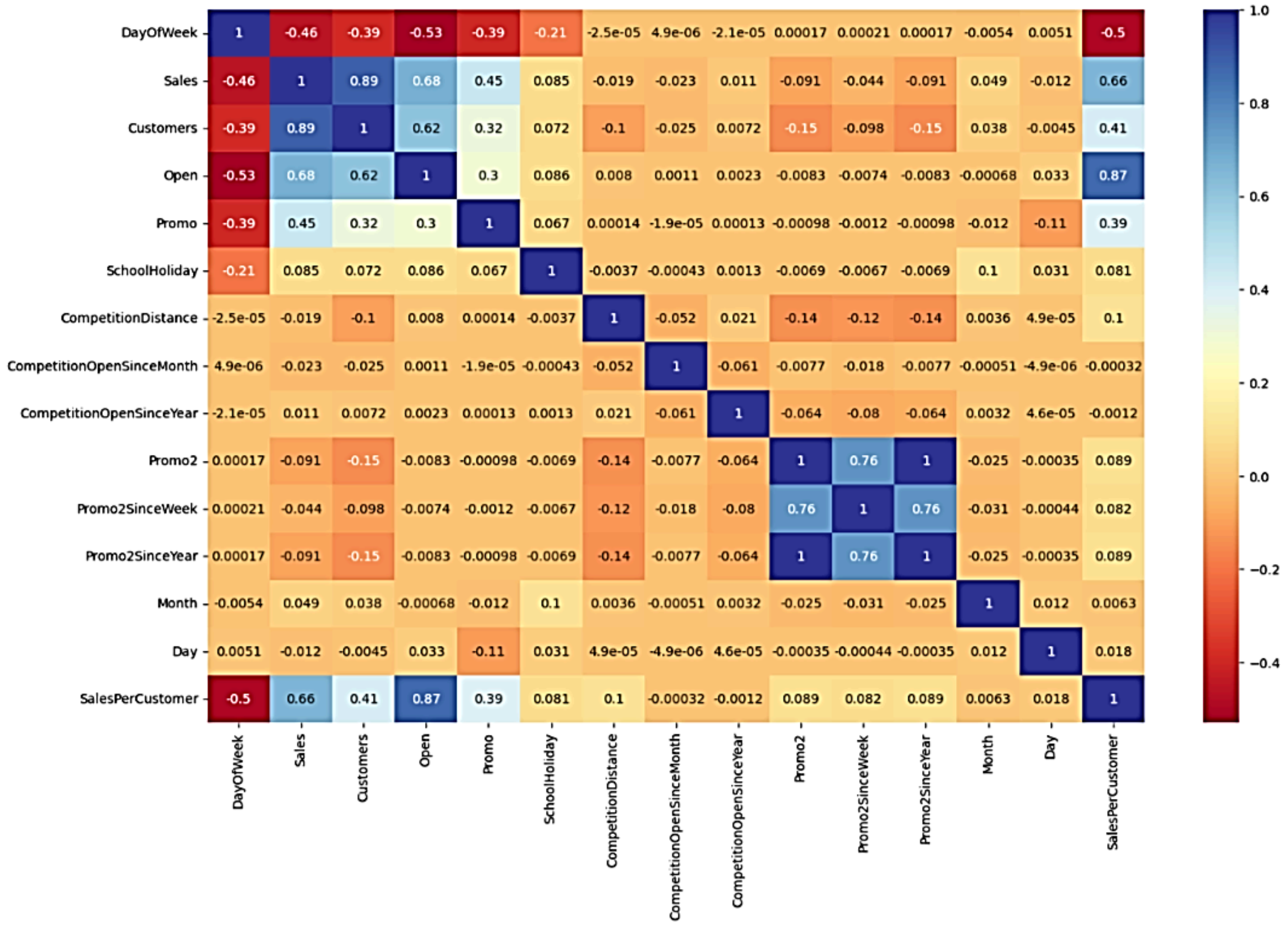


Figure.21 Correlation Matrix using Heatmap.

Observation:

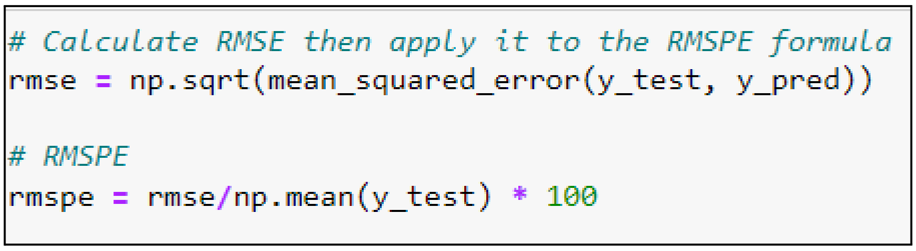
- Features like **Customers,Promo,Open** have a positive correlation with sales leading to increase in sales.
- CompetitionDistance is negatively correlated with sales, higher distance implies lower sales.
- DayOfTheWeek exhibits negative correlation, indicating lower sales on weekends.
- **Multicollinearity** is evident in similar features like Promo2, Promo2SinceWeek and Year.
- In conclusion the correlation values are well justifying our EDA findings.

4. Modelling

4.1 Key Evaluation Metric

Root Mean Square Percentage Error (RMSPE)

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



```
# Calculate RMSE then apply it to the RMSPE formula
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# RMSPE
rmspe = rmse/np.mean(y_test) * 100
```

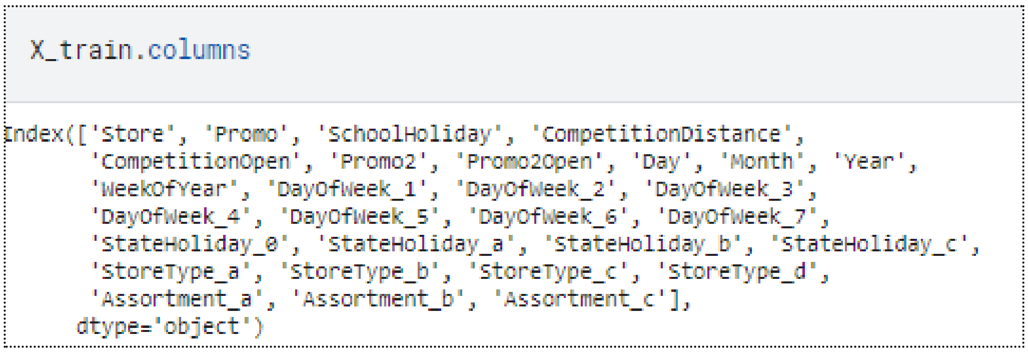
Figure .22 Using RMSE value to estimate RMSPE for sales forecasting evaluation

4.2 Model Selection

We selected XG**Boost(Gradient Boosting)** over Decision Tree(Base model) and Random Forest for its superior performance, regularisation ability and versatility to handle large complex data with outliers. In context of Sales forecasting for over 1000 stores. In presence of outliers, decision trees are prone to overfitting and Random Forest might not effectively mitigate overfitting, but XGBoost's parameters such as max_depth and min_weight, help control model complexity, reduce the influence of outliers and give accurate sales predictions.

The training dataset is divided in 80-20 ratio before model building. Initially, we built 3 distinct models and the model with the best validation score was selected for forecasting on a test dataset.

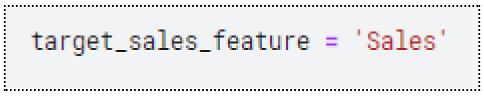
Before modelling, we drop correlated columns like SalesPerCustomer, Customers, PromoInterval. We also use new and encoded features.



```
X_train.columns
```

```
Index(['Store', 'Promo', 'SchoolHoliday', 'CompetitionDistance',
       'CompetitionOpen', 'Promo2', 'Promo2Open', 'Day', 'Month', 'Year',
       'WeekOfYear', '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'],
      dtype='object')
```

Figure.23 Input Features

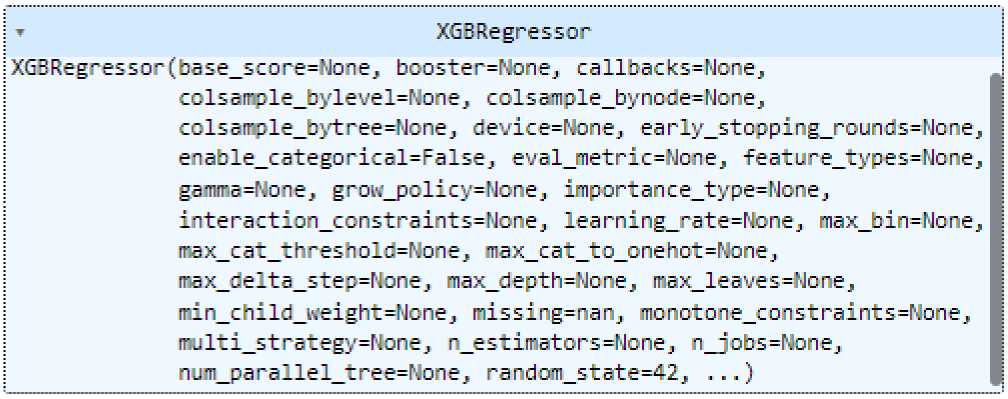


```
target_sales_feature = 'Sales'
```

Figure.24 Target Feature

4.3 Model Fitting

We will split our training dataset consisting of pre-processed input features into a training subset(80%) and validation subset(20%)**.** This dataset will be fitted along with our target variable 'Sales', to assess our model solely using the training data.

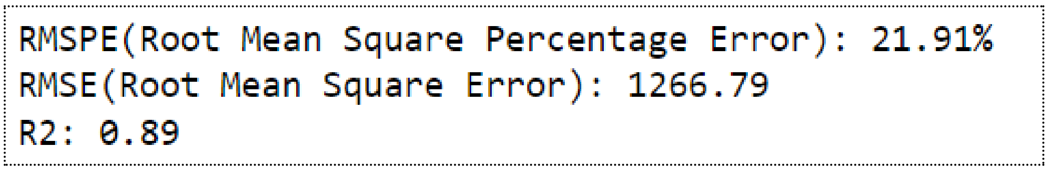


```
▾                              XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, 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=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

Figure.25 Model Fit XGBRegressor with default parameters.

Output:

After applying the XGBoost model with default optimum parameters on the training dataset, Validation **RMSPE is <u>21.91%</u>**



```
RMSPE(Root Mean Square Percentage Error): 21.91%
RMSE(Root Mean Square Error): 1266.79
R2: 0.89
```

Figure.26 Evaluation Metrics for default XGBoost model

4.4 Feature Importance

XGBoost provided us with feature importance scores and based on that we analyse the most commonly used features in sales prediction.

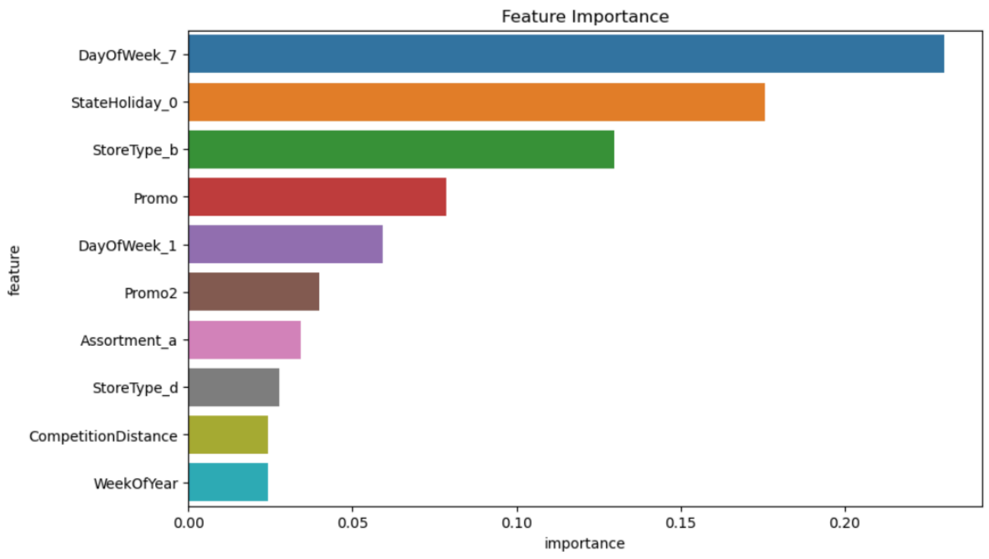
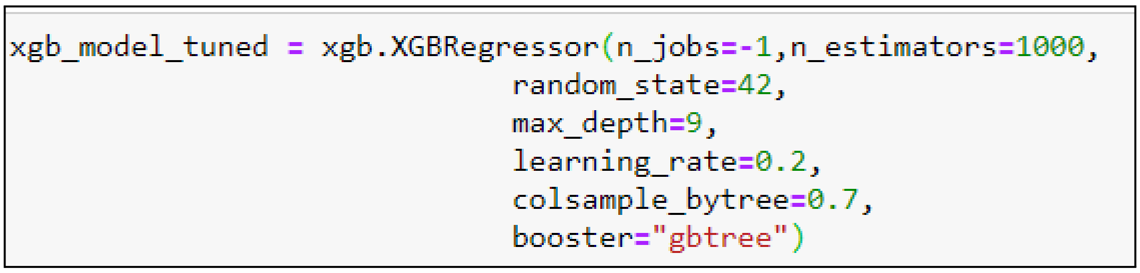


Figure.27 Feature Importance from XGBoost model.

DayofWeek7 & StateHolida_0 are the most valuable features, whereas WeekofYear has very little contribution in the model.

4.5 Model Tuning

We apply Hyper parameter tuning to fine tune the XGBoost parameters for an improved result. The tuned parameters are –

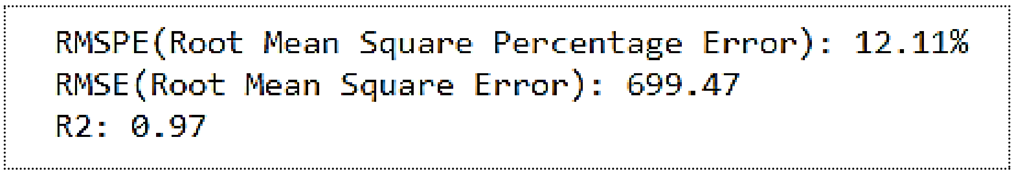


```python
xgb_model_tuned = xgb.XGBRegressor(n_jobs=-1,n_estimators=1000,
                                   random_state=42,
                                   max_depth=9,
                                   learning_rate=0.2,
                                   colsample_bytree=0.7,
                                   booster="gbtree")
```

Figure.28 Tuned Hyperparameters

DayofWeek7 & StateHolida_0 the most valuable feature, whereas WeekofYear has very little contribution in the model.

The Tuned-Model with above parameters performed well with very low **RMSPE of <u>12.11%</u>** on the validation subset.



```
RMSPE(Root Mean Square Percentage Error): 12.11%
RMSE(Root Mean Square Error): 699.47
R2: 0.97
```

Figure.29 Evaluation metrics on tuned model

5. Key Results

Tuned XGBoost model exhibits improved performance, we have forecasted the Sales for **1st-Aug-2015 till 17th-Sept-2015** and visualised the distribution.

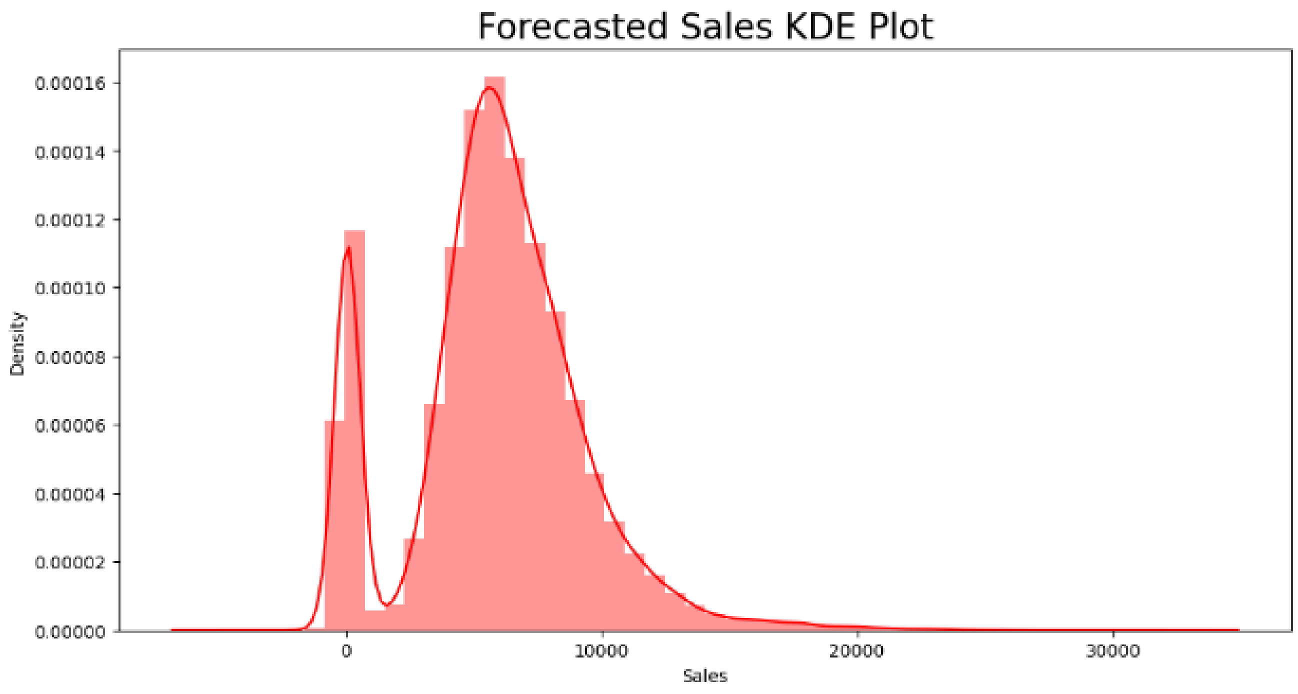


Figure.30 KDE distribution plot for forecasted sales on unseen data

The forecasted sales also follow a similar skewed distribution like our train dataset, which indicates our prediction is closer to the true sales dataset.

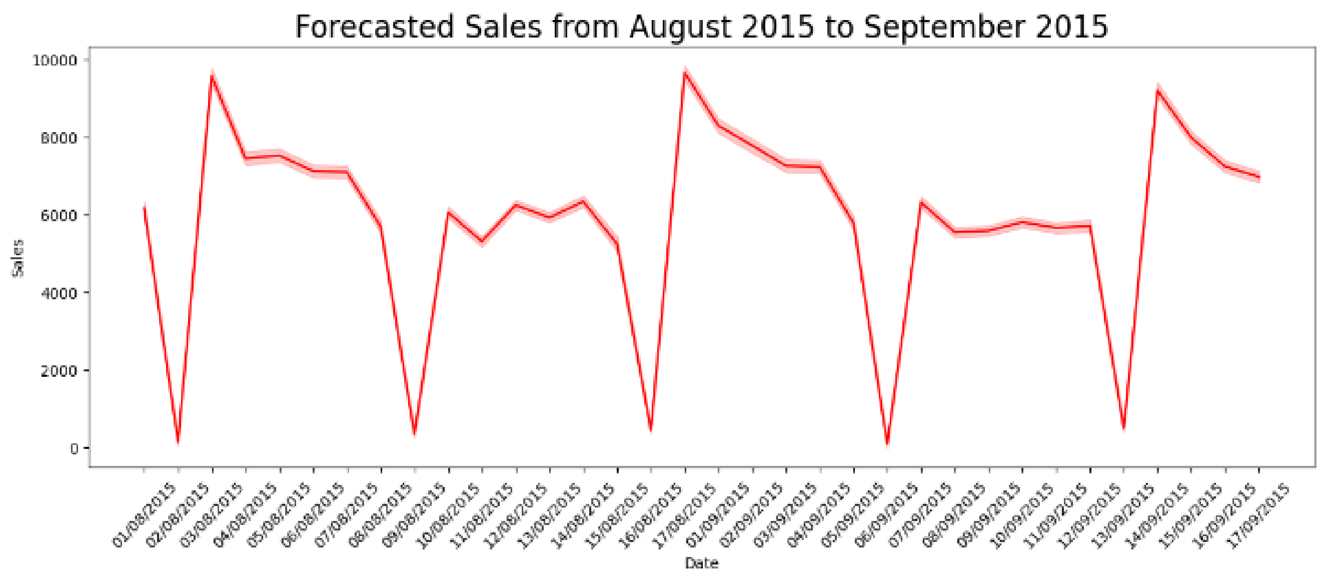


Figure.31 Predicted sales for 6 weeks (Aug,2015 to September,2015)

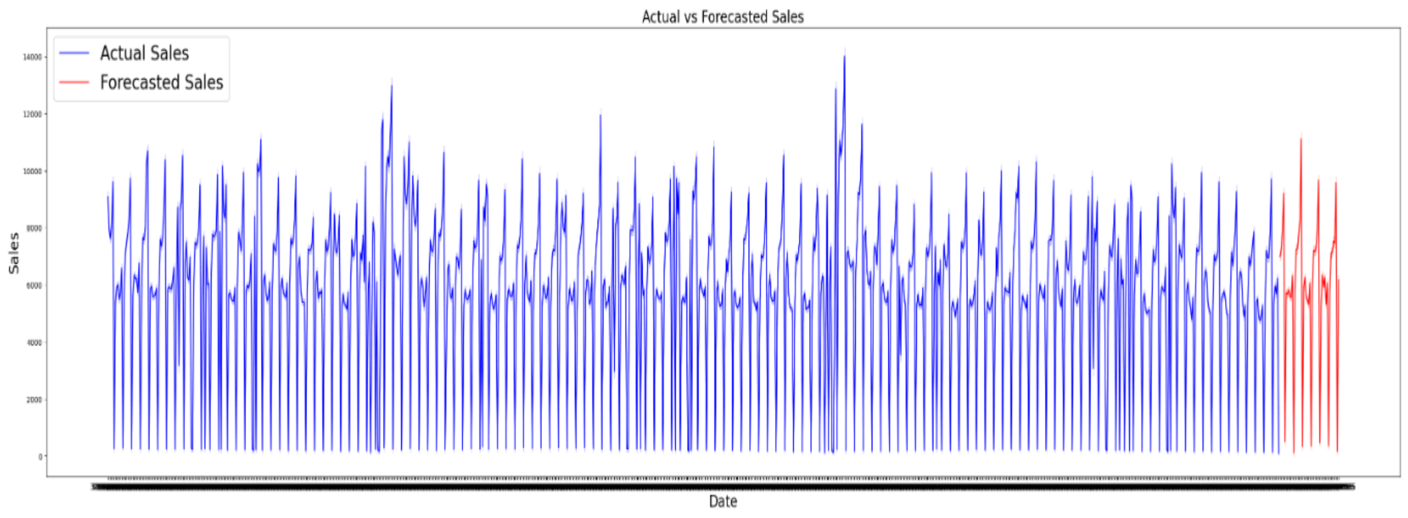


Figure32 Actual Sales (Blue) and forecasted sales(Red)

5. Conclusion

Overall, after extensive data preprocessing including wrangling, feature engineering, scaling and encoding and insights drawn from meaningful EDA, XGBoost emerged as the optimal model for sales forecasting. Its capacity to handle time variables, seasonal fluctuations and mitigating outliers were key factors in the selection process. Through parameter tuning, the model achieved a very low RMSPE of 12.11% signifying enhanced accuracy. XGBoost's ability to give stable predictions with low variance makes it the ideal choice for accurate and efficient sales forecasting in the retail industry.

6. Limitations

- Limited customer insights hinder understanding behaviour.
- Geographic bias due to store location exists.
- Limited information regarding economic factors like inflation rates affecting sales.
- Lack of external information like health crisis, technological shifts.
- Limited historical data, leading to constraint in analysing seasonality.

7. Recommendation

- The Outlier treatment can be done better in future by investigating the outlier patterns and causes by conducting an intensive EDA and using robust regression. We can mitigate the effect of outliers by consulting domain experts.
- Time-series analysis can bring more valuable insights in the future.

References

[1] Andrabi, Syed & Alice, Dr & Srivastava, Siddharth. (2022). Sales Forecasting using XGBoost. 10.36227/techrxiv.21444129.v1.

[2]https://medium.com/@amdeamd7/rossmann-pharmaceutical-sales-prediction-a-deep-learning-approach-9c4434fc809c

[3] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News, 2(3):18–22.

[4] Ghosh, R. & Purkayastha, P. (2017) „Forecasting profitability 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

[5] https://medium.com/analytics-vidhya/rossmann-store-sales-prediction-998161027abf#e216

[6] https://medium.com/analytics-vidhya/rossmann-store-sales-prediction-998161027abf

[7] https://plotly.com/

[8]https://learning.oreilly.com/library/view/hands-on-data-preprocessing/9781801072137/B17397_FM_Final_NM_ePub.xhtml

[9] Ma S, Fildes R (2021) Retail sales forecasting with meta-learning. Eur J Oper Res 288(1):111–128

[10] https://cs229.stanford.edu/proj2015/218_report.pdf