ROSSMAN STORES: A DEEP DIVE INTO SALES FORECASTING CHALLENGES

ID-11349153

INTRODUCTION

In the dynamic landscape of the retail industry, the ability to accurately forecast sales is a critical determinant of success for businesses worldwide. Forecasting sales accurately stands out as one of the most formidable challenges faced by retailers, owing to the multifaceted nature of factors influencing sales patterns. Reliable sales forecasts are paramount for store managers as they strive to enhance overall productivity, increase profitability, and elevate customer satisfaction within the retail domain. In our analysis, we will be dealing with the dataset of a renowned drug store chain operating across Europe especially Germany- ROSSMAN. The company, founded in 1972, is best-known for its brands like Isana (skin, hair and body care), Alterra (natural cosmetics), domol (cleaning and laundry detergents) alouette (paper tissues etc). Rossmann also offers promotional items like pet food, photo service ,perfume and a wide range of natural foods and wines.

Our objective here is to predict 6 weeks of daily sales for their 1,115 drug stores keeping in mind multiple factors affecting sales like promotions, competition, seasonality and locality. For our analysis we have been provided 3 datasets namely stores, train and test and based on stores and train data we need to predict sales on test dataset.

APPROACH

The process of working with a dataset in predictive analytics is not an easy task and involves several steps like data collection, data cleaning, data visualization, model building and finally model evaluation and prediction. We will discuss each step in detail eventually in this report.

DATASETS USED

Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
1	c	a	1270	9	2008	0			
2	a	a	570	11	2007	1	13	2010	Jan,Apr,Jul,Oct
3	a	a	14130	12	2006	1	14	2011	Jan,Apr,Jul,Oct
4	c	c	620	9	2009	0			
5	a	a	29910	4	2015	0			
6	a	a	310	12	2013	0			
7	a	c	24000	4	2013	0			
8	a	a	7520	10	2014	0			
9	a	c	2030	8	2000	0			
10	a	a	3160	9	2009	0			
11	a	c	960	11	2011	1	1	2012	Jan,Apr,Jul,Oct
12	a	c	1070			1	13	2010	Jan,Apr,Jul,Oct
13	d	a	310			1	45	2009	Feb,May,Aug,Nov
14	a	a	1300	3	2014	1	40	2011	Jan,Apr,Jul,Oct
15	d	c	4110	3	2010	1	14	2011	Jan,Apr,Jul,Oct
16	a	c	3270			0			

Figure 1.Store Dataset

This is the table which contains information about the 1115 stores of ROSSMAN. Variables used here are:-

Store: Unique identifier for each store

StoreType: Format of store

Assortment-Product types available in store

CompetitionDistance- Distance to the nearest competitor

CompetitionOpenSinceMonth: The month when the competitor store opened

CompetitionOpenSinceYear: The year when the competitor store opened

Promo2: A binary indicator (0 or 1) that represents whether the store is participating in a continuous promotion (Promo2).

Promo2SinceWeek: The week when Promo2 started.

Promo2SinceYear: The year when Promo2 started.

PromoInterval: Describes the consecutive intervals Promo2 is started, naming the months the promotion is active.

Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1	5	31/07/2015	5263	555	1	1	0	1
2	5	31/07/2015	6064	625	1	1	0	1
3	5	31/07/2015	8314	821	1	1	0	1
4	5	31/07/2015	13995	1498	1	1	0	1
5	5	31/07/2015	4822	559	1	1	0	1
6	5	31/07/2015	5651	589	1	1	0	1
7	5	31/07/2015	15344	1414	1	1	0	1
8	5	31/07/2015	8492	833	1	1	0	1
9	5	31/07/2015	8565	687	1	1	0	1
10	5	31/07/2015	7185	681	1	1	0	1
11	5	31/07/2015	10457	1236	1	1	0	1
12	5	31/07/2015	8959	962	1	1	0	1
13	5	31/07/2015	8821	568	1	1	0	0
14	5	31/07/2015	6544	710	1	1	0	1

Figure 2.Train Dataset

Variables used:

Store: Same as previous

DayOfWeek- Day of the week

Date- Date of sales

Sales- Actual sales made that day. This is the target variable for prediction.

Customers- Number of customers that day

Open- Indicator for store open/closed

Promo- Indicator for presence of promotion that day

StateHoliday: Indicator for state holiday ("0" = no holiday, "a" = public holiday, "b" = Easter holiday, "c" = Christmas).

SchoolHoliday: Indicator for school holiday

Test dataset is similar to train except that it doesn't contain information about customers and sales which we need to predict.

DATA PREPROCESSING

After loading data we merge both the datasets into a new dataframe for ease of analysis.

In [40]:
```
train=train.merge(store,on="Store",how="inner")

train.head()
```

Out[40]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	CompetitionOp
0	1	5	31/07/2015	5263	555	1	1	0	1	c	a	1270.0	
1	1	4	30/07/2015	5020	546	1	1	0	1	c	a	1270.0	
2	1	3	29/07/2015	4782	523	1	1	0	1	c	a	1270.0	
3	1	2	28/07/2015	5011	560	1	1	0	1	c	a	1270.0	
4	1	1	27/07/2015	6102	612	1	1	0	1	c	a	1270.0	

Figure 3.Merged Data

NULL VALUES

We have null values only in the features -CompetitionOpenSinceMonth,CompetitionDistance,CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear and PromoInterval.

In [88]:
```
train.isnull().sum()
```

Out[88]:
```
Store                             0
DayOfWeek                         0
Date                              0
Sales                             0
Customers                         0
Open                              0
Promo                             0
StateHoliday                      0
SchoolHoliday                     0
StoreType                         0
Assortment                        0
CompetitionDistance            2642
CompetitionOpenSinceMonth    323348
CompetitionOpenSinceYear     323348
Promo2                            0
Promo2SinceWeek              508031
Promo2SinceYear              508031
PromoInterval                508031
dtype: int64
```

Figure 4.Null value count

Having null values in our dataset would impact the predictions badly, so we need to deal with them by either imputing or removing the row. In this case removing data would lead to loss o f important features affecting sales so we will be imputing those values.

CompetitionDistance- This can be imputed by median value. Filling 0 here would mean there is no competitor present which might not be true and we are not using mean here because it can get impacted by outlier easily.

Features like Promo2SinceWeek and CompetitionOpenSinceMonth having null values sugg ests there is no competitor or promotion applicable, so here we can impute with 0.

```
In [47]: train["CompetitionDistance"].fillna(train["CompetitionDistance"].median(), inplace = True)


         train["CompetitionOpenSinceMonth"].fillna(0, inplace = True)
         train["CompetitionOpenSinceYear"].fillna(0, inplace = True)


         train["Promo2SinceWeek"].fillna(0, inplace = True)
         train["Promo2SinceYear"].fillna(0, inplace = True)
         train["PromoInterval"].fillna(0, inplace = True)

         store.info()

         <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 5.Data after imputation

DUPLICATE VALUES

Next, we drop the duplicate values from the dataset so that there is no redundancy which can lead to data inconsistency adversely impacting model predictions.

OUTLIERS

Treating outliers is important in data analysis otherwise it would lead to skewness towards certain data points leading to biased predictions. To identify outliers here we will be using boxplots.

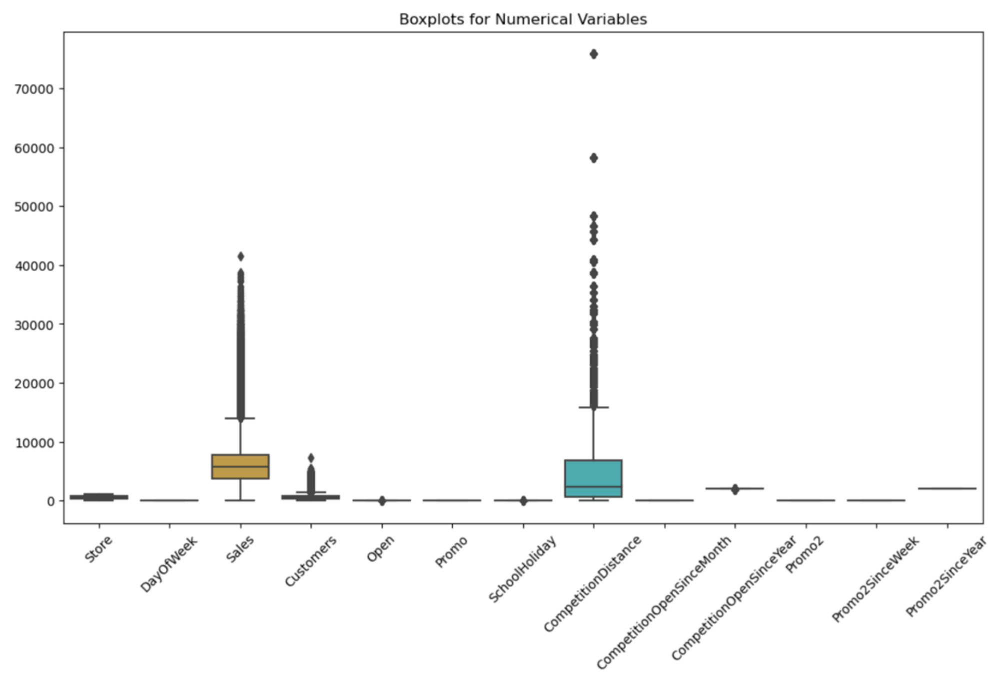

Figure 6.Boxplot detecting outlier

Now, to remove outliers we need to define the quantiles and exclude data which falls beyond that bound.

In [43]:
```
#REMOVING OUTLIERS

Q1 = train['Sales'].quantile(0.25)
Q3 = train['Sales'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = train[(train['Sales'] < lower_bound) | (train['Sales'] > upper_bound)]
print("Outliers in Sales:")
print(outliers[['Store', 'Date', 'Sales']])
```

```
Outliers in Sales:
         Store        Date  Sales
2280         3  30/06/2014  15689
2469         3  23/12/2013  14461
2475         3  17/12/2013  14555
2476         3  16/12/2013  14647
2918         4  30/04/2015  16106
...        ...         ...    ...
1016260   1114  07/01/2013  21237
1016262   1114  05/01/2013  18856
1016263   1114  04/01/2013  18371
1016264   1114  03/01/2013  18463
1016265   1114  02/01/2013  20642

[26694 rows x 3 columns]
```

Figure 7.Removing Outliers

BOXPLOT AFTER REMOVING OUTLIERS

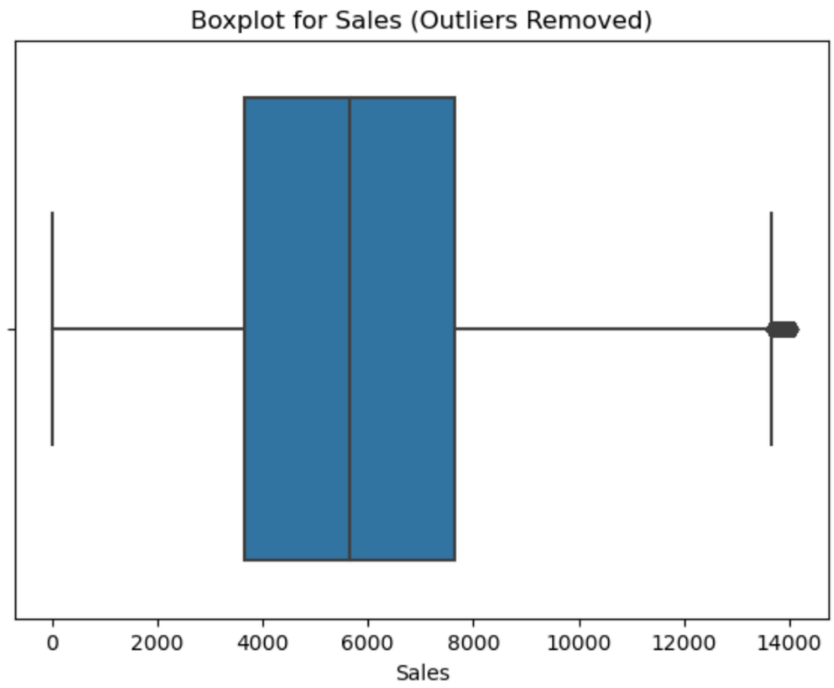

Figure 8

MISSING VALUES

Missing value means incomplete data so we have to impute values in this case as well to ensure data completeness and consistency. This is the current percentage of missing values in the dataset.

```
Store                          0.000000
StoreType                      0.000000
Assortment                     0.000000
CompetitionDistance            0.269058
CompetitionOpenSinceMonth     31.748879
CompetitionOpenSinceYear      31.748879
Promo2                         0.000000
Promo2SinceWeek               48.789238
Promo2SinceYear               48.789238
PromoInterval                 48.789238
dtype: float64
```

Figure 9.Percentage of Missing Values

After going through all the stages of data cleaning, this is how our train dataset look currently.

```
<class 'pandas.core.frame.DataFrame'>
Index: 990515 entries, 0 to 1017208
Data columns (total 18 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Store                      990515 non-null  int64
 1   DayOfWeek                  990515 non-null  int64
 2   Date                       990515 non-null  object
 3   Sales                      990515 non-null  int64
 4   Customers                  990515 non-null  int64
 5   Open                       990515 non-null  int64
 6   Promo                      990515 non-null  int64
 7   StateHoliday               990515 non-null  object
 8   SchoolHoliday              990515 non-null  int64
 9   StoreType                  990515 non-null  object
 10  Assortment                 990515 non-null  object
 11  CompetitionDistance        990515 non-null  float64
 12  CompetitionOpenSinceMonth  990515 non-null  float64
 13  CompetitionOpenSinceYear   990515 non-null  float64
 14  Promo2                     990515 non-null  int64
 15  Promo2SinceWeek            990515 non-null  float64
 16  Promo2SinceYear            990515 non-null  float64
 17  PromoInterval              990515 non-null  object
dtypes: float64(5), int64(8), object(5)
memory usage: 143.6+ MB
```

Figure 10.Cleaned Data

So, as we can see there are no NULL values in the dataset, hence we can proceed to the next stage which is feature engineering.

FEATURE ENGINEERING

This section is all about modifying the feature variables in such a way so that it helps in clustering similar data and thus ease the analysis process. It also helps in generating valuable insights which we might not be able to identify other way round. Key trends, locality and seasonality are some of them.

But to detect such patterns first we need to format the date column and disseminate it into day, month, week, year so that we can work with seasons & also identify consumer buying patterns like on which day of the week customers have purchased the most and vice versa.

Similarly, we can also identify the same for Holiday season like during which holiday season customer prefers to buy more and also we can check what impact promotions can have on sales.

Seasonality

```
train["Season"]=np.where(train["Month"].isin([3,4,5]),"Spring",
                                    np.where(train["Month"].isin([6,7,8]),"Summer",
                                             np.where(train["Month"].isin([9,10,11]),"Autumn",
                                                      np.where(train["Month"].isin([12,1,2]),"Winter","None"))))
```

Figure 11.Adding new feature

Here we are introducing a new variable “seasons” by clustering months as per seasons.

	Date	Day	Week	Month	Year	Season
0	2015-07-31	31	31	7	2015	Summer
1	2015-07-30	30	31	7	2015	Summer
2	2015-07-29	29	31	7	2015	Summer
3	2015-07-28	28	31	7	2015	Summer
4	2015-07-27	27	31	7	2015	Summer
5	2015-07-26	26	30	7	2015	Summer
6	2015-07-25	25	30	7	2015	Summer
7	2015-07-24	24	30	7	2015	Summer
8	2015-07-23	23	30	7	2015	Summer
9	2015-07-22	22	30	7	2015	Summer
10	2015-07-21	21	30	7	2015	Summer
11	2015-07-20	20	30	7	2015	Summer
12	2015-07-19	19	29	7	2015	Summer
13	2015-07-18	18	29	7	2015	Summer
14	2015-07-17	17	29	7	2015	Summer
15	2015-07-16	16	29	7	2015	Summer
16	2015-07-15	15	29	7	2015	Summer
17	2015-07-14	14	29	7	2015	Summer
18	2015-07-13	13	29	7	2015	Summer

Figure 12

Alternatively, we can also group stores based on sales level or sales statistics for better understanding.

```
def sales_stats(df, df2):

    df2['SalesMean'] = df.groupby('Store')['Sales'].transform('mean')
    df2['SalesStd'] = df.groupby('Store')['Sales'].transform('std')
    return df2

def sale_level(df2):

    # Define quartiles
    Q1 = df2['SalesMean'].quantile(0.25)
    Q2 = df2['SalesMean'].quantile(0.50)
    Q3 = df2['SalesMean'].quantile(0.75)

    # Using bins to create cluster
    bins = [float('-inf'), Q1, Q2, Q3, float('inf')]
    labels = [1, 2, 3, 4]
    df2['StoreGroup'] = pd.cut(df2['SalesMean'], bins=bins, labels=labels, include_lowest=True)

    return df2

store = sales_stats(train1, store.copy())
store_df2 = sale_level(store)
```

Figure 13 Store Clustering

Here we have classified the store groups into 4 categories based on quantile limits.

We could also see that there are multiple columns conveying similar info,like,CompetitonSinceYear, CompetitionSinceMonth as well as PromotionSinceYear, PromotionSinceWeek which are all date related info. So, here we have decided to create 2 new features Promotion and Competition and kept only the duration of months rather than mentioning year and month separately.

```
train["CompetitionOpen"]=12*(train["Year"]-train["CompetitionOpenSinceYear"])+(train["Month"]-train["CompetitionOpen

train["PromoOpen"]=12*(train["Year"]-train["Promo2SinceYear"])+(train["Week"]-train["Promo2SinceWeek"])/4.0
```

Figure 14.New feature variable

Doing this will help us reduce unnecessary data thus refining data for model fitting. The last step before moving towards the model fitting stage will be to identify the categorical and numerical columns in the data and apply encoding to categorical data and thereafter scale and standardize the entire dataset.

```
num=["Customers","CompetitionDistance","CompetitionOpen","PromoOpen"]
cat=["DayOfWeek","StateHoliday","SchoolHoliday", "StoreType","Assortment","Open","Promo","Promo2","Week","Month","Ye
```

Figure 15.Numerical & Categorical columns

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split,cross_val_score,KFold,GridSearchCV,cross_validate

# encoding
train[cat]=train[cat].astype("object")
le=LabelEncoder()
train.update(train[cat].apply(le.fit_transform))
train.head()
```

Figure 16.Encoding features

Now, we will be removing the sales column since we do not need that in the training data as it is our target variable.

```
col=train.columns.tolist()
col.remove("Sales")
col
# Standardization:
scaler = StandardScaler()
train[col]=scaler.fit_transform(train[col])
train.head()
```

Figure 17.Standardization

DATA AFTER STANDARDIZATION

	Store	DayOfWeek	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance	Promo2	Promo
0	-1.724753	0.487173	5263	-0.105586	0.459724	1.298114	-0.161019	2.153673	0.570338	-0.934203	-0.544301	-1.013462	-0.
1	-1.724753	-0.014712	5020	-0.128397	0.459724	1.298114	-0.161019	2.153673	0.570338	-0.934203	-0.544301	-1.013462	-0.
2	-1.724753	-0.516598	4782	-0.186691	0.459724	1.298114	-0.161019	2.153673	0.570338	-0.934203	-0.544301	-1.013462	-0.
3	-1.724753	-1.018483	5011	-0.092913	0.459724	1.298114	-0.161019	2.153673	0.570338	-0.934203	-0.544301	-1.013462	-0.
4	-1.724753	-1.520368	6102	0.038883	0.459724	1.298114	-0.161019	2.153673	0.570338	-0.934203	-0.544301	-1.013462	-0.

Figure 18.Final dataset

EXPLORATORY DATA ANALYSIS

This section is about exploring various features present in the dataset, their relations and identifying useful patterns. Let’s look at some of the visualizations below

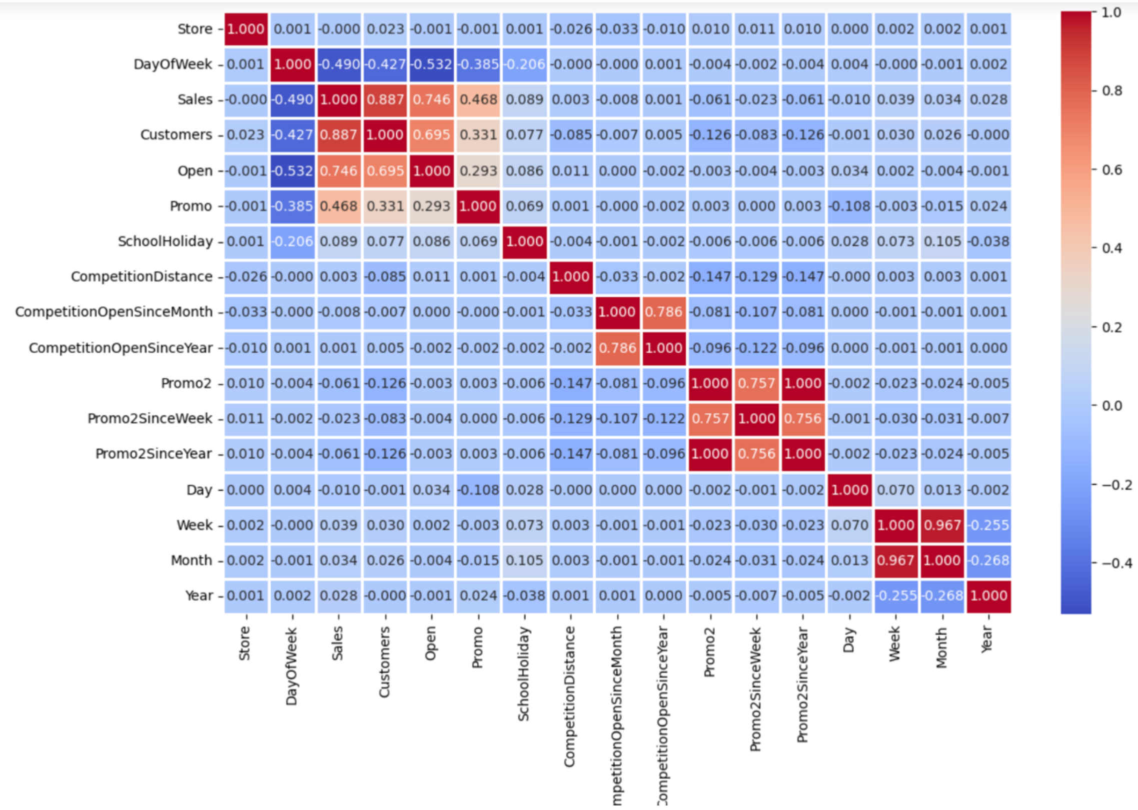

Figure 19.Correlation Heatmap

i

From this heatmap we can understand that Open, customers and promotions mostly impact on sales.

SALES TRENDS

MONTHLY

Coming to sales, we can see similar trends in sales for all 3 years from January to July with a significant spike around year end because of holiday season.

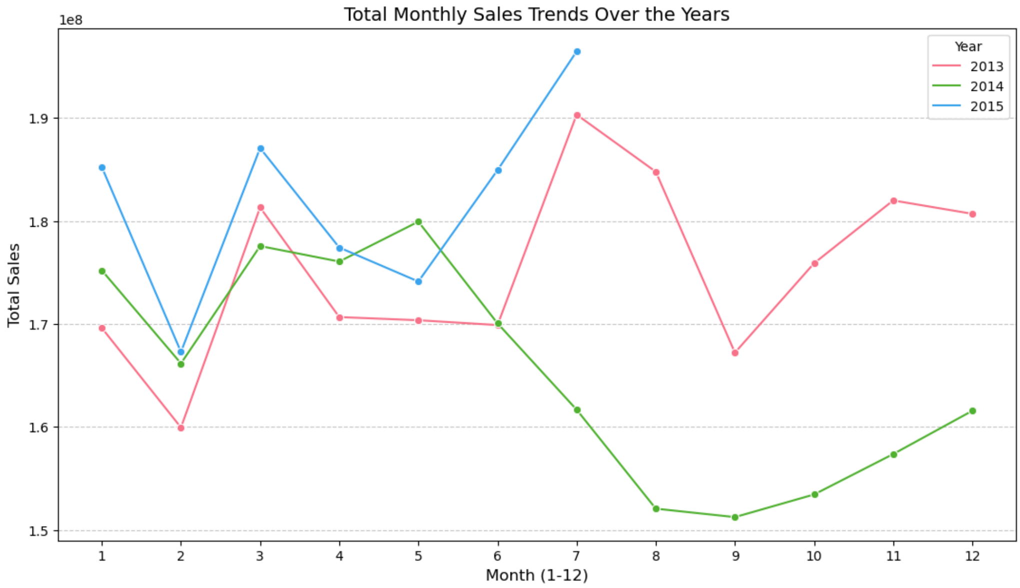

Figure 20

HOLIDAY

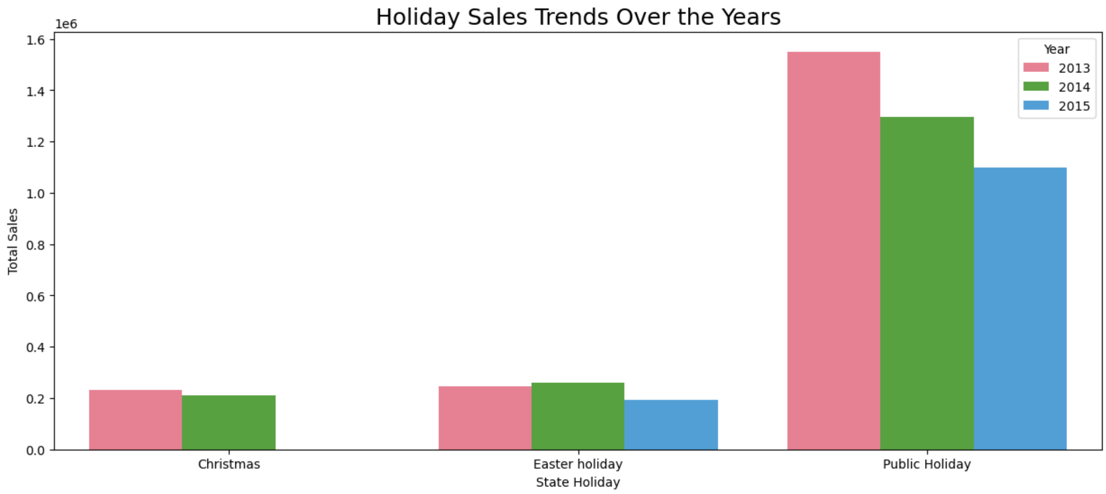

Figure 21

In this plot we can observe high sales pattern during public holidays throughout all three years with an unusual gradual dip in 2015 because we don't have data for the entire year.

DAILY

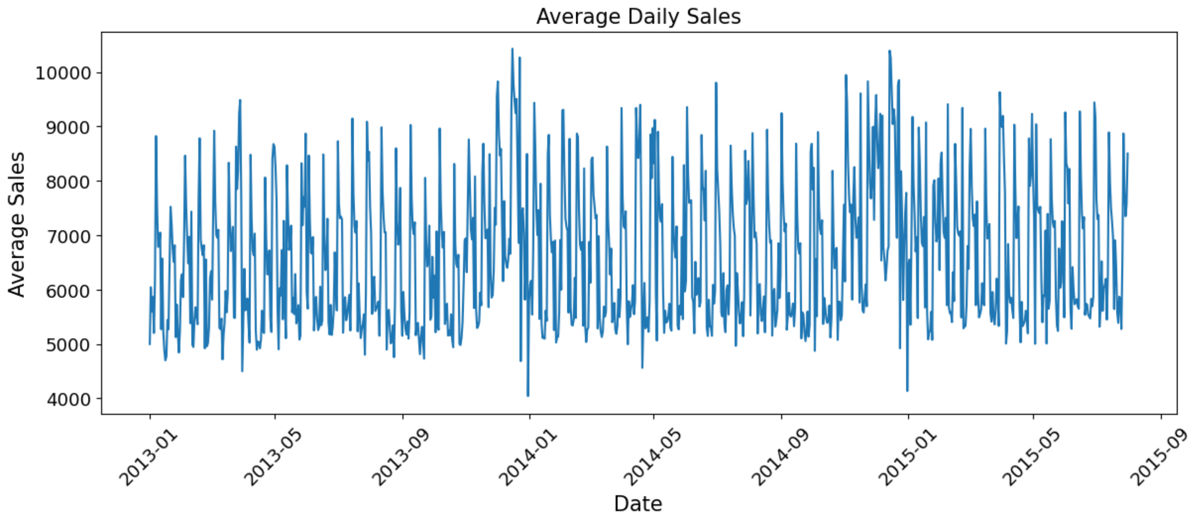

Figure 22

From daily sales, we can identify periodic spikes in sales indicating seasonality, which can be used to create sales optimization strategies.

PROMOTIONS

DISTRIBUTION OF PROMOTION INTERVALS

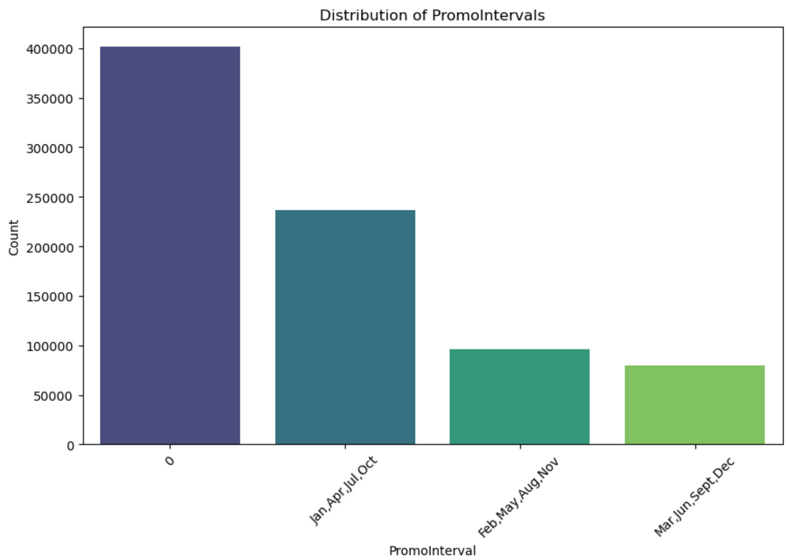

Figure 23

By this plot, we can infer that most of the promotional campaigns have taken place during January ,April, July and October to increase sales. Now, let's look at the plot below to see if the strategies were successful.

IMPACT OF PROMOTION ON SALES

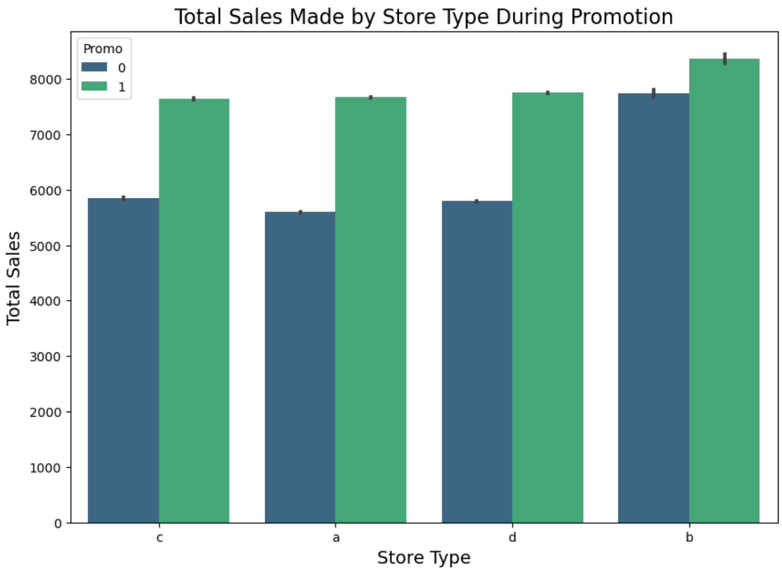

Figure 24

From this plot, one thing is evident that promotion has a strong impact on sales of all the stores but store type "b" have benefitted the most..

PROMOTIONS VS DAY OF WEEK

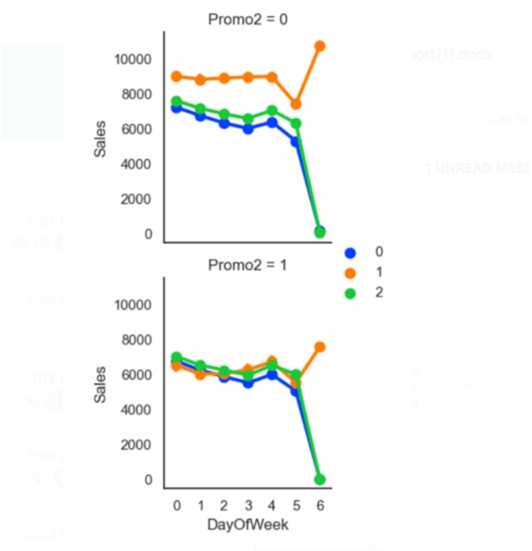

Figure 25

From this figure we can identify that type b products are bestselling and its sales remain intact even on Sundays while others doesn’t.

CUSTOMERS BY STORE TYPE

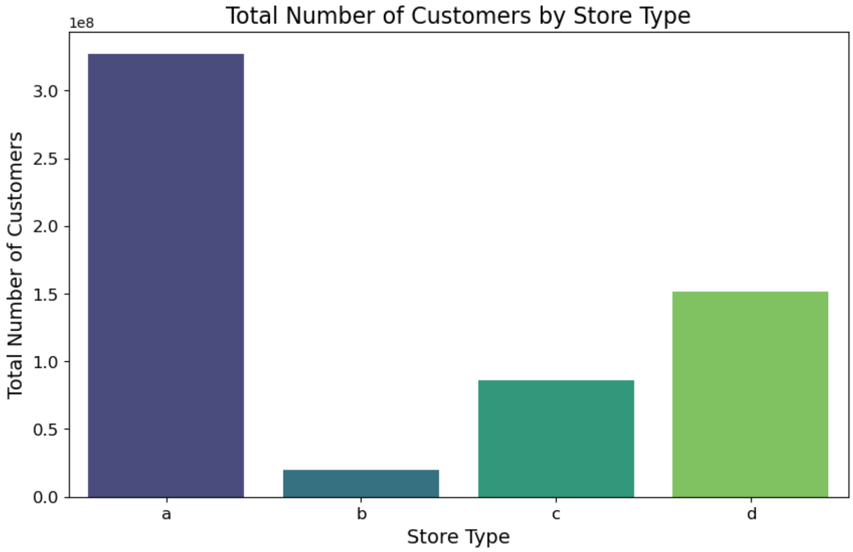

Figure 26

Previously, we have seen that store b had the highest sales because of promotions. This is most probably because they have less customers, hence they offered heavy promotions to improve sales and customer engagement

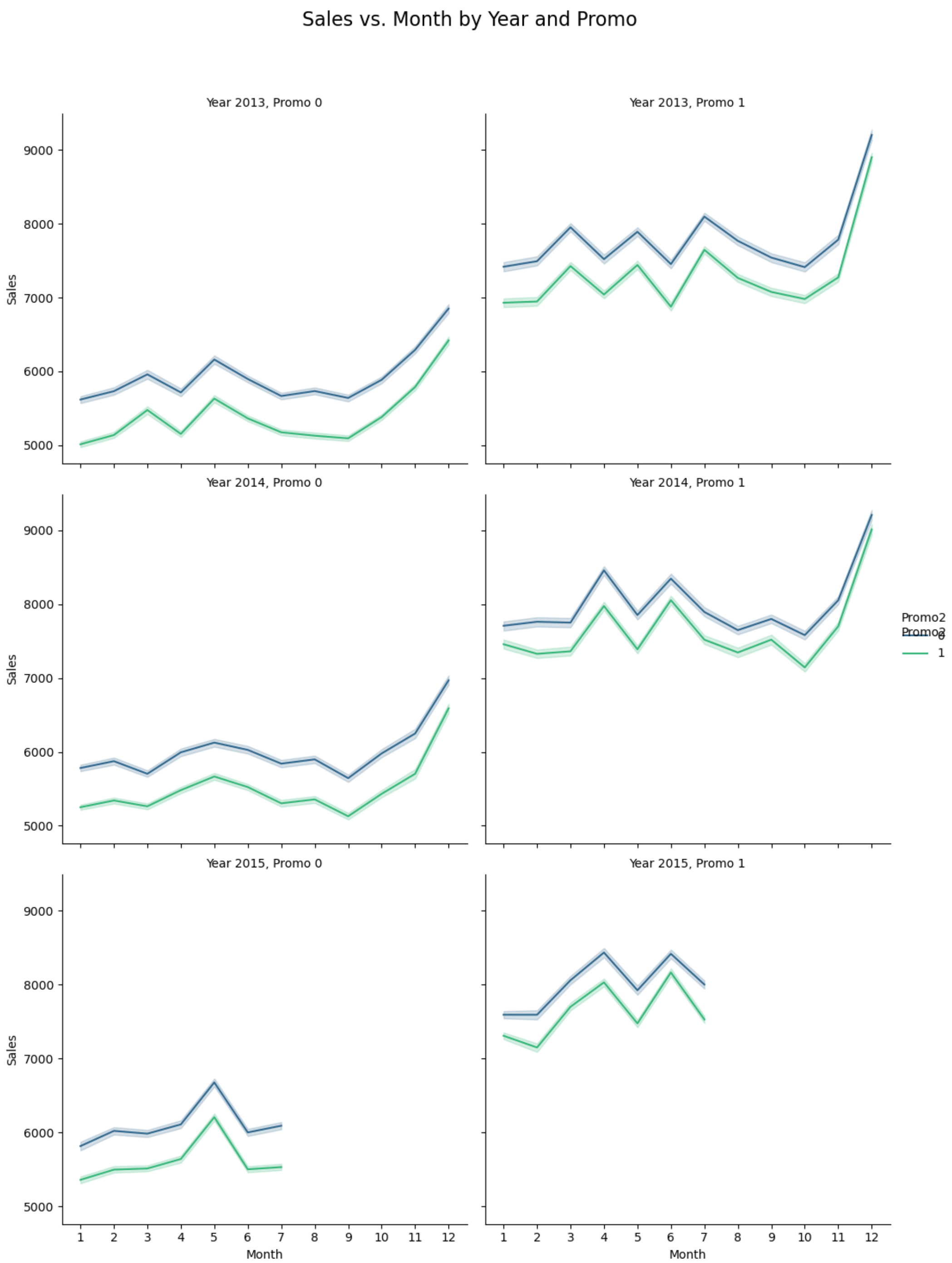

Figure 27

This plot also shows a consistent trend in sales from January to September and then gradually increase in the last 4 months but with promotions, sales have improved.

MODEL DESIGN & RESULTS

In this last step we need to build a machine learning model to predict the outcome that is sales from August till 17th September. Here we have used the random forest regressor model for prediction because it is quite versatile in terms of working with classification and regression problems as well as it highly emphasizes on features used in analysis, which is really crucial for time series sales forecasting. So, in this case we didn't have to split the data as we already had the test dataset. We have fitted those data in the model and used optimized hyperparamaters like "max depth", "max features" and "n_estimators". Next up we have calculated the negative mse and R score for our model using the train and test datasets.

To better the accuracy, we can explicitly fine tune the hyperparameters as well. Post tuning, we again fit the data and make predictions. Final outcome:-

```
R2 for RF: 0.9543533296019707
Root Mean Squared Error for RF: 719.9074196499485
```

The graph below justifies the model accuracy.

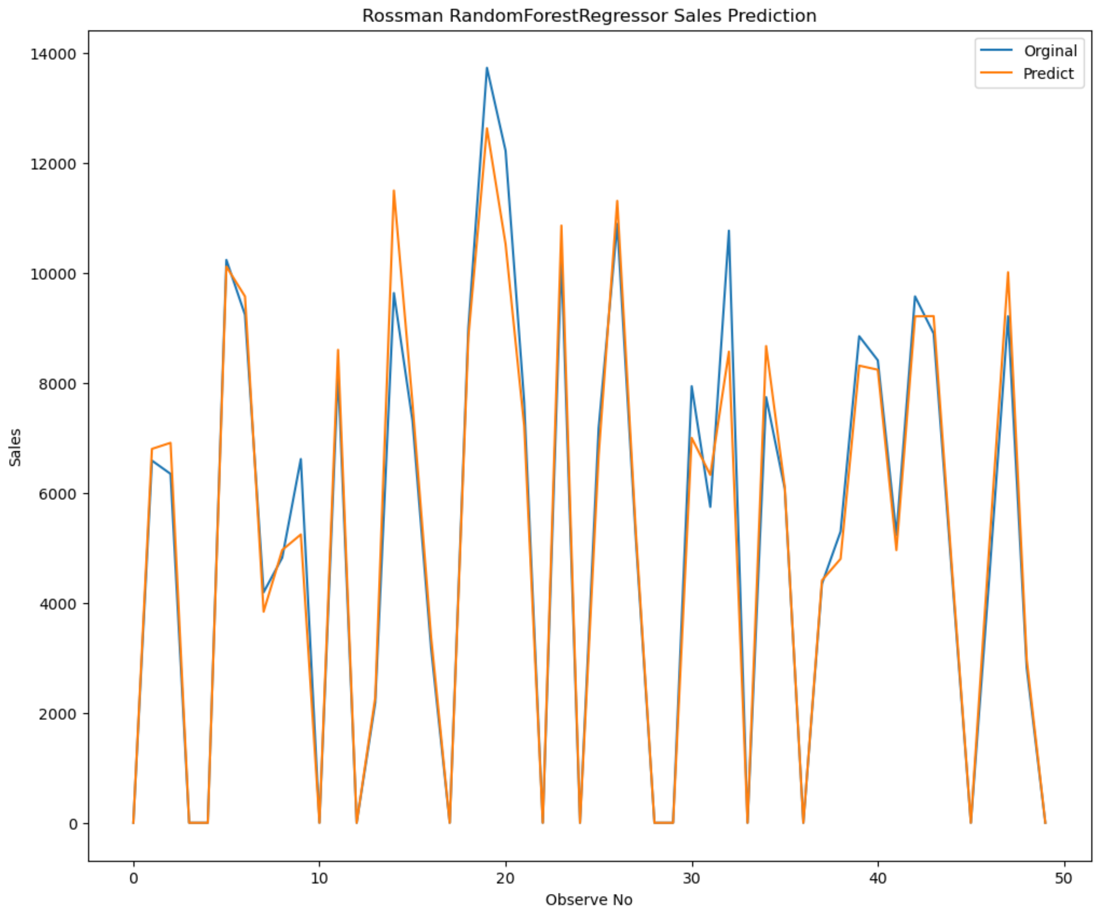

Figure 28

CONCLUSION

As a concluding remark, we can say that this comprehensive prediction analysis can be quite useful for any retail firm to draw insights from their current sales and accordingly plan and implement future business strategies to enhance their growth. Also having factors like competition, seasonality can help them in deciding on campaigns in an effective way to succeed against competitors. Recommendations on this would be to periodically try different parameters in the model and identify the best sales approach to target customers in this ever changing market dynamics.