# UDATE REPORT

**STUDENT ID: 11340063**

**WORD COUNT: 1648**

This coursework explores sales forecasting in the retail industry, focusing on Rossmann, a German drug store chain. The goal is to identify potential in raw data by navigating historical sales data analysis. The data is preprocessed, analysed, and used to understand the retail store's operations, enabling future sales forecasting.

## Dataset Description:

This coursework uses three datasets: Store, Train, and Test. The Store dataset contains information on 1115 Rossmann drug stores in Germany, including store type, assortment, proximity to competitors, and promotions. StoreType and Assortment are categorical data, representing store models and assortment levels. CompetitionDistance, CompetitionOpenSinceMonth and CompetitionOpenSinceYear are numerical data representing the nearest competitor store's distance, opening date and time. Promo2 field indicates if the store is engaged in ongoing promotions, with 1 denoting participation. Promo2SinceWeek and Promo2SinceYear show the week and year the store started participating in promotions. PromoInterval indicates the months the promotion has started anew. Stores that do not run promotions have the Promo2 field set to null, which also sets the fields Promo2SinceWeek, Promo2SinceYear, and PromoInterval to null.

The train dataset is a historical sales data set from 01/01/2013 to 31/07/2015, containing details like customer numbers, store-specific promotions, sales turnover, and any holidays. It contains 10,17209 records of sales. The store field stores the anonymized number assigned to a store, while the DayOfWeek and Date variables indicate the day of the week and recorded sales date. The sales field provides the sales turnover, while the customers field describes the number of customers. The Promo field indicates if the store is running a store-specific promotion on that day. StateHoliday and SchoolHoliday variables describe if the store is affected by any kind of holidays on the given day. The test dataset is identical to the train dataset, except that sales and number of customers are unknown for the period of 01/08/2015 to 17/09/2015.

## Preprocessing:

Effective data preprocessing is crucial for preparing historical sales data for reliable forecasting models. This involves addressing inherent difficulties during data review. In the store dataset,

categorical variables like StoreType and Assortment are encoded using the Label Encoder, assigning each category an integer starting from 0. This helps in analysing data based on categories, providing crucial information about the store type. Null values of PromoInterval are filled with 0 to indicate if it is not present, which was then categorised using Label Encoder. The StateHoliday field is also categorised using the Label Encoder, and the same operation is done for the train and test datasets.

The operations preprocessed each dataset individually, but merging them can aid in analysis and forecasting future sales. The store dataset is merged with the train and test datasets to form merged_train and merged_test.

On analysing the merged datasets, it was found that the merged_train dataset had 52 records where the variable Open was equal to 1 but the number of customers was equal to 0. This would mean that 52 records of sales had a situation where the store was open but no customers had visited that given day, which is highly unlikely and this brings attention to the fact that the customer values might be missing completely at random. Since these 52 records are very small in number when comparing to the entire dataset, these rows are dropped from the dataset. The Date field on its own would not be useful while fitting the dataset into a model. Hence the Date is split into 4 different variables each describing week, day, month and year.

The fields CompetitionOpenSinceMonth and CompetitionOpenSinceYear have missing values. When the CompetitionDistance is not null, these fields have a value of null. This would mean that these fields are missing completely at random. Thus they have to be imputed with an appropriate value. Using a KNN imputation method is not accurate in this case as it results in decimal values in the CompetitionOpenSinceMonth and CompetitionOpenSinceYear fields. To understand more about these fields, a Histogram and KDE graph is plotted.

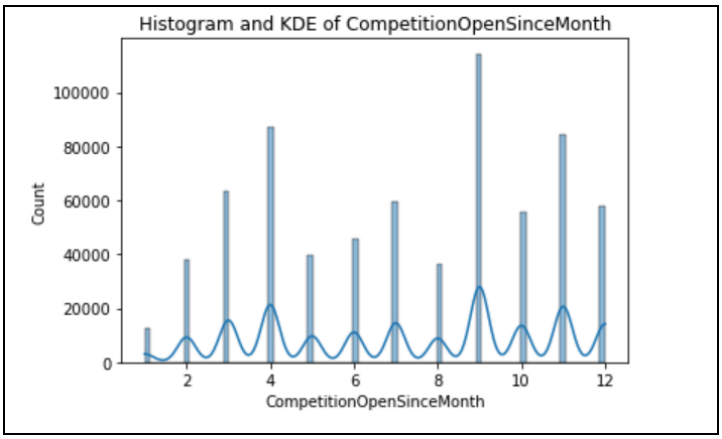



*Figure 1: Histogram and Kernel Density Estimate Graph for describing CompetitonOpenSinceMonth field*

From Figure 1 it is found that CompetitionOpenSinceMonth is not normally distributed. Because it is multimodal, it acts like a categorical variable, and thus the field should be imputed with mode values based on the store types. From Figure 2 it is found that the CompetitionOpenSinceYear is also not normally distributed but is left skewed. Hence imputing this field with the median value based on the store type is appropriate.

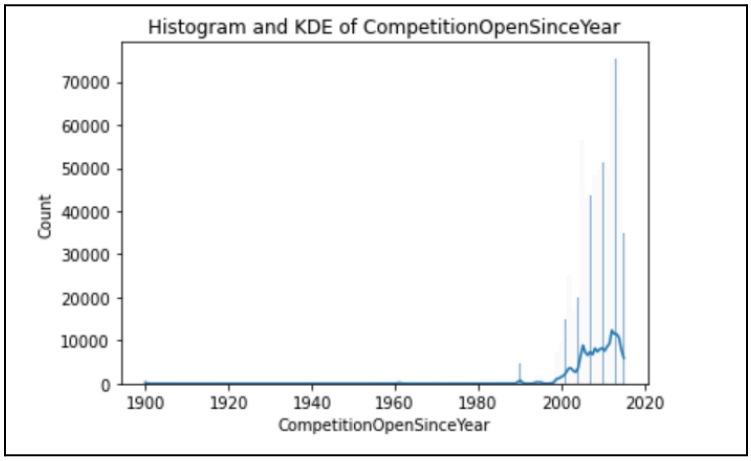



*Figure 2: Histogram and Kernel Density Estimate Graph for describing CompetitonOpenSinceYear field*

The variables CompetitionOpenSinceMonth and CompetitionOpenSinceYear are not useful on their own for describing the prediction model, so two new composite variables, CompetitionOpenTotalMonth and Promo2TotalWeek, are created to provide the total months and weeks since competition and promotion began, enabling sales trends prediction.

A new variable is created called the isStateHoliday which removes the categorical data from StateHoliday and makes it into an indicator variable, where 1 represents the day with a State Holiday. This makes it easier for understanding the pattern associated with store openings and thus the sales. The merged_test dataset is likewise subjected to all of the aforementioned preprocessing procedures.

## Exploratory Data Analysis:

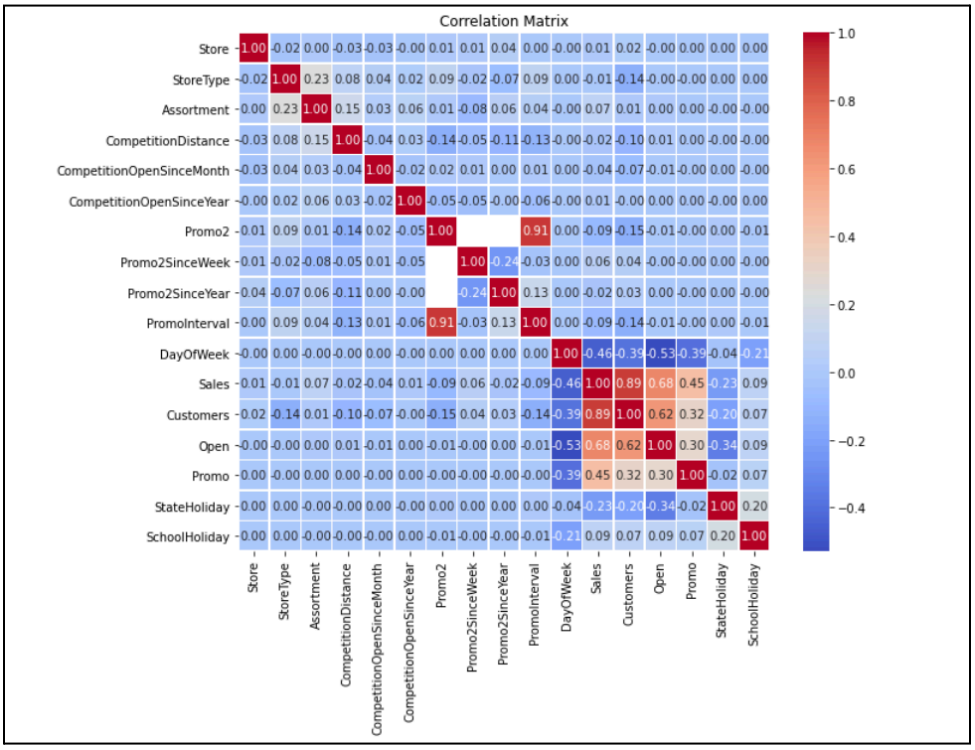



*Figure 3: Correlation Heatmap of the variables in merged_train dataset*

Exploratory Data Analysis (EDA) is a crucial step in data preparation, involving finding patterns, evaluating data quality, and drawing insightful conclusions for sales forecasting. Figure 3 reveals a strong positive correlation between Customers and Sales explaining how sales increase when customers increase. It also shows strong positive correlation between Promo2 and PromoInterval, and Sales and Open. This helps in understanding the logic behind sales forecasting.

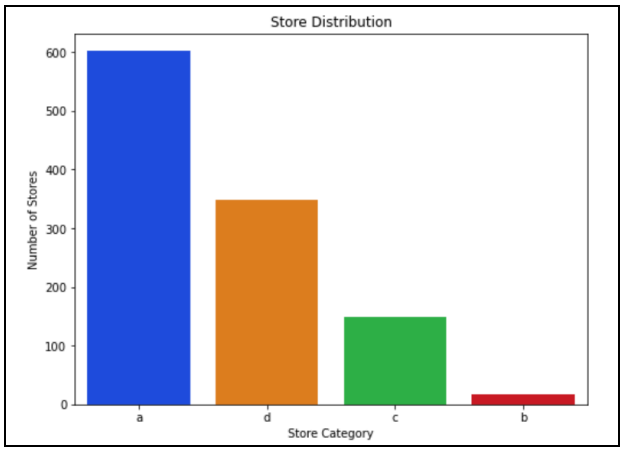



*Figure 4: Store Distribution based on different Store Categories*

Figure 4 shows that the majority of the stores are of type a followed by stores of type d. Though the number of stores of type b are the least, from Figure 5 it is clear that over the years type b stores stand out with the highest aggregate sales. The sales structures of the other store types, however, remain consistent throughout the year with little changes.

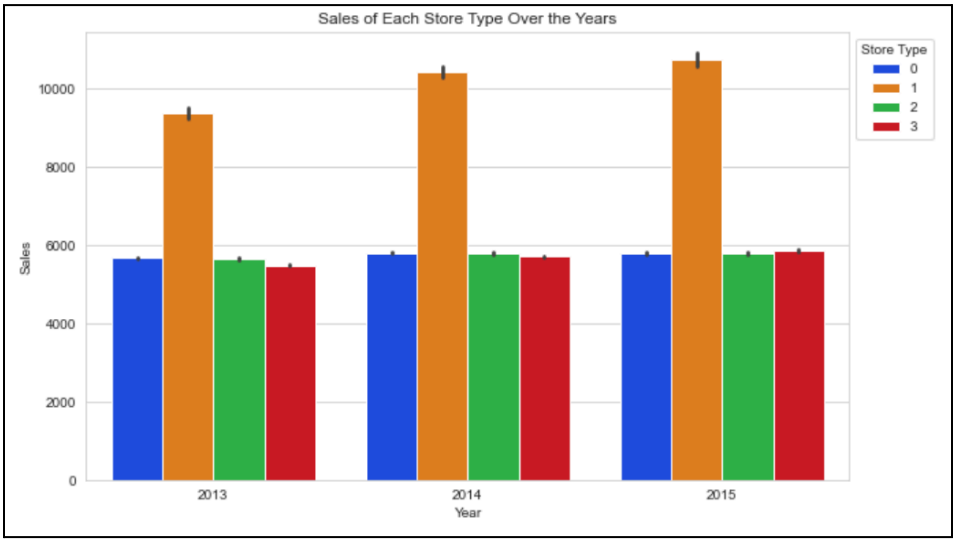



*Figure 5: Sales of each Store type from 2013 to 2015.*

From Figure 6, it is clear that store type b outperforms all the other store types and there is a slight increase in sales during the Month of May and December.

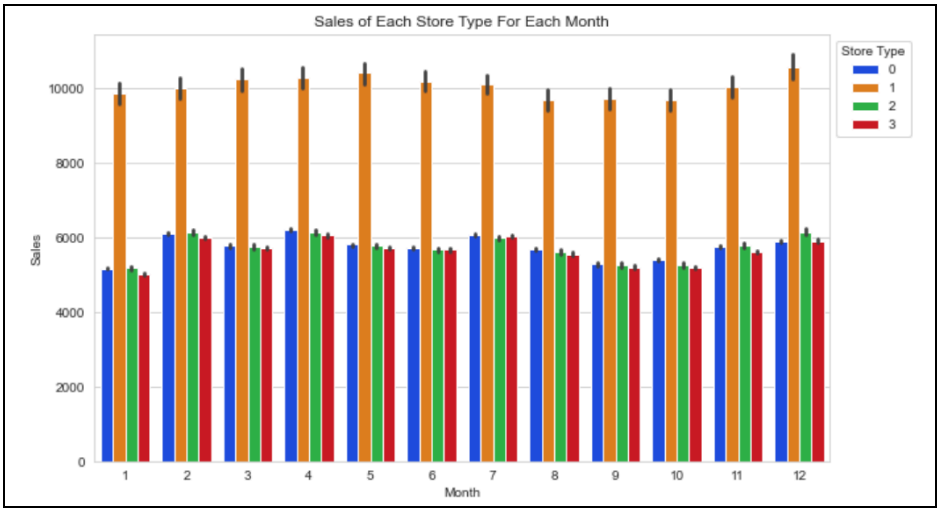



*Figure 6: Sales of each Store type for Each Month*

Whereas from Figure 7, it is clear that the majority of the stores are closed during the State Holidays and thereby affecting the sales of the store.

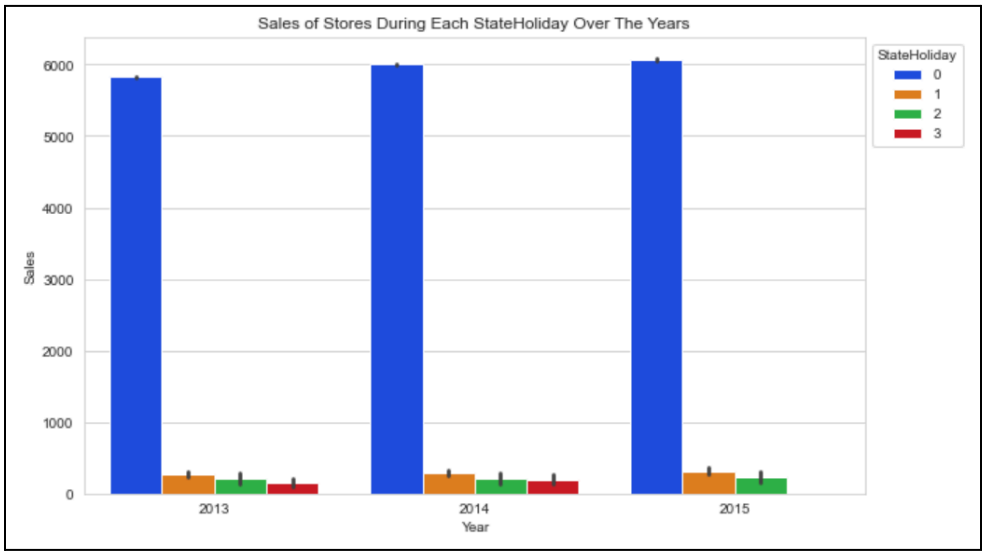



*Figure 7: Sales of Stores during each StateHoliday over the year. 0 represents no holiday, 1 represents public holiday, 2 represents Easter holiday and 3 represents Christmas holiday.*

Figure 8 shows a huge dip in the stores contributing to the sales on a Sunday. This is due to the fact that Germany is one of the countries with most of the shops being closed on Sundays. But it can also be seen that there is a spike of sales on Monday, which is the beginning of a new week.


*Figure 8: Sales of stores during each day of the week*

Figure 9 shows that most of the customers shop at store type b, which explains the reason why store type b had the maximum sales. It can be also noted that the assortment level of extra exists only at store type b. This might be the reason why store type b outperforms when compared to other store types. It can be also noted that the assortment level extended shows a huge spike in the number of customers than the other store types having the same assortment level.

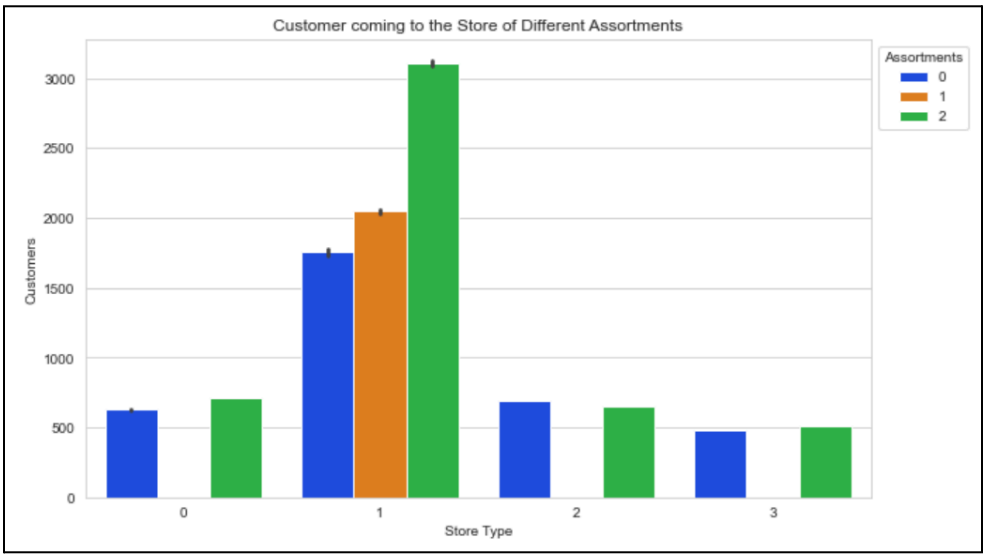



*Figure 9: Customers coming to different store types of different assortment levels*

Figure 10 shows that it is a right skewed distribution. It can also be noted that most of the competitor stores are located within a 10 kilometre radius.

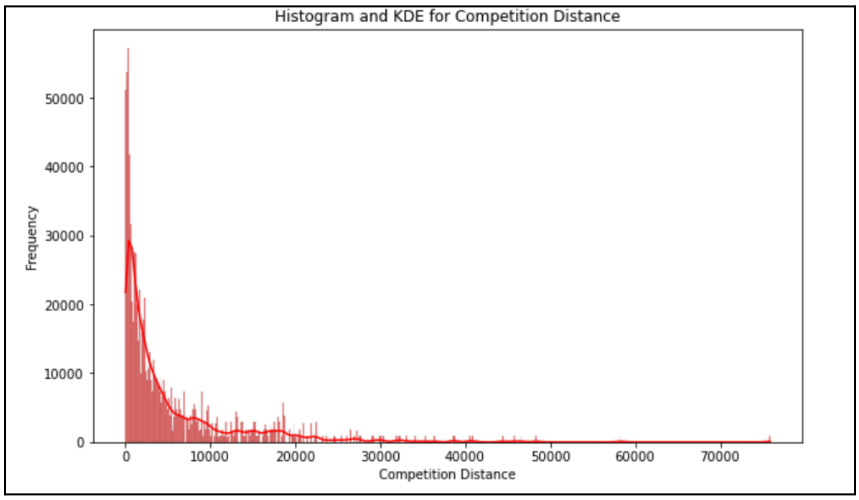



*Figure 10: Histogram and Kernel Density Estimate Graph for CompetitionDistance*

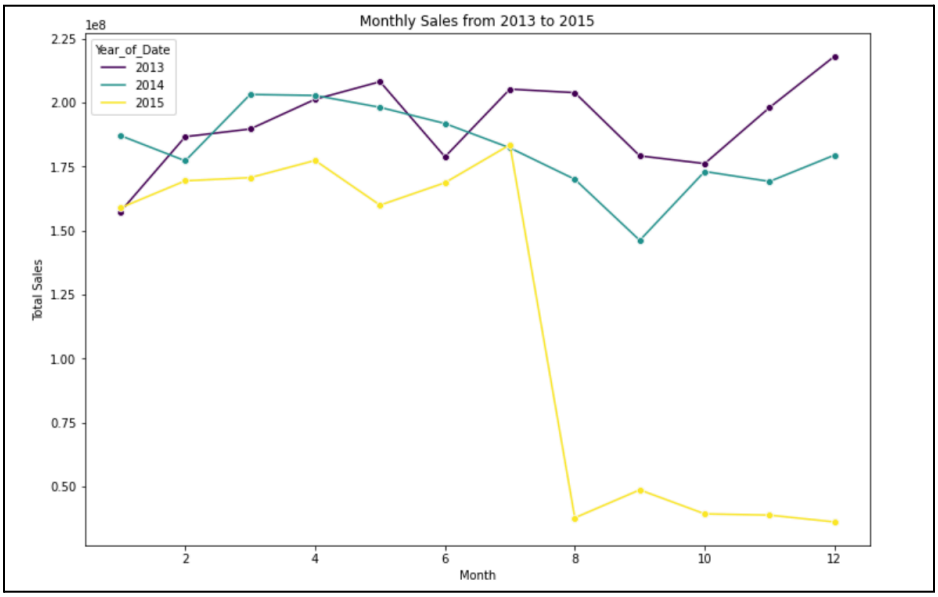



*Figure 11: Monthly Sales of all the stores combined from 2013 to 2015*

From Figure 11, it can be noted that 2013 shows the maximum peak in sales. As the years pass by, the overall sales are decreasing. It can be also seen that there is a drop from the month of August in the 2015 graph. This is due to the lack of data after 31/07/2015.

## Forecasting Models:

The focus now shifts to predictive models for sales forecasting, focusing on the customer field. A random forest regressor is used to train the model and predict the customer field. The efficiency of the model and prediction is evaluated using Root Mean Square Percentage Error (RMSPE), with a 11.76 percentage obtained. The predicted customer values are rounded off and stored in the merged_test dataset.

Two models were used for sales prediction: XGBoost and Random Forest Regressor. XGBoost is a machine learning algorithm that uses a combination of learning methods, including boosting algorithm, gradient boosting, and decision trees. Random Forest Regressor is an ensemble learning technique that combines multiple decision trees for more reliable outcomes. The XGBoost model had a poor RMSPE of 48417397726827.98%, indicating poor data preprocessing or a lack of accuracy in data analysis. The Random Forest Regressor, on the other hand, had an RMSPE of 6.23%, indicating almost accurate predictions. This suggests that the Random Forest Regressor is a better model than the XGBoost model, as shown in Figure 12.

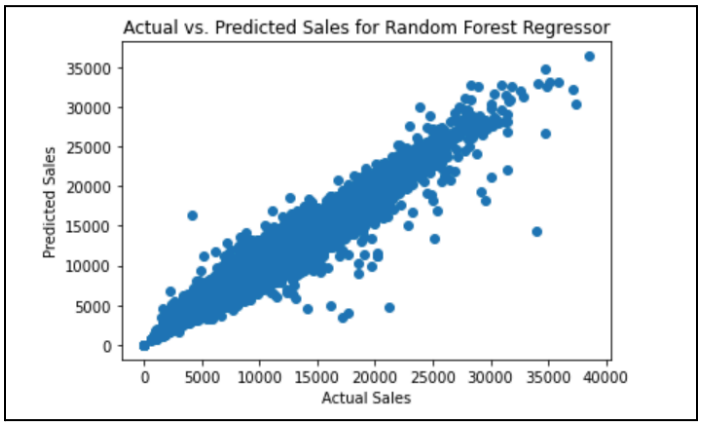



*Figure 12: Actual vs Predicted Sales for Random Forest Regressors*

## Conclusion:

The sales forecasting study's conclusions require a thorough examination of the dataset's constraints, including large null values, computationally intensive models, and delayed predictions. The dataset's complexity makes it difficult to use time series models for seasonality. Sales forecasting is expanding with the rise of e-commerce, offering new ways to analyse sales trends, customer segmentation, and personalization, and offering future scopes for growth.

# Appendix

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from statistics import mode,median, StatisticsError
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
```

# Reading csv Files

```
store=pd.read_csv("store.csv")

train=pd.read_csv("train.csv")

test=pd.read_csv("test.csv")
```

# Preprocessing

## Label Encoding Store Type

```
lb=LabelEncoder()
lb.fit(["a","b","c","d"])
store["StoreType"]=lb.transform(store["StoreType"])

store.head(20)
```

| | Store | StoreType | Assortment | CompetitionDistance | \ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | a | 1270.0 | |
| 1 | 2 | 0 | a | 570.0 | |
| 2 | 3 | 0 | a | 14130.0 | |
| 3 | 4 | 2 | c | 620.0 | |
| 4 | 5 | 0 | a | 29910.0 | |
| 5 | 6 | 0 | a | 310.0 | |
| 6 | 7 | 0 | c | 24000.0 | |
| 7 | 8 | 0 | a | 7520.0 | |
| 8 | 9 | 0 | c | 2030.0 | |
| 9 | 10 | 0 | a | 3160.0 | |
| 10 | 11 | 0 | c | 960.0 | |
| 11 | 12 | 0 | c | 1070.0 | |
| 12 | 13 | 3 | a | 310.0 | |

```
13     14          0          a                1300.0
14     15          3          c                4110.0
15     16          0          c                3270.0
16     17          0          a                  50.0
17     18          3          c               13840.0
18     19          0          c                3240.0
19     20          3          a                2340.0

    CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                         9.0                    2008.0       0
1                        11.0                    2007.0       1
2                        12.0                    2006.0       1
3                         9.0                    2009.0       0
4                         4.0                    2015.0       0
5                        12.0                    2013.0       0
6                         4.0                    2013.0       0
7                        10.0                    2014.0       0
8                         8.0                    2000.0       0
9                         9.0                    2009.0       0
10                       11.0                    2011.0       1
11                        NaN                       NaN       1
12                        NaN                       NaN       1
13                        3.0                    2014.0       1
14                        3.0                    2010.0       1
15                        NaN                       NaN       0
16                       12.0                    2005.0       1
17                        6.0                    2010.0       1
18                        NaN                       NaN       1
19                        5.0                    2009.0       1

    Promo2SinceWeek  Promo2SinceYear     PromoInterval
0               NaN              NaN               NaN
1              13.0           2010.0   Jan,Apr,Jul,Oct
2              14.0           2011.0   Jan,Apr,Jul,Oct
3               NaN              NaN               NaN
4               NaN              NaN               NaN
5               NaN              NaN               NaN
6               NaN              NaN               NaN
7               NaN              NaN               NaN
8               NaN              NaN               NaN
9               NaN              NaN               NaN
10              1.0           2012.0   Jan,Apr,Jul,Oct
11             13.0           2010.0   Jan,Apr,Jul,Oct
12             45.0           2009.0   Feb,May,Aug,Nov
13             40.0           2011.0   Jan,Apr,Jul,Oct
14             14.0           2011.0   Jan,Apr,Jul,Oct
15              NaN              NaN               NaN
16             26.0           2010.0   Jan,Apr,Jul,Oct
17             14.0           2012.0   Jan,Apr,Jul,Oct
```

```
18              22.0           2011.0  Mar,Jun,Sept,Dec
19              40.0           2014.0   Jan,Apr,Jul,Oct
```

## Label Encoding Assortment

```
lb.fit(["a","b","c"])
store["Assortment"]=lb.transform(store["Assortment"])

store.head(20)
```

```
    Store  StoreType  Assortment  CompetitionDistance  \
0       1          2           0               1270.0
1       2          0           0                570.0
2       3          0           0              14130.0
3       4          2           2                620.0
4       5          0           0              29910.0
5       6          0           0                310.0
6       7          0           2              24000.0
7       8          0           0               7520.0
8       9          0           2               2030.0
9      10          0           0               3160.0
10     11          0           2                960.0
11     12          0           2               1070.0
12     13          3           0                310.0
13     14          0           0               1300.0
14     15          3           2               4110.0
15     16          0           2               3270.0
16     17          0           0                 50.0
17     18          3           2              13840.0
18     19          0           2               3240.0
19     20          3           0               2340.0

    CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                         9.0                    2008.0       0
1                        11.0                    2007.0       1
2                        12.0                    2006.0       1
3                         9.0                    2009.0       0
4                         4.0                    2015.0       0
5                        12.0                    2013.0       0
6                         4.0                    2013.0       0
7                        10.0                    2014.0       0
8                         8.0                    2000.0       0
9                         9.0                    2009.0       0
10                       11.0                    2011.0       1
11                        NaN                       NaN       1
12                        NaN                       NaN       1
13                        3.0                    2014.0       1
14                        3.0                    2010.0       1
15                        NaN                       NaN       0
16                       12.0                    2005.0       1
```

```
17                        6.0                    2010.0       1
18                        NaN                       NaN       1
19                        5.0                    2009.0       1
```

```
    Promo2SinceWeek  Promo2SinceYear     PromoInterval
0               NaN              NaN               NaN
1              13.0           2010.0   Jan,Apr,Jul,Oct
2              14.0           2011.0   Jan,Apr,Jul,Oct
3               NaN              NaN               NaN
4               NaN              NaN               NaN
5               NaN              NaN               NaN
6               NaN              NaN               NaN
7               NaN              NaN               NaN
8               NaN              NaN               NaN
9               NaN              NaN               NaN
10              1.0           2012.0   Jan,Apr,Jul,Oct
11             13.0           2010.0   Jan,Apr,Jul,Oct
12             45.0           2009.0   Feb,May,Aug,Nov
13             40.0           2011.0   Jan,Apr,Jul,Oct
14             14.0           2011.0   Jan,Apr,Jul,Oct
15              NaN              NaN               NaN
16             26.0           2010.0   Jan,Apr,Jul,Oct
17             14.0           2012.0   Jan,Apr,Jul,Oct
18             22.0           2011.0  Mar,Jun,Sept,Dec
19             40.0           2014.0   Jan,Apr,Jul,Oct
```

## Filling 0 for null values in PromoInterval and Label Encoding PromoInterval

```
store["PromoInterval"]=store["PromoInterval"].fillna("0")

lb.fit(["0","Feb,May,Aug,Nov","Jan,Apr,Jul,Oct","Mar,Jun,Sept,Dec"])
store["PromoInterval"]=lb.transform(store["PromoInterval"])
```

## Label Encoding State Holiday for Training Data

```
train["StateHoliday"]=train["StateHoliday"].replace(0,'0')

lb.fit(["0","a","b","c"])
train["StateHoliday"]=lb.transform(train["StateHoliday"])

train.head(20)
```

```
    Store  DayOfWeek        Date  Sales  Customers  Open  Promo
StateHoliday  \
0       1          5  31/07/2015   5263        555     1      1
0
1       2          5  31/07/2015   6064        625     1      1
0
2       3          5  31/07/2015   8314        821     1      1
```

```
0
3        4          5  31/07/2015  13995        1498      1       1
0
4        5          5  31/07/2015   4822         559      1       1
0
5        6          5  31/07/2015   5651         589      1       1
0
6        7          5  31/07/2015  15344        1414      1       1
0
7        8          5  31/07/2015   8492         833      1       1
0
8        9          5  31/07/2015   8565         687      1       1
0
9       10          5  31/07/2015   7185         681      1       1
0
10      11          5  31/07/2015  10457        1236      1       1
0
11      12          5  31/07/2015   8959         962      1       1
0
12      13          5  31/07/2015   8821         568      1       1
0
13      14          5  31/07/2015   6544         710      1       1
0
14      15          5  31/07/2015   9191         766      1       1
0
15      16          5  31/07/2015  10231         979      1       1
0
16      17          5  31/07/2015   8430         946      1       1
0
17      18          5  31/07/2015  10071         936      1       1
0
18      19          5  31/07/2015   8234         718      1       1
0
19      20          5  31/07/2015   9593         974      1       1
0

    SchoolHoliday
0               1
1               1
2               1
3               1
4               1
5               1
6               1
7               1
8               1
9               1
10              1
11              1
```

```
12              0
13              1
14              1
15              1
16              1
17              1
18              1
19              0
```

## Label Encoding State Holiday for Test Data

```
lb.fit(["0","a"])
test["StateHoliday"]=lb.transform(test["StateHoliday"])

test.head(20)
```

```
    Store  DayOfWeek        Date  Sales  Customers  Open  Promo
StateHoliday  \
0       1          4  17/09/2015    NaN        NaN   1.0      1
0
1       3          4  17/09/2015    NaN        NaN   1.0      1
0
2       7          4  17/09/2015    NaN        NaN   1.0      1
0
3       8          4  17/09/2015    NaN        NaN   1.0      1
0
4       9          4  17/09/2015    NaN        NaN   1.0      1
0
5      10          4  17/09/2015    NaN        NaN   1.0      1
0
6      11          4  17/09/2015    NaN        NaN   1.0      1
0
7      12          4  17/09/2015    NaN        NaN   1.0      1
0
8      13          4  17/09/2015    NaN        NaN   1.0      1
0
9      14          4  17/09/2015    NaN        NaN   1.0      1
0
10     15          4  17/09/2015    NaN        NaN   1.0      1
0
11     16          4  17/09/2015    NaN        NaN   1.0      1
0
12     19          4  17/09/2015    NaN        NaN   1.0      1
0
13     20          4  17/09/2015    NaN        NaN   1.0      1
0
14     21          4  17/09/2015    NaN        NaN   1.0      1
0
15     22          4  17/09/2015    NaN        NaN   1.0      1
0
```

```
16     23           4  17/09/2015    NaN         NaN   1.0       1
0
17     24           4  17/09/2015    NaN         NaN   1.0       1
0
18     25           4  17/09/2015    NaN         NaN   1.0       1
0
19     27           4  17/09/2015    NaN         NaN   1.0       1
0
```

```
    SchoolHoliday
0               0
1               0
2               0
3               0
4               0
5               0
6               0
7               0
8               0
9               0
10              0
11              0
12              0
13              0
14              0
15              0
16              0
17              0
18              0
19              0
```

## Merge Store to Train and Test Data

```
merged_train=pd.merge(store, train, on="Store")

merged_test=pd.merge(store, test, on="Store")

merged_train.head(10)
```

```
   Store  StoreType  Assortment  CompetitionDistance  \
0      1          2           0               1270.0
1      1          2           0               1270.0
2      1          2           0               1270.0
3      1          2           0               1270.0
4      1          2           0               1270.0
5      1          2           0               1270.0
6      1          2           0               1270.0
7      1          2           0               1270.0
8      1          2           0               1270.0
9      1          2           0               1270.0
```

```
   CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                        9.0                    2008.0       0
1                        9.0                    2008.0       0
2                        9.0                    2008.0       0
3                        9.0                    2008.0       0
4                        9.0                    2008.0       0
5                        9.0                    2008.0       0
6                        9.0                    2008.0       0
7                        9.0                    2008.0       0
8                        9.0                    2008.0       0
9                        9.0                    2008.0       0

   Promo2SinceWeek  Promo2SinceYear  PromoInterval  DayOfWeek
Date  \
0              NaN              NaN              0          5
31/07/2015
1              NaN              NaN              0          4
30/07/2015
2              NaN              NaN              0          3
29/07/2015
3              NaN              NaN              0          2
28/07/2015
4              NaN              NaN              0          1
27/07/2015
5              NaN              NaN              0          7
26/07/2015
6              NaN              NaN              0          6
25/07/2015
7              NaN              NaN              0          5
24/07/2015
8              NaN              NaN              0          4
23/07/2015
9              NaN              NaN              0          3
22/07/2015

   Sales  Customers  Open  Promo  StateHoliday  SchoolHoliday
0   5263        555     1      1             0              1
1   5020        546     1      1             0              1
2   4782        523     1      1             0              1
3   5011        560     1      1             0              1
4   6102        612     1      1             0              1
5      0          0     0      0             0              0
6   4364        500     1      0             0              0
7   3706        459     1      0             0              0
8   3769        503     1      0             0              0
9   3464        463     1      0             0              0
```

## Training Data

Deleting rows where Customer is 0 when Open is 1 in Train Data

```
merged_train.drop(merged_train[(merged_train['Open'] == 1) &
(merged_train['Customers'] == 0)].index, inplace=True)
```

Splitting Date into Components in Train Data

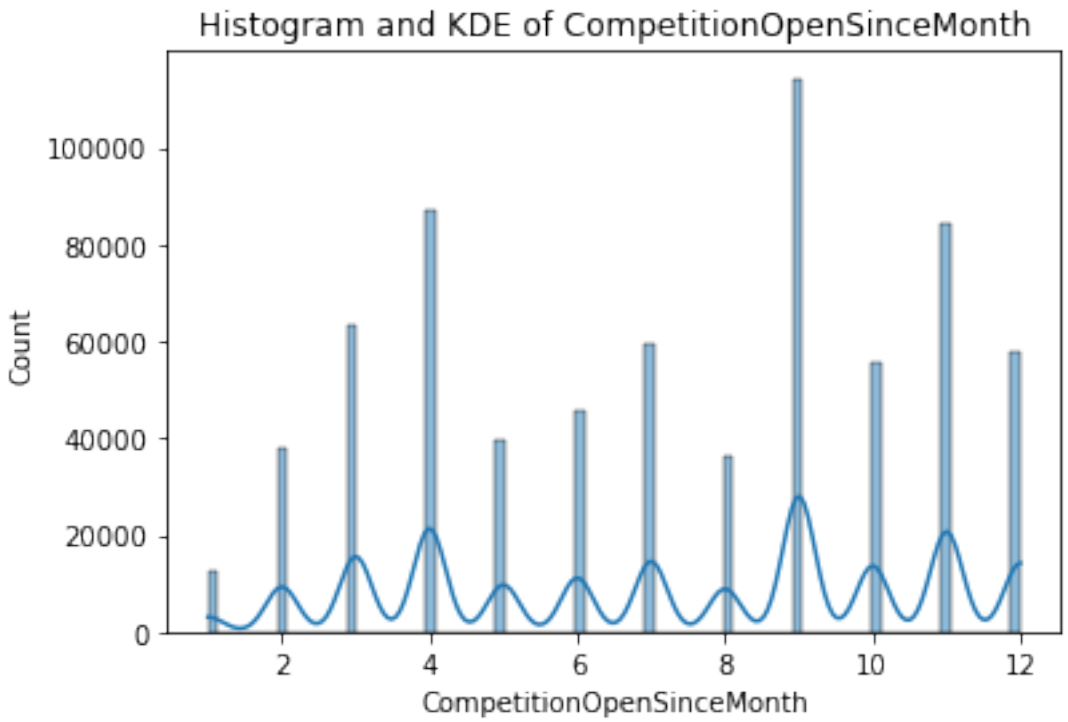
```
merged_train["Date"]=pd.to_datetime(merged_train["Date"])
merged_train["Week_of_Date"]=merged_train["Date"].dt.isocalendar().wee
k
merged_train["Day_of_Date"]=merged_train["Date"].dt.day
merged_train["Month_of_Date"]=merged_train["Date"].dt.month
merged_train["Year_of_Date"]=merged_train["Date"].dt.year
```

Calculating the Total no of Weeks of Implementing Promo2 from Given Date in Train Data

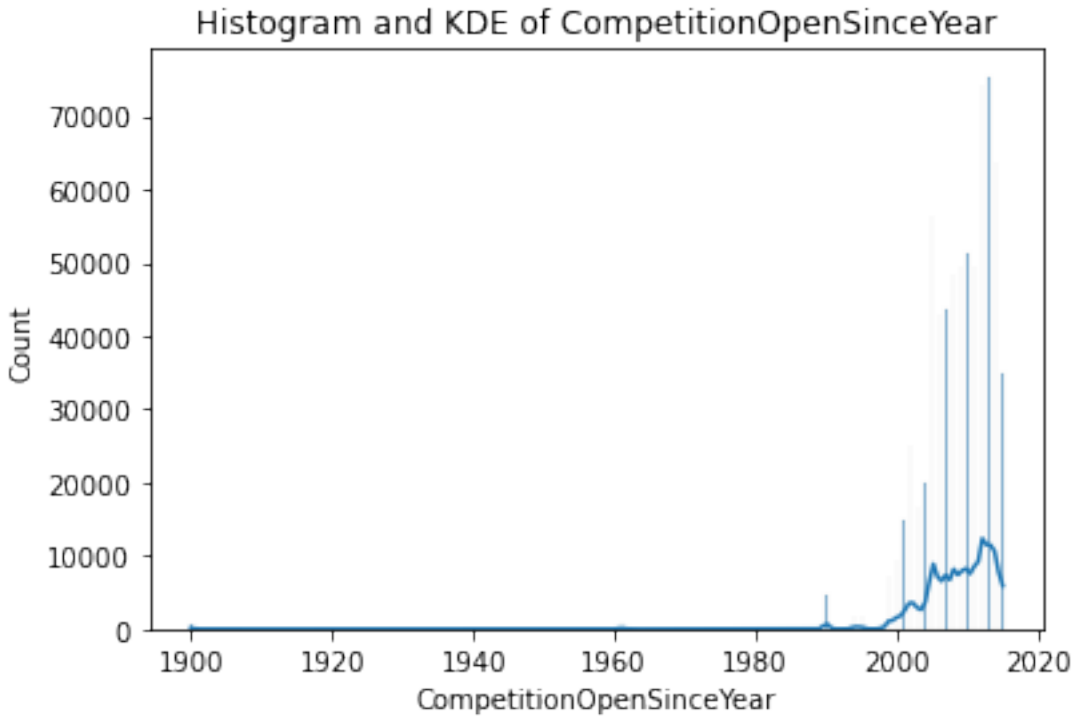
```
def calc_promo2_week(row):
    if(row["Promo2"]==1):
        return ((row["Year_of_Date"]-row["Promo2SinceYear"])*52)+
(row["Week_of_Date"]-row["Promo2SinceWeek"])
    else:
        return 0

merged_train["Promo2TotalWeek"]=merged_train.apply(calc_promo2_week,
axis=1)
```

Imputing CompetitionOpenSinceYear and CompetitionOpenSinceMonth with Mean and Median values in Train Data

```
sns.histplot(merged_train['CompetitionOpenSinceMonth'], kde=True)
plt.title('Histogram and KDE of CompetitionOpenSinceMonth')
plt.show()
```




```
sns.histplot(merged_train['CompetitionOpenSinceYear'], kde=True)
plt.title('Histogram and KDE of CompetitionOpenSinceYear')
plt.show()
```



```
merged_train["CompetitionDistance"]=merged_train["CompetitionDistance"
].fillna(0)

merged_train["StoreType"]=merged_train["StoreType"].astype('category')
merged_train["Assortment"]=merged_train["Assortment"].astype('category
')

merged_train['CompetitionOpenSinceMonth'] =
merged_train.groupby(['StoreType'])
['CompetitionOpenSinceMonth'].transform(lambda x:
x.fillna(x.mode().iloc[0]))

merged_train['CompetitionOpenSinceYear'] =
merged_train.groupby(['StoreType'])
['CompetitionOpenSinceYear'].transform(lambda x: x.fillna(x.median()))
```

Calculating the Total no of Months of Competition from Given Date in Train Data

```
def calc_competition_month(row):
    if(row["CompetitionDistance"]!=0):
        return ((row["Year_of_Date"]-
row["CompetitionOpenSinceYear"])*12)+(row["Month_of_Date"]-
row["CompetitionOpenSinceMonth"])
    else:
        return 0

merged_train["CompetitionOpenTotalMonth"]=merged_train.apply(calc_comp
etition_month, axis=1)

merged_train.head(10)
```

```
   Store StoreType Assortment  CompetitionDistance
CompetitionOpenSinceMonth  \
0      1         2          0               1270.0
9.0
1      1         2          0               1270.0
9.0
2      1         2          0               1270.0
9.0
3      1         2          0               1270.0
9.0
4      1         2          0               1270.0
9.0
5      1         2          0               1270.0
9.0
6      1         2          0               1270.0
9.0
7      1         2          0               1270.0
9.0
8      1         2          0               1270.0
9.0
```

```
9      1          2           0                1270.0
9.0

   CompetitionOpenSinceYear  Promo2  Promo2SinceWeek  Promo2SinceYear
\
0                    2008.0       0              NaN              NaN

1                    2008.0       0              NaN              NaN

2                    2008.0       0              NaN              NaN

3                    2008.0       0              NaN              NaN

4                    2008.0       0              NaN              NaN

5                    2008.0       0              NaN              NaN

6                    2008.0       0              NaN              NaN

7                    2008.0       0              NaN              NaN

8                    2008.0       0              NaN              NaN

9                    2008.0       0              NaN              NaN


   PromoInterval  ...  Open Promo  StateHoliday  SchoolHoliday
Week_of_Date  \
0              0  ...     1     1             0              1
31
1              0  ...     1     1             0              1
31
2              0  ...     1     1             0              1
31
3              0  ...     1     1             0              1
31
4              0  ...     1     1             0              1
31
5              0  ...     0     0             0              0
30
6              0  ...     1     0             0              0
30
7              0  ...     1     0             0              0
30
8              0  ...     1     0             0              0
30
9              0  ...     1     0             0              0
30

   Day_of_Date  Month_of_Date  Year_of_Date  Promo2TotalWeek  \
```

```
0            31                7           2015                0.0
1            30                7           2015                0.0
2            29                7           2015                0.0
3            28                7           2015                0.0
4            27                7           2015                0.0
5            26                7           2015                0.0
6            25                7           2015                0.0
7            24                7           2015                0.0
8            23                7           2015                0.0
9            22                7           2015                0.0

   CompetitionOpenTotalMonth
0                       82.0
1                       82.0
2                       82.0
3                       82.0
4                       82.0
5                       82.0
6                       82.0
7                       82.0
8                       82.0
9                       82.0

[10 rows x 24 columns]
```

Creating isStateHoliday with only category as 0 or 1

```
merged_train["isStateHoliday"]=(merged_train["StateHoliday"]!
=0).astype(int)
```

Converting merge_train to csv file

```
merged_train.to_csv("merged_train.csv", index=False)
```

## Testing Data

```
merged_test.head(10)

   Store  StoreType  Assortment  CompetitionDistance  \
0      1          2           0               1270.0
1      1          2           0               1270.0
2      1          2           0               1270.0
3      1          2           0               1270.0
4      1          2           0               1270.0
5      1          2           0               1270.0
6      1          2           0               1270.0
7      1          2           0               1270.0
8      1          2           0               1270.0
9      1          2           0               1270.0
```

```
   CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                        9.0                    2008.0       0
1                        9.0                    2008.0       0
2                        9.0                    2008.0       0
3                        9.0                    2008.0       0
4                        9.0                    2008.0       0
5                        9.0                    2008.0       0
6                        9.0                    2008.0       0
7                        9.0                    2008.0       0
8                        9.0                    2008.0       0
9                        9.0                    2008.0       0

   Promo2SinceWeek  Promo2SinceYear  PromoInterval  DayOfWeek
Date  \
0              NaN              NaN              0          4
17/09/2015
1              NaN              NaN              0          3
16/09/2015
2              NaN              NaN              0          2
15/09/2015
3              NaN              NaN              0          1
14/09/2015
4              NaN              NaN              0          7
13/09/2015
5              NaN              NaN              0          6
12/9/2015
6              NaN              NaN              0          5
11/9/2015
7              NaN              NaN              0          4
10/9/2015
8              NaN              NaN              0          3
9/9/2015
9              NaN              NaN              0          2
8/9/2015

   Sales  Customers  Open  Promo  StateHoliday  SchoolHoliday
0    NaN        NaN   1.0      1             0              0
1    NaN        NaN   1.0      1             0              0
2    NaN        NaN   1.0      1             0              0
3    NaN        NaN   1.0      1             0              0
4    NaN        NaN   0.0      0             0              0
5    NaN        NaN   1.0      0             0              0
6    NaN        NaN   1.0      0             0              0
7    NaN        NaN   1.0      0             0              0
8    NaN        NaN   1.0      0             0              0
9    NaN        NaN   1.0      0             0              0
```

Splitting Date into Components in Test Data

```
merged_test["Date"]=pd.to_datetime(merged_test["Date"])
merged_test["Week_of_Date"]=merged_test["Date"].dt.isocalendar().week
merged_test["Day_of_Date"]=merged_test["Date"].dt.day
merged_test["Month_of_Date"]=merged_test["Date"].dt.month
merged_test["Year_of_Date"]=merged_test["Date"].dt.year
```

Calculating the Total no of Weeks of Implementing Promo2 from Given Date in Test Data

```
def calc_promo2_week(row):
    if(row["Promo2"]==1):
        return ((row["Year_of_Date"]-row["Promo2SinceYear"])*52)+
(row["Week_of_Date"]-row["Promo2SinceWeek"])
    else:
        return 0

merged_test["Promo2TotalWeek"]=merged_test.apply(calc_promo2_week,
axis=1)
```

Imputing CompetitionOpenSinceYear and CompetitionOpenSinceMonth with Mean and Median values in Test Data

```
merged_test["CompetitionDistance"]=merged_test["CompetitionDistance"].
fillna(0)

merged_test["StoreType"]=merged_test["StoreType"].astype('category')

merged_test['CompetitionOpenSinceMonth'] =
merged_test.groupby(['StoreType'])
['CompetitionOpenSinceMonth'].transform(lambda x:
x.fillna(x.mode().iloc[0]))

merged_test['CompetitionOpenSinceYear'] =
merged_test.groupby(['StoreType'])
['CompetitionOpenSinceYear'].transform(lambda x: x.fillna(x.median()))
```

Calculating the Total no of Months of Competition from Given Date in Test Data

```
def calc_competition_month(row):
    if(row["CompetitionDistance"]!=0):
        return ((row["Year_of_Date"]-
row["CompetitionOpenSinceYear"])*12)+(row["Month_of_Date"]-
row["CompetitionOpenSinceMonth"])
    else:
        return 0

merged_test["CompetitionOpenTotalMonth"]=merged_test.apply(calc_compet
ition_month, axis=1)

merged_test.head(10)
```

```
   Store StoreType  Assortment  CompetitionDistance  \
0      1         2           0                1270.0
1      1         2           0                1270.0
2      1         2           0                1270.0
3      1         2           0                1270.0
4      1         2           0                1270.0
5      1         2           0                1270.0
6      1         2           0                1270.0
7      1         2           0                1270.0
8      1         2           0                1270.0
9      1         2           0                1270.0

   CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                        9.0                    2008.0       0
1                        9.0                    2008.0       0
2                        9.0                    2008.0       0
3                        9.0                    2008.0       0
4                        9.0                    2008.0       0
5                        9.0                    2008.0       0
6                        9.0                    2008.0       0
7                        9.0                    2008.0       0
8                        9.0                    2008.0       0
9                        9.0                    2008.0       0

   Promo2SinceWeek  Promo2SinceYear  PromoInterval  ...  Open Promo  \
0              NaN              NaN              0  ...   1.0     1
1              NaN              NaN              0  ...   1.0     1
2              NaN              NaN              0  ...   1.0     1
3              NaN              NaN              0  ...   1.0     1
4              NaN              NaN              0  ...   0.0     0
5              NaN              NaN              0  ...   1.0     0
6              NaN              NaN              0  ...   1.0     0
7              NaN              NaN              0  ...   1.0     0
8              NaN              NaN              0  ...   1.0     0
9              NaN              NaN              0  ...   1.0     0

   StateHoliday  SchoolHoliday  Week_of_Date  Day_of_Date
Month_of_Date  \
0             0              0            38           17
9
1             0              0            38           16
9
2             0              0            38           15
9
3             0              0            38           14
9
4             0              0            37           13
9
5             0              0            50            9
12
```

```
6              0               0              46              9
11
7              0               0              41              9
10
8              0               0              37              9
9
9              0               0              32              9
8

   Year_of_Date  Promo2TotalWeek  CompetitionOpenTotalMonth
0          2015              0.0                       84.0
1          2015              0.0                       84.0
2          2015              0.0                       84.0
3          2015              0.0                       84.0
4          2015              0.0                       84.0
5          2015              0.0                       87.0
6          2015              0.0                       86.0
7          2015              0.0                       85.0
8          2015              0.0                       84.0
9          2015              0.0                       83.0

[10 rows x 24 columns]
```

Creating isStateHoliday with only category as 0 or 1

```
merged_test["isStateHoliday"]=(merged_test["StateHoliday"]!
=0).astype(int)
```

Dropping the rows where Open is Null

```
merged_test = merged_test.dropna(subset=['Open'])
```

Converting merge_test to csv file

```
merged_test.to_csv("merged_test.csv", index=False)
```

# Predicting Customers in Test Data

```
final_train=merged_train[["Store","StoreType","Assortment","Competitio
nDistance","Promo2","PromoInterval","DayOfWeek","Customers","Open","Pr
omo","isStateHoliday","SchoolHoliday",
"Week_of_Date","Day_of_Date","Month_of_Date","Year_of_Date","Promo2Tot
alWeek","CompetitionOpenTotalMonth"]]

final_train.head(10)

   Store  StoreType  Assortment  CompetitionDistance  Promo2
PromoInterval  \
0      1          2           0                 1270       0
```

```
0
1      1          2           0                1270       0
0
2      1          2           0                1270       0
0
3      1          2           0                1270       0
0
4      1          2           0                1270       0
0
5      1          2           0                1270       0
0
6      1          2           0                1270       0
0
7      1          2           0                1270       0
0
8      1          2           0                1270       0
0
9      1          2           0                1270       0
0

   DayOfWeek  Customers  Open  Promo  isStateHoliday  SchoolHoliday  \
0          5        555     1      1               0              1
1          4        546     1      1               0              1
2          3        523     1      1               0              1
3          2        560     1      1               0              1
4          1        612     1      1               0              1
5          7          0     0      0               0              0
6          6        500     1      0               0              0
7          5        459     1      0               0              0
8          4        503     1      0               0              0
9          3        463     1      0               0              0

   Week_of_Date  Day_of_Date  Month_of_Date  Year_of_Date
Promo2TotalWeek  \
0            31           31              7          2015
0
1            31           30              7          2015
0
2            31           29              7          2015
0
3            31           28              7          2015
0
4            31           27              7          2015
0
5            30           26              7          2015
0
6            30           25              7          2015
0
7            30           24              7          2015
```

```
0
8            30            23                7           2015
0
9            30            22                7           2015
0
```

```
   CompetitionOpenTotalMonth
0                         82
1                         82
2                         82
3                         82
4                         82
5                         82
6                         82
7                         82
8                         82
9                         82
```

```
X = final_train.drop('Customers', axis=1)
y = final_train['Customers']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

rf_cust_model = RandomForestRegressor(n_estimators=100,
random_state=42)
rf_cust_model.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=42)
```

```
y_pred = rf_cust_model.predict(X_test)

epsilon = 1e-10
rmspe = np.sqrt(np.mean(((y_test - y_pred) / (y_test + epsilon)) **
2)) * 100

print(f'RMSPE: {rmspe:.2f}%')
```

```
RMSPE: 11.76%
```

```
final_test=merged_test[["Store","StoreType","Assortment","CompetitionD
istance","Promo2","PromoInterval","DayOfWeek","Customers","Open","Prom
o","isStateHoliday","SchoolHoliday",
"Week_of_Date","Day_of_Date","Month_of_Date","Year_of_Date","Promo2Tot
alWeek","CompetitionOpenTotalMonth"]]

final_test_X=final_test.drop('Customers', axis=1)

cust_pred=rf_cust_model.predict(final_test_X)

final_test_copy=final_test.copy()
```

```
final_test_copy["Customers"]=cust_pred

final_test_copy.head(10)
```

```
   Store  StoreType  Assortment  CompetitionDistance  Promo2
PromoInterval  \
0      1          2           0                 1270       0
0
1      1          2           0                 1270       0
0
2      1          2           0                 1270       0
0
3      1          2           0                 1270       0
0
4      1          2           0                 1270       0
0
5      1          2           0                 1270       0
0
6      1          2           0                 1270       0
0
7      1          2           0                 1270       0
0
8      1          2           0                 1270       0
0
9      1          2           0                 1270       0
0

   DayOfWeek  Customers  Open  Promo  isStateHoliday  SchoolHoliday  \
0          4     508.33   1.0      1               0              0
1          3     524.88   1.0      1               0              0
2          2     538.05   1.0      1               0              0
3          1     568.93   1.0      1               0              0
4          7       0.00   0.0      0               0              0
5          6     573.43   1.0      0               0              0
6          5     447.64   1.0      0               0              0
7          4     448.85   1.0      0               0              0
8          3     470.34   1.0      0               0              0
9          2     478.07   1.0      0               0              0

   Week_of_Date  Day_of_Date  Month_of_Date  Year_of_Date
Promo2TotalWeek  \
0            38           17              9          2015
0
1            38           16              9          2015
0
2            38           15              9          2015
0
3            38           14              9          2015
0
4            37           13              9          2015
```

```
0
5              50              9              12          2015
0
6              46              9              11          2015
0
7              41              9              10          2015
0
8              37              9               9          2015
0
9              32              9               8          2015
0
```

```
   CompetitionOpenTotalMonth
0                         84
1                         84
2                         84
3                         84
4                         84
5                         87
6                         86
7                         85
8                         84
9                         83
```

```
final_test_copy["Customers"]=final_test_copy["Customers"].round()

final_test_copy["Customers"]=final_test_copy["Customers"].astype(int)
```

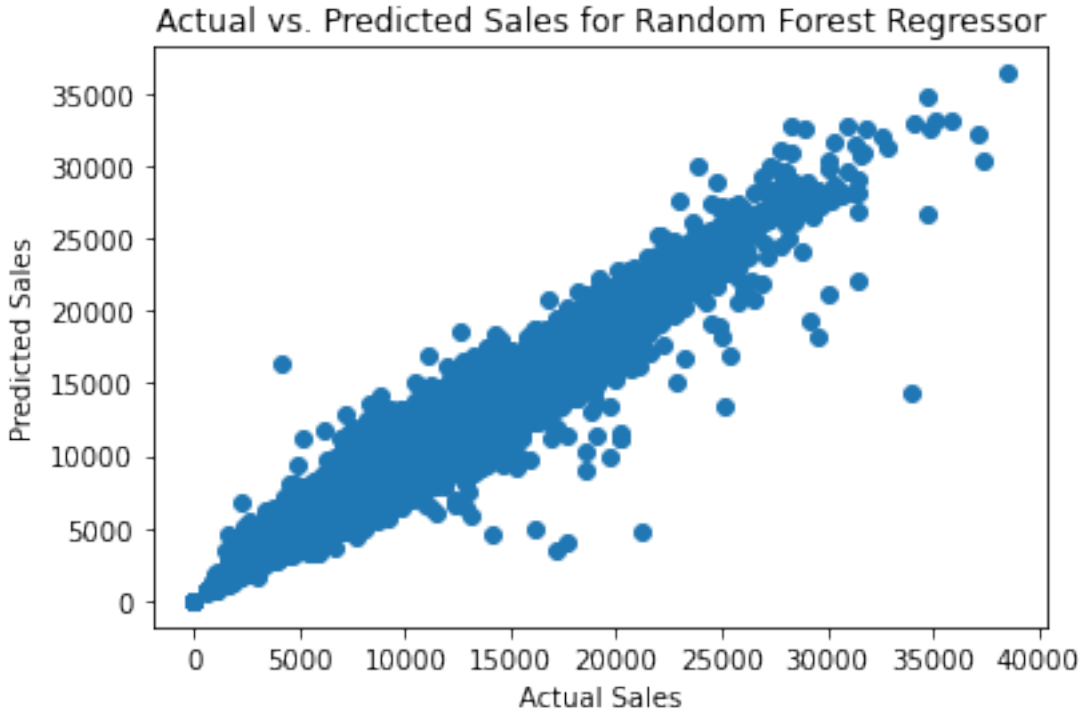
# Models for predicting Sales

```
merged_train=pd.read_csv("merged_train.csv")
merged_test=pd.read_csv("merged_test.csv")
```

## Random Forest

```
final_train=merged_train[["Store","StoreType","Assortment","CompetitionDistance","DayOfWeek","Sales",
"Customers","Promo","Day_of_Date","Month_of_Date","Year_of_Date","Promo2TotalWeek","CompetitionOpenTotalMonth"]]
X = final_train.drop('Sales', axis=1)
y = final_train['Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

rf_sales_2 = RandomForestRegressor(n_estimators=100, random_state=42)
rf_sales_2.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=42)

y_pred = rf_sales_2.predict(X_test)

epsilon = 1e-10
rmspe = np.sqrt(np.mean(((y_test - y_pred) / (y_test + epsilon)) **
2)) * 100

print(f'RMSPE: {rmspe:.2f}%')

RMSPE: 6.23%

plt.scatter(y_test, y_pred)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales for Random Forest Regressor')
plt.show()
```




## XGBoost

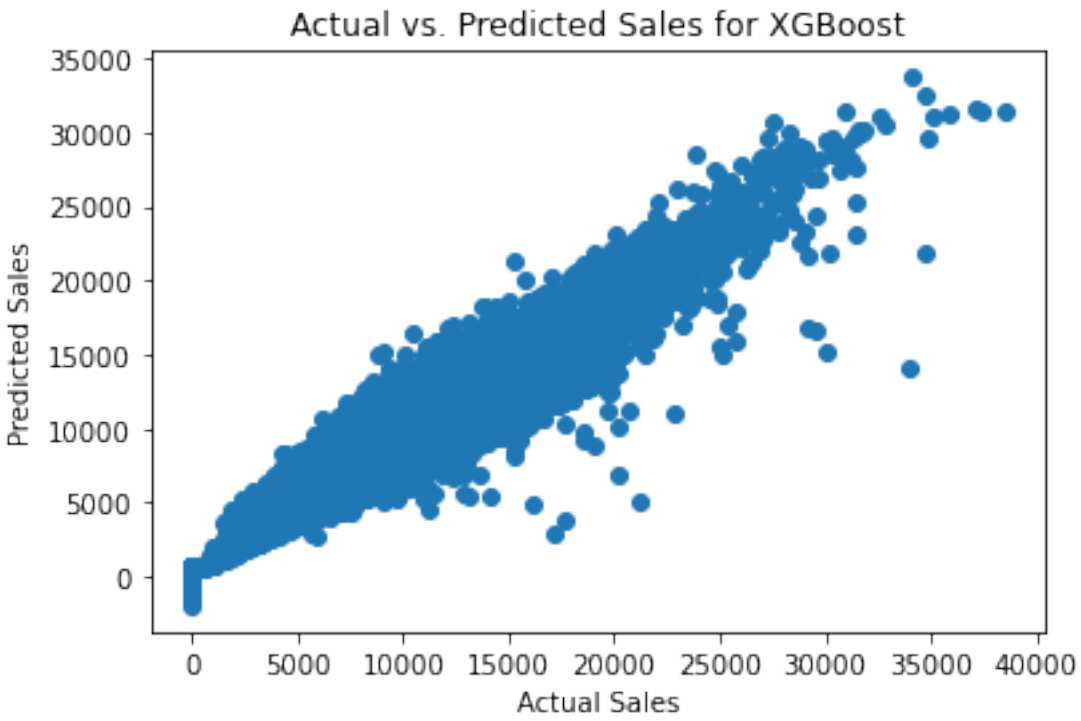
```
pip install xgboost

Collecting xgboostNote: you may need to restart the kernel to use
updated packages.
  Downloading xgboost-1.5.2-py3-none-win_amd64.whl (106.6 MB)
Requirement already satisfied: scipy in c:\users\reeth\anaconda3\envs\
60711\lib\site-packages (from xgboost) (1.5.2)
Requirement already satisfied: numpy in c:\users\reeth\anaconda3\envs\
```

```
60711\lib\site-packages (from xgboost) (1.19.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.5.2
```

```
import xgboost as xgb

#final_train=merged_train[["Store","StoreType","Assortment","Competitio
nDistance","Promo2","PromoInterval","DayOfWeek","Sales",
"Customers","Open","Promo","isStateHoliday","SchoolHoliday",
"Week_of_Date","Day_of_Date","Month_of_Date","Year_of_Date","Promo2Tot
alWeek","CompetitionOpenTotalMonth"]]

final_train=merged_train[["Store","StoreType","Assortment","Competitio
nDistance","DayOfWeek","Sales",
"Customers","Open","Promo","Day_of_Date","Month_of_Date","Year_of_Date
","Promo2TotalWeek","CompetitionOpenTotalMonth"]]
X = final_train.drop('Sales', axis=1)
y = final_train['Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

train = xgb.DMatrix(X_train, label=y_train)
test = xgb.DMatrix(X_test, label=y_test)

params = {
    'objective': 'reg:squarederror',
    'colsample_bytree': 0.8,
    'learning_rate': 0.1,
    'max_depth': 5,
    'alpha': 10
}


model = xgb.train(params, train, num_boost_round=100)

y_pred = model.predict(test)

epsilon = 1e-10
rmspe = np.sqrt(np.mean(((y_test - y_pred) / (y_test + epsilon)) **
2)) * 100

print(f'RMSPE: {rmspe:.2f}%')
```

```
RMSPE: 48417397726827.98%
```

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales for XGBoost')
plt.show()
```


Actual vs. Predicted Sales for XGBoost
Predicted Sales
Actual Sales
35000
30000
25000
20000
15000
10000
5000
0
0
5000
10000
15000
20000
25000
30000
35000
40000

# EDA

```
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns

merged_train=pd.read_csv("merged_train.csv")
store=pd.read_csv("store.csv")

merged_train.head(10)
```

```
   Store  StoreType  Assortment  CompetitionDistance  \
0      1          2           0                 1270
1      1          2           0                 1270
2      1          2           0                 1270
3      1          2           0                 1270
4      1          2           0                 1270
5      1          2           0                 1270
6      1          2           0                 1270
7      1          2           0                 1270
8      1          2           0                 1270
9      1          2           0                 1270

   CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                          9                      2008       0
1                          9                      2008       0
2                          9                      2008       0
3                          9                      2008       0
4                          9                      2008       0
5                          9                      2008       0
6                          9                      2008       0
7                          9                      2008       0
8                          9                      2008       0
9                          9                      2008       0

   Promo2SinceWeek  Promo2SinceYear  PromoInterval  ...  Promo
StateHoliday  \
0              NaN              NaN              0  ...      1
0
1              NaN              NaN              0  ...      1
0
2              NaN              NaN              0  ...      1
0
3              NaN              NaN              0  ...      1
0
4              NaN              NaN              0  ...      1
0
```

```
5               NaN               NaN               0  ...       0
0
6               NaN               NaN               0  ...       0
0
7               NaN               NaN               0  ...       0
0
8               NaN               NaN               0  ...       0
0
9               NaN               NaN               0  ...       0
0

   SchoolHoliday  Week_of_Date  Day_of_Date  Month_of_Date
Year_of_Date  \
0              1            31           31              7
2015
1              1            31           30              7
2015
2              1            31           29              7
2015
3              1            31           28              7
2015
4              1            31           27              7
2015
5              0            30           26              7
2015
6              0            30           25              7
2015
7              0            30           24              7
2015
8              0            30           23              7
2015
9              0            30           22              7
2015

   Promo2TotalWeek  CompetitionOpenTotalMonth  isStateHoliday
0                0                         82               0
1                0                         82               0
2                0                         82               0
3                0                         82               0
4                0                         82               0
5                0                         82               0
6                0                         82               0
7                0                         82               0
8                0                         82               0
9                0                         82               0

[10 rows x 25 columns]
```

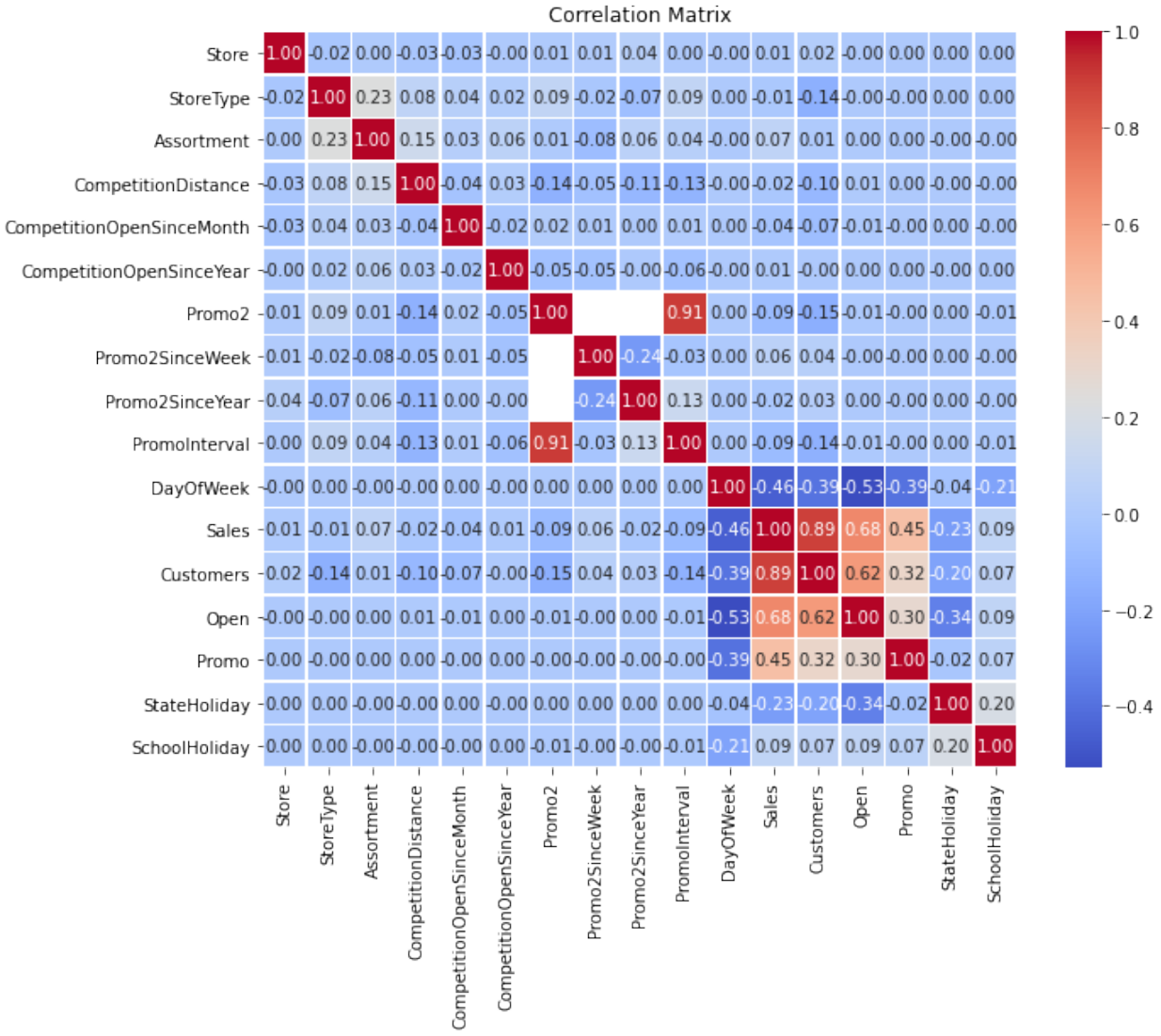
```
cols=["Store","StoreType","Assortment","CompetitionDistance","Competit
ionOpenSinceMonth","CompetitionOpenSinceYear","Promo2","Promo2SinceWee
```

```
k","Promo2SinceYear","PromoInterval","DayOfWeek","Sales","Customers","
Open","Promo","StateHoliday","SchoolHoliday"]
eda_df=merged_train[cols]
eda_df.head(10)
```

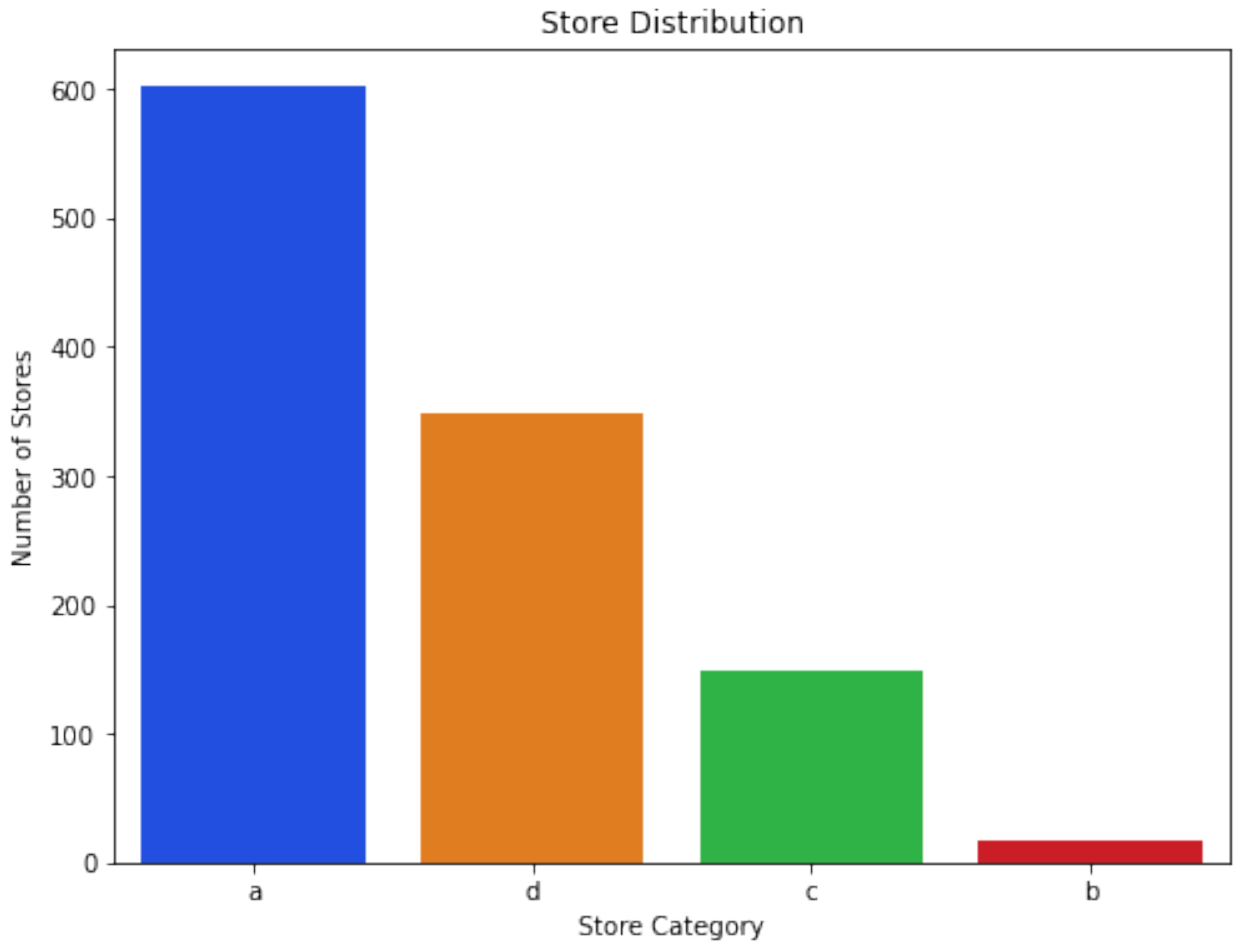
```
   Store  StoreType  Assortment  CompetitionDistance  \
0      1          2           0                 1270
1      1          2           0                 1270
2      1          2           0                 1270
3      1          2           0                 1270
4      1          2           0                 1270
5      1          2           0                 1270
6      1          2           0                 1270
7      1          2           0                 1270
8      1          2           0                 1270
9      1          2           0                 1270

   CompetitionOpenSinceMonth  CompetitionOpenSinceYear  Promo2  \
0                          9                      2008       0
1                          9                      2008       0
2                          9                      2008       0
3                          9                      2008       0
4                          9                      2008       0
5                          9                      2008       0
6                          9                      2008       0
7                          9                      2008       0
8                          9                      2008       0
9                          9                      2008       0

   Promo2SinceWeek  Promo2SinceYear  PromoInterval  DayOfWeek
Sales  \
0              NaN              NaN              0          5   5263

1              NaN              NaN              0          4   5020

2              NaN              NaN              0          3   4782

3              NaN              NaN              0          2   5011

4              NaN              NaN              0          1   6102

5              NaN              NaN              0          7      0

6              NaN              NaN              0          6   4364

7              NaN              NaN              0          5   3706

8              NaN              NaN              0          4   3769

9              NaN              NaN              0          3   3464
```

```
   Customers  Open  Promo  StateHoliday  SchoolHoliday
0        555     1      1             0              1
1        546     1      1             0              1
2        523     1      1             0              1
3        560     1      1             0              1
4        612     1      1             0              1
5          0     0      0             0              0
6        500     1      0             0              0
7        459     1      0             0              0
8        503     1      0             0              0
9        463     1      0             0              0
```

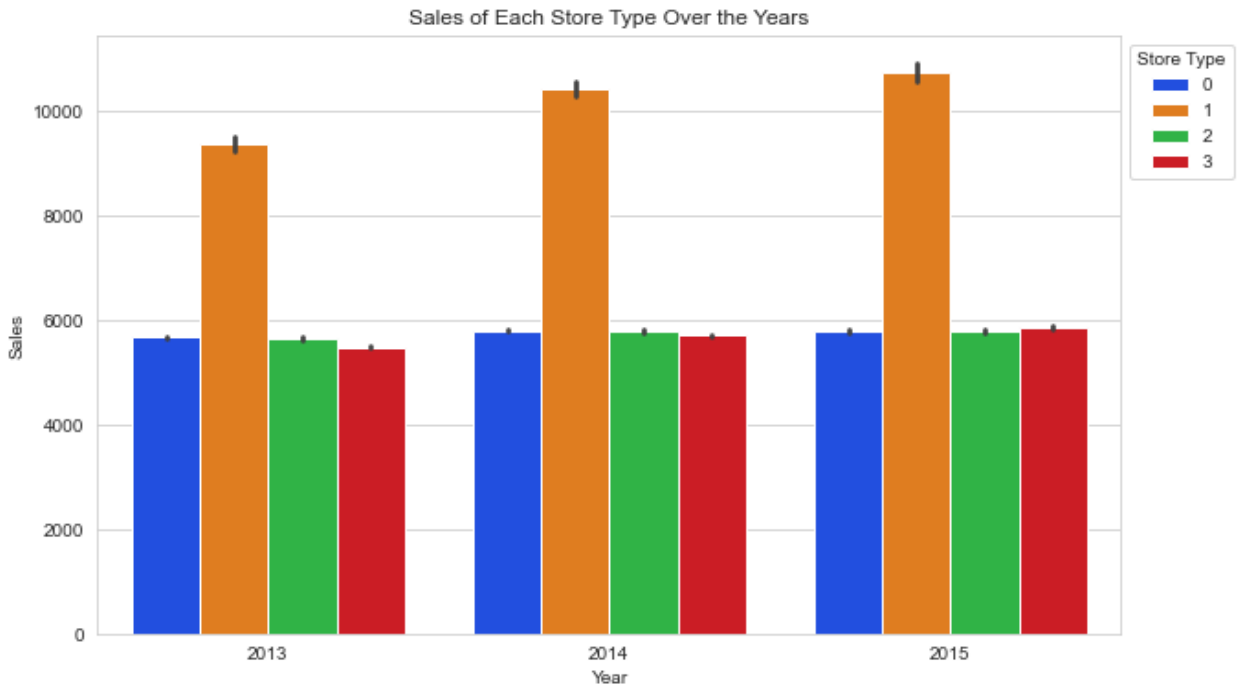
```
corr_matrix = eda_df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.title("Correlation Matrix")
plt.show()
```



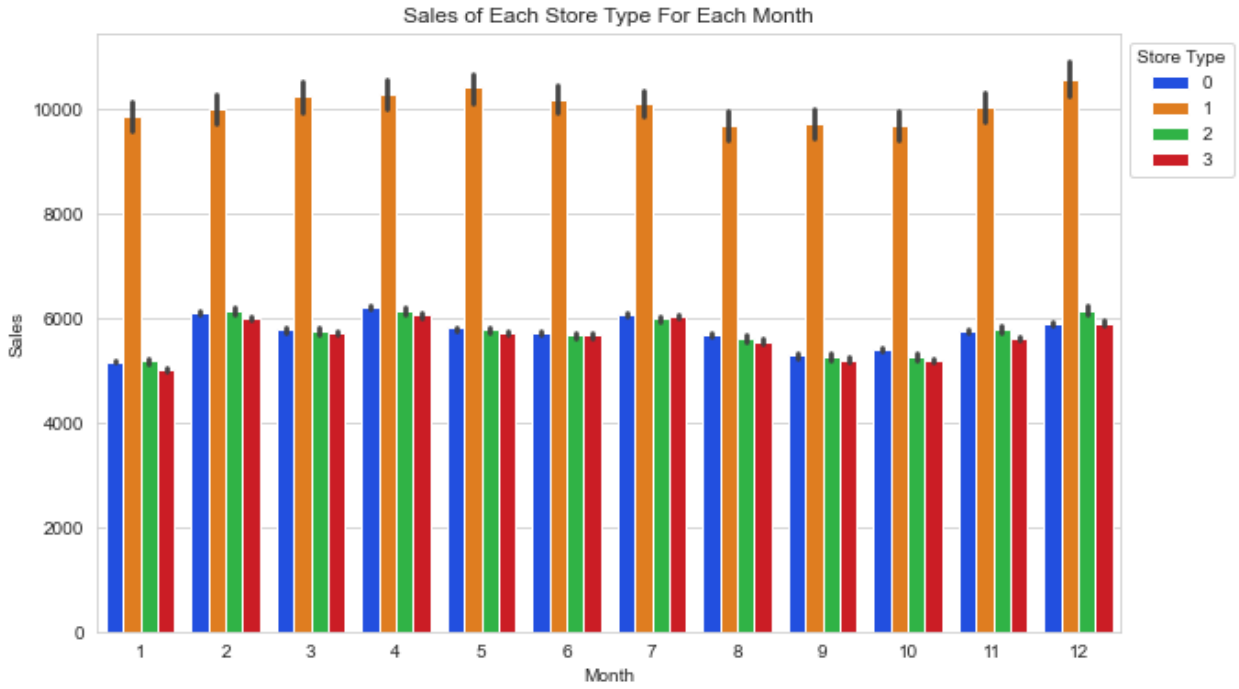

```
store = store['StoreType'].value_counts().reset_index()

plt.figure(figsize=(8, 6))
sns.barplot(x='index', y='StoreType', data=store, palette='bright')
plt.title('Store Distribution')
plt.xlabel('Store Category')
plt.ylabel('Number of Stores')
plt.show()
```



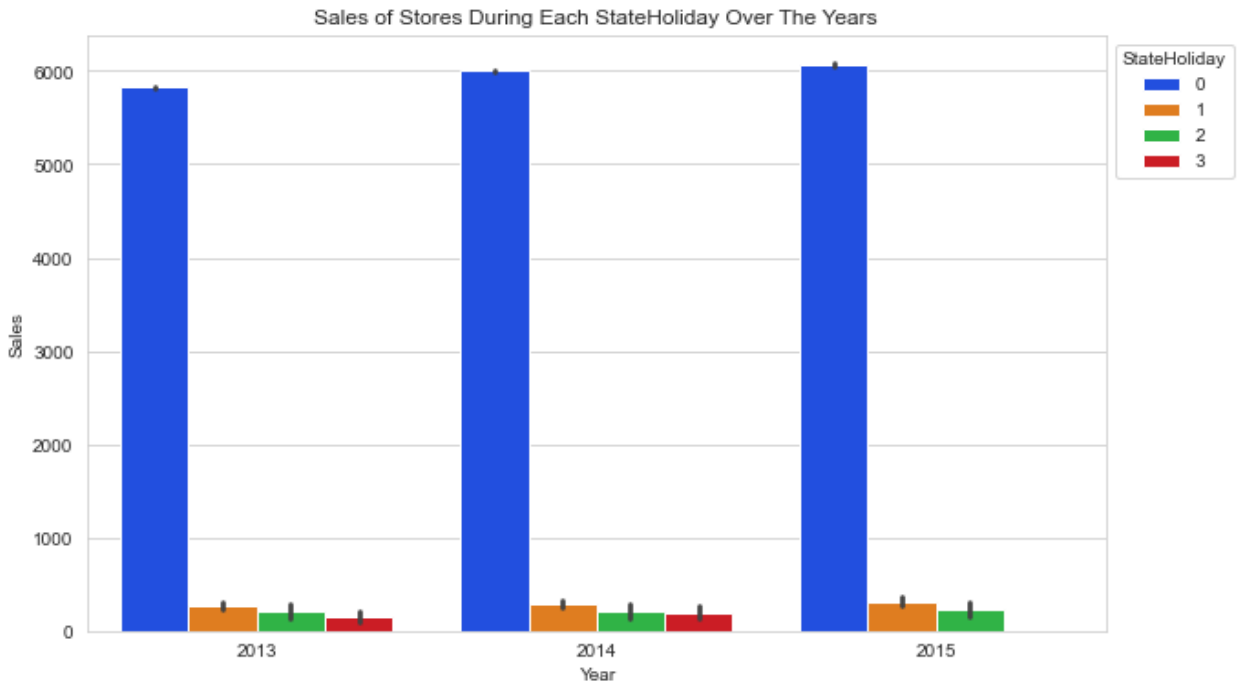

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Year_of_Date', y='Sales', hue='StoreType',
data=merged_train[['Year_of_Date', 'StoreType', 'Sales']],
palette=sns.color_palette("bright"))
plt.title('Sales of Each Store Type Over the Years')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.legend(title='Store Type', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



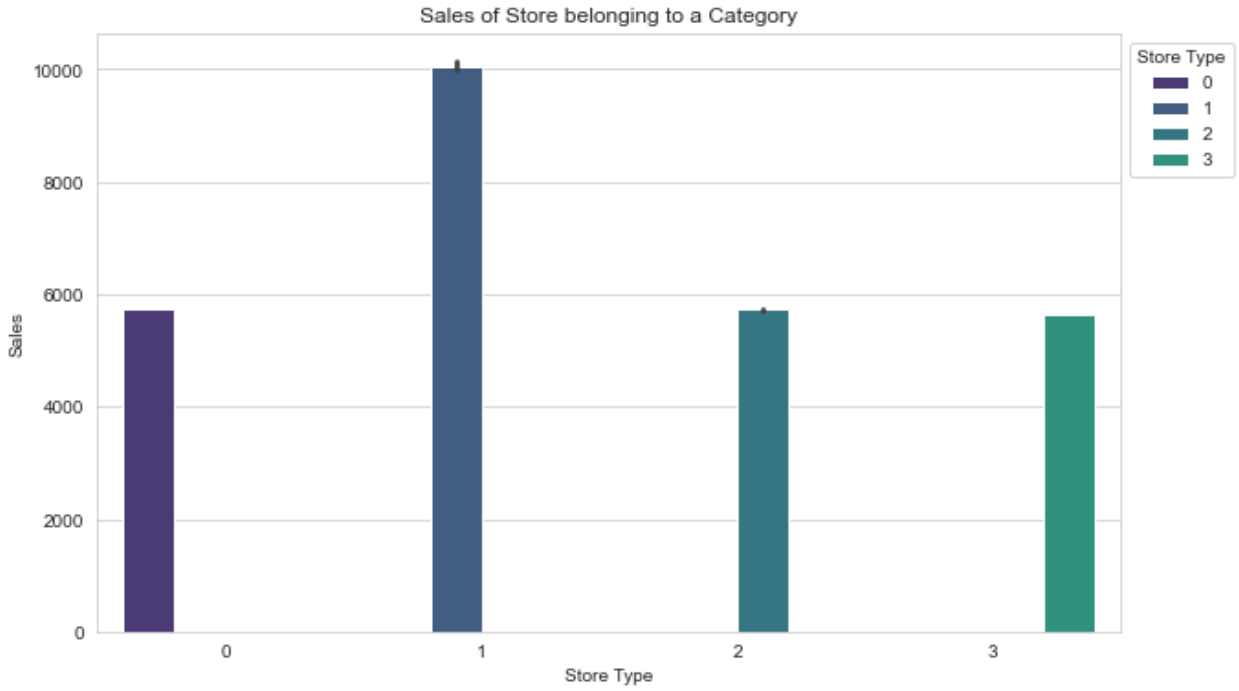

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Month_of_Date', y='Sales', hue='StoreType',
data=merged_train[['Month_of_Date', 'StoreType', 'Sales']],
palette=sns.color_palette("bright"))
plt.title('Sales of Each Store Type For Each Month')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend(title='Store Type', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



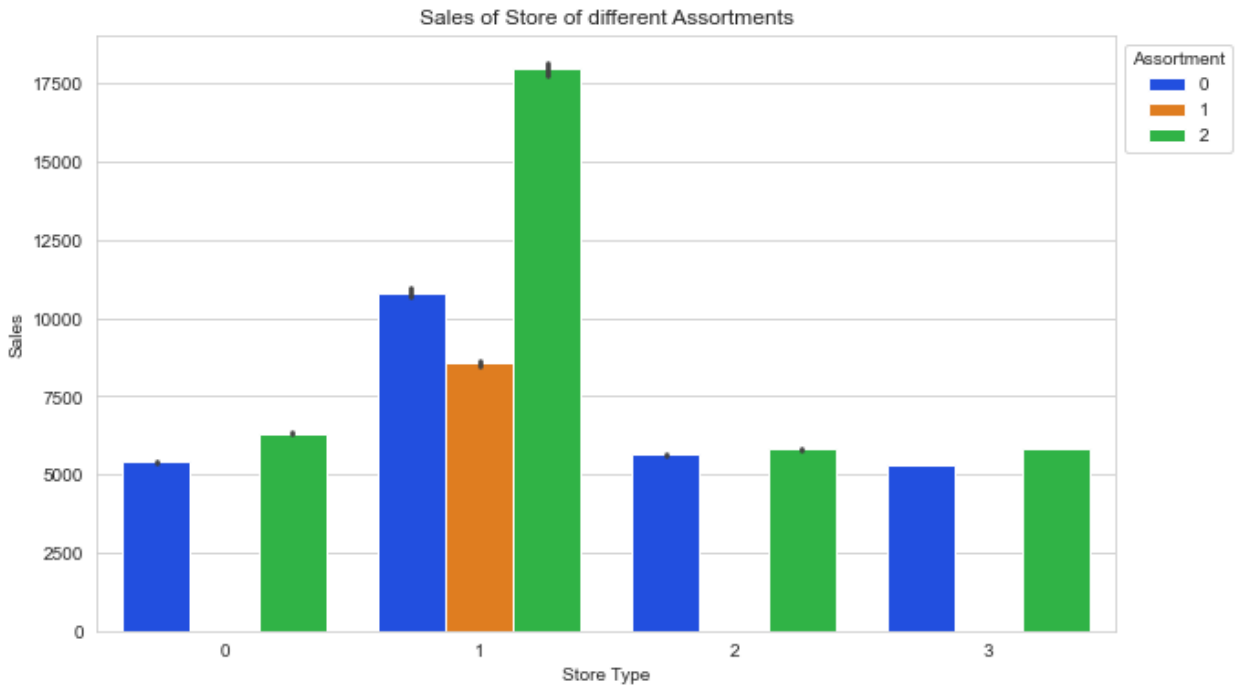

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Year_of_Date', y='Sales', hue='StateHoliday',
data=merged_train[['Year_of_Date', 'StateHoliday' ,'Sales']],
palette=sns.color_palette("bright"))
plt.title('Sales of Stores During Each StateHoliday Over The Years')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.legend(title='StateHoliday', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



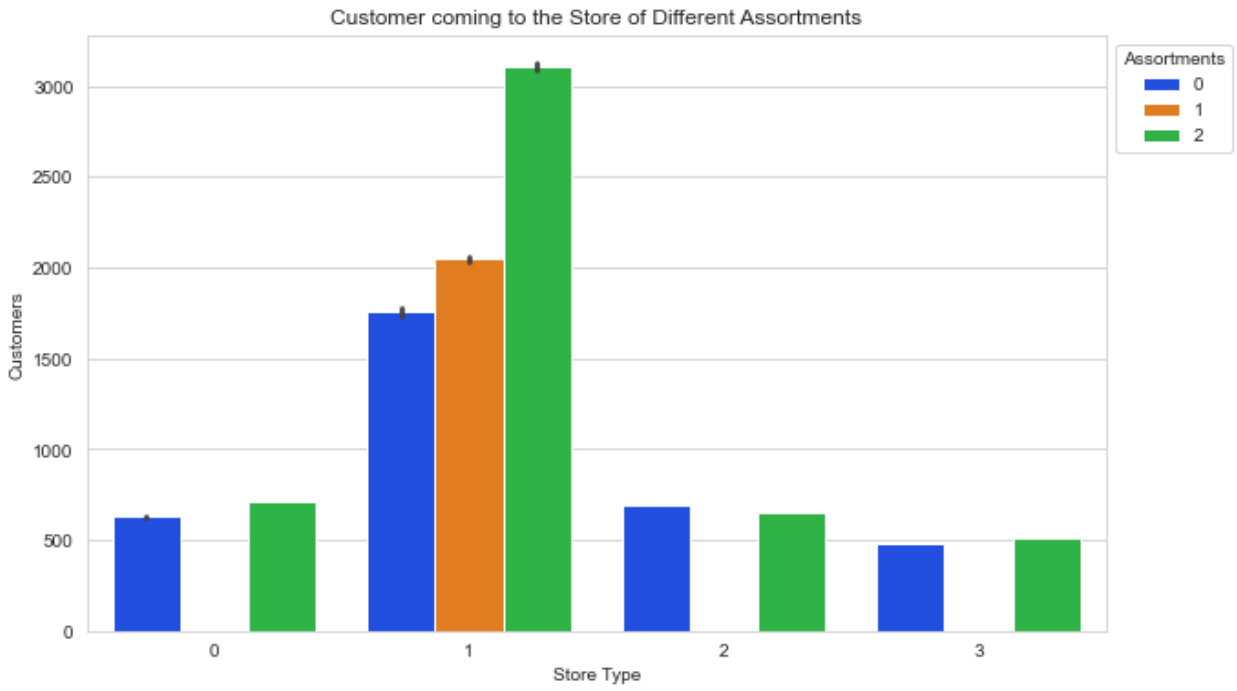

```
plt.figure(figsize=(10, 6))
sns.barplot(x='StoreType', y='Sales', hue='StoreType',
data=merged_train[['StoreType','Sales']],
palette=sns.color_palette("viridis"))
plt.title('Sales of Store belonging to a Category')
plt.xlabel('Store Type')
plt.ylabel('Sales')
plt.legend(title='Store Type', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



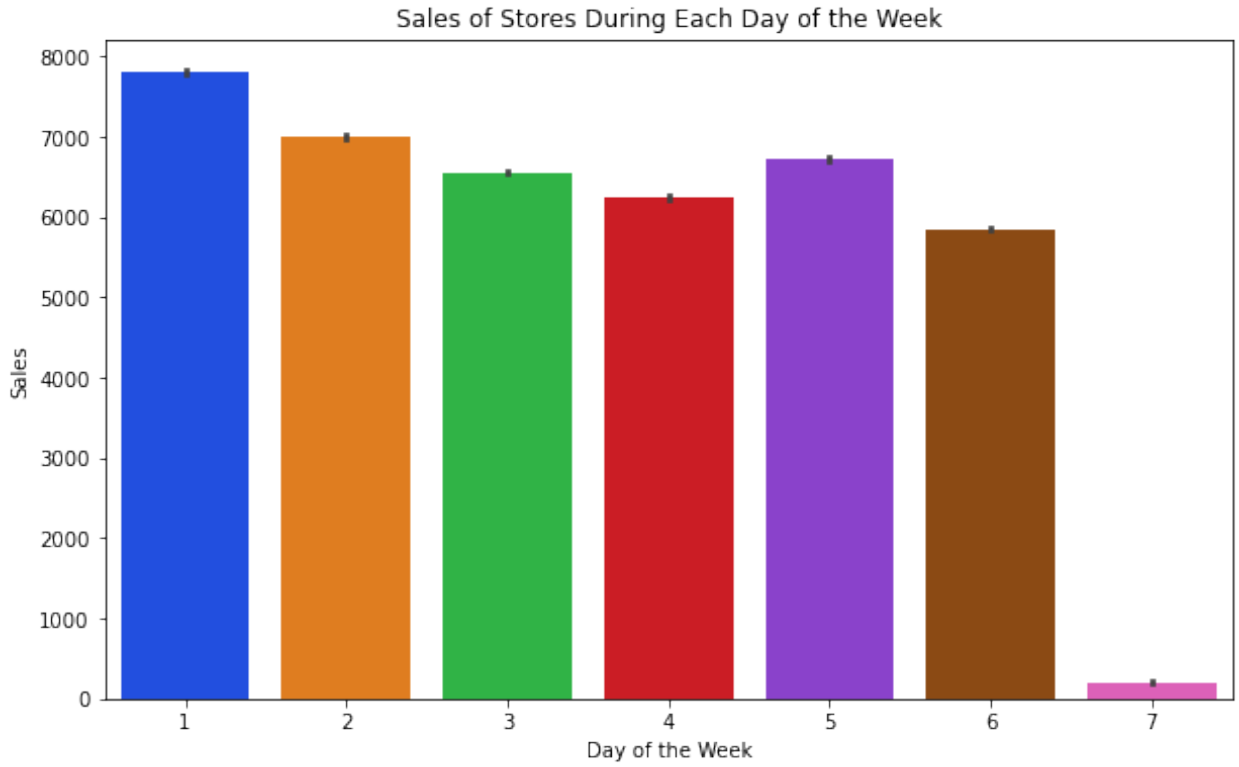

```
plt.figure(figsize=(10, 6))
sns.barplot(x='StoreType', y='Sales', hue='Assortment',
data=merged_train[['StoreType','Assortment','Sales']],
palette=sns.color_palette("bright"))
plt.title('Sales of Store of different Assortments')
plt.xlabel('Store Type')
plt.ylabel('Sales')
plt.legend(title='Assortment', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



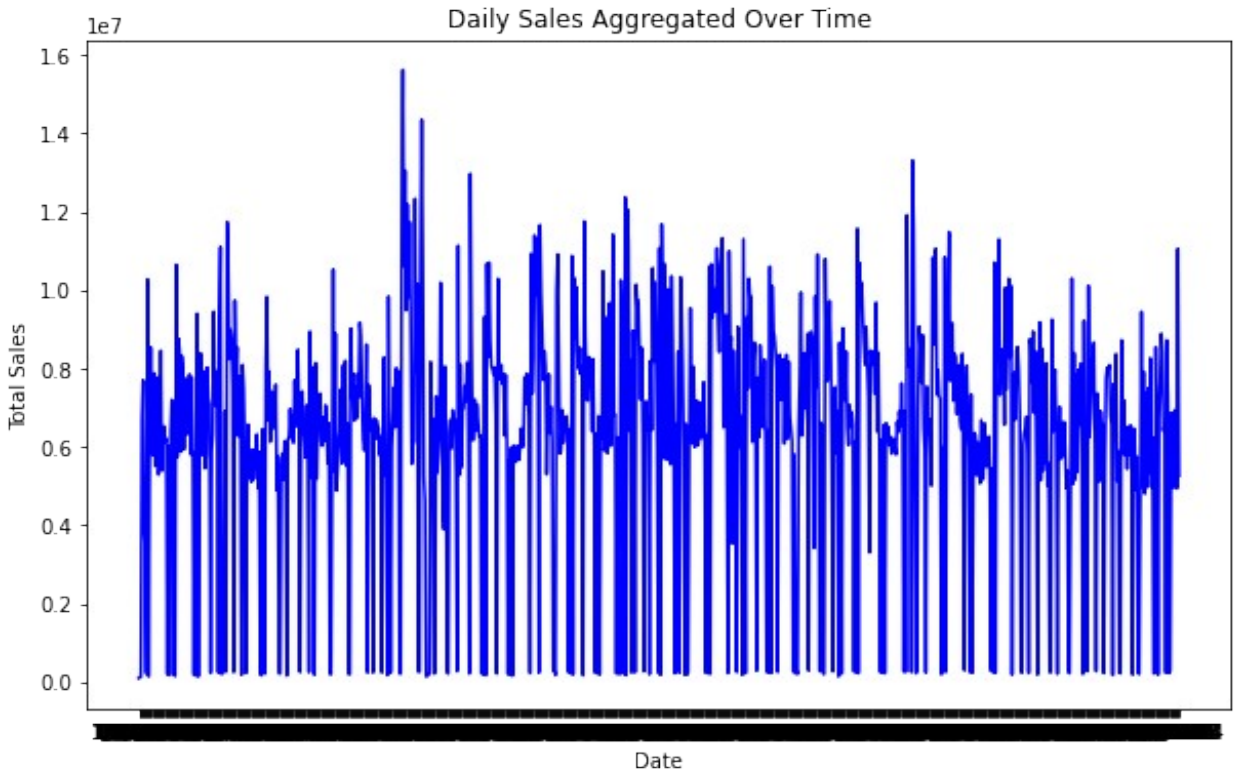

```
plt.figure(figsize=(10, 6))
sns.barplot(x='StoreType', y='Customers', hue='Assortment',
data=merged_train[['StoreType','Assortment','Customers']],
palette=sns.color_palette("bright"))
plt.title('Customer coming to the Store of Different Assortments')
plt.xlabel('Store Type')
plt.ylabel('Customers')
plt.legend(title='Assortments', loc='upper left', bbox_to_anchor=(1,
1))
plt.show()
```



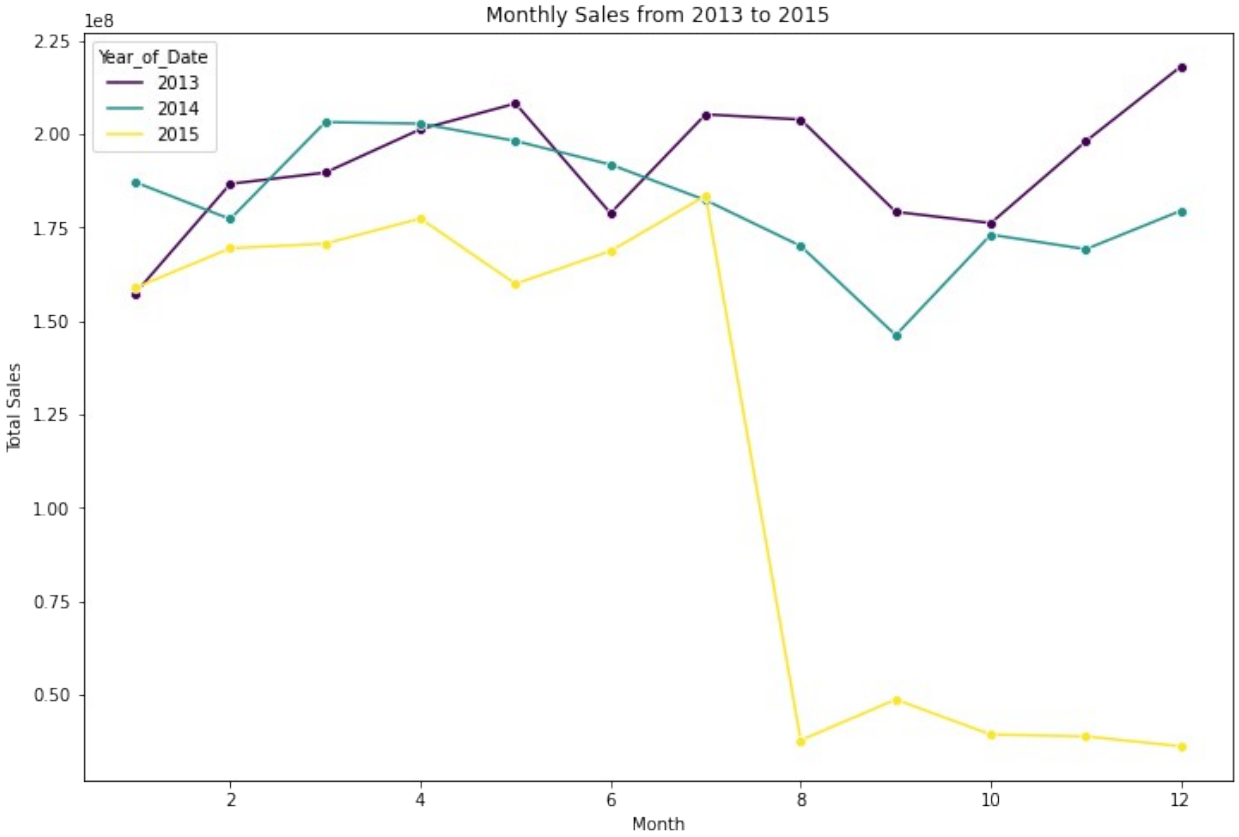

```
plt.figure(figsize=(10, 6))
sns.barplot(x='DayOfWeek', y='Sales',
data=merged_train[['DayOfWeek','Sales']],
palette=sns.color_palette("bright"))
plt.title('Sales of Stores During Each Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Sales')
plt.show()
```



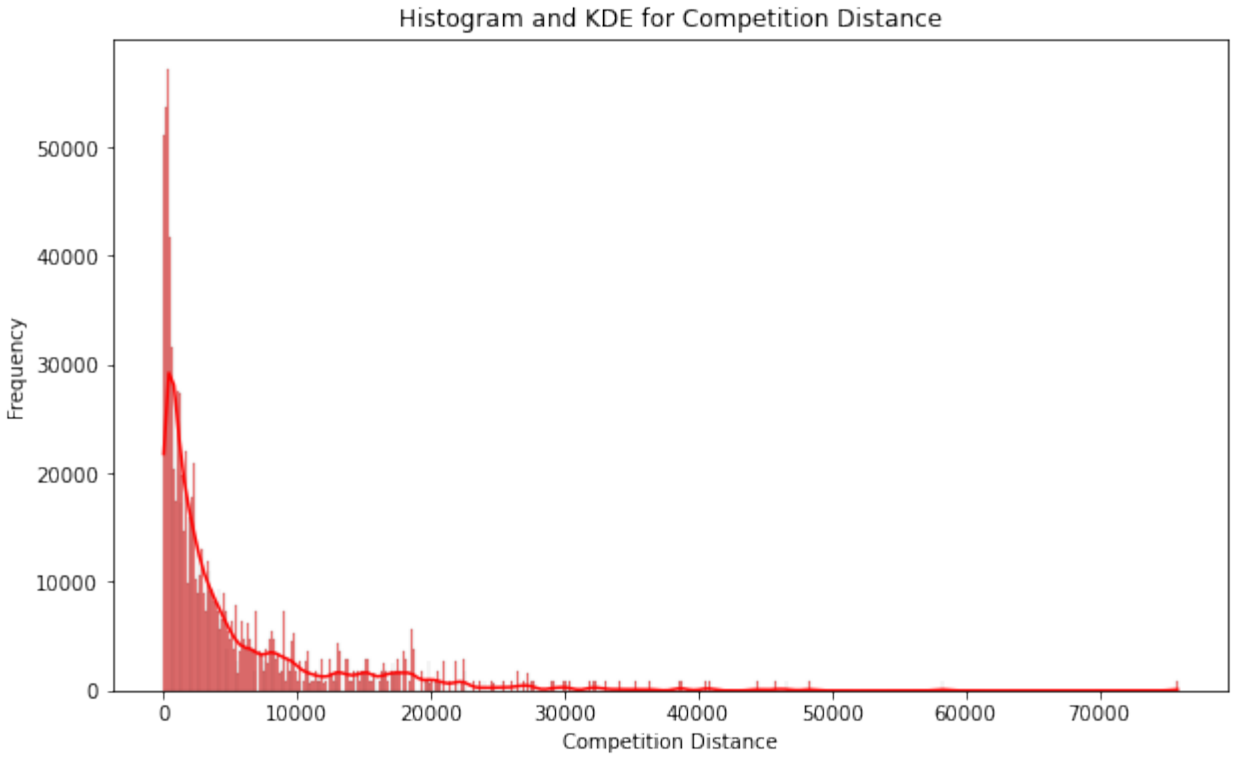

```
sales = merged_train.groupby('Date')['Sales'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.lineplot(x='Date', y='Sales', data=sales, color='b')
plt.title('Daily Sales Aggregated Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')

plt.show()
```




```
month_sales = merged_train.groupby(['Year_of_Date', 'Month_of_Date'])
['Sales'].sum().reset_index()

plt.figure(figsize=(12, 8))
sns.lineplot(x='Month_of_Date', y='Sales', hue='Year_of_Date',
data=month_sales, marker='o', palette='viridis')
plt.title('Monthly Sales from 2013 to 2015')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.histplot(merged_train['CompetitionDistance'], kde=True,
color='red')
plt.title('Histogram and KDE for Competition Distance')
plt.xlabel('Competition Distance')
plt.ylabel('Frequency')
plt.show()
```


Histogram and KDE for Competition Distance
Frequency
50000
40000
30000
20000
10000
0
0
10000
20000
30000
40000
50000
60000
70000
Competition Distance