EDA summary - Group 5

Emily Ye zy367@drexel.edu
Junkai Ge jg3944@drexel.edu
Jerry Li jl4533@drexel.edu
Shengyang Dong sd3666@drexel.edu

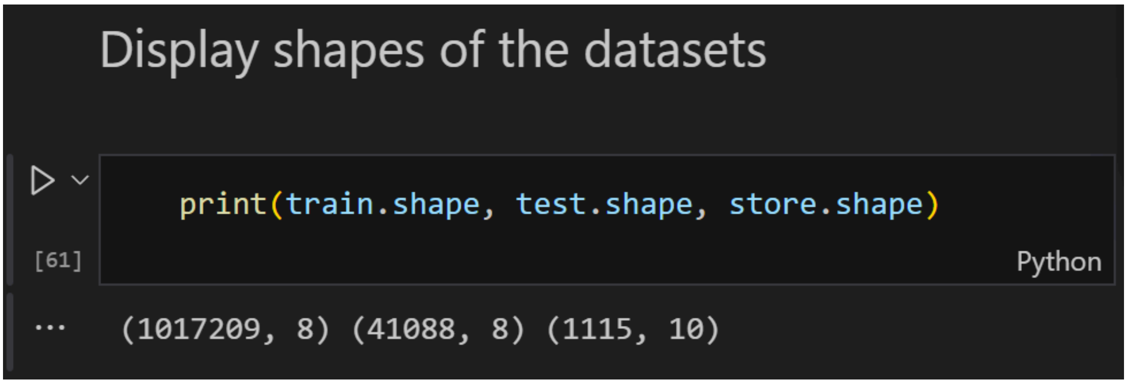
The objective of this exploratory data analysis (EDA) is to predict daily sales for Rossmann, a company that operates over 3,000 drug stores in seven European countries. The challenge is to forecast sales for up to six weeks in advance by understanding various sales patterns and identifying key factors influencing sales.

Steps in EDA

Dataset Overview

The dataset used is the Rossmann store data, which includes sales information from January 2013 to 2015.

The initial step involved displaying the shapes of the datasets to understand their dimensions. This included the training, test, and store datasets.



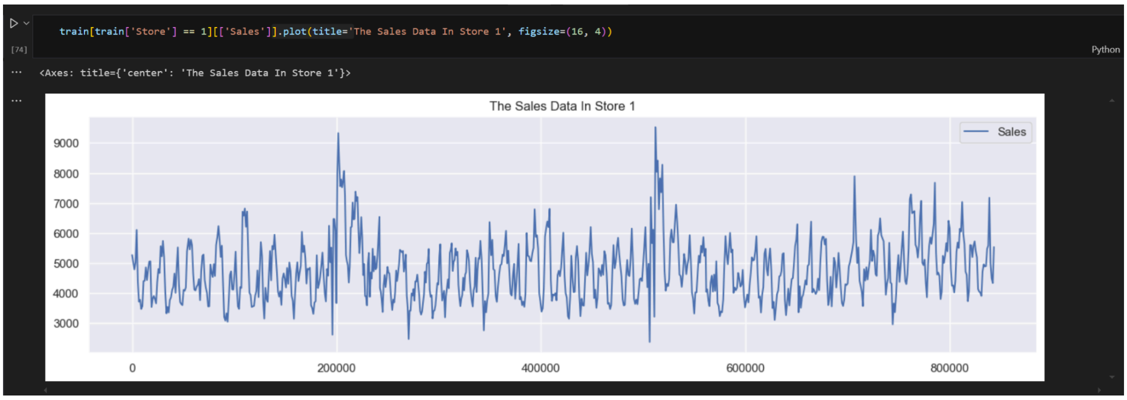
Display shapes of the datasets

```
print(train.shape, test.shape, store.shape)
```

```
(1017209, 8) (41088, 8) (1115, 10)
```

Sales Data Visulization

To get a visual sense of the overall sales changes over time, we plotted the sales data for the store with serial number 1. This helped in understanding the trend and seasonal patterns in the sales data.



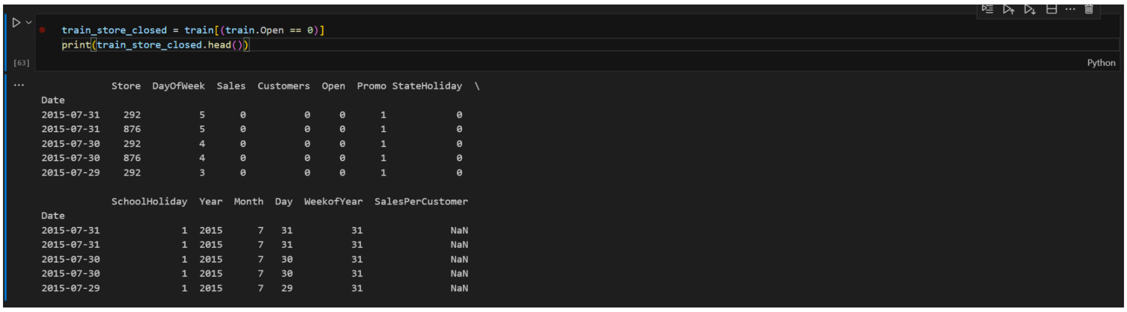
```
train[train['Store'] == 1][['Sales']].plot(title='The Sales Data In Store 1', figsize=(16, 4))
```

```
<Axes: title={'center': 'The Sales Data In Store 1'}>
```

Store Closure Analysis

We examined the data to identify when the stores were closed. This involved plotting histograms for the days when the store was closed.

The analysis revealed that stores were predominantly closed on Sundays.



```
train_store_closed = train[(train.Open == 0)]
print(train_store_closed.head())
```

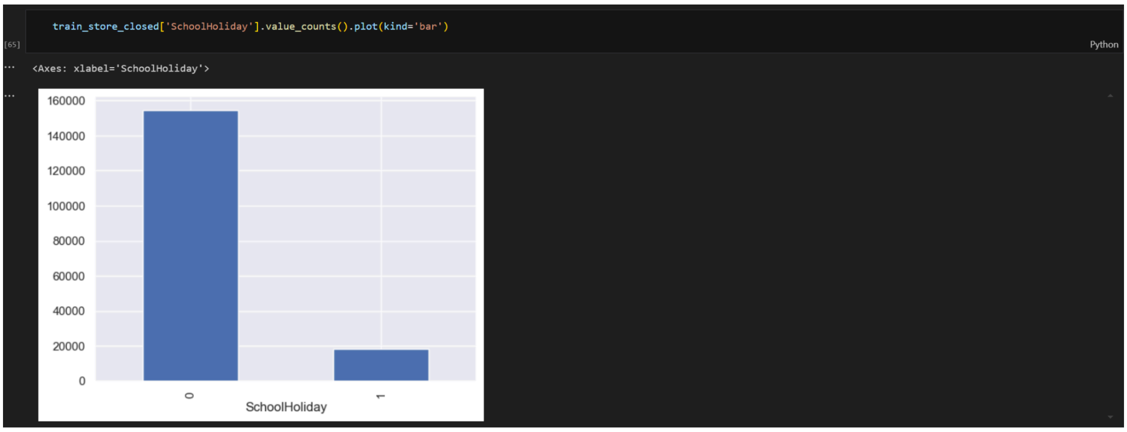
```
            Store  DayOfWeek  Sales  Customers  Open  Promo StateHoliday  \
Date
2015-07-31    292          5      0          0     0      1            0
2015-07-31    876          5      0          0     0      1            0
2015-07-30    292          4      0          0     0      1            0
2015-07-30    876          4      0          0     0      1            0
2015-07-29    292          3      0          0     0      1            0

            SchoolHoliday  Year  Month  Day  WeekofYear  SalesPerCustomer
Date
2015-07-31              1  2015      7   31          31               NaN
2015-07-31              1  2015      7   31          31               NaN
2015-07-30              1  2015      7   30          31               NaN
2015-07-30              1  2015      7   30          31               NaN
2015-07-29              1  2015      7   29          31               NaN
```

School Holidays

We analyzed the data to determine whether the stores were closed due to school holidays. This involved checking the distribution of school holidays and plotting their frequency.

The holidays were categorized as: '1' indicating the school was closed and '0' indicating it was not.



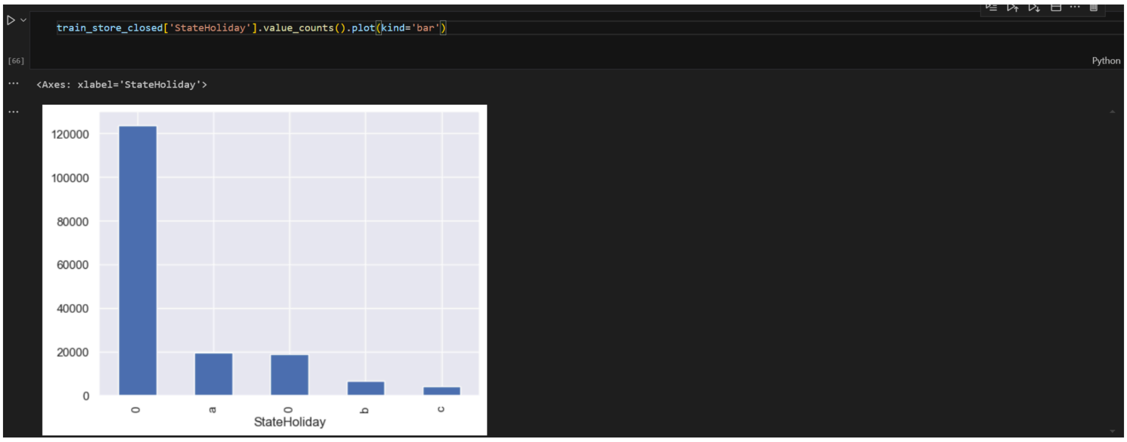
```
train_store_closed['SchoolHoliday'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='SchoolHoliday'>
```

State Holidays

The analysis included checking the impact of state holidays on store operations. The state holidays were categorized into public holidays (a), Easter holidays (b), Christmas (c), and none (0).

We plotted the frequency of each type of state holiday to understand their distribution.



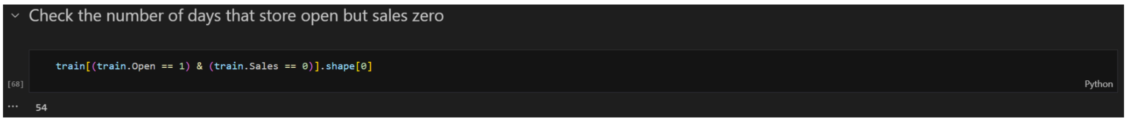
```
train_store_closed['StateHoliday'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='StateHoliday'>
```

Sales Analysis on Open Days

We calculated and plotted the number of days when stores were closed.

Additionally, we checked the number of days the store was open but had zero sales to understand anomalies in sales data.



Check the number of days that store open but sales zero

```
train[(train.Open == 1) & (train.Sales == 0)].shape[0]
```

```
54
```

Feature Engineering

Several new features were created to enhance the analysis:

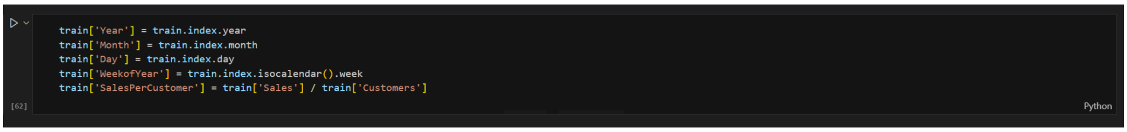
'Year': Extracted from the date index.

'Month': Extracted from the date index.

'Day': Extracted from the date index.

'WeekofYear': Extracted from the date index.

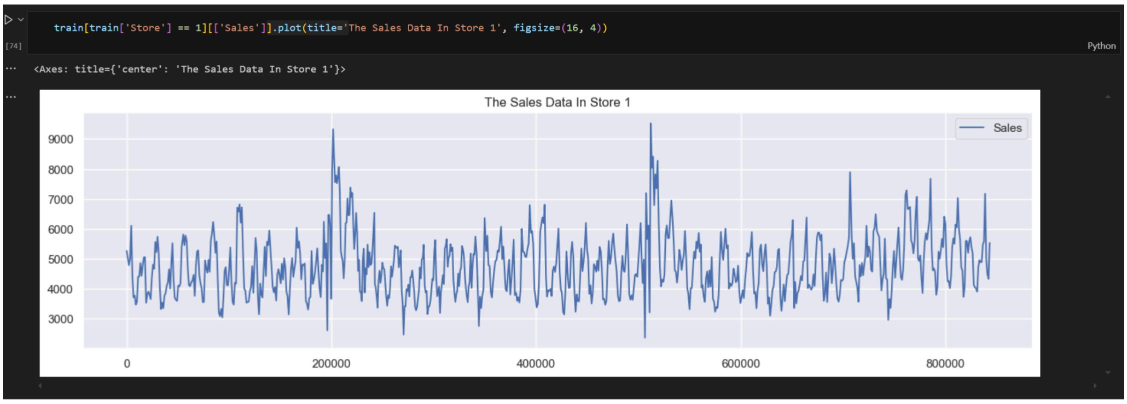
'SalesPerCustomer': Calculated by dividing the 'Sales' by the number of 'Customers'.



```
train['Year'] = train.index.year
train['Month'] = train.index.month
train['Day'] = train.index.day
train['WeekofYear'] = train.index.isocalendar().week
train['SalesPerCustomer'] = train['Sales'] / train['Customers']
```

Line Graph of Sales Data:

We plotted a line graph to visualize the trend in sales over time for a specific store.

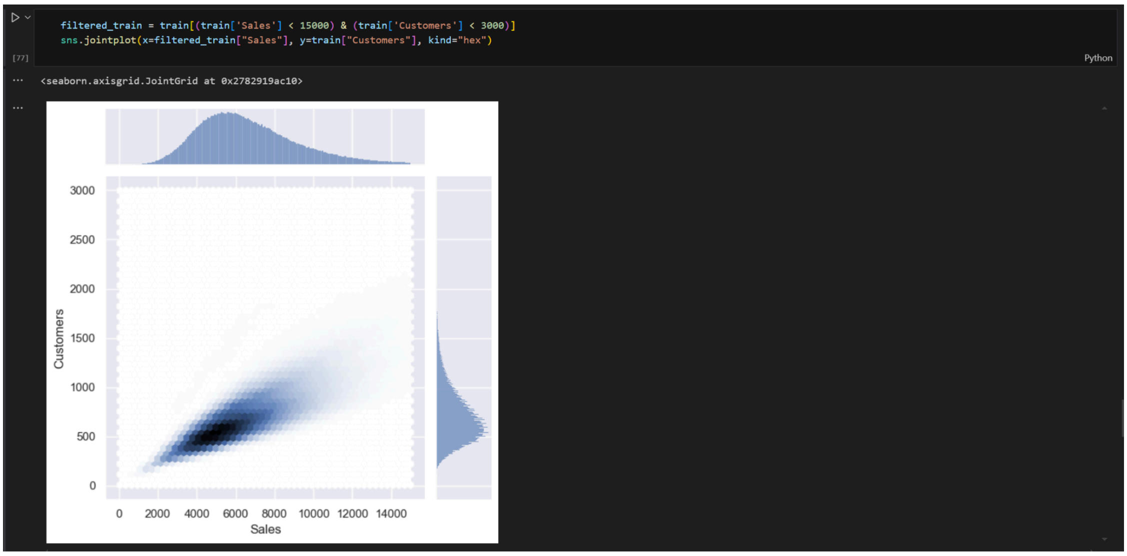


```
train[train['Store'] == 1][['Sales']].plot(title='The Sales Data In Store 1', figsize=(16, 4))
```

```
<Axes: title={'center': 'The Sales Data In Store 1'}>
```

Joint Distribution Plots

To analyze the relationship between different features, we used joint distribution plots. This helped in understanding the correlation between variables like 'Sales' and 'Customers'.

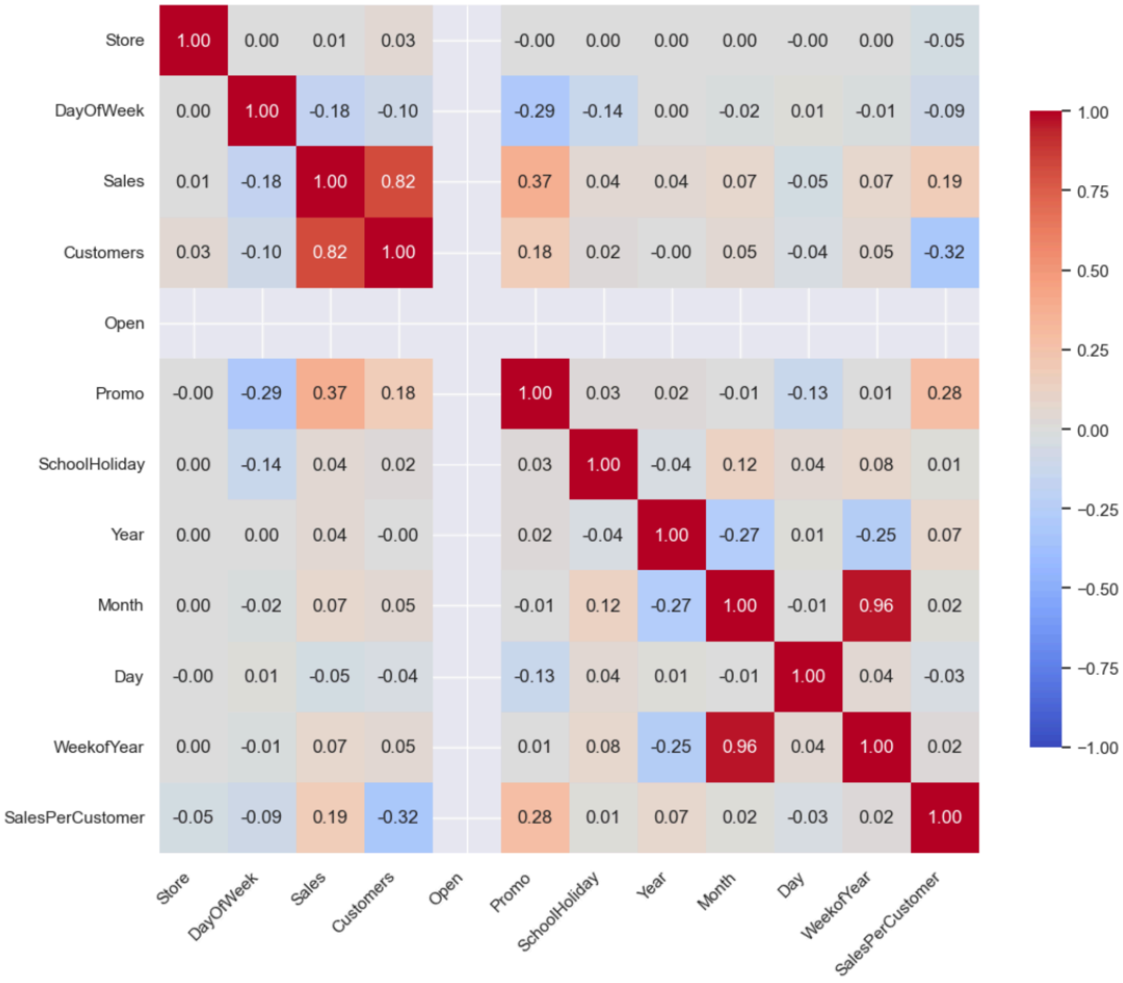


```
filtered_train = train[(train['Sales'] < 15000) & (train['Customers'] < 3000)]
sns.jointplot(x=filtered_train["Sales"], y=train["Customers"], kind="hex")
```

```
<seaborn.axisgrid.JointGrid at 0x2782919ac10>
```

Heatmap of Correlations

We created a heatmap to visualize the correlations between different numerical features in the dataset. This provided insights into the strength and direction of relationships between variables.



Summary

The EDA process provided valuable insights into the patterns of store closures and their impact on sales, as well as the influence of holidays on sales trends. The visualizations, including line graphs, joint distribution plots, and heatmaps, helped in understanding the data's underlying structure. The feature engineering steps created new dimensions for analysis, which will be crucial for the predictive modeling phase.