Rossmann Report

Student ID: 10724837

Introduction

The quality of the Rossmann dataset is decent. The data given is certainly enough to make useful conclusions about store performance. The main issues are some missing records between 2014-07-01 and 2014-12-31, and a considerable number of null values in certain columns. Some additional context would have been useful as well. For example, it is not clear what the different store types and assortments really represent. Caution must also be taken in noting that there is a difference between 'Promo' (in-store on-the-day promo schemes) and 'Promo2' (continuous schemes).

Initial Pre-Processing

The goal in this section is to prepare the datasets for EDA and graph visualisation. Since the train and test datasets have similar structures, they are pre-processed simultaneously, whereas the store dataset is pre-processed separately. Throughout this project, it was decided to keep dates in string form, not datetime, because we do not use time series analysis. At the time, it was believed that much of the information in the date columns could be captured by creating new variables. In hindsight, using dates would have led to better visualisations and model performance.

The Store Dataset

The 3 nulls in the CompetitionDistance column are mean imputed. This is acceptable because it is such a small fraction of the total data. However, there was a much greater proportion of nulls in the CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear and PromoInterval columns. Due to the discrete nature of these columns and the quantity of these nulls, there was no means of imputing these without making dramatic assumptions. Thus, these nulls were kept.

We then converted the CompetitionOpenSinceMonth and CompetitionOpenSinceYear columns into a single column, CompetitionSinceDate, which captures this information more concisely. The same was repeated with the respective Promo2 columns, ensuring Promo2SinceWeek was converted to months beforehand.

Two new features were created, MonthsSinceCompetition and MonthsSincePromo2. These new variables capture whether having competition stores nearby or running continuous promos for longer periods of time affects store performance. In order to create these, an assumption was made that the current date was 17/09/2015, as this was the last date to appear in any of the datasets.

PromoInterval was removed because every store running continuous promos refreshes them every 3 months. The only insight this column may have provided is whether the timing of these refreshes affects sales.

Upon observation of the distribution of the StoreType and Assortment columns, there are very few stores of StoreType ‘b’ and Assortment ‘b’. The sales and customers for these are considerably higher in comparison to their counterparts. While we expect such stores to perform better, they may be more sensitive to fluctuations compared to other predictors in our model.

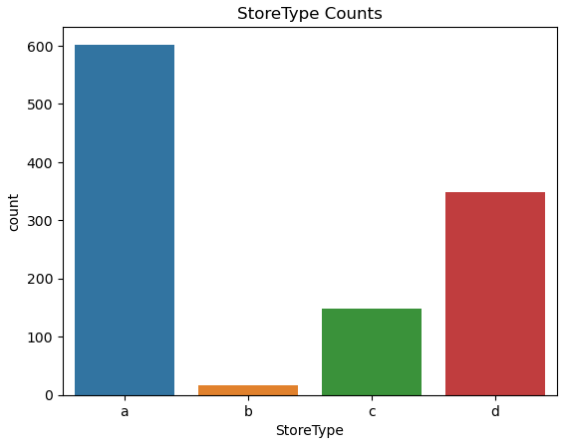

Figure 1: Distribution of store types.

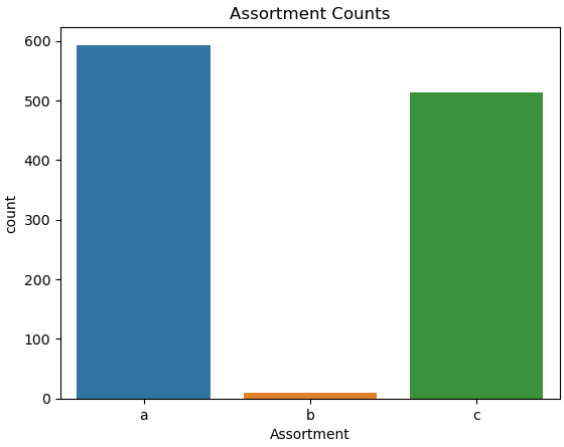

Figure 2: Distribution of assortments.

The Train and Test Datasets

There are only 11 nulls in the Open column of the test dataset. By observing the relationship between DayOfWeek and Open, we see that stores overwhelmingly tend to be open Monday to Saturday, then closed on Sundays. We impute these nulls according to this condition.

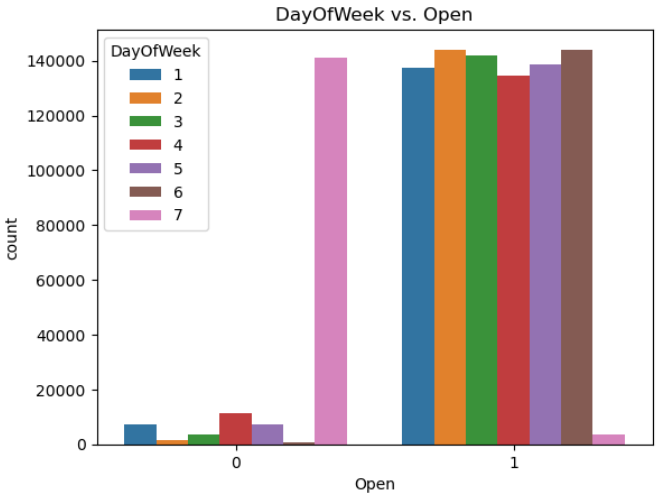

Figure 3: Open vs. closed stores on different weekdays.

181 stores have missing records within the range 2014-07-01 to 2014-12-31. Since we do not know the reason for these missing values, and since so much data is missing, it is difficult to impute these. They have been left as is. Regardless, in our upcoming analysis, we look more at averages of sales and customers across all stores, so these missing records will have little effect.

When looking at records where a store is closed, the sales and customer numbers are always 0. When predicting values later, any records in the test dataset with Open equal to 0 can be assigned 0 sales and customers accordingly. Such records in the train dataset will be of no use when training our model, so we can drop them, along with the Open column.

From the Date column, we can create Month and Year columns. The Month column will be useful because we can observe seasonal changes, as well as how important times of year, such as Christmas, will impact store performances. The Year variable will be less representative because the data we have only spans around two and a half years. However, the average number of sales and customers increases steadily within each of these years. Given that all the records in the test dataset are in 2015, their sales and customer numbers should be slightly higher than average to reflect this.

The StateHoliday columns contain mixtures of strings ('0', 'a', 'b', 'c') as well as integers (0). Change all integer 0s to string '0's for consistency before creating graphs. Within the test dataset, there are no records with 'b' (Easter) or 'c' (Christmas) values. Using such records to train the model, when we do not have any in the test dataset, may hurt performance. At this point, we can either drop all 'b' and 'c' records from the training dataset, or we can change all state holidays to a single binary value 1.

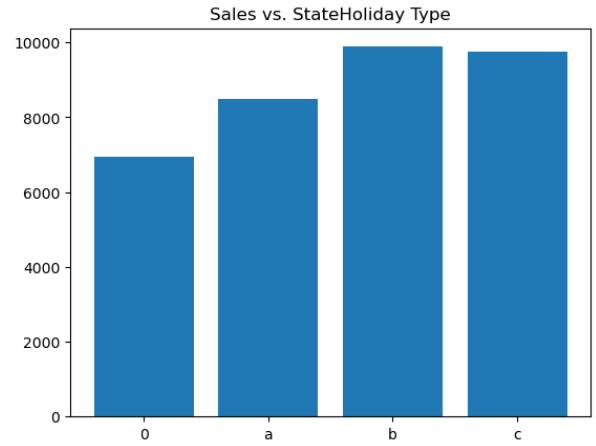

Figure 4: Sales on different state holidays.

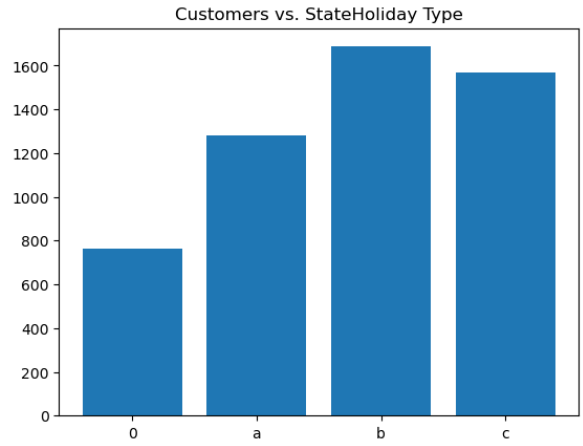

Figure 5: Customers on different state holidays.

We can see that 'a' holidays perform differently to 'b' and 'c' holidays. Hence, we should drop the 'b' and 'c' records. Now that there are only '0' and 'a' values left in the StateHoliday columns, they are changed to binary 0s and 1s respectively.

Before moving on, it was worth noting that there are only 694 records with state holidays in the train dataset, in comparison to 843,482 records without (0.082%). This is a small percentage, so the

StateHoliday may be an unreliable predictor. The corresponding percentage for the test dataset was at 0.074%. These two percentages being close in value means the StateHoliday variable will be a good representative.

Merging Datasets

The store dataset is merged with the training and testing datasets respectively, letting us make some more visualisations (many in Tableau) and adding features which will be used when building the models.

Creating Visualisations

For the sake of brevity, the graphs in this section only show notable findings on sales numbers. Please note that all the trends discussed here are largely mirrored by the customer numbers as well.

Sales increase slightly between March and July, around the summer holidays. A big increase is also seen in December, coinciding with Christmas.

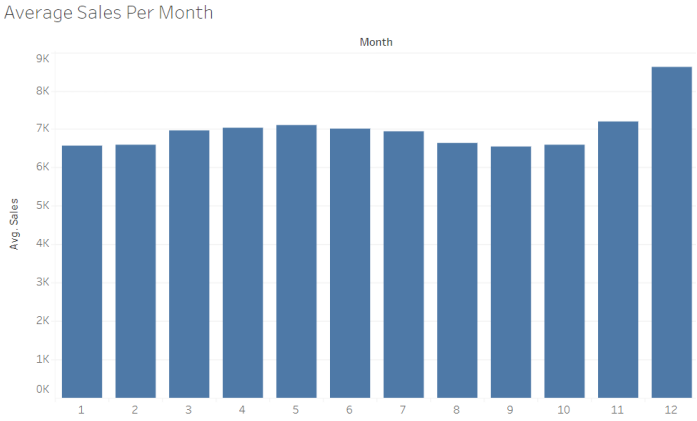

Figure 6: Sales per month.

Sales see an increase on Sundays as expected. A large increase is seen on Mondays, and a large decrease is seen on Saturdays, which is unexpected. It is unclear why this is the case.

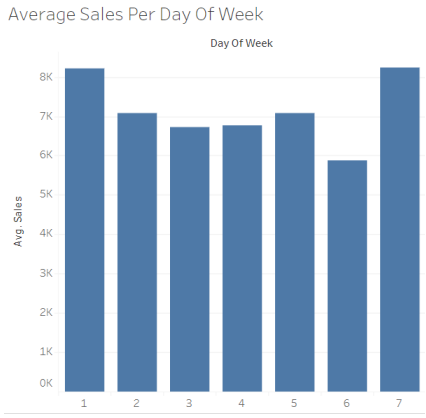

Figure 7: Sales per weekday.

In-store promos greatly increase sales. Unexpectedly, sales are lower when coupon promos are active. This does not necessarily show causation. It may be because coupon promos are only run in stores which are performing poorly.

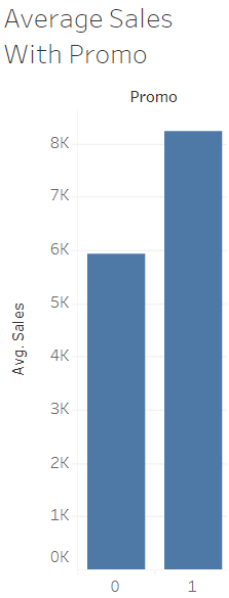

Figure 8: Effect of in-store promos on sales.

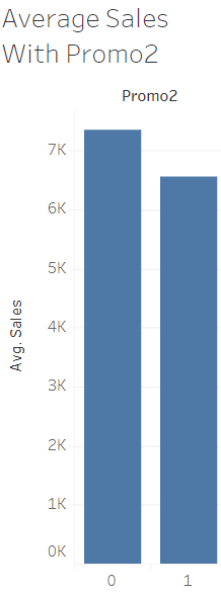

Figure 9: Effect of continuous promos on sales.

Unexpectedly, CompetitionDistance has little to no correlation with performance.

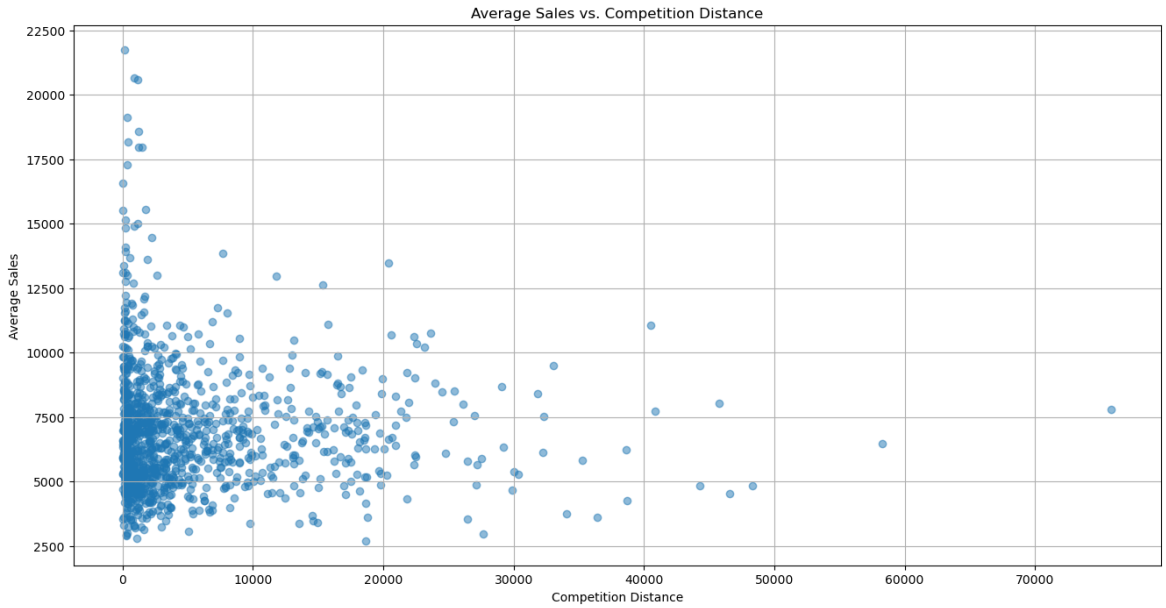

Figure 10: Effect of competition distance on sales.

Looking at some specific stores, we can see the positive effect that continuous promo campaigns can bring, as well as the potential negative effect of competition stores opening on performance.

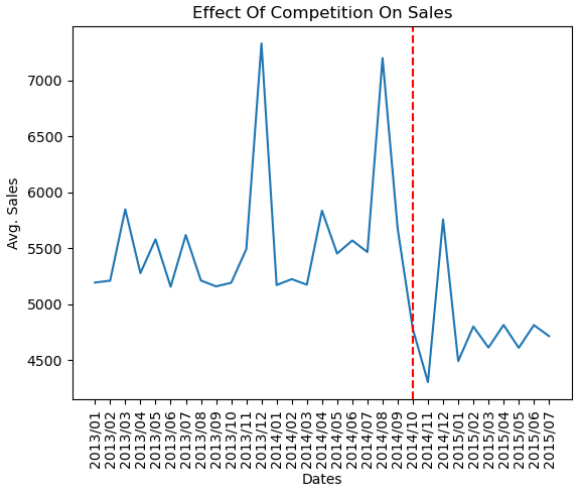

Figure 11: Effect of competition on sales.

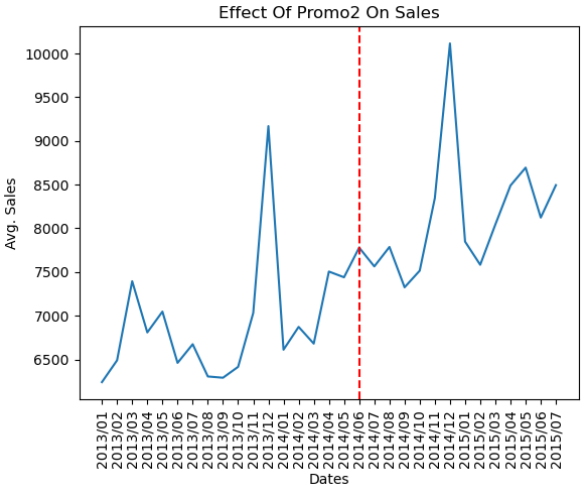

Figure 12: Effect of continuous promos on sales.

Preparation For Models

At this point, we have many variables which were useful for visualisation, but will need to be changed or dropped before building the models. Drop the Date, CompetitionSinceDate and Promo2SinceDate columns, as we are not using time series analysis. Some of the information from these columns will be captured in the Month, Year, MonthsSinceCompetition and MonthsSincePromo2 variables.

Plotting a correlation heatmap for the continuous variables, we see there is little to no correlation between the predictors and the target variables. This lack of linearity suggests that linear regression would not be an ideal model. With so many categorical variables, decision trees would be a better option. XGBoost uses gradient boosting to increase performance and efficiency. Although interpretability is reduced, this is not of interest in our context, so XGBoost is a great choice.

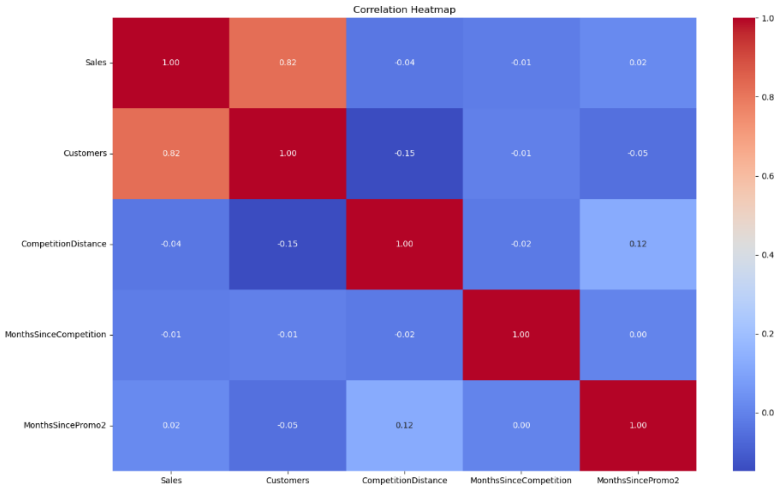

Figure 13: Correlation heatmap for continuous variables.

We prepare all categorical variables using label encoding. Since we are including variables such as Store, of which there are over 1000, one-hot encoding would not work. We convert these columns into the 'category' datatype.

Model Performance and Predictions

Using all predictors, build a model each for Sales and Customers.

```
Sales RMSE Mean: 946.8669928542924
Sales MAE Mean: 633.78712467228
Sales R-squared Mean: 0.90683267954199
Customers RMSE Mean: 86.08912311944172
Customers MAE Mean: 55.55499813211736
Customers R-squared Mean: 0.9538151165450068
```

Figure 14: Model performance metrics.

Some attempts were made to improve performance. MonthsSinceCompetition and MonthsSincePromo2 had little correlation with sales or customer numbers, but removing these greatly hindered performance, so they are kept in the models. CompetitionDistance is positively skewed, so a log transform was applied. This had a small negative effect on performance, so it was reverted. Removing it also greatly reduced performance. Therefore, with the features we have, these are the final models.

These models were then used to predict the missing values in the merged_test dataset. As discussed earlier, the final step was to directly assign any records where the store is closed with 0 sales and 0 customers.

Conclusion

In reflection, the biggest error in this project was to avoid the use of time series analysis. In an attempt to simplify the data in pre-processing, overlooking the dates columns led to model performance which was sub-optimal. The creation of some more features may also have led to better predictions, as the final model ended up being relatively simple.