ADS1002
Group Report

Analysis of Rossmann Drugstore Sales

Group Members

Koh Xuan Qing	**33521719**
Jiang Wen Feng	**34595147**
Tashvin Ramesh	**34675280**
Daniel Ong	**34897887**

Table Of Contents

Executive Summary

Background

Dirk Rossmann, born in 1946 into an entrepreneurial family in Hanover, founded Rossmann drugstores in 1972 (Rossmann, n.d.). He created Germany's first self-service drugstore. Within a decade, he owned 100 stores in northern Germany (Rossmann, n.d.). Rossmann credits the company's success to economic freedom, allowing independent pricing. His philosophy of perseverance helped navigate challenges during the early '90s, including financial strains and market expansions into Eastern Europe (Rossmann, n.d.). Today, Rossmann has become one of the leading drugstore chains in Germany, known for its commitment to quality, affordability, and social responsibility. The chain operates over 4000 branches in nine countries with the help of approximately 62000 employees (Forbes, n.d.). In 2023, the franchise generated €13.9 billion in revenue (Petruzzi, 2024).

The Problem

Sales forecasting is an important task in companies as it provides a great source of information for planning and decision making. Demand forecasting is the foundation of many managerial decisions which includes budgeting, production planning, human resource planning and many more. Nevertheless, demand forecasting can be difficult as there are many uncertainties that are associated with them. Some of the uncertainties may include economic situation, weather, promotions and holidays. All of these factors have an impact on the demand. In this report, we are going to discuss our process and findings in terms of predicting the Rossmann Drugstore sales for four weeks.

Findings

Our findings revealed that, on a yearly basis, the stores reach a sales peak during festive holidays such as Christmas and New Years. On a weekly basis, revenues are highest on Mondays. Additionally, the main factors that directly affect sales are the number of customers, store promotions, the store's open status, and the day of the week. Moreover, based on the prediction model that we have made, it shows a clear weekly pattern of the sales done. This cyclical behavior suggests a recurring trend in sales, possibly reflecting typical shopping habits or store operations. Nevertheless, the stability in prediction indicates that while daily and weekly fluctuations exist, they follow a predictable rhythm without significant external factors altering the sales trajectory.

Data Cleaning

Details of The Supplied Data

There are three datasets given on Moodle, which are the test.csv, the train.csv and the store.csv. The train.csv contains the historical data for 1115 Rossmann stores, including the Sales. In contrast, the test.csv contains the historical data for 1115 Rossmann stores, excluding the Sales. Additionally, the store.csv contains the supplemental information about the stores. Some of the features that are included in the datasets include Store where a unique ID is given for each store, Sales, Customers, Open as indicator for whether the store was open, StateHoliday and SchoolHoliday. Looking into the amount of data each dataset contains, it is found that the train dataset contains 1017209 rows and 9 columns, the test dataset contains 41088 rows and 8 columns whereas the store dataset contains 1115 rows and 10 columns. However, it is stated in the task description that we only need to work with the train dataset. Hence, the store dataset would be completely ignored in this task.

Focusing on the training dataset, we have 9 columns which resemble 9 features. The following table would show the details of the supplied features.

Features	Details
Store	The value given is the unique ID for each store
DayOfWeek	The day of the week that is represented in numerical form 1 = Monday, 2 = Tuesday, 3 = Wednesday and so on
Date	The date where the data was recorded
Sales	The turnover for any given day
Customers	The number of customers on a given day
Open	A binary indicator for whether the store was open 0 = closed, 1 = open
Promo	A binary indicator for whether the store had promotion on the day 0 = no promotion, 1 = have promotion
StateHoliday	An indicator for whether there is a state holiday on a given day
SchoolHoliday	A binary indicator for whether there is a school holiday on the day 0 = no school holiday, 1 = have school holiday

Table 1: A table that shows the details of the features in the training dataset

Data Cleaning

Before we did any processing or manipulation with the dataset, we had a look into the dataset and checked for its missing values. It is found that there are no missing values in the training dataset. However, we found that there are 11 missing values in the testing dataset. All of the missing values are located at the variable 'Open'.

Store	0		
DayOfWeek	0	Id	0
Date	0	Store	0
Sales	0	DayOfWeek	0
Customers	0	Date	0
Open	0	Open	11
Promo	0	Promo	0
StateHoliday	0	StateHoliday	0
SchoolHoliday	0	SchoolHoliday	0

Figure 1: The amount of missing values in the training and the testing dataset

Due to the presence of the missing values in the testing dataset, we decided to remove the rows with the missing value using the .dropna() function. After removing the missing values, we have a clean testing dataset with no missing values. Now, our testing dataset has only 41077 rows and 8 columns. This aligns with our cleaning process where 11 rows are removed from the dataset.

For the training dataset, we decided to take the top 20% of stores in terms of sales. This is because the original dataset contains more than a million rows, which might be difficult to interpret. By doing this, we can focus on the variables that have the most significant contribution to sales. To do this, we began with grouping the data by store and calculating the total sales for each store using the .groupby() function. This allows us to determine the total sales each store had generated throughout the period. Next, we identified the top 20% of stores based on their total sales. This is done using the .nlargest() function which extracts the highest values from our total sales data. We used 0.2 as the factor to get the top 20% of stores wanted. The indices, which is also the store ID, were then saved. With the IDs that we have obtained, we proceeded to filter the original training dataset and select only those rows which correspond to the ID that we have saved. This step involved the use of the .isin() method. Now, our newest training dataset contains the stores that have the highest 20% of sales. From 1115 stores contained in the training dataset, we reduced it to 223 stores. Additionally, the new training dataset has 208226 rows of data and 9 columns.

Exploratory Data Analysis

Statistical Description

	Sales	Customers	Open	Promo	SchoolHoliday
count	208226.000000	208226.000000	208226.000000	208226.000000	208226.000000
mean	8748.276330	1009.026490	0.840433	0.381989	0.181313
std	5086.094263	687.268818	0.366205	0.485875	0.385278
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	6728.000000	688.000000	1.000000	0.000000	0.000000
50%	8995.000000	970.000000	1.000000	0.000000	0.000000
75%	11494.000000	1297.000000	1.000000	1.000000	0.000000
max	41551.000000	7388.000000	1.000000	1.000000	1.000000

Figure 2: The statistical description on the more insightful features

To understand the training dataset better, we investigated the statistical description on the features that we think is more insightful among the 9 columns. Based on the figure shown above, it is seen that there is an average sales of 8748.28 whereas the average customer is 1009 for each store per day. However, both of this data might not be very meaningful if we take into account the days where some of the stores are closed. Looking at the 'Open' feature, we can see that about 75% of the data shows the value 1, which means that most of the stores are open throughout the period given. For the 'Promo' feature, it is seen that the binary value 1 is seen at approximately the third quartile of the data. This means that promotions are done less than half of the time. For the 'SchoolHoliday' feature, it is seen that the binary value 1 is shown as the maximum value. This means that most of the time, there is no school holiday.

Distribution of The Sales

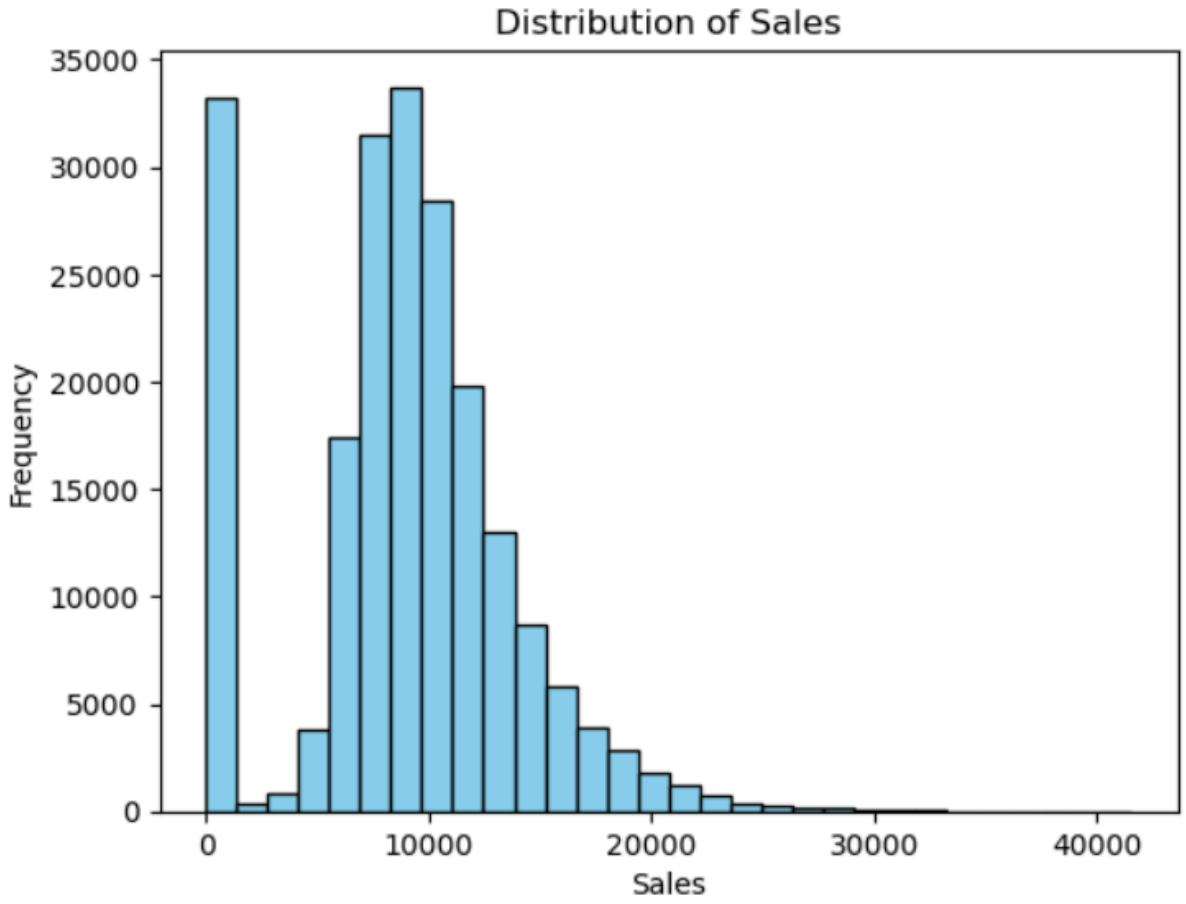

Figure 3: The sales distribution throughout the period

Besides that, we have analyzed the daily sales performance of each store to assess how they are performing throughout the period. From the histogram, we can see that there is a relatively high frequency for the value 0. This is due to the amount of stores that are closed throughout the period. If we ignore the zero sales and only focus on the stores that are open during the period, we can observe a right-skewed distribution for the sales. Moreover, we can see that most of the sales are concentrated at the range between 5000 and 15000, with the highest frequency being a value slightly lower than 10000. Additionally, sales that are more than 20000 are less common.

Monthly Sales Trend

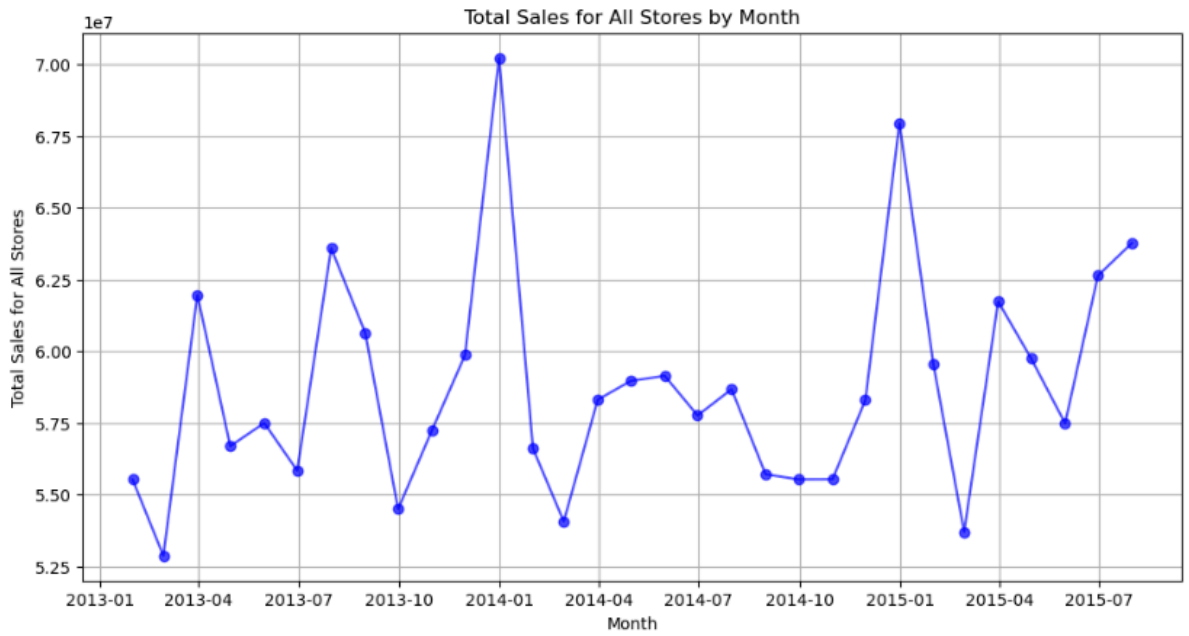

Figure 4: The monthly sales trend for all the stores

Other than that, we have investigated the monthly sales trend to observe various patterns, such as seasonal fluctuations, peaks during promotions or holidays, and any potential long-term growth or decline in sales. This analysis helps identify months with consistently higher or lower sales, revealing periods of increased demand, marketing effectiveness, or operational bottlenecks. Looking at the line graph, there are two clear peaks seen in January 2014 and 2015, with a value of 70 million and 68 million respectively. This could be due to the festive seasons as there is Christmas in the late December and New Years in the early January. Since these holidays are so close to each other, some consumers might prefer to celebrate them together, which resulted in a sharp increase in sales. Additionally, the lowest trough is seen in March 2013, reflected by the value of only 53 million. This could be influenced by the low amount of holidays during the month. It could also be influenced by other factors such as competitions and economic conditions during that time.

Daily Sales Trend

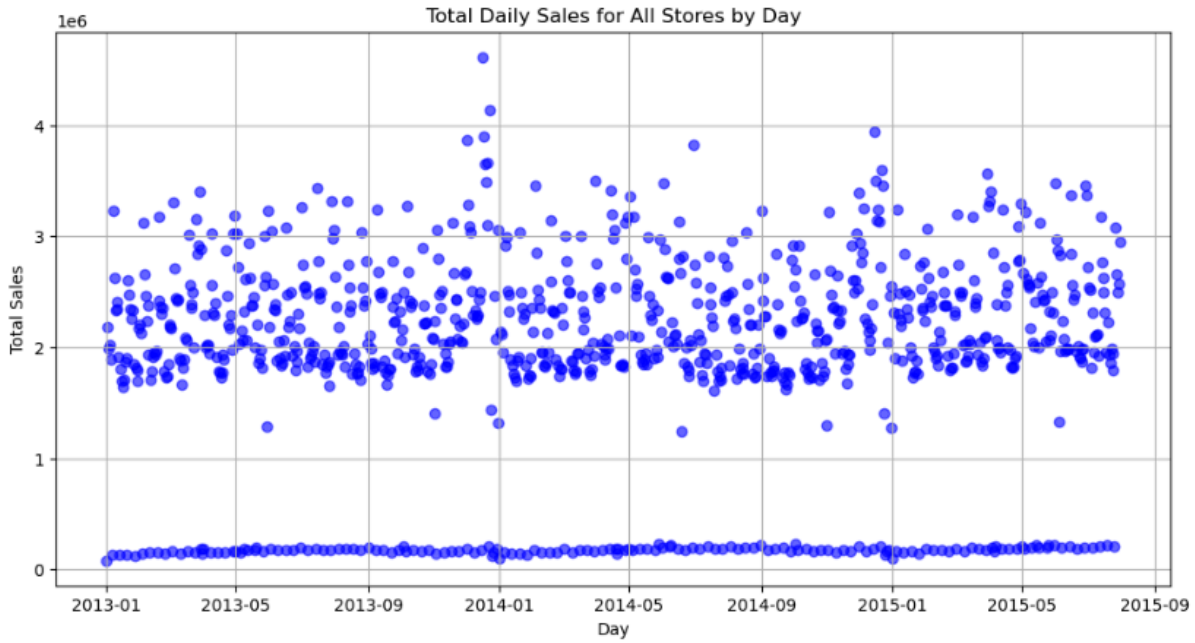

Figure 5: The daily sales trend for all the stores

Since the monthly sales pattern may not provide sufficient clarity for predicting sales over the next four weeks, we have also examined daily sales trends to gain more relevant insights for short-term forecasting. Based on the scatter plot shown above, it is clearly seen that there is a maximum data point shown in January 2014, which aligns with the observed trend in Figure 4. Looking at the plot, most daily sales are distributed in the range between 1.5 million and 3 million, with the majority clustering below 2 million. This suggests a typical sales range for the stores, which reflects a consistent business operation. Additionally, there is a row of data points shown in the region near to the value 0. The data in this region might be influenced by the store's closure, which could reduce the sales dramatically on certain days. Besides that, unusual conditions such as operational disruptions and unforeseen market conditions could also lead to the drop in sales, which causes these outliers.

Feature Correlation

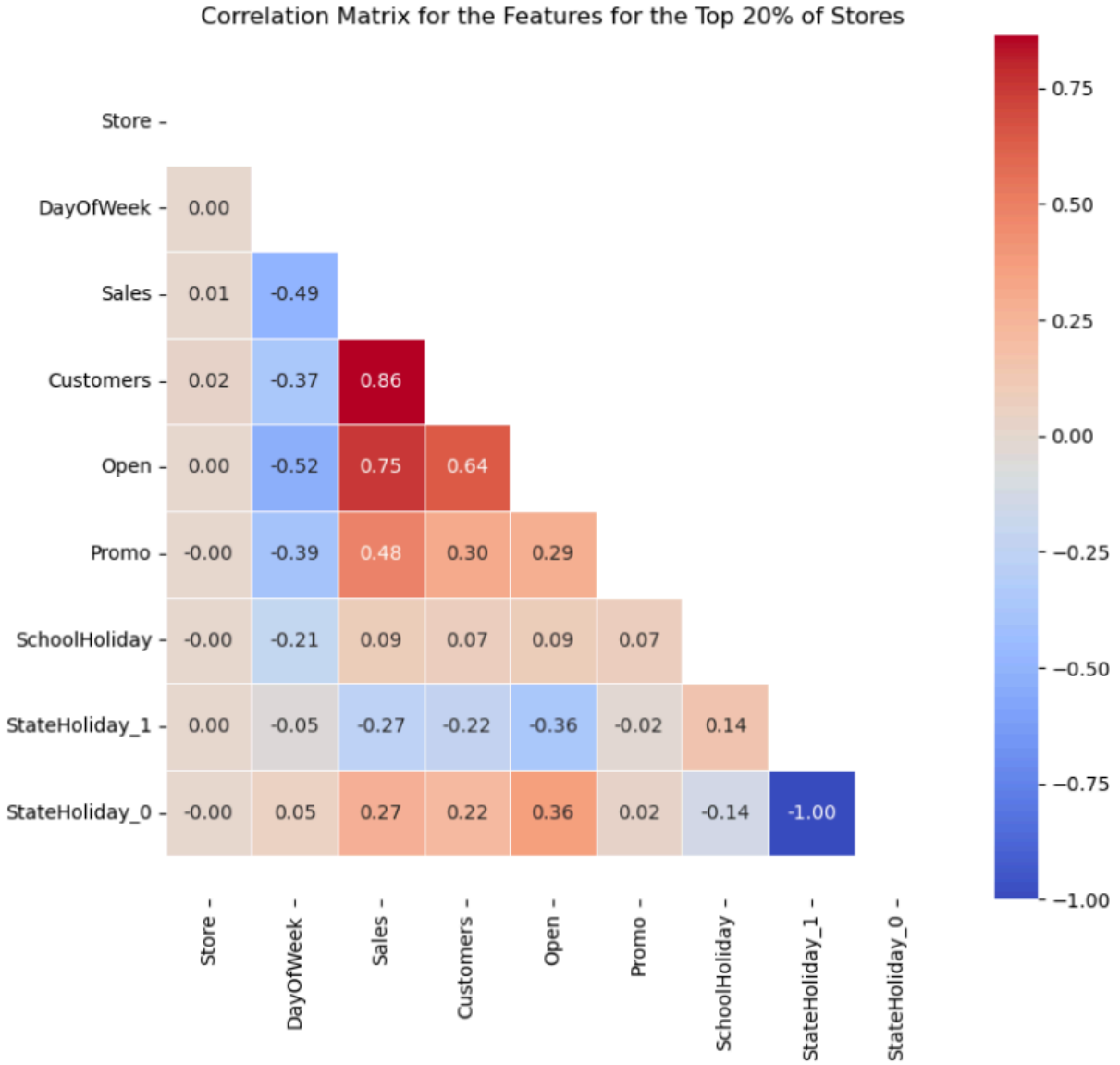

Figure 6: The correlation matrix for all features using heatmap

To investigate the relationships between each feature, we have constructed a correlation matrix in the form of a heatmap. Essentially, the correlation of each variable is denoted by a value ranging from -1 to +1. This means that the more negative a value is, the lesser it correlates with the corresponding feature. Besides that, from the heatmap shown, we have converted the string values in the 'StateHoliday' feature, which are 0, a, b and c, into binary form, where 0 means that there is no state holiday and 1 means that the day is a state holiday. This gives us a clearer idea on how state holiday impacts the sales of the stores. Additionally, we have removed the 'Date' feature as each date is unique and does not impact the sales.

Feature Correlation with Sales

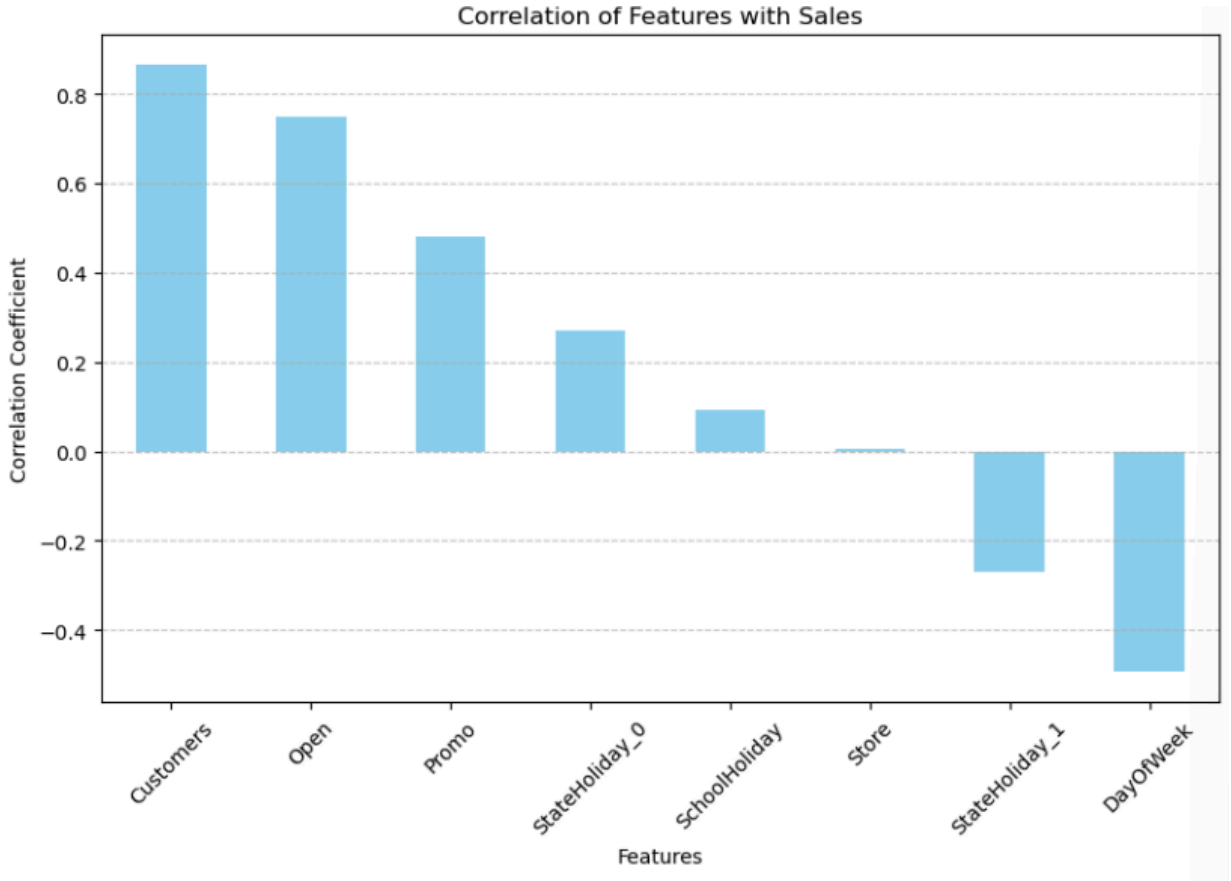

Figure 7: The correlation of each features with sales

From the figure above, it is seen that the four highest correlated features with sales are Customers, Open, Promo and DayOfWeek. These are also highlighted with a darker region of color in the heatmap shown previously. As seen from the figure above, the highest correlated feature is ‘Customers’, with a value of 0.86, followed by ‘Open’, with a value of 0.75 and ‘Promo’, with a value of 0.48. Although DayOfWeek is also quite highly correlated with a value of -0.49, the negative sign shows that it has an inverse relationship with sales. This indicates that as the 'DayOfWeek' variable increases, sales tend to decrease. This inverse relationship may reflect consumer behavior, such as lesser traffic during weekdays compared to weekends.

Sales Trend with The Day of The Week

Since it is obvious that more customers will bring in more sales and only open stores can make sales, we decided to only investigate the sales trend for the other two highly correlated features, which are 'DayOfWeek' and 'Promo'.

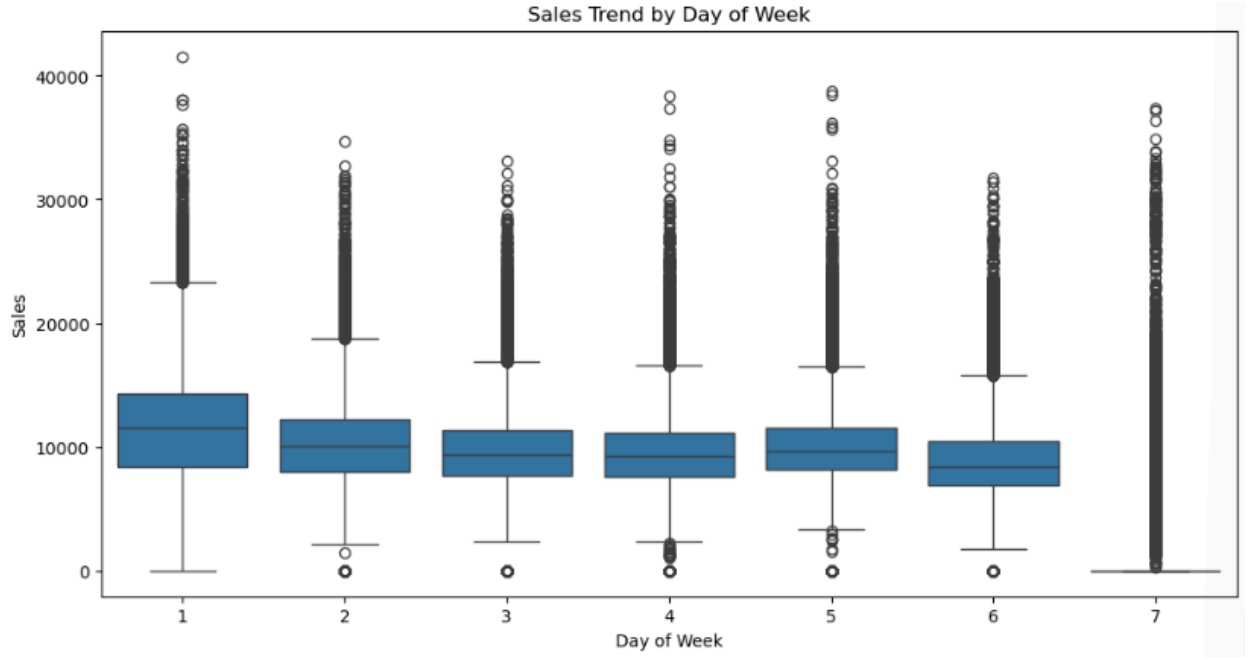

Figure 8: The sales trend by the day of the week

Looking at the box plot above, it can be seen that the sales generated from Monday to Saturday hovered around the 10,000 median sales range, with the highest median sales achieved on Monday. This is possibly due to the restocking done after the weekend or specific promotional activities initiated at the beginning of the week. Moreover, it is noticeable that there is no 'box' seen on Sunday. This can be concluded that most of the stores were closed. Hence, no sales could possibly be generated on the day. Additionally, there are many 'o' noticed above and below the boxes. These are the outliers. A factor that could contribute to the outliers is the competition with other stores. For instance, a nearby competitor may run a special promotion or event that draws customers away, resulting in lower sales for the stores represented in the dataset and vice versa.

Sales Trend with Promo

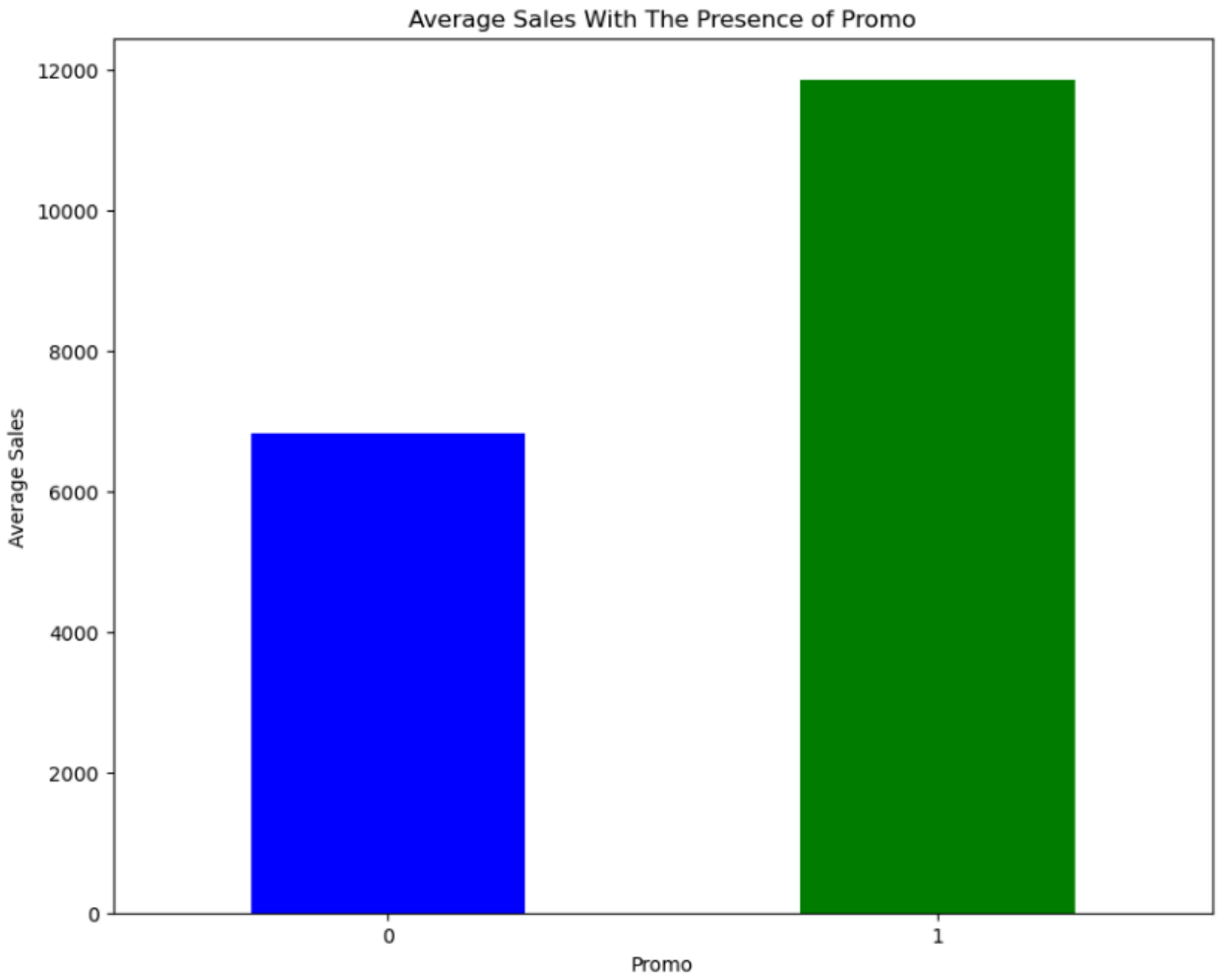

Figure 9: The sales trend with the presence of promotion

Moving on to the investigation of sales trends with the presence of promotion, we used a histogram to clearly show the impact of promotion on the average sales. Looking at the x-axis, 0 stands for no promotion whereas 1 stands for having promotion. As we can see, there is quite a noticeable difference in terms of average sales when there are promotions. During promotional periods, the average sales nearly doubled the amount of days without promotions. This suggests that customers are easily attracted to these offers since during these times, prices of products are lower than retail. Customers who are on a budget can make use of these promotions as a buying opportunity.

Summary of The Exploratory Data Analysis

From our investigation on the new training dataset that we obtained, it can be concluded that the 223 stores are open most of the time and promotion is done less than half of the time in the given period. Additionally, most stores have a sale between 5000 to 15000 daily and huge sales more than 20000 are very rare. There would be a significant increase of sales in January due to festivals such as Christmas and New Year. Most total daily sales are distributed in the range of 1.5 million to 3 million, with most of them being under 2 million.

Moreover, only four out of the eight features are correlated to the sales. These four features are 'Customers', 'Open', 'Promo' and 'DayOfWeek'. The highest correlated feature is 'Customers', with a value of 0.86, followed by 'Open', with a value of 0.75 and 'Promo', with a value of 0.48. Although DayOfWeek is also quite highly correlated with a value of -0.49, the negative sign shows that it has an inverse relationship with sales.

It is also seen that the sales generated from Monday to Saturday hovered around the 10,000 median sales range, with the highest median sales achieved on Monday. This could be due to the restocking done on the weekend. Besides that, most stores are closed on Sunday. Lastly, it is also seen that during promotional periods, the average sales nearly doubled the amount of days without promotions. This reflects customers preference towards the presence of promotions.

Data Modelling

The Lasso and Ridge Regression

Both Lasso and Ridge form regularization features in linear regression, providing a great role in the model improvement. Both these algorithms helped us in tuning our model not to overfit and provide the best predictive ability. The graphs given below illustrate the relationship between predicted sales and actual sales using Lasso and Ridge regression.

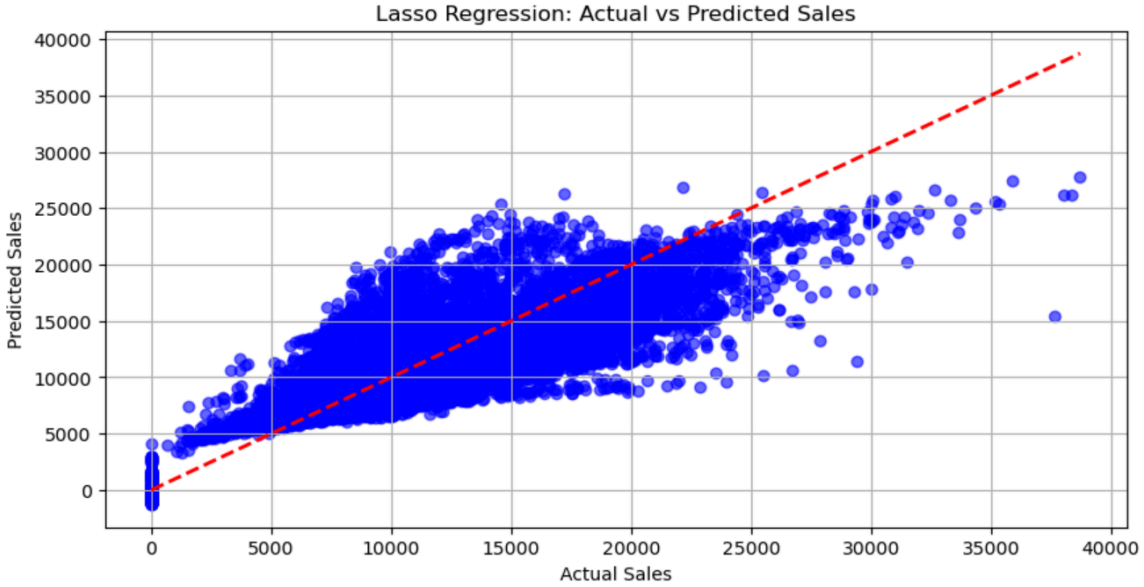

Figure 10: The Lasso Regression Model

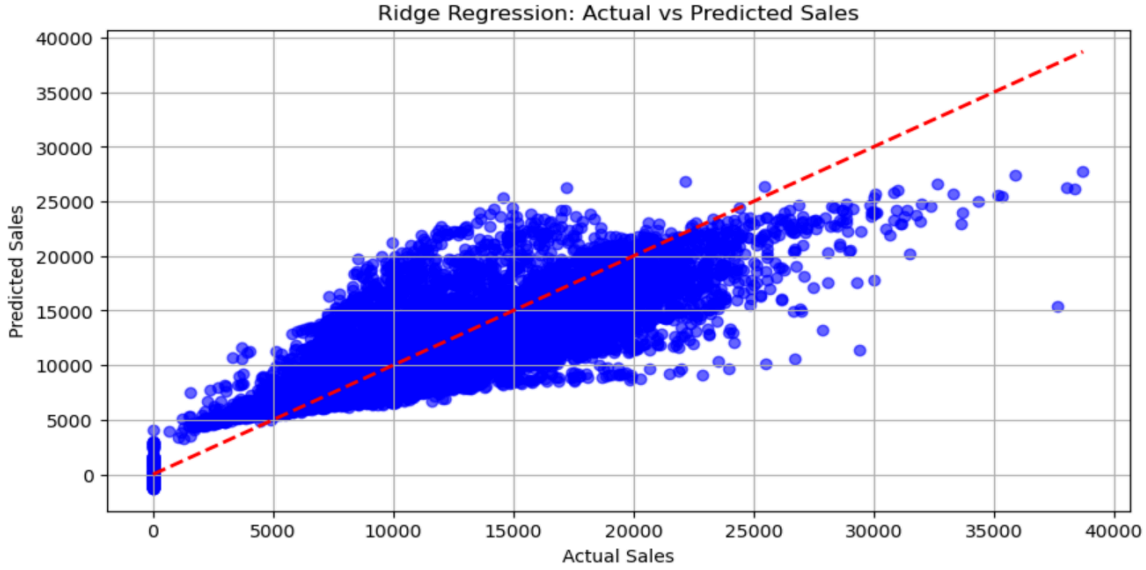

Figure 11: The Ridge Regression Model

By referring to these two graphs, we can see that the blue data points are clustered around the 5,000 to 20,000 range for actual sales. As a whole, we can see that there is a deviation from the red dotted line of best fit in both regressions. This infers that the model's prediction was not accurate enough and there are more actual sales generated than what was predicted. The results below illustrate the R^2, RMSE and MAE values as the evaluations of the models' performances.

	R^2 Value	RMSE	MAE
Train	0.853	1949.203	1286.378
Test	0.855	1929.433	1276.956

Table 2: The performance matrix for Ridge Regression model

	R^2 Value	RMSE	MAE
Train	0.853	1949.226	1285.381
Test	0.855	1929.438	1276.007

Table 3: The performance matrix for Lasso Regression model

Both regressions gave an accuracy R^2 value of 0.853 and 0.855 for both the training and testing datasets. There were slight differences in the root mean square error, RMSE, and the mean absolute error, or MAE values, but they were still relatively high. Although the accuracy was fairly good, we believe that we could utilize another tool to increase the accuracy and make the model more robust.

Random Forest

Random Forest is an ensemble learning technique. This means that it works based on a principle of constructing multiple decision trees and the random selection of features. When a prediction is made, the input data is predicted independently by each tree. The values of all the trees are then averaged to give a final predicted value, with improved accuracy and model performance.

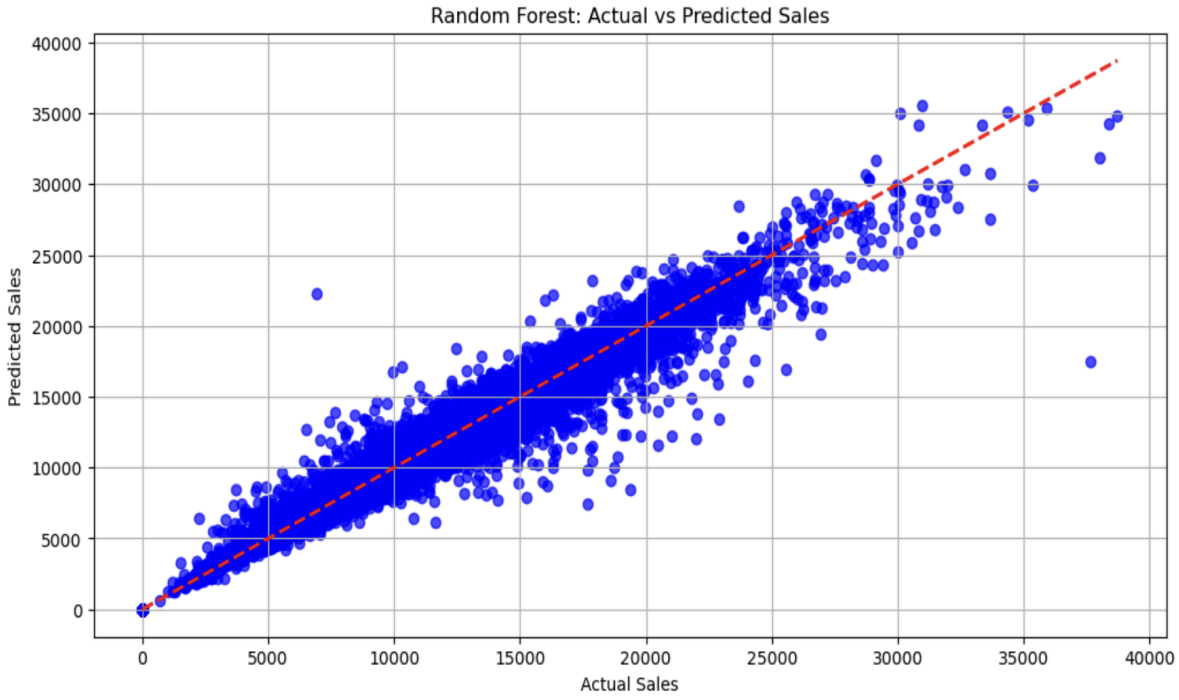

Figure 12: The Random Forest Model

Looking at the graph shown above, we noticed that, as compared to the Ridge and Lasso Regression models used previously, the blue dots seem to fit the red dotted line better. There is also reduced clustering. There is a more uniform distribution of the blue data points and they appear to be less deviated from the red line. This shows that there is a better fit even though there are still some points that are away from the red line. Overall, the trend exhibits a strong positive correlation between the actual and predicted sales. Hence, the model performed well in the vast majority of cases.

Additionally, we have also generated a bar plot to display the important features using the Random Forest model as this gave us a clear visual representation of how important certain features are when it comes to sales.

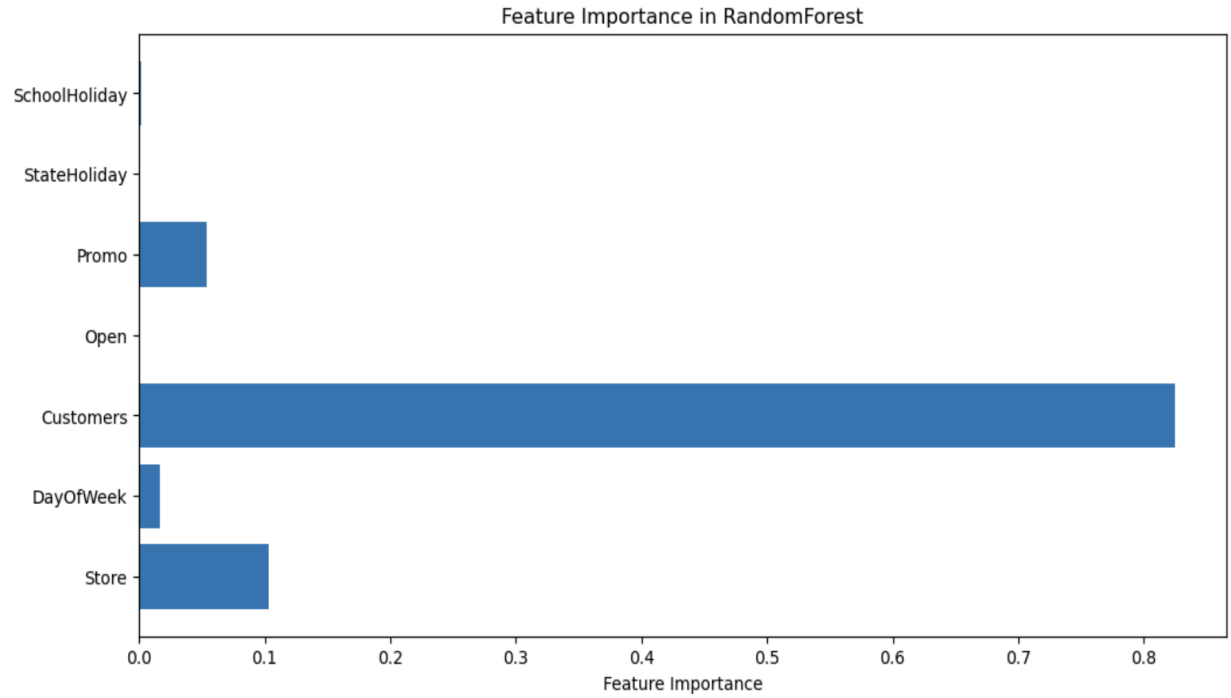

Figure 13: The Feature Importance for Random Forest

From this picture, we can see that 'Customers' is the most important feature, with a feature importance of close to 0.8. Thus, this feature contributes the most to the model's performance. The next two influential features are 'Promo' and 'Store', indicating that these two features have some importance but the size of the bars are relatively small. However, in this case, it is more sensible if the feature, 'Open' also shows its importance when it comes to predicting. This is because only stores that are open can make sales. Nevertheless, since Random Forest is an integrated model, it is difficult to know how individual features affect the final result. This makes the feature 'Open' difficult to explain in terms of why it is not considered important.

	R^2 Value	RMSE	MAE
Train	0.997	299.197	182.918
Test	0.978	752.076	460.716

Table 3: The performance matrix for Random Forest model

The table above shows the performance evaluation results of the Random Forest prediction model on the testing and training datasets. Based on these results, we can see that the R^2 value of 0.997 for the training data. This indicates that the model performed extremely well as it almost achieved perfect accuracy in fitting the model. The R^2 value for the testing dataset was 0.978 which was also very good even though it did not fit as well as compared to the training dataset. Both the training and testing datasets had a significantly lower RMSE value compared to the regularization techniques discussed earlier. Both values were below the 1000 mark, indicating that the model has improved in performance. However, the testing dataset had a much higher RMSE of 752.053, meaning there is a larger prediction error on unseen data. There is also a large reduction in the MAE values in both datasets. However, the testing dataset has a higher value of 460.714 as compared to the value, 182.918 shown for the training dataset. This could also mean that the model has a larger prediction error on the testing dataset.

Overall, we can conclude that although the R^2 values are high for both datasets, the higher values for the RMSE and MAE on the testing dataset shows an increase in error, which may mean that there is a slight amount of overfitting.

Prediction Analysis

The final stage of this project was to come up with a predictive analysis scheme to forecast the sales of the store for the future 4 weeks. This is a crucial technique as it helps businesses like the Rossmann drugstore to make more informative decisions to ensure that the actual sales generated matches the prediction models.

Time Series Forecasting

We developed a time series forecasting model to predict total sales across all stores for each day over the next 4 weeks. The graph below illustrates these predictions.

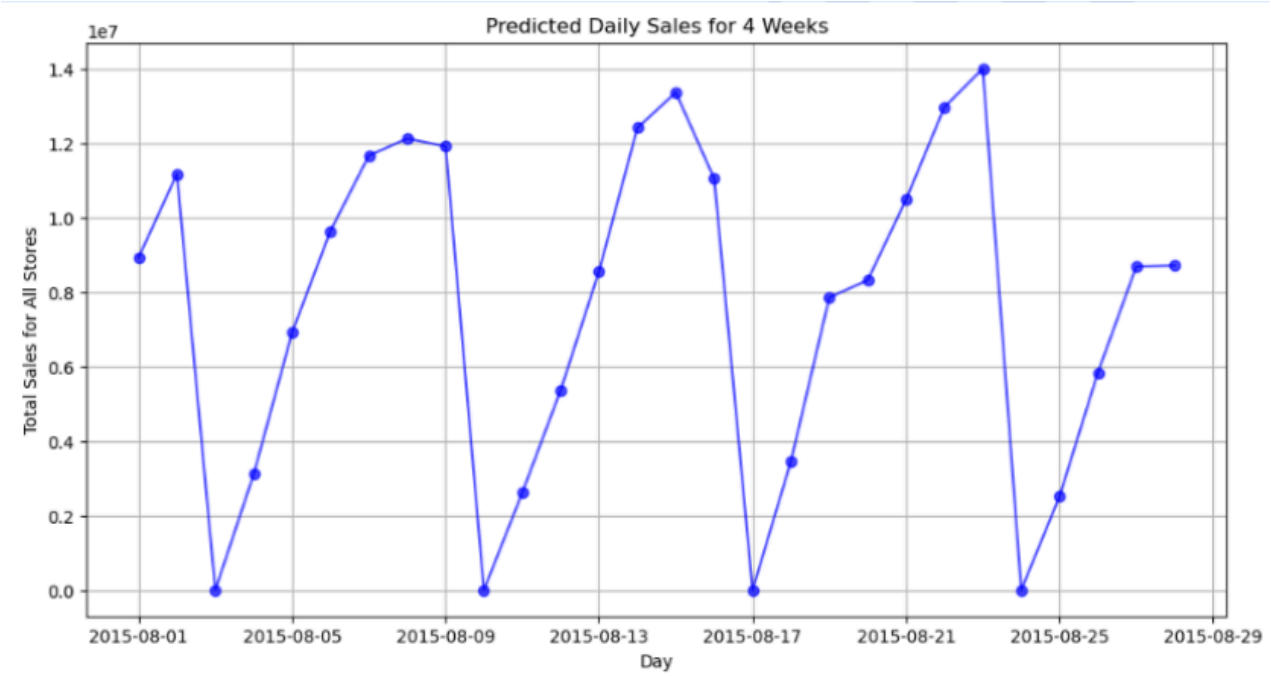

Figure 14: The prediction of the daily sales for the next 4 weeks

As seen from the graph, there is a distinct up-and-down pattern occurring on a weekly basis over the forecasted 4-week period. This cyclical behavior suggests a recurring trend in sales, possibly reflecting typical shopping habits or store operations. Despite the weekly fluctuations, the sales remain within a range of 3 million to 14 million. The recurring drops to zero likely correspond to Sundays when most stores are closed, further reinforcing this weekly cycle.

Although there is a clear short-term pattern of rising and falling sales within each week, there is no overall positive or negative trend over the 4-week span. This stability indicates that while daily and weekly fluctuations exist, they follow a predictable rhythm without significant external factors altering the sales trajectory.

Further Suggestions

We have also come up with some possible suggestions that could possibly help the Rossmann franchise see a boost in its sales. First, opening the stores for more days including Sundays could benefit from the extra sales that could be generated. Since there are more people who are free during the weekends to do shopping or for leisure, this can be a perfect selling opportunity for the store to generate income. Additionally, having more promotions is another aspect that can increase the growth of the store and improve revenue. Reducing the prices of items in the store can attract a large group of consumers, especially those who are on a budget. Moreover, enhancing the appearance of the store by using more visually appealing decorations is an eye-attracting factor to new or existing customers. The look of the store acts as a magnet to draw the customers' attention.

Conclusion

In conclusion, we successfully developed an efficient model for forecasting the sales of Rossmann drugstores by employing multiple machine learning techniques, including Lasso, Ridge and Random Forest. Among these, the Random Forest model demonstrated the best performance. It has achieved an R^2 value of 0.997 for the training data, which indicates that it has a high accuracy and reliability in predicting sales. Besides that, our prediction analysis using a time series forecasting model revealed that sales fluctuated between 3 to 14 million. This displays a clear short-term weekly pattern of rising and falling sales. This insight helps in understanding the cyclical nature of sales, with peaks and troughs likely corresponding to operational factors and consumer shopping behaviors.

To further enhance our sales forecasting model, we would be exploring some advanced ensemble techniques such as bagging, boosting and stacking. Bagging, exemplified by Random Forest, trains multiple models on bootstrapped subsets of data, reducing variance and improving stability. Boosting builds models sequentially, focusing on correcting prior errors for greater accuracy. Stacking combines multiple models by using their predictions as inputs for a meta-model, leveraging the strengths of various algorithms. By incorporating these techniques, we can improve the robustness, reduce error rates and capture more complex sales trends effectively.

References

Forbes. (n.d.). *Rossmann | Company Overview & News*. Forbes. https://www.forbes.com/companies/rossmann/

Petruzzi, D. (2024, January 30). *Rossmann's annual revenue from 2008 to 2023*. Statista. https://www.statista.com/statistics/1009587/rossmann-s-annual-revenue/

Rossmann. (n.d.). *From a bicycle with a basket to a drugstore empire*. Rossmann. https://www.rossmann.pl/firma/en-us/about-us/dirk-rossmann?srsltid=AfmBOoqprZnKanVhK9wLIPRc-xwkXSL4JTov3zER2ffCCK3p0Xdfgrm_

Appendix

```
# Divide the data into two parts
X = top_20_stores_data.drop(['Date', 'Sales'], axis=1)
Y = top_20_stores_data['Sales']

# Split the data into training and test sets (80% train, 20% test), with a random seed of 42 for reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Standardize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create a list of regularization strengths to test
alfas = np.logspace(-2, 2, num=21)

# Initialize a RidgeCV model
ridge_model = RidgeCV()
ridge_model.fit(X_train_scaled, y_train)

# Calculate the R2 score
train_socre = ridge_model.score(X_train_scaled, y_train)
test_socre = ridge_model.score(X_test_scaled, y_test)

# Get the optimal alpha
optimal_alpha = ridge_model.alpha_

# Predict the target variable for the training data
train_prediction = ridge_model.predict(X_train_scaled)

# Predict the target variable for the test data
test_prediction = ridge_model.predict(X_test_scaled)

# Calculate the RMSE
train_rmse = np.sqrt(mean_squared_error(y_train, train_prediction))
test_rmse = np.sqrt(mean_squared_error(y_test, test_prediction))

# Calculate the MAE
train_mae = mean_absolute_error(y_train, train_prediction)
test_mae = mean_absolute_error(y_test, test_prediction)

print(f'optimal score: {optimal_alpha:.3f}')
print(f'\nTrain R2 score: {train_socre:.3f}')
print(f'Test R2 score: {test_socre:.3f}')
print(f'\nTrain RMSE: {train_rmse:.3f}')
print(f'Test RMSE: {test_rmse:.3f}')
print(f'\nTrain MAE: {train_mae:.3f}')
print(f'Test MAE: {test_mae:.3f}')
```

Figure 15: The code to obtain the performance matrix for Ridge Regression model

```
# Get the feature importances from the trained Rideg model
festure_importance = ridge_model.coef_

# Get the feature names from the columns of X
festure_names = X.columns

plt.figure(figsize=(12, 6))
plt.barh(festure_names, festure_importance)
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Ridge Regression')
plt.show()
```

Figure 16: The code to obtain the important features for Ridge Regression model

```
# Use the trained Ridge regression model to predict the sales values for the test data
y_pred = ridge_model.predict(X_test_scaled)

plt.figure(figsize=(10, 8))
plt.scatter(y_test, y_pred, alpha=0.6, color='b')

# This line connects the minimum and maximum values of the actual sales to form a diagonal reference
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Ridge Regression: Actual vs Predicted Sales')
plt.grid()
plt.show()
```

Figure 17: The code to plot the Ridge Regression model

```
# Create a list of regularization strengths to test
alfas = np.logspace(-2, 2, num=21)

# Initialize a LassoCV model
lasso_model = LassoCV()
lasso_model.fit(X_train_scaled, y_train)

# Calculate the R2 score
train_socre = lasso_model.score(X_train_scaled, y_train)
test_socre = lasso_model.score(X_test_scaled, y_test)

# Get the optimal alpha
optimal_alpha = lasso_model.alpha_

# Predict the target variable for the training data
train_prediction = lasso_model.predict(X_train_scaled)

# Predict the target variable for the testing data
test_prediction = lasso_model.predict(X_test_scaled)

# Calculate the RMSE
train_rmse = np.sqrt(mean_squared_error(y_train, train_prediction))
test_rmse = np.sqrt(mean_squared_error(y_test, test_prediction))

# Calculate the MAE
train_mae = mean_absolute_error(y_train, train_prediction)
test_mae = mean_absolute_error(y_test, test_prediction)

print(f'optimal score: {optimal_alpha:.3f}')
print(f'\nTrain R2 score: {train_socre:.3f}')
print(f'Test R2 score: {test_socre:.3f}')
print(f'\nTrain RMSE: {train_rmse:.3f}')
print(f'Test RMSE: {test_rmse:.3f}')
print(f'\nTrain MAE: {train_mae:.3f}')
print(f'Test MAE: {test_mae:.3f}')
```

Figure 18: The code to obtain the performance matrix for Lasso Regression model

```
# Use the trained Lasso regression model to predict the sales values for the test data
y_pred = lasso_model.predict(X_test_scaled)

plt.figure(figsize=(12, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='b')

# This line connects the minimum and maximum values of the actual sales to form a diagonal reference
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Lasso Regression: Actual vs Predicted Sales')
plt.grid()
plt.show()
```

Figure 19: The code to plot the Lasso Regression model

```
# Initialize a RandomForestRegressor model with 1000 decision trees (n_estimators) and a random seed of 42
rf_model = RandomForestRegressor(n_estimators=1000, random_state=42)

# Fit the model to the training data
rf_model.fit(X_train_scaled, y_train)

# Make predictions using the trained model on the training data
x_pred = rf_model.predict(X_train_scaled)

# Make predictions using the trained model on the test data
y_pred = rf_model.predict(X_test_scaled)

# Calculate the Mean Absolute Error (MAE) for the training data
train_mae = mean_absolute_error(y_train, x_pred)

# Calculate the MAE for the test data
test_mae = mean_absolute_error(y_test, y_pred)

# Calculate the Root Mean Squared Error (RMSE) for the training data
train_rmse = np.sqrt(mean_squared_error(y_train, x_pred))

# Calculate the RMSE for the test data
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Calculate the R-squared (R2) score for the training data
train_r2 = r2_score(y_train, x_pred)

# Calculate the R-squared (R2) score for the test data
test_r2 = r2_score(y_test, y_pred)

print(f'Train RMSE: {train_rmse:.3f}')
print(f'Test RMSE: {test_rmse:.3f}')

print(f'Train MAE: {train_mae:.3f}')
print(f'Test MAE: {test_mae:.3f}')

print(f'Train R2 Score: {train_r2:.3f}')
print(f'Test R2 Score: {test_r2:.3f}')
```

Figure 20: The code to obtain the performance matrix for Random Forest model

```
# Get the feature importances from the trained RandomForest model
festure_importance = rf_model.feature_importances_

# Get the feature names from the columns of X
festure_names = X.columns

plt.figure(figsize=(12, 6))
plt.barh(festure_names, festure_importance)
plt.xlabel('Feature Importance')
plt.title('Feature Importance in RandomForest')
plt.show()
```

Figure 21: The code to obtain the important features for Random Forest model

```
# Set the figure size for the plot to 12 inches by 6 inches
plt.figure(figsize=(12, 6))

# Create a scatter plot with actual sales and predicted sales
plt.scatter(y_test, y_pred, alpha=0.7, color='b')

# This line connects the minimum and maximum values of the actual sales to form a diagonal reference
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)

plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Random Forest: Actual vs Predicted Sales')
plt.grid()
plt.show()
```

Figure 22: The code to plot the Random Forest model