Document

Team Members :

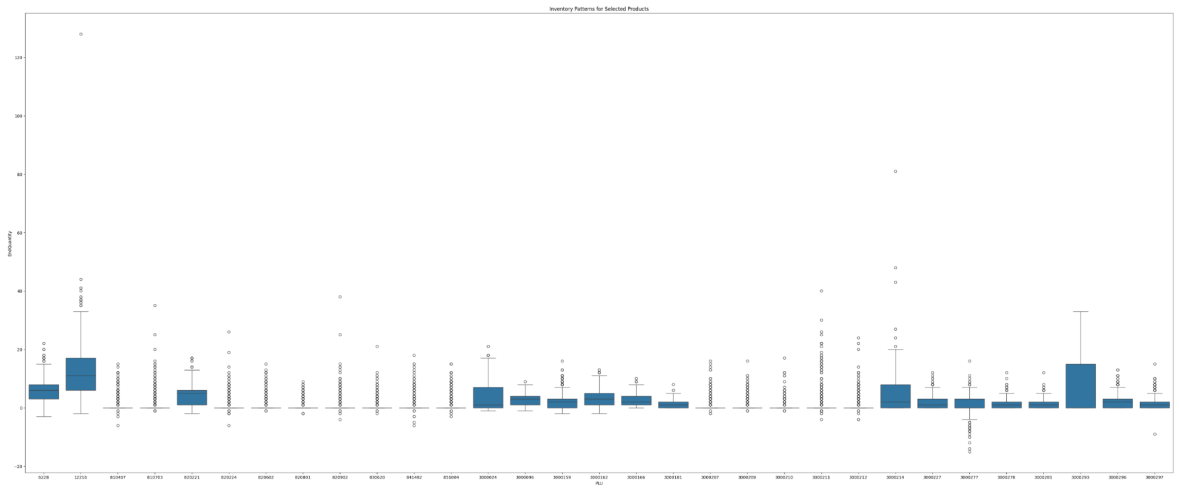
Name	CSULB ID	Work
Sampath Abhishek Ayyalasomayajula	030834256	Focused on analyzing the data of three specified stores (StoreID=18, 117, 332)
Sandeep Kumar Kulathuran Pillai Suba	030065592	Focused on the data of the entire 132 stores.
Sritharni Chellan Natarajan	028269772	Focused on the data of the entire 132 stores.

Section 1: Data Visualization (Coding and chapter one of the report)

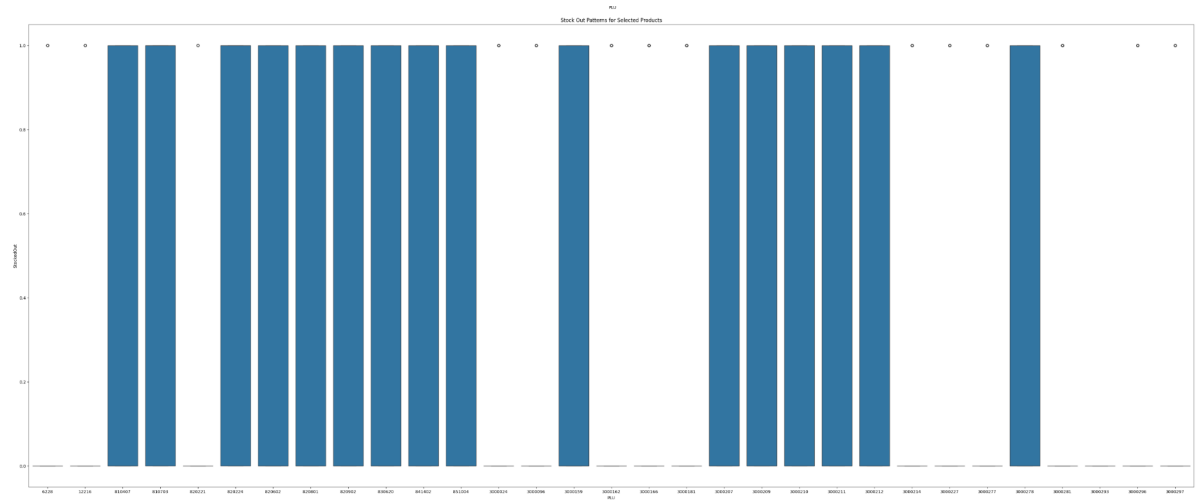
Part A: Analyze the data of individual stores and draw insights:

1.Provide the box plots and statistics of 27 products, inventory patterns, stock out patterns and missed sales.

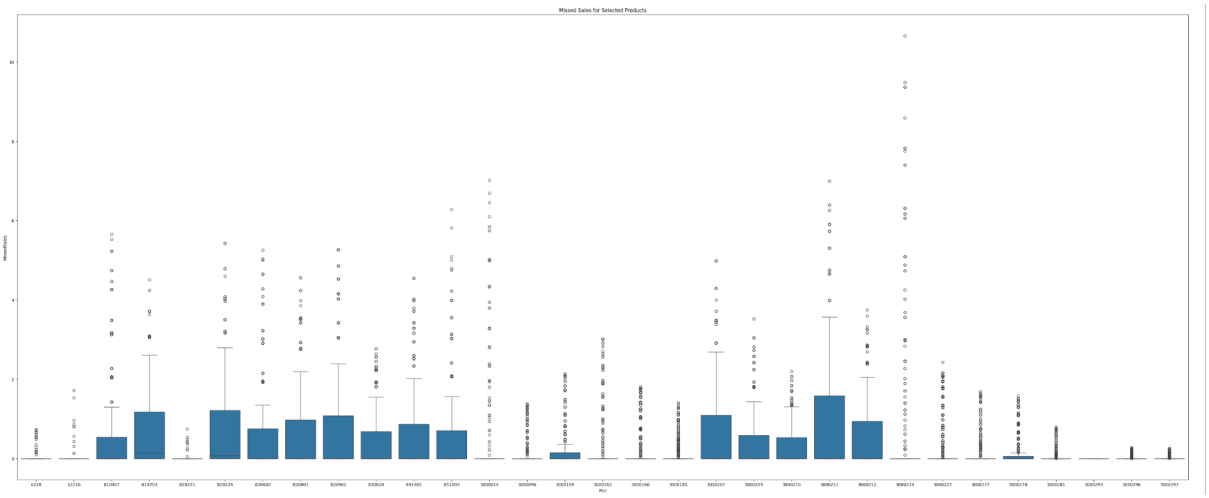
Inventory Patterns for the products:



Stockout Patterns for the products

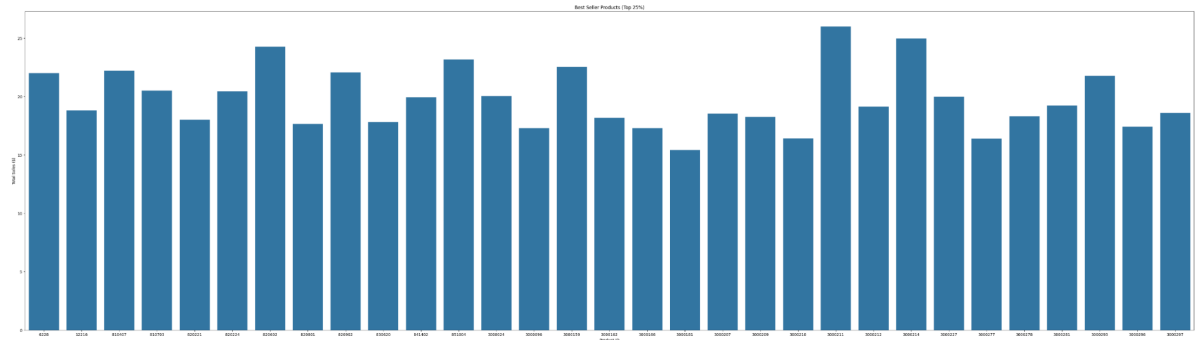


Missed sales for the products

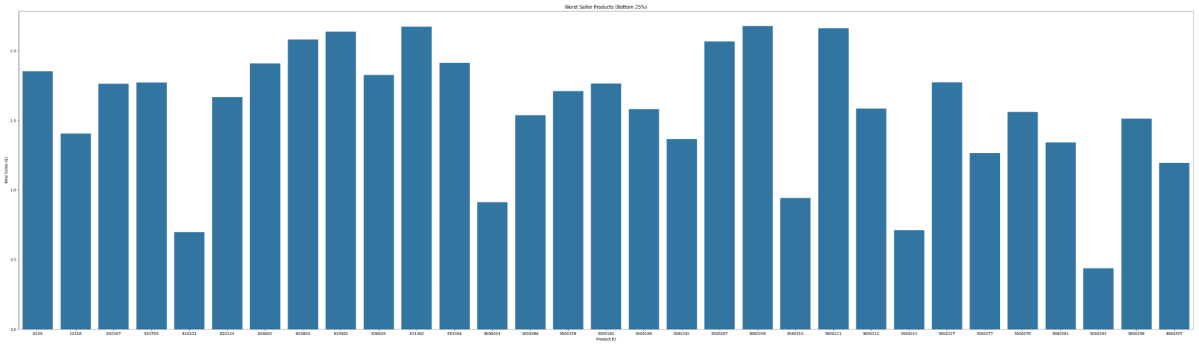


2.Show graphs of best seller and worst seller products of top 25% and bottom 25% and provide your insight into data.

Best Seller:



Worst Seller:



Insights:(Best seller)

- The Bake case is the most popular product which includes muffins, danishes and bagels.
- From the graph we can infer that these products are in high demand showing positive sales.
- The average total sales for the top products sums up to $15.

Recommendations:(Worst seller)

- Extra efforts can be made to make sure that the best selling products are manufactured in surplus amounts to meet the demands.
- Marketing strategies can be used revolving around the top selling products.

Insights:(Worst seller)

- Caramel cake pop, coffee cake with streusel and new york cheesecake are the least selling products.
- The average total sales of these underperforming products are $3.

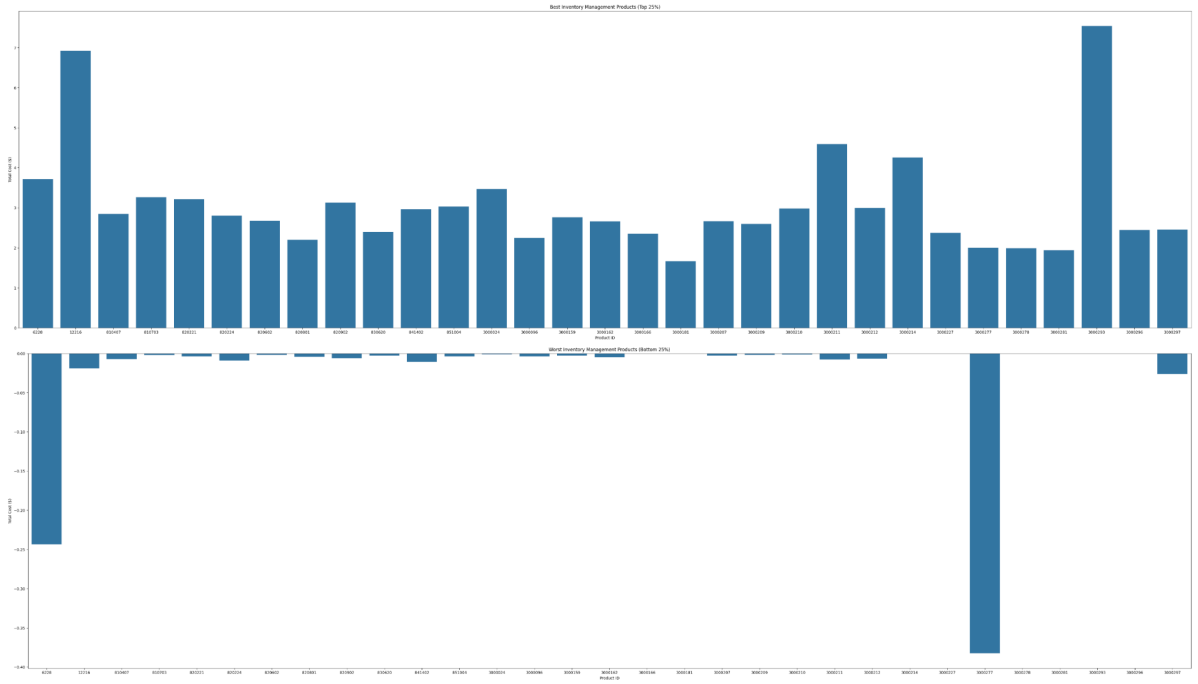
Recommendations:(Worst seller)

- Consistently low performing products can be filtered out and can be excluded from the menu.
- Customer surveys can be done to analyze and understand the reasons for quantity(lesser) being sold.

So what:

- **Market trends**: Constant monitoring of market trends seasonality external factors may be recorded and analyzed.
- **Inventory optimization**: Streamline and optimize the best selling and worst selling products for the better success of the business.
- **Customer choices**: Understand customer preferences in order to gain insights which could be useful when deciding upon the product that has to be marketed well and the product that fails to sell closer to the expected quantity, could be removed.

3. Show graphs of best and worst products based on their inventory management - Top 25% and bottom 25% and provide your insight into data (the average cost of a product is 0.5$) Identify where/when the store gets rid of the non purchased products



```
Average total cost for best inventory management products: $3.36
Average total cost for worst inventory management products: $-0.02
```

Top 25:

Insights:

- We could notice that few of the best performing items are out of stocks in certain inventory which requires forecasting the demand and efficient supply chain management.
- Top performing products show favorable sales to cost ratios.It can be inferred that these products not only sell well but also contribute positively for the profit.
- We could also notice that the best selling products are regularly replaced from the consistent order patterns

So what:

- Predictive analysis could be used to forecast the demand patterns thus enhancing customer satisfaction.
- Strategic pricing and marketing strategies could be implemented to increase the profitability.

Bottom 25

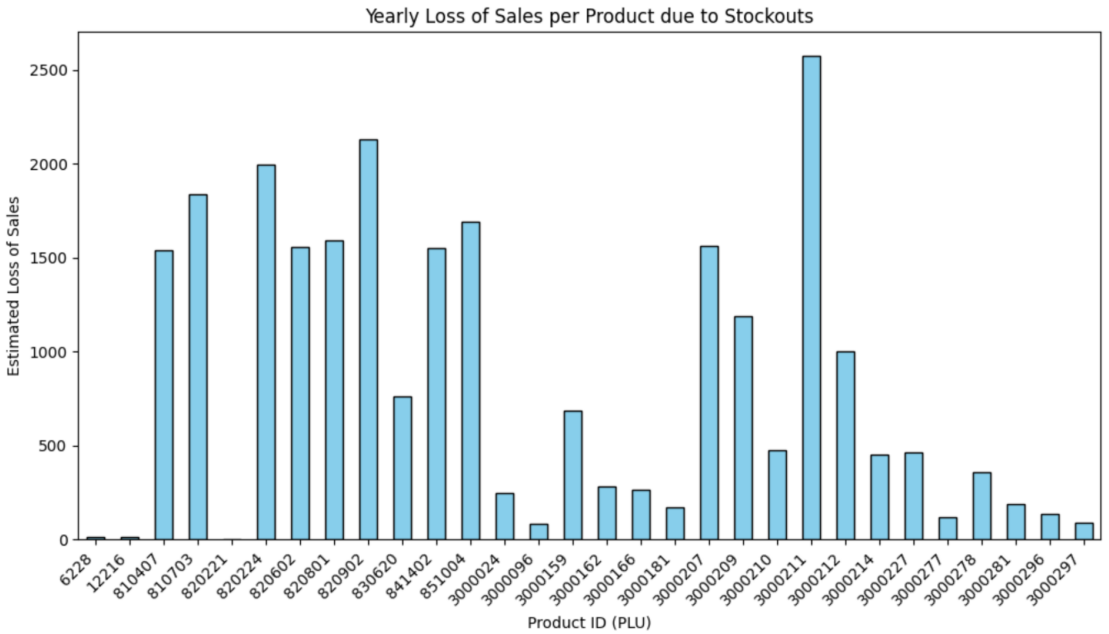
Insights:

- Products that are left unpurchased with low sales can be discarded due to lack of demand or their expiring nature.
- The low performing button 25% products exhibit frequent misales and stockouts. We could infer that the inventory is managed very poorly.
- Low performing products have high cost to sales ratio indicating negative misales.

So what:

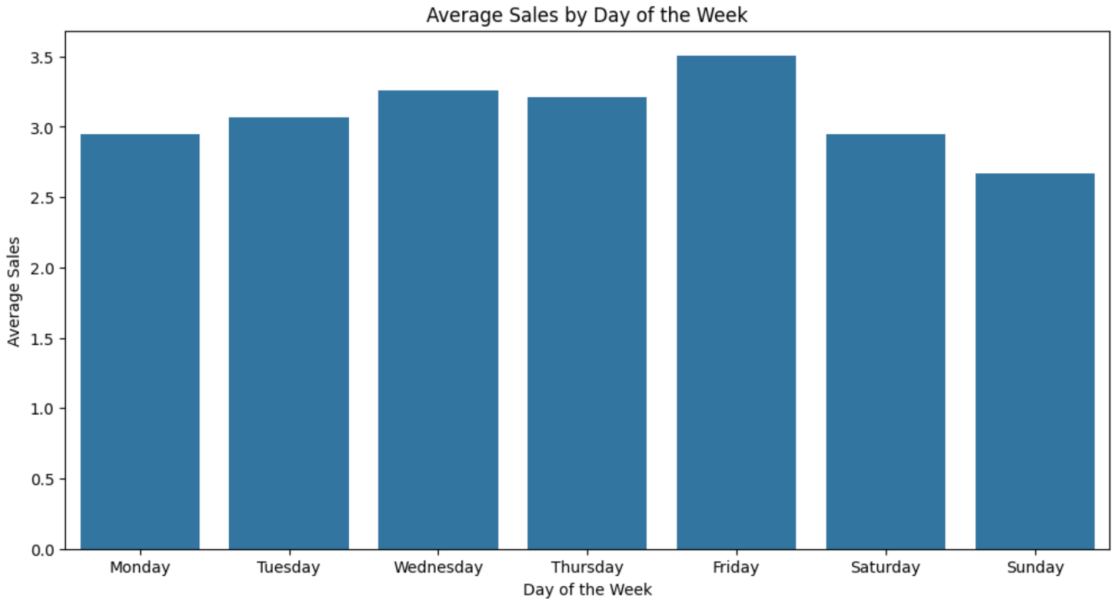
- Investigate the reasons for stockout and misales and implement corrective measures.
- Geographically get a sense of demand for the products and the inventory could be optimized based on the geographical demands and needs.

4. Identify stock outs and estimate the loss of sales per year per product. Assume when we are out of stock, we conservatively lose 75% of the average of sales in the previous 4 weeks on the same weekday.

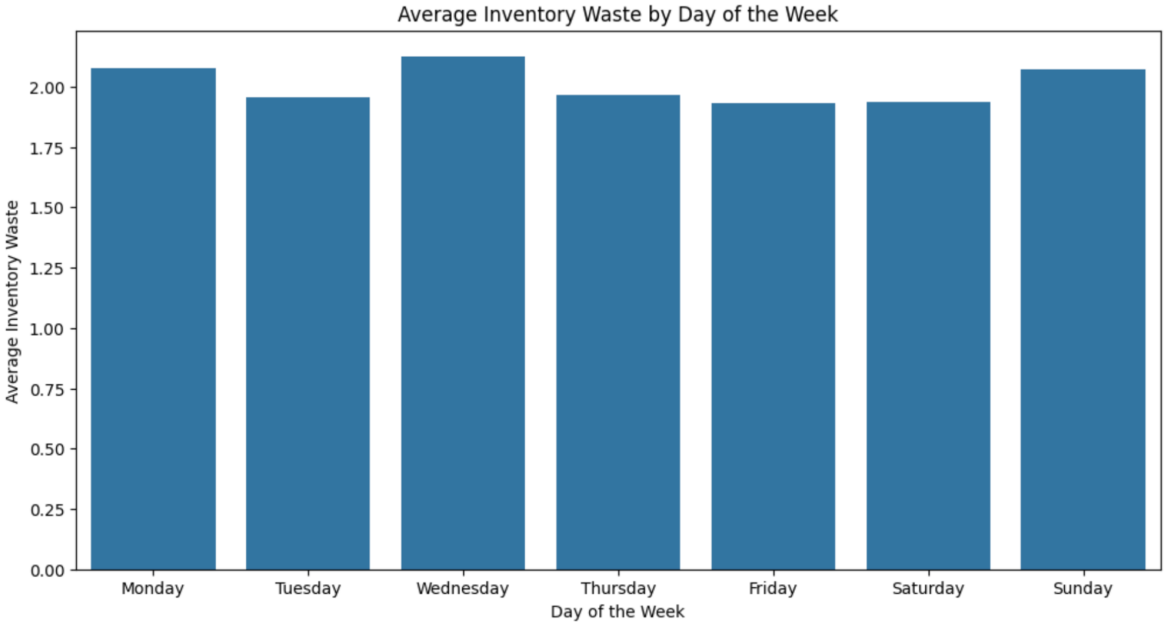


5. Show graphically how the product sales and inventory waste change. Investigate

a. Impact of day of the week on sales and stocks (7 days)

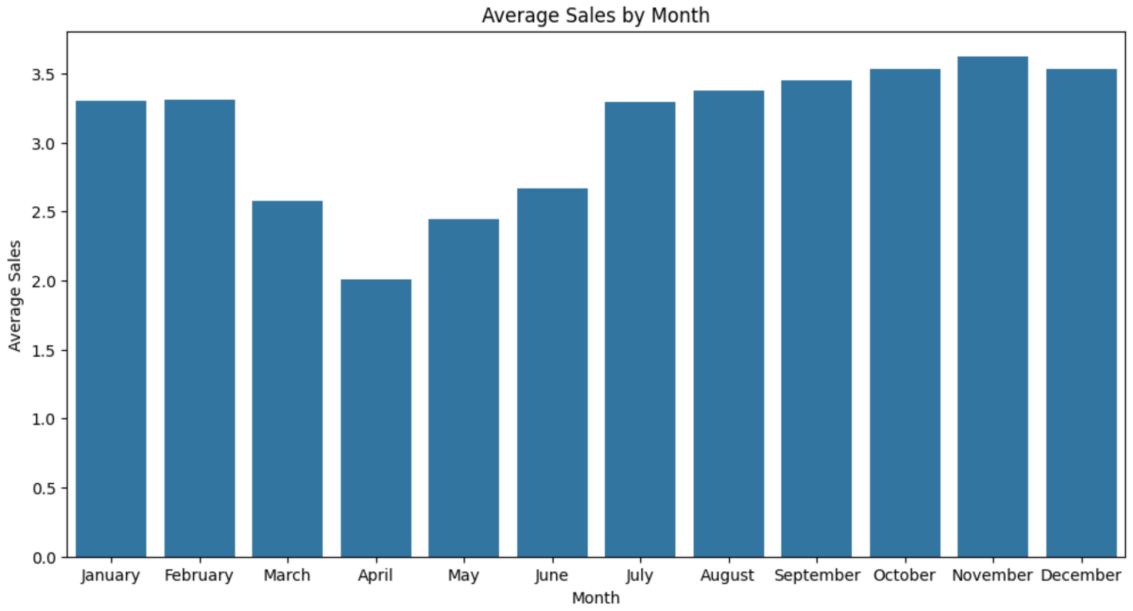


- We could notice from the graph plot that there is a hike in average sales on friday and there is a sales drop on sundays. We could assume that sundays could be a rest day for people leading to reduced sale of the products.

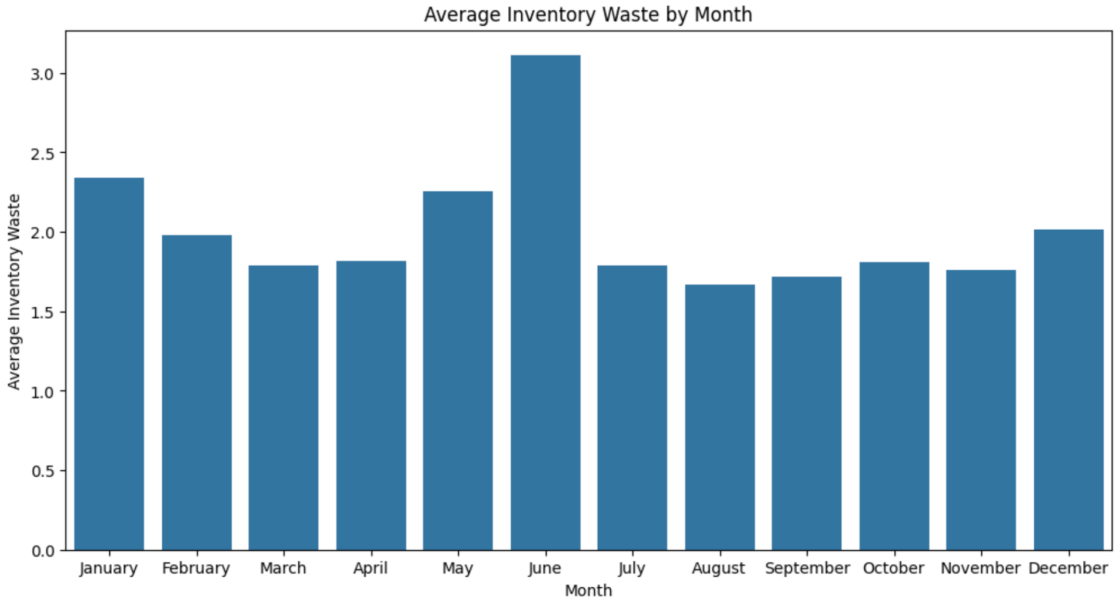


- We could notice that Wednesdays and Sundays are the two days of the week where there is increased inventory waste.

b. Monthly changes and patterns (for the duration of the data)



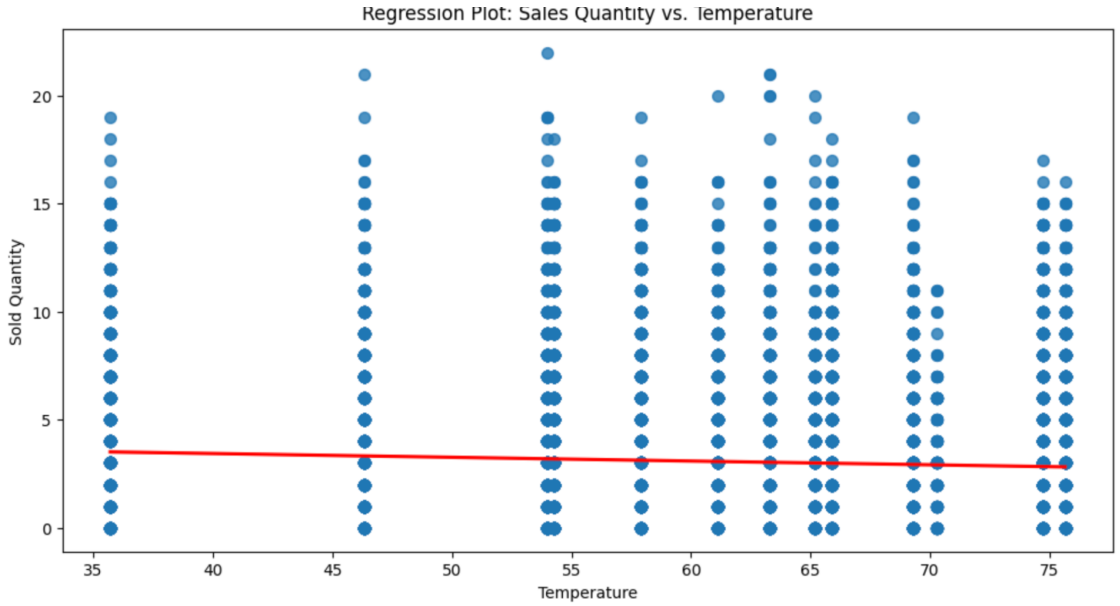
- We could notice that average sale of the products reached the maximum during november and the least in april.We could also observe a pattern that the average sale continues to increase from april to november .There is a drop in sale during the month of december since its the end of the year.



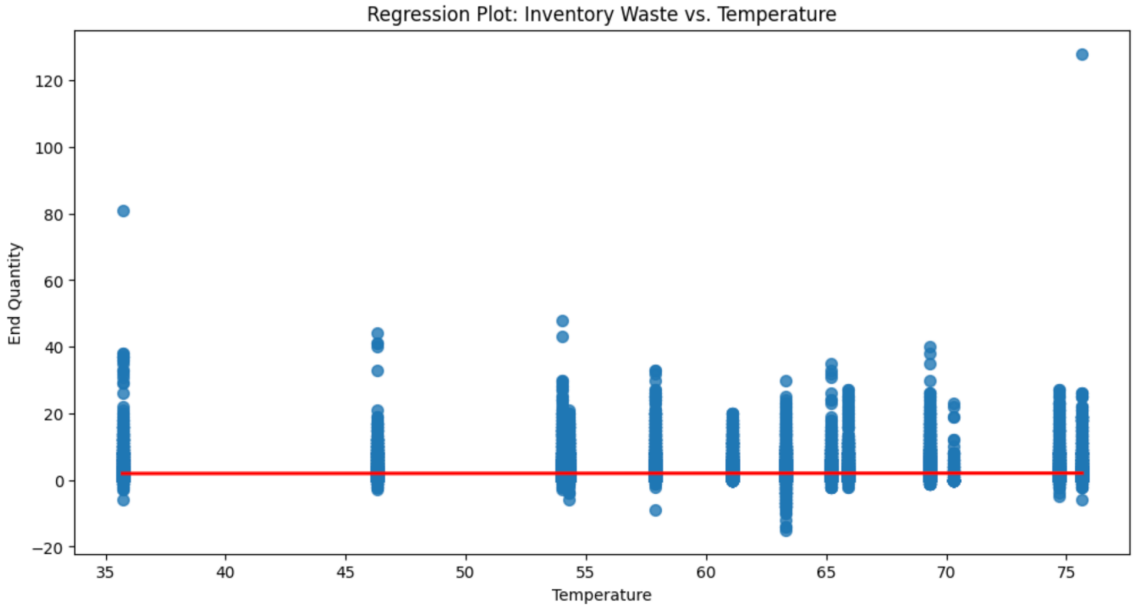
- We can understand from the above graph that the inventory waste reached the maximum during the month of June and the minimum during the month of august.
- Proper investigation and analysis could be made in order to understand the root cause for the inventory waste in the month of June and necessary steps could be taken to arrest that.

c. Impact of weather condition based on two factors:

i. Temperature

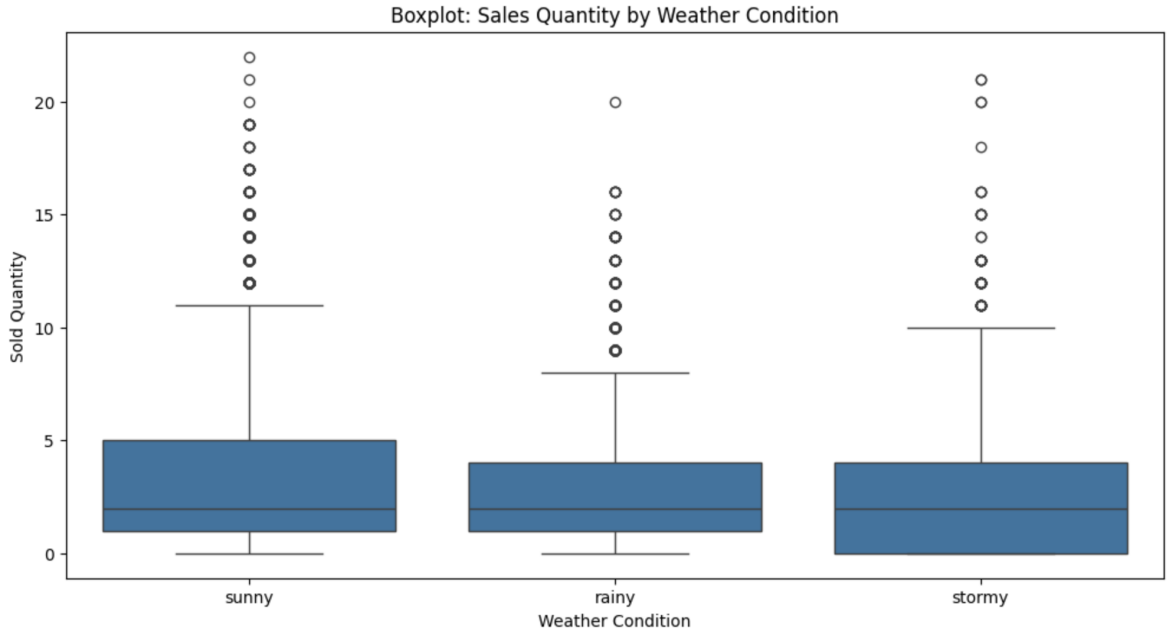


- It depicts that the sale decreases and the temperature increases.
- Note: Temperature has been added as an additional feature which is taken from the US historical temperature dataset available on the internet.



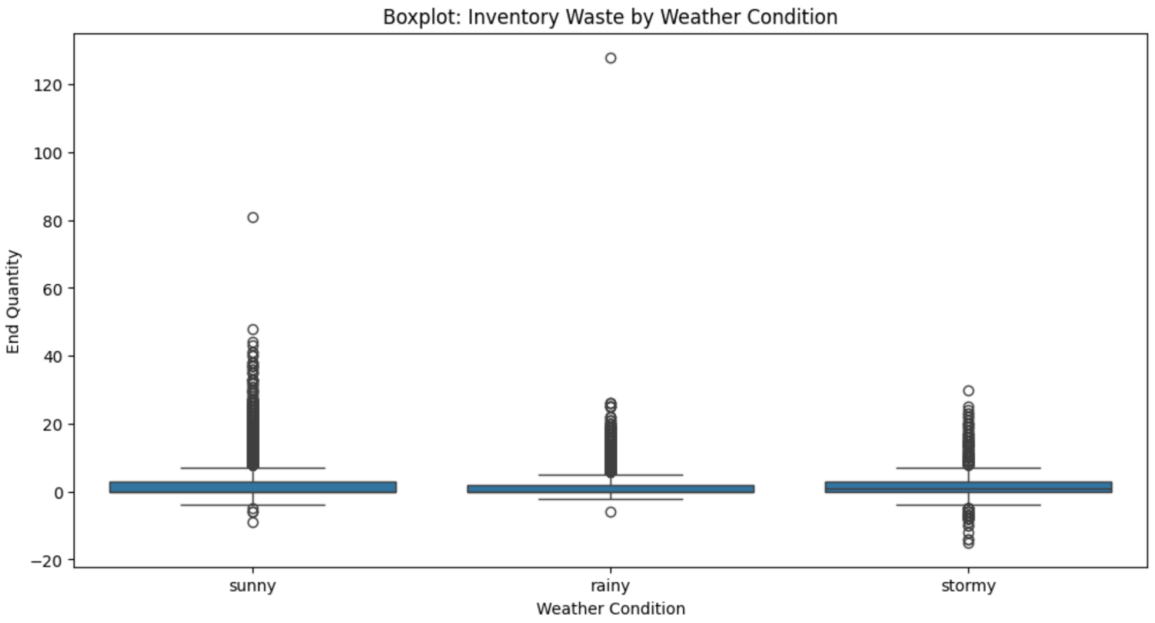
- The graph depicts that the inventory wastage rises and the temperature rises.
- The root cause should be studied and necessary steps need to be taken to address the issue which includes installing a cooking system which prevents expiration.

ii. Weather conditions (sunny, rainy, cloudy, etc.)



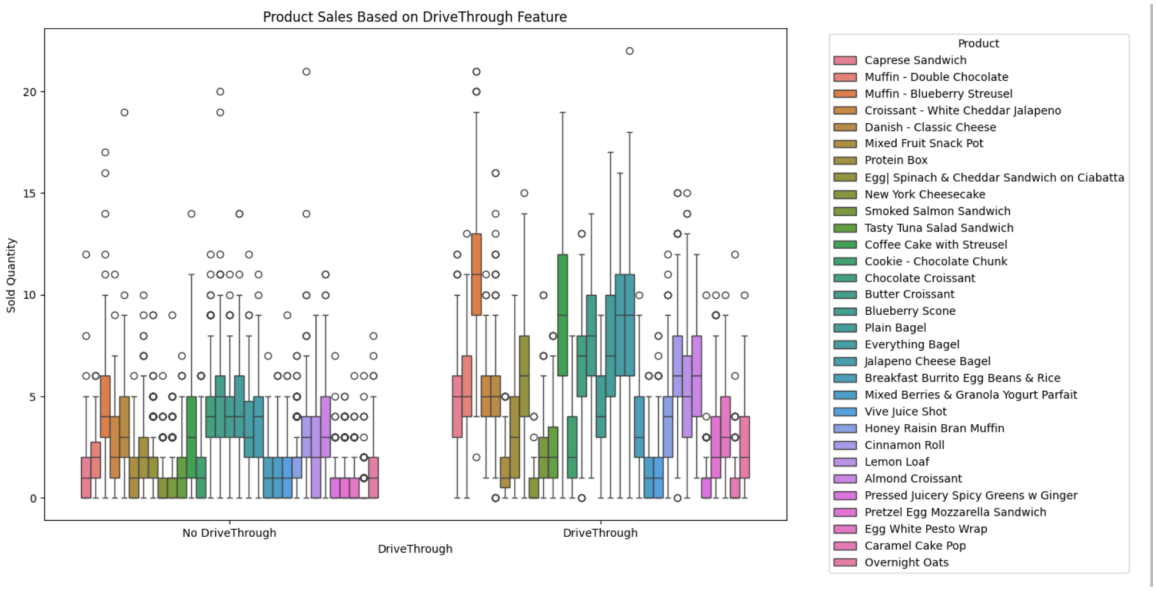
Note: Weather is added as an additional feature which is dependent on three other features present in the dataset namely temperature,humidity and precipitation.

- It could be understood from the box plot that there are increased sales when the weather is sunny and decreased sales during the rainy season. People might find it difficult to commute to the store and purchase products.Hence weather creates a significant impact on the sale of products and revenue of the company



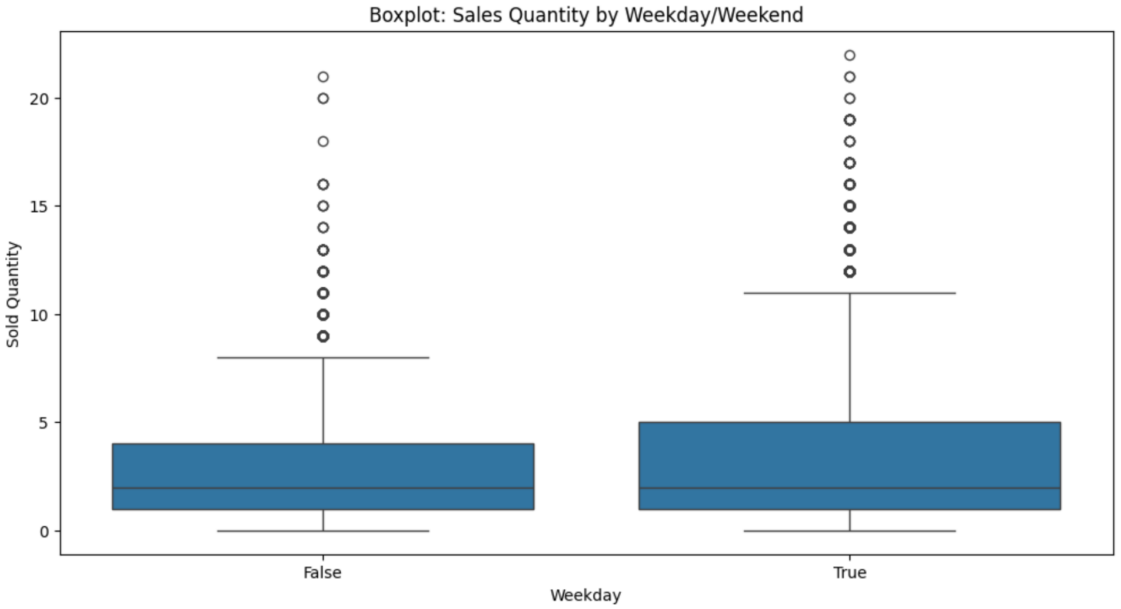
- Inventory waste has peaked in the sunny season whereas it has dropped noticeably in the rainy season.

6.Investigate whether drive thru feature causes certain products to sell better or worse

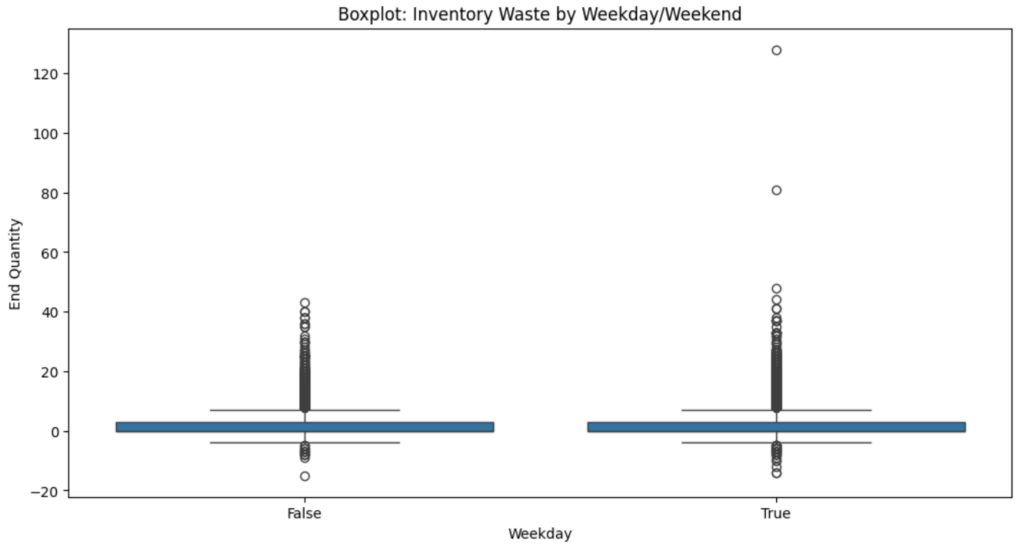


- Yes, some products like butter croissants, blueberry muffins,plain bagel get sold more in drive through rather than dining.
- This could happen due to the busy schedule of the people.

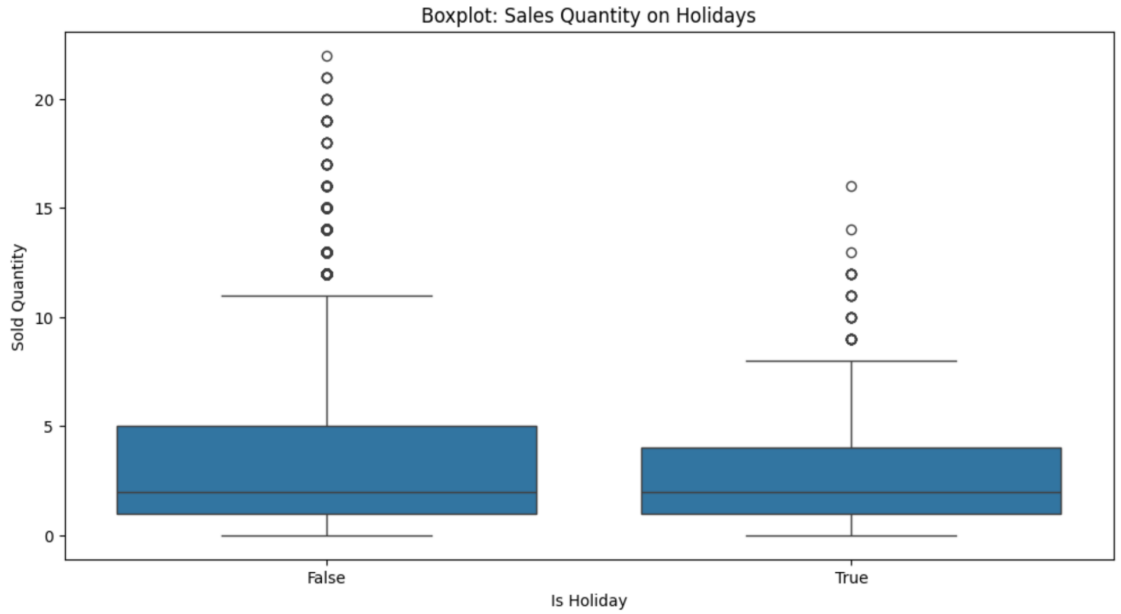
7.Investigate the impact of weekday/weekends and National Holidays by adding extra features.

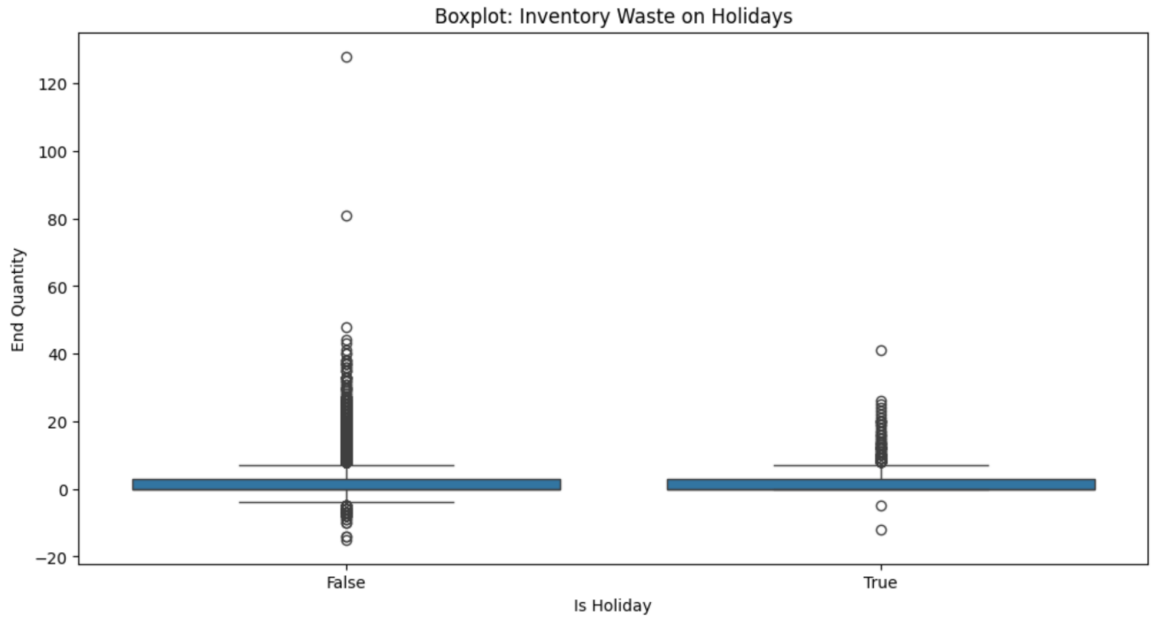


- During weekdays sales quantity increases.
- People with busy schedules may quickly grab something from the coffee shop before or after work.



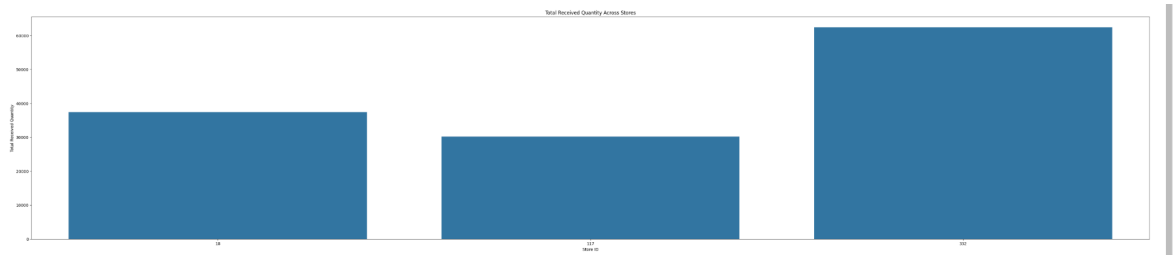
- During weekdays, More products expire. This is happening because more products are produced because of the demand and there is not enough supply.





- In the below graph, higher bars indicate that a particular store receives a larger quantity of products, while lower bars suggest smaller quantities.

8.Based on the store data, identify the stocking patterns across multiple stores. Are they provided with new products every day or restocking happens less frequently based on your insights?



- Higher bars indicate that a particular store receives a larger quantity of products, while lower bars suggest smaller quantities.This indicates variations in the stocking strategies or demand patterns across stores.

9.Draw conclusions and suggest a recommendation to optimize the stocking. We will have a deeper dive into it in section 3

Identify Top and Bottom Products:

- Identified top-performing and bottom-performing products based on sales quantity and inventory management.
- Products in the top 25% consistently perform well, while those in the bottom 25% may need attention.

Seasonal and Weather Impact:

- Analyzed the impact of weather conditions on sales.
- Consider factors such as temperature, humidity, and precipitation to understand their influence on product sales.

Drive-Through Impact:

- Investigated the impact of the drive-through feature on product sales.
- Analyzed whether certain products sell better or worse when the drive-through option is available.

Sensitivity Analysis:

- Conducted a sensitivity analysis on model parameters to fine-tune the default values.
- Explored opportunities to improve machine learning models, which is crucial for accurate sales predictions.

Sales Forecasting:

- Utilized machine learning models to predict sales for different time horizons (1 day, 3 days, and 10 days ahead).
- Evaluated the performance of Random Forest, Gradient Boosting, and XGBoost models for forecasting.

Sales Forecasting:

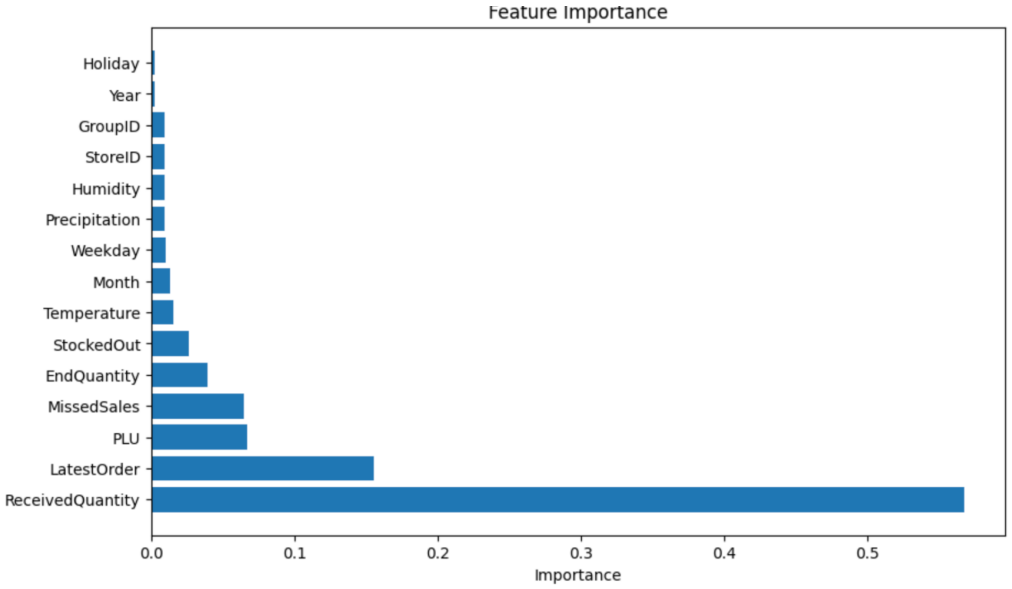
- Utilized machine learning models to predict sales for different time horizons (1 day, 3 days, and 10 days ahead).

- Evaluated the performance of Random Forest, Gradient Boosting, and XGBoost models for forecasting

Part B: Analyze the data of stores and draw insights based on the information:

SECTION 2:

2. Iterate through different combinations of features to identify the optimal features and remove potential correlated features (if any) for your predictions. Add weather, weekdays, holidays and temperature data to your features.



- SoldQuantity is the target variable here.
- Among the features present in the dataset the most important feature seems to be the ReceivedQuantity.This is done by
- After analysis the following features are selected as optimal features (received quantity, latest order,end quantity).
- The above mentioned optimal features were selected using correlation analysis, recursive feature elimination and L1 regularization. The threshold value used to detect highly correlated features is 0.5.
- Recursive feature elimination was used to remove the least important feature (holiday,group ID and store ID).

3. Start with a quick linear regression to get a sense of data. Linear regression

may not result in a great prediction

```
# 3. (optional) Start with a quick linear regression to get a sense of data. Linear regression
# may not result in a great prediction
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df[selected_features]
y = df[target_feature]

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

model = LinearRegression()
model.fit(X_train, y_train)

from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'LinearRegression Mean Squared Error: {mse}')
print(f'for LinearRegression R-squared: {r2}')
```

```
LinearRegression Mean Squared Error: 3.7833389486966955e-27
for LinearRegression R-squared: 1.0
```

4.Use ensemble models. Develop the following models and compare the accuracies by appropriate metric.
a. Random Forest
b. Gradient Boosting Machine
c. XGBoost
Perform a quick sensitivity analysis on the parameters of the model and try to finetune the default values where you see an opportunity to improve the model. Improving the machine learning models is where a data scientists will shine in their career and ahead of the game from others

Output
Random Forest Mean Squared Error: 0.0001660456828122468
Random Forest R-squared: 0.9999806628534806
Gradient Boosting Mean Squared Error: 3.251981650886302
Gradient Boosting R-squared: 0.6212846693974187
XGBoost Mean Squared Error: 3.7885913879969206
XGBoost R-squared: 0.5587928241746679

- Sensitive analysis was carried out to identify the key hyperparameters that significantly impact the model. In order to determine the best values for the hyperparameter, grid search method was used. This is followed by training the model with the best parameters and evaluating the performance.
- Finally cross validation was used to estimate the model's performance.

5. Document and highlight model improvements. Extra credit will be considered for team's effort on improvements.(5 points)

Random Forest:
Model Type: RandomForestClassifier
Default Parameters: n_estimators=100, max_depth=None, min_samples_split=2

Optimize Hyperparameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 50}

Mean Squared Error on Test Set: 0.0007822378641916154

Gradient Boosting Machine:

Model Type: Gradient Boosting

Default Parameters: n_estimators=100, random_state=42

Optimize Hyperparameters: {'learning_rate': 0.01, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}

Mean Squared Error on Test Set: 3.2519816508863015

XGBoost:

Model Type: XGBoost

Default Parameters: n_estimators=100, random_state=42

Best Hyperparameters for XGBoost: {'colsample_bytree': 0.8, 'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight': 1, 'n_estimators': 50, 'subsample': 0.8}

Mean Squared Error on Test Set: 3.6479464563640542

6. As we are dealing with time-series data, we would like to compare the results of the previous models with the following deep methods:

a. CNN

```
Epoch 1/10
772/772 [==============================] - 1s 1ms/step - loss: 0.9300 - val_loss: 0.9427
Epoch 2/10
772/772 [==============================] - 1s 1ms/step - loss: 0.9075 - val_loss: 0.9493
Epoch 3/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8982 - val_loss: 0.9503
Epoch 4/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8934 - val_loss: 0.9436
Epoch 5/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8894 - val_loss: 0.9389
Epoch 6/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8862 - val_loss: 0.9365
Epoch 7/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8825 - val_loss: 0.9305
Epoch 8/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8799 - val_loss: 0.9344
Epoch 9/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8754 - val_loss: 0.9350
Epoch 10/10
772/772 [==============================] - 1s 1ms/step - loss: 0.8727 - val_loss: 0.9383
193/193 [==============================] - 0s 406us/step - loss: 0.9383
Test Loss: 0.9382756352424622
193/193 [==============================] - 0s 414us/step
CNN Mean Squared Error: 0.9382760009814237
CNN R-squared: 0.0906252687406599
```

b. LSTM

```
Epoch 1/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8758 - val_loss: 0.8747
Epoch 2/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8574 - val_loss: 0.8710
Epoch 3/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8528 - val_loss: 0.8694
Epoch 4/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8509 - val_loss: 0.8675
Epoch 5/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8488 - val_loss: 0.8736
Epoch 6/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8465 - val_loss: 0.8774
Epoch 7/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8461 - val_loss: 0.8687
Epoch 8/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8437 - val_loss: 0.8640
Epoch 9/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8419 - val_loss: 0.8674
Epoch 10/10
772/772 [==============================] - 2s 2ms/step - loss: 0.8406 - val_loss: 0.8684
193/193 [==============================] - 0s 753us/step - loss: 0.8684
Test Loss: 0.8683895468711853
193/193 [==============================] - 0s 756us/step
LSTM Mean Squared Error: 0.8683893346380612
LSTM R-squared: 0.15835924931581125
```

c. Transformer or any sequence-to-sequence model with attention mechanism. You only need to implement one, but if you implement transformer and any other network with attention (for example, RNN with attention) I will add 5 bonus points to your final project grade

```
Epoch 1/10
772/772 [==============================] - 1s 1ms/step - loss: 8.9186 - val_loss: 7.1924
Epoch 2/10
772/772 [==============================] - 1s 1ms/step - loss: 8.4190 - val_loss: 7.1371
Epoch 3/10
772/772 [==============================] - 1s 987us/step - loss: 8.4097 - val_loss: 7.0609
Epoch 4/10
772/772 [==============================] - 1s 994us/step - loss: 8.3999 - val_loss: 7.1882
Epoch 5/10
772/772 [==============================] - 1s 1ms/step - loss: 8.3766 - val_loss: 7.3143
Epoch 6/10
772/772 [==============================] - 1s 1ms/step - loss: 8.3694 - val_loss: 7.1205
Epoch 7/10
772/772 [==============================] - 1s 1ms/step - loss: 8.3847 - val_loss: 7.1169
Epoch 8/10
772/772 [==============================] - 1s 1ms/step - loss: 8.3840 - val_loss: 7.3180
Epoch 9/10
772/772 [==============================] - 1s 998us/step - loss: 8.3667 - val_loss: 7.1285
Epoch 10/10
772/772 [==============================] - 1s 990us/step - loss: 8.3513 - val_loss: 7.2720
193/193 [==============================] - 0s 578us/step
Transformer Mean Squared Error: 7.2720073639811424
Transformer R-squared: -0.8587222764315681
```

RNN with attention

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras optimizer instead, located at `t
f.keras.optimizers.legacy.Adam`.
Epoch 1/10
  1/772 [..............................] - ETA: 11:07 - loss: 12.9108
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1702700761.583798       1 op_level_cost_estimator.cc:699] Error in PredictCost() for the op: op: "Softmax" attr { key: "T" value { type: DT_FLOAT } }
inputs { dtype: DT_FLOAT shape { unknown_rank: true } } device { type: "CPU" model: "0" frequency: 2400 num_cores: 10 environment { key: "cpu_instruction_set" val
ue: "ARM NEON" } environment { key: "eigen" value: "3.4.90" } l1_cache_size: 16384 l2_cache_size: 524288 l3_cache_size: 524288 memory_size: 268435456 } outputs {
dtype: DT_FLOAT shape { unknown_rank: true } }
769/772 [============================>.] - ETA: 0s - loss: 7.9996
W0000 00:00:1702700764.896391       1 op_level_cost_estimator.cc:699] Error in PredictCost() for the op: op: "Softmax" attr { key: "T" value { type: DT_FLOAT } }
inputs { dtype: DT_FLOAT shape { unknown_rank: true } } device { type: "CPU" model: "0" frequency: 2400 num_cores: 10 environment { key: "cpu_instruction_set" val
ue: "ARM NEON" } environment { key: "eigen" value: "3.4.90" } l1_cache_size: 16384 l2_cache_size: 524288 l3_cache_size: 524288 memory_size: 268435456 } outputs {
dtype: DT_FLOAT shape { unknown_rank: true } }
772/772 [==============================] - 4s 4ms/step - loss: 7.9946 - val_loss: 6.9268
Epoch 2/10
772/772 [==============================] - 3s 4ms/step - loss: 7.6580 - val_loss: 6.9545
Epoch 3/10
772/772 [==============================] - 3s 5ms/step - loss: 7.6148 - val_loss: 6.8675
Epoch 4/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5808 - val_loss: 7.4529
Epoch 5/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5503 - val_loss: 7.0899
Epoch 6/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5335 - val_loss: 7.6047
Epoch 7/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5053 - val_loss: 7.3895
Epoch 8/10
772/772 [==============================] - 4s 5ms/step - loss: 7.5132 - val_loss: 7.0797
Epoch 9/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5041 - val_loss: 7.4997
Epoch 10/10
772/772 [==============================] - 3s 4ms/step - loss: 7.5137 - val_loss: 7.2880
134/193 [===================>.........] - ETA: 0s
W0000 00:00:1702700796.052302       1 op_level_cost_estimator.cc:699] Error in PredictCost() for the op: op: "Softmax" attr { key: "T" value { type: DT_FLOAT } }
inputs { dtype: DT_FLOAT shape { unknown_rank: true } } device { type: "CPU" model: "0" frequency: 2400 num_cores: 10 environment { key: "cpu_instruction_set" val
ue: "ARM NEON" } environment { key: "eigen" value: "3.4.90" } l1_cache_size: 16384 l2_cache_size: 524288 l3_cache_size: 524288 memory_size: 268435456 } outputs {
dtype: DT_FLOAT shape { unknown_rank: true } }
193/193 [==============================] - 0s 1ms/step
RNN with Attention Mean Squared Error: 7.2880413889678675
RNN with Attention R-squared: -0.8587222764315681
```

7.Based on the results from the models above, we would like to predict sales based on the following scheme:

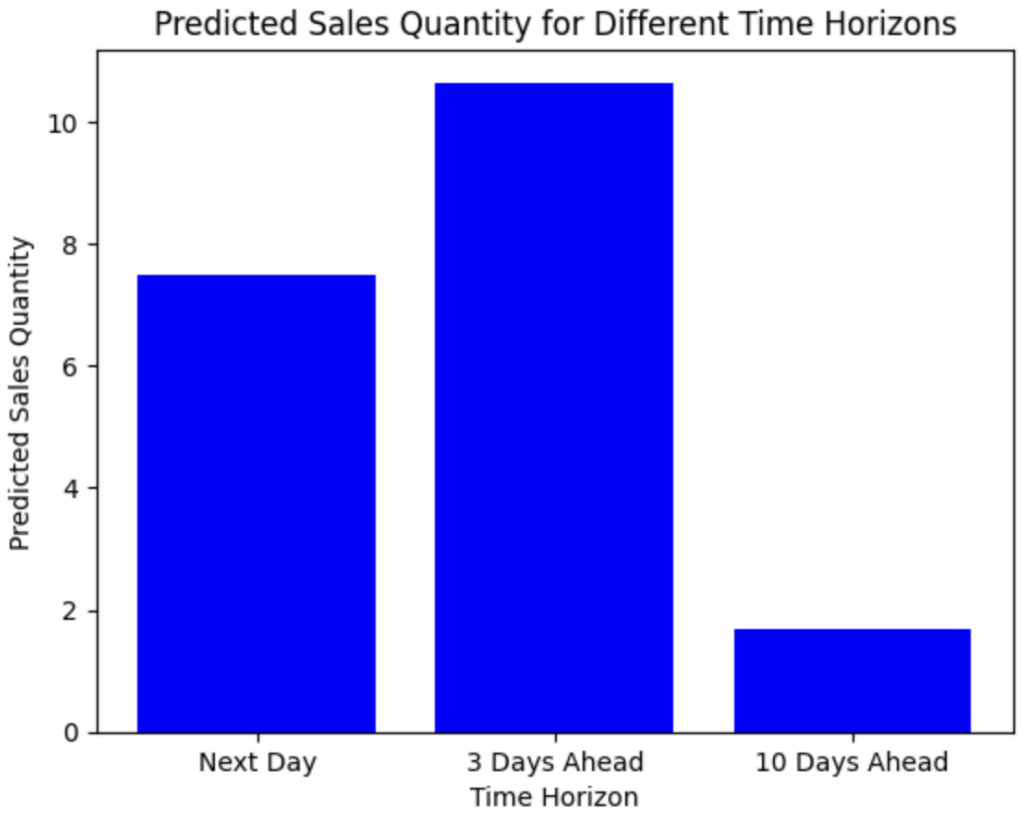
a. Weather forecast

i) 1 day ahead = [3.53442039]

ii) 3 days ahead (shipping from corporate warehouse) = [4.11685876]

iii)10 days ahead (distributor order to the manufacturer) = [2.37407712]

```
Predicted Sales Quantity for the Next Day: [7.48758333]
Predicted Sales Quantity for 3 Days Ahead: [10.64047619]
Predicted Sales Quantity for 10 Days Ahead: [1.6915]
```

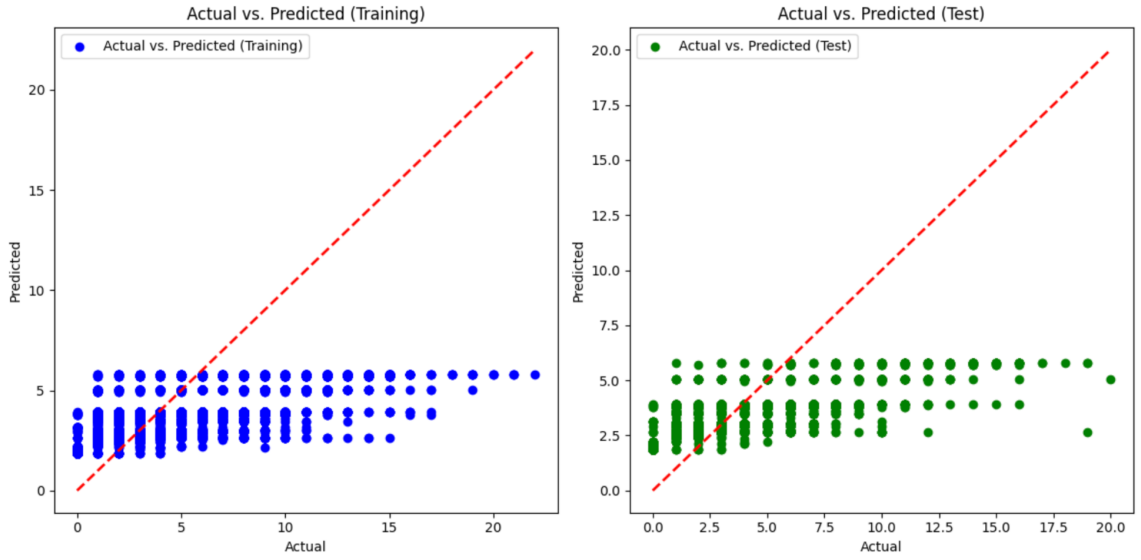


b. What models provide the best prediction for the sales forecast 1 day from today, 3 days from today and 10 days from today

Best Model for 1 Day(s) Ahead: XGBoost, MSE: 8.865983987487438
Best Model for 3 Day(s) Ahead: XGBoost, MSE: 8.875286801542119
Best Model for 10 Day(s) Ahead: XGBoost, MSE: 8.898980086184634

8.Use 80% of data for training and 20% of data for testing. Compare the model accuracy for training and test data sets

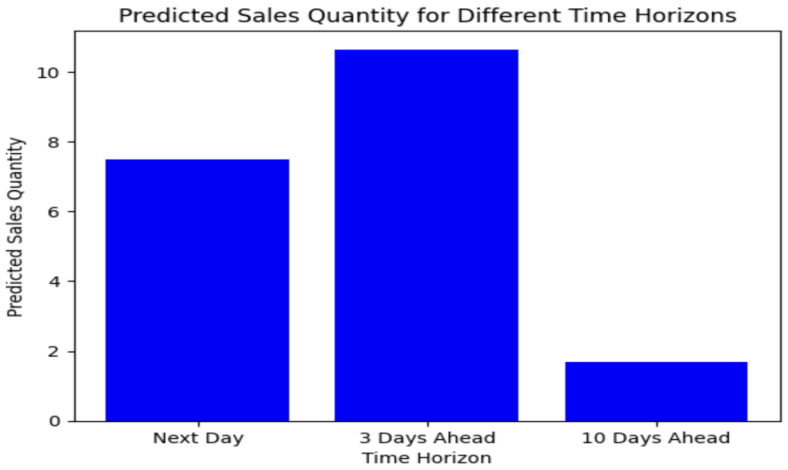
```
MSE for Training Set: 5.026506588727003
MSE for Test Set: 5.114052855670698
R-squared for Training Set: 0.4105126376824487
R-squared for Test Set: 0.410314461777483
```



10.Find the best models across all stores (the model with best predictions, i.e., the lowest error) Apply the individual store model to 10 other stores. Discuss the accuracy and where to improve. Note if the models heavily depend on the weather data, you may need to remove that feature from your data set for predictions of the remaining stores. Average US daily temperature could be a great substitute

Best Model for 1 Day(s) Ahead: XGBoost, MSE: 8.865983987487438
Best Model for 3 Day(s) Ahead: XGBoost, MSE: 8.875286801542119
Best Model for 10 Day(s) Ahead: XGBoost, MSE: 8.898980086184634

```
Predicted Sales Quantity for the Next Day: [7.48758333]
Predicted Sales Quantity for 3 Days Ahead: [10.64047619]
Predicted Sales Quantity for 10 Days Ahead: [1.6915]
```



Explainer Dashboard

Sales Prediction Dashboard

Download

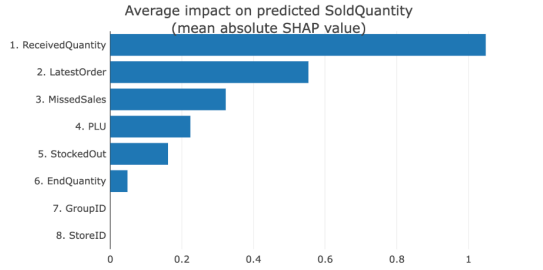
Feature Importances | Regression Stats | Individual Predictions | What if... | Feature Dependence | Decision Trees

Shap Summary

Ordering features by shap value

Depth: Summary Type

Aggregate

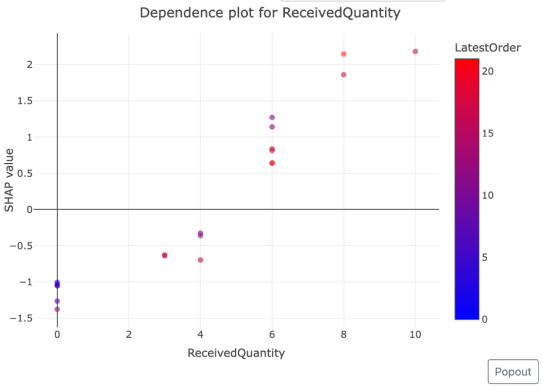


Popout

Shap Dependence

Relationship between feature value and SHAP value

Feature: ReceivedQuantit Color feature: LatestOrder Index: Select Index here...



Popout

Sales Prediction Dashboard

Download

Feature Importances | Regression Stats | Individual Predictions | What if... | Feature Dependence | Decision Trees

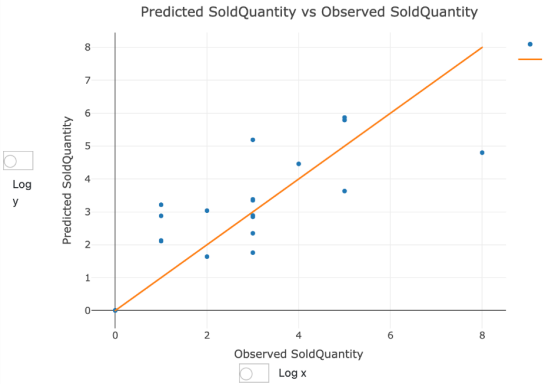
Model Summary

Quantitative metrics for model performance

metric	Score
mean-squared-error	1.646
root-mean-squared-error	1.283
mean-absolute-error	0.975
mean-absolute-percentage-error	2251799813685.74
R-squared	0.55

Predicted vs Actual

How close is the predicted value to the observed?



Popout

Sales Prediction Dashboard

Download

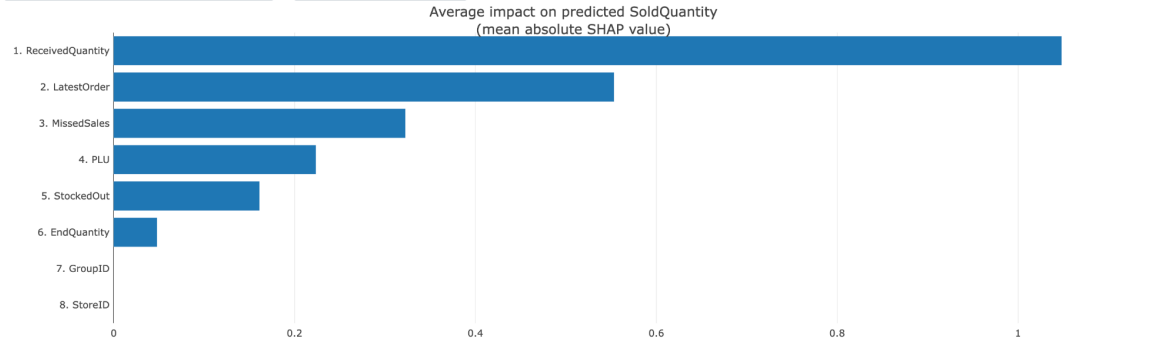
Feature Importances | Regression Stats | Individual Predictions | What if... | Feature Dependence | Decision Trees

Feature Importances

Which features had the biggest impact?

Importances type: SHAP values

Depth:



Popout