4/23/2025

MLG382 CYO Project

Waldo Blom (578068)

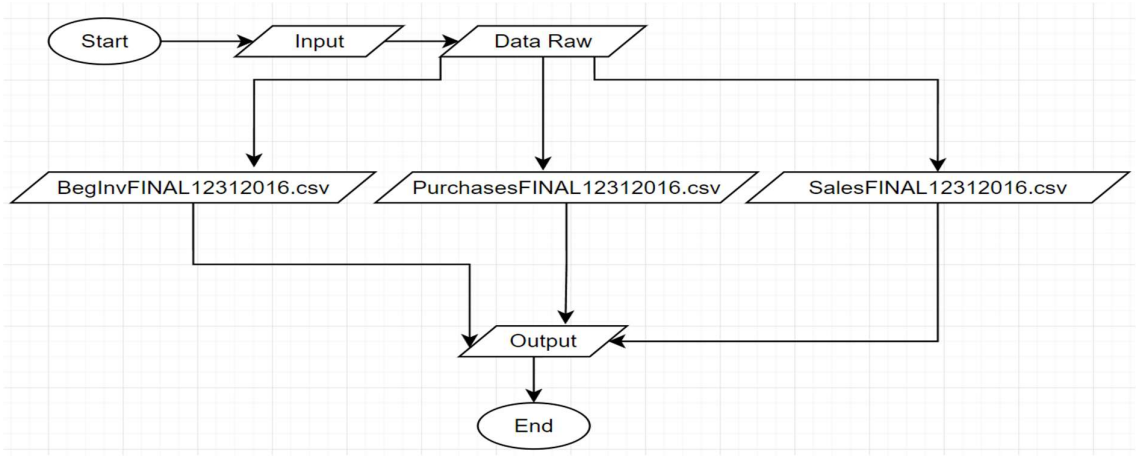
Erin David Cullen (600531)

Brandon Alfonso Lemmer (578062)

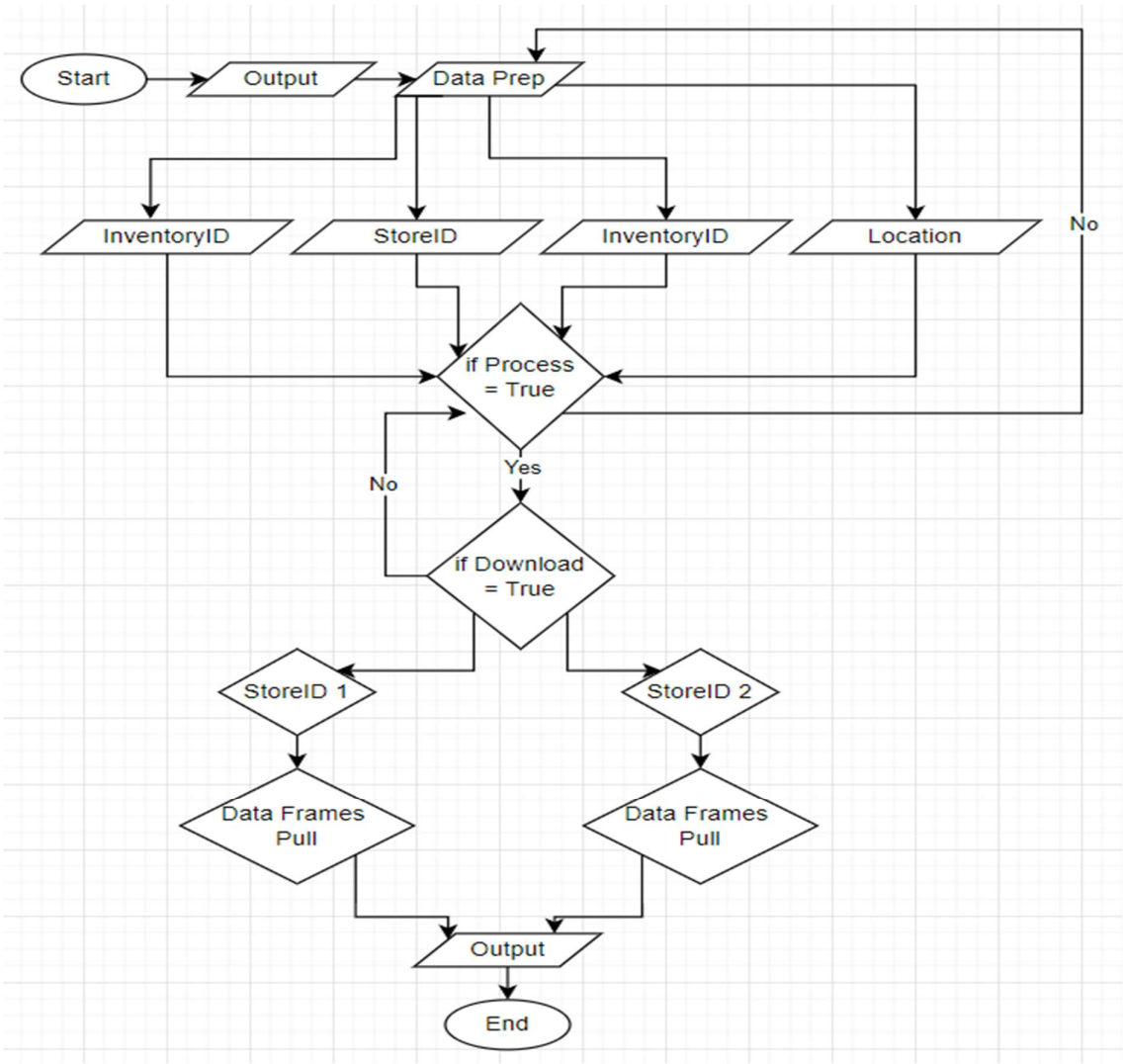
Tristan James Ball (601541)

Contents:

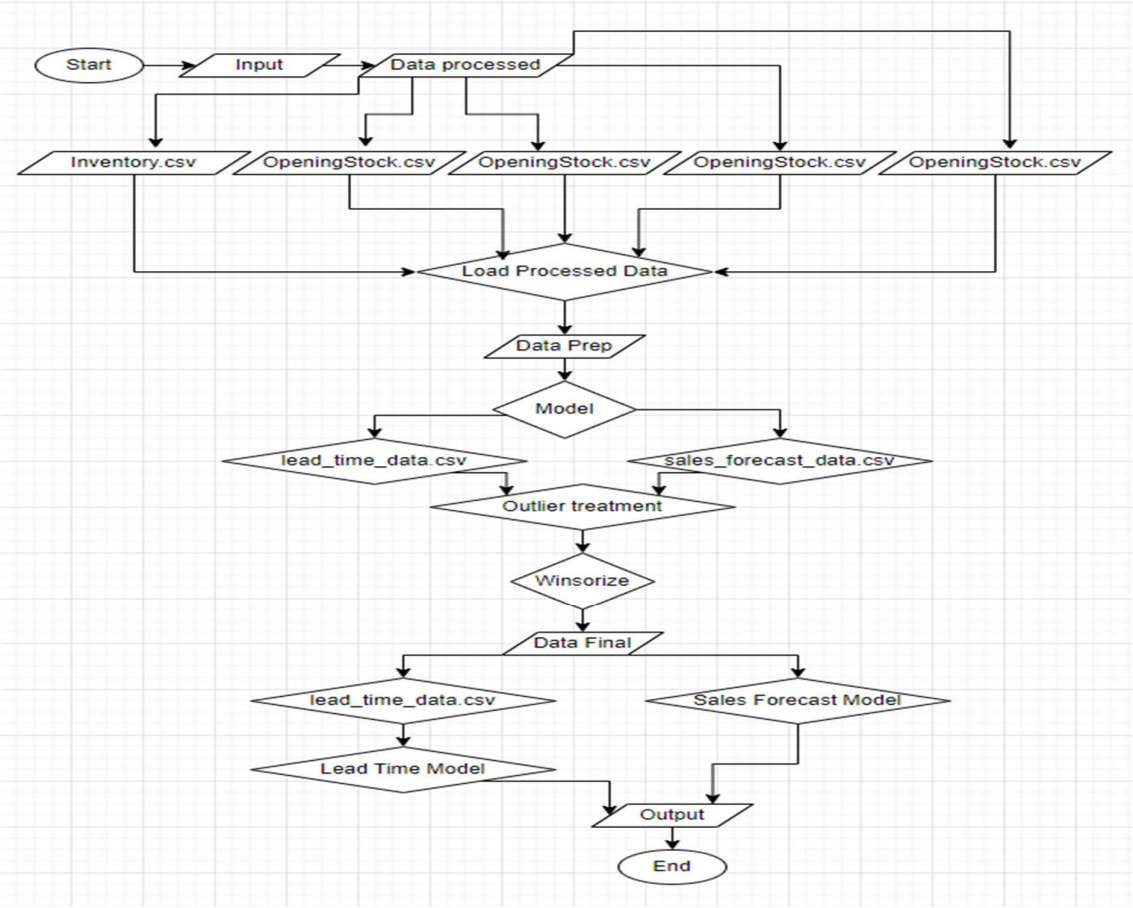
Flow chart to visualize our data flow prosses:



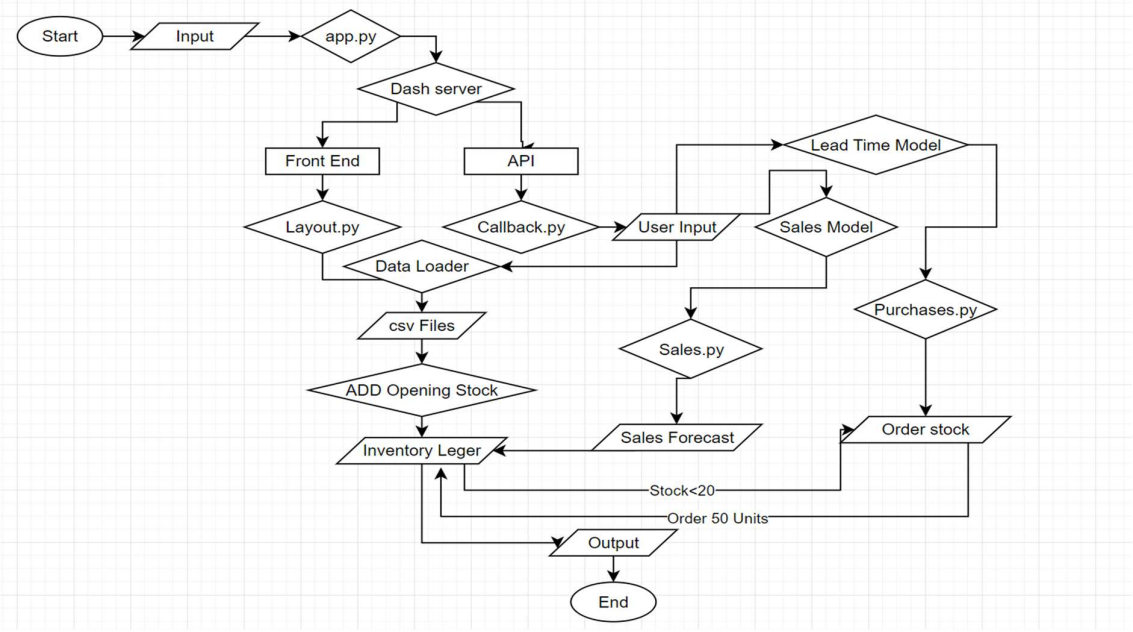
Flow chart to visualize our data preparation prosses:



Flow chart to visualize our data processing prosses through our models:



Flow chart to visualize our Web app process:



Problem statement

This is data is based on a dataset from Kaggle that has been modified. Link to Kaggle dataset: https://www.kaggle.com/datasets/bhanupratapbiswas/inventory-analysis-case-study

Optimizing Inventory Management for Tops Liquor Stores in South Africa:

Tops, a leading liquor retail chain in South Africa, operates across multiple store locations and carries a wide variety of alcoholic beverages. The brand seeks to optimize its inventory management system to enhance operational efficiency, reduce carrying costs, and ensure product availability across all its stores.

Currently, inventory data is scattered across various sources including purchase records, sales data, opening stock levels, and individual store information. However, the absence of a centralized, predictive system often results in **stockouts**, **overstocking** and inefficient **reorder processes** negatively impacting sales and increasing operational costs.

The goal of this project is to develop a data-driven inventory management solution for Tops that:

- Accurately forecasts a specific product demand (sales) at a specific store based on historical sales with the help of a XG boost machine learning model applied.
- Accurately forecasts lead times of product orders, so they know when to order new products.
- Optimizes stock levels to avoid both overstocking and stockouts.

Using available datasets on inventory, purchases, sales, and store locations, this system will help Tops make smarter inventory decisions and better align stock levels with customer demand.

Data Understanding

Data prep and load:

EDA analysis:

- Missing value analysis

```
Info of "inventory" dataframe
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11613 entries, 0 to 11612
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ItemID       11613 non-null  int64
 1   Description  11613 non-null  object
dtypes: int64(1), object(1)
memory usage: 181.6+ KB
```

From this we can see there is no data missing.

Check for duplicates:

- We only check for duplicate products as that is the only field that should not contain any duplicates. It is possible that the same purchase or sale could take place on the same day, so we do not check for any duplicates.

```
Empty DataFrame
Columns: [ItemID, Description]
Index: []
```

We can see the index is empty so there are no duplicates.

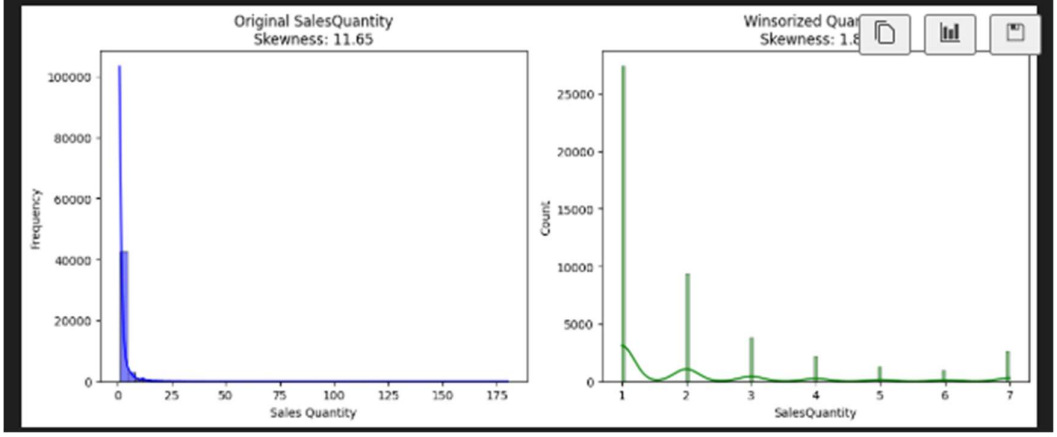
Describe data:

	count	mean	min	25%	50%	75%	max	std
StoreID	70426.00	1.40	1.00	1.00	1.00	2.00	2.00	0.49
PODate	70426	2016-06-30 16:42:12.564677888	2015-12-20 00:00:00	2016-04-03 00:00:00	2016-07-03 00:00:00	2016-10-02 00:00:00	2016-12-23 00:00:00	NaN
ReceivingDate	70426	2016-07-08 16:12:21.408002816	2016-01-01 00:00:00	2016-04-11 00:00:00	2016-07-11 00:00:00	2016-10-10 00:00:00	2016-12-31 00:00:00	NaN
Quantity	70426.00	14.34	1.00	6.00	11.00	12.00	407.00	21.39
ItemID	70426.00	12386.16	58.00	3612.00	6429.00	18743.00	90631.00	12624.44

We can gain valuable insights form the table like count, mean and standard deviation.

Checking for missing values and comparing original data against winsorized data for Sales Quantity:

```
Missing values in SalesQuantity: 0
```



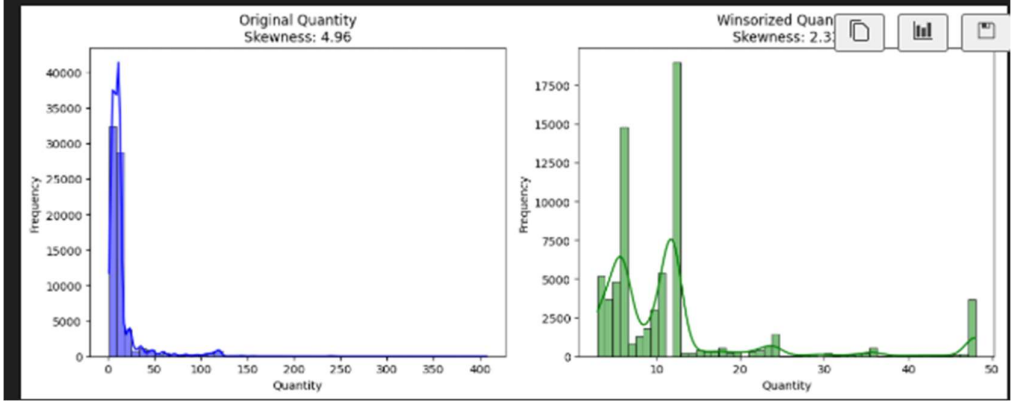

```
Original SalesQuantity Statistics:
count   47477.00
mean        2.32
std         3.30
min         1.00
25%         1.00
50%         1.00
75%         2.00
max       180.00
Name: SalesQuantity, dtype: float64

Winsorized SalesQuantity Statistics:
count   47477.00
mean        2.03
std         1.67
min         1.00
25%         1.00
50%         1.00
75%         2.00
max         7.00
Name: SalesQuantity, dtype: float64
```

We can see no data is missing and all values were successfully replaced with acceptable values.

Checking for missing values and comparing original data against winsorized data for Quantity:

```
Missing values in Quantity: 0
```

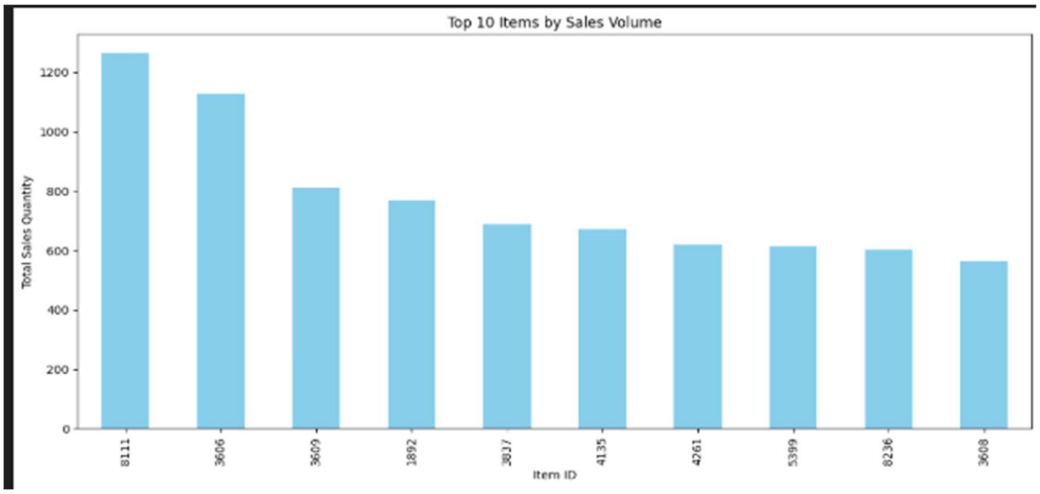



```
Original Quantity Statistics:
count   70830.00
mean       14.33
std        21.38
min         1.00
25%         6.00
50%        11.00
75%        12.00
max       407.00
Name: Quantity, dtype: float64

Winsorized Quantity Statistics:
count   70830.00
mean       12.06
std        10.63
min         3.00
25%         6.00
50%        11.00
75%        12.00
max        48.00
Name: Quantity, dtype: float64
```

We can see no data is missing and all values were successfully replaced with acceptable values.

Sales analysis:



Here we can see Item 8111 sells the best.

Purchases analysis:

Here we can see that the month Jully had the most sales.

Lead time analysis

```
Lead Time Summary Statistics:
count   70426.00
mean        7.98
std         2.23
min         3.00
25%         6.00
50%         8.00
75%        10.00
max        14.00
Name: lead_time, dtype: float64
```

Here we can see that we will wait on average 8 days for placing a new stock order till receiving it.

Final data prep:

Load the Prepped Data into:

- lead_time_data.csv
- sales_forecast_data.csv

Both files need to go through some final processing before they are ready to train models

2 functions to read, both are notarized

- create_lead_time_data
- create_sales_forecast_data

Lead_time_data.csv:

Column Name | Description

- PODate | Date the item was ordered (purchases_df) (yyyy-mm-dd)
- ReceivingDate | Date the item was received (purchases_df) (yyyy-mm-dd)
- LeadTimeDays | Target = ReceivingDate - PODate (in days)
- ItemID | Item identifier (purchases_df)
- Description | Item description (inventory_df)
- StoreID | Store identifier (purchases_df)
- Location | Store location (stores_df)
- Quantity | Quantity ordered (purchases_df)
- Week | Week number of the PODate (optional feature)
- Month | Month of PODate (optional feature)
- Day | Day of week PODate was placed (optional feature)

Sales_forecast.csv:

Column Name | Description

- SalesDate | Date of sale (daily or weekly aggregated)
- ItemID | Item identifier
- StoreID | Store identifier
- SalesQuantity | Target: quantity sold
- Lag_1 | Sales one day/week before
- Lag_7 | Sales one week before
- RollingAvg_7 | 7-day rolling average
- Month | Month of sale
- DayOfWeek | Day of week of sale
- StockOnHand | (optional) On-hand inventory if available
- ReceivedQty | Quantity received in last x days (optional)

Lead Time DataFrame:

	PODate	ReceivingDate	LeadTimeDays	ItemID	Description	StoreID	Location	Quantity	Week	Month	Day
0	2015-12-22	2016-01-01	10	5255	TGI Fridays Ultimte Mudslide	1	HARDERSFIELD	6	52	12	1
1	2015-12-20	2016-01-01	12	8358	Bacardi 151 Proof	1	HARDERSFIELD	12	51	12	6
2	2015-12-20	2016-01-01	12	4233	Castillo Silver Label Rum	1	HARDERSFIELD	23	51	12	6
3	2015-12-20	2016-01-01	12	3830	Grey Goose L'Orange Vodka	1	HARDERSFIELD	6	51	12	6
4	2015-12-20	2016-01-01	12	4670	Bacardi Dragon Berry Rum	1	HARDERSFIELD	11	51	12	6

Sales Forecast DataFrame:

	SalesDate	ItemID	StoreID	SalesQuantity	Lag_1	Lag_7	RollingAvg_7	Month	DayOfWeek
0	2016-01-01	1004	1	1	0.0	0.0	0.0	1	4
1	2016-01-02	1004	1	2	1.0	0.0	0.0	1	5
2	2016-01-03	1004	1	1	2.0	0.0	0.0	1	6
3	2016-01-08	1004	1	1	1.0	0.0	0.0	1	4
4	2016-01-09	1005	1	2	0.0	0.0	0.0	1	5

All data has been saved to the correct file, collum and row.

Data load and cleaning

Load and clean data:

Creates 5 clean tables and exports them to Data/Processed:

Inventory.csv:

- ItemID - Gives the ItemID
- Description - Gives the Item Description

OpeningStock.csv:

- StoreID - StoreID
- onHand - How Much Stock is on hand
- startDate - 2015-12-31 stock taken for the start of 2016
- ItemId - Gives the ItemID

Purchases.csv

- StoreID – StoreID
- PODate - Purchase Order Date
- ReceivingDate - Order Revieved Date
- Quantity - Quantity Recieved in Order
- ItemId - Gives the ItemID

Sales.csv

- StoreID - StoreID
- SalesQuantity - Quantity of item sold
- SalesDate - Date that the sale took place
- ItemId - Gives the ItemID

Stores.csv

- StoreID - StoreID
- Location - Location of the store

Web application development process

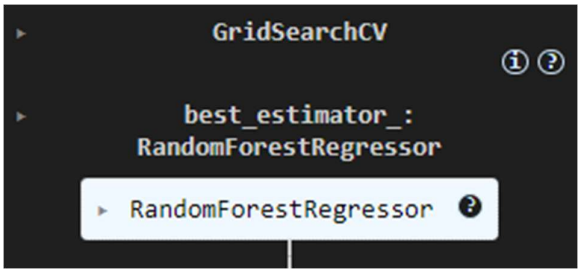
Build a dash app in repo:

- In the dash app, set up server = app.server.
- deploy repo to GitHub.
- Start a new project that runs on python in render.com.
- Link to GitHub repo.
- In Start Command set gunicorn to WebApp.app:server.

Models

Lead time model

Train Random Forest Model:



We have now successfully trained our model.

Evaluate Model:

```
MAE: 1.0788
RMSE: 1.3865
R^2: 0.6130
```

Mean Absolute Error (MAE):

- Value: 1.0788
- This model performed well as the predictions are, on average, around 1 day off actual lead time.

Root Mean Squared Error (RMSE):

- Value: 1.3865
- This metric makes up for what MAE lacks in that RMSE heavily penalizes larger errors and outliers. Its value indicates that are not many large errors as it is close to the value of MAE.

R^2:

- Value: 0.6130/~61%
- This model explains around 61% of the variance in the target. This is a solid value considering the difficulty of predicting lead times.

Sales model

Train XG Boost Model:

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=10,
             enable_categorical=False, eval_metric='rmse', feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.1, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=100,
             n_jobs=None, num_parallel_tree=None, ...)
```

Here we can see all the parameters we used to train the XG boost model.

Predictions & Evaluation:

```
Mean Absolute Error: 0.9914
Root Mean Square Error: 1.3993
R² Score: 0.2323
```

1. Mean Absolute Error (MAE: 0.9914)

- Meaning: Predictions deviate by approximately 0.99 units of Sales Quantity on average.
- Implication: Good performance on typical days (6-12 units) but may miss larger spikes (near 48).

2. Root Mean Square Error (RMSE: 1.3993)

- Meaning: Higher than MAE, indicating some larger prediction errors.
- Implication: Model struggles with outliers/high-sales events (30-48 range).

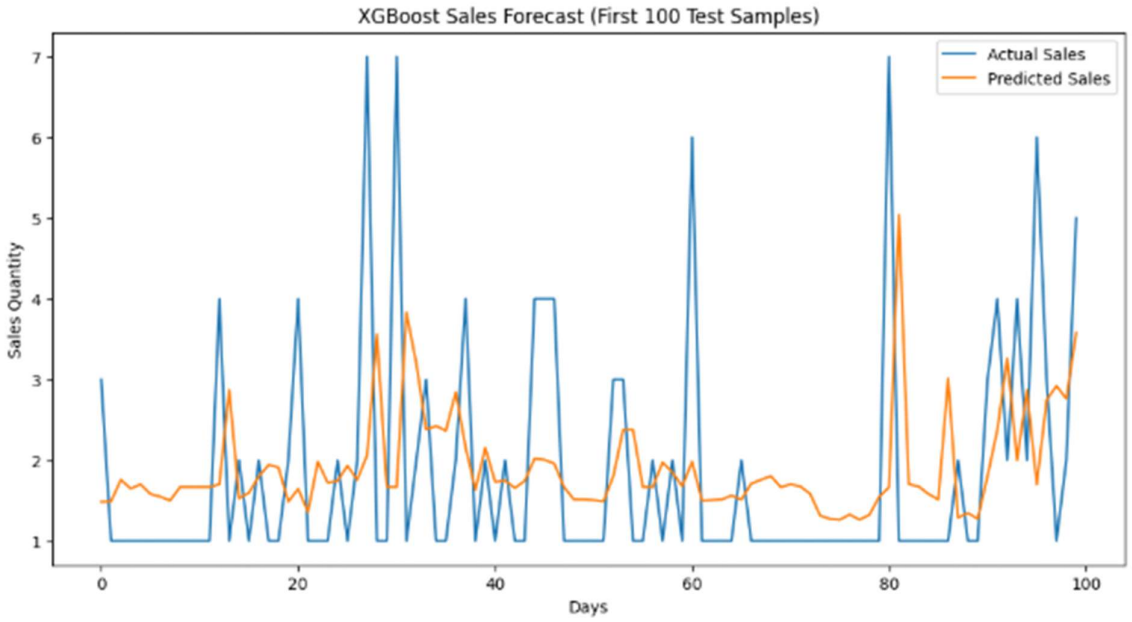
3. R^2 Score (0.2323)

- Meaning: The model explains approximately 23.23% of the variance in sales.
- Implication: Missing critical explanatory variables for high-sales days.

Strengths & Limitations

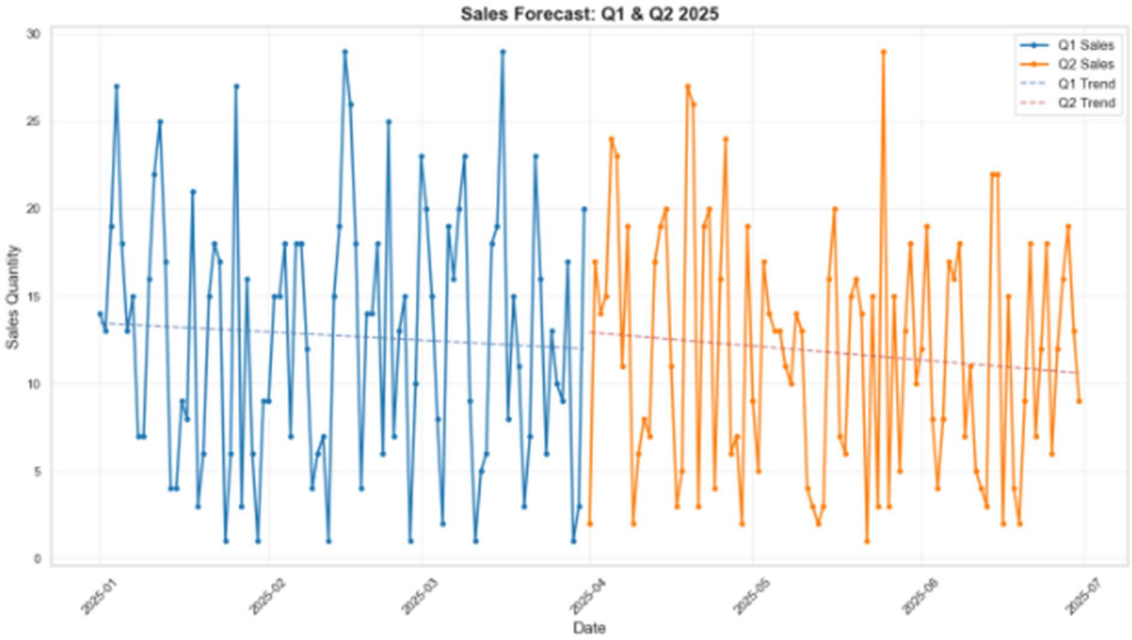
- Model performs well for average conditions.
- Struggles with extreme values, which inflate RMSE due to squared error weighting
- Model likely underpredicts high-sales events in the 30-48 range

Plot Predictions vs Actual (First 100 Days):



Here we can see the accuracy of our model. It predicts accurately within 1-3 units of stock.

Example of how the model would work:



Here we predict the next 6 months of sales.

Link to GitHub repository

[BeardedSeal77/MLG382_CYO_Project]

Link to Dash App
CYO Project on Render

Key Findings
WebApp is the folder that the web server runs in (where all the website files are).

Challenges faced
Acquiring the data was a challenge, understanding how the data would lead to predictions, dash app deployment and how to set up the dash app.

Lessons Learned
How to deploy dash app.