# 

Select type of CT to forecast:

CostcoBrand1 v

You selected: CostcoBrand1

Previous Forecast Summary of CostcoBrand1

📈 Avg Volume

▲ NORM %

MAPE MAPE

▲ AHT

1000

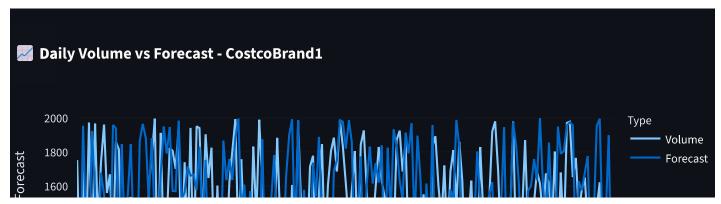
102%

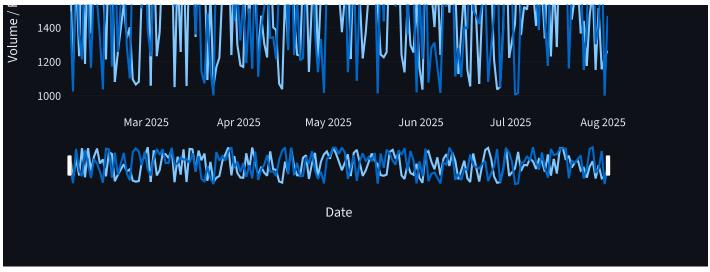
12

250

#### Week over week Summary

	Week Start Date	Volume	AHT	NORM (%)
0	2025-06-30	1821	337	107.69
1	2025-07-07	2023	359	96.12
2	2025-07-14	1591	368	102.83
3	2025-07-21	1469	197	91.95
4	2025-07-28	1669	178	91.48





Forecast for CostcoBrand1

## 

### **Thought**

First, I needed to prepare the data for forecasting. I started by taking the <code>model\_data</code> dataframe and running it through a preprocessing step. This was to ensure the data was clean, correctly formatted, and had all the necessary features, like holiday encoding, for the model to understand. The result was a dataframe ready for analysis with 1826 rows and 13 columns.

Once the data was prepared, my next step was to generate the forecast. I used the preprocessed model\_data as input for a pre-trained LSTM (Long Short-Term Memory) model. This model is specifically designed for time-series data like call volumes. The model successfully generated a 180-day forecast, predicting call volumes from July 1, 2025, to December 27, 2025.

Finally, to understand how well the model is performing, I needed to calculate its accuracy on recent data. I calculated the Mean Absolute Percentage Error (MAPE) by comparing the model's predictions to the actual call volumes from the last seven days. This gave me a concrete metric, 28.21%, to quantify the model's recent prediction error.

#### Reason

• **preprocess**: This tool was used to prepare the raw <code>model\_data</code> dataframe for the forecasting model. It handles essential tasks like cleaning data, creating features (e.g., <code>HOLIDAY\_ENCODED</code>), and setting the

correct time-series index ( REPORT\_DT ). This step is crucial because machine learning models require data to be in a specific, structured format to function correctly.

- LSTM\_Prediction: This tool was used to generate the future call volume forecast. It employs a pretrained Long Short-Term Memory (LSTM) model, which excels at identifying patterns in sequential data like time-series. It was chosen to predict the next 180 days based on the historical patterns learned from the preprocessed data.
- Last\_week\_Mape: This tool was used to evaluate the model's performance. It calculates the Mean Absolute Percentage Error (MAPE) for the last 7 days, providing a clear and standardized metric of the model's accuracy on recent, known data. This helps to understand the model's reliability and how much its predictions deviate from actual outcomes.

#### **Forecasted DataFrame:**

	Date	Forecast
0	2025-07-01 00:00:00	18657.9382
1	2025-07-02 00:00:00	19991.6474
2	2025-07-03 00:00:00	19417.1629
3	2025-07-04 00:00:00	21643.0905
4	2025-07-05 00:00:00	13856.4958
5	2025-07-06 00:00:00	14884.2501
6	2025-07-07 00:00:00	24759.5929
7	2025-07-08 00:00:00	15918.6279
8	2025-07-09 00:00:00	19002.7656
9	2025-07-10 00:00:00	14709.791

## **@ AGENT FestivCast**

Here is the restructured output:

### **Thought**

The agent began with a three-step plan to adjust a call volume forecast for holidays. First, it aimed to identify all the holiday dates within the <code>forecast\_df</code> dataset. Second, it planned to evaluate the historical impact of these holidays on call volume. The final step was to apply these findings to create an adjusted forecast.

The agent successfully executed the first two steps, identifying the holidays and then calculating their impact. However, it encountered an error when attempting the third step, <code>final\_adjustment</code>. The agent realized this tool failed because it required a single, combined dataset containing both the holiday dates and their calculated impact values, which were currently in two separate outputs.

To resolve this, the agent proposed merging the results from the first two steps into a new dataframe. After this intermediate data preparation step, the agent re-ran the final\_adjustment tool on the newly merged data. This attempt was successful, and the agent concluded by presenting the final, holidayadjusted forecast in a table.

#### Reason

- identify\_holidays: This tool was used to scan the forecast\_df and locate all dates that are recognized as holidays. This was the necessary first step to know which dates in the forecast needed special attention.
- **holiday\_impact:** This tool was used to analyze historical data and quantify the effect each holiday has on call volume. This provided the specific adjustment values (e.g., "-500 calls" or "+10% calls") needed for the forecast.
- **final\_adjustment:** This tool was used to apply the calculated impact from the holiday\_impact tool to the specific holiday dates identified by the identify\_holidays tool. It takes the base forecast, the holidays, and their impact as input to produce the final, adjusted call volume forecast. It was used twice: once unsuccessfully on separate data inputs, and a second time successfully on a merged dataset.

#### **Forecasted DataFrame:**

	Date	Holiday	Day of Week	Forecast	dow_avg_3w	Impact_Captured
0	2025-07-04 00:00:00	Independence Day	Friday	21643.0905	27750.3333	-22.01
1	2025-09-01 00:00:00	Labor Day	Monday	16431.6707	None	None
2	2025-10-13 00:00:00	Columbus Day	Monday	27283.9337	None	None
3	2025-10-31 00:00:00	Halloween	Friday	28916.8433	None	None
4	2025-11-11 00:00:00	Veterans Day	Tuesday	34469.8929	None	None
5	2025-11-27 00:00:00	Thanksgiving Day	Thursday	30024.8261	None	None
6	2025-11-28 00:00:00	Black Friday	Friday	27103.126	None	None
7	2025-12-01 00:00:00	Cyber Monday	Monday	33119.3207	None	None
8	2025-12-24 00:00:00	Christmas Eve	Wednesday	23508.3026	None	None
9	2025-12-25 00:00:00	Christmas Day	Thursday	20888.3591	None	None

## 

Here is the restructured output:

## **Thought**

First, I needed to gather the necessary data for a seasonal comparison. My initial goal was to extract data for the same date range from two different sources: historical data from previous years contained in the model\_data dataframe, and the corresponding future period from the forecast\_df dataframe.

After successfully extracting both the historical and forecast data, I confirmed the step was complete and checked if I should proceed.

My final task was to apply a "last-mile" seasonal adjustment to the forecasted call volumes. This is a final refining step to ensure the forecast accurately reflects expected seasonal patterns. I executed this adjustment and then presented a sample of the final, adjusted data to show the result.

### Reason

• **get\_historical\_periods\_dynamic\_actual**: This tool was used to extract historical data from the model\_data dataframe. It was necessary to pull actual call volume data from previous years that

corresponded to the forecast's date range for comparison and adjustment.

- **get\_historical\_periods\_dynamic\_forecast**: This tool was used to extract the relevant data slice from the <code>forecast\_df</code> dataframe. This isolated the specific forecast period that needed to be adjusted.
- **Final\_adjustment\_Seso**: This tool was used to apply the final seasonal adjustment to the forecasted data. After gathering the historical and forecast periods, this tool performed the calculation to refine the forecast based on observed seasonal patterns, creating the LM\_Seasonal\_Adjustment column.

#### **Forecasted DataFrame:**

	Date	DAY_OF_WEEK	Forecast	Seasonal_Factor	LM_Seasonal_Adjustment
0	2025-07-01 00:00:00	Tuesday	18657.9382	0.9304	17359.3151
1	2025-07-02 00:00:00	Wednesday	19991.6474	0.8863	17718.358
2	2025-07-03 00:00:00	Thursday	19417.1629	1.2199	23686.998
3	2025-07-04 00:00:00	Friday	21643.0905	1.1384	24638.3543
4	2025-07-05 00:00:00	Saturday	13856.4958	1.3931	19303.1168
5	2025-07-06 00:00:00	Sunday	14884.2501	0.9471	14096.2943
6	2025-07-07 00:00:00	Monday	24759.5929	1.0304	25511.7385
7	2025-07-08 00:00:00	Tuesday	15918.6279	0.9304	14810.6653
8	2025-07-09 00:00:00	Wednesday	19002.7656	0.8863	16841.9239
9	2025-07-10 00:00:00	Thursday	14709.791	1.2199	17944.4748