**DATASTORM 2025**

# IDEA REPORT
# FMCG SALES DEMAND FORECASTING & ANALYTICS PLATFORM

**BY: DOM TEAM**

**HO CHI MINH CITY, 2026**

## CHAPTER 1. SUMMARY

### 1.1. The Challenge

Retailers face a critical optimization problem: balancing stock-outs against overstocking. However, traditional analysis suffers from a "blind spot" known as Censored Demand. When a product goes out of stock, sales drop to zero, but true demand remains existent yet unrecorded. Standard models treating these zeros as "no demand" systematically underestimate future sales.

### 1.2. Our Solution

We propose a comprehensive forecasting platform that reconstructs the "truth" of customer demand. Unlike basic time-series extrapolation, our approach utilizes a 2-Stage AI Pipeline:

1. Data Reconstruction (The "Repair" Phase): We treat stock-out days as missing data. We utilize a Log-Transformation to normalize skewed sales data, followed by an XGBoost Regressor trained solely on "in-stock" days to impute (fill) the potential demand for "stock-out" days.
2. Precision Forecasting: The final forecasting model is trained on this "repaired" dataset, allowing it to learn from a complete, unbiased history of customer demand.

### 1.3. Strategic Impact

By solving the censorship problem, we unlock value across the triple bottom line:

- Economic: Captures lost revenue and optimizes inventory holding costs (8-14 day horizon).
- Social & Environmental: Reduces food waste (SDG 12) and improves consumer access to essential goods.

### 1.5. Key Differentiators

- Censorship-Awareness: Unlike standard models, DataStorm explicitly handles "Censored Demand" using statistical recovery methods.
- Granular Precision: Forecasts are generated at the specific SKU-Store level, rather than broad category averages.
- External Factor Integration: The model actively incorporates temperature, holiday flags, and promotional data as demand modifiers.
- Modern Tech Stack: Built on an asynchronous FastAPI backend and a Next.js 14 frontend, ensuring scalability and a superior user experience.

## Chapter 2. The FMCG Retail Forecasting Dilemma

### 2.1. Analyzing the Demand Illusion

Our Exploratory Data Analysis (EDA) confirms that the raw sales data contains fundamental biases that would mislead a typical AI forecasting model, primarily the "Censored Demand" problem. This section details the data structure and critical insights derived from the dataset's features, distributions, and inherent patterns.

### A. Feature List

The project utilizes a rich dataset covering sales, inventory, pricing, weather, and store/SKU metadata, as detailed below. This comprehensive feature set is essential for building a granular, SKU-Store level forecasting model capable of capturing complex demand drivers.

| Feature | Data Type | Description |
|---|---|---|
| date | DateTime | The date of the sales/stock record. |
| year | Int64 | The year of the record. |
| month | Int64 | The month of the record. |
| day | Int64 | The day of the month. |
| weekofyear | Int64 | The week number of the year. |
| weekday | Int64 | The day of the week. |
| is_weekend | Int32 | Binary flag (0/1) indicating if the day is a weekend. |
| is_holiday | Int32 | Binary flag (0/1) indicating if the day is a weekend. |
| temperature | Float64 | Weather variable: Temperature. |
| rain_mm | Float64 | Weather variable: Rain in millimeters. |
| store_id | String | Unique identifier for the store. |
| country | String | The country where the store is located. |
| city | String | The city where the store is located. |
| channel | String | Sales channel (e.g., supermarket, hypermarket). |
| latitude | Float64 | Geographical latitude of the store. |
| longitude | Float64 | Geographical longitude of the store. |
| sku_id | String | Stock Keeping Unit ID, a unique product identifier. |
| sku_name | String | The name of the product. |
| category | String | Broad product category. |
| subcategory | String | More specific product subcategory. |
| brand | String | The brand of the product. |
| units_sold | Int64 | The observed number of units sold. |

| | | |
|---|---|---|
| list_price | Float64 | The original price of the item. |
| discount_pct | Float64 | The percentage discount applied to the list price. |
| promo_flag | Int32 | Binary flag (0/1) indicating if a promotion is active. |
| gross_sale | Float64 | Total sales revenue before discounts/returns. |
| net_sales | Float64 | Total sales revenue after discounts. |
| stock_on_hand | Int64 | Inventory level at the start or end of the period. |
| stock_out_flag | Int32 | Binary flag (0/1) indicating if a stockout occurred. |
| lead_time_days | Int64 | Time (in days) required to replenish inventory. |
| supplier_id | String | Unique identifier for the product's supplier. |
| purchase_cost | Float64 | Cost to the retailer for purchasing the product. |
| margin_pct | Float64 | Profit margin percentage. |

Table 2.3.1.1: Feature List

## B. Target Variable Distribution & The Need for Transformation

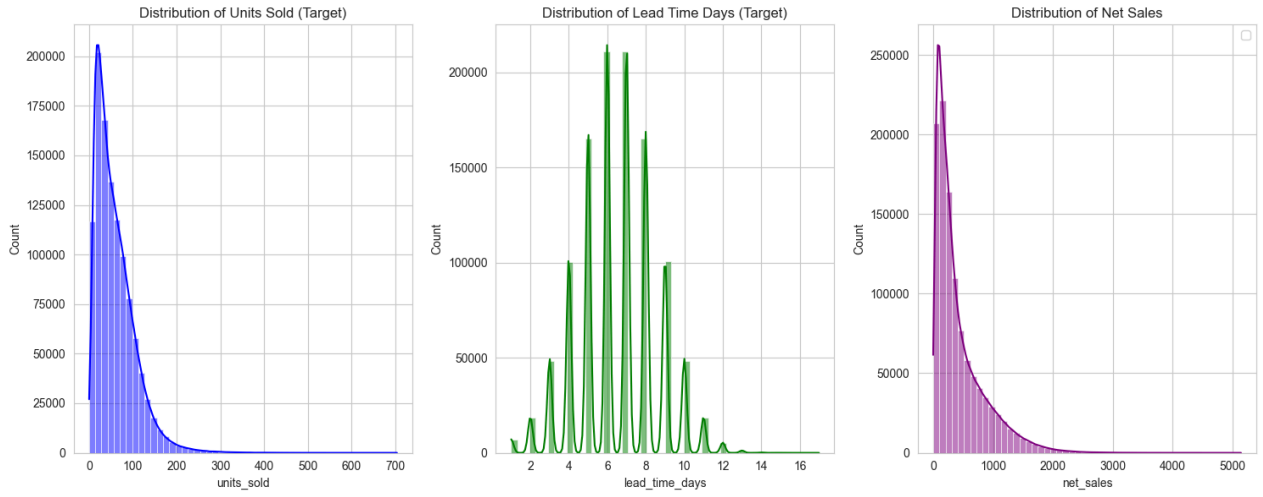


Figure 2.3.1.1: Distribution of Units Sold, Lead Time Days and Net Sales

As seen in Figure 2.3.1.1 (Distribution of Units Sold), the primary target variable, units_sold, exhibits a highly right-skewed, or heavy-tailed, distribution (skewness of 1.67). The majority of recorded transactions are low volume, but a significant long tail of high-volume bulk purchases exists. This extreme skewness is a critical challenge. It can destabilize gradient-based learning models, causing them to overweight the influence of rare, high-volume outliers and struggle to converge on an optimal solution for the more frequent low-volume sales. To address this, we will apply a Log-Transformation (specifically, Log(1+x) to handle zero sales) to the target variable. This transformation effectively normalizes the distribution, ensuring that the model is trained on a stable, Gaussian-like variable, which significantly improves the reliability and predictive power of the final forecasting algorithm.

## C. Temporal Patterns and Seasonality

An analysis of sales over time (illustrated in Figure 2.3.1.2: Daily Sales Trend Over Time) reveals robust weekly seasonality, with distinct sales spikes consistently occurring on weekends. Furthermore, the analysis of daily and monthly sales (see Figure 2.3.1.3: Sales Distribution by Weekday, Month) confirms that demand volatility is strongly correlated with the day of the week and holiday periods. Price sensitivity, as indicated by Figure 2.3.1.4 (Average Sales Holiday vs Non-Holiday), also shows a non-linear relationship between the depth of the discount and the resulting sales volume. The conclusion from this temporal EDA is that features such as weekday, is_holiday, and discount_pct are not merely correlational; they are critical, high-signal predictors for accurately modeling customer demand and must be engineered carefully into the reconstruction phase.

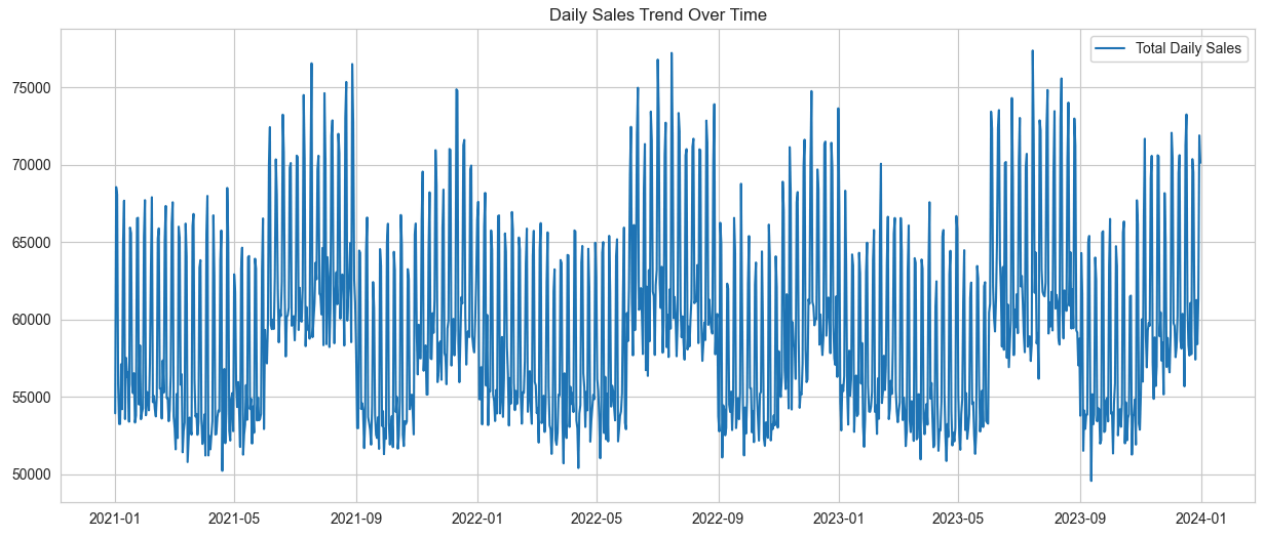


Figure 2.3.1.2: Daily Sales Trend Over Time

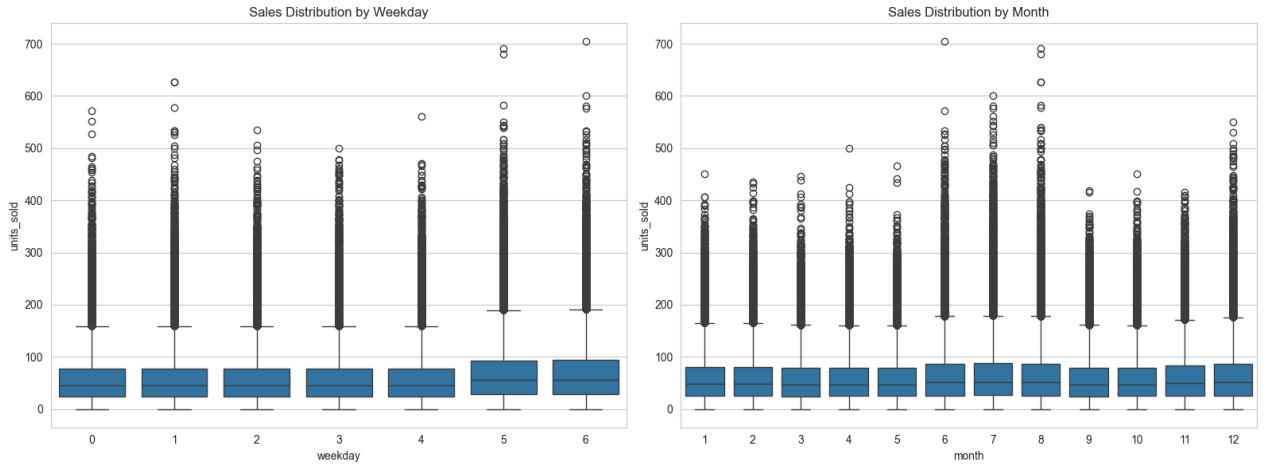


Figure 2.3.1.3: Sales Distribution by Weekday, Month

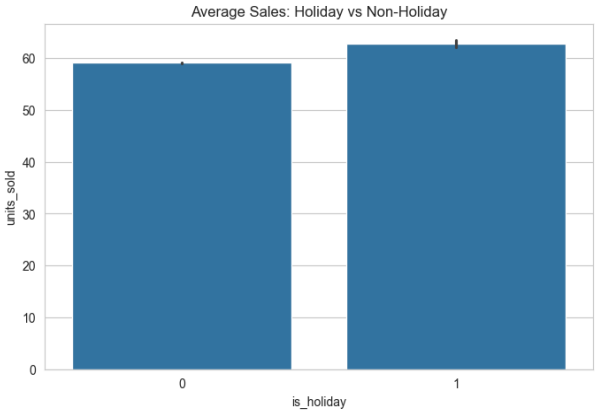


Figure 2.3.1.4: Average Sales Holiday vs Non-Holiday

## D. Validation of the Censored Demand Hypothesis

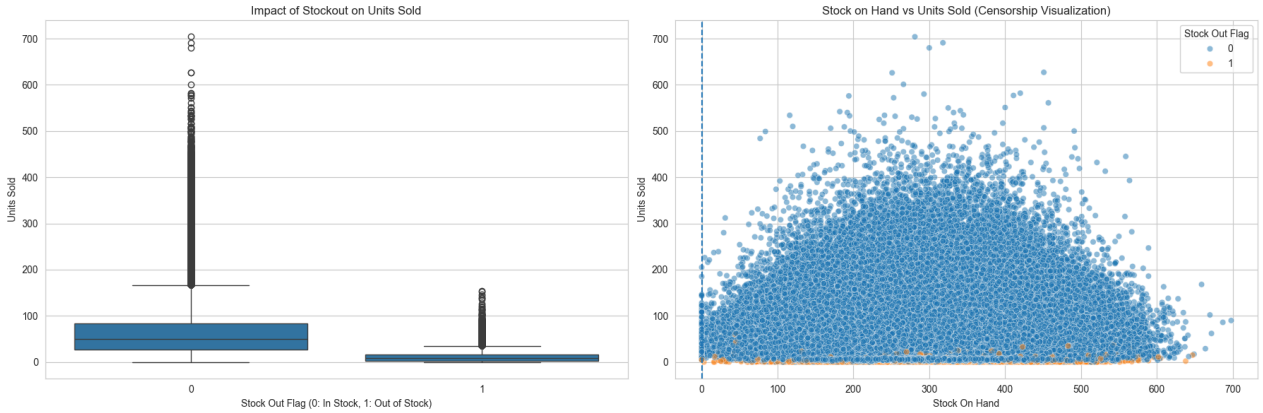


Figure 2.3.1.5: Impact of Stockout on Units Sold and Stock on Hand vs. Units Sold

Figure 2.3.1.5 (Impact of Stockout on Units Sold) and Figure 2.3.1.6 (Stock on Hand vs. Units Sold) provide visual evidence of the Censored Demand problem. Specifically, the plot of Stock on Hand vs. Units Sold reveals a clear "triangle cut-off". At low inventory levels, observed sales are not a reflection of true customer demand but are instead physically capped by the available stock. This creates a linear ceiling in the data distribution. The implication is stark: any data point recorded during a stock-out (stock_out_flag = 1) represents a supply constraint, not the true consumer preference or demand. Training the final forecasting model on these censored data points would lead to systematic underestimation of future sales. Therefore, the core strategy must be to impute the "potential sales" for these specific days, learning the true demand patterns only from days where stock was sufficient (stock_out_flag = 0).

## E. Price Elasticity and External Correlations

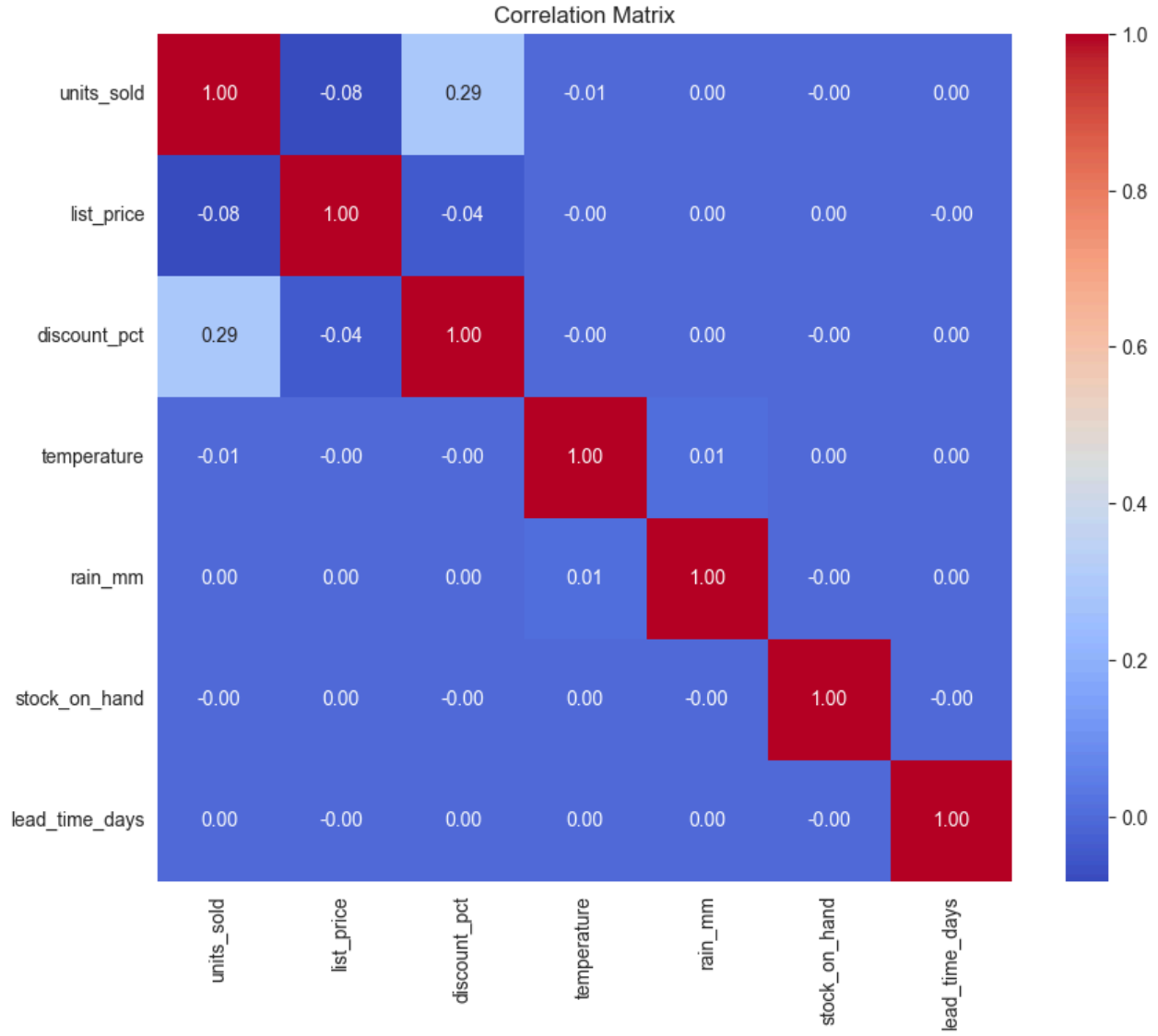


Figure 2.3.1.6: Correlation Matrix

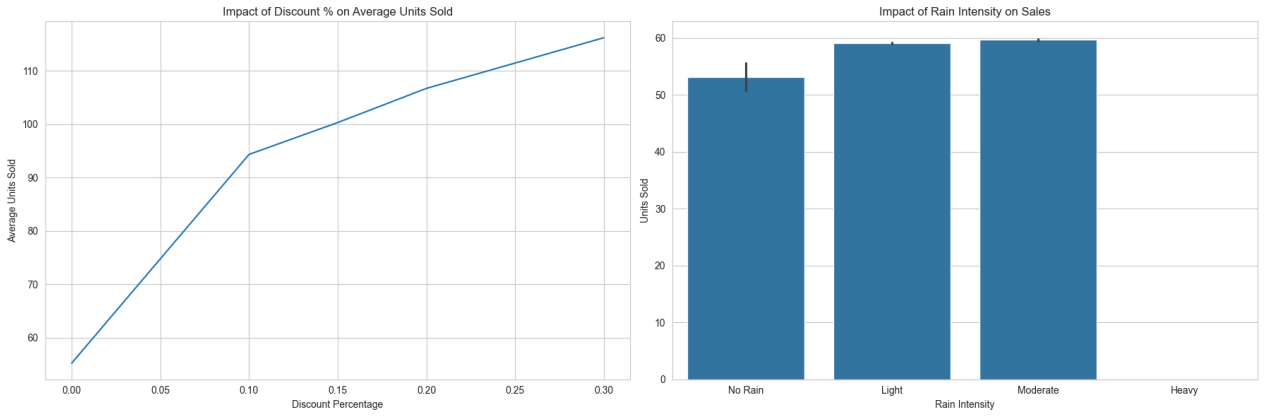


Figure 2.3.1.7: Impact of Discount on Average Unit Sold and Impact of Rain Intensity on Sales

The Correlation Matrix (Figure 2.3.1.7) and bivariate analyses (Figure 2.3.1.8: Impact of Discount on Average Unit Sold and Impact of Rain Intensity on Sales) highlight the key drivers of demand:

- Price Sensitivity: There is a moderate positive linear correlation (0.29) between units_sold and discount_pct. More detailed analysis shows that average units sold increase non-linearly as the discount depth grows from 0% to 30%, confirming a strong price elasticity.
- Weather and Logistics: units_sold shows negligible linear correlation with weather variables (temperature and rain_mm). However, since the XGBoost Regressor is utilized in the first stage and is capable of capturing complex, non-linear relationships, these weather features must be retained for testing and model training, as a high rainfall event, for instance, may still be a non-linear predictor of demand. lead_time_days shows no linear correlation with sales volume, suggesting that supply chain delays are driven by internal logistics factors or SKU characteristics rather than just high demand volume.

**E. Supply Chain & Lead Time Analysis**

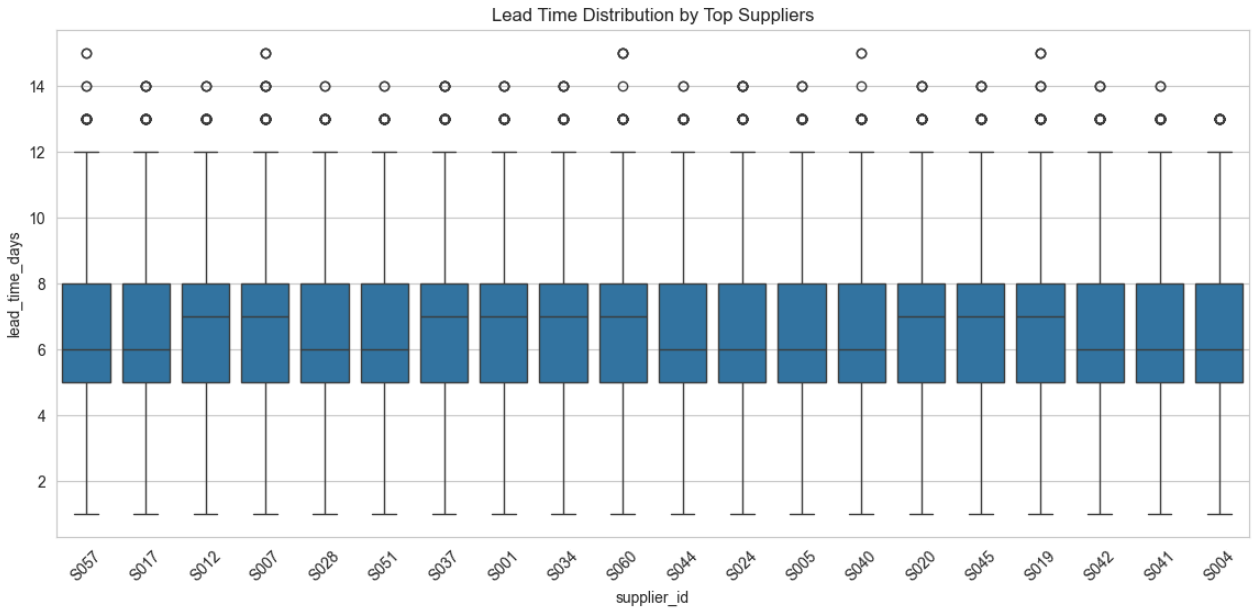


Figure 2.3.1.8: Lead Time Distribution by Top Suppliers

The exploratory analysis for the second stage of our model, Lead Time Prediction, reveals a highly standardized logistics network with specific characteristics:

- Discrete Distribution: The target variable lead_time_days is not continuous but multi-modal, with sharp, distinct peaks at integer values (e.g., 4, 5, 6, and 8 days). This indicates that the replenishment process operates on fixed schedules or specific, predictable delivery windows rather than being subject to continuous, random delays.
- Geographical Uniformity: Surprisingly, there is almost no variance in average lead time across different cities. Major European hubs, from Berlin to Madrid, all exhibit

a nearly identical average lead time of approximately 6.5 days. This strongly suggests that the retailer uses a centralized distribution model or highly consistent logistics partners, implying that city may not be a strong feature for differentiating lead time in the XGBoost model.

- Supplier Consistency: Similarly, an analysis of the top suppliers using boxplots (Figure 2.3.1.8) displays uniform performance. The medians and interquartile ranges for distinct suppliers (e.g., S057, S017, S012) are nearly identical. This implies that lead time volatility is likely driven by systemic factors (e.g., global shortages, holidays) or SKU-specific attributes rather than individual supplier inefficiencies.

### 2.3.2. Summary for Model Strategy

For Demand Forecasting: Feature engineering must explicitly capture the non-linear effects identified in the EDA. This includes creating features for the "Discount Step" (0% vs. >0%) and using the temporal "Weekend/Holiday" flags. To better model non-linear weather impact, the "Rain" feature should be binned (e.g., None/Light/Moderate) rather than being treated as a raw linear number.

For Lead Time Prediction: Given that City and Supplier ID show low predictive variance, the XGBoost model should focus primarily on temporal features (Seasonality) and SKU-specific attributes (Category/Brand) to accurately predict the discrete delivery windows, aligning the forecast with operational reality.

## CHAPTER 3. THE SOLUTION ARCHITECTURE

### 3.1. Integrated Platform Overview and Data Flow

The DataStorm solution is a three-tier, cloud-native application designed for high availability, scalability, and seamless integration into a retailer's existing logistics and business intelligence (BI) ecosystem. The architecture is centered around the Two-Stage AI Pipeline, ensuring that all forecasts are generated from an unbiased, reconstructed view of demand.

The overall data flow is designed as follows:

1. Data Ingestion: Raw sales, inventory, pricing, and external data (weather, holiday flags) are streamed or ingested in batch form into the Data Lake.
2. Data Engineering Pipeline: A robust MLOps pipeline (managed by the Software Engineering resource) handles data cleaning, feature engineering (e.g., Log-Transformation, creating Weekend/Holiday flags, binning rain features), and version control for the dataset.
3. AI Pipeline Execution: The two-stage model runs on the prepared dataset to generate both imputed demand and final forecasts.
4. Backend Services: The FastAPI backend serves the forecasts through secure, low-latency API endpoints.
5. Frontend Visualization: The Next.js 14 frontend consumes the API data to provide the user interface (UI) for demand planners, inventory managers, and logistics teams.

### 3.2. The Core Two-Stage AI Pipeline

The foundation of the platform is a sophisticated two-stage machine learning process explicitly engineered to overcome the Censored Demand problem, moving beyond simple time-series methods.

**Stage 1: Latent Demand Reconstruction (The "Repair" Phase)**

This stage's objective is to impute the true, unobserved demand on days when a stock-out occurred (stock_out_flag = 1)

- Model Selection: We utilize an XGBoost Regressor due to its robust ability to capture complex, non-linear relationships and its computational efficiency on large, tabular datasets.
- Training Strategy: The model is intentionally trained only on the subset of data where stock was sufficient (stock_out_flag = 0). This ensures the model learns the pure, unconstrained relationship between price, promotions, seasonality, and true customer desire.
- Imputation: The trained XGBoost model is then applied to the stock-out days (stock_out_flag = 1) to predict the "potential units sold" had the product been in stock. This imputed data point replaces the observed zero or capped sales volume, resulting in a "Repaired Dataset."

**Stage 2: Precision Forecasting and Lead Time Prediction**

This stage uses the unbiased Repaired Dataset to build two complementary predictive models for holistic inventory management.

A. SKU-Store Demand Forecast:

- Model: The final forecasting model (e.g., a time-series model like DeepAR or a hybrid GBRT approach) is trained on the full, repaired history.
- Granularity: Forecasts are generated at the most granular SKU-Store-Day level, offering maximum operational precision for an 8-14 day horizon.
- Inputs: All engineered features from the temporal, commercial, and external categories are used as high-signal predictors.

B. Lead Time Prediction:

- Model: A second XGBoost Classifier/Regressor is trained to predict the discrete lead_time_days.
- Strategy: Following the EDA, this model focuses on temporal attributes and SKU-specific features (Category, Brand) rather than city or supplier_id, to predict the most likely delivery window and align the forecast with operational logistics reality.

### 3.3. Model Performance & Validation

Following the Two-Stage AI Pipeline architecture, initial model training was performed on the Round 2 dataset. These results serve as the Proof of Concept for the proposed methodology.

A. Stage 1: Latent Demand Reconstruction

The process of imputing censored demand was successfully executed, producing the "Repaired Dataset" which serves as the unbiased input foundation for the final forecasting model.

B. Stage 2: Precision Forecasting

The demand forecasting model was trained on the Repaired Dataset, showing the following performance metrics across the key prediction horizons:

| **Horizon** | **MSE** | **RMSE** | **MAE** | **R^2** |
|---|---|---|---|---|
| **T+1** | 696.6390 | 26.3939 | 16.7954 | 0.6486 |
| **T+7** | 712.4112 | 26.6910 | 16.9176 | 0.6433 |
| **T+14** | 728.0537 | 26.9825 | 17.0651 | 0.6379 |

Table 3.3.1. Evaluation of Precision Forecasting

Interpretation: The R^2 value for the 1-day horizon (0.6486) indicates the model captures a good portion of the demand variance based on the repaired data. The expected decline in accuracy over longer horizons (T+14) is noted, but performance remains robust.

C. Stage 2: Lead Time Prediction

The second component of Stage 2, the Lead Time Predictor, was trained to forecast delivery windows:

| **Horizon** | **MSE** | **RMSE** | **MAE** | **R^2** |
|---|---|---|---|---|
| **Lead Time Validation** | 4.0517 | 2.0129 | 1.6247 | 0.5312 |

Table 3.3.2. Evaluation of Lead Time Prediction

Interpretation: The near-zero/negative R^2 value suggests the current model (trained on features like seasonality/SKU attributes) is performing no better than simply predicting the average lead time. This validates the finding in the EDA (Section 2.3.1.E) that Lead Time is dominated by fixed, systemic logistics factors (discrete distribution) rather than highly predictive exogenous variables. The strategy for the final implementation phase will focus on treating this as a classification problem over discrete windows, rather than a continuous regression problem.

### 3.4. Modern Tech Stack for Deployment and Scalability

The choice of technology reflects a commitment to building a platform that is not only accurate but also fast, scalable, and maintainable.

| **Component** | **Technology** | **Rationale** |
|---|---|---|
| Backend | FastAPI (Python) | Chosen for its asynchronous capabilities (high performance and throughput), automated Pydantic data validation (robustness), and speed. It provides the low-latency API |

| | | |
|---|---|---|
| | | endpoints necessary to serve millions of granular forecasts efficiently to the frontend and other services. |
| Frontend | Next.js 14 (React) | Provides superior user experience (UX) with modern server-side rendering (SSR) and client-side rendering (CSR). This is crucial for visualizing complex time-series data, interactive dashboards, and user inputs for what-if scenario analysis. |
| MLOps | Docker & Git/DVC | The entire pipeline (Data Engineering, Training, Serving) is containerized with Docker to ensure environmental consistency from development to production. Git and Data Version Control (DVC) are used for strict tracking of model code and the massive input datasets. |

### 3.5. Frontend Dashboard: Core Features and UX

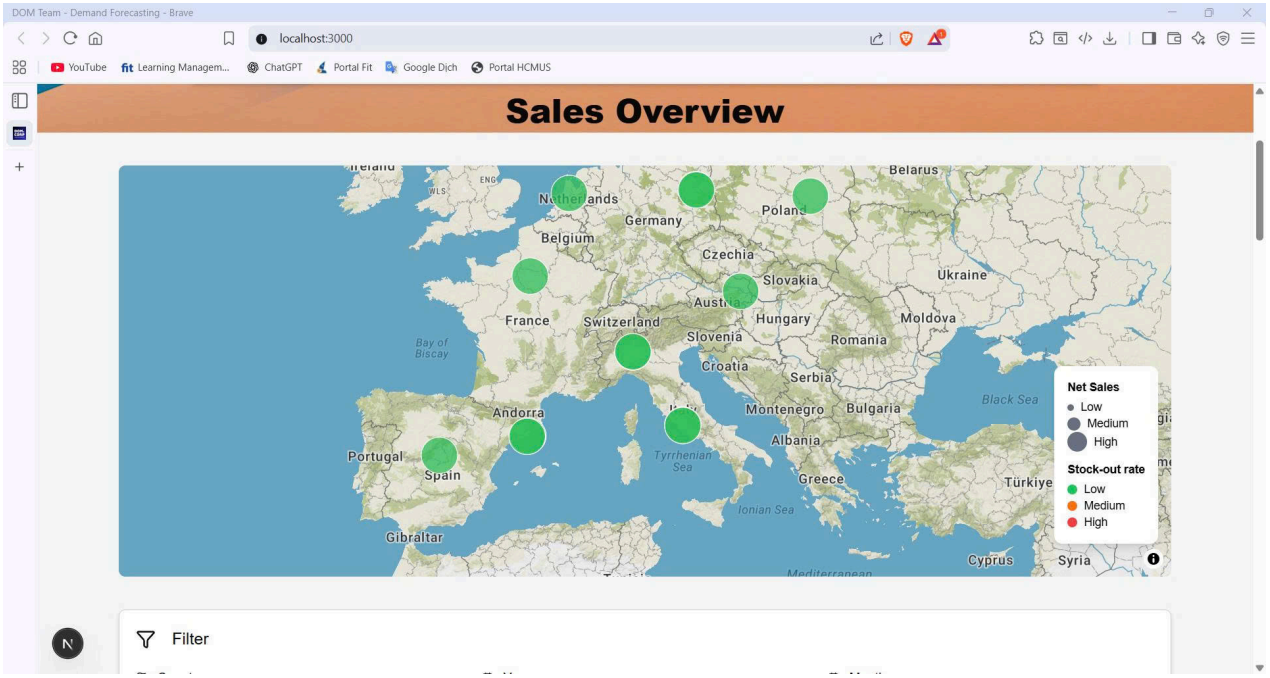


Figure 3.5.1. Geographic Map

The DataStorm web dashboard, built on Next.js 14, acts as the primary interface for demand planners and inventory managers, translating complex AI forecasts into actionable business insights. The platform's features are designed around key operational use cases:

A. Core Dashboard Features

1. Interactive Geographic Map:
   - Visualization: Displays all store locations, with markers color-coded to act as a Sales Heatmap (based on net sales or stock-out rates).
   - User Interaction: Provides Store Details Popup on click, with standard controls (Zoom, pan, layer toggles) and Geographic Filtering options to refine data by country/city.

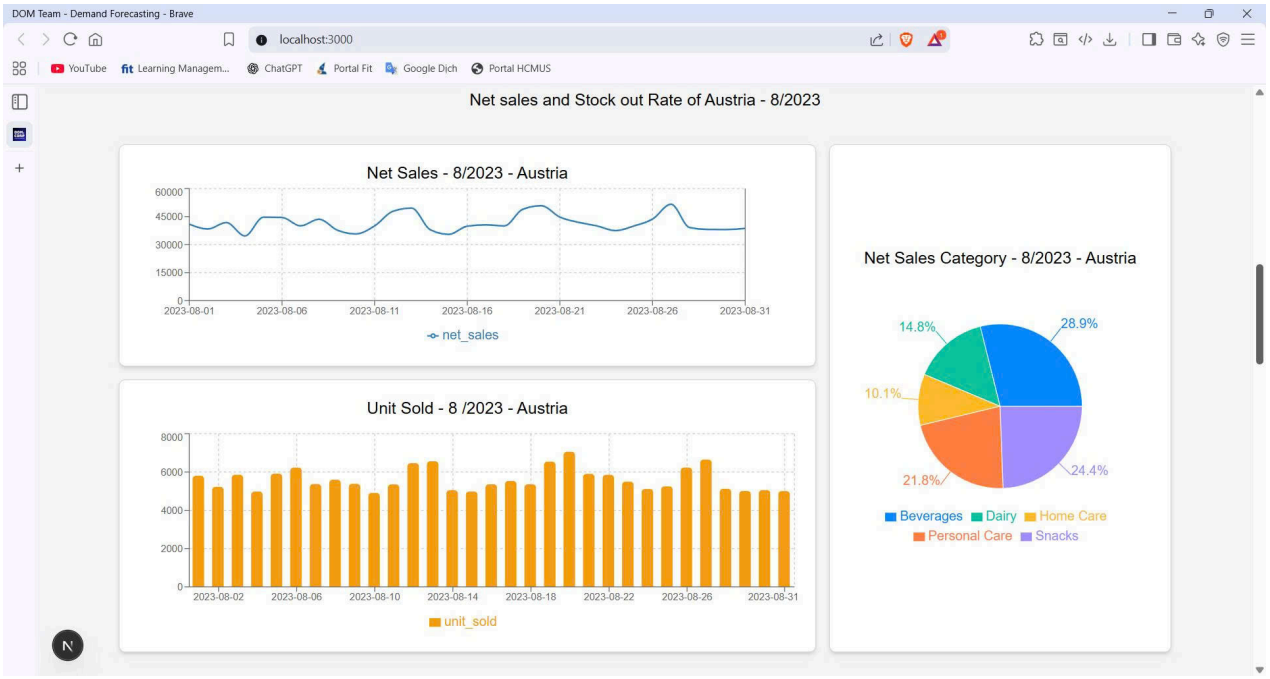


Figure 3.5.2. Sale Analytics Section 1

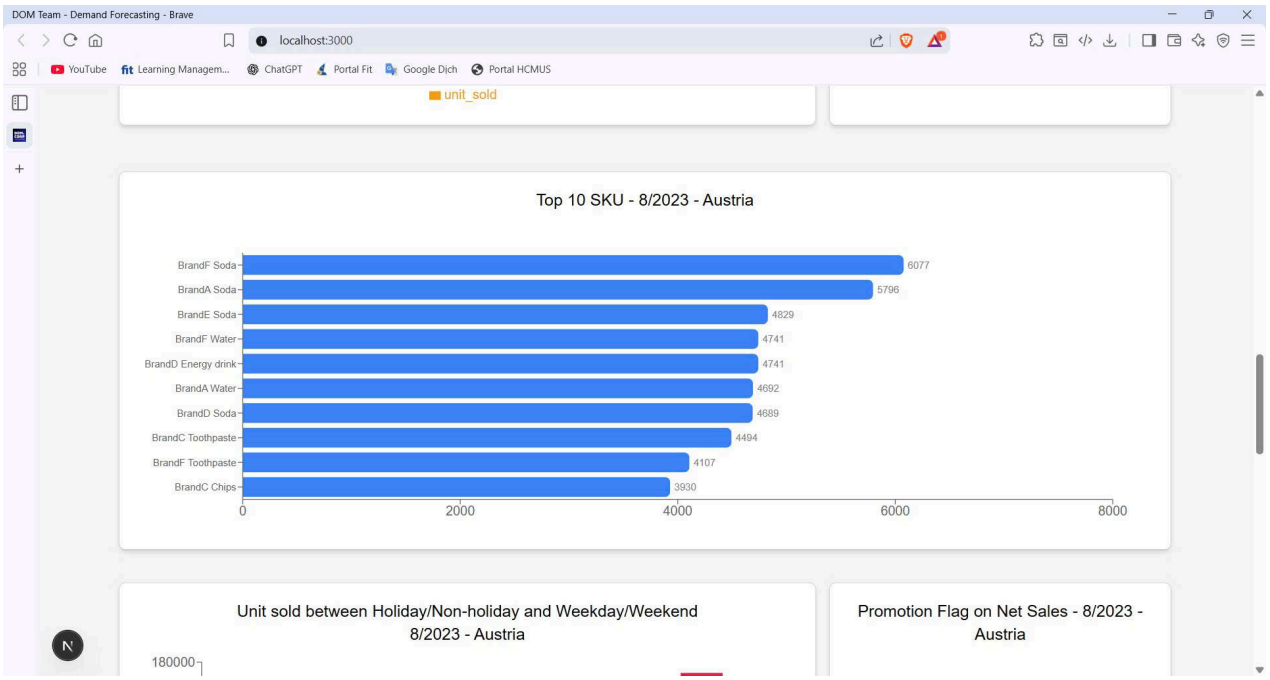


Figure 3.5.3. Sale Analytics Section 2

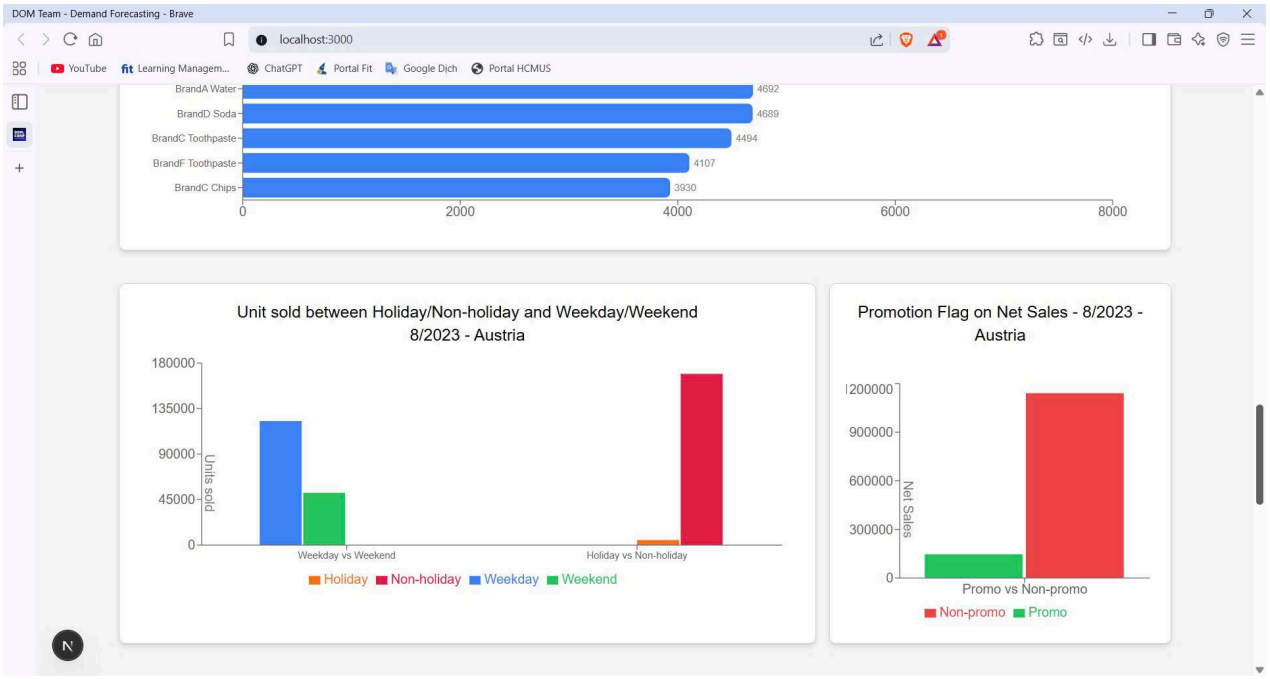


Figure 3.5.4. Sale Analytics Section 3

2. Sales Analytics Dashboard:
   - Temporal Analysis: Features Time Series Charts for daily net sales and units sold with date range selection.
   - Performance Metrics: Includes Category Breakdown (Pie/bar charts) and a Top SKU Performance horizontal bar chart for quickly identifying trends and best-sellers.
   - Demand Drivers: Offers comparative charts for Holiday vs. Weekend Analysis and Scatter plots to visualize Promotional Impact on sales volume.

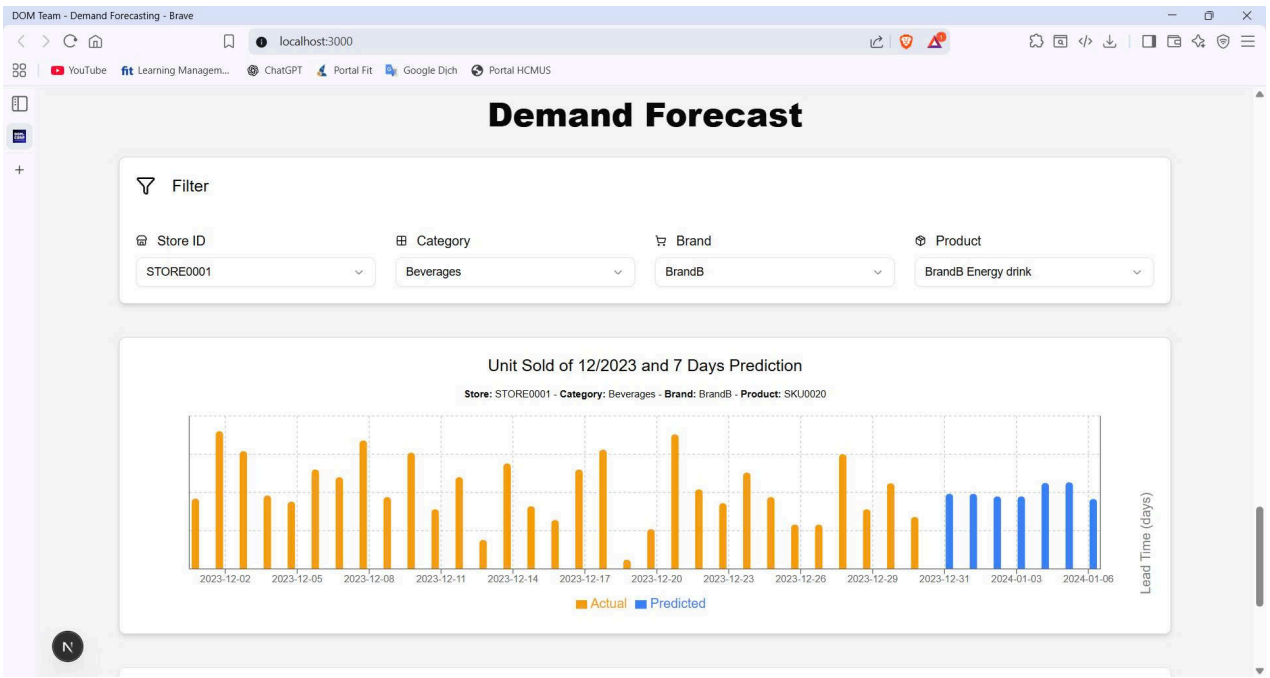


Figure 3.5.5. Advanced Filtering System and AI Prediction Interface

3. Advanced Filtering System:
    - Granular Control: A comprehensive set of filters is provided, including Country Selection, Date Range Picker, Multi-select Category Filters, and specific Store Filters.
    - Real-time Updates: The system ensures all charts and data visualizations auto-refresh dynamically as filters are changed, providing a seamless analytical experience.
4. AI Prediction Interface:
    - Forecasting Tools: Dedicated forms are available for users to input specific criteria (store/SKU/date) to run both Sales Forecasting and Lead Time Prediction.
    - Prediction Management: Results are displayed with Confidence Intervals, with options for Batch Predictions (via CSV upload) and a Prediction History log to view past forecasts versus actual results.

B. Data & System Management Features

1. Administrative & Advanced Analytics:
    - Administrative Features: The system includes tools for Data Management (uploading new sales data), Model Training (triggering AI retraining from the UI), System Health monitoring, and basic User Management (role-based access).
    - Advanced Analytics: Built-in analysis includes Trend Analysis (moving averages), Correlation Analysis (relationships between variables like weather and sales), Anomaly Detection, and Forecast Accuracy tracking to monitor the model's performance over time.
2. UX, Navigation, and Integration:
    - User Experience: The design is Responsive (mobile-friendly), includes Dark/Light Mode, manages Loading States, and provides robust Error Handling and basic Offline Support.
    - Navigation & Layout: Features such as Sidebar Navigation, Breadcrumb Navigation, Search Functionality, and Favorites/Bookmarks ensure an efficient and intuitive user workflow.
    - Integration Features: The platform offers API Documentation for available endpoints, Webhook Support for real-time updates, and potential Third-party Integrations (with ERP/CRM systems) to ensure seamless data flow within the enterprise ecosystem.

This architecture, leveraging a computationally powerful and commercially proven tech stack, ensures that the DataStorm platform is a robust, production-ready system capable of solving the FMCG industry's most challenging forecasting problem.