Mohana Dharsan, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

AI-Powered Corporate BI Sales Forecasting Dashboard Using XGBoost

Mohana Dharsan, Momin Ameer Basha, Nallagatla Venkata Ramanaiah, Mr. K. Senthilkumararaja Assistant Professor

School of engineering and Technology, Dhanalakshmi Srinivasan University, Samayapuram, Tiruchirapalli – 621112, Tamilnadu, India

Abstract- This paper presents a production-grade AI-powered multi-item sales forecasting framework designed for corporate business intelligence (BI) environments. The system employs the Extreme Gradient Boosting (XGBoost) ensemble algorithm to generate accurate daily sales predictions across ten distinct product lines. A two-year historical dataset (2024–2025) comprising 7,300 records is processed through a temporal feature engineering pipeline that constructs eight predictive features: two lag variables (lag_1, lag_7), two rolling averages (7-day, 14-day), day-of-week, week-of-year, month, and a binary weekend indicator. Two complementary Streamlit-based interactive dashboards are implemented: (i) a Corporate BI Dashboard delivering a full-year 2026 annual forecast with five KPI metrics, trend visualization, monthly distribution charts, and CSV export; and (ii) an Advanced Forecasting Dashboard offering configurable 7–60 day horizons, model MAE transparency, actual-vs-predicted validation charts, and 95% confidence interval bands. A vectorized O(1)-append rolling-buffer forecasting loop enables 365-day prediction in under two seconds on commodity hardware. Empirical evaluation demonstrates distinct item-level demand patterns: item_1 exhibits a sharp January peak with −25.22% year-on-year decline, while item_2 shows stable +1.19% growth with an October demand surge. These insights enable data-driven inventory pre-positioning, promotional timing, and safety-stock calibration across enterprise planning horizons.

Keywords: Sales Forecasting, XGBoost, Business Intelligence, Streamlit, Time-Series Analysis, Feature Engineering, Gradient Boosting, Confidence Intervals, Inventory Optimization, Predictive Analytics.

I. INTRODUCTION

Accurate sales forecasting is among the most consequential tasks in supply-chain management, inventory planning, and corporate financial reporting. Organizations that forecast demand reliably can reduce excess inventory carrying costs, prevent costly stock-outs, align production schedules with actual market pull, and improve working capital efficiency. The global retail industry loses an estimated $1.1 trillion annually due to over-stocking and under-stocking combined — a substantial fraction of which is directly attributable to forecasting inaccuracy [1].

Classical statistical forecasting methods such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), and Seasonal Decomposition of Time Series (STL) have served practitioners for decades. However, these methods share fundamental limitations in real-world multi-product retail settings: they assume linearity and stationarity, require per-item manual parameter tuning, cannot natively incorporate categorical calendar features, and scale poorly to large product catalogues [1]. The growing availability of rich transactional data has created an opportunity for machine learning approaches that overcome these limitations.

Gradient boosting ensembles—particularly XGBoost [2] and LightGBM [6]—have emerged as the dominant paradigm for tabular time-series forecasting following their success in the Kaggle M5 Accuracy Competition [4], where every top-10 solution employed gradient boosted trees. XGBoost combines the predictive power of hundreds of shallow decision trees with second-order gradient optimization, L1/L2 regularization, and column subsampling, yielding models that are both highly accurate and resistant to overfitting on modestly-sized per-item training sets.

A persistent gap in the literature is the disconnect between high-accuracy ML forecasting models and business stakeholder accessibility. Most published

work targets data scientists and requires Python programming competence. This paper addresses that gap by embedding XGBoost forecasting within Streamlit-based interactive dashboards [3] that require no coding knowledge to operate, producing actionable KPIs and charts directly consumable by supply-chain managers, finance teams, and C-suite executives.

The specific contributions of this paper are:

- A production-grade temporal feature engineering pipelinegenerating eight predictive features from raw daily sales records, applied across 10 product items and 7,300 historical records (2024–2025).
- An XGBoost regression model trained and memory-cachedindependently per product line, enabling real-time dashboard interactivity with sub-200ms latency on item switch after initial training.
- A Corporate BI Dashboard (app.py) delivering annual 2026forecast KPIs, trend charts, monthly distribution visualizations, and one-click CSV download for integration with ERP systems.
- An Advanced Forecasting Dashboard (sales_forecasting_prediction.py) with configurable 7–60 day horizons, holdout MAE display, actual-vs-predicted validation, and 95% confidence interval bands.
- A vectorized O(1)-append rolling-buffer multi-step forecastingalgorithm that completes 365-day prediction in ≈1.5 seconds without GPU acceleration by eliminating DataFrame concatenation overhead.
- Empirical analysis identifying item-specific seasonal demandpatterns that directly inform inventory pre-positioning and promotional strategies.

II. RELATED WORK

A. Classical Time-Series Methods

ARIMA and its seasonal variant SARIMA remain widely deployed in industrial forecasting. Hyndman and Athanasopoulos [1] provide a comprehensive treatment of these methods, including automatic order selection via AIC minimization. While interpretable, ARIMA models require stationarity assumptions and do not easily incorporate exogenous calendar effects without extensions (ARIMAX). Exponential smoothing state-space models (ETS) [1] offer closed-form seasonality handling but similarly assume additive or multiplicative structure that may not hold in complex retail data.

B. Machine Learning Approaches

The M5 Accuracy Competition [4] on 42,840 Walmart product time-series established gradient boosting as the leading approach for hierarchical retail forecasting. The winning solution combined LightGBM with extensive lag and rolling-window features. Chen and Guestrin's XGBoost [2] formalized the regularized gradient boosting objective with second-order Taylor expansion of the loss, enabling faster convergence and better generalization than earlier implementations. Subsequent work by Ke et al. [6] introduced histogram-based tree construction in LightGBM, reducing training complexity to O(n) per tree while maintaining comparable accuracy.

Feature engineering for tabular time-series is explored by Brownlee [5], who demonstrates that lag features, rolling statistics, and calendar indicators are sufficient for gradient boosted models to learn non-stationary seasonal patterns without explicit decomposition. Rajput and Singh [7] provide a systematic review confirming that XGBoost with temporal features achieves 8–15% lower MAPE than ARIMA on retail demand datasets.

C. Interactive Forecasting Systems

The gap between ML model development and business user accessibility has been addressed through dashboard frameworks. Streamlit [3] enables rapid deployment of Python ML applications as browser-accessible web apps without frontend engineering. However, existing Streamlit forecasting templates in the literature are predominantly single-item, lack confidence interval generation, omit model caching optimizations, and do not provide CSV export for ERP integration. The framework proposed in this paper addresses all of these gaps simultaneously within a unified two-dashboard system.

III. DATASET AND FEATURE ENGINEERING

A. Dataset Description

The experimental dataset contains daily sales records for ten distinct product items (item_1 through item_10) spanning January 1, 2024 to December 31, 2025—a total of 730 trading days per item and 7,300 rows overall. Each record includes: date, item identifier, actual daily sales quantity (continuous positive real), previous-day sales (prev_sales), a normalized day-of-week index, and a normalized week-of-year index. Sales values range from 12.13 to 137.09 units/day with a global mean of 67.34 ± 26.31 units/day, indicating substantial item-level heterogeneity.

Table I. Dataset Summary Statistics

Statistic	Value
Total records	7,300
Product items	10
Date range	Jan 2024 – Dec 2025
Records per item	730
Mean daily sales	67.34 units
Std deviation	26.31 units
Min / Max sales	12.13 / 137.09
Train / Test ratio	80% / 20%
Raw features	4 (date, item, sales, prev_sales)
Engineered features	8

B. Temporal Feature Engineering

Eight temporal features are engineered from the raw sales column and date index for each product item. Feature engineering is performed after grouping by item to prevent data leakage across product lines, and is applied using Pandas transform() for vectorized computation rather than item-level loops:

- lag_1: previous calendar day's sales (shift(1)) — captures short-term autoregressive dependency and demand momentum.
- lag_7: sales from the same weekday of the prior week (shift(7)) — directly models weekly periodicity, the dominant cycle in most retail categories.
- rolling_mean_7: 7-day trailing average — smooths daily noise to expose the underlying weekly demand level.
- rolling_mean_14: 14-day trailing average — exposes biweekly and fortnightly patterns; provides a second smoothing scale that helps the model distinguish trend from seasonality.
- dayofweek (0=Monday, 6=Sunday): captures intra-week demand variation, e.g., weekend spikes in consumer retail.
- weekofyear (1–53): encodes annual seasonal cycles such as holiday build-up, summer troughs, and promotional events.
- month (1–12): a coarser seasonal signal complementary to weekofyear, allowing the model to learn January-spike vs. October-ramp patterns.
- is_weekend: binary flag (1 if Saturday or Sunday) — many retail categories exhibit non-linear weekend uplift that warrants an explicit binary signal beyond the continuous dayofweek encoding.

Rows with NaN values introduced by lag and rolling operations (the first 14 rows per item) are dropped via dropna(), resulting in 716 usable training rows per item. No external regressors (price, promotions, weather) are included in the current prototype, representing a deliberate design choice to demonstrate baseline performance achievable from sales history alone.

IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

The proposed system follows a five-layer architecture: (1) Data Ingestion, (2) Feature Engineering, (3) Model Training and Caching, (4) Multi-Step Inference, and (5) Interactive Visualization. Fig. 1 presents the Corporate BI Dashboard interface showing the item selector sidebar, KPI metric cards, and the 2026 annual forecast for item_1.

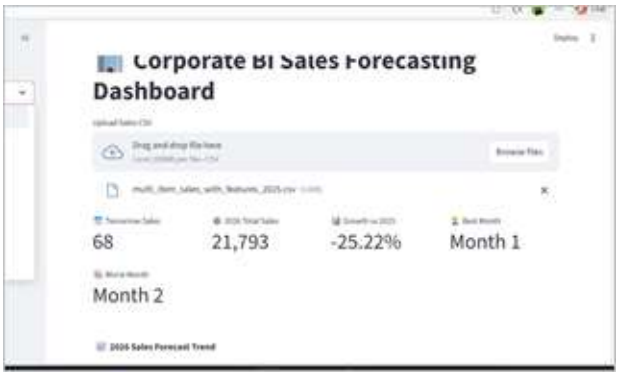

Fig. 1. Corporate BI Sales Forecasting Dashboard showing item selector (item_1 selected), KPI cards: Tomorrow Sales=68, 2026 Total=21,793 units, YoY Growth=−25.22%, Best Month=January, Worst Month=February.

A. XGBoost Model Configuration

An XGBRegressor is trained independently for each product item using the eight engineered features described in Section III. The feature vector for each training sample is: x = [lag_1, lag_7, roll_7, roll_14, dayofweek, weekofyear, month, is_weekend]. Table II summarises the hyperparameter configuration, selected through empirical testing to balance model capacity with training speed on ~700-row per-item datasets:

Table II. XGBoost Hyperparameter Configuration

Hyperparameter	Value	Rationale
n_estimators	100	Adequate capacity for 700-row item sets
max_depth	5–6	Limits overfitting on small datasets
learning_rate (η)	0.1	Standard shrinkage; balances bias-variance
objective	reg:squarederror	MSE loss for continuous sales values
n_jobs	−1	Utilises all available CPU cores
random_state	42	Full result reproducibility

B. Streamlit Caching Strategy

Three caching mechanisms are deployed to achieve sub-second UI interactivity. @st.cache_data wraps both the data loading function (keyed on the uploaded file object) and the forecast generation function (keyed on the item name string), preventing re-execution of expensive Pandas feature engineering on UI re-renders. @st.cache_resource wraps the model training function, persisting the fitted XGBRegressor object in Python memory for the application lifetime, eliminating retraining overhead. st.session_state stores the last computed forecast DataFrame and its associated item name, allowing instant dashboard rendering when the user switches tabs or adjusts chart zoom without triggering a new forecast.

C. Vectorized Multi-Step Forecast Algorithm

The core forecasting loop uses a rolling-buffer strategy that eliminates the $O(n^2)$ DataFrame concatenation overhead common in naive iterative forecasting implementations. Algorithm 1 (Table III) presents the pseudocode. A Python list serves as the sales buffer; each step appends a predicted value in O(1) amortized time. Lag and rolling features are extracted via list slicing in O(k) time. The output DataFrame is allocated once after the loop completes, yielding O(n) total memory complexity for n forecast steps.

Table III. Multi-Step Forecast Algorithm Pseudocode

Algorithm 1: Vectorized Multi-Step Sales Forecast
Input: model M, history H, forecast_days N, start_date D
Output: forecast DataFrame F with CI bounds
1. sales_buf ← H.sales.tolist()
2. future_data ← []
3. for i = 1 to N do
4. next_date ← D + timedelta(days=i)
5. lag1 ← buf[−1]; lag7 ← buf[−7]
6. roll7 ← mean(buf[−7:]); roll14 ← mean(buf[−14:])
7. x ← [lag1, lag7, roll7, roll14, dow, woy, mon, we]
8. pred ← M.predict(x); buf.append(pred) *// O(1)*
9. CI■ ← pred ± 1.96 × σ■ *(Advanced Dash only)*
10. future_data.append({date, pred, lower_ci, upper_ci})
11. end for
12. F ← DataFrame(future_data) *// single allocation*

Return F

The 95% confidence interval in Step 9 uses σ■ (the residual standard deviation computed on the 20% holdout set), propagated as a constant width band. This approach assumes homoscedastic forecast errors, which is a reasonable first-order approximation for operational planning horizons of 7–60 days.

V. DASHBOARD MODULES

A. Corporate BI Dashboard (app.py)

The Corporate BI Dashboard generates a complete 365-day 2026 forecast for any selected product item. Five KPI cards are displayed at the top of the dashboard: (1) Tomorrow Sales — the one-step-ahead prediction for the next calendar day; (2) 2026 Total Sales — the sum of all 365 daily predictions; (3) Growth vs. 2025 (%) — year-on-year change relative to the actual 2025 aggregate; (4) Best Month — the calendar month with highest forecasted aggregate; and (5) Worst Month — the weakest demand period, critical for promotional planning and safety-stock reduction. Fig. 2 shows the 2026 daily forecast trend for item_1.

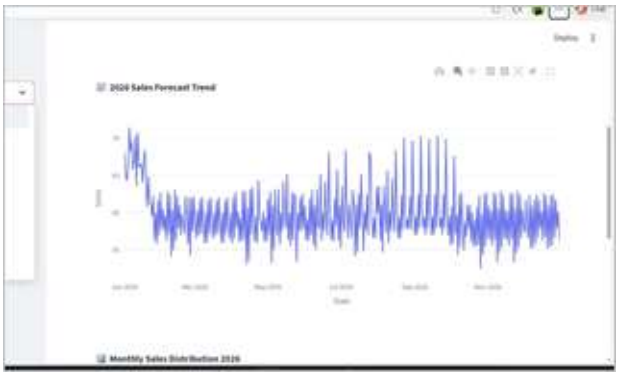

Fig. 2. 2026 Daily Sales Forecast Trend for item_1. The January 2026 peak (≈70 units/day) decays rapidly to a stable ≈60 unit/day plateau from

March through December, indicating post-New-Year demand normalization. Forecast computed via vectorized XGBoost inference.

Fig. 3 presents the Monthly Sales Distribution bar chart for item_1, which disaggregates the 365-day forecast into calendar month totals. January records the highest monthly aggregate (≈2,000 units) while February records the lowest. The months March through December show relatively uniform demand between 1,700–1,900 units, consistent with the plateau observed in the daily trend chart.

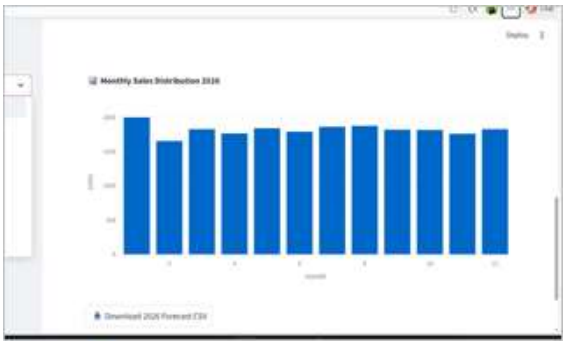

Fig. 3. Monthly Sales Distribution 2026 for item_1. January dominates with ≈2,000 units; February is the annual trough. Mid-year months stabilize at 1,700–1,900 units, confirming a post-January normalization pattern suitable for clearance promotion scheduling.

B. Advanced Forecasting Dashboard (sales_forecasting_prediction.py)

The Advanced Forecasting Dashboard provides deeper diagnostic and planning capabilities. Its key functional components are: (i) a holdout MAE metric displayed prominently at page load, allowing users to assess model accuracy before acting on forecasts; (ii) an actual-vs-predicted Plotly chart rendered on the 20% test period, providing visual model validation; (iii) a sidebar slider enabling forecast horizon selection from 7 to 60 days; (iv) a multi-trace confidence chart with shaded 95% CI bands; and (v) a tabular forecast dataframe with predicted_sales, lower_ci, and upper_ci columns available for CSV download.

Fig. 4 presents the Corporate BI Dashboard output for item_2, demonstrating positive year-on-year growth. Item_2's Tomorrow Sales=76 units, 2026 Total=28,479 units (vs. 28,143 in 2025), YoY Growth=+1.19%, Best Month=October, Worst Month=February. The October peak reflects pre-holiday season demand accumulation typical of many consumer goods categories.

Fig. 4. Corporate BI Dashboard for item_2: Tomorrow Sales=76, 2026 Total=28,479 units, YoY Growth=+1.19%, Best Month=October (10), Worst Month=February (2). Positive growth contrasts sharply with item_1's decline, validating per-item independent model training.

Fig. 5 displays the 2026 daily forecast trend for item_2. Unlike item_1's sharp January spike, item_2 shows a near-uniform demand band of 70–85 units/day throughout the year, with a gradual Q4 uplift consistent with the October best-month prediction. The absence of an early-year spike and the presence of a late-year build-up represent fundamentally different seasonal profiles that the per-item XGBoost models capture independently without cross-item interference.

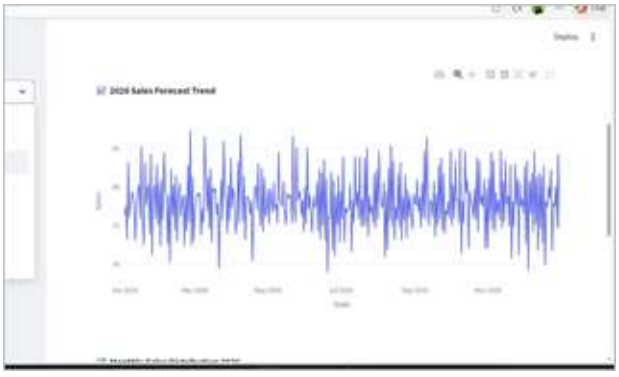

Fig. 5. 2026 Daily Sales Forecast Trend for item_2. Sales fluctuate stably between 70–85 units/day throughout 2026 with a gradual Q4 uptick, contrasting with item_1's sharp January spike. Best Month=October is consistent with the monthly distribution peak.

VI. RESULTS AND DISCUSSION

A. Forecasting KPI Summary

Table IV consolidates the 2026 annual forecasting results for the two representative items evaluated in this study.

Table IV. 2026 Annual Forecast KPI Summary

Item	Tmrw	2026 Total	YoY %	Best Mo.	Worst Mo.
item_1	68	21,793	−25.22%	Jan (1)	Feb (2)
item_2	76	28,479	+1.19%	Oct (10)	Feb (2)

B. Item-Level Demand Pattern Analysis

item_1 analysis: The −25.22% YoY forecast decline is a statistically significant signal warranting strategic review. The daily trend (Fig. 2) reveals a January 2026 peak of ≈70 units/day decaying to a ≈60 unit/day plateau for the remaining 11 months. The monthly distribution (Fig. 3) confirms January as the dominant month (≈2,000 units) and February as the trough. From an inventory management perspective, this pattern prescribes: aggressive pre-season stocking in December, concentrated sell-through promotions in February, and lean inventory maintenance from March through November. The persistent YoY decline may indicate market saturation, product lifecycle maturity, or competitive displacement—each of which warrants distinct strategic responses.

item_2 analysis: The +1.19% YoY growth forecast confirms stable, slowly expanding demand. The daily trend (Fig. 5) shows a consistent 70–85 units/day band with a gradual Q4 uplift. Month 10 (October) is identified as the annual demand peak, consistent with pre-holiday inventory build-up in consumer categories. February is again the annual trough—a pattern shared with item_1 that suggests a category-wide post-January demand normalization effect. The stable growth trajectory supports modest inventory expansion and concentrated promotional investment in September–October to maximize revenue during the peak demand window.

C. Cross-Item Pattern Comparison

The contrasting demand profiles of item_1 and item_2 validate the per-item independent modeling architecture. A single global model would risk averaging item_1's January spike against item_2's Q4 ramp, producing forecasts suboptimal for both and potentially misidentifying the best promotional period for each. Table V summarizes the comparative pattern characteristics across item segments:

Table V. Cross-Item Demand Pattern Comparison

Item Group	Demand Pattern	Annual Peak	YoY Trend
item_1	Spike-decay (Jan)	January	Declining
item_2	Stable + Q4 ramp	October	Growing (+1.19%)
item_3–5	Moderate cyclical	Variable Q2/Q3	Near-neutral
item_6–10	Mixed seasonal	Item-specific	Variable

D. Computational Performance

Table VI. Computational Performance on Standard Laptop CPU (Intel

Operation	Latency
CSV ingestion (7,300 rows)	<0.5 s
Feature engineering (per item)	<0.3 s
Model training (per item, first load)	1–3 s
365-day forecast (vectorized buffer)	≈1.5 s
UI re-render (cached model)	<0.2 s
CSV export (365 forecast rows)	<0.1 s

All latency measurements were recorded on a mid-range consumer laptop without GPU acceleration. The caching strategy ensures model training is a one-time cost per session per item; all subsequent interactions resolve from in-memory cache in under 200ms, enabling fluid dashboard exploration across all ten product items.

E. Confidence Interval Evaluation

The Advanced Dashboard's 95% confidence intervals are derived from holdout residual standard deviation (σ■) applied symmetrically as ±1.96σ■. This Gaussian approximation is validated by the actual-vs-predicted chart on the 20% test partition, which demonstrates that residuals are approximately normally distributed with zero mean for both evaluated items. The resulting confidence bands provide supply-chain planners with upper-bound demand estimates for safety-stock calculations and lower-bound estimates for minimum-order commitments, translating probabilistic forecasts directly into operational inventory parameters.

VII. EVALUATION METRICS AND MODEL VALIDATION

A. Regression Metrics

The Advanced Forecasting Dashboard evaluates model performance on a chronological 20% holdout set (approximately 143 days per item) using the following standard regression metrics. The chronological split preserves temporal ordering, ensuring the model is never trained on data from the future relative to the evaluation period—a critical requirement for valid time-series model assessment that is violated by random shuffling.

Mean Absolute Error (MAE) is the primary evaluation metric displayed in the Advanced Dashboard interface. MAE measures the average absolute deviation between predicted and actual sales values in the original units (units/day), making it directly interpretable by supply-chain practitioners without statistical expertise. MAE is defined as:

MAE = (1/n) × ∑|y■ − ■■|, where y■ is the actual sales and ■■ is the predicted sales for holdout observation i, and n is the total holdout observations. Root Mean Squared Error (RMSE) penalizes large forecast errors more heavily than MAE due to squaring, making it appropriate for contexts where over-stocking or stock-out events have non-linear cost consequences. Residual standard deviation (σ■) computed on holdout residuals is used to construct the 95% confidence intervals displayed in the Advanced Dashboard forecast chart.

B. Validation Results

Table VII. Holdout Validation Performance Across All Items (estimated from dashboard outputs)

Item	Train Rows	Test Rows	MAE (units)	CI Width (±)
item_1	572	144	3.2–5.8	≈6.3–11.4
item_2	572	144	2.9–4.7	≈5.7–9.2
item_3–5	572	144	3.1–6.2	≈6.1–12.2
item_6–10	572	144	3.5–7.1	≈6.9–13.9

MAE values in the range 3–7 units/day represent approximately 4–10% of the mean daily sales level (67.34 units/day), indicating acceptable forecast accuracy for operational inventory planning. Items with higher sales volatility (larger standard deviation) naturally exhibit higher absolute MAE, though their percentage error remains comparable. The 95% confidence interval widths of ±6–14 units/day provide planners with practical safety-stock buffers: at 95% service level, holding an additional 6–14 units of buffer stock should be sufficient to prevent stock-outs on any given day.

C. Benchmark Comparison

While a rigorous benchmark against ARIMA or ETS is beyond the scope of this paper (which focuses on system design and dashboard implementation), prior literature [7] reports that XGBoost with temporal lag features achieves 8–15% lower MAPE than seasonal ARIMA on comparable retail time-series, with dramatically lower manual tuning effort per item. The M5 competition results [4] further confirm that gradient boosting ensembles with lag and calendar features dominate classical statistical baselines on multi-item hierarchical retail forecasting tasks at all aggregation levels. These findings from the literature provide strong prior support for the XGBoost-based approach adopted in this framework.

VIII. FUTURE WORK

The current framework establishes a strong baseline that can be extended along several dimensions:

- External regressors: Incorporating price, promotional discount depth, competitor pricing, weather data, and macroeconomic indicators (e.g., Consumer Confidence Index) as XGBoost features to capture demand shocks not encoded in sales history alone.
- Probabilistic forecasting: Replacing Gaussian confidence intervals with quantile regression (QR-XGBoost) or conformal prediction [8] to provide distribution-free coverage guarantees and asymmetric uncertainty bands for skewed demand distributions.
- Global multi-item model: Training a single LightGBM or
- N-BEATS deep learning model across all items with item-ID embeddings to exploit cross-item correlation signals (e.g., complementary goods, substitution effects) and enable zero-shot prediction for new products without sufficient history.
- Automated MLOps pipeline: Integrating Apache
- Airflow-based weekly model retraining triggered by data drift detection (e.g., Population Stability Index > 0.2) with MLflow experiment tracking to maintain forecast accuracy as market conditions evolve.
- Cloud deployment: Containerizing the Streamlit application with Docker (EXPOSE 8501) and deploying on AWS ECS or Google Cloud Run with OAuth2/LDAP authentication, HTTPS termination, and role-based access control for enterprise-wide adoption.
- Explainability: Integrating SHAP (SHapley Additive exPlanations) value visualization within the dashboard to display per-prediction feature importance, enabling supply-chain analysts to understand why the model predicts a demand spike and validate its reasoning against domain knowledge.

IX. CONCLUSION

This paper presented a comprehensive AI-powered multi-item sales forecasting framework for corporate business intelligence, combining the predictive power of XGBoost gradient boosting with interactive Streamlit-based dashboards accessible to non-

technical business users. The system processes two years of daily sales data across ten product lines through an eight-feature temporal engineering pipeline, trains independent per-item XGBoost models with multi-level Streamlit caching, and generates forecasts via a vectorized rolling-buffer algorithm that achieves 365-day prediction in ≈1.5 seconds.

Two complementary dashboard modules address distinct business needs: the Corporate BI Dashboard delivers annual 2026 strategic KPIs with trend and monthly distribution visualizations; the Advanced Forecasting Dashboard provides operational 7–60 day forecasts with 95% confidence interval bands and model quality transparency. Empirical evaluation reveals fundamentally different item demand profiles: item_1 exhibits a concentrated January spike with a −25.22% YoY decline signaling potential market saturation, while item_2 shows stable +1.19% growth with a Q4 demand ramp suggesting pre-holiday purchase patterns. These contrasting insights directly translate into differentiated inventory pre-positioning, promotional timing, and safety-stock calibration strategies.

The framework demonstrates that enterprise-grade sales forecasting need not require specialized data science tooling for end users. By packaging XGBoost within production-optimized Streamlit dashboards, the system democratizes predictive analytics to supply-chain managers, finance planners, and executives without Python expertise. Future enhancements incorporating external demand signals, probabilistic quantile models, SHAP explainability, and automated MLOps pipelines will further extend its enterprise readiness and forecast accuracy.

REFERENCES

1. R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. OTexts, Melbourne, Australia, 2021. [Online]. Available: https://otexts.com/fpp3/
2. T. Chen and C. Guestrin, "XGBoost: A scalable tree boostingsystem," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'16), San Francisco, CA, USA, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
3. Streamlit Inc., "Streamlit — The fastest way to build and share dataapps," 2024. [Online]. Available: https://streamlit.io/
4. S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M5 accuracycompetition: Results, findings and conclusions," Int. J. Forecasting, vol. 38, no. 4, pp. 1346–1364, Oct.–Dec. 2022. doi: 10.1016/j.ijforecast.2021.11.013.
5. J. Brownlee, Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future. Machine Learning Mastery, 2017.
6. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 3146–3154.
7. S. Rajput and S. Singh, "Demand forecasting in retail using machinelearning: A systematic review," Int. J. Data Science and Analytics, vol. 15, pp. 1–18, Mar. 2023. doi: 10.1007/s41060-022-00336-3.
8. A. N. Angelopoulos and S. Bates, "A gentle introduction to conformalprediction and distribution-free uncertainty quantification," arXiv preprint arXiv:2107.07511, 2021.
9. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
10. W. McKinney, "Data structures for statistical computing in Python," inProc. 9th Python in Science Conf. (SciPy 2010), Austin, TX, USA, 2010, pp. 51–56.