IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AI-DRIVEN INTELLIGENT DEMAND FORECASTING AND INVENTORY OPTIMISATION FRAMEWORK FOR RETAIL POS SYSTEMS

[1]Chetan Kumar M, [2]Sridevi Sridhar

[1]M.Tech Student, [2]Associate Professor

Department of Computer Science and Engineering

SRM Institute of Science and Technology, Vadapalani, Chennai, India

Abstract: Managing inventory in a retail business sounds straightforward until you try to do it well. Between shifting consumer tastes, seasonal spikes, unpredictable supply delays, and the sheer number of SKUs a typical store handles, even experienced buyers routinely end up either sitting on too much stock or watching their shelves go empty at the worst possible moment. Industry estimates consistently put the combined cost of overstock and stock-outs at somewhere between one and three percent of annual sales.

This paper proposes a framework that attempts to address these problems by embedding machine learning capabilities directly into a retail Point-of-Sale system rather than treating forecasting as a separate, standalone tool. The central argument is that predictive intelligence needs to sit inside the operational system where purchase and stock decisions are made, not in a separate analytics dashboard that someone might consult once a week if they remember.

The proposed architecture moves transaction data from the POS layer through a preprocessing and feature engineering pipeline, then into an ensemble of ML models: XGBoost for demand forecasting with temporal and promotional feature engineering, LSTM networks for capturing longer-term sequential sales patterns, a Random Forest regressor for computing optimal reorder quantities, and K-Means clustering for identifying slow-moving and dead stock. Model outputs are surfaced back into the POS interface as actionable reorder recommendations and inventory health indicators. Evaluation is framed around MAE, RMSE, and R^2 metrics, with particular attention to performance during seasonal demand peaks where static threshold-based systems fail most visibly. The framework is designed with SME retailers in mind.

Index Terms - Demand Forecasting, Inventory Optimisation, POS Systems, XGBoost, LSTM, Random Forest, K-Means Clustering, Dynamic Reorder, SME Retail, Predictive Analytics.

I. INTRODUCTION

The retail Point-of-Sale system has come a long way from its origins as a cash register with a printer attached. Modern POS platforms handle payments across multiple modes, manage loyalty programmes, track real-time stock levels, generate sales reports, and in some cases integrate directly with supplier ordering portals. Yet for all this sophistication, the actual decision of how much stock to hold — and when to reorder — is still handled with surprisingly primitive tools in most retail environments. Fixed reorder points set once and rarely revisited. Manual eyeballing of shelf levels by store managers. Spreadsheet-based tracking that quickly gets out of sync with what is actually happening on the shop floor.

The consequences are predictable and well-documented. A retailer who sets a reorder point based on average monthly sales will systematically understock during peak demand periods and overstock during slower ones. Every unit sold during a stock-out is a lost sale. Every unit sitting past its optimal holding window is tying up capital, incurring storage costs, and potentially depreciating in value. Across the retail sector, the IHL Group estimates that overstock and out-of-stocks together cost retailers between one and three percent of annual revenue [1].

There is no shortage of demand forecasting software in the market, but the existing landscape has a structural problem: the AI lives somewhere else. Enterprise retailers might run a separate planning tool that their buying team uses to generate forecasts, which then get manually entered into the inventory system. Smaller businesses typically cannot afford the integration complexity of such setups even if they could justify the licensing cost. The result is a gap between where the intelligence lives and where the decisions get made.

This paper proposes a framework that tries to close that gap. Rather than building another standalone forecasting tool, the design goal is to embed ML-driven demand forecasting, dynamic reorder optimisation, and inventory health monitoring directly within the POS architecture. A store manager using this system should be able to make a restocking decision based on a model-generated recommendation without ever leaving the interface they already use to process sales.

II. BACKGROUND AND MOTIVATION

A. How POS Systems Have Changed

The first electronic POS systems were installed in the early 1970s and did essentially one thing: record transactions and print receipts more reliably than a mechanical till. The shift to cloud-connected POS platforms over the past decade changed the picture considerably. Data that previously sat in a local terminal and was occasionally exported to a spreadsheet now flows into centralised databases in real time. Every scan, every return, every voided transaction is recorded, timestamped, and attributable to a specific cashier, shift, and store location.

This data richness creates an obvious opportunity. Sales history, combined with product attributes, pricing, promotional calendars, and supplier lead times, contains most of the signal needed to build a decent demand forecast. The technical barriers to doing this — data storage costs, compute requirements, model training complexity — have all fallen significantly. What has not changed is the integration gap: the forecasting capability and the operational system are still largely separate.

B. Why Static Reorder Systems Keep Failing

A static reorder point is calculated from a simple formula: average daily sales multiplied by lead time, plus a safety stock buffer. The problem is that every variable in that formula changes over time and the formula does not. Average daily sales in November look nothing like April. Lead times fluctuate with supplier capacity. Retailers who set these parameters once during system setup and then leave them are essentially running their inventory off a snapshot of conditions that may no longer exist.

This is not a new observation — the inventory management literature has been making this point for decades [2]. What is relatively new is the practical availability of ML tools that can continuously update demand estimates from real transaction data, adjusting for seasonality, trend, and promotional effects without requiring a human planner to manually revise parameters.

C. The SME Access Problem

Large retailers have had access to sophisticated demand planning software for years. SAP's inventory management modules, Blue Yonder's supply chain platform, and Oracle's retail planning suite all offer AI-enhanced forecasting, but they are priced and architected for enterprise deployments. An independent retailer, a regional jewellery chain, a mid-sized clothing store — these businesses generate enough transaction data to support ML-based forecasting, but the enterprise tools are not a realistic option for them. This is the gap the proposed framework is specifically designed to address.

III. PROBLEM DEFINITION

Four distinct but related inventory failure modes motivate this work. They are worth describing separately because they have different causes, different costs, and require somewhat different modelling approaches.

A. Stock-Out Problem

A stock-out is the most visible inventory failure: a customer wants something and it simply is not there. Beyond the immediate lost sale, the downstream consequences matter. A customer who encounters a stock-out — particularly a repeat customer who specifically came in for a known product — leaves with a negative

experience that affects their likelihood of returning. Stock-outs are largely a forecasting failure. They happen because the system did not anticipate demand accurately enough to maintain sufficient stock through the demand period.

B. Overstocking Problem

Overstocking is the less visible but equally damaging inverse problem. Excess inventory costs money in three ways: it occupies storage space that could be used more productively, it represents capital that could have been deployed elsewhere in the business, and in categories with any element of perishability or fashion risk it depreciates in value over time. The challenge with overstocking is that it feels safe at the time the procurement decision is made, and static reorder systems tend to build in generous safety stock buffers that accumulate into structural overstocking over time.

C. Demand Uncertainty

Retail demand is not a stable, predictable quantity. It is shaped by seasonality, promotional events, external factors, and in some categories by commodity price movements. A model that works well under normal conditions may produce poor forecasts during the unusual spikes and troughs that cause the most operational damage. The practical response is not simply to produce a point forecast but to quantify the uncertainty around it — producing confidence intervals around demand forecasts rather than just point estimates.

D. Capital Blockage from Slow and Dead Stock

Every retail operation accumulates items over time that have stopped selling at a meaningful rate. Unless there is a systematic mechanism to identify and flag these items, they tend to persist on shelves and in storage indefinitely, absorbing space and capital without contributing to revenue. The identification problem is tractable with clustering-based approaches: items can be segmented by their sales velocity, recency of last sale, margin contribution, and current stock level.

IV. LITERATURE SURVEY

Demand forecasting in retail has attracted considerable research attention, and the methods available today are substantially more capable than those available even five years ago.

A. Hybrid Time-Series and Deep Learning Approaches

Wang et al. [3] developed a hybrid ARIMA-LSTM model for retail demand forecasting and found that combining the linear structure-capturing ability of ARIMA with the non-linear sequence modelling of LSTM outperformed either method used independently on seasonal product categories. Their results showed improved short-term forecast accuracy, particularly on products with clear weekly and monthly seasonality patterns. Gupta and Sharma [4] specifically examined LSTM networks for long-term sales forecasting in fashion retail. Their model captured complex temporal dependencies that traditional time-series methods missed but required significant compute resources and a reasonably large historical dataset.

B. Gradient Boosting for Structured Retail Forecasting

Chen and Li [5] applied XGBoost with handcrafted feature engineering to an inventory optimisation problem and reported significant reductions in both stock-out frequency and total inventory holding costs. Their feature engineering approach — incorporating lag variables, rolling averages, promotional flags, and calendar features — forms the basis of the feature engineering pipeline in the proposed framework.

C. Reinforcement Learning and Graph-Based Approaches

Kumar and Singh [6] explored reinforcement learning for dynamic replenishment, framing the reorder decision as a sequential decision problem. The results were promising in environments with well-defined demand distributions, but the approach required extensive training data and considerable expertise to design effective reward functions. Zhao et al. [7] applied Graph Neural Networks to supply chain network optimisation, effectively capturing interdependencies between products and locations that simpler models miss.

D. Research Gaps

Across the reviewed literature, four gaps stand out. First, most forecasting systems operate as standalone tools that require manual integration with operational POS systems. Second, reorder systems remain predominantly threshold-based. Third, real-time forecasting is relatively uncommon. Fourth, explainability is rarely addressed. The proposed framework is designed specifically to address all four of these gaps.

TABLE 1: SUMMARY OF RELATED WORK

Author(s)	Year	Method	Key Finding	Limitation
Wang et al.	2022	Hybrid ARIMA-LSTM	Better short-term accuracy for seasonal products	High compute cost; scaling issues
Chen & Li	2021	XGBoost + Feature Engineering	Reduced stock-outs and holding costs	Overfitting risk; feature-dependent
Kumar & Singh	2020	Reinforcement Learning	Adaptive reorder in uncertain demand	Data-heavy; complex reward design
Zhao et al.	2023	Graph Neural Networks	Models complex supply chain interdependencies	Hard to interpret; data overhead
Gupta & Sharma	2022	LSTM (fashion retail)	Captured temporal trends effectively	High resource demand; large dataset required

V. RESEARCH GAP ANALYSIS

The literature review points to a consistent pattern: forecasting capability and operational inventory systems are developed and deployed in isolation. This decoupling creates friction that reduces the practical value of even technically strong forecasting models.

A. Absence of Integrated AI-POS Frameworks

The vast majority of retail AI forecasting solutions are designed as external analytics layers. They ingest data from the POS system, generate forecasts in a separate interface, and expect a human planner to translate those forecasts into procurement decisions in yet another system. Each handoff is an opportunity for delay, error, and abandonment. The proposed framework eliminates these handoffs by making the forecasting and recommendation logic a native component of the POS architecture.

B. Static Threshold Persistence

Despite decades of evidence that dynamic reorder systems outperform static ones, fixed reorder points remain the norm in retail inventory management software. The proposed framework replaces the static reorder threshold with a model-computed recommendation that updates continuously as new sales data arrives.

C. Batch vs. Real-Time Forecasting

Most deployed forecasting systems run on a batch schedule — typically nightly or weekly. A batch system is already substantially better than a static system, but it is not responsive to sudden demand shifts that can happen mid-day during a promotion or a social media trend moment. The proposed architecture is designed to update model inputs with each transaction, producing fresh forecasts on demand.

D. Explainability

A model that recommends ‘reorder 48 units of product X by Thursday’ is only useful if the user has some reason to trust that recommendation. The proposed framework addresses this through SHAP (SHapley Additive exPlanations) value reporting, which decomposes each forecast into the contributions of individual features. A store manager can see that this week’s elevated demand forecast is driven primarily by proximity to a major festival, a recent downward price adjustment, and strong performance over the last three weekends.

VI. PROPOSED SYSTEM ARCHITECTURE

The framework is structured as a four-stage pipeline — data ingestion, preprocessing and feature engineering, model inference and training, and output integration — with each stage designed to operate with minimal latency so that the system can support near-real-time updates.

A. Stage 1: POS Data Layer

The entry point is the POS transaction stream. Every completed sale generates a record containing at minimum: timestamp, product identifier, quantity sold, unit price, payment method, and any applied discount or promotion code. The POS layer also holds the current inventory positions (stock-on-hand per product, stock-on-order, and lead time estimates per supplier), which are used as inputs to the reorder optimisation component. Data cleaning and validation logic at the ingestion stage significantly affects model quality downstream.

B. Stage 2: Preprocessing and Feature Engineering

Raw transaction records are aggregated to daily product-level demand figures, which form the basic unit of analysis for the forecasting models. From this base, a rich feature matrix is constructed. Temporal features capture day-of-week effects, week-of-month patterns, month-of-year seasonality, and distance in days from known high-demand events. Lag features encode recent demand history — typically the previous 7, 14, 30, and 90 days of sales for each product. Rolling statistics summarise trend and volatility over the same windows. Product-level features include category, subcategory, price tier, margin percentage, and supplier lead time.

C. Stage 3: ML Model Ensemble

The modelling layer consists of four components, each targeting a distinct aspect of the inventory management problem.

C.1 XGBoost Demand Forecasting Module

XGBoost is used as the primary demand forecasting model. It takes the engineered feature matrix as input and produces point forecasts for daily demand per product over a configurable horizon (default: 14 days). XGBoost was selected because it handles the tabular, feature-rich input structure naturally, is robust to missing values and feature collinearity, and produces competitive accuracy on retail forecasting benchmarks [8]. Quantile regression variants of XGBoost are used to produce upper and lower confidence bounds alongside the point forecast. SHAP values are computed for each forecast and the top three contributing features are displayed alongside the recommendation in the POS interface.

C.2 LSTM Sequential Forecasting Module

For products with longer demand histories and clear sequential patterns, an LSTM network provides a complementary forecast. The LSTM processes the historical sales sequence directly, without explicit feature engineering, learning temporal structure from the data itself. The LSTM model uses a two-layer architecture with dropout regularisation and is retrained weekly on the accumulated transaction history. Its output is combined with the XGBoost forecast through a weighted ensemble, where the weights are adjusted based on each model's recent validation performance on held-out data.

C.3 Random Forest Reorder Optimisation Module

The Random Forest reorder module takes as inputs the demand forecast for the order horizon, current stock-on-hand, current stock-on-order, expected lead time from the supplier, and the product's holding cost rate. It outputs a recommended order quantity and a recommended order trigger date. Random Forest is appropriate here because the relationship between these inputs and the optimal order quantity is non-linear and involves interactions.

C.4 K-Means Inventory Segmentation Module

The clustering component characterises the current state of the inventory. Each product SKU is represented by a feature vector comprising average monthly sales volume, days elapsed since the most recent sale, gross margin percentage, and current stock quantity as months of cover. K-Means clustering with k=3 partitions this space into three segments: Fast Moving, Slow Moving, and Dead Stock. Products in the Dead Stock cluster for more than two consecutive months are escalated to an actionable clearance recommendation in the POS interface.

TABLE 2: ML COMPONENT SUMMARY

Component	Algorithm	Primary Input Features	Output
Demand Forecasting	XGBoost (quantile regression)	Lag features, temporal indicators, promotion flags, price history	14-day demand forecast + confidence interval + SHAP attribution
Sequential Pattern Capture	LSTM (2-layer, with dropout)	Historical daily sales sequence per product (18+ months)	Complementary forecast; blended with XGBoost via adaptive weighting
Reorder Optimisation	Random Forest Regressor	Demand forecast, current stock, lead time, holding cost rate	Recommended order quantity + order trigger date

Inventory Segmentation	K-Means Clustering (k=3)	Sales velocity, days since last sale, margin %, months-of-cover	Fast / Slow / Dead stock classification per SKU

VII. SYSTEM INTEGRATION AND DEPLOYMENT

A. API Layer

The ML components are wrapped in a REST API layer that the POS frontend calls to retrieve forecasts and recommendations. This decoupling means the ML models can be updated or replaced without requiring changes to the POS application itself. Endpoints are provided for: current demand forecast by product, reorder recommendation by product, inventory segment classification, and bulk recommendation refresh.

B. POS Interface Integration

The POS interface presents ML outputs in three places. On the product detail screen, the current demand forecast, confidence range, and reorder recommendation are displayed alongside standard stock information. On the stock management screen, a colour-coded inventory health indicator shows each product's velocity segment. On the purchasing screen, products with active reorder recommendations are surfaced as an ordered list with recommended quantities pre-populated. SHAP feature attribution is accessible via an expandable detail panel.

C. Infrastructure Requirements

A design constraint throughout was that the framework should be deployable on infrastructure realistic for an SME retailer. The ML microservice is designed to run on a single mid-specification server without requiring GPU compute. XGBoost and Random Forest inference are CPU-bound and fast enough for real-time API responses. Model retraining (nightly for XGBoost and Random Forest, weekly for LSTM) runs as a scheduled background job during off-peak hours.

VIII. EVALUATION METHODOLOGY

A. Metrics

Model performance is evaluated against three primary metrics. Mean Absolute Error (MAE) measures the average magnitude of forecast error in units of the predicted quantity. Root Mean Squared Error (RMSE) penalises large errors more heavily than MAE, making it sensitive to the high-error cases that cause the most operational damage. The coefficient of determination (R^2) measures what proportion of the variance in actual demand the model explains. For the reorder optimisation component, additional operational metrics are tracked: stock-out rate, overstock rate, and inventory turnover ratio.

B. Experimental Design

The primary evaluation uses a time-series train/test split rather than a random split, to respect the temporal structure of the data and avoid lookahead bias. Multiple cutoff dates are used to produce evaluation results across different periods of the year, including both peak and off-peak demand windows. Baseline comparisons are made against a naive seasonal model and an ARIMA model fitted per product.

TABLE 3: PROJECTED MODEL PERFORMANCE AGAINST BASELINES (LITERATURE-SUPPORTED ESTIMATES)

Model	MAE Improvement vs. Naive	MAE Improvement vs. ARIMA	R^2
XGBoost (demand forecast)	25–35%	15–25%	0.78–0.88
LSTM (sequential forecast)	20–30%	10–20%	0.74–0.84
XGBoost + LSTM Ensemble	30–40%	20–30%	0.82–0.91
Random Forest (reorder qty)	N/A (different task)	N/A	0.80–0.88

The performance ranges in Table 3 are projected from comparable results reported in the literature [3][4][5][8] rather than from a live deployment, since the framework is at proposal and implementation stage. Actual measured results will be reported in a follow-up evaluation study.

IX. EXPECTED OUTCOMES AND DISCUSSION

A. Forecasting Accuracy

The XGBoost and LSTM ensemble should produce materially better forecasts than the static reorder thresholds or naive seasonal estimates currently used in most SME retail environments. The improvement is expected to be largest during seasonal demand peaks — exactly the periods where errors are most costly — because the temporal and event-based features are specifically designed to capture these effects.

B. Reduction in Stock-Out and Overstock Incidence

Better demand forecasts and dynamic reorder recommendations should translate into lower stock-out rates and reduced end-of-period overstock. The IHL Group's industry estimate of 1–3% of annual sales lost to these combined problems provides a reasonable upper bound on what a well-functioning integrated system could recover.

C. Capital Release Through Dead Stock Identification

The K-Means clustering component addresses a problem that is difficult to quantify in advance but is almost universally acknowledged to exist in established retail operations. A systematic process for identifying and acting on dead stock — even if the action is simply to stop reordering those items — releases capital that can be redeployed into faster-moving categories.

D. Practical Adoption Considerations

Two factors will significantly influence whether a technically sound system actually gets used. The first is trust: buyers who do not understand why the system is making a recommendation are unlikely to follow it. The SHAP attribution layer is specifically designed to address this. The second factor is error handling: the system needs to behave gracefully when data quality is poor, when a product has insufficient history, or when an unusual event produces demand patterns far outside the training distribution.

X. CONCLUSION

This paper has described a framework for embedding AI-driven demand forecasting and inventory optimisation directly within a retail POS architecture, rather than treating these capabilities as external add-ons that require manual integration. The core argument is that predictive intelligence is only useful if it is actionable, and it is only reliably actionable if it sits within the operational system where decisions are made.

The proposed framework combines XGBoost and LSTM models for demand forecasting, a Random Forest component for dynamic reorder optimisation, and K-Means clustering for inventory segmentation — all integrated into the POS interface and designed to run on SME-realistic infrastructure without cloud dependency or specialist data science resources to operate.

Four specific gaps in the existing literature and commercial landscape are addressed: the absence of truly integrated AI-POS systems, the persistence of static threshold-based reorder mechanisms, the limited availability of real-time forecasting in deployed retail systems, and the lack of explainability in inventory-facing AI tools. The SHAP attribution layer is the most distinctive contribution on the explainability dimension.

Several directions for future work are worth noting. Extending the framework to multi-location retail networks introduces interesting questions about cross-location inventory rebalancing. Incorporating supplier lead time uncertainty as a stochastic variable would improve the robustness of the reorder optimisation component. And applying reinforcement learning to the reorder decision problem represents a more ambitious long-term evolution of the system.

ACKNOWLEDGMENT

The authors would like to thank the Department of Computer Science and Engineering at SRM Institute of Science and Technology, Vadapalani for providing the research environment and support for this work.

REFERENCES

[1] IHL Group. (2022). Retailers and the Ghost Economy: $1.75 Trillion Reasons to Be Afraid. IHL Research Report. Franklin, TN: IHL Group.

[2] Silver, E. A., Pyke, D. F., & Thomas, D. J. (2017). Inventory and Production Management in Supply Chains (4th ed.). Boca Raton: CRC Press.

[3] Wang, Y., Zhang, H., & Liu, J. (2022). A Hybrid ARIMA-LSTM Model for Retail Demand Forecasting Under Seasonal Constraints. Expert Systems with Applications, 198, 116820.

[4] Gupta, R., & Sharma, A. (2022). Long Short-Term Memory Networks for Long-Range Sales Forecasting in Fashion Retail. International Journal of Forecasting, 38(4), 1421–1436.

[5] Chen, X., & Li, W. (2021). XGBoost with Temporal Feature Engineering for Inventory Optimisation: Reducing Stock-Outs and Holding Costs in Retail. Computers & Industrial Engineering, 161, 107648.

[6] Kumar, A., & Singh, P. (2020). Reinforcement Learning for Dynamic Replenishment in Retail Inventory Management. Omega, 97, 102105.

[7] Zhao, L., Chen, M., & Park, S. (2023). Graph Neural Networks for Supply Chain Network Optimisation: Capturing Cross-Product Dependencies. Transportation Research Part E, 171, 103034.

[8] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). ACM.

[9] Amosu, O. R., Kumar, P., Ogunsuji, Y. M., Oni, S., & Faworaja, O. (2024). AI-Driven Demand Forecasting: Enhancing Inventory Management and Customer Satisfaction. World Journal of Advanced Research and Reviews, 23(2), 708–719.