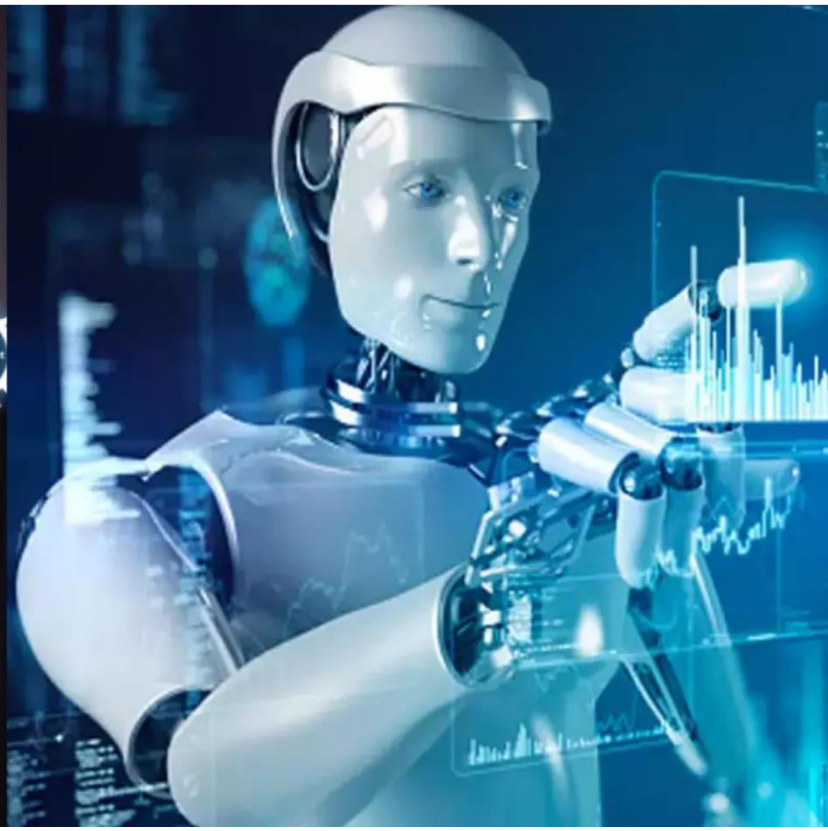
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 15, Issue 4, April 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

Inventory Redistribution Software for Enterprise Resource Planning Using Machine Learning

N. Anithaa, Adrian Jubal S, Ajay R, Mythraeyan S

Assistant Professor, Department of Computer Science and Engineering, Bharath Institute of Science and Technology (BIST), Tambaram, Chennai, Tamil Nadu, India

U.G. Student, Department of Computer Engineering, Bharath Institute of Science and Technology (BIST), Tambaram, Chennai, Tamil Nadu, India

U.G. Student, Department of Computer Engineering, Bharath Institute of Science and Technology (BIST), Tambaram, Chennai, Tamil Nadu, India

U.G. Student, Department of Computer Engineering, Bharath Institute of Science and Technology (BIST), Tambaram, Chennai, Tamil Nadu, India

ABSTRACT: Traditional inventory replenishment systems depend heavily on centralized warehouses to manage stock shortages across retail outlets. This approach frequently results in delays, higher logistics costs, and inefficient use of inventory when demand varies across store locations. This paper proposes an intelligent Inventory Redistribution Software designed to balance inventory across retail outlets by enabling automated stock transfers between nearby stores. The system continuously monitors inventory levels and sales data to identify surplus and deficit locations. Machine learning techniques are applied to analyze demand patterns and predict short-term sales trends. A clustering-based approach groups nearby stores to enable efficient peer-to-peer stock redistribution before warehouse replenishment is requested. The solution is implemented as a web-based ERP application. A FastAPI backend provides RESTful APIs that connect the frontend dashboard with the backend machine learning modules, enabling real-time inventory monitoring and automated redistribution decisions.

KEYWORDS: Inventory management, ERP systems, machine learning, supply chain optimization, demand forecasting, FastAPI.

I. INTRODUCTION

Retail organizations operating multiple outlets often experience uneven distribution of inventory across locations. Some stores face high product demand and quickly run out of stock, while other stores accumulate surplus inventory due to lower demand. Traditional enterprise resource planning (ERP) systems rely heavily on centralized warehouses for inventory replenish- ment. When a store experiences a stock shortage, a request is sent to the warehouse, which then dispatches the required stock. While this approach is widely used, it introduces several inefficiencies. These include delays in restocking, increased transportation costs, and poor utilization of surplus inventory already available within the retail network. This research proposes an Inventory Redistribution Software that enables peer-to-peer stock transfers between nearby stores. By analyzing real-time inventory data and sales trends, the system identifies opportunities to move products from overstocked stores to stores experiencing shortages. This localized redistribution approach improves stock utilization, reduces dependency on warehouses, and enhances supply chain responsiveness.

II. MOTIVATION

Retail chains frequently experience simultaneous stock shortages and excess inventory across different store locations. Local demand patterns vary significantly due to demographic differences, seasonal trends, and consumer preferences. Relying exclusively on warehouse-based replenishment results in several operational challenges:

- Delayed stock replenishment
- Increased transportation and logistics costs

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

- Idle surplus inventory
- Reduced customer satisfaction due to stockouts

The motivation for this project is to design a system capable of intelligently redistributing inventory between nearby stores. By implementing localized stock transfers, organizations can reduce supply chain delays and improve inventory turnover.

III. PROBLEM STATEMENT

Multi-outlet retail businesses often suffer from inventory imbalances caused by variations in local demand. High-demand stores frequently experience stockouts, while low-demand locations accumulate excess inventory. Traditional ERP systems rely primarily on centralized warehouses to resolve shortages. This centralized approach fails to utilize surplus inventory that already exists within nearby stores. The primary challenge is the absence of an intelligent, location-aware system capable of identifying surplus inventory and redistributing it efficiently between stores. A smart redistribution system is required to analyze sales trends, detect inventory imbalances, and perform automated stock transfers.

IV. LITERATURE SURVEY

Several studies have addressed the problem of inventory management and stock redistribution in distributed retail environments. Kapur et al. proposed a two-location inventory model incorporating transshipment and decentralized decision making. Their work demonstrates that coordinated transshipment strategies can improve supply chain efficiency. Shi introduced a transshipment mechanism for dual-channel supply chains using optimization techniques to reduce operational costs. Cheng examined the influence of accounting methods on inventory management strategies and highlighted the importance of financial considerations in operational decision making. Sagar et al. presented an API-based inventory management system enabling real-time integration across distributed retail platforms. These studies highlight the importance of automated inventory visibility, decentralized decision making, and API-driven systems in improving inventory management.

V. EXISTING SYSTEM

Current inventory management approaches fall into three categories.

A. Centralized ERP Systems. Most ERP systems rely on warehouse-based replenishment.

Working Mechanism

- Stores report daily sales data
- Reorder requests are sent to the central warehouse
- The warehouse dispatches stock to stores

Limitations

- High dependency on warehouse replenishment
- Delayed stock delivery
- Inefficient utilization of surplus inventory

B. Manual Store Transfers. In many retail chains, store managers manually coordinate stock transfers via phone calls or emails.

Issues

- Time consuming
- Lack of automation
- Prone to human errors

C. Basic Inventory Software. Basic systems provide stock tracking, alerts, and sales reports but lack advanced analytics such as clustering and demand forecasting.

VI. PROPOSED SYSTEM

The proposed system introduces an intelligent framework for automated inventory redistribution across retail stores. The system workflow includes the following stages.

A. Feature Extraction. Extracted longitude and latitude from store coordinates, and store data in regards to:

- Inventory levels

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

- Historical sales
- Store geographic locations

B. Data Processing. Data preprocessing includes cleaning missing records, aggregating daily sales, and calculating average sales values.

C. Store Clustering. Stores are grouped into clusters based on geographical proximity. This enables efficient redistribution between nearby stores.

D. Demand Analysis. The system analyzes sales trends and inventory levels to identify surplus and deficit stores.

E. Decision Engine. The decision engine matches surplus stores with deficit stores using a cluster-aware, multi-source allocation strategy. For each deficit store, the engine first attempts to source stock exclusively from surplus stores within the same geographic cluster, sorted by proximity. Cross-cluster sources are considered only after same-cluster surplus is fully exhausted. A single deficit store may draw from multiple surplus stores simultaneously, enabling partial fulfilment from several nearby locations rather than relying on a single source.

F. Redistribution. The redistribution function automatically balances inventory between nearby stores using a two-phase approach. In the first phase, intra-cluster transfers are prioritised: the engine iterates over deficit stores in descending order of need and allocates stock from the nearest surplus stores within the same cluster. In the second phase, if intra-cluster surplus is insufficient, the engine draws from cross-cluster surplus stores within the permitted distance threshold. Any unfulfilled deficit after both phases triggers a warehouse replenishment order. Each transfer record carries a scope tag (intra-cluster, cross-cluster, or mixed) to support logistics auditing.

VII. SYSTEM ARCHITECTURE

The proposed system follows a multi-layer architecture consisting of a data processing pipeline, machine learning modules, backend API services, and a visualization interface. The architecture is designed to automatically analyze inventory data, generate redistribution plans, and provide geographic visualization of transfer routes.

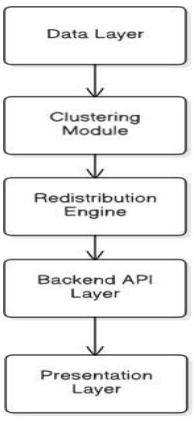


Figure 1: Architecture of the proposed inventory redistribution system.

A. Data Layer. The data layer stores structured inventory information including product IDs, store IDs, stock quantities, and store coordinates. This data is loaded from a CSV dataset and used as the primary input for clustering and redistribution analysis. The dataset also contains geographic coordinates that enable spatial clustering and distance calculations between stores.

B. Processing Layer. The processing layer performs inventory analysis and store clustering. The system first loads inventory data and groups stores using the K-Means clustering algorithm based on their geographic coordinates. The optimal number of clusters is determined using the Elbow method.

Once clusters are formed, the redistribution algorithm analyzes stock levels within each cluster. For each product, the system computes the average stock level across stores and identifies surplus and deficit locations. Surplus stores are matched with deficit stores to generate redistribution plans.

To minimize transportation distance, the system calculates the nearest store using the Haversine distance formula between geographic coordinates. This ensures that inventory transfers occur between geographically close stores.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

C. Redistribution Engine. The redistribution engine processes inventory data cluster-wise and determines optimal transfer quantities between stores. For each product, the algorithm calculates a target stock level based on the cluster average. Stores with inventory above the target are considered surplus stores, while stores below the target are considered deficit stores.

The engine then selects the nearest surplus store for each deficit store and calculates the transfer quantity required to balance inventory. Transfer orders are generated and stored in a structured JSON format that contains product IDs, source stores, destination stores, transfer quantities, and estimated distances.

D. API Layer. The backend API is implemented using the FastAPI framework. The API provides endpoints that allow the frontend interface to retrieve store locations and transfer routes. The API reads the generated redistribution data and returns structured JSON responses containing transfer instructions and route information.

E. Presentation Layer. The presentation layer consists of a web-based dashboard that visualizes store locations and recommended transfer routes on a geographic map. The frontend retrieves redistribution data from the backend API and displays transfer routes between stores, allowing administrators to easily interpret redistribution decisions and logistics paths.

This architecture enables automated inventory balancing, real-time data visualization, and scalable integration with enterprise resource planning systems.

VIII. SYSTEM MODULES

A. Store Clustering and Location Analysis. Stores are grouped using the K-Means clustering algorithm based on geographic coordinates. Pairwise distances between stores are precomputed using the Haversine formula and stored in a symmetric distance matrix, eliminating redundant computation during allocation. The cluster assignment is used directly by the redistribution engine: surplus stores are partitioned into same-cluster and cross-cluster candidate pools relative to each deficit store, ensuring that geographically proximate stores are always preferred as transfer sources.

B. Demand Forecasting Module. Demand forecasting models analyze historical sales data using techniques such as moving averages and linear regression.

C. Surplus–Deficit Detection. This module compares stock levels with predicted demand to identify surplus and deficit stores.

D. Inventory Update and ERP Integration. After redistribution decisions are made, inventory databases are updated automatically.

E. FastAPI Backend Module. The FastAPI backend provides REST API endpoints that allow the frontend ERP interface to communicate with backend analytics modules. The API enables operations such as retrieving inventory data, triggering redistribution algorithms, and updating stock levels.

IX. IMPLEMENTATION

The system was implemented as a web-based ERP application integrating data processing, clustering, and redistribution modules.

A. Data Collection. Inventory, sales, and store location data were stored in structured tables containing product IDs, store IDs, stock quantities, and geographic coordinates.

B. Data Preprocessing. Missing values were cleaned and historical sales data was aggregated to compute average daily demand.

C. Store Clustering. Stores were grouped using the K-Means clustering algorithm based on geographic proximity. The Elbow method was used to determine the optimal number of clusters by minimizing the Within-Cluster Sum of Squares (WCSS).

$$\mathrm{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \quad (1)$$

Cluster centroids were computed as the mean position of all data points in each cluster.

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \quad (2)$$

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

D. Demand Analysis. Stock levels were compared with predicted demand to identify surplus and deficit stores.

$$\text{Surplus} = \max(\text{Stock} - \text{PredictedDemand},\ 0) \quad (3)$$

$$\text{Deficit} = \max(\text{PredictedDemand} - \text{Stock},\ 0) \quad (4)$$

E. Decision Engine. The redistribution engine processes deficit stores in descending order of need. For each deficit store, the allocation procedure operates in two phases.

Phase 1 — Intra-cluster allocation. Surplus stores belonging to the same cluster as the deficit store are sorted by distance and drawn from sequentially until the deficit is met or intra-cluster surplus is exhausted:

$$\text{Move}_i = \min(\text{RemainingNeed},\ \text{Surplus}_i) \quad (5)$$

where i iterates over same-cluster surplus stores ordered by ascending distance.

Phase 2 — Cross-cluster allocation. If need remains after Phase 1, the engine repeats the same procedure over surplus stores from other clusters, subject to the same distance threshold $d_{\max}$.
Any deficit not resolved after both phases is escalated to a warehouse order. Transfer records include a scope field (intra-cluster, cross-cluster, or mixed) and a distance km field for logistics traceability. A global transfer limit cap is enforced across all products; once reached, remaining deficits are routed to warehouse orders rather than being silently dropped.

F. Backend API Implementation. FastAPI was used to implement backend services. The API enables the frontend dashboard to retrieve inventory data, trigger redistribution logic, and update stock records.

X. RESULTS AND DISCUSSION

The proposed inventory redistribution system was evaluated using simulated multi-store inventory datasets designed to reflect realistic variations in stock levels across different locations. During testing, the system effectively analyzed these datasets to identify surplus and deficit stores within each cluster, ensuring that imbalances were accurately detected. Based on this analysis, it generated optimal transfer plans that prioritized nearby locations, thereby supporting efficient and practical inventory movement. The results demonstrate that the system can consistently produce reliable redistribution strategies while maintaining a focus on minimizing transfer distances and improving overall stock balance within clusters.

A. Inventory Transfer Decision Output. The backend decision engine performs a comprehensive analysis of stock distribution at the cluster level, systematically evaluating the inventory status of each store to identify imbalances between surplus and deficit locations. Based on this analysis, it determines the most efficient transfer quantities required to achieve optimal redistribution while adhering to predefined constraints such as cluster boundaries and availability limits. The engine processes multiple products simultaneously, ensuring that allocation decisions are both scalable and consistent across varying inventory scenarios. Its output reflects not only the calculated transfer quantities but also the underlying logic used to arrive at those decisions, providing transparency in how stock movements are prioritized and executed.
Furthermore, the system generates a structured terminal output that captures these allocation decisions in a clear and interpretable format, allowing users to trace the flow of inventory between stores. This output includes detailed transfer instructions, typically represented as discrete actions, which collectively illustrate how the redistribution plan is implemented across different clusters and product categories. Fig. 2 shows an example of this terminal output, highlighting how the system consolidates and presents the results of its analysis after processing multiple products and store clusters.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

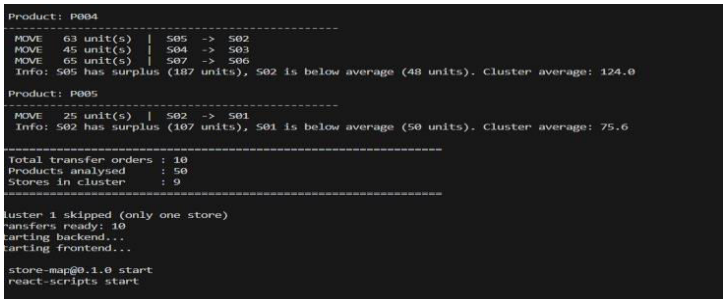


Figure 2: Backend output showing detected surplus stores, deficit stores, and recommended inventory transfers.

The system evaluates product demand across stores and automatically calculates redistribution quantities based on surplus–deficit analysis. In the example shown, several transfer operations are generated between stores within the same cluster. This demonstrates the ability of the system to dynamically rebalance stock levels and reduce dependency on centralized warehouses.

The enhanced allocation engine further demonstrates multi-source splitting behaviour, wherein a single deficit store can receive stock from two or more surplus stores within the same cluster when no individual source possesses sufficient inventory to fulfill the requirement on its own. This ensures that demand is met efficiently without over-reliance on a single supply point. In such scenarios, the output log clearly reflects this behavior by displaying multiple MOVE entries that share the same destination store, with each entry explicitly tagged according to its respective transfer scope. This level of detail in the logging mechanism helps in tracing how inventory is distributed across sources. Additionally, cross-cluster transfers are observed only in situations where the available surplus within the originating cluster has been completely utilized. This confirms that the engine consistently prioritizes intra-cluster balancing and adheres to the cluster boundary preference, only expanding the search to other clusters when it becomes absolutely necessary.

B. Store Location and Transfer Route Visualization. To support efficient logistics planning, the system provides a geographic visualization of store locations alongside the potential transfer routes generated by the redistribution algorithm. Each store's coordinates are accurately plotted on a digital map, offering a clear spatial overview of both surplus and deficit points within the network. The system then highlights the optimal path between a surplus store and a deficit store, visually representing the most efficient route for inventory movement based on the allocation decisions. This mapping not only aids in understanding how stock is redistributed but also assists in practical route planning and operational coordination. By presenting these routes in a visual format, the system makes it easier to interpret transfer decisions and evaluate their feasibility in real-world conditions. Fig. 3 illustrates an example route between two retail locations within the same cluster, demonstrating how the recommended movement is conveyed through the geographic interface.

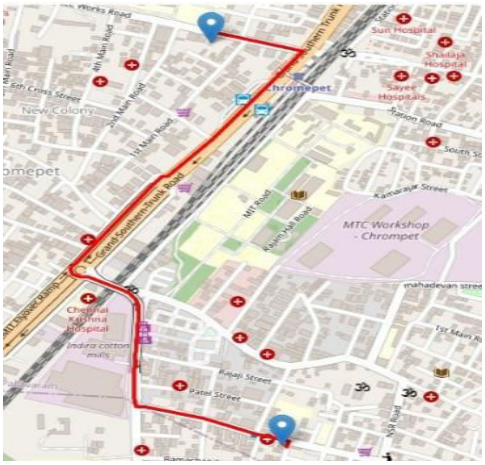


Figure 3: Visualization of store locations and transfer route generated for inventory redistribution.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504028|

The visualization helps administrators clearly identify the origin and destination of the transfer, along with the route that should be followed for transportation. By presenting this information on a map, the system enables decision makers to better understand the spatial relationships between stores and evaluate the practicality of the redistribution plan. The map-based visualization helps administrators identify the most efficient redistribution paths between stores. By clustering nearby stores and calculating optimal routes, the system reduces transportation time and operational costs while ensuring faster stock balancing across the retail network. Overall, the experimental results demonstrate that the proposed system can effectively detect inventory imbalances, recommend redistribution strategies, and provide clear visualizations for operational decision making.

XI. CONCLUSION

This paper presents an intelligent inventory redistribution system designed to improve inventory utilisation across multi- store retail networks. By enabling automated, cluster-aware stock transfers between nearby stores, the system reduces dependency on centralised warehouses and improves supply chain responsiveness. The system integrates a two-phase multi-source allocation engine with K-Means geographic clustering and a Haversine- based precomputed distance matrix. Intra-cluster surplus is always exhausted before cross-cluster sources are considered, minimising transportation distance. A single deficit store may draw from multiple surplus stores simultaneously, eliminating the single-source bottleneck present in conventional redistribution approaches. The FastAPI-based backend exposes these decisions as structured JSON with per-transfer scope and distance metadata. Future work will focus on improving demand prediction accuracy, evaluating the system on large-scale retail datasets, and incorporating dynamic re-clustering as store demand profiles shift over time.

XII. ACKNOWLEDGMENT

The authors express sincere gratitude to Mrs. N. Anithaa for her guidance throughout this research project. We thank the Department of Computer Science and Engineering at Bharath Institute of Science and Technology (BIST) for providing computational resources and support.

REFERENCES

[1] S. Kapur, N. Rudi, and D. Pyke, "A Two-Location Inventory Model with Transshipment and Local Decision Making," *Management Science*, vol. 47, no. 12, pp. 1668–1680, 2001.
[2] M. Shi, "Research on Order-Inventory Transshipment Mechanism in Dual-Channel Distribution Supply Chain," *Open Journal of Business and Management*, vol. 7, pp. 1120–1130, 2019.
[3] X. Cheng, "Research on Inventory Management in Modern Business," *Advances in Economics, Management and Political Sciences*, vol. 60, pp. 1–8, 2024.
[4] M. Sagar, M. Dharaniya, M. Nesly, and K. S. Rethina, "Inventory Management Using API," *International Scientific Journal of Engineering and Management*, vol. 4, 2025.
[5] S. Nasseri, M. Rezaei, and A. Karimi, "Applying Machine Learning in Retail Demand Prediction," *Applied Sciences*, vol. 13, no. 19, p. 11112, 2023.
[6] K. Taparia, R. Gupta, and P. Sharma, "Improved Demand Forecasting of a Retail Store Using Machine Learning Algorithms,"
Journal of Green Engineering and Urban Development, vol. 4, no. 2, pp. 45–58, 2024.
[7] S. Mukherjee, G. J. Y. Chen, and H. Wang, "AR-MDN: Associative and Recurrent Mixture Density Networks for e-Retail Demand Forecasting," in *Proc. ACM Conf. on Recommender Systems*, 2018, pp. 62–69.
[8] N. Fatima and M. Salam, "Predictive Framework for Inventory Optimization Using Machine Learning," *arXiv preprint*, arXiv:2601.05033, 2026.
[9] S. A. Torabi, M. Baghersad, and S. A. Mansouri, "Fulfillment Allocation and Inventory Transshipment in E-commerce Systems,"
Transportation Research Part E: Logistics and Transportation Review, vol. 79, pp. 128–147, 2015.
[10] J. Dennis and T. A. Hu¨ bner, "Operational Flexibility Through Inventory Transshipment," *Omega*, vol. 39, no. 6, pp. 595–605, 2011.
[11] Y. Yu, L. Liang, and Q. Qi, "Inventory Transshipment Strategy Alliances in Supply Chain Systems," *Sustainability*, vol. 11, no. 3,
p. 708, 2019.
[12] E. Arikan, M. Fichtinger, and A. Syntetos, "Risk Pooling Benefits in Inventory Systems with Lateral Transshipments," *International Journal of Production Economics*, vol. 199, pp. 19–28, 2018.
[13] H. Turan, M. Selim, and O. Aydin, "A Multi-Location Inventory Redistribution Model for Retail Supply Chains," *Computers and Industrial Engineering*, vol. 112, pp. 569–579, 2017.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details