Integrating Demand Prediction with Inventory Optimization for E-commerce Logistics

Project ID: 121

Kexin Chen, Chun-Ming Chang, Yucy Yang, Mario Zhang, Loris Emanuelli

Executive Summary

This project addresses a fundamental challenge in modern e-commerce operations: the disconnect between demand forecasting and operational decision-making. In many real-world systems, predictive models are optimized for statistical accuracy without considering how their outputs are used in downstream decisions such as inventory allocation and fulfillment planning. To bridge this gap, our project develops an integrated pipeline that combines machine learning–based demand prediction with an optimization model for multi-warehouse inventory and delivery planning.

Our team designed a data-driven framework using transaction-level e-commerce data, where we first mitigate data sparsity through SKU clustering and then predict next-day demand at the cluster–warehouse level. We evaluated multiple forecasting approaches, including baseline heuristics and machine learning models such as XGBoost, using metrics like MAE, RMSE, and WAPE. These predictions are then embedded into a mixed-integer optimization model that determines inventory allocation and fulfillment decisions under capacity and delivery constraints. Importantly, we assess model performance not only by prediction accuracy but also by operational outcomes, such as total cost and service level.

Through this process, we gained key insights into the trade-offs between model complexity and practical impact, as well as the importance of aligning predictive objectives with downstream decisions. Our results demonstrate that modest improvements in prediction, when integrated with optimization, can lead to meaningful operational gains.

For future work, we recommend extending the dataset to longer time horizons to better capture seasonality and promotion effects, incorporating richer feature engineering (e.g., pricing and user behavior), and exploring decision-focused learning approaches that directly optimize operational objectives during model training.

I. Introduction: Integrating Prediction and Optimization Improves E-commerce Operations

E-commerce logistics systems face increasing complexity due to the interaction between consumer behavior, inventory allocation, and fulfillment operations. In large multi-warehouse networks, firms must not only predict what customers will buy, but also decide where inventory should be placed and how orders should be fulfilled across locations. These decisions are tightly connected, since errors in one part of the system can quickly affect delivery speed, stock availability, and total operating cost. While modern machine learning models can often predict short-term demand reasonably well, operational decisions such as inventory positioning, cross-warehouse fulfillment, and replenishment are still frequently made in separate planning stages. This disconnect can create inefficiencies such as excessive cross-warehouse shipments, delivery delays, and unnecessary inventory holding costs.

This problem becomes even more challenging in multi-warehouse systems, where decisions at one location can affect cost and service performance across the rest of the network. In practice, replenishment decisions often involve lead times and broader planning constraints, while fulfillment decisions are more immediate and reactive, which makes the two difficult to coordinate [1]. As a result, better forecast accuracy alone does not necessarily translate into better operational performance. Existing research

suggests that predictive models should ultimately be evaluated based on the decisions they support, rather than by forecast error alone [2]. This makes integrated planning especially important in e-commerce logistics, where fulfillment decisions must be made quickly under network and inventory constraints.

Thesis: Our results show that integrating consumer-side demand prediction with a daily multi-warehouse optimization model leads to better operational outcomes than treating forecasting and operations as separate stages. By aligning demand forecasts with fulfillment decisions under network constraints, the integrated framework provides a more effective basis for reducing cost and improving service performance. More broadly, this project highlights why demand forecasting in logistics should be judged not only by statistical accuracy, but also by the operational decisions it enables.

II. Background: Existing Systems Fail to Align Prediction with Decisions

Despite significant advances in demand forecasting, many real-world supply chain systems still operate under a fragmented paradigm in which prediction and operational decision-making are developed and executed separately. Forecasting models are typically optimized for statistical accuracy—using metrics such as RMSE or MAE—without explicitly considering how predictions will be used in downstream decisions such as inventory allocation, fulfillment routing, or procurement. As highlighted in our project context, this separation can lead to suboptimal outcomes, including cross-warehouse fulfillment, delayed deliveries, and excess inventory holding .

This disconnect is widely observed in industry. Large e-commerce platforms such as Amazon and JD.com rely on sophisticated machine learning models to generate SKU-level demand forecasts. However, these predictions are often passed into rule-based or heuristic-driven operational systems. From an engineering leadership perspective, this reflects a broader systems integration challenge: optimizing individual components does not guarantee optimal system-level performance. In practice, even small forecast errors can propagate through the supply chain and lead to amplified inefficiencies, such as stockouts in high-demand regions and overstocking in low-demand locations. This phenomenon is closely related to the bullwhip effect, where demand variability increases upstream due to forecasting errors, lead times, and decentralized decision-making [3].

At the same time, inventory decisions in many organizations remain driven by static heuristics, such as fixed safety stock rules or base-stock policies. While these approaches are computationally simple and operationally robust, they are inherently reactive and lack the flexibility to adapt to short-term demand fluctuations caused by promotions, pricing changes, or shifts in consumer behavior. Prior work on "forecasting at scale" emphasizes that operational success depends not only on predictive accuracy but also on building robust pipelines that integrate forecasting, monitoring, and decision-making processes [4]. From a leadership standpoint, this highlights the importance of deploying ML systems in a way that directly supports operational objectives rather than treating them as standalone analytical tools.

These challenges become even more pronounced in multi-warehouse networks. In such systems, inventory placement decisions affect not only local fulfillment performance but also cross-warehouse transfers, delivery times, and overall network efficiency. Research in supply chain optimization shows that the positioning of inventory can be as important as the total inventory level itself, as it directly influences service levels and system-wide costs [5]. Misalignment between predicted demand and inventory

positioning can result in increased shipping distances, higher transfer costs, and degraded customer experience. This creates a complex trade-off between holding costs, transportation costs, and delivery performance, which cannot be effectively addressed through isolated decision-making.

Recent research has begun to bridge this gap by proposing integrated frameworks that connect prediction and optimization. The "predict-then-optimize" paradigm argues that predictive models should be evaluated based on their impact on downstream decisions rather than purely on statistical accuracy [6]. This perspective represents a shift toward decision-centric evaluation, where the value of a model is measured by its contribution to business outcomes such as cost reduction and service improvement. In parallel, contextual optimization methods further emphasize the importance of incorporating uncertainty, behavioral signals, and operational constraints into a unified decision-making framework [7].

In the context of e-commerce, demand is strongly influenced by dynamic factors such as pricing, promotions, and user behavior. Transaction-level analyses have shown that price elasticity varies significantly across products and contexts, suggesting that models incorporating behavioral and economic signals are better suited to capturing real demand patterns [8]. This reinforces the need for integrated systems that combine predictive modeling with operational decision-making to fully leverage data.

From an engineering leadership perspective, the persistence of fragmented systems also reflects organizational challenges. Data science teams are often evaluated based on prediction accuracy, while operations teams are evaluated based on cost and service metrics, leading to misaligned incentives. Effective system design requires cross-functional alignment, where both prediction and decision components are optimized toward shared objectives. This includes establishing feedback loops between forecasting and operations, defining metrics that reflect end-to-end performance, and designing systems that can adapt dynamically to changing conditions.

In summary, the primary challenge is not the lack of advanced forecasting models or optimization techniques, but the absence of integrated systems that align prediction with decision-making in real operational environments. Addressing this gap requires both technical integration and leadership-driven system design, ensuring that predictive insights translate into measurable business value.

III. Data: Multi-Source Signals Enable Demand and Decision Modeling

3.1 Dataset description

The analysis in this project is based on seven tables from the JD.com transaction-level dataset, which together capture both customer-side behavior and warehouse-level operations[9].

These tables include:

- Orders: transaction-level purchase records
- Clicks: browsing behavior prior to purchase
- SKUs: product attributes and categories
- Users: customer characteristics

- Delivery: shipment and fulfillment timelines
- Inventory: SKU availability at each warehouse
- Network: warehouse-to-region mapping and routing structure

The key strength of this dataset is that it connects demand generation (customer behavior) with fulfillment execution (warehouse operations). This enables us to model the entire pipeline from prediction to decision-making, rather than treating them as separate problems.

3.2 Parameters setting

3.2.1 Customer Side

On the customer side, demand-related inputs are constructed using aggregated transaction data and clustering-based preprocessing. To reduce sparsity at the SKU level, SKUs are grouped into clusters using K-means clustering based on features including demand statistics, user behavior (clicks), pricing, and promotion indicators. The resulting SKU-to-cluster mapping is used to define the aggregation level for all downstream variables.

Feature inputs for demand forecasting are constructed from historical transaction data. These include lagged demand variables and aggregated pricing and promotion features at the cluster–warehouse level. All features are computed consistently across the observation window to ensure alignment with the temporal structure of the data.

The forecasting model is selected from a set of candidate methods, including linear regression, random forest, LightGBM, and XGBoost, along with a baseline approach using lag-1 predictor. Based on empirical performance, XGBoost is chosen as the primary model, with hyperparameters tuned using validation data to balance predictive accuracy and generalization.

Finally, sensitivity analysis is conducted by perturbing key input variables, including price and lagged demand, within a predefined range to evaluate the robustness of predicted demand. This ensures that the generated forecasts remain stable and suitable for integration into the downstream optimization model.

3.2.2 Inventory Side

On the inventory side, we construct all model parameters based on a combination of empirical data and standard assumptions. The cluster-level price p_s is computed by first estimating SKU-level prices using order data. For each SKU, we calculate its unit price by averaging its original_unit_price across all transactions. The cluster price is then defined as the average price of all SKUs belonging to that cluster based on the SKU-to-cluster mapping data. Demand $D_{s,j}$ is constructed using both training and testing daily demand datasets, covering a 31-day period. For each warehouse j and cluster s, we aggregate daily demand and compute the average daily demand over the full period, treating missing observations as zero to ensure consistency.

Due to the absence of explicit inventory quantities in the data, the initial inventory $I^0_{s,j}$ is approximated using SKU availability. Specifically, each observed SKU record is treated as one unit of inventory, and the

cluster-level inventory at each warehouse is calculated as the total number of available SKUs within that cluster. The initial inventory is then defined as the average daily inventory over the same 31-day period. The procurement eligibility parameter W_j is determined using network data. Only when a warehouse is classified as a large warehouse, can it purchase additional SKUs, which is expressed by setting $W_j = 1$. The route availability parameter $r_{n,m}$ is defined as a binary indicator equal to 1 if a corresponding (n,m) route exists in the dataset and 0 otherwise.

Finally, the cost parameters are set based on industry standards and economic intuition. The holding cost h is assumed to be 20% annually and converted to a daily rate, while the purchasing cost u is set to 5% to reflect procurement premiums. The delivery time cost k is calibrated such that a one-day delay corresponds to approximately a 1% loss in value, converted to an hourly rate consistent with the time unit in the dataset.

IV. Customer-Side Prediction: Clustering Enables Scalable Demand Forecasting

4.1 Motivation and Problem Setting

Accurate demand forecasting is a critical component of the proposed predict–optimize framework. In large-scale e-commerce settings, demand is naturally defined at the SKU–warehouse–day level. However, directly modeling demand at this granularity presents significant challenges due to high sparsity, limited historical observations per SKU, and substantial noise in daily demand patterns.

To address these challenges, we reformulate the prediction task at the *cluster–warehouse* level. Let $D_{s,j,t}$ denote the demand for cluster *s* at warehouse *j* on day *t*. The objective is to learn a predictive function:

$$\widehat{D}_{s,j,t+1} = F(X_{s,j,t})$$

where $X_{s,j,t}$ represents historical features derived from transaction, pricing, and promotion data. This formulation ensures that predictions are both statistically stable and directly aligned with the decision variables used in the downstream optimization model.

4.2 Clustering for Demand Aggregation

Directly modeling demand at the SKU–warehouse level presents significant challenges due to extreme sparsity and variability in the data. Our empirical analysis shows that the raw SKU–warehouse demand tensor has a density of only 1.5%, implying over 98% sparsity. Even when aggregated to the SKU–region level, sparsity remains above 94%, meaning that most SKU–location pairs exhibit zero demand on the majority of days. This lack of observations makes it difficult for standard forecasting models to learn stable and generalizable patterns.

To address these issues, we aggregate SKUs into clusters prior to demand modeling. This approach is motivated by three key considerations. First, aggregation reduces sparsity by increasing the proportion of non-zero observations, thereby providing richer signals for model training. Second, clustering improves statistical stability, as demand at the cluster level tends to exhibit smoother temporal dynamics and lower variance compared to individual SKUs. Third, it enhances computational scalability by significantly reducing the number of time series that must be modeled.

Empirically, clustering leads to substantial improvements in data quality and structure. The total number of time series is reduced by over 96%, making the modeling problem more tractable. At the same time, the non-zero rate increases significantly—by approximately 4.58× in the training set and 1.76× in the test set—indicating a much denser and more informative dataset. Additionally, the proportion of high-activity series (defined as those with a non-zero rate of at least 0.5) increases by about 63%, further demonstrating improved data richness.

From a variability standpoint, clustering also stabilizes demand patterns. The coefficient of variation decreases by approximately 70%, suggesting a substantial reduction in relative variability. Distributional analysis confirms that cluster-level demand not only exhibits higher non-zero rates but also more concentrated variance compared to raw SKU-level data. Together, these improvements create a more reliable and learnable structure for downstream forecasting models.

Methodology and Train–Test Consistency

To ensure a realistic evaluation setting, we construct the training and test datasets using a time-based split that respects the temporal nature of demand. Specifically, we use the first portion of the dataset (e.g., March 1–24) as the training set and reserve the remaining period (e.g., March 25–31) as the test set. This setup mimics a real-world deployment scenario in which models are trained on historical data and used to predict future demand.

We apply K-means clustering using features computed from the training period only, including demand statistics, pricing, and promotion indicators. All features are standardized prior to clustering, and the number of clusters is selected based on silhouette score analysis.

To avoid information leakage, the learned SKU-to-cluster mapping is fixed and applied to both training and test datasets. Demand forecasting models are similarly trained on the training set and evaluated on a held-out test set, ensuring a realistic deployment setting.

4.3 Model Selection and Feature Sensitivity

Feature Set Design

We construct a sequence of feature sets to evaluate the contribution of different information sources:

- **Feature Set A**: Base identifiers + lag-1 demand
- **Feature Set B**: Feature Set A + lag-2 and lag-3 demand
- **Feature Set C**: Feature Set B + rolling statistics and demand growth
- **Feature Set D**: Feature Set C + calendar features
- **Feature Set E**: Feature Set D + aggregate cluster and warehouse demand signals
- **Feature Set F**: Feature Set E + user behavior features (clicks and conversions)

This progression enables a structured sensitivity analysis from simple autoregressive inputs to richer contextual features.

Model Comparison

We evaluate XGBoost, LightGBM, and Random Forest across all feature sets. Among these, XGBoost with Feature Set B achieves the best performance, with:

- WAPE = 0.3267
- RMSE = 19.36
- MAE = 6.32
- MAPE ≈ 0.61

Tree-based models consistently outperform simpler baselines, indicating the presence of nonlinear relationships in demand dynamics. However, performance differences across advanced models are relatively small, suggesting that feature design is more critical than model choice.

Among the evaluation metrics, **WAPE (Weighted Absolute Percentage Error)** is the most appropriate for this problem setting. Unlike RMSE and MAE, which measure absolute error magnitude, WAPE normalizes errors by total demand, making it more robust to scale differences across clusters and better aligned with aggregate business impact. In a multi-SKU, multi-warehouse environment where demand levels vary significantly, WAPE provides a more meaningful measure of how well the model captures overall demand distribution.

Given that downstream decisions depend on aggregated demand inputs to the optimization model, WAPE offers the best balance between interpretability and operational relevance. Nevertheless, we report all metrics to provide a comprehensive view of model performance. Detailed comparisons of how these prediction metrics translate into operational outcomes, including cost and service performance in the optimization model, are presented in Section 7.

Comparison with Baseline

We compare the best-performing model against a lag-1 baseline:

- **XGBoost (Set B)**: WAPE = 0.3267
- **Lag-1 baseline**: WAPE = 0.3556

This corresponds to an improvement of approximately 8% in WAPE, demonstrating that the proposed model captures meaningful structure beyond simple temporal persistence. At the same time, the strong baseline performance highlights that demand is highly autocorrelated, and much of the predictive signal comes from recent observations.

4.4 Limitations

Despite the effectiveness of the proposed approach, several limitations remain.

First, the use of clustering introduces a loss of granularity. While aggregation improves statistical stability, it may obscure important differences between individual SKUs within the same cluster, potentially leading to suboptimal predictions for specific items. In addition, clustering relies on features

derived from historical demand and pricing, which may introduce bias and limit the model's ability to generalize to unseen patterns. The clustering structure is also static and does not adapt to temporal changes in demand behavior.

Second, the dataset itself imposes important constraints. The available data spans only a short time horizon (approximately one month), which limits the model's ability to capture longer-term seasonality and structural trends. This is particularly relevant for features such as promotions and user behavior, whose effects may require longer observation periods to be reliably estimated. As a result, the limited impact of promotion and click-based features observed in the sensitivity analysis may be driven by insufficient temporal coverage rather than their true predictive value.

Finally, the high level of aggregation and the short time window together restrict the modelâ€™s ability to fully exploit richer feature sets. This may explain why simpler models with lag-based features outperform more complex specifications in our experiments. These same constraints also limit the scope of sensitivity analysis in the current report: the robustness checks we add below are local one-factor stress tests rather than a full uncertainty study over all reconstructed parameters.

V. Inventory Optimization: Translating Forecasts into Decisions

5.1 Optimization Model

5.1.1 Problem Description:

Given predicted next-day demand derived from customer interaction and transaction data, the system must determine how to allocate inventory, route orders, and replenish stock across warehouses. These decisions are interdependent, as inventory placement and fulfillment routing jointly affect delivery time, cost, and service performance. The objective is to minimize total operational cost while satisfying demand and respecting inventory and network constraints.

5.1.2 Model Description:

We propose a Predict-and-Optimize framework in which demand forecasts are directly embedded into a multi-warehouse optimization model. The model jointly determines:

- Fulfillment decisions: how customer demand is served across warehouses
- Transshipment decisions: how inventory is reallocated between warehouses
- Replenishment decisions: how inventory is restocked at selected locations.

The model captures operational trade-offs between delivery speed, inventory holding, and purchasing cost. It also incorporates realistic constraints derived from the data, including warehouse capacity, delivery network connectivity, and inventory availability.
The problem is formulated as a Mixed-Integer Linear Programming (MILP) and solved on a rolling daily basis.

5.1.3 Parameters & Sets:

$s \in S$: set of SKU clusters

$j, n, m \in W$: set of warehouses (regions)
$D_{s,j}$: predicted demand of SKU cluster s at warehouse j
$I_{s,j}^{0}$: initial inventory of SKU cluster s at warehouse j
C_j: capacity of warehouse j
p_s: price of SKU cluster s
$t_{n,m}$: delivery time from warehouse n to m
$r_{n,m} \in \{0,1\}$: route availability between warehouses
$W_j \in \{0,1\}$: indicator if warehouse j can replenish inventory
k: delivery delay cost coefficient (as % of price)
h: inventory holding cost coefficient (as % of price)
u: replenishment (purchasing) cost coefficient (as % of price)

5.1.4 Decision Variables:
$f_{s,n,m} \geq 0$: quantity of SKU cluster s shipped from n to satisfy demand at m
$y_{s,,n,m} \geq 0$: transshipment quantity from n to m
$q_{s,,j} \geq 0$: replenishment quantity at warehouse j
$I_{s,j}^{1} \geq 0$: end-of-day inventory of SKU cluster s in warehouse j

5.1.5 Objective:

$$Min \sum_{s} \left[k \sum_{n} \sum_{m} p_s \; t_{n,m} \; f_{s,n,m} + h \sum_{j} p_s I_{s,j}^{1} + u \sum_{n} p_s q_{s,n} \right]$$

Minimizes total cost including:

- delivery cost
- inventory holding cost
- replenishment cost

5.1.6 Constraints:
(1) Inventory Balance

$$I_{s,j}^{1} = I_{s,j}^{0} - \sum_{m} f_{s,j,m} + q_{s,j} + \sum_{n} y_{s,n,j} - \sum_{m} y_{s,j,m}, \forall s, j$$

(2) Capacity Constraint

$$\sum_{s} I_{s,j}^{1} \leq C_j, \; \forall j$$

(3) Demand Satisfaction

$$\sum_{n} f_{s,n,m} = D_{s,m}, \forall s, m$$

(4) Inventory Availability

$$\sum_{m} f_{s,n,m} \leq I^{0}_{s,n}, \forall s, n$$

$$\sum_{m} y_{s,n,m} \leq I^{0}_{s,n}, \forall s, n$$

(5) Routing Feasibility

$$f_{s,n,m} \leq M r_{n,m} \quad , \forall s, n, m$$
$$y_{s,n,m} \leq M r_{n,m} \quad , \forall s, n, m$$

(6) Replenishment Feasibility

$$q_{s,j} \leq M W_{j}, \forall s, j$$

5.2 Baseline Model

5.2.1 Policy Description:

We construct a rule-based baseline policy that reflects common operational practice in e-commerce logistics systems using the same JD.com data inputs.

This policy operates in a sequential and reactive manner, without global coordination. Fulfillment and replenishment decisions are made independently, based only on current inventory and realized demand.

Specifically, the policy follows three steps:

1. Prioritize local fulfillment using available inventory
2. Use cross-warehouse fulfillment when local inventory is insufficient
3. Perform end-of-day replenishment to match observed demand

This baseline serves as a benchmark to evaluate the benefits of integrated optimization.

5.2.2 Decision Rules:

Step 1: Local Fulfillment

$f_{s,j,j} = min(D_{s,j}, I^{0}_{s,j}), \forall s, j$ *(The demands are prioritized for local fulfillment.)*

$R_{s,j} = D_{s,j} - f_{s,j,j}$ *(Remaining demand)*

$\widehat{I_{s,j}} = I^{0}_{s,j} - f_{s,j,j}$ *(Remaining Inventory)*

Step 2: Cross-Warehouse Fulfillment

$$While\ R_{s,m} > 0:$$

Candidate warehouses $N_{s,m} = \{n \neq m \mid r_{n,m} = 1, \widehat{I_{s,n}} > 0\}$

From $N_{s,m}$ *select warehouse n with minimum* $t_{n,m}$

$$f_{s,n.m} = min(R_{s,m}, \widehat{I_{s,n}})$$

$$R_{s,m} \leftarrow R_{s,m} - f_{s,n.m}$$

$$\widehat{I_{s,n}} \leftarrow \widehat{I_{s,n}} - f_{s,n.m}$$

Step 3: End-of-Day Inventory Replenishment

If $\widehat{I_{s,j}} < D_{s,j}$:

$$q_{s,j} = min(D_{s,j} - \widehat{I_{s,J}}, C_j - \widehat{I_{s,J}})$$

$$I^1_{s,j} = \widehat{I_{s,J}} + q_{s,j}, \forall s, j$$

Step 4: Cost Evaluation

$$Total\ Cost = \sum_s \left[k \sum_n \sum_m p_s\ t_{n,m}\ f_{s,n,m} + h \sum_j p_s I^1_{s,j} + u \sum_n p_s q_{s,n} \right]$$

VI. Integrated System: Prediction Quality Should Be Measured by Decision Outcomes

In traditional demand forecasting systems, model performance is typically evaluated using statistical error metrics such as RMSE, MAE, or WAPE. While these metrics quantify how closely predictions match observed demand, they do not necessarily reflect how useful those predictions are for downstream operational decision-making. In supply chain contexts, the ultimate objective is not to minimize prediction error, but to improve decisions related to inventory allocation, fulfillment, and delivery.

Our project is built as a decision system rather than a standalone forecasting exercise. Several candidate forecast files are tested on the same warehouse-cluster demand panel, including the proposed XGBoost file, two LightGBM variants, a Random Forest variant, and a lag-1 baseline that carries forward the previous day's realized demand. Those forecasts feed the same planning model and are judged on the decisions they produce under realized demand.

Forecast quality is evaluated through the operating decisions it produces, not as an isolated prediction score.

Figure 1. Candidate forecast files feed a receding-horizon optimization loop and are evaluated on realized operational outcomes.

6.1 Receding-horizon optimization

Planning is executed through a daily receding-horizon linear program. At the start of each day, the model takes the current inventory state and the forecasted demand for each warehouse-cluster pair, then decides procurement, transshipment, and ending inventory subject to capacity, procurement eligibility, route feasibility, and delivery-time penalties. Once the day is simulated, the realized ending inventory becomes the next day's starting state. This creates a realistic link between forecast quality and how the network evolves over the week.

6.2 Why operational evaluation matters

This structure matters because forecasting metrics alone do not reveal whether a model creates a good operating plan. In a multi-warehouse network, the important question is not only which forecast is closest to demand, but which forecast places inventory more effectively, reduces avoidable transfers, controls congestion, and preserves service. That is why the main evaluation combines forecast accuracy with realized total cost, service level, shortage, overflow, procurement, and transfer volume.

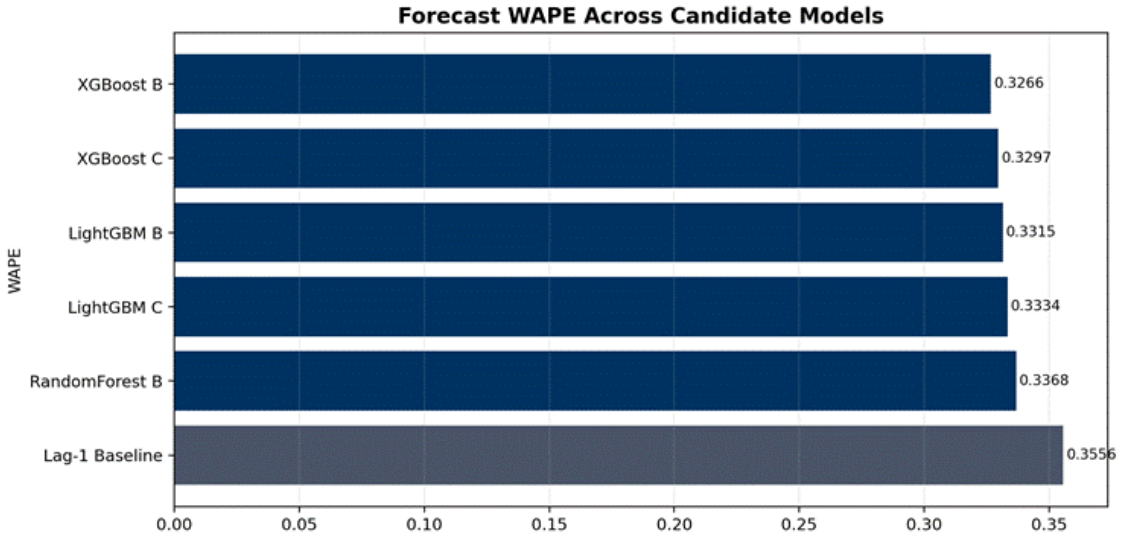

Figure 2. Forecast WAPE across all candidate models. XGBoost B is the strongest file on pure forecast fit; the operational implications of that ranking are examined in Sections 7.2 through 7.5.

VII. Findings: Prediction Alone is Insufficient, Integration is Key

7.1 Forecast ranking

Across all tested candidate files, XGBoost B is the strongest forecasting model. It achieves the lowest WAPE at 0.3266, the lowest RMSE at 18.9127, and the lowest weighted proxy cost at 321,407.37. XGBoost C follows closely, then LightGBM B, LightGBM C, and RandomForest B. The lag-1 baseline is the weakest on forecast fit, with WAPE of 0.3556 and the highest RMSE in the group.

Model	WAPE	Realized Cost	Service Level	Shortage Units	WAPE Rank	Cost Rank
XGBoost B	0.3266	653,495.01	0.8069	32,979.03	1	1
Lag-1 Baseline	0.3556	654,617.88	0.8103	32,395.39	6	2
XGBoost C	0.3297	656,223.19	0.8062	33,102.16	2	3
LightGBM B	0.3315	659,989.31	0.8032	33,602.64	3	4
LightGBM C	0.3334	662,897.49	0.8020	33,818.36	4	5
RandomForest B	0.3368	663,276.27	0.8011	33,958.99	5	6

Table 1. Candidate ranking across forecast fit and weekly operating outcomes.

7.2 Optimization ranking

The same ordering largely carries into the optimization layer. XGBoost B also produces the lowest realized weekly total cost at 653,495.01. Lag-1 is second at 654,617.88, followed by XGBoost C, LightGBM B, LightGBM C, and RandomForest B. This is an important result because it shows that the model with the best forecast accuracy also delivers the best operating cost in this experiment.

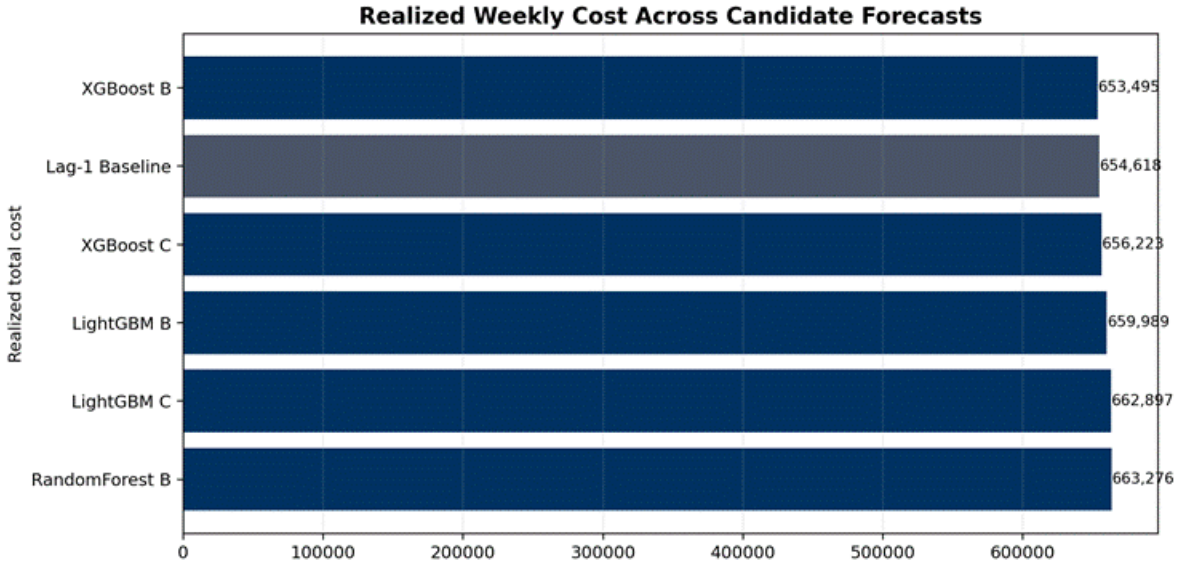

Figure 3. Realized weekly cost across candidate forecasts. The optimization layer separates the candidate models more clearly than the forecast metrics alone.

7.3 Cost versus service

The operational picture is still more nuanced than a single ranking. Lag-1 achieves the highest realized service level at 0.8103 and the fewest shortage units at 32,395.39, while XGBoost B comes in slightly below that on service at 0.8069 and with 32,979.03 shortage units. In contrast, XGBoost B materially reduces overflow, transfer volume, procurement, and ending inventory relative to lag-1. The practical interpretation is that lag-1 is a slightly more conservative service policy, while XGBoost B is the more efficient network policy.

7.4 What the benchmark shows

The benchmark confirms two things. First, forecasting quality does matter: weaker forecast files also tend to produce weaker operating outcomes. Second, forecasting quality is not the whole story: the same set of models produces a cost-service trade-off once the optimization layer is introduced. A model can be statistically better and still require interpretation through the operating objective the business actually cares about.

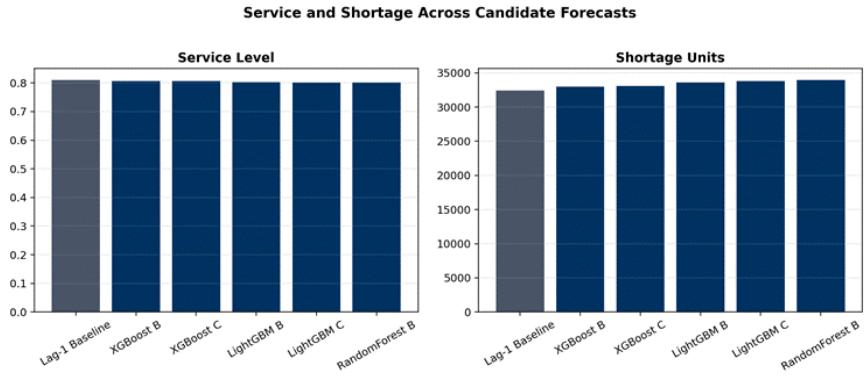

Figure 4. Service and shortage across candidate models. Lag-1 remains the strongest on service, while XGBoost B leads on cost efficiency.

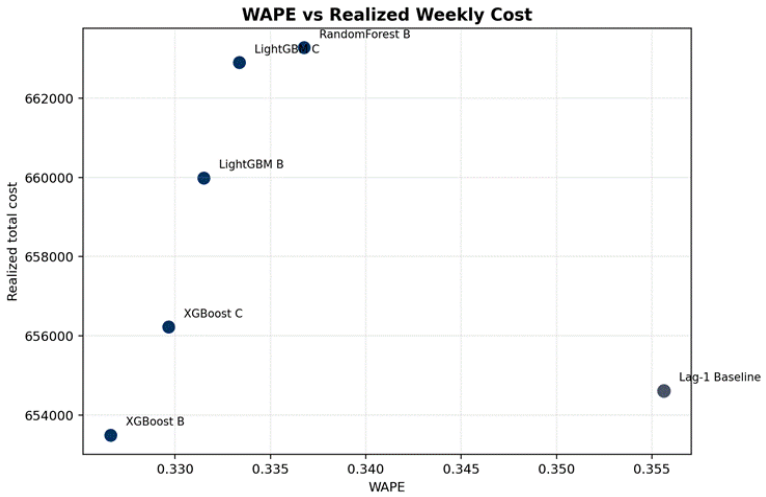

Figure 5. WAPE versus realized weekly cost. The benchmark shows that better fit generally helps, but the cost-service relationship still needs operational interpretation.

7.5 Comparison Between Baseline Model and Optimization Model

A stronger comparison holds the forecast input fixed and asks whether the optimization model actually improves on a simpler operating policy. We therefore evaluate the best forecast file and the lag-1 baseline under two decision rules: the receding-horizon optimization model and the rule-based baseline policy from Section 5.2. This 2x2 design separates the value of better prediction from the value of better decision logic.

Prediction Input	Optimization Model	Baseline Policy
Best Prediction	Cost 653,495 Service 80.69% Shortage 32,979 Transfers 86,619	Cost 1,072,510 Service 36.80% Shortage 107,930 Transfers 297
Lag-1 Baseline	Cost 654,618 Service 81.03% Shortage 32,395 Transfers 88,374	Cost 1,063,028 Service 37.51% Shortage 106,706 Transfers 768

Table 2. Two-by-two comparison of prediction input and decision policy.

The minimum-cost combination is Best Prediction plus the optimization model at 653,495.01. Lag-1 plus the optimization model is a close second at 654,617.88. Both optimized policies are far better than the rule-based baseline policy, whose weekly cost remains above 1,063,027.53 and whose service level stays below 40 percent.

With the best forecast held fixed, the optimization model lowers weekly cost by 419,014.80 relative to the baseline policy, raises service by 43.9 percentage points, and reduces shortage by 74,950.57 units. With lag-1 held fixed, the optimization gain is similarly large: 408,409.65 lower cost and 43.5 percentage points higher service.

Better prediction still matters inside the optimizer, but the effect is narrower than the policy effect. Best Prediction plus optimization saves 1,122.87 versus lag-1 plus optimization, while lag-1 retains a small service advantage of 0.34 percentage points. Under the baseline policy, the better forecast does not win: lag-1 plus the baseline policy is cheaper by 9,482.28. Taken together, this benchmark shows that most of the value is coming from the decision model, not from minor differences in forecast fit by themselves.

7.6 Local Sensitivity of the Optimization Model

To assess whether the optimization results are overly dependent on a narrow parameter calibration, we conducted a local one-factor sensitivity analysis around the best forecast file (XGBoost B) combined with the receding-horizon optimization model. The stress test perturbs forecast demand by +/-10% and perturbs the main cost coefficients from Section 3.2.2 - holding cost h, shortage cost u, and transfer-time penalty k - by +/-20%, while keeping the same March 25-31 test week and the same network constraints.

Scenario	Realized Cost	Delta Cost	Service	Delta Service
Baseline	653,495.01	+0.00%	80.69%	+0.00 pp
Demand -10%	712,717.38	+9.06%	75.55%	-5.14 pp
Demand +10%	611,841.76	-6.37%	84.71%	+4.02 pp

Holding -20%	653,050.40	-0.07%	80.69%	+0.00 pp
Holding +20%	653,978.74	+0.07%	80.69%	-0.00 pp
Shortage -20%	586,456.89	-10.26%	80.53%	-0.16 pp
Shortage +20%	719,773.53	+10.14%	80.69%	+0.00 pp
Transfer -20%	590,650.50	-9.62%	81.53%	+0.85 pp
Transfer +20%	716,853.50	+9.70%	80.53%	-0.15 pp

Table 3. Local sensitivity analysis around the best forecast plus optimization policy.

The largest cost movement in this local sweep comes from Shortage -20%, which changes realized total cost by -10.26% relative to the base case. The largest service movement comes from Demand -10%, which changes realized service by -5.14 percentage points. In particular, demand misspecification is materially more influential than moderate perturbations to h or k, which indicates that forecast level remains the most important driver of downstream decisions in this experiment.

These results should be interpreted cautiously. The analysis varies one input at a time, over a single test week, and does not propagate joint uncertainty in the inventory proxy, capacity layer, or SKU-to-cluster mapping. It is therefore best understood as a targeted robustness check that supports the main findings, not as a comprehensive stochastic sensitivity analysis.

VIII. Implications and Future Work

8.1 Implications

The main implication is that forecasting models for a warehouse network should be benchmarked on decisions, not only on fit. In this project, the scatter between WAPE and realized weekly cost makes that relationship visible: the strongest forecast file also gives the lowest cost, but service and shortage do not move in lockstep with cost. That is a more useful result than a pure forecast leaderboard because it shows how model choice changes the operating policy.

The comparison across several candidate files also gives a more credible benchmark than a single proposed-versus-baseline test. The XGBoost and LightGBM variants cluster fairly closely on forecast performance, but the optimization layer separates them more clearly. That suggests that small differences in forecast fit can still matter once they are translated into procurement and transshipment decisions under capacity constraints.

8.2 Future work

- Better SKU-to-cluster mapping. A stronger bridge between SKU-level attributes and cluster IDs would improve both feature construction and parameter assignment.
- More accurate inventory initialization. A direct warehouse-cluster inventory snapshot would reduce uncertainty in the first-stage state of the optimizer.
- Richer exogenous demand features. Promotions, site traffic, holiday effects, and stronger calendar structure would likely improve the candidate models beyond lag-based behavior.
- Multi-period optimization with explicit lead-time carryover. Extending the daily LP into a fuller multi-period model would make procurement and transshipment decisions more realistic.

- Longer rolling validation. Testing these candidate models over more weeks would provide a more stable view of ranking and robustness.
- Deployment-style daily re-optimization. A production-oriented loop that refreshes forecasts and decisions each day would be the natural next step toward a usable planning tool.

Broader multi-parameter sensitivity analysis. Future work should test joint changes in demand, shortage penalties, holding costs, transfer penalties, and service targets so that the optimizer can be evaluated under a wider range of plausible operating conditions.

IX. Conclusion

This capstone built an integrated workflow for multi-warehouse planning: generate demand forecasts, translate them into procurement and transshipment decisions through a receding-horizon optimization model, and evaluate the resulting policy on realized operations. That framing shifts the project from pure demand estimation to decision support.

The strongest result is that XGBoost B is the best model in the benchmark on both forecast quality and weekly operating cost. At the same time, lag-1 still delivers the highest service level among the tested models, which makes the central trade-off explicit: the model that minimizes cost is not necessarily the model that maximizes service. That distinction is exactly why the problem has to be evaluated at the system level.

Taken together, the benchmark shows that forecasting should be judged through the decisions it drives. The remaining limitations are mostly about data fidelity, inventory state, and time horizon rather than the logic of the pipeline itself. Even in its current form, the project provides a credible foundation for a practical planning system that can be tuned to the business objective it is meant to support.

References

[1] F. Chen, Z. Drezner, J. K. Ryan, and D. Simchi-Levi, “Quantifying the bullwhip effect in a

simple supply chain: The impact of forecasting, lead times, and information,” Management

Science, vol. 46, no. 3, pp. 436–443, 2000.

[2] A. N. Elmachtoub and P. Grigas, “Smart ‘Predict, then Optimize’,” arXiv:1710.08005, 2017.

[3] F. Chen, Z. Drezner, J. K. Ryan, and D. Simchi-Levi, “Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting, lead times, and information,” *Management Science*, vol. 46, no. 3, pp. 436–443, 2000.

[4] S. J. Taylor and B. Letham, “Forecasting at scale,” *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018, doi: 10.1080/00031305.2017.1380080.

[5] S. C. Graves and S. P. Willems, “Optimizing strategic safety stock placement in supply chains,” *Manufacturing & Service Operations Management*, vol. 2, no. 1, pp. 68–83, 2000, doi: 10.1287/msom.2.1.68.23267.

[6] A. N. Elmachtoub and P. Grigas, “Smart ‘Predict, then Optimize’,” *arXiv preprint arXiv:1710.08005*, 2017.

[7] D. Bertsimas, A. Kallus, V. Gupta, and A. Paschalidis, “A survey of contextual optimization methods for decision making under uncertainty,” *arXiv preprint arXiv:2306.10374*, 2023.

[8] E. Liu and Y. Yang, “Quantitative analysis of price elasticity in e-commerce using machine learning: A case study of JD.com transaction-level data,” in *Proc. 2024 5th Int. Conf. Big Data Economy and Information Management (BDEIM)*, 2024, pp. 314–319, doi: 10.1145/3724154.3724206.

[9] M. Shen, C. S. Tang, D. Wu, R. Yuan, and W. Zhou, *“JD.com: Transaction-Level Data for the 2020 MSOM Data Driven Research Challenge,”* Jan. 5, 2020.