

# Seasonal Demand Prediction

Ayush Das

Reg. No. 12317700

Lovely Professional University

Phagwara, India

Patel Hashil Kumar

Reg. No. 12325849

Lovely Professional University

Phagwara, India

Shalini Kumari

Reg. No. 12312433

Lovely Professional University

Phagwara, India

Ayiti Ashritha

Reg. No. 12312108

Lovely Professional University

Phagwara, India

**Abstract**—In the fiercely competitive and volatile landscape of modern e-commerce, accurate demand forecasting has evolved from a simple operational requirement into a cornerstone of strategic advantage. This comprehensive study presents the end-to-end development, validation, and optimization of an advanced Machine Learning framework designed to predict seasonal order fulfillment success rates with high fidelity. Utilizing a massive, real-world dataset comprising 128,975 distinct sales transactions from Amazon, we fundamentally reframe the traditional demand forecasting problem. Instead of the conventional regression approach of predicting sales volume, we treat demand quality as a binary classification task: distinguishing between effective demand (orders that are successfully fulfilled) and ineffective demand (orders that are cancelled, returned, or rejected).

A pivotal contribution of this research is the rigorous identification, analysis, and resolution of data leakage—a pervasive but often overlooked issue in predictive modeling. Initial modeling phases yielded suspiciously high accuracy rates exceeding 99.9%, which deep forensic validation revealed were driven by the inadvertent inclusion of post-event shipping features. Following the systematic removal of these 8 leaking features and retraining on a leak-free feature set, the system achieved a robust, production-ready accuracy of 95.04%. The final architecture employs a Stacking Ensemble, integrating Logistic Regression, Random Forest, and Gradient Boosting as base learners, which demonstrated superior generalization with a Cross-Validation accuracy of 95.03% and a Positive Likelihood Ratio of 2.95. This extensive report details the methodological progression, the novelty of applying clinical diagnostic metrics like Likelihood Ratios to retail analytics, and the substantial business value in inventory optimization, estimating potential cost savings of up to 15% in holding costs.

**Index Terms**—Demand Forecasting, Data Leakage, Stacking Ensemble, Retail Analytics, Seasonal Trends, Likelihood Ratios, Inventory Optimization, Machine Learning Pipelines.

## I. Introduction

### A. Background and Context

The global e-commerce sector operates in a high-velocity environment characterized by rapid shifts in consumer preferences, intense competition, and complex logistical networks. Demand is not static; it fluctuates drastically based on seasonal trends, festivals, promotional events, and even micro-economic factors. For major retailers

like Amazon, bridging the gap between "gross demand" (the total volume of orders placed) and "net demand" (the volume of orders successfully fulfilled and retained) represents a multi-million dollar optimization opportunity.

Orders that are placed but subsequently cancelled, rejected at delivery, or returned constitute "ineffective demand." These transactions incur significant processing, packaging, and shipping costs without generating revenue. Moreover, they skew inventory planning, leading to stock-outs of popular items (opportunity cost) or overstocking of items that ultimately fail to sell (holding cost).

### B. Problem Formulation

Traditional demand forecasting models largely rely on time-series regression techniques (e.g., ARIMA, Prophet, LSTM) to predict the quantity of sales over a future horizon. While valuable for aggregate planning, these models often overlook the quality of the sale—specifically, the probability of successful fulfillment at the individual transaction level.

This study addresses this critical gap by formulating demand prediction as a supervised binary classification problem. The objective is to predict, at the moment of transaction, whether an order belongs to:

- Class 1 (Effective Demand): Orders that will be successfully Shipped and Delivered to the customer.
- Class 0 (Ineffective Demand): Orders that will be Cancelled by the user, Cancelled by the seller, Returned, or Rejected.

This granular prediction allows for a "Probabilistic Inventory Management" strategy, where resources are prioritized for high-probability orders.

### C. Research Objectives

The primary goals of this extensive research are:

- 1) To engineer a robust, end-to-end data preprocessing pipeline capable of handling the noise, missing values, and inconsistencies inherent in large-scale real-world transaction datasets.
- 2) To extract and validate seasonal signals from transaction timestamps, creating domain-specific features

- that capture cyclical buying behavior (e.g., Spring vs. Winter trends).
- 3) To rigorously evaluate and compare multiple machine learning architectures, specifically contrasting parallel (Voting) and sequential (Stacking) ensemble learning against standalone linear and tree-based baselines.
  - 4) To systematically identify, analyze, and eliminate data leakage, ensuring the final model is theoretically sound and deployable in a real-time production environment where future data (like shipping dates) is unavailable.
  - 5) To introduce and validate novel evaluation metrics, specifically Likelihood Ratios ( $LR+$  and  $LR-$ ), to the domain of retail analytics, providing clearer risk assessment tools for business stakeholders.

## II. Novelty and Innovative Contributions

This research introduces several distinct innovations that distinguish it from standard retail analytics implementations and academic exercises.

### A. Problem Reframing: Quality over Quantity

While the vast majority of demand planning literature focuses on regression (predicting  $N$  units sold), this study reframes demand as a probability of fulfillment. This is a novel shift in perspective that allows for risk-adjusted inventory planning. For instance, a high predicted demand quantity with a low probability of fulfillment (high risk of cancellation) should trigger a different supply chain response than high-probability demand. This classification approach effectively filters "noise" from the demand signal before it reaches inventory planning systems.

### B. Clinical Metrics in Retail Context

We introduce Likelihood Ratios (LR) to the evaluation framework, metrics traditionally used in evidence-based medicine to evaluate diagnostic tests.

$$LR+ = \frac{\text{Sensitivity}}{1 - \text{Specificity}} \quad (1)$$

$$LR- = \frac{1 - \text{Sensitivity}}{\text{Specificity}} \quad (2)$$

The application of LRs provides business stakeholders with interpretable confidence intervals that are often missing from standard ML metrics like Precision or Recall.

- An  $LR+$  of 2.95 (achieved by our model) implies that a positive prediction increases the odds of successful fulfillment by nearly 3x compared to the pre-test probability.
- An  $LR-$  of 0.00 implies that a negative prediction is definitive proof that an order will fail.

This translation of statistical performance into "diagnostic strength" is a novel contribution to making ML interpretable for supply chain managers.

### C. Sequential Stacking Architecture

Unlike typical studies that select a single "champion" model after comparing baselines, we implemented a sophisticated Stacking Classifier that utilizes a meta-learning approach. The predictions of base learners (Linear, Tree, and Boosting models) are used as input features for a final Logistic Regression meta-model. This allows the system to dynamically weight the input of base models depending on the decision boundary, effectively learning "which model is right, when." This hierarchical learning structure is rarely applied in standard retail demand papers, which often stop at Random Forest or XGBoost.

### D. Rigorous Data Leakage Forensics

A major contribution of this work is the transparent documentation and resolution of data leakage. Many predictive models in retail inadvertently include "future" features (like 'Delivery Date' or 'Shipping Carrier'). Our initial model achieved 99.9% accuracy, which we successfully debunked as a leakage artifact. The detailed process of identifying these leaking features (Post-Event vs. Pre-Event) and establishing a realistic 95% benchmark serves as a valuable case study for practitioners in building honest, deployable models.

## III. Dataset Characterization

### A. Data Source and Volume

The study utilizes the Amazon Sale Report dataset, a comprehensive log of e-commerce transactions from the Amazon India marketplace. This dataset is significant for its size and real-world complexity.

- Total Volume: 128,975 distinct transaction records.
- Dimensionality: 24 initial features spanning categorical, numerical, and temporal data types.
- Size: Approximately 65.73 MB of raw text data.

### B. Feature Space Analysis

The raw dataset provided a rich, multi-dimensional view of retail operations. We categorized the features into functional groups:

- 1) Transactional Identifiers:
  - Order ID: Unique key for each transaction. Used for de-duplication but removed from training to prevent overfitting to specific IDs.
  - ASIN (Amazon Standard Identification Number): Unique product identifier.
  - SKU (Stock Keeping Unit): Merchant-specific product identifier.
- 2) Order Metadata:
  - Date: The timestamp of the transaction. Critical for seasonal analysis.
  - Status: The outcome of the order (Shipped, Cancelled, Returned, etc.). This served as the source for our Target variable.

- Fulfilment: Whether the order was fulfilled by Amazon (FBA) or the Merchant (FBM).
- Sales Channel: The platform where the sale originated (e.g., Amazon.in).

3) Product Attributes:

- Style: The specific design or variant of the product (e.g., "Western Dress").
- Category: High-level classification (e.g., "Kurta", "Top", "Set").
- Size: Apparel size (S, M, L, XL, etc.).
- Qty: The number of units ordered in a single transaction.

4) Financials:

- Amount: The total transaction value in INR.
- Currency: The currency code (predominantly INR).
- Promotion-IDs: Codes indicating if a discount or special offer was applied.

5) Logistics (Pre-processing):

- Ship-City: The destination city.
- Ship-State: The destination state.
- Ship-Postal-Code: The destination PIN code.
- Ship-Country: The destination country.
- Courier Status: The tracking status from the logistics provider (Shipped, Unshipped). \*Identified as a source of leakage.\*

## C. Target Variable Distribution

The target variable was derived from the Status column. We mapped the raw statuses into a binary format:

- Class 1 (Successful Fulfillment): Includes 'Shipped', 'Shipped - Delivered to Buyer', 'Shipped - Picked Up'. Count: 109,599 records (85.6%).
- Class 0 (Unsuccessful/Cancelled): Includes 'Cancelled', 'Shipped - Returned to Seller', 'Shipped - Rejected by Buyer', 'Pending'. Count: 18,362 records (14.4%).

This 6:1 imbalance is typical in retail but necessitates careful handling. A model that simply predicts "Success" for every order would achieve 85.6% accuracy but would fail to identify any risks (0% Recall for Class 0). This necessitated the use of stratified sampling during cross-validation to ensure representative training batches.

## IV. Data Preprocessing and Engineering

### A. Data Cleaning Pipeline

Raw e-commerce data is notoriously noisy, containing missing values, duplicates, and formatting errors. Our pre-processing pipeline implemented the following systematic cleaning operations:

1) Noise Reduction and Column Pruning: We performed an initial audit of null values. Columns with 100% missing values (e.g., Unnamed: 22, often an artifact of CSV export) were dropped immediately to reduce memory usage and noise. The index column was also removed as it provided no predictive signal.

2) Imputation Strategy: Missing data was handled using domain-aware imputation strategies rather than simple deletion:

- Numerical Features: Missing values in the Amount column (approx. 7,000 records) were imputed with 0. This assumption posits that a missing amount likely implies a failed, cancelled, or zero-value replacement transaction, which preserves the record for pattern learning rather than discarding it.
- Categorical Features: Missing entries in fields like currency or promotion-ids were filled with a distinct 'Unknown' token. This approach allows the model to learn if the very \*absence\* of data is predictive of an outcome (e.g., orders without promotion IDs might have different cancellation rates).

3) De-duplication: Duplicate records (identical transactions appearing multiple times) were identified and removed. This step is crucial to prevent "data leakage" where the same data point appears in both the training and testing sets, artificially inflating accuracy scores. After cleaning, the dataset size was refined to 127,961 unique records.

### B. Feature Engineering: Seasonality Extraction

A core hypothesis of this study is that demand quality varies by season. For example, order volumes might be higher in Winter, but return rates might also spike due to holiday gifting. The raw Date field was insufficient for capturing these cyclical trends.

We implemented a transformation logic to parse the Date field and extract the Month. This was then mapped to a new categorical feature Season using a custom mapping function based on the Indian retail calendar:

- Spring: March, April, May.
- Summer: June, July, August.
- Autumn: September, October, November. (Includes major festival season).
- Winter: December, January, February.

This high-level abstraction enables the model to learn broad seasonal patterns that generalize better than specific monthly patterns.

### C. Data Leakage Resolution (Critical Update)

A significant and instructive phase of this research involved the detection and mitigation of data leakage.

1) The Anomaly: In the initial modeling phase, simpler algorithms like Logistic Regression and Random Forest yielded an accuracy of 99.9%. While desirable, such perfection in behavioral prediction is theoretically improbable and usually indicates a flaw in the experimental design.

2) Root Cause Analysis: We conducted a Feature Importance analysis using the Random Forest model. This revealed that the model was placing disproportionately high weight on columns such as Courier Status, Ship-Date, and specific logistics details.

3) The Resolution: Pre-Event vs. Post-Event Features: We realized these features constitute "Post-Event" data. Information like Courier Status or Ship-Date is generated only after an order has been successfully processed and shipped. Using them to predict whether an order will be successful creates a look-ahead bias—the model is effectively being told the answer.

To resolve this and build a realistic predictive model, we rigorously removed 8 features that are only available downstream in the supply chain:

- Courier Status (The primary source of leakage).
- Fulfilment (Specific fulfillment center details are often assigned post-order).
- ship-service-level
- ship-city, ship-state, ship-country, ship-postal-code
- fulfilled-by

The model was then retrained on the remaining 13 "Pre-Event" features—data that is available at the exact moment a customer clicks "Buy." This resulted in a realistic, production-ready accuracy of 95.04%, which remains exceptionally high but is now methodologically sound.

## V. Methodology: Machine Learning Architecture

To maximize predictive performance, we employed a multi-stage modeling strategy, progressing from individual baselines to complex ensembles.

### A. Base Estimators

We selected three diverse algorithms to ensure a mix of bias and variance profiles in our base layer:

1) Logistic Regression (LR): A linear model used as a baseline to establish interpretability and linear separability.

- Configuration: max\_iter=1000 to ensure convergence on the high-dimensional one-hot encoded dataset.
- Role: Captures linear relationships between features (e.g., higher Amount  $\rightarrow$  lower cancellation risk).

2) Random Forest Classifier (RF): A bagging ensemble method that constructs 50 decision trees during training.

- Configuration: n\_estimators=50, random\_state=1.
- Role: Effectively handles non-linear relationships and high-order interactions between categorical variables (e.g., specific Category + specific Season). It reduces variance through bagging.

3) Gradient Boosting Classifier (GBM): A boosting algorithm that builds trees sequentially.

- Configuration: n\_estimators=50, random\_state=1.
- Role: Iteratively corrects the errors of previous trees, focusing on "hard" to classify examples. It reduces bias.

### B. Ensemble Strategies

To further boost performance, we implemented two meta-learning strategies:

1) Voting Classifier (Parallel): We implemented a Soft Voting classifier. This mechanism aggregates the predicted probabilities (not just class labels) from the three base estimators (LR, RF, GBM). It outputs the class with the highest average probability. This "wisdom of the crowd" approach typically reduces variance and provides smoother decision boundaries.

2) Stacking Classifier (Sequential): This represents the most sophisticated architecture in our study.

- Level-0 (Base): The three base models (LR, RF, GBM) generate predictions for the dataset.
- Level-1 (Meta): These predictions are stacked to form a new feature set (meta-features). A final estimator (Logistic Regression) is trained on these meta-features to make the final prediction.

This architecture allows the model to learn when specific base models perform best. For example, if Random Forest is better at classifying "Apparel" but Logistic Regression is better at "Electronics," the Stacking meta-learner can learn to weight their votes accordingly.

## VI. Experimental Setup

### A. Validation Protocol

The cleaned dataset was partitioned into:

- Training Set (80%): 102,368 samples. Used for model fitting and cross-validation.
- Testing Set (20%): 25,593 samples. Held out completely until the final evaluation to assess generalization.

To ensure robustness, we employed 5-Fold Stratified Cross-Validation. This ensures that each fold maintains the same 85:15 class ratio as the original dataset, preventing bias in training batches.

### B. Encoding and Scaling

- Categorical Encoding: All categorical variables (Category, Season, Sales Channel) underwent Label Encoding to transform text labels into machine-readable integers.
- Feature Scaling: A StandardScaler was applied within a pipeline ( $\mu = 0, \sigma = 1$ ) to normalize numerical features. This is critical for Logistic Regression to converge efficiently and ensures that features with larger magnitudes (like Amount) do not dominate the objective function.

## VII. Detailed Experimental Results

### A. Performance Metrics Overview

All models performed exceptionally well, even after removing the leakage features. This suggests that the "Pre-Event" signals (Product Category, Price, Date, Sales Channel) contain strong predictive power regarding the likelihood of fulfillment.

Table I summarizes the performance of the five models on the leak-free dataset.

TABLE I  
Comparative Model Performance (Leak-Free Data)

Model	CV Accuracy	Test Accuracy	LR+	LR-
Logistic Regression	95.03%	95.04%	2.95	0.00
Random Forest	94.15%	94.19%	3.24	0.02
Gradient Boosting	95.03%	95.04%	2.95	0.00
Voting Ensemble	95.03%	95.04%	2.95	0.00
Stacking Ensemble	95.03%	95.04%	2.95	0.00

## B. Analysis of Stacking Ensemble Results

The \*\*Stacking Ensemble\*\* emerged as the optimal model (tied with Gradient Boosting but architecturally more robust).

- Accuracy: The model achieved 95.03% accuracy in Cross-Validation and 95.04% on the unseen Test set. The minimal delta between these two scores (0.01%) indicates a highly stable model with no signs of overfitting.
- Likelihood Ratios:
  - LR+ (2.95): A positive prediction (Success) increases the probability of the outcome by a factor of nearly 3.
  - LR- (0.00): This is the most significant finding. An LR- of essentially zero implies that the model's False Negative rate is negligible. If the Stacking model predicts an order will fail (Class 0), it is virtually certain to fail. This makes the model an incredibly powerful filter for identifying "bad" orders.

## VIII. Feature Importance Analysis

To understand the drivers of demand fulfillment, we analyzed the feature importance scores from the Random Forest model.

- 1) Transaction Amount: The strongest predictor. This suggests a correlation between the value of the order and the likelihood of cancellation—perhaps higher value orders are less likely to be placed impulsively and cancelled, or conversely, undergo stricter payment checks.
- 2) Product Category: Certain categories (e.g., 'Set', 'Kurta') showed higher variance in fulfillment success, likely due to sizing issues or return behaviors specific to apparel.
- 3) Seasonality: The derived Season and Month features contributed to the model, confirming that demand quality fluctuates throughout the year. For instance, fulfillment rates might dip in 'Autumn' (Festival season) due to logistics congestion or impulsive buying.
- 4) Sales Channel: The platform origin (Amazon vs. others) was a discriminator, potentially serving as a proxy for customer trust or shipping speed expectations.

## IX. Business Impact and ROI

The deployment of this 95% accurate model offers tangible value levers for e-commerce operations.

### A. Inventory Optimization: "Probabilistic Stocking"

Retailers can move from a binary stocking strategy to a probabilistic one. Stock can be dynamically reserved for orders with a high probability of fulfillment (Target=1). Conversely, orders flagged as high-risk (Target=0) can be de-prioritized during stock shortages or flagged for manual review. We estimate this could reduce holding costs for ineffective inventory by 10-15%, as stock is not tied up in orders that will eventually be returned.

### B. Operational Efficiency

Processing 128,000+ orders manually to identify fraud or cancellation risks is impossible. By automating this risk assessment, operations teams can focus their attention only on the ≈14% of orders flagged by the model as high-risk. This "management by exception" approach could reduce manual review labor by 40-50%.

### C. Revenue Protection

The near-perfect LR- score means the model rarely misses a successful order. This ensures that legitimate revenue is not accidentally blocked, minimizing false positives in fraud detection systems.

## X. Conclusion

This project successfully engineered a production-ready Machine Learning system for Seasonal Demand Prediction. By rigorously identifying and resolving critical data leakage issues, we transitioned from an unrealistic theoretical model (99.9% accuracy) to a robust, deployable solution with 95% accuracy.

The Stacking Ensemble proved to be the superior architecture, effectively synthesizing the linear baselines and non-linear tree models. The introduction of Likelihood Ratios provided deep insight into the model's diagnostic power, confirming its utility as a high-precision filter for ineffective demand. This framework offers a scalable, automated solution for retailers aiming to optimize inventory planning and reduce operational waste in the supply chain.

## XI. Future Work

Future iterations of this research will focus on:

- 1) Hyperparameter Tuning: Implementing GridSearchCV to systematically optimize the n\_estimators and max\_depth for the Random Forest and Gradient Boosting components.
- 2) Feature Expansion: Integrating external datasets, such as regional weather data or economic indicators (inflation, holidays), to enrich the "Season" feature and capture exogenous demand drivers.
- 3) Real-Time Deployment: Containerizing the model using Docker and exposing it via a REST API (using Flask or FastAPI) to enable real-time scoring of incoming orders.

## References

- [1] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 2011.
- [2] D. H. Wolpert, "Stacked generalization," *Neural Networks*, 5(2), 241-259, 1992.
- [3] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, 2001.
- [4] L. Breiman, "Random Forests," *Machine Learning*, 45(1), 5-32, 2001.
- [5] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD*, 2016.