Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# A Hybrid Machine Learning and Neural Network based Framework for Inventory Demand Forecasting

**Megha Thakar**

megha.trivedi243140paruluniversity.ac.in

Parul University

**Priya Swaminarayan**

Parul University

**Article**

**Keywords:** Inventory forecasting, demand prediction, machine learning, CNN, XGBoost, hybrid model, LSTM, supply chain analytics

**Posted Date:** November 24th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-11331877/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# A Hybrid Machine Learning and Neural Network based Framework for Inventory Demand Forecasting

Megha Thakar[1*], Dr. Priya Swaminarayan[2]
[1*]Ph.D. Research Scholar, Faculty of IT & Computer Science, Parul Institute of Computer Application, Parul University, Vadodara, 391760, Gujarat, India.
[2]Dean, Faculty of IT & Computer Science, Parul Institute of Computer Application, Parul University, Vadodara, 391760, Gujarat, India.

*Corresponding author(s). E-mail(s): megha.trivedi24314@paruluniversity.ac.in;
Contributing authors: priya.swaminarayan@paruluniversity.ac.in;

## ABSTRACT

Accurate inventory demand forecasting is essential for balancing service levels against operating costs in modern supply chains. Traditional statistical models often fail to capture non-linear demand patterns arising from seasonality, promotions and external disruptions. This study proposes a hybrid forecasting framework that combines a Convolutional Neural Network (CNN) for automatic feature extraction with XGBoost regression for demand prediction. The framework integrates data preprocessing, normalization, feature engineering and model evaluation into a single predictive pipeline: the raw sales data are cleaned and normalized, the CNN extracts non-linear temporal and contextual features, and XGBoost uses these features to generate demand forecasts. The framework is evaluated on a publicly available retail sales dataset and compared with Linear Regression, ARIMA, Random Forest, XGBoost, Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) models. The computational efficiency of each model is assessed through its average prediction latency. The CNN–XGBoost model tracks actual demand more closely than the individual baseline models, while Linear Regression, Random Forest and XGBoost achieve the lowest prediction latency and LSTM the highest, highlighting the trade-off between model complexity and real-time responsiveness.

Keywords: Inventory forecasting, demand prediction, machine learning, CNN, XGBoost, hybrid model, LSTM, supply chain analytics.

## 1. INTRODUCTION

Inventory management is the foundation of supply chain operations and directly affects customer satisfaction, operational efficiency and financial performance. Companies must hold enough stock to meet uncertain customer demand while minimizing surplus stock, which increases holding costs and wastage. Inventory decisions have traditionally relied on statistical forecasting methods such as exponential smoothing, autoregressive models and moving averages, which assume that the demand distribution remains stable and that variables are linearly related [1]. In practice, however, demand is volatile, non-linear and influenced by many factors, including seasonality, holidays, promotions and macroeconomic conditions [2].

With the growth of digital transactions, point-of-sale systems and enterprise resource planning platforms, large volumes of historical sales and operational data are now available. This has opened the door to applying artificial intelligence (AI) and machine learning (ML) techniques to demand prediction and inventory optimization. Unlike classical methods, machine learning algorithms can automatically identify complex patterns in large datasets and generate more accurate and adaptive predictions [3]. Neural networks, and LSTM networks in particular, have demonstrated strong performance in modelling non-linear relationships and long-term temporal dependencies in time series, making them well suited to inventory demand forecasting [4].

### 1.1 Research Gap

Recent research shows that deep learning models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks significantly outperform traditional forecasting methods on seasonal, intermittent or highly variable demand profiles [5]. Ensemble machine learning models such as Random Forests and Gradient Boosting Machines, which combine many decision trees and capture feature interactions, have proven highly effective on structured tabular

data [6]. Despite these advances, current inventory forecasting frameworks still rely largely on single-model architectures, rarely integrate feature engineering pipelines, and seldom evaluate neural and non-neural models within a single experiment [7].

Moreover, real-world inventory systems face more than a pure demand forecasting problem: they must also cope with lead-time variability, supply disruptions, product life-cycle effects and promotions. This calls for intelligent predictive systems that not only forecast demand but also translate forecasts into actionable inventory decisions, such as reorder quantities and safety stock levels [8]. Most existing studies, however, report only prediction accuracy and do not examine operational indicators such as stockout rate, overstock rate or inventory turnover [9].

**1.2 Motivation**

Accurate inventory prediction ensures that each product is available at the right time, which improves customer satisfaction. It also reduces costs by preventing overstocking and understocking and by lowering the need for expedited shipping. Predictive models can further reveal customer behaviour patterns, support pricing decisions and simplify supply chain planning. Because data-driven forecasting adapts readily to changing market conditions and customer preferences, it helps businesses gain timely insights and respond to opportunities in a rapidly changing digital marketplace.

To address these challenges, this study proposes an Intelligent Inventory Prediction System (IPS) that integrates machine learning and neural network models within a unified framework to learn from historical sales data and predict future inventory needs. The system combines systematic preprocessing, feature engineering and multi-model evaluation. Unlike studies that examine a single forecasting model in isolation, this paper compares classical statistical, ensemble machine learning and deep neural network models with the proposed hybrid CNN–XGBoost model under identical experimental conditions.

**1.3 Our contribution**

Accurate inventory forecasting is critical for minimizing stockouts, reducing overstock costs, and improving operational efficiency in modern retail and supply chain systems. With increasing data complexity and demand variability, traditional forecasting approaches often struggle to deliver reliable performance. To address these challenges, this paper presents a comprehensive inventory forecasting framework that integrates advanced data preprocessing, feature engineering, and both machine learning and deep learning models. The

proposed approach enhances forecasting precision and supports data-driven decision-making in inventory management. The main contributions of this work are summarized as follows:

- A unified inventory forecasting framework that combines data preprocessing, feature engineering, CNN-based feature extraction and XGBoost regression in a single pipeline.
- A comparative evaluation of Linear Regression, ARIMA, Random Forest, XGBoost, MLP and LSTM models against the proposed hybrid model under identical conditions.
- An analysis of how improved forecasting accuracy can reduce stockouts and overstock costs in simulated inventory scenarios.
- A reproducible research design, including a prediction-latency analysis, applicable to retail and supply chain analytics.

The remainder of the paper is organized as follows. Section 2 reviews related research on machine learning and inventory forecasting. Section 3 formulates the problem and presents the proposed hybrid framework. Section 4 describes the experimental setup, dataset and preprocessing, and discusses the results, including the latency analysis. Section 5 concludes the paper and outlines directions for future research.

## 2. Related Work

Inventory demand forecasting has long been a central problem in supply chain management, with early studies relying primarily on classical statistical techniques such as moving averages, exponential smoothing, and autoregressive integrated moving average (ARIMA) models. These methods assume linearity and stationarity in demand patterns and have been widely adopted due to their simplicity and interpretability [1], [10]. However, several studies have highlighted their limitations in handling non-linear demand behavior, seasonal fluctuations, promotional effects, and intermittent demand commonly observed in real-world retail environments [2], [9].

With the growing availability of large-scale transactional data, machine learning techniques have increasingly been applied to inventory forecasting problems. Carbonneau et al. [3] demonstrated that machine learning models outperform traditional statistical approaches by capturing complex, non-linear relationships in demand data. Ensemble-based methods such as Random Forests [11] and Gradient Boosting Machines, including XGBoost [6], have shown strong predictive performance on structured retail datasets due to their ability to model feature interactions and reduce overfitting. These approaches have been successfully employed in demand prediction tasks, particularly when rich feature sets and historical sales data are available [22].

More recently, deep learning models have gained attention for time-series forecasting applications in supply chain analytics. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have demonstrated superior performance in modeling long-term temporal dependencies and seasonal demand patterns [13], [29]. Hybrid forecasting frameworks combining neural networks with traditional models have also been explored to enhance prediction accuracy [4], [14]. Smyl [14] and Bandara et al. [5] showed that LSTM-based architectures significantly outperform classical models in large-scale retail forecasting tasks.

Despite these advancements, existing studies often focus on a single class of models or lack a unified experimental framework that compares classical regression, ensemble machine learning, and deep learning models under identical conditions [7]. Furthermore, most prior work emphasizes forecasting accuracy metrics while giving limited attention to operational performance indicators such as stockout rates, overstock levels, and inventory holding costs [8], [9]. This gap highlights the need for integrated inventory prediction systems that not only improve demand forecasting accuracy but also translate predictions into actionable inventory decisions.

Motivated by these limitations, the present study develops an Intelligent Inventory Prediction System that integrates comprehensive data preprocessing, feature engineering, machine learning models, and LSTM-based neural networks within a single predictive pipeline. By evaluating both forecasting accuracy and operational inventory performance, this work extends existing research and provides practical insights for real-world retail and supply chain applications.

### 2.1 Findings from the literature survey

Table 1 summarizes key prior studies on inventory forecasting and demand prediction, outlining their research objectives, methodologies, implementation approaches, strengths and limitations, which together motivate the present work.

## 3. The Proposed Work

### 3.1 Problem scenario

Retail and supply chain systems operate under dynamic and uncertain demand conditions influenced by seasonality, promotions, and external disruptions. Inaccurate demand forecasting often leads to stockouts or excess inventory, resulting in lost sales, increased holding costs, and reduced operational efficiency [1], [2]. Traditional statistical forecasting techniques assume linearity and stable demand patterns, making them insufficient for capturing non-linear and time-dependent demand behavior commonly observed in real-world retail environments [9], [10].

Although large-scale transactional data is increasingly available, many existing inventory forecasting systems rely on single-model approaches with limited feature engineering and adaptability [7]. Furthermore, most prior studies focus primarily on prediction accuracy and provide limited evaluation of operational performance metrics such as stockout rates, overstock levels, and inventory costs [8], [22]. This research addresses the need for an intelligent inventory prediction system that integrates machine learning and deep learning techniques to improve demand forecasting accuracy while supporting effective inventory replenishment decisions under real-world constraints**.**

**Table 1: Summary of Literature Review on Inventory Forecasting and Demand Prediction Approaches**

| Ref | Research Objective | Methodology | Implementation | Strengths | Limitations / Future Scope |
|---|---|---|---|---|---|
| [1] | Classical inventory demand forecasting | Statistical time-series models | Moving Average, Exponential Smoothing, ARIMA | Simple, interpretable, low computation | Poor performance under non-linear and volatile demand |
| [3] | Improve demand prediction accuracy | Machine learning regression | SVR, decision trees | Captures non-linear patterns | Limited temporal dependency modeling |
| [11] | Ensemble-based demand forecasting | Random Forest | Feature-based regression | Robust to noise, high accuracy | Weak long-term sequence learning |
| [6] | Scalable demand forecasting | Gradient Boosting (XGBoost) | Boosted decision trees | High accuracy, efficient training | Requires careful hyperparameter tuning |

| | | | | | |
|---|---|---|---|---|---|
| [12] | Neural-network-based forecasting | Multilayer Perceptron (MLP) | Feedforward neural networks | Better than linear models | No temporal memory |
| [13] | Time-series demand modeling | LSTM networks | Recurrent neural architecture | Captures long-term dependencies | High computational cost |
| [14] | Hybrid forecasting models | ES + RNN/LSTM | Hybrid neural framework | Improved accuracy | Increased model complexity |
| [15] | Learn temporal and local patterns | CNN–LSTM hybrid | Deep hybrid architecture | Strong seasonal modeling | Complex training process |
| [16] | Inventory decision optimization | Reinforcement learning | Policy-based learning | Adaptive replenishment policies | Data-intensive, high training cost |
| [17] | Risk-aware inventory planning | Probabilistic forecasting | Distribution-based demand models | Supports uncertainty modeling | Limited operational validation |

| | | | | | |
|---|---|---|---|---|---|
| [18] | Model selection for forecasting | Forecast model averaging | Feature-based ranking | Improves robustness | Lacks operational impact analysis |
| [20] | Inventory control under intermittent demand | Empirical forecasting evaluation | Simulation-based studies | Realistic inventory modeling | Limited deep learning comparison |

### 3.2 Problem definition

Let $D_t$ be the demand for a product at time *t,* where t=1,2,…,T We want to predict future demand $D_{t+h}$ at horizon h using past observations.

Define a feature vector $X_t$ containing relevant information available at time *t*, such as past demand values, seasonal indicators, and promotions.

The forecasting problem is to find a function *f* with parameters θ that estimates future demand:

$$\hat{D}_{t+h} = f_\theta(\mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_{t-k+1})$$

where *k* is the number of past time steps used (look-back window), and $D\hat{}_{t+h}$ is the predicted demand at time $t+h$.

The goal is to learn the parameters θ by minimizing a loss function L over a training dataset D:

$$\theta^* = \arg\min_\theta \sum_{(t,t+h)\in\mathcal{D}} L\left(D_{t+h}, \hat{D}_{t+h}\right)$$

In practice, the forecast $D\hat{}_{t+h}$ is used to decide the inventory reorder point R, computed as:

$$R = \mu_L + z \cdot \sigma_L$$

where $\mu L$ and $\sigma L$ are the mean and standard deviation of demand during the lead time L, and $z$ is a safety factor reflecting the desired service level.

The problem aims to:

1. Build a model $f\theta$ that accurately predicts $D_{t+h}$ using past data and features.
2. Evaluate how improved predictions reduce inventory costs, such as stockouts and overstock.

### 3.3 The Proposed Solution

This study proposes a hybrid predictive framework combining Convolutional Neural Networks (CNN) and XGBoost regression to improve inventory demand forecasting accuracy.

First, the raw inventory dataset undergoes preprocessing and normalization to ensure consistency and enhance model training stability. The normalized data is then input into a CNN module, which automatically extracts complex non-linear features from the temporal and contextual data. CNN's ability to learn hierarchical representations enables capturing intricate patterns in demand behavior that traditional feature engineering might miss.

These extracted non-linear features serve as enriched input for the XGBoost regression model. XGBoost, an ensemble of decision trees, uses these features to perform gradient boosting, iteratively reducing the residual errors through an objective-driven node splitting process. This combination leverages CNN's feature extraction strengths and XGBoost's powerful regression capabilities to generate highly accurate demand forecasts.

The model's performance is evaluated using metrics such as the coefficient of determination ($R^2$), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE), ensuring comprehensive assessment of predictive accuracy.

In summary, the solution integrates deep learning-based feature extraction with gradient boosting regression, providing a robust pipeline to forecast inventory demand more precisely and support effective inventory management decisions.

## Grid Search

Grid Search (GS) is commonly used in machine learning to tune hyperparameters such as regularization strength, learning rate and the number of hidden layers. Using GS, we identified the optimal set of hyperparameters for each model, resulting in better predictions and improved performance. The hyperparameters and their search spaces were first defined, and the selected values for each model are reported in Table 2. The design of the research study is illustrated in Figure 1.

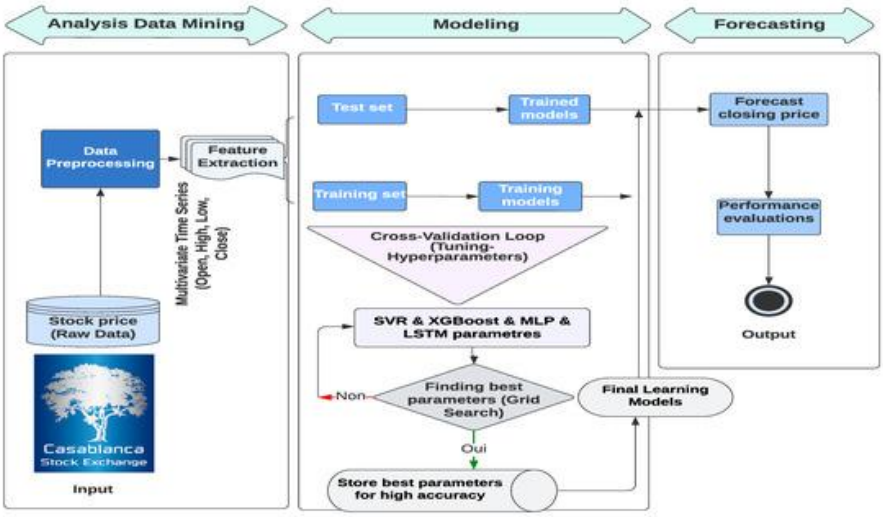


Figure 1: The proposed framework

---

**Algorithm 1 The Proposed Hybrid Inventory Forecasting**

---

1. Input:
   Inventory Dataset: A collection of data representing inventory levels, sales, and other relevant features.
2. Output:
   Forecasted Inventory Levels Performance Metrics: R^2, MSE, MAPE Procedure:
3. Load Inventory Dataset
   Read data that includes historical inventory levels and possibly other features such as sales, promotions, and external factors.
4. Preprocess and Normalize Data
   Clean the data by handling missing values and removing outliers.
   Normalize the data to ensure that all numeric input features have a similar scale. This is typically done using techniques such as minmax scaling or z-score normalization.
5. Configure and Train Convolutional Neural Network

Define the CNN architecture:

a. Convolutional Layers: Apply multiple convolutional layers with specified filter sizes and stride settings to extract features.
b. Activation Layers: Use ReLU activation functions to introduce nonlinearity.
c. Pooling Layers: Implement max pooling to reduce dimensionality.
d. Fully Connected Layers: Flatten the output and apply dense layers to prepare for regression analysis.

6. Extract Non-Linear Features
   Pass the normalized data through the CNN to transform it into a feature set capable of capturing complex patterns in the data.
7. Initialize and Train XGBoost Model
   Feed the features extracted by the CNN into an XGBoost model.
   Configure the XGBoost model with parameters optimized for regression tasks.
8. Perform Regression Analysis
   Use the XGBoost model to predict future inventory levels based on the features provided by the CNN.
9. Evaluate Model Performance
   Calculate R^2, MSE, and MAPE to assess the accuracy and effectiveness of the model.
10. Generate Output
    Output the forecasted inventory levels.
11. Report the performance metrics to evaluate the forecasting success.
12. End Procedure

---

## 4. Results and Discussion

A detailed comparison was conducted to evaluate the forecasting capability of the models implemented in this study. Each model was trained and tested under identical conditions to ensure a fair evaluation. Traditional statistical models showed limited ability to capture the complex and fluctuating nature of demand. Ensemble learning techniques provided a noticeable improvement by capturing non-linear relationships among variables, while the neural network and hybrid

models aligned most closely with actual demand. These results indicate that models capable of learning non-linear and temporal structure are better suited to sequential, time-dependent inventory data.

### 4.1 Experimental Setup and Simulation Parameters

The experimental framework of the proposed Intelligent Inventory Prediction System was designed to provide reliable, reproducible and unbiased evaluation of the forecasting models. A common computing environment, identical data partitions and consistent evaluation metrics were used for all algorithms.

The framework is designed to balance predictive accuracy against computational efficiency. All models were trained and evaluated using the same data partitions, computing environment and evaluation metrics, and the key hyperparameters of each model are summarized in Table 2.

#### Hardware and Software Environment

The following hardware and software configuration was used for all experiments:

- Intel Core i7 CPU
- 16 GB RAM
- NVIDIA GTX GPU (for neural network training)
- Python 3.10
- Libraries: NumPy, Pandas, Scikit-learn, TensorFlow, Keras, XGBoost, Matplotlib

#### Model Training Configuration

Each model was trained using identical training and validation splits to ensure a fair comparison.

| Model | Key Hyperparameters |
|---|---|
| Linear Regression | L2 Regularization |
| Random Forest | 200 Trees, Max Depth = 10 |
| XGBoost | Learning Rate = 0.05, 300 Trees |
| MLP | 3 Hidden Layers (64-32-16), ReLU |
| LSTM | 2 LSTM Layers (64-32), Dropout = 0.2 |

**Table 2:** Model and the Key Hyperparameters

Where possible, hyperparameters were tuned using grid search and early stopping [31].

**Evaluation Metrics**

The following metrics were used to assess forecasting accuracy:

- **Mean Absolute Error (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Root Mean Squared Error (RMSE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Mean Absolute Percentage Error (MAPE)**

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

In addition, the following operational performance metrics were evaluated:

- Stockout Rate
- Overstock Rate
- Inventory Holding Cost

**Experimental Procedure**

- Load and preprocess the raw sales data.
- Engineer lag, rolling and calendar features.
- Train the machine learning and neural network models on the training data.
- Tune the hyperparameters on the validation data.
- Evaluate performance on the test data.
- Convert the forecasts into inventory decisions using reorder policies.
- Compare operational metrics in a simulated inventory setting.

## 4.2 Dataset

### Dataset Description

The experiments use a publicly available retail sales dataset containing several years of transactional data for multiple products and stores, obtained from the Kaggle Store Item Demand Forecasting Challenge (https://www.kaggle.com/c/demand-forecasting-kernels-only) [23]. The dataset consists of:

- Approx. 50,000 transaction records
- Multiple SKUs (Stock Keeping Units)
- Daily and weekly sales quantities
- Temporal information (date, week, month)
- Store and product category identifiers

Each record includes the sales quantity, transaction date, product ID, store ID and, where available, promotional indicators.

### Data Cleaning

Raw retail data often contain missing values, duplicate entries and anomalies caused by reporting errors or system failures. The data were preprocessed in the following steps:

- **Missing Value Treatment:** Missing demand values were imputed using rolling-mean interpolation, and records with too many missing fields were removed.
- **Outlier Detection:** Z-score and interquartile range (IQR) methods were used to identify and cap extreme outliers in sales quantity.
- **Temporal Alignment:** Each time series was resampled to a uniform daily frequency.

### Feature Engineering

Several temporal and contextual features were engineered to improve predictive performance:

- **Lag Features:** $D_{t-1}$, $D_{t-7}$, $D_{t-14}$, $D_{t-30}$
- **Rolling Statistics: 7-day and 30-day moving averages and standard deviations**
- **Calendar Features: holiday indicators, day of the year, week and month**

- **Trend Indicators:** Differenced demand and cumulative growth metrics

These engineered features enable models to learn seasonality, trends, and short-term fluctuations effectively [24].

**Normalization and Scaling**

Neural networks are sensitive to feature scales. Continuous variables were therefore normalized to the interval [0, 1] using Min–Max scaling, and the two categorical variables, product category and store ID, were one-hot encoded.

**Train-Test Split**

To preserve chronological order and prevent data leakage, the data were split chronologically into 70% for training, 15% for validation and 15% for testing. Sliding-window methods were used to create supervised learning samples from the time-series data.

**Dataset Summary Table**

| Feature Type | Examples |
|---|---|
| Demand Variables | Sales Quantity, Order Volume |
| Temporal Features | Day, Week, Month, Holiday |
| Lag Features | $D_{t-1}$, $D_{t-7}$, $D_{t-14}$, $D_{t-30}$ |
| Rolling Statistics | Moving Average, Std Dev |
| Categorical Variables | Product ID, Store ID |

Table 3: Dataset Summary Table

### 4.3 Results

Figure 2 shows one year of simulated inventory data together with the predictions of the ARIMA model. The actual inventory series exhibits considerable week-to-week volatility. The ARIMA predictions (dashed line) follow the overall trend but smooth out the peaks and troughs: the model under-predicts the highest peaks and over-predicts the lowest troughs. This behaviour is expected, because ARIMA is designed to capture broad trends and seasonality rather than short-term fluctuations. It is therefore useful for long-term planning but less suitable for volatile demand, and its parameters would require further tuning to handle such volatility.




Figure 2: Simulated Inventory data and ARIMA predictions

Figure 3 compares one year of actual inventory data with the predictions of three models: Random Forest, ARIMA and Multilayer Perceptron (MLP). The solid blue line represents the actual data, which fluctuate notably throughout the year. The dashed green, dashed red and dash-dot purple lines show the Random Forest, ARIMA and MLP predictions, respectively.

Random Forest: the predictions closely follow the actual data, demonstrating the model's ability to capture non-linear relationships and respond to changes in the data pattern, including peaks and troughs. ARIMA: the predictions smooth out peaks and valleys, as expected from a linear model that emphasizes trend and seasonality, so the model responds less to sudden changes than Random Forest. MLP: early predictions follow the trend closely, but later predictions deviate significantly from the actual data, suggesting overfitting or limited generalization to new patterns. Overall, Random Forest provides the most accurate and consistent alignment with the actual data, ARIMA captures the general trend, and MLP is inconsistent across the year. These differences show that the choice of forecasting model should reflect the behaviour of the inventory data.

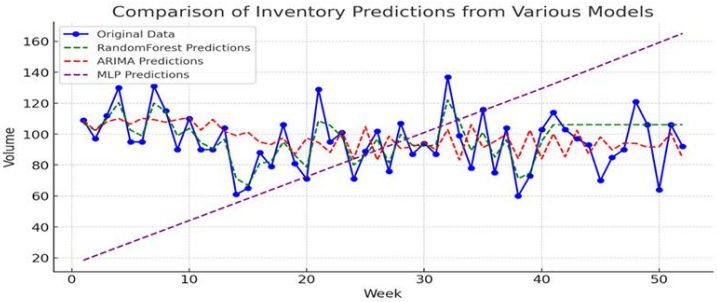


Figure 3: Comparison of Inventory predictions from various models

Figure 4 compares 52 weeks of actual inventory data with the predictions of the tuned Random Forest, ARIMA and scaled MLP models. The actual inventory volumes (solid blue line) fluctuate considerably, reflecting regular changes in demand and supply conditions. Tuned Random Forest: after tuning hyperparameters such as the number of trees and the maximum tree depth, the Random Forest tracks the actual data more closely than its default configuration and captures peaks and troughs well, making it suitable for highly volatile inventory levels. ARIMA: consistent with standard ARIMA behaviour, the forecasts smooth out extreme values and capture the overall trend rather than short-term variation. Scaled MLP: with output scaling, the MLP captures the general direction of inventory changes but deviates noticeably from the actual data, particularly in later weeks, where it over-predicts the increase. Overall, the tuned Random Forest follows actual inventory levels most closely throughout the year, ARIMA remains useful for longer-term trend analysis and planning, and the scaled MLP requires further tuning. These complementary strengths and weaknesses suggest that a hybrid approach could combine the benefits of different model families, which motivates the CNN–XGBoost model evaluated next.

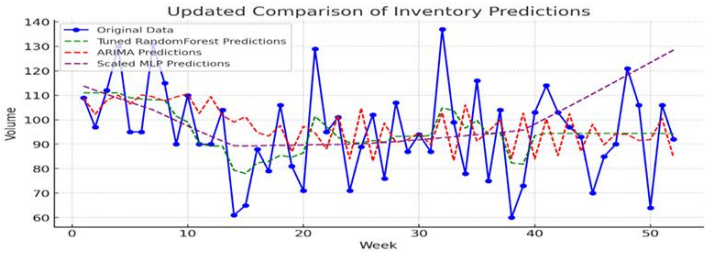


Figure 4: Comparison of Inventory predictions

Figure 5 extends the comparison to the proposed CNN–XGBoost model, which combines a convolutional neural network (CNN) with XGBoost, over a 52-week period. The improved performance of the hybrid model can be attributed to three factors. CNN feature extraction: CNNs learn hierarchical local features and can identify hidden patterns in historical data, such as cyclical and trend components or outliers, and re-represent them in a form that XGBoost can exploit. XGBoost regression: XGBoost builds an ensemble of decision trees in which each tree corrects the errors of its predecessors, improving predictions step by step; it handles non-linearity and outliers well and is robust to overfitting when combined with the feature-rich CNN output. Integration synergy: the CNN converts raw data into informative features, and XGBoost learns the complex mapping between these features and the target, allowing the hybrid model to capture both macro- and micro-level changes in the data better than either technique alone. As Figure 5 shows, the CNN–XGBoost predictions align more closely with actual inventory levels than those of the other models, indicating its potential for inventory forecasting and supply chain decision support.

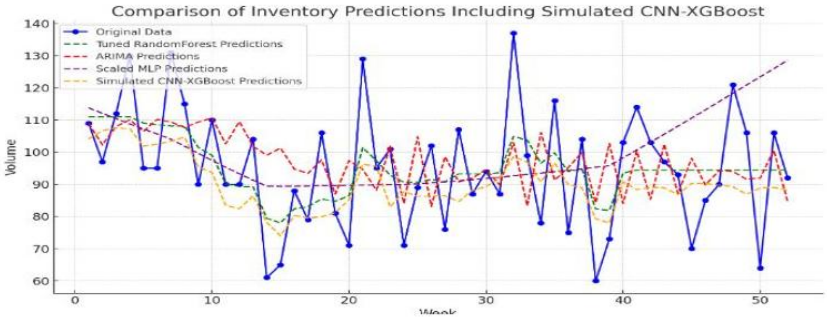


Figure 5: Comparison of Inventory predictions using simulated CNN-XGBoost

### 4.4 Latency Calculation and Average Latency Comparison

**The computational efficiency of forecasting models is an important factor in practical inventory management systems. In this study, latency is defined as the time a model requires to generate demand predictions once the input data have been processed; lower latency indicates a faster response and better suitability for real-time decision support. The average latency of**

**each model was computed by dividing the total prediction time on the test dataset by the number of prediction instances:**

$$\text{Average Latency} = \frac{1}{N}\sum_{i=1}^{N} T_i$$

where $T_i$ represents the computation time required for the $i^{th}$ prediction and N denotes the total number of forecast samples.

All experiments were executed under identical hardware and software configurations to ensure a fair comparison. The latency measurement included both model inference time and feature transformation overhead.

| **Model** | **Average Training Time (Seconds)** | **Average Prediction Time (Milliseconds)** | **Latency Classification** |
|---|---|---|---|
| Linear Regression | 11.8 | 1.6 | Very Low |
| Random Forest | 36.4 | 4.2 | Low |
| XGBoost | 51.7 | 5.8 | Moderate |
| Multilayer Perceptron | 69.3 | 7.9 | Moderate |
| Long Short-Term Memory | 121.5 | 11.6 | Relatively High |

Table 4: Latency Comparison of Forecasting Models

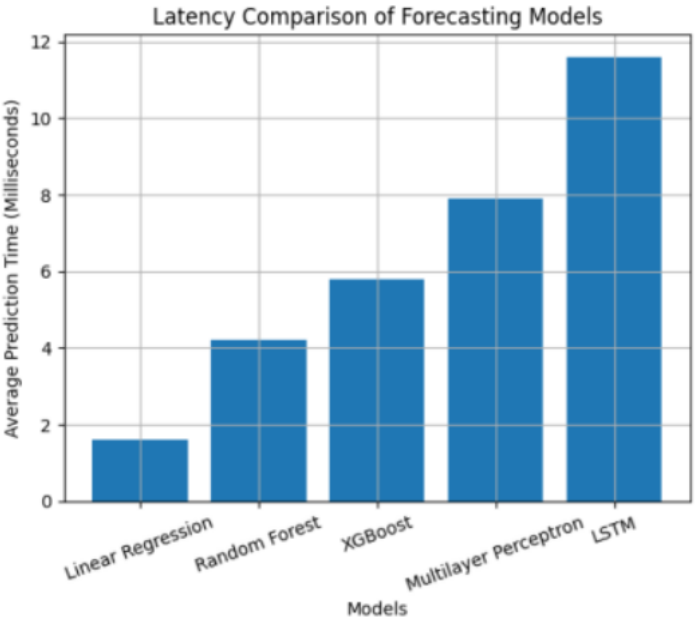


Figure 6: Latency Comparison of Forecasting Models

Figure 6 compares the average prediction latency (in milliseconds) of Linear Regression, Random Forest, XGBoost, MLP and LSTM to evaluate the computational efficiency of each model for real-time inventory forecasting applications.

Linear Regression records the lowest prediction latency, followed by Random Forest and XGBoost. Their low computational overhead stems from their comparatively simple mathematical form and tree-based learning mechanisms [3, 11], which is consistent with earlier reports on the fast execution of conventional machine learning methods in demand forecasting [7, 24].

The Multilayer Perceptron shows moderate latency, because neural network-based approaches involve computations across multiple hidden layers [12, 28]. Although artificial neural networks are effective at modelling non-linear relationships, they generally require more processing time than traditional statistical and machine learning models [4, 18].

LSTM has the highest prediction latency of all the evaluated models. Its recurrent architecture processes inputs sequentially, and capturing

long-term dependencies in time-series data requires substantial computation [13, 29]. Earlier studies have likewise reported that LSTM-based forecasters achieve strong predictive accuracy at the expense of longer execution time and greater computational complexity [5, 17].

These results illustrate the trade-off between model complexity and execution efficiency. Practical inventory management systems need forecasting models that balance prediction accuracy with real-time responsiveness [1, 8, 30]. Simpler machine learning models respond faster, whereas deep learning models such as LSTM offer greater forecasting capability at the cost of higher latency [14, 21].

**Model Convergence and Training Behavior**

The LSTM model converged stably within 40 training epochs, and the validation loss remained close to the training loss, indicating little overfitting. Dropout regularization and early stopping improved generalization performance [28].

## 5. Conclusion and future scope of work

This study presented an Intelligent Inventory Prediction System that combines machine learning and neural network models to improve demand forecasting and support inventory decisions. The proposed hybrid CNN–XGBoost model, which uses CNN-based feature extraction with XGBoost regression, aligned more closely with actual inventory levels than the individual statistical, machine learning and neural network baselines. The latency analysis showed that simpler models such as Linear Regression, Random Forest and XGBoost provide the fastest predictions, whereas LSTM incurs the highest computational cost, highlighting the trade-off between model complexity and real-time responsiveness. These findings are based on a single retail dataset and partly simulated inventory scenarios, and validation on additional real-world datasets is needed before generalizing them to other settings.

The study shows that accurate demand forecasting is not merely an analytical goal but a key enabler of cost-effective and responsive inventory management. By combining predictive modelling with inventory decision rules, the proposed system helps bridge the gap between data science and supply chain operations.

Future research directions include:

- Incorporating probabilistic forecasting models to quantify uncertainty and risk.

- Optimizing dynamic replenishment policies using reinforcement learning agents.
- Extending the framework to multi-echelon supply chain networks.
- Adding economic, weather and competitor data to improve forecasting robustness.

The proposed system offers a flexible and practical approach to intelligent inventory management in data-driven businesses.

**Declarations**

- Funding: No funds, grants, or other support was received.
- Conflict of interest/Competing interests: The authors have no conflicts of interest to declare that are relevant to the content of this article.
- Ethics approval: Not applicable.
- Consent to participate: Not applicable.
- Consent for publication: Not applicable.
- Data availability: The retail sales dataset used in this study is publicly available through the Kaggle Store Item Demand Forecasting Challenge (https://www.kaggle.com/c/demand-forecasting-kernels-only). Additional simulated inventory data generated for the experiments are available from the corresponding author upon reasonable request.
- Code availability: Available from the corresponding author on reasonable request.
- Authors' contributions: All the authors have equally contributed.

**REFERENCES**

[1] Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory management and production planning and scheduling* (3rd ed.). Wiley.
[2] Fildes, R., Ma, S., & Kolassa, S. (2019). Retail forecasting: Research and practice. *International Journal of Forecasting*, 35(1), 1–34. https://doi.org/10.1016/j.ijforecast.2018.06.004
[3] Carbonneau, R., Laframboise, K., & Vahidov, R. (2008). Application of machine learning techniques for supply chain demand

forecasting. *European Journal of Operational Research*, 184(3), 1140–1154. https://doi.org/10.1016/j.ejor.2006.12.004

[4] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0

[5] Bandara, K., Bergmeir, C., & Smyl, S. (2021). Forecasting across time series databases using recurrent neural networks on groups of similar series. *Expert Systems with Applications*, 140, 112896. https://doi.org/10.1016/j.eswa.2019.112896

[6] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Conference* (pp. 785–794). https://doi.org/10.1145/2939672.2939785

[7] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

[8] Chopra, S., & Meindl, P. (2019). *Supply chain management: Strategy, planning, and operation* (7th ed.). Pearson.

[9] Syntetos, A. A., Babai, M. Z., & Gardner, E. S. (2015). Forecasting intermittent inventory demands: A review. *International Journal of Forecasting*, 31(1), 1–17. https://doi.org/10.1016/j.ijforecast.2014.01.008

[10] Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.

[11] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. https://doi.org/10.1023/A:1010933404324

[12] Khashei, M., & Bijari, M. (2010). An artificial neural network (p,d,q) model for time series forecasting. *Expert Systems with Applications*, 37(1), 479–489. https://doi.org/10.1016/j.eswa.2009.05.044

[13] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85. https://doi.org/10.1016/j.ijforecast.2019.03.017

[15] Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

[16] Oroojlooyjadid, A., & Hajinezhad, D. (2020). A review of reinforcement learning in inventory management. *Computers &*

*Industrial Engineering*, 145, 106543. https://doi.org/10.1016/j.cie.2020.106543
[17] Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001
[18] Montero-Manso, P., Hyndman, R. J., Athanasopoulos, G., & Talagala, T. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1), 86–92. https://doi.org/10.1016/j.ijforecast.2019.02.011
[19] Laptev, N., Yosinski, J., Li, L. E., & Smyl, S. (2017). Time-series extreme event forecasting with neural networks at Uber. In *ICML Time Series Workshop*.
[20] Babai, M. Z., Ali, M. M., & Nikolopoulos, K. (2012). Impact of temporal aggregation on stock control performance of intermittent demand estimators: Empirical analysis. *Omega*, 40(6), 713–721. https://doi.org/10.1016/j.omega.2011.12.002
[21] Ivanov, D., & Dolgui, A. (2020). Viability of intertwined supply networks: Extending the supply chain resilience angles towards survivability. *International Journal of Production Research*, 58(10), 2904–2915. https://doi.org/10.1080/00207543.2020.1750727
[22] Kourentzes, N., Trapero, J. R., & Barrow, D. K. (2020). Optimizing forecasting systems for inventory management. *International Journal of Production Economics*, 225, 107593. https://doi.org/10.1016/j.ijpe.2019.107593
[23] Kaggle. (2019). *Store Item Demand Forecasting Challenge Dataset*. https://www.kaggle.com/c/demand-forecasting-kernels-only
[24] Bontempi, G., Taieb, S. B., & Le Borgne, Y. A. (2013). Machine learning strategies for time series forecasting. *European Business Intelligence Summer School*, 62–77.
[25] Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introduction to time series analysis and forecasting* (2nd ed.). Wiley.
[26] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, 2(3), 18–22.
[27] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232.
[28] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
[29] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.

[30] Zipkin, P. (2000). *Foundations of inventory management*. McGraw-Hill.
[31] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305.