Journal of Artificial Intelligence and Metaheuristic

♦ ISSN: 2833-5597 Vol: 11, No. 02, 2026 ♦ PP. 74–86

Article DOI: https://doi.org/10.54216/JAIM.110205

FULL LENGTH ARTICLE

# A Smart Retail Point-of-Sale System Using Machine Learning Forecasting and Application-Level Failover

**Sara albuainain**[1,⋆]

[1] School of ICT, Faculty of Engineering, Design and Information & Communications Technology (EDICT), Bahrain Polytechnic, PO Box 33349, Isa Town, Bahrain

**Email:** 202003580@student.polytechnic.bh

Received: January 08, 2026 Revised: March 10, 2026 Accepted: May 12, 2026 ⋆ Corresponding author

## ABSTRACT

Point-of-Sale (POS) systems support retail transaction processing, inventory management, and operational decision-making. However, many traditional POS systems rely on a single centralized database, creating a critical point of failure during server faults, database errors, or network disruptions. Conventional POS systems may also lack intelligent forecasting capabilities, causing inventory decisions to depend on manual estimation or fixed reorder rules. These limitations can lead to operational interruption, inventory inconsistency, and inefficient stock management. This study proposes a high-availability smart POS system integrated with AI-based inventory forecasting. The framework uses a dual-database architecture consisting of a primary database and a secondary backup database. Application-level failover redirects operations automatically when the primary database becomes unavailable, while an event-driven backup mechanism synchronizes important data after critical transactions. The forecasting component was developed using the Rossmann Store Sales dataset. Several models were evaluated, including Random Forest (RF), Bagging, CatBoost, XGBoost, and KNN Regressor. Optimization algorithms including PSO, MVO, HHO, and DE were also integrated with RF. The results showed that baseline RF achieved the best overall performance, with MSE = 0.080454, RMSE = 0.283643, MAE = 0.184264, $R^2$ = 0.919712, and WI = 0.978542. Among optimized models, PSO + RF achieved the strongest optimizer-based result, with MSE = 0.081480 and RMSE = 0.285446. The implemented system also demonstrated successful database connectivity, failover readiness, and automatic backup execution. The findings confirm that the proposed framework can support continuous retail operation while providing accurate AI-based forecasting and inventory recommendations.

**Keywords:** Smart POS ▪ Machine learning forecasting ▪ Application-level failover ▪ Inventory management ▪ Random Forest

## 1. INTRODUCTION

Point-of-Sale (POS) systems have become fundamental components of modern retail environments because they support transaction processing, inventory management, sales monitoring, and operational decision-making. Contemporary POS platforms are no longer limited to billing and payment activities; rather, they have evolved into intelligent business management systems that integrate transactional functions with analytical and predictive capabilities [1, 2]. The digital transformation of retail has increased organizational dependence on POS infrastructures for operational continuity, customer experience improvement, and real-time business intelligence [3, 4]. Consequently, reliability, availability, and forecasting

capability have become essential requirements for modern POS architectures.

Despite this development, many conventional POS solutions still depend on centralized database infrastructures, where all operational activities rely on a single database server. Although such architectures simplify deployment and maintenance, they introduce a single point of failure. Hardware faults, database corruption, software errors, or network interruptions may stop retail operations and cause transaction loss, inventory inconsistency, inaccurate reporting, and customer dissatisfaction, especially in high-volume retail environments [5]. These limitations have increased interest in fault-tolerant and high-availability mechanisms capable of maintaining service continuity during failure conditions.

High availability in retail computing is often achieved through enterprise clustering, distributed databases, cloud redundancy, or database replication [3]. While these approaches provide strong reliability, they are commonly associated with high cost, deployment complexity, and maintenance overhead, making them difficult for small and medium-sized enterprises (SMEs). Application-level failover offers a practical alternative by placing monitoring and switching logic at the application layer rather than relying entirely on database-level clustering. In this approach, operations are redirected from a primary database to a secondary database when failure occurs, offering a lightweight and cost-effective solution for retail environments that require continuity without expensive infrastructure.

In parallel, modern retail systems increasingly require intelligent decision support. Inventory management is particularly critical because inaccurate planning may cause stock shortages, overstocking, increased operational cost, and reduced customer satisfaction. Traditional inventory control techniques commonly rely on fixed reorder thresholds, manual estimation, or simple statistical methods that cannot fully adapt to dynamic consumer behavior and market conditions [6]. The availability of transactional retail data has therefore encouraged the use of Artificial Intelligence (AI) and Machine Learning (ML) for inventory forecasting and demand prediction [7, 8].

Machine learning models can analyze historical sales data and identify hidden consumption patterns that are difficult to detect using conventional methods. Through predictive modeling, retailers can estimate future demand more accurately and improve replenishment decisions [7]. Ensemble learning, regression models, and optimization-enhanced ML techniques have shown strong potential for improving forecasting accuracy, feature selection, parameter tuning, and inventory-related decision-making [9, 10].

This research proposes a unified intelligent POS framework that combines high-availability database infrastructure with AI-driven inventory forecasting. The proposed architecture uses a dual-database redundancy model composed of primary and secondary databases controlled by an application-level failover mechanism. Under normal operation, the primary database handles transactions while the secondary database remains synchronized as a standby backup. When failure occurs, operations are redirected to the backup database without interrupting retail activities. Alongside this reliability component, the system integrates an AI-based forecasting module trained using the Rossmann Store Sales dataset from Kaggle, which contains historical retail sales records for 1,115 stores and includes sales volume, customers, promotions, holidays, store type, assortment, and competition-related attributes.

Several ML models were evaluated, including Random Forest, Bagging, CatBoost, XGBoost, and KNN Regressor. Metaheuristic optimization algorithms were also integrated with Random Forest, including PSO, MVO, HHO, and DE. The experimental results showed that the baseline Random Forest model achieved the strongest overall performance, with MSE = 0.080454, RMSE = 0.283643, MAE = 0.184264, $R^2$ = 0.919712, and WI = 0.978542. The optimizer-enhanced models did not outperform the baseline Random Forest, indicating that the baseline configuration was already stable and accurate for the selected dataset.

The contribution of this work lies in integrating reliability engineering and AI-based forecasting within a unified retail management platform. While previous studies often focus either on high availability or forecasting, limited attention has been given to combining both within a practical POS architecture suitable for SMEs [5, 1, 4]. The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the methodology and system design. Section 4 discusses the results. Section 5 concludes the study and presents future work.

## 2. LITERATURE REVIEW

AI, ML, and Deep Learning (DL) have significantly transformed retail demand forecasting and inventory optimization by improving prediction accuracy, operational efficiency, and decision support. Recent studies increasingly combine predictive analytics, optimization algorithms, enterprise integration, and external contextual data to overcome the limitations of traditional statistical forecasting methods.

Early work demonstrated that AI-based forecasting models outperform conventional techniques such as ARIMA and Exponential Smoothing Techniques (ETS). ML-based retail forecasting achieved approximately 20% inventory savings and improved product availability by 15%, highlighting the importance of continuous calibration and real-time data integration [11]. Similarly, a hybrid LSTM and ensemble forecasting framework using historical sales, sentiment, weather, and economic indicators reduced forecasting errors by up to 25% compared with conventional statistical models [12]. These findings confirm the suitability of nonlinear learning approaches for complex retail demand patterns.

Several studies embedded forecasting directly into operational retail systems. BillWise integrated Random Forest forecasting with automated inventory replenishment in a POS microservices architecture for SMEs [13]. An ERP-integrated framework used XGBoost with FastAPI and Odoo ERP to automate replenishment, improving inventory turnover by more than 130% and reducing costs by 8.51% [14]. These studies demonstrate the practical value of integrating forecasting intelligence into daily retail workflows.

Deep learning approaches have also become prominent because they capture temporal dependencies and nonlinear demand behavior. Comparative studies of LSTM, GRU, TCN, and TFT reported that TFT achieved 96.1% accuracy with the

lowest MAE and RMSE in tire demand forecasting [15]. Similarly, LSTMixer outperformed classical ML and other DL methods in retail time-series forecasting [16]. Hybrid models have shown similar strength: StackRetailNet combined LightGBM, RF, XGBoost, LSTM, Transformer, and FFNN through meta-learning and achieved 98.95% accuracy [17], while a CNN-LSTM model using holidays, salary periods, protests, and weather achieved a MAPE of 4.16% [18].

External variables are increasingly recognized as critical for forecasting. Exogeneity-aware forecasting reduced errors by up to 22% using weather, economic, and market indicators [19], while pandemic-aware feature engineering improved post-pandemic forecasting by representing COVID-19 restriction levels [20]. Inventory optimization studies have also integrated forecasting with decision-support mechanisms. A hybrid approach using ARM, Random Forest, and FP-Growth supported EOQ, ROP, and safety-stock decisions [21]; meanwhile, a transformer-based multi-agent deep reinforcement learning framework reduced forecasting error by 18.2% and stockout rates by 23.5% using IoT, RFID, and smart shelf data [22].

Metaheuristic optimization has also been used to improve forecasting models. A Bi-GRU model optimized with the Mayfly Algorithm achieved RMSE = 0.06 and MAPE = 3.1%, outperforming traditional DL methods [23]. Other work explored clustering, AutoML, and geospatial intelligence for category-level forecasting and market potential analysis [24, 25]. AI applications have further expanded into smart shopping carts, edge computing, cloud analytics, and POS integration [26]. Review studies emphasize the importance of feature engineering, hyperparameter tuning, exogenous variables, transfer learning, scalability, benchmarking, and hybrid ML/DL forecasting systems [27, 28, 29]. Human judgment has also been integrated with AI through a Demand Forecasting Decision Model that combines expert estimates, time-series models, ML models, and bias adjustment [30].

Overall, the reviewed literature shows that retail forecasting is moving toward integrated, adaptive systems that combine ML, DL, reinforcement learning, optimization, feature engineering, enterprise integration, and contextual information. However, challenges remain in interpretability, intermittent demand, scalability, standardized benchmarking, and real-world deployment. Table 1 summarizes the selected studies.

**Table 1.** Summary of reviewed studies on AI-based retail forecasting and inventory optimization

| Ref | Focus and Methodology | Key Findings and Contributions |
|---|---|---|
| [11] | **AI-based retail forecasting**<br>Machine learning forecasting models compared with ARIMA and ETS | Improved forecasting accuracy, achieved 20% inventory savings, and increased product availability by 15%. |
| [12] | **Supply chain demand forecasting**<br>Hybrid LSTM and ensemble learning using sales, sentiment, weather, and economic data | Reduced forecasting errors by up to 25% and improved inventory cost control and order fulfillment. |
| [13] | **SME inventory and POS forecasting**<br>BillWise POS system using Random Forest regression within Spring Boot and FastAPI microservices | Generated timely reorder alerts and integrated forecasting directly into operational inventory decisions. |
| [15] | **Tire demand forecasting and inventory optimization**<br>LSTM, GRU, TCN, and TFT deep learning models | TFT achieved the best performance with 96.1% accuracy, MAE of 2.9, and RMSE of 4.7. |
| [24] | **Retail sales forecasting by product category**<br>TabNet, DNNs, ANNs, regression models, and clustering methods including K-Means, HC, and PAM | TabNet and the second DNN model achieved the lowest RMSE and MAE. |
| [26] | **Smart retail shopping and inventory analytics**<br>AI smart cart using RL, LSTM, edge computing, cloud analytics, and POS integration | Improved tracking, navigation, inventory forecasting, billing automation, and customer experience. |
| [20] | **Post-pandemic demand forecasting**<br>Feature engineering using COVID-19 restriction indicators with deep probabilistic forecasting models | Improved forecasting accuracy across product aggregation levels and retail datasets affected by pandemic bias. |
| [21] | **Inventory optimization and transaction pattern mining**<br>Random Forest forecasting combined with Association Rule Mining and FP-Growth | Supported EOQ, ROP, and safety stock decisions while reducing stockouts and excess inventory. |
| [25] | **Grid-based sales forecasting and site selection**<br>AutoML combined with geospatial intelligence | Outperformed regression models and supported market-potential prediction and location selection strategies. |
| [14] | **ERP-integrated inventory forecasting**<br>XGBoost deployed through FastAPI and integrated with Odoo ERP and PostgreSQL | Improved inventory turnover by over 130% and reduced operational costs by 8.51%. |
| [19] | **Exogenous-variable-aware retail forecasting**<br>Machine learning framework for selecting weather, economic, and market trend variables | Reduced forecast error by up to 22% in a European online retail case study. |
| [27] | **Intermittent demand forecasting review**<br>Review of ML methods, feature engineering, hyperparameter tuning, exogenous variables, and transfer learning | Identified industrialization challenges, benchmarking gaps, and future opportunities for ML-based intermittent forecasting. |
| [30] | **Human judgment and AI forecasting integration**<br>Demand Forecasting Decision Model combining expert estimates, time-series models, ML models, and bias adjustment | Produced more accurate forecasts than common practical forecasting methods and supported short-term decisions. |
| [22] | **Joint demand forecasting and inventory optimization**<br>Transformer sequence modeling with multi-agent deep reinforcement learning using IoT, RFID, and smart shelf data | Reduced forecast error by 18.2% and stockout rates by 23.5%. |
| [17] | **Hybrid retail forecasting and inventory classification**<br>StackRetailNet using LightGBM, RF, XGBoost, LSTM, RNN, Transformer, FFNN, and meta-learning | Achieved 98.95% accuracy and improved forecast-aware inventory classification. |
| [23] | **Sales forecasting in supply chain management**<br>Bi-GRU optimized using the Mayfly Algorithm | Achieved RMSE of 0.06 and MAPE of 3.1%, outperforming traditional deep learning models. |
| [28] | **Predictive analytics in SCM**<br>AI and ML-based predictive analytics for demand forecasting | Highlighted the role of predictive analytics in improving forecasting accuracy, efficiency, and responsiveness. |
| [29] | **ML/DL sales forecasting review**<br>Systematic review of ML and DL forecasting studies from 2020–2024 | Emphasized hybrid models, scalability, external variables, and industry-specific customization. |
| [16] | **Retail time-series forecasting**<br>TFT, TCN, Time-Series Mixer, and proposed LSTMixer model | LSTMixer achieved superior forecasting performance compared with statistical and ML baselines. |
| [18] | **Multivariate retail sales forecasting**<br>Hybrid CNN-LSTM using holidays, salary days, protests, weather, and demographic variables | Achieved MAPE of 4.16% and improved forecasting reliability through contextual feature integration. |

## 3. METHODOLOGY AND SYSTEM DESIGN

The methodology is based on designing and implementing an integrated intelligent POS system that combines high-availability database architecture with AI-based inventory forecasting. The system addresses two limitations of conventional POS environments: vulnerability of centralized databases to failure and dependence on manual or static inventory estimation. A modular system engineering approach was adopted, dividing the application into user interface, service, and database layers. This structure improves maintainability, reduces complexity, and allows failover, backup, and forecasting functions to operate within the same workflow.

The system was implemented using Python, Tkinter for the graphical user interface, and MySQL for database management. The forecasting module uses historical sales data to estimate demand and generate reorder recommendations. Thus, the AI component is not treated as an isolated experimental module but as a decision-support function integrated with POS operations.

### 3.1 Overall System Architecture

The proposed architecture consists of three layers: user interface, service, and database. The user interface layer includes login, dashboard, product management, sales transaction, and forecasting interfaces. It enables users to authenticate, manage products, process sales, and request demand forecasts. The service layer contains business logic for transaction handling, database communication, failover control, backup execution, and AI forecasting. The database layer stores operational and analytical data and consists of primary and secondary relational databases. Fig. 1 presents the general architecture.

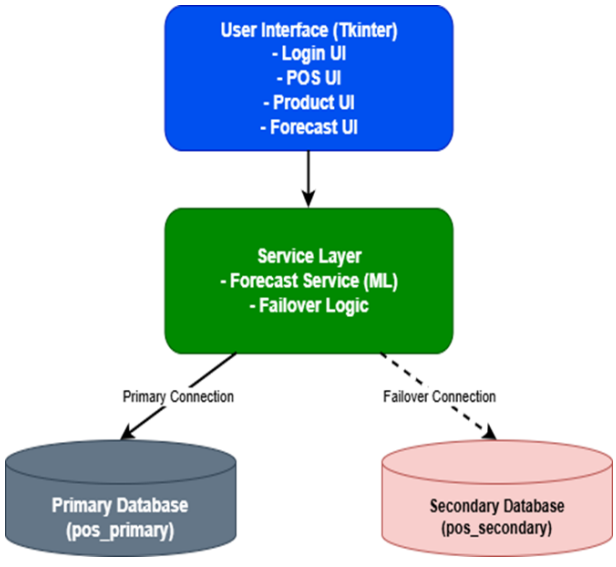



**Figure 1.** Overall architecture of the proposed smart POS system.

The layered design separates presentation, processing, and storage responsibilities. The user interface triggers service-layer functions, the service layer manages both reliability and forecasting processes, and the database layer supports redundancy and historical data storage. This modularity enables later extension to cloud synchronization, advanced forecasting, web interfaces, or additional security controls.

### 3.2 Database Design and Redundancy

The database design supports transaction reliability and inventory forecasting. The proposed system uses a dual-database structure consisting of a primary database, `pos_primary`, and a secondary database, `pos_secondary`. During normal operation, the primary database stores products, sales, stock updates, and operational records. The secondary database has the same schema and acts as a standby database for failover operation, following an active–passive redundancy model suitable for SMEs. Fig. 2 illustrates this design.

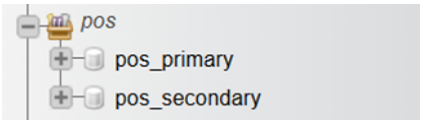



**Figure 2.** Primary and secondary database structure used in the proposed POS system.

The main database tables are shown in Fig. 3. The current implementation focuses on the products and sales tables because they contain the core data required for inventory management, transaction processing, and forecasting.

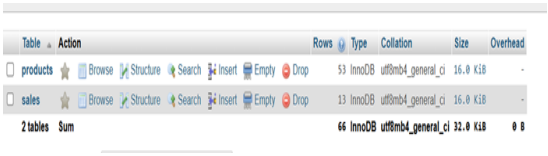


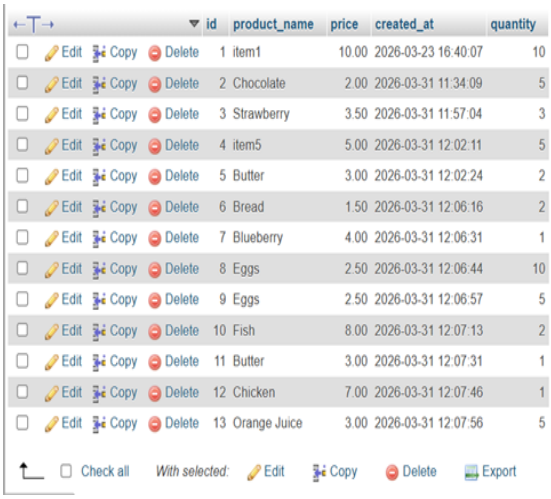

**Figure 3.** Main tables created within the POS database.

The products table stores product identifier, name, price, and quantity, as shown in Fig. 4. It is used by product management, sales, and forecasting modules. The sales table, shown in Fig. 5, stores transaction identifier, product name, quantity sold, price, and transaction date. It provides both operational transaction history and the historical data foundation for forecasting.

| | product_id | name | price | quantity |
|---|---|---|---|---|
| Edit Copy Delete | 6 | Milk | 3.50 | 100 |
| Edit Copy Delete | 7 | Chocolate | 2.00 | 75 |
| Edit Copy Delete | 8 | Bread | 1.50 | 118 |
| Edit Copy Delete | 9 | Eggs | 2.50 | 75 |
| Edit Copy Delete | 10 | Cheese | 4.00 | 60 |
| Edit Copy Delete | 11 | Butter | 3.00 | 47 |
| Edit Copy Delete | 12 | Yogurt | 2.20 | 70 |
| Edit Copy Delete | 13 | Orange Juice | 3.00 | 75 |
| Edit Copy Delete | 14 | Apple Juice | 3.00 | 70 |
| Edit Copy Delete | 15 | Water Bottle | 1.00 | 200 |
| Edit Copy Delete | 16 | Soda | 1.50 | 150 |
| Edit Copy Delete | 17 | Cereal | 4.50 | 60 |
| Edit Copy Delete | 18 | Rice | 5.00 | 100 |
| Edit Copy Delete | 19 | Pasta | 2.50 | 90 |
| Edit Copy Delete | 20 | Tomato Sauce | 2.00 | 70 |
| Edit Copy Delete | 21 | Olive Oil | 6.00 | 40 |
| Edit Copy Delete | 22 | Salt | 1.00 | 120 |
| Edit Copy Delete | 23 | Sugar | 2.00 | 100 |
| Edit Copy Delete | 24 | Flour | 3.00 | 80 |
| Edit Copy Delete | 25 | Coffee | 5.50 | 50 |
| Edit Copy Delete | 26 | Tea | 3.50 | 60 |
| Edit Copy Delete | 27 | Cookies | 2.50 | 70 |

**Figure 4.** Structure and sample records of the products table.

Both databases use identical tables and fields, allowing backup and failover operations without structural conflict.



| | id | product_name | price | created_at | quantity |
|---|---|---|---|---|---|
| Edit Copy Delete | 1 | item1 | 10.00 | 2026-03-23 16:40:07 | 10 |
| Edit Copy Delete | 2 | Chocolate | 2.00 | 2026-03-31 11:34:09 | 5 |
| Edit Copy Delete | 3 | Strawberry | 3.50 | 2026-03-31 11:57:04 | 3 |
| Edit Copy Delete | 4 | item5 | 5.00 | 2026-03-31 12:02:11 | 5 |
| Edit Copy Delete | 5 | Butter | 3.00 | 2026-03-31 12:02:24 | 2 |
| Edit Copy Delete | 6 | Bread | 1.50 | 2026-03-31 12:06:16 | 2 |
| Edit Copy Delete | 7 | Blueberry | 4.00 | 2026-03-31 12:06:31 | 1 |
| Edit Copy Delete | 8 | Eggs | 2.50 | 2026-03-31 12:06:44 | 10 |
| Edit Copy Delete | 9 | Eggs | 2.50 | 2026-03-31 12:06:57 | 5 |
| Edit Copy Delete | 10 | Fish | 8.00 | 2026-03-31 12:07:13 | 2 |
| Edit Copy Delete | 11 | Butter | 3.00 | 2026-03-31 12:07:31 | 1 |
| Edit Copy Delete | 12 | Chicken | 7.00 | 2026-03-31 12:07:46 | 1 |
| Edit Copy Delete | 13 | Orange Juice | 3.00 | 2026-03-31 12:07:56 | 5 |

Check all With selected: Edit Copy Delete Export

**Figure 5.** Structure and sample records of the sales table.

New product and sales records are copied from the primary database to the secondary database, keeping the backup database synchronized and ready for operation during failure.

### 3.3 Failover and Backup Mechanism

The proposed system adopts an application-level failover and backup mechanism to reduce interruption risk during database failure. The application first attempts to connect to the primary database. If the connection fails because of server unavailability, network interruption, authentication failure, or other database errors, the system automatically attempts to connect to the secondary database. This process allows transaction processing, stock updating, product retrieval, and forecasting-related access to continue through the available database. Fig. 6 shows the workflow.

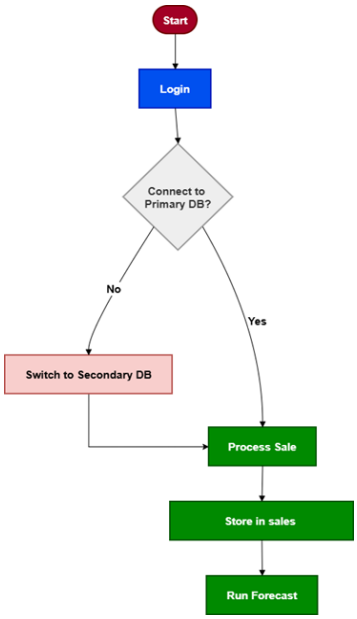



**Figure 6.** Workflow of the proposed POS system showing database connection handling, failover switching, transaction processing, and forecasting data flow.

**Algorithm 1** Application-level database connection and failover process

```
Define primary database configuration PRIMARY_DB
Define secondary database configuration SECONDARY_DB
Attempt to connect to PRIMARY_DB
if primary database connection is successful then
    Set active database as primary database
    Return primary connection and primary status
else
    Attempt to connect to SECONDARY_DB
    if secondary database connection is successful then
        Set active database as secondary database
        Return secondary connection and secondary status
    else
        Return null connection and failure status
    end if
end if
```

The system also performs event-driven backup after critical operations such as product insertion, stock update, and sales transaction. Records are copied from the primary to the secondary database using replacement-based synchronization to avoid duplication and preserve consistency. This supports both transaction continuity and forecasting reliability because the AI module depends on historical sales data.

**Algorithm 2** Real-time backup and synchronization process

```
Establish connection with the primary database
Establish connection with the secondary database
Retrieve the list of tables from the primary database
for each table in the primary database do
    Retrieve all records from the current table
    for each record in the table do
        Insert or replace the record in the corresponding secondary database table
    end for
end for
Commit all changes in the secondary database
Display backup completion message
```

The mechanism is lightweight because it does not require distributed consensus or database clustering. However, frequent real-time backups may introduce overhead under high transaction volumes; therefore, future versions may use incremental synchronization, transaction logs, or asynchronous replication.

### 3.4 Dataset Description

The AI forecasting component was developed and evaluated using the Rossmann Store Sales dataset. The dataset contains historical sales data for 1,115 stores and supports a supervised regression task in which `Sales` is predicted for store-date records. It includes `train.csv`, `store.csv`, and `sample_submission.csv`. The `train.csv` file contains daily sales records, customer counts, promotions, store opening status, and holiday indicators. The `store.csv` file provides store type, assortment, competition distance, competition opening information, and long-term promotion attributes.

The target variable is `Sales`, representing daily store turnover. Input features include operational variables such as `Store`, `Customers`, and `Open`; promotional and holiday variables such as `Promo`, `StateHoliday`, and `SchoolHoliday`; and store-level variables such as `StoreType`, `Assortment`, `CompetitionDistance`, `Promo2`, and `PromoInterval`. The `train.csv` and `store.csv` files were merged using the `Store` field to create a richer feature space. Although the dataset was not generated by the prototype POS system, it provides a realistic benchmark for evaluating the forecasting module before integration into the POS workflow.

The time-series decomposition, monthly sales distribution, and feature importance analysis are presented in Figures 7–9. These visualizations clarify trend, seasonal, residual, monthly, and feature-contribution patterns that support forecasting model interpretation.

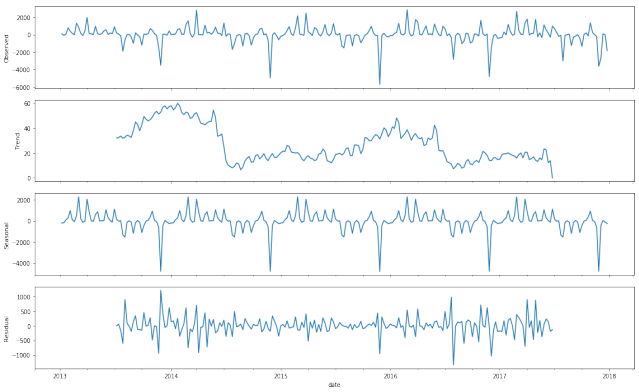


**Figure 7.** Time-series decomposition showing the observed, trend, seasonal, and residual components of the dataset.

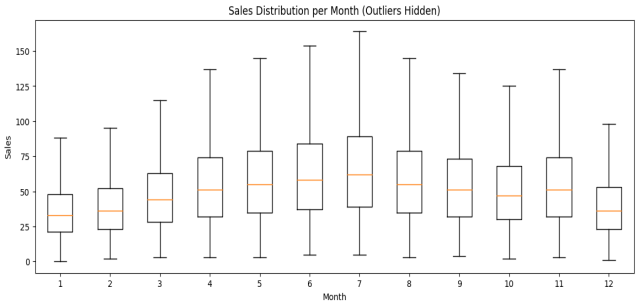



**Figure 8.** Boxplot representation of monthly sales distribution with outliers hidden.

### 3.5 Data Preprocessing

Data preprocessing was performed to transform the Rossmann dataset into a clean and machine-learning-ready form. The workflow included merging datasets, removing closed-store and invalid zero-sales observations, handling missing values, extracting temporal features, encoding categorical variables, normalizing numerical variables, and splitting the data into training and testing subsets. Algorithm 3 summarizes the workflow.

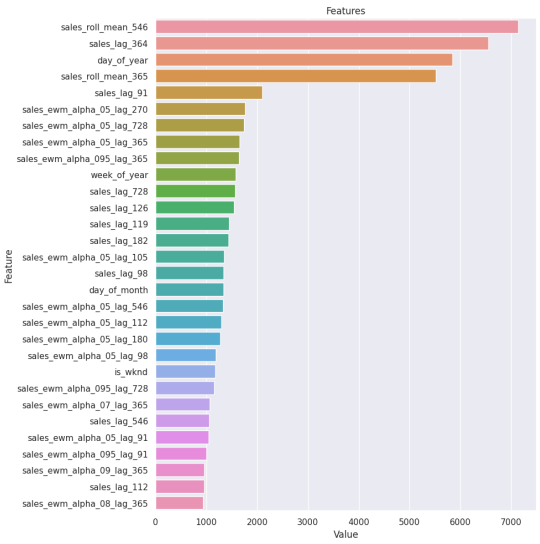



**Figure 9.** Feature importance ranking of the variables used in the forecasting model.

**Algorithm 3** Preprocessing workflow for the AI-based sales forecasting dataset

1: Load the historical sales dataset from `train.csv`
2: Load the supplementary store dataset from `store.csv`
3: Merge the datasets using the `Store` identifier
4: Remove records corresponding to closed stores
5: Remove invalid or zero-sales observations
6: Handle missing values in numerical and categorical features
7: Extract temporal attributes from the `Date` feature
8: Encode categorical variables into numerical representations
9: Normalize numerical features
10: Split the processed dataset into training and testing subsets

The merging stage combines daily sales records with store-level information, allowing models to learn from both transactional and contextual variables. Closed-store records and invalid sales observations were removed because they do not represent normal demand behavior. Missing numerical and categorical values, especially those related to competition and long-term promotions, were replaced using consistent strategies to maintain dataset completeness. The `Date` feature was transformed into temporal variables such as day, month, year, week number, and day of week to capture seasonal and periodic patterns. Categorical variables such as `StateHoliday`, `StoreType`, `Assortment`, and `PromoInterval` were encoded into numerical form, and numerical features were normalized to support distance-based models such as KNN. Finally, the dataset was split into training and testing subsets to evaluate generalization performance.

### 3.6 Machine Learning Models

The forecasting task was formulated as supervised regression. Five models were evaluated: Random Forest (RF), Bagging, CatBoost, XGBoost, and KNN Regressor. These models represent ensemble, boosting, and instance-based learning paradigms suitable for structured retail forecasting.

#### Random Forest (RF)

Random Forest combines multiple decision trees trained on random subsets of data and features, reducing variance and improving generalization [31]. For regression, the prediction is:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

where $h_t(x)$ is the prediction of tree $t$ and $T$ is the number of trees. Feature importance can be estimated by mean decrease in impurity:

$$I(f_j) = \frac{1}{T}\sum_{t=1}^{T}\sum_{n\in N_t(j)} p(n)\Delta i(n)$$

RF is suitable for retail forecasting because it captures nonlinear interactions among sales, customers, promotions, holidays, competition, and store characteristics.

#### Bagging

Bagging trains multiple learners on bootstrap samples and averages their predictions to reduce variance [32]:

$$\hat{y} = \frac{1}{M}\sum_{m=1}^{M} h_m(x)$$

where $M$ is the number of learners. This improves robustness against noisy retail fluctuations caused by promotions, seasonality, and behavioral changes.

#### CatBoost

CatBoost is a gradient boosting algorithm designed to handle categorical features efficiently using ordered boosting and advanced encoding [33]. Its prediction is:

$$F(x) = \sum_{k=1}^{K} \gamma_k h_k(x)$$

and the loss minimization objective is:

$$L = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i).$$

CatBoost is relevant because the Rossmann dataset contains categorical variables such as `StoreType`, `Assortment`, `StateHoliday`, and `PromoInterval`.

#### XGBoost

XGBoost is an optimized gradient boosting framework for structured data [34]. Its prediction model is:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathscr{F}$$

with objective:

$$\mathscr{L} = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \quad \Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2.$$

The regularization term helps control model complexity and reduce overfitting.

#### KNN Regressor

KNN Regressor predicts a sample using the average output of its nearest neighbors [35]. The Euclidean distance is:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

and the prediction is:

$$\hat{y} = \frac{1}{K}\sum_{i=1}^{K} y_i.$$

Because KNN is scale-sensitive, normalization was applied during preprocessing.

### 3.7 Role of Metaheuristics in Hyperparameter Optimization

Hyperparameter optimization is essential because model performance depends on parameters controlling complexity, learning behavior, tree structure, neighborhood size, and convergence stability. Manual tuning, grid search, and random search may be inefficient in large, nonlinear, and irregular search spaces. Metaheuristic optimizers provide a flexible alternative because they can search complex spaces without requiring gradient information. Their main advantage is the balance between exploration, which investigates diverse regions, and exploitation, which refines promising solutions.

In this study, metaheuristics were used to optimize forecasting models by treating each candidate solution as a hyperparameter configuration. Each candidate was evaluated by training the model and calculating forecasting error. The optimizer then updated candidates iteratively until termination.

**Proposed Particle Swarm Optimization (PSO).** PSO is a population-based optimizer inspired by bird flocking and fish schooling [36]. Each particle represents a possible hyperparameter configuration and moves according to personal and global best experiences. The particle and velocity matrices are:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix}, \tag{1}$$

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,d} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,d} \end{bmatrix}. \tag{2}$$

Velocity and position updates are:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (p_i^t - x_i^t) + c_2 r_2 (g^t - x_i^t), \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{4}$$

The fitness function minimizes forecasting error:

$$Fitness(x_i) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2. \tag{5}$$

Personal and global bests are updated as:

$$p_i^{t+1} = \begin{cases} x_i^{t+1}, & \text{if } Fitness(x_i^{t+1}) < Fitness(p_i^t), \\ p_i^t, & \text{otherwise,} \end{cases} \quad (6)$$

$$g^{t+1} = \arg\min_{p_i^{t+1}} Fitness(p_i^{t+1}). \quad (7)$$

**Algorithm 4** Particle Swarm Optimization for Hyperparameter Tuning

**Require:** Training data, validation data, search space, swarm size $n$, maximum iterations $T_{\max}$, inertia weight $\omega$, acceleration coefficients $c_1$ and $c_2$
**Ensure:** Best hyperparameter configuration $g$
Initialize particle positions $X$ randomly within the defined search space
Initialize particle velocities $V$ randomly
**for** each particle $i = 1$ to $n$ **do**
    Train the machine learning model using configuration $x_i$
    Evaluate particle fitness using forecasting error
    Set personal best $p_i = x_i$
**end for**
Determine the initial global best solution $g$
**for** each iteration $t = 1$ to $T_{\max}$ **do**
    **for** each particle $i = 1$ to $n$ **do**
        Update velocity $v_i^{t+1}$
        Update position $x_i^{t+1}$
        Apply boundary constraints
        Train the model using $x_i^{t+1}$
        Evaluate updated particle fitness
        Update personal best $p_i$
    **end for**
    Update global best $g$
**end for**
**return** $g$

**Benchmark Optimizers.** MVO is a population-based optimizer inspired by multiverse theory in which candidate universes share information based on inflation rates and exploit promising regions through wormhole mechanisms [37]. HHO is a population-based optimizer that alternates between global exploration and local exploitation [38]. DE is an evolutionary optimizer that uses mutation, crossover, and selection to generate improved candidate solutions [39]. Together, PSO, MVO, HHO, and DE provide a diverse optimization framework for RF hyperparameter tuning.

### 3.8 Model Evaluation Metrics

The forecasting models were evaluated using MSE, RMSE, MAE, coefficient of determination ($R^2$), and Willmott Index (WI). These metrics measure error magnitude, prediction stability, explanatory ability, and agreement between predicted and observed values.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

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

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

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

$$WI = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}\left(|\hat{y}_i - \bar{y}| + |y_i - \bar{y}|\right)^2}$$

Lower MSE, RMSE, and MAE indicate better accuracy, while higher $R^2$ and WI values indicate stronger explanatory performance and agreement.

### 3.9 Integration of AI Forecasting with the POS System

The AI forecasting module is integrated into the service layer of the POS system. When a user requests a forecast, the service layer retrieves historical sales data from the active database, prepares input variables, applies the trained forecasting model, and returns the predicted demand value to the interface. Thus, forecasting is embedded in the same workflow that manages products, sales, database connectivity, backup, and failover.

Forecast generation can be represented as:

$$\hat{D}_{t+h} = f(X_t), \quad (8)$$

where $\hat{D}_{t+h}$ is predicted future demand, $X_t$ is the input feature vector, and $f(\cdot)$ is the trained model. The reorder quantity is calculated as:

$$RQ = \max(\hat{D}_{t+h} - S_t, 0), \quad (9)$$

where $S_t$ is current stock. This rule converts predictions into practical inventory recommendations.

**Algorithm 5** AI-based forecast generation and reorder recommendation

**Require:** Active database connection, selected product, trained forecasting model
**Ensure:** Recommended reorder quantity $RQ$
Retrieve historical sales records for the selected product
Construct the input feature vector $X_t$
Generate future demand prediction $\hat{D}_{t+h}$
Retrieve current stock level $S_t$
**if** $\hat{D}_{t+h} > S_t$ **then**
    Compute $RQ = \hat{D}_{t+h} - S_t$
**else**
    Set $RQ = 0$
**end if**
Display the recommended reorder quantity
**return** $RQ$

Because the forecasting module uses the currently active database, it remains available during failover. Under normal conditions, it uses the primary database; during failure, it uses the secondary database. Therefore, backup synchronization supports both transaction continuity and forecasting reliability.

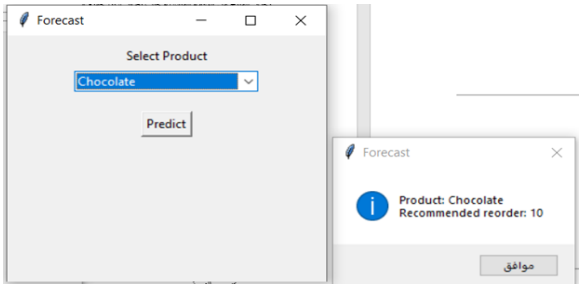



**Figure 10.** Forecasting interface integrated within the proposed smart POS system for generating demand predictions and reorder recommendations.

### 3.10 System Implementation and User Interfaces

The proposed POS system was implemented as a desktop application composed of database, login, dashboard, product management, sales transaction, forecasting, and main execution modules. The database module handles primary/secondary connections, queries, record storage, backup operations, and failover-aware access. The login module verifies credentials and displays database status, as shown in Fig. 11. The dashboard provides access to product management, sales, forecasting, and system controls. The sales module processes transactions, updates stock, and stores sales records, as shown in Fig. 12. The forecasting module generates demand predictions and reorder recommendations using historical sales information, as previously shown in Fig. 10.

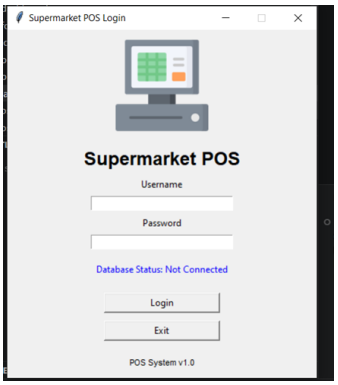



**Figure 11.** Login interface of the proposed smart POS system.

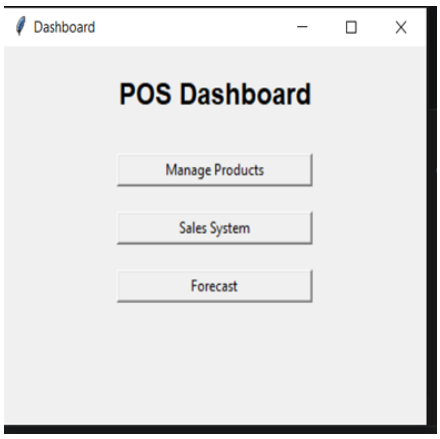



**Figure 12.** Sales transaction interface of the proposed smart POS system.

The user interfaces were designed for usability, operational clarity, and workflow integration. They allow retailers to authenticate, process transactions, manage inventory, monitor database status, and generate AI-based reorder recommendations without interacting directly with the database or ML framework.

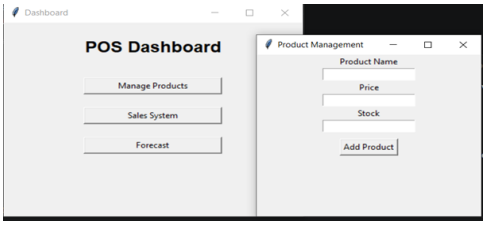



**Figure 13.** Forecast interface used for generating demand prediction and reorder recommendations within the proposed smart POS system.

## 4. RESULTS AND DISCUSSION

This section evaluates the proposed system from two perspectives: operational reliability and forecasting performance. Reliability evaluation focuses on database connectivity, failover readiness, backup, and data protection. Forecasting evaluation compares baseline and optimized ML models using the metrics defined in Section 3.8.

### 4.1 Database Connectivity and Failover Results

The first evaluation stage verified whether the application could communicate with both primary and secondary databases. Fig. 14 shows successful startup and database connection messages, including `PROGRAM STARTED`, `Launching UI...`, `Login UI started`, `Connected to PRIMARY database`, and `Connected to SECONDARY database`. These outputs confirm that the user interface, main execution module, and database layer were correctly integrated.

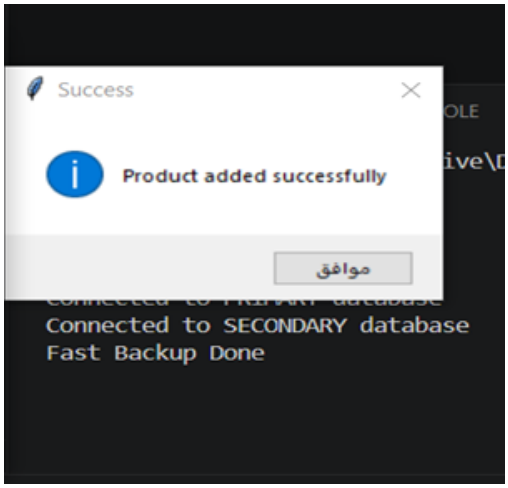



**Figure 14.** System startup and database connectivity result showing successful initialization and connection to both primary and secondary databases.

The result demonstrates failover readiness because the secondary database was available as a standby alternative. The sequential connection strategy allows the system to attempt the primary database first and then switch to the secondary database if the primary connection fails. This confirms the

feasibility of application-level failover as a cost-effective reliability mechanism for SMEs. However, full failover validation should also include primary shutdown, network interruption, transaction interruption during failover, and recovery after primary restoration.

### 4.2 Backup and Data Protection Results

The second evaluation stage examined backup and data protection. The execution log showed that the automatic backup mechanism was triggered immediately after critical database activity, confirmed by the message `Fast Backup Done`. This indicates that backup is integrated directly into the operational workflow rather than executed only as a scheduled offline process.

The proposed event-driven backup reduces the time gap between data modification and backup execution. It improves transaction safety, reduces recovery time, and minimizes the risk of losing recent product or sales records. Table 2 compares the proposed strategy with traditional scheduled backups.

**Table 2.** Comparison between traditional backup and the proposed event-driven backup mechanism

| Feature | Traditional Backup | Proposed System |
|---|---|---|
| Execution Time | Scheduled (daily/hourly) | Real-time (event-driven) |
| Data Loss Risk | Medium | Very low |
| Performance Impact | Low | Optimized |
| Reliability | Moderate | High |

The result confirms that the proposed architecture supports reliability and fault tolerance by preserving critical data immediately after execution. Future improvements may include incremental backup, compressed backup storage, and cloud synchronization to improve scalability.

### 4.3 Baseline Forecasting Results

The baseline experiment evaluated RF, Bagging, CatBoost, XGBoost, and KNN Regressor using the same preprocessing pipeline and evaluation metrics. Table 3 presents the results.

**Table 3.** Baseline forecasting model results

| Model | MSE | RMSE | MAE | $R^2$ | WI |
|---|---|---|---|---|---|
| RF | 0.080454 | 0.283643 | 0.184264 | 0.919712 | 0.978542 |
| Bagging | 0.089274 | 0.298788 | 0.194540 | 0.910910 | 0.976290 |
| CatBoost | 0.116772 | 0.341720 | 0.240584 | 0.883468 | 0.966051 |
| XGBoost | 0.127568 | 0.357167 | 0.254092 | 0.872695 | 0.962517 |
| KNN Regressor | 0.146145 | 0.382289 | 0.246768 | 0.854156 | 0.959206 |

RF achieved the best baseline performance, with the lowest MSE, RMSE, and MAE and the highest $R^2$ and WI. Bagging ranked second, confirming the strength of ensemble approaches for retail sales forecasting. CatBoost and XGBoost produced competitive results but higher errors than RF and Bagging under the current configuration. KNN performed weakest, likely because distance-based estimation is less effective in high-dimensional retail data containing heterogeneous numerical and encoded categorical features.

Figures 15 and 16 visualize the comparative model behavior. The radar and error-metric plots confirm the numerical findings: RF had the strongest overall performance, followed by Bagging, while KNN showed the weakest performance.

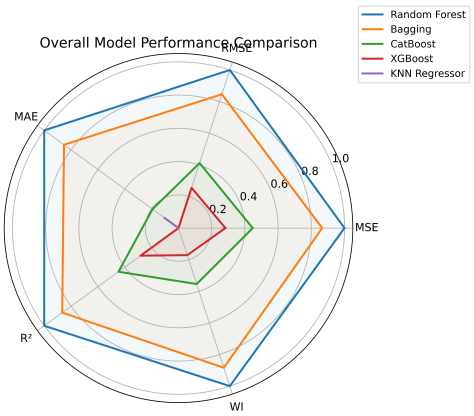



**Figure 15.** Overall model performance comparison using multiple evaluation metrics.

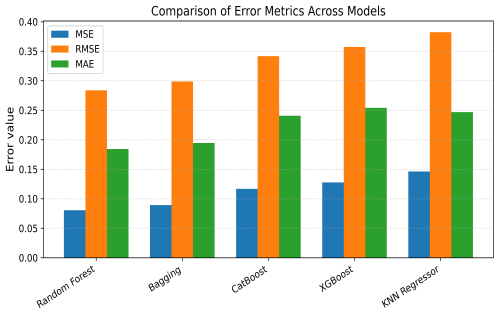



**Figure 16.** Comparison of MSE, RMSE, and MAE across different machine learning models.

### 4.4 Optimized Forecasting Results

After RF achieved the strongest baseline performance, PSO, MVO, HHO, and DE were integrated with RF for hyperparameter optimization. Table 4 presents the optimized forecasting results.

**Table 4.** Optimized forecasting model results

| Model | MSE | RMSE | MAE | $R^2$ | WI |
|---|---|---|---|---|---|
| PSO + RF | 0.081480 | 0.285446 | 0.185172 | 0.918688 | 0.978083 |
| MVO + RF | 0.082868 | 0.287867 | 0.187100 | 0.917303 | 0.977656 |
| HHO + RF | 0.089089 | 0.298478 | 0.195655 | 0.911095 | 0.975704 |
| DE + RF | 0.089346 | 0.298908 | 0.194056 | 0.910838 | 0.975615 |

Among the optimized models, PSO + RF achieved the strongest result, followed by MVO + RF. HHO + RF and DE + RF produced comparable but weaker performance. However, the original RF baseline remained the best overall model. This finding is important because it shows that optimization does not automatically improve forecasting accuracy; rather, the baseline RF configuration was already near an effective solution region.

Figures 17–19 visualize the optimized model results. The normalized heatmap compares all metrics, while the RMSE figures highlight small performance differences among optimizers.

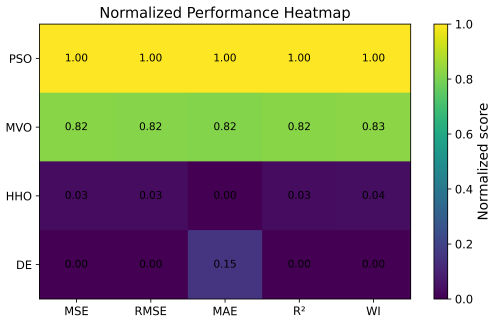



**Figure 17.** Normalized performance heatmap of optimization algorithms across multiple evaluation metrics.

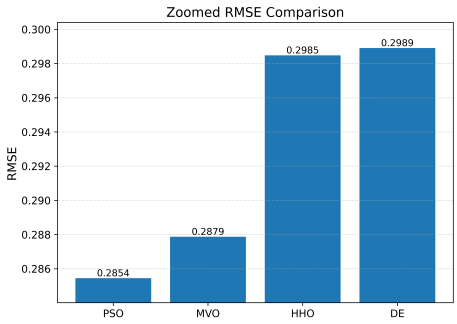



**Figure 18.** Zoomed comparison of RMSE values for different optimization algorithms.

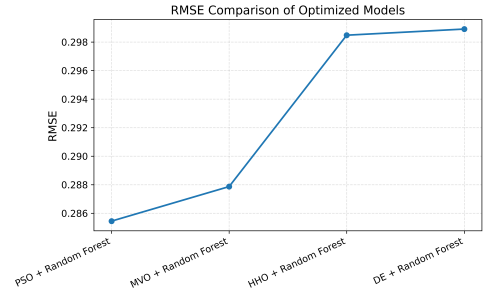



**Figure 19.** RMSE comparison of optimized hybrid Random Forest models using different optimization algorithms.

Overall, the results confirm that ensemble learning can generate reliable retail demand forecasts and that metaheuristic optimization is feasible for RF tuning. Nevertheless, empirical comparison is necessary because the optimized models did not exceed the baseline RF performance in this experiment.

## 5. CONCLUSION

This study presented a high-availability smart POS system integrated with AI-based inventory forecasting. The system combines a dual-database architecture, application-level failover, event-driven backup, and ML-based demand prediction within a unified retail e nvironment. The proposed architecture addresses a major limitation of conventional POS systems: dependence on a single centralized database. By using primary and secondary databases, the system reduces the risk of interruption caused by database failure and supports continuity for SMEs without requiring expensive enterprise infrastructure.

The database connectivity results confirmed that the application could communicate with both database instances, indicating failover readiness. The backup results further showed that event-driven backup was executed after critical operations, reducing the risk of recent data loss and improving transaction safety. These findings demonstrate that application-level failover and backup can provide a practical reliability mechanism for retail POS environments.

The forecasting component was evaluated using the Rossmann Store Sales dataset. Among the baseline models, RF achieved the strongest performance with MSE = 0.080454, RMSE = 0.283643, MAE = 0.184264, $R^2$ = 0.919712, and WI = 0.978542. Optimizer-enhanced models were also evaluated using PSO + RF, MVO + RF, HHO + RF, and DE + RF. PSO + RF achieved the best optimizer-based result, but the baseline RF model remained the strongest overall. This confirms that optimization should be evaluated empirically rather than assumed to improve performance automatically.

The integration of AI forecasting with the POS system adds practical value by converting historical sales data into demand predictions and reorder recommendations. Instead of relying on manual estimation or fixed reorder thresholds, the proposed system supports data-driven inventory planning and can help reduce both stockouts and overstocking. Therefore, the system contributes to both transaction reliability and intelligent retail management.

Despite these contributions, limitations remain. The failover mechanism depends on proper synchronization between databases; incomplete synchronization may cause inconsistency during failover. The system was evaluated in a controlled development environment, so further testing in real retail conditions is required. Forecasting performance also depends on the quality and completeness of historical sales data. Future work will focus on incremental replication, transaction logging, cloud deployment, security mechanisms, advanced forecasting models, larger datasets, and extension of the desktop POS application into web-based or mobile-supported platforms.

In conclusion, the proposed high-availability smart POS system provides a balanced solution that combines reliability, data protection, and AI-supported inventory forecasting. The results confirm that the system can support continuous retail operation while generating accurate forecasting outputs for inventory decision support, making it a practical foundation for intelligent and cost-effective POS systems in future retail applications.

## REFERENCES

[1] G. Venkatasubbu and R. Devabhaktuni, "Integrating generative ai into retail checkout systems: A case study in cloud and application integration," in *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*, 2026, pp. 1–6.

[2] V. R. Polamreddy, "Architecting financially compliant enterprise point-of-sale systems: Data integrity and revenue recognition at scale," *International Journal of Advanced Research in Computer Science & Technology (IJARCST)*, vol. 8, no. 5, pp. 12 993–13 104, 2025.

[3] U. Kumar, D. Krishnamoorthy, and R. Ghadiyaram, "Omnichannel retail analytics: A scalable big data architecture using data vault 2.0 and apache spark for retail

intelligence," in *2025 IEEE Conference on Cloud and Big Data Computing (CBDCom)*, 2025, pp. 207–212.

[4] S. Chen and H. Ding, "The application and optimization method of intelligent decision support systems in chain operation performance management," *International Journal of Information Technologies and Systems Approach (IJITSA)*, vol. 19, no. 1, pp. 1–17, 2026.

[5] M. Rana, T. Joshi, and M. Malik, "Enhancing retail system resilience through integrated cloudless ai and aiops: A framework for real-time market adaptation and consumer behavior response," in *2025 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, 2025, pp. 407–412.

[6] A. Handojo, L. P. Dewi, and C. Liadi, "Optimizing inventory control through hybrid abc-ved segmentation and adaptive reorder points: A case study in electrical wholesale operations," in *2025 12th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2025, pp. 735–741.

[7] D. Guru, "Ai-driven retail replenishment: Deriving patterns and adaptive policies from sales and external data streams," in *2025 5th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, 2025, pp. 475–481.

[8] I. Kostenko, "Event-driven demand model: Semantic mapping of non-transactional intent signals to product skus in proactive e-commerce systems," *Preprint*, 2025.

[9] S. Rizal and D.-S. Kim, "Enhancing blockchain consensus mechanisms: A comprehensive survey on machine learning applications and optimizations," *Blockchain: Research and Applications*, vol. 6, no. 4, p. 100302, 2025.

[10] V. Athiththan and P. Sivasothy, "Enhancing blockchain security through hybrid pos–pbft consensus and machine learning-based anomaly detection," in *2026 IEEE International Research Conference on Smart Computing and Systems Engineering (SCSE)*, vol. 9, 2026, pp. 1–6.

[11] N. Tiwari and S. K. Prasad, "Agentic ai-driven optimizing demand forecasting in retail systems with ai-based predictive analytics," in *2025 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, 2025, pp. 1–5.

[12] G. Radhakrishnan, P. P. Gaikwad, V. Pimplapure, I. A. Mohammed, A. D. Todmal, and S. H. V. Sanne, "Ai-based demand forecasting system for supply chain optimization," in *2025 Second International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM)*, 2025, pp. 1–7.

[13] A. Kumar, S. P. Singh, and G. Sakthi, "Ai-driven inventory forecasting pos system," in *2026 5th International Conference on Innovative Practices in Technology and Management (ICIPTM)*, 2026, pp. 1–6.

[14] S. Pungky and J. Wiratama, "Integrating machine learning based sales forecasting with odoo erp for automated inventory management in a retail company," in *2025 4th International Conference on Electronics Representation and Algorithm (ICERA)*, 2025, pp. 01–06.

[15] R. A. Perumal, A. Maurya, C. Stalin, A. K. Mathur, M. Sunitha, and N. Keshari, "Ai-driven predictive analytics and deep learning for demand forecasting and inventory optimization in supply chain management," in *Proceedings of Data Analytics and Management*. Cham: Springer Nature Switzerland, 2026, pp. 86–96.

[16] G. Theodoridis and A. Tsadiras, "Retail demand forecasting: A comparative analysis of deep neural networks and the proposal of lstmixer, a linear model extension," *Information*, vol. 16, no. 7, p. 596, 2025.

[17] S. Praveena and S. P. Devi, "Optimizing retail operations through hybrid machine learning and deep learning techniques in demand forecasting," *Cybernetics and Systems*, vol. 57, no. 1, pp. 1–43, 2026.

[18] S. Mansur, K. Sattar, S. E. Hosseini, S. Pervez, I. Ahmad, K. Saleem, and A. Z. Elhendi, "Sales forecasting for retail stores using hybrid neural networks and sales-affecting variables," *PeerJ Computer Science*, vol. 11, p. e3058, 2025.

[19] T. U. Aye, A. A. Neto, and E. R. da Silva, "Knowing your surroundings: Tactical retail demand forecasting through exogeneity-aware machine learning," *IEEE Engineering Management Review*, pp. 1–26, 2025.

[20] C. Riachy, M. He, S. Joneidy, S. Qin, T. Payne, G. Boulton, A. Occhipinti, and C. Angione, "Enhancing deep learning for demand forecasting to address large data gaps," *Expert Systems with Applications*, vol. 268, p. 126200, 2025.

[21] B. Mohanty, S. Meher, S. Pattnaik, R. Das, S. R. Mishra, and S. Tripathy, "Enhancing inventory management and demand forecasting in retail: Integrating machine learning with association rule mining," in *2025 International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU)*, 2025, pp. 1–6.

[22] Y. Yang, M. Wang, J. Wang, P. Li, and M. Zhou, "Multi-agent deep reinforcement learning for integrated demand forecasting and inventory optimization in sensor-enabled retail supply chains," *Sensors*, vol. 25, no. 8, p. 2428, 2025.

[23] G. S. Sajja, M. K. Meesala, S. R. Addula, and P. Ravipati, "Optimizing retail supply chain sales forecasting with a mayfly algorithm-enhanced bidirectional gated recurrent unit," *SN Computer Science*, vol. 6, no. 6, p. 737, 2025.

[24] M. Pattnaik, S. K. Padhi, L. Panda, U. Naushad, R. K. Behera, A. Mishra, and A. Pattnaik, "Ai-driven sales forecasting in retail: Insights from big bazaar's sales data," *International Journal of Management Science and Engineering Management*, vol. 21, no. 1, pp. 1–14, 2026.

[25] H. Hu, D. Tan, P. Thaichon, B. Wang, and Z. Zhu, "Grid-based market sales forecasting for retail businesses using automated machine learning and geospatial intelligence," *Expert Systems with Applications*, vol. 284, p. 127869, 2025.

[26] M. I. Zulfiqar, A. Khalid, A. Siddig, M. J. Nawaz, and S. Saay, "Ai-driven smart shopping carts with real-time tracking and inventory forecasting for enhanced retail efficiency," *IEEE Access*, vol. 13, pp. 55 576–55 585, 2025.

[27] P. G. Giannopoulos, T. K. Dasaklis, I. Tsantilis, and C. Patsakis, "Machine learning algorithms in intermittent demand forecasting: A review," *International Journal of Production Research*, vol. 0, no. 0, pp. 1–43, 2025.

[28] H. Kagalwala, G. V. Radhakrishnan, I. A. Mohammed, R. R. Kothinti, and N. Kulkarni, "Predictive analytics in supply chain management: The role of ai and machine learning in demand forecasting," *Advances in Consumer Research*, vol. 2, pp. 142–149, 2025.

[29] M. L. Y. Lin, T. M. Hao, M. Mukred, and M. I. Nofal, "Recommended machine learning and deep learning models in improving sales forecasting across diverse industries: A review analysis," in *2025 1st International Conference on Computational Intelligence Approaches and Applications (ICCIAA)*, 2025, pp. 1–8.

[30] S. Punia, "Medium- to long-term demand forecasting in retail and manufacturing organizations: Integration of machine learning, human judgment, and interval variable," *Journal of Forecasting*, vol. 45, no. 1, pp. 122–134, 2026.

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[33] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794.

[35] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[37] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Computing and Applications*, vol. 27, no. 2, pp. 495–513, 2016.

[38] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[39] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.