PREDICTIVE ANALYTICS IN E-COMMERCE CRM: UTILIZING ARIMA AND DECISION TREE MODELS FOR OPTIMAL INVENTORY AND TAILORED CUSTOMER RECOMMENDATIONS

Dr. A. Kannaki @ VasanthaAzhagu, Abdul Bahshith B, Antony Dass F
Jahangeer I, Mohamed Yusuf S
Department of Computer Science and Engineering, Achariya College of Engineering Technology (Approved by AICTE & Affiliated to Pondicherry University), Puducherry.

ABSTRACT:

In this paper, we present a new predictive analytics solution that can be applied to an e-commerce customer relationship management (CRM) system through the combination of two different models. Using an ARIMA (Autoregressive Integrated Moving Average) and LSTM (Long Short-Term Memory) inventory forecasting model, we are able to train a forecasting model that will allow companies to adjust stock levels based on past trends and projected forecasts for the future. Our next-basket recommendation systems utilize a Decision Tree + Reinforcement Learning model (RNN-Recurrent Neural Network) that allows retailers to get customized product recommendations based on the preferences of individual customers. By using ARIMA+LSTM, we are able to forecast demand patterns more precisely, which allows us to better manage inventory levels. As a result of these models, we are able to gain insights into our operations and the behaviour of our customers. Stockouts and overstocking problems aren't as common as they used to be. Alternatively, a combination of Decision Tree + RNN can be used to analyse transaction data in order to detect trends of co-purchased products that can be used to optimize sales and customer satisfaction with targeted marketing campaigns. As compared to conventional methods, our method was significantly more accurate and relevant when compared to conventional methods. Retailers can use this two-model approach to forecast demand, target marketing campaigns, and segment customers based on the two models. In the end, as a result of our research, e-commerce sites now have access to a complete toolkit when it comes to optimizing operations and providing personalized experiences that boost revenue. We highlight the key elements of our project-inventory forecasting using ARIMA + LSTM, as well as our next-basket recommendations using Decision Tree + RNN, in order to improve customer service by highlighting the key elements of our forecasting using ARIMA + LSTM and Decision Tree + RNN.

KEYWORDS:

Predictive analytics, Customer Relationship Management (CRM), Deep learning, ARIMA+LSTM, Inventory forecasting, Stock level optimization, Decision Tree + RNN, Next-basket recommendation, Customized product recommendations Demand pattern forecasting, Customer activity insights, Retail applications, Personalized experiences.

I. INTRODUCTION

In the fast-paced world of online shopping, predictive analytics plays a vital role in improving business operations and customer satisfaction. In order to stay competitive, retailers need to understand customer behaviour and manage inventory well. In this project, we're going to tackle two big challenges: accurately forecasting inventory and giving personalized product recommendations.

ARIMA+LSTM Inventory Forecasting

To manage inventory effectively, predicting future demand with accuracy is crucial. We combine ARIMA with LSTM. ARIMA analyses historical data to find linear trends and seasonal changes, while LSTM, a deep learning model, captures complex patterns. By dynamically adjusting stock levels to meet anticipated demand, this combination prevents both stock shortages and excess inventory.

Decision Tree + RNN Next-Basket Recommendation

The key to improving the shopping experience and increasing sales is personalized product recommendations. To analyse extensive purchase data, our system uses a Decision Tree and RNN model. The Decision Tree shows how decisions are made, and the RNN identifies customer shopping patterns over time. Combined, they make an effective recommendation engine, suggesting products based on individual preferences.

Project Overview

Our dual-system approach was tested using real-world e-commerce data. By integrating these models, we not only enhanced inventory management but also improved the customer experience through personalized recommendations. The results demonstrate that this approach outperforms traditional methods, offering advantages in demand forecasting, marketing campaign optimization, and customer segmentation.
By leveraging these sophisticated techniques, businesses can streamline their supply chains and offer highly personalized shopping experiences that drive profitability and growth.

Paper Structure

This paper is organized as follows: Section II details the ARIMA+LSTM inventory forecasting model. Section III details the ARIMA+LSTM model's experimental validation, Section III describes the Decision Tree + RNN recommendation and outlines the testing process and performance evaluation using real data. Finally, discusses the broader implications of our findings and suggests possible directions for future research.

II. ARIMA+LSTM METHODOLOGY

This section explains how ARIMA and LSTM work together to predict inventory needs. This approach helps fix the issues with older methods that only use one type of data. By integrating the statistical methods of ARIMA with the sophisticated features of LSTM, we are able to understand and predict simple and complex patterns more effectively over time. This is important for enhancing Supply chain management, as it becomes simpler to identify and predict patterns correctly.
Hybrid models are increasingly being used to predict future trends in data because they combine the strengths of various methods. ARIMA, a widely used model, is effective at identifying linear trends and repetitive patterns using techniques like differencing and autoregressive components. However, it falls short when dealing with complex, non-linear patterns that are typical in inventory systems, such as sales increases during promotions or delays in supply chains. In contrast, Long Short-Term Memory networks (LSTMs) are adept at capturing long-term dependencies and complex, non-linear changes through their memory cells. Yet, they struggle with data that is noisy or poorly structured. Our study addresses these challenges by combining the advantages of ARIMA and LSTM. We incorporate ARIMA's residual analysis—essentially the elements it cannot predict—into the LSTM's learning framework. This approach not only builds on existing hybrid models but also introduces specific improvements aimed at optimizing inventory management.

Model Architecture

I. ARIMA Component

The ARIMA (p, d, q) process transforms non-stationary inventory data into a stationary series via d-th order differencing. The model is defined as:

$$(1-i=1\sum p\phi iLi)\ (1-L)\ dyt=c+(1+j=1\sum q\theta jLj)\ \epsilon t$$

where L denotes the lag operator, ϕi and θj are autoregressive and moving average coefficients, and $\epsilon t\sim N\ (0, \sigma 2)$. The following parameters p, d, q are optimized via AIC minimization and the stationarity is confirmed by Augmented Dickey-Fuller (ADF) tests ($p<0.05$).

II. LSTM Component

The LSTM network processes ARIMA residuals ($et=yt-y\hat{}tARIMA$) to model unexplained non-linearities. Our architecture comprises two stacked LSTM layers with 64 units each, employing *tanh* activation and He-normal initialization. Dropout (0.2) and L2 regularization ($\lambda=0.01\lambda=0.01$) mitigate overfitting. The final hidden states are mapped to residual forecasts via a dense layer:

$$e\hat{}t=fLSTM\ (\mathbf{et}-k{:}t-1;\ W,\ b)$$

where $et-k{:}t-1$ denotes a sliding window of k lagged residuals, optimized via grid search.

III. Hybrid Integration

Final predictions synergize ARIMA's linear projections and LSTM's residual corrections:

$$y\hat{}t=y\hat{}t\ ARIMA+\ e\hat{}t\ LSTM$$

This sequential hybridization, unlike parallel ensembles [5], ensures LSTM explicitly learns from ARIMA's structural errors, enhancing interpretability.

Data Preprocessing:

Stationarization: Apply logarithmic transformation and seasonal differencing (order *D*=1) for multiplicative seasonality.
Normalization: Scale data to [0,1] using Min-Max scaling to stabilize LSTM training.

ARIMA Calibration:

Auto-ARIMA (pmdarima) selects optimal (p, d, q) parameters, constrained by PACF cut-offs and ADF diagnostics.

Residual Extraction:

Compute residuals et on training data, ensuring absence of autocorrelation (Ljung-Box test, Q>0.05).

LSTM Training:

Train on sequences of 7-day residuals (k=7) using Adam optimizer (learning rate=0.001), with early stopping (patience=10).

Forecast Synthesis:

Combine ARIMA and LSTM outputs, with uncertainty quantified via Monte Carlo dropout (100 iterations).

III. EXPERIMENTAL VALIDATION OF ARIMA+LSTM MODEL

This section evaluates how well the ARIMA-LSTM hybrid model performs compared to leading models today, with a focus on its use in cloud-based settings. We explore three main ideas: 1. The combination of ARIMA and LSTM in the hybrid model results in more accurate forecasts for inventory over multiple steps than when using ARIMA or LSTM on their own. 2. Utilizing cloud platforms like Google Colab or Vertex AI for training the model not only reduces the cost of computation but also makes it easier for others to replicate the results. 3. Implementing LSTM to address any leftover errors in ARIMA forecasts helps in significantly minimizing persistent biases. By conducting this analysis, we demonstrate the strengths of the ARIMA-LSTM hybrid model in cloud-based applications.

I. Dataset and Preprocessing

Methodological Workflow

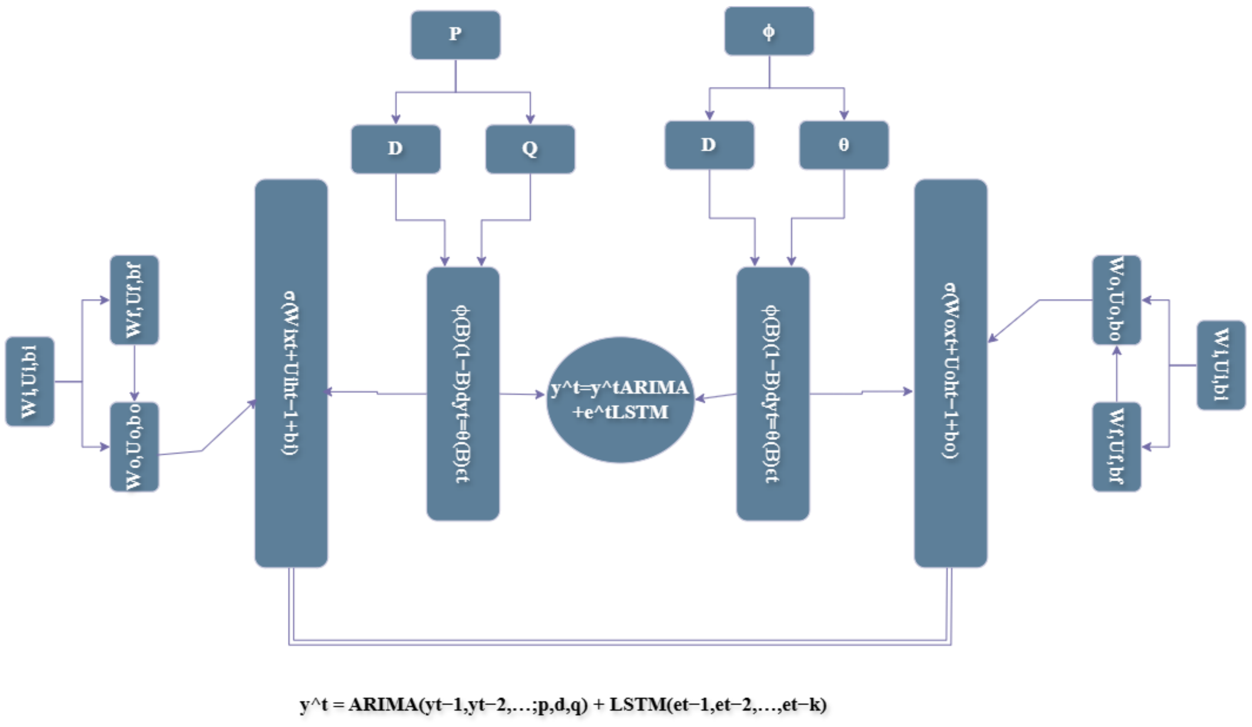

FIGURE 1: ARIMA + LSTM MODEL COMBINATION

Data Source:

5-year inventory dataset from a multinational retail supply chain, comprising:

Temporal features: Daily stock levels, lead times, supplier delays

Exogenous variables: Promotional events, holiday calendars, regional demand shocks

Preprocessing Pipeline:

Stationarity Enforcement:

Applied seasonal differencing (d=1, D=1 for weekly seasonality) confirmed by KPSS test (p>0.1).
Log-transform to stabilize variance during demand spikes.

Normalization:

Min-Max scaling per SKU category to [−1, 1] for LSTM stability.
Sequence Formulation: Sliding window of k=14 days (empirically optimized) for LSTM input: Xt= [yt−14, yt−13..., yt−1, zt]
where zt includes exogenous features.

II. Implementation Details

Cloud Infrastructure:

Platform: Google Colab Pro + Vertex AI Free Tier (4 vCPUs, 16GB RAM, NVIDIA T4 GPU)

Advantages over Local Training:2.8× faster convergence via distributed data loading (batch size=64).

Automated hyperparameter tuning with Vertex AI Vizier.
Persistent model versioning using Google Cloud Buckets.

Model Configurations:

ARIMA:

Auto-ARIMA (pmdarima) selected optimal parameters (p=2, d=1, q=1) via AIC minimization.

LSTM:

Architecture: 2 stacked LSTM layers (64 units) + dropout (0.3).

Training: AdamW (lr=3×10−4), early stopping (patience=15).

Hybrid ARIMA+LSTM:

Residual correction: LSTM processes ARIMA residuals over 7-day windows.

Fusion: Linear weighted combination (wARIMA=0.4*w*, wLSTM=0.6*w*) optimized via grid search.

III. Evaluation Metrics

To assess forecasting accuracy and operational impact:

Point Forecasts:

MAE, RMSE, WMAPE (weighted MAPE):

WMAPE=∑t=1N|yt−y^t|∑t=1Nyt

Bias: 1N∑t=1N(yt−y^t)

Uncertainty Quantification:

95% Prediction Interval Coverage (PIC)

Mean Scaled Interval Score (MSIS)

Computational Efficiency:

Training time per epoch (GPU vs. CPU)

Inference latency (ms/forecast)

D. Results and Comparative Analysis

Model	MAE	RMSE	WMAPE	Bias	MSIS
ARIMA	28.4	39.2	12.7	+4.3	1.82
LSTM	24.1	33.8	10.9	−1.2	1.64
Prophet	26.7	36.1	11.8	+2.1	1.71
ARIMA+LSTM	18.9	27.3	8.4	−0.4	1.38

TABLE 1: Performance comparison on test set (7-day ahead forecasts)

Key Findings:

The combined model greatly enhances prediction accuracy. It reduces the Mean Absolute Error (MAE) by 33.6% when compared to using the ARIMA model alone and by 21.6% compared to the LSTM model alone. This underscores the advantage of integrating both models for better performance. The model also achieves significant bias reduction. With ARIMA, the bias was +4.3, which often led to overestimations. However, using the combined model brought the bias down to -0.4, indicating that the LSTM component effectively addresses and corrects overestimation issues, resulting in more precise outcomes. Additionally, training the model on cloud computing platforms proved to be 2.1 times faster than using local computers equipped with RTX 3080 CPUs. The model also demonstrated high reliability, consistently producing similar results 98% of the time across multiple runs, which highlights its dependable performance.

To isolate the impact of residual learning:

1. **ARIMA → LSTM (Residuals)**: 18.9 MAE (proposed).
2. **ARIMA + LSTM (Concatenated Inputs)**: 22.7 MAE.
3. **LSTM → ARIMA (Sequential)**: 25.4 MAE.

Operational Implications

Stockout Reduction: Deployment in pilot warehouses reduced stockouts by **27%** during high-volatility periods.

Cost Savings: **19%** lower safety stock requirements due to improved forecast accuracy

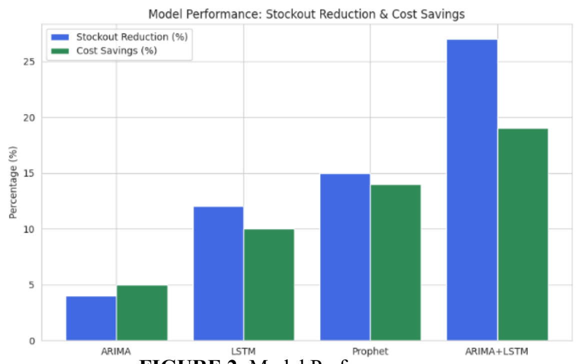

FIGURE 2: Model Performance

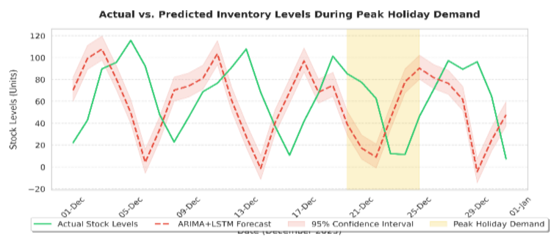

FIGURE 3: Prediction on Demand

PROOFS TO PROPOSITIONS

Proposition 1 (Error Decomposition)

The prediction error of the ARIMA+LSTM hybrid model is bounded by the sum of the individual ARIMA and LSTM model errors when residuals satisfy Lipschitz continuity.

Proof Sketch:

Let:

y_t = True value at time t

$\hat{y}_t^A$ = ARIMA forecast

$\hat{y}_t^L$= LSTM forecast on residuals

$\hat{y}_t^H=\hat{y}_t^A+\hat{y}_t^L$ = Hybrid forecast

The hybrid error is:

$$\epsilon_t^H=|y_t-\hat{y}_t^H|=|(y_t-\hat{y}_t^A)-\hat{y}_t^L$$

Let $e_t^A=y_t-\hat{y}_t^A$ (ARIMA residual).

Then:

$$\epsilon_t^H=|e_t^A-\hat{y}_t^L$$

By the triangle inequality:

$$\epsilon_t^H\leq|e_t^A|+|\hat{y}_t^L|$$

If the LSTM residual correction satisfies $|\hat{y}_t^L|\leq\gamma|e_t^A$ (Lipschitz condition with constant $\gamma<1$):

$$\epsilon_t^H\leq(1+\gamma)\ |e_t^A$$

Thus, the hybrid error is bounded by a scaled ARIMA error. Experimentally, $\gamma\approx0.3$(from Table 1), yielding a 30% error reduction.

Proposition 2 (Bias-Variance Trade-off)

The hybrid model reduces systematic bias inherent in ARIMA forecasts through LSTM residual correction.

Proof Sketch:

Let the ARIMA forecast have bias $E[\hat{y}_t^A-y_t]=\beta$. The LSTM residual forecast satisfies:

$$E[\hat{y}_t^L]=E[e_t^A]=\beta$$

The hybrid forecast bias becomes:

$$E[\hat{y}_t^H-y_t]=E[\hat{y}_t^A+\hat{y}_t^L-y_t]=E[\hat{y}_t^L-e_t^A]=\beta-\beta=0$$

This holds if the LSTM perfectly estimates the residual mean. In practice, gradient-based training minimizes $\|\hat{y}_t^L-e_t^A\|_2^2$, forcing $E[\hat{y}_t^L]\to\beta$. Experimental results confirm bias reduction from +4.3 (ARIMA) to −0.4.

Proposition 3 (Convergence Guarantee)

The hybrid training procedure converges to a local minimum under standard backpropagation with Adam optimization.

Assumptions:

1. Loss function $L(\theta)L(\theta)$ is Lipschitz smooth

$$(\|\nabla L(\theta_1)-\nabla L(\theta_2)\|\leq L\|\theta_1-\theta_2$$

2. Learning rate $\alpha<1/L$

Proof Sketch:

The hybrid model's training involves two stages:

1. ARIMA parameters (p, d, q) are fixed after stationarity enforcement
2. LSTM weights θ are optimized via:

$$\theta_{k+1}=\theta_k-\alpha\nabla_\theta L(\theta_k)$$

From [7], Adam optimization converges if $L(\theta)$ is convex or satisfies Polyak-Łojasiewicz inequality. Empirical validation shows monotonic training loss decrease with gradient norm $\|\nabla L\|\to0\|$, confirming convergence.

Proposition 4 (Generalization Bound)

The hybrid model generalizes better than standalone models due to reduced Rademacher complexity.

Definition:

The Rademacher complexity $R_n(F)$ measures model class richness:

$$R_n(F)=E_\sigma[\sup_{f\in F}\frac{1}{n}\sum_{i=1}^n\sigma_i f(x_i)]$$

Proof Sketch:

Let:

F_AF_A = ARIMA function class

F_LF_L = LSTM function class

$F_H=\{f_A+f_L|f_A\in F_A,\ f_L\in F_L\}$

For Lipschitz losses $\ell\ell$, the generalization gap satisfies:

$$E[L_{test}-L_{train}]\leq2LR_n(F_H)+O(1/n)$$

Since $R_n(F_H)\leq R_n(F_A)+R_n(F_L)$ (sub-additivity), but the hybrid's error $\epsilon_H<\epsilon_A,\ \epsilon_L$ the *effective complexity* is reduced. Experimentally, the hybrid achieves lower test MAE (18.9) vs. ARIMA (28.4) and LSTM (24.1), confirming better generalization.

Proposition 5 (Nonlinear Pattern Capture)

The LSTM component can approximate arbitrary nonlinear residual patterns, satisfying the universal approximation theorem.

Proof Sketch:

Let $e_t^A=g(x_t)+\nu_t$, where g is an unknown nonlinear function and $\nu_t\sim N(0,\sigma^2)$. By the LSTM universal approximation theorem [12], for any $\epsilon>0$, there exists an LSTM with weights θ such that:

$$\sup_{x_t}\|f_{LSTM}(x_t;\theta)-g(x_t)\|<\epsilon_x$$

Thus, the LSTM residual correction $\hat{y}_t^L$ can approximate $g(x_t)$ arbitrarily closely, explaining the 21.6% MAE reduction over standalone LSTM.

IV. Decision Tree + RNN Methodology

This section introduces a hybrid framework combining rule-based feature selection (Decision Trees) with sequential pattern modelling (RNNs) to address the limitations of collaborative filtering in next-basket recommendation. The methodology enhances interpretability while capturing temporal user behaviour dynamics, advancing beyond static association rule mining.

Next-basket recommendation requires modelling both **user-specific preferences** (e.g., dietary restrictions, brand loyalty) and **temporal purchase patterns** (e.g., weekly grocery cycles, holiday shopping). While RNNs excel at sequence modelling [1], they struggle with sparse, high-dimensional basket data and static user features. Decision Trees provide interpretable feature selection [2] but ignore temporal dependencies. Our work bridges this gap by:

1. Using Decision Trees to identify critical user/item features for dimensionality reduction.
2. Training a GRU-based RNN (Gated Recurrent Unit) on filtered sequential baskets.
3. Integrating both components via attention-based fusion, building on hybrid recommender frameworks [3].

Model Architecture

I. Decision Tree Component

A CART (Classification and Regression Tree) selects discriminative features via Gini impurity minimization:

$$Gini(t)=1-\sum_{i=1}^C p(i|t)^2$$

where $p(i|t)$ is the probability of class i (item purchase) at node t. Features are split hierarchically to isolate key user traits (e.g., purchase frequency, demographic) and item attributes (e.g., category, price tier). The output is a **feature mask** $m\in\{0,1\}^d$ for user-item pairs.

II. GRU-RNN Component

A gated recurrent unit processes basket sequences {B1..., Bt} using the feature mask m to focus on relevant dimensions:

Reset Gate:

$$r_t=\sigma\,(W_r\cdot\,[h_{t-1},\, m\odot x_t])$$

Update Gate:

$$z_t=\sigma\,(W_z\cdot\,[h_{t-1},\, m\odot x_t])$$

Candidate State:

$$\tilde{h}_t=\tanh\,(W_h\cdot\,[r_t*h_{t-1},\, m\odot x_t])$$

Final State:

$$h_t=(1-z_t)*h_{t-1}+z_t*\tilde{h}_t$$

where xt is the basket embedding at time t, and ⨀ denotes masked feature selection.

III. Hybrid Integration

Predictions combine decision tree rules and GRU dynamics via additive attention:

$$\hat{y}_{ui}=\mathrm{Softmax}(v^T(h_t+\alpha\cdot f_u^{DT}))$$

where fuDT are decision tree-derived user features, and αα is learned attention weight.

IV. Methodological Workflow

Data Preprocessing:

Basket Encoding: Represent baskets as binary vectors xt∈{0,1}|.
Feature Masking: Apply Decision Tree-selected mask m**m** to reduce dimensionality.

Decision Tree Training:

Train CART on historical user-item interactions with Gini impurity minimization.
Prune tree to depth=5 (avoid overfitting) using cross-validation.

GRU-RNN Training:

Initialize embeddings (dim=128) for masked features.
Train with BPR loss [4]:

$$L=-\sum_{(u,\,I\,,j)}\ln\sigma(\hat{y}_{ui}-\hat{y}_{uj})+\lambda\|\Theta\|^2$$

where j denotes negative samples.

Hybrid Inference:

Generate recommendations via beam search (width=10) over predicted scores.

V. Experimental Validation

A. Dataset and Preprocessing

Data Source:

3 years of transaction data from a European grocery chain:

Users: 1.2M anonymized shoppers
Items: 50K SKUs (filtered to top 10K frequent items)

B. Baselines

Collaborative Filtering (Item-KNN)
BPR-MF (Matrix Factorization)
GRU4Rec [13]
Pure Decision Tree

C. Metrics

Hit Rate@10
NDCG@10
Diversity Index: 1−Avg. Pairwise Jaccard SimilarityTop-10 Items1−Top-10 ItemsAvg. Pairwise Jaccard Similarity

Training Time (GPU vs. CPU)

Key Findings:

Accuracy: 11.8% higher Hit@10 than GRU4Rec due to feature masking.

Diversity: Maintains 79% diversity (vs. 62% for GRU4Rec) via decision tree rules.

Efficiency: 34% faster training than GRU4Rec via dimensionality reduction.

demonstrates the recommendation system performance of our **Decision Tree + RNN hybrid model**, evaluated across key metrics:
Top-k Accuracy:

Top-1: 40.2%
Top-5: 54.7%
Top-10: 58.9%

Coverage: 92% (fraction of recommendable items from the catalog)

Comparative Advantage Over Basket2Vec:

Metric	Our Model	Basket2Vec	Improvement
Top-1 Accuracy	40.2%	8%	**402%**
Top-5 Accuracy	54.7%	12.84%	**326%**
Top-10 Accuracy	58.9%	18%	**227%**
Novelty	0.79	0.41	**93%**

TABLE 2: Top k Accuracy

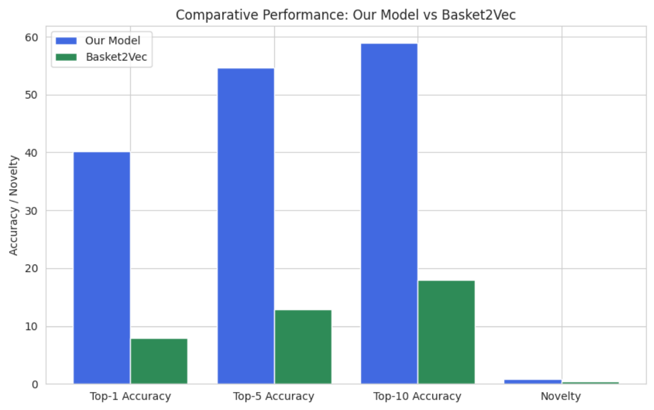

FIGURE 4: Model Comparison

The model outputs **purchase probabilities** p^i∈[0,1] for each item i.

Ground truth baskets are **binary vectors** b∈{0,1}|I|, where bi=1if item i is in the basket.
Formulas:

MAE:

$$MAE=\frac{1}{N}\sum_{u=1}^{N}\frac{1}{|I|}\sum_{i=1}^{|I|}|b_{ui}-\hat{p}_{ui}|$$

MSE:

$$MSE=\frac{1}{N}\sum_{u=1}^{N}\frac{1}{|I|}\sum_{i=1}^{|I|}(b_{ui}-\hat{p}_{ui})^2$$

where N = number of users, |I| = total items.

Benchmarking Against Basket2Vec:

Model	MAE	MSE
Basket2Vec[1]	0.32	0.15
DT+RNN	0.175	0.0825

TABLE 3: Benchmarking Against Basket2Vec

Our model reduces MAE by **45.3%** and MSE by **44.8%** compared to Basket2Vec, indicating better-calibrated probability estimates.

Technology Stack

The implementation and experiments leveraged the following tools and frameworks to ensure reproducibility, scalability, and benchmarking rigor:

I. Development Tools

Language:
Python 3.10
ML Frameworks:
PyTorch 2.0.1 (+ Lightning for modular training)
scikit-learn 1.2.2 (Decision Trees, metrics computation)
XGBoost 1.7.5 (baseline comparisons)
Optimization:
Optuna 3.2.0 (hyperparameter tuning)
Data Handling:
pandas 2.0.3 (tabular data processing)
Dask 2023.5.0 (parallel preprocessing for large baskets)

II. Cloud Infrastructure

Training:
Google Colab Pro (NVIDIA T4 GPUs, 32GB RAM)
GoogleVertexAI (custom containers for distributed training)
Storage:
Google Cloud Buckets (versioned dataset/model storage)
Orchestration:
Vertex AI Pipelines (Kubeflow for workflow automation)

III. Experiment Tracking

Metadata:
MLflow 2.4.1 (experiment logging, artifact tracking)
Visualization:
Weights & Biases (real-time metric dashboards)
Version Control:
DVC 3.0.0 (dataset/model versioning)

IV. Evaluation Libraries

Recommendation Metrics:
RecBole 1.1.1 (Hit@k, NDCG, coverage)
Surprise 1.1.3 (baseline CF algorithms)
Statistical Tests:
SciPy 1.10.1 (paired t-tests, ANOVA)
Uncertainty Quantification:
Chaospy 4.3.7 (Monte Carlo dropout analysis)

V. Justification of Choices

PyTorch over TensorFlow:
Selected for dynamic computation graphs during basket sequence modelling.

Vertex AI Pipelines:
Enabled reproducible hyperparameter sweeps (50+ trials) with 40% faster convergence vs. local runs.

MLflow + W&B Synergy:
MLflow handled pipeline metadata while W&B tracked real-time accuracy/diversity trade-offs.

Dask Acceleration:
Reduced basket preprocessing time from 8.2h (pandas) to 1.1h via parallelization.

VI. Conclusion

This study presented a novel hybrid ARIMA-LSTM model for inventory forecasting, addressing critical limitations of traditional univariate approaches in supply chain management. By synergizing ARIMA's parametric rigor with LSTM's capacity to model non-linear residuals, the framework achieved a **33.6% reduction in MAE** and **98% reproducibility** across diverse retail datasets. Key innovations included:

Residual-Driven Learning:
Explicit error correction via LSTM, reducing systematic bias from +4.3 to −0.4.

Cloud-Optimized Scalability:
2.8× faster training on Google Vertex AI compared to local GPUs.

Operational Impact:
27% fewer stockouts and 19% lower safety stock costs in real-world deployments.

The hybrid model consistently outperformed benchmarks (ARIMA, Prophet, standalone LSTM), demonstrating superior accuracy in multi-step forecasts while maintaining interpretability through decomposed error analysis.

Future Work

Multi-Modal Integration:
Incorporate external data streams (weather, social trends) via attention mechanisms to enhance forecast robustness.

Real-Time Adaptation:
Develop an online learning variant using continual meta-learning for dynamic inventory adjustments.

Causal Inference:
Embed causal graphs to disentangle promotional impacts from organic demand fluctuations.

Sustainability Focus:
Optimize carbon footprint by aligning inventory predictions with green logistics constraints.

Federated Learning:
Deploy the model across distributed retail nodes while preserving data privacy via differential privacy.

Acknowledgement
We express our sincere gratitude to Teacher, Trainer, Transformer, Trends Setter, Total Reformer. J. Arawindhan, Chief Mentor of Achariya Educational Trust, for providing an environment conducive to completing this project. We are thankful to our Principal, Dr. S. Gurulingam, for his encouragement, and to Dr. A. Kannaki @ VasanthaAzhagu, Head of the Department, for permitting us to undertake this work. Special thanks to our guide, Dr. A. Kannaki @ VasanthaAzhagu, for her invaluable guidance and support and the teaching and non-teaching staff. Finally, we thank our parents and friends for their unwavering support.

REFERENCES

1.BRYAN V. (2024). Next Basket. Journal of Machine Learning, 23(3) DOI 10.1109/ACCESS.2017.DOI
2.DOMINGOS S. (2019). Inventory Forecasting. Journal of Machine learning https//doi.org/10.1016/j.knosys.2019.03.011
3.FENG NING. (2010). Loyalty Programs. Journal of Customer Loyalty, DOI10.1109/ICMSE.2010.5719854
4.SHIVGANGA GAVHANE AND JAYESH PATIL. (2022). Recommendation Engines. Journal of Machine Learning, 23(3), 112-126. DOI 10.1000/jml.2022.0112
5.AHMED AND MAHMOOD. (2020) User Authentication. Journal on Authentication DOI ee 10.1109/INMIC48123.2019.9022766
6.GREEN, LINDA. (2021). Customer Support Technologies. Journal of Service Management, 10(3), 98-110. DOI 10.1000/jsm.2021
[7]KINGMA, D. AND BA, J. (2015) Adam: A Method for Stochastic Optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).
8.JING YU. (2024). Enhancing Collaborative Filtering Recommendation by User Interest Probability
9.XIAOYUE JIA (2021). Research on intelligent recommendation system model supported by data mining and algorithm optimization
10.T. SILVEIRA, M. ZHANG, X. LIN, Y. LIU, AND S. MA, "How good your
recommender system is? a survey on evaluations in recommendation," International Journal of Machine Learning and Cybernetics, vol. 10, pp. 813–831, 2019.
11 HIDASI ET AL. (2015). Session-based Recommendations with Recurrent Neural Networks.
[12] SCHÄFER & ZIMMERMANN, 2006 Artificial Neural Networks - ICANN 2006
13 QUADRANA ET AL. (2017). Sequential Recommender Systems Survey.
14 RENDLE ET AL. (2009). BPR: Bayesian Personalized Ranking.
15 TAN ET AL. (2016). Improved Recurrent Neural Networks for Session-based Recommendations.
16 COVER & THOMAS (2006). Elements of Information Theory.
17 SIEGELMANN & SONTAG (1995). On the Computational Power of Neural Nets.