University of West Florida

Capston Project In Data Science IDC-6940 | Professor Dr. Shusen Pu

Edwin Quijano

Monday, September 29, 2025

Proposal

Capturing Trends for Accurate Forecasting: An Intelligent Inventory Optimization

The focus of this project is on building intelligent systems that improve inventory forecasting and replenishment decision-making. Key questions include: Which products are likely to face stockouts or excess inventory? How can we use reinforcement learning to dynamically adjust reorder policies based on demand fluctuations and supplier performance? These questions matter because effective inventory management reduces operational costs, prevents lost sales, and promotes sustainability by minimizing waste.

1. Research Problem

Given historical transactions and inventory records, the goal of the project is to find policies that minimize expected long-run inventory costs while achieving target service levels under demand and lead-time uncertainty. For each SKU–location pair and time period t, decide order quantity a_t to balance:

- Shortage costs (lost sales, expediting, penalty costs),
- Holding costs (capital, storage, obsolescence),
- Ordering costs (fixed + variable), and
- Service constraints (e.g., ≥ 95 fill rate or $\leq X$ stockout probability).

Research questions.

- How much incremental performance does an ML-augmented forecasting model add versus classical statistical baselines for intermittent/seasonal demand?
- What is the impact of lead-time variability and supplier reliability on learned reorder policies?
- Which features and modeling choices (hierarchical aggregation, promotions, calendar effects) consistently improve out-of-sample performance across categories?

2. Objectives & Deliverables

1. Forecasting module with feature store, hyperparameter search, and backtesting; outputs demand distributions or quantiles per SKU–location–week.
2. Control module learning replenishment policies via deep RL; supports service-level constraints and capacity.
3. Simulator calibrated from data (lead-time distributions, ordering calendars, MOQs, capacity).
4. Evaluation suite comparing baselines, and full pipeline; statistical significance tests and business KPI dashboards.

3. Methods

The methodology for Intelligent Inventory Optimization combines deep learning and reinforcement learning to create an intelligent inventory management system. The process begins with understanding the business goals to optimizing inventory replenishment decisions. The Global Product Inventory Dataset 2025 is then explored to identify relevant features such as sales quantity, stock levels, reorder quantities, and lead times. Next, the data is cleaned and enriched through feature engineering, including time-based variables like rolling averages and reorder point calculations. For demand forecasting, deep learning models such as LSTM and CNN-LSTM are trained to capture temporal sales patterns. Simultaneously, a reinforcement learning agent using a Deep Q-Network (DQN) is developed in a simulated inventory environment to learn optimal restocking policies. These models are evaluated using metrics such as MAE and RMSE for forecasting and cost-reduction or service level for the reinforcement learning agent. The final solution aims to deliver accurate forecasts and adaptive inventory strategies that reduce costs and avoid stockouts.

3.1 Data Engineering & Feature Pipeline

Entity definition: SKU–location (warehouse) time series at weekly granularity can extend to daily where stable.

Cleaning: duplicate transactions; align time buckets; impute small gaps with Kalman/locf; remove impossible negatives.

Feature families

Temporal: lags, rolling means/medians, EWMA, STL/Loess seasonality signals; holiday/week-of-year; price/promo flags.

Inventory state: on-hand, in-transit, backorders, open POs, reorder level, safety stock proxy.

Supply risk: empirical lead-time distribution, supplier reliability, late-delivery rate.

Hierarchical signals: category/brand/warehouse aggregates (bottom-up & top-down reconciliation).

Train/validation splits: rolling-origin evaluation, avoiding leakage from future inventory states.

3.2 Forecasting Models

Baselines

Naïve/seasonal naïve; Croston-style for intermittent demand; SES/Holt-Winters; ARIMA/SARIMA; simple Prophet-style decomposition for seasonal promotion-heavy series.

Machine learning

Tree-based: XGBoost, LightGBM with monotonic constraints where appropriate; quantile objectives to get predictive intervals.

Deep time-series: LSTM/GRU; CNN-LSTM for local temporal patterns; optional attention/Temporal Fusion Transformer-style layers if time permits.

Hierarchical reconciliation: MinT or simple proportional reconciliation across SKU/category/warehouse.

Loss & metrics

Point: RMSE/MAE/MAPE (MAPE only when non-zero demand).

Probabilistic: Pinball loss across quantiles; WIS/CRPS for distributional accuracy.

Decision-aligned: newsvendor cost under candidate service levels to test whether better forecast translates to lower expected cost.

3.3 Inventory Control via Reinforcement Learning

Inventory MDP.

State s_t: on-hand, pipeline (by age), backorders, recent demand features, calendar, supplier state, capacity.

Action a_t: order quantity (bounded; may include reorder timing).

Transition: stochastic demand and lead times; receipt of aged pipeline orders; backorder fulfillment.

Reward r_t: −(holding + ordering + shortage + penalty for service constraint violations); optional carbon/aging costs.

Algorithms.

Start with DQN/DDQN on discretized actions; extend to Dueling DQN. For continuous actions or larger action spaces, test PPO or SAC.

Safety constraints: reward shaping and Lagrangian methods to enforce service-level target; or convert to constrained RL with dual variables.

Baselines for control

Heuristics: (s, S) policy, (R, Q) policy, and newsvendor-derived base-stock.

Stochastic optimization: myopic expected-shortage cost minimization using forecast quantiles.

Evaluation

Rolling policy evaluation in simulator with bootstrapped confidence intervals.

KPIs: total cost per unit time; fill rate/customers served; stockout frequency; average & percentile on-hand; turns; cash tied up.

Ablations: tree vs deep forecasters; perfect-forecast oracle to upper-bound achievable control performance; no-uncertainty lead time vs empirical distribution.

3.4 Statistical Testing & Robustness

Diebold–Mariano tests for forecast accuracy comparisons.

Paired bootstrap for cost/KPI differences.

Stress tests: demand spikes, lead-time shocks, promotion bursts, cold-start SKUs.

Fairness/coverage: per-category error parity; calibration curves for predictive intervals.

3.5 Lead-Time & Supply Modeling

Empirical distributions: Fit discrete lead-time distributions per supplier–lane using histogram or kernel smoothing over observed purchase order receipts. Capture multimodality (e.g., biweekly sailings) via mixture weights. Store quantiles (P50, P80, P95, P99) and an on-time delivery rate.

Calendars & batching: Respect non-working days and cut-offs; enforce MOQs, case-pack multiples, and order calendars (e.g., orders only Mondays). Model partial shipments and cancellations with probabilities estimated from history.

Reliability features: Late delivery rate, average delay conditional on lateness, and a supplier "reliability score" that feeds both forecasting covariates and the policy state.

3.6 Inventory Simulator

Implementation of a discrete-time simulator at the SKU–warehouse grain. Each period executes: receive pipeline, realize stochastic demand, fulfill orders/backorders, place a new order, and accrue costs (holding, ordering fixed/variable, shortage/backorder, optional obsolescence).

The simulator: Supports capacity, MOQs, order calendars, and shelf-life limits.

3.7 Forecasting Details

Targets: For each horizon h train horizon-specific models and optionally a direct multi-horizon model. Produce point forecasts and quantiles (e.g., 10th/50th/90th/95th) to inform safety stock and service-constrained decisions.

Cross-validation: Rolling-origin splits (e.g., 6 windows). Models are selected via multi-objective score (average pinball loss at key quantiles + RMSE), with ties broken by calibration error (PI coverage vs nominal).

Feature governance: Maintain feature catalogs with data lineage and drift checks; freeze feature sets per experiment to ensure comparability.

3.8 Decision Policies & RL Training

Baselines

(s, S) and (R, Q) policies tuned via grid search against target service (e.g., 95%).

Myopic stochastic optimization that orders to a quantile target given the predictive distribution; strong non-RL baseline for risk-averse control.

RL Design

State: on-hand, pipeline by age bucket, backorders, inventory position, recent demand features (lags/rolls), season/promo flags, lead-time stats, capacity flags.

Action: discrete order quantities on a grid (0, MOQ, 2×MOQ, …, S_max). For continuous control, switch to PPO with squashed Gaussian outputs.

Reward: negative total cost with a Lagrangian penalty for service-level violations; the dual variable is updated every N steps to target the desired fill rate.

Exploration: epsilon-greedy decay or parameter noise; prioritized replay for DQN variants.

Stability: target networks, gradient clipping, delayed policy updates (for actor–critic), and reward normalization.

Train RL for a fixed number of environment steps per SKU-equivalent, checkpoint best models by validation cost and service.

Evaluate on held-out periods and stress scenarios (demand spikes, delayed receipts). Report mean/median cost, fill rate, stockout frequency, inventory turns, and cash tied up with 95% bootstrap CIs.

4. Preferred Programming Languages & Tools

Primary: Python 3.11+ (NumPy, pandas, scikit-learn, XGBoost/LightGBM, TensorFlow/Keras or PyTorch, statsmodels, orca/rl implementations via Stable-Baselines3/TF-Agents), Hydra for configs.

Secondary: R (forecast, fable) for baseline sanity checks; SQL (DuckDB/Postgres) for data prep; Jupyter for EDA; MLflow/Weights & Biases for experiments; Plotly/Matplotlib for visualization; Poetry/Conda for env management; Docker for reproducibility.

5. Dataset

Name: Global Product Inventory Dataset 2025 (Kaggle).\ Source/Access: Kaggle dataset by Keyush Nisar; accessed via Kaggle API and stored locally as CSV/Parquet. Size & scope: >100,000 rows spanning multiple product categories, warehouses, suppliers; includes time-stamped records enabling time-series analysis. Key variables (anticipated): date, product_id, category, warehouse_id, supplier_id, sales_qty/demand, on_hand/stock_level, in_transit, reorder_level, lead_time_days, backorders, price, promotion_flag. Use in pipeline:

Create SKU–warehouse panels; 2) engineer temporal/supply features; 3) fit forecasters with rolling backtests; 4) calibrate simulator (empirical demand & lead-time distributions); 5) train and evaluate control policies; 6) report KPIs. Ethics & licenses: Follow Kaggle terms; avoid re-sharing raw data; document transformations and ensure privacy compliance.

6. Work Plan & Milestones (7 weeks)

- Week 5 - 6: Data audit, schema, feature store, EDA, baseline forecasts; initial report.
- Week 7: ML forecasters (GBTs, LSTM/GRU), hyperparameter search, probabilistic outputs.
- Week 8: Hierarchical reconciliation; decision-aligned loss experiments.
- Week 9: Build calibrated inventory simulator; validate against historical KPIs.
- Week 10: RL control (DQN→DDQN→PPO), constraint handling, sanity checks vs (s, S).
- Week 11: Integrated evaluation; ablations; robustness/stress tests.
- Week 12: Dashboard & documentation; model card; draft paper. Polish results; finalize slides & manuscript.

7. Risks, Limitations, and Mitigations

Data quality & intermittency: Use Croston/Tsb baselines and zero-inflated models; aggregate to weekly buckets; borrow strength hierarchically.

Non-stationarity: rolling re-training; drift detectors; decay weights.

Simulation fidelity: validate with backtests and historical policies; sensitivity analyses on lead-time distributions.

Compute/time: prioritize trees + simple RNNs first; scale deep models only if incremental value justifies.

Overfitting to cost weights: scenario analysis with multiple cost vectors; report Pareto frontiers of (cost, service).

8. Expected Contributions

A clear comparison of forecast accuracy vs decision quality, demonstrating when better forecasts matter.

An open, modular prototype that can be adapted to new SKUs/warehouses and cost regimes.

Practical guidance on when to use tree-based versus deep time-series models and when RL beats tuned classical policies.

9. Evaluation Metrics & Reporting

Forecast: RMSE/MAE, MAPE (filtered), pinball loss, coverage of PIs.

Control: total cost, service level (fill rate), stockout frequency, average inventory & turns, cash tied up.

Statistical confidence: DM tests, paired bootstrap.

Business reporting: category-level scorecards; waterfall of cost savings vs baseline policy; SHAP for tree models; saliency/attention for deep models where applicable.

Appendix A: Brief Math Appendix

1. Inventory flow and cost (per period) $Sales_t = min(I_t, D_t); B_{t+1} = max(0, D_t - I_t); I_{t+1} = I_t - Sales_t + R_t. Cost_t = h * max(I_{t+1}, 0) + k * 1a_t > 0 + c * a_t + p * B_{t+1}$.

2. Newsvendor and base-stock Single period optimum: $q* = F_D^{-1}(c_s/(c_h + c_s))$ Periodic review order-up-to for service target eta: $S = \mu_L + z_e ta * \sigma_L$, where $\mu_L = E(sum_{i=1..L} D_{t+i})$ and $\sigma_L^2 = Var(sum_{i=1..L} D_{t+i})$

3. Quantile (pinball) loss For quantile τ in (0,1): $loss_t au(y, \hat{q}) = (\tau - 1y < \hat{q}) * (y - \hat{q})$. Prediction-interval coverage for $[\hat{q}_{alpha/2}, \hat{q}_{1-\alpha/2}]$ is the fraction of outcomes that fall inside the interval.

4. Diebold–Mariano test (multi-step) Let d_t be loss differentials across two models. $DM = mean(d_t)/sqrt(var_N W(d_t)/T)$, where $var_N W$ uses Newey–West autocovariances up to horizon h-1. Under $H0$: equal accuracy, DM ~ N(0,1) asymptotically.

5. Hierarchical reconciliation (MinT) With summing matrix S and error-covariance proxy W, reconciled forecasts are: y_tilde = S (S' W^{-1} S)^{-1} S' W^{-1} y_hat.

6. Constrained RL with service target Maximize discounted return subject to E[g(s,a)] >= eta. Lagrangian objective uses r_lambda = r + lambda * (g - eta). Bellman equation: Qlambda(s,a) = r_lambda(s,a) + gamma * E{s'|s,a}[max_{a'} Q_lambda(s',a')].

7. CVaR (tail risk) For cost X and level alpha: CVaR_alpha(X) = average cost in the worst (1 - alpha) quantile. Estimate via bootstrap by averaging the largest (1 - alpha) fraction of sample costs.

8. Bootstrap confidence intervals Resample SKUs with replacement B times; compute KPI each time; percentile CI is [alpha/2, 1 - alpha/2] quantiles of the bootstrap distribution.

9. Demand censoring heuristic When stockouts censor true demand, approximate D_hat_t = Sales_t + E[Unmet_t | stockout], estimated from nearby non-stockout periods or category peers. Sensitivity: report policy metrics with and without this correction.

Tentative References (selected)

Bastos, A. S. T. (2024). Machine learning in digital retail: Demand forecasting for inventory management in a sportswear company. ProQuest Dissertations & Theses.

Pasupuleti, V., Thuraka, B., Kodete, C. S., & Malisetty, S. (2024). Enhancing supply chain agility and sustainability through machine learning: Optimization techniques for logistics and inventory management. Logistics, 8(3), 73.

Polo-Triana, S., Gutierrez, J. C., & Leon-Becerra, J. (2024). Integration of machine learning in the supply chain for decision making: A systematic literature review. Journal of Industrial Engineering and Management, 17(2), 344.

Pramodhini, R., Kumar, S., Bhardwaj, S., Agrahari, N., Pandey, S., & Harakannanavar, S. S. (2023). E-commerce inventory management system using machine learning approach. ICDSNS 2023, 1–7.

Sathish, T., SaiKumar, D., Patil, S., Saravanan, R., Giri, J., & Aly, A. A. (2024). Exponential smoothing method vs gradient boosting for materials forecasting. AIP Advances, 14(6).