Machine Learning Methods for Retail

Introduction

This research investigates how to forecast next-season demand at the **Stock-Keeping Unit (SKU)–size–store** level while explicitly correcting for **Out-of-Stock (OOS)** censoring (i.e., sales = min(true demand, stock)). Traditional models misread OOS periods as low demand and distort the **size curve** (e.g., "XS doesn't sell" simply because it wasn't available). Our dataset is small, so methods must generalize with limited history, share information across related items/stores, and run quickly on a local machine.

Aim

Identify, compare, and select a bias-aware, small-data-friendly forecasting approach that delivers accurate, app-ready **size-by-SKU-by-store** predictions. We will evaluate candidate models, prioritize accuracy, and recommend a final model to implement.

Abbreviations

OOS: Out-of-Stock

SKU: Stock-Keeping Unit

WAPE: Weighted Absolute Percentage Error

MAPE: Mean Absolute Percentage Error

RNN: Recurrent Neural Network

LSTM: Long Short-Term Memory

GRU: Gated Recurrent Unit

TFT: Temporal Fusion Transformer

ETS: Exponential Smoothing (Error/Trend/Seasonal)

TKF: Tobit Kalman Filter

TETS: Tobit Exponential Smoothing

SHAP: SHapley Additive exPlanations

CV: Cross-Validation

Potential Machine Learning Models

Bayesian Hierarchical Models

Probabilistic approach that models **latent true demand** per SKU-size-store and treats observed sales as **censored by stock**. A **hierarchical** (multi-level) structure "shares strength" across related items/stores, which is ideal for small datasets. Use a **censored likelihood** so OOS periods don't drag demand down. (PyMC/Stan can do this directly.) Recent applied notes show censored-likelihood Bayesian models reduce bias vs. classical methods.

Censored Data: Observations truncated by a bound (here: stock).

Censored Likelihood: Log-likelihood accounts for the probability mass above the stock bound (e.g., PyMC pm.Censored).

Hierarchical Pooling (shrinkage): Partial pooling across SKUs/stores to stabilize small series.

Posterior Intervals: Uncertainty bands for safety stock, etc.

Pros	Cons
Handles OOS **by construction** → big bias reduction.	Needs modeling effort (priors, diagnostics).
Works well with **small data** (priors + pooling).	MCMC/VI can be slower (still fine for small data on M3).
Delivers uncertainty for planning.	

State-Space Models: Tobit Kalman Filter (TKF) & Tobit Exponential Smoothing (TETS)

State-space approach where **latent (unobserved) true demand** evolves over time and **observed sales** are **censored by stock**: **$sales_t = \min(true\_demand_t, stock_t)$**.

Tobit Kalman Filter (TKF) = Kalman Filter with a Tobit (censored) observation model.

Tobit Exponential Smoothing (TETS) = Exponential Smoothing with a Tobit layer, so you keep ETS speed/interpretability while fixing out-of-stock (OOS) bias.

State-Space Model: Represents level/trend/season as hidden "states"; updated each period with the **Kalman Filter (KF)**.

Tobit Model (Censoring): Observation is truncated at a bound (here, stock), i.e., we only observe up to the available inventory.

Truncated Normal: Likelihood used when observations hit the stock boundary.

Exponential Smoothing (ETS – Error/Trend/Seasonal): Classical forecasting family; TETS adds censoring logic on top.

Time Aggregation: Combine sub-daily signals to daily/weekly to correctly treat intra-day stock outs.

Pros	Cons
One-stage OOS correction; materially reduces downward bias.	Limited off-the-shelf tooling; often needs a small custom implementation.
Lightweight & fast (ETS heritage), great for small datasets.	Requires reliable **stock/OOS flags** (ideally stock-on-hand quantities).
Interpretable components (level/trend/season).	

Gradient-Boosted Trees (LightGBM / XGBoost)

Turn forecasting into supervised regression across all **Stock-Keeping Unit (SKU)**–size–store series: engineer features (lags, rolling stats, seasonality, product/store metadata, promotions/price, **in-stock flag**, **days-in-stock**, **sell-through**) and train a **global** model (one model for all series). **Light Gradient Boosting Machine (LightGBM)** and **Extreme Gradient Boosting (XGBoost)** are fast, accurate, and easy to ship.

Global Model: Single model trained on all series with identifiers and metadata → small series benefit via pooling.

Feature Engineering: Lags, moving averages, seasonal dummies, holiday flags, promo/price, inventory features.

Rolling-Origin Cross-Validation (CV): Time-aware CV for robust evaluation.

SHAP (SHapley Additive exPlanations): Explains feature contributions for interpretability.

Pros	Cons
Strong accuracy vs. complexity; **very fast** training/inference on laptop.	Does not natively "fix" censoring → best when paired with **OOS features** or **two-stage recovery**.
Handles many covariates; explainable via feature importance/SHAP.	Quality depends on good features (you must encode seasonality, promos, etc.).
Works well with **limited data** when trained globally.	

Recurrent Neural Networks (RNNs: LSTM / GRU)

Sequence models that learn temporal patterns directly from history and covariates. **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** can ingest multiple inputs (price, promo, stock mask, weather, etc.). Train **globally** so short series borrow signal from others.

Sequence-to-Sequence / Sequence-to-One: Predict full horizon or next step from a sliding window.

Embeddings: Dense vectors for high-cardinality IDs (SKU, store, category).

Regularization: Dropout, weight decay, early stopping to control overfitting.

Teacher Forcing: Training trick when predicting multi-step sequences.

Pros	Cons
Captures nonlinear and long-range dynamics; flexible with many covariates.	**Data-hungry**; easy to overfit on small datasets.
Global training shares information across series.	Slower to tune/train than trees; still needs OOS handling (mask or two-stage).
	Less interpretable without extra tooling.

Transformers & Modern Architectures (TFT, TimesNet)

Temporal Fusion Transformer (TFT) uses attention to blend static and time-varying covariates with interpretability hooks. **TimesNet** models temporal "patches" (2D variation) and excels at **imputation + forecasting** on many series. Both are typically trained as **global models** and pair well with explicit OOS handling or **two-stage latent demand recovery**.

Attention: Learns which time steps and features matter for each forecast.

Static vs. Time-Varying Covariates: TFT separates item/store attributes (static) from evolving drivers (price, promo, stock).

Positional Encoding / Patching: Lets the model learn periodicity; TimesNet converts time to structured 2D patches.

Quantile Loss: Train for full predictive distributions (P10/P50/P90).

Pros	Cons
Highest accuracy ceiling when you have many related series and rich covariates.	More compute and engineering; may be **overkill for tiny datasets**.
Can natively leverage complex drivers and long-range dependencies.	Still needs stock masks or a recovery stage to avoid censoring bias.
Plays well with **latent demand recovery** for OOS-heavy data.	Harder to maintain/deploy than trees for small teams.

Latent Demand Recovery (Two-Stage Pipeline)

Fix censoring **before** forecasting.

Stage-1 (Imputation): Estimate **latent (unobserved) demand** during out-of-stock periods using rules (baseline mix, in-stock-days scaling, substitution) or a learned imputer (e.g., TimesNet, simple Bayesian model).

Stage-2 (Forecast): Train your forecaster (LightGBM/TFT/ETS) on the **recovered demand** series; assume availability in the future horizon.

Out-of-Stock (OOS) Mask: Binary/continuous signal of availability used by the imputer and/or forecaster.

Sell-Through: Sold / (sold + ending inventory); helps spot constrained weeks.

Weighted Absolute Percentage Error (WAPE) / Mean Absolute Percentage Error (MAPE): Accuracy metrics.

Bias: Systematic over/under-forecast; goal is ~0% after recovery.

Pros	Cons
Works with almost any forecaster; **large bias reduction** when OOS is frequent.	Two artifacts to build/maintain (imputer + forecaster).
Transparent: you can audit Stage-1 adjustments and explain them.	Forecast quality depends on **imputation quality**.

Hierarchical Forecasting (Product/Store → Size)

Forecast at multiple levels (total product, store, **size**) and **reconcile** so lower levels add up to upper levels (**coherence**). Useful when sizes have sparse histories: forecast total, then allocate to sizes via a **learned size curve** (e.g., softmax) or a **Dirichlet** share model; or use bottom-up/top-down/middle-out strategies.

Coherent Forecasts: Reconciled so child forecasts sum to parents.

Top-Down / Bottom-Up / Middle-Out: Different reconciliation strategies.

Dirichlet (Compositional) Modeling: Size shares constrained to the simplex (sum to 1).

Minimum Trace (MinT) Reconciliation: Statistical method to optimally reconcile hierarchies.

Pros	Cons
Stabilizes **size curves** with limited data; aligns with retail planning.	Requires clean hierarchies and a little extra plumbing.
Ensures consistency across reporting levels (store → region → total).	If parent forecasts are biased, allocation inherits that bias (pair with OOS correction).

Sell-Through–Based Baseline (Supportive)

Simple, explainable control logic to diagnose and partially correct constraints. Use **sell-through** and **in-stock days** to scale observed sales to an **unconstrained** estimate; optionally apply a **substitution matrix** (e.g., XS demand spills to S/M when XS is OOS). Great as a sanity check and as a **Stage-1** starting point.

In-Stock Days: Number of days the item was available within the period; scale sales by 7/instock_days for weekly normalization.

Lost Sales: max(0, expected – actual) during constrained periods.

Substitution Matrix: Probabilities of demand shifting across sizes when a size is OOS.

Pros	Cons
Excel-friendly and **transparent**; easy to explain to stakeholders.	Heuristic; not optimal on its own.
Good baseline for **latent demand recovery** before ML.	Needs careful caps/guards to avoid over-uplifting.

ARIMA / SARIMA (Classical Time Series)

Autoregressive Integrated Moving Average (ARIMA) models capture autocorrelation and trends; **Seasonal ARIMA (SARIMA)** adds seasonal terms. Train one series per SKU-size-store or use external regressors. Works best when histories are moderately long and relatively clean (few structural breaks).

AR / I / MA: Autoregressive, differencing (Integrated), moving average components.

SARIMA: Seasonal ARIMA with seasonal AR/MA and seasonal differencing.

SARIMAX: SARIMA with e**X**ogenous regressors (price, promo, events).

Stationarity: After differencing, mean/variance stable over time.

Pros	**Cons**
Strong baseline; interpretable and fast.	One-series-per-model scales poorly; weak for very short histories.
With **SARIMAX**, can include covariates (promo/price/holiday).	**Does not handle censoring**; must pair with OOS recovery or masks.
	Limited capacity vs. ML/deep models for complex nonlinear effects.

Random Forest Regressor (Tree-Based, Bagging)

Decision-tree ensemble via **bagging** (bootstrap aggregation). Similar setup to Gradient-Boosted Trees (features + global model), but **averages** many trees instead of boosting. Often more robust but less sharp than boosting on tabular forecasting tasks.

Bagging: Train trees on bootstrapped samples; average predictions.

Out-of-Bag (OOB) Error: Built-in validation using samples not seen by a tree.

Feature Importance: Frequency/impact of feature splits across trees.

Pros	**Cons**
Robust, low-tuning, quick to try; decent accuracy with minimal fuss.	Typically underperforms **LightGBM/XGBoost** on tabular forecasting.
Works with small datasets; interpretable via feature importance.	Still needs OOS features or two-stage recovery to avoid censoring bias.

Comparison of Models in Use-Cases

Models	OOS Correction Quality	Small-Data Robustness	Accuracy Ceiling	Speed and Footprint	Implementation Complexity	Interpretability
Tobit Kalman Filter (TKF) / Tobit Exponential Smoothing (TETS)	High (built-in censoring)	High	Medium-High	High (fast)	Medium (light custom mode)	High (level/trend/season)
Bayesian Hierarchical (censored likelihood)	High (by construction)	High (pooling + priors)	High	Medium (ok for small data)	High (modeling effort)	Medium-High (probabilistic)
Two-Stage: Latent Demand Recovery → Light Gradient Boosting Machine (LightGBM)	High (via Stage-1)	High (global model)	High	Very High (fast)	Medium (two artifacts)	Medium-High (feature importances/SHAP)
Two-Stage: Latent Demand Recovery → Temporal Fusion Transformer (TFT)	High (via Stage-1)	Medium (Needs breadth)	Very High	Medium (heavier)	High	Medium (some interpretability)
Long Short-Term Memory (LSTM) / Gated Recurrent Unit (GRU)	Medium (needs masks/two-stage)	Medium-Low (overfit risk)	High (with data)	Medium	Medium-High	Low-Medium
Autoregressive Integrated Moving Average (ARIMA) / Seasonal ARIMA (SARIMA)	Low (no censoring)	Medium	Medium	High	Low-Medium	Medium-High
Random Forest (bagging)	Medium (needs masks/two-stage)	Medium-High	Medium	High	Low	Medium

Conclusion

Recommendation

If our priorities are **top predictive accuracy**, **small-data robustness**, and **explicit Out-of-Stock (OOS) handling**, the **Bayesian Hierarchical Model (BHM) with a censored likelihood** is the primary choice. It (1) models **latent true demand** directly, (2) **shares strength** across Stock-Keeping Unit (SKU), size, and store via partial pooling (critical when histories are short), and (3) treats **sales = min(true demand, stock)** natively through a censored likelihood. This combination typically yields **lower bias** and **better calibration** on limited data.

At the same time, **Tobit Exponential Smoothing (TETS)** and **Tobit Kalman Filter (TKF)** remain **excellent one-stage baselines**: fast, interpretable, and purpose-built for censoring. If you need **very low engineering overhead** and **high speed** with good accuracy, TETS/TKF is a strong practical alternative.

Choose Bayesian (censored hierarchical) when accuracy and small-data stability are paramount, and you can afford modest modeling effort.

Choose Tobit (TETS/TKF) when you want a **lean, fast, interpretable** solution that still corrects OOS in one shot.

Ideally, run **both**: ship Tobit as a transparent baseline, and adopt Bayesian as the accuracy-focused production model.

Potential Categorizing Feature Using Hierarchical Structure of the Models

Both shortlisted paths—**Bayesian Hierarchical (censored)** and **Tobit (TETS/TKF)** —naturally support a **hierarchical structure** across *store* → *region* → *chain* and *year* → *season* → *week*. That means the model can **interpret and forecast by sections of the timeline** (e.g., early/peak/late season, SS vs FW, specific years) while correcting for **Out-of-Stock (OOS)** bias.

Bayesian (censored, multi-level)

Uses **random effects** for store/season/year with **partial pooling**, so small or new segments borrow signal from the whole while keeping their own nuances. You get **per-segment size curves**, **time-slice effects** (e.g., early vs late season), and **credible intervals** for each slice—great for scenario planning and buyer storytelling.

Tobit State-Space (TETS/TKF)

Encodes **level/trend/season** as states and can **share seasonal templates** hierarchically across regions or store types. It cleanly separates **timeline sections** (e.g., seasonal indices by phase) and remains **OOS-aware** via the Tobit observation, delivering fast, interpretable per-group forecasts.

Acting as a Selling Point

By exploiting a hierarchical structure, both the Bayesian (censored, multi-level) and Tobit state-space models learn stable group effects across **store/region/chain** and **year/season/week**, plus time-slice patterns (early/peak/late season). The **Potential Categorizing Feature Using Hierarchical Structure** turns those learned effects into automatic, explainable cohorts (e.g., store-season segments) that get tailored size curves and allocations from day one—enabling credible **cold-start** forecasts, timeline-aware actions, and fewer stockouts/overstocks. Because categories and effects are explicit and auditable, planners gain trust, re-plans stay fast in a single global pipeline, and the capability becomes a clear commercial differentiator.

Sources

1. Pedregal, D. J.; Trapero, J. R.; Holgado, E. “Demand forecasting under lost-sales stock policies.” (ScienceDirect / International Journal of Forecasting). https://www.sciencedirect.com/science/article/abs/pii/S0169207023000961
2. PyMC Labs Blog. “Probabilistic Forecasting with Censored Likelihoods.” https://www.pymc-labs.com/blog-posts/probabilistic-forecasting
3. Wang, Y. et al. (2025). “FreshRetailNet-50K: A Stockout-Annotated Censored Demand Dataset for Latent Demand Recovery and Forecasting in Fresh Retail.” (arXiv). Paper: https://arxiv.org/abs/2505.16319 PDF: https://arxiv.org/pdf/2505.16319
4. FreshRetailNet-50K Baseline Code (latent demand recovery → forecasting). https://github.com/Dingdong-Inc/frn-50k-baseline
5. Bhutani, K. “Evaluating Time Series Models for Real-World Forecasting: A Practical Comparison.” (Medium). https://medium.com/%40karanbhutani477/evaluating-time-series-models-for-real-world-forecasting-a-practical-comparison-5c9622618715
6. Marree, R. (AH Tech Blog, 2024). “Accounting for Stock-Out Substitution in Demand Forecasting at Scale.” https://blog.ah.technology/accounting-for-stock-out-substitution-in-demand-forecasting-at-scale-88d264102ee4
7. Nixtla — Nixtlaverse Documentation (for fast experimentation): MLForecast (machine-learning TS): https://nixtlaverse.nixtla.io/mlforecast StatsForecast (ARIMA/ETS, etc.): https://nixtlaverse.nixtla.io/statsforecast