# Group Project 2: Rossmann Store Sales Forecasting

Time-Series Forecasting with Classical Baselines, Prophet, and Deep Learning

Sec 04 Group 8: Aarathi Saranya Pandravada & Chirag Mahaveer Chivate

December 10, 2025

**Abstract**

This project forecasts daily sales for the Rossmann retail chain using a progression of models from a strong seasonal-naive baseline to a classical time-series model (Facebook Prophet) and exploratory deep learning (LSTM variants). We emphasize realistic, chronological splits, feature engineering driven by EDA, and business-facing evaluation (short-term vs long-term accuracy).

## 1 Dataset Description and Preprocessing

### 1.1 Dataset overview

We use the Rossmann Store Sales dataset (daily store-level observations) and store metadata. The target variable is `Sales`, and key drivers include promotions, holidays, and competition.

**Core fields (examples).**

- `Date`, `Store`: time index and store identifier.
- `Sales`, `Customers`: daily demand signals.
- `Promo`, `Promo2`, `PromoInterval`: short/long promotion indicators.
- `StateHoliday`, `SchoolHoliday`, `DayOfWeek`: calendar effects.
- `CompetitionDistance`, `CompetitionOpenSince*`: local competition context.

### 1.2 Cleaning and merging strategy

Based on EDA, we apply a universal preprocessing pipeline to make the data consistent across models:

- **Key standardization:** standardize `StateHoliday` typing to avoid category-splitting.
- **Store metadata missingness:**
  - Fill the few missing `CompetitionDistance` values with the median (robust to skew).
  - Treat `Promo2`-related missingness as structural and fill with 0 / `None`.
  - Use 0 placeholders for unknown `CompetitionOpenSince` fields; duration is computed later.
- **Join:** left-join training rows with store metadata on `Store`.
- **Filter:** keep only `Open == 1` and `Sales > 0` to model demand magnitude.

### 1.3 Feature engineering

We engineer features to encode cyclic structure and business rules:

- **Date decomposition:** Year, Month, Day, WeekOfYear.
- **Cyclical encoding:** sine/cosine transforms for Month and DayOfWeek so the model understands wrap-around (Dec ≈ Jan; Sun ≈ Mon).
- **Dynamic Promo2:** parse `PromoInterval` and compute `IsPromo2Active` (true only when the current month is inside the store's specific promo months and after the Promo2 start).
- **Competition duration:** compute `MonthsSinceCompetition` with floor at 0.

## 2 EDA Results with Visualizations

### 2.1 Data integrity and temporal structure

EDA confirms that the train timeline is continuous with no missing calendar dates, and all store IDs are present. However, the daily reporting volume is not stable across the full horizon: the number of reporting stores drops notably during mid-2014 to early-2015, producing two store groups with different history lengths. This directly impacts time-series splitting and lag/sequence creation (some stores will not have a full window of recent history).

| **Integrity check** | **EDA result** |
|---|---|
| Train date range | 2013-01-01 to 2015-07-31 |
| Unique days observed | 942 (perfect continuity) |
| Total unique stores | 1115 (Store IDs 1–1115 all present) |
| Records per store (range) | 758 to 942 days |
| History-length groups | Group A: ∼942 days; Group B: ∼758 days (missing ∼6 months) |
| Example full-history stores | 1, 726, 708, 709, 713 (942 days) |
| Example shorter-history stores | 159, 637, 636, 633, 155 (758 days) |

Table 1: Temporal and store-history integrity checks. The two history-length groups motivate store-aware validation and robust lag construction.

**Implication:** A blind global time split can unfairly penalize stores with shorter histories (missing recent blocks). We therefore design splits and lag features to be resilient to incomplete per-store history (e.g., store-aware validation windows, careful handling of missing lag windows, and avoiding assumptions that every store has the same recent context).

### 2.2 Key distributions and drivers

**Sales** is mildly right-skewed and contains a large mass at zero caused by store closures. The boxplot analysis shows extreme high values that appear to be genuine demand spikes rather than data errors.

| Variable | Mean | Std | Min | 25% | Median | 75% | Max |
|---|---|---|---|---|---|---|---|
| Sales | 5773.82 | 3849.93 | 0 | 3727 | 5744 | 7856 | 41551 |
| Customers | 633.15 | 464.41 | 0 | 405 | 609 | 837 | 7388 |

Table 2: Numerical distribution summary (train set).

| Variable | Skewness |
|---|---|
| Sales | 0.64 |
| Customers | 1.60 |
| CompetitionDistance | 2.93 |
| CompetitionOpenSinceYear | -8.01 |
| Promo2SinceWeek | 0.07 |

Table 3: Skewness diagnostics for selected variables (magnitude indicates non-normality).

**Open and Promo** are key operational drivers and are reasonably balanced for learning effects, while **Customers** tracks demand strongly but is not available at prediction time.

| Feature | Category | Count |
|---|---|---|
| Open | 1 (Open) | 844,392 |
| Open | 0 (Closed) | 172,817 |
| Promo | 1 | 388,080 |
| Promo | 0 | 629,129 |
| SchoolHoliday | 1 | 181,721 |
| SchoolHoliday | 0 | 835,488 |

Table 4: Operational and calendar feature balance (train set).

**Implications:** (i) We filter or explicitly handle closed days because zeros are deterministic outcomes of `Open==0`. (ii) `Customers` cannot be used as a direct future feature (unknown tomorrow), but it is useful as lagged history. (iii) `Promo` is a high-impact external regressor and is included across model families.

**Competition distance** is strongly skewed, motivating median imputation and robust modeling choices.

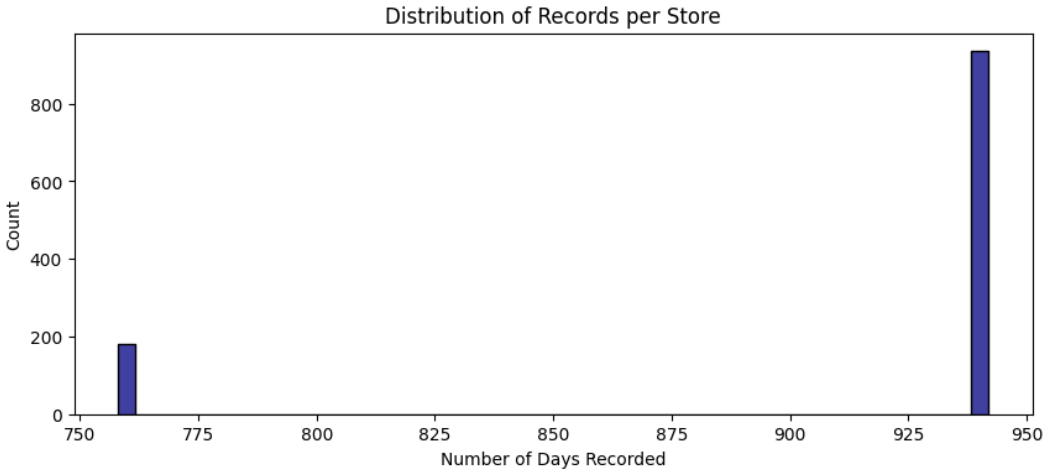


Figure 1: EDA: Distribution of `CompetitionDistance` (skewed), motivating median fill and log compression.

### 2.3 Holiday effects and weekly seasonality

**StateHoliday** is heavily imbalanced and contains mixed type representations (e.g., 0 vs. '0'), so it must be standardized during preprocessing. Given scarcity of non-zero holiday types, we consider both: (a) full one-hot encoding and (b) a simplified binary `IsStateHoliday` flag.

| Feature | Category | Count |
|---|---|---|
| StateHoliday | 0 | 986,159 |
| StateHoliday | a | 20,260 |
| StateHoliday | b | 6,690 |
| StateHoliday | c | 4,100 |

Table 5: State holiday imbalance (train set).

Weekly seasonality is strong: `DayOfWeek` is roughly uniform for days 1–6 with a clear drop on day 7 (Sunday), which is consistent with retail closures or reduced demand on Sundays.

| DayOfWeek | Count |
|---|---|
| 5 | 145,845 |
| 4 | 145,845 |
| 3 | 145,665 |
| 2 | 145,664 |
| 1 | 144,730 |
| (7) Sunday | (substantially lower than 1–6) |

Table 6: Day-of-week distribution highlights strong weekly structure (Sunday is notably lower).

**Implication:** For models that do not natively capture periodicity, we encode weekly cycles explicitly (e.g., sine/cosine cyclic encoding or day dummies) and ensure the validation scheme respects time order.

## 2.4 Autocorrelation and seasonality

For individual stores, autocorrelation analysis confirms weekly seasonality and longer-range temporal structure.

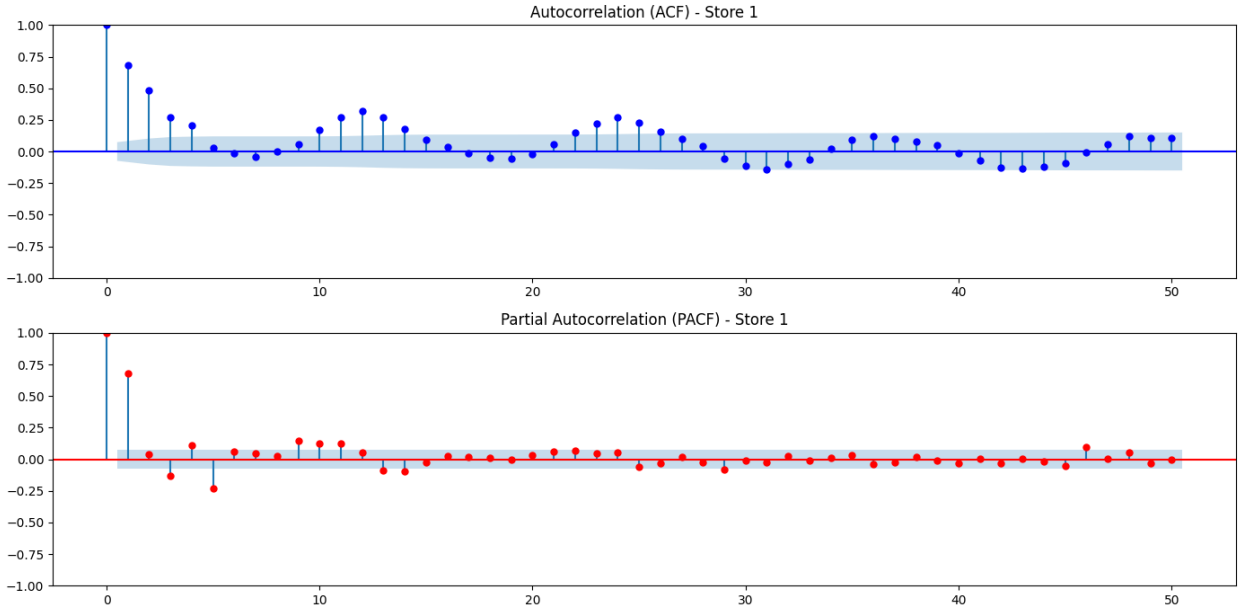


Figure 2: EDA: ACF/PACF example (Store 1) showing strong weekly patterns and structured temporal dependence.

## 2.5 Store metadata signals and missingness

Store metadata shows (i) strong class imbalance in store categories, (ii) heavy skew in competitor proximity, and (iii) non-random missingness in competition open dates and Promo2 fields.

| Store feature | Missing | Mean | Median | 25% | 75% | Max |
|---|---|---|---|---|---|---|
| CompetitionDistance (m) | 3 | 5404.90 | 2325.00 | 717.50 | 6882.50 | 75860.00 |
| CompetitionOpenSinceYear | 354 | 2008.67 | 2010 | 2006 | 2013 | 2015 |

Table 7: Selected store-level numeric signals and missingness (store.csv).

| Feature | Category | Stores |
|---|---|---|
| StoreType | a | 602 |
| StoreType | d | 348 |
| StoreType | c | 148 |
| StoreType | b | 17 |
| Assortment | a | 593 |
| Assortment | c | 513 |
| Assortment | b | 9 |
| Promo2 | 1 | 571 |
| Promo2 | 0 | 544 |

Table 8: Store-level categorical distributions (store.csv), highlighting rare classes (risk of overfitting).

Promo2-related missingness is structural: `Promo2SinceYear`, `Promo2SinceWeek`, and `PromoInterval` are missing for exactly the stores where `Promo2==0`. We therefore treat these missing values as "not applicable" rather than noisy data errors.

| PromoInterval pattern | Stores |
|---|---|
| Jan,Apr,Jul,Oct | 335 |
| Feb,May,Aug,Nov | 130 |
| Mar,Jun,Sept,Dec | 106 |
| Missing (Promo2==0) | 544 |

Table 9: PromoInterval patterns (store.csv). Missingness corresponds to non-participation in Promo2.

**Implications:** (i) `CompetitionDistance` is median-imputed and log-compressed (`log1p`) to reduce the impact of extreme outliers. (ii) `CompetitionOpenSince*` missingness is large (∼32%), so we avoid dropping data and instead engineer robust features (e.g., `MonthsSinceCompetition` plus a `HasCompetitionInfo` flag). (iii) Rare `StoreType` / `Assortment` classes increase generalization risk; splits should avoid isolating rare store types entirely in validation, and models should use regularization to prevent overfitting on tiny subgroups.

# 3 Model Design and Rationale

## 3.1 Evaluation protocol and metrics

We evaluate all models on the same date-based split (train → validation → test) to respect temporal causality and avoid leakage. Performance is reported using MAE and RMSE:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

To connect accuracy to operational planning, we additionally report a **short-term vs long-term** MAE split (e.g., first week vs later horizon) and a **degradation** percentage:

$$\text{Degradation}(\%) = 100 \cdot \frac{\text{MAE}_{\text{long}} - \text{MAE}_{\text{short}}}{\text{MAE}_{\text{short}}}.$$

This horizon analysis helps detect whether a model is only good for near-term forecasting or stays stable as the forecast window grows.

### 3.2 Tier 1: Seasonal Naive (yearly lookback baseline)

For each store $s$ and date $t$, the yearly seasonal naive baseline predicts sales by copying the observation from exactly 52 weeks earlier (364 days), which preserves the day-of-week alignment:

$$\hat{y}_{s,t} = y_{s,t-364}.$$

If $y_{s,t-364}$ is unavailable (missing history or a reporting gap), we fall back to a robust store-level statistic computed on training data, e.g. the store median:

$$\hat{y}_{s,t} = \begin{cases} y_{s,t-364}, & \text{if available,} \\ \text{median}\{y_{s,\tau} : \tau \in \mathcal{T}_{\text{train}}\}, & \text{otherwise.} \end{cases}$$

**Why this model:** Retail demand often exhibits strong year-over-year structure driven by recurring seasonality, promotions, and calendar effects. This baseline is parameter-free, very hard to "overfit," and sets a strong yardstick: any more complex model must beat it to justify added complexity.

### 3.3 Tier 2: Recursive Regularized Linear Regression (global Ridge model)

We train a global linear model (shared across stores) with engineered predictors $x_{s,t} \in \mathbb{R}^d$:

$$\hat{z}_{s,t} = \beta^\top x_{s,t}, \qquad z_{s,t} = \log(1 + y_{s,t}).$$

We use the log transform to reduce right-skew and stabilize variance, then invert with $\hat{y}_{s,t} = \exp(\hat{z}_{s,t}) - 1$. To address multicollinearity from correlated lag features, we fit Ridge regression:

$$\hat{\beta} = \arg\min_{\beta} \sum_{(s,t)\in\mathcal{T}_{\text{train}}} (z_{s,t} - \beta^\top x_{s,t})^2 + \lambda \|\beta\|_2^2.$$

Key time-series signal enters through lag features motivated by autocorrelation structure (e.g., immediate momentum and weekly seasonality):

$$x_{s,t} \ \ni \ \big(y_{s,t-1},\ y_{s,t-2},\ y_{s,t-7},\ \text{Promo}_{s,t},\ \text{Calendar}_t,\ \text{Competition}_{s,t}, \dots\big).$$

**Recursive multi-step forecasting (leakage-free):** For a horizon $k \geq 1$, future lag inputs are constructed from *predicted* values:

$$\hat{y}_{s,t+k} = f(\hat{y}_{s,t+k-1}, \hat{y}_{s,t+k-2}, \hat{y}_{s,t+k-7},\ \text{known covariates at } t+k)\,,$$

rather than using ground-truth lags from the validation/test period. This mirrors real deployment (you never know tomorrow's true sales today) and avoids the optimistic bias of one-step-ahead validation. In practice, we also apply a safety clip on recursive outputs (e.g., bounded by a multiple of max training sales) to prevent runaway error accumulation.

**Why this model:** This tier provides interpretability (coefficients quantify marginal effects like promotions) and serves as an explainable bridge between heuristics and more complex non-linear models, while the recursive evaluation stress-tests stability over longer horizons.

### 3.4 Tier 3: Facebook Prophet (generalized additive time-series model)

Prophet models each store's sales as an additive decomposition:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t,$$

where $g(t)$ is a (piecewise) trend component, $s(t)$ captures seasonality (weekly/annual patterns, often parameterized via Fourier features), and $h(t)$ encodes holiday/event effects through indicator regressors.

**Why Prophet (and not ARIMA):** Classical ARIMA-family pipelines typically require per-series parameter identification (e.g., selecting $(p, d, q)$ via ACF/PACF and iterative tuning). With 1,000+ stores, doing that store-by-store is inefficient. Prophet is designed to automate seasonality/holiday decomposition across many time series and can handle missing date blocks more gracefully, which matters under the observed reporting gaps.

### 3.5 Tier 4: Deep learning (LSTM variants; exploratory)

We model each store's history as a supervised sequence learning problem. Given a window of length $L$ with per-day covariates,

$$X_{s,t} = \{x_{s,t-L+1}, \ldots, x_{s,t}\},$$

we predict next-day sales $\hat{y}_{s,t+1}$. A standard LSTM maps inputs to hidden states:

$$(h_\ell, c_\ell) = \mathrm{LSTM}(x_{s,t-L+\ell},\, h_{\ell-1},\, c_{\ell-1}), \quad \ell = 1, \ldots, L,$$

followed by a regression head (vanilla variant):

$$\hat{y}_{s,t+1} = w^\top h_L + b.$$

**Attention variant:** Instead of relying only on $h_L$, we learn weights over the full history:

$$\alpha_\ell = \mathrm{softmax}\Big(v^\top \tanh(W h_\ell)\Big), \qquad c = \sum_{\ell=1}^{L} \alpha_\ell h_\ell, \qquad \hat{y}_{s,t+1} = w^\top c + b.$$

Optionally, store identity can be represented via a learned embedding $e_s$ and concatenated to each day's features:

$$x'_{s,t} = [x_{s,t}; e_s],$$

allowing the model to learn store-specific latent behavior while still training a single global network.

**Why this model:** Linear models capture only additive/linear effects, while LSTMs can learn non-linear temporal interactions (e.g., promotion effectiveness varying by store type and time of year). Attention further helps by letting the model emphasize the most informative days in the lookback window. These deep models address the "advanced" requirement but are treated as exploratory due to higher tuning and compute cost.

# 4 Evaluation Results with Tables/Figures

## 4.1 Tier 1–3 comparison (baseline, linear regression, prophet)

Table 10: Model comparison across Tier 1–3 using global and horizon-based MAE (from the project notebook).

| Model | Global MAE | Global RMSE | Short-Term MAE | Long-Term MAE |
|---|---|---|---|---|
| Seasonal Naive (Yearly) | 875.59 | 1277.36 | 910.10 | 843.88 |
| Recursive Linear Regression | 2916.41 | 5012.08 | 2556.51 | 2948.61 |
| Facebook Prophet | 1136.10 | 1438.93 | 833.11 | 1271.55 |

## 4.2 Tier 4 (LSTM) results (exploratory)

Table 11: LSTM variants on the held-out test period (as reported in the notebook).

| Model | MAE | RMSE |
|---|---|---|
| Vanilla LSTM | 4935.51 | 6338.39 |
| LSTM + Attention | 1721.02 | 2141.06 |

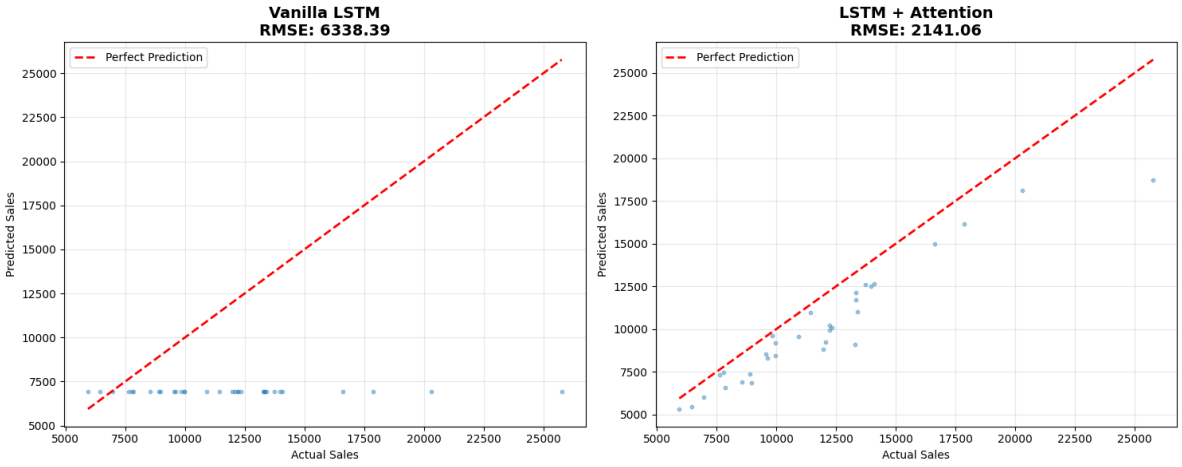


Figure 3: Attention visualization (example) showing which recent days the model weighted most for prediction.

# 5 Key Insights, Limitations, and Future Improvements

## 5.1 Key insights

- **Year-over-year seasonality is the dominant signal.** The Tier 1 Seasonal Naive (52-week lookback) achieved the best *global* error among the classical baselines, showing that "same time last year" captures most of the predictable structure in this retail dataset.

- **The best model depends on the forecast horizon.** Prophet is strongest for *short-term* (next-week) forecasting because explicit weekly seasonality and holiday effects improve tactical accuracy, but it degrades over longer horizons. In contrast, the Seasonal Naive baseline remains robust (and even improves) as the validation window moves into a more stable seasonal regime.

- **Recursive multi-step evaluation exposes real-world drift.** The recursive linear regression setup (feeding predictions back into lag features) is intentionally strict and leakage-resistant. Its weak performance highlights a key lesson: models that rely heavily on short lags can become unstable when forced to self-generate their own momentum. Despite this, the linear model is still valuable for interpretation: promotions emerge as strong positive drivers, while holiday/specific weekday effects act as suppressors.

- **Attention helps deep models, but feature depth matters.** The Tier 4 experiments show a large gain from adding attention over a vanilla LSTM, indicating that weighting recent days can improve sequence-to-one forecasting. However, the deep models did not surpass the strongest baseline, suggesting that architecture alone is insufficient without richer covariates, store identity signals, and broader tuning.

### 5.2 Limitations

- **Non-uniform history length and reporting gaps.** EDA identified store-history irregularities ("Group B" style gaps), which complicate sequence creation, horizon evaluation, and fair comparisons across stores.

- **Metric aggregation can hide heterogeneity.** Reporting global MAE/RMSE (averaged across stores) can obscure which store types or regimes (e.g., promotion vs non-promotion, holiday weeks, low-volume stores) are driving error.

- **Baseline brittleness under structural change.** A 52-week copy rule can fail when a store undergoes regime shifts (renovation, competitive entry/exit, assortment changes) or when holiday timing drifts year-to-year.

- **Prophet configuration is intentionally narrow.** Prophet was used primarily for decomposition/holiday handling and robustness to gaps; without strong exogenous regressors, it can underfit medium/long-term year-over-year shifts.

- **Deep learning was exploratory, not fully optimized.** The LSTM variants used a compact feature set and a single date-based split; performance is sensitive to window length, scaling, store encoding, and hyperparameters that were not exhaustively tuned.

### 5.3 Future improvements

- **Data handling and evaluation.** Use rolling-origin backtests (multiple cutoffs), stratify results by store group/history length, and add horizon-by-horizon dashboards (errors vs $h$) to diagnose drift regimes.

- **Richer feature engineering.** Incorporate stronger competitive/promotion dynamics (e.g., Promo2 activity windows, time-since-competition-open, interaction terms like Promo $\times$ DayOfWeek), and explicitly model `Open`/closure days to avoid mixing structural zeros with normal demand.

- **Stronger global ML baselines.** Add boosted-tree models (e.g., LightGBM/CatBoost) with calendar + lag features as a high-performing, scalable middle ground between linear regression and deep learning.
- **Improved neural sequence modeling.** Add store embeddings and train a single global sequence model across stores; consider direct multi-horizon prediction (predict a vector of future days) rather than recursive rollouts to reduce error accumulation.
- **Hybrid strategy (recommended).** Use the Seasonal Naive baseline as a strong seasonal anchor, then train a second-stage residual model (ML or neural) focused on deviations driven by promotions, holidays, and competition effects.