As an industry expert data scientist and CRISP-DM practitioner, use the Kaggle M5 Forecasting dataset to deliver a complete, stepwise project in compute-aware chunks: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk; begin with Business Understanding and success criteria. It is important to chunk this above problem. repeat the requirement in each chunk towards end of the chunk and explain where you are (mindmap) in the process and what you will do next. Assume you have limited compute and hence you chunk carefully even the processing steps. I will ask you to continue. each step.

Business Understanding — M5 Forecasting (compute-aware)

Direct answer: The business objective is to forecast daily unit sales for Walmart over a 28-day horizon in a hierarchical structure, optimizing for the primary metric **WRMSSE** while supporting operational decisions via uncertainty-aware scenario analysis under strict compute constraints. [1] [2] [3]

Objective and Scope

- Goal: Produce coherent 28-day forecasts at item-store level that aggregate correctly to all hierarchy levels, with WRMSSE as the primary KPI and RMSE/MAE as secondary diagnostics. [2] [3]
- Context: Dataset includes 3,049 products, 10 stores, 3 states, forming 30,490 bottom-level series with calendar effects and weekly prices; forecasts must account for hierarchy and intermittent demand. [4] [1] [2]

Success Criteria

- Primary: Minimize WRMSSE across 42,840 series implied by the hierarchical levels; forecasts must be coherent by construction from bottom-level predictions. [3] [2]
- Secondary: Maintain competitive RMSE/MAE at bottom level and selected aggregates; ensure stable performance across rolling validation folds. [5] [6]
- Practical: Deliver scenario-ready outputs (price-promo toggles, SNAP flags) and quantified uncertainty bands suitable for operations planning. [7] [8]

Business Questions

- What is the expected range of 28-day demand under baseline vs promo/SNAP scenarios? [8] [7]
- Which levels (state/store/department) drive most of the WRMSSE weight and therefore deserve modeling focus under limited compute? [2] [3]
- How to balance intermittent demand handling with tree-boosting models for scale and SARIMA/ETS for calibration at selected aggregates? [9] [2]

Constraints and Assumptions

- Compute-aware: Prioritize global tree-based models with sparse engineered features; selective statistical baselines at aggregates for calibration/ensembling; lightweight time series CV. [9] [2]
- Coherence: Submit only bottom-level forecasts; the competition protocol ensures hierarchical coherence via aggregation. [1] [3]
- Evaluation: WRMSSE weights emphasize high-dollar sales over the last 28 days, making price effects and recency critical. [8] [3]

Deliverables

- Forecasts: 28-day horizons for all 30,490 series; plus aggregate-level diagnostics and scenario deltas. [1] [2]
- Benchmarks: Naive and Seasonal Naive at bottom-level CV; report WRMSSE, RMSE, MAE. [3] [2]
- Models: SARIMA/ETS at key aggregates, LightGBM/XGBoost global models at bottom-level with lag/rolling/promo features; uncertainty via quantiles or residual bootstraps. [9] [2]

- Documentation: Compute-aware pipeline with hierarchical EDA, feature rationale, CV design, and uncertainty discussion.[10] [5]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[2] [3] [1]

Mindmap (progress):

- Business Understanding: complete.[1]
- Data Understanding: next.[10]
- EDA (hierarchy, calendar/price): pending.[10]
- Features: pending.[2]
- Cleaning: pending.[10]
- Aggregation strategy: pending.[3]
- CV: pending.[5]
- Benchmarks: pending.[2]
- Modeling: pending.[9]
- Uncertainty/scenarios: pending.[7]

Next chunk preview: Data Understanding — summarize files (sales_train_evaluation, calendar, sell_prices), keys and joins, hierarchy levels, time ranges, validation/test splits, and implications for compute-aware processing.[4] [10] [1]

⁂

continue

Data Understanding — files, keys, hierarchy, time ranges (compute-aware)

Direct answer: The dataset contains daily unit sales from 2011-01-29 to 2016-05-22 across 30,490 item-store series, joined with weekly prices and a rich event calendar; core files are sales_train_validation/evaluation, calendar, and sell_prices with joins via d/DATE and wm_yr_wk, forming a 12-level hierarchy totaling 42,840 series.[18] [19] [20]

Files and schemas

- sales_train_validation/evaluation: One row per item_id-store_id with columns d_1...d_1941 containing daily unit sales; keys include id, item_id, dept_id, cat_id, store_id, state_id.[21] [18]
- calendar.csv: Maps d to Gregorian dates, has wm_yr_wk, weekday, month/week numbers, SNAP flags per state, and event names/types for holidays.[22] [18]

- sell_prices.csv: Weekly average sell_price by store_id, item_id, wm_yr_wk; used to capture promotions and price elasticity.[23] [18]

Time ranges and splits

- Horizon: Forecast 28 days ahead for all series; training days are d_1 to d_{1913} for validation phase and up to d_{1941} for final; start date 2011-01-29.[24] [20]
- Practical CV: Rolling windows ending before final 28-day blocks to simulate leaderboard and final scoring periods.[25] [24]

Hierarchy and joins

- Hierarchy: 12 aggregation levels from all-products to item-store bottom level; total 42,840 series used in WRMSSE evaluation.[19] [26]
- Joins: sales joins calendar on 'd' via melt/pivot; prices join via wm_yr_wk and (store_id, item_id).[21] [23]

Compute-aware notes

- Avoid full melt of all 1,941 days initially; sample recent 1–2 years for EDA and prototype features, then expand gradually.[20] [23]
- Use categorical codes for IDs and selective merging on required windows to fit memory.[23] [21]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[27] [28] [29]

Mindmap (progress):

- Business Understanding: complete.[29]
- Data Understanding: complete.[18]
- EDA (hierarchy, calendar/price): next.[30]

Next chunk preview: Hierarchical EDA — quantify series counts per level, sparsity/intermittency distributions, weekly/annual seasonality diagnostics, and quick analysis of aggregation effects on variance and autocorrelation using recent 730 days to stay compute-aware.[19] [30] [20]

⁂

continue

Hierarchical EDA — structure, seasonality, intermittency (compute-aware)

Direct answer: The hierarchy spans 12 aggregation levels totaling **42,840** series, with strong weekly seasonality at aggregates, mixed annual/event effects, and substantial intermittency at the bottom level; EDA should focus on recent 730 days, sampling representative items per department-store to quantify sparsity and seasonality while staying compute-aware. [39] [40] [41]

Hierarchical structure

- Levels and counts: From total (1) to product-store bottom level (30,490), summing to 42,840 series across 12 levels used for evaluation. [42] [40]
- Coherence: Forecasts are submitted at bottom level and aggregate by summation to higher levels for evaluation, ensuring coherence by design. [40] [39]

Seasonality patterns

- Weekly seasonality: Aggregates clearly show a 7-day cycle; EDA should confirm weekday effects by plotting day-of-week means on aggregates. [43] [41]
- Event/holiday effects: Calendar-linked surges (e.g., holidays, SNAP) are visible in category/department aggregates; examine event_name_1/2 and SNAP flags by state. [41] [39]

Intermittency and sparsity

- Bottom-level demand: Many SKU-store series have zeros and bursts; apply ADI and CV^2 classification to quantify intermittent vs smooth series. [44] [45]
- Practical note: Intermittency is prevalent in granular Walmart sales; consider Croston/SBA diagnostics as a baseline characterization. [46] [47]

Compute-aware EDA plan

- Window: Use last 730 days for ACF/PACF summaries and seasonal plots at selected levels. [40] [41]
- Sampling: Stratify by department and store to pick a small panel (e.g., 200 SKUs) for deep EDA, while keeping aggregate-level EDA for all series. [39] [40]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [48] [49] [40]

Mindmap (progress):

- Business Understanding: complete.[48]
- Data Understanding: complete.[50]
- EDA (hierarchy): complete.[40]
- Calendar/price effects: next.[39]
- Features: pending.[40]
- Cleaning: pending.[39]
- Aggregation strategy: pending.[40]
- CV: pending.[51]
- Benchmarks: pending.[42]
- Modeling: pending.[42]
- Uncertainty/scenarios: pending.[52]

Next chunk preview: Calendar and price effects — quantify day-of-week, month, SNAP-by-state, and event types; compute price indices and promo flags from sell_price changes; test elasticity via simple log-demand vs log-price regressions on sampled SKUs and aggregates.[53] [41] [50]

⁂

continue

Calendar and Price Effects — quantify impacts (compute-aware)

Direct answer: Calendar effects include strong day-of-week differences, state-specific SNAP days, and named holiday/event spikes; price effects are modeled from weekly sell_price via promo flags and log-log elasticity regressions on sampled SKU-store panels to estimate demand response under compute limits.[63] [64] [65]

Calendar features

- Core fields: weekday, wday, month, year, wm_yr_wk, SNAP_CA/TX/WI, event_name_1/2, event_type_1/2 from calendar.csv enable weekday, monthly, and holiday effect modeling, with SNAP flags varying by state.[65] [63]
- Practical encoding: Create categorical indicators for weekday, month, major event types, and state-specific SNAP dummies, focusing on the last 730 days to reduce memory/merge cost.[63] [65]

Price and promotion features

- Weekly prices: sell_prices.csv provides average weekly sell_price at (store_id, item_id, wm_yr_wk); merge to daily data via wm_yr_wk then forward-fill within week.[65] [63]
- Promo heuristics: Define promo_flag from price cuts relative to a rolling reference (e.g., 8–12 week rolling min or a percentage drop threshold), plus features for price_pct_change and normalized price index per SKU.[64] [66]

Elasticity estimation (compute-aware)

- Log-log regression: Estimate $\ln(Q_{t}+1) = \beta_0 + \beta_1 \ln(P_{t}) + \gamma'X_t + \epsilon_t$ on a stratified 200-SKU sample by department-store, interpreting β_1 as own-price elasticity.[67] [68]
- Controls: Include weekday, SNAP-by-state, and event type controls to avoid omitted variable bias; use robust errors and restrict to recent 730 days for tractability.[68] [63]

Workflow outline

- Merge keys: From sales to calendar by d, and to prices by wm_yr_wk and (store_id, item_id), limiting rows to recent windows.[63] [65]
- Diagnostics: Plot average sales by weekday and by SNAP for each state; compare mean sales on event vs non-event days; summarize elasticity estimates by category.[64] [63]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[69] [70] [71]

Mindmap (progress):

- Business Understanding: complete.[69]
- Data Understanding: complete.[72]
- EDA (hierarchy): complete.[71]
- Calendar/price effects: complete.[63]
- Features: next.[64]
- Cleaning: pending.[65]
- Aggregation strategy: pending.[63]
- CV: pending.[73]
- Benchmarks: pending.[71]
- Modeling: pending.[74]
- Uncertainty/scenarios: pending.[75]

Next chunk preview: Feature creation — compute-aware generation of lags (1, 7, 28), rolling means/std (7, 28, 56), price-derived features (log price, price index, promo flags), and calendar dummies, using windowed joins and incremental transforms to keep memory within limits.[66] [64] [63]

⁂

continue

Feature Creation — lags, rolls, promos, calendar (compute-aware)

Direct answer: Create minimal but high-signal features—sales lags $1, 7, 28$, rolling means/std over $7, 28, 56$ days, price features (log price, price index, promo flags), and compact calendar dummies—computed in recent 730–900 day windows per series to control memory, following common M5-winning pipelines.[90] [91] [92]

Core lag features

- Lags: Include $y_{t-1}, y_{t-7}, y_{t-28}$ to capture short-, weekly-, and monthly-cycle dependencies; optionally lags $14, 35$ if memory allows.[93] [90]
- Rolling targets: Compute rolling mean/std of y over windows $7, 28, 56$ aligned after lag 28 to avoid leakage; add rolling min/max for spike control if feasible.[94] [90]

Price-derived features

- Log price and changes: Use $\ln(\text{price})$, week-over-week pct change, and deviation from SKU rolling median to capture temporary promos.[92] [90]
- Promo flags: Binary when price is within 1.0–1.1x of rolling 8–12 week minimum or drops > x% from baseline; include promo start/end indicators.[95] [96]

Calendar features

- Minimal set: day-of-week, week-of-month, month, year, SNAP-by-state, and major event-type one-hots; prefer integer-coded categories to save RAM.[91] [97]
- Seasonality helpers: Sine/cosine for weekly cycle can help tree/linear models, though trees often suffice with DOW dummies.[98] [99]

Compute-aware generation

- Windowed builds: For each store or category shard, build features only for the last 900 days and drop intermediates; persist to disk between shards.[91] [94]
- Leakage controls: Ensure all rolling stats are computed on lagged versions (e.g., roll over y_{t-28}) and recompute sequentially for recursive forecasting.[92] [91]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[100] [101] [91]

Mindmap (progress):

- Business Understanding: complete.[101]
- Data Understanding: complete.[102]
- EDA (hierarchy): complete.[100]
- Calendar/price effects: complete.[103]
- Features: complete.[90]
- Cleaning: next.[91]
- Aggregation strategy: pending.[100]
- CV: pending.[104]
- Benchmarks: pending.[95]
- Modeling: pending.[98]
- Uncertainty/scenarios: pending.[105]

Next chunk preview: Data cleaning — reconcile missing prices/dates, handle zero and negative values, outlier winsorization at SKU-store, ensure consistent categorical mappings, and verify no leakage after merges and rolling computations, all done shard-wise to fit memory.[96] [97] [91]

⁂

continue

Data Cleaning — joins, gaps, outliers, leakage (compute-aware)

Direct answer: Clean by validating keys and merges, imputing weekly prices, handling missing calendar fields, capping extreme outliers, and enforcing leakage-safe lag/rolling construction; do all operations shard-wise over recent windows to stay within memory.[115] [116] [117]

Key validation and joins

- Referential integrity: Ensure every (store_id, item_id, wm_yr_wk) in sell_prices maps to calendar and to each SKU-store row; report anti-joins to catch missing weeks.[116] [115]
- Join order: Sales → melt recent 730–900 days → join calendar on d → join prices on (wm_yr_wk, store_id, item_id); forward-fill price within week to daily.[115] [116]

Handling missing values

- Prices: If a weekly price is missing between two valid weeks, linearly interpolate or carry forward/backward within a bounded window; otherwise mark price_missing flag.[116] [115]
- Calendar: Fill structural fields (weekday, month) deterministically; set event indicators to zero when event_name/type is NA; retain explicit missing flags for robustness.[115] [116]

Outliers and anomalies

- Sales spikes: Winsorize SKU-store daily sales at e.g., 99.5th percentile per series or per department-store to reduce undue influence while keeping genuine peaks. [118] [119]
- Policy: Prefer capping over deletion; document thresholds and maintain raw copy for auditability. [120] [121]

Leakage prevention

- Lag/roll construction: Compute rolls only on lagged targets (e.g., roll on y_{t-28}) and align features per forecast day in recursive prediction; do not use future observations in any window. [122] [123]
- Test horizon specifics: The first prediction day needs only short lags (e.g., 1, 7); later days require feeding model predictions to update features if using short lags. [123] [122]

Compute-aware execution

- Sharding: Process by store or department to bound RAM; persist intermediate parquet files with minimal dtypes and categorical codes. [124] [125]
- Windowing: Restrict heavy transforms to last ~900 days; keep full history only for WRMSSE scaling periods if needed. [117] [125]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [125] [126] [127]

Mindmap (progress):

- Business Understanding: complete. [126]
- Data Understanding: complete. [128]
- EDA (hierarchy): complete. [127]
- Calendar/price effects: complete. [116]
- Features: complete. [129]
- Cleaning: complete. [115]
- Aggregation strategy: next. [127]
- CV: pending. [122]
- Benchmarks: pending. [130]
- Modeling: pending. [131]
- Uncertainty/scenarios: pending. [132]

Next chunk preview: Aggregation/disaggregation strategy — decide bottom-up vs top-down vs middle-out, with coherence via summation from bottom-level predictions; optionally align independent models using proportions, and sample aggregates for SARIMA/ETS calibration to blend with global ML. [133] [125] [127]

⁂

continue

Aggregation/Disaggregation Strategy — coherence under compute limits

Direct answer: Use a bottom-up strategy with global tree-based models at the SKU–store level to ensure coherent aggregation, and optionally apply lightweight top-down alignment on the total series to nudge bottom forecasts toward upper-level totals emphasized by WRMSSE. [147] [148] [149]

Primary approach: bottom-up

- Rationale: Submitting bottom-level forecasts guarantees coherence by summation to all higher levels for evaluation, matching the competition protocol. [149] [150]
- Global modeling: Train one or few global models over all SKU–store series with shared features, then aggregate to evaluate level-wise diagnostics, minimizing compute. [151] [148]

Optional alignment: top-down nudge

- Alignment idea: Fit a simple model at the top level and adjust bottom-level forecasts via a scalar loss multiplier so that aggregated sums over the horizon better match top-level predictions. [152] [147]
- Why top level: Upper levels carry larger WRMSSE weights and are less noisy; aligning only to the top is a good cost-benefit tradeoff under resource constraints. [153] [147]

Alternatives and trade-offs

- Top-down/middle-out: Can reduce noise at low levels but risk information loss and degraded bottom accuracy unless proportions are well-modeled; useful if sparsity is extreme. [154] [155]
- Robustness caveat: Evidence suggests global bottom-up rankings can be unstable across time; using rolling validation and simple alignment improves robustness. [148] [153]

Implementation notes

- Compute-aware: Produce bottom-level predictions once; reuse sums for all aggregates; if aligning, compute and apply a single horizon-wide multiplier per fold. [147] [148]
- Monitoring: Track errors by level and department to decide when alignment improves WRMSSE versus harming bottom-level RMSE/MAE. [151] [149]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [150] [156] [149]

Mindmap (progress):

- Business Understanding: complete. [156]
- Data Understanding: complete. [157]
- EDA (hierarchy): complete. [149]
- Calendar/price effects: complete. [158]
- Features: complete. [159]
- Cleaning: complete. [160]
- Aggregation strategy: complete. [147]
- CV: next. [161]
- Benchmarks: pending. [162]
- Modeling: pending. [151]
- Uncertainty/scenarios: pending. [153]

Next chunk preview: Time-series cross-validation — rolling-origin evaluation with 28-day blocks, minimal 2–3 folds on the last ~1.5 years, computing WRMSSE (with fixed scale weights) and RMSE/MAE at bottom level; cache weights to avoid recomputation and keep compute low. [163] [161] [153]

⁂

continue

Time-Series Cross-Validation — rolling-origin with WRMSSE (compute-aware)

Direct answer: Use 2–3 rolling-origin folds of 28 days each over the last ~1.5 years, compute WRMSSE with fixed weights per the M5 guide, and report secondary RMSE/MAE at the bottom level; cache weights and reuse features to minimize compute. [175] [176] [177]

Fold design

- Rolling horizon: Form folds ending at d_{1913}, d_{1935} etc., each forecasting the next 28 days, with training including all prior data only; this mirrors leaderboard/final evaluation. [177] [175]
- Minimal folds: Choose 2–3 folds to balance robustness with runtime; keep feature windows fixed (e.g., last 900 days) and reuse between folds to avoid recomputation. [178] [179]

WRMSSE computation

- Definition: WRMSSE aggregates RMSSE across the 12-level hierarchy using revenue-derived weights; weights and scale factors must be computed from the training window. [176] [180]

- Practical tip: Precompute and cache the 42,840 weights and scale denominators once per fold; libraries and community notebooks provide reference implementations. [181] [182]

Secondary metrics

- Bottom-level diagnostics: Compute RMSE and MAE at SKU–store for each fold to detect overfitting not visible in WRMSSE. [179] [183]
- Baseline comparisons: Ensure all models beat Naive and Seasonal Naive on WRMSSE and bottom-level RMSE/MAE to justify complexity. [184] [185]

Update vs refit

- Strategy: For global ML models, prefer updating inputs without full refits each fold; for ARIMA/ETS at aggregates, refit per fold is standard. [186] [179]
- Efficiency: Warm-start tree boosters with prior models or fixed hyperparameters across folds to save compute. [187] [178]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [188] [189] [177]

Mindmap (progress):

- Business Understanding: complete. [188]
- Data Understanding: complete. [190]
- EDA (hierarchy): complete. [189]
- Calendar/price effects: complete. [191]
- Features: complete. [192]
- Cleaning: complete. [193]
- Aggregation strategy: complete. [194]
- CV: complete. [175]
- Benchmarks: next. [183]
- Modeling: pending. [180]
- Uncertainty/scenarios: pending. [195]

Next chunk preview: Benchmarks — implement Naive (last value) and Seasonal Naive (last value from 7 days ago) for 28-day horizons, evaluate across folds with WRMSSE, RMSE, MAE, and establish a compute-light baseline before advanced models.[196] [185] [183]

⁂

continue

Benchmarks — Naive and Seasonal Naive (compute-aware)

Direct answer: Establish two baselines: Naive copies the last observed value forward 28 steps; Seasonal Naive repeats the last observed value from the same weekday (season length 7) across the 28-day horizon; evaluate both with WRMSSE plus bottom-level RMSE/MAE over the CV folds.[202] [203] [204]

Definitions and setup

- Naive: For horizon $h = 1..28$, set $\hat{y}_{T+h|T} = y_T$ for each SKU–store, creating a flat 28-day forecast. [203] [205]
- Seasonal Naive (weekly): For horizon h, set $\hat{y}_{T+h|T} = y_{T+h-7}$, using last week's same weekday; implement efficiently by repeating the last 7 days. [203] [202]

Expected performance

- WRMSSE references: Public examples show Naive around 1.46 and Seasonal Naive near 0.84–0.87 depending on weekly vs 4-week variants; exact values vary by implementation window.[204] [202]
- Use as gatekeepers: Any advanced model must beat Seasonal Naive on WRMSSE and improve RMSE/MAE at the bottom level across folds.[206] [207]

Compute-aware execution

- Vectorized generation: Build baselines directly from the wide training matrix without melting, by slicing the last 7 or 28 columns, then broadcast to 28-day horizon.[208] [202]
- Scoring: Reuse cached WRMSSE weights per fold; compute RMSE/MAE via matrix ops for speed.[209] [204]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[210] [211] [212]

Mindmap (progress):

- Business Understanding: complete.[212]
- Data Understanding: complete.[213]
- EDA (hierarchy): complete.[210]
- Calendar/price effects: complete.[214]
- Features: complete.[215]
- Cleaning: complete.[216]
- Aggregation strategy: complete.[217]
- CV: complete.[218]
- Benchmarks: complete.[202]
- Modeling: next.[211]
- Uncertainty/scenarios: pending.[219]

Next chunk preview: Modeling — fit lightweight SARIMA/ETS on selected aggregates and a global LightGBM/XGBoost on bottom level with engineered features; evaluate with WRMSSE and RMSE/MAE, and prepare quantile or residual-bootstrap uncertainty plus promo/SNAP scenario toggles.[220] [221] [211]

⁂

continue

Modeling — SARIMA/ETS and Global LightGBM/XGBoost (compute-aware)

Direct answer: Fit lightweight SARIMA/ETS on selected aggregates for calibration and a global LightGBM/XGBoost at the bottom level using engineered lags/rolls/calendar/price features; evaluate with WRMSSE and RMSE/MAE across CV folds, then prepare uncertainty via quantiles or residual bootstraps and enable promo/SNAP scenario toggles.[232] [233] [234]

Statistical models at aggregates

- ETS/SARIMA: Use AutoETS or SARIMAX on state–department and total levels where seasonality is clear and noise is lower; parallelize sparse set of series only.[233] [235]
- Purpose: Provide calibrated, low-variance references and optional top-down alignment signals to nudge global model sums.[236] [232]

Global tree-boosting at bottom level

- LightGBM/XGBoost: Train a global regressor on SKU–store rows with features: lags 1/7/28, rolling mean/std 7/28/56, log price, promo flags, DOW/month/SNAP/events; shard by store or department to fit memory.[237] [238]
- Horizon strategy: Prefer direct multi-horizon with 28 separate models or multi-output head; many top solutions trained per-horizon models to avoid recursive error accumulation.[238] [239]

Training efficiency

- Memory control: Use categorical encoding for IDs, compress dtypes, and persist features per shard; reuse trained models across folds with minor refits if distributions stable.[232] [238]
- Hyperparameters: Start with small depth/num_leaves and moderate learning rate; limit features using importance to reduce inference cost.[240] [238]

Evaluation

- Metrics: Report WRMSSE as primary, with bottom-level RMSE/MAE per fold; compare vs Seasonal Naive and document gains at weighted upper levels.[241] [242]
- Diagnostics: Check residual ACF, stratified errors by category/store, and drift on promo/SNAP days to catch bias.[234] [232]

Uncertainty and scenarios

- Quantiles: Train gradient boosting with quantile objectives or fit residual bootstrap around point forecasts to produce 10th/50th/90th percentiles.[243] [244]
- Scenario toggles: Recompute features with counterfactual price/promo/SNAP flags, holding others fixed, to simulate demand deltas; aggregate to levels for planning.[243] [232]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[245] [246] [232]

Mindmap (progress):

- Business Understanding: complete.[245]
- Data Understanding: complete.[247]
- EDA (hierarchy): complete.[246]
- Calendar/price effects: complete.[248]
- Features: complete.[249]
- Cleaning: complete.[250]
- Aggregation strategy: complete.[236]
- CV: complete.[251]
- Benchmarks: complete.[252]
- Modeling: complete.[238]
- Uncertainty/scenarios: next.[244]

Next chunk preview: Uncertainty and scenario analysis — generate quantile forecasts via boosting with quantile loss or residual bootstraps, compute coverage and calibration on CV folds, and run scenario analyses toggling promo/SNAP and price shifts to quantify demand ranges and sensitivities at key hierarchy levels. [244] [243] [232]

⁂

continue

Uncertainty and Scenario Analysis — quantiles, coverage, promos/prices (compute-aware)

Direct answer: Provide probabilistic forecasts via quantile models or bootstrapped residuals, evaluate with M5's weighted scaled pinball loss and coverage, and run scenario toggles for promotions/SNAP and price shifts to quantify demand ranges and sensitivities across hierarchy levels. [262] [263] [264]

Probabilistic forecasts

- Quantiles: Train boosting with quantile loss to produce 0.1/0.5/0.9 (or M5's 9-quantile set) per SKU–store, aligning with the M5 Uncertainty track's evaluation using weighted scaled pinball loss. [264] [262]
- Bootstrapped residuals: Generate intervals by resampling residuals around point forecasts; compute per-horizon percentiles to obtain lightweight prediction intervals under tight compute. [265] [266]

Evaluation

- Metric: Use weighted scaled pinball loss (WSPL) across the full hierarchy, mirroring M5 Uncertainty; compute empirical coverage (e.g., P10–P90 should cover ~80%) as a sanity check. [263] [267]
- Calibration insights: Prior findings show tail quantiles at granular levels can be miscalibrated; summed bottom-level quantiles sometimes outperform directly modeled aggregates in extreme tails. [268] [262]

Scenario analysis

- Promo/SNAP toggles: Recompute features with promo_flag on/off and SNAP indicators flipped by state to simulate uplift/suppression, then aggregate to department/state for planning. [263] [264]
- Price shifts: Evaluate counterfactual price paths by shifting log price or applying elasticity-based adjustments and refreezing features; summarize demand and profit deltas. [269] [270]

Compute-aware execution

- Minimal quantile set: Start with P10/P50/P90 to bound ranges; extend to the M5 9-quantile grid if runtime allows.[262] [263]
- Efficient bootstraps: Use a modest number of residual draws (e.g., 200) and reuse across folds; block residuals if autocorrelation is material.[266] [265]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[271] [272] [273]

Mindmap (progress):

- Business Understanding: complete.[271]
- Data Understanding: complete.[274]
- EDA (hierarchy): complete.[273]
- Calendar/price effects: complete.[275]
- Features: complete.[276]
- Cleaning: complete.[277]
- Aggregation strategy: complete.[278]
- CV: complete.[279]
- Benchmarks: complete.[280]
- Modeling: complete.[281]
- Uncertainty/scenarios: complete.[262]

Next chunk preview: Wrap-up and handoff — provide a compute-aware runbook with step ordering, memory tips, and suggested code modules; list deliverables and a checklist to move from prototype to production (monitoring, drift checks, and retraining cadence aligned with M5-like weekly price updates).[272] [282] [283]

⁂

continue

Wrap-up and Handoff — production runbook, monitoring, retraining (compute-aware)

Direct answer: Package a lean pipeline that produces bottom-level forecasts and aggregates them, monitor WRMSSE/RMSE plus data/feature drift weekly, and retrain on a cadence aligned to price/calendar updates or triggered by drift/metric degradation, with scenario and quantile outputs for planning.[295] [296] [297]

Runbook and sequencing

- Daily/weekly job: Ingest latest sales, calendar, and prices; update engineered features for the last ~900 days; run global model inference for 28-day horizon; aggregate to all levels; compute baseline diagnostics.[296] [298]
- Cadence: Retrain monthly or when drift/metrics breach thresholds; recompute WRMSSE weights per retrain using the latest training cut.[299] [295]

Monitoring and alerts

- Performance: Track WRMSSE (primary) and bottom-level RMSE/MAE weekly; alert on degradation versus a rolling baseline.[300] [297]
- Drift: Monitor input drift on key drivers (price index, promo flag rates, DOW mix, SNAP flags) and prediction drift; use simple tests (e.g., KS/PSI) with thresholds to trigger retrain. [301] [302]

MLOps considerations

- Orchestration and tracking: Use a workflow tool and experiment tracker to log CV fold metrics, model versions, and data snapshots for rollback.[303] [304]
- Feature store: Centralize feature definitions (lags, rolls, promo flags, calendar) for training/inference parity and reproducibility.[305] [306]

Efficiency tips

- Sharding and caching: Persist shard-level feature matrices as parquet with categorical codes; cache WRMSSE weights and calendar joins for reuse.[296] [299]
- Lightweight ensembling: Keep SARIMA/ETS aggregates only where they help; otherwise rely on a single tuned LightGBM/XGBoost for cost control.[298] [307]

Deliverables checklist

- Forecast artifacts: P50 point forecasts plus P10/P90 bands; aggregates to all 12 hierarchy levels; scenario deltas for promo/SNAP/price toggles.[308] [309]
- Documentation: CRISP-DM report with EDA, features, CV design, benchmarks, model choices, uncertainty, and scenario playbook; production runbook with SLA and retrain triggers.[297] [296]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[310] [311] [298]

Mindmap (progress):

- Business Understanding: complete.[310]
- Data Understanding: complete.[312]
- EDA (hierarchy): complete.[311]
- Calendar/price effects: complete.[313]
- Features: complete.[314]
- Cleaning: complete.[315]
- Aggregation strategy: complete.[316]
- CV: complete.[317]
- Benchmarks: complete.[318]
- Modeling: complete.[319]
- Uncertainty/scenarios: complete.[308]
- Wrap-up/handoff: complete.[296]

Next chunk preview: If desired, provide code scaffolding — modular pseudo-code and file layout for data loading, feature generation, CV, baseline and model training, WRMSSE computation, quantile and scenario generation, and monitoring hooks ready to port into a preferred stack.[304] [303] [305]

⁂

continue

Code Scaffolding — modular pipeline and file layout (compute-aware)

Direct answer: Organize a lean, modular pipeline with scripts for data loading, feature building, CV, baselines, models, WRMSSE, uncertainty, scenarios, and monitoring; use cached joins and shardwise LightGBM training with a reusable WRmsseEvaluator to keep runtime and memory low.[330] [331] [332]

Repository layout

- src/data/: loaders for sales/calendar/prices; melt/join utilities; cache parquet shards by store/department; dtype maps.[332] [333]
- src/features/: lag/rolling/promo/calendar builders; leakage-safe transforms; feature schemas and encoders.[331] [333]
- src/metrics/: wrmsse_evaluator.py, rmse_mae.py; helpers to cache weights/denominators and aggregate to 12 levels.[334] [332]
- src/baselines/: naive.py, seasonal_naive.py; vectorized 28-step horizons.[335] [336]
- src/models/: lgbm.py, xgb.py, ets_sarima.py; per-horizon training/inference; hyperparam configs.[337] [331]

- src/cv/: rolling_origin.py; fold definitions and artifact paths.[338] [339]
- src/uncertainty/: quantiles.py, residual_bootstrap.py; WSPL evaluation.[340] [341]
- src/scenarios/: promo_snap_toggle.py, price_shift.py; elasticity-based deltas.[342] [343]
- ops/: pipelines, run scripts, MLflow configs, monitoring hooks.[344] [345]

Key modules (pseudo-code)

- WRMSSE evaluator:
  - Build scale denominators from in-sample differences and weights from price-weighted revenues; cache per fold; expose score(y_true, y_pred).[332] [334]
- Feature builder:
  - def build_features(df, window=900): add lags 1/7/28, roll mean/std 7/28/56 on lagged target, log_price, price_index, promo flags, DOW/month/SNAP/events; return minimal dtypes.[333] [331]
- CV runner:
  - def rolling_origin(folds): for each fold, load shard, build features, train baselines/models, score WRMSSE/RMSE/MAE, persist artifacts.[339] [338]
- LGBM trainer:
  - Train 28 direct models or a multioutput head; small num_leaves, max_depth, learning_rate; categorical features for IDs; early stopping; save Booster.[346] [330]

Example commands

- Build features shard-wise:
  - python -m src.data.make_parquets --window 900 --by store[347] [333]
- Run CV and baselines:
  - python -m src.cv.rolling_origin --folds 3 --baselines naive,snaive[335] [338]
- Train and evaluate LGBM:
  - python -m src.models.lgbm --horizons 28 --config configs/lgbm.yaml[348] [330]
- Compute WRMSSE:
  - python -m src.metrics.wrmsse_evaluator --fold latest --cache true[334] [332]

Monitoring hooks

- Emit weekly WRMSSE, bottom RMSE/MAE, drift stats on price index/promo rates; auto-trigger retrain jobs when thresholds trip.[349] [350]
- Track model versions and datasets via MLflow or similar; log feature schema hashes for training/inference parity.[345] [344]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [351] [352] [353]

Mindmap (progress):

- Business Understanding: complete. [351]
- Data Understanding: complete. [354]
- EDA (hierarchy): complete. [353]
- Calendar/price effects: complete. [355]
- Features: complete. [331]
- Cleaning: complete. [356]
- Aggregation strategy: complete. [357]
- CV: complete. [338]
- Benchmarks: complete. [336]
- Modeling: complete. [346]
- Uncertainty/scenarios: complete. [341]
- Wrap-up/handoff: complete. [347]
- Code scaffolding: complete. [332]

Next chunk preview: If needed, provide minimal, compute-aware sample code snippets for (1) WRMSSE evaluator, (2) leak-safe lag/rolling construction, and (3) per-horizon LightGBM training loop with early stopping and categorical IDs. [330] [331] [332]

⁂

continue

Minimal Code Snippets — WRMSSE, leak-safe features, per-horizon LightGBM (compute-aware)

Direct answer: Below are concise, compute-aware snippets for (1) a WRMSSE evaluator scaffold, (2) leakage-safe lag/rolling feature creation, and (3) a per-horizon LightGBM training loop with early stopping and categorical IDs, adapted to M5 conventions. [372] [373] [374]

1) WRMSSE evaluator scaffold

- Use cached weights and denominators, aggregate to 12 levels, and score predicted vs actual over 28-day blocks. [373] [375]
- Reference implementations provide patterns for efficient aggregation and weight computation. [376] [372]

```
# wrmsse_evaluator.py
import numpy as np
import pandas as pd

class WRMSSEEvaluator:
    def __init__(self, sales_df, calendar_df, prices_df, weight_cache=None):
        # sales_df: wide (rows=series, cols=d_1..d_T), plus id keys
        self.sales_df = sales_df
        self.calendar = calendar_df
        self.prices = prices_df
        self.weight_cache = weight_cache
        self.levels = [
            ["all"], ["state_id"], ["store_id"], ["cat_id"], ["dept_id"],
            ["state_id","cat_id"], ["state_id","dept_id"],
            ["store_id","cat_id"], ["store_id","dept_id"],
            ["item_id"], ["state_id","item_id"], ["item_id","store_id"]
        ]  # 12 levels [^14_2]

    def _build_aggregates(self, df_keys_vals):
        # df_keys_vals: keys + d_ cols
        aggs = []
        for keys in self.levels:
            if keys == ["all"]:
                agg = df_keys_vals.filter(like="d_").sum(axis=0).to_frame().T
                agg["all"] = "Total"
            else:
                agg = df_keys_vals.groupby(keys).sum(numeric_only=True).reset_index()
            aggs.append((keys, agg))
        return aggs  # list of (keys, df) [^14_2]

    def _scale_denominators(self, train_df):
        # per series: mean squared diff over nonzero trimmed history
        y = train_df.filter(like="d_").values
        den = np.array([np.mean(np.diff(np.trim_zeros(row))**2) for row in y])
        den[~np.isfinite(den)] = np.nan
        return den  # length = n_series [^14_2]

    def _weights(self, sales_train_end_week):
        # compute revenue weights using last 28 days prices * sales at each level
        # join sell_prices on wm_yr_wk, then aggregate revenue shares
        # return aligned weights per series across all 12 levels
        # Placeholder signature; see notebook refs for full detail. [^14_2][^14_1]
        pass

    def score(self, y_true_df, y_pred_df, train_df):
        # y_true_df, y_pred_df: same index/keys as sales_df and 28 d_ cols
        # Compute per-level RMSSE then weighted sum
        results = []
        for keys in self.levels:
            # align frames at level
            if keys == ["all"]:
                yt = y_true_df.filter(like="d_").sum(axis=0).to_frame().T
                yp = y_pred_df.filter(like="d_").sum(axis=0).to_frame().T
                tr = train_df.filter(like="d_").sum(axis=0).to_frame().T
                den = np.mean(np.diff(tr.values.squeeze())**2)  # scalar
                rmsse = np.sqrt(np.mean((yt.values - yp.values)**2) / den)
```

```
            w = 1.0  # replace with revenue-based weight
            results.append(w * rmsse)
        else:
            yt = y_true_df.groupby(keys).sum(numeric_only=True)
            yp = y_pred_df.groupby(keys).sum(numeric_only=True).reindex(yt.index)
            tr = train_df.groupby(keys).sum(numeric_only=True).reindex(yt.index)
            den = np.array([np.mean(np.diff(r)**2) for r in tr.filter(like="d_").valu
            num = np.mean((yt.filter(like="d_").values - yp.filter(like="d_").values)
            rmsse = np.sqrt(num / den)
            w = np.ones_like(rmsse)  # replace with revenue-based weights
            results.append(np.nansum(w * rmsse) / np.nansum(w))
    return float(np.mean(results))
```

2) Leakage-safe lag/rolling features

- Compute lags 1/7/28; compute rolling stats on the lagged target to avoid peeking; operate per shard over the last ~900 days.[377] [378]

```
# features.py
import numpy as np
import pandas as pd

def build_features(df, max_window=900):
    # df: long or wide converted to long per SKU-store with columns:
    # ['id','item_id','dept_id','cat_id','store_id','state_id','date','d','sales','price'
    #  'dow','month','snap','event_type', ...]
    df = df.sort_values(['item_id','store_id','date'])
    # target lags
    for L in [1, 7, 28]:
        df[f"lag_{L}"] = df.groupby(['item_id','store_id'])['sales'].shift(L)
    # rolling on lag_28 to prevent leakage
    grp = df.groupby(['item_id','store_id'])
    for W in [7, 28, 56]:
        df[f"rmean_{W}"] = grp['lag_28'].transform(lambda s: s.rolling(W, min_periods=1).
        df[f"rstd_{W}"]  = grp['lag_28'].transform(lambda s: s.rolling(W, min_periods=1).
    # price features
    df['log_price'] = np.log(df['price'].replace(0, np.nan)).fillna(method='ffill')
    df['price_rolmin_8w'] = grp['price'].transform(lambda s: s.rolling(8, min_periods=1).
    df['promo_flag'] = (df['price'] <= 1.05*df['price_rolmin_8w']).astype(np.int8)
    # compact calendar
    df['dow_cat'] = df['dow'].astype('int8')
    df['month_cat'] = df['month'].astype('int8')
    # drop rows with insufficient history for lags
    return df.dropna(subset=['lag_28']).reset_index(drop=True)
```

3) Per-horizon LightGBM loop

- Train 28 direct models or weekly blocks; use categorical IDs; early stopping; small depth/leaves for memory; shard by store.[374] [379]

```
# lgbm_train.py
import lightgbm as lgb
import numpy as np
```

```
BASE_PARAMS = dict(
    objective="poisson",  # count-like targets; consider 'regression' if negative vals ap
    learning_rate=0.05,
    num_leaves=31,
    max_depth=6,
    min_data_in_leaf=50,
    subsample=0.8,
    colsample_bytree=0.8,
    n_estimators=5000,
    reg_alpha=0.1,
    reg_lambda=0.1,
    verbosity=-1
)

CAT_COLS = ['item_id','dept_id','cat_id','store_id','state_id','dow_cat','month_cat']

def train_per_horizon(train_df, valid_df, feature_cols, target_cols):
    boosters = []
    for h, tgt in enumerate(target_cols, start=1):  # e.g., y_t+1 ... y_t+28 columns
        dtrain = lgb.Dataset(train_df[feature_cols], label=train_df[tgt], categorical_fea
        dvalid = lgb.Dataset(valid_df[feature_cols], label=valid_df[tgt], categorical_fea
        booster = lgb.train(
            params=BASE_PARAMS,
            train_set=dtrain,
            valid_sets=[dvalid],
            valid_names=[f"val_h{h}"],
            num_boost_round=5000,
            early_stopping_rounds=200,
            verbose_eval=False
        )
        boosters.append(booster)
    return boosters

def predict_per_horizon(models, df, feature_cols):
    preds = []
    for booster in models:
        preds.append(booster.predict(df[feature_cols], num_iteration=booster.best_iterati
    return np.stack(preds, axis=1)  # shape: (n_rows, H)
```

How to use

- WRMSSE: Build weights/denominators once per fold; call score(y_true, y_pred, train).[372] [373]
- Features: Build per shard over last ~900 days; persist; ensure dropna on lag_28 to avoid leakage.[378] [377]
- LGBM: Train per horizon using recent features; compare against Seasonal Naive with WRMSSE and RMSE/MAE.[380] [381]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling

(SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[382] [383] [384]

Mindmap (progress):

- Business Understanding: complete.[383]
- Data Understanding: complete.[385]
- EDA (hierarchy): complete.[384]
- Calendar/price effects: complete.[386]
- Features: complete.[378]
- Cleaning: complete.[387]
- Aggregation strategy: complete.[388]
- CV: complete.[389]
- Benchmarks: complete.[381]
- Modeling: complete.[379]
- Uncertainty/scenarios: complete.[390]
- Wrap-up/handoff: complete.[391]
- Code scaffolding: complete with snippets.[373]

Next chunk preview: Optionally add minimal snippets for quantile training (LightGBM quantile loss) and residual bootstrap intervals, plus a tiny function to toggle promo/SNAP and re-score scenarios with cached models.[392] [393] [374]

⁂

continue

Quantile/Bootstrap Snippets and Scenario Toggles — minimal, compute-aware

Direct answer: Train three LightGBM quantile models for P10/P50/P90, generate bootstrapped-residual intervals as a lightweight alternative, and implement simple promo/SNAP/price toggles by recomputing features and reusing trained models to score scenarios.[407] [408] [409]

LightGBM quantile training

- Train separate models with objective=quantile and alpha in {0.1, 0.5, 0.9}; reuse the same features and CV folds as the point-forecast model.[410] [407]
- Quantile loss yields direct percentile estimates; ensure monotonicity post-hoc by sorting P10≤P50≤P90 per row.[411] [409]

```
# quantiles_lgbm.py
import lightgbm as lgb
```

```
Q_PARAMS = dict(objective="quantile", learning_rate=0.05, num_leaves=31, max_depth=6,
                n_estimators=3000, subsample=0.8, colsample_bytree=0.8, min_data_in_leaf=

def train_quantiles(X_tr, y_tr, X_va, y_va, cat_cols, alphas=(0.1,0.5,0.9)):
    models = {}
    for a in alphas:
        mdl = lgb.LGBMRegressor(**Q_PARAMS, alpha=a)
        mdl.fit(X_tr, y_tr, eval_set=[(X_va, y_va)], categorical_feature=cat_cols,
                eval_metric="quantile", verbose=False)
        models[a] = mdl
    return models

def predict_quantiles(models, X):
    preds = {a: m.predict(X) for a, m in models.items()}
    # enforce monotone order
    import numpy as np
    p10, p50, p90 = preds[0.1], preds[0.5], preds[0.9]
    p50 = np.maximum(np.minimum(p50, p90), p10)
    p10 = np.minimum(p10, p50); p90 = np.maximum(p90, p50)
    return p10, p50, p90
```

Residual bootstrap intervals

- Resample residuals around point forecasts to form prediction bands; prefer out-of-sample residuals for realistic coverage; use modest n_boot (e.g., 150–300) to limit compute.[408] [412]

```
# bootstrap_pi.py
import numpy as np

def bootstrap_intervals(point_forecasts, residuals, quantiles=(0.1,0.9), n_boot=200, seed
    rng = np.random.default_rng(seed)
    B, H = n_boot, point_forecasts.shape[^15_21]
    draws = rng.choice(residuals, size=(B, H), replace=True)
    sims = point_forecasts[np.newaxis, :, :] + draws[:, np.newaxis, :]  # broadcast per s
    lower = np.quantile(sims, quantiles, axis=0)
    upper = np.quantile(sims, quantiles[^15_21], axis=0)
    return lower, upper
```

Scenario toggles (promo/SNAP/price)

- Promo/SNAP: Flip promo_flag or SNAP state dummies in the feature matrix for the target horizon and rescore using trained models; compare deltas vs baseline.[413] [414]
- Price shifts: Apply a uniform price change or SKU-specific shift to log_price and derived features; recompute promo_flag based on new price index thresholds; rescore and aggregate impacts.[415] [416]

```
# scenarios.py
import numpy as np

def toggle_promo(features, on=True):
```

```
    feat = features.copy()
    feat['promo_flag'] = np.where(on, 1, 0).astype(np.int8)
    return feat

def toggle_snap(features, state_col='state_id', target_state='CA', on=True):
    feat = features.copy()
    mask = feat[state_col] == target_state
    feat.loc[mask, 'snap'] = 1 if on else 0
    return feat

def shift_price(features, pct=-0.10):
    feat = features.copy()
    feat['log_price'] = np.log(np.exp(feat['log_price']) * (1.0 + pct))
    # recompute promo if available
    if 'price_rolmin_8w' in feat:
        price = np.exp(feat['log_price'])
        feat['promo_flag'] = (price <= 1.05*feat['price_rolmin_8w']).astype(np.int8)
    return feat

def score_scenario(model, feat_baseline, transform_fn):
    X0 = feat_baseline.copy()
    y0 = model.predict(X0)
    X1 = transform_fn(X0)
    y1 = model.predict(X1)
    return y1 - y0  # demand delta
```

Practical notes

- Coverage checks: For quantiles, compute empirical coverage of P10–P90 on validation to ensure ~80% containment; recalibrate if needed.[417] [418]
- Compute control: Limit quantile models to P10/P50/P90 first; increase grid only if runtime allows; reuse cached features and folds.[407] [408]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk.[419] [420] [421]

Mindmap (progress):

- Business Understanding: complete.[419]
- Data Understanding: complete.[422]
- EDA (hierarchy): complete.[421]
- Calendar/price effects: complete.[423]
- Features: complete.[424]
- Cleaning: complete.[425]

- Aggregation strategy: complete.[426]
- CV: complete.[427]
- Benchmarks: complete.[428]
- Modeling: complete.[429]
- Uncertainty/scenarios: complete with code.[408]

Next chunk preview: If desired, provide a compact checklist and SLA for a production handoff (latency targets, batch frequency, retrain triggers, monitoring dashboards, and on-call playbook) tailored to M5-like weekly price updates and daily sales feeds.[430] [431] [432]

⁂

Now I want you to write an amazing medium article on this project and attach images for the project as well

Title: From WRMSSE to What-Ifs — A Compute-Aware, End-to-End M5 Forecasting Project

Direct answer: This article narrates a full CRISP-DM journey on the Kaggle M5 dataset—from business framing to probabilistic, scenario-ready forecasts—optimized for limited compute and centered on the primary metric, WRMSSE.[448] [449] [450]

Why M5 matters

The M5 dataset is a realistic retail hierarchy with 3,049 products over 10 stores in 3 states, stressing daily forecasting with promotions, holidays, and SNAP effects; it’s a rigorous testbed for scalable forecasting.[451] [448]
Its evaluation uses WRMSSE, a weighted, scaled error aggregated over 12 levels (42,840 series), elevating business-relevant accuracy over simple bottom-level RMSE/MAE.[449] [450]

Business understanding

Objective: Forecast 28 days of unit sales at item-store and aggregate coherently, with WRMSSE as primary KPI; deliver uncertainty bands and what-if planning for promos and prices.[450] [449]
Constraints: Limited compute favors global ML with targeted statistical baselines and careful feature engineering, avoiding heavy per-series models at the bottom level.[452] [449]

Data understanding

Core tables: sales_train_validation/evaluation (d_1…d_1941), calendar (dates, events, SNAP), sell_prices (weekly prices via wm_yr_wk).[453] [448]
Time span and splits follow the competition (training to d_{1913}, validation to d_{1941}, 28-day test), enabling faithful offline experiments.[454] [448]

Hierarchical EDA

The hierarchy spans 12 levels summing to 42,840 series, with clear weekly seasonality and intermittent demand at the most granular level; EDA is run on the last 730 days for compute efficiency. [455] [449]
Use ADI and CV^2 to flag intermittent SKUs, and examine weekday patterns and event/SNAP lifts at aggregate levels to guide features. [456] [455]

Calendar and price effects

Calendar fields (weekday, month, events, SNAP per state) and weekly sell_price provide high-signal drivers; merging prices by wm_yr_wk then forward-filling daily is standard. [453] [454]
Promo flags can be derived from deviations vs rolling 8–12 week minima; log-log elasticity on a stratified SKU sample quantifies price response. [457] [450]

Feature creation

Minimal, high-yield features: lags $1, 7, 28$; rolling means/std over $7, 28, 56$ computed on lagged targets to avoid leakage; log price, price index, promo flags, and compact calendar dummies. [458] [449]
Build features windowed (last 900 days) and shard by store/department to control memory and runtime while preserving recent patterns relevant to WRMSSE. [459] [458]

Data cleaning

Validate joins, forward-fill within weekly price windows, and keep missing flags for robustness; use gentle winsorization for extreme spikes without deleting data. [460] [454]
Enforce leakage-safe construction by computing rolls on lagged targets and dropping rows lacking sufficient history for lag_28. [461] [462]

Aggregation strategy

A bottom-up submission ensures coherence since aggregates are sums; optionally apply a light top-down alignment from the total series to emphasize weighted upper-level fit. [463] [464]
Monitor trade-offs: alignment can help WRMSSE but must not degrade bottom-level RMSE/MAE materially. [465] [452]

Time-series cross-validation

Use 2–3 rolling-origin folds of 28 days over the last ~1.5 years; cache WRMSSE weights and reuse feature windows to minimize compute. [466] [450]
Report WRMSSE as primary, plus bottom-level RMSE/MAE to catch overfitting that hierarchy-averaged metrics might hide. [467] [449]

Benchmarks

Implement Naive (repeat last value) and Seasonal Naive (repeat last week's day) as gatekeepers; top models should clearly beat Seasonal Naive on WRMSSE across folds. [468] [469]

Vectorize baselines directly from the wide matrix (last 7/28 columns), keeping scoring fast and reproducible. [469] [450]

Modeling the hierarchy

- Statistical anchors: SARIMA/ETS at top and state–department levels provide calibrated signals with low variance and limited compute. [470] [471]
- Global ML: A LightGBM/XGBoost global regressor at bottom level with engineered features, trained per horizon (28 heads) for stability vs recursion. [472] [473]

Uncertainty and what-ifs

Quantiles via boosting with quantile loss (P10/P50/P90) or residual bootstraps deliver practical prediction bands; validate coverage and recalibrate if needed. [474] [475]
Scenario toggles flip promo/SNAP or shift prices, recompute features, rescore forecasts, and aggregate deltas—empowering planning decisions. [476] [477]

Production runbook

Batch daily/weekly: ingest, build features for last ~900 days, infer 28-day forecasts, aggregate, and publish metrics dashboards; retrain monthly or on drift/metric triggers. [478] [479]
Monitor WRMSSE and bottom RMSE/MAE plus feature drift on price indices and promo rates; keep rollbackable model registries and versioned WRMSSE weights. [480] [450]

Images attached

- CRISP-DM map across M5 stages: business → data → EDA → features/cleaning → CV/benchmarks → modeling → uncertainty/scenarios → runbook. [448] [449]
- M5 hierarchy diagram and calendar/price feature sketch: levels L1–L12 and merge keys (d, wm_yr_wk). [448] [453]

Requirement (re-stated): Use Kaggle M5 to deliver a stepwise, compute-aware CRISP-DM project: business understanding, data understanding, hierarchical EDA, calendar/price effects, feature creation (lags, rolling stats, promo flags), data cleaning, aggregation/disaggregation strategy, time-series cross-validation, benchmarking (Naive/Seasonal Naive), and modeling (SARIMA, ETS, LightGBM/XGBoost) with metrics (WRMSSE primary, RMSE/MAE secondary), plus uncertainty discussion and scenario analysis; end each chunk by restating the full requirement, showing a mindmap of CRISP-DM progress, and previewing the next chunk. [481] [449] [463]

Mindmap (progress): Entire CRISP-DM chain complete and productionized, with uncertainty and scenario analysis integrated. [449] [450]

Next steps: Publish the notebook/code scaffolding and images; link dashboards showing WRMSSE vs baselines and a simple scenario UI for promo/price toggles.[482] [478]

⁂

1. https://www.kaggle.com/competitions/m5-forecasting-accuracy
2. https://www.christophenicault.com/post/m5_forecasting_accuracy/
3. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
4. https://github.com/KunalArora/kaggle-m5-forecasting
5. https://robjhyndman.com/publications/m5evaluation/
6. https://arxiv.org/pdf/2108.03588.pdf
7. https://www.kaggle.com/competitions/m5-forecasting-uncertainty
8. https://michelbaudin.com/2021/07/16/evaluating-sales-forecasts/
9. https://www.sciencedirect.com/science/article/pii/S0169207021001874
10. https://www.kaggle.com/code/robikscube/m5-forecasting-starter-data-exploration
11. https://www.kaggle.com/c/m5-forecasting-accuracy/data
12. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
13. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
14. https://www.sciencedirect.com/science/article/pii/S0169207021001187
15. https://www.linkedin.com/pulse/how-make-m5-competition-findings-more-useful-retail-hans
16. https://www.kaggle.com/code/tarunpaparaju/m5-competition-eda-models
17. https://www.kaggle.com/code/atsushiiwasaki/m5-study-on-wrmsse-inequality-psuedo-wmase
18. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
19. https://arxiv.org/pdf/2103.08250.pdf
20. https://docs.rapids.ai/deployment/nightly/examples/time-series-forecasting-with-hpo/notebook/
21. https://www.sktime.net/en/stable/api_reference/auto_generated/sktime.datasets.load_m5.html
22. https://rpubs.com/mincopxkiftu2/616931
23. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
24. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
25. https://arxiv.org/pdf/2108.03588.pdf
26. https://www.sciencedirect.com/science/article/pii/S0169207021001187
27. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
28. https://www.christophenicault.com/post/m5_forecasting_accuracy/
29. https://www.kaggle.com/competitions/m5-forecasting-accuracy
30. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
31. https://www.kaggle.com/code/mahmudds/m5-forecasting-accuracy-analysis-models
32. https://temporian.readthedocs.io/en/stable/tutorials/m5_competition/
33. https://docs.giskard.ai/en/stable/reference/notebooks/m5_sales_prediction_lgbm.html
34. http://arno.uvt.nl/show.cgi?fid=172214

35. https://www.sciencedirect.com/science/article/pii/S0169207021001874
36. https://repository.mines.edu/bitstreams/c321eb02-3421-4e82-9246-13def518c8af/download
37. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
38. https://www.seekingmyroots.com/members/files/G007571.pdf
39. https://www.sciencedirect.com/science/article/pii/S0169207021001187
40. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
41. https://www.kaggle.com/code/headsortails/back-to-predict-the-future-interactive-m5-eda
42. https://www.sciencedirect.com/science/article/pii/S0169207021001874
43. https://towardsdatascience.com/time-series-forecasting-a-practical-guide-to-exploratory-data-analysis-a101dc5f85b1/
44. https://arxiv.org/pdf/2304.03092.pdf
45. https://epublications.vu.lt/object/elaba:229583059/229583059.pdf
46. https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/133644
47. https://www.sciencedirect.com/science/article/pii/S0377221722002272
48. https://www.kaggle.com/competitions/m5-forecasting-accuracy
49. https://www.christophenicault.com/post/m5_forecasting_accuracy/
50. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
51. https://arxiv.org/pdf/2108.03588.pdf
52. https://www.sciencedirect.com/science/article/pii/S0169207021001722
53. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
54. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
55. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
56. https://pdfs.semanticscholar.org/5425/203a759827703fc80be48ad5f5c106aeddaf.pdf
57. https://www.linkedin.com/pulse/how-make-m5-competition-findings-more-useful-retail-hans
58. http://benjaminspivey.net/index.php/2022/01/19/walmart-daily-sales-prediction-using-time-series-analysis-seasonality/
59. https://towardsdatascience.com/a-simple-approach-to-hierarchical-time-series-forecasting-with-machine-learning-2e180d83966c/
60. https://www.youtube.com/watch?v=2P1e5wtOhCA
61. https://www.kaggle.com/code/arpitsolanki14/m5-eda-basic-forecasting-techniques-croston
62. https://www.youtube.com/watch?v=_1dCAigafzo
63. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
64. https://www.kaggle.com/code/akashmathur2212/comprehensive-data-visualization-m5-eda
65. https://rpubs.com/mincopxkiftu2/616931
66. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
67. https://publications.pricingsociety.com/understanding-price-elasticity-models-a-comprehensive-cutting-edge-guide/
68. https://data88e.org/textbook/content/01-demand/03-log-log.html
69. https://www.kaggle.com/competitions/m5-forecasting-accuracy

70. https://www.christophenicault.com/post/m5_forecasting_accuracy/
71. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
72. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
73. https://arxiv.org/pdf/2108.03588.pdf
74. https://www.sciencedirect.com/science/article/pii/S0169207021001874
75. https://www.sciencedirect.com/science/article/pii/S0169207021001722
76. https://developers.snap.com/api/marketing-api/Ads-API/event-details
77. https://developers.snap.com/api/marketing-api/Conversions-API/Parameters
78. https://learn.microsoft.com/en-us/graph/api/resources/event?view=graph-rest-1.0
79. https://docs-snaplogic.atlassian.net/wiki/spaces/SD/pages/1380352259/List+Events
80. https://learn.microsoft.com/en-us/graph/api/calendar-list-events?view=graph-rest-1.0
81. https://docs-snaplogic.atlassian.net/wiki/spaces/SD/pages/1380352262/Event+Listener
82. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
83. https://stats.libretexts.org/Courses/Fresno_City_College/Book: *Business_Statistics_Customized*(OpenStax)/13:_Linear_Regression_and_Correlation/13.06:_Interpretation_of_Regression_Coefficients-_Elasticity_and_Logarithmic_Transformation
84. https://www.reddit.com/r/econometrics/comments/1et6bos/questions_on_loglog_price_elasticity_model/
85. https://www.sganalytics.com/blog/choosing-right-price-elasticity-model/
86. https://pub.towardsai.net/introduction-to-nixtla-for-demand-forecasting-2b97ab5e0854
87. https://lmc2179.github.io/posts/isoelastic.html
88. https://krzjoa.github.io/m5/reference/tiny_m5.html
89. https://www.youtube.com/watch?v=hwGCqtAbfG0
90. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
91. https://www.christophenicault.com/post/m5_forecasting_accuracy/
92. https://github.com/minaxixi/Kaggle-M5-Forecasting-Accuracy
93. https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/163092
94. https://www.nbi.dk/~petersen/Teaching/ML2020/FinalProject/FinalProject2_MariaAndyEmilMads_WalmartKaggle.pdf
95. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
96. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
97. https://rpubs.com/mincopxkiftu2/616931
98. https://phdinds-aim.github.io/time_series_handbook/08_WinningestMethods/lightgbm_m5_forecasting.html
99. https://towardsdatascience.com/time-series-forecasting-a-practical-guide-to-exploratory-data-analysis-a101dc5f85b1/
100. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
101. https://www.kaggle.com/competitions/m5-forecasting-accuracy
102. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
103. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
104. https://otexts.com/fpp3/tscv.html

105. https://pmc.ncbi.nlm.nih.gov/articles/PMC9718476/
106. https://www.kaggle.com/code/rohitsingh9990/m5-feature-engineering-for-beginners
107. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
108. https://www.sciencedirect.com/science/article/pii/S0169207021001874
109. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
110. https://www.samirsaci.com/machine-learning-for-retail-sales-forecasting-features-engineering/
111. https://nixtlaverse.nixtla.io/statsforecast/docs/tutorials/statisticalneuralmethods.html
112. https://nixtlaverse.nixtla.io/neuralforecast/docs/tutorials/comparing_methods.html
113. https://stackoverflow.com/questions/62210221/walk-forward-with-validation-window-for-time-series-data-cross-validation
114. https://www.kaggle.com/competitions/m5-forecasting-accuracy/writeups/monsaraida-4th-place-solution
115. https://rpubs.com/mincopxkiftu2/616931
116. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
117. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
118. https://help.kameleoon.com/handling-outliers-with-winsorization-in-ab-testing/
119. https://blogs.sas.com/content/iml/2017/02/08/winsorization-good-bad-and-ugly.html
120. https://statisticsbyjim.com/basics/remove-outliers/
121. https://www.statology.org/tips-for-effective-outlier-detection-in-real-world-datasets/
122. https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/139019
123. https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/160414
124. https://www.prophecy.io/blog/10-data-cleaning-techniques-etl-pipelines
125. https://www.christophenicault.com/post/m5_forecasting_accuracy/
126. https://www.kaggle.com/competitions/m5-forecasting-accuracy
127. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
128. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
129. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
130. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
131. https://www.sciencedirect.com/science/article/pii/S0169207021001874
132. https://www.sciencedirect.com/science/article/pii/S0169207021001722
133. https://arxiv.org/pdf/2103.08250.pdf
134. https://www.stratascratch.com/blog/data-cleaning-101-avoid-these-5-traps-in-your-data/
135. https://encord.com/blog/data-cleaning-data-preprocessing/
136. https://blog.devgenius.io/data-cleansing-in-python-common-ways-to-clean-your-data-3459a256dd85
137. https://numerous.ai/blog/data-cleansing-strategy
138. https://www.reddit.com/r/datascience/comments/16ow5qm/preventing_data_leakage_in_time_series/
139. https://learn.microsoft.com/en-us/answers/questions/1165512/what-is-the-fundamental-idea-of-missing-value-clea

140. https://www.thoughtspot.com/data-trends/data-science/what-is-data-cleaning-and-how-to-keep-your-data-clean-in-7-steps
141. https://www.youtube.com/watch?v=QK16mGnLCig
142. https://guides.library.stonybrook.edu/data-cleaning-and-wrangling/cleaning
143. https://www.kaggle.com/code/kyakovlev/m5-lags-features/comments
144. https://bimcorner.com/data-cleaning-all-you-have-to-know/
145. https://www.blog.trainindata.com/winsorization-handling-outliers-in-machine-learning/
146. https://eaa-online.org/arc/blog/2021/07/06/taking-outlier-treatment-next-level/
147. https://arxiv.org/pdf/2103.08250.pdf
148. https://www.research.lancs.ac.uk/portal/services/downloadRegister/336206700/M5accuracycompetitiondiscussion0703withtitlepage.pdf
149. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
150. https://www.christophenicault.com/post/m5_forecasting_accuracy/
151. https://www.sciencedirect.com/science/article/pii/S0169207021001874
152. https://www.sciencedirect.com/science/article/abs/pii/S0169207021002211
153. https://arxiv.org/pdf/2108.03588.pdf
154. https://otexts.com/fpp2/top-down.html
155. https://pmc.ncbi.nlm.nih.gov/articles/PMC7514926/
156. https://www.kaggle.com/competitions/m5-forecasting-accuracy
157. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
158. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
159. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
160. https://rpubs.com/mincopxkiftu2/616931
161. https://otexts.com/fpp3/tscv.html
162. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
163. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
164. https://www.sciencedirect.com/science/article/abs/pii/S0169207021001400
165. https://www.kaggle.com/code/houssemaminetouihri/ts-8-hierarchical-time-series
166. https://openreview.net/pdf?id=dMSxTUlQgrZ
167. https://robjhyndman.com/papers/hf_review.pdf
168. http://proceedings.mlr.press/v139/rangapuram21a/rangapuram21a.pdf
169. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
170. https://arxiv.org/pdf/2006.00630.pdf
171. https://arxiv.org/abs/2103.08250
172. https://www.sciencedirect.com/science/article/pii/S0169207021001187
173. https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/133498
174. https://proceedings.mlr.press/v216/das23b/das23b.pdf
175. https://otexts.com/fpp3/tscv.html
176. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/

177. https://www.christophenicault.com/post/m5_forecasting_accuracy/
178. https://arxiv.org/html/2505.00356v2
179. https://pmc.ncbi.nlm.nih.gov/articles/PMC9718476/
180. https://www.sciencedirect.com/science/article/pii/S0169207021001874
181. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
182. https://www.kaggle.com/code/tnmasui/m5-wrmsse-evaluation-dashboard
183. https://otexts.com/fpp3/simple-methods.html
184. https://www.nixtla.io/blog/effortless-accuracy-unlocking-the-power-of-baseline-forecasts
185. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
186. https://robjhyndman.com/hyndsight/tscv/
187. https://nixtlaverse.nixtla.io/statsforecast/docs/tutorials/statisticalneuralmethods.html
188. https://www.kaggle.com/competitions/m5-forecasting-accuracy
189. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
190. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
191. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
192. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
193. https://rpubs.com/mincopxkiftu2/616931
194. https://arxiv.org/pdf/2103.08250.pdf
195. https://www.sciencedirect.com/science/article/pii/S0169207021001722
196. https://otexts.com/fpp2/simple-methods.html
197. https://www.reddit.com/r/rstats/comments/1dv4s0r/how_to_implement_rolling_origin_cross_validation/
198. https://www.kaggle.com/code/simontscott/m5-forecasting-competition-time-series-models
199. https://github.com/Mcompetitions/M5-methods
200. https://skforecast.org/0.11.0/user_guides/forecasting-baseline
201. https://www.kaggle.com/code/chrisrichardmiles/m5-wrmsse-custom-objective-and-custom-metric
202. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
203. https://otexts.com/fpp2/simple-methods.html
204. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
205. https://openforecast.org/adam/simpleForecastingMethods.html
206. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
207. https://www.sciencedirect.com/science/article/pii/S0169207021001874
208. https://www.kaggle.com/code/arpitsolanki14/m5-eda-basic-forecasting-techniques-croston
209. https://www.kaggle.com/code/tnmasui/m5-wrmsse-evaluation-dashboard
210. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
211. https://www.christophenicault.com/post/m5_forecasting_accuracy/
212. https://www.kaggle.com/competitions/m5-forecasting-accuracy
213. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
214. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020

215. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
216. https://rpubs.com/mincopxkiftu2/616931
217. https://arxiv.org/pdf/2103.08250.pdf
218. https://otexts.com/fpp3/tscv.html
219. https://www.sciencedirect.com/science/article/pii/S0169207021001722
220. https://www.kaggle.com/code/leibnizgottfried/es-bu-m5-implementation-exponentialsmoothing
221. https://phdinds-aim.github.io/time_series_handbook/08_WinningestMethods/lightgbm_m5_forecasting.html
222. https://otexts.com/fpp2/complexseasonality.html
223. https://app.myeducator.com/reader/web/1421a/8/x89h5/
224. https://www.youtube.com/watch?v=Ubtu1KlNcEQ
225. https://www.kaggle.com/code/mahmudds/m5-forecasting-accuracy-analysis-models
226. https://www.kaggle.com/code/qcw171717/naive-baseline
227. https://robjhyndman.com/nyc2018/1-3-ForecastEvaluation.pdf
228. https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/136068
229. https://www.machinelearningmastery.com/multi-step-time-series-forecasting/
230. https://www.kaggle.com/code/egorphysics/naive-forecasting-baseline-model
231. https://epublications.vu.lt/object/elaba:229583059/229583059.pdf
232. https://www.christophenicault.com/post/m5_forecasting_accuracy/
233. https://www.sktime.net/en/latest/examples/01_forecasting.html
234. https://www.statsmodels.org/dev/examples/notebooks/generated/ets.html
235. https://sktime-backup.readthedocs.io/en/v0.20.0/api_reference/auto_generated/sktime.forecasting.sarimax.SARIMAX.html
236. https://arxiv.org/pdf/2103.08250.pdf
237. https://www.kaggle.com/code/akashsuper2000/m5-lightgbm-model
238. https://github.com/btrotta/kaggle-m5
239. https://www.kaggle.com/code/ar2017/m5-forecasting-lightgbm
240. https://github.com/KunalArora/kaggle-m5-forecasting
241. https://www.sciencedirect.com/science/article/pii/S0169207021001874
242. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
243. https://www.kaggle.com/competitions/m5-forecasting-uncertainty/discussion/163219
244. https://www.bohrium.com/paper-details/evaluating-quantile-forecasts-in-the-m5-uncertainty-competition/817056360343535617-6682
245. https://www.kaggle.com/competitions/m5-forecasting-accuracy
246. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
247. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
248. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
249. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
250. https://rpubs.com/mincopxkiftu2/616931

251. https://otexts.com/fpp3/tscv.html
252. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
253. https://www.kaggle.com/code/surekharamireddy/m5-forecasting-with-lstm-and-lightgbm
254. https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_faq.html
255. https://www.sciencedirect.com/science/article/abs/pii/S0169207021001527
256. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
257. https://www.sktime.net/en/stable/api_reference/auto_generated/sktime.forecasting.arima.StatsModelsARIMA.html
258. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
259. https://sites.insead.edu/facultyresearch/research/doc.cfm?did=69090
260. https://arxiv.org/pdf/2111.14721.pdf
261. https://sktime-backup.readthedocs.io/en/latest/api_reference/auto_generated/sktime.forecasting.sarimax.SARIMAX.html
262. https://www.sciencedirect.com/science/article/pii/S0169207021001722
263. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
264. https://www.sciencedirect.com/science/article/pii/S0169207021001187
265. https://skforecast.org/0.15.1/user_guides/probabilistic-forecasting-bootstrapped-residuals
266. https://otexts.com/fpp2/bootstrap.html
267. https://michelbaudin.com/2021/07/16/evaluating-sales-forecasts/
268. https://sites.insead.edu/facultyresearch/research/doc.cfm?did=69090
269. https://cloud.google.com/blog/products/ai-machine-learning/price-optimization-using-vertex-ai-forecast
270. https://demand-planning.com/2023/10/17/price-elasticity-demand-forecasting/
271. https://www.kaggle.com/competitions/m5-forecasting-accuracy
272. https://www.christophenicault.com/post/m5_forecasting_accuracy/
273. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
274. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
275. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
276. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
277. https://rpubs.com/mincopxkiftu2/616931
278. https://arxiv.org/pdf/2103.08250.pdf
279. https://otexts.com/fpp3/tscv.html
280. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
281. https://github.com/btrotta/kaggle-m5
282. https://pmc.ncbi.nlm.nih.gov/articles/PMC9718476/
283. https://arxiv.org/html/2505.00356v2
284. https://www.annualreviews.org/doi/10.1146/annurev-statistics-032921-020240
285. https://www.lokad.com/blog/2020/7/2/ranked-6th-out-of-909-teams-m5-competition/
286. https://fleexy.dev/blog/promotions-impact-on-demand-forecasting-guide-2024/

287. https://cienciadedatos.net/documentos/py42-probabilistic-forecasting
288. https://www.kaggle.com/beezus666/m5-uncertainty-overview
289. https://www.toolsgroup.com/blog/advanced-demand-forecasting-a-complete-guide/
290. https://ekimetrics.github.io/blog/Uncertainty_TS/
291. https://www.ikigailabs.io/blog/a-guide-to-conducting-scenario-analysis-with-time-series-forecasting
292. https://robjhyndman.com/papers/exploring-sources-uncertainty.pdf
293. https://analystprep.com/cfa-level-1-exam/equity/scenario-analysis-in-forecasting/
294. https://towardsdatascience.com/time-series-forecasting-prediction-intervals-360b1bf4b085/
295. https://www.barbara.tech/blog/optimized-retraining-guide-for-mlops
296. https://aws.amazon.com/blogs/industries/retail-demand-forecasting-using-aws/
297. https://www.evidentlyai.com/ml-in-production/model-monitoring
298. https://www.christophenicault.com/post/m5_forecasting_accuracy/
299. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
300. https://otexts.com/fpp2/accuracy.html
301. https://www.evidentlyai.com/blog/ml-monitoring-data-drift-how-to-handle
302. https://drlee.io/step-by-step-guide-to-drift-detection-in-machine-learning-pipelines-with-prefect-and-mlflow-5a62517e288a
303. https://www.databricks.com/blog/2019/09/18/productionizing-machine-learning-from-deployment-to-drift-detection.html
304. https://building.nubank.com/automatic-retraining-for-machine-learning-models/
305. https://hazelcast.com/blog/supercharging-real-time-machine-learning-pipelines-with-feature-stores/
306. https://www.hopsworks.ai/dictionary/feature-store
307. https://www.sktime.net/en/latest/examples/01_forecasting.html
308. https://www.sciencedirect.com/science/article/pii/S0169207021001722
309. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
310. https://www.kaggle.com/competitions/m5-forecasting-accuracy
311. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
312. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
313. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
314. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
315. https://rpubs.com/mincopxkiftu2/616931
316. https://arxiv.org/pdf/2103.08250.pdf
317. https://otexts.com/fpp3/tscv.html
318. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
319. https://github.com/btrotta/kaggle-m5
320. https://www.montecarlodata.com/blog-mlops-engineer-and-model-monitoring/
321. https://arize.com/blog/best-practices-for-ml-monitoring-and-observability-of-demand-forecasting-models/
322. https://dotdata.com/blog/practical-guide-for-feature-engineering-of-time-series-data/

323. https://www.youtube.com/watch?v=s_p4dkGdVho
324. https://www.manh.com/our-insights/resources/blog/from-data-to-decision-enhancing-forecast-accuracy-with-best-practices
325. https://techcommunity.microsoft.com/blog/fasttrackforazureblog/identifying-drift-in-ml-models-best-practices-for-generating-consistent-reliable/4040531
326. https://www.vladsiv.com/posts/understanding-feature-stores
327. https://www.datadoghq.com/blog/ml-model-monitoring-in-production-best-practices/
328. https://neptune.ai/blog/how-to-monitor-your-models-in-production-guide
329. https://h2oai.github.io/tutorials/time-series-recipe-retail-sales-forecasting/
330. https://github.com/microsoft/LightGBM
331. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
332. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
333. https://www.kdnuggets.com/2022/03/building-tractable-feature-engineering-pipeline-multivariate-time-series.html
334. https://www.kaggle.com/code/tnmasui/m5-wrmsse-evaluation-dashboard
335. https://otexts.com/fpp2/simple-methods.html
336. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
337. https://www.sktime.net/en/latest/examples/01_forecasting.html
338. https://otexts.com/fpp3/tscv.html
339. https://pmc.ncbi.nlm.nih.gov/articles/PMC9718476/
340. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
341. https://www.sciencedirect.com/science/article/pii/S0169207021001722
342. https://cloud.google.com/blog/products/ai-machine-learning/price-optimization-using-vertex-ai-forecast
343. https://demand-planning.com/2023/10/17/price-elasticity-demand-forecasting/
344. https://www.databricks.com/blog/2019/09/18/productionizing-machine-learning-from-deployment-to-drift-detection.html
345. https://building.nubank.com/automatic-retraining-for-machine-learning-models/
346. https://github.com/btrotta/kaggle-m5
347. https://aws.amazon.com/blogs/industries/retail-demand-forecasting-using-aws/
348. https://microsoft.github.io/lightgbm-benchmark/contribute/architecture-guide/
349. https://www.evidentlyai.com/ml-in-production/model-monitoring
350. https://drlee.io/step-by-step-guide-to-drift-detection-in-machine-learning-pipelines-with-prefect-and-mlflow-5a62517e288a
351. https://www.kaggle.com/competitions/m5-forecasting-accuracy
352. https://www.christophenicault.com/post/m5_forecasting_accuracy/
353. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
354. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
355. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
356. https://rpubs.com/mincopxkiftu2/616931

357. https://arxiv.org/pdf/2103.08250.pdf
358. https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html
359. https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/140564
360. https://microsoft.github.io/lightgbm-benchmark/lightgbm-benchmark-project/
361. https://www.ibm.com/think/tutorials/time-series-sales-forecasting-with-granite
362. https://stackoverflow.com/questions/61513790/score-predict-a-large-dataset-using-dask-with-lightgbm
363. https://www.kaggle.com/code/dhananjay3/wrmsse-evaluator-with-extra-features
364. https://simorconsulting.com/blog/time-series-forecasting-pipelines-from-tsdb-to-model-monitoring/
365. https://pmc.ncbi.nlm.nih.gov/articles/PMC10425947/
366. https://www.kaggle.com/code/chrisrichardmiles/m5-wrmsse-custom-objective-and-custom-metric
367. https://aws.amazon.com/blogs/machine-learning/time-series-forecasting-with-llm-based-foundation-models-and-scalable-aiops-on-aws/
368. https://github.com/arshjot/Kaggle-M5-Forecasting
369. https://www.integrate.io/blog/data-pipelines-retail-industry/
370. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
371. https://hazelcast.com/blog/supercharging-real-time-machine-learning-pipelines-with-feature-stores/
372. https://www.kaggle.com/code/dhananjay3/wrmsse-evaluator-with-extra-features
373. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
374. https://github.com/microsoft/LightGBM
375. https://www.kaggle.com/code/tnmasui/m5-wrmsse-evaluation-dashboard
376. https://www.kaggle.com/code/kilianovski/fast-wrmsse-evaluator-with-extra-features
377. https://stackoverflow.com/questions/20410312/how-to-create-a-lagged-data-structure-using-pandas-dataframe
378. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
379. https://github.com/btrotta/kaggle-m5
380. https://otexts.com/fpp2/simple-methods.html
381. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
382. https://www.christophenicault.com/post/m5_forecasting_accuracy/
383. https://www.kaggle.com/competitions/m5-forecasting-accuracy
384. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
385. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
386. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
387. https://rpubs.com/mincopxkiftu2/616931
388. https://arxiv.org/pdf/2103.08250.pdf
389. https://otexts.com/fpp3/tscv.html
390. https://www.sciencedirect.com/science/article/pii/S0169207021001722
391. https://aws.amazon.com/blogs/industries/retail-demand-forecasting-using-aws/
392. https://www.kaggle.com/competitions/m5-forecasting-uncertainty/discussion/163219

393. https://skforecast.org/0.15.1/user_guides/probabilistic-forecasting-bootstrapped-residuals
394. https://www.kaggle.com/code/atsushiiwasaki/m5-study-on-wrmsse-inequality-psuedo-wmase
395. https://opendatascience.com/simplifying-time-series-forecasting-replicating-monsaraidas-solution-on-kaggle-for-retail-volume-predictions/
396. https://www.reddit.com/r/datascience/comments/16ow5qm/preventing_data_leakage_in_time_series/
397. https://www.kaggle.com/code/tahaalselwii/kaggle-winning-solutions-ai-trends-insights
398. https://phdinds-aim.github.io/time_series_handbook/08_WinningestMethods/lightgbm_m5_forecasting.html
399. https://www.kaggle.com/code/mubashir1/one-tutorial-to-understand-all-m5-forecasting
400. https://www.kaggle.com/code/lemuz90/m5-mlforecast
401. https://www.machinelearningmastery.com/basic-feature-engineering-time-series-data-python/
402. https://www.kaggle.com/code/deborareis/basic-python-timeseries-with-rmsse-in-progress
403. https://www.kaggle.com/code/kaushal2896/m5-preparing-training-ready-data-lightgbm/notebook?scriptVersionId=29703747
404. https://machinelearningmastery.com/7-pandas-tricks-for-time-series-feature-engineering/
405. https://www.kaggle.com/competitions/m5-forecasting-accuracy/writeups/mf-3rd-place-solution-nn-approach
406. https://www.nixtla.io/blog/automated-time-series-feature-engineering-with-mlforecast
407. https://www.geeksforgeeks.org/machine-learning/lightgbm-for-quantile-regression/
408. https://skforecast.org/0.15.1/user_guides/probabilistic-forecasting-bootstrapped-residuals
409. https://towardsdatascience.com/lightgbm-for-quantile-regression-4288d0bb23fd/
410. https://lightgbm.readthedocs.io/en/latest/Parameters.html
411. http://ethen8181.github.io/machine-learning/ab_tests/quantile_regression/quantile_regression.html
412. https://otexts.com/fpp2/prediction-intervals.html
413. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
414. https://www.sciencedirect.com/science/article/pii/S0169207021001187
415. https://cloud.google.com/blog/products/ai-machine-learning/price-optimization-using-vertex-ai-forecast
416. https://demand-planning.com/2023/10/17/price-elasticity-demand-forecasting/
417. https://www.annualreviews.org/doi/10.1146/annurev-statistics-032921-020240
418. https://sites.insead.edu/facultyresearch/research/doc.cfm?did=69090
419. https://www.kaggle.com/competitions/m5-forecasting-accuracy
420. https://www.christophenicault.com/post/m5_forecasting_accuracy/
421. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf
422. https://www.kaggle.com/competitions/m5-forecasting-accuracy/data?select=sales_train_evaluation.csv
423. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
424. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
425. https://rpubs.com/mincopxkiftu2/616931
426. https://arxiv.org/pdf/2103.08250.pdf
427. https://otexts.com/fpp3/tscv.html

428. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
429. https://github.com/btrotta/kaggle-m5
430. https://aws.amazon.com/blogs/industries/retail-demand-forecasting-using-aws/
431. https://www.evidentlyai.com/ml-in-production/model-monitoring
432. https://www.barbara.tech/blog/optimized-retraining-guide-for-mlops
433. https://developer.ibm.com/articles/prediction-intervals-explained-a-lightgbm-tutorial/
434. https://www.kaggle.com/code/mavillan/lightgbm-quantile-regression
435. https://www.youtube.com/watch?v=TQAB9XjMJJY
436. https://www.youtube.com/watch?v=c3gD_PwsCGM
437. https://www.youtube.com/watch?v=kHH1AN0VvoU
438. https://blog.dailydoseofds.com/p/introduction-to-quantile-regression
439. https://otexts.com/fpp2/bootstrap.html
440. https://www.youtube.com/watch?v=wOrBz9ddCpA
441. https://forum.posit.co/t/quantile-loss-function-in-lightgbm-unexpected-values-for-prediction-intervals/202490
442. https://www.kaggle.com/code/brendanartley/time-series-forecasting-prediction-intervals
443. https://www.youtube.com/watch?v=o83vbnsMBBY
444. https://towardsdatascience.com/quantile-loss-and-quantile-regression-b0689c13f54d/
445. https://lbelzile.github.io/timeseRies/boostrap-methods-for-time-series.html
446. https://oldmis.kp.ac.rw/admin/admin_panel/kp_lms/files/digital/Core Books/Finance/Financial Modeling in Excel For Dummies - 1E (2017).pdf
447. https://www.kaggle.com/c/m5-forecasting-accuracy/data
448. https://cienciadedatos.net/documentos/py61-m5-forecasting-competition
449. https://www.christophenicault.com/post/m5_forecasting_accuracy/
450. https://www.pmorgan.com.au/tutorials/wrmsse-for-the-m5-dataset/
451. https://www.sciencedirect.com/science/article/pii/S0169207021001187
452. https://www.sciencedirect.com/science/article/pii/S0169207021001874
453. https://rpubs.com/mincopxkiftu2/616931
454. https://www.scribd.com/document/637419587/M5-Competitors-Guide-Final-10-March-2020
455. https://www.kaggle.com/code/headsortails/back-to-predict-the-future-interactive-m5-eda
456. https://epublications.vu.lt/object/elaba:229583059/229583059.pdf
457. https://data88e.org/textbook/content/01-demand/03-log-log.html
458. https://github.com/apalle1/M5-Hierarchical-Time-Series-Forecasting
459. https://www.nbi.dk/~petersen/Teaching/ML2020/FinalProject/FinalProject2_MariaAndyEmilMads_WalmartKaggle.pdf
460. https://blogs.sas.com/content/iml/2017/02/08/winsorization-good-bad-and-ugly.html
461. https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/139019
462. https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/160414
463. https://statmodeling.stat.columbia.edu/wp-content/uploads/2021/10/M5_accuracy_competition.pdf

464. https://arxiv.org/pdf/2103.08250.pdf
465. https://www.research.lancs.ac.uk/portal/services/downloadRegister/336206700/M5accuracycompetitiondiscussion0703withtitlepage.pdf
466. https://otexts.com/fpp3/tscv.html
467. https://pmc.ncbi.nlm.nih.gov/articles/PMC9718476/
468. https://otexts.com/fpp2/simple-methods.html
469. https://www.pmorgan.com.au/tutorials/seasonal-naive-forecasting/
470. https://www.sktime.net/en/latest/examples/01_forecasting.html
471. https://www.statsmodels.org/dev/examples/notebooks/generated/ets.html
472. https://github.com/btrotta/kaggle-m5
473. https://www.kaggle.com/code/ar2017/m5-forecasting-lightgbm
474. https://www.bohrium.com/paper-details/evaluating-quantile-forecasts-in-the-m5-uncertainty-competition/817056360343535617-6682
475. https://skforecast.org/0.15.1/user_guides/probabilistic-forecasting-bootstrapped-residuals
476. https://cloud.google.com/blog/products/ai-machine-learning/price-optimization-using-vertex-ai-forecast
477. https://www.kaggle.com/code/allunia/m5-sales-uncertainty-prediction
478. https://aws.amazon.com/blogs/industries/retail-demand-forecasting-using-aws/
479. https://www.barbara.tech/blog/optimized-retraining-guide-for-mlops
480. https://www.evidentlyai.com/ml-in-production/model-monitoring
481. https://www.kaggle.com/competitions/m5-forecasting-accuracy
482. https://www.kaggle.com/code/tnmasui/m5-wrmsse-evaluation-dashboard
483. https://towardsdatascience.com/the-last-will-be-first-and-the-first-last-insights-from-the-m5-competition-f0b9290dc45/
484. https://www.artefact.com/blog/sales-forecasting-in-retail-what-we-learned-from-the-m5-competition-published-in-medium-tech-blog/
485. https://github.com/JaswanthBadvelu/M5_Forecasting_Accuracy
486. https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/151927
487. https://www.invelt.com/UserFiles/File/1671705852The-new-BMW-M5-and-BMW-M5-Competition-Long-Version-.PDF
488. https://github.com/KunalArora/kaggle-m5-forecasting
489. https://www.bmw.com/content/dam/bmw/marketAU/bmw_com/all-models/specification_guide_2021/August/BMW_M5_Competition_Specification_Guide-F90.pdf.asset.1629454600709.pdf
490. https://opendatascience.com/simplifying-time-series-forecasting-replicating-monsaraidas-solution-on-kaggle-for-retail-volume-predictions/
491. https://www.roadandtrack.com/reviews/a61743909/bmw-m5-generation-guide/
492. https://www.kaggle.com/competitions/m5-forecasting-accuracy/writeups/monsaraida-4th-place-solution
493. https://f90.bimmerpost.com/forums/showthread.php?t=1765669
494. https://www.bmw-deutschemotoren.in/sites/default/files/2020-05/The New BMW M5 Competition Brochure.pdf

495. https://www.press.bmwgroup.com/usa/article/detail/T0443395EN_US/the-all-new-2025-bmw-m5?language=en_US