Demand Forecaster – Analyst Guide

1. Introduction

This guide is for analysts who want to customize, extend, and deeply understand the Demand Forecaster package. It explains the modeling approach, customization options, and advanced recipes included in the package.

2. Core Modeling Approach

- Historical patterns are captured using a **weighted moving average or median** of comparable periods.
- Comparable periods are defined by frequency:
• Daily: same weekday over user-defined past N periods
• Weekly: same week number across past N months
• Monthly: same month across available years
- Seasonal growth factors (monthly, weekly, yearly) are applied as multipliers.
- Promotions, discounts, and event/holiday effects are incorporated when datasets are provided.

3. Customization Options

- User can specify lookback window (N periods) per frequency.
- Choice of aggregation: **weighted moving average** (user-defined weights) or **median**.
- Growth adjustment: turn on/off or override with custom uplift factors.
- Categorical grouping supported (Region, Store, Item, etc.).

4. Advanced Recipes

The demo_notebook.ipynb includes advanced examples:
- Custom weights: Analyst can pass arrays of weights per frequency.
- Trimmed median: Remove outliers before median calculation.
- Machine Learning uplift: Fit regression/XGBoost to estimate promo or weather impacts.
- Weather factor: Merge sales with weather dataset for adjusted forecasts.
- Hierarchical overrides: Analyst can aggregate forecasts at Region/Store/Item level and apply top-down reconciliation.

5. Analyst Workflow

1. Explore historical sales patterns in Jupyter Notebook.
2. Define lookback windows and choose weighted moving average or median.
3. Add event/promotion datasets (map via dates).
4. Apply optional ML uplift or external factors (e.g., weather).
5. Run forecast and validate accuracy with historical holdout sets.
6. Share outputs with planners via CSV or BI tool dashboards.

6. Forecast Outputs

Forecasts are saved as CSV with columns: date, region, store, item, forecast. Intermediate calculations (growth factor, uplift applied) can be enabled for debugging.

7. Package Internals

The demand_forecaster package consists of modules: - data_loader.py: reads and validates datasets - forecaster.py: main forecasting logic - adjustments.py: growth, promotion, and event uplift logic - cli.py: command-line interface This modular structure makes it easy for analysts to extend the solution.

8. Extensibility

Analysts can extend the model by: - Adding new uplift factors (e.g., macroeconomic indicators) - Plugging in ML models (Prophet, XGBoost, LSTM) for selected SKUs - Overriding reconciliation logic for multi-level forecasts - Integrating with external APIs for real-time factors