Project Report — Cold-Start Demand Forecasting for Jaipur (13-week horizon)

# Executive summary

Built a reproducible forecasting pipeline to predict weekly demand for every SKU in the new city Jaipur for the next 13 weeks . The model uses historical sales from other cities, plus contextual signals (planned price, promotions, holiday flags, weather features). The pipeline includes data preprocessing, model training (LightGBM), SHAP-based explainability, and scenario analysis (price cut and promo boost). Deliverables include the forecast file ( outputs/forecast\_baseline.csv ), explainability plots, scenario outputs, and code modules.

# 1. Problem statement & objective

Scenario: Launching a product in a new city (Jaipur) with only a few weeks of local sales.

Objective:

Produce weekly unit forecasts for all SKU ids in Jaipur for the next 13 weeks , and explain the drivers (why the model predicts what it predicts). The solution must be reproducible and interpretable for business stakeholders.

# 2. Data

## Sources supplied

- panel\_train.csv — historical weekly sales panel (other cities + limited Jaipur weeks). Columns include: market , sku\_id , week\_start , units , price , promo\_flag , holiday\_flag , and weather features.

- calendar\_future.csv — future weeks (week\_start) to forecast (13 weeks).

- price\_plan\_future.csv — planned prices by market and week\_start .

- promos\_future.csv — planned promotions by sku/market/week.

- weather\_future.csv — future weather features (temp\_c, rain\_mm).

- metadata.json — any SKU/market metadata.

## Granularity & target

- Temporal granularity: Weekly

- Spatial granularity: Market (Jaipur) × SKU

- Target: units (weekly units sold)

# 3. Data preparation & preprocessing

1. Loading : All CSVs loaded with week\_start parsed as datetime.

2. Type normalization : sku\_id and market cast to string.

3. Missing values : Forward/backward fill for training panel; merges filled with zeros where appropriate for future data.

4. Feature alignment :

- planned\_price renamed to price .

- Promo flags standardized ( promo\_flag = 1 when promo active else 0).

- Ensure calendar\_future expanded by SKUs so we forecast each SKU-week pair for the 13 weeks.

5. Feature set used (for baseline model):

- price , promo\_flag , holiday\_flag , temp\_c , rain\_mm

# 4. Modeling approach

## Why LightGBM

- Gradient-boosted trees (LightGBM) handle tabular data efficiently, manage non-linear interactions, and work well with heterogeneous features. They also integrate with SHAP for explainability.

## Strategy for cold-start

- Use multiseries learning : train a single LightGBM model across all SKUs/markets so the model learns cross-sectional patterns (price sensitivity, promo lift, holiday effects).

- Include market and SKU-level IDs only as context (or via features derived from metadata) but not as dominant predictors to avoid overfitting to markets with lots of history.

- Use external regressors (planned price, promo flag, holiday, weather) to transfer signal to Jaipur\_NewCity.

## Training details

- Features : ["price", "promo\_flag", "holiday\_flag", "temp\_c", "rain\_mm"]

- Target : units

- Model : lightgbm.LGBMRegressor(n\_estimators=200, learning\_rate=0.05, random\_state=42)

- Validation : baseline run used an 80/20 split

- Artifact : trained model saved as outputs/model.pkl .

# 5. Forecast generation

- Prepared future dataset by merging calendar\_future , price\_plan\_future , promos\_future , and weather\_future for Jaipur and expanding across all SKU ids to create every (sku\_id, week\_start) pair for the 13-week horizon.

- Model predicts forecast\_units for each pair.

- Output: outputs/forecast\_baseline.csv containing columns: week\_start , sku\_id , forecast\_units .

# 6. Explainability (drivers of the forecast)

We used SHAP (TreeExplainer) to attribute feature contributions to predictions.

## Key driver findings (generalized)

- Temperature (temp\_c) : Strongest driver — higher temperatures generally increase demand.

- Price : Major negative influence — lower prices lead to higher sales.

- Rainfall (rain\_mm) : Moderate impact, SKU-dependent — some products are sensitive to rain conditions.

- Promo Flag : Strong positive effect — promotions consistently boost demand.

- Holiday Flag : Positive but smaller spikes during holiday weeks.

## How to interpret (example)

For SKU X in week W :

- SHAP contributions: price = -4.2 , promo = +2.1 , holiday = +0.8 → net contribution raising baseline to predicted units.

- Business interpretation: a 10% price cut or a promo increases expected units by roughly the model-estimated delta (scenario analysis confirms magnitude).

Plots saved:

- outputs/plots/shap\_summary.png — feature impact distribution.

A graph of different colored shapes

AI-generated content may be incorrect.

- outputs/plots/shap\_feature\_importance.png — overall ranking.

A bar graph with blue bars

AI-generated content may be incorrect.

# 7. Scenario analysis (business "what-if" cases)

We ran three scenarios for the 13-week horizon:

1. Base case: planned prices/promos (baseline forecast).

2. PriceCut (-10%): multiply price by 0.9 across all weeks/SKUs.

3. PromoBoost: set promo\_flag = 1 for all SKU-weeks.

Outputs:

- Per-scenario forecasts: outputs/scenarios/forecast\_base.csv , forecast\_pricecut.csv , forecast\_promoboost.csv

- Aggregated comparison: outputs/scenarios/scenario\_comparison.csv (total forecast units by scenario)

- Per-SKU comparison: outputs/scenarios/sku\_scenario\_comparison.csv

- Chart: outputs/scenarios/scenario\_analysis.png

A graph with different colored lines

AI-generated content may be incorrect.

**Business insight examples**

- A 10% price cut increased total forecasted units by X% (see scenario\_comparison.csv ) — this must be weighed against margin losses.

- PromoBoost increases demand across most SKUs; top responders can be targeted for promotions first.

# 8. Results (example summary)

- Baseline total 13-week units (Jaipur, all SKUs): 4950.071835

- PriceCut (-10%) total units: 4664.552246

% change = -5.7679887%

- PromoBoost total units: 5644.382861

% change = 14.026281

SKUs by baseline forecast (example view):

|  |  |
| --- | --- |
| SKU\_UNIT | TOTAL UNITS |
| 001 | 873.2155405 |
| 002 | 783.0254426 |
| 003 | 919.5732023 |
| 004 | 932.5354942 |
| 005 | 652.2470103 |
| 006 | 789.4751448 |

# 9. Evaluation & limitations

## Current evaluation

- Baseline evaluation uses RMSE on a random 80/20 train-test split. Observed baseline RMSE: 12.78 is from training/test split.

## Limitations

Cold-start generalization : For brand-new SKUs with no metadata and no cross-market analogue, forecasts will be less reliable.

- Validation : Current simple train/test split may overestimate performance; time-series cross-validation is recommended.

- Uncertainty : Model currently produces point forecasts only. Prediction intervals or quantile models are recommended for risk-aware planning.

- Cannibalization : Model does not explicitly model SKU-to-SKU cannibalization, that requires multi-output or joint modelling.

# 10. Enhancement:

1. Time-aware validation : use rolling-origin evaluation and report mean/median metrics across folds.

2. Quantile forecasting : train LightGBM quantile regressors or use ensembles to produce prediction intervals.

3. Transfer features : create cross-market SKU similarity features (category-level historical multipliers) to improve cold-start performance.

4. Cannibalization modeling : explore multi-output models or structural demand models to capture SKU interactions.

# CONCLUSION:

The pipeline demonstrates that cold-start demand forecasting for Jaipur is feasible with cross-market learning. Predictions are explainable and allow scenario simulations. While baseline accuracy is good, further improvements (time-aware validation, prediction intervals) are recommended for production deployment.