1. Role of Data Preprocessing

Overview of the Forecasting Pipeline

PIPELINE OVERVIEW

INPUT → STEP 1 → STEP 2 → OUTPUT

Raw Data
Sales, Calendar, Prices

Cleaned & Linked
Joins, Date Parsing

Engineered Features
Lags, Rolling Stats

Model Ready
Training Dataset

What the pipeline does

Systematically **cleans** errors, **links** separate data sources, and **enriches** the raw logs with mathematical patterns needed for forecasting.

Why it is required

Raw retail data is noisy and disconnected. Preprocessing **reduces noise**, aligns timelines, and adds context to **prevent data leakage**.

2. Raw Datasets Used

Overview of the primary data sources required for retail forecasting

Sales Data

KEY COLUMNS

- `item_id`
- `store_id`
- `day (d_1, d_2...)`
- `units_sold`

PURPOSE

*Defines historical demand: **what** was sold, **where**, and **how much**.*

Calendar Data

KEY COLUMNS

- `day (d_1...)`
- `date (YYYY-MM-DD)`
- `weekday`
- `event_name / type`

PURPOSE

*Provides temporal context: connects abstract "Day 1" to real dates and **holidays**.*

Sell Prices

KEY COLUMNS

- `item_id`
- `store_id`
- `wm_yr_wk (week id)`
- `sell_price`

PURPOSE

Tracks value: Price changes often drive demand shifts and promotions.

Data Flow Overview

End-to-end transformation pipeline from raw sources to model-ready features

RAW INPUTS

- **sales_train_validation.csv** — Daily unit sales
- **calendar.csv** — Dates & Events
- **sell_prices.csv** — Price variations

TRANSFORMATION

Preprocessing Pipeline

- ✓ Merge & Link
- ✓ Reshape (Melt)
- ✓ Feature Engineering

FINAL OUTPUT

READY FOR ML

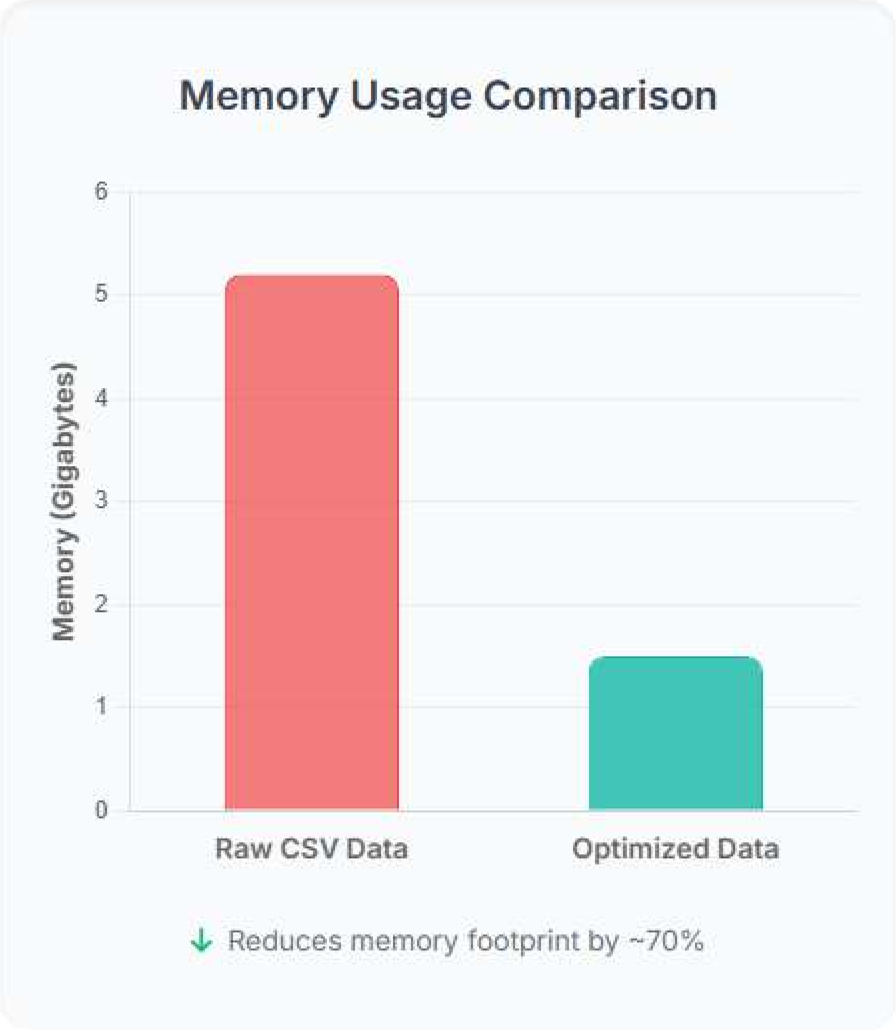
processed_dataset.pkl

A memory-optimized, feature-rich dataset ready for LightGBM training.

Binary Format | Optimized

3. Why Large Retail Data Needs Memory Optimization

Handling millions of rows efficiently before processing



The Challenge

Retail datasets contain millions of transactions (rows) multiplied by thousands of items. Loading raw CSV files directly often causes **Out of Memory (OOM)** errors on standard computers.

Optimization Actions

Downcast Numeric Types

Convert 64-bit numbers to 16/32-bit (e.g., `float64` → `float32`). We don't need 15 decimal places for sales counts.

Categorical Encoding

Store repeated strings (like "State_CA") as compact integers internally, saving massive string overhead.

Why this matters

Ensures the entire dataset fits in RAM, prevents crashes, and speeds up subsequent feature engineering steps.

Why Large Retail Data Needs Memory Optimization

Handling Millions of Transactions Efficiently

1. Raw Data Problem - 40+ GB
2. Memory Issues - Computer Crashes
3. Processing Speed - Hours to Load
4. Solution - Smart Data Types & Batching

4. Dataset Linkage Strategy

Merging Disconnected Sources into a Single Timeline

DATA INTEGRATION

Join Keys (Logic)

Sales Calendar using `day_id`

Sales Prices using `item, store, week`

Why Linkage is Critical

Machine learning models require a **single consistent view**. Linking ensures every "unit sold" is matched with its exact selling price and whether that day was a holiday.

Dataset Linkage Strategy

Connecting sales, price, and calendar data for forecasting

DATA PREPROCESSING

Relationship Diagram

calendar.csv
Time features, ev
SNAP detail

sell_prices.csv
eekly price per
store/item

CENTRAL HUB
sales_train.csv
Historical Unit Sales

PRIMARY JOIN KEYS

`id` `item_id` `store_id` `state_id` `wm_yr_wk`

ID Structure Analysis

Every row in the dataset is uniquely identified by a composite string containing hierarchical information.

EXAMPLE ID: FOODS_1_018_CA_1_EVALUATION

FOODS	1	018	CA	1	evaluation
Category	Dept	Item	State	Store	Data Split

Why this structure matters:

- Allows aggregation at multiple levels (State, Store, Category, Department).
- `store_id` + `item_id` forms the crucial link to pricing data.

5. Wide Format vs Long Format Sales Data

DATA TRANSFORMATION

Reshaping data structure for effective machine learning

↔ Wide Format (Raw Input)

ITEM_ID	D_1	D_2	...	D_1913
HOBBIES_1	0	0	...	1
HOBBIES_2	0	1	...	0
... 30,000 items ...				

- **Hard to Link:** Cannot easily join "Day 1" with a calendar date column.
- **Bad for ML:** Models expect a fixed number of features, but days keep increasing.
- **Structure:** 1 row per item, many columns.

VS

↕ Long Format (Target)

ITEM_ID	DAY	SALES
HOBBIES_1	d_1	0
HOBBIES_1	d_2	0
HOBBIES_1	d_3	0
... millions of rows ...		

- **Easy Linkage:** "Day" becomes a key to join with Calendar/Prices.
- **ML Ready:** Standard "Samples x Features" matrix format.
- **Structure:** 1 row per transaction (Item-Day).

DECISION:

Convert to Long Format

Enables Joins
Merges seamlessly with Calendar & Prices

Time Features
Allows calculating lags (t-1, t-7)

Model Input
Standard input shape for Forecasting Models

6. Time-Based Feature Engineering

Extracting Demand Signals from Dates

FEATURE ENGINEERING

↓ Decomposition of Raw 'Date' Column

Basic Components
Fundamental timeline tracking

Month (1-12) | Year | Day of Month

Weekly Cycles
Captures shopping habits

Day of Week | Week Number | Is Weekend?

Holidays & Events
Impactful external factors

Event Name | Event Type | SNAP / Payday

Seasonality
Long-term weather/trends

Quarter (1-4) | Season | Day Index

25

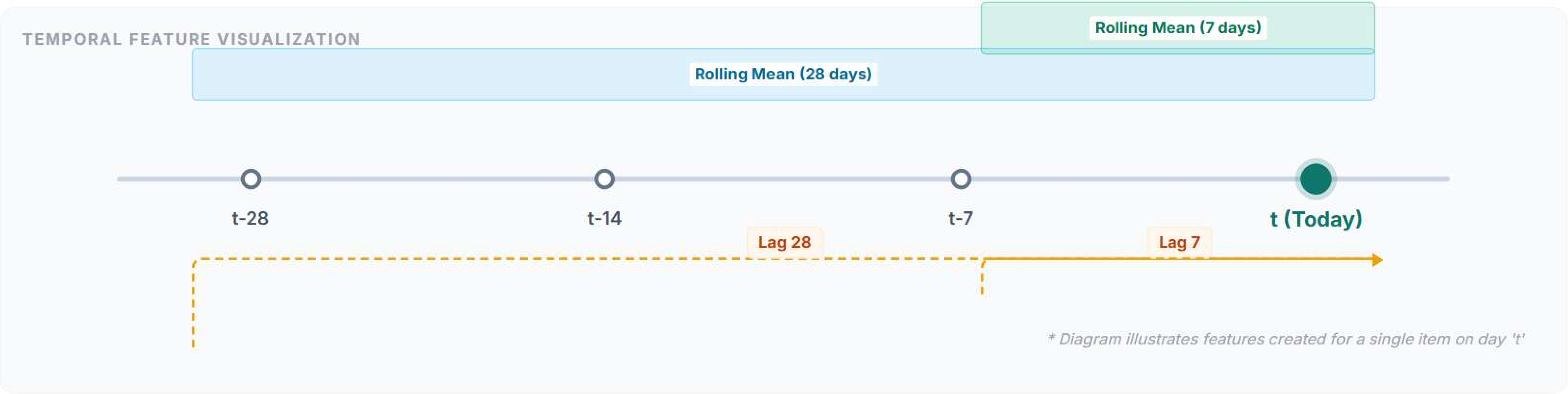
"Machine Learning models cannot understand a raw date like '2016-12-25'. They need numbers representing concepts."

Why it matters

- Retail sales follow strict **weekly cycles** (e.g., Saturday peak).
- **Holidays** cause massive demand spikes that models miss without explicit flags.
- Numerical features allow the model to learn **recurring patterns**.

7. Lag and Rolling Window Features

Teaching the model to look back in time



Lag Features

Definition: The exact sales value from specific days in the past (e.g., exactly 7 days ago).

Why it helps: Captures *seasonality*. If sales are high every Saturday, the "Lag 7" feature tells the model to expect high sales this Saturday.

Rolling Window Features

Definition: Statistical summaries (Mean, Max, Std Dev) over a past window (e.g., average of last 28 days).

Why it helps: Captures *recent trends* and stabilizes predictions. If the average is rising, the model predicts an increase.

Lag and Rolling Window Features

Learning from Past Sales History

Step 1

Lag Features - Sales from 1, 7, 28 days ago

Step 2

Rolling Average - Average sales over past 7, 30 days

Step 3

Rolling Statistics - Min, max, standard deviation of past sales

Feature Engineering Overview

Transforming raw time-series data into predictive signals

Lag Features

28 Days

35 Days

42 Days

"Captures recent sales volume trends directly"

Rolling Means

7 Days (Short)

28 Days (Medium)

60 Days (Long)

"Smooths volatility to reveal underlying patterns"

Calendar

Day of Week

Month / Year

Weekend Flag

SNAP / Holidays

"Encodes critical seasonality and event markers"

Price Momentum

Price Change %

Max/Min Ratio

Std. Deviation

"Quantifies price elasticity and discount impact"

8. Handling Categorical Variables

Translating Text Labels into Machine-Readable Signals

DATA PREPARATION

PROCESS 1: LABEL ENCODING

Category: "Hobbies" TEXT

↓

Category_ID: 1 INT

PROCESS 2: ADDING CONTEXT

Store_ID: "CA_3"

Look up historical average...

Avg. Daily Sales `2,450 units`

Why convert to numbers?

Machine Learning models operate on mathematical matrices. They cannot multiply or subtract text strings like "California" or "Foods". We must map every unique text label to a unique integer.

Grouping adds context

Converting "Store A" to "1" is necessary, but providing the **"Average Sales of Store A"** gives the model a stronger signal about that store's typical performance magnitude.

CRITICAL PREPROCESSING RULE

When calculating group averages (e.g., average sales per store), use **only past data**. Using future data causes "Data Leakage" and invalidates the model.

9. Handling Zero-Inflated Demand

Solving the "Intermittent Sales" Problem

DATA ENGINEERING

The Distribution Problem

Frequency of Daily Units Sold

Insight: Most days have 0 sales. A standard model treats 0 as just a "low number," failing to capture that 0 is a distinct state (No Stock / No Demand).

ENGINEERED SIGNALS

Zero Flag (Binary)

Explicitly tells the model: "Was there a sale today?" (0 or 1).

Recency Gap

"Days since the last non-zero sale." Helps predict when the next sale is due.

Non-Zero Rate

Rolling probability of a sale occurring (e.g., "Sold on 20% of days last month").

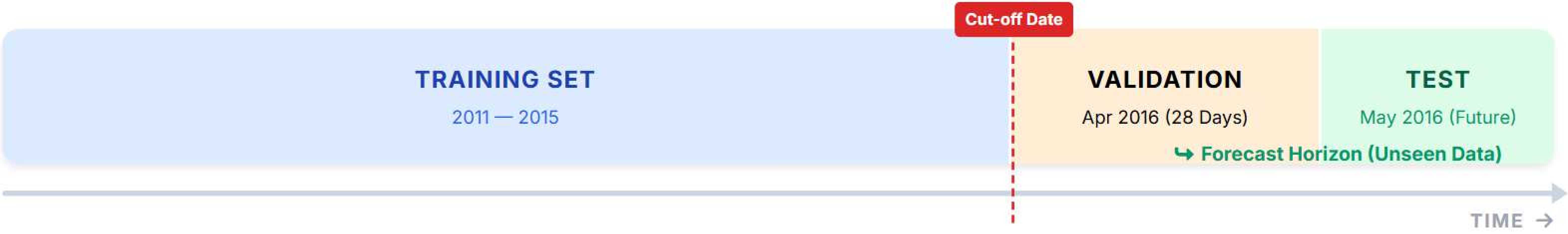
Why it matters

Without these specific hints, regression models tend to "hedge" their bets and predict impossible values like **0.1 units**. These features help the model separate "Zero Days" from "Sales Days."

10. Preventing Data Leakage

Time-Based Splitting Strategy

MODEL VALIDATION



Split by Time, Not Randomly

Never shuffle. Unlike image classification, sales data depends on sequence. We must train on the past to predict the future, preserving the order of events.

Block Future Information

No peeking. Features for a specific day (e.g., "7-day average") must strictly use data *prior* to that day. Using next week's sales to predict today is "leakage."

Realistic Validation

Mimic production. The gap between Training and Validation ensures our model is tested exactly how it will be used in the real world: predicting the unknown.

11. Final Processed Dataset

Structure of the Model-Ready Input Matrix

PIPELINE OUTPUT

SINGLE ROW STRUCTURE (ITEM-STORE-DAY)

IDS & KEYS		TIME FEATURES			PRICE	EVENTS		HISTORY		TARGET
item_id	store_id	day_int	wday	month	sell_price	event_name	snap_CA	lag_7	roll_28	sales (y)
1432	3	1854	6	2	2.48	0	1	5.0	3.2	4
1432	3	1855	7	2	2.48	1	1	4.0	3.1	?

Target Variable (What we predict) Categorical Encoded Engineered Numeric

Dataset Specs

GRANULARITY

Item × Store × Day

TOTAL FEATURES

~35 Columns

DATA TYPES

Int16 / Float32

Memory Optimized

MISSING VALUES

Handled (0 or Imputed)

MODEL READY

VALIDATED FOR TRAINING

Offline Pre-Aggregation

Optimizing the data pipeline architecture for performance and scalability.

Join & Clean

Merge raw CSVs and handle missing values.

Feature Build

Generate lags, rolling means, and calendar features.

Compress

Downcast types and optimize memory usage.

Save Dataset

Export optimized pickle for rapid loading.

Why Offline?

Critical for large-scale retail datasets

Faster Loading

Eliminates repetitive processing time during model training iterations.

Reduced RAM

Optimized datatypes significantly lower runtime memory footprint.

Performance

Enables rapid experimentation and smoother application performance.

Handling Retail-Specific Challenges

Key complexities addressed during data preprocessing

Intermittent Demand

Prevalence of Zero Sales Days

Retail data often contains many days with 0 units sold. Requires specific handling (e.g., Tweedie loss) rather than standard regression.

Hierarchical Structure

Multi-Level Aggregation

Data exists at Item → Department → Category → Store → State levels. Trends at aggregate levels may differ from individual item trends.

Seasonal Patterns

Event & Holiday Effects

Significant sales spikes driven by events (Super Bowl, Christmas, SNAP days). Mapped via `calendar.csv` features.

State & Store Variations

Local Price & Demand Dynamics

Prices and preferences vary by location (CA, TX, WI). 30,490 distinct product-store combinations must be modeled.

12. Why This Pipeline Improves Accuracy

From Raw Data to High-Performance Signals

PERFORMANCE IMPACT

Noise Reduction

Handling missing values and zero-inflated demand prevents the model from learning "garbage" patterns.

Context Awareness

Correctly linking prices and calendar events explains *why* sales spikes happen (e.g., promotions).

Stronger Signals

Lags and rolling windows allow the model to "see" trends and seasonality directly.

Honest Validation

Time-based splitting ensures the accuracy we measure is realistic and reliable for future use.

FORECAST ERROR (RMSE)

~30% Error Reduction

Result: A model-ready dataset that yields higher, more stable accuracy and faster training.

PREPROCESSING PIPELINE OUTPUT

Final Processed Dataset

processed_dataset.pkl

Serialized Python Object

`2.4 GB` `Pickle Protocol 4`

Dataset Contents

Clean, linked data
Merged from sales, calendar, and pricing sources

Engineered features
Lag variations (28-42 days) & Rolling windows (7-60 days)

Encoded categoricals
Label encoding for products, stores, and events

Memory-optimized dtypes
Downcasted numerics for efficient loading

STATUS

Ready for ML Model Training