Reproducible Market Analytics Pipeline

Coffee (Arabica) Time-Series Analysis with LLM-Assisted Reporting

January 26, 2026

1 Introduction

This document describes the design, implementation, and rationale of a reproducible data science pipeline developed for analysing Coffee (Arabica) market time-series data. The primary goals of the pipeline are:

- to provide a clean, automated, and reproducible analytical workflow,
- to support iterative experimentation and extension,
- to separate data processing, modelling, and reporting concerns,
- and to explore the integration of a local large language model (LLM) as an analytical reporting layer.

The pipeline is orchestrated using `Snakemake` and is designed to be executed end-to-end with a single command.

2 High-Level Pipeline Overview

The workflow follows a linear but modular structure:

$$\texttt{fetch} \rightarrow \texttt{store} \rightarrow \texttt{features} \rightarrow \texttt{model} \rightarrow \texttt{report}$$

Each stage produces explicit artifacts that serve as inputs to downstream steps. This ensures transparency, reproducibility, and ease of debugging.

3 Data Ingestion

3.1 Objective

The ingestion step retrieves raw Coffee (Arabica) price data from the Federal Reserve Economic Data (FRED) database and converts it into a clean, validated time series.

3.2 Implementation

Data is fetched via an HTTP request to the FRED CSV endpoint. The ingestion logic enforces a strict schema consisting of:

- `date`: timestamp
- `value`: numeric price

Non-numeric values and missing observations are removed, and the data is sorted chronologically.

3.3 Design Rationale

This step is intentionally minimal. No transformations beyond validation and cleaning are performed, ensuring that raw data remains interpretable and auditable.

4 Data Storage

4.1 Objective

Persist the cleaned time series in a lightweight relational format to act as a stable checkpoint for downstream analysis.

4.2 Implementation

The validated CSV is written into a SQLite database table. The table is replaced on each run to guarantee reproducibility.

4.3 Design Rationale

Using SQLite provides:

- a realistic production-style storage interface,
- decoupling between ingestion and feature engineering,
- and a convenient boundary for future extensions (e.g., multiple assets or metadata tables).

5 Feature Engineering

5.1 Objective

Transform the raw time series into a supervised learning dataset suitable for predictive modelling.

5.2 Feature Construction

The following transformations are applied:

- Log price transformation (ensuring strictly positive values),
- Log returns computed as first differences of log prices,
- Lagged return features,
- Rolling-window statistics (mean and standard deviation of returns),
- One-step-ahead log return as the prediction target.

Rows containing missing values introduced by lagging or rolling operations are removed.

5.3 Design Rationale

This feature set provides a simple but interpretable baseline for time-series modelling. It avoids domain-specific assumptions while capturing short- and medium-term temporal structure.

6 Modelling and Evaluation

6.1 Objective

Train and evaluate a baseline predictive model for next-step log returns.

6.2 Model Choice

A Ridge regression model is used as a baseline. The model is intentionally simple, providing:

- interpretability,
- numerical stability,
- and a reference point for future, more complex models.

6.3 Evaluation Protocol

A time-ordered train/test split is applied to avoid information leakage. Model performance is evaluated using:

- Mean Absolute Error (MAE),
- Root Mean Squared Error (RMSE).

Both metrics and pointwise predictions are saved as versioned artifacts.

7 LLM-Assisted Reporting

7.1 Objective

Automate the generation of concise analytical market reports based on validated model outputs.

7.2 Architecture

Rather than exposing raw data to the LLM, a compact summary bundle is constructed containing:

- model metadata and performance metrics,
- recent prediction error statistics,
- a small window of recent predictions.

This bundle is passed to a locally hosted LLM via a structured prompt with strict constraints on tone, content, and formatting.

7.3 Design Rationale

This approach ensures:

- reproducibility and grounding of generated text,
- avoidance of speculation or hallucination,
- separation between analytical computation and narrative generation.

The LLM functions as a reporting layer, not a decision-making component.

8 Workflow Orchestration

All steps are orchestrated using `Snakemake`. Each rule declares explicit inputs and outputs, allowing:

- automatic dependency resolution,
- incremental recomputation,
- full pipeline execution with a single command.

9 Limitations and Future Extensions

Potential future improvements include:

- richer feature sets or alternative targets,
- probabilistic or non-linear models,
- uncertainty estimation,
- multi-asset support,
- more structured LLM outputs (e.g., JSON summaries),
- evaluation of LLM-generated text quality.

The current design prioritises clarity, reproducibility, and extensibility over model sophistication.

10 Conclusion

This pipeline serves as a reusable template for reproducible time-series analytics. Its modular design enables iterative improvement while maintaining transparency and automation. The integration of an LLM as a reporting layer demonstrates a practical, constrained approach to combining traditional data science workflows with modern language models.