1. Pipeline Architecture & Documentation

This document details the end-to-end data flow, explaining how raw data is transformed into a volatility prediction.

1.1 High-Level Pipeline Diagram

The system follows a linear pipeline architecture, ensuring data consistency from ingestion to inference.

1.2 Pipeline Stages Description

Stage 1: Data Ingestion

- **Source**: The system reads the dataset.csv file.
- **Action**:
    - The CSV is loaded into a Pandas DataFrame.
    - Column names are standardized (e.g., crypto_name, date, close).
- **Output**: A raw DataFrame containing historical OHLCV data.

Stage 2: Preprocessing

- **Objective**: Prepare raw data for analysis.
- **Steps**:
    1. **Type Casting**: The date column is converted to datetime objects to enable time-series operations.
    2. **Sorting**: Data is sorted by crypto_name and then date. This is critical; unsorted data would lead to incorrect volatility calculations (e.g., calculating changes between dates out of order).
    3. **Missing Value Handling**: Rows with NaN values (often found at the very beginning of a dataset after lag calculations) are dropped.

Stage 3: Feature Engineering

- **Objective**: Create numerical features that represent market volatility.
- **Key Features Created**:
    1. **Log Returns**: $\ln(P_t / P_{t-1})$. This normalizes price changes, making them comparable across different assets and price ranges.
    2. **Rolling Volatility (7-Day)**: The rolling standard deviation of Log Returns over the last 7 days. This is the primary input feature.
    3. **Liquidity Ratio**: Calculated as Volume / Market Cap. This helps the model understand if a price move is backed by significant trading activity.
    4. **Previous Close**: A lag feature allowing the model to see the most recent price level.

Stage 4: Model Training & Evaluation

- **Algorithm**: Random Forest Regressor.
- **Splitting Strategy**: **Time-Series Split**.

    - *Why?* You cannot use random shuffling for time-series data because using future data to predict the past (data leakage) would create a fake "perfect" score. We strictly use the first 80% of dates for training and the last 20% for testing.
- **Target Variable**: The model attempts to predict the volatility_7d for the *next* time step (simulated in the code by shifting the target variable).

Stage 5: Deployment (Inference)

- **Interface**: Streamlit Web App.
- **Flow**:
    1. User selects a cryptocurrency.
    2. App filters the dataset for that specific asset.
    3. App calculates the latest available technical indicators.
    4. Model predicts the likely volatility level for the upcoming week.
    5. Results are displayed as interactive Plotly charts.