Pipeline Architecture Cryptocurrency Volatility Prediction System

**Contents**

1. [Overview](#_bookmark0) 2
2. [Pipeline Stages](#_bookmark1) 2
   1. [Data Ingestion](#_bookmark2) 2
   2. [Data Preprocessing](#_bookmark3) 2
   3. [Exploratory Data Analysis (EDA)](#_bookmark4) 2
   4. [Model Training](#_bookmark5) 2
   5. [Model Evaluation](#_bookmark6) 3
   6. [Model Deployment](#_bookmark7) 3
3. [Diagram](#_bookmark8) 3
4. [Notes](#_bookmark9) 3

# Overview

The pipeline architecture outlines the data flow from raw input to volatility prediction, ensuring modularity, scalability, and reproducibility. The pipeline consists of sequential stages that process data, train a model, and deploy predictions.

# Pipeline Stages

## Data Ingestion

* + - **Input**: Raw CSV file (dataset.csv).
    - **Process**: Load data using Pandas, validate columns and data types.
    - **Output**: Raw DataFrame.

## Data Preprocessing

* + - **Input**: Raw DataFrame.
    - **Process**:
      * Drop Unnamed: 0 column and rows with missing critical values.
      * Convert date to datetime.
      * Sort by crypto\_name and date.
      * Engineer features: daily return, 7-day rolling volatility, liquidity ratio, 7-day moving average.
      * Normalize numerical features using StandardScaler.
    - **Output**: Processed DataFrame, fitted scaler.

## Exploratory Data Analysis (EDA)

* + - **Input**: Processed DataFrame.
    - **Process**:
      * Compute summary statistics.
      * Generate visualizations: Bitcoin close price, 7-day volatility, liquidity ratio.
    - **Output**: Statistics file, visualization PNGs (bitcoin\_close\_price.png, bitcoin\_volatility.png, bitcoin\_liquidity.png).

## Model Training

* + - **Input**: Processed DataFrame.
    - **Process**:
      * Select features and target (shifted rolling\_volatility\_7d).
      * Use TimeSeriesSplit for 5-fold cross-validation.
      * Train RandomForestRegressor.
    - **Output**: Trained model, evaluation metrics.

## Model Evaluation

* + - **Input**: Trained model, test data splits.
    - **Process**: Compute RMSE, MAE, and R² for each fold, average metrics.
    - **Output**: Evaluation metrics.

## Model Deployment

* + - **Input**: Trained model, fitted scaler.
    - **Process**:
      * Save model and scaler using joblib.
      * Create Streamlit app with input fields for features.
      * Normalize inputs, predict volatility, display results.
    - **Output**: Web interface for predictions.

# Diagram

[*RawCSV* ] *→* [*DataIngestion*] *→* [*DataPreprocessing*] *→* [*EDA*] *→* [*ModelTraining*] *→* [*ModelEvaluation*

# Notes

* The pipeline ensures temporal order using TimeSeriesSplit.
* Intermediate outputs (processed data, plots, metrics) are saved for reproducibility.
* The Streamlit app enables user interaction for volatility predictions.