Low-Level Design Document: Cryptocurrency Volatility Prediction System

Contents

1 Introduction

This Low-Level Design (LLD) document outlines the theoretical design of the cryptocurrency volatility prediction system, encompassing components implemented in Jupyter notebooks for data processing, feature engineering, model training, and a Streamlit application for user interaction. The system predicts 7-day volatility for various cryptocurrencies using historical market data and provides an interactive interface for predictions and visualizations. Each component is described in terms of its purpose, dependencies, inputs/outputs, and error handling, focusing on theoretical aspects to ensure clarity and maintainability.

2 Data Ingestion

2.1 Purpose

The data ingestion module is responsible for loading the raw cryptocurrency dataset from a CSV file into a structured format suitable for subsequent processing and analysis in both the analytical pipeline and the user-facing application.

2.2 Dependencies

- Python environment for data manipulation.
- Library for handling tabular data structures.
- Library for managing file system paths in the application.

2.3 Input/Output

- Input: A CSV file containing historical cryptocurrency data with columns such as open, high, low, close, volume, market capitalization, timestamp, cryptocurrency name, and date.
- Output: A structured data table with 72,946 records, maintaining the original or processed column set (10 columns for raw data, up to 74 for processed data).

2.4 Error Handling

- Validates the existence of the input file to prevent loading errors.
- Ensures the file format is compatible with expected column structure.
- Manages exceptions related to file access or parsing issues.

3 Data Preprocessing

3.1 Purpose

The preprocessing module cleans the dataset by removing redundant or unnecessary columns, standardizing date and timestamp formats, and verifying data integrity to ensure reliability for downstream analysis.

3.2 Dependencies

- Python environment for data manipulation.
- Library for handling tabular data and datetime conversions.

3.3 Input/Output

- Input: Raw data table with 10 columns, including an index column and datetime fields.
- Output: Cleaned data table with 9 columns, where redundant columns are removed and date/timestamp fields are in a standardized datetime format.

3.4 Error Handling

- Validates date and timestamp formats to prevent conversion errors.
- Checks for missing or duplicate records to ensure data quality.
- Handles inconsistencies in data types or unexpected values.

4 Exploratory Data Analysis (EDA)

4.1 Purpose

The EDA module analyzes the dataset to uncover patterns, trends, and statistical properties in cryptocurrency price, volume, and market capitalization data, facilitating informed feature engineering and model development.

4.2 Dependencies

- Python environment for data analysis.
- Libraries for data filtering and visualization, including histogram generation with kerneldensity estimation.

4.3 Input/Output

- Input: Cleaned data table with 9 columns, including price and cryptocurrency identifier fields.
- Output: Visualizations (e.g., histograms with density curves) for open, close, high, and low prices per cryptocurrency, highlighting distribution characteristics like volatility and skewness.

4.4 Error Handling

- Ensures the presence of cryptocurrency identifier fields.
- Validates data subsets to prevent visualization errors due to empty or invalid data.
- Manages issues related to visualization library compatibility.

5 Feature Engineering

5.1 Purpose

The feature engineering module enhances the dataset by creating new features, such as moving averages, volatility metrics, liquidity ratios, and time-based attributes, to improve the predictive power of the volatility forecasting model.

5.2 Dependencies

- Python environment for data manipulation.
- Libraries for tabular data operations, categorical encoding, and numerical computations.

5.3 Input/Output

- Input: Cleaned data table with 9 columns.

- Output: Enhanced data table with 74 columns, including simple moving averages (7, 14, 30 days), one-hot encoded cryptocurrency identifiers, time-based features (year, month, day, day of week), and technical indicators (log returns, 14- and 30-day volatility, liquidity ratio, Bollinger band width, true range, average true range).

5.4 Error Handling

- Manages missing values in rolling window calculations, allowing initial nulls where applicable.
- Prevents division-by-zero errors in ratio calculations by validating denominators.
- Ensures categorical encoding does not produce duplicate or conflicting columns.

6 Model Training and Evaluation

6.1 Purpose

The model training module develops and evaluates multiple regression models to predict 7-day volatility, selecting the best-performing model (LightGBM) and optimizing its hyperparameters using time-series cross-validation.

6.2 Dependencies

- Python environment for machine learning.
- Libraries for data splitting, model training, evaluation metrics, and advanced regressionalgorithms (including gradient boosting).

6.3 Input/Output

- Input: Enhanced data table with 74 columns, including features and the target variable (7-day volatility).
- Output: A trained LightGBM model with optimized hyperparameters, achieving an RMSE of approximately 27.03 and Rš of 0.51, along with performance metrics for multiple regression models.

6.4 Error Handling

- Ensures sufficient data for time-series splits to maintain chronological integrity.
- Handles deprecated parameters in evaluation metrics to ensure compatibility.
- Manages exceptions during model training and hyperparameter optimization.

7 Model Deployment

7.1 Purpose

The model deployment module serializes the trained LightGBM model to a file, enabling its use in the predictive application for real-time volatility forecasting.

7.2 Dependencies

- Python environment for serialization.
- Library for object serialization to files.

7.3 Input/Output

- Input: Trained LightGBM model.
- Output: A serialized file containing the model for later retrieval.

7.4 Error Handling

- Verifies write permissions for the output file.
- Handles file operation errors during serialization.

8 Streamlit Application

8.1 Purpose

The Streamlit application provides an interactive web interface for users to input cryptocurrency data, predict 7-day volatility using the trained model, and visualize results through candlestick charts and simulated price paths.

8.2 Dependencies

- Python environment for web application development.
- Libraries for web interface creation, data handling, numerical computations, model deserialization, feature scaling, interactive visualizations, and date manipulation.

8.3 Input/Output

- Input:
  - Serialized LightGBM model.
  - Processed dataset for feature reference.

    - User inputs: cryptocurrency selection, date, and numerical features (open, high, low, close, volume, market capitalization, average true range, Bollinger band width).

- Output:
    - Predicted 7-day volatility value.
    - Candlestick chart displaying open, high, low, and close prices for the input date with a volatility annotation.
    - Simulated 7-day price paths with a mean path and 95% confidence interval.

8.4 Error Handling

- Validates the presence of model and data files to prevent loading errors.
- Ensures numerical inputs fall within predefined ranges to maintain data validity.
- Provides fallback values for derived features if calculations fail due to insufficient data.
- Verifies feature alignment with model expectations to avoid prediction errors.
- Manages exceptions during feature scaling and prediction processes.

9 Conclusion

This LLD provides a theoretical overview of the cryptocurrency volatility prediction system, detailing the data ingestion, preprocessing, exploratory data analysis, feature engineering, model training, deployment, and interactive application components. The system is designed to process historical market data, generate predictive features, train a robust model, and deliver userfriendly predictions and visualizations, ensuring reliability and extensibility.