High-Level Design (HLD)

Project Title

Cryptocurrency Volatility Prediction System

Objective

To design a machine learning–based system that predicts short-term cryptocurrency volatility using historical market data such as OHLC prices, trading volume, and market capitalization.

System Overview

The system processes historical cryptocurrency data, performs feature engineering to derive volatility-related indicators, trains a machine learning regression model, and provides volatility predictions through a simple local web interface.

System Architecture

Components

1. **Data Source**
   Historical cryptocurrency price dataset (CSV format)

2. **Data Preprocessing Layer**

3. Missing value handling
4. Data cleaning and sorting

5. Data normalization

6. **Feature Engineering Layer**

7. Daily returns
8. Rolling volatility
9. Moving averages

10. Liquidity indicators

11. **Model Training Layer**

12. Machine learning regression model

13. Model persistence

14. **Evaluation Layer**

15. Performance metrics calculation

16. **Deployment Layer**

17. Streamlit-based user interface

Technology Stack

- Programming Language: Python
- Libraries: Pandas, NumPy, Scikit-learn
- Visualization: Matplotlib, Seaborn
- Deployment: Streamlit

Data Flow

Raw Data → Preprocessing → Feature Engineering → Model Training → Evaluation → Prediction Output

---

Low-Level Design (LLD)

Module Breakdown

1. Data Preprocessing Module

File: `data_preprocessing.py`

Responsibilities: - Load CSV data - Convert date column to datetime - Sort data by symbol and date - Handle missing values - Remove duplicates

Input: Raw CSV file
Output: Cleaned dataset

---

2. Feature Engineering Module

File: `feature_engineering.py`

Responsibilities: - Calculate daily returns - Generate rolling volatility (7-day) - Compute moving averages - Create liquidity ratio - Drop rows with insufficient rolling data

Input: Cleaned dataset
Output: Processed dataset with engineered features

3. Model Training Module

File: `train_model.py`

Responsibilities: - Select input features and target variable - Scale numerical features - Train Random Forest Regressor - Save trained model and scaler

Input: Processed dataset
Output: Trained model (`.pkl` file)

---

4. Model Evaluation Module

File: `evaluate_model.py`

Responsibilities: - Load trained model - Predict on test dataset - Compute RMSE, MAE, and R^2 score

Input: Test dataset, trained model
Output: Evaluation metrics

---

5. Deployment Module

File: `app.py`

Responsibilities: - Load trained model and scaler - Accept user-uploaded dataset - Predict volatility - Display results and graphs

Input: Processed dataset (CSV)
Output: Volatility predictions via UI

Error Handling

- Missing values handled via forward fill
- Rows with invalid rolling calculations removed
- Model input validation before prediction

Assumptions

- Data is daily and continuous per cryptocurrency
- Volatility is defined as rolling standard deviation of returns
- Local deployment is sufficient for evaluation

Security & Performance Considerations

- Read-only local file access
- No external API dependency
- Efficient batch prediction

Pipeline Architecture

End-to-End Machine Learning Pipeline

The cryptocurrency volatility prediction system follows a structured and sequential machine learning pipeline to ensure data consistency, model reliability, and reproducibility.

Pipeline Stages

1. Data Ingestion

- Historical cryptocurrency data is ingested from CSV files.
- Data includes OHLC prices, volume, and market capitalization for multiple cryptocurrencies.

2. Data Preprocessing

- Date column conversion to datetime format.
- Sorting data by cryptocurrency symbol and date.
- Handling missing values using forward fill.
- Removing duplicate records.

Output: Cleaned and consistent dataset

---

3. Feature Engineering

- Calculation of daily returns and log returns.
- Generation of rolling volatility (7-day standard deviation of returns).
- Computation of moving averages (7-day and 14-day).
- Creation of liquidity indicators such as volume-to-market-cap ratio.
- Removal of rows with insufficient rolling window data.

Output: Feature-rich processed dataset

---

4. Feature Scaling

- Numerical features are standardized using `StandardScaler`.
- Ensures uniform feature contribution to the model.

---

5. Model Training

- Random Forest Regressor is trained on processed features.
- Time-based train-test split is applied to prevent data leakage.
- Trained model and scaler are saved for reuse.

Output: Trained machine learning model

---

6. Model Evaluation

- Model predictions generated on unseen test data.
- Evaluation metrics calculated:
- RMSE
- MAE
- R^2 Score

---

7. Deployment Pipeline

- Trained model is loaded into a Streamlit application.
- User uploads processed data.
- Model predicts volatility values.
- Results displayed in tabular and graphical format.

Pipeline Flow Diagram (Textual Representation)

Raw Data → Preprocessing → Feature Engineering → Scaling → Model Training → Evaluation → Deployment

---

Final Report

1. Introduction

Cryptocurrency markets are known for their extreme price fluctuations, making volatility prediction a critical task for traders, investors, and financial institutions. This project focuses on building a machine learning–based system to predict short-term cryptocurrency volatility using historical market data.

---

2. Problem Statement

The objective of this project is to predict cryptocurrency volatility levels using historical OHLC prices, trading volume, and market capitalization. Accurate volatility prediction helps in risk management, portfolio optimization, and informed trading decisions.

3. Dataset Description

The dataset consists of daily historical records for multiple cryptocurrencies and includes the following attributes: - Date - Symbol - Open, High, Low, Close prices - Trading Volume - Market Capitalization

The data spans multiple years and covers more than 50 cryptocurrencies.

---

4. Data Preprocessing

The following preprocessing steps were applied: - Conversion of date column to datetime format - Sorting data by cryptocurrency symbol and date - Handling missing values using forward fill - Removing duplicate records

These steps ensured data consistency and reliability.

---

5. Feature Engineering

Several financial and statistical features were engineered to capture market behavior: - Daily returns - Log returns - 7-day rolling volatility (target variable) - 7-day and 14-day moving averages - Liquidity ratio (volume / market capitalization) - Intraday volatility (high-low spread)

Rolling volatility was chosen as the target variable to represent short-term market risk.

---

6. Exploratory Data Analysis (EDA)

EDA revealed key insights: - High volatility periods often coincide with high trading volume - Cryptocurrencies with smaller market capitalization exhibit higher volatility - Liquidity ratio shows a positive correlation with volatility

Visualizations such as price trends, volatility trends, distributions, and correlation heatmaps were used to support these observations.

---

7. Model Selection and Training

A Random Forest Regressor was selected due to its ability to capture non-linear relationships and robustness to noise. The dataset was split using a time-based approach to avoid data leakage. Feature scaling was applied using StandardScaler.

---

8. Model Evaluation

The model performance was evaluated using the following metrics: - Root Mean Squared Error (RMSE) - Mean Absolute Error (MAE) - R^2 Score

The trained model demonstrated good predictive performance and generalization capability on unseen data.

---

9. Deployment

The trained model was deployed locally using a Streamlit application. Users can upload processed cryptocurrency data and obtain predicted volatility values along with graphical visualizations.

---

10. Results and Key Insights

- The model effectively predicts short-term volatility trends
- Liquidity and volume play a significant role in volatility prediction
- Ensemble-based models perform well for financial time-series regression

---

11. Limitations

- Model relies on historical data only
- External factors such as news and sentiment are not considered
- Predictions are short-term and may not capture long-term market cycles

---

12. Future Scope

- Integration of real-time data
- Inclusion of sentiment analysis from news and social media
- Use of deep learning models such as LSTM or GRU
- Multi-horizon volatility forecasting

Conclusion

This project successfully demonstrates an end-to-end machine learning pipeline for cryptocurrency volatility prediction. The system meets all project requirements and provides valuable insights into market risk, making it suitable for academic and practical applications.