

Cryptocurrency Volatility Prediction

Problem Statement



Problem Statement

Cryptocurrency markets are highly volatile, and understanding and forecasting this volatility is crucial for market participants. Volatility refers to the degree of variation in the price of a cryptocurrency over time, and high volatility can lead to significant risks for traders and investors. Accurate volatility prediction helps in risk management, portfolio allocation, and developing trading strategies.

In this project, you are required to build a machine learning model to predict cryptocurrency volatility levels based on historical market data such as OHLC (Open, High, Low, Close) prices, trading volume, and market capitalization. The objective is to anticipate periods of heightened volatility, enabling traders and financial institutions to manage risks and make informed decisions.

Your final model should provide insights into market stability by forecasting volatility variations, allowing stakeholders to proactively respond to changing market conditions.

Dataset Information

You will use a dataset that includes historical daily cryptocurrency price, volume, and market capitalization data for multiple cryptocurrencies.

Dataset:

[Cryptocurrency Historical Prices Dataset](#)

The dataset consists of daily records for over 50 cryptocurrencies, including features such as date, symbol, open, high, low, close, volume, and market cap.

Data Preprocessing Required

- Handle missing values and ensure data consistency
- Normalize and scale numerical features
- Engineer new features related to volatility and liquidity trends

Project Development Steps

- Data Collection: Gather historical OHLC, volume, and market cap data from the provided dataset
- Data Preprocessing: Handle missing values, clean data, and normalize numerical features
- Exploratory Data Analysis (EDA): Analyze data patterns, trends, and correlations
- Feature Engineering: Create relevant features such as moving averages, rolling volatility, liquidity ratios (e.g., volume/market cap), and technical indicators (e.g., Bollinger Bands, ATR)
- Model Selection: Choose appropriate machine learning models such as time-series forecasting, regression, or deep learning approaches
- Model Training: Train the selected model using the processed dataset
- Model Evaluation: Assess model performance using metrics such as RMSE, MAE, and R² score

Model Optimization and Deployment

- Hyperparameter Tuning: Optimize model parameters for better accuracy
- Model Testing & Validation: Test the model on unseen data and analyze predictions
- Local Deployment: Deploy the trained model locally using Flask or Streamlit for testing

Expected Deliverables

1. Machine Learning Model

- A trained model that predicts cryptocurrency volatility
- Evaluation metrics showing how well the model performs

2. Data Processing & Feature Engineering

- Cleaned and prepared dataset
- A brief explanation of new features added

3. Exploratory Data Analysis (EDA) Report

- Summary of dataset statistics
- Basic visualizations (trends, correlations, distributions)

4. Project Documentation

- High-Level Design (HLD) Document: Overview of system and architecture
- Low-Level Design (LLD) Document: Breakdown of how each component is implemented
- Pipeline Architecture: Explanation of data flow from preprocessing to prediction
- Final Report: A simple summary of findings, model performance, and key insights

Guidelines & Submission Requirements

- Code Documentation: Ensure all scripts are well-commented and easy to follow
- Report Structure: The report must be structured and should clearly explain the methodology followed
- Diagrams & Visuals: Use appropriate diagrams and plots to explain data processing, model selection, and performance evaluation
- Deployment: If possible, deploy the model using a simple interface (e.g., Streamlit or Flask API) for testing predictions

Submission Format

The project must be submitted as a GitHub repository or a zipped folder containing:

- Source Code
- EDA Report
- HLD & LLD Documents
- Pipeline Architecture and Documentation
- Final Report