Cryptocurrency Liquidity Prediction Project - Final Report

1. Introduction

This project aims to predict the liquidity of cryptocurrencies using historical data including prices, volumes, and market capitalization.

Liquidity prediction helps financial institutions, traders, and analysts in making better trading decisions and managing risks. The

solution leverages machine learning models trained on processed and engineered features from cryptocurrency market data.

2. Problem Statement

Cryptocurrency markets are volatile and less liquid compared to traditional financial markets. Predicting liquidity is crucial for

portfolio management, minimizing slippage, and assessing market stability. The goal is to create a model that predicts a liquidity

metric using historical market features.

3. Dataset Description

The dataset includes historical data of multiple cryptocurrencies with the following features:

- Date and Time
- Open, High, Low, Close Prices
- Trading Volume
- Market Capitalization

The target variable is a derived liquidity ratio based on volume and market cap.

4. Data Preprocessing

- Missing value imputation using forward-fill or interpolation
- Date conversion and sorting
- Normalization and scaling
- Removal of redundant or constant features

5. Feature Engineering

- Rolling window statistics (mean, std, etc.)
- Volatility indicators
- Ratio features (e.g., volume/market cap)
- Lag features to capture time-based patterns

6. Model Development

Two regression models were developed:

1. XGBoost Regressor
2. Bagging Regressor

Each model underwent hyperparameter tuning using GridSearchCV. The best model is saved using pickle for inference.

Cryptocurrency Liquidity Prediction Project - Final Report

7. Model Evaluation

Performance metrics used:

- RMSE (Root Mean Square Error)

- MAE (Mean Absolute Error)

- R^2 Score (Coefficient of Determination)

Models were evaluated on a test set split from the original dataset to ensure unbiased performance measurement.

8. Application Architecture

The application is divided into:

- Data Layer: Reads CSV and processes data

- ML Layer: Handles feature engineering, training, evaluation

- API Layer: Flask app with endpoints for training (/train) and prediction (/predict)

- UI Layer: Web interface for interaction

9. Deployment

The model and Flask application can be deployed locally or within a Docker container. Dependencies are managed using requirements.txt.

Steps to run:

1. Clone repository

2. Install dependencies

3. Run app with `python app.py`

10. Challenges and Future Work

Challenges:

- High volatility and noise in cryptocurrency data

- Limited historical data for lesser-known coins

Future Work:

- Use deep learning models like LSTMs for time series

- Extend to multi-asset forecasting

- Integrate real-time data ingestion APIs