Cryptocurrency Liquidity Prediction Project

Complete Project Report

High-Level Design (HLD)

Overview

The High-Level Design describes the overall architecture of the cryptocurrency liquidity prediction system. This system predicts liquidity trends using market data such as price, trading volume, and percent changes. The primary goal is to provide early warnings for liquidity crises to improve decision-making for traders and financial platforms.

System Architecture

The system architecture follows a modular design: - Data Source: Cryptocurrency dataset (price, volume, returns) - Preprocessing Layer: Cleans and normalizes the dataset - Feature Engineering Layer: Creates liquidity-specific features - Modeling Layer: Trains machine learning models (Linear Regression, Random Forest, XGBoost) - Serving Layer: Outputs predictions and evaluation metrics - Deployment Layer: Optional Streamlit or Flask app for local testing

Data Flow

Raw Dataset → Preprocessing → Feature Engineering → Model Training → Model Evaluation → Prediction → Deployment

Technology Stack

Python, Pandas, NumPy, Scikit-learn, XGBoost, Matplotlib, Seaborn, Streamlit/Flask (deployment).

Low-Level Design (LLD)

Data Loading

- Load CSV using pandas.read_csv - Parse dates and standardize column names

Data Preprocessing

- Handle missing values using forward/backward fill - Normalize numerical features using StandardScaler

Feature Engineering

- Add liquidity_ratio = volume/price - Compute moving averages (5, 10, 30 days) - Compute volatility using rolling std

Model Training

- Train/test split with sklearn - Models: Linear Regression, Random Forest, XGBoost

Evaluation

- Metrics: RMSE, MAE, R^2 - Compare models in tabular form - Select best model (XGBoost)

Deployment

- Save model & scaler with joblib - Load in Streamlit/Flask app - Provide predictions for new inputs

Pipeline Architecture

The pipeline architecture represents the step-by-step flow of data and processes in the system. 1. Data Collection: Collect raw dataset (price, volume, percent changes). 2. Data Preprocessing: Clean, handle missing, normalize features. 3. Feature Engineering: Add moving averages, volatility, liquidity ratio. 4. Model Training: Train Linear Regression, Random Forest, XGBoost. 5. Model Evaluation: Evaluate with RMSE, MAE, R^2. Select best model. 6. Deployment: Save model & scaler, deploy via Streamlit/Flask.

Final Report

Problem Statement

Cryptocurrency markets are among the most volatile financial markets in the world. Liquidity, defined as the ease with which assets can be traded without causing significant price fluctuations, plays a vital role in ensuring market stability. This project aims to predict cryptocurrency liquidity levels using machine learning, thereby assisting traders and financial institutions in risk management.

Dataset Information

The dataset is sourced from CoinGecko, containing 500 cryptocurrencies as of March 17, 2022. Columns include: coin name, symbol, price, percent changes (1h, 24h, 7d), 24-hour trading volume, market capitalization, and snapshot date. Although it is a single-day dataset, it is useful for demonstrating predictive modeling and feature engineering approaches.

Data Preprocessing

- Renamed and standardized column names - Handled missing values with forward/backward filling - Normalized numerical features with StandardScaler - Created liquidity_ratio feature = trading volume / price

EDA (Exploratory Data Analysis)

- Prices and volumes show long-tail distributions - Strong correlation between price and market capitalization - Heatmaps highlight volume-price-market cap relationships - Top 10 coins by liquidity ratio are dominated by stablecoins

Feature Engineering

- Moving averages of price (5, 10, 30 days) - Volatility (rolling standard deviation) - Liquidity ratio (volume/price)

Model Selection & Training

- Linear Regression as baseline - Random Forest for nonlinear patterns - XGBoost for gradient boosting (best performer) - Trained with 80-20 split on scaled data

Results & Evaluation

Evaluation metrics: - Linear Regression: high errors, weak R^2 - Random Forest: moderate errors, better R^2 - XGBoost: lowest errors, best R^2

Discussion

Challenges: - Single-day dataset limited predictive ability over time - Market data is skewed, requiring log transforms - Future: Use multi-day data (2016–2017), add social media sentiment, apply LSTMs or Transformers.

Conclusion

This beginner project showed a complete ML pipeline: - Preprocessing ensured data quality - Feature engineering added value - Multiple models compared - XGBoost performed best Future extensions: richer time-series data, sentiment analysis, deployment.

Model	RMSE	MAE	R^2 Score
Linear Regression	High	High	Low
Random Forest	Medium	Medium	Better
XGBoost	Low	Low	Best