**Cryptocurrency Liquidity Prediction for Market Stability**

– Low Level Design (LLD)

Domain: Machine Learning

Project Name: Cryptocurrency Liquidity Prediction System

Author: Naveen Jangid

Date: 14/05/2025

Version: 1.0

Email: nvnjan95@gmail.com

# Document Version Control

Date | Version | Description | Author

----------------------------|----------------------------------|-----------------------------|------------------------

14/05/2025 | 1.0 | First Draft | Naveen Jangid

# Contents

- Introduction

- Objective

- Scope

- Architecture

- Architecture Description

- Data Pipeline

- Data Description

- Data Preprocessing

- Feature Engineering

- Model Development

- Model Implementation

- Hyperparameter Tuning

- Model Evaluation

- Web Deployment

- API Design

- Unit Test Cases

# Introduction

What is a Low-Level Design Document?

The Low-Level Design (LLD) document outlines the detailed internal logic of the Cryptocurrency Liquidity Prediction project. It covers the structure, data transformation, model pipeline, API flow, and user interaction of the system.

# Objective

To develop and deploy a machine learning model that predicts the liquidity ratio of a cryptocurrency and classifies it into Low, Medium, or High categories, with the help of a user-friendly web interface.

# Scope

- Data Cleaning and Transformation

- Feature Engineering (Liquidity Ratio & Volatility Score)

- Model Training and Stacking Ensemble

- Flask API and Web Interface

- Cloud Deployment on Render

# Architecture Overview

User Input → Flask App → Model (joblib) → log1p + prediction → expm1 → Liquidity Classification → Output on Web UI

# Data Pipeline

Data Description:

- Source: CoinGecko (snapshots from 2022-03-16 & 2022-03-17)

- Features: Price, 1h / 24h / 7d % Change, 24h Volume, Market Cap

Data Preprocessing:

- Dropped rows with missing values (less than 1%)

- Converted data types to float

- Renamed columns for consistency

Feature Engineering:

- Liquidity Ratio: volume / market\_cap

- Volatility Score: (|1h| + |24h| + |7d|) / 3

- Target Transformation: log1p(liquidity\_ratio)

# Model Development

Model Implementation:

- Used StackingRegressor with passthrough

- Base Models: RandomForestRegressor, XGBRegressor, SVR

- Meta Model: LinearRegression

- Saved model as: final\_liquidity\_stack\_model.pkl

Hyperparameter Tuning:

- GridSearchCV used for individual models

- Evaluation metrics: R², MAE, RMSE

Model Evaluation:

Metric | Value

-------|------

MAE | 0.00836

RMSE | 0.01932

R² | 0.95221

# Web Deployment

- Framework: Flask

- Frontend: HTML + Jinja2 template

- Deployment Platform: Render

- Start Command: gunicorn app:app

- Hosted link: [insert Render link here]

# API Design

Endpoints:

1. Home Page

- Method: GET

- URL: /

- Response: HTML form

2. Prediction

- Method: POST

- URL: /predict

- Input Format: Form Data

{ 'price': 2784.67, '1h': 0.14, '24h': 1.21, '7d': -2.57, '24h\_volume': 58000000, 'mkt\_cap': 320000000, 'volatility\_score': 1.31 }

- Response Format:

{ 'Predicted Liquidity Ratio': '0.129', 'Liquidity Classification': 'Medium' }

# Unit Test Cases

Test Case | Description | Expected Result

---------------------|----------------------------------|-------------------------------

Valid Input | All fields correct | Returns predicted liquidity

Missing Input Fields | One or more fields missing | Returns validation error

Non-numeric Input | Input contains invalid symbols | Returns input validation error

Model Not Loaded | Joblib file missing/corrupt | Returns server error

Render App Down | Server inaccessible | Returns 500 internal server error

# Conclusion

This LLD serves as a detailed implementation plan for the cryptocurrency liquidity prediction system. It includes data processing, model stacking, classification, Flask-based serving logic, and Render deployment, providing an end-to-end solution from input to deployment.