**Cryptocurrency Liquidity Prediction for Market Stability**

– Architecture Document

Domain: Machine Learning, Financial Forecasting

Author: Naveen Jangid

Date: 14/05/2025

Version: 1.0

Email: nvnjan95@gmail.com

# Cryptocurrency Liquidity Prediction – Architecture Document

\*\*Domain\*\*: Machine Learning, Financial Forecasting

\*\*Author\*\*: Naveen Jangid

\*\*Date\*\*: 14/05/2025

\*\*Version\*\*: 1.0

\*\*Email\*\*: nvnjan95@gmail.com

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to detail the complete system architecture for the project titled \*\*"Cryptocurrency Liquidity Prediction"\*\*, which leverages machine learning models to forecast the liquidity ratio of cryptocurrencies using real-time market data. The system aims to aid traders, investors, and financial platforms by providing accurate liquidity predictions through a web-based interface.

### 1.2 Scope

The project utilizes public market data from CoinGecko to compute derived features and train an ensemble machine learning model. The system allows users to input features such as price, 1h/24h/7d price changes, volume, market capitalization, and volatility, and returns the predicted liquidity ratio along with a classification label (Low, Medium, High). The entire pipeline is deployed on a cloud platform (Render) using Flask as the web framework.

### 1.3 Objectives

- To predict cryptocurrency liquidity based on current market metrics.

- To classify liquidity into actionable levels: Low, Medium, High.

- To design and deploy a full-stack ML web application.

- To ensure scalability and usability via a cloud-based hosting environment.

## 2. High-Level Architecture

### 2.1 Overview

The system is divided into four primary layers:

1. \*\*Data Preparation Layer\*\*: Cleans the raw dataset, handles missing values, and generates engineered features.

2. \*\*Modeling Layer\*\*: Trains a stacking ensemble using multiple base regressors and a meta-model.

3. \*\*Application Layer\*\*: Flask web server that receives input, performs predictions, and renders results.

4. \*\*Deployment Layer\*\*: Cloud infrastructure (Render) for hosting the web service with automatic build and deployment.

### 2.2 Data Flow Diagram

```

User Input (Form)

↓

Flask App (app.py)

↓

Model Load (joblib)

↓

Prediction (log liquidity → expm1)

↓

Liquidity Classification (Low/Medium/High)

↓

Result Rendered on Web Page

```

...