Pipeline Architecture Document

Project Title: Cryptocurrency Liquidity Prediction for Market Stability

1. Introduction This document outlines the architecture of the end-to-end machine learning pipeline for the Cryptocurrency Liquidity Prediction project. The pipeline is designed to be a sequential and automated workflow, transforming raw data into actionable predictions.

2. Pipeline Stages The pipeline consists of four distinct stages, executed in sequence as defined in main.py.

Stage 1: Data Ingestion and Cleaning

- **Description:** The pipeline begins by loading historical cryptocurrency data from specified CSV files (coin_gecko_2022-03-16.csv, coin_gecko_2022-03-17.csv). The load_and_clean_data() function is called to merge these files and handle any missing or inconsistent data points.
- **Component:** src/preprocessing.py

Stage 2: Data Transformation and Scaling

- **Description:** Key numerical features (price, 24h_volume, mkt_cap, etc.) are normalized using the scale_data() function. This step is crucial for ensuring that features with different scales do not disproportionately influence the model's learning process.
- **Component:** src/preprocessing.py

Stage 3: Feature Engineering

- **Description:** The add_features() function is executed to create higher-level features that are more predictive of liquidity. This includes calculating a 3-day moving average of the price (price_ma) and the target variable, liquidity_ratio.
- **Component:** src/feature_engineering.py

Stage 4: Model Training and Serialization

- **Description:** The final stage involves training a Random Forest Regressor model using the fully processed data. The train_model() function is called with liquidity_ratio as the target. Upon completion, the trained model is serialized (saved to disk) for later use by the prediction service.
- **Component:** src/model.py

3. Data Flow Diagram

[CSV Data Files] -> [1. Ingest & Clean] -> [2. Scale Features] -> [3. Engineer Features] -> [4. Train Model] -> [Saved Model] ->[Streamlit App] -> [Prediction]

4. Execution The entire pipeline is orchestrated and executed by the main.py script, which calls the functions from each module in the correct order to produce the final, trained model.