Pipeline Architecture Report: Cryptocurrency Liquidity Prediction

1. Overview

This report outlines the end-to-end pipeline architecture of the Cryptocurrency Liquidity Prediction project.

The pipeline encompasses all key stages from data ingestion to model deployment.

2. Pipeline Stages

1. Data Ingestion:

   - Raw historical data is loaded from CSV files or external APIs.

   - Initial sanity checks and schema validation are performed.

2. Data Preprocessing:

   - Handles missing values using forward-fill/interpolation.

   - Converts data types, sorts by date, and normalizes/standardizes numeric columns.

3. Feature Engineering:

   - Creates rolling averages, liquidity ratio, volatility indicators, and lag features.

   - Adds time-based features like day-of-week, month, etc.

4. Model Training:

   - Splits dataset into train/test.

   - Trains multiple models (e.g., Linear Regression, XGBoost).

   - Evaluates using RMSE, MAE, and R².

   - Selects and serializes the best-performing model.

5. Model Evaluation:

   - Compares model performance metrics.

   - Performs residual analysis to assess prediction error distribution.

6. Model Deployment:

   - Deploys model using a Flask web server.

   - API endpoints: `/train` for training and `/predict` for prediction.

   - Accepts input data and returns predicted liquidity.

3. Tools and Technologies

- Python 3.x

- Pandas, NumPy for data manipulation

- scikit-learn, XGBoost for ML modeling

- Flask for deployment

- Matplotlib, Seaborn for visualization

- Docker (optional) for containerization

4. Pipeline Automation (Optional Enhancements)

- Use Airflow or Prefect to schedule and monitor pipeline steps.

- Add real-time data ingestion from crypto exchanges.

- Include automated retraining and model versioning.

5. Conclusion

The pipeline is modular, scalable, and easy to maintain. It supports rapid experimentation and production-ready deployment.

Each component can be extended or replaced as project needs evolve.