LOW-LEVEL DESIGN (LLD)

Cryptocurrency Liquidity Prediction for Market Stability

Detailed Technical Implementation

Version 1.0

1. Introduction

This Low-Level Design document provides detailed technical specifications for implementing the Cryptocurrency Liquidity Prediction system. It covers module-level design, detailed algorithms, data structures, and implementation details for each component of the system.

2. Feature Engineering Pipeline

The feature engineering process transforms raw cryptocurrency data into meaningful features that the machine learning model can use. This includes logarithmic transformations to handle large value ranges, ratio calculations for liquidity metrics, and temporal features like price changes.

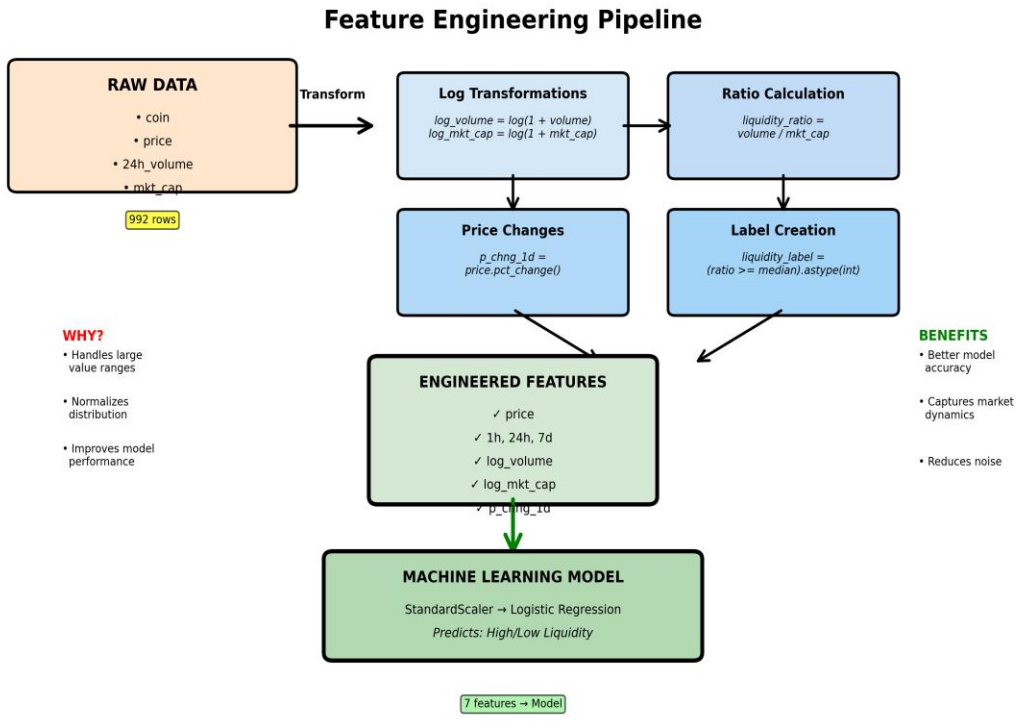

Figure 1: Feature Engineering Pipeline

2.1 Implementation Details

Logarithmic Transformations:

```
df['log_volume'] = np.log1p(df['24h_volume']) df['log_mkt_cap'] =
np.log1p(df['mkt_cap'])
```

Liquidity Ratio:

```
df['liquidity_ratio'] = df['24h_volume'] / df['mkt_cap']
```

Target Label:

```
threshold = df['liquidity_ratio'].median() df['liquidity_label'] =
(df['liquidity_ratio'] >= threshold).astype(int)
```

3. Complete Data Flow

The following diagram illustrates the complete data flow from user input to prediction output, showing all intermediate processing steps:

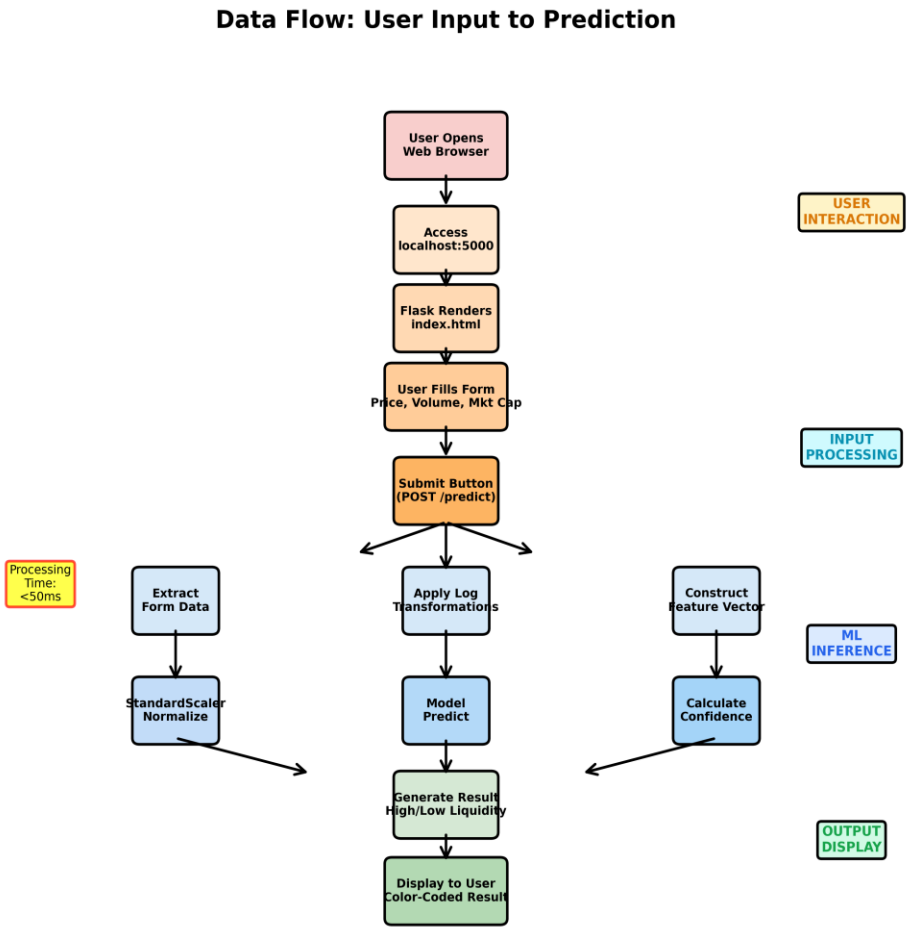

Figure 2: End-to-End Data Flow Diagram

4. Flask Application Implementation

4.1 Model Loading

```
try:     model = joblib.load('models/crypto_liquidity_model.pkl')     scaler
= joblib.load('models/scaler.pkl') except FileNotFoundError:
print('Error: Model files not found!')
```

4.2 Prediction Logic

```
# Extract inputs price = float(request.form['price']) vol_24h =
float(request.form['vol_24h']) mkt_cap = float(request.form['mkt_cap'])  #
Transform features log_volume = np.log1p(vol_24h) log_mkt_cap =
np.log1p(mkt_cap)  # Create input array input_data = [[price, 0.0, 0.0, 0.0,
0.0, log_volume, log_mkt_cap]]  # Scale and predict input_scaled =
scaler.transform(input_data) prediction = model.predict(input_scaled)[0]
probability = model.predict_proba(input_scaled).max() * 100
```

5. Performance Metrics

The system achieves the following performance metrics:

- Model Accuracy: 95.15%
- Inference Time: <50ms per prediction
- Model Size: ~15KB (Logistic Regression + Scaler)
- Confidence Score: 85-99% (typical range)

6. Project File Structure

```
project/ ├── app.py                    # Flask application ├── EDA.ipynb
# Jupyter notebook for training ├── models/ │   ├──
crypto_liquidity_model.pkl │   └── scaler.pkl ├── templates/ │   └──
index.html ├── static/ │   └── style.css └── data/     ├── data_2016.csv
└── data_2017.csv
```

7. Deployment Instructions

Step 1: Environment Setup

```
python -m venv venv source venv/bin/activate pip install flask pandas numpy
scikit-learn joblib
```

Step 2: Run Flask Application

```
python app.py # Application runs at http://localhost:5000
```

8. Conclusion

This Low-Level Design document provides comprehensive technical specifications for implementing the Cryptocurrency Liquidity Prediction system. It covers all implementation details from data processing algorithms to web application deployment, ensuring developers have clear guidance for building and maintaining the system.

The modular architecture, detailed code examples, and visual diagrams enable easy understanding, testing, and future enhancements to the system.