EDA Report Summary

Source Data

- Two CSV files from CoinGecko:
    - `coin_gecko_2022-03-16.csv`
    - `coin_gecko_2022-03-17.csv`
- Columns include: `coin`, `price`, `1h`, `24h`, `7d`, `24h_volume`, `mkt_cap`, `date`

Cleaning Steps

- Merged the two files with an added `date` column
- Removed unnecessary columns like `id`, `name`, `symbol`, `image`
- Dropped rows with missing or invalid values
- Renamed columns to match standard ML terms (e.g., `24h_volume → total_volume`)

Data Insights

- Price ranged from fractions of a dollar to over $40,000 (Bitcoin)
- Market caps vary widely — from low caps to hundreds of billions
- Volume data was highly skewed: a few coins dominate trading volume
- Significant price changes observed over 24h: useful for feature engineering

Key Observations

- Highly imbalanced liquidity in the market — just a few coins are truly liquid
- Some fields had formatting issues (%, commas, symbols) that were cleaned

- Missing values mostly from very new or inactive coins — filtered out

Final Cleaned Dataset

- Rows: ~8,000 (after cleaning)
- Columns used: `coin`, `price`, `24h_volume`, `mkt_cap`, `price_change_24h`, `date`

Pipeline Architecture

Visual Diagram

Raw CSVs → Data Preparation → Cleaned CSV

↓

Feature Engineering → Engineered CSV

↓

Model Training → Model + Scaler

↓

Streamlit App

Pipeline Architecture Document

1. **Data Collection**
    - Historical crypto market data from CoinGecko in CSV format.
    - Two separate dates: `2022-03-16` and `2022-03-17`.
2. **Data Preparation (`data_preparation_ml_project_sid.py`)**
    - Reads both files, appends a `date` column.

- Drops unused columns and rows with NaN values.
- Saves output to `merged_cleaned_crypto_data.csv`.

3. **Feature Engineering (`feature_engineering_ml_project_sid.py`)**
    - Computes:
        - `liquidity_ratio = volume / market_cap`
        - `price_change_ratio = price_change_percentage / current_price`
        - Other helper features
    - Converts liquidity into `High`, `Medium`, `Low` using `qcut`
    - Saves to `engineered_crypto_data.csv`
4. **Model Training (`model_training_ml_project_sid.py`)**
    - Splits into train/test sets, scales features
    - Trains `RandomForestClassifier`
    - Saves model as `trained_model.pkl` and scaler as `scaler.pkl`
5. **Prediction Layer (`app_ml_project_sid.py`)**
    - Streamlit app takes user inputs
    - Computes ratios, scales inputs
    - Predicts class using trained model

This end-to-end flow follows a modular, reusable design where each step can be reused for any new market dataset with minimal code changes.