Low Level Design (LLD)

Project Title:

Cryptocurrency Liquidity Prediction for Market Stability

Objective:

To build a predictive system that forecasts liquidity levels for major cryptocurrencies, aiding market participants and regulators in detecting potential instability.

Data Flow Diagram:

Data Collection —> Data Preprocessing —> Feature Engineering
|
Model Training (ML)
|
Model Evaluation
|
Deployment & API

Major Components:

1. **Data Ingestion:**
    - Two CSV files containing cryptocurrency data were loaded using the pandas library.
        - ➢ march16.csv
        - ➢ march17.csv
    - Merged the dataset into a single DataFrame to enable unified analysis and modeling.
2. **Data Cleaning**:

- Dropped rows with missing values using dropna().
- Removed duplicate records using drop_duplicates().
- Converted columns to appropriate data types:

    ➢ Dates : datetime using pd.to_datetime()
    ➢ Numeric columns : float using astype()

3. Feature Engineering:

- Created moving average features:
    ➢ Price_ma_2 = 2-day moving average of price
    (price + price.shift(1)) / 2

    ➢ Mkt_cap_ma_2 = 2-day moving average of market cap
    (market_cap + market_cap.shift(1)) / 2

- Crated volatility feature:
    ➢ Volatility = rolling standard deviation of returns
    (price.max() - price.min()) / price.mean()

- Calculated liquidity ratio:
    ➢ liquidity_ratio = volume_24h / market cap

4. Exploratory Data Analysis (EDA):

- Plotted the Bitcoin price trend over time using Matplotlib.
- Created a correlation heatmap using the `seaborn.heatmap()` function to visualize relationships between numerical variables.
- Displayed summary statistics using the `df.describe()` function to understand the central tendency, dispersion, and shape of the dataset's distribution.

5. Model Building:

- The dataset was split the data into training and testing sets using the `train_test_split()` function from Scikit-learn.
- A basic Linear Regression model was initially developed to establish a baseline.
- A Random Forest Regressor was selected as the final model due to its better performance.
- Hyperparameter tuning was performed using grid search or randomized search, if required, to optimize model accuracy.

6. **Model Evaluation:**

The models performance was evaluated using the following metrics:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R2 Score (Coefficient of Determination)

These metrics helped assess prediction accuracy and model generalization capability.

7. **Model Saving:**

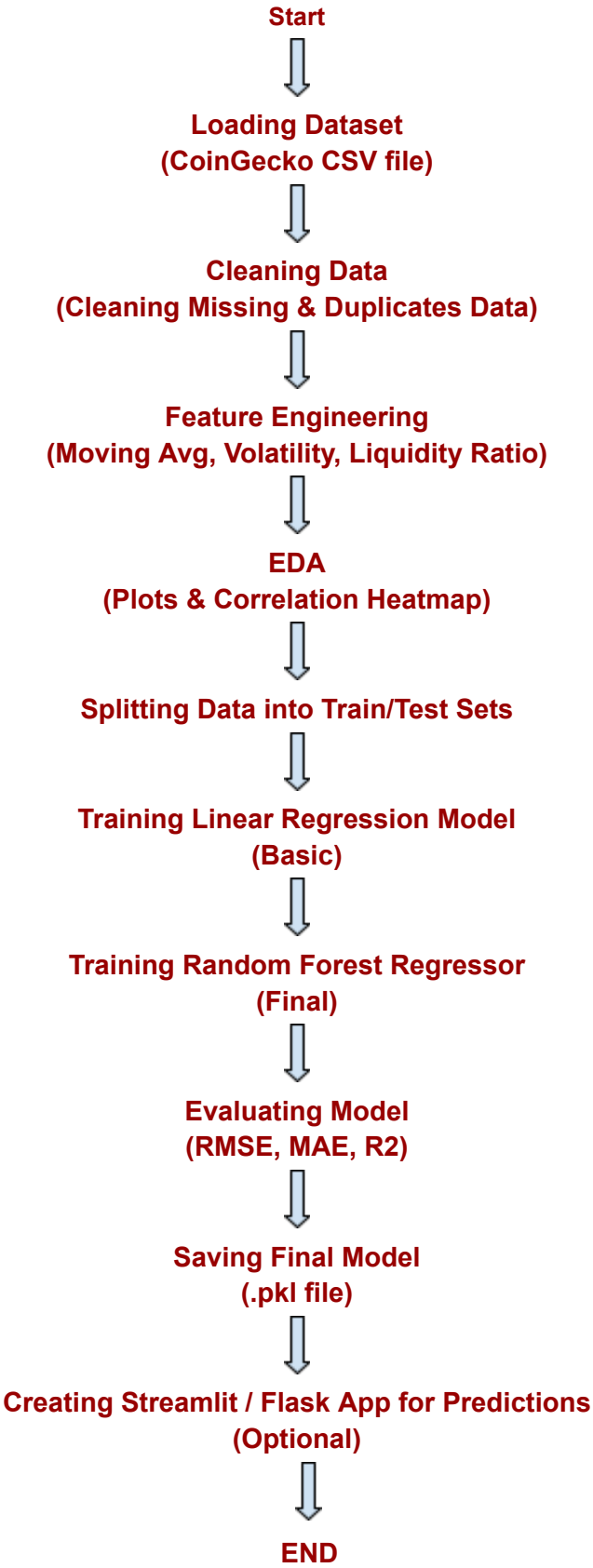
- The best- performing model was saved using Joblib for deployment purposes:

  joblib.dump(model, 'liquidity_prediction_model.pkl')

8. **Local Deployment (Optional):**

- A simple Streamlit or Flask application can be created to:
  - ➢ Load the saved .pkl model
  - ➢ Accept new input features
  - ➢ Predict and display the liquidity output in real-time**.**

Low level Design


Start
Loading Dataset
(CoinGecko CSV file)
Cleaning Data
(Cleaning Missing & Duplicates Data)
Feature Engineering
(Moving Avg, Volatility, Liquidity Ratio)
EDA
(Plots & Correlation Heatmap)
Splitting Data into Train/Test Sets
Training Linear Regression Model
(Basic)
Training Random Forest Regressor
(Final)
Evaluating Model
(RMSE, MAE, R2)
Saving Final Model
(.pkl file)
Creating Streamlit / Flask App for Predictions
(Optional)
END