High Level Design (HLD)

1. Project Overview

The Cryptocurrency Volatility Prediction system is developed to analyze historical cryptocurrency market data and predict daily price volatility. The system uses machine learning techniques to identify patterns in price movement, trading volume, and market capitalization. The predicted volatility can help traders and analysts understand market risk and take informed decisions.

---

2. System Objectives

- Analyze historical cryptocurrency price data
- Engineer meaningful features related to volatility and liquidity
- Build a machine learning model to predict volatility
- Evaluate model performance using standard regression metrics

---

3. Input and Output

Input Data

- Timestamp
- Date
- Cryptocurrency Name
- Open Price
- High Price
- Low Price
- Close Price
- Trading Volume
- Market Capitalization

Output

- Predicted volatility value
- Model evaluation metrics (R^2, MAE, MSE)

---

4. System Architecture

Overall Flow:

Data Source (CSV Dataset) → Data Preprocessing → Feature Engineering → Exploratory Data Analysis (EDA) → Model Training → Model Evaluation → Volatility Prediction Output

5. Major System Components

5.1 Data Ingestion Module

- Loads dataset from CSV file
- Removes unnecessary columns
- Converts date and timestamp columns into datetime format

5.2 Data Preprocessing Module

- Checks for missing values
- Sorts data based on date
- Scales numerical features using StandardScaler

5.3 Feature Engineering Module

- Calculates daily volatility
- Generates rolling volatility (7-day and 14-day)
- Computes moving averages
- Calculates liquidity ratio

5.4 EDA Module

- Generates statistical summaries
- Performs correlation analysis
- Visualizes correlations using heatmaps

5.5 Model Training Module

- Uses Random Forest Regressor
- Trains model on processed dataset

5.6 Model Optimization Module

- Applies GridSearchCV for hyperparameter tuning
- Selects best-performing model

5.7 Model Evaluation Module

- Evaluates model using R^2 score, MAE, and MSE

6. Tools and Technologies

- Programming Language: Python
- Libraries: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
- Development Environment: Jupyter Notebook / VS Code

7. Conclusion

The system provides an effective approach to analyze and predict cryptocurrency volatility using machine learning. Modular design ensures scalability and easy maintenance.

Low Level Design (LLD)

1. Data Loading

The dataset is loaded from a CSV file using Pandas. An unnamed index column is removed to clean the dataset.

```
data = pd.read_csv('dataset.csv')
data = data.iloc[:, 1:]
```

2. Data Inspection

- Check for missing values
- Verify data types
- Inspect dataset structure using `info()`

3. Date-Time Processing

Timestamp and date columns are converted into datetime format. The dataset is sorted by date to maintain time-series order.

```
data['timestamp'] = pd.to_datetime(data['timestamp'])
data['date'] = pd.to_datetime(data['date'])
data.sort_values('date', inplace=True)
```

4. Feature Scaling

Numerical features are scaled using StandardScaler to normalize feature ranges.

```
scaler = StandardScaler()
num_col = ['open','high','low','close','volume','marketCap']
data[num_col] = scaler.fit_transform(data[num_col])
```

---

5. Feature Engineering

5.1 Volatility Calculation

Volatility is calculated using the formula:

```
data['volatility'] = (data['high'] - data['low']) / data['close']
```

5.2 Rolling Volatility

```
data['volatility_7d'] = data['close'].rolling(7).std()
data['volatility_14d'] = data['close'].rolling(14).std()
```

5.3 Moving Averages

```
data['ma7'] = data['close'].rolling(7).mean()
data['ma14'] = data['close'].rolling(14).mean()
```

5.4 Liquidity Ratio

```
data['liquidity_ratio'] = data['volume'] / data['marketCap']
```

After feature generation, rows containing null values are removed.

---

6. Exploratory Data Analysis

- Compute correlation matrix
- Visualize correlations using heatmap

```
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
```

---

7. Model Building

7.1 Feature and Target Selection

```
X = data.drop(['volatility','timestamp','date','crypto_name'], axis=1)
y = data['volatility']
```

7.2 Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
```

---

8. Model Training

Random Forest Regressor is used to train the model.

```
model = RandomForestRegressor(n_estimators=100, random_state=1)
model.fit(X_train, y_train)
```

---

9. Model Evaluation

Model performance is evaluated using regression metrics.

```
r2_score(y_test, y_pred)
mean_absolute_error(y_test, y_pred)
mean_squared_error(y_test, y_pred)
```

---

10. Model Optimization

GridSearchCV is applied to tune hyperparameters and select the best model.

---

11. Final Output

The optimized model predicts cryptocurrency volatility with improved accuracy and reliability.

---

12. Conclusion

The Low-Level Design clearly explains each implementation step of the system, ensuring transparency, reproducibility, and ease of understanding.