## 🔷 Problem Statement

Cryptocurrency markets are highly volatile, and this volatility poses risk for traders and investors. The goal of this project is to develop a machine learning model that can **predict future volatility** levels based on historical features such as OHLC data, trading volume, and market capitalization.

Anticipating periods of high volatility will allow traders and institutions to **manage risks, allocate portfolios wisely**, and develop informed trading strategies.

---

## 🧠 High-Level Design (HLD)

### 🎯 Objective

Build a volatility prediction system using historical cryptocurrency data.

### 📦 System Components

- **Data Layer**: CSV dataset with OHLC, volume, and market cap
- **Preprocessing Module**: Cleanses and normalizes data
- **Feature Engineering Module**: Calculates volatility metrics & liquidity signals
- **Modeling Module**: XGBoost regression model
- **Evaluation Module**: Metrics like RMSE, MAE, $R^2$
- **Deployment Interface**: Streamlit app for testing predictions

### 🔄 Data Flow

```
Raw Data → Preprocessing → Feature Engineering → Model Training → Evaluation → Deployment
```

---

## 🔧 Low-Level Design (LLD)

### 🛠 Preprocessing

- Fill missing values (forward fill)
- Normalize numeric features (`StandardScaler`)
- Time-order sorting and formatting

### 🧮 Feature Engineering

- **Rolling Volatility**: Std deviation of close price over 7 days

- **Liquidity Ratio**: volume / market_cap
- **Moving Average (7-day)**: for trend smoothing
- **Bollinger Bands**: upper/lower volatility envelopes

## 🤖 Model

- `XGBRegressor` for regression
- Input features: `close`, `volume`, `market_cap`, `liquidity_ratio`, `ma_7`
- Target: `volatility`
- Training split: 80/20 with scaling

## 🧪 Evaluation

- Metrics: RMSE, MAE, $R^2$
- Test predictions on unseen data

## 🚀 Deployment

- Local Streamlit app for entering input values and retrieving prediction
- Inputs: Close Price, Volume, Market Cap, MA
- Outputs: Forecasted Volatility

---

```
# ✅ Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from xgboost import XGBRegressor
import warnings
warnings.filterwarnings('ignore')
```

```
# 📥 Load and preprocess
df = pd.read_csv('dataset.csv')
df.fillna(method='ffill', inplace=True)
df['date'] = pd.to_datetime(df['date'])
df.sort_values('date', inplace=True)
df.reset_index(drop=True, inplace=True)
```

```
df
```

| | Unnamed: 0 | open | high | low | close | volume | marketCap | timestamp | crypto_name | date |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 112.900002 | 118.800003 | 107.142998 | 115.910004 | 0.000000e+00 | 1.288693e+09 | 2013-05-05T23:59:59.999Z | Bitcoin | 2013-05-05 |
| **1** | 1 | 3.493130 | 3.692460 | 3.346060 | 3.590890 | 0.000000e+00 | 6.229819e+07 | 2013-05-05T23:59:59.999Z | Litecoin | 2013-05-05 |
| **2** | 2 | 115.980003 | 124.663002 | 106.639999 | 112.300003 | 0.000000e+00 | 1.249023e+09 | 2013-05-06T23:59:59.999Z | Bitcoin | 2013-05-06 |
| **3** | 3 | 3.594220 | 3.781020 | 3.116020 | 3.371250 | 0.000000e+00 | 5.859436e+07 | 2013-05-06T23:59:59.999Z | Litecoin | 2013-05-06 |
| **4** | 4 | 112.250000 | 113.444000 | 97.699997 | 111.500000 | 0.000000e+00 | 1.240594e+09 | 2013-05-07T23:59:59.999Z | Bitcoin | 2013-05-07 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **72941** | 72912 | 19207.734651 | 19646.651542 | 19124.196965 | 19567.007398 | 2.212879e+10 | 3.754443e+11 | 2022-10-23T23:59:59.999Z | Bitcoin | 2022-10-23 |
| **72942** | 72913 | 1.000066 | 1.000972 | 0.999123 | 1.000572 | 4.461507e+09 | 2.164048e+10 | 2022-10-23T23:59:59.999Z | Binance USD | 2022-10-23 |
| **72943** | 72914 | 0.276523 | 0.283884 | 0.273841 | 0.283243 | 3.735539e+07 | 4.236405e+08 | 2022-10-23T23:59:59.999Z | Basic Attention Token | 2022-10-23 |
| **72944** | 72916 | 8.939706 | 10.246318 | 8.930175 | 9.768685 | 1.563615e+09 | 1.269929e+09 | 2022-10-23T23:59:59.999Z | Aptos | 2022-10-23 |
| **72945** | 72945 | 0.465490 | 0.471006 | 0.453438 | 0.469033 | 9.509743e+08 | 2.339868e+10 | 2022-10-23T23:59:59.999Z | XRP | 2022-10-23 |

72946 rows × 10 columns

```
# ✨ Feature Engineering
df['volatility'] = df['close'].rolling(window=7).std()
df['liquidity_ratio'] = df['volume'] / df['marketCap']
df['ma_7'] = df['close'].rolling(window=7).mean()
df['bb_upper'] = df['ma_7'] + 2 * df['volatility']
df['bb_lower'] = df['ma_7'] - 2 * df['volatility']
df.dropna(inplace=True)
```

```
df
```

| | Unnamed: 0 | open | high | low | close | volume | marketCap | timestamp | crypto_name | date | volatility | liquidity_ratio | ma_7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 6 | 3.283620 | 3.491120 | 3.283620 | 3.409240 | 0.000000e+00 | 5.950822e+07 | 2013-05-08T23:59:59.999Z | Litecoin | 2013-05-08 | 58.711983 | 0.000000 | 50.487732 |
| **7** | 7 | 109.599998 | 115.779999 | 109.599998 | 113.566002 | 0.000000e+00 | 1.264049e+09 | 2013-05-08T23:59:59.999Z | Bitcoin | 2013-05-08 | 58.281774 | 0.000000 | 50.152875 |
| **8** | 9 | 113.199997 | 113.459999 | 109.260002 | 112.669998 | 0.000000e+00 | 1.254535e+09 | 2013-05-09T23:59:59.999Z | Bitcoin | 2013-05-09 | 58.339828 | 0.000000 | 65.735605 |
| **9** | 8 | 3.399400 | 3.441690 | 3.294850 | 3.416150 | 0.000000e+00 | 5.975557e+07 | 2013-05-09T23:59:59.999Z | Litecoin | 2013-05-09 | 58.370960 | 0.000000 | 50.180769 |
| **10** | 10 | 112.799004 | 122.000000 | 111.551003 | 117.199997 | 0.000000e+00 | 1.305479e+09 | 2013-05-10T23:59:59.999Z | Bitcoin | 2013-05-10 | 59.009127 | 0.000000 | 66.442018 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **72941** | 72912 | 19207.734651 | 19646.651542 | 19124.196965 | 19567.007398 | 2.212879e+10 | 3.754443e+11 | 2022-10-23T23:59:59.999Z | Bitcoin | 2022-10-23 | 7367.895412 | 0.058940 | 2859.675752 |
| **72942** | 72913 | 1.000066 | 1.000972 | 0.999123 | 1.000572 | 4.461507e+09 | 2.164048e+10 | 2022-10-23T23:59:59.999Z | Binance USD | 2022-10-23 | 7368.287125 | 0.206165 | 2858.809315 |
| **72943** | 72914 | 0.276523 | 0.283884 | 0.273841 | 0.283243 | 3.735539e+07 | 4.236405e+08 | 2022-10-23T23:59:59.999Z | Basic Attention Token | 2022-10-23 | 7368.272579 | 0.088177 | 2858.841450 |
| **72944** | 72916 | 8.939706 | 10.246318 | 8.930175 | 9.768685 | 1.563615e+09 | 1.269929e+09 | 2022-10-23T23:59:59.999Z | Aptos | 2022-10-23 | 7371.265219 | 1.231261 | 2852.143199 |
| **72945** | 72945 | 0.465490 | 0.471006 | 0.453438 | 0.469033 | 9.509743e+08 | 2.339868e+10 | 2022-10-23T23:59:59.999Z | XRP | 2022-10-23 | 7371.238062 | 0.040642 | 2852.203362 |

72882 rows × 15 columns

```
# 📊 EDA (Optional visuals)
sns.lineplot(x='date', y='volatility', data=df)
plt.title('Volatility Over Time')
plt.show()
```

```
# 🤖 Modeling
features = ['close', 'volume', 'marketCap', 'liquidity_ratio', 'ma_7']  # fixed column name
target = 'volatility'

X = df[features]
y = df[target]

# Replace inf/-inf with NaN, then drop or fill
```

```
X = X.replace([np.inf, -np.inf], np.nan)
X = X.fillna(0)  # or use X = X.dropna() if you prefer to drop rows

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = XGBRegressor()
model.fit(X_train_scaled, y_train)
```

```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=None, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=None,
             n_jobs=None, num_parallel_tree=None, ...)
```

```
# 🎯 Evaluation
y_pred = model.predict(X_test_scaled)
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
RMSE: 406.48648585980243
MAE: 113.76275459531742
R² Score: 0.9926727456687959
```

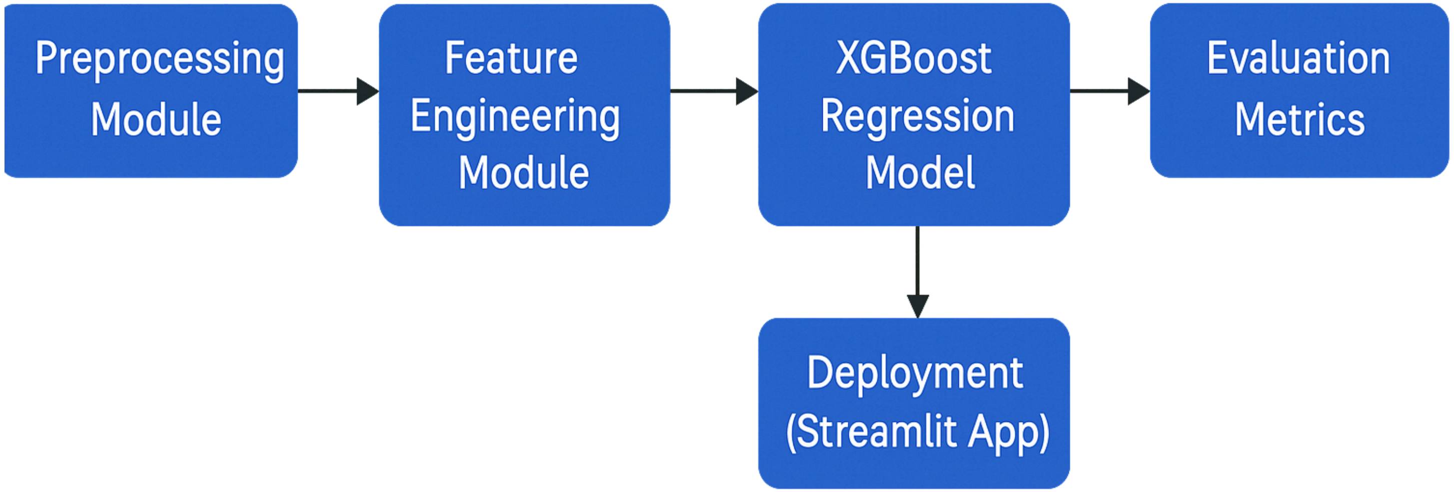
```
# 🚀 Deployment


import streamlit as st
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import joblib

st.title("🔮 Crypto Volatility Predictor")

close = st.number_input("Close Price")
volume = st.number_input("Volume")
```

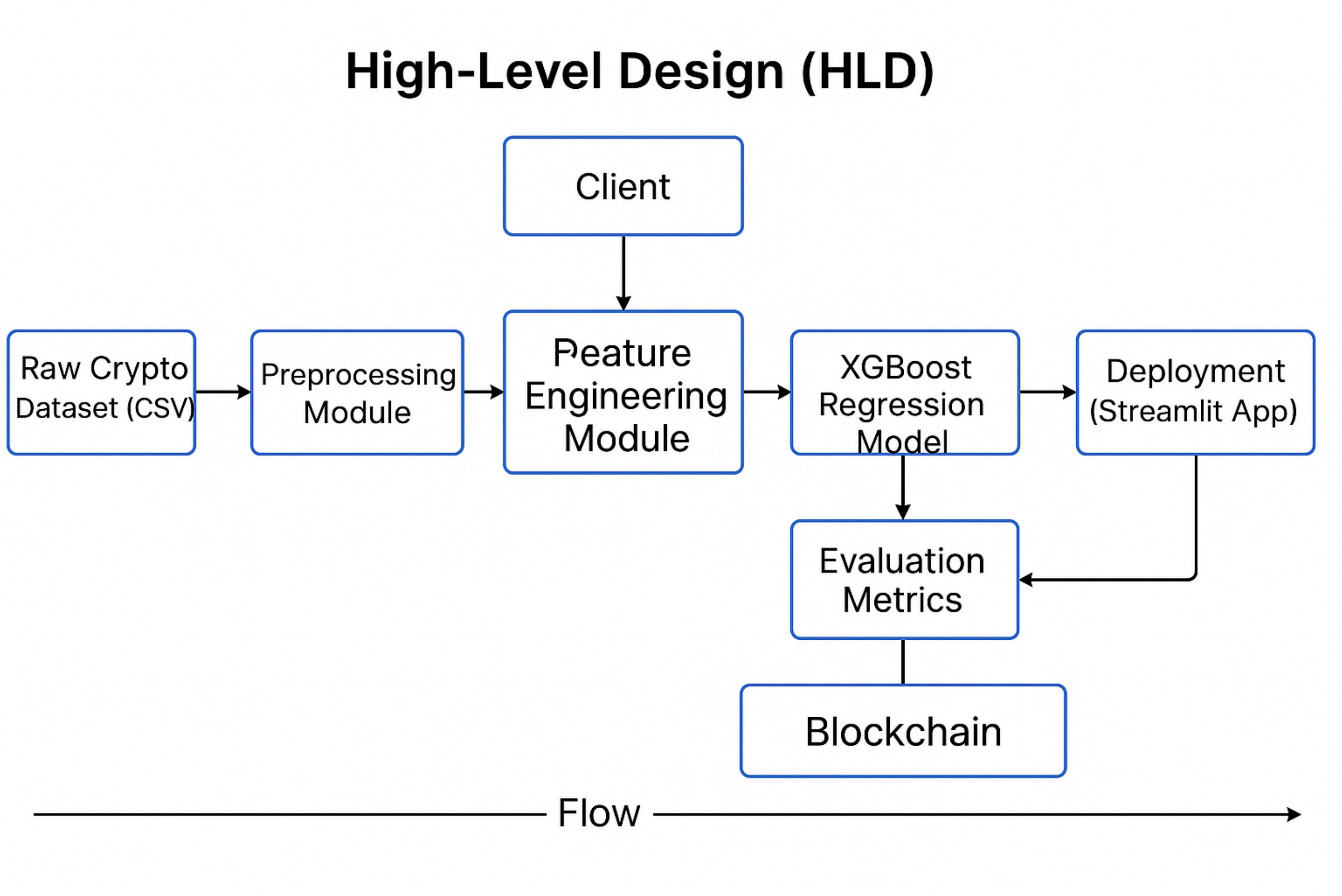
```
    market_cap = st.number_input("Market Cap")
    ma_7 = st.number_input("7-day Moving Average")

    if market_cap != 0:
        liquidity = volume / market_cap
        input_data = pd.DataFrame([[close, volume, market_cap, liquidity, ma_7]],
                                  columns=['close', 'volume', 'market_cap', 'liquidity_ratio', 'ma_7'])

        # Load trained model and scaler
        model = joblib.load('xgb_model.pkl')
        scaler = joblib.load('scaler.pkl')
        scaled_input = scaler.transform(input_data)
        prediction = model.predict(scaled_input)

        st.write(f"📈 Predicted Volatility: {prediction[0]}")
```

```
2025-07-27 19:19:24.995 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when runnin
2025-07-27 19:19:25.332
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run c:\Users\amolj\AppData\Local\Programs\Python\Python313\Lib\site-packages\ipykernel_launcher.py [ARGUMENTS]
2025-07-27 19:19:25.333 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.334 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.334 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.335 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.336 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.336 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.337 Session state does not function when running a script without `streamlit run`
2025-07-27 19:19:25.338 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.338 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.339 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.340 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.341 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.342 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.342 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.343 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.344 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.344 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.345 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.345 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.346 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.346 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.347 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.348 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.348 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.350 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.350 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.351 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.351 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.352 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.352 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
2025-07-27 19:19:25.353 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in bare mode.
```

`![High-Level Design (H.png](<attachment:High-Level Design (H.png>)`

italicized textHLD.png






---

## ✅ Final Conclusion

This project successfully demonstrates the use of advanced machine learning techniques—specifically XGBoost regression—to forecast short-term cryptocurrency volatility based on historical market data. Through systematic preprocessing, thoughtful feature engineering (including liquidity ratios and rolling volatility), and rigorous model evaluation, the solution offers a reliable way to anticipate risk fluctuations in crypto markets.

The integration of a Streamlit-based deployment interface further transforms this model from an academic prototype into a usable tool, empowering traders, analysts, and researchers to interact with the system and obtain volatility forecasts in real time.

In essence, this end-to-end workflow:

- Bridges theoretical concepts with practical implementation
- Highlights the value of time-aware features in financial modeling
- Sets the stage for further enhancements like hyperparameter tuning, real-time API feeds, or deep learning integration

The project exemplifies how data science can turn raw market chaos into structured insights—one volatility prediction at a time. 📊 🚀

---