**BTC Price Movement Prediction – Project Documentation**

**Overview**

This project is an end-to-end machine learning pipeline that predicts the next-hour price movement of Bitcoin using real-time data. It includes live data ingestion, feature engineering, model training (XGBoost), a Streamlit-based frontend, Docker deployment, and a lightweight model monitoring dashboard.

**1. Process Map**

A diagram of a software development process

Description automatically generated

Pipeline Steps:

CryptoCompare API → Feature Engineering → XGBoost Model → Streamlit App → Docker Deployment → Monitoring Dashboard

Each step is modular and designed for retraining, model updates, and scaling.

**2. Data Ingestion**

Source: [ CryptoCompare API ](https://min-api.cryptocompare.com/)

Data Types:

- Hourly OHLCV (Open, High, Low, Close, Volume)

- Real-time price and volume (current data)

- Frequency: Hourly window (rolling)

**3. Predictive Model**

Model Type: Binary classification (Up vs Down/Neutral)

Framework: XGBoost

Features:

- Momentum, rolling volatility, SMA-24, SMA-168

- Return over 1h, 3h, and 6h

- Target: Next-hour close minus current close (target = close[t+1] - close[t])

- Threshold: Price increase > $50 = class 1 (up)

**4. Streamlit Application**

App: `btc\_predictor.py`

Features:

- Current hour prediction with confidence

- Last 10 hourly predictions with actual vs predicted

- In-progress candle display

- Live BTC price + 24h change

**5. Docker Deployment**

- Dockerized for platform-independent deployment

- Run locally:

Bash

docker build -t btc-app .

docker run -p 8501:8501 btc-app

- View app: http://localhost:8501

**6. Monitoring Dashboard (Embedded in App)**

Tracks:

- Recent prediction distribution (↑ vs ↓)

- Average prediction confidence

- Real-time prediction logs

- In-progress data vs model inputs

**7. Risks in Production**

Model Drift: Market behavior may change (volatility regimes)

Data Availability: CryptoCompare may limit API calls or change structure

Confidence Misuse: Low confidence predictions may be over-trusted

Lag in Real-Time Data: Partial candles could mislead if used incorrectly

**8. Repository Structure**

bitcoin\_mlops/

── data/

── btc\_hourly\_ohlc\_.csv

── data\_injestion.py

── model/

── btc\_xgb\_classifier.pkl

── model\_features\_train.ipynb

── docker/

── app.py

── requirements.txt

── Dockerfile

── btc\_hourly\_ohlc\_.csv

── .streamlit/

── config.toml

**9. Getting Started**

Bash

* Build & run the app locally

cd app/

docker build -t btc-app .

docker run -p 8501:8501 btc-app

* Cloud deployment to https://streamlit.io/cloud using the app folder contents.