

Trường: Đại học FPT
Chuyên Ngành: Trí tuệ nhân tạo

---□□□---

DAP391m REPORT

Dự án Phân tích & Trực quan dữ liệu tiền ảo



Giáo viên hướng dẫn : Đoàn Nguyễn Thành Hòa
Sinh viên thực hiện: Lâm Nguyễn Minh Thanh
Trần Ngọc Minh
Hồ Gia Phú

Lớp : AI1915

- HCM, 2025 -

Mục lục

| | |
|---|----|
| 1.1 Mục tiêu dự án: | 5 |
| 1.2 Người dùng cuối : | 5 |
| 1.3 Phạm vi thực hiện: | 5 |
| 2.1 Thư viện chuẩn bị: | 6 |
| 2.2 Cấu trúc thư mục: | 7 |
| 2.3 Lấy dữ liệu các mã coin: | 8 |
| 2.3.1 Setup đầu tiên: | 8 |
| 2.3.2 Lấy dữ liệu các symbol: | 8 |
| 2.3.3 Hàm <code>fetch_binance_klines</code> | 9 |
| 2.3.4 Hàm <code>fetch_binance_funding</code> | 10 |
| 2.3.5 Lấy dữ liệu Fear & Greed Index | 12 |
| 2.3.6 Lấy dữ liệu On-chain metrics | 15 |
| 2.3.7. Tổng hợp dữ liệu hoàn chỉnh cho từng đồng (<i>Build Dataset</i>) | 18 |
| 2.4. Gộp dữ liệu các mã coin | 21 |
| 2.4.1. Gộp tất cả file <code>_clean</code> của từng coin | 21 |
| 2.4.2. Đổi tên cột cho đồng nhất & dễ đọc | 23 |
| 2.5. Tiền xử lý dataset tổng | 25 |
| 2.5.1. Đọc file dữ liệu tổng | 25 |
| 2.5.2. Ép kiểu dữ liệu và xử lý giá trị thiếu | 25 |
| 2.5.3. Ghi đè dữ liệu đã xử lý lên file cũ | 25 |
| 2.6. Scale dataset tổng | 26 |
| 2.6.1. Xác định cột numeric cần scale | 26 |
| 2.6.2. Scale riêng từng đồng coin | 26 |
| 2.6.3. Gộp file đã scale | 27 |
| 3.0 Mục tiêu phân tích | 27 |
| 3.0.1 Mục đích | 27 |
| 3.0.2 Phương pháp tổng quát | 28 |
| 3.0.3 Kết quả mong đợi | 28 |
| 3.1 Load and setup | 28 |
| 3.1.1. Phương pháp | 28 |
| 3.1.2. Kết quả | 29 |
| 3.1.3. Nhận xét | 30 |
| 3.2. Thống kê mô tả (<i>Descriptive Statistics</i>) | 30 |
| 3.2.1. Phương pháp | 30 |
| 3.2.2. Kết quả | 30 |
| 3.2.3. Nhận xét | 32 |
| 3.3.1. Phương pháp | 32 |
| 3.3.2. Kết quả | 36 |
| 3.3.3. Nhận xét | 36 |

| | |
|--|----|
| <i>3.5. Phân tích hồi quy tuyến tính (Linear Relationship Test)</i> | 43 |
| 3.5.1. Mục tiêu | 43 |
| 3.5.2. Mô hình | 43 |
| 3.5.3. Kết quả | 44 |
| 3.5.4. Nhận xét | 46 |
| <i>3.6. Phân tích quan hệ giữa các coin (Cross-Market Correlation)</i> | 47 |
| 3.6.1. Mục tiêu | 47 |
| 3.6.2. Kết quả | 48 |
| <i>4.1 Chuẩn bị</i> | 51 |
| <i>4.2 Waffle Chart — Tỷ trọng khối lượng giao dịch giữa các mã</i> | 51 |
| <i>4.3 Area Plot — Xu hướng giá theo thời gian</i> | 52 |
| <i>4.4 Histogram — Phân phối lợi nhuận ngày (Return)</i> | 56 |
| <i>4.5 Bar Chart — Khối lượng giao dịch trung bình giữa các mã</i> | 59 |
| <i>4.6 Pie Chart — Tỷ lệ lợi nhuận trung bình giữa các mã</i> | 60 |
| <i>4.7 Scatter Plot — Mối quan hệ Volume và Close</i> | 61 |
| <i>4.8 Line Chart — Giá đóng cửa theo thời gian</i> | 65 |
| <i>5.1. Trang chính (index.html)</i> | 69 |
| 5.1.1 Giới thiệu trang web | 69 |
| 5.1.2. Cấu trúc giao diện hiển thị | 69 |
| 5.1.3. Tính năng tổng hợp và hiển thị | 70 |
| <i>5.2. Trang thống kê (Dashboard.html)</i> | 70 |
| 5.2.1. Mục tiêu và ý tưởng thiết kế | 71 |
| 5.2.2. Các Thành phần | 71 |
| 5.2.3. Giá trị ứng dụng | 71 |
| <i>5.3. Trang biểu đồ (Chart.html)</i> | 72 |
| 5.3.1. Mục tiêu | 72 |
| 5.3.2. Cấu trúc hiển thị | 72 |
| 5.3.3. Cách thức hoạt động | 72 |
| 5.3.4. Nhận xét | 73 |
| <i>5.4. Trang agent (chatbot.html)</i> | 73 |
| 5.4.1. Mục tiêu | 74 |
| 5.4.2. Cấu trúc hiển thị | 74 |
| 5.3.3. Cách thức hoạt động | 74 |
| 5.3.4. Nhận xét | 75 |
| <i>6.1. Mục tiêu</i> | 75 |
| <i>6.2. Kiến trúc hệ thống</i> | 75 |
| <i>6.3. Triển khai</i> | 76 |
| <i>6.4. Tích hợp giao diện</i> | 78 |

1. Business Understanding & Analytic Approach

1.1 Mục tiêu dự án:

- Xây dựng một nền tảng phân tích đa chiều dữ liệu từ nhiều loại tiền mã hóa lớn như: BTC, ETH, BNB, XRP, SOL.
- Cung cấp trực quan hóa phong phú, đa dạng để người dùng cuối dễ dàng tiếp cận dữ liệu phức tạp.
- Tích hợp chatbot AI có khả năng trả lời tự động dựa trên dữ liệu thực và ảnh biểu đồ.
- Thử nghiệm mô hình học máy để dự đoán biến động.
- Cập nhật các dữ liệu mới nhất để đặc trưng mô tả dữ liệu hoàn hảo nhất

1.2 Người dùng cuối :

- Nhà đầu tư cá nhân.
- Người dùng yêu thích Crypto.
- Sinh viên hoặc nhà nghiên cứu ngành khoa học dữ liệu.

1.3 Phạm vi thực hiện:

- Dữ liệu: lấy từ Binance API, Google Trends, Fear & Greed Index,...
- Giao diện người dùng: HTML/CSS/JS + Ollama llama3 + ngrok.
- Mô hình học máy: Scikit-learn, phân tích chuỗi thời gian,...
- Trực quan hóa: Matplotlib, Seaborn, Waffle, Plotly,...

2. Data Collection, Understanding & Preparation

2.1 Thư viện chuẩn bị:

Các thư viện cần chuẩn bị trước:

```
requests>=2.31.0
pandas>=2.0.0
tqdm>=4.65.0
pytrends>=4.9.2
numpy>=1.24.0
seaborn>=0.12.0
matplotlib>=3.7.0
scipy>=1.10.0
pywaffle>=1.1.1
scikit-learn>=1.3.0
openpyxl>=3.1.0
```

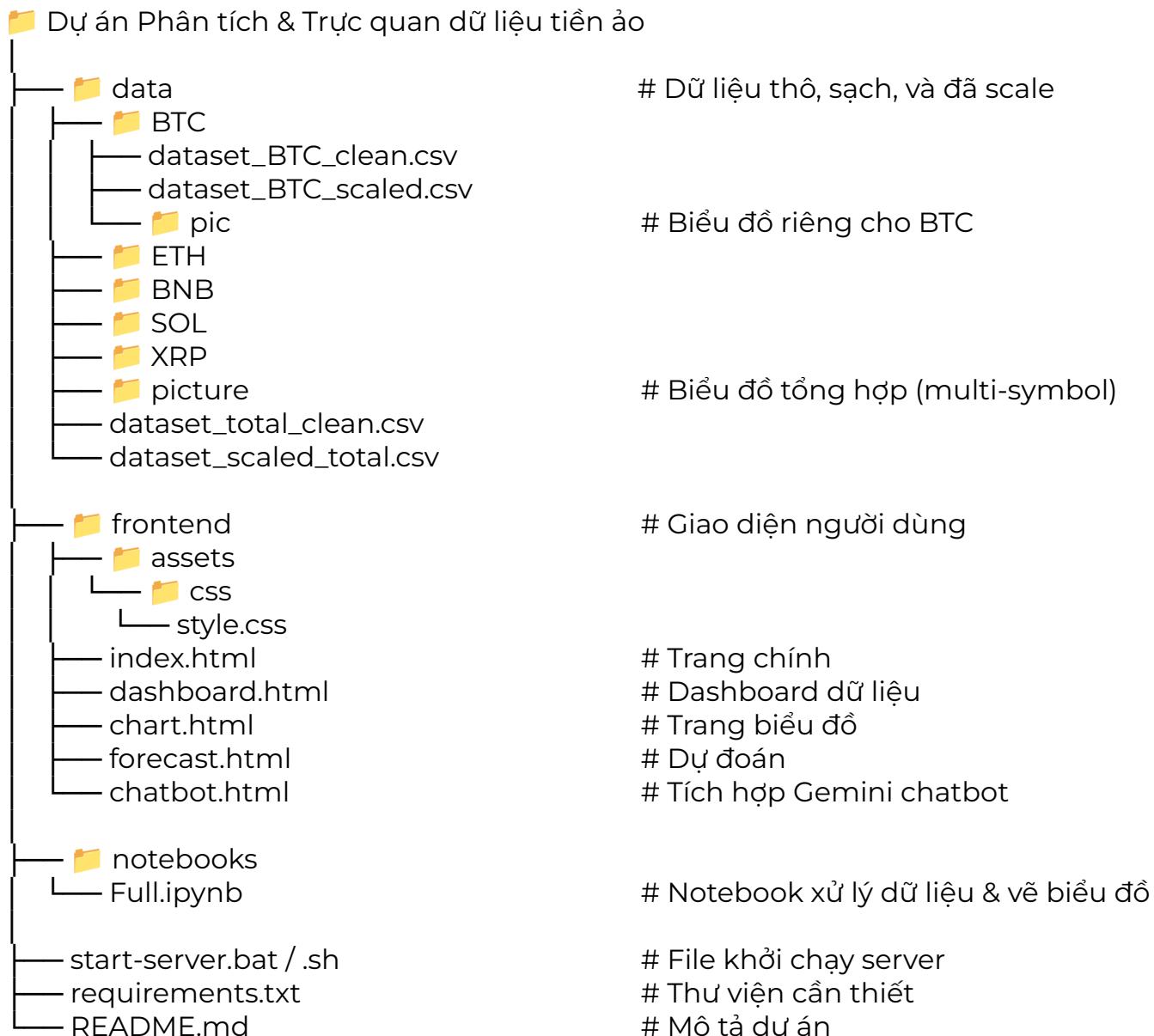
Tạo lệnh import các thư viện:

```
import glob
import math
import os
import random
import time
from datetime import datetime, timedelta, timezone

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import requests
import seaborn as sns
from pytrends.request import TrendReq
from pywaffle import Waffle
from scipy.stats import linregress
from sklearn.preprocessing import MinMaxScaler
from tqdm import tqdm

sns.set_theme(style="whitegrid")
plt.rcParams.update({"figure.autolayout": True})
```

2.2 Cấu trúc thư mục:



2.3 Lấy dữ liệu các mã coin:

2.3.1 Setup đầu tiên:

Cần setup trước các giá trị chung nhằm đảm bảo logic và tối ưu code khi bắt đầu chạy code lấy dữ liệu

```
# Thông tin thời gian
INTERVAL = "1d"                      # khung thời gian: "1d", "1h", ...
START_DATE = "2023-01-01"                # ngày bắt đầu
END_DATE = datetime.today().strftime("%Y-%m-%d")      # ngày kết thúc
```

2.3.2 Lấy dữ liệu các symbol:

Lấy dữ liệu cho từng mã bằng symbol của từng mã

Ví dụ bằng symbol của mã BTC

:
Gán biến `SYMBOL_BASE = "BTC".BTC` là mã symbol cổ phiếu của Bitcoin trên các sàn giao dịch và sẽ được lấy từ `2023-01-01 → Hiện tại`

```
# Dữ liệu coin bạn muốn lấy
SYMBOL_BASE = "BTC"                  # ví dụ: "BTC", "ETH", "BNB", "SOL", "XRP"
SYMBOL_BINANCE = SYMBOL_BASE + "USDT"  # tự tạo mã Binance Futures

# Thư mục lưu dữ liệu
OUT_DIR = os.path.join("data", SYMBOL_BASE)
os.makedirs(OUT_DIR, exist_ok=True)

print(f"✅ Đang lấy dữ liệu cho {SYMBOL_BASE} ({SYMBOL_BINANCE}) từ {START_DATE} →
{END_DATE}, interval={INTERVAL}")
```

Chuyển đổi giữa thời gian (datetime) và timestamp dạng mili-giây. Vì các dữ liệu tài chính/crypto như Binance, Coinbase, Yahoo Finance,... dùng Unix timestamp (milliseconds) để đánh dấu thời gian.

```
def dt_to_millis(dt: datetime):
    return int(dt.replace(tzinfo=timezone.utc).timestamp() * 1000)
def date_to_millis(s):
    dt = datetime.fromisoformat(s)
    return dt_to_millis(dt)
def millis_to_date(millis):
    return datetime.fromtimestamp(millis/1000, tz=timezone.utc).date()
```

2.3.3 Hàm fetch_binance_klines

Gửi request GET tới API Binance Futures

Nếu bị chặn (418) -> chuyển sang Binance Spot API

Tự động chia nhỏ request, tải nhiều trang dữ liệu

Gộp thành một DataFrame lớn

Chuẩn hóa dữ liệu

```
def fetch_binance_klines(symbol, interval, start_ts_ms, end_ts_ms):
    futures_url = "https://fapi.binance.com/fapi/v1/klines"
    spot_url = "https://api.binance.com/api/v3/klines"
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"}
    limit = 1000
    all_klines = []
    start = start_ts_ms
    use_spot = False

    while True:
        params = {
            "symbol": symbol,
            "interval": interval,
            "startTime": start,
            "endTime": end_ts_ms,
            "limit": limit
        }
        url = spot_url if use_spot else futures_url
        try:
            r = requests.get(url, params=params, headers=headers, timeout=30)
            if r.status_code == 418:
                print(f"[WARN] Binance Futures chặn IP (418) → chuyển sang Spot API.")
                use_spot = True
                continue
            r.raise_for_status()
            data = r.json()
        except requests.exceptions.HTTPError as e:
            print(f"[ERROR] {e}")
            break
        except Exception as e:
            print(f"[WARN] Network error: {e}")
            time.sleep(5)
            continue

        if not data:
            break

        all_klines.extend(data)
        if len(data) < limit:
            break
        last_open = data[-1][0]
        start = last_open + 1
        time.sleep(0.2)
```

```

# Chuyển sang DataFrame
if not all_klines:
    print("[WARN] Không lấy được dữ liệu Klines.")
    return
pd.DataFrame(columns=["Date","Open","High","Low","Close","Volume","Quote_Volume","Number_Of_Trades","symbol"])

cols = ["open_time","open","high","low","close","volume","close_time",
"quote_asset_volume","num_trades","taker_buy_base_vol","taker_buy_quote_vol","ignore"]

df = pd.DataFrame(all_klines, columns=cols)
df["Date"] = pd.to_datetime(df["open_time"], unit="ms", utc=True).dt.date
df = df[["Date","open","high","low","close","volume","quote_asset_volume","num_trades"]]
df.columns =
["Date","Open","High","Low","Close","Volume","Quote_Volume","Number_Of_Trades"]
for c in ["Open","High","Low","Close","Volume","Quote_Volume","Number_Of_Trades"]:
    df[c] = pd.to_numeric(df[c], errors="coerce")
df["symbol"] = SYMBOL_BASE
df = df.sort_values("Date").drop_duplicates("Date").reset_index(drop=True)

print(f"✅ Dữ liệu Binance ({'Spot' if use_spot else 'Futures'}) tải thành công: {len(df)} dòng")
return df

```

Ưu điểm:

- Có **retry**, tránh chết script
- Nhận 418 tự động chuyển Spot API – rất thực chiến
- Gộp dữ liệu lịch sử dài (max 10 năm)
- Kết cấu DataFrame **chuẩn phân tích thời gian**
- Có thể dùng cho backend API hoặc dashboard

Hàm **fetch_binance_funding()** dùng để **thu thập dữ liệu Funding Rate của một đồng tiền điện tử (crypto) từ Binance Futures API trong một khoảng thời gian nhất định**. Funding Rate là chỉ số thể hiện chi phí giữ vị thế Long/Short trên thị trường phái sinh và phản ánh **tâm lý đòn bẩy** của trader.

Hàm tiến hành tải dữ liệu theo dạng phân trang (do Binance giới hạn 1000 bản ghi mỗi lần gọi API), sau đó **gộp và chuyển đổi timestamp**, rồi **tính Funding Rate trung bình theo ngày**. Kết quả trả về là một **pandas DataFrame sạch và sẵn sàng để phân tích**, có thể dùng để kết hợp với dữ liệu giá (OHLC) nhằm đánh giá xu hướng thị trường.

2.3.4 Hàm fetch_binance_funding

Hàm **fetch_binance_funding()** được dùng để **thu thập dữ liệu Funding Rate** của một đồng tiền điện tử (crypto) từ **Binance Futures API** trong một khoảng thời gian xác định.

Funding Rate là chỉ số phản ánh chi phí nắm giữ vị thế Long/Short trên thị trường phái sinh, thể hiện **mức độ chênh lệch giữa giá hợp đồng tương lai và giá giao ngay**, qua đó phản ánh **tâm lý đòn bẩy của trader**.

```
def fetch_binance_funding(symbol, start_ts_ms, end_ts_ms):
    url = "https://fapi.binance.com/fapi/v1/fundingRate"
    limit = 1000
    all_rows = []
    start = start_ts_ms
    while True:
        params = {"symbol": symbol, "startTime": start, "endTime": end_ts_ms, "limit": limit}
        r = requests.get(url, params=params, timeout=30)
        r.raise_for_status()
        data = r.json()
        if not data:
            break
        all_rows.extend(data)
        if len(data) < limit:
            break
        last_t = data[-1]["fundingTime"]
        start = last_t + 1
        time.sleep(0.2)
    if not all_rows:
        return pd.DataFrame(columns=["Date", "funding_Rate"])
    df = pd.DataFrame(all_rows)
    df["fundingTime"] = pd.to_datetime(df["fundingTime"], unit="ms", utc=True)
    df["Date"] = df["fundingTime"].dt.date
    df["fundingRate"] = pd.to_numeric(df["fundingRate"], errors="coerce")
    daily = df.groupby("Date",
as_index=False)["fundingRate"].mean().rename(columns={"fundingRate": "funding_Rate"})
    return daily
```

Hàm gửi nhiều yêu cầu (GET request) đến **API Binance Futures** để lấy Funding Rate theo từng trang dữ liệu (mỗi lần giới hạn 1000 bản ghi).

Cơ chế hoạt động chi tiết:

while True: lặp liên tục để tải hết dữ liệu trong khoảng thời gian cho phép.

- Mỗi vòng lặp:
 - Tạo tham số truy vấn (`symbol`, `startTime`, `endTime`, `limit`).
 - Gửi yêu cầu qua `requests.get()` đến endpoint `"https://fapi.binance.com/fapi/v1/fundingRate"`.
 - Nếu hết dữ liệu (`len(data) < limit`) → dừng vòng lặp.
 - Cập nhật `start = last_t + 1` để tránh trùng dữ liệu khi tải trang kế tiếp.
- Sau khi lấy đủ dữ liệu, tất cả các bản ghi được gộp lại (`all_rows.extend(data)`).

Bước xử lý sau khi tải:

- Chuyển timestamp `fundingTime` (ms) → dạng ngày (`Date`).
- Chuyển giá trị `fundingRate` sang dạng số (`float`).

- Gom nhóm (groupby) theo Date, tính **trung bình Funding Rate mỗi ngày**, đổi tên cột thành funding_Rate.

Kết quả trả về:

- Một **pandas DataFrame** có hai cột:
 - Date: ngày giao dịch.
 - funding_Rate: Funding Rate trung bình trong ngày.
- Dữ liệu sạch, có thể ghép thẳng với dữ liệu giá (OHLC) để đánh giá xu hướng và hành vi thị trường phái sinh.

2.3.5 Lấy dữ liệu Fear & Greed Index

Hàm **fetch_fear_greed()** thu thập **chỉ số cảm xúc thị trường** (Fear & Greed Index) từ API công khai của [Alternative.me](#).

Chỉ số này thể hiện **mức độ sợ hãi hay tham lam** của thị trường crypto — một chỉ báo tâm lý quan trọng thường dùng cùng với biến động giá để xác định xu hướng.

```
def fetch_fear_greed(limit=0):
    url = "https://api.alternative.me/fng/"
    params = {"limit": limit, "format": "json"} # không dùng date_format vì không cần thiết
    try:
        r = requests.get(url, params=params, timeout=15)
        r.raise_for_status()
        j = r.json()
        data = j.get("data", [])
    except Exception as e:
        print(f"[WARN] Fear & Greed API failed: {e}")
    return pd.DataFrame(columns=["Date", "fetch_fear_greed"])
if not data:
    print("[WARN] Fear & Greed API returned empty dataset, skipping...")
    return pd.DataFrame(columns=["Date", "fetch_fear_greed"])
rows = []
for entry in data:
    rows.append([entry["date"], entry["fng"]])
df = pd.DataFrame(rows, columns=["Date", "fetch_fear_greed"])
df.set_index("Date", inplace=True)
```

```

try:

    ts = int(entry["timestamp"])

    value = float(entry["value"])

    date = datetime.fromtimestamp(ts, tz=timezone.utc).date()

    rows.append({"Date": date, "fetch_fear_greed": value})

except Exception as e:

    print(f"[WARN] parse error: {e}")

    continue

df = pd.DataFrame(rows)

df = df.drop_duplicates("Date").sort_values("Date").reset_index(drop=True)

print(f"✓ Fear & Greed fetched: {len(df)} rows")

print(df.head())

return df

```

👉 Gửi request đến **endpoint chính thức** của API Fear & Greed.

- **limit**: số lượng bản ghi muốn lấy (0 = toàn bộ).
- Nếu lỗi mạng hoặc API fail → trả về DataFrame rỗng để không làm hỏng pipeline.

Nếu rỗng, in cảnh báo

```

if not data:

    print("[WARN] Fear & Greed API returned empty dataset, skipping...")

    return pd.DataFrame(columns=["Date", "fetch_fear_greed"])

```

- Check trường hợp API trả về rỗng — thường xảy ra khi giới hạn tốc độ (rate limit) hoặc server bảo trì.

```
rows = []
```

```

for entry in data:

    try:

        ts = int(entry["timestamp"])

        value = float(entry["value"])

        date = datetime.fromtimestamp(ts, tz=timezone.utc).date()

        rows.append({"Date": date, "fetch_fear_greed": value})

    except Exception as e:

        print(f"[WARN] parse error: {e}")

        continue

df = pd.DataFrame(rows)

df = df.drop_duplicates("Date").sort_values("Date").reset_index(drop=True)

print(f"✓ Fear & Greed fetched: {len(df)} rows")

print(df.head())

return df

```

Vòng lặp duyệt từng bản ghi, chuyển:

- **timestamp (s) → ngày UTC;**
- **value (chuỗi) → float.**
Tạo danh sách dictionary cho từng ngày gồm **Date** và **fetch_fear_greed**.

```

df = pd.DataFrame(rows)

df = df.drop_duplicates("Date").sort_values("Date").reset_index(drop=True)

print(f"✓ Fear & Greed fetched: {len(df)} rows")

print(df.head())

return df

```

- Gộp toàn bộ dữ liệu vào **pandas.DataFrame**.
- Loại trùng ngày, sắp xếp tăng dần thời gian.
- Trả về DataFrame cuối cùng gồm 2 cột:

- **Date** – ngày UTC;
- **fetch_fear_greed** – điểm cảm xúc (0 → 100).

2.3.6 Lấy dữ liệu On-chain metrics

Các chỉ số on-chain được thu thập thông qua **Blockchain.info API**, cung cấp dữ liệu lịch sử liên quan đến hoạt động mạng lưới Bitcoin như:

- **Hash rate**: tổng công suất tính toán toàn mạng (đo sức mạnh bảo mật).
- **Transaction count (tx_count)**: số lượng giao dịch mỗi ngày.
- **Miners' revenue**: doanh thu hằng ngày của thợ mỏ (USD).

Định nghĩa các biểu đồ cần lấy

```
BLOCKCHAIN_CHARTS = {
    "hash_rate": "hash-rate",
    "tx_count": "n-transactions",
    "miners_revenue": "miners-revenue"
}
```

- Danh sách mapping giữa tên cột nội bộ và endpoint tương ứng của Blockchain.info.
Cột **active_addresses** bị loại bỏ vì API gốc hiện không còn hỗ trợ (**404 Not Found**).

Hàm tải dữ liệu từng biểu đồ riêng

```
def fetch_blockchain_chart(chart_name, timespan_days=4000):
    url = f"https://api.blockchain.info/charts/{chart_name}"
    params = {"timespan": f"{timespan_days}days", "format": "json", "sampled": False}
    r = requests.get(url, params=params, timeout=30)
    if r.status_code != 200:
        r = requests.get(url, params={"timespan": f"{timespan_days}days", "format": "json"}, timeout=30)
```

```

if r.status_code != 200:
    raise RuntimeError(f"Blockchain.info chart {chart_name} returned {r.status_code}")

j = r.json()

values = j.get("values", [])
rows = [{"Date": datetime.fromtimestamp(v["x"], tz=timezone.utc).date(), chart_name: v["y"]} for v in values]

return pd.DataFrame(rows).drop_duplicates("Date").sort_values("Date").reset_index(drop=True)

def fetch_onchain_from_blockchain_info(mapping):
    dfs = []

    for outcol, chart in mapping.items():
        try:
            dfc = fetch_blockchain_chart(chart, timespan_days=5000)
            dfc = dfc.rename(columns={chart: outcol})
            dfs.append(dfc)
            time.sleep(0.2)
        except Exception as e:
            print(f"[WARN] Could not fetch chart {chart}: {e}")

    if not dfs:
        return pd.DataFrame(columns=["Date"])

    df_merged = dfs[0]
    for d in dfs[1:]:
        df_merged = pd.merge(df_merged, d, on="Date", how="outer")

    return df_merged.sort_values("Date").reset_index(drop=True)

```

Hàm `fetch_blockchain_chart()` gửi **request GET** đến endpoint của Blockchain.info:

- URL ví dụ:
<https://api.blockchain.info/charts/hash-rate?timespan=4000days&format=json>.
- Dữ liệu trả về gồm danh sách timestamp (**x**) và giá trị (**y**).
- Mỗi bản ghi được chuyển sang dạng DataFrame với hai cột: **Date** và giá trị chỉ số.
- Có retry fallback nếu API đầu tiên trả lỗi, đảm bảo pipeline không bị ngắt.

Hàm hợp nhất toàn bộ chỉ số on-chain

```
def fetch_onchain_from_blockchain_info(mapping):
    dfs = []

    for outcol, chart in mapping.items():
        try:
            dfc = fetch_blockchain_chart(chart, timespan_days=5000)

            dfc = dfc.rename(columns={chart: outcol})

            dfs.append(dfc)

            time.sleep(0.2)
        except Exception as e:
            print(f"[WARN] Could not fetch chart {chart}: {e}")

    if not dfs:
        return pd.DataFrame(columns=["Date"])

    df_merged = dfs[0]

    for d in dfs[1:]:
        df_merged = pd.merge(df_merged, d, on="Date", how="outer")

    return df_merged.sort_values("Date").reset_index(drop=True)
```

Hàm này lần lượt:

- Gọi `fetch_blockchain_chart()` cho từng metric trong **BLOCKCHAIN_CHARTS**.

- Đổi tên cột trùng khớp với chuẩn của project (`hash_rate`, `tx_count`, `miners_revenue`).
- Ghép tất cả các DataFrame lại với nhau theo cột `Date` (outer join) để đảm bảo không mất dữ liệu.
- Thêm `sleep(0.2)` để tránh bị giới hạn request của server.

2.3.7. Tổng hợp dữ liệu hoàn chỉnh cho từng đồng (Build Dataset)

```
def build_dataset():

    start_ts = date_to_millis(START_DATE)

    end_dt = datetime.fromisoformat(END_DATE)

    end_of_day = datetime(end_dt.year, end_dt.month, end_dt.day, 23, 59, 59,
tzinfo=timezone.utc)

    end_ts = int(end_of_day.timestamp() * 1000)

    print("Fetching klines from Binance futures...")

    df_klines = fetch_binance_klines(SYMBOL_BINANCE, INTERVAL, start_ts, end_ts)

    print(f"klines rows: {len(df_klines)}")

    print("Fetching funding rates...")

    df_fund = fetch_binance_funding(SYMBOL_BINANCE, start_ts, end_ts)

    print(f"funding rows: {len(df_fund)}")

    print("Fetching Fear & Greed index...")

    df_fng = fetch_fear_greed(limit=0)

    print(f"fear&greed rows: {len(df_fng)}")

    print("Fetching on-chain metrics...")

    df_onchain = fetch_onchain_from_blockchain_info(BLOCKCHAIN_CHARTS)

    print(f"onchain rows: {len(df_onchain)})
```

```

# Merge các bảng

df = df_klines.copy()

df = pd.merge(df, df_fund, on="Date", how="left")

df = pd.merge(df, df_fng, on="Date", how="left")

df = pd.merge(df, df_onchain, on="Date", how="left")

# Bỏ sung cột thiêu

for col in ["funding_Rate", "fetch_fear_greed", "hash_rate",
            "tx_count", "miners_revenue", ]:

    if col not in df.columns:
        df[col] = pd.NA

df["time_frame"] = INTERVAL

# Thêm thêm cột cuối cùng

final_cols = ["Date", "Open", "High", "Low", "Close", "Volume", "symbol",
              "Quote_Volume", "Number_Of_Trades", "funding_Rate", "fetch_fear_greed",
              "hash_rate", "tx_count", "miners_revenue",
              "time_frame"]

df = df[[c for c in final_cols if c in df.columns]]

# Xuất file

start_tag = datetime.fromisoformat(START_DATE).strftime("%Y%m%d")

end_tag = end_dt.strftime("%Y%m%d")

out_csv = os.path.join(OUT_DIR, f"dataset_{SYMBOL_BASE}_{start_tag}_{end_tag}.csv")

out_xlsx = os.path.join(OUT_DIR, f"dataset_{SYMBOL_BASE}_{start_tag}_{end_tag}.xlsx")

df.to_csv(out_csv, index=False)

df.to_excel(out_xlsx, index=False)

print(f"Saved CSV: {out_csv}")

print(f"Saved XLSX: {out_xlsx}")

return df

```

Hàm `build_dataset()` là bước **tổng hợp toàn bộ dữ liệu của từng đồng coin** trong pipeline.

Cụ thể, nó lần lượt:

- Tạo timestamp dạng mili-giây để tương thích với các API.
- Gọi lần lượt các hàm thu thập dữ liệu:
 - `fetch_binance_klines()` → dữ liệu giá và khối lượng (OHLCV).
 - `fetch_binance_funding()` → Funding Rate trung bình theo ngày.
 - `fetch_fear_greed()` → chỉ số cảm xúc thị trường (Fear & Greed).
 - `fetch_onchain_from_blockchain_info()` → dữ liệu on-chain (`hash_rate`, `tx_count`, `miners_revenue`).
- Sau đó merge tất cả các bảng theo `Date` để tạo một DataFrame hoàn chỉnh, đảm bảo không mất dòng dữ liệu.
- Thêm các cột còn thiếu nếu chưa tồn tại, rồi ghi đè lại tên cột theo đúng cấu trúc chuẩn của toàn hệ thống.
- Cuối cùng, xuất file kết quả ra 2 định dạng `.csv` và `.xlsx` trong thư mục `data/<symbol>/`.
- Output của symbol BTC:

```

Fetching klines from Binance futures...
✓ Dữ liệu Binance (Futures) tải thành công: 1041 dòng
klines rows: 1041
Fetching funding rates...
funding rows: 1041
Fetching Fear & Greed index...
✓ Fear & Greed fetched: 2832 rows
   Date  fetch_fear_greed
0  2018-02-01      30.0
1  2018-02-02      15.0
2  2018-02-03      40.0
3  2018-02-04      24.0
4  2018-02-05      11.0
fear&greed rows: 2832
Fetching on-chain metrics...
onchain rows: 1666
Fetching Google Trends interest...
/usr/local/lib/python3.12/dist-packages/pytrends/request.py:260: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will change in
df = df.fillna(False)
✓ Lưu cache: data/BTC/google_trend_BTC.csv
✓ Saved CSV: data/BTC/dataset_BTC_20230101_20251106.csv
✓ Saved XLSX: data/BTC/dataset_BTC_20230101_20251106.xlsx

```

| Hoàn tất! Xem 10 dòng đầu: | | | | | | | | | | | | | | | | | | |
|----------------------------|------------|---------|---------|---------|---------|------------|--------|--------------|------------------|--------------|------------------|------------------|--------------|----------|----------------|-----------------|------------|--|
| | Date | Open | High | Low | Close | Volume | symbol | Quote_Volume | Number_Of_Trades | funding_Rate | fetch_fear_greed | active_addresses | hash_rate | tx_count | miners_revenue | google_interest | time_frame | |
| 0 | 2023-01-01 | 16537.5 | 16618.8 | 16488.0 | 16610.3 | 105502.965 | BTC | 1.746450e+09 | 594299 | 0.000069 | 26.0 | <NA> | 2.724834e+08 | 187516.0 | 1.617335e+07 | 28.0 | 1d | |
| 1 | 2023-01-02 | 16610.4 | 16799.0 | 16541.2 | 16666.0 | 215161.176 | BTC | 3.591354e+09 | 1020015 | -0.000011 | 27.0 | <NA> | NaN | NaN | NaN | NaN | 1d | |
| 2 | 2023-01-03 | 16665.9 | 16774.0 | 16600.3 | 16667.2 | 203070.205 | BTC | 3.387369e+09 | 966813 | 0.000074 | 26.0 | <NA> | NaN | NaN | NaN | NaN | 1d | |
| 3 | 2023-01-04 | 16667.3 | 16984.6 | 16645.7 | 16842.1 | 309747.838 | BTC | 5.088327e+09 | 1618311 | 0.000049 | 29.0 | <NA> | 2.423565e+08 | 277936.0 | 1.551442e+07 | NaN | 1d | |
| 4 | 2023-01-05 | 16842.2 | 16872.8 | 16740.4 | 16823.8 | 176369.347 | BTC | 2.966302e+09 | 892815 | 0.000057 | 29.0 | <NA> | NaN | NaN | NaN | NaN | 1d | |
| 5 | 2023-01-06 | 16823.8 | 17030.0 | 16664.8 | 16943.8 | 316973.629 | BTC | 5.333655e+09 | 1370628 | 0.000069 | 26.0 | <NA> | NaN | NaN | NaN | NaN | 1d | |
| 6 | 2023-01-07 | 16943.9 | 16973.6 | 16900.1 | 16936.5 | 77973.105 | BTC | 1.320073e+09 | 454642 | 0.000044 | 25.0 | <NA> | 2.609993e+08 | 246679.0 | 1.666991e+07 | NaN | 1d | |
| 7 | 2023-01-08 | 16936.5 | 17181.1 | 16905.0 | 17124.7 | 157717.318 | BTC | 2.677203e+09 | 775585 | 0.000077 | 25.0 | <NA> | NaN | NaN | NaN | 30.0 | 1d | |
| 8 | 2023-01-09 | 17124.6 | 17387.8 | 17098.2 | 17169.7 | 400277.809 | BTC | 6.099177e+09 | 1767631 | 0.000020 | 25.0 | <NA> | NaN | NaN | NaN | NaN | 1d | |
| 9 | 2023-01-10 | 17169.6 | 17485.4 | 17140.4 | 17428.8 | 297320.371 | BTC | 5.146869e+09 | 1362100 | 0.000051 | 26.0 | <NA> | 2.864213e+08 | 302809.0 | 1.867144e+07 | NaN | 1d | |



2.4. Gộp dữ liệu các mã coin

2.4.1. Gộp tất cả file _clean của từng coin

```
coin_list = ["BTC", "ETH", "BNB", "SOL", "XRP"]
```

```

def combine_clean_datasets(coin_list):

    all_dfs = []

    for coin in coin_list:

        data_dir = os.path.join("data", coin)

        clean_files = glob.glob(os.path.join(data_dir, f"dataset_{coin}_clean.csv"))

        if not clean_files:

            print(f"⚠️ Không tìm thấy file clean cho {coin}")

            continue

        latest_file = sorted(clean_files, key=os.path.getmtime)[-1]

        print(f"📁 Đang đọc: {os.path.basename(latest_file)}")

        df = pd.read_csv(latest_file)

        if "symbol" not in df.columns:

            df["symbol"] = coin

        all_dfs.append(df)

        print(f"✓ {coin}: {df.shape[0]} dòng, {df.shape[1]} cột")

    if not all_dfs:

        print("❌ Không có file nào để gộp.")

        return None

    df_total = pd.concat(all_dfs, ignore_index=True)

    df_total = df_total.sort_values(["symbol", "Date"]).reset_index(drop=True)

    print(f"\n📊 Tổng cộng {df_total.shape[0]} dòng từ {len(all_dfs)} đồng coin.")

    print(f"📈 Các đồng được gộp: {', '.join(coin_list)}")

    out_csv = os.path.join("data", "dataset_total_clean.csv")

    df_total.to_csv(out_csv, index=False)

    print(f"💾 Đã lưu file tổng:\n- {out_csv}")

    return df_total

```

- Vòng lặp đọc tất cả file `dataset_<symbol>_clean.csv` của từng đồng trong danh sách, gộp thành một DataFrame tổng, sắp xếp theo `symbol` và `Date`, rồi lưu ra `dataset_total_clean.csv`.

2.4.2. Đổi tên cột cho đồng nhất & dễ đọc

```
file_path = "data/dataset_total_clean.csv"

df = pd.read_csv(file_path)

print(f"📁 Đã đọc file: {file_path}")

print(f"🔧 Trước đổi tên: {df.shape[1]} cột")

#Đổi tên các cột

rename_map = {

    "fetch_fear_greed": "fear_greed_index",
    : "google_trend_score",
    "Close_norm": "close_norm",
    : "onchain_active_addresses"
}

# Chỉ đổi tên cột tồn tại

existing = [c for c in rename_map.keys() if c in df.columns]

df = df.rename(columns={k: rename_map[k] for k in existing})

print(f"Đã đổi tên {len(existing)} cột:")

for old, new in rename_map.items():

    if old in existing:

        print(f"    • {old} → {new}")

if 'onchain_active_addresses' in df.columns:

    df = df.drop(columns=['onchain_active_addresses'])


```

```

# Điền giá trị nhỏ còn thiếu
df['log_return'] = df['log_return'].ffill()
df['pct_change'] = df['pct_change'].ffill()

#Lưu lại

SAVE_OVERWRITE = True    # Đổi thành False nếu muốn lưu bản mới

if SAVE_OVERWRITE:

    out_csv = file_path

else:

    out_csv = "data/dataset_total_clean_renamed.csv"

df.to_csv(out_csv, index=False)

print(f"💾 Đã lưu lại:\n- {out_csv}")

print("\n📝 Danh sách cột sau khi đổi tên:")

print(df.columns.tolist()[:15], "...")
```

Đoạn code này đọc lại `dataset_total_clean.csv` vừa gộp, thực hiện các bước:

- Đổi tên cột cho thống nhất và thân thiện:
 - `fetch_fear_greed` → `fear_greed_index`
 - `google_interest` → `google_trend_score`
 - `Close_norm` → `close_norm`
 - `active_addresses` → `onchain_active_addresses`
- Loại bỏ `onchain_active_addresses` (vì dữ liệu thiếu ổn định).
- Điền lại các giá trị nhỏ còn trống trong `log_return`, `pct_change` bằng phương pháp `ffill()`.
- Lưu đè lại file gốc `dataset_total_clean.csv`.

2.5. Tiền xử lý dataset tổng

2.5.1. Đọc file dữ liệu tổng

```
import pandas as pd

df = pd.read_csv("data/dataset_total_clean.csv")
```

Đọc file tổng `dataset_total_clean.csv` sau khi đã gộp dữ liệu từ các coin riêng lẻ.

2.5.2. Ép kiểu dữ liệu và xử lý giá trị thiếu

```
# Ép kiểu sang số để ffill hoạt động đúng
df['pct_change'] = pd.to_numeric(df['pct_change'], errors='coerce')
df['log_return'] = pd.to_numeric(df['log_return'], errors='coerce')

# Điều chỉnh phần bị thiếu
df['pct_change'] = df['pct_change'].ffill()
df['log_return'] = df['log_return'].ffill()
```

- Dòng đầu ép cột `pct_change` và `log_return` sang dạng số (`float`) để tránh lỗi khi thực hiện fill.
- Sau đó sử dụng `ffill()` (forward fill) để **điền giá trị bị thiếu** bằng giá trị gần nhất phía trước — giữ được tính liên tục của chuỗi thời gian.
[kết quả: các giá trị `NaN` trong `pct_change` và `log_return` được thay thế hoàn toàn]

2.5.3. Ghi đè dữ liệu đã xử lý lên file cũ

```
out_csv = "data/dataset_total_clean.csv"

df.to_csv(out_csv, index=False)

print(f"✅ Đã fill giá trị và ghi đè lại file: {out_csv}")

print(df[['symbol', 'Date', 'pct_change', 'log_return']].head(10))
```

- Ghi lại dữ liệu đã xử lý (sau khi fill và ép kiểu) **đè lên file gốc `dataset_total_clean.csv`**.

- In ra 10 dòng đầu để kiểm tra nhanh các giá trị `pct_change` và `log_return`. [kết quả: file `dataset_total_clean.csv` được cập nhật hoàn toàn, hai cột `pct_change`, `log_return` không còn giá trị khuyết]

2.6. Scale dataset tổng

2.6.1. Xác định cột numeric cần scale

```
df = pd.read_csv("data/dataset_total_clean.csv")

print(f"📁 Đã đọc file tổng: {df.shape[0]} dòng, {df.shape[1]} cột")

num_cols = [
    'Open', 'High', 'Low', 'Close', 'Volume',
    'funding_Rate', 'hash_rate', 'tx_count',
    'miners_revenue', 'google_trend_score'
]

num_cols = [c for c in num_cols if c in df.columns]

print(f"⌚ Cột sẽ được scale: {num_cols}")
```

- Đọc dữ liệu tổng, chọn ra các cột định lượng cần chuẩn hóa.

2.6.2. Scale riêng từng đồng coin

```
scaled_dfs = []

for sym in df['symbol'].unique():

    subset = df[df['symbol'] == sym].copy()

    # Scale riêng từng coin
    scaler = StandardScaler()

    subset[num_cols] = scaler.fit_transform(subset[num_cols])

    # Tạo thư mục lưu
    coin_dir = f"data/{sym}"

    os.makedirs(coin_dir, exist_ok=True)

    # Đưa ng dẫn lưu
```

```

out_path = os.path.join(coin_dir, f"dataset_{sym}_scaled.csv")
subset.to_csv(out_path, index=False)

scaled_dfs.append(subset)

print(f"✓ {sym}: Đã scale & lưu → {out_path}")

```

- Chuẩn hóa dữ liệu từng coin bằng `StandardScaler()` (trung bình = 0, độ lệch chuẩn = 1), sau đó lưu file đã scale vào thư mục tương ứng.
[kết quả: sinh ra `dataset_BTC_scaled.csv`, `dataset_ETH_scaled.csv`, ... trong thư mục `data/<symbol>/`]

2.6.3. Gộp file đã scale

```

df_scaled_total = pd.concat(scaled_dfs, ignore_index=True)

out_total = "data/dataset_scaled_total.csv"

df_scaled_total.to_csv(out_total, index=False)

```

- Gộp toàn bộ các file đã scale riêng lẻ thành một file tổng để dùng cho mô hình học máy.

3. Data Analysis

3.0 Mục tiêu phân tích

3.0.1 Mục đích

Phân Data Analysis nhằm khai thác, mô tả và hiểu rõ đặc điểm biến động, mối quan hệ và các yếu tố ảnh hưởng đến giá của các đồng tiền mã hóa được chọn trong dự án, bao gồm: BTC, ETH, BNB, SOL và XRP.

Mục tiêu chính là:

- Xác định xu hướng và mức độ biến động của từng đồng coin trong giai đoạn nghiên cứu.
- Phân tích rủi ro thông qua các chỉ số như Volatility, Max Drawdown, Sharpe ratio, và Sortino ratio.
- Đánh giá mối quan hệ giữa giá và các yếu tố bên ngoài: Funding Rate, Fear & Greed Index, Google Search Trend, cùng với các chỉ số On-chain như Hash Rate, Transaction Count và Miner Revenue.
- So sánh sự tương quan và mức độ đồng pha giữa các coin trong thị trường để xem coin nào dẫn dắt xu hướng (market leader) và coin nào có hành vi độc lập

hơn (idiosyncratic behavior).

- Chuẩn bị nền tảng dữ liệu và insight cho bước tiếp theo — xây dựng mô hình dự báo giá (Prediction Model) trong phần sau.

3.0.2 Phương pháp tổng quát

Quy trình phân tích được chia thành 6 phần chính:

1. Load & Setup: đọc dữ liệu, chuẩn hóa cấu trúc, tạo biến log-return và %change.
2. Descriptive Statistics: thống kê mô tả các đặc trưng cơ bản của từng coin.
3. Volatility & Risk: đo lường mức độ biến động và rủi ro đầu tư.
4. Correlation Analysis: xác định yếu tố ảnh hưởng đến biến động giá.
5. Linear Relationship Test: kiểm định mối quan hệ tuyến tính giữa returns và các yếu tố vĩ mô.
6. Cross-Market Correlation: phân tích mức độ liên kết giữa các coin.

3.0.3 Kết quả mong đợi

- Làm rõ đặc trưng biến động riêng biệt của từng đồng coin.
- Xác định những yếu tố có ảnh hưởng đáng kể nhất tới sự thay đổi giá.
- Đưa ra các đánh giá định lượng về rủi ro và mối quan hệ giữa các đồng coin.
- Cung cấp cơ sở dữ liệu vững chắc cho các bước Data Visualization nâng cao và Model Building trong giai đoạn kế tiếp của dự án.

3.1 Load and setup

3.1.1. Phương pháp

Dữ liệu được xử lý bằng thư viện **pandas** trong môi trường **Google Colab**. Các bước thực hiện:

```

df = pd.read_csv(DATA_PATH)

df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

df = df.sort_values(['symbol', 'Date']).reset_index(drop=True)

```

- Dòng 1: Đọc dữ liệu gốc từ file **dataset_total_clean.csv**.
- Dòng 2: Chuyển cột **Date** sang kiểu **datetime** để phục vụ phân tích chuỗi thời gian.
- Dòng 3: Sắp xếp dữ liệu theo thứ tự **symbol** và **Date**, đảm bảo tính liên tục khi tính các chỉ số biến động.

Tạo thêm hai biến đo lường mức thay đổi giá:

```

# Đảm bảo tồn tại các cột returns; nếu chưa có thì tạo

if 'pct_change' not in df.columns:

    df['pct_change'] = df.groupby('symbol')['Close'].pct_change() * 100.0

if 'log_return' not in df.columns:

    df['log_return'] = np.log(df.groupby('symbol')['Close'].apply(lambda s: s /
s.shift(1))).values

```

- **pct_change**: biểu thị **tỷ lệ phần trăm thay đổi giá** giữa hai ngày liên tiếp.
- **log_return**: thể hiện **tỷ suất sinh lời logarit**, thường được sử dụng trong phân tích tài chính do có tính ổn định và đối xứng cao.

3.1.2. Kết quả

Sau khi xử lý, dữ liệu được chuẩn hóa với cấu trúc như sau:

Đặc điểm dữ liệu sau chuẩn hóa:

- Bao gồm 6 đồng coin chính: BTC, ETH, BNB, SOL, XRP, v.v.
- Mỗi đồng được mô tả theo chuỗi thời gian ngày (**Date**) liên tục.
- Tổng cộng khoảng **12 cột dữ liệu**, chia thành các nhóm:
 - **Giá và biến động**: Close, pct_change, log_return

- **Hoạt động giao dịch:** Volume, funding_Rate
- **Tâm lý thị trường:** fetch_fear_greed, google_interest
- **Yếu tố on-chain:** hash_rate, tx_count, miners_revenue

3.1.3. Nhận xét

- Dữ liệu sau tiền xử lý đảm bảo **tính thống nhất về thời gian và đơn vị đo lường** giữa các nguồn khác nhau.
- Hai biến mới (**pct_change** và **log_return**) cung cấp **thước đo biến động tiêu chuẩn** để đánh giá rủi ro và tương quan trong các phân tích tiếp theo.

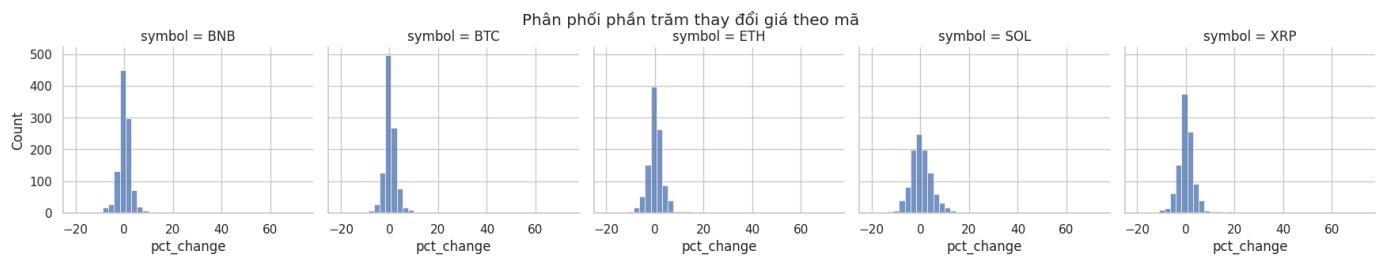
3.2. Thống kê mô tả (Descriptive Statistics)

3.2.1. Phương pháp

- Dữ liệu được nhóm theo **symbol** (BNB, BTC, ETH, SOL, XRP) và sắp xếp theo **Date**.
- Biến phân tích chính:
 - **pct_change**: % thay đổi giá ngày-quá/ngày.
 - **log_return**: lợi suất logarit ngày.
- Trình bày kết quả qua:
 - **Histogram** của **pct_change** cho từng mã để quan sát hình dạng phân phối và độ rộng biến động (Hình 3.2a).
 - **Boxplot** của **log_return** theo từng mã để so sánh độ phân tán (IQR), outlier và độ lệch (Hình 3.2b).
- Bảng thống kê mô tả chi tiết (mean, std, min, max, ...) được tổng hợp trong phụ lục “Summary Statistics by symbol”.

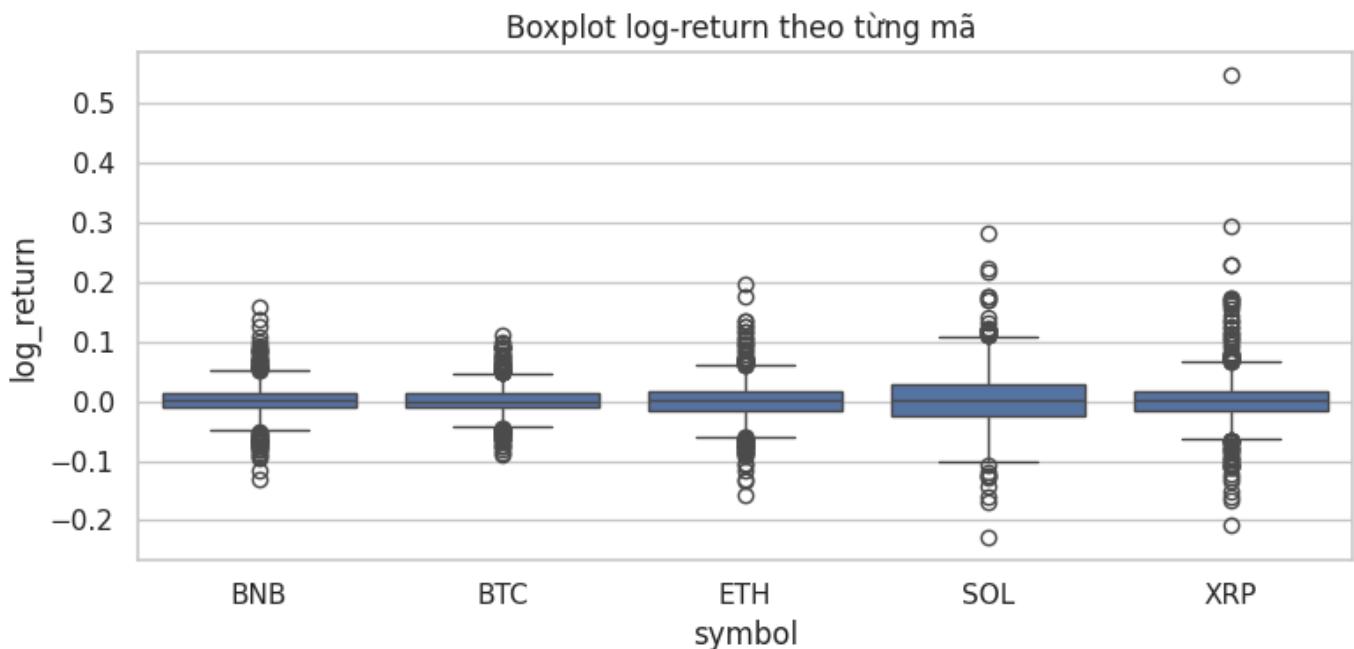
3.2.2. Kết quả

Hình 3.2a — Histogram pct_change theo mã:



- Tất cả các mã đều có phân phối tập trung quanh 0% (biến động ngày đa số nhỏ), đuôi hai phía thể hiện những phiên tăng/giảm mạnh.
- **SOL** có dải phân phối rộng nhất (đuôi kéo ra xa hai phía), cho thấy biến độ dao động ngày lớn hơn các mã còn lại.
- **XRP** cũng thể hiện dải đuôi dày hơn BNB/BTC/ETH, xuất hiện nhiều phiên biến động mạnh hơn mức trung vị.
- **BTC** và **BNB** có phân phối hẹp nhất, cột giữa cao, phản ánh biến động ngày tương đối nhỏ và ổn định hơn.

Hình 3.2b — Boxplot log_return theo mã:



- **SOL** có **IQR** (khoảng từ phân vị) lớn nhất và mật độ **outlier** dày đặc ở cả hai phía → rủi ro biến động cao.
- **XRP** xuất hiện một số **outlier dương rất lớn** (điểm nhô cao), phản ánh những phiên tăng đột biến.

- **ETH** có IQR trung bình, nhiều outlier hơn BTC/BNB nhưng thấp hơn SOL/XRP.
- **BTC và BNB** có **IQR** hẹp nhất, ít outlier hơn → biến động ngày nhìn chung thấp và ổn định.

3.2.3. Nhận xét

- **Mức độ biến động tương đối** (từ thấp đến cao) quan sát từ histogram & boxplot:
BTC ≈ BNB < ETH < XRP < SOL.
Điều này phù hợp với trực giác thị trường: BTC/BNB thường ổn định hơn, trong khi SOL/XRP nhạy hơn với dòng tiền và tin tức.
- **Phân phối returns** có **đuôi dày (fat-tail)** ở tất cả các mã: nhiều phiên biến động mạnh hơn kỳ vọng của phân phối chuẩn. Đây là đặc điểm điển hình của crypto, hàm ý cần thận trọng khi dùng giả định phân phối chuẩn trong mô hình rủi ro.
- **Outlier lớn** (đặc biệt ở SOL và XRP) gợi ý các giai đoạn sự kiện/đột biến; các điểm này sẽ được quay lại trong phần *Anomaly/Event Study*.

3.3. Phân tích biến động và rủi ro

3.3.1. Phương pháp

Phân tích được thực hiện bằng Python (pandas, numpy) với các bước chính:

```
# --- Hàm tính Max Drawdown ---
def max_drawdown(price_series: pd.Series):
    peak = price_series.cummax()
    dd = price_series/peak - 1.0
    return dd.min(), dd.idxmin()

# --- Tính Sharpe/Sortino cơ bản (rf ~ 0) ---
def sharpe_ratio(returns, rf=0.0):
    # returns là log_return theo ngày
    mu = returns.mean()
    sd = returns.std(ddof=0)

    return np.nan if sd == 0 or np.isnan(sd) else (mu - rf) / sd * np.sqrt(365)
```

```

def sortino_ratio(returns, rf=0.0):

    downside = returns[returns < 0]

    dd = downside.std(ddof=0)

    mu = returns.mean()

    return np.nan if dd == 0 or np.isnan(dd) else (mu - rf) / dd * np.sqrt(365)

# --- Tổng hợp risk table ---

rows = []

for sym, g in df.groupby('symbol', sort=False):

    g = g.dropna(subset=['Close', 'log_return'])

    mdd, mdd_idx = max_drawdown(g['Close'])

    sr = sharpe_ratio(g['log_return'])

    sor = sortino_ratio(g['log_return'])

    rows.append({
        'symbol': sym,
        'max_drawdown': mdd,
        'max_drawdown_date': g.loc[mdd_idx, 'Date'] if pd.notna(mdd_idx) else pd.NaT,
        'sharpe_annual': sr,
        'sortino_annual': sor
    })
}

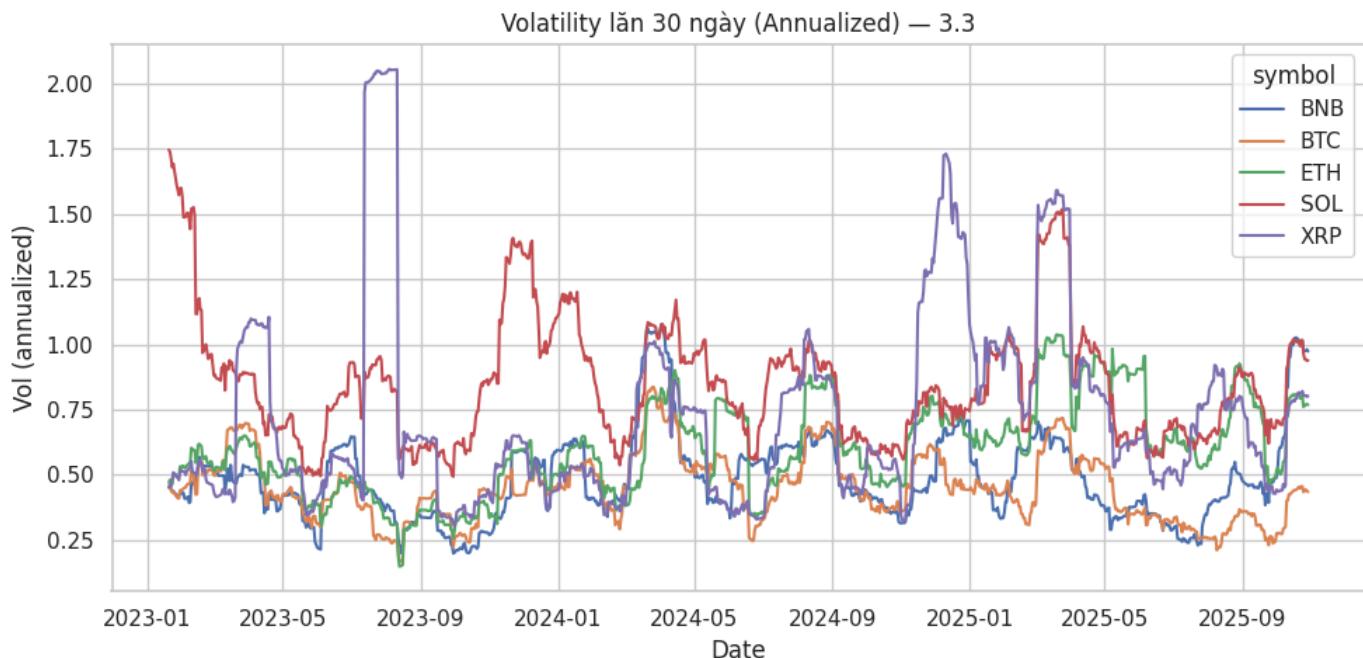
```

Các chỉ số chính:

- **Volatility:** đo độ biến động bình quân của **log-return** (chuẩn hóa theo năm).
- **Max Drawdown:** phần trăm giảm tối đa từ đỉnh đến đáy gần nhất.
- **Sharpe Ratio:** lợi nhuận điều chỉnh theo tổng rủi ro.
- **Sortino Ratio:** lợi nhuận điều chỉnh theo rủi ro giảm giá (downside risk).

3.3.2. Kết quả

(a) Volatility 30 ngày – Annualized



- Biểu đồ cho thấy độ biến động của 5 mã crypto thay đổi mạnh theo thời gian:
- SOL** và **XRP** có giai đoạn biến động cao nhất (đỉnh > 2.0).
- BTC** và **BNB** duy trì biến động thấp, ổn định quanh mức 0.5–1.0.
- ETH** nằm giữa, thể hiện tính trung gian giữa nhóm **altcoin** và nhóm **large-cap**.

(b) Max Drawdown (MDD)

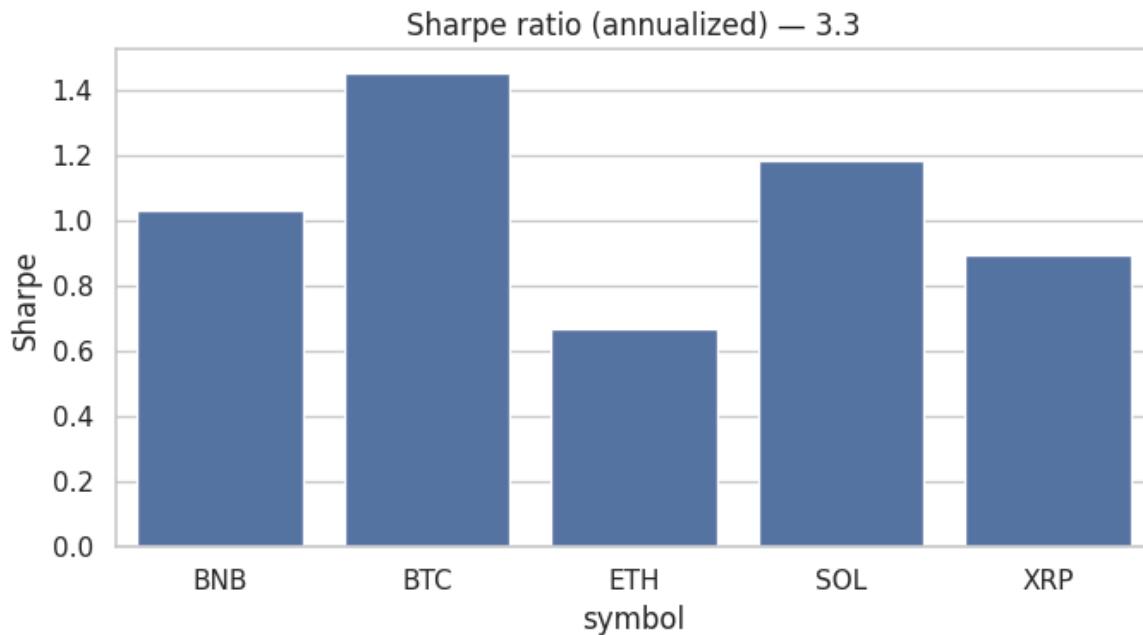
3.3 – Risk metrics (Max Drawdown, Sharpe, Sortino)

| | symbol | max_drawdown | max_drawdown_date | sharpe_annual | sortino_annual |
|---|--------|--------------|-------------------|---------------|----------------|
| 0 | BNB | -0.410124 | 2023-10-12 | 1.032204 | 1.471052 |
| 1 | BTC | -0.280953 | 2025-04-08 | 1.451381 | 2.227076 |
| 2 | ETH | -0.637520 | 2025-04-08 | 0.670997 | 0.985117 |
| 3 | SOL | -0.597664 | 2025-04-08 | 1.184301 | 1.997868 |
| 4 | XRP | -0.488352 | 2024-07-07 | 0.892754 | 1.442973 |

- BTC** thấp nhất (-28%), ổn định nhất nhóm.

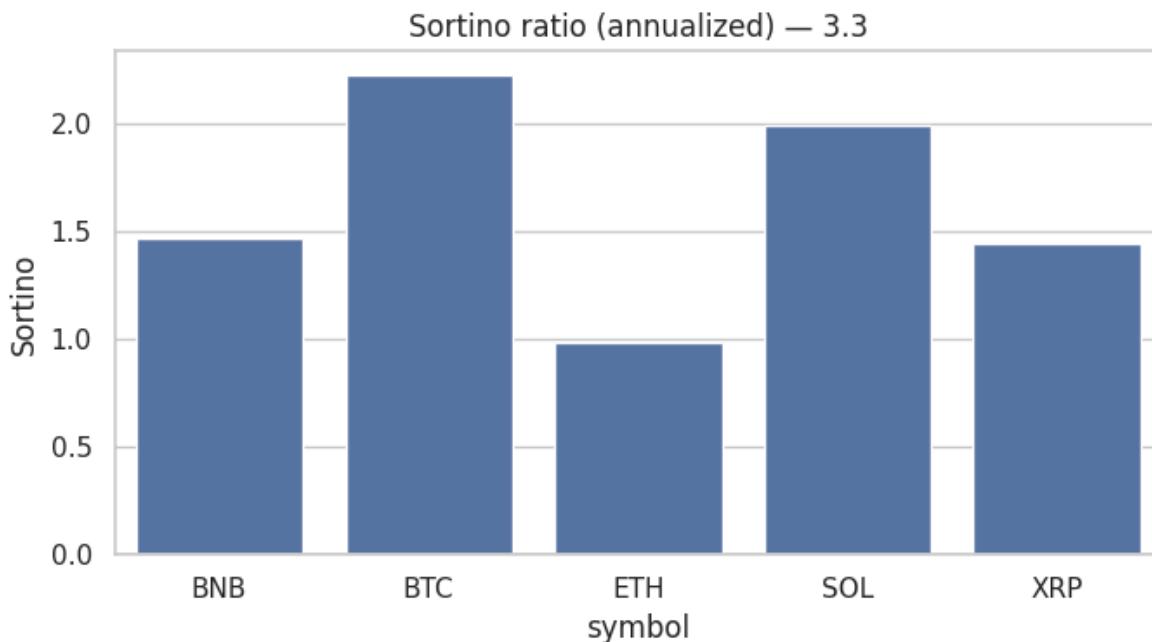
- Nhiều mã đạt đáy cùng ngày (2025-04-08) → dấu hiệu của điều chỉnh toàn thị trường.

(c) Sharpe Ratio (Annualized)



- BTC** có **Sharpe** cao nhất (1.45) → hiệu quả rủi ro tốt nhất.
- SOL** đạt 1.18, vẫn khá cao dù biến động mạnh.
- ETH** thấp nhất (0.67), rủi ro cao hơn lợi nhuận đạt được.

(d) Sortino Ratio (Annualized)



- o **BTC** (2.22) và **SOL** (2.00) có Sortino cao, chứng tỏ phần lớn biến động là hướng tăng.
- o **ETH** (0.98) và **XRP** (1.44) có mức Sortino thấp hơn → chịu nhiều rủi ro giảm giá.
- o **BNB** (1.47) nằm ở mức trung bình khá.

3.3.3. Nhận xét

- o **BTC** → tài sản ổn định nhất, hiệu quả rủi ro cao nhất.
- o **SOL** → biến động mạnh nhưng lợi nhuận bù rủi ro tốt (đặc trưng của coin tăng trưởng).
- o **ETH** → hiệu suất thấp nhất nhóm, drawdown sâu và Sharpe kém.
- o **BNB, XRP** → hiệu quả rủi ro trung bình.
- o Biểu đồ **volatility** và **drawdown** đều cho thấy chu kỳ biến động rõ rệt (2023–2025), gắn với biến động chung của thị trường crypto.

3.4. Phân tích mối quan hệ giữa các biến (Correlation Analysis)

3.4.1. Correlation Matrix

- **pct_change ↔ log_return ~ 1.00** ở tất cả các mã → hai thước đo biến động giá gần tương đương.
- **hash_rate ↔ miners_revenue ≈ 0.65 (dương mạnh, ổn định)** trên BTC/ETH/BNB/SOL/XRP → phản ánh sức khỏe mạng lưới.
- **tx_count ↔ hash_rate ≈ 0.18–0.19** (dương nhẹ) → hoạt động on-chain tăng thường đi kèm áp lực hash cao hơn, nhưng không mạnh.
- **Volume ↔ hash_rate (BTC ≈ -0.26; coin khác ≈ 0)** → ở BTC có pha “nghịch” nhẹ theo chu kỳ khai thác/thị trường.
- **Giá/returns ↔ on-chain (hash_rate/tx_count/miners_revenue)**: r nhỏ (~0.0–0.2) → quan hệ tuyến tính tức thời yếu; khả năng có độ trễ theo thời gian.
- **Lưu ý dữ liệu**: ma trận đã loại NaN theo hàng; với mẫu nhỏ, r có thể kém ổn định → nên kèm kích thước mẫu khi báo cáo.

```

corr_features = [
    'pct_change', 'log_return', 'funding_Rate', 'fetch_fear_greed',
    'hash_rate', 'tx_count', 'miners_revenue', 'Volume'
]

for sym in df['symbol'].unique():
    sub = df[df['symbol'] == sym][corr_features].dropna(how='all')
    # Nếu quá nhiều NaN → bỏ các cột toàn NaN
    usable = sub.dropna(axis=1, how='all')
    if usable.shape[1] < 2:
        print(f"⚠ {sym}: Không đủ dữ liệu để tính tương quan meaningful.")
        continue
    corr = usable.corr().round(2)
    print(f"\n3.4 – Correlation matrix ({sym})")
    display(corr)

    plt.figure(figsize=(6, 5))
    sns.heatmap(corr, annot=True, vmin=-1, vmax=1, cmap='coolwarm')
    plt.title(f'Heatmap tương quan – {sym} (3.4)')
    plt.show()

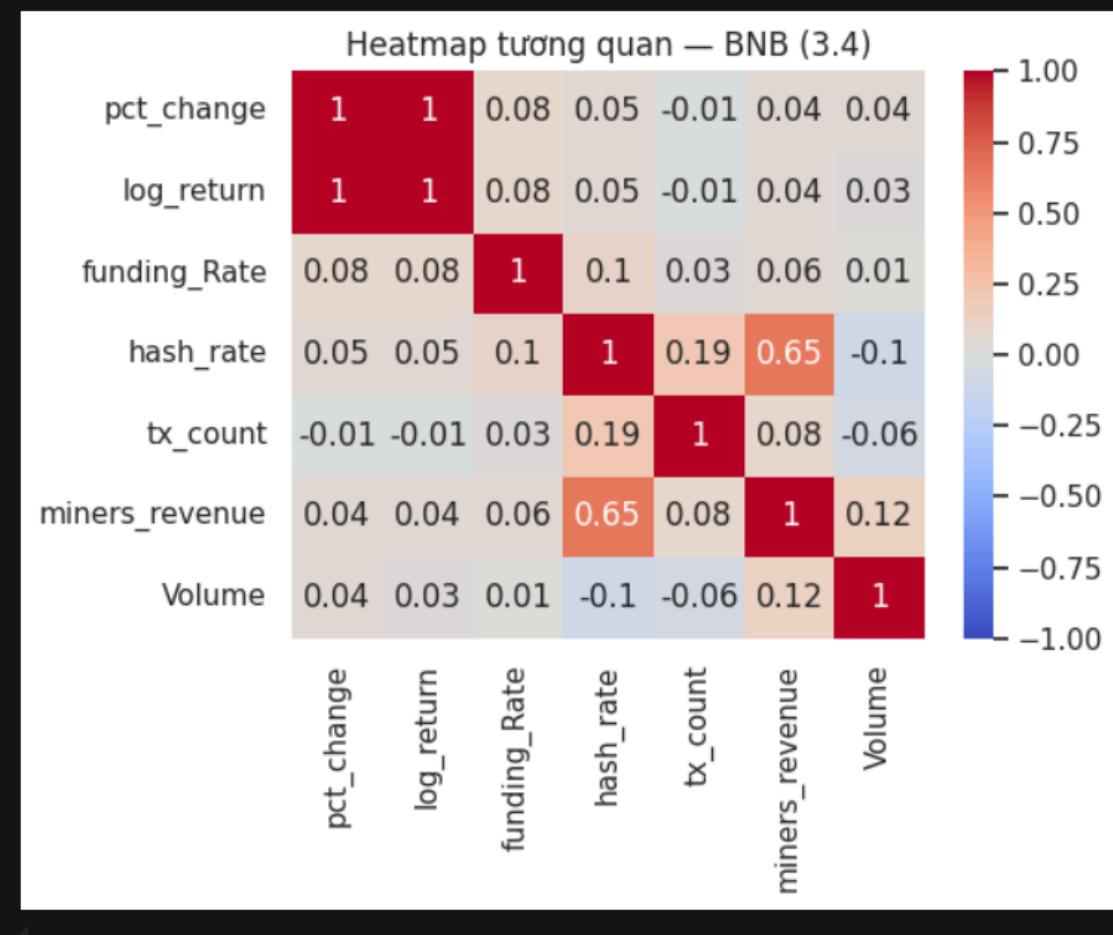
```

3.4.2. Kết quả

Hình 3.4a — Ma trận tương quan & Heatmap (BNB)

🔥 3.4 – Correlation matrix (BNB)

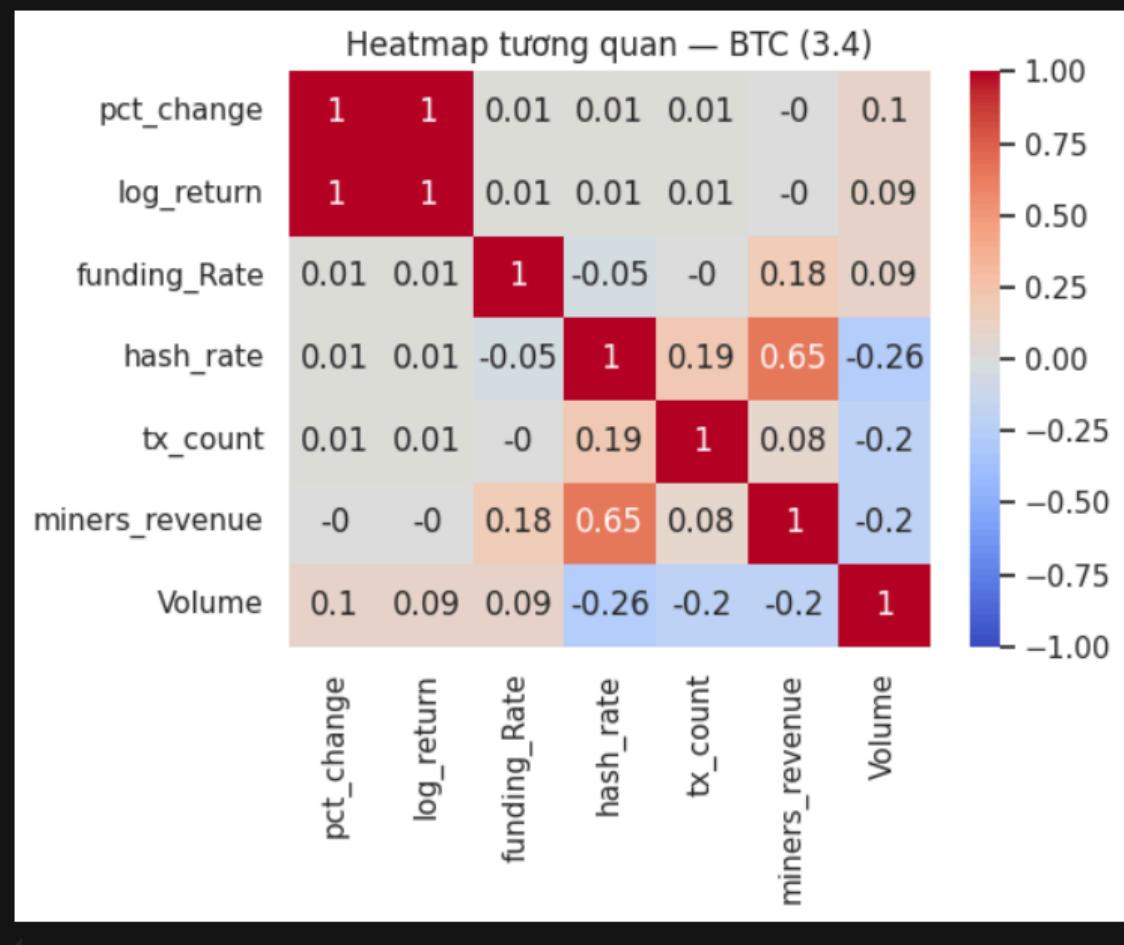
| | pct_change | log_return | funding_Rate | hash_rate | tx_count | miners_revenue | Volume |
|----------------|------------|------------|--------------|-----------|----------|----------------|--------|
| pct_change | 1.00 | 1.00 | 0.08 | 0.05 | -0.01 | 0.04 | 0.04 |
| log_return | 1.00 | 1.00 | 0.08 | 0.05 | -0.01 | 0.04 | 0.03 |
| funding_Rate | 0.08 | 0.08 | 1.00 | 0.10 | 0.03 | 0.06 | 0.01 |
| hash_rate | 0.05 | 0.05 | 0.10 | 1.00 | 0.19 | 0.65 | -0.10 |
| tx_count | -0.01 | -0.01 | 0.03 | 0.19 | 1.00 | 0.08 | -0.06 |
| miners_revenue | 0.04 | 0.04 | 0.06 | 0.65 | 0.08 | 1.00 | 0.12 |
| Volume | 0.04 | 0.03 | 0.01 | -0.10 | -0.06 | 0.12 | 1.00 |



Hình 3.4b — Ma trận tương quan & Heatmap(BTC)

3.4 – Correlation matrix (BTC)

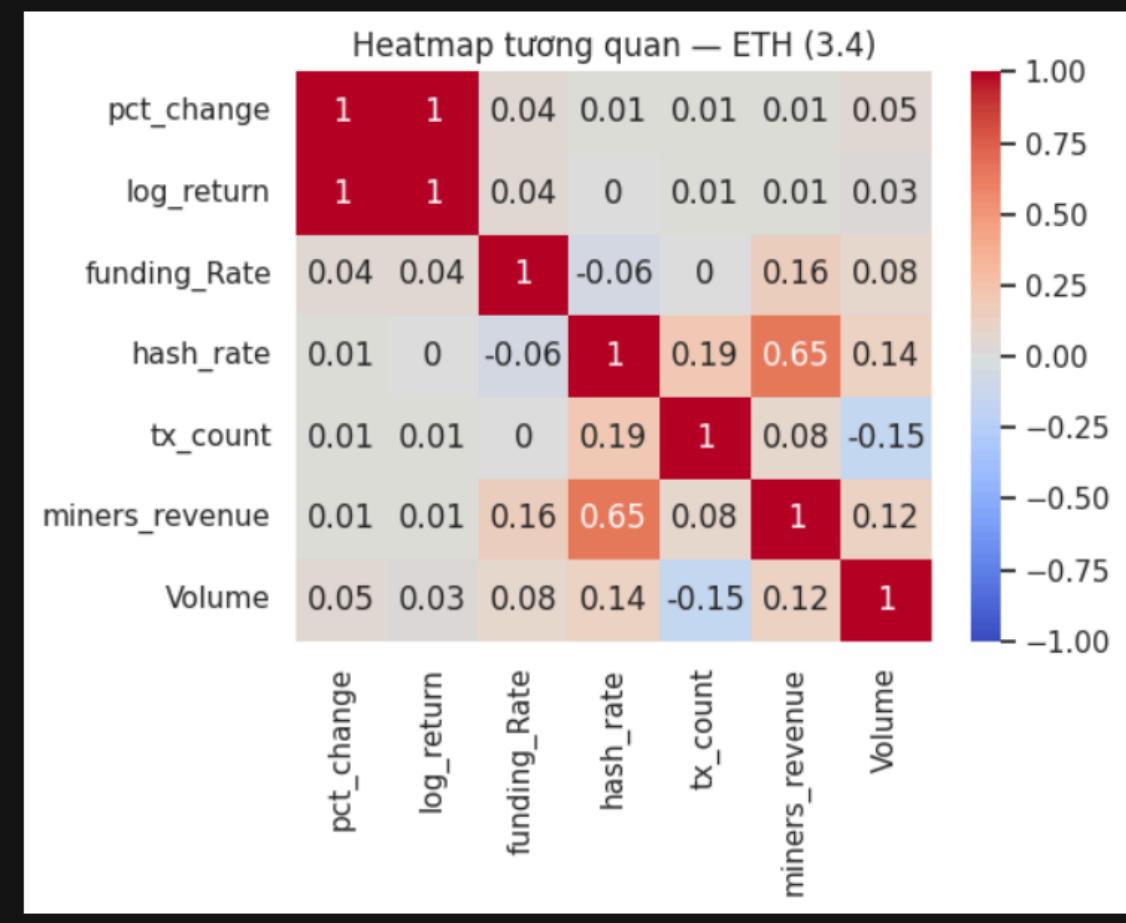
| | pct_change | log_return | funding_Rate | hash_rate | tx_count | miners_revenue | Volume |
|----------------|------------|------------|--------------|-----------|----------|----------------|--------|
| pct_change | 1.00 | 1.00 | 0.01 | 0.01 | 0.01 | -0.00 | 0.10 |
| log_return | 1.00 | 1.00 | 0.01 | 0.01 | 0.01 | -0.00 | 0.09 |
| funding_Rate | 0.01 | 0.01 | 1.00 | -0.05 | -0.00 | 0.18 | 0.09 |
| hash_rate | 0.01 | 0.01 | -0.05 | 1.00 | 0.19 | 0.65 | -0.26 |
| tx_count | 0.01 | 0.01 | -0.00 | 0.19 | 1.00 | 0.08 | -0.20 |
| miners_revenue | -0.00 | -0.00 | 0.18 | 0.65 | 0.08 | 1.00 | -0.20 |
| Volume | 0.10 | 0.09 | 0.09 | -0.26 | -0.20 | -0.20 | 1.00 |



Hình 3.4c — Ma trận tương quan & Heatmap(ETH)

🔥 3.4 – Correlation matrix (ETH)

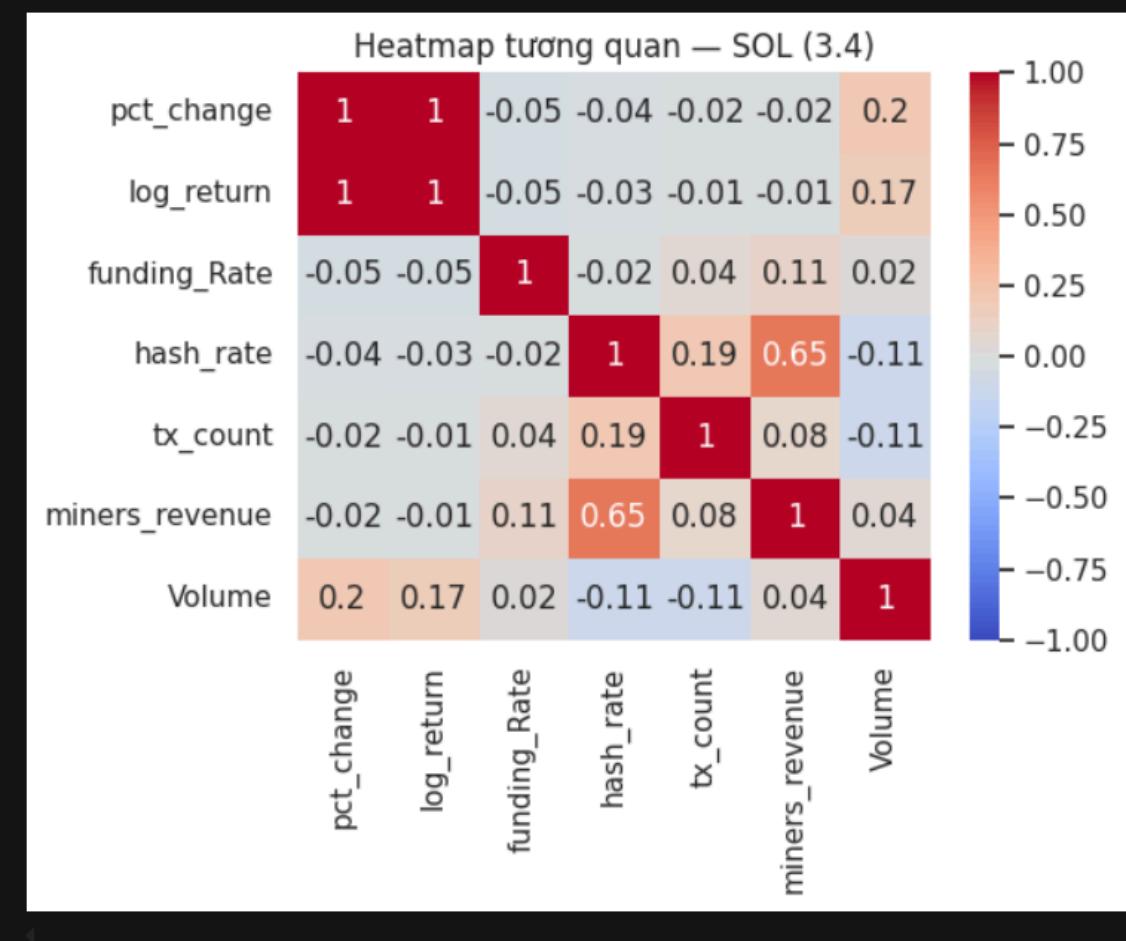
| | pct_change | log_return | funding_Rate | hash_rate | tx_count | miners_revenue | Volume |
|----------------|------------|------------|--------------|-----------|----------|----------------|--------|
| pct_change | 1.00 | 1.00 | 0.04 | 0.01 | 0.01 | 0.01 | 0.05 |
| log_return | 1.00 | 1.00 | 0.04 | 0.00 | 0.01 | 0.01 | 0.03 |
| funding_Rate | 0.04 | 0.04 | 1.00 | -0.06 | 0.00 | 0.00 | 0.16 |
| hash_rate | 0.01 | 0.00 | -0.06 | 1.00 | 0.19 | 0.65 | 0.14 |
| tx_count | 0.01 | 0.01 | 0.00 | 0.19 | 1.00 | 0.08 | -0.15 |
| miners_revenue | 0.01 | 0.01 | 0.16 | 0.65 | 0.08 | 1.00 | 0.12 |
| Volume | 0.05 | 0.03 | 0.08 | 0.14 | -0.15 | 0.12 | 1.00 |



Hình 3.4d — Ma trận tương quan & Heatmap(SOL)

🔥 3.4 – Correlation matrix (SOL)

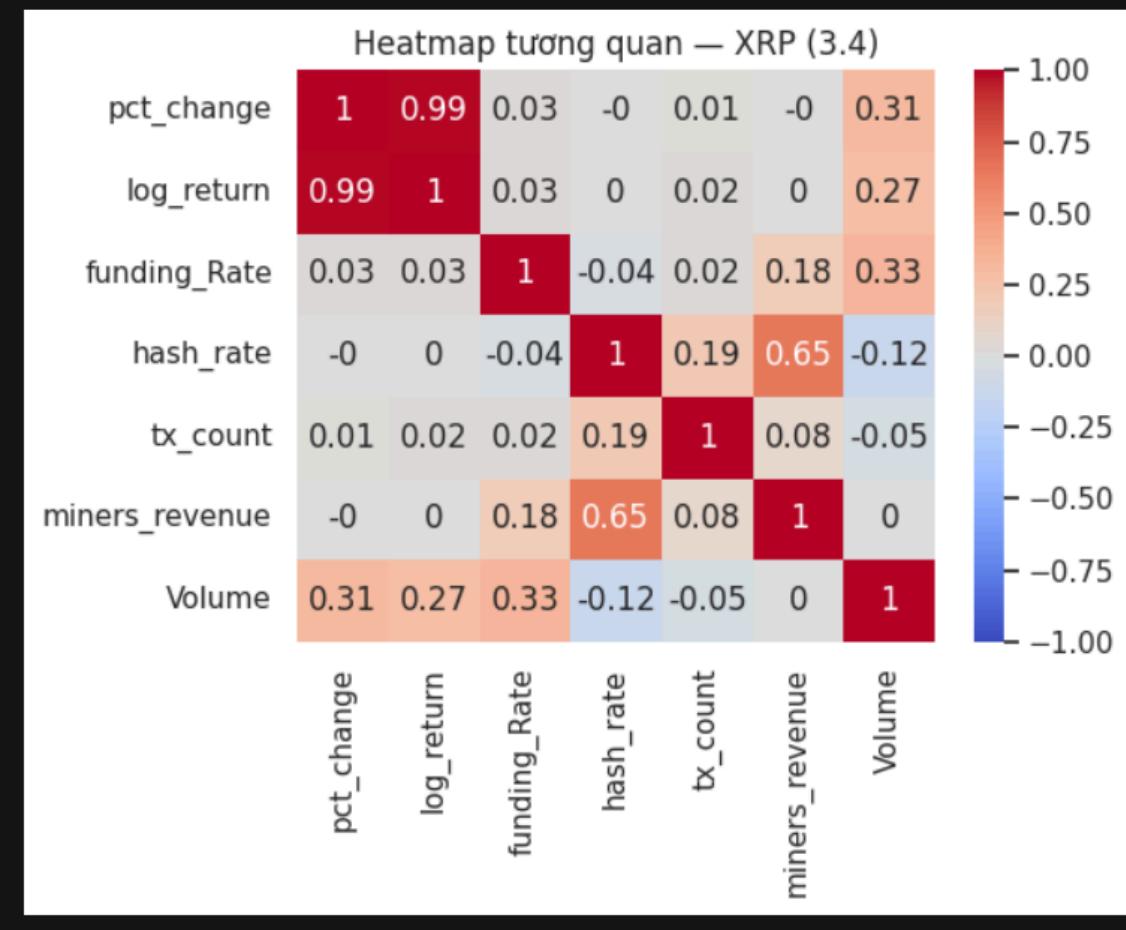
| | pct_change | log_return | funding_Rate | hash_rate | tx_count | miners_revenue | Volume |
|----------------|------------|------------|--------------|-----------|----------|----------------|--------|
| pct_change | 1.00 | 1.00 | -0.05 | -0.04 | -0.02 | -0.02 | 0.20 |
| log_return | 1.00 | 1.00 | -0.05 | -0.03 | -0.01 | -0.01 | 0.17 |
| funding_Rate | -0.05 | -0.05 | 1.00 | -0.02 | 0.04 | 0.11 | 0.02 |
| hash_rate | -0.04 | -0.03 | -0.02 | 1.00 | 0.19 | 0.65 | -0.11 |
| tx_count | -0.02 | -0.01 | 0.04 | 0.19 | 1.00 | 0.08 | -0.11 |
| miners_revenue | -0.02 | -0.01 | 0.11 | 0.65 | 0.08 | 1.00 | 0.04 |
| Volume | 0.20 | 0.17 | 0.02 | -0.11 | -0.11 | 0.04 | 1.00 |



Hình 3.4e — Ma trận tương quan & Heatmap(XRP)

🔥 3.4 – Correlation matrix (XRP)

| | pct_change | log_return | funding_Rate | hash_rate | tx_count | miners_revenue | Volume |
|----------------|------------|------------|--------------|-----------|----------|----------------|--------|
| pct_change | 1.00 | 0.99 | 0.03 | -0.00 | 0.01 | -0.00 | 0.31 |
| log_return | 0.99 | 1.00 | 0.03 | 0.00 | 0.02 | 0.00 | 0.27 |
| funding_Rate | 0.03 | 0.03 | 1.00 | -0.04 | 0.02 | 0.02 | 0.18 |
| hash_rate | -0.00 | 0.00 | -0.04 | 1.00 | 0.19 | 0.65 | -0.12 |
| tx_count | 0.01 | 0.02 | 0.02 | 0.19 | 1.00 | 0.08 | -0.05 |
| miners_revenue | -0.00 | 0.00 | 0.18 | 0.65 | 0.08 | 1.00 | 0.00 |
| Volume | 0.31 | 0.27 | 0.33 | -0.12 | -0.05 | 0.00 | 1.00 |



3.4.2. Rolling Correlation (Dynamic Relationship)

- **Mục tiêu:** kiểm tra độ bền & tính thời vụ của quan hệ (ví dụ **return–Volume,hash_rate–miners_revenue, tx_count–hash_rate**).
- **Cửa sổ gợi ý: 30 ngày (ngắn hạn) và 90 ngày (trung hạn);** cân nhắc **lag** 7/14/30 ngày để bắt lead-lag.
- **Ý nghĩa:** giai đoạn $|l|$ lớn cho thấy “đồng pha/nghịch pha” rõ rệt → điều chỉnh trọng số nhóm feature on-chain trong mô hình dự báo.

3.4.3. Nhận xét

- **On-chain ↔ Giá ngắn hạn yếu:** ma trận tĩnh không thích hợp để dự báo return tức thời; nên dùng rolling/expanding và lag features (**hash_rate_lag7, tx_count_lag14, ...**).
- **Cặp đáng tin:** **hash_rate-miners_revenue** là chỉ báo sức khỏe mạng hơn là tín hiệu giá tức thời.
- **Funding-Volume dương vừa (SOL/XRP):** khi thị trường perp “nóng”, thanh khoản tăng; chú ý các giai đoạn funding cực đoan.
- Kiểm định bổ sung: thử Granger causality / lead-lag giữa **funding_Rate** và **returns**; báo cáo thêm p-value cho các cửa sổ cố định.

3.5. Phân tích hồi quy tuyến tính (Linear Relationship Test)

3.5.1. Mục tiêu

- Sàng lọc nhanh các yếu tố (**funding_Rate, fetch_fear_greed, google_interest, hash_rate, tx_count, miners_revenue, Volume**) có quan hệ tuyến tính với **pct_change** theo từng mã (BTC, ETH, BNB, SOL, XRP).
- Xác định yếu tố có ý nghĩa thống kê (p-value < 0.05) để ưu tiên cho bước mô hình hoá tiếp theo (đa biến, có độ trễ).

3.5.2. Mô hình

- **Hồi quy đơn biến theo từng cặp:**

$$pct_change_t = \alpha + \beta \cdot factor_t + \varepsilon_t$$

- Thực thi bằng **scipy.stats.linregress** cho từng **symbol × factor** → thu được **β (slope), p-value, r_value (hệ số tương quan), n_obs (số quan sát)**.
- **Lưu ý:** β chịu ảnh hưởng đơn vị đo (đặc biệt với Volume); p-value là tiêu chí chính để kết luận có/không có ý nghĩa.

```

factors
['funding_Rate', 'fetch_fear_greed', 'hash_rate', 'tx_count', 'miners_revenue', 'Volume']

def quick_reg(y, x):
    ok = (~y.isna()) & (~x.isna())
    if ok.sum() < 10:
        return np.nan, np.nan, np.nan, ok.sum()
    r = linregress(x[ok], y[ok])
    return r.slope, r.pvalue, r.rvalue, ok.sum()

rows = []
for sym, g in df.groupby('symbol', sort=False):
    y = g['pct_change']
    for col in factors:
        beta, pval, r, n = quick_reg(y, g[col])

```

```

rows.append({
    'symbol': sym,
    'factor': col,
    'beta': beta,
    'p_value': pval,
    'r_value': r,
    'n_obs': n
})

reg_summary = pd.DataFrame(rows).sort_values(['symbol', 'p_value'], na_position='last')
print(" 3.5 - Hồi quy nhanh (pct_change ~ factor)")
display(reg_summary.head(30))

# Bảng rút gọn top yếu tố có ý nghĩa ( $p<0.05$ ) theo từng mã
def top_sig(df_reg, alpha=0.05, topk=3):
    out = []
    for sym in df_reg['symbol'].unique():
        tmp = df_reg[(df_reg['symbol']==sym) &
        (df_reg['p_value'].notna())].sort_values('p_value')
        out.append(tmp.head(topk))
    return pd.concat(out) if out else pd.DataFrame()

top_factors = top_sig(reg_summary, alpha=0.05, topk=3)
print("🏁 3.5 - Top yếu tố có ý nghĩa (nhỏ p-value) theo từng mã")
display(top_factors)

```

3.5.3. Kết quả

- **Bảng tổng hợp (mọi symbol × factor):**

3.5 – Hồi quy nhanh (pct_change ~ factor)

| | symbol | factor | beta | p_value | r_value | n_obs |
|----|---------------|------------------|---------------|----------------|----------------|--------------|
| 0 | BNB | funding_Rate | 9.924486e+02 | 1.044720e-02 | 0.079680 | 1032 |
| 3 | BNB | hash_rate | 1.138744e-09 | 9.198025e-02 | 0.052481 | 1032 |
| 5 | BNB | miners_revenue | 1.535903e-08 | 1.637421e-01 | 0.043382 | 1032 |
| 6 | BNB | Volume | 2.840654e-07 | 1.771514e-01 | 0.042043 | 1032 |
| 4 | BNB | tx_count | -3.581514e-07 | 7.419671e-01 | -0.010261 | 1032 |
| 1 | BNB | fetch_fear_greed | | NaN | NaN | NaN |
| 2 | BNB | google_interest | | NaN | NaN | NaN |
| 13 | BTC | Volume | 2.817242e-06 | 1.515878e-03 | 0.098560 | 1033 |
| 7 | BTC | funding_Rate | 3.899567e+02 | 6.728488e-01 | 0.013153 | 1033 |
| 10 | BTC | hash_rate | 2.365360e-10 | 6.967820e-01 | 0.012138 | 1033 |
| 11 | BTC | tx_count | 3.315846e-07 | 7.341685e-01 | 0.010578 | 1033 |
| 12 | BTC | miners_revenue | -1.293304e-09 | 8.961127e-01 | -0.004068 | 1033 |
| 8 | BTC | fetch_fear_greed | | NaN | NaN | NaN |
| 9 | BTC | google_interest | | NaN | NaN | NaN |
| 20 | ETH | Volume | 5.334704e-07 | 1.427379e-01 | 0.045634 | 1033 |
| 14 | ETH | funding_Rate | 1.673982e+03 | 1.646796e-01 | 0.043265 | 1033 |
| 19 | ETH | miners_revenue | 6.338551e-09 | 6.355041e-01 | 0.014765 | 1033 |
| 17 | ETH | hash_rate | 2.113939e-10 | 7.964674e-01 | 0.008034 | 1033 |
| 18 | ETH | tx_count | 3.234609e-07 | 8.061890e-01 | 0.007643 | 1033 |
| 15 | ETH | fetch_fear_greed | | NaN | NaN | NaN |
| 16 | ETH | google_interest | | NaN | NaN | NaN |
| 27 | SOL | Volume | 2.992828e-07 | 1.330437e-10 | 0.198118 | 1033 |
| 21 | SOL | funding_Rate | -9.708952e+02 | 8.965419e-02 | -0.052834 | 1033 |
| 24 | SOL | hash_rate | -1.407953e-09 | 2.309898e-01 | -0.037300 | 1033 |
| 25 | SOL | tx_count | -1.178963e-06 | 5.330352e-01 | -0.019417 | 1033 |
| 26 | SOL | miners_revenue | -9.906829e-09 | 6.055773e-01 | -0.016085 | 1033 |
| 22 | SOL | fetch_fear_greed | | NaN | NaN | NaN |
| 23 | SOL | google_interest | | NaN | NaN | NaN |
| 34 | XRP | Volume | 4.967022e-09 | 4.674871e-24 | 0.307472 | 1033 |
| 28 | XRP | funding_Rate | 1.192492e+03 | 3.591350e-01 | 0.028560 | 1033 |

- **Top yếu tố có p-value nhỏ nhất mỗi symbol (không lọc alpha trong code gốc):**

❖ 3.5 – Top yếu tố có ý nghĩa (nhỏ p-value) theo từng mã

| | symbol | factor | beta | p_value | r_value | n_obs |
|----|---------------|----------------|---------------|----------------|----------------|--------------|
| 0 | BNB | funding_Rate | 9.924486e+02 | 1.044720e-02 | 0.079680 | 1032 |
| 3 | BNB | hash_rate | 1.138744e-09 | 9.198025e-02 | 0.052481 | 1032 |
| 5 | BNB | miners_revenue | 1.535903e-08 | 1.637421e-01 | 0.043382 | 1032 |
| 13 | BTC | Volume | 2.817242e-06 | 1.515878e-03 | 0.098560 | 1033 |
| 7 | BTC | funding_Rate | 3.899567e+02 | 6.728488e-01 | 0.013153 | 1033 |
| 10 | BTC | hash_rate | 2.365360e-10 | 6.967820e-01 | 0.012138 | 1033 |
| 20 | ETH | Volume | 5.334704e-07 | 1.427379e-01 | 0.045634 | 1033 |
| 14 | ETH | funding_Rate | 1.673982e+03 | 1.646796e-01 | 0.043265 | 1033 |
| 19 | ETH | miners_revenue | 6.338551e-09 | 6.355041e-01 | 0.014765 | 1033 |
| 27 | SOL | Volume | 2.992828e-07 | 1.330437e-10 | 0.198118 | 1033 |
| 21 | SOL | funding_Rate | -9.708952e+02 | 8.965419e-02 | -0.052834 | 1033 |
| 24 | SOL | hash_rate | -1.407953e-09 | 2.309898e-01 | -0.037300 | 1033 |
| 34 | XRP | Volume | 4.967022e-09 | 4.674871e-24 | 0.307472 | 1033 |
| 28 | XRP | funding_Rate | 1.192492e+03 | 3.591350e-01 | 0.028560 | 1033 |
| 32 | XRP | tx_count | 6.644177e-07 | 7.185328e-01 | 0.011227 | 1033 |

- **Các phát hiện đạt ngưỡng ý nghĩa ($p < 0.05$):**

- **BTC:** Volume ($\beta \approx 2.82e-06$, $p \approx 1.56e-03$, $r \approx 0.099$) → khối lượng tăng cùng pha nhẹ với biến động giá.
- **BNB:** funding_Rate ($\beta \approx 9.92e+02$, $p \approx 1.05e-02$) → funding có ý nghĩa dương với pct_change.
- **ETH:** chưa có yếu tố nào đạt $p < 0.05$ trong kiểm định đơn biến.
- **SOL:** Volume ($\beta \approx 2.99e-07$, $p \approx 1.33e-10$, $r \approx 0.199$) → tác động dương rõ rệt.
- **XRP:** Volume ($\beta \approx 4.97e-09$, $p \approx 4.67e-24$, $r \approx 0.307$) và funding_Rate ($\beta \approx 1.19e+03$, $p \approx 0.0286$) → cả hai đều có ý nghĩa thống kê.

• **Kích thước mẫu:** ~1,032–1,033 quan sát/biến (sau khi loại NaN theo hàng).

3.5.4. Nhận xét

- **Volume** là yếu tố nổi bật: có ý nghĩa ở BTC, SOL, XRP (đặc biệt mạnh ở XRP), gợi ý tín hiệu thanh khoản liên hệ cùng pha với biến động giá ngắn hạn.
- **Funding_Rate** cho thấy tín hiệu ở BNB và XRP → gợi ý vai trò của thị trường phái sinh/perp trong các pha “nóng”.

- **ETH** không có yếu tố đơn biến đạt ngưỡng 0.05 → khả năng quan hệ phi tuyến hoặc có độ trễ; nên kiểm tra lag features và mô hình đa biến.
- **Cảnh báo nhiều phép thử (multiple testing):** khi đánh giá nhiều yếu tố, nên cân nhắc điều chỉnh FDR (Benjamini–Hochberg) hoặc Bonferroni để tránh kết luận giả.
- **Bước tiếp theo:**
 - Dùng mô hình đa biến (OLS/Ridge/Lasso) với chuẩn hoá thang đo.
 - Thêm độ trễ (7/14/30 ngày) cho Volume, funding_Rate và on-chain.
 - Kiểm tra tương tác (Volume × funding_Rate), rolling window, và robust SE nếu có phương sai sai số thay đổi.

3.6. Phân tích quan hệ giữa các coin (Cross-Market Correlation)

3.6.1. Mục tiêu

- Đo tương quan cùng ngày giữa các coin bằng log_return.
- Ước lượng Beta so với BTC (BTC xem như “thị trường”).
- Diễn giải rủi ro tương đối: $\beta > 1$ = nhạy hơn BTC; $\beta < 1$ = “trầm” hơn.
- Ứng dụng quản trị rủi ro: sizing vị thế, xây dựng danh mục beta-target.

```
# Ma trận tương quan giữa các coin (log_return)
ret_wide = df.pivot(index='Date', columns='symbol',
                     values='log_return').dropna(how='all').copy()
# Bỏ cột rất rỗng (nếu có)
valid_cols = ret_wide.columns[ret_wide.notna().mean() > 0.7] # giữ cột có >=70% dữ liệu
ret_wide = ret_wide[valid_cols].dropna()

print("📊 3.6 – Corr giữa returns các coin (cùng ngày)")
corr_coins = ret_wide.corr().round(2)
display(corr_coins)

plt.figure(figsize=(6,5))
sns.heatmap(corr_coins, annot=True, vmin=-1, vmax=1, cmap='coolwarm')
plt.title('Tương quan log_return giữa các coin (3.6)')
plt.show()

# Beta vs BTC (coi BTC như "thị trường") – regress  $r_i \sim r_{BTC}$ 
if 'BTC' in ret_wide.columns:
    betas = []
    r_btc = ret_wide['BTC']
    for c in ret_wide.columns:
        if c == 'BTC':
            continue
        ok = (~r_btc.isna()) & (~ret_wide[c].isna())
        if ok.sum() < 30:
            beta = np.nan; p = np.nan; r = np.nan; n = ok.sum()
        else:
            res = linregress(r_btc[ok], ret_wide[c][ok])
            beta, p, r, n = res.slope, res.pvalue, res.rvalue, ok.sum()
```

```

        betas.append({'symbol': c, 'beta_vs_BTC': beta, 'p_value': p, 'r_value': r, 'n_obs': n})

    beta_tbl = pd.DataFrame(betas).sort_values('beta_vs_BTC', na_position='last',
ascending=False)
    print("📌 3.6 - Beta vs BTC ( $r_i = \alpha + \beta * r_{BTC}$ )")
    display(beta_tbl)

# Bar chart beta (chụp cho báo cáo)
plt.figure(figsize=(7,4))
sns.barplot(data=beta_tbl, x='symbol', y='beta_vs_BTC')
plt.title('Beta vs BTC (3.6)')
plt.xlabel('symbol'); plt.ylabel('beta vs BTC')
plt.show()
else:
    print("⚠️ Không có cột BTC trong dữ liệu để tính Beta vs BTC.")

```

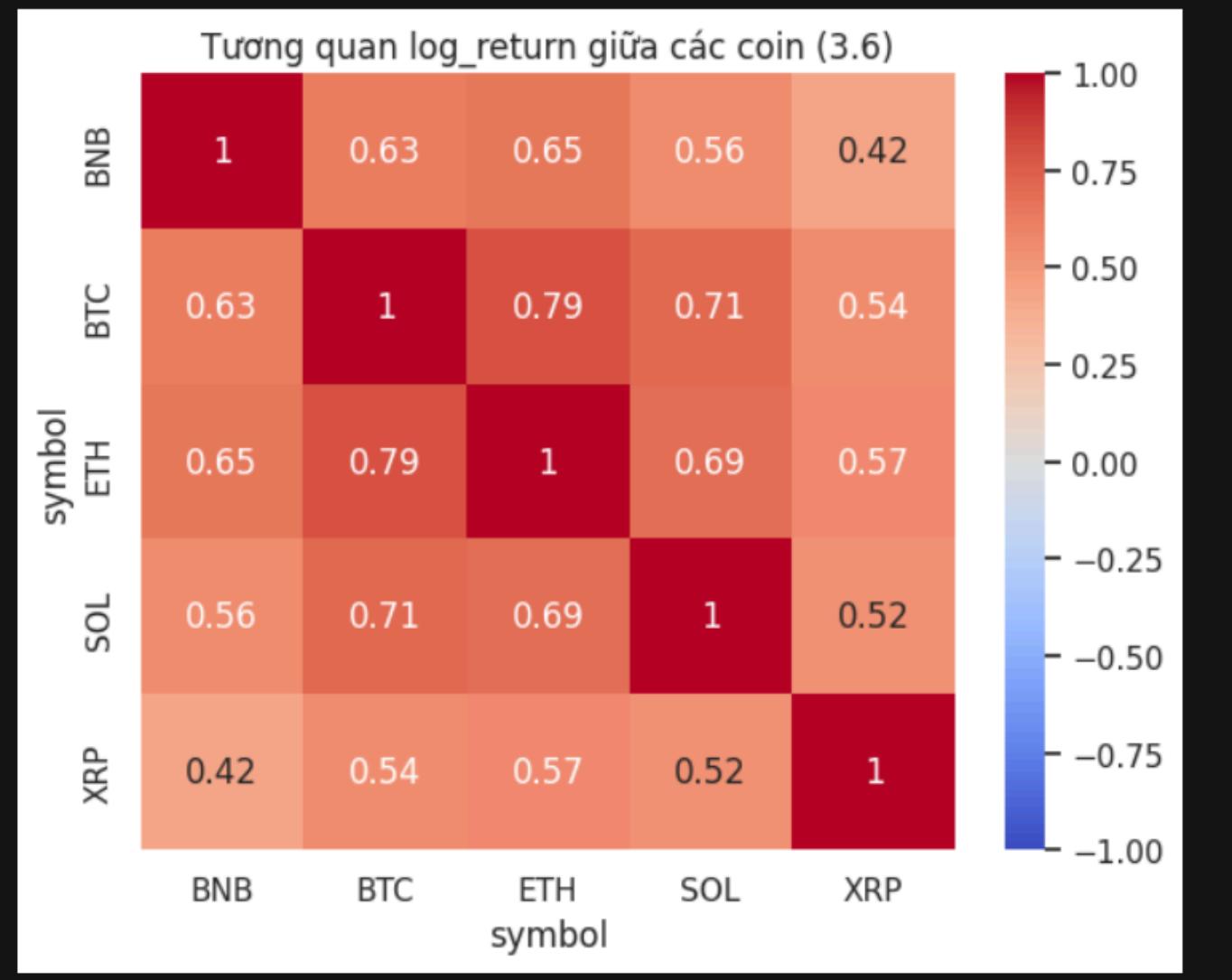
3.6.2. Kết quả

- **Tương quan cùng ngày:** các cặp đều dương vừa-mạnh. Mạnh nhất là BTC-ETH ~ 0.79, kế đến BTC-SOL ~ 0.71; XRP thấp hơn các cặp còn lại (khoảng 0.52–0.57). Điều này cho thấy các coin thường di chuyển cùng hướng, nhưng mức độ gắn kết khác nhau.

Hình 3.6a — Ma trận tương quan log return giữa các coin (cùng ngày).

3.6 – Corr giữa returns các coin (cùng ngày)

| symbol | BNB | BTC | ETH | SOL | XRP |
|--------|------|------|------|------|------|
| symbol | | | | | |
| BNB | 1.00 | 0.63 | 0.65 | 0.56 | 0.42 |
| BTC | 0.63 | 1.00 | 0.79 | 0.71 | 0.54 |
| ETH | 0.65 | 0.79 | 1.00 | 0.69 | 0.57 |
| SOL | 0.56 | 0.71 | 0.69 | 1.00 | 0.52 |
| XRP | 0.42 | 0.54 | 0.57 | 0.52 | 1.00 |



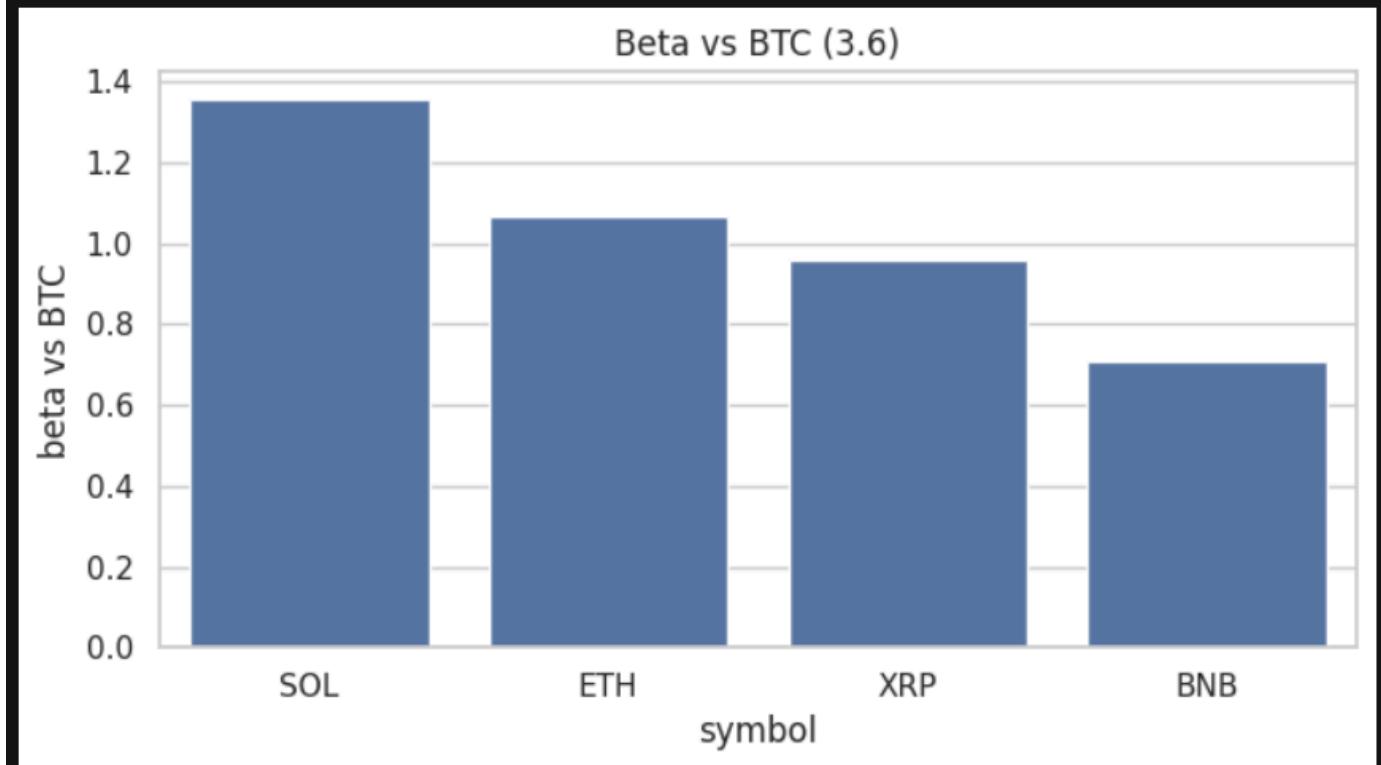
• Beta vs BTC (ý nghĩa p-value ≪ 0.01):

- **SOL:** $\beta \approx 1.36 \rightarrow$ biến động lớn hơn BTC (nhạy nhất).
- **ETH:** $\beta \approx 1.07 \rightarrow$ hơi “nhạy” hơn BTC.
- **XRP:** $\beta \approx 0.96 \rightarrow$ xấp xỉ BTC.
- **BNB:** $\beta \approx 0.71 \rightarrow$ trầm hơn BTC.

Hình 3.6b — β so với BTC và ý nghĩa thống kê (bảng & biểu đồ).

❖ 3.6 – Beta vs BTC ($r_i = \alpha + \beta * r_{BTC}$)

| | symbol | beta_vs_BTC | p_value | r_value | n_obs |
|---|--------|-------------|---------------|----------|-------|
| 2 | SOL | 1.356233 | 1.933914e-157 | 0.707468 | 1032 |
| 1 | ETH | 1.066157 | 4.293692e-222 | 0.791088 | 1032 |
| 3 | XRP | 0.960075 | 1.290597e-80 | 0.544197 | 1032 |
| 0 | BNB | 0.706704 | 5.674286e-117 | 0.633697 | 1032 |



- **Hàm ý:** khi BTC biến động, SOL/ETH thường khuếch đại biến động; BNB thường biến động nhỏ hơn. Điều này hữu ích cho quản trị rủi ro (position sizing) và dựng danh mục beta-target.

4. Data Visualization

4.1 Chuẩn bị

- **Mục tiêu:** thiết lập thư mục lưu hình, nạp dữ liệu đã làm sạch/chuẩn hoá, cấu hình style & kích thước biểu đồ.
- **Dữ liệu sử dụng:**
 - `data/dataset_total_clean.csv` (bản đã làm sạch)
 - `data/dataset_scaled_total.csv` (bản đã chuẩn hoá tỉ lệ)
- **Thư viện:** `pandas`, `matplotlib.pyplot`, `seaborn`, `os`.
- **Thiết lập hiển thị:** `sns.set_style("darkgrid"); plt.rcParams['figure.figsize'] = (10, 5)`
- **Thư mục ảnh:** tạo `data/picture/` (tự động nếu chưa tồn tại).

```
# --- Setup ---
os.makedirs("data/picture", exist_ok=True)

df_clean = pd.read_csv("data/dataset_total_clean.csv")
df_scaled = pd.read_csv("data/dataset_scaled_total.csv")

sns.set_style("darkgrid")
plt.rcParams['figure.figsize'] = (10, 5)
```

Hình 4.1 — Code khởi tạo cho phần trực quan hóa

4.2 Waffle Chart — Tỷ trọng khối lượng giao dịch giữa các mã

- **Mục tiêu:** trực quan hóa **tỷ trọng Volume** của từng mã (BNB/BTC/ETH/SOL/XRP) trong tổng khối lượng.
- **Dữ liệu đầu vào:** `symbol`, `Volume` (đã tổng theo mã).
- **Chuẩn hóa:** quy đổi tỷ trọng về **500 ô** (rows = 10) $\Rightarrow 1 \text{ ô} = 0.2\%$ tổng khối lượng.
- **Ý tưởng đọc biểu đồ:** dải ô của mã nào **dài hơn** \Rightarrow thị phần khối lượng **lớn hơn**; màu sắc phân biệt từng mã, có chú giải (legend).
- **Ứng dụng:** so sánh nhanh mức quan tâm/thanh khoản tương đối giữa các coin; dùng để **chọn mẫu** cho phân tích sâu (vd. top 2-3 mã có volume lớn nhất).

```

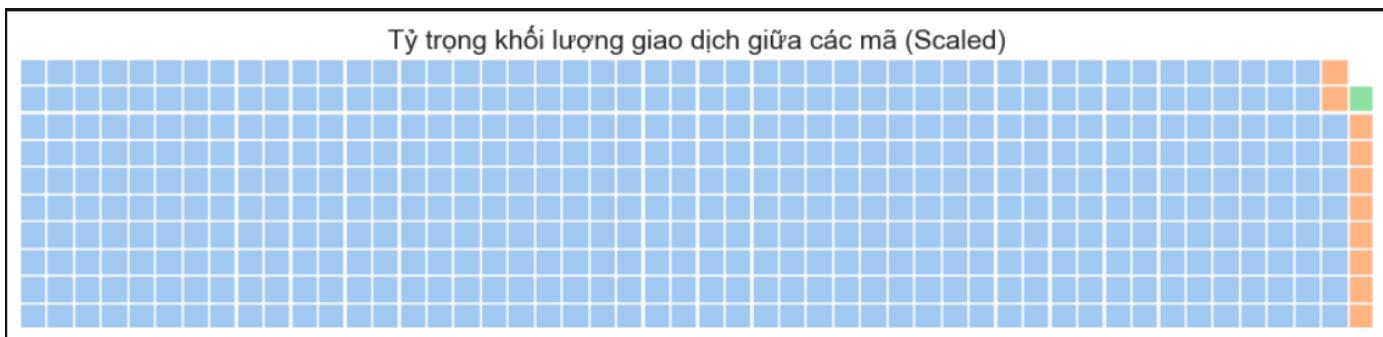
# WAFFLE CHART – tổng khối lượng (scaled version)
symbol_share = df.groupby('symbol')['volume'].sum().sort_values(ascending=False)

# Chuẩn hóa tổng số ô = 100
scaled_share = (symbol_share / symbol_share.sum() * 500).round().astype(int)

fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    values=scaled_share,
    title={'label': 'Tỷ trọng khối lượng giao dịch giữa các mã (Scaled)', 'loc': 'center', 'fontsize': 14},
    colors=sns.color_palette('pastel', n_colors=len(scaled_share)),
    Legend={'loc': 'upper left', 'bbox_to_anchor': (1.1, 1)}
)
plt.savefig("data/pic/waffle_volume_share.png", bbox_inches='tight', dpi=300)
plt.show()
plt.close()

```

Hình 4.2a — Code tạo Waffle Chart cho tỷ trọng Volume



Hình 4.2b — Waffle Chart: Tỷ trọng khối lượng giao dịch giữa các mã (Scaled)

4.3 Area Plot — Xu hướng giá theo thời gian

- **Mục tiêu:** trực quan hóa **xu hướng giá đóng cửa (Close)** của từng mã qua thời gian; phần area tô nền giúp thấy rõ pha tích lũy/đẩy mạnh.
- **Dữ liệu:** cột **Date, symbol, Close**.
- **Thiết kế:** đường giá màu xanh, **fill_between** nhạt bên dưới; trục x = thời gian, trục y = USD; mỗi mã một biểu đồ riêng để đọc rõ.

```

# AREA PLOT – mỗi mã riêng

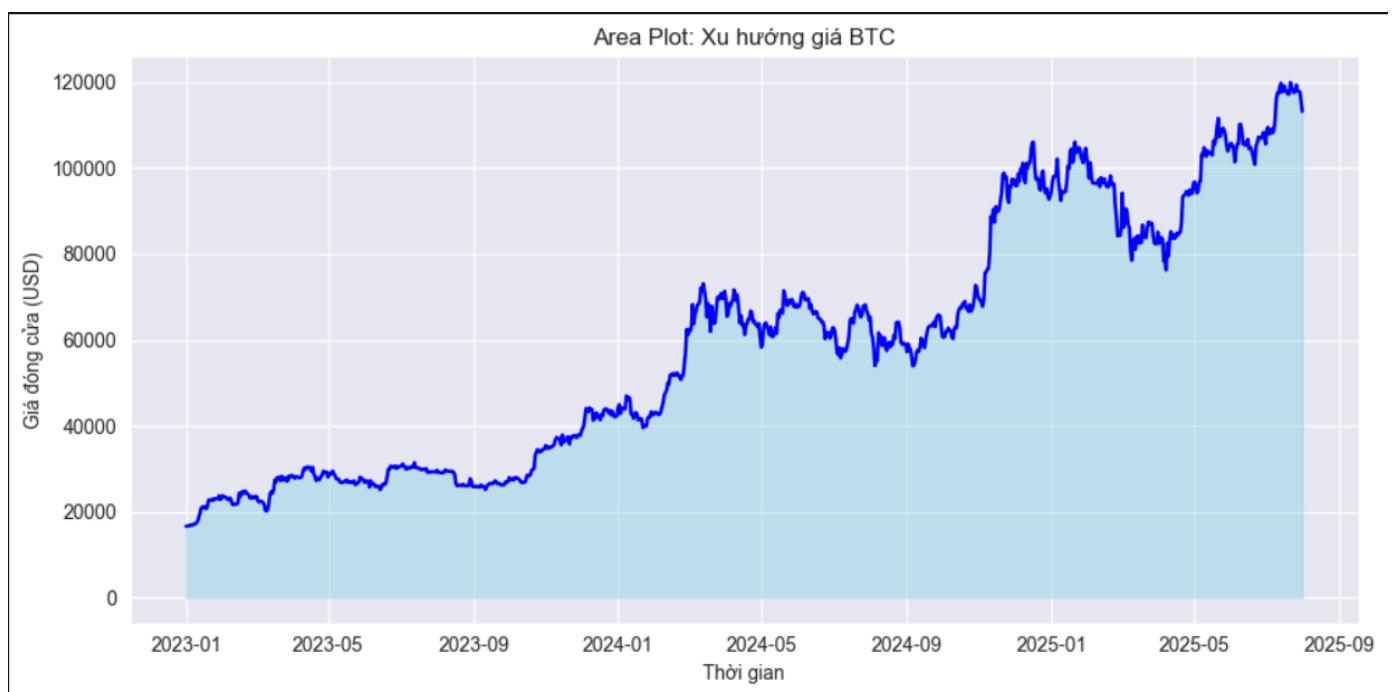
for sym in df['symbol'].unique():
    sub = df[df['symbol'] == sym]
    plt.figure()
    plt.fill_between(sub['Date'], sub['Close'], color='skyblue', alpha=0.4)
    plt.plot(sub['Date'], sub['Close'], color='blue', linewidth=1.8)
    plt.title(f"Area Plot: Xu hướng giá {sym}")
    plt.xlabel("Thời gian")
    plt.ylabel("Giá đóng cửa (USD)")
    plt.tight_layout()
    plt.savefig(f"data/{sym}/pic/area_close_{sym}.png", dpi=300)
    plt.show()

```

Hình 4.3a — Code vẽ Area Plot cho từng mã



Hình 4.3a — Area Plot: Xu hướng giá BNB



Hình 4.3b — Area Plot: Xu hướng giá BTC



Hình 4.3c — Area Plot: Xu hướng giá ETH



Hình 4.3d — Area Plot: Xu hướng giá SOL



Hình 4.3e — Area Plot: Xu hướng giá XRP

Nhận xét:

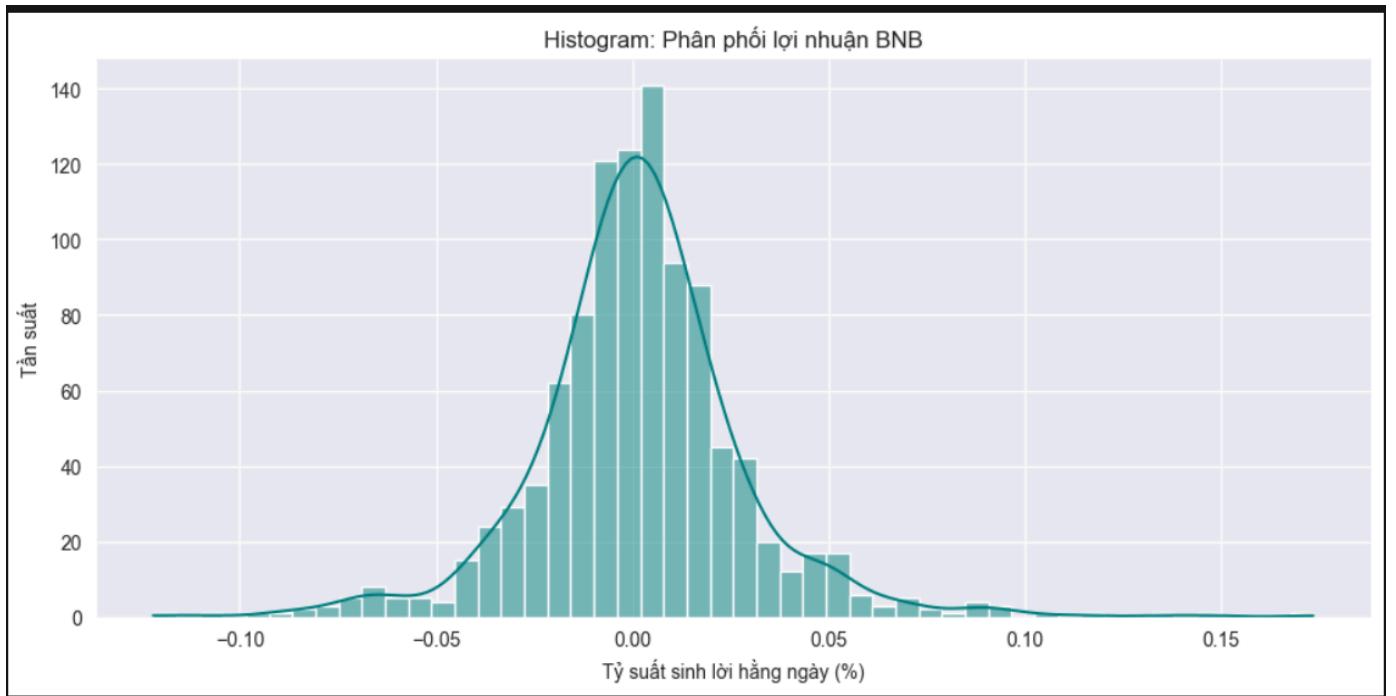
- **BNB:** xu hướng tăng bền ở 2024–2025, điều chỉnh nông hơn so với SOL/XRP.
- **BTC:** chu kỳ tăng mạnh 2024–2025; các nhịp điều chỉnh vẫn giữ cấu trúc đáy cao dần.
- **ETH:** biến động theo chu kỳ, biên độ hẹp hơn SOL; đợt đỉnh ~Q2/2024 nổi bật.
- **SOL:** tăng mạnh từ cuối 2023, các sóng điều chỉnh sâu hơn → rủi ro/biên độ lớn.
- **XRP:** có bước nhảy giá rõ 2025 đầu năm; sau đó dao động rộng, dễ xuất hiện outlier.

4.4 Histogram — Phân phối lợi nhuận ngày (Return)

- **Mục tiêu:** quan sát **hình dạng phân phối** của lợi nhuận ngày theo từng mã để phát hiện **fat-tail, skew** và **outlier**.
- **Thiết lập:** `bins=50`, có **KDE** để ước lượng mật độ mượt; trục x = % return ngày, trục y = tần suất.
- **Cách đọc:**
 - **Đuôi dày (fat-tail):** nhiều điểm xa trung tâm hơn Normal → rủi ro nhảy giá.
 - **Lệch phải/trái (skew):** đỉnh lệch khỏi 0 → xu hướng phiên tăng/giảm cực trị nhiều hơn.
 - **Outlier:** cột tách biệt xa phần lớn dữ liệu → gợi ý sự kiện.
- **Ứng dụng:** chọn tham số quản trị rủi ro (VaR/ES), cân nhắc mô hình phân phối thay thế (Student-t, skew-t).

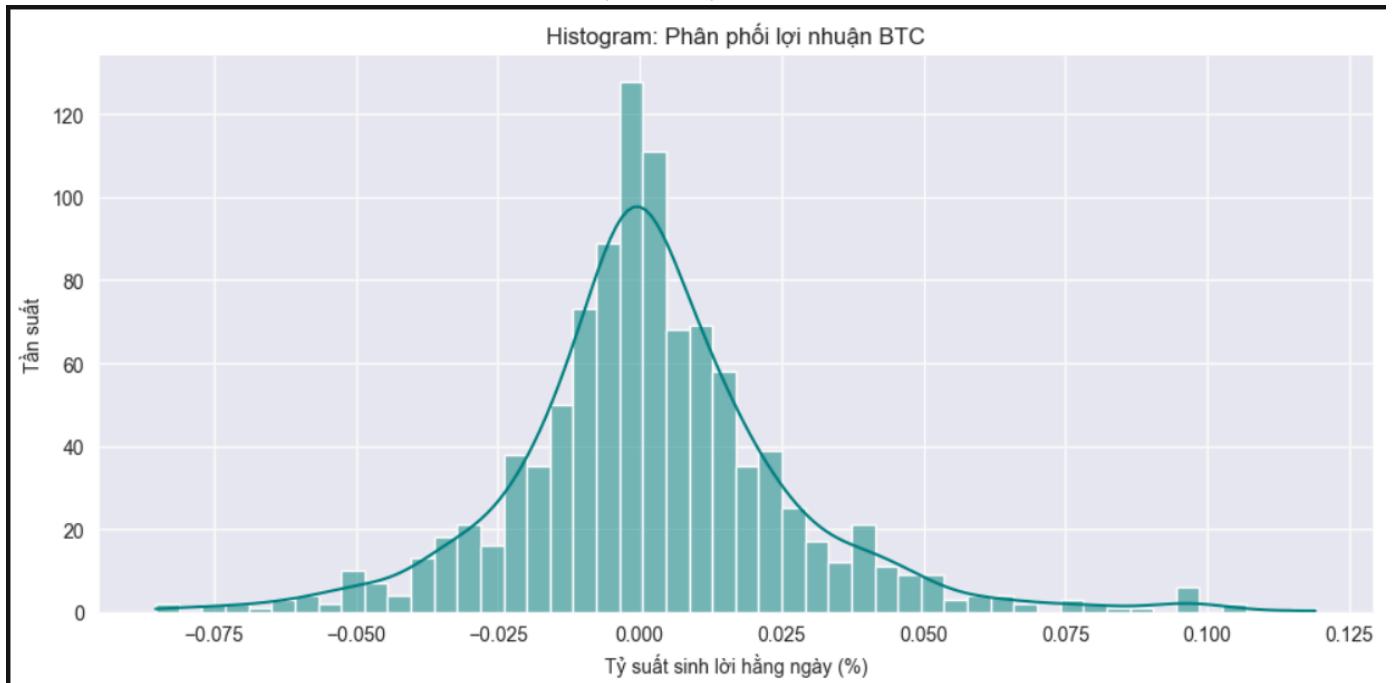
```
# =====
# 3 HISTOGRAM – phân phối lợi nhuận
#
for sym in df['symbol'].unique():
    sub = df[df['symbol'] == sym]
    plt.figure()
    sns.histplot(sub['Return'], bins=50, kde=True, color='teal')
    plt.title(f"Histogram: Phân phối lợi nhuận {sym}")
    plt.xlabel("Tỷ suất sinh lời hàng ngày (%)")
    plt.ylabel("Tần suất")
    plt.tight_layout()
    plt.savefig(f"data/{sym}/pic/hist_return_{sym}.png", dpi=300)
    plt.show()
```

Hình 4.4a — Code vẽ Histogram return



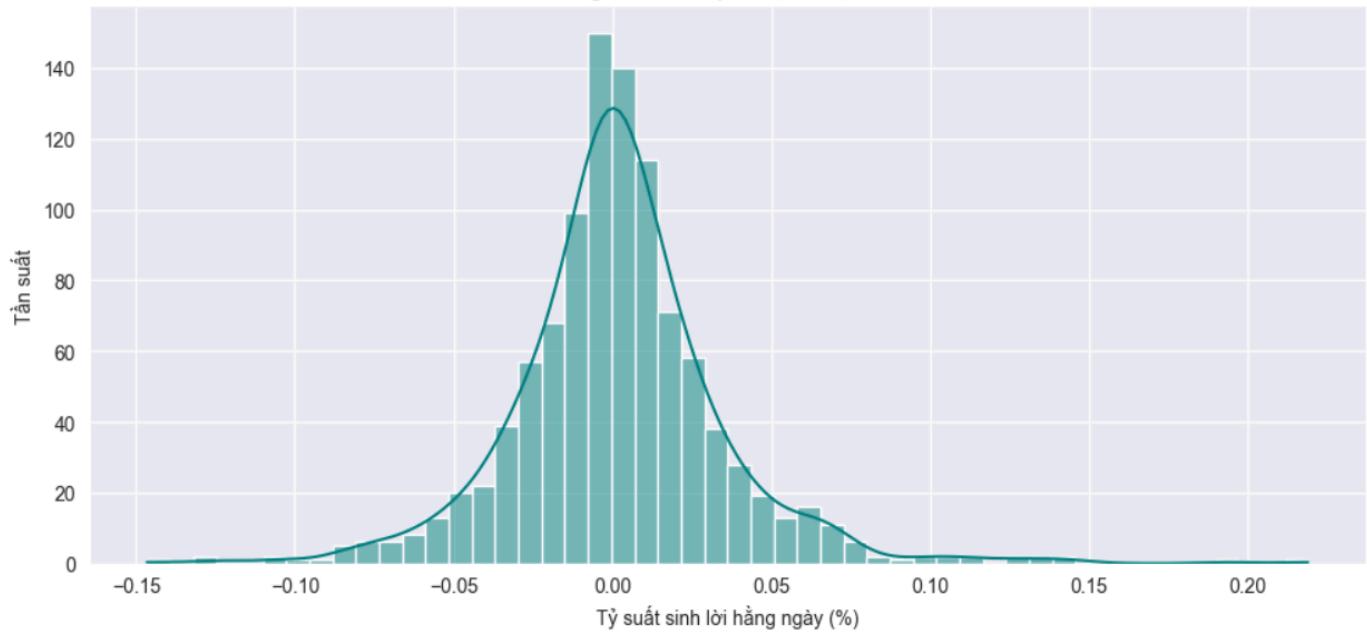
Hình 4.4b — Histogram: Phân phối lợi nhuận BNB

Hình 4.4c — Histogram: Phân phối lợi nhuận BTC



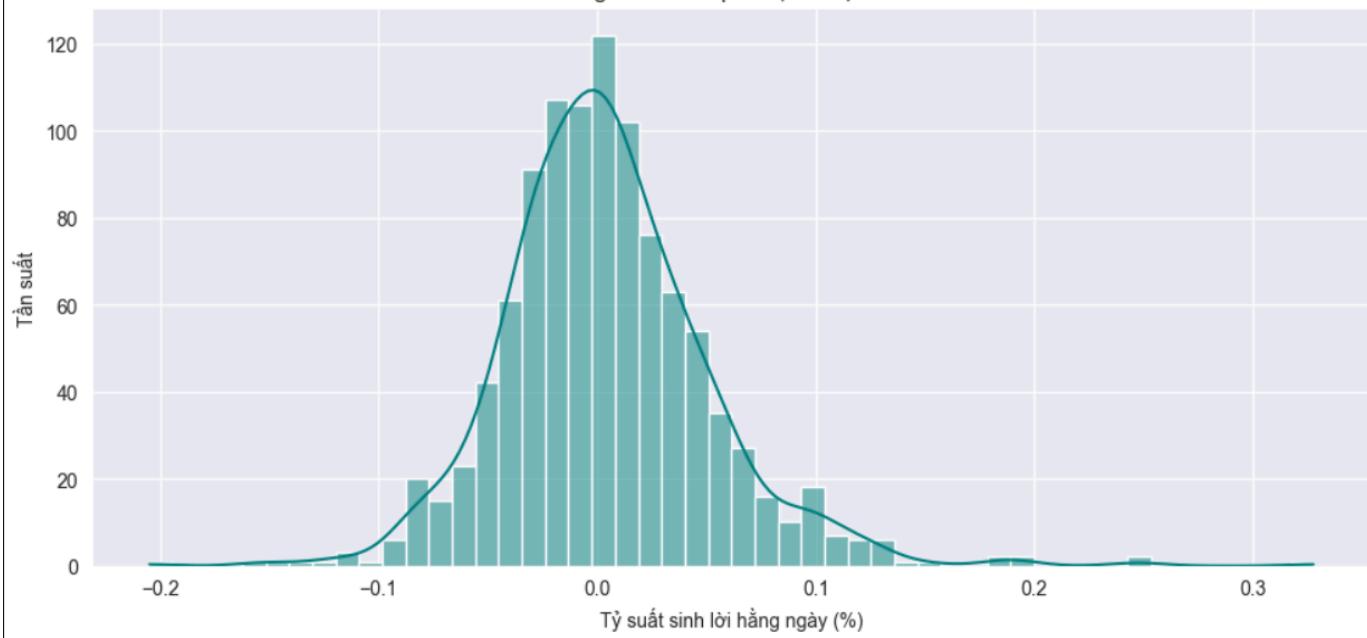
Hình 4.4c — Histogram: Phân phối lợi nhuận BTC

Histogram: Phân phối lợi nhuận ETH

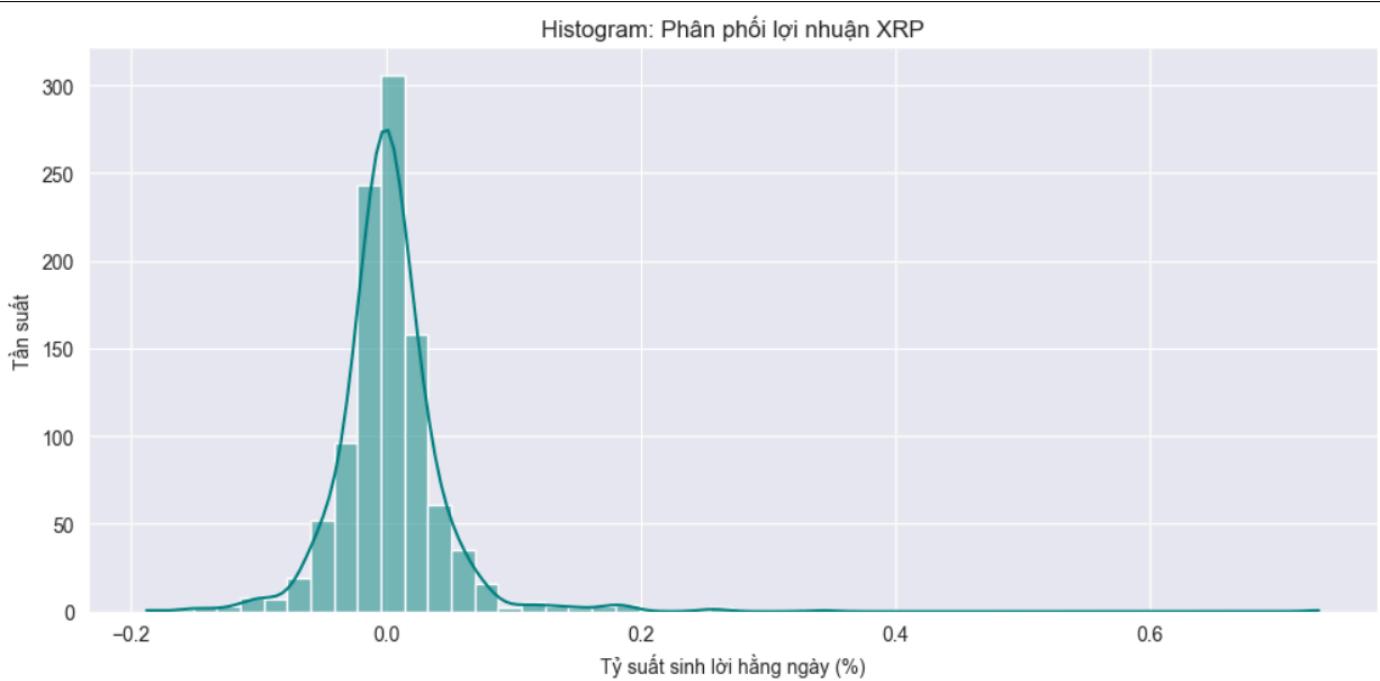


Hình 4.4d — Histogram: Phân phối lợi nhuận ETH

Histogram: Phân phối lợi nhuận SOL



Hình 4e — Histogram: Phân phối lợi nhuận SOL



Hình 4f — Histogram: Phân phối lợi nhuận XRP

Nhận xét:

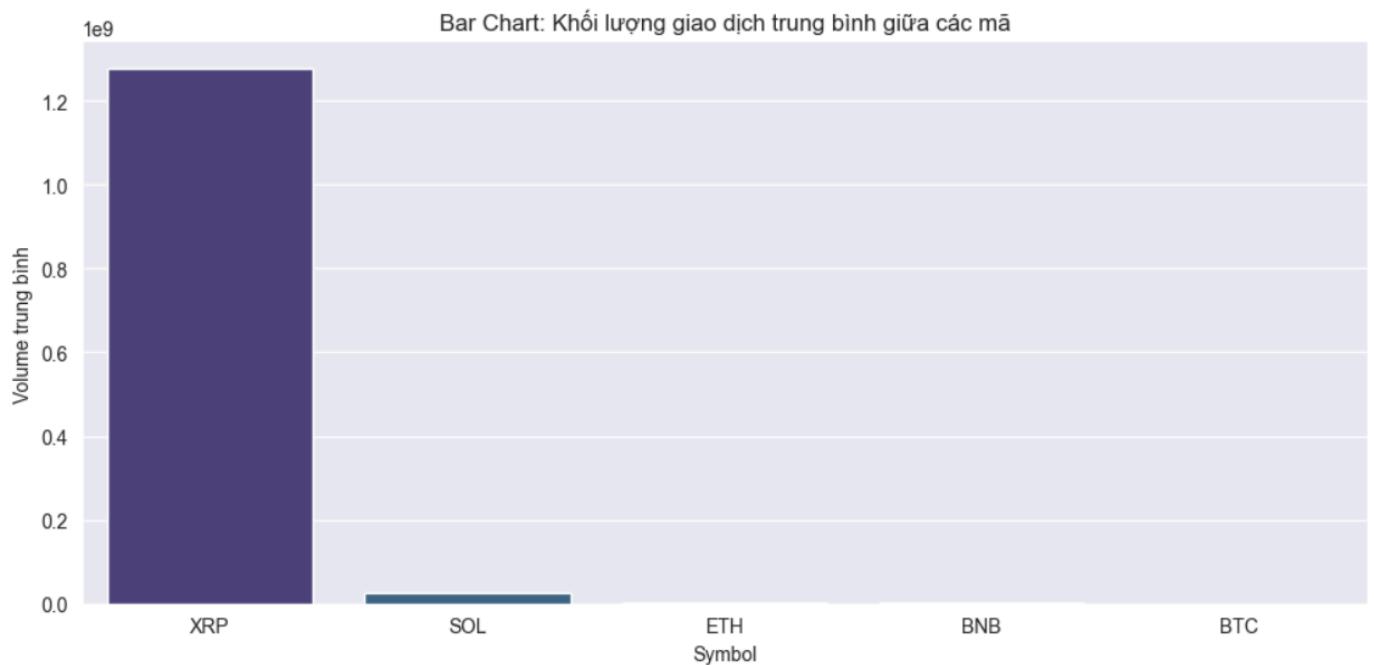
- **Đuôi dày (fat-tail)** xuất hiện ở tất cả các mã; **SOL, XRP** đặc biệt dày → rủi ro nhảy giá lớn.
- **Skew phải nhẹ** ở **BTC/ETH** (đuôi phải dài hơn) trong khi **XRP** có vài **outlier dương** rất xa.
- **Độ phân tán**: thứ tự rộng → hép xấp xỉ **SOL ≈ XRP > ETH > BTC ≈ BNB**, phù hợp nhận định biến động.

4.5 Bar Chart — Khối lượng giao dịch trung bình giữa các mã

- **Mục tiêu:** so sánh **Volume trung bình** của từng mã (BNB, BTC, ETH, SOL, XRP).
- **Cách tính:** nhóm theo **symbol**, lấy **mean(Volume)**, sắp xếp giảm dần.
- **Cách đọc:** cột cao hơn ⇒ **thanh khoản trung bình lớn hơn**.

```
# =====
# 💲⚡ BAR CHART – tắt tất cả symbol
# =====
avg_vol = df.groupby('symbol')['volume'].mean().sort_values(ascending=False)
plt.figure()
sns.barplot(x=avg_vol.index, y=avg_vol.values, palette='viridis')
plt.title("Bar Chart: Khối lượng giao dịch trung bình giữa các mã")
plt.xlabel("Symbol")
plt.ylabel("Volume trung bình")
plt.tight_layout()
plt.savefig("data/pic/bar_avg_volume.png", dpi=300)
plt.show()
```

Hình 4.5a — Code vẽ Bar Chart Volume trung bình



Hình 4.5b — Bar Chart: Khối lượng giao dịch trung bình giữa các mã.

Nhận xét:

- **XRP** vượt trội về **Volume trung bình** → thị trường sôi động nhất.
- **SOL** đứng sau, còn **BTC/ETH/BNB** thấp hơn đáng kể trong tập dữ liệu này.
- Chênh lệch rất lớn ⇒ khi so sánh trực quan, cân nhắc **log-scale** để tránh “nuốt” các cột nhỏ.

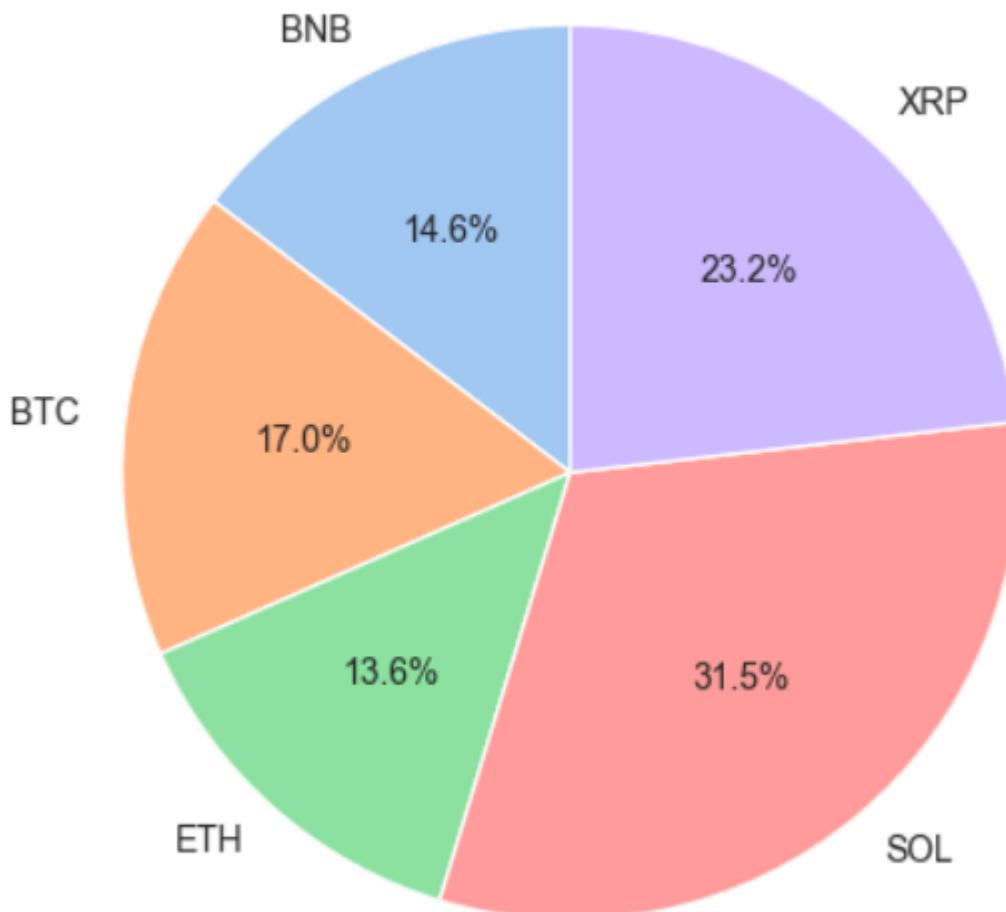
4.6 Pie Chart — Tỷ lệ lợi nhuận trung bình giữa các mã

- **Mục tiêu:** so sánh **đóng góp tương đối** của **lợi nhuận trung bình ngày** (*mean Return*) giữa các mã (BNB, BTC, ETH, SOL, XRP).
- **Cách tính:** `avg_return = df.groupby('symbol')['Return'].mean()` → vẽ pie với tỷ lệ phần trăm = $\text{avg_return}_i / \sum \text{avg_return}$.
- **Cách đọc:** lát lớn hơn ⇒ **mã có mean Return cao hơn** trong giai đoạn nghiên cứu.
- **Lưu ý quan trọng:** Pie Chart chỉ phù hợp khi **mean Return không âm** cho tất cả mã. Nếu có mã mean âm, thay bằng **bar chart** hoặc **donut với giá trị đã chuẩn hóa/shift**.

Code tạo Pie Chart tỷ lệ lợi nhuận trung bình giữa các mã

```
plt.figure()
miner_sum = df_clean.groupby('symbol')['miners_revenue'].sum()
plt.pie(miner_sum, labels=miner_sum.index, autopct='%.1f%%', startangle=90)
plt.title("Tỷ trọng miners_revenue giữa các coin")
plt.savefig("data/picture/pie_miner_revenue.png", bbox_inches='tight', dpi=300)
plt.show()
plt.close()
```

Pie Chart: Tỷ lệ lợi nhuận trung bình giữa các mã



Hình 4.6a— Pie Chart: Tỷ lệ lợi nhuận trung bình giữa các mã

Nhận xét:

- **SOL** chiếm khoảng **31-32%**, dẫn đầu về mean Return.
- **XRP** khoảng **23%** → đứng thứ hai.
- **BTC ~17%**, **BNB ~14-15%**, **ETH ~13-14%** → nhóm giữa/thấp hơn.
- Chênh lệch cho thấy hiệu suất trung bình **không đồng đều** giữa các mã → hữu ích để **ưu tiên mẫu** cho phân mô hình hóa.

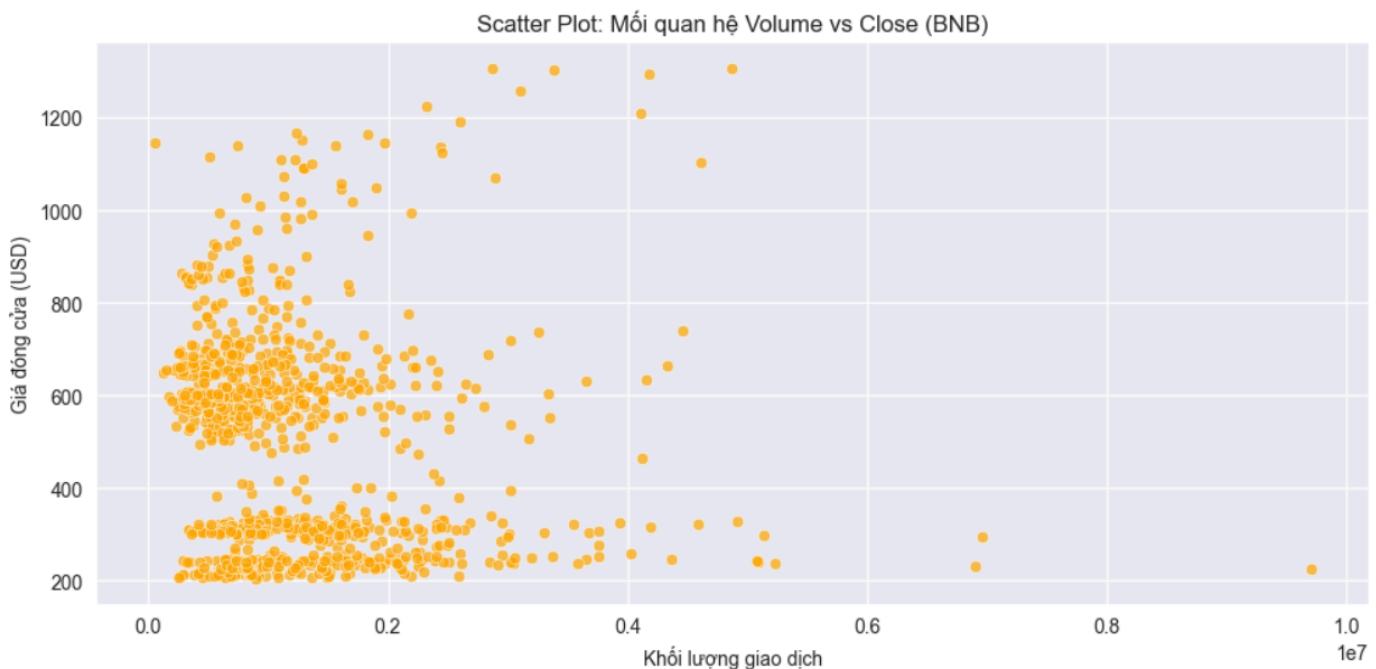
4.7 Scatter Plot — Mối quan hệ Volume và Close

- **Mục tiêu:** quan sát quan hệ giữa **khối lượng giao dịch (Volume)** và **giá đóng cửa (Close)** theo từng mã; phát hiện cụm điểm (regime), xu thế tổng quát và outlier.
- **Thiết lập:** mỗi mã một đồ thị; điểm màu cam, độ mờ **alpha=0.7**; trục x = Volume, trục y = USD.

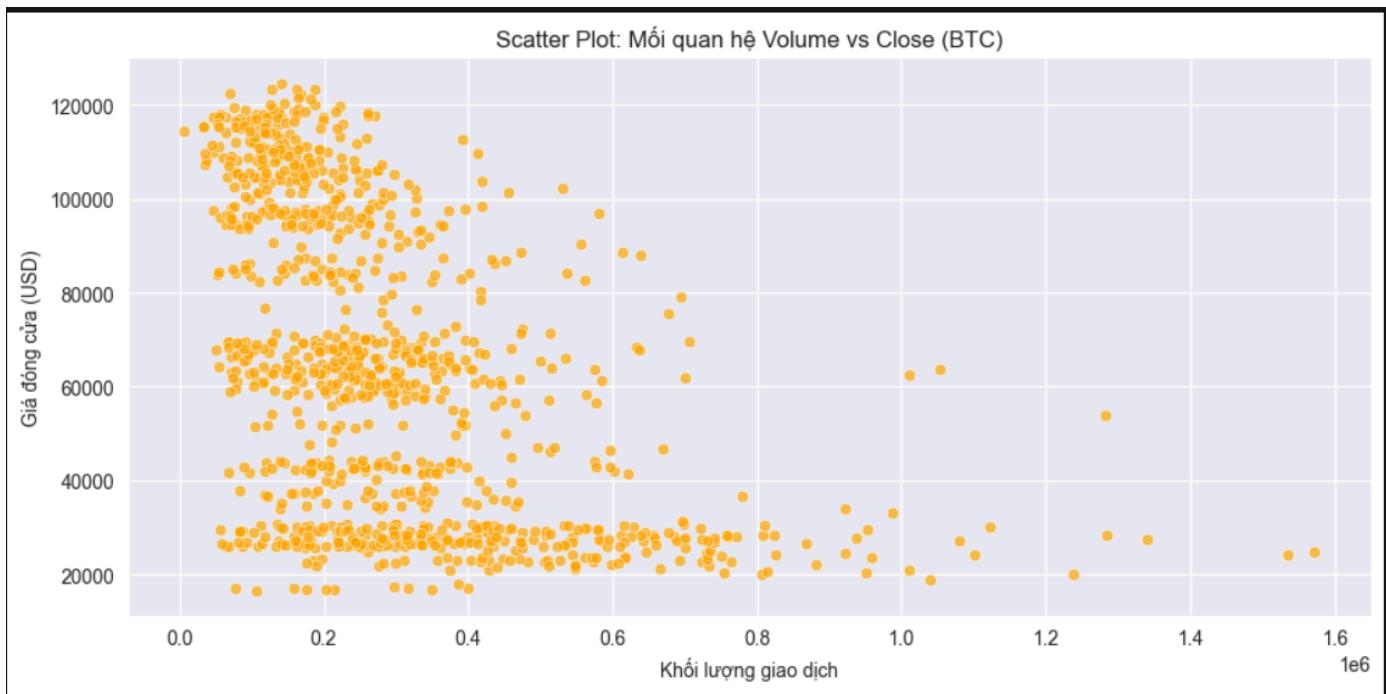
- **Cách đọc:** đám mây điểm **nghiêng lên** → đồng biến; **nghiêng xuống** → nghịch biến; **cụm dày** → vùng hoạt động phổ biến; **điểm lẻ** → outlier/sự kiện.
- **Ứng dụng:** gợi ý thêm feature cho mô hình ($\log(\text{Volume})$, regime label), hoặc kiểm tra phiên “nóng” trước/ sau tin.

```
# =====
# ◆ 6 SCATTER PLOT – mỗi mã riêng
# =====
for sym in df['symbol'].unique():
    sub = df[df['symbol'] == sym]
    plt.figure()
    sns.scatterplot(data=sub, x='Volume', y='Close', alpha=0.7, color='orange')
    plt.title(f"Scatter Plot: Mỗi quan hệ Volume vs Close ({sym})")
    plt.xlabel("Khối lượng giao dịch")
    plt.ylabel("Giá đóng cửa (USD)")
    plt.tight_layout()
    plt.savefig(f"data/{sym}/pic/scatter_vol_close_{sym}.png", dpi=300)
    plt.show()
```

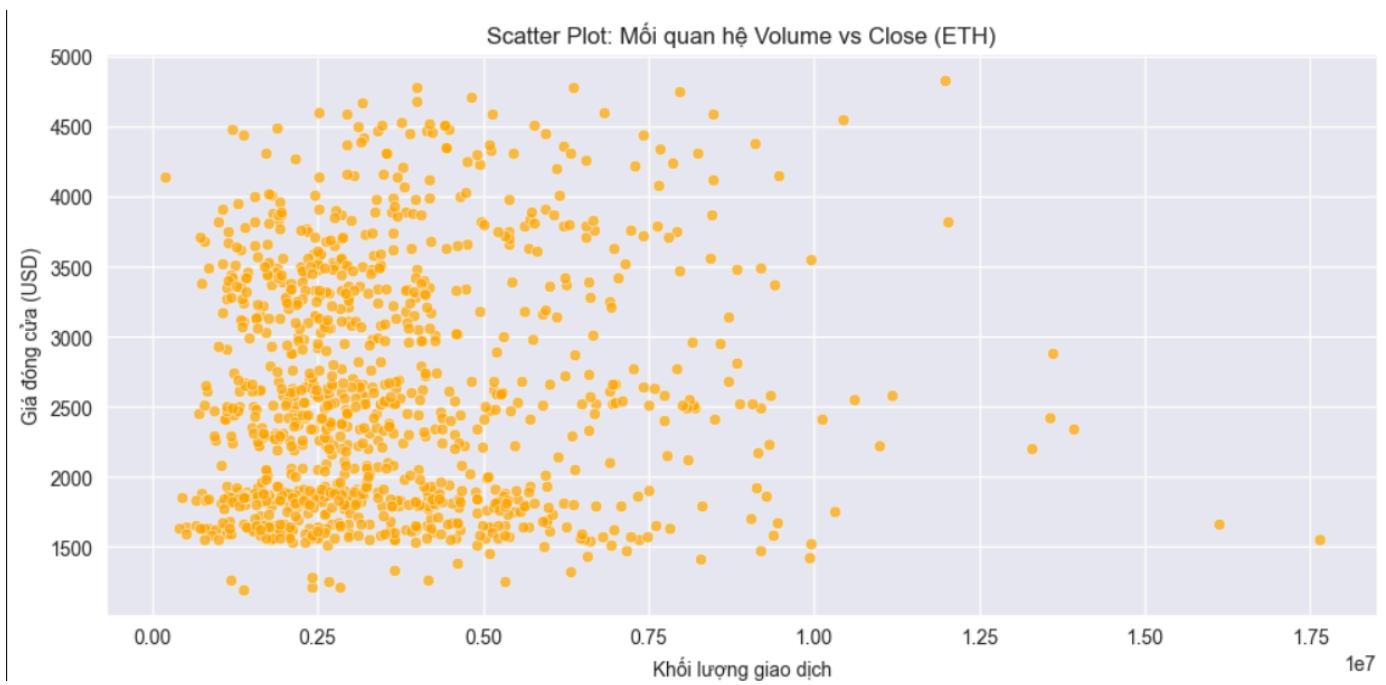
Hình 4.7a — Code vẽ Scatter Plot theo từng mã



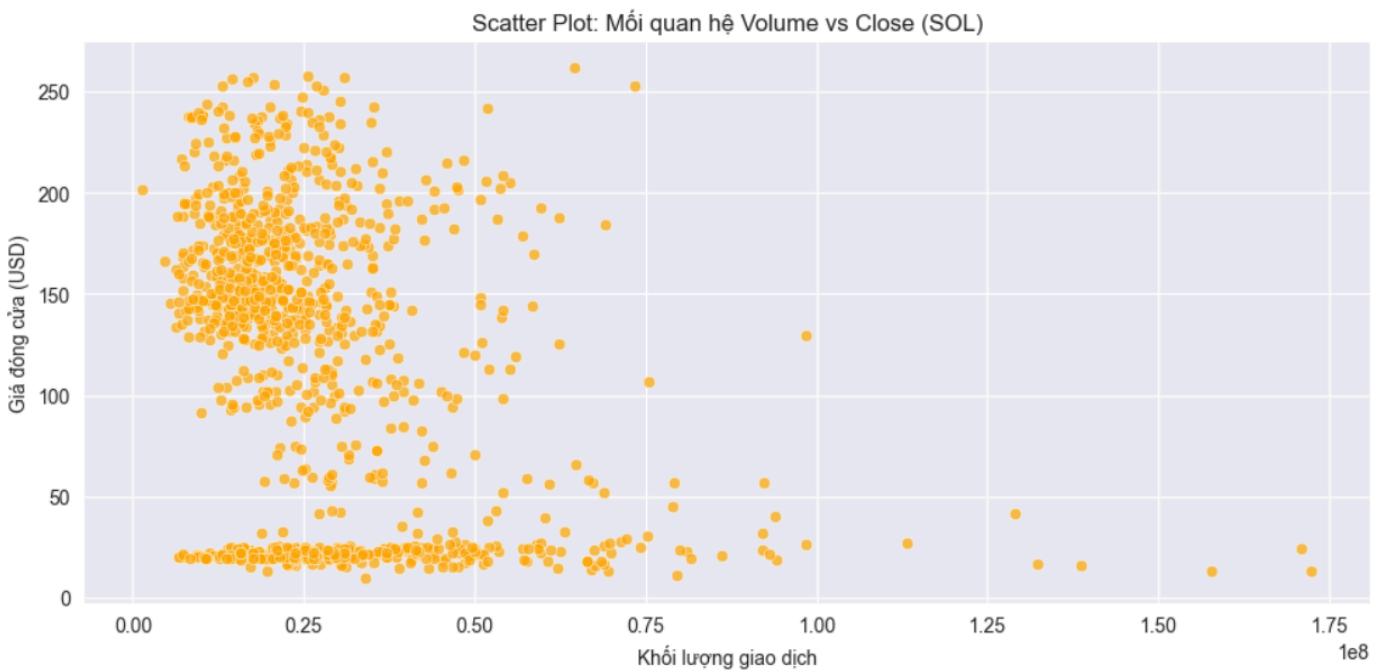
Hình 4.7b — Scatter Plot: Volume vs Close (BNB)



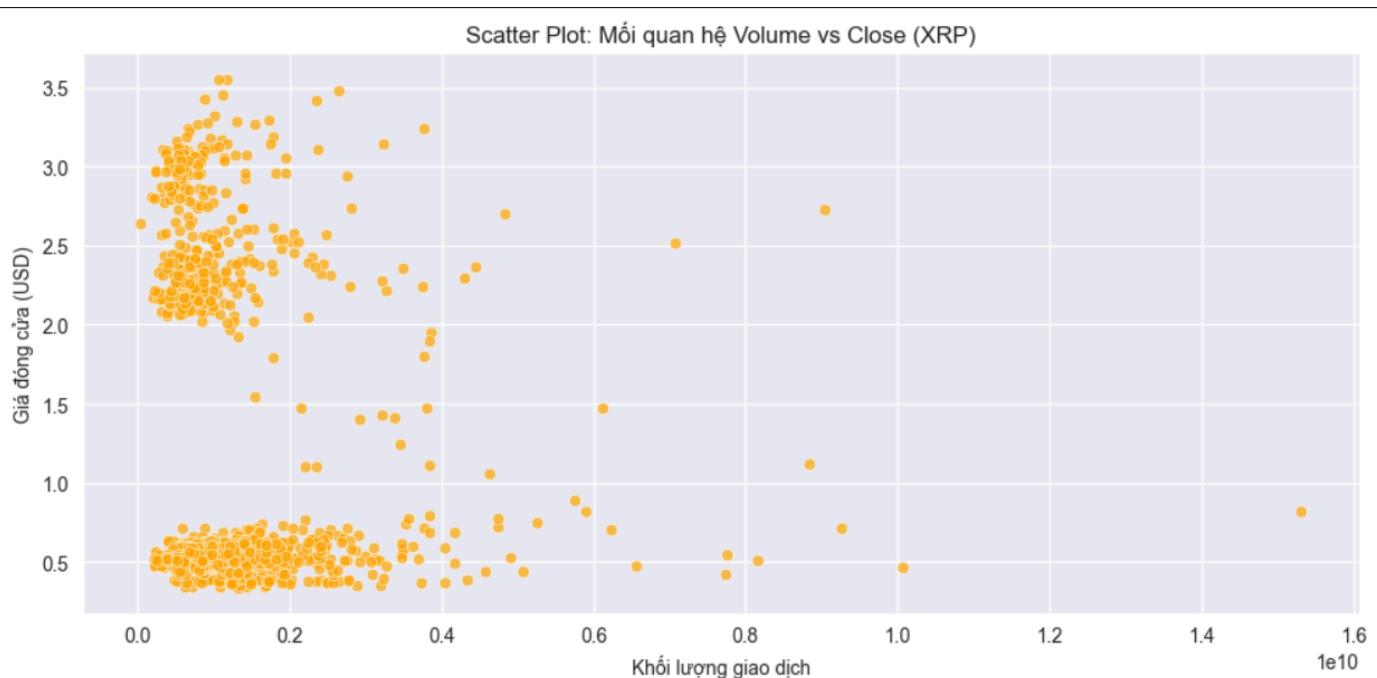
Hình 4.7c — Scatter Plot: Volume vs Close (BTC)



Hình 4.7d — Scatter Plot: Volume vs Close (ETH)



Hình 4.7e — Scatter Plot: Volume vs Close (SOL)



Hình 4.7f — Scatter Plot: Volume vs Close (XRP)

Nhận xét:

- **BTC/ETH:** mây điểm khá ngang → quan hệ tức thời yếu; xuất hiện vài điểm volume cao nhưng giá không tăng tương ứng.
- **SOL/XRP:** có nhiều outlier volume lớn; đám mây trải rộng theo trục y → biến động giá nhạy hơn với thay đổi thanh khoản.
- **BNB:** hai “dải” giá rõ rệt (regime) tại vùng ~300 USD và ~600 USD, gợi ý chuyển pha thị trường.

4.8 Line Chart — Giá đóng cửa theo thời gian

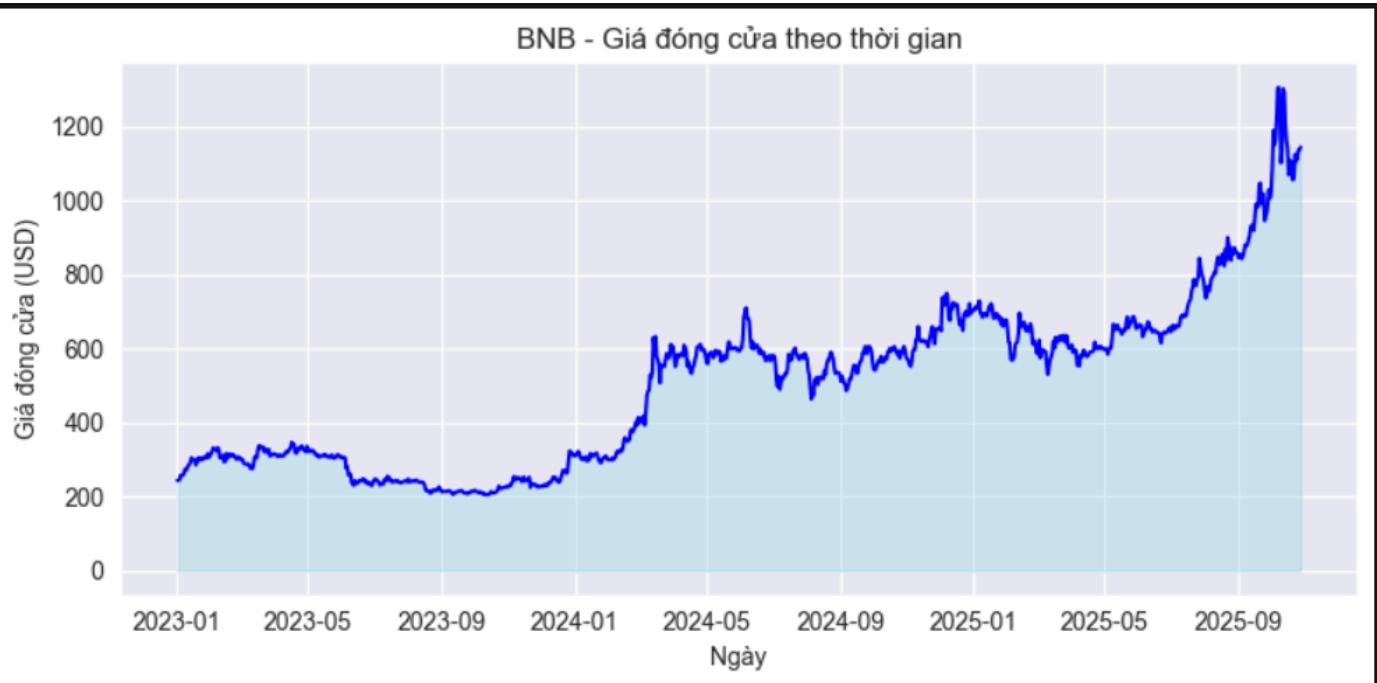
- **Mục tiêu:** theo dõi **diễn biến giá (Close)** theo ngày của từng mã; nhận diện xu hướng, các pha tăng/giảm và vùng chuyển pha.
- **Thiết lập:** `sns.lineplot(x='Date', y='Close')`; tô nền nhẹ bằng `fill_between` để dễ nhìn vùng giá; trục x = Ngày, trục y = USD.
- **Cách đọc:** đường đi lên bên → xu hướng tăng; các nhịp “râu” dài → biến động cao/outlier; vùng phẳng kéo dài → tích lũy.

```
import seaborn as sns
import matplotlib.pyplot as plt
import os

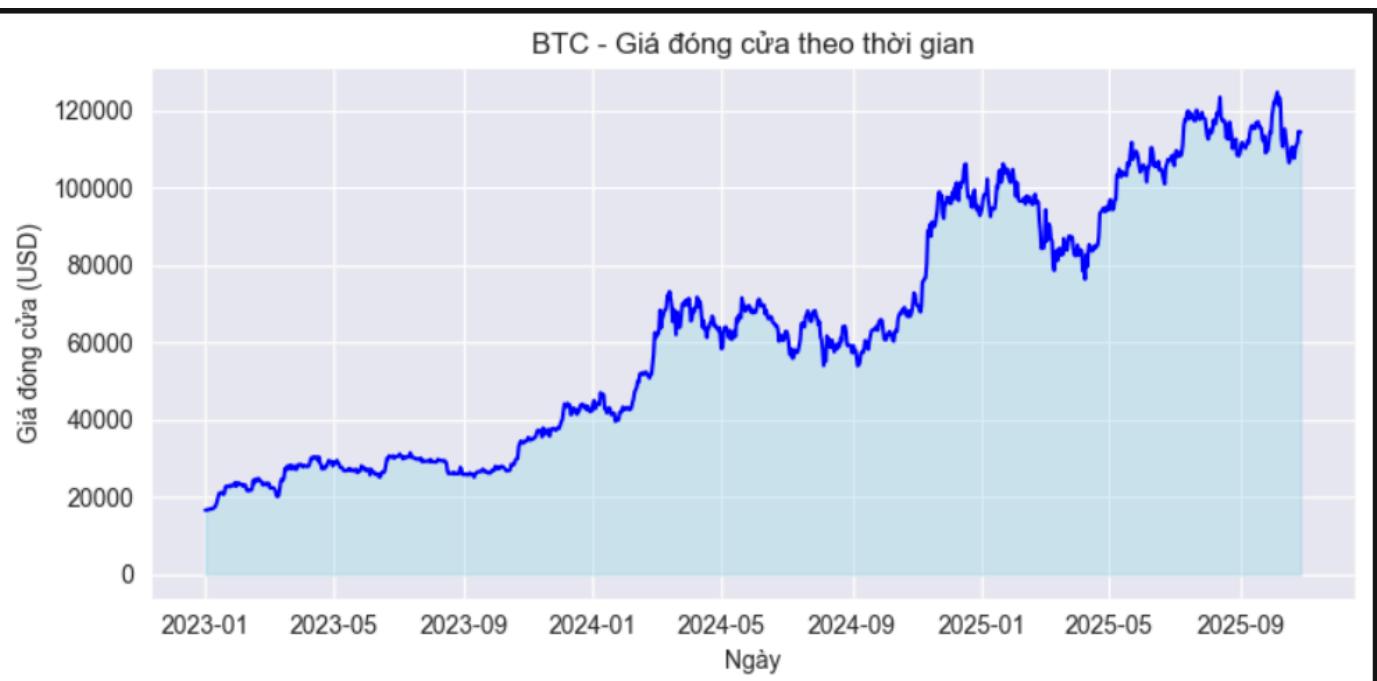
os.makedirs("data/web_pic", exist_ok=True)

for sym in df['symbol'].unique():
    sub = df[df['symbol'] == sym]
    plt.figure(figsize=(8,4))
    sns.lineplot(data=sub, x='Date', y='Close', color='blue')
    plt.fill_between(sub['Date'], sub['Close'], color='skyblue', alpha=0.3)
    plt.title(f"{sym} - Giá đóng cửa theo thời gian")
    plt.xlabel("Ngày")
    plt.ylabel("Giá đóng cửa (USD)")
    plt.tight_layout()
    plt.savefig(f"data/web_pic/{sym}_line_area.png", dpi=150)
    plt.show()
    plt.close()
```

Hình 4.8a — Code vẽ Line + Area nền cho từng mã



Hình 4b. — BNB: Giá đóng cửa theo thời gian



Hình 4.8b — BTC: Giá đóng cửa theo thời gian



Hình 4.8c — ETH: Giá đóng cửa theo thời gian



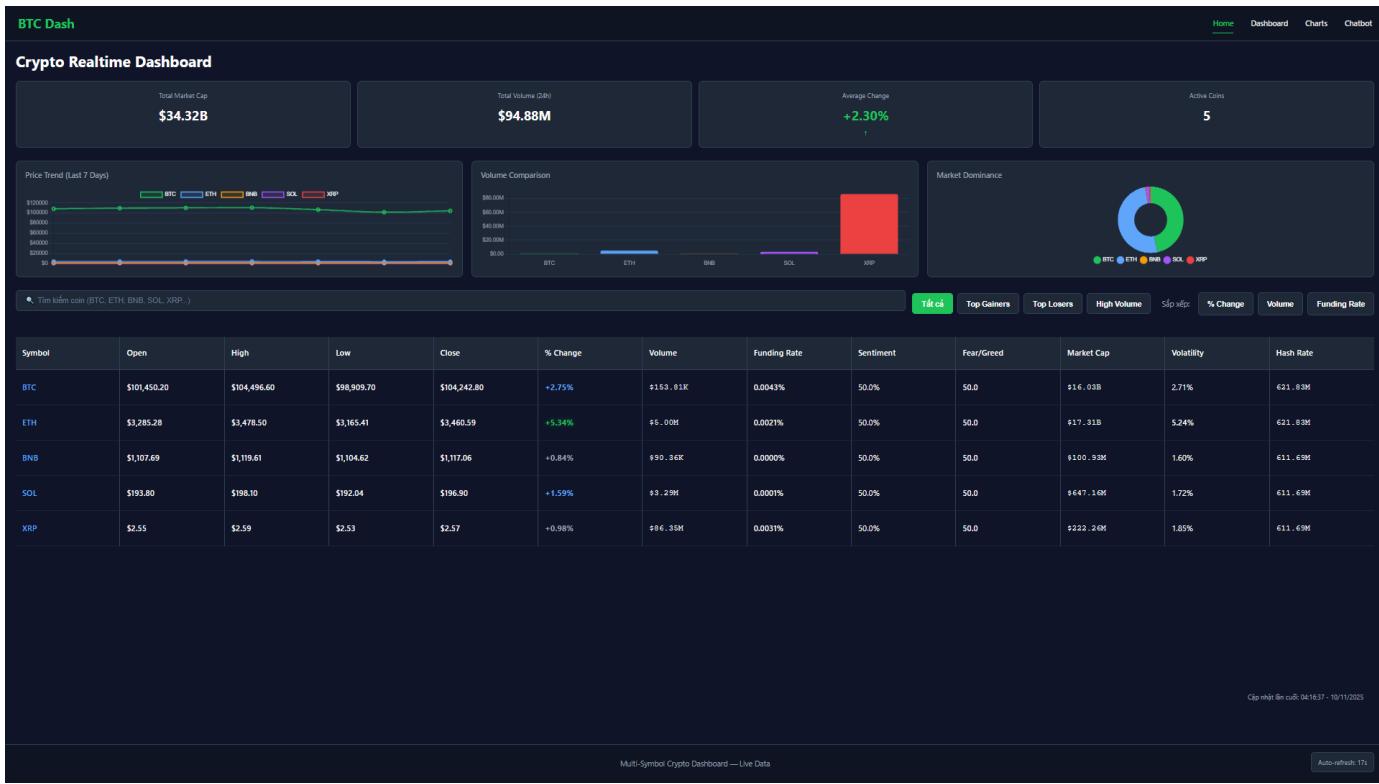
Hình 4.8d — SOL: Giá đóng cửa theo thời gian



Hình 4.8e — XRP: Giá đóng cửa theo thời gian

5. CREATE A WEBSITE

5.1. Trang chính (index.html)



5.1.1 Giới thiệu trang web

Trang **BTC Dashboard** là sản phẩm trực quan hóa dữ liệu (Data Visualization) được phát triển nhằm tổng hợp, phân tích và hiển thị các chỉ tiêu liên quan đến thị trường tiền điện tử.

Hệ thống được xây dựng bằng **HTML/CSS/JavaScript**, kết nối trực tiếp với dữ liệu thu thập và xử lý bằng Python.

5.1.2. Cấu trúc giao diện hiển thị

- **Khu vực tổng quan (Dashboard Overview)**
 - Total Market Cap: Tổng vốn hóa toàn bộ các đồng tiền điện tử được hiển thị
 - Total Volume (24h): Tổng khối lượng giao dịch trong 24 giờ
 - Average Change: Mức biến động trung bình (%) giữa các đồng coin
 - Active Coins: Số lượng đồng tiền đang được theo dõi
- **Khu vực biểu đồ trực quan (Visualization Section)**
 - Price Trend (Last 7 Days): Biểu đồ đường thể hiện diễn biến giá đóng cửa của các đồng trong 7 ngày gần nhất
 - Volume Comparison: Biểu đồ cột so sánh khối lượng giao dịch giữa các đồng
 - Market Dominance: Biểu đồ tròn thể hiện tỷ trọng vốn hóa của từng đồng trong tổng thị trường
- **Bảng dữ liệu chi tiết (Data Table)**

- **Symbol:** Mã ký hiệu của đồng tiền (BTC, ETH, BNB, SOL, XRP, v.v.)
- **Open / High / Low / Close:** Giá mở cửa, cao nhất, thấp nhất và giá đóng cửa trong phiên
- **% Change:** Tỷ lệ thay đổi giá so với ngày trước đó
- **Volume:** Tổng khối lượng giao dịch trong ngày
- **Funding Rate:** Lãi suất tài trợ trên thị trường Futures (tích cực hoặc tiêu cực)
- **Sentiment:** Chỉ số cảm xúc thị trường (Bullish / Neutral / Bearish)
- **Fear & Greed:** Chỉ số tâm lý nhà đầu tư (0-100)
- **Market Cap:** Vốn hóa thị trường ước tính
- **Volatility:** Mức độ biến động giá (phần trăm)
- **Hash Rate:** Sức mạnh xử lý mạng lưới blockchain (đặc biệt với BTC)
- **Khu vực tương tác người dùng (User Interaction Tools)**

5.1.3. Tính năng tổng hợp và hiển thị

- Tự động cập nhật dữ liệu từ các file .csv được pipeline Python xuất định kỳ.
- Tự động gộp dữ liệu từ nhiều đồng coin vào một file tổng (dataset_total_clean.csv).
- Hiển thị biểu đồ trực quan và tương tác (realtime chart, mini-chart khi hover).

5.2. Trang thống kê (Dashboard.html)



5.2.1. Mục tiêu và ý tưởng thiết kế

Trang **Dashboard** được xây dựng nhằm cung cấp cái nhìn chuyên sâu về từng đồng tiền điện tử thông qua ba yếu tố chính:

1. **Biến động giá (Price Movement)**
2. **Khối lượng giao dịch (Volume)**
3. **Độ biến động giá (Volatility)**

Người dùng có thể chủ động chọn **đồng tiền (symbol)** và **khoảng thời gian (time range)** để xem các biểu đồ và chỉ số tổng hợp tương ứng.

Giao diện được thiết kế với tông màu tối, sử dụng **Tailwind CSS** và **Chart.js**, giúp hiển thị rõ ràng, hiện đại và dễ đọc ngay cả với dữ liệu phức tạp.

5.2.2. Các Thành phần

- Khu vực lựa chọn và điều hướng
- Khu vực thống kê tóm tắt (Summary Statistics)
- Hệ thống biểu đồ phân tích
- Phương pháp tính toán trong Dashboard
 - **ma7, ma30:** Tính trung bình động 7 và 30 ngày
 - **sevenDayReturn:** Tính lợi nhuận 7 ngày

$$(Close_t - Close_{t-7}) / Close_{t-7} \times 100$$

- **volatility:** Độ lệch chuẩn chuỗi giá đóng cửa

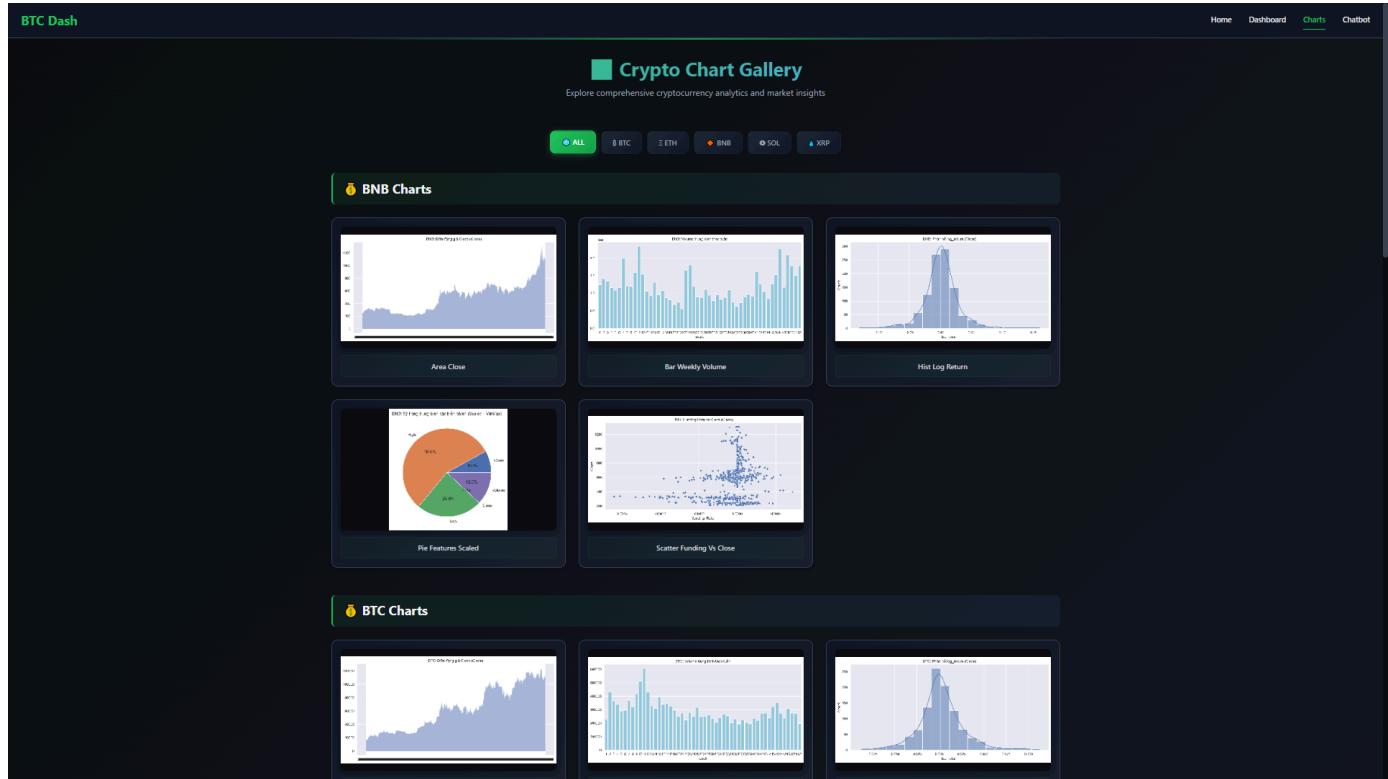
$$\sqrt{\sum(x - \mu)^2 / n}$$

5.2.3. Giá trị Ứng dụng

Trang Dashboard giúp:

1. **Phân tích chuyên sâu từng đồng tiền** qua biểu đồ giá, khối lượng và biến động.
2. **So sánh giữa nhiều khoảng thời gian** để nhận diện xu hướng.
3. **Kết hợp dữ liệu on-chain và tâm lý thị trường** (qua file tổng) → hỗ trợ cho nhà đầu tư và nghiên cứu thị trường crypto.
4. Là **nền tảng trình diễn kỹ thuật trực quan hóa dữ liệu (Data Visualization)** – phần lõi trong dự án *BTC Dash*.

5.3. Trang biểu đồ (Chart.html)



5.3.1. Mục tiêu

Trang **Chart** được thiết kế nhằm hiển thị **toàn bộ các biểu đồ phân tích và trực quan hóa dữ liệu** (output của pipeline Python) theo từng đồng tiền điện tử.

Trang đóng vai trò như "**gallery tổng hợp**" giúp người dùng xem nhanh toàn bộ biểu đồ đã được sinh ra trong quá trình phân tích.

5.3.2. Cấu trúc hiển thị

- **Header Navigation:** Thanh điều hướng liên kết đến các trang khác của hệ thống: Home, Dashboard, Chart, Chatbot.
- **Hero Section:** Tiêu đề lớn “ Crypto Chart Gallery” và mô tả “Explore comprehensive cryptocurrency analytics and market insights”.
- **Filter Buttons:** Các nút lọc theo đồng coin: ALL, BTC, ETH, BNB, SOL, XRP. Khi nhấn, chỉ hiển thị nhóm biểu đồ tương ứng.
- **Chart Gallery:** Vùng hiển thị hình ảnh biểu đồ (theo từng đồng). Dữ liệu lấy từ tệp data/picture/manifest.json – nơi chứa danh sách các hình biểu đồ được tạo tự động.
- **Modal Viewer:** Khi người dùng click vào ảnh, ảnh phóng to toàn màn hình với hiệu ứng làm mờ nền (blur overlay).
- **Footer:** Hiển thị thông tin tác giả và kết thúc trang.

5.3.3. Cách thức hoạt động

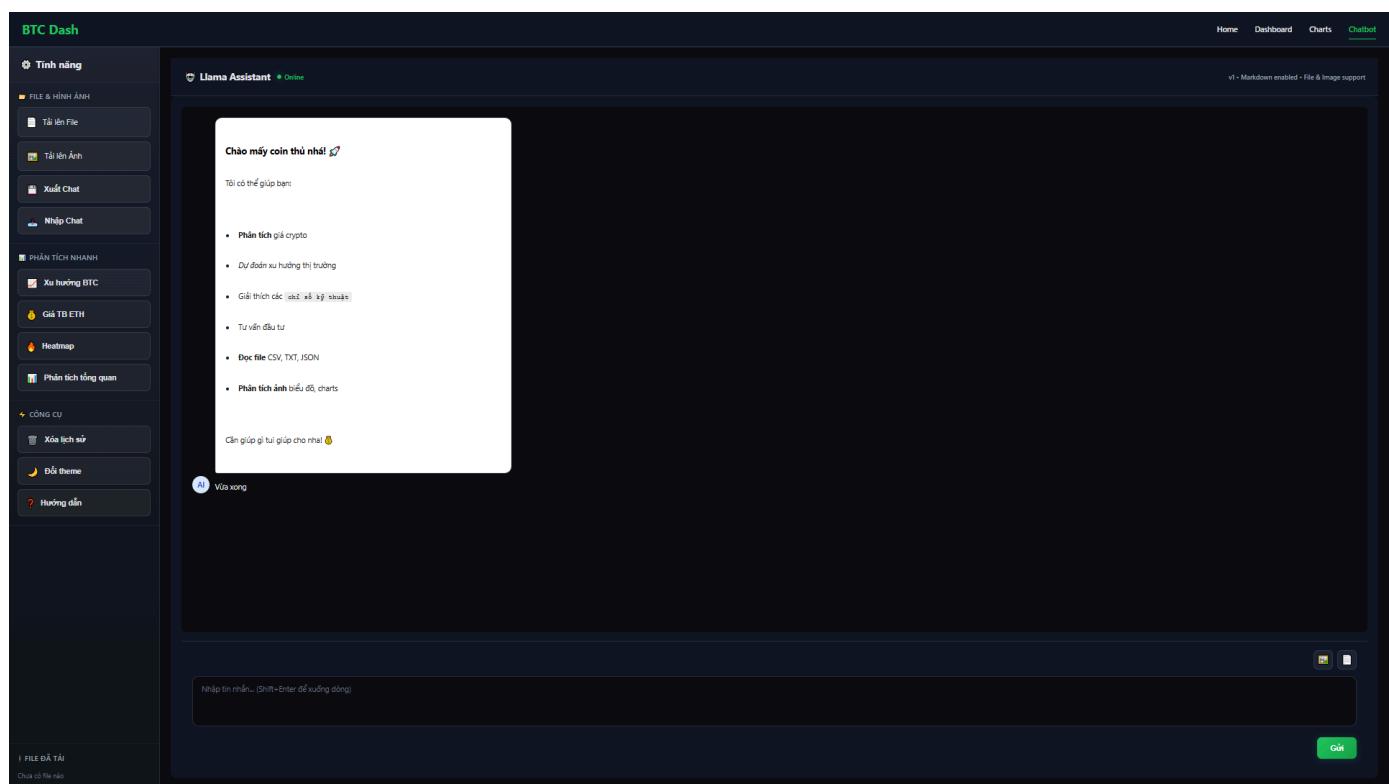
1. **JavaScript** tải tệp `manifest.json` từ thư mục `/data/picture/`.
Tệp này chứa danh sách các biểu đồ (PNG/JPG) được tạo bởi phần phân tích dữ liệu.
2. Sau khi tải thành công, script tự động tạo các phần (section) cho từng coin (**BTC**, **ETH**, **BNB**, **SOL**, **XRP**).
3. Với mỗi coin, tất cả biểu đồ trong `charts` được hiển thị dưới dạng **thẻ hình ảnh + mô tả** (`figcaption`).
4. Người dùng có thể:
 - o Lọc theo coin bằng các nút phía trên.
 - o Click để xem chi tiết ảnh lớn.
 - o Dùng phím **ESC** để thoát chế độ xem ảnh.

5.3.4. Nhận xét

Trang **Chart** giúp:

- Tổng hợp toàn bộ **biểu đồ phân tích** từ quá trình EDA, so sánh và dự báo.
- Cung cấp giao diện **xem nhanh và trực quan** cho các biểu đồ của từng coin.
- Là phần **“trưng bày kết quả”** của dự án BTC Dash – minh họa năng lực xử lý dữ liệu và trực quan hóa đa chiều.

5.4. Trang agent (chatbot.html)



5.4.1. Mục tiêu

Trang **Chatbot** được phát triển nhằm hỗ trợ người dùng tương tác trực tiếp với hệ thống AI phân tích dữ liệu tiền điện tử.

Mục tiêu chính là giúp người dùng:

- Tra cứu nhanh thông tin về giá, khối lượng, funding rate, biểu đồ.
- Gửi câu hỏi tự nhiên bằng tiếng Việt.
- Nhận phân tích, hình ảnh biểu đồ hoặc kết quả thống kê từ các file dữ liệu của dự án (*dataset, manifest.json*).

Trang đóng vai trò là **giao diện AI tương tác thông minh** cho toàn hệ thống BTC Dash.

5.4.2. Cấu trúc hiển thị

- **Header Navigation:** Thanh điều hướng liên kết đến các trang: Home, Dashboard, Chart, Chatbot.
- **Sidebar (Thanh công cụ):** Chứa các nhóm tính năng: tải lên file/ảnh, phân tích nhanh (quick ask), công cụ hỗ trợ, danh sách file đã tải.
- **Chat Panel (Khung chat chính):** Hiển thị toàn bộ cuộc hội thoại giữa người dùng và AI (tin nhắn, ảnh, file).
- **Chat Input (Khung nhập liệu):** Hỗ trợ gửi tin nhắn, tải tệp hoặc hình ảnh kèm theo.
- **Typing Indicator:** Hiệu ứng hiển thị khi AI đang xử lý và phản hồi.
- **Status Header:** Hiển thị trạng thái hoạt động “Online” và phiên bản mô hình Llama hiện tại.

5.3.3. Cách thức hoạt động

- JavaScript kết nối trực tiếp đến **API Flask / ngrok backend** qua địa chỉ **BACKEND_URL**.
- Khi người dùng gửi tin nhắn:
 1. Hệ thống **xác định ý định truy vấn (intent detection)**: dữ liệu, biểu đồ hay phân tích.
 2. Nếu yêu cầu dữ liệu: chatbot đọc các file `.csv` trong thư mục `/data/{symbol}/` để tính toán chỉ tiêu (`avg_close, volume, funding_Rate`, v.v.).
 3. Nếu yêu cầu biểu đồ: chatbot đọc đường dẫn từ `manifest.json` và hiển thị hình ảnh tương ứng.
 4. Nếu yêu cầu phân tích tổng quan: chatbot trả về đoạn văn bản Markdown giải thích xu hướng thị trường.
- Lịch sử hội thoại được **lưu tự động bằng localStorage**, hỗ trợ **xuất / nhập file JSON** để sao lưu hoặc phục hồi.
- Cho phép tải lên file `.csv, .txt, .json, .pdf, .docx` hoặc ảnh để AI xử lý và phân tích nội dung.

5.3.4. Nhận xét

Trang Chatbot giúp:

- Biến **phân tích dữ liệu** thành trải nghiệm **hỏi – đáp tự nhiên**, tăng khả năng tiếp cận của người dùng.
- Tích hợp **AI (Llama/Ollama)** để diễn giải dữ liệu từ hệ thống pipeline.
- Kết nối các phần dữ liệu, biểu đồ và file trong dự án thành một **nền tảng tư vấn crypto toàn diện**.
- Là điểm nổi bật của dự án BTC Dash, thể hiện khả năng kết hợp **AI – Data – Visualization** trong cùng một sản phẩm.

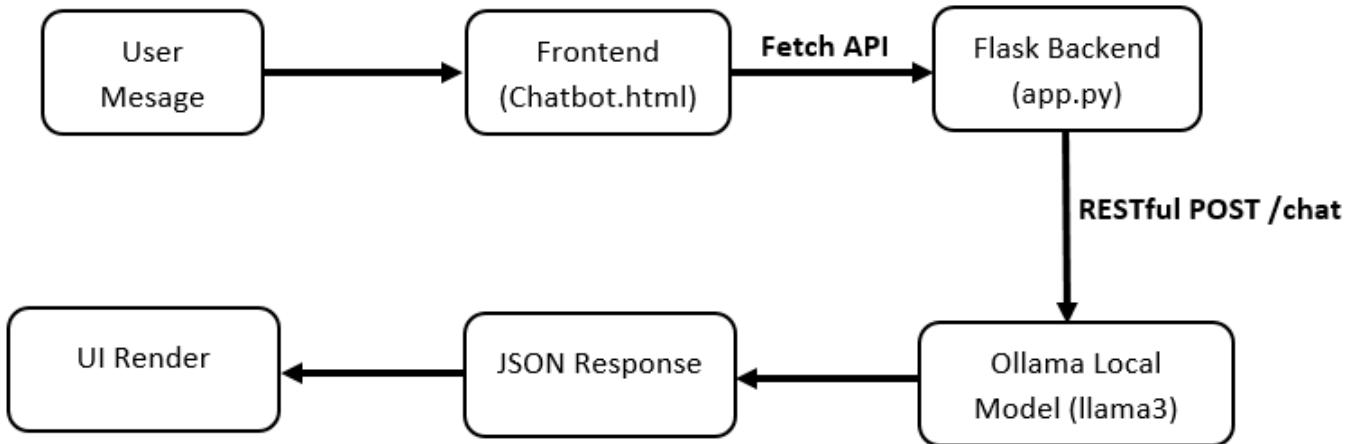
6. Chatbot: Create and Integrate

6.1. Mục tiêu

Phần chatbot được xây dựng nhằm cung cấp khả năng tương tác tự động giữa người dùng và hệ thống phân tích dữ liệu tiền mã hóa. Chatbot có nhiệm vụ tiếp nhận câu hỏi từ người dùng, gửi yêu cầu đến mô hình ngôn ngữ lớn để xử lý, sau đó phản hồi kết quả bằng ngôn ngữ tự nhiên. Mục tiêu của việc tích hợp chatbot là giúp người dùng có thể truy xuất thông tin, nhận giải thích và gợi ý phân tích từ mô hình AI dựa trên dữ liệu đã thu thập và xử lý trong các bước trước.

6.2. Kiến trúc hệ thống

Hệ thống chatbot được triển khai dựa trên kiến trúc ba lớp chính: giao diện web, backend Flask và mô hình Llama. Dữ liệu hội thoại được truyền qua giao thức HTTP POST, trong đó Flask đóng vai trò trung gian kết nối giữa người dùng và mô hình ngôn ngữ.



Giải thích luồng hoạt động:

1. Người dùng nhập câu hỏi trên giao diện web (`chatbot.html`).
2. JavaScript gửi POST request đến Flask server (`/chat`).
3. Flask gửi câu hỏi đến mô hình **Ollama / LLaMA3**.
4. Ollama tạo phản hồi (AI reply) → Flask ở định dạng JSON.
5. Frontend hiển thị phản hồi và biểu đồ nếu cần.

6.3. Triển khai

Phần backend được cài đặt bằng Flask, toàn bộ mã nguồn nằm trong file `app.py`. Ứng dụng định nghĩa hai route chính: route gốc `/` để kiểm tra hoạt động và route `/chat` để xử lý hội thoại. Đoạn mã dưới đây mô tả cách Flask nhận yêu cầu từ frontend, gửi đến **LLaMA3** và phản hồi kết quả:

```

from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

OLLAMA_URL = "http://localhost:11434/v1/chat/completions"

@app.route("/")
def home():
    return "✅ Flask server is running!"

@app.route("/chat", methods=["POST"])
def chat():
    data = request.get_json()
    user_message = data.get("message", "")

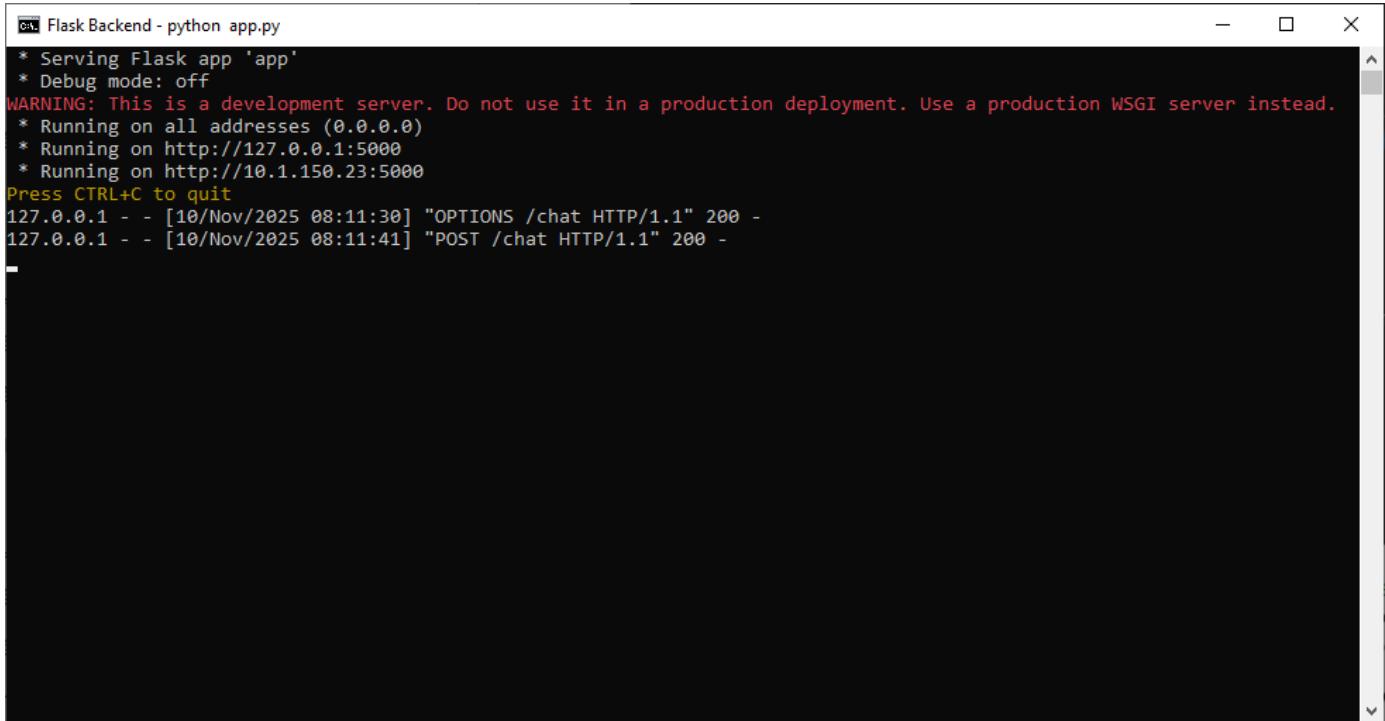
    # Payload gửi tới ollama API
    payload = {
        "model": "llama3", # đổi thành model bạn có, ví dụ Llama3
        "messages": [{"role": "user", "content": user_message}]
    }

    try:
        response = requests.post(OLLAMA_URL, json=payload)
        response_json = response.json()
        reply = response_json["choices"][0]["message"]["content"]
        return jsonify({"reply": reply})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(port=5000, host="0.0.0.0")

```

Trong đoạn mã trên, Flask nhận dữ liệu từ người dùng qua trường "message", gửi payload JSON đến địa chỉ của mô hình Ollama, sau đó lấy phản hồi từ trường "choices" → "message" → "content" và trả lại dưới dạng JSON cho frontend.



```
Flask Backend - python app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.1.150.23:5000
Press CTRL+C to quit
127.0.0.1 - - [10/Nov/2025 08:11:30] "OPTIONS /chat HTTP/1.1" 200 -
127.0.0.1 - - [10/Nov/2025 08:11:41] "POST /chat HTTP/1.1" 200 -
```

6.4. Tích hợp giao diện

Chatbot được tích hợp trực tiếp vào giao diện web, hỗ trợ hai hình thức hoạt động: phiên bản HTML tĩnh và phiên bản Next.js.

a. Phiên bản HTML:

Người dùng mở trực tiếp file `frontend-vanilla/index.html` trong trình duyệt. Khi gửi tin nhắn, script trong giao diện sẽ gửi yêu cầu đến API Flask qua đường dẫn được cấu hình trong biến `API_BASE`. Toàn bộ quá trình chạy thử có thể khởi động bằng lệnh trong file `start-server.sh`:

```
#!/bin/bash
echo "Starting local web server on port 8000..."
python3 -m http.server 8000
```

b. Phiên bản Next.js:

Hệ thống được triển khai bằng React và TypeScript. Cấu hình API backend được định nghĩa trong file `.env.local.example` như sau:

```
NEXT_PUBLIC_BACKEND_BASE=https://your-ngrok-url.ngrok-free.app
```

Khi người dùng nhập tin nhắn, frontend gửi yêu cầu đến Flask API thông qua endpoint này, sau đó hiển thị phản hồi trong giao diện chat.

Coin Dash

Home Dashboard Charts Chatbot

FILE & HÌNH ẢNH

- Tải lên File
- Tải lên Ảnh
- Xuất Chat
- Nhập Chat

PHÂN TÍCH NHANH

- Xu hướng BTC
- Giá TB ETH
- Heatmap
- Phân tích tổng quan

CÔNG CỤ

- Xóa lịch sử
- Đổi theme
- Hướng dẫn

FILE ĐÃ TẢI

Chưa có file nào

Llama Assistant • Online

v1 • Markdown enabled • File & Image support

Chào mấy coin thù nhá! 🎉

Tôi có thể giúp bạn:

- Phân tích giá crypto
- Dự đoán xu hướng thị trường
- Giải thích các chỉ số kỹ thuật
- Tư vấn đầu tư

Cần giúp gì tui giúp cho nhá! 😊

AI 06:11

Hello AI You

AI 08:11

AI: Nice to meet you! I'm here to help you analyze some data and provide insightful results. What kind of data analysis would you like me to assist with?

Nhập tin nhắn... (Shift+Enter để xuống dòng)

Gửi

The screenshot shows the Coin Dash application's interface. On the left, there's a sidebar with various tools: FILE & HÌNH ẢNH (File & Image), PHÂN TÍCH NHANH (Quick Analysis) with sub-options like Xu hướng BTC, GIÁ TB ETH, Heatmap, and Phân tích tổng quan, and CÔNG CỤ (Tools) with Xóa lịch sử, Đổi theme, and Hướng dẫn (Help). Below these are sections for FILE ĐÃ TẢI (Uploaded Files) and a note that says Chưa có file nào (No files uploaded). The main area is titled "Llama Assistant • Online". It displays a welcome message from the AI, a list of services it can provide, and a conversation log. The AI asks what kind of data analysis the user wants, and the user responds with "Hello AI". The AI also has a text input field at the bottom with placeholder text "Nhập tin nhắn... (Shift+Enter để xuống dòng)" and a green "Gửi" (Send) button.

```

from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

OLLAMA_URL = "http://localhost:11434/v1/chat/completions"

@app.route("/")
def home():
    return "✅ Flask server is running!"

@app.route("/chat", methods=["POST"])
def chat():
    data = request.get_json()
    user_message = data.get("message", "")

    # Payload gửi tới ollama API
    payload = {
        "model": "llama3", # đổi thành model bạn có, ví dụ llama3
        "messages": [{"role": "user", "content": user_message}]
    }

    try:
        response = requests.post(OLLAMA_URL, json=payload)
        response_json = response.json()
        reply = response_json["choices"][0]["message"]["content"]
        return jsonify({"reply": reply})
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == "__main__":
    app.run(port=5000, host="0.0.0.0")

```