**Bài tập** Các vấn đề Tiền xử lý dữ liệu

**Dẫn nhập**

**Dữ liệu thô/gốc**

* Có cấu trúc, bán cấu trúc, phi cấu trúc
* Được đưa vào từ các nguồn dữ liệu trong các hệ thống xử lý tập tin (file processing systems) và/hay

các hệ thống cơ sở dữ liệu (database systems)

**Chất lượng dữ liệu** (data quality): tính chính xác, tính hiện hành, tính toàn vẹn, tính nhất quán

**A. Tóm tắt mô tả về dữ liệu**

1. Xác định các thuộc tính (properties) tiêu biểu của dữ liệu về xu hướng chính (central tendency) và sự phân tán (dispersion) của dữ liệu

1.1. Các độ đo về xu hướng chính: mean, median, mode, midrange

Bài tập Các độ đo về xu hướng chính (Central Tendency)

a. Dữ liệu mẫu

Giả sử ta có tập dữ liệu về Thu nhập hàng tháng (triệu VND) của 10 nhân viên:

[5, 7, 8, 6, 10, 12, 7, 9, 15, 100]

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Cần tìm xu hướng trung tâm của dữ liệu:

• Mean (trung bình cộng)

• Median (trung vị)

• Mode (mốt)

• Midrange (trung điểm)

- Lưu ý: dữ liệu có một giá trị ngoại lai (outlier) = 100, có thể ảnh hưởng lớn đến mean.

c. Xử lý bằng tay

1. Mean:

Mean = (5+7+8+6+10+12+7+9+15+100) / 10 = 179 / 10 = 17.9

2. Median:

Dữ liệu sắp xếp: [5, 6, 7, 7, 8, 9, 10, 12, 15, 100]

Số phần tử chẵn (10) → Median = (8+9)/2 = 8.5

3. Mode:

Giá trị xuất hiện nhiều nhất: 7 (xuất hiện 2 lần) → Mode = 7

4. Midrange:

Midrange = (Max + Min)/2 = (100 + 5)/2 = 52.5

d. Giá trị sau xử lý

- Mean: 17.9

- Median: 8.5

- Mode: 7

- Midrange: 52.5

Nhận xét: Mean và Midrange bị ảnh hưởng mạnh bởi outlier = 100, trong khi Median và Mode phản ánh dữ liệu trung tâm tốt hơn.

e. Code Python để hiện thực xử lý

import numpy as np

import statistics as stats

# Dữ liệu mẫu

data = [5, 7, 8, 6, 10, 12, 7, 9, 15, 100]

# Mean

mean\_val = np.mean(data)

# Median

median\_val = np.median(data)

# Mode

mode\_val = stats.mode(data)

# Midrange

midrange\_val = (min(data) + max(data)) / 2

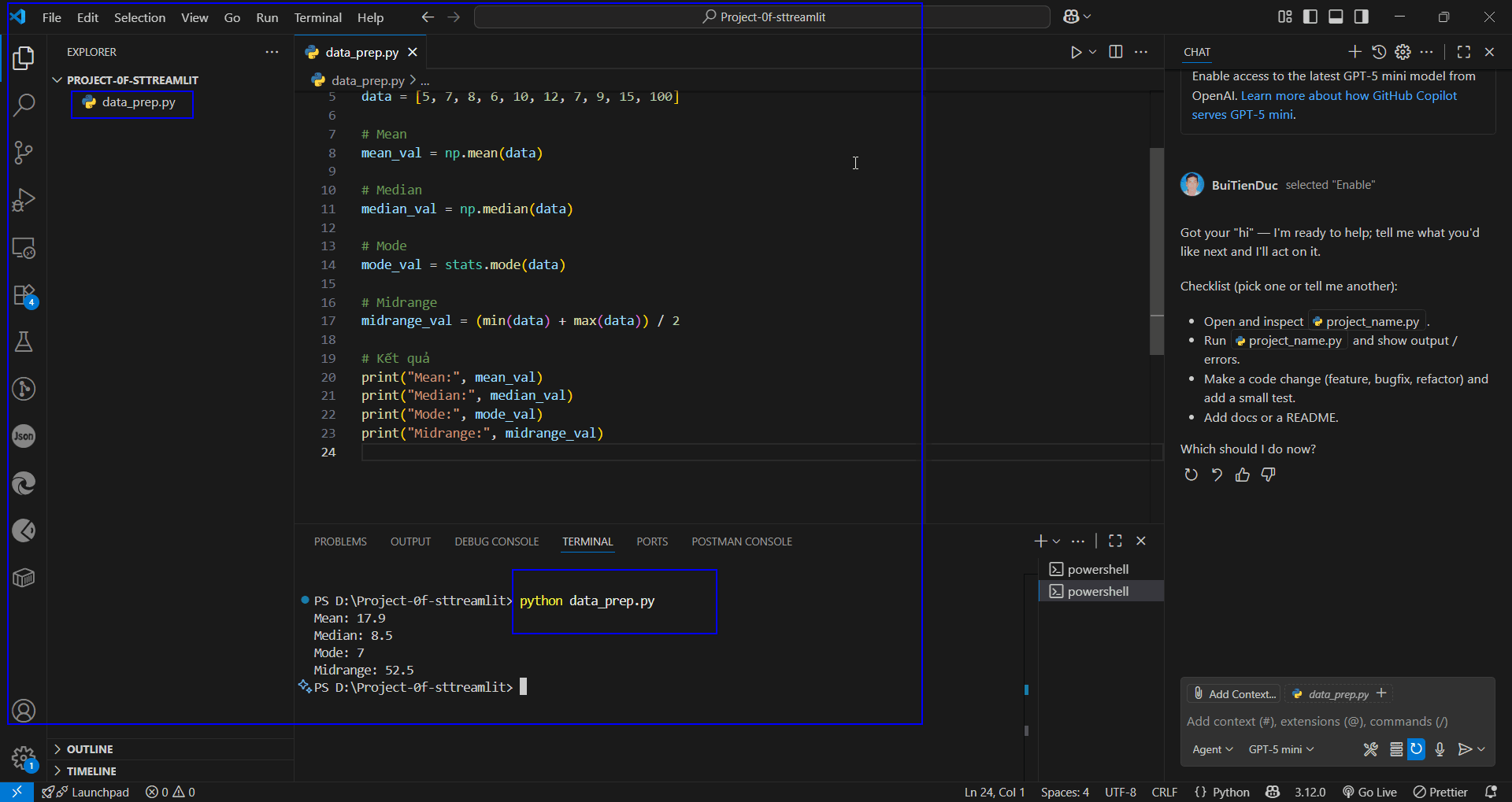
# Kết quả

print("Mean:", mean\_val)

print("Median:", median\_val)

print("Mode:", mode\_val)

print("Midrange:", midrange\_val)



1.2. Các độ đo về sự phân tán: quartiles, interquartile range (IQR), variance

Bài tập Các độ đo về sự phân tán (Dispersion Measures)

a. Dữ liệu mẫu

Giả sử ta có dữ liệu về số giờ học mỗi tuần của 12 sinh viên:

[5, 6, 8, 7, 6, 9, 12, 15, 7, 8, 20, 25]

b. Định nghĩa vấn đề trong dữ liệu mẫu

Cần đo lường mức độ phân tán (dispersion) của dữ liệu, thay vì chỉ xem xét trung tâm:

- Quartiles (Q1, Q2, Q3): chia dữ liệu thành 4 phần bằng nhau.

- Interquartile Range (IQR): đo độ trải rộng trung tâm, công thức: IQR = Q3 - Q1.

- Variance (phương sai): trung bình bình phương khoảng cách tới mean, công thức: σ² = Σ(xi - μ)² / n.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Sắp xếp dữ liệu:

[5, 6, 6, 7, 7, 8, 8, 9, 12, 15, 20, 25]

2. Quartiles:

- Q1 = (6+7)/2 = 6.5

- Q2 (Median) = (8+8)/2 = 8

- Q3 = (12+15)/2 = 13.5

3. IQR = Q3 - Q1 = 13.5 - 6.5 = 7

4. Mean = 128/12 ≈ 10.67

5. Variance: Tính Σ(xi - μ)² = 432.66 → Variance = 432.66/12 = 36.06

d. Giá trị sau xử lý vấn đề trong dữ liệu mẫu

- Q1 = 6.5

- Q2 (Median) = 8

- Q3 = 13.5

- IQR = 7

- Variance = 36.06

Nhận xét: Dữ liệu có độ phân tán lớn do tồn tại giá trị ngoại lai (20, 25).

e. Code Python để hiện thực xử lý

import numpy as np

# Dữ liệu mẫu

data = [5, 6, 8, 7, 6, 9, 12, 15, 7, 8, 20, 25]

# Quartiles

Q1 = np.percentile(data, 25)

Q2 = np.median(data)

Q3 = np.percentile(data, 75)

# IQR

IQR = Q3 - Q1

# Variance

variance = np.var(data)

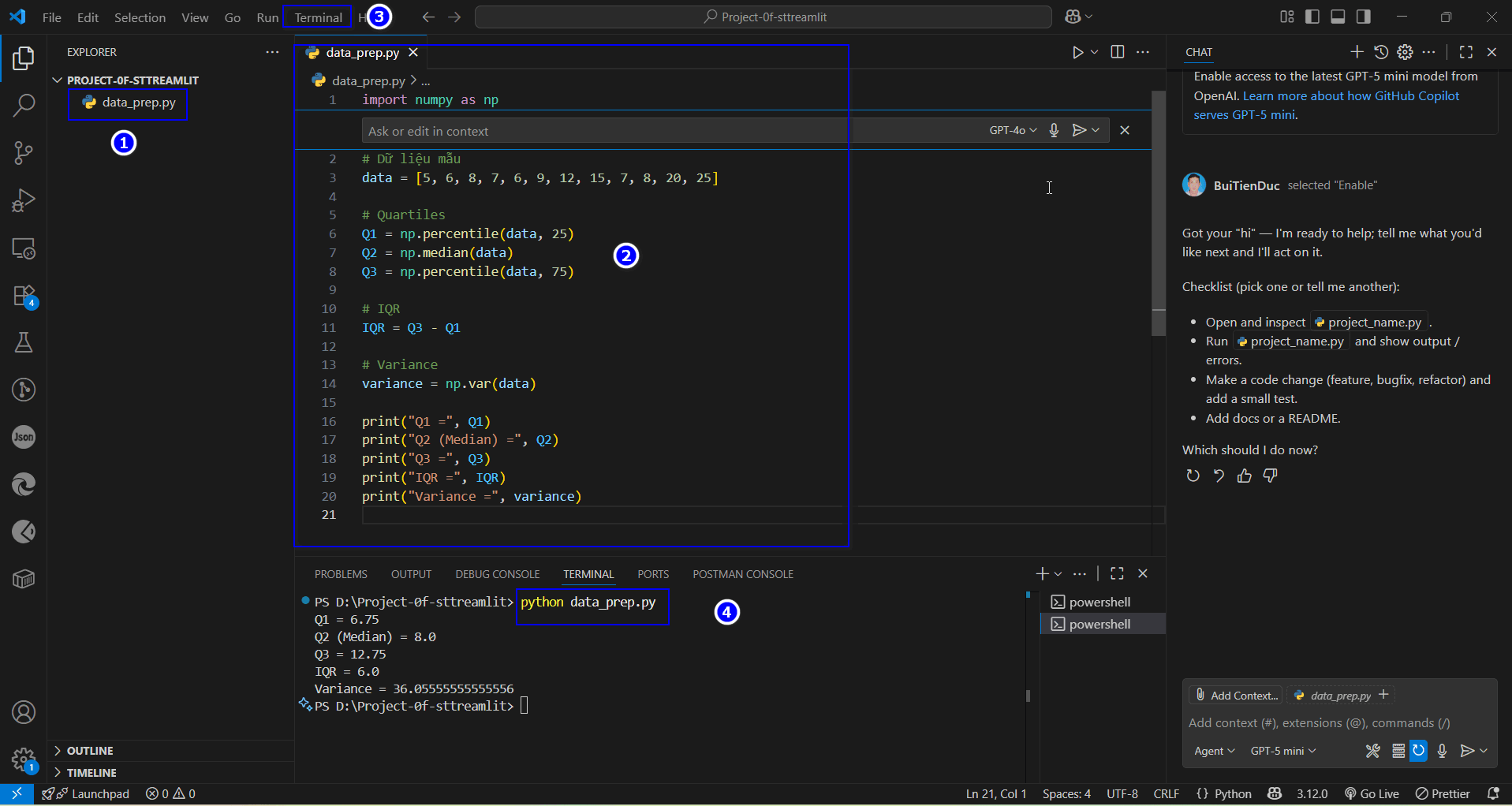
print("Q1 =", Q1)

print("Q2 (Median) =", Q2)

print("Q3 =", Q3)

print("IQR =", IQR)

print("Variance =", variance)



2. Làm nổi bật các giá trị dữ liệu nên được xem như nhiễu (noise) hoặc phần tử biên (outliers), cung cấp cái nhìn tổng quan về dữ liệu

Bài tập Nhận diện nhiễu và phần tử biên (Outliers & Noise Detection)

a. Dữ liệu mẫu

Giả sử ta có dữ liệu điểm kiểm tra (thang 100) của một lớp học:

[45, 50, 52, 48, 47, 51, 49, 46, 95, 10, 50, 52, 48, 47, 49, 100]

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Noise: Các giá trị bất thường, không tuân theo xu hướng chung.

- Outliers: Các phần tử biên cách xa phần lớn dữ liệu, làm sai lệch kết quả.

Trong dữ liệu mẫu, đa số điểm nằm quanh 45–55, nhưng có các giá trị xa lạ như 10, 95, 100 → có thể coi là outliers.

c. Xử lý bằng tay

1. Sắp xếp dữ liệu:

[10, 45, 46, 47, 47, 48, 48, 49, 49, 50, 50, 51, 52, 52, 95, 100]

2. Tính Q1, Q3, IQR:

- Q1 = 47

- Q3 = 52

- IQR = Q3 - Q1 = 5

3. Ngưỡng phát hiện outlier (IQR method):

- Lower bound = 39.5

- Upper bound = 59.5

4. So sánh:

Các giá trị < 39.5 hoặc > 59.5 được xem là outlier.

Outliers = {10, 95, 100}

d. Giá trị sau xử lý trong dữ liệu mẫu

Bộ dữ liệu sau khi loại bỏ outliers:

[45, 50, 52, 48, 47, 51, 49, 46, 50, 52, 48, 47, 49]

Dữ liệu còn lại phản ánh chính xác phân bố điểm kiểm tra của lớp.

e. Code Python để hiện thực xử lý

import numpy as np

# Dữ liệu mẫu

data = [45, 50, 52, 48, 47, 51, 49, 46, 95, 10, 50, 52, 48, 47, 49, 100]

# Sắp xếp dữ liệu

data\_sorted = sorted(data)

# Tính Q1, Q3, IQR

Q1 = np.percentile(data\_sorted, 25)

Q3 = np.percentile(data\_sorted, 75)

IQR = Q3 - Q1

# Tính ngưỡng

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Phát hiện outliers

outliers = [x for x in data\_sorted if x < lower\_bound or x > upper\_bound]

# Loại bỏ outliers

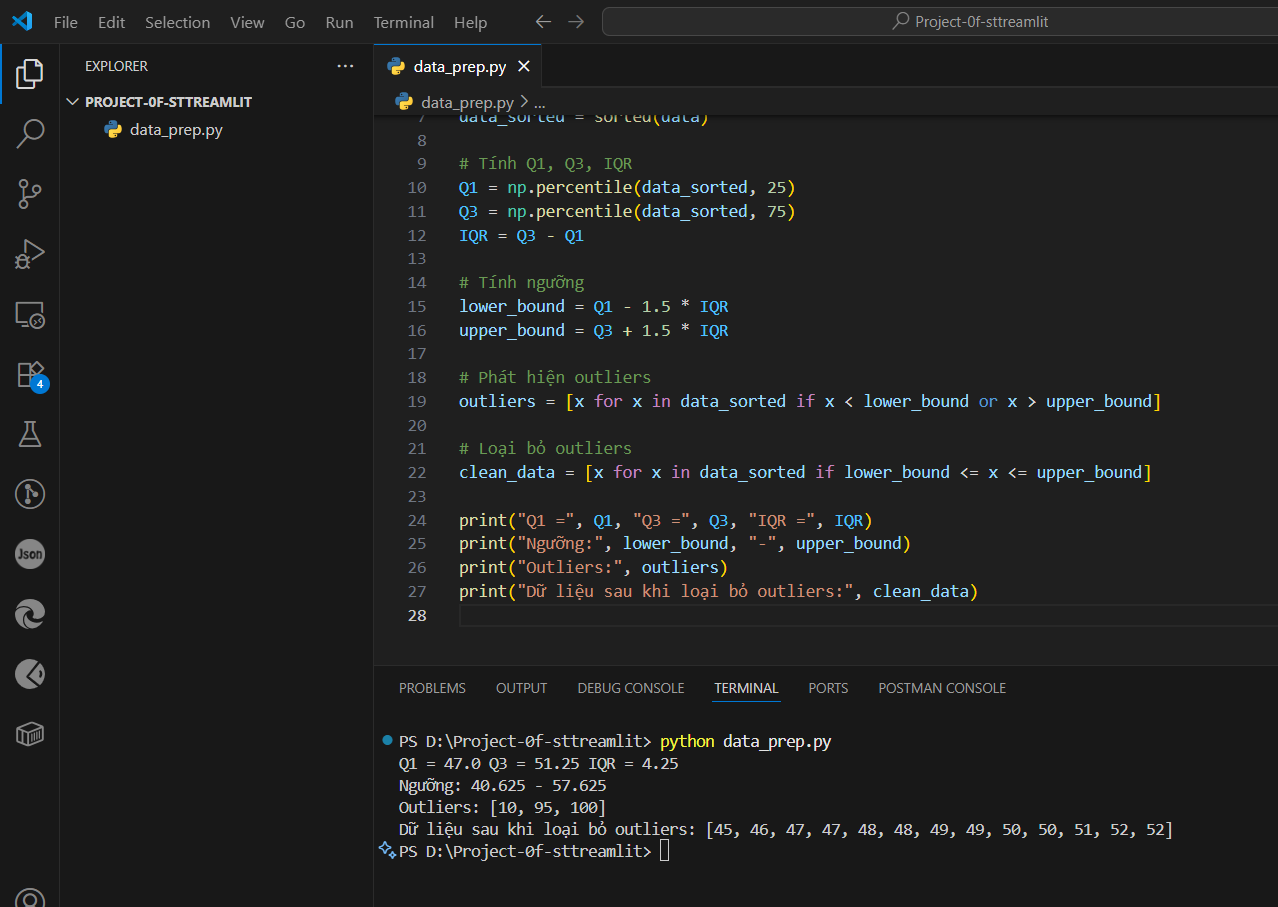
clean\_data = [x for x in data\_sorted if lower\_bound <= x <= upper\_bound]

print("Q1 =", Q1, "Q3 =", Q3, "IQR =", IQR)

print("Ngưỡng:", lower\_bound, "-", upper\_bound)

print("Outliers:", outliers)

print("Dữ liệu sau khi loại bỏ outliers:", clean\_data)



**B. Các kỹ thuật tiền xử lý dữ liệu**

**B.1. Làm sạch dữ liệu**

Làm sạch dữ liệu (data cleaning/cleansing): loại bỏ nhiễu (remove noise), hiệu chỉnh những phần dữ liệu không nhất quán (correct data inconsistencies)

Làm sạch dữ liệu (data cleaning/cleansing)

1. Tóm tắt hoá dữ liệu: nhận diện đặc điểm chung của dữ liệu và sự hiện diện của nhiễu hoặc các phần tử kì dị (outliers)

Bài tập Tóm tắt hoá dữ liệu và phát hiện nhiễu (Data Summarization & Outlier Detection)

a. Dữ liệu mẫu (dạng bảng)

ID Tuổi Thu nhập (triệu VND) Chi tiêu hàng tháng (triệu VND)

1 25 8 5

2 28 10 6

3 35 12 7

4 40 15 9

5 50 20 12

6 60 100 15

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Cần tóm tắt dữ liệu để nắm được đặc điểm chung: trung bình, trung vị, phương sai, min, max.

- Phát hiện sự hiện diện của nhiễu (noise) hoặc outliers.

- Trong dữ liệu, Thu nhập = 100 của ID=6 khác biệt rõ rệt so với phần còn lại (8–20).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Thống kê mô tả – Thu nhập:

- Min = 8

- Max = 100

- Mean = 27.5

- Median = 13.5

- Variance ≈ 1129.58

2. Nhận xét:

- Hầu hết dữ liệu Thu nhập nằm trong khoảng 8–20.

- Giá trị 100 cách biệt rõ → có thể là outlier.

3. Chi tiêu hàng tháng: dao động từ 5–15, không có giá trị bất thường.

d. Giá trị sau xử lý

- Tuổi: trung bình ≈ 39.7, trong khoảng 25–60.

- Thu nhập: trung bình 27.5 nhưng bị kéo lệch bởi outlier = 100, median = 13.5.

- Chi tiêu: khá ổn định, trung bình ≈ 9.0.

- Outlier phát hiện: Thu nhập = 100 (ID=6).

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5, 6],

"Tuổi": [25, 28, 35, 40, 50, 60],

"Thu nhập": [8, 10, 12, 15, 20, 100],

"Chi tiêu": [5, 6, 7, 9, 12, 15]

}

df = pd.DataFrame(data)

# Thống kê mô tả

summary = df.describe()

# Phát hiện outliers theo IQR cho cột "Thu nhập"

Q1 = df["Thu nhập"].quantile(0.25)

Q3 = df["Thu nhập"].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5\*IQR

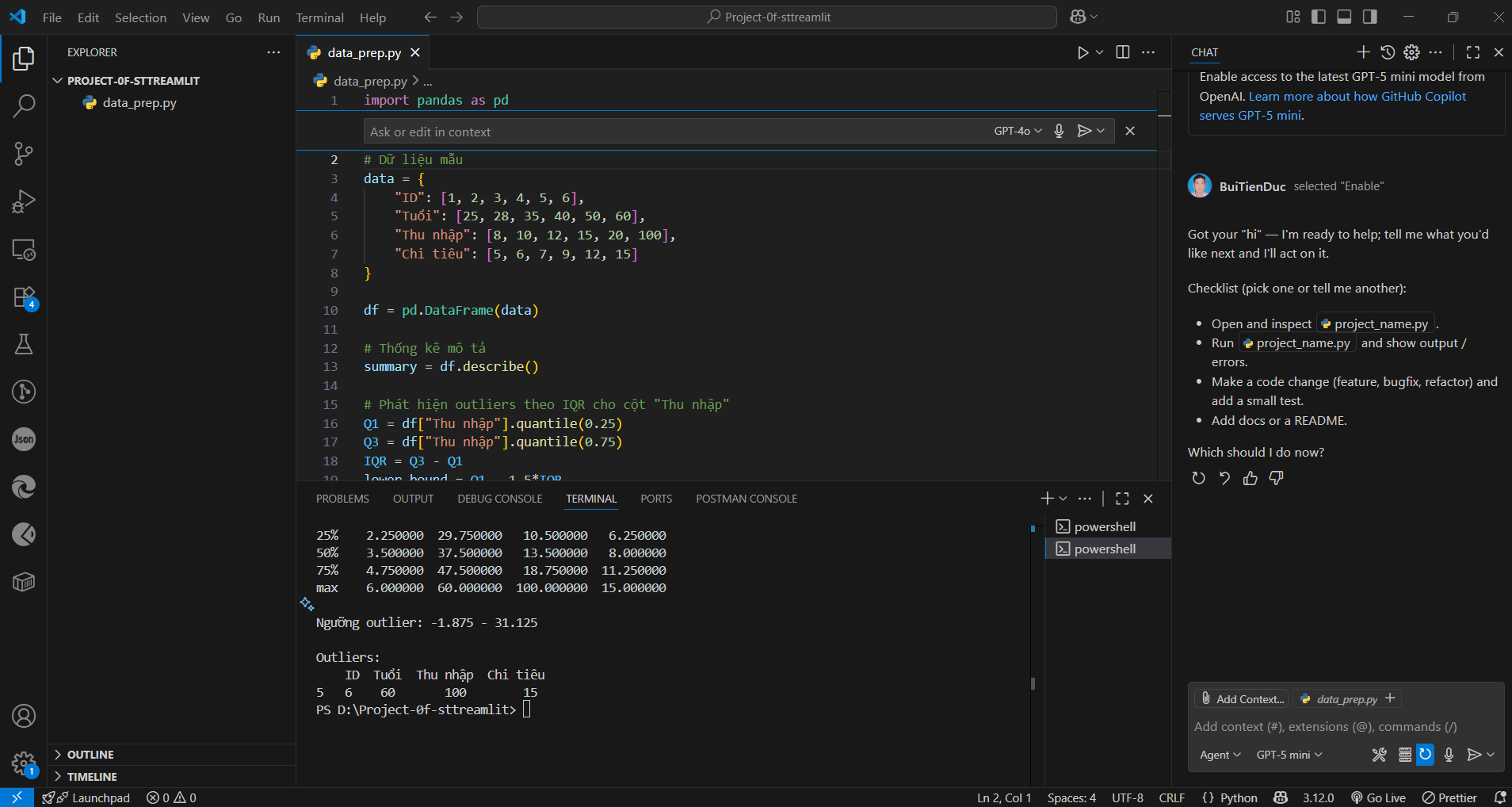
upper\_bound = Q3 + 1.5\*IQR

outliers = df[(df["Thu nhập"] < lower\_bound) | (df["Thu nhập"] > upper\_bound)]

print("Tóm tắt dữ liệu:\n", summary)

print("\nNgưỡng outlier:", lower\_bound, "-", upper\_bound)

print("\nOutliers:\n", outliers)



1. Xử lý dữ liệu bị thiếu (missing data)

Bài tập Xử lý dữ liệu bị thiếu (Missing Data Handling)

a. Dữ liệu mẫu

ID Tuổi Thu nhập (triệu VND) Giới tính

1 25 8 Nam

2 30 NaN Nữ

3 NaN 12 Nam

4 40 15 NaN

5 35 20 Nữ

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Dữ liệu bị thiếu (missing values) gây khó khăn cho phân tích và khai phá dữ liệu.

- Các phương pháp xử lý:

1. Loại bỏ bản ghi (deletion).

2. Thay thế bằng trung bình/trung vị/mốt (imputation).

3. Suy luận từ thuộc tính khác (prediction-based imputation).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Tuổi bị thiếu (ID=3): Trung bình tuổi = 32.5 ≈ 33 → thay thế = 33.

2. Thu nhập bị thiếu (ID=2): Trung bình thu nhập = 13.75 ≈ 14 → thay thế = 14.

3. Giới tính bị thiếu (ID=4): Mốt giới tính = 'Nữ' → thay thế = 'Nữ'.

d. Giá trị sau xử lý

ID Tuổi Thu nhập (triệu VND) Giới tính

1 25 8 Nam

2 30 14 Nữ

3 33 12 Nam

4 40 15 Nữ

5 35 20 Nữ

Dữ liệu sau xử lý đã đầy đủ, không còn giá trị thiếu.

e. Code Python để hiện thực xử lý

import pandas as pd

import numpy as np

# Tạo dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5],

"Tuổi": [25, 30, np.nan, 40, 35],

"Thu nhập": [8, np.nan, 12, 15, 20],

"Giới tính": ["Nam", "Nữ", "Nam", np.nan, "Nữ"]

}

df = pd.DataFrame(data)

print("Dữ liệu gốc:\n", df)

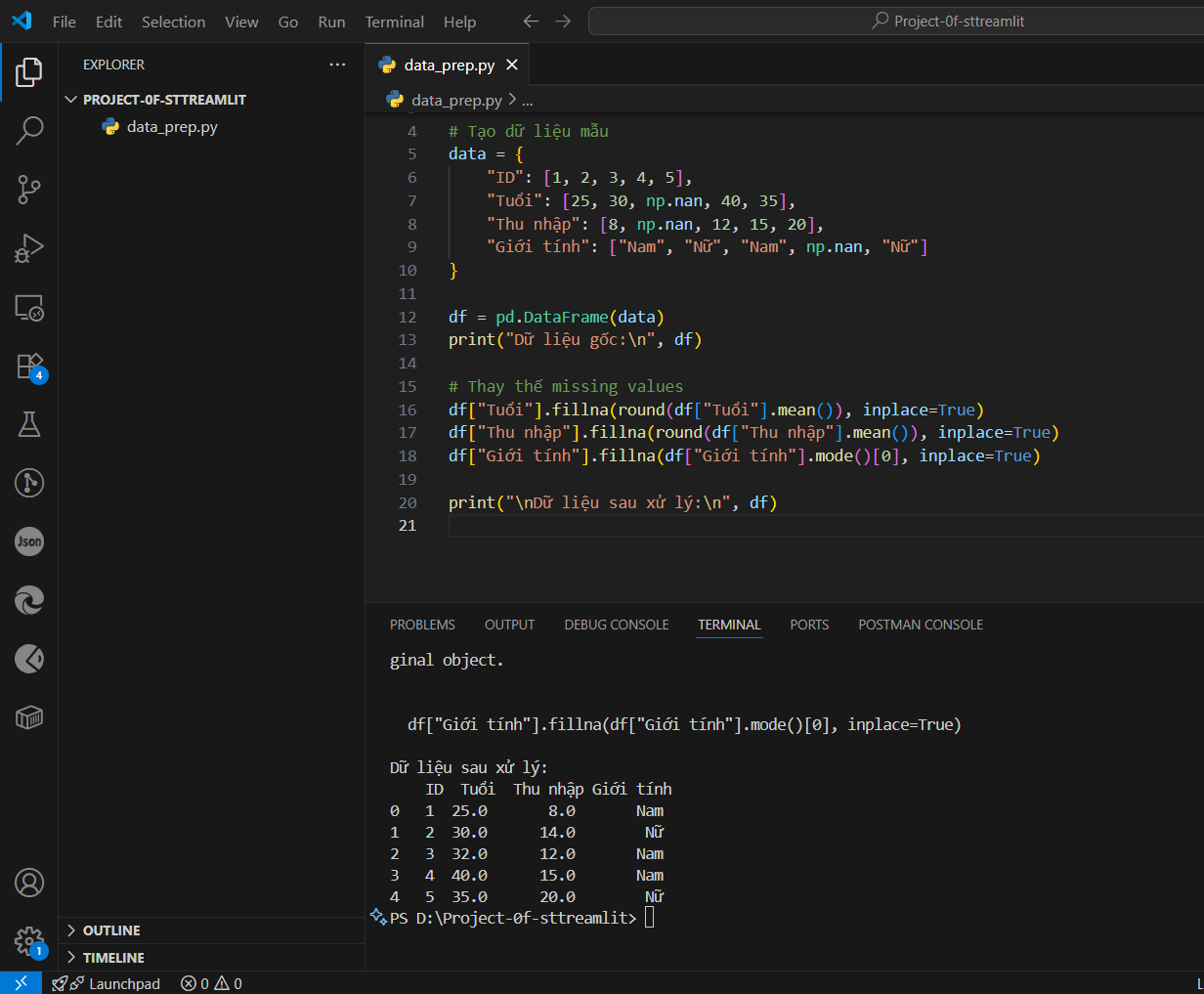
# Thay thế missing values

df["Tuổi"].fillna(round(df["Tuổi"].mean()), inplace=True)

df["Thu nhập"].fillna(round(df["Thu nhập"].mean()), inplace=True)

df["Giới tính"].fillna(df["Giới tính"].mode()[0], inplace=True)

print("\nDữ liệu sau xử lý:\n", df)



1. Xử lý dữ liệu bị nhiễu (noisy data)

Bài tập Xử lý dữ liệu bị nhiễu (Noisy Data Handling)

a. Dữ liệu mẫu (dạng bảng)

ID Chiều cao (cm)

1 165

2 170

3 168

4 400

5 172

6 169

7 -50

8 171

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Noisy data là các giá trị bị lỗi, bất thường hoặc phi thực tế trong dữ liệu.

- Nếu không xử lý, chúng sẽ gây sai lệch khi phân tích hoặc huấn luyện mô hình.

- Các kỹ thuật xử lý phổ biến: Binning, Regression, Clustering.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Dữ liệu hợp lý nằm trong khoảng 150–200 cm.

2. Các giá trị 400 và -50 được xem là nhiễu.

3. Thay thế bằng trung bình của các giá trị hợp lệ:

- Mean (giá trị hợp lệ) = (165+170+168+172+169+171)/6 = 169.2 ≈ 169

- Thay thế 400 → 169, -50 → 169.

d. Giá trị sau xử lý

ID Chiều cao (cm)

1 165

2 170

3 168

4 169

5 172

6 169

7 169

8 171

Bộ dữ liệu sau xử lý đã loại bỏ giá trị nhiễu và trở nên hợp lý hơn.

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5, 6, 7, 8],

"Chiều cao": [165, 170, 168, 400, 172, 169, -50, 171]

}

df = pd.DataFrame(data)

print("Dữ liệu gốc:\n", df)

# Giả sử giá trị hợp lý nằm trong khoảng 150–200 cm

valid = df[(df["Chiều cao"] >= 150) & (df["Chiều cao"] <= 200)]

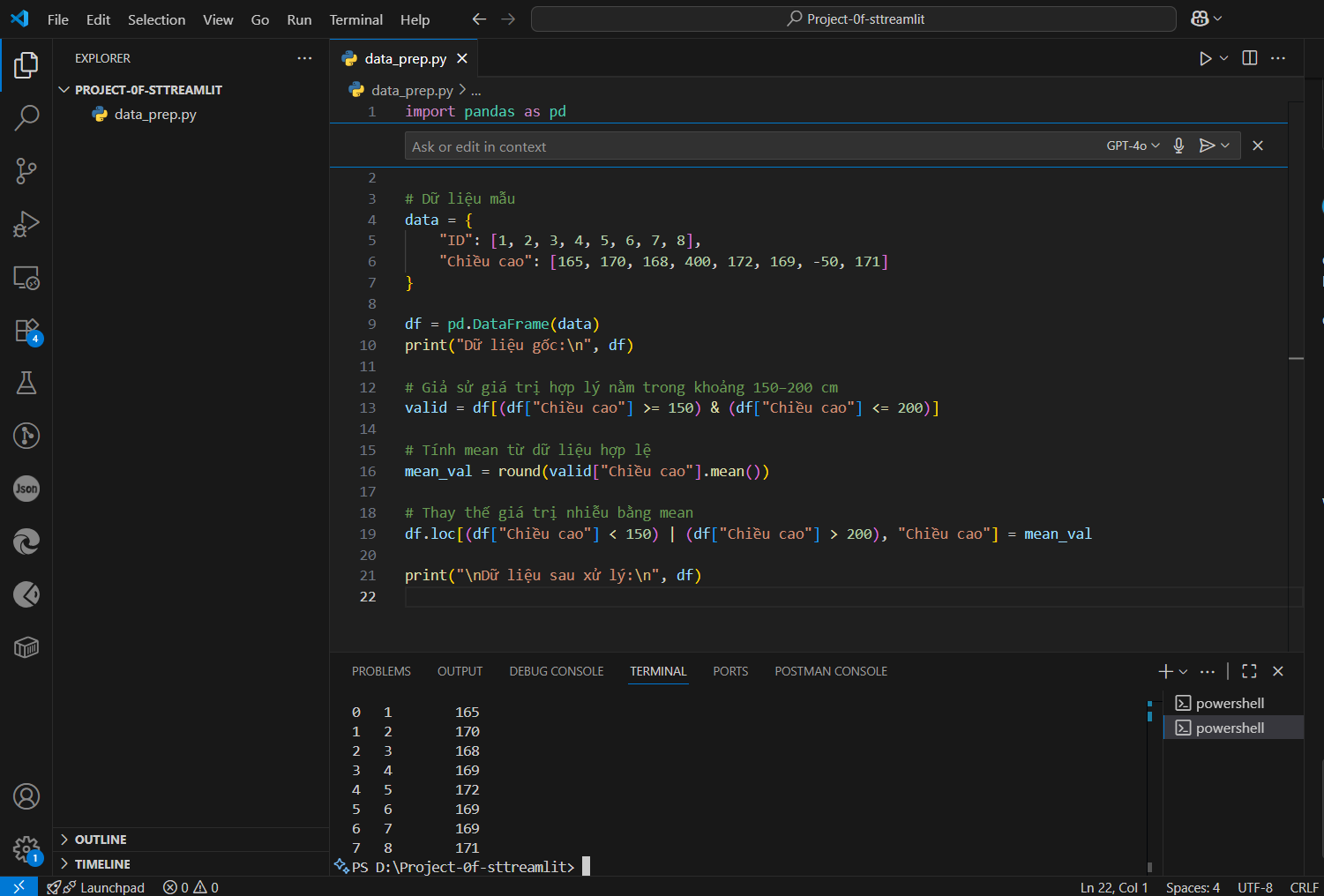
# Tính mean từ dữ liệu hợp lệ

mean\_val = round(valid["Chiều cao"].mean())

# Thay thế giá trị nhiễu bằng mean

df.loc[(df["Chiều cao"] < 150) | (df["Chiều cao"] > 200), "Chiều cao"] = mean\_val

print("\nDữ liệu sau xử lý:\n", df)



1. Xử lý dữ liệu không nhất quán (inconsistent data)

Bài tập: Xử lý dữ liệu không nhất quán (Inconsistent Data Handling)

a. Dữ liệu mẫu (dạng bảng)

ID Tên Ngày sinh Tuổi Thành phố

1 An 2000-05-10 25 Hà Nội

2 Bình 1998-07-21 20 HCM City

3 Chi 1985-03-15 30 Hồ Chí Minh

4 Dũng 1990-10-30 40 Ha Noi

5 Hạnh 2002-12-01 22 Đà Nẵng

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Inconsistent data là dữ liệu không đồng nhất, mâu thuẫn giữa các trường hoặc cách ghi.

- Các dạng phổ biến:

1. Tuổi không khớp với ngày sinh.

2. Tên địa phương viết khác nhau.

- Nếu không xử lý, sẽ gây sai lệch trong phân tích.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Tính tuổi thực từ ngày sinh (năm 2025):

- ID=2: 1998 → 27, nhưng ghi 20 → sửa thành 27.

- ID=3: 1985 → 40, nhưng ghi 30 → sửa thành 40.

- ID=4: 1990 → 35, nhưng ghi 40 → sửa thành 35.

- ID=5: 2002 → 23, nhưng ghi 22 → sửa thành 23.

2. Chuẩn hóa tên thành phố:

- 'HCM City' → 'Hồ Chí Minh'.

- 'Ha Noi' → 'Hà Nội'.

d. Giá trị sau xử lý

ID Tên Ngày sinh Tuổi Thành phố

1 An 2000-05-10 25 Hà Nội

2 Bình 1998-07-21 27 Hồ Chí Minh

3 Chi 1985-03-15 40 Hồ Chí Minh

4 Dũng 1990-10-30 35 Hà Nội

5 Hạnh 2002-12-01 23 Đà Nẵng

Bộ dữ liệu đã được đồng bộ tuổi và chuẩn hóa tên địa phương.

e. Code Python để hiện thực xử lý

import pandas as pd

from datetime import datetime

# Dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5],

"Tên": ["An", "Bình", "Chi", "Dũng", "Hạnh"],

"Ngày sinh": ["2000-05-10", "1998-07-21", "1985-03-15", "1990-10-30", "2002-12-01"],

"Tuổi": [25, 20, 30, 40, 22],

"Thành phố": ["Hà Nội", "HCM City", "Hồ Chí Minh", "Ha Noi", "Đà Nẵng"]

}

df = pd.DataFrame(data)

# Tính lại tuổi từ ngày sinh (năm 2025)

current\_year = 2025

df["Ngày sinh"] = pd.to\_datetime(df["Ngày sinh"])

df["Tuổi\_tính"] = current\_year - df["Ngày sinh"].dt.year

# Thay thế tuổi không nhất quán

df["Tuổi"] = df["Tuổi\_tính"]

# Chuẩn hóa tên thành phố

df["Thành phố"] = df["Thành phố"].replace({

"HCM City": "Hồ Chí Minh",

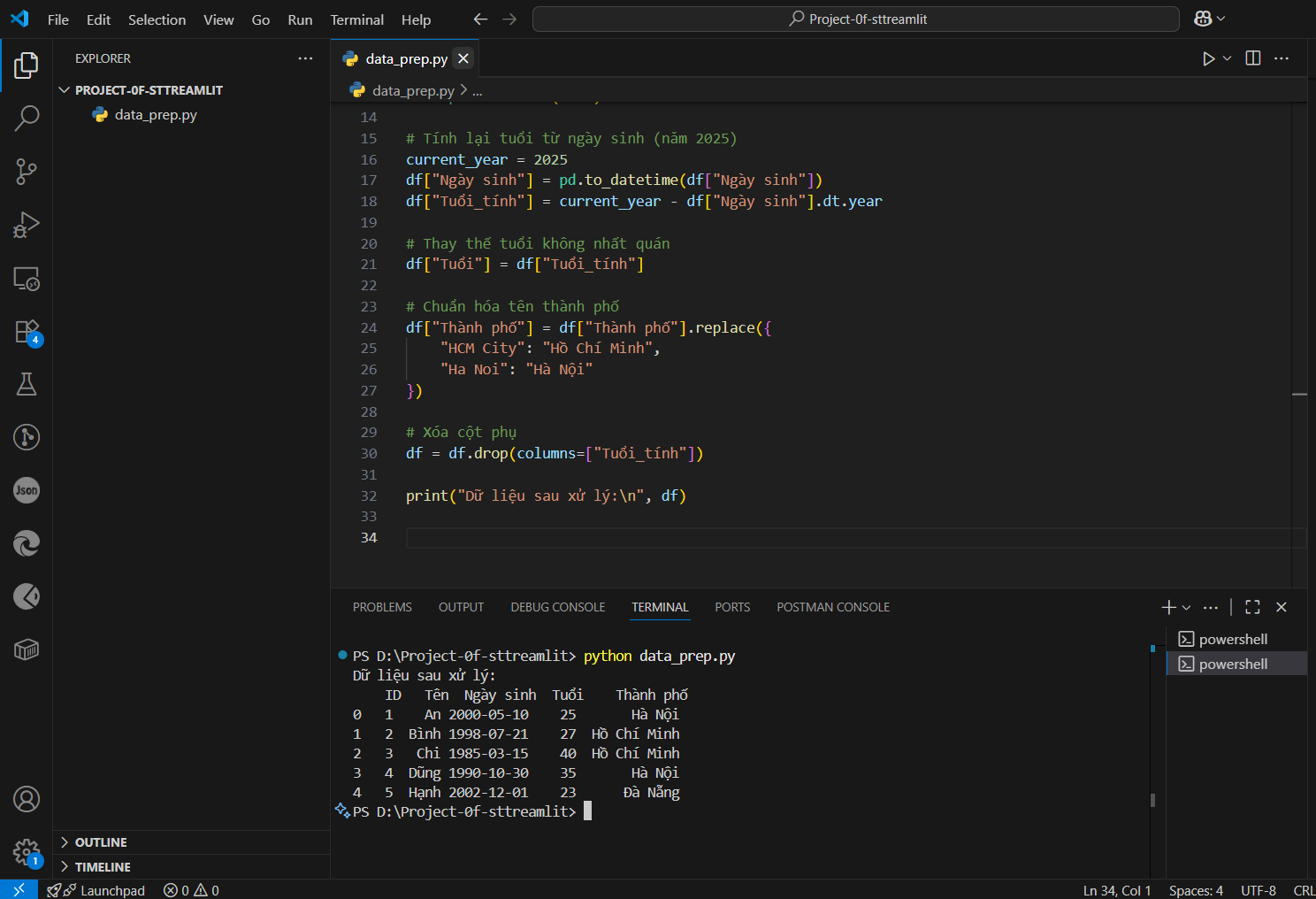
"Ha Noi": "Hà Nội"

})

# Xóa cột phụ

df = df.drop(columns=["Tuổi\_tính"])

print("Dữ liệu sau xử lý:\n", df)



**B.2. Tích hợp dữ liệu**

Tích hợp dữ liệu (data integration): trộn dữ liệu (merge data) từ nhiều nguồn khác nhau vào một kho dữ liệu

Tích hợp dữ liệu (data integration)

1. Tích hợp lược đồ (schema integration) và so trùng đối tượng (object matching)

Bài tập: Tích hợp lược đồ (Schema Integration) và so trùng đối tượng (Object Matching)

a. Dữ liệu mẫu (dạng bảng)

Bảng 1 – Customers\_A

Cust\_ID Name Age City Phone

1 An 25 Ha Noi 0901234567

2 Binh 30 HCM City 0912345678

3 Chi 28 Da Nang 0987654321

Bảng 2 – Customers\_B

CustomerNo FullName BirthYear Location Mobile

C001 Nguyễn An 2000 Hà Nội 0901234567

C002 Trần Bình 1995 Hồ Chí Minh 0912345678

C003 Lê Chi 1997 Đà Nẵng 0987654321

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Schema integration: Hai bảng có cùng bản chất nhưng tên cột khác nhau (Cust\_ID vs CustomerNo, City vs Location).

- Object matching: Cùng khách hàng có thể xuất hiện với cách ghi khác nhau (Ha Noi vs Hà Nội).

- Cần hợp nhất hai bảng về cùng một lược đồ và so khớp đối tượng dựa trên Phone.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Mapping schema:

- Cust\_ID ↔ CustomerNo

- Name ↔ FullName

- Age ↔ BirthYear (quy đổi)

- City ↔ Location

- Phone ↔ Mobile

2. So trùng đối tượng:

- ID=1 (An) ↔ C001 (Nguyễn An)

- ID=2 (Binh) ↔ C002 (Trần Bình)

- ID=3 (Chi) ↔ C003 (Lê Chi)

3. Quy đổi BirthYear sang Age (năm 2025): 2000→25, 1995→30, 1997→28

d. Giá trị sau xử lý

CustomerID FullName Age City Phone

1 Nguyễn An 25 Hà Nội 0901234567

2 Trần Bình 30 Hồ Chí Minh 0912345678

3 Lê Chi 28 Đà Nẵng 0987654321

Hai bảng đã được tích hợp về một lược đồ chung và đối tượng trùng lặp được hợp nhất.

e. Code Python để hiện thực xử lý

import pandas as pd

# Bảng A

customers\_a = pd.DataFrame({

"Cust\_ID": [1, 2, 3],

"Name": ["An", "Binh", "Chi"],

"Age": [25, 30, 28],

"City": ["Ha Noi", "HCM City", "Da Nang"],

"Phone": ["0901234567", "0912345678", "0987654321"]

})

# Bảng B

customers\_b = pd.DataFrame({

"CustomerNo": ["C001", "C002", "C003"],

"FullName": ["Nguyễn An", "Trần Bình", "Lê Chi"],

"BirthYear": [2000, 1995, 1997],

"Location": ["Hà Nội", "Hồ Chí Minh", "Đà Nẵng"],

"Mobile": ["0901234567", "0912345678", "0987654321"]

})

# Quy đổi BirthYear → Age

current\_year = 2025

customers\_b["Age"] = current\_year - customers\_b["BirthYear"]

# Chuẩn hóa schema B

customers\_b\_renamed = customers\_b.rename(columns={

"CustomerNo": "Cust\_ID",

"FullName": "Name",

"Location": "City",

"Mobile": "Phone"

})[["Cust\_ID", "Name", "Age", "City", "Phone"]]

# Merge hai bảng dựa trên Phone

merged = pd.merge(customers\_a, customers\_b\_renamed, on="Phone", suffixes=("\_A", "\_B"))

# Tạo bảng hợp nhất

unified = pd.DataFrame({

"CustomerID": merged["Cust\_ID\_A"],

"FullName": merged["Name\_B"],

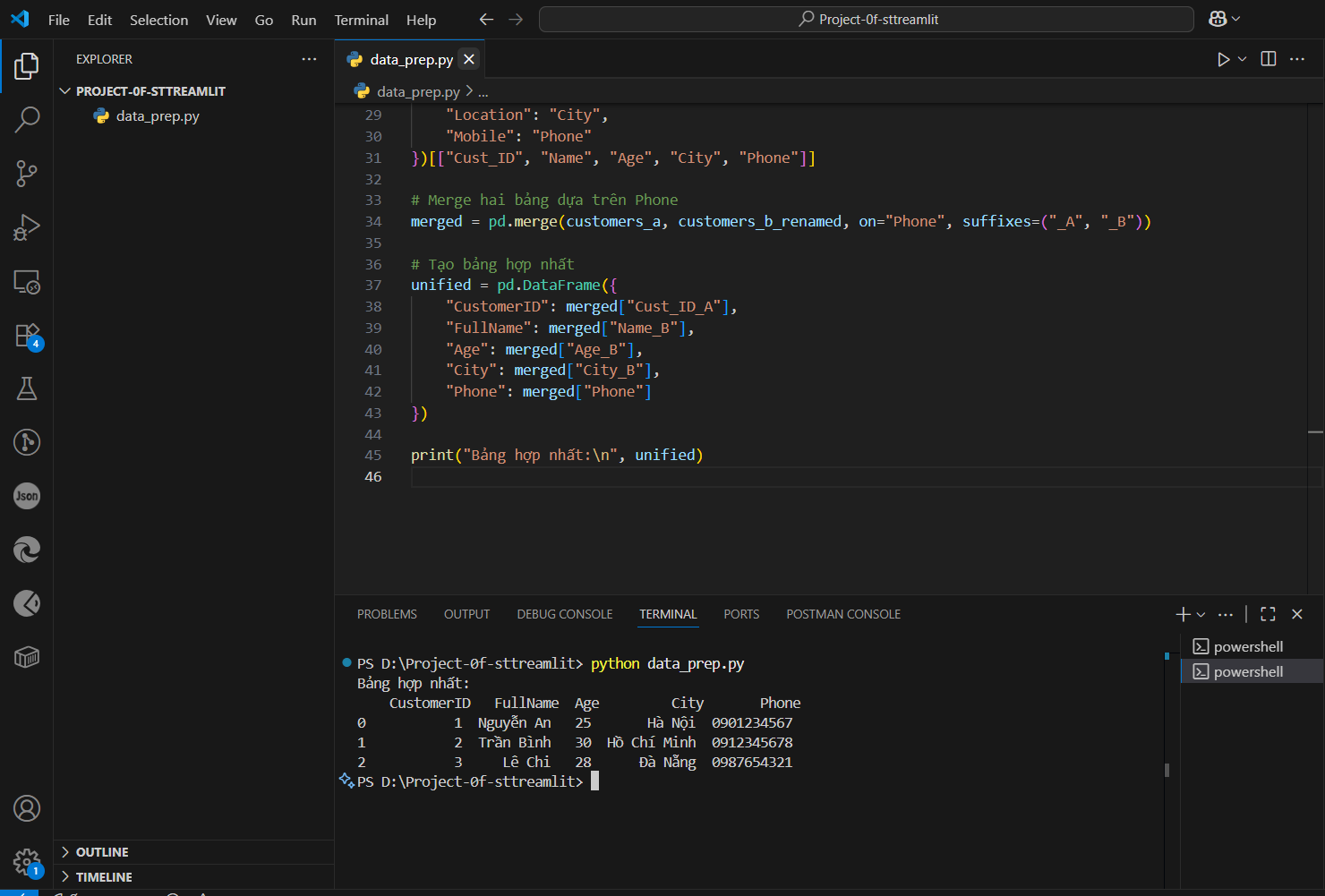
"Age": merged["Age\_B"],

"City": merged["City\_B"],

"Phone": merged["Phone"]

})

print("Bảng hợp nhất:\n", unified)



1. Vấn đề dư thừa (redundancy)

Bài tập: Vấn đề dư thừa (Redundancy) trong Tích hợp Dữ liệu

a. Dữ liệu mẫu (dạng bảng)

Bảng 1 – Sales\_A

ID Name City Phone AnnualIncome (triệu VND)

1 An Ha Noi 0901234567 120

2 Binh HCM City 0912345678 180

3 Chi Da Nang 0987654321 144

4 Binh HCM City 0912345678 180

Bảng 2 – CRM\_B

CustomerNo FullName Location Mobile MonthlyIncome (triệu VND) Sex

C001 Nguyễn An Hà Nội 0901234567 10 Nam

C002 Trần Bình Hồ Chí Minh 0912345678 15 Nam

C003 Lê Chi Đà Nẵng 0987654321 12 Nữ

C004 Lê Chi Đà Nẵng 0987654321 12 Nữ

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Dư thừa (redundancy) xuất hiện khi nhiều nguồn lưu trữ thông tin trùng lặp hoặc có mối phụ thuộc hàm giữa các thuộc tính.

- Biểu hiện trong ví dụ:

• Trùng bản ghi: cùng một khách hàng lặp lại trong Sales\_A (ID=2 & 4) và CRM\_B (C003 & C004) theo cùng số điện thoại.

• Thuộc tính trùng nghĩa: City ≈ Location; Name ≈ FullName; Phone ≈ Mobile.

• Thuộc tính dư thừa theo phụ thuộc hàm: MonthlyIncome = AnnualIncome / 12 (hệ số tương quan ~ 1.0).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1) Chuẩn hoá lược đồ (schema mapping):

- ID ↔ CustomerNo; Name ↔ FullName; City ↔ Location; Phone ↔ Mobile; AnnualIncome ↔ 12×MonthlyIncome.

2) Chuẩn hoá giá trị (standardization):

- Ha Noi/ Hà Nội → 'Hà Nội'; HCM City/ Hồ Chí Minh → 'Hồ Chí Minh'; Da Nang/ Đà Nẵng → 'Đà Nẵng'.

3) So trùng đối tượng (object matching) theo Phone/Mobile:

- 0901234567 → (An ↔ Nguyễn An); 0912345678 → (Binh ↔ Trần Bình); 0987654321 → (Chi ↔ Lê Chi).

4) Xử lý trùng bản ghi:

- Giữ bản ghi đầu tiên, loại bỏ lặp: Sales\_A(ID=4) & CRM\_B(C004).

5) Gộp thuộc tính dư thừa:

- Giữ AnnualIncome, loại MonthlyIncome (vì MonthlyIncome = AnnualIncome/12).

d. Giá trị sau xử lý

Bảng hợp nhất (Unified\_Customers) – không trùng bản ghi, lược đồ thống nhất, bỏ thuộc tính dư thừa:

CustomerID FullName City Phone AnnualIncome (triệu VND)

1 Nguyễn An Hà Nội 0901234567 120

2 Trần Bình Hồ Chí Minh 0912345678 180

3 Lê Chi Đà Nẵng 0987654321 144

Kết quả: loại bỏ trùng lặp, chuẩn hoá giá trị, và loại thuộc tính dư thừa có phụ thuộc hàm.

e. Code Python để hiện thực xử lý

import pandas as pd

# Nguồn A (Sales\_A)

a = pd.DataFrame({

"ID": [1, 2, 3, 4],

"Name": ["An", "Binh", "Chi", "Binh"],

"City": ["Ha Noi", "HCM City", "Da Nang", "HCM City"],

"Phone": ["0901234567", "0912345678", "0987654321", "0912345678"],

"AnnualIncome": [120, 180, 144, 180]

})

# Nguồn B (CRM\_B)

b = pd.DataFrame({

"CustomerNo": ["C001", "C002", "C003", "C004"],

"FullName": ["Nguyễn An", "Trần Bình", "Lê Chi", "Lê Chi"],

"Location": ["Hà Nội", "Hồ Chí Minh", "Đà Nẵng", "Đà Nẵng"],

"Mobile": ["0901234567", "0912345678", "0987654321", "0987654321"],

"MonthlyIncome": [10, 15, 12, 12],

"Sex": ["Nam", "Nam", "Nữ", "Nữ"]

})

# Chuẩn hoá thành phố

map\_city = {

"Ha Noi": "Hà Nội", "Hà Nội": "Hà Nội",

"HCM City": "Hồ Chí Minh", "Hồ Chí Minh": "Hồ Chí Minh",

"Da Nang": "Đà Nẵng", "Đà Nẵng": "Đà Nẵng"

}

a["City"] = a["City"].map(map\_city)

b["Location"] = b["Location"].map(map\_city)

# Loại trùng bản ghi theo số điện thoại trong từng nguồn

a = a.drop\_duplicates(subset=["Phone"], keep="first")

b = b.drop\_duplicates(subset=["Mobile"], keep="first")

# Đổi tên cột cho cùng lược đồ

b\_std = b.rename(columns={

"CustomerNo": "ID\_B",

"FullName": "Name\_B",

"Location": "City\_B",

"Mobile": "Phone",

"MonthlyIncome": "MonthlyIncome"

})

# Hợp nhất theo Phone

merged = pd.merge(a, b\_std, on="Phone", how="inner")

# Xác nhận dư thừa: MonthlyIncome ≈ AnnualIncome/12

merged["Annual\_from\_month"] = merged["MonthlyIncome"] \* 12

# Tạo bảng hợp nhất, bỏ thuộc tính dư thừa MonthlyIncome

unified = pd.DataFrame({

"CustomerID": merged["ID"],

"FullName": merged["Name\_B"],

"City": merged["City"],

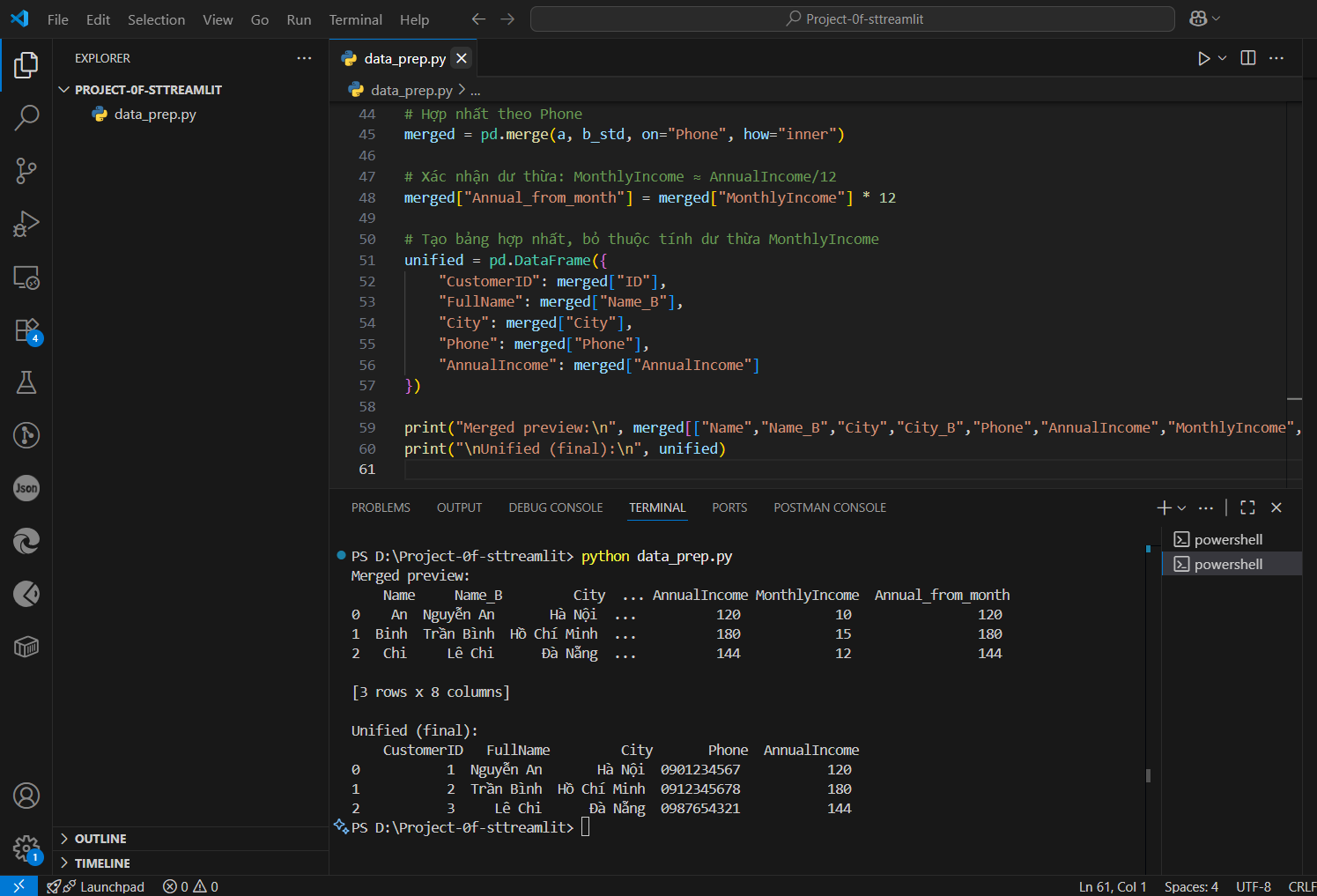
"Phone": merged["Phone"],

"AnnualIncome": merged["AnnualIncome"]

})

print("Merged preview:\n", merged[["Name","Name\_B","City","City\_B","Phone","AnnualIncome","MonthlyIncome","Annual\_from\_month"]])

print("\nUnified (final):\n", unified)



1. Phát hiện và xử lý mâu thuẫn giá trị dữ liệu (detection and resolution of data value conflicts)

Bài tập: Phát hiện và xử lý mâu thuẫn giá trị dữ liệu (Detection and Resolution of Data Value Conflicts)

a. Dữ liệu mẫu (dạng bảng)

CustomerID Name BirthDate Gender Income (USD) Income\_Source

C001 Nguyễn An 1990-05-21 Male 1200 Job

C002 Trần Bình 1985-08-14 M 1500 Salary

C003 Lê Cường 1992-12-02 Nam 2000 Job

C004 Phạm Dung 1987-07-19 Female 1800 Job

C005 Lê Cường 1992-12-02 Male 2500 Salary

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Cột Gender có mâu thuẫn biểu diễn: Male/M/Nam.

- Cột Income có mâu thuẫn giá trị: cùng khách hàng (Lê Cường) nhưng thu nhập khác nhau.

- Cột Income\_Source chưa đồng nhất: Job vs Salary.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Chuẩn hoá Gender: M, Nam → Male.

2. Chuẩn hoá Income\_Source: Job → Salary.

3. Với khách hàng Lê Cường (cùng Name & BirthDate): Income có hai giá trị 2000 và 2500 → lấy trung bình = 2250.

d. Giá trị sau xử lý

CustomerID Name BirthDate Gender Income (USD) Income\_Source

C001 Nguyễn An 1990-05-21 Male 1200 Salary

C002 Trần Bình 1985-08-14 Male 1500 Salary

C003 Lê Cường 1992-12-02 Male 2250 Salary

C004 Phạm Dung 1987-07-19 Female 1800 Salary

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"CustomerID": ["C001", "C002", "C003", "C004", "C005"],

"Name": ["Nguyễn An", "Trần Bình", "Lê Cường", "Phạm Dung", "Lê Cường"],

"BirthDate": ["1990-05-21", "1985-08-14", "1992-12-02", "1987-07-19", "1992-12-02"],

"Gender": ["Male", "M", "Nam", "Female", "Male"],

"Income": [1200, 1500, 2000, 1800, 2500],

"Income\_Source": ["Job", "Salary", "Job", "Job", "Salary"]

}

df = pd.DataFrame(data)

# Chuẩn hóa Gender

df["Gender"] = df["Gender"].replace({"M": "Male", "Nam": "Male"})

# Chuẩn hóa Income\_Source

df["Income\_Source"] = df["Income\_Source"].replace({"Job": "Salary"})

# Phát hiện và xử lý mâu thuẫn giá trị theo (Name, BirthDate)

df\_resolved = (

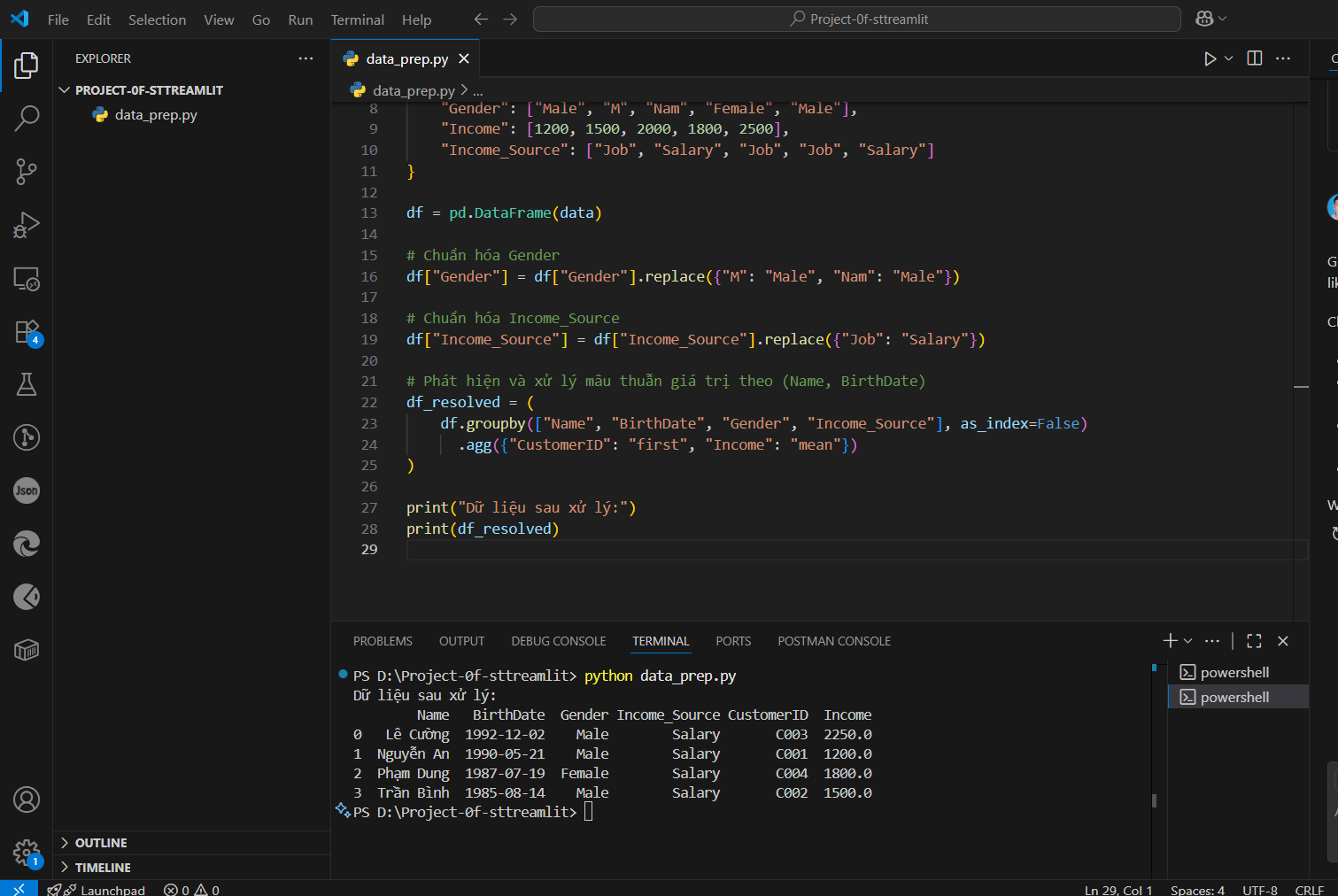
df.groupby(["Name", "BirthDate", "Gender", "Income\_Source"], as\_index=False)

.agg({"CustomerID": "first", "Income": "mean"})

)

print("Dữ liệu sau xử lý:")

print(df\_resolved)



1. Phân tích tương quan giữa hai thuộc tính số A và B

Bài tập: Phân tích tương quan giữa hai thuộc tính số A và B

a. Dữ liệu mẫu (dạng bảng)

StudentID Hours\_Study (A) Exam\_Score (B)

S01 2 40

S02 3 45

S03 4 50

S04 5 65

S05 6 70

S06 7 75

S07 8 85

S08 9 88

b. Định nghĩa vấn đề trong dữ liệu mẫu

Phân tích mối tương quan giữa hai thuộc tính số:

- A = Hours\_Study (số giờ học)

- B = Exam\_Score (điểm kiểm tra)

Câu hỏi: Liệu số giờ học có quan hệ tuyến tính với điểm kiểm tra?

Sử dụng hệ số tương quan Pearson (r).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Mean(A) = 5.5, Mean(B) = 64.75.

2. Tính (Ai - MeanA)(Bi - MeanB) cho từng sinh viên → tổng = 308.0.

3. ∑(Ai - MeanA)^2 = 42.0; ∑(Bi - MeanB)^2 = 2275.5.

4. √(42 × 2275.5) ≈ 309.2.

5. r = 308.0 / 309.2 ≈ 0.996.

d. Giá trị sau xử lý

- Hệ số tương quan Pearson r ≈ 0.996 → tương quan tuyến tính dương rất mạnh.

- Kết luận: Số giờ học càng nhiều thì điểm kiểm tra càng cao.

e. Code Python để hiện thực xử lý

import pandas as pd

import numpy as np

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04","S05","S06","S07","S08"],

"Hours\_Study": [2,3,4,5,6,7,8,9],

"Exam\_Score": [40,45,50,65,70,75,85,88]

}

df = pd.DataFrame(data)

# Tính hệ số tương quan Pearson

corr = df["Hours\_Study"].corr(df["Exam\_Score"])

print("Hệ số tương quan Pearson r =", corr)

# Kiểm tra thêm bằng numpy

r\_matrix = np.corrcoef(df["Hours\_Study"], df["Exam\_Score"])

print("Ma trận hệ số tương quan:\n", r\_matrix)

Hệ số tương quan Pearson (r) giữa số giờ học và điểm kiểm tra.

1. Vai trò của hệ số tương quan

Đo lường mức độ tuyến tính giữa hai biến số (ở đây là Hours\_Study và Exam\_Score).

Giá trị r nằm trong [-1, 1]:

r ≈ +1: quan hệ tuyến tính dương mạnh (x tăng thì y tăng).

r ≈ -1: quan hệ tuyến tính âm mạnh (x tăng thì y giảm).

r ≈ 0: gần như không có mối quan hệ tuyến tính.

Trong kết quả của bạn, r ≈ 0.99 → nghĩa là số giờ học có quan hệ tuyến tính dương rất mạnh với điểm kiểm tra.

2. Trong trích chọn thuộc tính (feature selection) cho mô hình

Mục tiêu: chọn ra các thuộc tính có giá trị giải thích/ảnh hưởng mạnh tới biến mục tiêu (target).

Hệ số tương quan giúp:

Xác định biến hữu ích: Thuộc tính nào có tương quan cao với biến mục tiêu → giữ lại.

Loại bỏ biến dư thừa: Nếu hai thuộc tính có tương quan rất cao với nhau (multicollinearity), ta chỉ giữ một để tránh trùng lặp thông tin.

Cải thiện hiệu năng mô hình: ít thuộc tính nhiễu, tốc độ huấn luyện nhanh hơn, mô hình ổn định hơn.

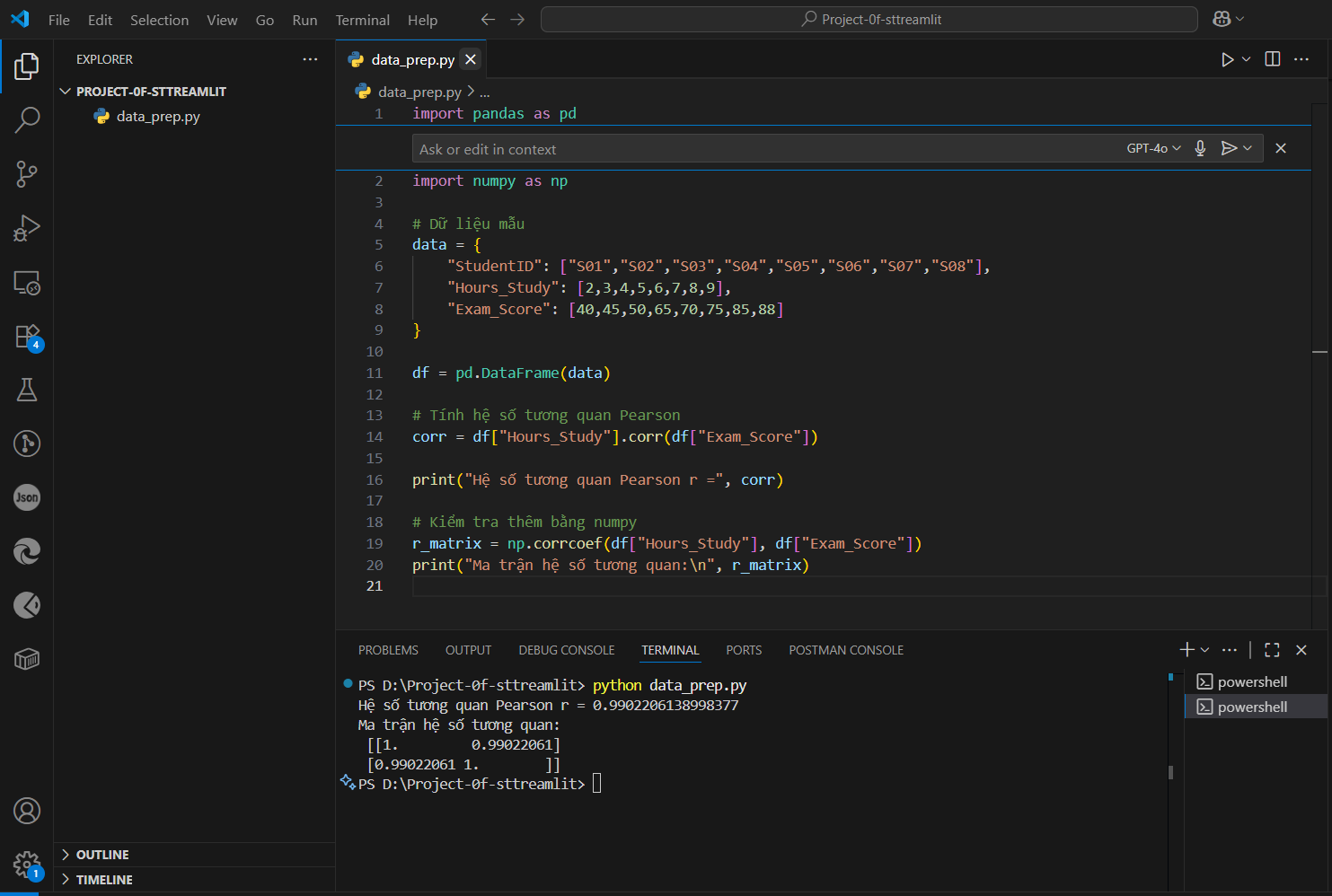
Ví dụ:

Nếu có nhiều biến đầu vào, có thể tính hệ số tương quan giữa từng biến với biến mục tiêu (label).

Giữ các biến có |r| cao, loại bỏ các biến có |r| thấp (không giải thích được target).

Giữa các biến độc lập, nếu chúng quá tương quan với nhau, chỉ giữ 1 biến để tránh dư thừa.

* Nói ngắn gọn: Hệ số tương quan giúp bạn quyết định biến nào nên giữ/lọc khi xây dựng mô hình học máy.



Ví dụ:

Bài tập: Trích chọn thuộc tính bằng hệ số tương quan (Feature Selection using Correlation)

a. Dữ liệu mẫu (dạng bảng)

StudentID Hours\_Study TV\_Hours Gaming\_Hours Extra\_Class Exam\_Score

S01 2 5 6 1 40

S02 3 6 5 1 45

S03 4 6 5 2 50

S04 5 7 4 2 65

S05 6 7 4 3 70

S06 7 8 3 3 75

S07 8 9 2 4 85

S08 9 9 2 4 88

b. Định nghĩa vấn đề trong dữ liệu mẫu

Ta muốn dự đoán điểm thi (Exam\_Score) dựa trên nhiều thuộc tính đầu vào như giờ học (Hours\_Study), giờ xem TV (TV\_Hours), giờ chơi game (Gaming\_Hours), và số lần đi học thêm (Extra\_Class).

Không phải tất cả các thuộc tính đều có ảnh hưởng mạnh tới điểm thi. Cần dùng hệ số tương quan để xác định thuộc tính nào quan trọng nhất và nên giữ lại.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Tính hệ số tương quan Pearson giữa mỗi thuộc tính đầu vào với Exam\_Score.

2. Nhận xét:

- Hours\_Study có tương quan dương mạnh với Exam\_Score.

- Gaming\_Hours có tương quan âm mạnh với Exam\_Score.

- TV\_Hours và Extra\_Class có mức tương quan trung bình.

3. Đặt ngưỡng chọn |r| > 0.7 → giữ lại Hours\_Study và Gaming\_Hours.

d. Giá trị sau xử lý

- Các thuộc tính được chọn cho mô hình: Hours\_Study, Gaming\_Hours.

- Các thuộc tính bị loại: TV\_Hours, Extra\_Class.

=> Nhờ đó mô hình gọn nhẹ hơn, tránh dư thừa và tập trung vào các yếu tố quan trọng nhất.

e. Code Python để hiện thực xử lý

import pandas as pd

# Tạo dữ liệu mẫu

data = {

"Hours\_Study": [2,3,4,5,6,7,8,9],

"TV\_Hours": [5,6,6,7,7,8,9,9],

"Gaming\_Hours": [6,5,5,4,4,3,2,2],

"Extra\_Class": [1,1,2,2,3,3,4,4],

"Exam\_Score": [40,45,50,65,70,75,85,88]

}

df = pd.DataFrame(data)

# Tính ma trận tương quan

corr\_matrix = df.corr()

print("Ma trận tương quan:")

print(corr\_matrix)

# Chọn thuộc tính có tương quan cao với Exam\_Score

target = "Exam\_Score"

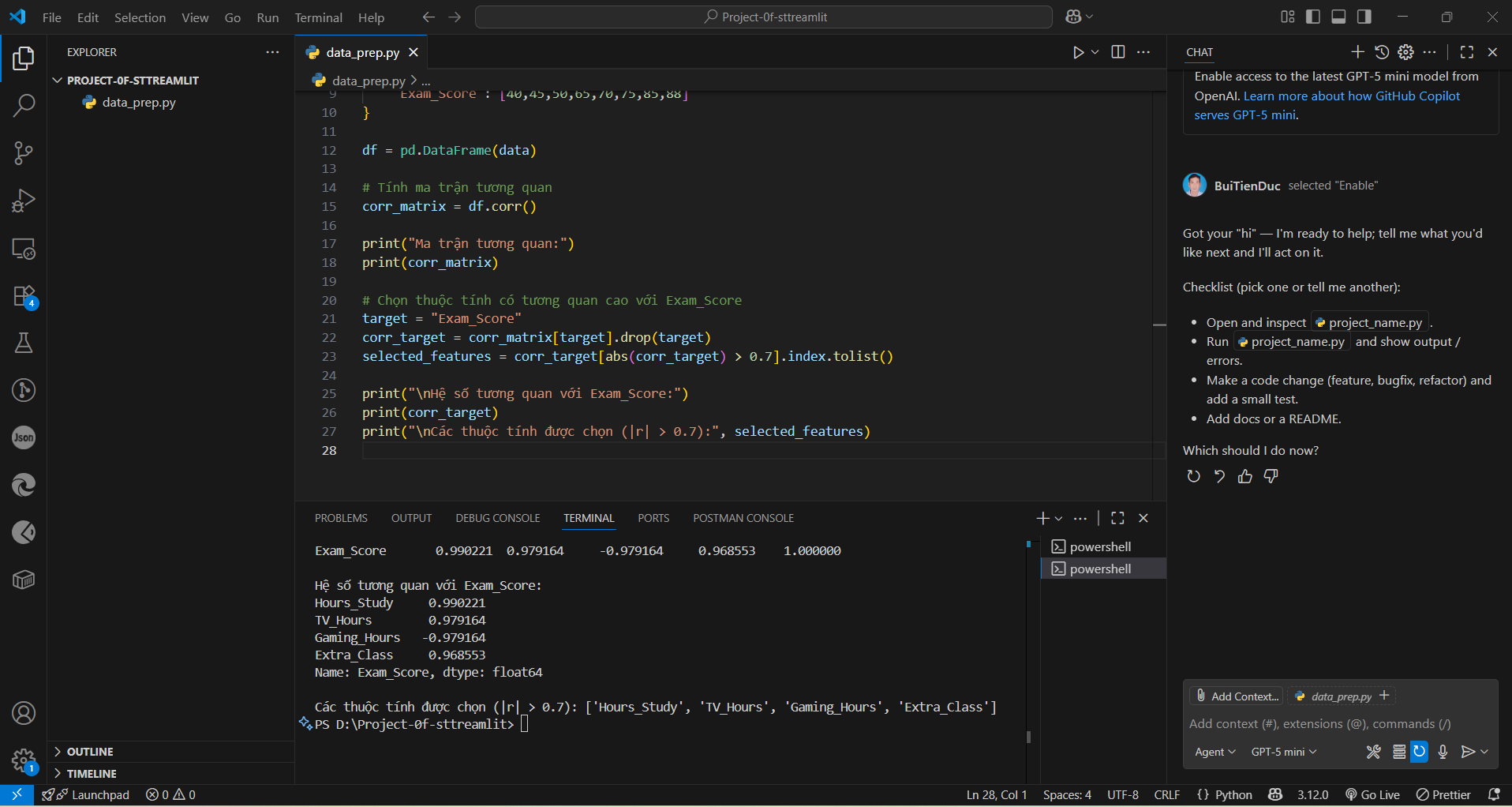
corr\_target = corr\_matrix[target].drop(target)

selected\_features = corr\_target[abs(corr\_target) > 0.7].index.tolist()

print("\nHệ số tương quan với Exam\_Score:")

print(corr\_target)

print("\nCác thuộc tính được chọn (|r| > 0.7):", selected\_features)



Vẽ Histogram để dễ quan sát (biểu đồ histogram để quan sát phân bố dữ liệu)

#Code mẫu

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Tạo dữ liệu mẫu

data = {

    "Hours\_Study": [2,3,4,5,6,7,8,9],

    "TV\_Hours": [5,6,6,7,7,8,9,9],

    "Gaming\_Hours": [6,5,5,4,4,3,2,2],

    "Extra\_Class": [1,1,2,2,3,3,4,4],

    "Exam\_Score": [40,45,50,65,70,75,85,88]

}

df = pd.DataFrame(data)

# Tính ma trận tương quan

corr\_matrix = df.corr()

print("Ma trận tương quan:")

print(corr\_matrix)

# Chọn thuộc tính có tương quan cao với Exam\_Score

target = "Exam\_Score"

corr\_target = corr\_matrix[target].drop(target)

selected\_features = corr\_target[abs(corr\_target) > 0.7].index.tolist()

print("\nHệ số tương quan với Exam\_Score:")

print(corr\_target)

print("\nCác thuộc tính được chọn (|r| > 0.7):", selected\_features)

# Vẽ histogram (Exam\_Score màu đỏ)

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

axes = axes.flatten()

for i, column in enumerate(df.columns):

    color = 'red' if column == "Exam\_Score" else 'blue'

    df[column].hist(bins=5, ax=axes[i], edgecolor='black', color=color)

    axes[i].set\_title(column)

# Ẩn subplot thừa nếu có

for j in range(i+1, len(axes)):

    fig.delaxes(axes[j])

plt.suptitle("Histogram các thuộc tính trong dữ liệu", fontsize=14)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.show()



Thêm đường trung bình (mean) vào từng histogram để dễ phân tích trực quan phân bố dữ liệu so với trung bình

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Tạo dữ liệu mẫu

data = {

    "Hours\_Study": [2,3,4,5,6,7,8,9],

    "TV\_Hours": [5,6,6,7,7,8,9,9],

    "Gaming\_Hours": [6,5,5,4,4,3,2,2],

    "Extra\_Class": [1,1,2,2,3,3,4,4],

    "Exam\_Score": [40,45,50,65,70,75,85,88]

}

df = pd.DataFrame(data)

# Tính ma trận tương quan

corr\_matrix = df.corr()

print("Ma trận tương quan:")

print(corr\_matrix)

# Chọn thuộc tính có tương quan cao với Exam\_Score

target = "Exam\_Score"

corr\_target = corr\_matrix[target].drop(target)

selected\_features = corr\_target[abs(corr\_target) > 0.7].index.tolist()

print("\nHệ số tương quan với Exam\_Score:")

print(corr\_target)

print("\nCác thuộc tính được chọn (|r| > 0.7):", selected\_features)

# Vẽ histogram (Exam\_Score màu đỏ, thêm đường mean)

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

axes = axes.flatten()

for i, column in enumerate(df.columns):

    color = 'red' if column == "Exam\_Score" else 'blue'

    df[column].hist(bins=5, ax=axes[i], edgecolor='black', color=color)

    mean\_val = df[column].mean()

    axes[i].axvline(mean\_val, color='green', linestyle='dashed', linewidth=2)

    axes[i].set\_title(f"{column} (Mean={mean\_val:.2f})")

# Ẩn subplot thừa nếu có

for j in range(i+1, len(axes)):

    fig.delaxes(axes[j])

plt.suptitle("Histogram các thuộc tính trong dữ liệu (có đường mean)", fontsize=14)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.show()



**Mở rộng**

Bài tập: Đánh giá sự ảnh hưởng của thuộc tính lên mô hình phân loại (Feature Importance in Classification)

a. Dữ liệu mẫu (dạng bảng)

StudentID Hours\_Study Gaming\_Hours Extra\_Class Attendance (%) Result

S01 2 6 1 60 Fail

S02 3 5 1 65 Fail

S03 5 4 2 75 Pass

S04 6 3 3 80 Pass

S05 7 2 3 85 Pass

S06 8 2 4 90 Pass

S07 4 5 1 70 Fail

S08 9 1 4 95 Pass

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Trong trích chọn thuộc tính, cần biết thuộc tính nào ảnh hưởng mạnh nhất đến kết quả phân loại.

- Các kỹ thuật đánh giá sự ảnh hưởng (feature importance) thường dùng:

1. Hệ số tương quan.

2. Information Gain (Entropy).

3. Gini Index (Random Forest).

4. Permutation Importance.

5. SHAP values (mạnh mẽ nhất, phổ biến trong Explainable AI).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

- Quan sát bảng dữ liệu:

• Hours\_Study và Attendance cao → dễ Pass.

• Gaming\_Hours nhiều → dễ Fail.

• Extra\_Class bổ sung tín hiệu tích cực nhưng yếu hơn.

- Kết luận sơ bộ:

• Ảnh hưởng mạnh: Hours\_Study, Attendance.

• Ảnh hưởng vừa: Gaming\_Hours (ngược chiều).

• Ảnh hưởng yếu: Extra\_Class.

d. Giá trị sau xử lý

- Các thuộc tính cần giữ: Hours\_Study, Attendance, Gaming\_Hours.

- Có thể loại Extra\_Class nếu dữ liệu lớn mà không cải thiện nhiều.

- Ví dụ kết quả Random Forest Feature Importance:

• Hours\_Study: 0.40

• Attendance: 0.35

• Gaming\_Hours: 0.20

• Extra\_Class: 0.05

e. Code Python để hiện thực xử lý

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

# Dữ liệu mẫu

data = {

"Hours\_Study": [2,3,5,6,7,8,4,9],

"Gaming\_Hours": [6,5,4,3,2,2,5,1],

"Extra\_Class": [1,1,2,3,3,4,1,4],

"Attendance": [60,65,75,80,85,90,70,95],

"Result": ["Fail","Fail","Pass","Pass","Pass","Pass","Fail","Pass"]

}

df = pd.DataFrame(data)

# Chuẩn hóa nhãn

le = LabelEncoder()

df["Result"] = le.fit\_transform(df["Result"]) # Fail=0, Pass=1

X = df.drop("Result", axis=1)

y = df["Result"]

# Huấn luyện mô hình Random Forest

model = RandomForestClassifier(random\_state=42)

model.fit(X, y)

# Trích xuất feature importance

importances = model.feature\_importances\_

feature\_importance = pd.DataFrame({

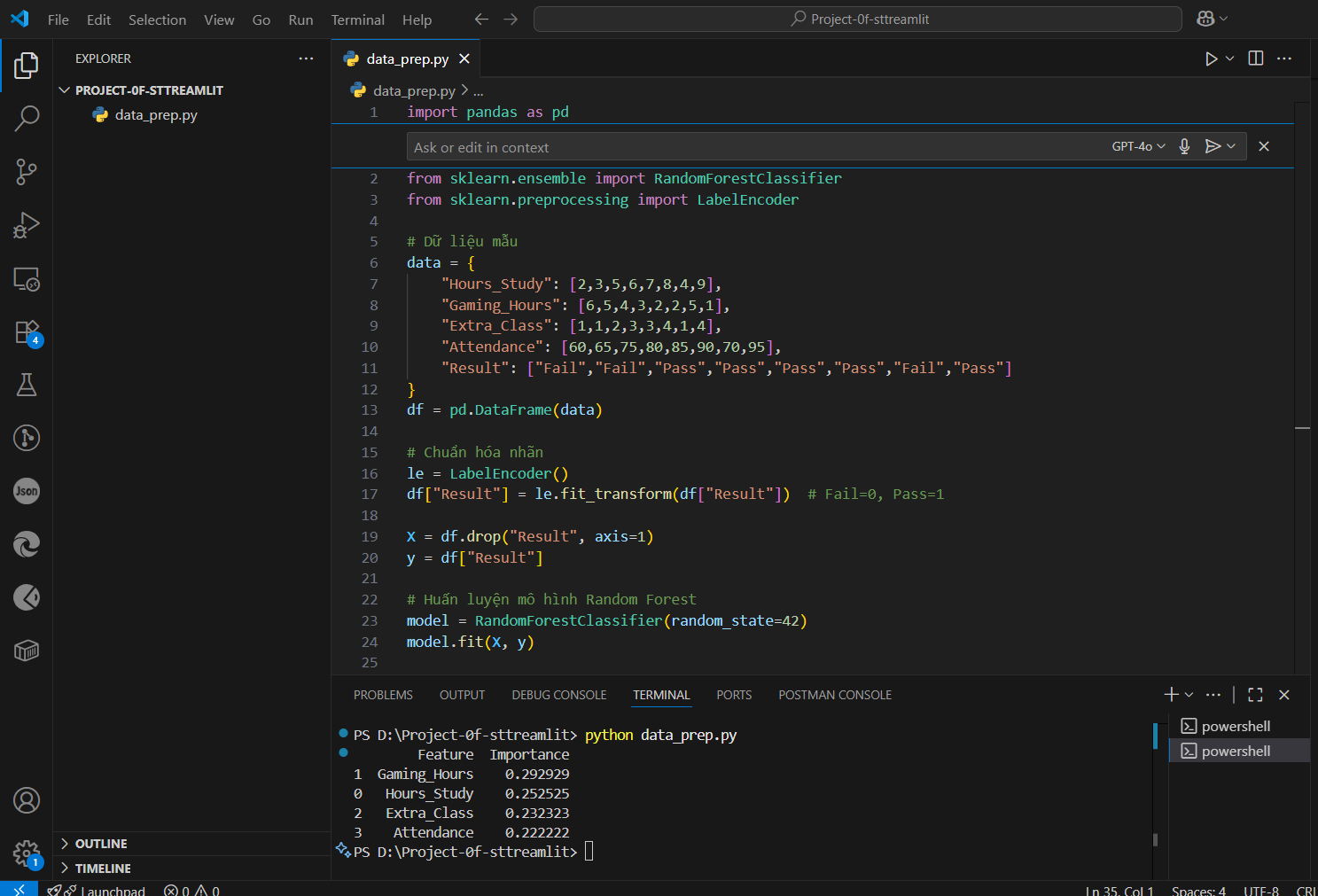
"Feature": X.columns,

"Importance": importances

}).sort\_values(by="Importance", ascending=False)

print(feature\_importance)

**Nhận xét:** Trong thực tế, phương pháp mạnh mẽ nhất hiện nay để đánh giá sự ảnh hưởng của thuộc tính là SHAP values (SHapley Additive exPlanations), vì dựa trên lý thuyết trò chơi Shapley.



Code Python có thêm đoạn vẽ Histogram cho các thuộc tính được chọn (selected\_features) và bỏ qua các thuộc tính không chọn

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt

# Dữ liệu mẫu

data = {

    "Hours\_Study": [2,3,5,6,7,8,4,9],

    "Gaming\_Hours": [6,5,4,3,2,2,5,1],

    "Extra\_Class": [1,1,2,3,3,4,1,4],

    "Attendance": [60,65,75,80,85,90,70,95],

    "Result": ["Fail","Fail","Pass","Pass","Pass","Pass","Fail","Pass"]

}

df = pd.DataFrame(data)

# Chuẩn hóa nhãn

le = LabelEncoder()

df["Result"] = le.fit\_transform(df["Result"])  # Fail=0, Pass=1

X = df.drop("Result", axis=1)

y = df["Result"]

# Huấn luyện mô hình Random Forest

model = RandomForestClassifier(random\_state=42)

model.fit(X, y)

# Trích xuất feature importance

importances = model.feature\_importances\_

feature\_importance = pd.DataFrame({

    "Feature": X.columns,

    "Importance": importances

}).sort\_values(by="Importance", ascending=False)

print("Feature Importance:")

print(feature\_importance)

# Chọn các thuộc tính có importance > 0.1

selected\_features = feature\_importance[feature\_importance["Importance"] > 0.1]["Feature"].tolist()

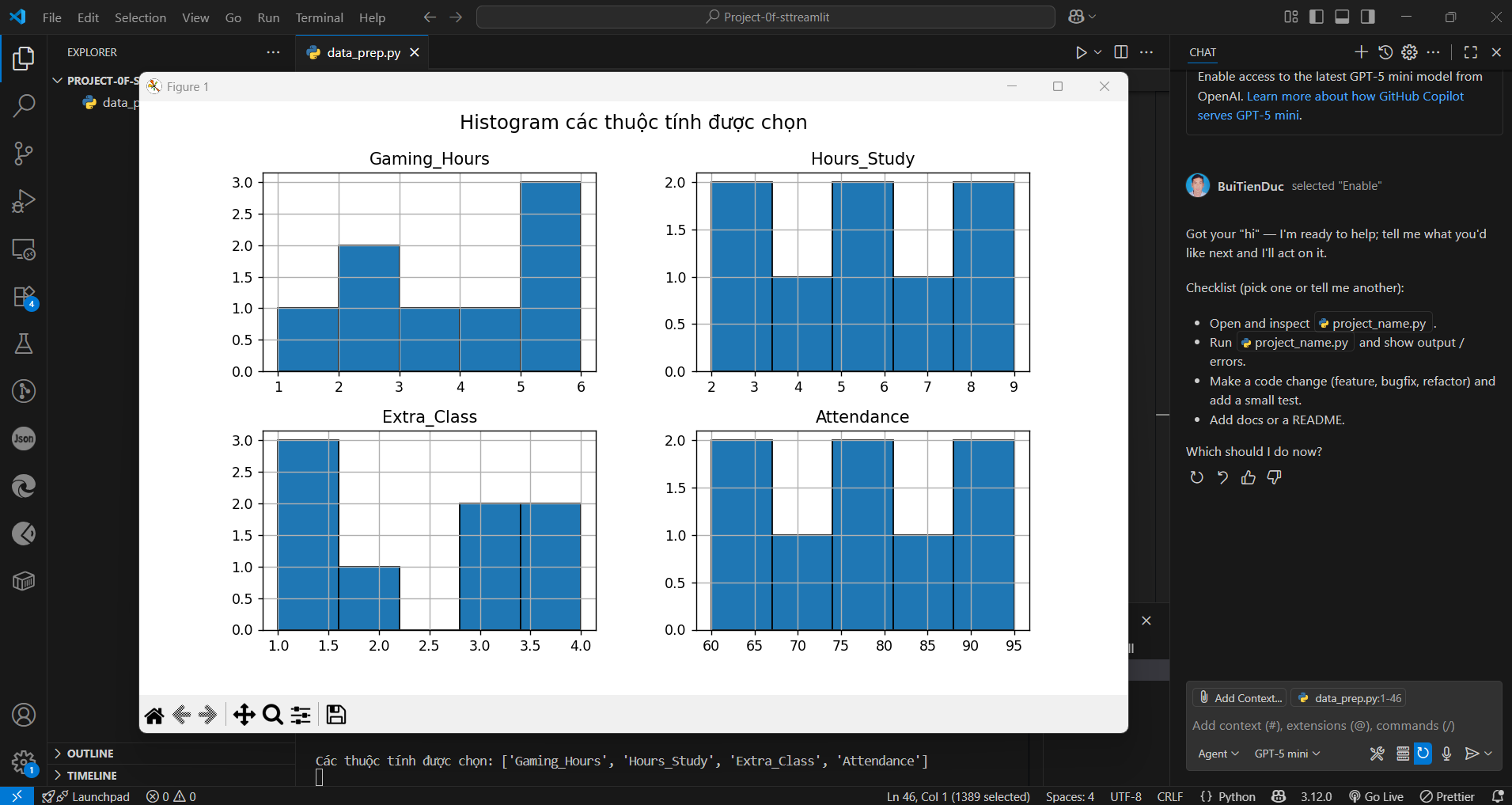
print("\nCác thuộc tính được chọn:", selected\_features)

# Vẽ histogram cho các thuộc tính được chọn

df[selected\_features].hist(bins=5, figsize=(10, 6), edgecolor='black')

plt.suptitle("Histogram các thuộc tính được chọn", fontsize=14)

plt.show()



**Quay lại Làm tiền xử lý dữ liệu tiếp** 😊

Phép kiểm thống kê chi-square kiểm tra giả thuyết liệu A và B có độc lập với nhau dựa trên một mức quan trọng (significance level) với độ tự do (degree of freedom).

Nếu giả thuyết bị loại bỏ thì A và B có sự liên hệ với nhau dựa trên thống kê.

Bài tập: Kiểm định Chi-square cho sự độc lập giữa hai thuộc tính

a. Dữ liệu mẫu dạng bảng

Gender Purchased=Yes Purchased=No

Male 40 30

Female 20 50

b. Định nghĩa vấn đề trong dữ liệu mẫu

Bài toán: Kiểm tra xem hai thuộc tính (Gender và Purchased) có độc lập với nhau không.

- Giả thuyết H0: Gender và Purchased độc lập.

- Giả thuyết H1: Gender và Purchased không độc lập.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

- Tính tổng theo hàng, cột và tổng toàn bộ.

- Sử dụng công thức giá trị kỳ vọng E = (Tổng hàng \* Tổng cột) / Tổng.

- Tính thống kê chi-square: χ² = Σ ( (O - E)² / E ).

- So sánh với giá trị tới hạn (critical value) dựa trên mức ý nghĩa α và bậc tự do (df).

- df = (số hàng - 1) \* (số cột - 1).

d. Giá trị sau xử lý

- Nếu p-value < α (ví dụ α = 0.05), bác bỏ H0 → Gender và Purchased có liên hệ.

- Nếu p-value ≥ α, không bác bỏ H0 → Gender và Purchased độc lập.

e. Code Python để hiện thực xử lý

import pandas as pd

import scipy.stats as stats

# Dữ liệu mẫu

data = [[40, 30],

        [20, 50]]

df = pd.DataFrame(data, columns=['Purchased=Yes', 'Purchased=No'], index=['Male','Female'])

# Thực hiện kiểm định Chi-square

chi2, p, dof, expected = stats.chi2\_contingency(df)

print("Giá trị Chi-square:", chi2)

print("p-value:", p)

print("Bậc tự do:", dof)

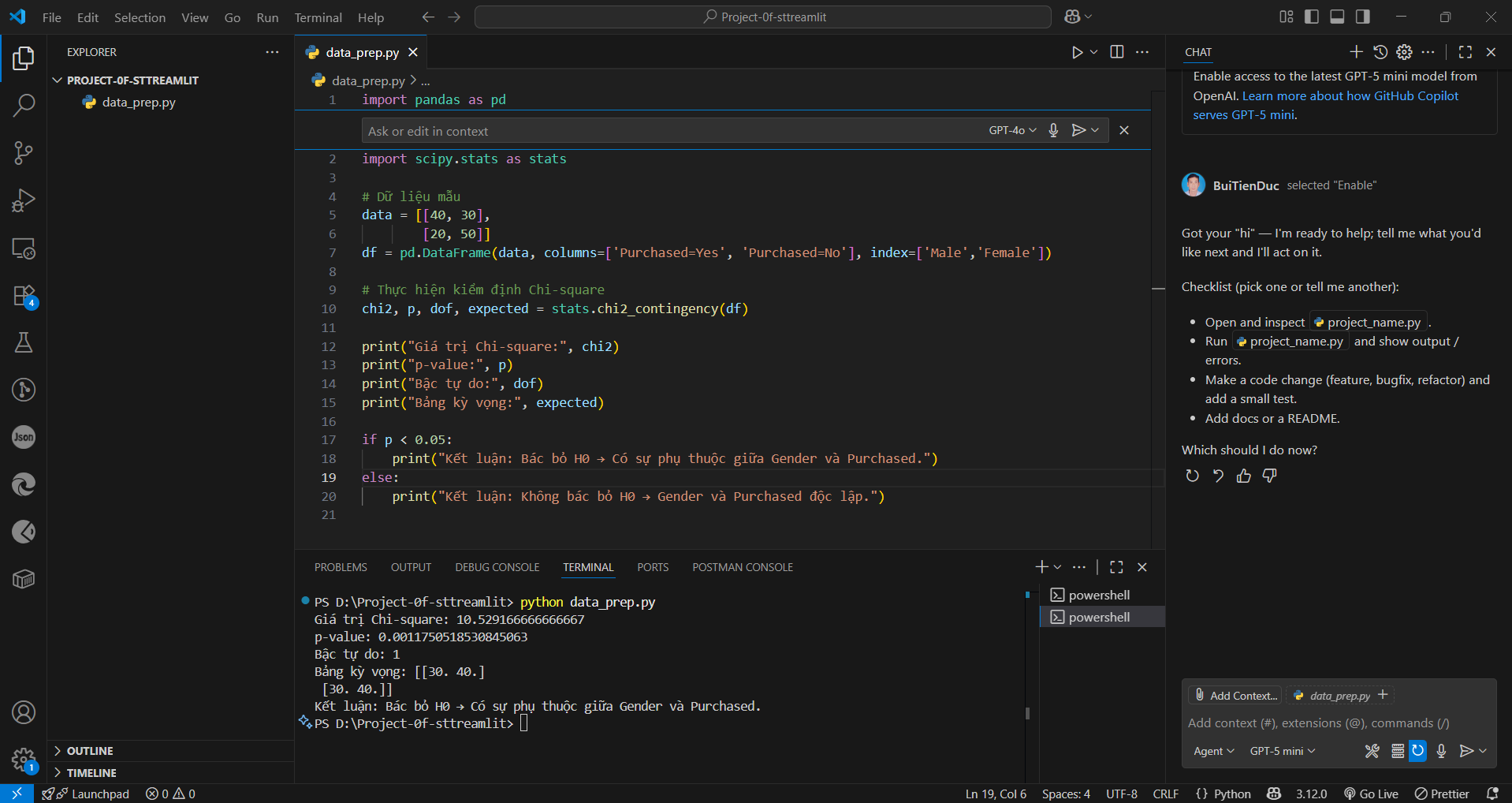
print("Bảng kỳ vọng:", expected)

if p < 0.05:

    print("Kết luận: Bác bỏ H0 → Có sự phụ thuộc giữa Gender và Purchased.")

else:

    print("Kết luận: Không bác bỏ H0 → Gender và Purchased độc lập.")



Bổ sung luôn phần so sánh với giá trị tới hạn (critical value) để hiểu rõ hơn cách ra quyết định, thay vì chỉ dựa vào p-value

#code python

import pandas as pd

import scipy.stats as stats

# Dữ liệu mẫu

data = [[40, 30],

        [20, 50]]

df = pd.DataFrame(data, columns=['Purchased=Yes', 'Purchased=No'], index=['Male','Female'])

# Thực hiện kiểm định Chi-square

chi2, p, dof, expected = stats.chi2\_contingency(df)

print("Giá trị Chi-square:", chi2)

print("p-value:", p)

print("Bậc tự do:", dof)

print("Bảng kỳ vọng:\n", expected)

# Mức ý nghĩa

alpha = 0.05

# Giá trị tới hạn từ phân phối Chi-square

critical\_value = stats.chi2.ppf(1 - alpha, dof)

print("Giá trị tới hạn (critical value) ở mức α=0.05:", critical\_value)

# Ra quyết định

if chi2 > critical\_value:

    print("Kết luận (theo critical value): Bác bỏ H0 → Có sự phụ thuộc giữa Gender và Purchased.")

else:

    print("Kết luận (theo critical value): Không bác bỏ H0 → Gender và Purchased độc lập.")

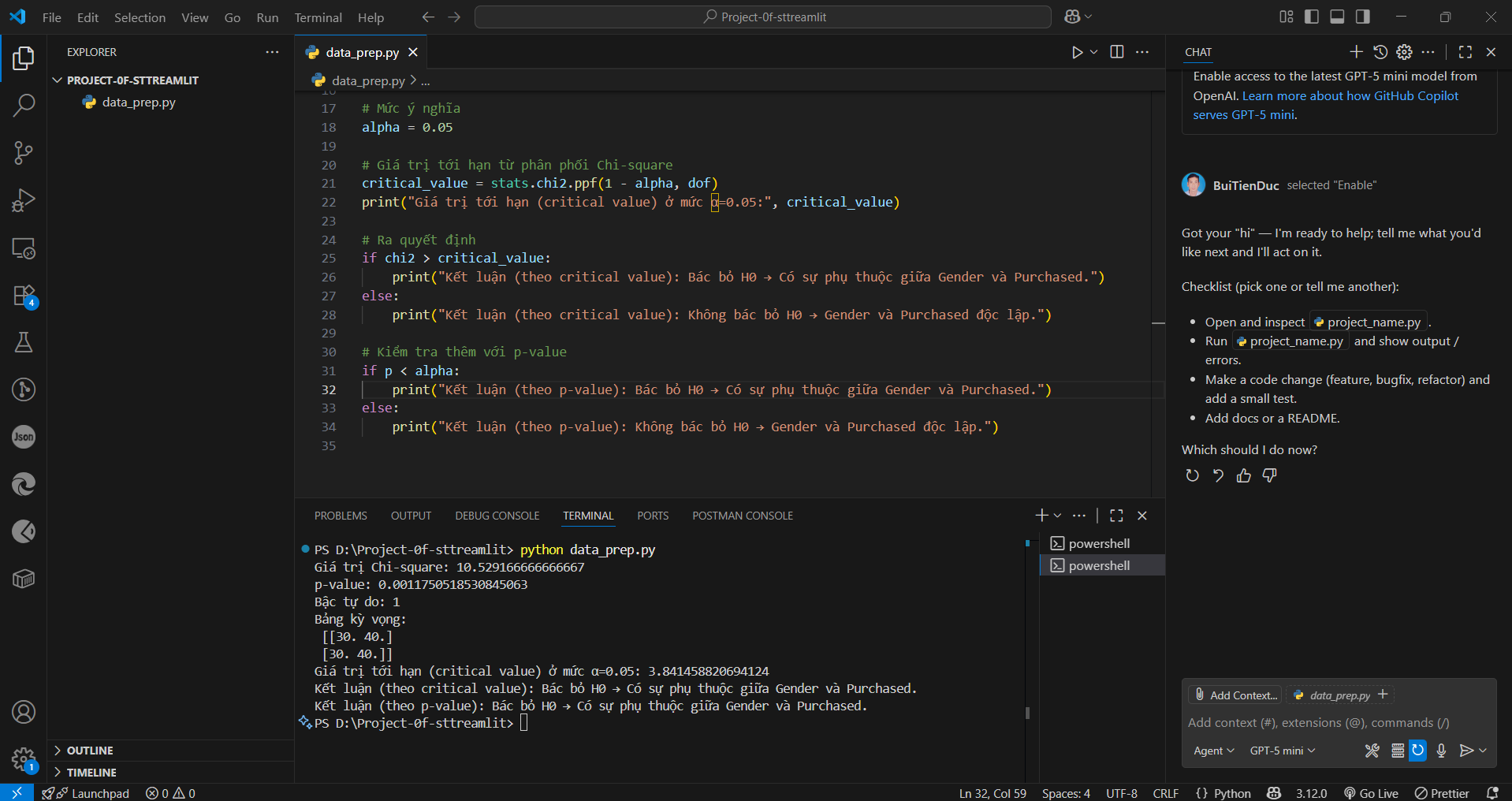
# Kiểm tra thêm với p-value

if p < alpha:

    print("Kết luận (theo p-value): Bác bỏ H0 → Có sự phụ thuộc giữa Gender và Purchased.")

else:

    print("Kết luận (theo p-value): Không bác bỏ H0 → Gender và Purchased độc lập.")



Độ tự do (degree of freedom): (r-1)\*(c-1)

Tra bảng phân bố chi-square để xác định giá trị x2

Nếu giá trị tính toán được nhỏ hơn hay bằng trị tra bảng được thì hai thuộc tính A và B độc lập nhau (giả thuyết đúng).

Kiểm định Chi-square – Độ tự do và so sánh giá trị tới hạn

a. Dữ liệu mẫu (dạng bảng)

Gender Thích sản phẩm X Không thích Row total

Male 25 15 40

Female 20 40 60

Col total 45 55 100

b. Định nghĩa vấn đề trong dữ liệu mẫu

Kiểm định giả thuyết:

- H0: Hai thuộc tính Gender và Product Preference độc lập.

- H1: Hai thuộc tính có quan hệ.

Độ tự do df = (r-1)(c-1) = (2-1)(2-1) = 1.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Tính giá trị kỳ vọng (Expected frequencies):

- Male-Yes: (40\*45)/100 = 18

- Male-No: (40\*55)/100 = 22

- Female-Yes: (60\*45)/100 = 27

- Female-No: (60\*55)/100 = 33

2. Tính thống kê Chi-square:

χ² = Σ (O-E)²/E = 2.72 + 2.23 + 1.81 + 1.48 = 8.24

3. So sánh với giá trị tới hạn:

- Với df=1, α=0.05 → χ²\_critical = 3.841.

- Vì 8.24 > 3.841 ⇒ bác bỏ H0.

d. Giá trị sau xử lý

- Thống kê tính toán χ² ≈ 8.24

- Giá trị tới hạn 3.841

- Kết luận: Gender và Product Preference có quan hệ, không độc lập.

e. Code Python để hiện thực xử lý

import pandas as pd

from scipy.stats import chi2\_contingency, chi2

# Bảng quan sát

data = [[25, 15],

[20, 40]]

df = pd.DataFrame(data, columns=['Yes', 'No'], index=['Male', 'Female'])

# Kiểm định Chi-square

chi2\_stat, p, dof, expected = chi2\_contingency(df, correction=False)

print("Chi-square statistic:", chi2\_stat)

print("p-value:", p)

print("Degrees of freedom:", dof)

print("Expected frequencies:\n", expected)

# Giá trị tới hạn với alpha = 0.05

alpha = 0.05

critical\_value = chi2.ppf(1 - alpha, dof)

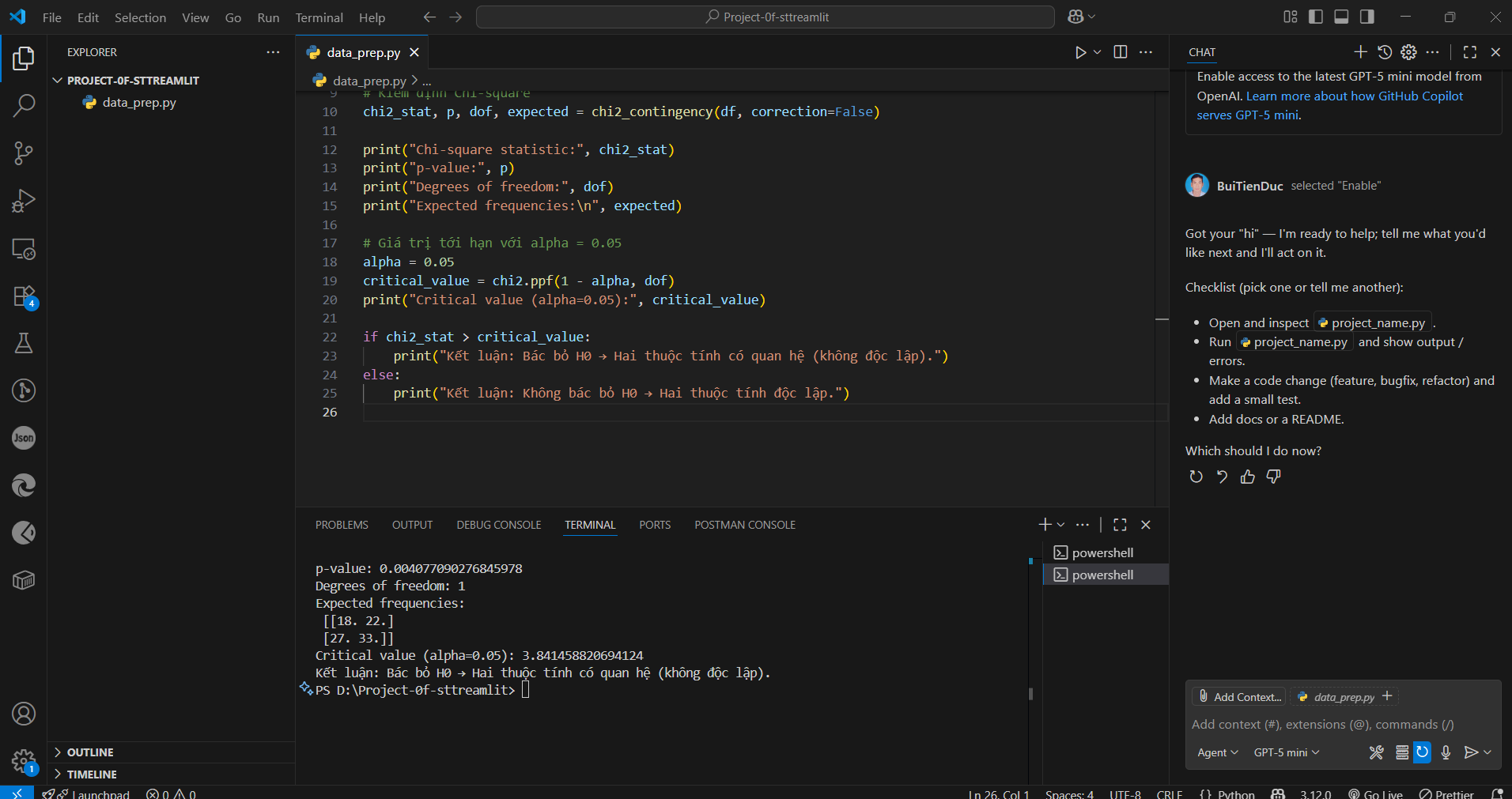
print("Critical value (alpha=0.05):", critical\_value)

if chi2\_stat > critical\_value:

print("Kết luận: Bác bỏ H0 → Hai thuộc tính có quan hệ (không độc lập).")

else:

print("Kết luận: Không bác bỏ H0 → Hai thuộc tính độc lập.")



**B.3. Biến đổi dữ liệu**

Biến đổi dữ liệu (data transformation): chuẩn hoá dữ liệu (data normalization)

Biến đổi dữ liệu (data transformation)

1. Làm trơn dữ liệu (smoothing)

Bài tập: Biến đổi dữ liệu – Làm trơn dữ liệu (Smoothing)

a. Dữ liệu mẫu (dạng bảng)

Tuần Doanh số

1 52

2 55

3 58

4 90

5 54

6 53

7 52

8 91

9 59

10 58

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu quan sát có thể chứa nhiễu (noise) hoặc biến động quá lớn. Ví dụ: tuần 4 và 8 có giá trị đột biến (90, 91) so với xu hướng chung. Cần áp dụng kỹ thuật làm trơn (smoothing) để giảm nhiễu và phản ánh xu hướng chính xác hơn.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Sử dụng phương pháp Moving Average với cửa sổ 3 tuần:

- Tuần 2 = (52+55+58)/3 = 55.0

- Tuần 3 = (55+58+90)/3 ≈ 67.7

- Tuần 4 = (58+90+54)/3 ≈ 67.3

- Tuần 5 = (90+54+53)/3 ≈ 65.7

- Tuần 6 = (54+53+52)/3 ≈ 53.0

- Tuần 7 = (53+52+91)/3 ≈ 65.3

- Tuần 8 = (52+91+59)/3 ≈ 67.3

- Tuần 9 = (91+59+58)/3 ≈ 69.3

d. Giá trị sau xử lý

Tuần Doanh số (gốc) Doanh số (sau smoothing – MA-3)

1 52 -

2 55 55.0

3 58 67.7

4 90 67.3

5 54 65.7

6 53 53.0

7 52 65.3

8 91 67.3

9 59 69.3

10 58 -

Dữ liệu sau làm trơn phản ánh xu hướng chung tốt hơn, giảm tác động của các giá trị đột biến.

e. Code Python để hiện thực xử lý

import pandas as pd

import matplotlib.pyplot as plt

# Dữ liệu mẫu

data = {

"Week": [1,2,3,4,5,6,7,8,9,10],

"Sales": [52,55,58,90,54,53,52,91,59,58]

}

df = pd.DataFrame(data)

# Làm trơn bằng Moving Average (cửa sổ 3 tuần)

df["Smoothed"] = df["Sales"].rolling(window=3).mean()

print("Dữ liệu sau smoothing:")

print(df)

# Vẽ biểu đồ trực quan

plt.plot(df["Week"], df["Sales"], label="Gốc", marker="o")

plt.plot(df["Week"], df["Smoothed"], label="Smoothed (MA-3)", marker="x")

plt.xlabel("Tuần")

plt.ylabel("Doanh số")

plt.title("Smoothing doanh số bằng Moving Average")

plt.legend()

plt.show()



1. Kết hợp dữ liệu (aggregation)

Bài tập: Biến đổi dữ liệu – Kết hợp dữ liệu (Aggregation)

a. Dữ liệu mẫu (dạng bảng)

Ngày Doanh số (triệu VND)

01/01 5.2

02/01 4.8

03/01 6.0

04/01 6.5

05/01 5.5

06/01 7.0

07/01 6.2

08/01 4.9

09/01 5.1

10/01 5.8

11/01 6.4

12/01 6.8

13/01 7.2

14/01 6.9

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu ở mức chi tiết hàng ngày → khó quan sát xu hướng. Ta cần kết hợp dữ liệu (aggregation) để tổng hợp thành mức tuần (weekly sales). Lợi ích: giảm số lượng bản ghi, phát hiện xu hướng và so sánh dễ hơn.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Chia dữ liệu thành 2 tuần:

- Tuần 1 (01/01–07/01): 5.2 + 4.8 + 6.0 + 6.5 + 5.5 + 7.0 + 6.2 = 41.2

- Tuần 2 (08/01–14/01): 4.9 + 5.1 + 5.8 + 6.4 + 6.8 + 7.2 + 6.9 = 43.1

d. Giá trị sau xử lý

Tuần Doanh số tổng (triệu VND)

1 41.2

2 43.1

Sau khi kết hợp dữ liệu, ta dễ dàng so sánh xu hướng giữa các tuần hơn.

e. Code Python để hiện thực xử lý

import pandas as pd

import matplotlib.pyplot as plt

# Dữ liệu mẫu

data = {

"Date": pd.date\_range("2023-01-01", periods=14, freq="D"),

"Sales": [5.2, 4.8, 6.0, 6.5, 5.5, 7.0, 6.2,

4.9, 5.1, 5.8, 6.4, 6.8, 7.2, 6.9]

}

df = pd.DataFrame(data)

# Thêm cột 'Week' để nhóm

df["Week"] = df["Date"].dt.isocalendar().week

# Kết hợp dữ liệu theo tuần

weekly\_sales = df.groupby("Week")["Sales"].sum().reset\_index()

print("Dữ liệu sau khi kết hợp (aggregation):")

print(weekly\_sales)

# Vẽ biểu đồ

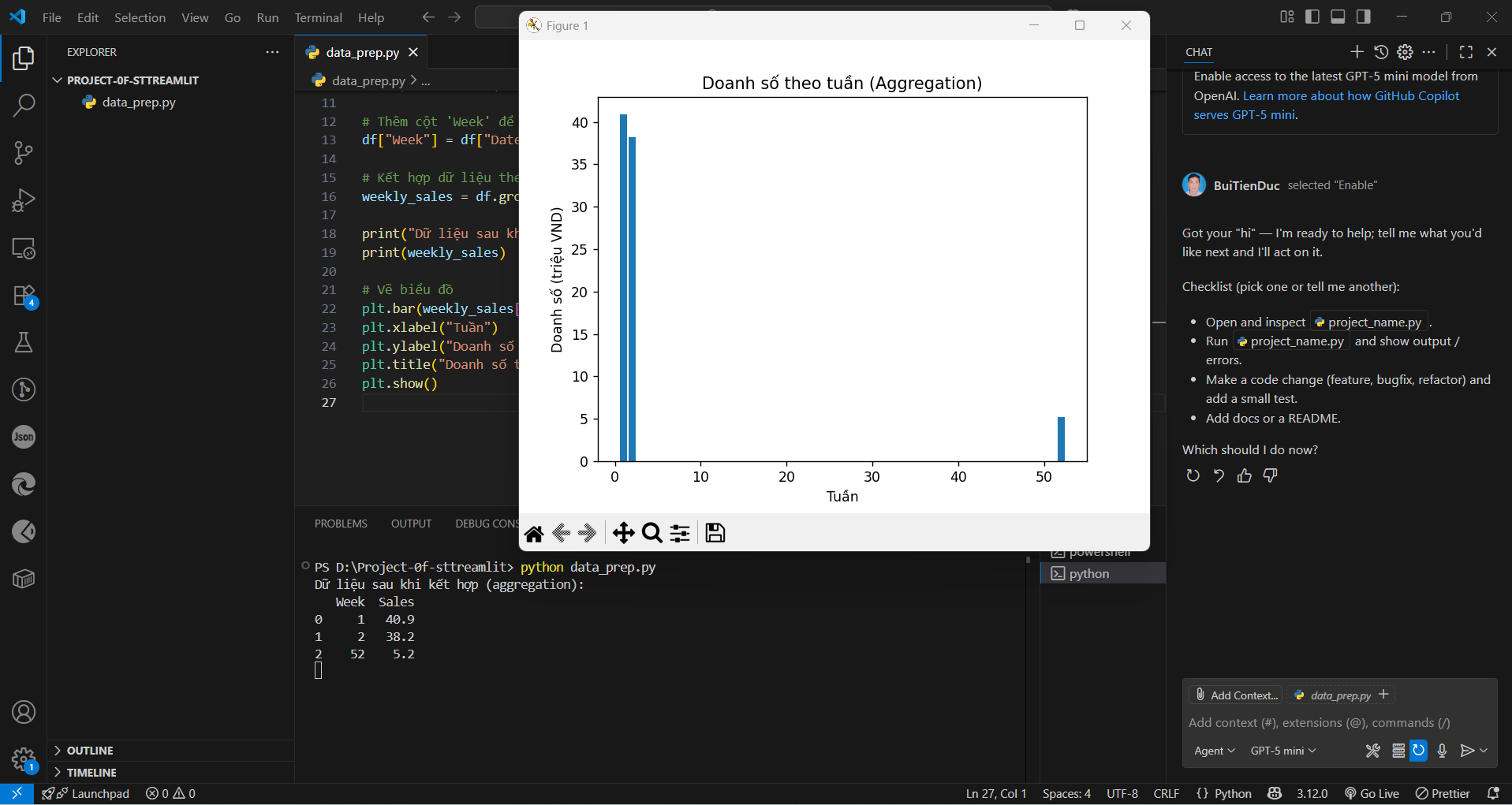
plt.bar(weekly\_sales["Week"], weekly\_sales["Sales"])

plt.xlabel("Tuần")

plt.ylabel("Doanh số (triệu VND)")

plt.title("Doanh số theo tuần (Aggregation)")

plt.show()



1. Tổng quát hóa dữ liệu (generalization)

Bài tập: Biến đổi dữ liệu – Tổng quát hóa dữ liệu (Generalization)

a. Dữ liệu mẫu (dạng bảng)

CustomerID Tuổi Nghề nghiệp Thành phố Quốc gia

C001 23 Sinh viên Hà Nội Việt Nam

C002 35 Kỹ sư IT TP.HCM Việt Nam

C003 42 Giáo viên Đà Nẵng Việt Nam

C004 29 Nhân viên bán hàng Cần Thơ Việt Nam

C005 55 Bác sĩ Hà Nội Việt Nam

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu đang ở mức chi tiết (tuổi số, nghề nghiệp cụ thể, thành phố). Để khai phá xu hướng chung, cần tổng quát hóa dữ liệu: Tuổi → nhóm tuổi, Nghề nghiệp → nhóm nghề, Thành phố → vùng.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

- Tuổi → nhóm: 20–29 = Thanh niên, 30–39 = Trung niên, 40–49 = Trung cao niên, 50+ = Cao tuổi.

- Nghề nghiệp → lĩnh vực: Sinh viên, Giáo viên → Giáo dục; Kỹ sư IT → Công nghệ; Nhân viên bán hàng → Thương mại; Bác sĩ → Y tế.

- Thành phố → Vùng: Hà Nội = Miền Bắc; Đà Nẵng = Miền Trung; TP.HCM, Cần Thơ = Miền Nam.

d. Giá trị sau xử lý

CustomerID Nhóm tuổi Nhóm nghề nghiệp Vùng Quốc gia

C001 Thanh niên Giáo dục Miền Bắc Việt Nam

C002 Trung niên Công nghệ Miền Nam Việt Nam

C003 Trung cao niên Giáo dục Miền Trung Việt Nam

C004 Thanh niên Thương mại Miền Nam Việt Nam

C005 Cao tuổi Y tế Miền Bắc Việt Nam

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"CustomerID": ["C001","C002","C003","C004","C005"],

"Age": [23, 35, 42, 29, 55],

"Job": ["Sinh viên","Kỹ sư IT","Giáo viên","Nhân viên bán hàng","Bác sĩ"],

"City": ["Hà Nội","TP.HCM","Đà Nẵng","Cần Thơ","Hà Nội"],

"Country": ["Việt Nam"]\*5

}

df = pd.DataFrame(data)

# Tổng quát hóa Age → AgeGroup

def age\_group(age):

if 20 <= age <= 29: return "Thanh niên"

elif 30 <= age <= 39: return "Trung niên"

elif 40 <= age <= 49: return "Trung cao niên"

else: return "Cao tuổi"

df["AgeGroup"] = df["Age"].apply(age\_group)

# Tổng quát hóa Job → JobCategory

job\_map = {

"Sinh viên": "Giáo dục",

"Giáo viên": "Giáo dục",

"Kỹ sư IT": "Công nghệ",

"Nhân viên bán hàng": "Thương mại",

"Bác sĩ": "Y tế"

}

df["JobCategory"] = df["Job"].map(job\_map)

# Tổng quát hóa City → Region

city\_map = {

"Hà Nội": "Miền Bắc",

"Đà Nẵng": "Miền Trung",

"TP.HCM": "Miền Nam",

"Cần Thơ": "Miền Nam"

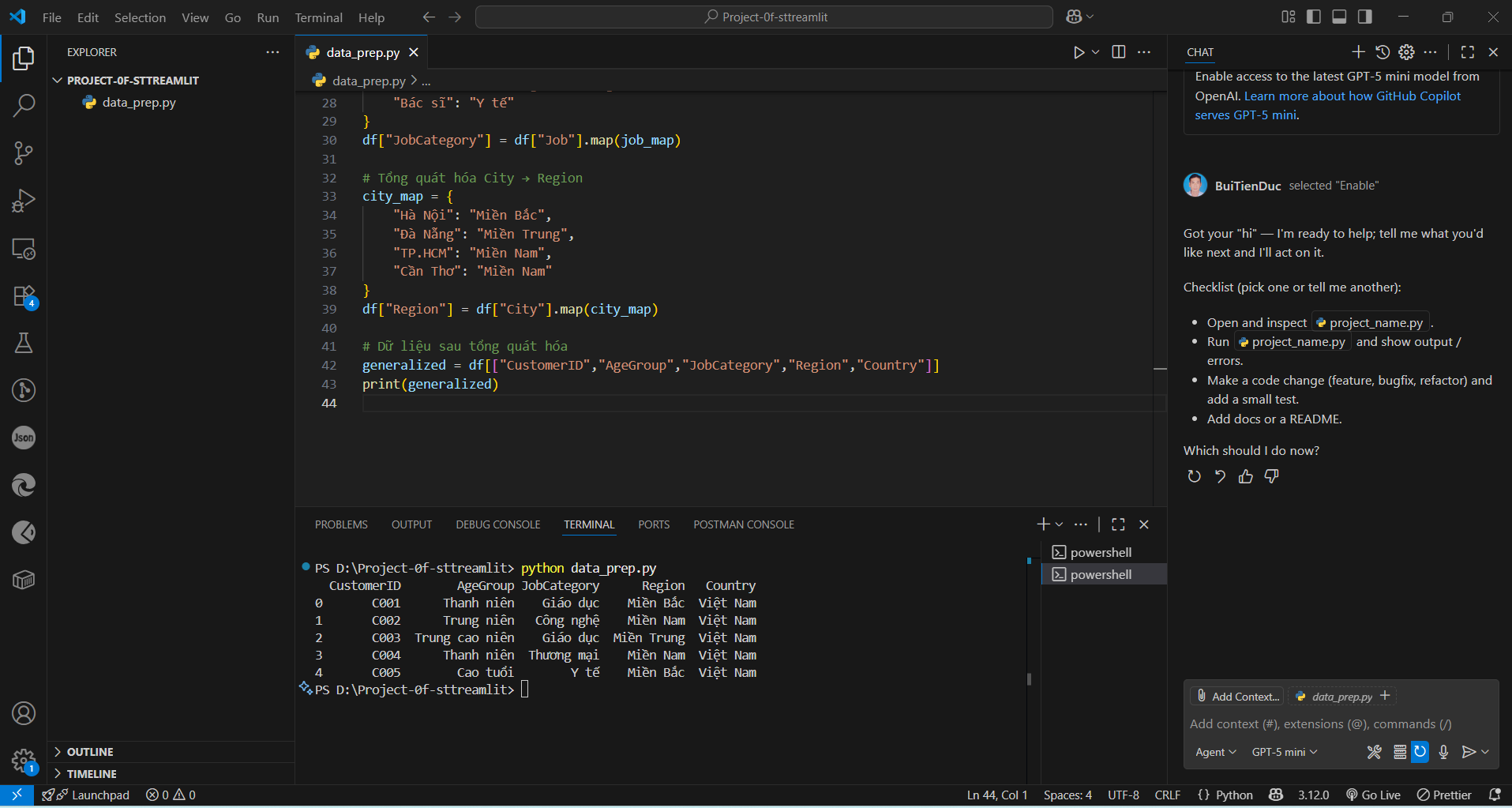
}

df["Region"] = df["City"].map(city\_map)

# Dữ liệu sau tổng quát hóa

generalized = df[["CustomerID","AgeGroup","JobCategory","Region","Country"]]

print(generalized)



1. Chuẩn hóa dữ liệu (normalization)
   1. min-max normalization

Bài tập: Biến đổi dữ liệu – Chuẩn hóa dữ liệu (Normalization) bằng Min-Max

a. Dữ liệu mẫu (dạng bảng)

StudentID Toán Lý Hóa

S01 65 78 82

S02 90 88 95

S03 72 92 85

S04 55 60 58

S05 80 85 88

b. Định nghĩa vấn đề trong dữ liệu mẫu

Các thuộc tính (Toán, Lý, Hóa) có thang đo khác nhau (55–95). Một số thuật toán (K-means, Neural Networks) nhạy cảm với độ lớn dữ liệu, nên cần chuẩn hóa. Sử dụng Min-Max Normalization để đưa về [0,1].

Công thức: x' = (x - min(x)) / (max(x) - min(x))

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Ví dụ với cột Toán (min=55, max=90):

- S01: (65-55)/35 = 0.286

- S02: (90-55)/35 = 1.000

- S03: (72-55)/35 = 0.486

- S04: (55-55)/35 = 0.000

- S05: (80-55)/35 = 0.714

Tương tự cho Lý (min=60, max=92) và Hóa (min=58, max=95).

d. Giá trị sau xử lý

StudentID Toán (norm) Lý (norm) Hóa (norm)

S01 0.286 0.562 0.649

S02 1.000 0.875 1.000

S03 0.486 1.000 0.730

S04 0.000 0.000 0.000

S05 0.714 0.781 0.811

e. Code Python để hiện thực xử lý

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04","S05"],

"Math": [65, 90, 72, 55, 80],

"Physics": [78, 88, 92, 60, 85],

"Chemistry": [82, 95, 85, 58, 88]

}

df = pd.DataFrame(data)

# Áp dụng Min-Max Normalization

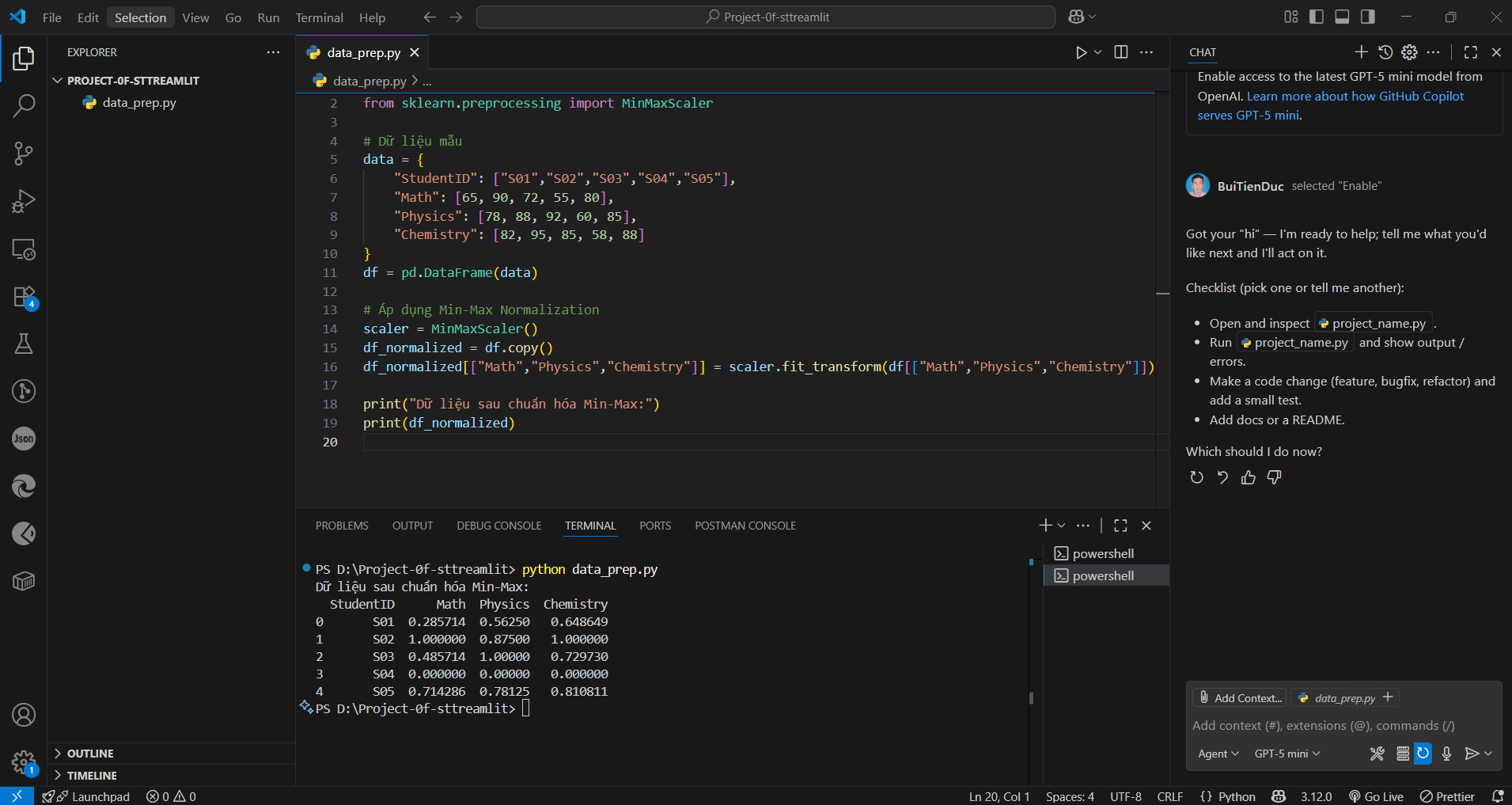
scaler = MinMaxScaler()

df\_normalized = df.copy()

df\_normalized[["Math","Physics","Chemistry"]] = scaler.fit\_transform(df[["Math","Physics","Chemistry"]])

print("Dữ liệu sau chuẩn hóa Min-Max:")

print(df\_normalized)



* 1. z-score normalization

Bài tập: Biến đổi dữ liệu – Chuẩn hóa dữ liệu (Normalization) bằng Z-Score

a. Dữ liệu mẫu (dạng bảng)

StudentID Toán Lý Hóa

S01 65 78 82

S02 90 88 95

S03 72 92 85

S04 55 60 58

S05 80 85 88

b. Định nghĩa vấn đề trong dữ liệu mẫu

Các thuộc tính (Toán, Lý, Hóa) có phân bố và thang đo khác nhau. Một số thuật toán học máy yêu cầu dữ liệu chuẩn hóa quanh giá trị trung bình 0 và độ lệch chuẩn 1. Z-Score Normalization được dùng để chuẩn hóa:

x' = (x - μ) / σ

Trong đó μ là giá trị trung bình, σ là độ lệch chuẩn.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Ví dụ với cột Toán:

- Mean = 72.4, Std ≈ 12.76

- S01: (65-72.4)/12.76 ≈ -0.58

- S02: (90-72.4)/12.76 ≈ 1.38

- S03: (72-72.4)/12.76 ≈ -0.03

- S04: (55-72.4)/12.76 ≈ -1.36

- S05: (80-72.4)/12.76 ≈ 0.59

Tương tự với cột Lý và Hóa.

d. Giá trị sau xử lý

StudentID Toán (z-score) Lý (z-score) Hóa (z-score)

S01 -0.58 -0.02 -0.02

S02 1.38 0.62 1.19

S03 -0.03 0.86 0.29

S04 -1.36 -1.69 -1.58

S05 0.59 0.23 0.12

e. Code Python để hiện thực xử lý

import pandas as pd

from sklearn.preprocessing import StandardScaler

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04","S05"],

"Math": [65, 90, 72, 55, 80],

"Physics": [78, 88, 92, 60, 85],

"Chemistry": [82, 95, 85, 58, 88]

}

df = pd.DataFrame(data)

# Áp dụng Z-score Normalization

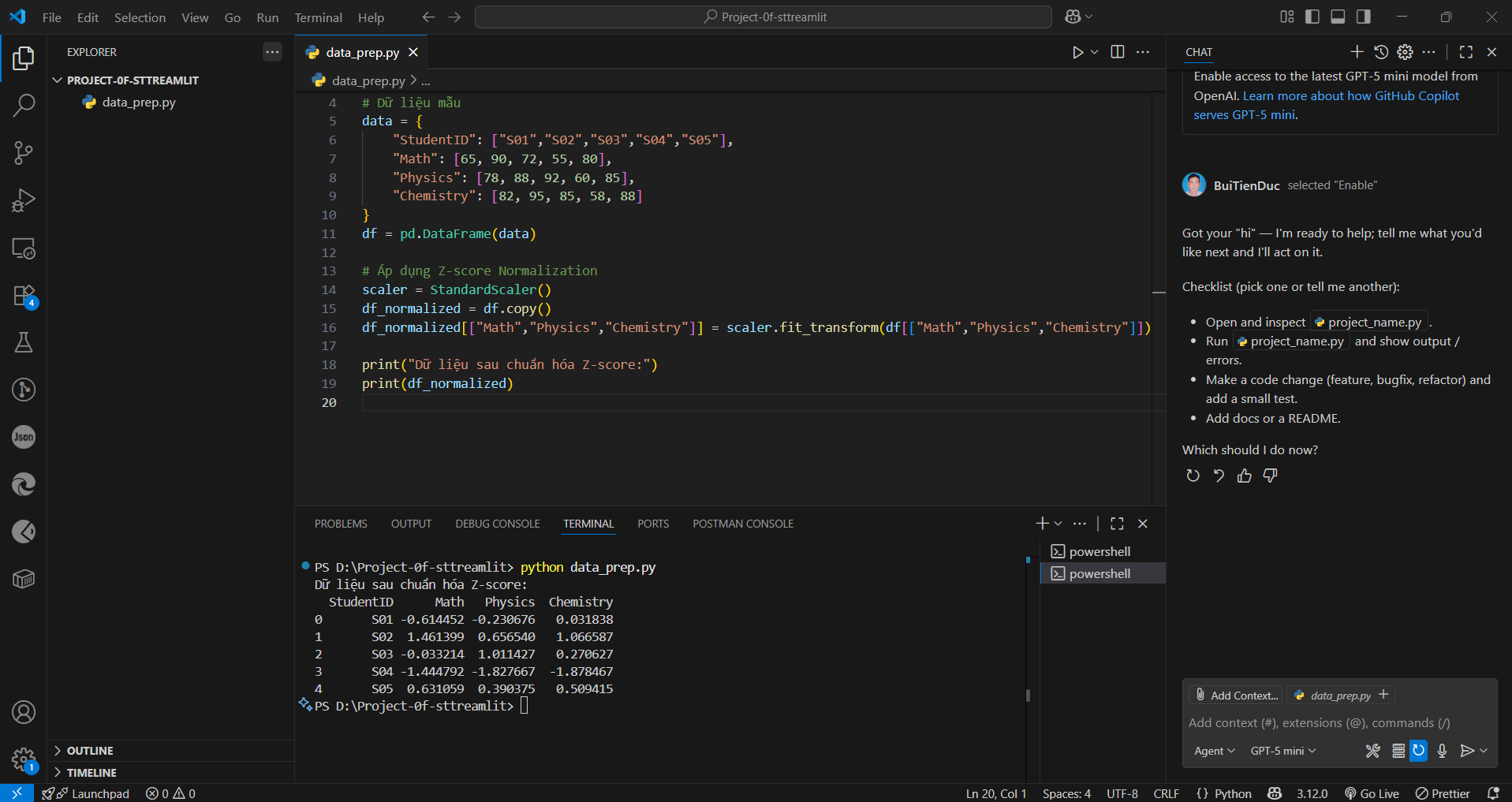
scaler = StandardScaler()

df\_normalized = df.copy()

df\_normalized[["Math","Physics","Chemistry"]] = scaler.fit\_transform(df[["Math","Physics","Chemistry"]])

print("Dữ liệu sau chuẩn hóa Z-score:")

print(df\_normalized)



* 1. Normalization by decimal scaling

Bài tập: Biến đổi dữ liệu – Chuẩn hóa dữ liệu (Normalization) bằng Decimal Scaling

a. Dữ liệu mẫu (dạng bảng)

StudentID Toán Lý Hóa

S01 65 78 82

S02 90 88 95

S03 72 92 85

S04 55 60 58

S05 800 850 920

b. Định nghĩa vấn đề trong dữ liệu mẫu

Một số thuộc tính có giá trị quá lớn (ví dụ: Toán=800) khiến thang đo không đồng đều. Decimal Scaling Normalization chuẩn hóa dữ liệu bằng cách chia cho 10^j, với j là số chữ số sao cho giá trị chuẩn hóa có trị tuyệt đối nhỏ hơn 1.

Công thức: x' = x / 10^j

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Với cột Toán (max=800 → j=3):

- S01: 65/1000 = 0.065

- S02: 90/1000 = 0.090

- S03: 72/1000 = 0.072

- S04: 55/1000 = 0.055

- S05: 800/1000 = 0.800

Tương tự cho Lý (max=850 → j=3) và Hóa (max=920 → j=3).

d. Giá trị sau xử lý

StudentID Toán (decimal norm) Lý (decimal norm) Hóa (decimal norm)

S01 0.065 0.078 0.082

S02 0.090 0.088 0.095

S03 0.072 0.092 0.085

S04 0.055 0.060 0.058

S05 0.800 0.850 0.920

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04","S05"],

"Math": [65, 90, 72, 55, 800],

"Physics": [78, 88, 92, 60, 850],

"Chemistry": [82, 95, 85, 58, 920]

}

df = pd.DataFrame(data)

# Decimal Scaling Normalization

df\_normalized = df.copy()

for col in ["Math","Physics","Chemistry"]:

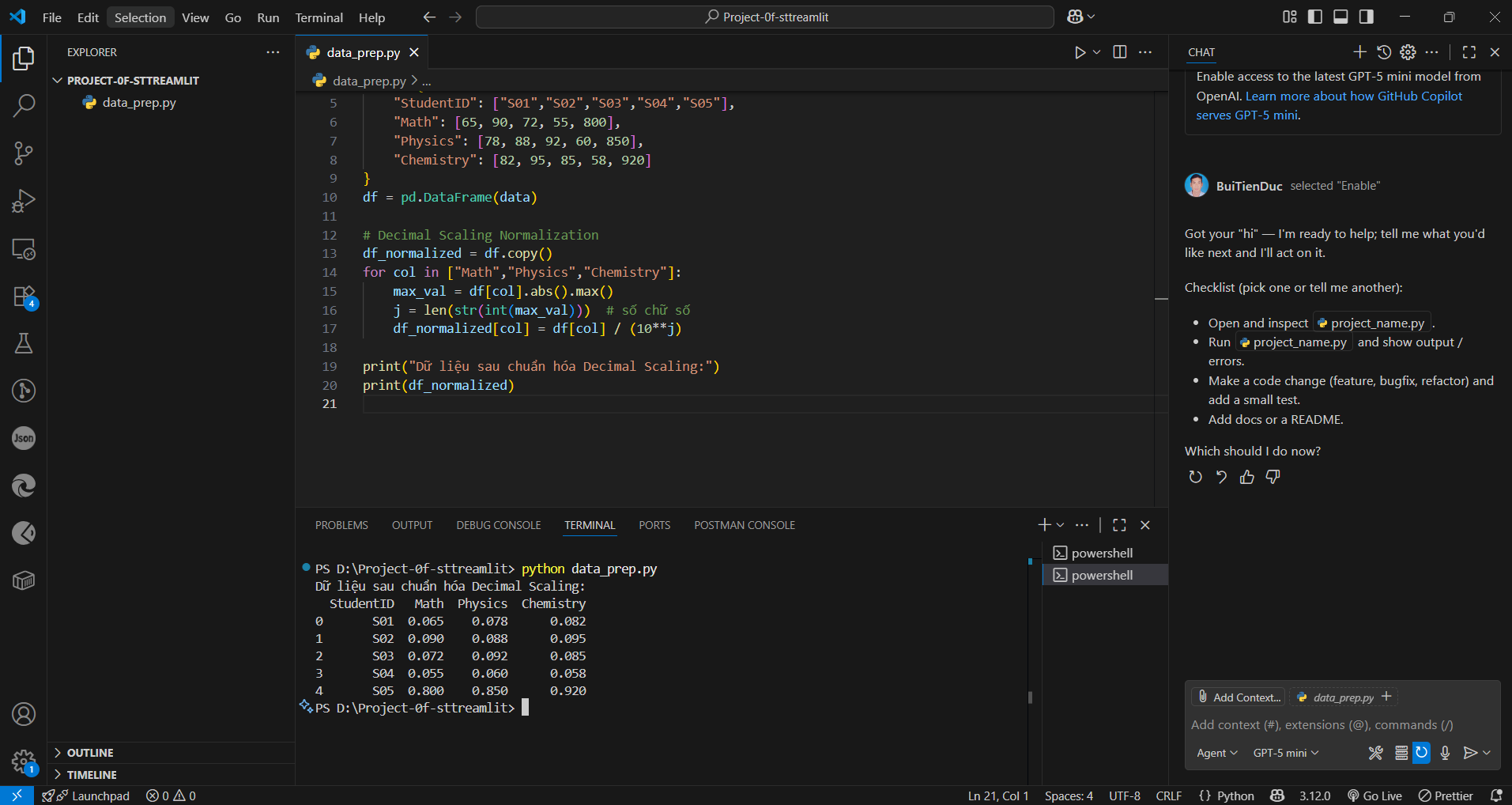
max\_val = df[col].abs().max()

j = len(str(int(max\_val))) # số chữ số

df\_normalized[col] = df[col] / (10\*\*j)

print("Dữ liệu sau chuẩn hóa Decimal Scaling:")

print(df\_normalized)



* Các giá trị thuộc tính được chuyển đổi vào một miền trị nhất định được định nghĩa trước.

1. Xây dựng thuộc tính/đặc tính (attribute/feature construction)

Bài tập: Biến đổi dữ liệu – Xây dựng thuộc tính (Feature Construction)

a. Dữ liệu mẫu (dạng bảng)

StudentID Họ Tên Năm sinh Điểm Toán Điểm Lý Điểm Hóa

S01 Nguyễn An 2002 7.5 8.0 7.0

S02 Trần Bình 2001 9.0 8.5 9.2

S03 Lê Cường 2003 6.5 7.0 6.8

S04 Phạm Dung 2002 8.0 8.2 7.8

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu hiện tại chỉ có các thuộc tính gốc (họ, tên, năm sinh, điểm từng môn). Để phục vụ phân tích hoặc xây dựng mô hình tốt hơn, cần xây dựng thêm các thuộc tính mới như: tuổi, điểm trung bình, học lực, họ tên đầy đủ.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Tuổi = 2025 - Năm sinh → S01: 23, S02: 24, S03: 22, S04: 23.

2. Điểm trung bình = (Toán+Lý+Hóa)/3 → S01: 7.5, S02: 8.9, S03: 6.8, S04: 8.0.

3. Học lực (dựa vào điểm TB): ≥8.5 = Giỏi; ≥7.0 = Khá; ≥5.0 = Trung bình; <5.0 = Yếu.

→ S01: Khá, S02: Giỏi, S03: Trung bình, S04: Khá.

4. Họ tên đầy đủ = Họ + ' ' + Tên.

d. Giá trị sau xử lý

StudentID Họ tên đầy đủ Tuổi Điểm TB Học lực

S01 Nguyễn An 23 7.5 Khá

S02 Trần Bình 24 8.9 Giỏi

S03 Lê Cường 22 6.8 Trung bình

S04 Phạm Dung 23 8.0 Khá

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04"],

"Ho": ["Nguyễn","Trần","Lê","Phạm"],

"Ten": ["An","Bình","Cường","Dung"],

"NamSinh": [2002,2001,2003,2002],

"Toan": [7.5, 9.0, 6.5, 8.0],

"Ly": [8.0, 8.5, 7.0, 8.2],

"Hoa": [7.0, 9.2, 6.8, 7.8]

}

df = pd.DataFrame(data)

# Xây dựng thuộc tính mới

df["HoTen"] = df["Ho"] + " " + df["Ten"]

df["Tuoi"] = 2025 - df["NamSinh"]

df["DiemTB"] = df[["Toan","Ly","Hoa"]].mean(axis=1)

def xep\_loai(diem):

if diem >= 8.5: return "Giỏi"

elif diem >= 7.0: return "Khá"

elif diem >= 5.0: return "Trung bình"

else: return "Yếu"

df["HocLuc"] = df["DiemTB"].apply(xep\_loai)

# Kết quả

print(df[["StudentID","HoTen","Tuoi","DiemTB","HocLuc"]])

**B.4. Thu giảm dữ liệu**

Thu giảm dữ liệu (data reduction): thu giảm kích thước dữ liệu (nghĩa là giảm số phần tử) bằng kết hợp dữ liệu (data aggregation), loại bỏ các đặcđiểm dư thừa (redundant features) (nghĩa là giảm số chiều/ thuộc tính dữ liệu), gom cụm dữ liệu

Thu giảm dữ liệu (data reduction)

1. Kết hợp khối dữ liệu (data cube aggregation)

Bài tập: Thu giảm dữ liệu – Kết hợp khối dữ liệu (Data Cube Aggregation)

a. Dữ liệu mẫu (dạng bảng)

Quý Vùng Sản phẩm Doanh thu (triệu VND)

Q1 Bắc Laptop 120

Q1 Bắc Điện thoại 100

Q1 Nam Laptop 150

Q1 Nam Điện thoại 130

Q2 Bắc Laptop 140

Q2 Bắc Điện thoại 110

Q2 Nam Laptop 160

Q2 Nam Điện thoại 120

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu hiện tại ở mức chi tiết (Quý – Vùng – Sản phẩm). Để thu giảm dữ liệu và quan sát xu hướng tổng quan hơn, ta dùng Data Cube Aggregation để tổng hợp: 1) Tổng doanh thu theo Quý và Vùng (bỏ chi tiết sản phẩm), 2) Tổng doanh thu theo Quý (bỏ chi tiết vùng và sản phẩm).

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

1. Theo Quý – Vùng:

- Q1 – Bắc = 120 + 100 = 220

- Q1 – Nam = 150 + 130 = 280

- Q2 – Bắc = 140 + 110 = 250

- Q2 – Nam = 160 + 120 = 280

2. Theo Quý (tổng tất cả vùng và sản phẩm):

- Q1 = 220 + 280 = 500

- Q2 = 250 + 280 = 530

d. Giá trị sau xử lý

\*\*Tổng hợp theo Quý – Vùng\*\*

Quý Vùng Doanh thu (triệu VND)

Q1 Bắc 220

Q1 Nam 280

Q2 Bắc 250

Q2 Nam 280

\*\*Tổng hợp theo Quý\*\*

Quý Doanh thu (triệu VND)

Q1 500

Q2 530

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"Quy": ["Q1","Q1","Q1","Q1","Q2","Q2","Q2","Q2"],

"Vung": ["Bac","Bac","Nam","Nam","Bac","Bac","Nam","Nam"],

"SanPham": ["Laptop","Dien thoai","Laptop","Dien thoai",

"Laptop","Dien thoai","Laptop","Dien thoai"],

"DoanhThu": [120,100,150,130,140,110,160,120]

}

df = pd.DataFrame(data)

# Tổng hợp theo Quý – Vùng

agg\_quy\_vung = df.groupby(["Quy","Vung"])["DoanhThu"].sum().reset\_index()

# Tổng hợp theo Quý

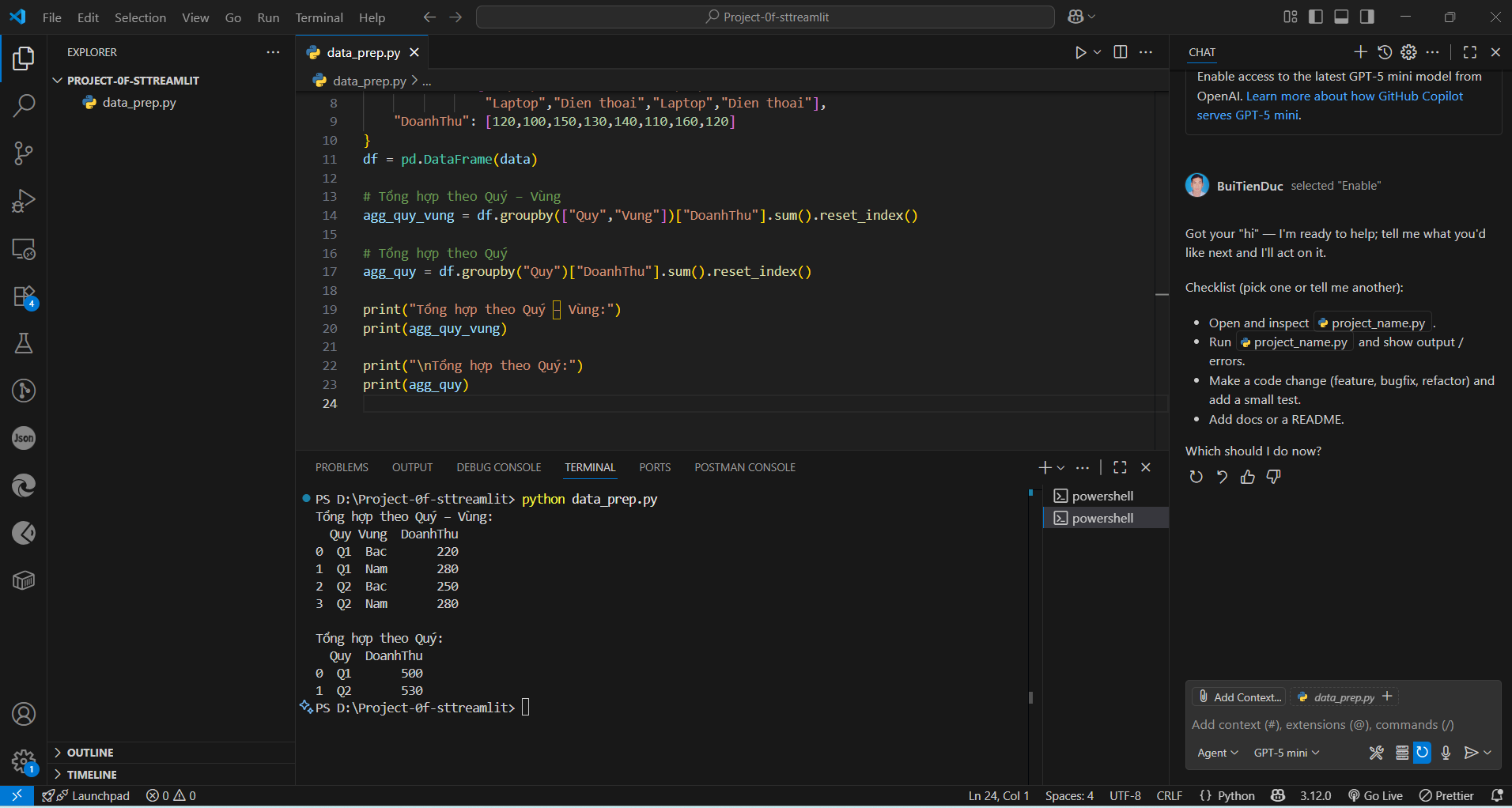
agg\_quy = df.groupby("Quy")["DoanhThu"].sum().reset\_index()

print("Tổng hợp theo Quý – Vùng:")

print(agg\_quy\_vung)

print("\nTổng hợp theo Quý:")

print(agg\_quy)



1. Chọn tập con các thuộc tính (attribute subset selection)

Bài tập: Thu giảm dữ liệu – Chọn tập con các thuộc tính (Attribute Subset Selection)

a. Dữ liệu mẫu (dạng bảng)

CustomerID Tuổi Giới tính Thu nhập (triệu VND) Nghề nghiệp Thành phố Quốc gia

C001 23 Nam 15 Sinh viên Hà Nội Việt Nam

C002 35 Nữ 25 Kỹ sư TP.HCM Việt Nam

C003 42 Nam 40 Giáo viên Đà Nẵng Việt Nam

C004 29 Nữ 18 Bán hàng Cần Thơ Việt Nam

C005 55 Nam 60 Bác sĩ Hà Nội Việt Nam

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu có nhiều thuộc tính, nhưng không phải tất cả đều quan trọng cho mục tiêu phân tích. Ví dụ: nếu mục tiêu là dự đoán Thu nhập, thì thuộc tính Quốc gia (toàn bộ giống nhau) không mang thông tin. Do đó cần chọn tập con các thuộc tính quan trọng nhất, giúp giảm chiều dữ liệu, tăng hiệu quả mô hình và giảm nhiễu.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

- Thuộc tính quan trọng để dự đoán Thu nhập: Tuổi, Giới tính, Nghề nghiệp, Thành phố.

- Thuộc tính kém quan trọng: CustomerID (chỉ định danh), Quốc gia (không thay đổi).

→ Giữ lại 4 thuộc tính quan trọng nhất cùng với Thu nhập.

d. Giá trị sau xử lý

Tuổi Giới tính Nghề nghiệp Thành phố Thu nhập (triệu VND)

23 Nam Sinh viên Hà Nội 15

35 Nữ Kỹ sư TP.HCM 25

42 Nam Giáo viên Đà Nẵng 40

29 Nữ Bán hàng Cần Thơ 18

55 Nam Bác sĩ Hà Nội 60

e. Code Python để hiện thực xử lý

import pandas as pd

# Dữ liệu mẫu

data = {

"CustomerID": ["C001","C002","C003","C004","C005"],

"Age": [23,35,42,29,55],

"Gender": ["Nam","Nữ","Nam","Nữ","Nam"],

"Income": [15,25,40,18,60],

"Job": ["Sinh viên","Kỹ sư","Giáo viên","Bán hàng","Bác sĩ"],

"City": ["Hà Nội","TP.HCM","Đà Nẵng","Cần Thơ","Hà Nội"],

"Country": ["Việt Nam"]\*5

}

df = pd.DataFrame(data)

print("Dữ liệu gốc:")

print(df)

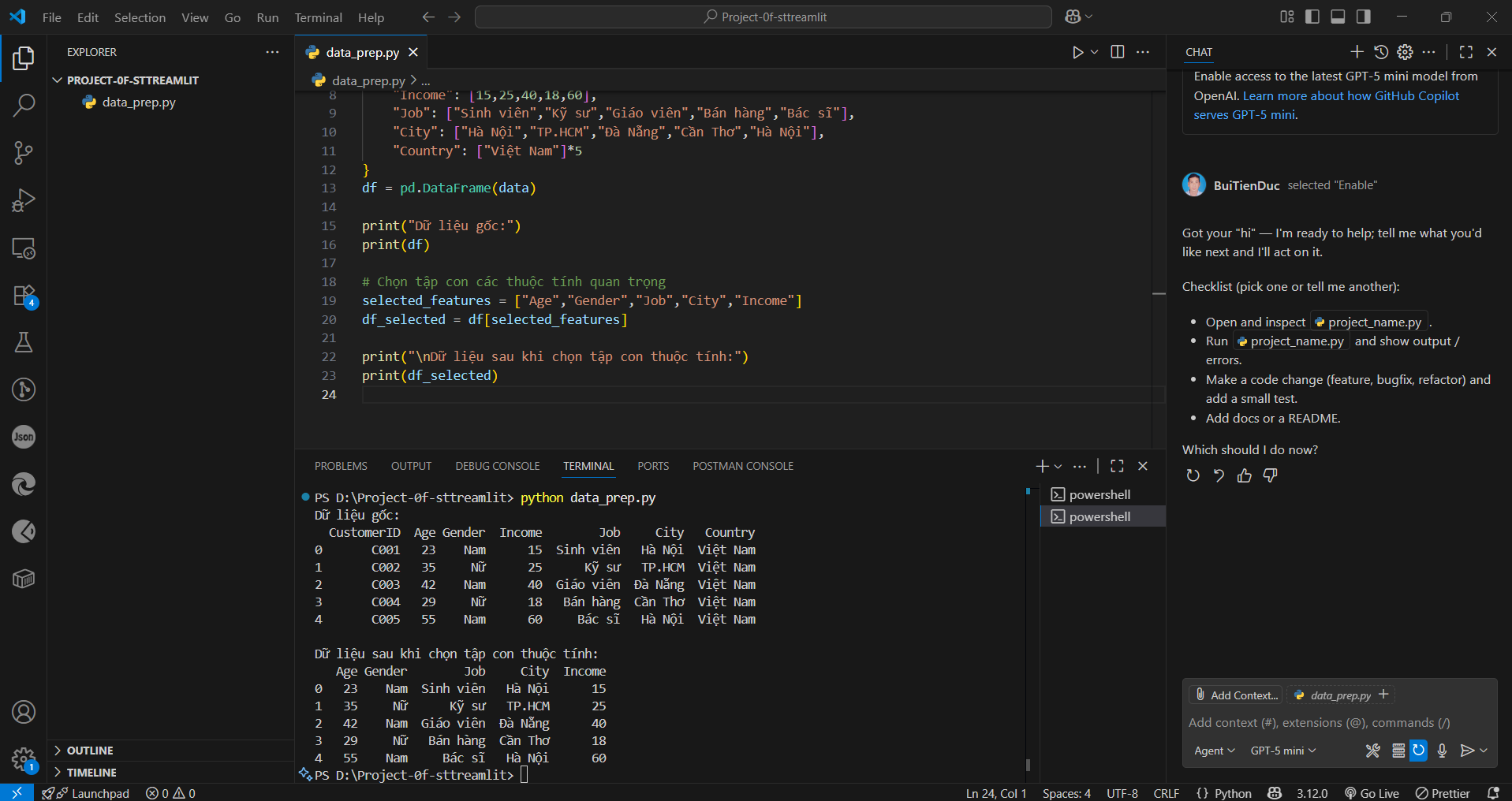
# Chọn tập con các thuộc tính quan trọng

selected\_features = ["Age","Gender","Job","City","Income"]

df\_selected = df[selected\_features]

print("\nDữ liệu sau khi chọn tập con thuộc tính:")

print(df\_selected)



1. Thu giảm chiều (dimensionality reduction)

Bài tập: Thu giảm dữ liệu – Thu giảm chiều (Dimensionality Reduction)

a. Dữ liệu mẫu (dạng bảng)

StudentID Toán Lý Hóa Sinh Văn Sử Địa Anh

S01 7.5 8.0 7.0 7.8 6.5 6.0 6.2 7.0

S02 9.0 8.5 9.2 8.8 7.0 7.5 7.8 8.5

S03 6.5 7.0 6.8 6.9 6.2 6.5 6.8 6.5

S04 8.0 8.2 7.8 8.0 7.2 7.0 7.5 8.0

S05 5.5 6.0 5.8 5.9 6.0 5.5 5.8 6.2

b. Định nghĩa vấn đề trong dữ liệu mẫu

Bộ dữ liệu có nhiều chiều (8 môn học). Một số môn có tương quan cao (ví dụ: Toán – Lý – Hóa, hoặc Văn – Sử – Địa). Giữ toàn bộ thuộc tính gây tăng chi phí tính toán và có thể gây nhiễu. Dimensionality Reduction giúp giảm số chiều nhưng vẫn giữ được thông tin quan trọng.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

- Gom nhóm Toán, Lý, Hóa, Sinh thành thuộc tính Tự nhiên (Natural).

- Gom nhóm Văn, Sử, Địa thành thuộc tính Xã hội (Social).

- Giữ nguyên môn Anh.

Ví dụ S01: Tự nhiên = (7.5+8.0+7.0+7.8)/4 = 7.6; Xã hội = (6.5+6.0+6.2)/3 ≈ 6.23; Anh=7.0.

d. Giá trị sau xử lý

StudentID Tự nhiên Xã hội Anh

S01 7.6 6.23 7.0

S02 8.9 7.43 8.5

S03 6.8 6.5 6.5

S04 8.0 7.23 8.0

S05 5.8 5.77 6.2

e. Code Python để hiện thực xử lý

import pandas as pd

from sklearn.decomposition import PCA

# Dữ liệu mẫu

data = {

"StudentID": ["S01","S02","S03","S04","S05"],

"Math": [7.5,9.0,6.5,8.0,5.5],

"Physics": [8.0,8.5,7.0,8.2,6.0],

"Chemistry": [7.0,9.2,6.8,7.8,5.8],

"Biology": [7.8,8.8,6.9,8.0,5.9],

"Literature": [6.5,7.0,6.2,7.2,6.0],

"History": [6.0,7.5,6.5,7.0,5.5],

"Geography": [6.2,7.8,6.8,7.5,5.8],

"English": [7.0,8.5,6.5,8.0,6.2]

}

df = pd.DataFrame(data)

# Áp dụng PCA để giảm chiều còn 3 thành phần chính

features = ["Math","Physics","Chemistry","Biology","Literature","History","Geography","English"]

X = df[features]

pca = PCA(n\_components=3)

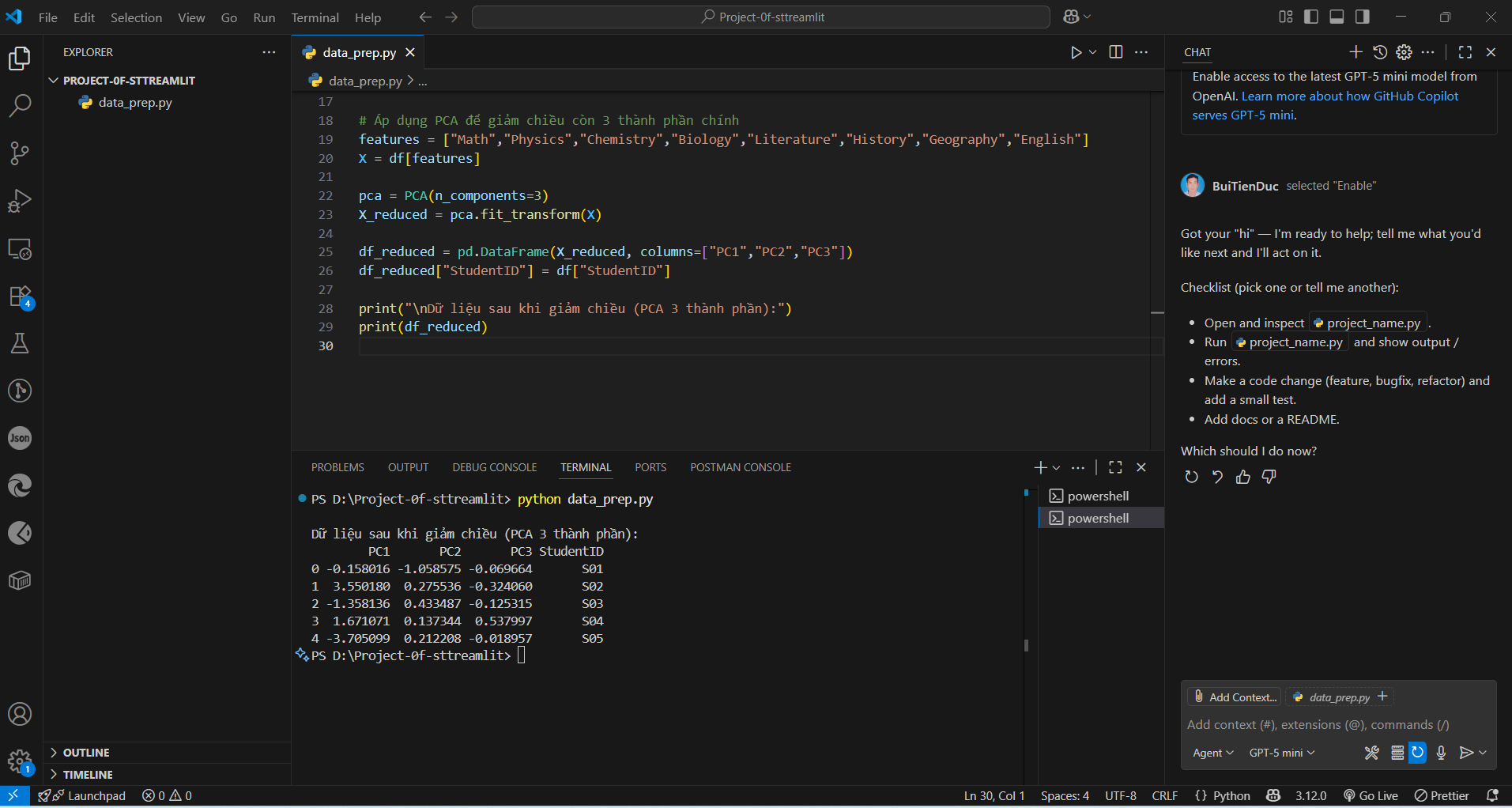
X\_reduced = pca.fit\_transform(X)

df\_reduced = pd.DataFrame(X\_reduced, columns=["PC1","PC2","PC3"])

df\_reduced["StudentID"] = df["StudentID"]

print("\nDữ liệu sau khi giảm chiều (PCA 3 thành phần):")

print(df\_reduced)



1. Thu giảm lượng (numerosity reduction)

Bài tập: Thu giảm dữ liệu – Thu giảm lượng (Numerosity Reduction)

a. Dữ liệu mẫu (dạng bảng)

Ngày Doanh số (triệu VND)

01/01 52

02/01 55

03/01 58

04/01 90

05/01 54

06/01 53

07/01 52

08/01 91

09/01 59

10/01 58

b. Định nghĩa vấn đề trong dữ liệu mẫu

Dữ liệu có nhiều bản ghi chi tiết, gây tốn bộ nhớ. Ta cần thu giảm lượng (numerosity reduction) bằng cách biểu diễn dữ liệu bằng mô hình hoặc xấp xỉ. Một cách phổ biến: dùng hồi quy tuyến tính (linear regression) để biểu diễn xu hướng thay vì lưu toàn bộ dữ liệu.

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

Giả sử mô hình hồi quy tuyến tính: Doanh\_so = a + b \* Ngày.

Quan sát dữ liệu cho thấy xu hướng tăng theo thời gian, dù có nhiễu (outlier: ngày 4 và 8).

Ước lượng sơ bộ: b ≈ 1.0, a ≈ 52 → Mô hình: Doanh\_so ≈ 52 + 1.0 \* Ngày.

d. Giá trị sau xử lý

Thay vì lưu 10 bản ghi, ta chỉ lưu mô hình: Doanh\_so = 52 + 1.0 \* Ngày.

Ví dụ dự đoán:

- Ngày 1 → 53 (so với 52 thực tế)

- Ngày 10 → 62 (so với 58 thực tế)

Mặc dù mất chi tiết, mô hình vẫn phản ánh được xu hướng chung.

e. Code Python để hiện thực xử lý

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Dữ liệu mẫu

data = {

"Day": np.arange(1,11),

"Sales": [52,55,58,90,54,53,52,91,59,58]

}

df = pd.DataFrame(data)

# Hồi quy tuyến tính

X = df[["Day"]]

y = df["Sales"]

model = LinearRegression()

model.fit(X, y)

a = model.intercept\_

b = model.coef\_[0]

print(f"Mô hình hồi quy: Sales = {a:.2f} + {b:.2f} \* Day")

# Dự đoán

df["Predicted"] = model.predict(X)

print("\nDữ liệu sau khi thu giảm (biểu diễn bằng mô hình hồi quy):")

print(df)

# Vẽ trực quan

plt.scatter(df["Day"], df["Sales"], color="blue", label="Thực tế")

plt.plot(df["Day"], df["Predicted"], color="red", label="Hồi quy")

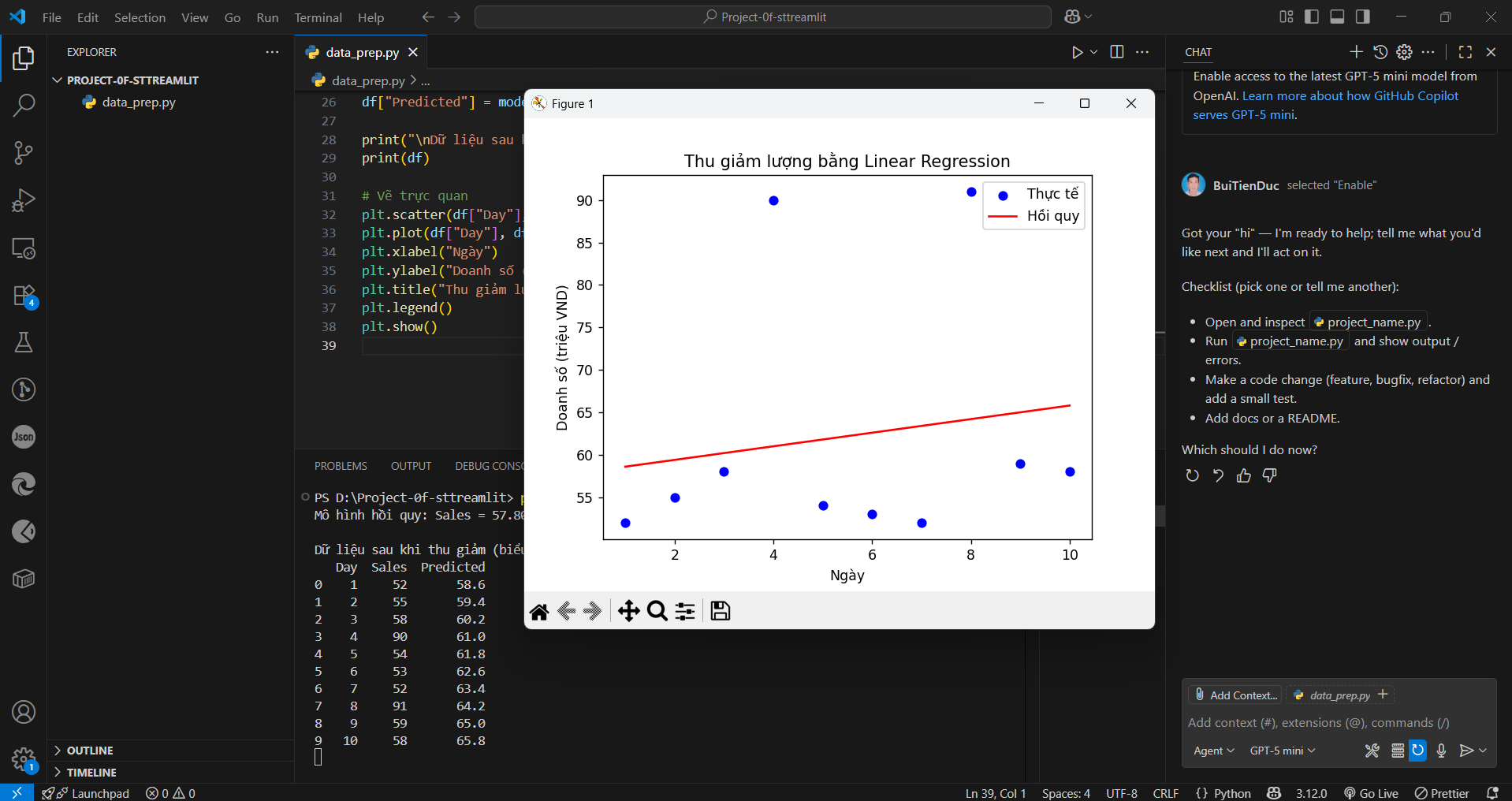
plt.xlabel("Ngày")

plt.ylabel("Doanh số (triệu VND)")

plt.title("Thu giảm lượng bằng Linear Regression")

plt.legend()

plt.show()



1. Tạo phân cấp ý niệm (concept hierarchy generation)

Bài tập: Phân cấp ý niệm (Concept Hierarchy Generation)

a. Dữ liệu mẫu dạng bảng

ID Vùng Tỉnh Thành phố Số tiền (triệu VND)

1 Miền Bắc Hà Nội Hoàn Kiếm 5

2 Miền Bắc Hà Nội Cầu Giấy 15

3 Miền Bắc Hải Phòng Lê Chân 7

4 Miền Trung Đà Nẵng Hải Châu 20

5 Miền Nam TP.HCM Quận 1 12

6 Miền Nam TP.HCM Quận 3 3

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Dữ liệu ban đầu có nhiều mức chi tiết: Thành phố, Tỉnh, Vùng.

- Cột Số tiền có giá trị liên tục, cần phân cấp thành các nhóm khái niệm.

Phân cấp ý niệm:

1. Địa lý: Thành phố → Tỉnh → Vùng

2. Số tiền:

- Nhỏ (0–5 triệu)

- Trung bình (6–15 triệu)

- Lớn (>15 triệu)

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

ID Vùng Tỉnh Thành phố Số tiền (triệu VND) Nhóm tiền

1 Miền Bắc Hà Nội Hoàn Kiếm 5 Nhỏ

2 Miền Bắc Hà Nội Cầu Giấy 15 Trung bình

3 Miền Bắc Hải Phòng Lê Chân 7 Trung bình

4 Miền Trung Đà Nẵng Hải Châu 20 Lớn

5 Miền Nam TP.HCM Quận 1 12 Trung bình

6 Miền Nam TP.HCM Quận 3 3 Nhỏ

d. Giá trị sau xử lý

Tổng số tiền theo Nhóm tiền:

Nhóm tiền Tổng số tiền (triệu VND)

Nhỏ 8

Trung bình 34

Lớn 20

Tổng số tiền theo Vùng:

Vùng Tổng số tiền (triệu VND)

Miền Bắc 27

Miền Trung 20

Miền Nam 15

e. Code Python để hiện thực xử lý + vẽ hình

import pandas as pd

import matplotlib.pyplot as plt

# Dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5, 6],

"Vùng": ["Miền Bắc", "Miền Bắc", "Miền Bắc", "Miền Trung", "Miền Nam", "Miền Nam"],

"Tỉnh": ["Hà Nội", "Hà Nội", "Hải Phòng", "Đà Nẵng", "TP.HCM", "TP.HCM"],

"Thành phố": ["Hoàn Kiếm", "Cầu Giấy", "Lê Chân", "Hải Châu", "Quận 1", "Quận 3"],

"Số tiền": [5, 15, 7, 20, 12, 3]

}

df = pd.DataFrame(data)

# Phân cấp ý niệm cho số tiền

def nhom\_tien(x):

if x <= 5:

return "Nhỏ"

elif x <= 15:

return "Trung bình"

else:

return "Lớn"

df["Nhóm tiền"] = df["Số tiền"].apply(nhom\_tien)

print("=== Dữ liệu sau phân cấp ===")

print(df)

# Tổng hợp theo Nhóm tiền

tong\_tien\_theo\_nhom = df.groupby("Nhóm tiền")["Số tiền"].sum()

print("\n=== Tổng số tiền theo Nhóm tiền ===")

print(tong\_tien\_theo\_nhom)

# Tổng hợp theo Vùng

tong\_tien\_theo\_vung = df.groupby("Vùng")["Số tiền"].sum()

print("\n=== Tổng số tiền theo Vùng ===")

print(tong\_tien\_theo\_vung)

# Vẽ histogram Nhóm tiền

tong\_tien\_theo\_nhom.plot(kind="bar", color="skyblue", edgecolor="black")

plt.title("Tổng số tiền theo Nhóm tiền")

plt.xlabel("Nhóm tiền")

plt.ylabel("Tổng số tiền (triệu VND)")

plt.show()

# Vẽ histogram Vùng

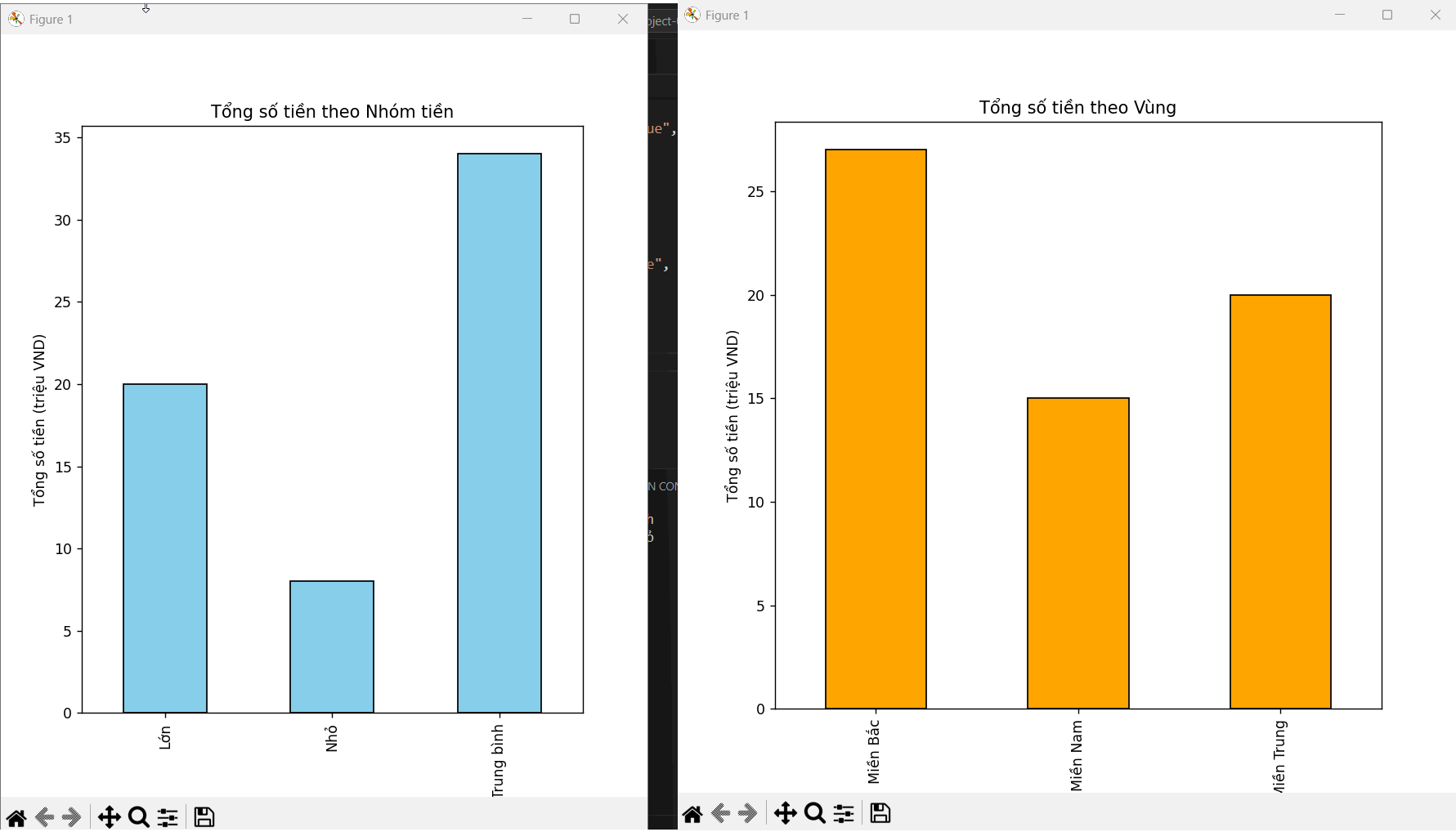
tong\_tien\_theo\_vung.plot(kind="bar", color="orange", edgecolor="black")

plt.title("Tổng số tiền theo Vùng")

plt.xlabel("Vùng")

plt.ylabel("Tổng số tiền (triệu VND)")

plt.show()



1. Rời rạc hóa dữ liệu (discretization)

Bài tập: Rời rạc hóa dữ liệu (Discretization)

a. Dữ liệu mẫu dạng bảng

ID Tên KH Tuổi Thu nhập (triệu VND)

1 A 23 5

2 B 35 12

3 C 42 20

4 D 51 30

5 E 62 50

6 F 28 8

b. Định nghĩa vấn đề trong dữ liệu mẫu

- Các thuộc tính Tuổi và Thu nhập ở dạng liên tục.

- Cần rời rạc hóa thành các khoảng để thuận tiện cho phân tích và thuật toán.

Rời rạc hóa đề xuất:

1. Tuổi:

- Trẻ (≤30)

- Trung niên (31–50)

- Cao tuổi (>50)

2. Thu nhập:

- Thấp (≤10)

- Trung bình (11–30)

- Cao (>30)

c. Xử lý bằng tay tất cả vấn đề trong dữ liệu mẫu

ID Tên KH Tuổi Nhóm tuổi Thu nhập (triệu VND) Nhóm thu nhập

1 A 23 Trẻ 5 Thấp

2 B 35 Trung niên 12 Trung bình

3 C 42 Trung niên 20 Trung bình

4 D 51 Cao tuổi 30 Trung bình

5 E 62 Cao tuổi 50 Cao

6 F 28 Trẻ 8 Thấp

d. Giá trị sau xử lý

Phân bố Nhóm tuổi:

Nhóm tuổi Số KH

Trẻ 2

Trung niên 2

Cao tuổi 2

Phân bố Nhóm thu nhập:

Nhóm thu nhập Số KH

Thấp 2

Trung bình 3

Cao 1

e. Code Python để hiện thực xử lý + vẽ hình histogram

import pandas as pd

import matplotlib.pyplot as plt

# Dữ liệu mẫu

data = {

"ID": [1, 2, 3, 4, 5, 6],

"Tên KH": ["A", "B", "C", "D", "E", "F"],

"Tuổi": [23, 35, 42, 51, 62, 28],

"Thu nhập": [5, 12, 20, 30, 50, 8]

}

df = pd.DataFrame(data)

# Hàm rời rạc hóa

def nhom\_tuoi(x):

if x <= 30:

return "Trẻ"

elif x <= 50:

return "Trung niên"

else:

return "Cao tuổi"

def nhom\_thu\_nhap(x):

if x <= 10:

return "Thấp"

elif x <= 30:

return "Trung bình"

else:

return "Cao"

df["Nhóm tuổi"] = df["Tuổi"].apply(nhom\_tuoi)

df["Nhóm thu nhập"] = df["Thu nhập"].apply(nhom\_thu\_nhap)

print("=== Dữ liệu sau rời rạc hóa ===")

print(df)

# Thống kê

print("\n=== Phân bố Nhóm tuổi ===")

print(df["Nhóm tuổi"].value\_counts())

print("\n=== Phân bố Nhóm thu nhập ===")

print(df["Nhóm thu nhập"].value\_counts())

# Vẽ histogram Tuổi

df["Nhóm tuổi"].value\_counts().plot(kind="bar", color="skyblue", edgecolor="black")

plt.title("Phân bố theo Nhóm tuổi")

plt.xlabel("Nhóm tuổi")

plt.ylabel("Số khách hàng")

plt.show()

# Vẽ histogram Thu nhập

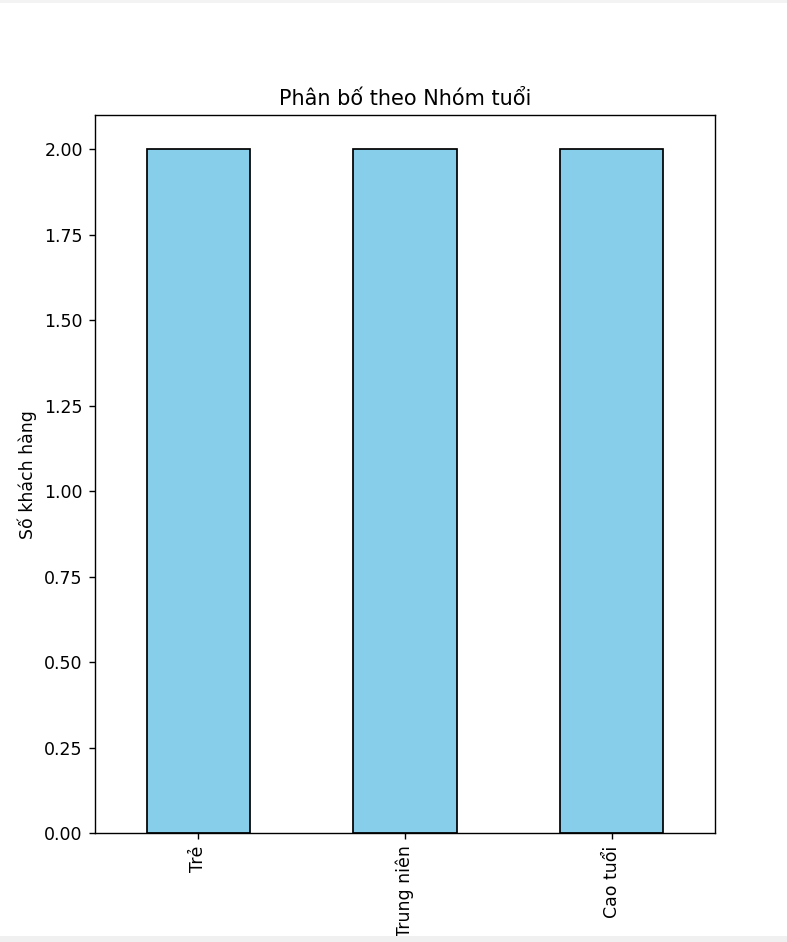
df["Nhóm thu nhập"].value\_counts().plot(kind="bar", color="orange", edgecolor="black")

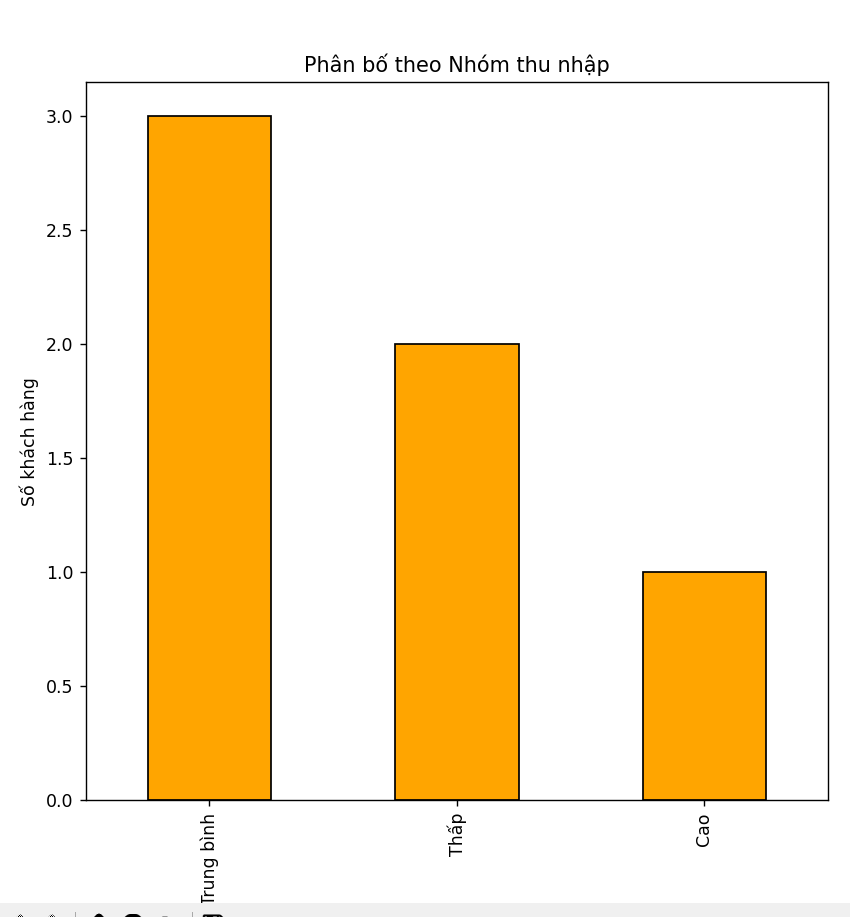
plt.title("Phân bố theo Nhóm thu nhập")

plt.xlabel("Nhóm thu nhập")

plt.ylabel("Số khách hàng")

plt.show()





Rời rạc hóa dữ liệu xuất hiện ở đâu?

Ban đầu cột Tuổi trong dữ liệu là giá trị liên tục (23, 28, 35, 42, 51, 62).

Sau khi rời rạc hóa, ta chia thành các khoảng rời rạc (intervals):

Trẻ (≤30)

Trung niên (31–50)

Cao tuổi (>50)

Biểu đồ cho thấy chính là kết quả sau khi rời rạc hóa: thay vì hiển thị các con số tuổi chi tiết, dữ liệu đã được gom nhóm thành ba nhãn rời rạc.

Rời rạc hóa nằm ở bước chuyển từ “Tuổi = 23, 28, 35, …” sang “Trẻ, Trung niên, Cao tuổi”.

Biểu đồ này không còn biểu diễn dữ liệu gốc (tuổi số), mà là dữ liệu sau rời rạc hóa.

**Xin cảm ơn!**

**Bùi Tiến Đức-010239**