# Họ và tên: Mai Hoàng Tùng

# Lớp: 64TTNT1

# MSV: 2251262656

# Đề: 2251262656%15+1=2

**Thực nghiệm phân tích chuỗi thời gian với mô hình Gaussian kết hợp biến thể Transformer**

# Tìm hiểu và Chuẩn bị Dữ liệu

1. **Truy cập dữ liệu:** [https://www.kaggle.com/datasets/kukuroo3/room-occupancy-detection-data-iot-sensor](Time-series)

# Khám phá dữ liệu (Dự kiến):

- *date:* Ngày ghi nhận dữ liệu.

- t*enperature:* Nhiệt độ

- h*umidity:* Độ ẩm

- l*ight:* Ánh sáng

- *CO2, Occupancy*

# **Xử lý:**

- Đọc và tiền xử lý dữ liệu từ file Occupancy.csv

- Chuyển đổi cột thời gian thành timestamp (nếu có)

- Tách đặc trưng (feature) và nhãn (label)

- Chuẩn hóa đặc trưng bằng StandardScaler

- Áp dụng GMM để phân cụm không giám sát

- Huấn luyện mô hình Transformer Encoder cho bài toán phân loại

- Đánh giá kết quả bằng classification report và confusion matrix

- Trực quan hóa kết quả bằng biểu đồ

# Triển khai **mô hình Gaussian kết hợp biến thể Transformer**

### ****1. Mô hình Gaussian Mixture Model (GMM)****

#### ****Ý tưởng:****

GMM là mô hình học **không giám sát**, giả định rằng dữ liệu được sinh ra từ một tổ hợp của nhiều phân phối chuẩn (Gaussian). Mỗi điểm dữ liệu thuộc về một trong các cụm (component) ẩn, và GMM dùng thuật toán EM để tìm tham số của mô hình.

#### ****Công thức tổng quát:****

Xác suất của một điểm dữ liệu **x** được tính như sau:

p(x) = π₁·N(x | μ₁, Σ₁) + π₂·N(x | μ₂, Σ₂) + ... + πₖ·N(x | μₖ, Σₖ)

**Trong đó:**

πₖ: hệ số hỗn hợp (tổng πₖ = 1)

N(x | μₖ, Σₖ): mật độ xác suất Gaussian với trung bình μₖ và hiệp phương sai Σₖ

#### ****Các bước của thuật toán EM:****

**Bước 1 (Khởi tạo)**:**** Chọn ngẫu nhiên giá trị ban đầu cho các tham số μₖ, Σₖ và πₖ

**Bước 2 (Expectation (E-step))**:**** Tính xác suất một điểm xᵢ thuộc về cụm k:

γᵢₖ = (πₖ·N(xᵢ | μₖ, Σₖ)) / ∑ⱼ πⱼ·N(xᵢ | μⱼ, Σⱼ)

**Bước 3 (Maximization (M-step)):** Cập nhật lại tham số:

Nₖ = ∑ᵢ γᵢₖ  
μₖ = (1/Nₖ)·∑ᵢ γᵢₖ·xᵢ  
Σₖ = (1/Nₖ)·∑ᵢ γᵢₖ·(xᵢ - μₖ)(xᵢ - μₖ)ᵀ  
πₖ = Nₖ / N

**Bước 4 (Phân loại):** Gán mỗi điểm xᵢ vào cụm có γᵢₖ lớn nhất:

yᵢ = argmaxₖ γᵢₖ

## ****2. Mô hình Transformer Encoder (Biến thể)****

### ****Ý tưởng tổng quan****

Transformer Encoder là mô hình sử dụng **cơ chế attention** để tìm ra các mối quan hệ giữa các đặc trưng đầu vào. Trong biến thể đơn giản của bạn, mỗi chuỗi dữ liệu được xem là một chuỗi ngắn với độ dài 1 (seq\_len = 1), nhưng vẫn áp dụng cơ chế attention và xử lý theo từng batch.

### ****Cấu trúc mô hình****

Gồm 2 phần chính:

**- Transformer Encoder Layer**

**- Lớp phân loại tuyến tính (Linear Layer)**

### ****Công thức mô hình****

#### ****a) Cơ chế Attention (Scaled Dot-Product Attention)****

Đầu tiên, các vector đặc trưng được chiếu vào ba không gian khác nhau:

Q = X × W\_Q

K = X × W\_K

V = X × W\_V

Sau đó, tính attention như sau:

Attention(Q, K, V) = softmax(Q × Kᵗ / √dₖ) × V

**Trong đó:**

Q, K, V: ma trận truy vấn, khóa và giá trị

dₖ: số chiều của vector khóa

W\_Q, W\_K, W\_V: ma trận trọng số học được

#### ****b) Multi-Head Attention****

Nếu có h heads, mỗi head thực hiện attention như trên và kết quả các head được nối lại:

MultiHead(Q, K, V) = Concat(head₁, ..., headₕ) × W\_O

**Trong đó:**

headᵢ = Attention(Qᵢ, Kᵢ, Vᵢ)

W\_O: ma trận học được sau khi nối

#### ****c) Position-wise Feedforward Network****

Sau attention là mạng fully-connected áp dụng cho từng vị trí (position):

FFN(x) = max(0, x × W₁ + b₁) × W₂ + b₂

#### ****d) Pooling và Phân loại****

Vì chuỗi có độ dài 1, ta lấy trung bình theo chiều sequence (thường là mean(dim=1)):

x\_pooled = mean(x, dim=1)

Sau đó đưa vào một lớp tuyến tính:

ŷ = softmax(x\_pooled × W\_c + b\_c)

**Trong đó:**

W\_c, b\_c: trọng số và bias của lớp phân loại

ŷ: xác suất dự đoán thuộc các lớp (Occupied hoặc Not Occupied)

# 3. Chạy Code và Public Lên GitHub

**a) Môi trường:** Thiết lập môi trường Python với các thư viện cần thiết: pandas, numpy, matplotlib, scikit-learn ......

### ****1. Tải và Tiền Xử Lý Dữ Liệu****

- Đọc tệp Occupancy.csv và kiểm tra định dạng.

- Chuyển đổi các cột thời gian về dạng timestamp (giây).

- Loại bỏ cột gốc thời gian sau khi chuyển đổi.

- Tách đặc trưng đầu vào gồm: Temperature, Humidity, Light, CO2, HumidityRatio và các timestamp (nếu có).

- Tách nhãn mục tiêu là Occupancy (0: Không người, 1: Có người).

### ****2. Phân Tích Dữ Liệu Ban Đầu****

- Chuẩn hóa dữ liệu đầu vào bằng StandardScaler.

- Trực quan hóa phân phối dữ liệu bằng biểu đồ PCA 2 chiều để quan sát phân tách cụm ban đầu (trước khi huấn luyện).

### ****3. Áp Dụng Gaussian Mixture Model (GMM)****

- Khởi tạo mô hình GMM với n\_components=2.

- Huấn luyện mô hình GMM trên toàn bộ dữ liệu đặc trưng (không dùng nhãn).

- Dự đoán nhãn GMM cho từng điểm dữ liệu.

- Dùng **majority voting (mode)** để ánh xạ nhãn GMM về nhãn thật (0 hoặc 1).

- Đánh giá kết quả phân loại bằng GMM bằng **Classification report và Confusion matrix**

### ****4. Chuẩn Bị Dữ Liệu Cho Transformer****

- Chuẩn hóa đặc trưng với StandardScaler.

- Chia tập dữ liệu thành **80% huấn luyện** và **20% kiểm tra**.

- Chuyển dữ liệu sang Tensor và thêm chiều sequence (seq\_len = 1).

- Đóng gói dữ liệu vào các DataLoader cho huấn luyện và đánh giá.

### ****5. Xây Dựng Mô Hình Transformer Encoder (Biến thể)****

- Xây dựng lớp TransformerClassifier gồm:

- Một TransformerEncoderLayer với attention đơn giản (nhead=1)

- Một lớp tuyến tính (Linear) để phân loại.

- Dùng cơ chế **mean pooling** để lấy đầu ra từ sequence.

- Khởi tạo mô hình, hàm mất mát CrossEntropyLoss, và tối ưu hóa Adam.

### ****6. Huấn Luyện và Dự Đoán Với Transformer:**** Huấn luyện mô hình trong 20 epoch bằng cách lặp qua train\_dataloader.

### ****7. Đánh Giá Mô Hình Transformer****

- Tính các chỉ số phân loại:

- Accuracy, Precision, Recall, F1-score

- In classification\_report cho cả tập huấn luyện và kiểm tra.

- Vẽ **ma trận nhầm lẫn (Confusion Matrix)** để trực quan hóa kết quả.

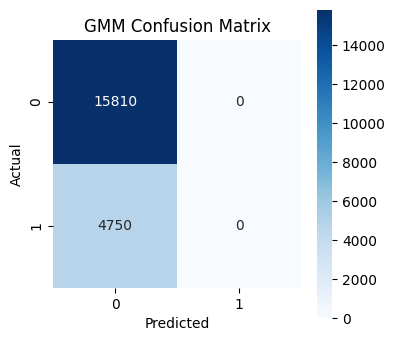
### ****8. Trực Quan Hóa Dữ Liệu****

- Giảm chiều dữ liệu đặc trưng bằng **PCA 2 chiều**.

- Vẽ biểu đồ phân tán thể hiện các cụm phân loại GMM sau giảm chiều.

- So sánh phân bố nhãn dự đoán của GMM và Transformer.

### ****9. So Sánh và Phân Tích Kết Quả****

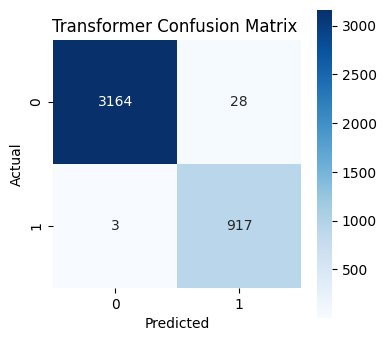


**Nhận xét:**

**- Toàn bộ mẫu đều bị dự đoán là lớp 0**, GMM **không phân biệt được giữa hai lớp**.

- Dù GMM là mô hình không giám sát (unsupervised), điều này cho thấy **khả năng phân cụm kém**, hoặc dữ liệu không thể phân tách tốt bằng mô hình Gaussian đơn giản.

- Có thể do đặc trưng chưa được chuẩn hóa tốt hoặc phân phối không phù hợp với giả định Gaussian.



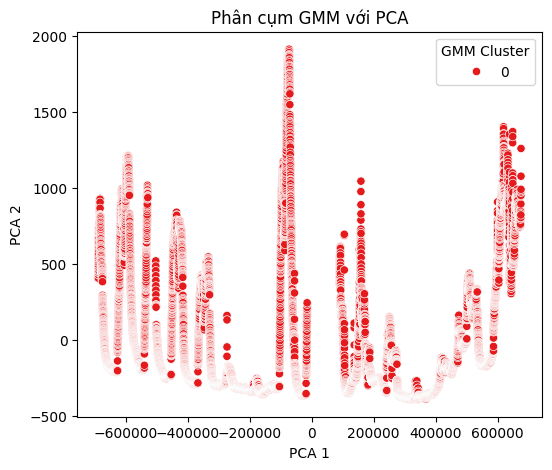
****Nhận xét:****

- Mô hình **Transformer hoạt động rất tốt**.

- Chỉ có **3 mẫu âm bị dự đoán sai** (false negative) và **28 mẫu dương bị sai** (false positive).

- Kết quả này chứng tỏ **mô hình có độ chính xác cao**, đặc biệt là trong phân biệt hai lớp (Occupancy: có người và không có người).

- Transformer Encoder tận dụng đặc trưng thời gian tốt hơn và có khả năng học phi tuyến hiệu quả.



**Nhận xét:**

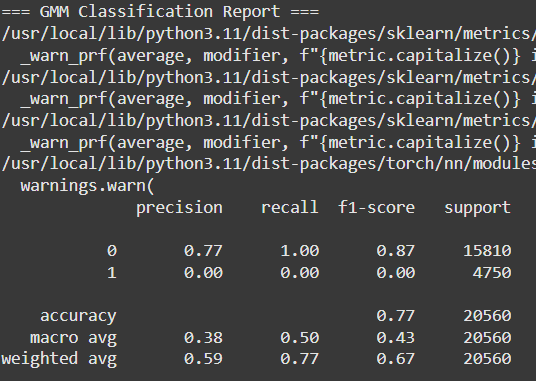
- PCA đã được áp dụng để giảm chiều dữ liệu xuống 2D để trực quan hóa.

- Tuy nhiên, biểu đồ **chỉ thể hiện 1 cụm duy nhất (Cluster 0)** → khớp với ma trận nhầm lẫn GMM cho thấy GMM **không phát hiện được cụm thứ 2**.

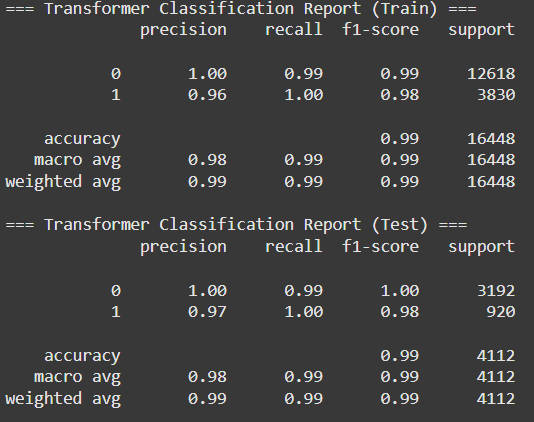
- Các điểm dữ liệu phân bố **rải rác và có vẻ không tuân theo phân phối Gaussian rõ ràng**, điều này khiến GMM khó hoạt động hiệu quả.

**- GMM không tìm được ranh giới phân cụm rõ ràng**, có thể do cấu trúc dữ liệu phức tạp hoặc cần chuẩn hóa/phân tích lại đặc trưng.

·**In Kết Quả:**



**Nhận xét:** GMM hoàn toàn **thiên lệch** về lớp 0. Mô hình không học được đặc trưng phân biệt giữa hai lớp -> mô hình **kém hiệu quả** với bài toán phân loại nhãn Occupancy.



**Nhận xét:** Mô hình Transformer cho kết quả rất **cân bằng**, nhận diện tốt cả hai lớp. Sai số rất nhỏ, thể hiện khả năng **tổng quát hóa** cao.

1. **Tài liệu tham khảo:**

[https://www.kaggle.com/code/dfoly1/gaussian-mixture-model#GMM-should-produce-something-similar](GMM & Transformer-MaiHoangTung-64TTNT1.docx)

[https://medium.com/@RobuRishabh/types-of-transformer-model-1b52381fa719](GMM & Transformer-MaiHoangTung-64TTNT1.docx)

[https://www.kaggle.com/code/hakim11/transformer-encoder-architecture](GMM & Transformer-MaiHoangTung-64TTNT1.docx)

# Phụ lục (Tùy chọn):

Link GitHub: [https://github.com/Tungmh/GMM-Transformer](GMM & Transformer-MaiHoangTung-64TTNT1.docx)