ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

MATH FOR AI

PCA, CLUSTERING

NHÓM 5

Giáo viên	Cấn Trần Thành Trung
	Nguyễn Ngọc Toàn
23122003	Nguyễn Văn Linh
23122022	Trần Hoàng Gia Bảo
23122026	Trần Chấn Hiệp
23122040	Nguyễn Thị Mỹ Kim

TP. HỒ CHÍ MINH, THÁNG 5/2025

Mục lục

1 Giới thiệu

1.1 Dữ liệu Iris

Bộ dữ liệu **Iris** là một trong những tập dữ liệu kinh điển và phổ biến nhất trong lĩnh vực học máy và thống kê. Nó được giới thiệu bởi nhà thống kê học Ronald A. Fisher vào năm 1936 trong bài báo "The use of multiple measurements in taxonomic problems". Bộ dữ liệu này thường được sử dụng để minh họa các thuật toán phân loại, phân tích thành phần chính (PCA) và trực quan hóa.

1.1.1 Mô tả dữ liệu

Bộ dữ liệu Iris chứa thông tin về 150 mẫu hoa thuộc 3 loài khác nhau trong họ *Iris*:

- `Iris-setosa`
- `Iris-versicolor`
- `Iris-virginica`

Mỗi mẫu có 4 đặc trưng định lượng:

- `sepal length (cm)`: chiều dài đài hoa
- `sepal width (cm)`: chiều rộng đài hoa
- `petal length (cm)`: chiều dài cánh hoa
- `petal width (cm)`: chiều rộng cánh hoa

Kích thước dữ liệu: 150 dòng (mẫu) và 5 cột (4 đặc trưng + 1 nhãn lớp)
Nạp và sử dụng dữ liệu iris:
- Dữ liệu được tải từ thư viện `scikit-learn`.

```
from sklearn.datasets import load_iris
iris = load_iris()
X = pd.DataFrame(data=iris.data, columns=iris.feature_names)
y = iris.target
```

1.2 Dữ liệu ABIDE II (updated)

1.2.1 Mô tả dữ liệu

Trong nghiên cứu này, chúng em sử dụng một phiên bản được cập nhật từ bộ dữ liệu **ABIDE II (Autism Brain Imaging Data Exchange II)** – một tập dữ liệu hình ảnh cộng hưởng từ (MRI/fMRI) mở, ban đầu được sử dụng trong các nghiên cứu liên quan đến rối loạn phổ tự kỷ. Bộ dữ liệu này từng được giới thiệu tại *NeuroHackademy 2020* bởi **GS. Tal Yarkoni** nhằm thúc đẩy nghiên cứu trong lĩnh vực khoa học thần kinh tính toán.

Tuy nhiên, phiên bản được sử dụng trong đồ án này đã được **điều chỉnh nhãn (re-labeled)** nhằm phục vụ cho mục tiêu phân tích bệnh lý ung thư. Cụ thể, cột nhãn `group` hiện tại bao gồm hai giá trị:

- `"Cancer"`: 463 bệnh nhân mắc ung thư
- `"Normal"`: 541 bệnh nhân bình thường

Tổng quan bộ dữ liệu:

- Số mẫu: **1004**
- Số đặc trưng: **1444** (dạng số thực)
- Cột nhãn: `group` (kiểu chuỗi, gồm 2 giá trị: `"Cancer"` và `"Normal"`)

Cấu trúc và đặc điểm của các đặc trưng

Mỗi mẫu dữ liệu bao gồm 1444 đặc trưng định lượng, được trích xuất từ ảnh MRI hoặc fMRI sử dụng phần mềm **FreeSurfer** – một công cụ phổ biến trong phân tích ảnh thần kinh học. Tên các đặc trưng tuân theo một quy tắc định danh nhất quán, ví dụ như:

`fsArea_L_V1_ROI`

Trong đó:

- `fsArea`, `fsLgi`, `fsCt`, `fsVol`: là loại chỉ số đo lường, tương ứng với:
  - `Area`: diện tích bề mặt vỏ não
  - `Lgi`: chỉ số độ nếp gấp vỏ não (Local Gyrification Index)
  - `Ct`: độ dày vỏ não (Cortical Thickness)
  - `Vol`: thể tích vùng não
- `L` hoặc `R`: biểu thị bán cầu não **trái (Left)** hoặc **phải (Right)**
- `V1` (và các mã khác): mã vùng não cụ thể (Region of Interest). Ví dụ: `V1` là vùng thị giác nguyên phát; ngoài ra còn rất nhiều mã khác tương ứng với các vùng khác nhau theo các atlas thần kinh học.
- `ROI`: viết tắt của "Region of Interest" – chỉ định đặc trưng này liên quan đến một vùng cụ thể trong não.

Hiện tại, do số lượng vùng ROI và các chỉ số giải phẫu là rất lớn và đa dạng, việc liệt kê toàn bộ là không khả thi trong phạm vi báo cáo. Tuy nhiên, các đặc trưng này đều mang ý nghĩa sinh học và giải phẫu có thể quan trọng trong việc phân biệt giữa hai nhóm bệnh nhân.

2 Phương pháp và chứng minh toán học

2.1 Pricipal Component Analysis

2.1.1 Giới thiệu

Principal Component Analysis(PCA) là thuật toán giúp giảm số chiều dữ liệu khi dữ liệu có quá nhiều thuộc tính bằng cách tạo ra cách thuộc tính ảo dữa trên dữ liệu gốc nhưng vẫn giữ được phần lớn thông tin quan trọng, kĩ thuật này được sử dụng phổ biến trong các mô hình học máy.

2.1.2 Thuật toán

Cho dữ liệu $\theta = \{\vec{x_1}, ..., \vec{x_n}\}, \vec{x_i} \in \mathbb{R}^d$

- Chuẩn hóa dữ liệu:
  Tính $\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n}\vec{x_i}$. Sau đó, đặt $\hat{X} = \begin{bmatrix} \vec{x_1}^T \\ \vdots \\ \vec{x_n}^T \end{bmatrix} \in M_{n\times d}(\mathbb{R})$ với $\vec{\hat{x_i}} = \vec{x_i} - \vec{\mu}$ hoặc $\vec{\hat{x_i}} = \frac{\vec{x_i}-\vec{\mu}}{\sigma}$
  (đồ án này sử dụng)
- Lập ma trận hiệp phương sai:
  Tính ma trận hiệp phương sai $A = \frac{1}{n}\hat{X}^T\hat{X}$
- Tìm các trị riêng và vector riêng của A và sắp xếp chúng theo thứ tự giảm dần là $\{\lambda_1, ..., \lambda_d\}$ và bộ cơ sở trực chuẩn của $\mathbb{R}^d$ tương ứng với từng trị riêng $\{\vec{v_1}, ..., \vec{v_n}\}$.
- Chọn ra k vector riêng từ bộ ứng với k trị riêng lớn nhất vừa tìm được tạo thành $W_{d\times k} = [\vec{v_1} \quad ... \quad \vec{v_k}]$
- Tính $Z_{n\times k} = \hat{X}W$ chính là dữ liệu với từ dữ liệu sau khi ta đã áp dụng kĩ thuật PCA.

2.1.3 Giải thích

Cho $\theta = \{\vec{x}_1, \ldots, \vec{x}_n\} \subset \mathbb{R}^d$ và $\vec{b}_1 \in \mathbb{R}^d$ là dữ liệu sau khi chuẩn hóa, độ khuếch tán theo phương $\vec{b}_1$ của θ là:

$$\mathrm{Var}_{\vec{b}_1}(\theta) = \frac{1}{n}(\|\operatorname{proj}_{\vec{b}_1}\vec{x}_1\|^2 + \cdots + \operatorname{proj}_{\vec{b}_1}\vec{x}_n\|^2)$$

Mục tiêu của ta là tìm $\vec{b_1}$ sao cho $Var_{\vec{b_1}}(\theta)$ đạt giá trị lớn nhất, từ việc tìm được 1 vector $\vec{b_1}$ ta sẽ mở rộng tìm bộ trực chuẩn k vector $\{\vec{b_1}, ..., \vec{b_k}\}$, đây chính là mục đích cốt lõi của thuật toán PCA.
Ta có định lý: Cho A là một ma trận đối xứng thực cấp n thì tồn tại một cơ sở trực chuẩn gồm toàn vector riêng của A.
Áp dụng cho ma trận hiệp phương sai có được từ bộ dữ liệu θ như đã trình bày ở trên ta được: Gọi $\{\vec{w}_1, \ldots, \vec{w}_d\}$ là cơ sở trực chuẩn của $\mathbb{R}^d$ gồm toàn vector riêng của ma trận hiệp phương sai A ứng với trị riêng $\{\lambda_1, ..., \lambda_d\}$ và $\lambda_1 \geq ... \geq \lambda_d$ thì:

$$\vec{b_1} = proj_{\mathbb{R}^d}\vec{b_1} = \sum_{i=1}^{d} proj_{\vec{w_i}}\vec{b_1} = \alpha_1\vec{w_1} + ... + \alpha_d\vec{w_d}$$

Phương $\vec{b}_1$ được gọi là thành phần chính thứ nhất của θ nếu $\underline{\mathrm{Var}_{\vec{b}_1}(\theta) \text{ đạt giá trị lớn nhất}}$
Giả sử θ là các dòng của $\hat{X}$ thì ma trận hiệp sai A là:

$$A = \frac{1}{n}\hat{X}^T\hat{X}$$

Ta có: $||proj_{\vec{b_1}}\vec{x_1}||^2 = proj_{\vec{b_1}}\vec{x_1}.proj_{\vec{b_1}}\vec{x_1} = (\frac{\vec{b_1}.\vec{x_1}}{\vec{b_1}.\vec{b_1}})\vec{b_1}.(\frac{\vec{b_1}.\vec{x_1}}{\vec{b_1}.\vec{b_1}})\vec{b_1} = (\vec{b_1}\vec{x_1})^2 = \vec{b_1}^T\vec{x_1}\vec{x_1}^T\vec{b_1}$ do ta chỉ quan tâm đến hướng của $\vec{b_1}$ nên ta chuẩn hóa $||\vec{b_1}|| = 1$.
Khi đó: $Var_{\vec{b_1}}(\theta) = \frac{1}{n}\vec{b_1}^T(\sum_{i=1}^{n}\vec{x_i}\vec{x_i}^T)\vec{b_1} = \vec{b_1}^T A\vec{b_1}$. Tới đây ta quy về tìm $\vec{b_1}$ sao cho $\vec{b_1}^T A\vec{b_1}$ đạt giá trị lớn nhất.

Mà ta có $\vec{b_1}^T A\vec{b_1} = \langle \vec{b_1}, A\vec{b_1} \rangle = \langle \vec{b_1}, A(\alpha_1\vec{w_1} + ... + \alpha_d\vec{w_d}) \rangle = \langle \vec{b_1}, (\alpha_1\lambda_1\vec{w_1} + ... + \alpha_d\lambda_d\vec{w_d}) \rangle = \langle (\alpha_1\vec{w_1} + ... + \alpha_d\vec{w_d}), (\alpha_1\lambda_1\vec{w_1} + ... + \alpha_d\lambda_d\vec{w_d}) = \lambda_1\alpha_1^2 + ... + \lambda_d\alpha_d^2 \leq \lambda_1(\alpha_1^2 + ... + \alpha_n^2) = \lambda_1$ với $\{\vec{w_1}, ..., \vec{w_d}\}$ là bộ cơ sở trực chuẩn gồm toàn vector riêng của A đã đề cập ở trên.
Ta thấy ngay nếu $\vec{b_1}$ là vector đơn vị riêng ứng với trị riêng lớn nhất λ_1 của A thì $\vec{b_1}^T A\vec{b_1} = \lambda_1$ và cũng chính là giá trị lớn nhất.
Các thành phần chính $\vec{w_k}$ kế tiếp phải thỏa thêm tính chất vừa để $\vec{w_k}^T A\vec{w_k}$ đạt giá trị lớn nhất và $\vec{w_k} \in span(\{\vec{w_1}, ..., \vec{w_{k-1}}\})^\perp$
Tới đây ta rút ra nhận xét là các thành phần chính sẽ là các vector riêng của A. Ta sẽ chỉ ra nhận xét sau nhờ vào quy nạp toán học.
Với mỗi n là số tự nhiên thì, và ma trận hiệp phương sai $A \in M_{n\times n}(\mathbb{R})$ có bộ cơ sở trực chuẩn $\{\vec{w_1}, ..., \vec{w_n}\}$ ứng với $\lambda_1 \geq ... \geq \lambda_n$
Với $k = 1$ thì ta thấy thành phần chính thứ nhất chính là $\vec{w_1}$ như đã trình bày ở trên.
Giả sử đúng tới $k = t - 1$ hay $\{\vec{w_1}, ..., \vec{w_{t-1}}\}$ là bộ các thành phần chính thứ nhất đến thứ t-1.
Ta cần chỉ ra vector thành phần chính thứ t là $\vec{w_t}$.
Ta xét: $\vec{w} = \alpha_1\vec{w_1} + ... + \alpha_n\vec{w_n}$. Để $\vec{w} \in span(\{\vec{w_1}, ..., \vec{w_{t-1}}\})^\perp$ thì $\alpha_1 = ... = \alpha_{t-1} = 0$ vì $\vec{w}.\vec{w_i} = \alpha_i, \quad i = \overline{1, t-1}$. Suy ra: $\vec{w}^T A\vec{w} = \langle \alpha_t\vec{w_t} + ... + \alpha_n\vec{w_n}, \alpha_t\lambda_t\vec{w_t} + ... + \alpha_n\lambda_n\vec{w_n} \rangle = \lambda_t\alpha_t^2 + ... + \lambda_n\alpha_n^2 \leq \lambda_t(\alpha_t^2 + ... + \alpha_n^2) = \lambda_t$
Tới đây ta thấy nếu $\vec{w} = \vec{w_t}$ thì ta có ngay $\vec{w}^T A\vec{w} = \lambda_t$ là giá trị lớn nhất và hơn nữa thì $\vec{w}$ hiển nhiên thuộc $span(\{\vec{w_1}, ..., \vec{w_{t-1}}\})^\perp$ nên theo nguyên lí quy nạp toán học thì ta có nhận xét đúng.

Đây chính là phần giải thích cho các bước của thuật toán PCA trong việc tìm ra k thành phần chính để giảm số chiều dữ liệu.

2.2 KMeans Clustering

2.2.1 Giới thiệu

Thuật toán KMeans là một trong các mô hình học không giám sát cho bài toán clustering khi ta đã biết trước số cụm.
Mô tả: Ta có bộ dữ liệu $\theta = \{\vec{x_1}, ..., \vec{x_n}\}, \vec{x_i} \in \mathbb{R}^d$ và ta muốn phân bộ dữ liệu trên thành k cụm thì KMeans sẽ giúp ta tìm ra một bộ $C = \{\vec{c_1}, ..., \vec{c_k}\}, \vec{c_i} \in \mathbb{R}^d$ được gọi là "centroid", bộ này gồm các vector/điểm được coi là tâm của các cụm và khi đó mỗi $\vec{x_i} \in \theta$ sẽ thuộc về cụm t sao cho khoảng cách Euclid $||\vec{x_i} - \vec{c_t}||$ là nhỏ nhất có thể trong các vector/điểm trong centroid.

2.2.2 Thuật toán

Thuật toán KMeans chính là tìm bộ centroid $\{\vec{c_1}, ..., \vec{c_k}\}$ làm nghiệm để hàm

$$inertia = \sum_{j=1}^{k} \sum_{\vec{x} \in S_j} ||\vec{x} - \vec{\mu_j}||^2$$

đạt giá trị nhỏ nhất, trong đó: S_j là cụm thứ j ứng với tâm $\vec{u_j}$.
Trước tiên ta sẽ tìm hiểu về thuật toán KMeans để tìm bộ centroid. Thuật toán có các bước cơ sở như sau.

- Thuật toán sẽ thực hiện n lần, với n cố định cho trước.
- Ở mỗi lần, gọi C_t là centroid thứ t, thuật toán sẽ bắt đầu từ một centroid $C_0 = \{\vec{\mu_1}, ..., \vec{\mu_k}\}$ được chọn ngẫu nhiên, thường là lấy k điểm ngẫu nhiên từ θ.

- Từng điểm dữ liệu $\vec{x_i}$ thuộc θ sẽ thuộc về cụm S_j sao cho khoảng cách Euclid từ điểm từ $\vec{x_i}$ đến tâm $\vec{\mu_j}$ là nhỏ nhất, hay:

$$x_i \in S_j \text{ với } j = \arg\min_z ||\vec{x_i} - \vec{\mu_z}||$$

- Tại bước thứ t, từ centroid ta tìm được từ bước $t-1$, ta sẽ tìm trọng tâm của từng cụm S_i, hay ta tìm:

$$\vec{\mu_i^*} = \frac{1}{|S_i|} \sum_{\vec{x} \in S_i} \vec{x}, \quad i = \overline{1,k}$$

- Và cuối cùng ta sẽ thu được centroid mới $C_t = \{\vec{\mu_1^*}, ..., \vec{\mu_k^*}\}$. Và ta sẽ kiểm tra nếu $C_t = C_{t-1}$ hay $M < \epsilon$ thì sẽ xem là đã hội tụ, nếu không thì ta sẽ tiếp tục quay lại cập nhật centroid.

- Sau khi centroid đã hội tụ thì ta sẽ tính inertia cho centroid hội tụ tìm được và đây là "điểm số" ứng với lần chạy đó. Ta sẽ so sánh điểm số giữa n lần chạy để tìm ra bộ tốt nhất.

2.2.3 Giải thích

Dưới đây là phần giải thích cho thuật toán. Như đã đề cập thì ta cần tìm bộ centroid sao cho $inertia = \sum_{j=1}^{k} \sum_{\vec{x} \in S_j} ||\vec{x} - \vec{\mu_j}||^2$ đạt được giá trị nhỏ nhất.
Trước tiên ta sẽ giải thích, vì sao lại chọn trọng tâm của từng cụm để tìm ra centroid mới.
Ta xét bài toán: Cho n điểm $A_1, ..., A_n \in \mathbb{R}^d$. Ta tìm điểm $M \in \mathbb{R}^d$ sao cho $S = \sum_{i=1}^{n} (MA_i)^2$ đạt giá trị nhỏ nhất.
Ta gọi G là trọng tâm của hệ $A_1, ..., A_n$ hay $\vec{OG} = \frac{1}{n} \sum \vec{OA_i}$. Từ đây ta có $n\vec{OG} = \sum \vec{OG} + \vec{GA_i} = n\vec{OG} + \sum \vec{GA_i} \Leftrightarrow \sum \vec{GA_i} = \vec{0}(*)$. Vậy ta có G là điểm duy nhất thỏa (*) vì giả sử tồn tại G' cũng thỏa (*) thì $\vec{GG'} = \vec{0}$
Ta xét $(MA_i)^2 = (\vec{MA_i})^2 = (\vec{MG} + \vec{GA})^2 = MG^2 + 2\langle \vec{MG}, \vec{GA_i} \rangle + GA_i^2$.
Khi đó $S = \sum_{i=1}^{n} MG^2 + 2\langle \vec{MG}, \vec{GA_i} \rangle + GA_i^2 = nMG^2 + \sum GA_i^2 + 2\langle \vec{MG}, \sum \vec{GA_i} \rangle = nMG^2 + C$ (bởi vì ta đã biết điểm G và $A_1, ..., A_n$ cố định nên $C = \sum GA_i^2$ là một hằng số). Vì vậy nên để S nhỏ nhất thì M phải là trọng tâm của hệ điểm.
Trở lại hàm inertia thì rõ ràng, cứ sau mỗi bước phân cụm thì ta lại chọn một bộ điểm làm centroid mới thì từ bài toán trên, ta đã thấy được chọn trọng tâm là tối ưu nhất ứng với từng cụm.
Tiếp đến, do ta đã biết thì số lượng dữ liệu là hữu hạn, từ đó dẫn tới có hữu hạn cách chia bộ dữ liệu thành k cụm là hữu hạn mà với mỗi cách chia thì ta sẽ tính được inertia tương ứng nên cũng sẽ có hữu hạn các giá trị của inertia. Theo nguyên lí cực hạn thì ta có inertia phải có giá trị nhỏ nhất.
Ta xét tới tính chất của inertian như sau:
Ta có inertia ứng với $C_{t-1} \geq$ inertia ứng với C_t. Thật vậy, ứng với cách phân cụm cho C_{t-1} thì ta có $inertia_{t-1} = \sum_{i=1}^{k} \sum_{\vec{x} \in S_i} ||\vec{x} - \vec{\mu_i}||^2 \geq \sum_{i=1}^{k} \sum_{\vec{x} \in S_i} ||\vec{x} - \vec{OG_i}||^2 \quad (\Delta)$ với G_i là trọng tâm của S_i mà đây cũng chính là thành phần của centroid C_t. Ta xét trường hợp có $\vec{x}^*$ thuộc cụm u ứng với C_{t-1} nhưng lại thuộc v ứng với C_t thì khi này rõ ràng $||\vec{x^*} - \vec{\mu_u}||^2 \geq ||\vec{x^*} - \vec{OG_v}||^2$, kết hợp với (Δ) thì $inertia_{t-1} \geq \sum_{i=1}^{k} \sum_{x \in S_i^*} ||\vec{x} - \vec{OG_i}||^2 = \sum_{i=1}^{k} \sum_{x \in S_i^*} ||\vec{x} - \vec{\mu_i^*}||^2 = inertia_t$.
Chính vì vậy nên inertia luôn không tăng khi ta cập nhật centroid, và inertia không âm nên theo Định lý Weierstrass thì ta có inertia hội tụ về 1 giá trị thực không âm.
Như ta đã đề cập thì mỗi lần xuất phát từ một bộ centroid ngẫu nhiên, thuật toán sẽ có xu thế hội tụ do ta thấy inertia sẽ phải hội tụ, và ta chỉ quan tâm xem inertia có thể giảm tiếp hay

không khi ta tiếp tục cập nhật centroid nên ta sẽ dừng khi inertia không còn thay đổi nhiều nữa. Trong thuật toán thực tế thì ta sẽ kiểm tra xem nếu centroid ở bước t giống với t-1 thì ta sẽ dừng hay centroid không còn thay đổi nhiều nữa.
Thuật toán này cũng có nhược điểm là tuy ta chạy n lần để tìm các bộ centroid nhưng ta hoàn toàn không thể khẳng định nó là giá trị nhỏ nhất của inertia vì nó có thể là cực trị địa phương, trên thực tế khi ta tăng số lần chạy thử càng nhiều thì càng chắc chắn hơn trong việc tìm ra giá trị nhỏ nhất cho inertia.

Dây chính là phần giải thích cho độ hợp lý của thuật toán KMeans mà nhóm em thực hiện cài đặt trong đồ án này.

3 Thực nghiệm

3.1 Bộ dữ liệu Iris

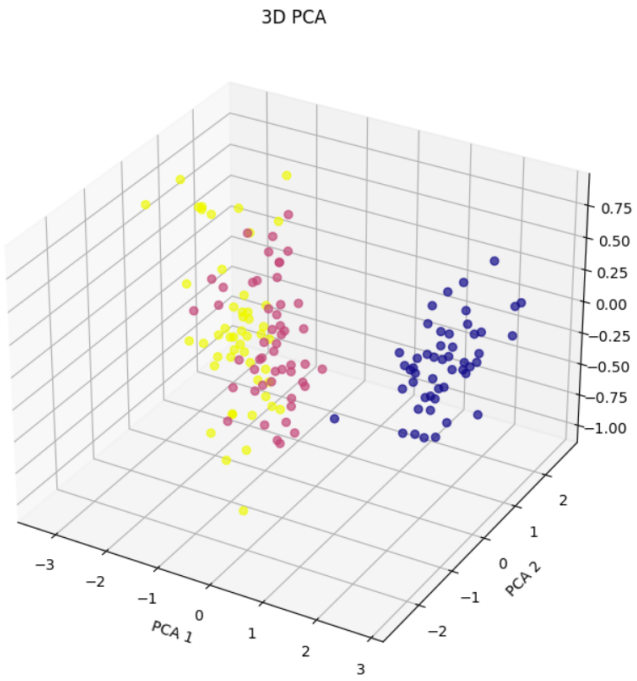
3.1.1 Tiền xử lý dữ liệu

Dữ liệu được chuẩn hóa về trung bình 0 và độ lệch chuẩn 1 để đảm bảo rằng tất cả các thuộc tính có cùng trọng số khi tính toán phương sai trong PCA. Việc chuẩn hóa giúp PCA hoạt động hiệu quả hơn và tránh bị chi phối bởi những thuộc tính có giá trị tuyệt đối lớn hơn.

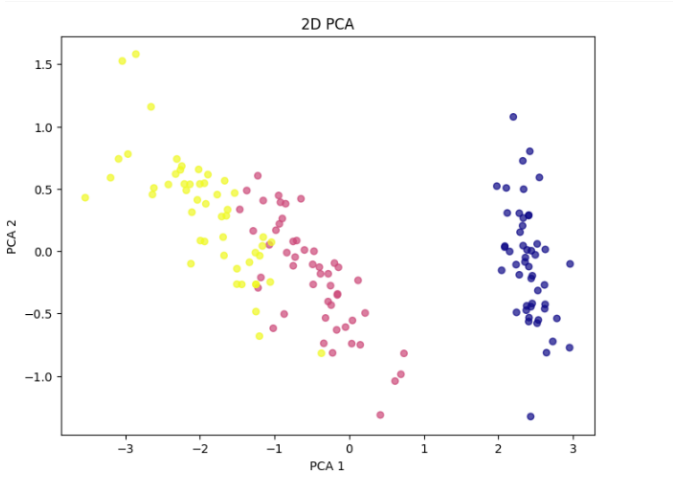
3.1.2 Áp dụng PCA trong bài toán Iris

Sau khi áp dụng PCA để giảm số chiều của dữ liệu Iris từ 4 xuống còn 3, ta thu được tập dữ liệu mới với ba thành phần chính (principal components), biểu diễn phần lớn phương sai của dữ liệu gốc.

Kết quả CEVR: Tổng tích lũy tỉ lệ phương sai được nắm giữ bởi các thành phần PCA trên 3 thành phần chính này lần lượt là: 0.72962445, 0.95813207, 0.99482129. Cho thấy chỉ với 2 thành phần chính đầu tiên chúng ta đã có thể giải thích được khoảng 95.81% phương sai của toàn bộ dữ liệu, tức là hầu hết thông tin quan trọng đã được giữ lại. Việc giảm chiều từ 4 xuống 3, hoặc thậm chí 2 thành phần chính, giúp giảm số chiều của dữ liệu và tăng hiệu quả tính toán mà không làm mất nhiều thông tin.



Hình 1: Biểu diễn dữ liệu Iris dạng 3D sau khi giảm chiều bằng PCA



Hình 2: Biểu diễn dữ liệu Iris dạng 2D sau khi giảm chiều bằng PCA

3.1.3 Phân tích kết quả

Hình 1 minh họa dữ liệu Iris sau khi được chiếu lên ba thành phần chính đầu tiên. Ta cũng có thể minh họa trên hai thành phần chính đầu tiên giống hình 2. Có thể thấy rõ:

- **Iris Setosa** (được tách biệt rất rõ ràng khỏi hai loài còn lại, chứng tỏ PCA đã giữ lại được đặc trưng quan trọng để phân biệt loài này.

- **Iris Versicolor** và **Iris Virginica** có phần chồng lấn lên nhau, cho thấy sự tương đồng cao hơn giữa hai loài này trong không gian đặc trưng đã giảm chiều.
- Tổng phương sai giữ lại bởi hai thành phần chính là rất lớn (thường trên 95%), cho thấy hiệu quả của việc giảm chiều.

3.1.4 Kết luận

Việc sử dụng PCA cho bộ dữ liệu Iris giúp giảm số chiều dữ liệu từ 4 xuống 3, thuận tiện cho việc trực quan hóa và hiểu dữ liệu. Mặc dù PCA là kỹ thuật tuyến tính và không xem xét đến nhãn lớp, nó vẫn cho thấy hiệu quả trong việc phân biệt giữa các lớp dữ liệu. Đặc biệt, khả năng tách biệt rõ ràng của Iris Setosa chứng minh rằng PCA là một công cụ tiền xử lý mạnh mẽ trong các bài toán phân loại hoặc trực quan hóa dữ liệu.

3.2 Bộ dữ liệu ABIDE II

Trong chi tiết thực nghiệm, nhóm thực nghiệm 2 phương pháp cho tiền xử lí dữ liệu là PCA trên toàn bộ dữ liệu và PCA 2 giai đoạn, mỗi cách đều cho n_component của PCA khác nhau và cải tiến để tăng điểm các metrics sử dụng. Sau khi tiền xử lí, dữ liệu được đưa vào KMeans để phân cụm và đánh giá.

3.2.1 Metrics đánh giá phân cụm KMeans

Dữ liệu sau khi tiền xử lí ở cả 2 hai phương pháp, đều được đưa vào thuật toán **KMeans Clustering** với các tham số như sau:

- Số cụm: 2 (tương ứng với hai lớp `Cancer` và `Normal`)
- k_init: 30, đây cũng là giá trị mặc định cho việc chạy kmeans bao nhiêu lần để tìm các bộ centroid mà inertia nhỏ nhất trong các lần chạy (tăng số lần chạy thử càng nhiều thì càng chắc chắn hơn trong việc tìm ra giá trị nhỏ nhất cho inertia, tuy nhiên dữ liệu không hoàn toàn phân cụm được do các thuộc tính của dữ liệu làm dữ liệu của nhãn `Cancer` và `Normal` blend vào nhau nên tăng số lần init có thể dẫn đến tìm centroid không hoàn toàn tối ưu)

Để đánh giá hiệu quả mô hình trên tập dữ liệu ABIDE-II, nhóm sử dụng các metric sau:

- **Accuracy**: tỷ lệ các dự đoán đúng trên tổng số mẫu.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall**: tỉ lệ các dự đoán positive đúng trên tổng số mẫu positive thật sự.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Precision**: là tỷ lệ các dự đoán positive đúng trên tổng số dự đoán positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **F1-score**: là trung bình điều hòa của Precision và Recall, cân bằng hai chỉ số này.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Trong đó, TP là số lượng True Positive, TN là số lượng True Negative, FP là số lượng False Positive và FN là số lượng False Negative.

Vì KMeans là thuật toán phân cụm chưa có nhãn tối ưu, nên nhãn đầu ra được ánh xạ lại để phù hợp hơn với nhãn thật (`y_true`) dựa vào metric ta chọn. Nhóm viết một hàm đánh giá đơn giản chọn nhãn thích hợp nhất với dữ liệu chỉ có hai nhãn đầu ra là 0 và 1 dựa trên điểm accuracy. `y_pred` nào có accuracy cao hơn thì chọn nhãn đó.

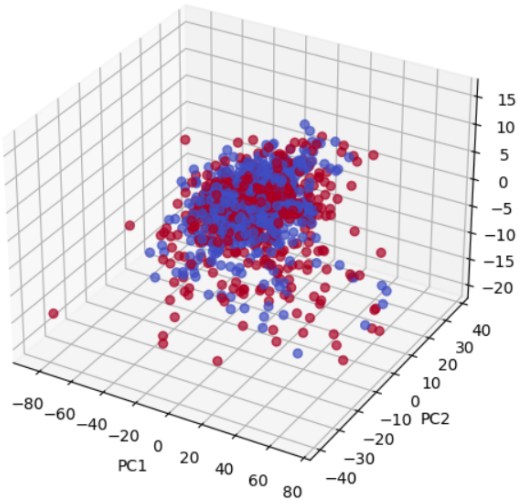
Ví dụ như sau, ta có `y_true` = [0, 1, 1, 0]. Giả sử ta còn có thêm `y_pred` = [0, 1, 1, 1] hoặc `1 - y_pred` = [1, 0, 0, 0] nhờ việc lật ngược lại giá trị dự đoán. Rõ ràng `y_pred` = [0, 1, 1, 1] có accuracy = 3 / 4 lớn hơn accuracy của `1 - y_pred` = [1, 0, 0, 0] là 1 / 4 nên ta chọn `y_pred` là kết quả cuối cùng.

Bảng 1: Pseudocode cho Best map binary

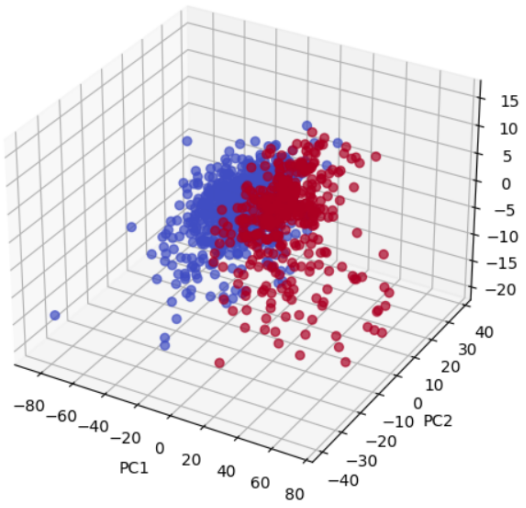
Best map binary
Input: `y_pred` từ KMeans và `y_true`
Output: `y_pred` hợp lí nhất
A1 ← Accuracy `(y_pred)`
A2 ← Accuracy `(1 - y_pred)`
Return `(y_pred)` if **A1** ≥ **A2** else `(1 - y_pred)`

3.2.2 PCA trên toàn bộ dữ liệu + KMeans

Sau đây là phương pháp PCA trên toàn bộ dữ liệu và đưa vào KMeans. Bộ dữ liệu đầu vào có kích thước lớn với 1444 đặc trưng, khi chọn PCA trên toàn bộ dữ liệu, các đặc trưng vẫn còn nhiễu đáng kể vì cho các các đặc trưng khiến các mẫu dữ liệu không phân cụm được.



Hình 3: Sau khi giảm chiều bằng PCA trên toàn bộ dữ liệu, biểu diễn các nhãn y_true



Hình 4: Sau khi giảm chiều bằng PCA trên toàn bộ dữ liệu, biểu diễn các nhãn y_pred

Thực nghiệm lần 1 đơn giản với PCA và KMeans, nhóm chọn n_components=3 (giá trị ngẫu nhiên nhóm chọn, chọn bằng 3 cũng để có thể biểu diễn lên không gian 3 chiều). Khi quan sát hình 3, dữ liệu X đã được PCA (X_pca) và giá trị nhãn thực tế (y_true) gần như blend vào nhau. Không thể phân nhóm dữ liệu.

Các metric đánh giá nhận được: Accuracy: 0.539841; Recall: 0.408207; Precision: 0.501326; f1_score: 0.45.

Rõ ràng ta cũng thấy giá trị accuracy là 0.539, precision là 0.5, gần như xấp xỉ giá trị khi ta cho random nhãn 0 hoặc 1 (xác suất $1/2 = 0.5$). Hình 4 biểu diễn phân cụm của y_pred cũng chưa cho thấy sự hiệu quả.

Thực nghiệm lần 2 tiếp tục, nhóm chọn n_components=50, n_components=100, n_components=500 (đều là các giá trị lớn hơn đáng kể so với 3). Kết quả đánh giá trên các metrics KHÔNG THAY ĐỔI (Accuracy: 0.539841; Recall: 0.408207; Precision: 0.501326; f1_score: 0.45). Điều này làm ta dự đoán đối với bất kì giá trị n_components nào thì kết quả đánh giá phân cụm dường như không đổi.

3.2.3 PCA 2 giai đoạn + KMeans

Nhóm đề xuất một phương án là **giảm chiều hai giai đoạn**. Trước tiên, ta tiến hành PCA riêng cho từng nhóm, giảm chiều trong từng nhóm để loại bỏ các đặc trưng dư thừa hoặc nhiễu trong nhóm đó. Đây là phương pháp giảm chiều hiệu quả từng nhóm trước khi kết hợp, tránh mất thông tin quan trọng khi làm PCA trực tiếp trên toàn bộ dữ liệu.

Tách đặc trưng theo nhóm cấu trúc

Dựa trên quy ước đặt tên các đặc trưng, em phân tách dữ liệu thành 4 nhóm chính:

- `fsArea`: diện tích vỏ não
- `fsCt`: độ dày vỏ não

- `fsLgi`: chỉ số độ nếp gấp vỏ não
- `fsVol`: thể tích vùng não

Khi thực hiện giảm chiều trên các nhóm, ta loại bỏ được các đặc trưng dư thừa và nhiễu trên mỗi nhóm đó. Sau đó, ta ghép các tập đặc trưng đã giảm chiều thành một tập chung rồi mới tiếp tục áp dụng PCA lần nữa trên tập đặc trưng chung này để giảm chiều tổng thể, tạo thành biểu diễn cuối cùng cho mỗi mẫu dữ liệu.

Khi đã hoàn thành tiền xử lí dữ liệu nhờ giảm chiều 2 giai đoạn, ta sử dụng KMeans để Cluster dữ liệu thành 2 nhóm.

Tìm n_components tối ưu

Việc tìm n_component cho từng nhóm là khá là phức tạp, nhóm em đã chọn số component sao cho tỉ lệ CEVR chiếm khoảng 70% ở mỗi nhóm, n_component của các nhóm lúc này là 46 thì CEVR là 70% (riêng fsVol mang kết quả khác). Nhóm chỉ muốn loại bỏ khoảng 30% các đặc trưng mang tính nhiễu trong từng nhóm vì còn thực hiện PCA lần 2 trên toàn bộ dữ liệu nữa. Dưới đây là CEVR của từng nhóm khi chọn n_component = 46.

- `fsArea`: 0.7085477462583457
- `fsCt`: 0.6927189445263189
- `fsLgi`: 0.7085522953180713
- `fsVol`: 0.6714283261495282

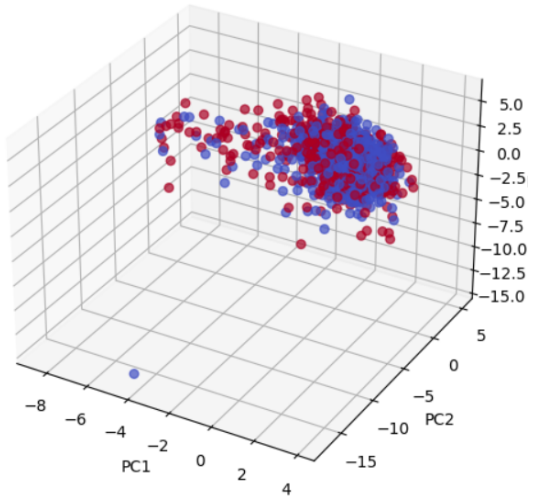
Tuy nhiên ở nhóm thể tích vùng não, n = 46 chỉ thể hiện tổng phương sai tích lũy xấp xỉ 0.67, khi n = 55 (tăng 9 thuộc tính) thì thể tích vùng não mới thể hiện tổng phương sai tích lũy được 70% của nhóm này. Nhóm Volume này có thể chứa nhiều thông tin nhiễu hơn, sau khi thực hiện giảm chiều PCA lần 1 cho các nhóm và thực hiện ghép các đặc trưng lại với nhau: ta có thể thử loại bỏ đặc trưng này để xem xét kết quả.

Thực nghiệm tìm n_component lần 1

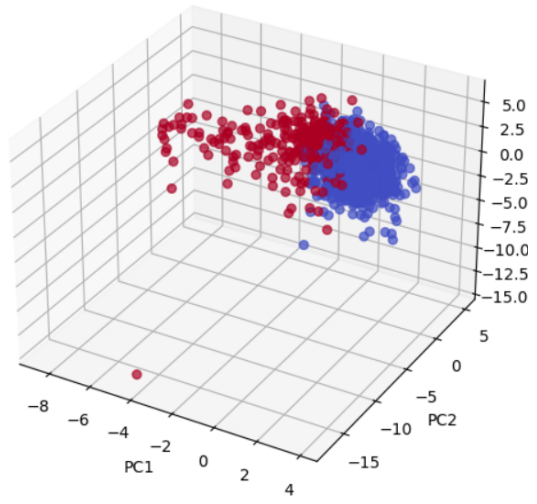
Nhóm kết hợp phương pháp PCA 2 lần. Lưu ý là PCA lần 1 có đầy đủ cả 4 nhóm và đều chọn n_component = 46. Khi thực hiện PCA cho các đặc trưng đã được nối lại với nhau. Nhóm thử các cái giá trị n_component từ 40 đến 100 và rút ra kết luận sau:

- n từ 19 đến 44: accuracy dao động trong khoảng 0.51 - 0.53
- n = 45: accuracy = 0.578685, CEVR=0.725
- n = 46: accuracy = 0.582669, CEVR=0.737
- n > 46: accuracy dao động từ 0.51 đến 0.58.

Tuy nhiên, kết quả accuracy cao nhất trên n = 46 khi đánh giá trên các metric khác: recall khá thấp (0.328294) dẫn đến tuy precision tăng (0.584615) nhưng điểm f1 chỉ xấp xỉ 0.420470



Hình 5: Thực nghiệm lần 1 cho PCA 2 lần, biểu diễn các nhãn y_true



Hình 6: Thực nghiệm lần 1 cho PCA 2 lần, biểu diễn các nhãn y_pred

Hình 5 khi thể hiện trên 3 thành phần chính đầu tiên để minh họa cho y_true vẫn còn blend vào nhau khá nhiều và hình 6 mình họa cho việc phân cụm dự liệu này cũng không tốt.

Thực nghiệm tìm n_component lần 2

Nhóm thử loại bỏ thuộc tính fsVol khỏi kết quả. Tức là thực hiện PCA lần 1 có chỉ có 3 nhóm. Khi thực hiện PCA cho các đặc trưng đã được nối lại với nhau. Nhóm thử các cái giá trị n_component

từ 3 đến 29 và rút ra kết luận sau:

- n từ 3 đến 5: accuracy dao động trong khoảng 0.53
- n từ 6 đến 14 và 18, 19: accuracy xấp xỉ 0.58
- n = 15, 16, 17: accuracy xấp xỉ 0.59
- n = 20: accuracy xấp xỉ 0.611554, CEVR = 0.346
- n = 21 đến 28: accurcacy khoảng 0.58
- n = 29: accuracy khoảng 0.56

Kết quả accuracy cao nhất trên n = 20 khi đánh giá trên các metric khác: recall tăng (0.496760), precision tăng (0.594315), f1 tăng (0.541176) đáng kể.

Tuy nhiên nhóm vẫn muốn cải thiện kết quả. Nhìn vào dữ liệu, nhóm đánh giá như sau:

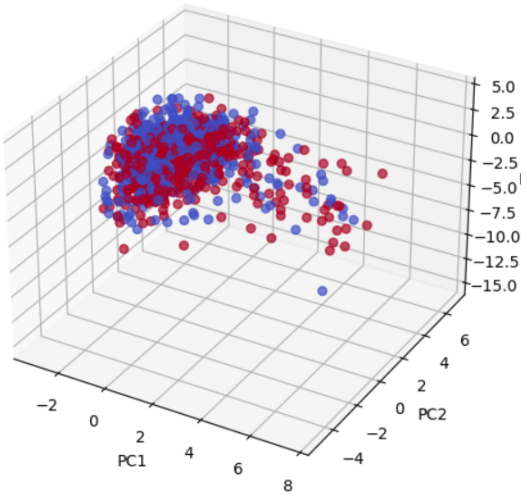
- fsCt – Độ dày vỏ não (Cortical thickness): Nhạy cảm với thoái hóa thần kinh (ví dụ Alzheimer, Parkinson) và thường được dùng để theo dõi sự thay đổi theo tuổi hoặc bệnh lý.
- fsArea – Diện tích vỏ não (Surface area): Gắn với di truyền và phát triển trí tuệ.
- fsLgi – Chỉ số độ nếp gấp (Local Gyrification Index): Gắn với sự phát triển thần kinh sớm, một số rối loạn phát triển thần kinh như tự kỷ, tâm thần phân liệt.

3 chỉ số mà ta quan tâm, dữ liệu gốc (chưa được chỉnh sửa lại có liên quan đến tự kỉ nên nhóm thử tăng n_component cho chỉ số NẾP GẤP, và quyết định chọn số 53 vì cho kết quả tổng thể cao).

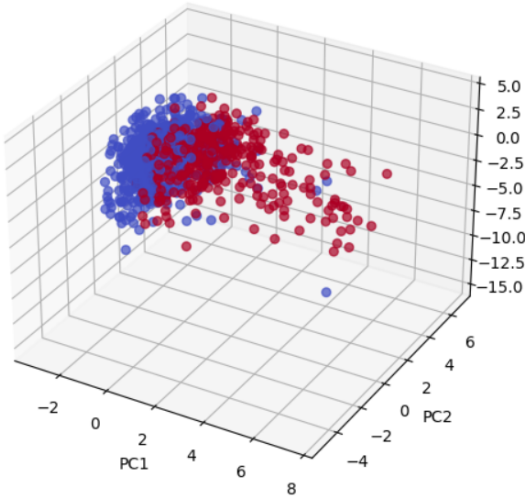
Khi chọn n_components cho lần PCA thứ 2, nhóm thử các cái giá trị n_component từ 6 đến 29 và rút ra kết luận sau:

- n từ 6 đến 19: accuracy dao động với giá trị 0.58, 0.59
- n = 22: accuracy = 0.6026
- n = 23: accuracy = 0.621514, CEVR: 0.37
- n từ 24 đến 30: accuracy dao động với giá trị 0.58, 0.59
- n = 31: accuracy = 0.51

Kết quả accuracy (0.621514) cao nhất trên n = 23 khi đánh giá trên các metric khác: recall tăng (0.498920), precision tăng (0.609499), f1 tăng (0.548694).



Hình 7: Thực nghiệm lần 2 cho PCA 2 lần, biểu diễn các nhãn y_true



Hình 8: Thực nghiệm lần 2 cho PCA 2 lần, biểu diễn các nhãn y_pred

Hình 7 khi thể hiện trên 3 thành phần chính đầu tiên để minh họa cho y_true tuy vẫn còn blend vào nhau nhưng đã có cải thiện và hình 8 mình họa cho việc phân cụm dự liệu này cũng cho kết quả khả quan hơn so với các lần phân cụm trước đó.

Đây là kết quả cao nhất mà nhóm đạt được và kết quả điều hòa giữa các metric hơn hết. Nên đây cũng là kết quả mà nhóm submit cho lab 2.

3.2.4 Kết luận

Chỉ số	Phương pháp 1	Phương pháp 2	Tăng (%)
Accuracy	0.539841	0.621514	+15.12%
Recall	0.408207	0.498920	+22.23%
Precision	0.501326	0.609499	+21.57%
F1-score	0.450000	0.548694	+21.93%

Bảng 2: So sánh hiệu quả giữa hai phương pháp

Kết quả so sánh cho thấy **PCA 2 lần có hiệu quả hơn** so với PCA 1 lần trên toàn bộ các chỉ số đánh giá. Cụ thể, accuracy tăng khoảng **15.12%**, recall tăng **22.23%**, precision tăng **21.57%**, và điểm F1-score (chỉ số tổng hợp giữa Recall và Precision) tăng **21.93%**.

Tài liệu

[1] Scikit-learn, "The Iris Dataset," [Online]. Available: `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`.

[2] Scikit-learn, "Principal Component Analysis (PCA)," [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`.

[3] Scikit-learn, "Clustering on the Iris dataset," [Online]. Available: `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html`.

[4] S. Raschka, "Unsupervised Learning with k-Means and PCA," [Online]. Available: `https://sebastianraschka.com/Articles/2014_pca_step_by_step.html`.

[5] Scikit-learn, "Preprocessing data," [Online]. Available: `https://scikit-learn.org/stable/modules/preprocessing.html`.

[6] Analytics Vidhya, "Why Data Scaling is Important in Machine Learning," [Online]. Available: `https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/`.

[7] J. Brownlee, "How to Use the Elbow Method to Find the Optimal Number of Clusters," [Online]. Available: `https://machinelearningmastery.com/k-means-clustering-in-python/`.