ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC

Học máy

Giảng viên: Cao Văn Chung

Mã lớp học phần: MAT3533

Nhận diện bảng chữ cái ngôn ngữ ký hiệu

Thành viên nhóm

Nguyễn Vĩ Chí Cường	(22000073)
Đỗ Tiến Dũng	(22000075)
Đỗ Hữu Đạt	(22000080)

Hà Nội, 2025

Lời nói đầu

Trong thời đại công nghệ số phát triển mạnh mẽ, trí tuệ nhân tạo (AI) và học máy (Machine Learning) ngày càng đóng vai trò quan trọng trong nhiều lĩnh vực của đời sống, đặc biệt là trong việc hỗ trợ giao tiếp và nâng cao chất lượng cuộc sống cho cộng đồng. Một trong những ứng dụng đầy tiềm năng của học máy là khả năng nhận diện ngôn ngữ ký hiệu – một phương tiện giao tiếp thiết yếu đối với người khiếm thính.

Đề tài này tập trung vào bài toán nhận diện ký hiệu tay trong ngôn ngữ ký hiệu Mỹ (ASL - American Sign Language) thông qua hình ảnh. Với sự hỗ trợ của bộ dữ liệu "ASL Handsign Dataset (Grayscaled & Thresholded)" từ nền tảng Kaggle, các phương pháp học máy và học sâu sẽ được áp dụng để xây dựng mô hình phân loại ký hiệu tay một cách hiệu quả và chính xác.

Thông qua đề tài này, người thực hiện mong muốn không chỉ làm quen và ứng dụng các kỹ thuật trong lĩnh vực thị giác máy tính mà còn góp phần nhỏ vào việc nghiên cứu các giải pháp công nghệ giúp hỗ trợ người khiếm thính trong việc giao tiếp và hòa nhập với cộng đồng.

Nội dung đề tài được trình bày thành các chương bao gồm: giới thiệu bộ dữ liệu, tiền xử lý, trực quan hóa, xây dựng mô hình phân loại, đánh giá hiệu suất mô hình và kết luận. Qua đó, đề tài nhằm mục tiêu khái quát toàn bộ quy trình xây dựng một hệ thống nhận diện ký hiệu tay dựa trên hình ảnh, từ bước thu thập dữ liệu đến triển khai mô hình.

Phân công công việc

STT	Họ và Tên	Công Việc Phụ Trách
1	Đỗ Hữu Đạt	Nghiên cứu và tổng hợp kiến thức về thuật toán K Nearest Neighbor, KMeans, giảm chiều PCA.
2	Đỗ Tiến Dũng	Nghiên cứu và tổng hợp kiến thức về thuật toán Logistic Regression, chuẩn hóa dữ liệu.
3	Nguyễn Vĩ Chí Cường	Nghiên cứu và tổng hợp kiến thức về thuật toán Naive Bayes, DBScan, tiền xử lý và trực quan hóa dữ liệu.

Mục lục

1 Giới thiệu bộ dữ liệu

1.1 Tổng quan về bộ dữ liệu ASL Handsign

Tập dữ liệu ASL Handsign Dataset (Grayscaled & Thresholded) là một tập dữ liệu hình ảnh được thiết kế cho bài toán nhận dạng ngôn ngữ ký hiệu Mỹ (ASL - American Sign Language). Mỗi hình ảnh trong tập dữ liệu thể hiện một ký hiệu tay tương ứng với các chữ cái trong bảng chữ cái tiếng Anh. Bộ dữ liệu đã được xử lý dưới dạng ảnh Grayscaled và nhị phân (thresholded), giúp tăng hiệu quả khi huấn luyện các mô hình học máy và học sâu.

Tập dữ liệu này rất phù hợp cho các bài toán phân loại ảnh trong lĩnh vực thị giác máy tính, đặc biệt là các bài toán về nhận diện ký hiệu ngôn ngữ.

1.2 Nguồn dữ liệu

Bộ dữ liệu được tải từ trang Kaggle với đường dẫn:

https://www.kaggle.com/datasets/furkanakdeniz/asl-handsign-dataset-grayscaled-thresholded

Bộ dữ liệu được chia sẻ công khai bởi người dùng furkanakdeniz.

1.3 Cấu trúc của bộ dữ liệu

Bộ dữ liệu được tổ chức thành hai thư mục chính:

- train/: Bao gồm các hình ảnh dùng để huấn luyện mô hình. Mỗi ảnh tương ứng với một chữ cái từ A đến Y (trừ J và Z – là các ký hiệu động, không được thể hiện bằng ảnh tĩnh). Mỗi lớp (class) là một thư mục chứa nhiều ảnh đại diện cho ký hiệu tương ứng. Số lượng ảnh train: Grayscale- 22.880 | Threshold-30.050
- test/: Chứa tập ảnh kiểm tra, được tổ chức tương tự như tập huấn luyện. Số lượng ảnh test Grayscale- 4.053 | Threshold-7.523
- Đặc điểm của các hình ảnh trong bộ dữ liệu:
  - Ảnh đen trắng (grayscale) đã được xử lý threshold để làm nổi bật phần ký hiệu tay.
  - Kích thước ảnh: 128x128 pixel, thuận tiện khi áp dụng các kiến trúc CNN đơn giản.
  - Mỗi ảnh thuộc về một trong 24 lớp (từ A đến Y, ngoại trừ J và Z).

2 Phương pháp tiền xử lý dữ liệu

2.1 Resize ảnh (Sử dụng hàm resize của thư viện opencv)

Resize ảnh là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.1.1 Mục đích

Biến đổi hình ảnh từ các kích thước ban đầu thành một kích thước chuẩn nhất định, giúp tạo ra sự đồng nhất về số chiều đặc trưng và cải thiện hiệu năng của mô hình học máy.

2.1.2 Tác động

- **Chuẩn hóa kích thước ảnh**: Đảm bảo tất cả hình ảnh có cùng kích thước (ví dụ: 64×64), tạo vector đặc trưng có độ dài cố định (4096 chiều), phù hợp với đầu vào của Gaussian Naive Bayes.
- **Giảm số chiều đặc trưng**: Giảm số lượng pixel (ví dụ: từ $128 \times 128 = 16384$ xuống $64 \times 64 = 4096$), giảm chi phí tính toán, nhiễu, và cải thiện khả năng tổng quát hóa của mô hình.
- **Bảo toàn thông tin hình ảnh**: Nội suy tuyến tính kép giữ chi tiết hình dạng tay, đảm bảo các ký hiệu ASL (như "A" và "E") vẫn phân biệt được, hỗ trợ Gaussian Naive Bayes mô hình hóa dữ liệu hiệu quả hơn.
- **Cải thiện hiệu suất huấn luyện**: Kích thước ảnh đồng nhất và số chiều giảm giúp quá trình tính toán xác suất trong Gaussian Naive Bayes nhanh hơn, đồng thời giảm nguy cơ sai số số học khi số chiều lớn.

2.2 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.2.1 Mục đích

Biến đổi dữ liệu từ các phạm vi giá trị ban đầu thành các phạm vi giá trị theo một chuẩn nhất định, giúp tạo ra sự cân bằng giữa các đặc trưng và cải thiện hiệu năng của mô hình học máy.

2.2.2 Phương pháp Standardization (Z-score normalization)

Nội dung phương pháp: Phương pháp này sẽ chuyển đổi dữ liệu về một phân bố trong đó giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

Công thức chuẩn hóa:

$$X' = \frac{X - \text{mean}(X)}{\text{std}(X)},$$

trong đó:

– $\text{mean}(X)$ là kỳ vọng (giá trị trung bình) của X,

– $\text{std}(X)$ là độ lệch chuẩn của X.

2.2.3 Tác động

- **Hỗ trợ quá trình huấn luyện học máy hội tụ nhanh hơn**: Chuẩn hóa dữ liệu làm cho các đặc trưng có cùng phạm vi giá trị, từ đó gradient của các đặc trưng sẽ có tầm ảnh hưởng tương đồng, giúp cho quá trình cập nhật tham số diễn ra một cách đồng đều và tránh được tình trạng *overshooting*, tạo điều kiện thuận lợi cho quá trình hội tụ tới lời giải tối ưu.
- **Đảm bảo cân bằng đặc trưng**: Các đặc trưng có phạm vi giá trị lớn hơn sẽ ảnh hưởng lớn hơn đến hàm mất mát, khiến mô hình thiên về các đặc trưng đó tạo ra một mô hình không cân bằng. Khi dữ liệu được chuẩn hóa, mọi đặc trưng đều có cùng phạm vi giá trị và cùng mức biến động, giúp khắc phục tình trạng trên.
- **Dễ dàng hơn trong điều chỉnh tham số**: Chuẩn hóa dữ liệu giúp làm giảm sự biến động của gradient, từ đó việc tinh chỉnh tốc độ học cho phù hợp với độ lớn của gradient trở nên dễ dàng hơn, giúp quá trình huấn luyện tránh xảy ra bất ổn.
- **Cải thiện độ chính xác cho mô hình**: Khi dữ liệu được chuẩn hóa, các mối quan hệ phức tạp trong dữ liệu được mô hình hóa hiệu quả hơn, từ đó cải thiện độ chính xác của mô hình.

2.3 Phân tích thành phần chính (PCA)

Phân tích thành phần chính (PCA) là một bước quan trọng trong tiền xử lý dữ liệu trước khi huấn luyện mô hình học máy, đặc biệt trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL) với cách tiếp cận chuyển hình ảnh 2D thành vector các pixel.

2.3.1 Mục đích

Giảm số chiều của dữ liệu bằng cách biến đổi các đặc trưng ban đầu thành một tập hợp các thành phần chính mới, giúp loại bỏ nhiễu, giảm chi phí tính toán và cải thiện hiệu năng của mô hình học máy.

2.3.2 Phương pháp PCA

Nội dung phương pháp: Phương pháp này tìm ra các thành phần chính (principal components) là các hướng trong không gian dữ liệu có phương sai lớn nhất, sau đó chiếu dữ liệu gốc lên các hướng này để giảm số chiều.

Các bước thực hiện:

– Chuẩn hóa dữ liệu: Đưa dữ liệu về phân bố có trung bình bằng 0 và độ lệch chuẩn bằng 1.

– Tính ma trận hiệp phương sai: Ma trận này thể hiện mức độ tương quan giữa các đặc trưng.

– Phân tích giá trị riêng và vector riêng: Tìm các giá trị riêng và vector riêng của ma trận hiệp phương sai, trong đó các vector riêng tương ứng với các thành phần chính.

– Sắp xếp các thành phần chính: Sắp xếp các vector riêng theo thứ tự giảm dần của giá trị riêng, chọn k thành phần chính đầu tiên (với k nhỏ hơn số chiều ban đầu).

– Chiếu dữ liệu: Chiếu dữ liệu gốc lên không gian mới được tạo bởi k thành phần chính.

Công thức chiếu dữ liệu: Giả sử dữ liệu gốc là X (ma trận $n \times d$ với n mẫu và d chiều), ma trận thành phần chính là W (ma trận $d \times k$ chứa k vector riêng), dữ liệu sau khi giảm chiều là:

$$X_{\text{PCA}} = X \cdot W,$$

trong đó:

– X_{PCA} là ma trận $n \times k$ chứa dữ liệu sau khi giảm chiều,

– W là ma trận chứa k thành phần chính (các vector riêng).

Ví dụ: Với vector pixel ban đầu có kích thước 784 chiều (từ ảnh 28×28), áp dụng PCA để giảm xuống 50 chiều. Giả sử sau khi tính toán, $k = 50$ thành phần chính được chọn, giữ lại 95% phương sai của dữ liệu. Dữ liệu sau khi chiếu sẽ có kích thước $n \times 50$, với n là số mẫu.

2.3.3 Tác động

- **Giảm số chiều đặc trưng**: Trong bài toán ASL, PCA giảm số chiều từ 4096 (ảnh 64×64) xuống còn 83, giảm chi phí tính toán và nhiễu, giúp mô hình Gaussian Naive Bayes hoạt động hiệu quả hơn.
- **Loại bỏ nhiễu và giữ thông tin chính**: PCA tập trung vào các thành phần có phương sai lớn nhất, loại bỏ các đặc trưng ít quan trọng (nhiễu), giữ lại thông tin chính về hình dạng tay trong các ký hiệu ASL.
- **Cải thiện hiệu suất huấn luyện**: Số chiều giảm làm giảm số lượng tham số cần ước lượng trong Gaussian Naive Bayes (như $\mu_{c,i}$, $\sigma^2_{c,i}$), giúp quá trình huấn luyện nhanh hơn và giảm nguy cơ quá khớp.
- **Hỗ trợ trực quan hóa dữ liệu**: Khi giảm xuống 2 hoặc 3 chiều, PCA cho phép trực quan hóa dữ liệu, giúp phân tích sự phân tách giữa các lớp (các ký hiệu từ A đến Z) trong không gian mới.

3 Phân tích và trực quan hóa dữ liệu

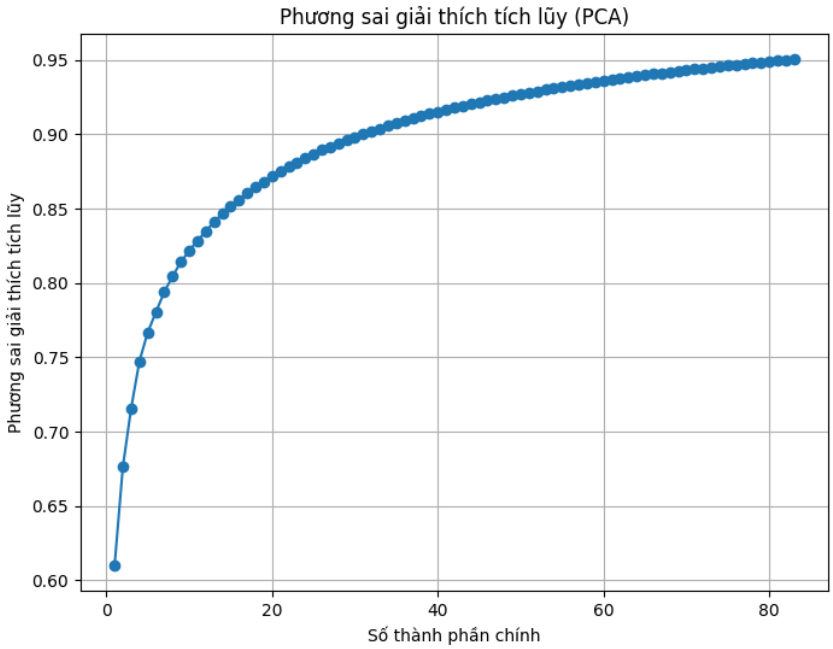
3.1 Phân tích

Phân bố nhãn

Bảng 1: Phân bố số lượng mẫu theo từng nhãn

Nhãn	Số lượng	Nhãn	Số lượng	Nhãn	Số lượng
D	1579	M	1082	P	931
E	1421	Q	1078	O	904
F	1247	I	1069		
N	1235	L	1047		
T	1185	V	1026		
B	1159	A	1023		
C	1153	H	1022		
K	1143	R	1021		
X	1134	W	1008		
S	1132				
Y	1130				
U	1108				
G	1096				

Lượng thông tin được bảo tồn theo phương sai giải thích:



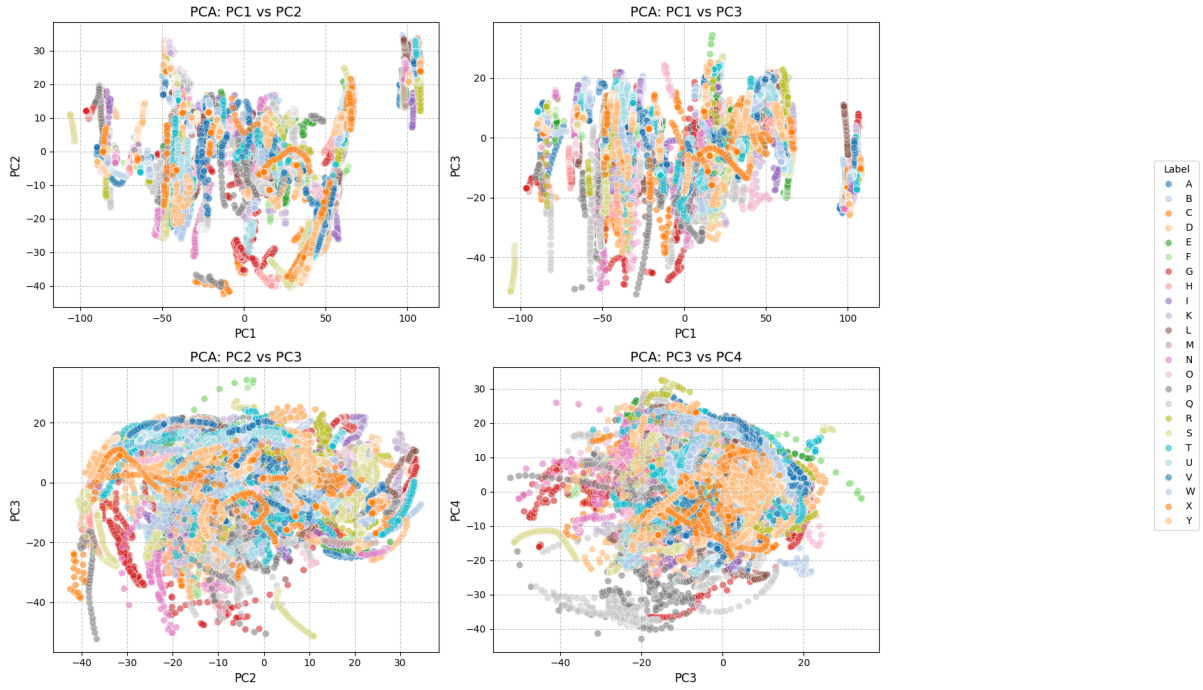
Hình 1: Biểu đồ phương sai giải thích tích lũy theo số lượng thành phần chính (PCA)

Nhận xét Từ biểu đồ Hình 1 và kết quả tính toán cụ thể, ta có một số nhận xét sau:

- Biểu đồ cho thấy phương sai giải thích tích lũy tăng nhanh trong những thành phần chính đầu tiên và có xu hướng hội tụ sau khoảng 40 thành phần.
- Với **4 thành phần chính (PC)**, tổng phương sai giữ lại là **0.747** ($\sim 74.7\%$), tức là gần 75% thông tin ban đầu đã được giữ lại.
- Khi tăng lên **6 thành phần chính**, tổng phương sai giữ lại là **0.781** ($\sim 78.1\%$), cho thấy việc thêm 2 thành phần chỉ cải thiện nhẹ khả năng giữ lại thông tin.
- Do sự cải thiện sau PC6 là không đáng kể, ta có thể chọn từ 4 đến 6 thành phần chính để cân bằng giữa độ phức tạp mô hình và lượng thông tin giữ lại.
- Ngoài ra, nếu yêu cầu giữ lại trên 90% phương sai, cần dùng khoảng từ 30 đến 40 thành phần chính, theo xu hướng biểu đồ.

3.2 Trực quan hóa

Trực quan hóa đối với dữ liệu theo từng cặp 02 thành phần chính.



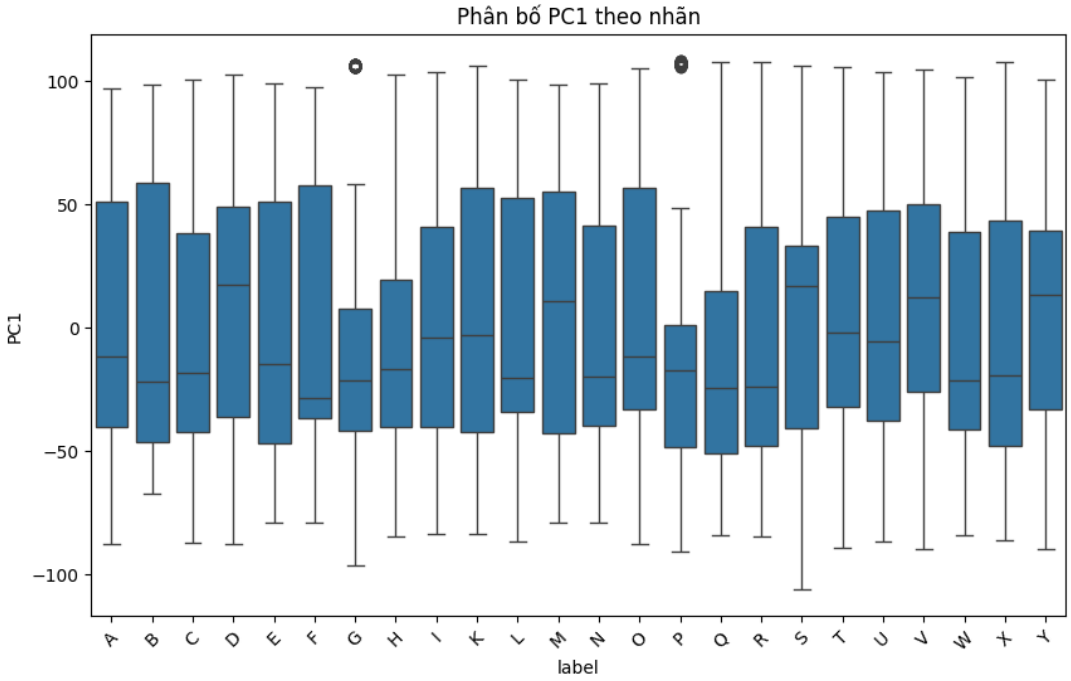
Hình 2: Biểu diễn dữ liệu sau khi giảm chiều bằng PCA theo các cặp thành phần chính

Nhận xét Từ Hình 2, ta có thể đưa ra một số nhận xét sau:

- Dữ liệu sau khi giảm chiều bằng PCA cho thấy một số cụm nhãn có xu hướng tách biệt tương đối rõ ràng, như các nhãn **A**, **B**, và **D**.
- Tuy nhiên, nhiều nhãn vẫn còn phân bố chồng lấn lên nhau, đặc biệt là ở các mặt phẳng PC1-PC2 và PC1-PC3, cho thấy việc phân loại sẽ gặp khó khăn nếu chỉ dựa trên 2 hoặc 3 thành phần chính.
- Các biểu đồ PCA thể hiện rõ tính chất phân bố không đồng đều của dữ liệu trong không gian đặc trưng, cho thấy cần xem xét thêm các đặc trưng bổ sung hoặc kỹ thuật tách tuyến tính/phi tuyến khác (như t-SNE, UMAP).

- Tổng thể, PCA giúp hình dung dữ liệu đa chiều dưới dạng trực quan, nhưng có thể chưa đủ để tách biệt rõ ràng tất cả các nhãn trong bài toán hiện tại.

Mối quan hệ của một số chiều dữ liệu chính với đầu ra



Hình 3: Phân bố thành phần chính đầu tiên (PC1) theo từng nhãn

Nhận xét

Dựa vào biểu đồ Hình 3 và hệ số tương quan giữa các thành phần chính với nhãn, ta có một số đánh giá như sau:

- Biểu đồ boxplot cho thấy **PC1 có sự phân bố khá đồng đều giữa các nhãn**, tuy nhiên không có sự tách biệt rõ ràng giữa các cụm nhãn. Điều này gợi ý rằng PC1 **không mang nhiều thông tin phân biệt các lớp**.
- Các hệ số tương quan Pearson giữa 6 thành phần chính đầu tiên với nhãn như sau:

Thành phần chính	Tương quan với nhãn
PC1	0.0017
PC2	-0.0297
PC3	0.0038
PC4	-0.0178
PC5	**-0.1507**
PC6	**0.1057**

- Nhìn chung, các hệ số tương quan đều rất nhỏ, cho thấy các thành phần chính không có mối quan hệ tuyến tính mạnh với nhãn. Tuy nhiên, **PC5 và PC6 có giá trị tương quan tương đối nổi bật** so với các thành phần còn lại. Điều này gợi ý rằng nếu có mối quan hệ giữa không gian đặc trưng PCA và nhãn, thì chúng có thể xuất hiện ở các chiều cao hơn.

- Ngoài ra, có thể cân nhắc tiếp tục trực quan hóa theo cặp (pairplot hoặc scatter 2D/3D) giữa PC5, PC6 và nhãn để kiểm tra khả năng hình thành cụm (clustering) trong không gian PCA.

4 Phân cụm

4.1 KMeans

4.1.1 Giới thiệu

Thuật toán K-means là một phương pháp học máy không giám sát (unsupervised learning) được sử dụng để phân cụm dữ liệu. Mục tiêu là chia tập dữ liệu thành K cụm sao cho các điểm trong cùng một cụm có đặc điểm tương tự, dựa trên khoảng cách Euclidean đến trung tâm cụm. Thuật toán này được ứng dụng rộng rãi trong phân đoạn thị trường, xử lý ảnh, phân tích văn bản, và nhiều lĩnh vực khác.[3]

4.1.2 Phân tích toán học

Ký hiệu toán học

Giả sử có N điểm dữ liệu được biểu diễn dưới dạng ma trận $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in R^{d\times N}$, trong đó $\mathbf{x}_i \in R^{d\times 1}$ là vector đặc trưng của điểm dữ liệu thứ i trong không gian d-chiều. $K < N$ là số cụm cần phân chia. Mục tiêu là tìm các trung tâm cụm $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K \in R^{d\times 1}$ và nhãn (label) của mỗi điểm dữ liệu.

Với mỗi điểm dữ liệu $\mathbf{x}_i$, nhãn được biểu diễn bởi vector one-hot $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iK}]$, trong đó: - $y_{ik} = 1$ nếu $\mathbf{x}_i$ thuộc cụm k. - $y_{ij} = 0\ \forall j \neq k$.[3]

Điều này đảm bảo mỗi điểm dữ liệu chỉ thuộc một cụm. Ràng buộc của $\mathbf{y}_i$ được viết dưới dạng toán học như sau:[3]

$$y_{ik} \in \{0, 1\}, \quad \sum_{k=1}^{K} y_{ik} = 1 \quad \forall i$$

Ma trận nhãn $\mathbf{Y} = [\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_N]$ và ma trận trung tâm $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K]$ lần lượt biểu diễn nhãn của tất cả điểm dữ liệu và trung tâm của các cụm.

Hàm mất mát[3]

Nếu coi $\mathbf{m}_k$ là đại diện (trung tâm) của cụm k, sai số khi gán điểm $\mathbf{x}_i$ vào cụm k là $\|\mathbf{x}_i - \mathbf{m}_k\|_2^2$. Vì $\mathbf{x}_i$ chỉ thuộc cụm k nếu $y_{ik} = 1$, sai số này có thể viết lại thành:[3]

$$y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Hàm mất mát tổng quát cho toàn bộ dữ liệu được định nghĩa như sau:

$$L(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Hàm $L(\mathbf{Y}, \mathbf{M})$ đo lường tổng bình phương khoảng cách từ các điểm dữ liệu đến trung tâm cụm tương ứng, với ràng buộc $y_{ij} \in \{0, 1\}$ và $\sum_{j=1}^{K} y_{ij} = 1\ \forall i$.

Bài toán tối ưu[3]

Mục tiêu của K-means là tìm $\mathbf{Y}$ và $\mathbf{M}$ để tối thiểu hóa hàm mất mát:

$$\mathbf{Y}, \mathbf{M} = \arg\min_{\mathbf{Y},\mathbf{M}} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

subject to:

$$y_{ij} \in \{0, 1\} \ \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1 \ \forall i$$

Thuật toán tối ưu hóa hàm mất mát[3]

Cố định $\mathbf{M}$*, tìm* $\mathbf{Y}$

Khi các trung tâm $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K]$ được cố định, bài toán trở thành tìm $\mathbf{y}_i$ cho mỗi điểm $\mathbf{x}_i$:

$$\mathbf{y}_i = \arg\min_{\mathbf{y}_i} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

subject to:

$$y_{ij} \in \{0, 1\} \ \forall j; \quad \sum_{j=1}^{K} y_{ij} = 1$$

Vì chỉ có một phần tử $y_{ij} = 1$, bài toán được đơn giản hóa thành:

$$j = \arg\min_{j} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Cố định $\mathbf{Y}$*, tìm* $\mathbf{M}$

Khi nhãn $\mathbf{Y}$ đã xác định (biết điểm nào thuộc cụm nào), nhiệm vụ là tìm trung tâm mới $\mathbf{m}_j$ cho mỗi cụm j để giảm thiểu hàm mất mát. Công thức tính trung tâm là:

$$\mathbf{m}_j = \arg\min_{\mathbf{m}_j} \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Để giải, ta lấy đạo hàm của hàm mất mát theo $\mathbf{m}_j$ và đặt bằng 0. Định nghĩa:

$$l(\mathbf{m}_j) = \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Đạo hàm riêng theo $\mathbf{m}_j$:

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2\sum_{i=1}^{N} y_{ij}(\mathbf{m}_j - \mathbf{x}_i) = 0$$

Giải phương trình:

$$\sum_{i=1}^{N} y_{ij}\mathbf{m}_j = \sum_{i=1}^{N} y_{ij}\mathbf{x}_i$$

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

Cách tính: tử số là tổng vector $\mathbf{x}_i$ của các điểm trong cụm j (khi $y_{ij} = 1$), mẫu số là số điểm trong cụm j. Nói cách khác, $\mathbf{m}_j$ là **trung bình cộng** của các điểm trong cụm j, giải thích tên K-means (K trung bình).

4.1.3 Các bước trong thuật toán K-means

Quá trình thực hiện thuật toán K-means được chia thành các bước cụ thể như sau, với mục tiêu lặp lại để gán các điểm vào cụm và cập nhật trung tâm cho đến khi hội tụ.

Quy trình của thuật toán:

1. Khởi tạo các tham số:
   - Người dùng chọn số lượng cụm K.
   - Khởi tạo ngẫu nhiên K trung tâm cụm $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$. Có thể dùng K-means++ để cải thiện:
     - Chọn ngẫu nhiên một điểm làm trung tâm đầu tiên.
     - Tính khoảng cách $D(\mathbf{x}_i)$ từ mỗi điểm $\mathbf{x}_i$ đến trung tâm gần nhất đã chọn.
     - Chọn trung tâm tiếp theo với xác suất tỷ lệ với $D(\mathbf{x}_i)^2$.
     - Lặp lại cho đến khi chọn đủ K trung tâm.
2. Gán các điểm vào cụm:
   - Với mỗi điểm dữ liệu $\mathbf{x}_i$:
     - Tính khoảng cách từ $\mathbf{x}_i$ đến tất cả các trung tâm $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$.
     - Gán $\mathbf{x}_i$ vào cụm có trung tâm gần nhất:

$$y_{ij} = \begin{cases} 1 & \text{nếu } j = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \\ 0 & \text{ngược lại} \end{cases}$$

3. Cập nhật trung tâm cụm:

- Với mỗi cụm j (từ 1 đến K):
  - Tính lại trung tâm cụm $\mathbf{m}_j$ bằng trung bình cộng của các điểm thuộc cụm:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

  - Nếu không có điểm nào thuộc cụm j, giữ nguyên trung tâm hoặc khởi tạo lại ngẫu nhiên.

4. Lặp lại và kiểm tra hội tụ:
   - So sánh các trung tâm mới với trung tâm cũ.
   - Nếu trung tâm không thay đổi (hoặc thay đổi nhỏ hơn ngưỡng ϵ) hoặc đạt số vòng lặp tối đa, dừng thuật toán.
   - Nếu không, quay lại bước 2 để tiếp tục gán các điểm và cập nhật trung tâm.

4.1.4 Ưu và nhược điểm

Ưu điểm

- Đơn giản, dễ thực hiện.

- Hiệu quả với dữ liệu lớn và cụm hình cầu.

- Tốc độ nhanh, độ phức tạp tuyến tính theo N.

Nhược điểm

- Cần chọn trước K, khó xác định.

- Nhạy với nhiễu và điểm ngoại lai.

- Không hiệu quả với cụm không hình cầu hoặc kích thước khác nhau.

- Phụ thuộc vào khởi tạo trung tâm.

4.2 DBScan

4.2.1 Giới thiệu phương pháp

DBScan: Density-Based Spatial Clustering of Applications with Noise.

DBScan là một thuật toán cơ sở để phân nhóm dựa trên mật độ điểm, rất phổ biến trong học máy và khai phá dữ liệu.

Áp dụng tốt trong trường hợp:

- Phát hiện ra các cụm có hình dạng tự do (không chỉ hình cầu/ elipse như K-Means/GMM).
- Xử lý được với điểm dữ liệu ngoại lệ/nhiều (outliers).

4.2.2 Các khái niệm quan trọng

[1]

Vùng lân cận epsilon (Eps-neighborhood) của một điểm dữ liệu P được định nghĩa là tập hợp tất cả các điểm dữ liệu nằm trong phạm vi bán kính epsilon (kí hiệu ϵ) xung quanh điểm P. Kí hiệu tập hợp những điểm này là:

$$N_{eps}(P) = \{Q \in D : d(P, Q) \leq \epsilon\}$$

Trong đó D là tập hợp tất cả các điểm dữ liệu của tập huấn luyện.

Khả năng tiếp cận trực tiếp mật độ (*directly density-reachable*) đề cập tới việc một điểm có thể tiếp cận trực tiếp tới một điểm dữ liệu khác. Cụ thể là một điểm Q được coi là có thể tiếp cận trực tiếp bởi điểm P tương ứng với tham số *epsilon* và *minPts* nếu như nó thoả mãn hai điều kiện:

1. Q nằm trong vùng lân cận *epsilon* của $P : Q \in N_{eps}(P)$
2. Số lượng các điểm dữ liệu nằm trong vùng lân cận *epsilon* tối thiểu là *minPts*: $|N_{eps}(Q)| \geq minPts$

Khả năng tiếp cận mật độ (*density-reachable*) liên quan đến cách hình thành một chuỗi liên kết điểm trong cụm. Cụ thể là trong một tập hợp chuỗi điểm $\{P_i\}_{i=1}^{n} \subset D$ mà nếu như bất kì một điểm P_i nào cũng đều có thể tiếp cận trực tiếp mật độ bởi P_{i-1} theo tham số *epsilon* và *minPts* thì khi đó ta nói điểm $P = P_n$ có khả năng kết nối mật độ tới điểm $Q = P_1$.

Từ định nghĩa trên có thể suy ra hai điểm P_i và P_j bất kì thuộc chuỗi $\{P_i\}_{i=1}^{n}$ thoả mãn $i < j$ thì P_j đều có khả năng kết nối mật độ tới P_i. Hai điểm bất kì có khả năng kết nối mật độ với nhau thì sẽ thuộc cùng một cụm. Từ đó suy ra các điểm trong chuỗi $\{P_i\}_{i=1}^{n}$ đều được phân về cùng cụm. Khả năng tiếp cận mật độ thể hiện sự mở rộng phạm vi của một cụm dữ liệu dựa trên liên kết theo chuỗi. Xuất phát từ một điểm dữ liệu ta có thể tìm được các điểm có khả năng kết nối mật độ tới nó theo lan truyền chuỗi để xác định cụm.

Trong thuật toán DBScan sử dụng **hai tham số chính**:

- ϵ (epsilon): Bán kính lân cận để xét mật độ điểm.
- *minPts*: Là một ngưỡng số lượng điểm tối thiểu trong một vùng bán kính ϵ để vùng đó được coi là vùng đủ mật độ.

Hai tham số trên giúp xác định **ba loại điểm**:

- Điểm lõi (Core Point): Nếu có ít nhất MinPts điểm nằm trong lân cận bán kính ϵ (kể cả chính nó).
- Điểm biên (Border Point): Không đủ MinPts trong lân cận bán kính ϵ, nhưng thuộc vùng lân cận của một điểm lõi.
- Điểm nhiễu (Noise Point): không phải điểm lõi cũng không phải điểm biên.

4.2.3 Các bước trong thuật toán DBScan

Các bước của thuật toán DBScan khá đơn giản. Thuật toán sẽ thực hiện lan truyền để mở rộng dần phạm vi của cụm cho tới khi chạm tới những điểm biên thì thuật toán sẽ chuyển sang một cụm mới và lặp lại tiếp quá trình trên.

Quy trình của thuật toán:

1. Xác định các tham số:
   - Người dùng chọn giá trị cho ϵ và MinPts.
   - Đánh dấu tất cả các điểm là chưa thăm (visited = False).
   - Đặt chỉ số cụm hiện tại là C = 0.
2. Duyệt qua tất cả các điểm dữ liệu - Với mỗi điểm chưa được phân cụm:
   - Đánh dấu điểm đã được thăm (visited = True).
   - Tìm láng giềng của điểm này trong phạm vi lân cận V_ϵ bán kính ϵ của nó.
   - Nếu số lượng láng giềng $\geq$ MinPts, khởi tạo cụm mới (C = C + 1) và mở rộng cụm (Expand Cluster) từ điểm đó.
   - Nếu số lượng láng giềng $<$ MinPts, đánh dấu là nhiễu tạm thời (có thể sau đó được gán vào cụm khác nếu là điểm biên).
3. Mở rộng cụm (Expand Cluster):
   - Với mỗi điểm láng giếng trong lân cận V_ϵ của Bước 2:
   - Nếu điểm đó chưa được thăm:
     - Đánh dấu đã thăm.
     - Lấy lân cận U_ϵ của điểm mới thăm này.
     - Nếu lân cận U_ϵ của nó cũng $\geq$ MinPts, thêm (những) điểm đó vào danh sách lân cận cần xét (để tiếp tục mở rộng cụm).
   - Nếu điểm đó chưa thuộc cụm nào, gán nó vào cụm hiện tại.
4. Lặp lại cho các điểm khác:
   - Khi một cụm được hoàn thành (không còn điểm để xét mở rộng), tiếp tục lặp lại quy trình từ Bước 2 cho các điểm chưa thăm.
   - Quá trình dừng khi tất cả các điểm đã được thăm.

4.2.4 Ưu và nhược điểm

Ưu điểm:

- Tự động xác định số lượng cụm (không cần chỉ định trước như K-Means/GMM).
- Phát hiện tốt các cụm có hình dạng phức tạp.
- Phát hiện nhiễu.

Nhược điểm:

- Khó chọn tham số ϵ và MinPts phù hợp.
- Không tốt nếu dữ liệu có mật độ không đồng đều hoặc chồng lấn nhau.
- Tính toán khoảng cách cho tất cả các cặp điểm (nên chi phí có thể cao với dữ liệu lớn).

5 Mô hình KNN

5.1 Giới thiệu mô hình

Thuật toán K-Nearest Neighbors (KNN) là một thuật toán học có giám sát dùng để dự đoán nhãn của một điểm dữ liệu mới dựa trên nhãn của k điểm láng giềng gần nhất trong tập huấn luyện. Thuật toán này thuộc nhóm phương pháp học lười (*lazy learning*), tức là không xây dựng mô hình trong giai đoạn huấn luyện mà chỉ thực hiện tính toán khi cần dự đoán. KNN được đánh giá cao vì tính dễ hiểu, dễ triển khai và không yêu cầu giả định về phân phối dữ liệu. Nó phụ thuộc vào khoảng cách giữa các điểm dữ liệu và thường sử dụng các thước đo như khoảng cách Euclid, Manhattan hoặc Minkowski.

5.2 Xây dựng mô hình[4]

5.2.1 Tập dữ liệu

Tập huấn luyện bao gồm n điểm dữ liệu có nhãn, được biểu diễn dưới dạng:

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Trong đó, mỗi $x_i \in R^p$ là một vector đặc trưng p chiều, và $y_i \in \{C_1, C_2, \ldots, C_m\}$ là nhãn lớp, với m là số lớp. Điểm dữ liệu mới cần phân loại là x.

5.2.2 Tính khoảng cách

Khoảng cách giữa điểm mới x và các điểm x_i trong tập huấn luyện được tính bằng các công thức phổ biến:

- **Khoảng cách Euclid**:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{p} (x_j - x_{ij})^2}$$

- **Khoảng cách Manhattan**:

$$d(x, x_i) = \sum_{j=1}^{p} |x_j - x_{ij}|$$

- **Khoảng cách Minkowski**:

$$d(x, x_i) = \left(\sum_{j=1}^{p} |x_j - x_{ij}|^q \right)^{1/q}$$

  với $q \geq 1$, trong đó $q = 2$ tương ứng với Euclid và $q = 1$ tương ứng với Manhattan.

5.2.3 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là bước cần thiết để đảm bảo các đặc trưng có đóng góp công bằng khi tính khoảng cách, tránh trường hợp đặc trưng có giá trị lớn chi phối kết quả.

5.2.4 Giá trị k

- Giá trị k ảnh hưởng lớn đến hiệu suất mô hình:
  - k nhỏ: Mô hình nhạy cảm với nhiễu, dễ bị *overfitting*.
  - k lớn: Mô hình tổng quát hơn nhưng có thể bỏ sót chi tiết, dẫn đến *underfitting*.
- Lựa chọn k tối ưu thường dựa trên phương pháp kiểm định chéo (*cross-validation*).

5.3 Quy tắc phân loại mô hình[4]

5.3.1 Chọn k láng giềng gần nhất

Sắp xếp các điểm huấn luyện theo khoảng cách tăng dần đến điểm mới x, sau đó lấy ra k điểm gần nhất để tạo thành tập $N_k(x)$.

5.3.2 Quy tắc phân loại

- **Đa số phiếu bầu**:
$$y = \arg\max_{C_j} \sum_{(x_i,y_i)\in N_k(x)} I(y_i = C_j)$$
- **Phiếu bầu có trọng số** (tùy chọn):
$$y = \arg\max_{C_j} \sum_{(x_i,y_i)\in N_k(x)} w_i \cdot I(y_i = C_j)$$
với trọng số $w_i = \frac{1}{d(x,x_i)}$ để ưu tiên các điểm gần hơn.

5.3.3 Công thức dự đoán

Xác suất một điểm thuộc lớp C_j được tính như sau:

$$P(C_j) = \frac{N_j}{k}$$

trong đó N_j là số láng giềng trong $N_k(x)$ thuộc lớp C_j. Nhãn dự đoán là:

$$\hat{y} = \arg\max_{C_j} P(C_j)$$

6 Mô hình Naive Bayes

6.1 Giới thiệu mô hình

Naive Bayes Classifier là một thuật toán phân loại đơn giản nhưng mạnh mẽ, đặc biệt phổ biến trong lĩnh vực học máy và xử lý ngôn ngữ tự nhiên. Mô hình này được xây dựng dựa trên Định lý Bayes—một công thức trong xác suất giúp tính toán khả năng xảy ra của một sự kiện khi đã biết thông tin từ trước. Với giả định rằng các đặc trưng (features) của dữ liệu độc lập với nhau khi biết lớp của dữ liệu, Naive Bayes trở thành một trong những thuật toán dễ hiểu và dễ triển khai nhất.

Ứng dụng thực tiễn của Naive Bayes rất đa dạng, từ phân loại thư rác (nhận diện và lọc bỏ email spam) đến phân tích cảm xúc (đánh giá cảm xúc trong các bình luận hoặc đánh giá trực tuyến). Khả năng xử lý nhanh chóng và hiệu quả trên cả dữ liệu nhỏ khiến nó trở thành lựa chọn phù hợp cho các tác vụ phân loại văn bản và các bài toán xử lý dữ liệu có chiều cao.

Thuật toán Naive Bayes là một phương pháp phân loại dựa trên Định lý Bayes với giả định rằng các đặc trưng (features) là độc lập với nhau. "Naive" nghĩa là "ngây thơ" vì nó giả định tính độc lập của các đặc trưng — giả định này không phải lúc nào cũng đúng, nhưng thường mang lại hiệu quả tốt trong nhiều ứng dụng thực tế.

6.2 Xây dựng Naive Bayes Classifier

[2]Trong bài toán phân loại, chúng ta có C phân lớp (class) $1, 2, ..., C$. Với một điểm dữ liệu $x \in R^d$ (tức là một vector đầu vào x có d chiều), mục tiêu là xác định xác suất để x thuộc về mỗi class $k \in 1, 2, \ldots, C$. Biểu thức này có dạng:

$$p(y = k \mid x) = p(k \mid x)$$

Sau khi tính được xác suất $p(k \mid x)$ cho từng lớp, chúng ta gán điểm dữ liệu x vào lớp có xác suất cao nhất, tức là:

$$y = c^* = \arg \max_{k \in \{1,...,C\}} p(k \mid x).$$

Biểu thức $p(k \mid x)$ thường rất khó tính toán trực tiếp. Vì vậy, quy tắc Bayes được áp dụng để chuyển biểu thức về dạng dễ tính hơn:

$$p(k \mid x) = \frac{p(x \mid k)p(k)}{p(x)}.$$

Do $p(x)$ không phụ thuộc vào k, ta có thể bỏ qua $p(x)$ và chỉ cần tối ưu hóa phần tử ở tử số:

$$y = c^* = \arg \max_{k \in \{1,...,C\}} p(x \mid k)p(k).$$

Trong đó:

- $p(k)$: Xác suất để một mẫu dữ liệu bất kỳ rơi vào lớp k. Thường được ước lượng bằng *Maximum Likelihood Estimation* (MLE), tức là tỷ lệ số lượng điểm dữ liệu trong lớp k so với tổng số điểm trong tập huấn luyện.
- $p(x \mid k)$: Phân phối xác suất của dữ liệu bên trong lớp k (xác suất để tồn tại một điểm dữ liệu x với điều kiện nó thuộc lớp k). Đây là một giá trị khó tính toán vì x có thể có nhiều chiều (nhiều đặc trưng).

Để đơn giản hóa việc tính toán, chúng ta giả định rằng các thành phần của x là độc lập với nhau nếu biết lớp k. Điều này dẫn đến biểu thức:

$$p(x \mid k) = p(x_1, x_2, \ldots, x_d \mid k) = \prod_{i=1}^{d} p(x_i \mid k).$$

Giả định này, dù đơn giản và không thực tế, nhưng mang lại kết quả tốt trong nhiều ứng dụng.

Trong quá trình huấn luyện:

- Ước lượng $p(k)$ từ tần suất xuất hiện của mỗi lớp trong tập huấn luyện.
- Xác định $p(x_i \mid k)$ cho từng đặc trưng x_i dựa vào loại dữ liệu (có thể là Gaussian, Multinomial, hoặc Bernoulli).

Trong quá trình kiểm tra, với một điểm dữ liệu mới x, lớp của nó được xác định qua:

$$y = c^* = \arg \max_{k \in \{1,\ldots,C\}} p(k) \prod_{i=1}^{d} p(x_i \mid k).$$

Để tránh sai số do tính toán với các xác suất nhỏ, ta thường dùng logarit của biểu thức trên:

$$y = c^* = \arg \max_{k \in \{1,\ldots,C\}} \left(\log p(k) + \sum_{i=1}^{d} \log p(x_i \mid k) \right).$$

Việc này không ảnh hưởng tới kết quả vì log là một hàm đồng biến trên tập các số dương.

6.3 Các phân phối thường dùng cho $p(x_i \mid k)$

- **Gausian Naive Bayes**

  Áp dụng cho dữ liệu liên tục, với giả định mỗi đặc trưng x_i trong lớp c có phân phối chuẩn. Xác suất có điều kiện $p(x_i \mid c)$ được tính như sau:

  $$p(x_i \mid c) = \frac{1}{\sqrt{2\pi\sigma_{c,i}^2}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2} \right),$$

  trong đó:

  - $\mu_{c,i}$: Giá trị trung bình của đặc trưng x_i trong lớp c.
  - $\sigma_{c,i}^2$: Độ lệch chuẩn của đặc trưng x_i trong lớp c.

- **Multinomial Naive Bayes** Phù hợp cho dữ liệu rời rạc, đặc biệt là khi đặc trưng là số lần xuất hiện (ví dụ: tần suất từ). Xác suất có điều kiện $p(x_i \mid c)$ được tính dựa trên tần suất từ:

$$p(x_i \mid c) = \frac{N_{c,i} + \alpha}{N_c + \alpha d},$$

  trong đó:

  - $N_{c,i}$: Số lần đặc trưng x_i xuất hiện trong lớp c.
  - N_c: Tổng số lần xuất hiện của tất cả các đặc trưng trong lớp c.
  - d: Số lượng đặc trưng (từ vựng).
  - α: Tham số điều chỉnh (*smoothing parameter*), thường dùng Laplace smoothing với $\alpha = 1$.

- **Bernoulli Naive Bayes** Áp dụng cho dữ liệu nhị phân (có hoặc không có từ nào đó). Mỗi đặc trưng x_i chỉ có hai trạng thái (0 hoặc 1). Xác suất có điều kiện $p(x_i \mid c)$ được tính như sau:

$$p(x_i \mid c) = p(x_i = 1 \mid c)x_i + \big(1 - p(x_i = 1 \mid c)\big)(1 - x_i),$$

  trong đó:

  - $p(x_i = 1 \mid c)$: Xác suất để đặc trưng x_i xuất hiện (tức là $x_i = 1$) trong các mẫu thuộc lớp c.
  - x_i: Giá trị của đặc trưng x_i trong một mẫu cụ thể, có thể là 1 (nếu đặc trưng xuất hiện) hoặc 0 (nếu đặc trưng không xuất hiện).

Trong bài toán nhận diện bảng chữ cái ngôn ngữ ký hiệu American Sign Language (ASL), dữ liệu được xử lý như sau:

- **Dữ liệu đầu vào**: Hình ảnh 2D của các ký hiệu ASL (từ A đến Z) được chuyển thành một vector các pixel.
- **Loại dữ liệu**: Giá trị pixel trong ảnh grayscale là liên tục (continuous) hoặc gần liên tục sau khi chuẩn hóa về khoảng $[0, 1]$.

Kết luận: Dữ liệu pixel trong cách tiếp cận này là liên tục, do đó cần một biến thể Naive Bayes phù hợp với dữ liệu liên tục, chẳng hạn như *Gaussian Naive Bayes*.

7 Mô hình Multinomial Logistic Regression

7.1 Giới thiệu mô hình

Softmax Regression, còn được gọi là **Multinomial Logistic Regression**, là một phương pháp phân loại tuyến tính phổ biến trong học máy, đặc biệt được sử dụng cho các bài toán *phân loại đa lớp* (multi-class classification). Khác với hồi quy logistic truyền thống vốn chỉ xử lý các bài toán nhị phân (2 lớp), Softmax Regression cho phép mở rộng lên nhiều lớp. Mỗi giá trị đầu ra thể hiện **xác suất** của dữ liệu thuộc về một lớp cụ thể, từ đó mô hình đưa ra quyết định phân loại cuối cùng bằng cách chọn lớp có xác suất cao nhất.

Mô hình này được xây dựng trên cơ sở các **hàm tuyến tính** và sử dụng **hàm softmax** để chuẩn hóa đầu ra thành phân phối xác suất. Hàm softmax không chỉ đảm bảo các xác suất là số dương và có tổng bằng 1, mà còn có tính chất đồng biến — tức giá trị đầu ra lớn hơn sẽ tương ứng với xác suất cao hơn.

Softmax Regression được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, tiêu biểu như: *phân loại chữ số viết tay* (ví dụ như bộ dữ liệu MNIST), *phân loại hình ảnh*, *nhận diện khuôn mặt*, và *phân tích văn bản* (như phân loại chủ đề hoặc cảm xúc trong câu).

Mặc dù là một mô hình tuyến tính, Softmax Regression có thể đạt hiệu quả cao trên các bài toán mà các lớp dữ liệu tách tuyến tính hoặc gần tuyến tính. Ngoài ra, nhờ vào cấu trúc đơn giản, nó có tốc độ huấn luyện nhanh và ít yêu cầu tài nguyên, vì vậy đây là lựa chọn lý tưởng trong các ứng dụng có giới hạn về thời gian hoặc thiết bị tính toán.

Với khả năng chuyển đổi đầu ra sang dạng xác suất và tích hợp tự nhiên với hàm mất mát cross-entropy, Softmax Regression không chỉ dễ triển khai mà còn có thể mở rộng linh hoạt trong các hệ thống học sâu, nơi nó thường được sử dụng như lớp đầu ra cuối cùng trong mạng nơ-ron phân loại.

7.2 Hàm softmax

Chúng ta cần một mô hình xác suất sao cho với mỗi input $\mathbf{x}$, a_i thể hiện xác suất để input đó rơi vào class i. Vậy điều kiện cần là các a_i phải dương và tổng của chúng bằng 1. Để có thể thỏa mãn điều kiện này, chúng ta cần *nhìn vào* mọi giá trị z_i và dựa trên quan hệ giữa các z_i này để tính toán giá trị của a_i.

Ngoài các điều kiện a_i lớn hơn 0 và có tổng bằng 1, chúng ta sẽ thêm một điều kiện nữa, đó là: giá trị $z_i = \mathbf{w}_i^\top \mathbf{x}$ càng lớn thì xác suất dữ liệu rơi vào class i càng cao. Điều kiện cuối này chỉ ra rằng chúng ta cần một hàm đồng biến đối với z_i.

Chú ý rằng z_i có thể nhận giá trị cả âm và dương. Một hàm số *mượt* đơn giản có thể chắc chắn biến z_i thành một giá trị dương, và hơn nữa, đồng biến, là hàm $\exp(z_i) = e^{z_i}$. Điều kiện *mượt* để thuận lợi hơn trong việc tính đạo hàm sau này. Điều kiện cuối cùng, tổng các a_i bằng 1 có thể được đảm bảo nếu:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

Hàm số này, tính trên tất cả các z_i, thỏa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1, giữ được *thứ tự* của các z_i. Hàm số này được gọi là **softmax function**. Chú ý rằng với cách định nghĩa này, không có xác suất a_i nào tuyệt đối bằng 0 hoặc tuyệt đối bằng 1, mặc dù chúng có thể rất gần 0 hoặc 1 khi rất nhỏ hoặc rất lớn so sánh với các z_j, $j \neq i$.

Lúc này, ta có thể giả sử rằng:

$$P(y_k = i \mid \mathbf{x}_k; \mathbf{W}) = a_i$$

Trong đó, $P(y = i \mid \mathbf{x}; \mathbf{W})$ được hiểu là xác suất để một điểm dữ liệu $\mathbf{x}$ rơi vào class thứ i nếu biết tham số mô hình (ma trận trọng số) là $\mathbf{W}$.

7.3 Phiên bản ổn định hơn của softmax function

Khi một trong các z_i quá lớn, việc tính toán $\exp(z_i)$ có thể gây ra hiện tượng tràn số (overflow), ảnh hưởng lớn tới kết quả của hàm softmax. Có một cách khắc phục hiện tượng này bằng cách dựa trên quan sát sau:

$$\frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp(-c)\exp(z_i)}{\exp(-c)\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp(z_i - c)}{\sum_{j=1}^{C} \exp(z_j - c)}$$

với c là một hằng số bất kỳ.

7.4 One hot coding

Với cách biểu diễn như trên, mỗi output sẽ không còn là một giá trị tương ứng với mỗi class nữa mà sẽ là một vector có đúng 1 phần tử bằng 1, các phần tử còn lại bằng 0. Phần tử bằng 1 nằm ở vị trí tương ứng với class đó, thể hiện rằng điểm dữ liệu đang xét rơi vào class này với xác suất bằng 1 (sự thật là như thế, không cần dự đoán).

Khi sử dụng mô hình Softmax Regression, với mỗi đầu vào $\mathbf{x}$, ta sẽ có đầu ra dự đoán là $\mathbf{a} = \text{softmax}(\mathbf{W}^T\mathbf{x})$. Trong khi đó, đầu ra thực sự chúng ta có là vector $\mathbf{y}$ được biểu diễn dưới dạng one-hot coding.

Hàm mất mát sẽ được xây dựng để tối thiểu sự khác nhau giữa đầu ra dự đoán $\mathbf{a}$ và đầu ra thực sự $\mathbf{y}$. Một lựa chọn đầu tiên ta có thể nghĩ tới là:

$$J(\mathbf{W}) = \sum_{i=1}^{N} ||\mathbf{a}_i - \mathbf{y}_i||_2^2$$

Tuy nhiên đây chưa phải là một lựa chọn tốt. Khi đánh giá sự khác nhau (hay khoảng cách) giữa hai phân bố xác suất (probability distributions), chúng ta có một đại lượng để đo đếm khác hiệu quả hơn. Đại lượng đó có tên là cross entropy.

7.5 Cross Entropy

Cross entropy giữa hai phân phối p và q được định nghĩa là:

$$H(p, q) = E_p[-\log q]$$

Với p và q là rời rạc (như $\mathbf{y}$ và $\mathbf{a}$ trong bài toán của chúng ta), công thức này được viết dưới dạng:

$$H(p, q) = -\sum_{i=1}^{C} p_i \log q_i \quad (1)$$

Chú ý: Hàm cross entropy không tính đối xứng $H(p, q) \neq H(q, p)$. Điều này có thể dễ dàng nhận ra ở việc các thành phần của p trong công thức (1) có thể nhận giá trị bằng 0, trong khi đó các thành phần của q phải là dương vì $\log 0$ không xác định. Chính vì vậy, khi sử dụng cross entropy trong các bài toán học giám sát, p thường là đầu ra thực sự vì đầu ra thực sự chỉ có 1 thành phần bằng 1, còn lại bằng 0 (one-hot), q thường là đầu ra dự đoán, khi mà không có xác suất nào tuyệt đối bằng 1 hoặc tuyệt đối bằng 0 cả.

Với Softmax Regression, trong trường hợp có C classes, loss giữa đầu ra dự đoán và đầu ra thực sự của một điểm dữ liệu $\mathbf{x}_i$ được tính bằng:

$$J(\mathbf{W}; \mathbf{x}_i, \mathbf{y}_i) = -\sum_{j=1}^{C} y_{ij} \log(a_i)_j$$

Với y_{ij} và $(a_i)_j$ lần lượt là phần tử thứ j của vector (xác suất) $\mathbf{y}_i$ và $\mathbf{a}_i$. Nhắc lại rằng đầu ra $\mathbf{a}_i$ phụ thuộc vào đầu vào $\mathbf{x}_i$ và ma trận trọng số $\mathbf{W}$.

7.6 Hàm mất mát cho Softmax Regression

Kết hợp tất cả các cặp dữ liệu $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N$, chúng ta sẽ có hàm mất mát cho Softmax Regression như sau:

$$J(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log(a_i)_j = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(\mathbf{w}_j^T \mathbf{x}_i)}{\sum_{k=1}^{C} \exp(\mathbf{w}_k^T \mathbf{x}_i)}\right)$$

Với ma trận trọng số $\mathbf{W}$ là biến cần tối ưu. Hàm mất mát này trông có vẻ đáng sợ, nhưng đừng sợ, đọc tiếp các bạn sẽ thấy đạo hàm của nó rất đẹp (và đáng yêu).

7.7 Tối ưu hàm mất mát

Một lần nữa, chúng ta lại sử dụng Stochastic Gradient Descent (SGD) ở đây.

Với mỗi cặp dữ liệu $(\mathbf{x}_i, \mathbf{y}_i)$, ta có:

$$
\begin{aligned}
&J_i(\mathbf{W})J(\mathbf{W};\mathbf{x}_i,\mathbf{y}_i) \\
&\quad = -\sum_{j=1}^{C} y_{ij} \log\left(\frac{\exp(\mathbf{w}_j^T\mathbf{x}_i)}{\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)}\right) \\
&\quad = -\sum_{j=1}^{C} y_{ij}\mathbf{w}_j^T\mathbf{x}_i + \sum_{j=1}^{C} y_{ij}\log\left(\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)\right) \quad (3)
\end{aligned}
$$

Trong biến đổi ở dòng cuối cùng, tôi đã sử dụng quan sát: $\sum_{j=1}^{C} y_{ij} = 1$ vì nó là tổng các xác suất.

Tiếp theo ta sử dụng công thức:

$$
\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = \left[\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_1}, \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_2}, ..., \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_C}\right] \quad (4)
$$

Trong đó, gradient theo từng được dựa theo dựa theo (3):

$$
\begin{aligned}
\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{w}_j} &= -y_{ij}\mathbf{x}_i + \frac{\exp(\mathbf{w}_j^T\mathbf{x}_i)}{\sum_{k=1}^{C}\exp(\mathbf{w}_k^T\mathbf{x}_i)}\mathbf{x}_i \\
&= -y_{ij}\mathbf{x}_i + (a_i)_j\mathbf{x}_i \\
&= ((a_i)_j - y_{ij})\mathbf{x}_i e_{ij}\mathbf{x}_i \quad (5)
\end{aligned}
$$

Giá trị $e_{ij} = (a_i)_j - y_{ij}$ có thể được coi là sai số dự đoán.

Đến đây ta đã có biểu thức thực sự rất đẹp. Kết hợp (4) và (5) ta có:

$$
\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}} = [e_{i1}\mathbf{x}_i, e_{i2}\mathbf{x}_i, ..., e_{iC}\mathbf{x}_i] = \mathbf{x}_i\mathbf{e}_i^T
$$

với $\mathbf{E} = \mathbf{A} - \mathbf{Y}$. Công thức tính gradient đơn giản này giúp cho cả Batch Gradient Descent, Stochastic Gradient Descent (SGD), và Mini-batch Gradient Descent đều có thể dễ dàng được áp dụng.

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số $\mathbf{W}$ sẽ là:

$$
\mathbf{W} = \mathbf{W} + \eta\mathbf{x}_i(\mathbf{y}_i - \mathbf{a}_i)^T
$$

8 Chuyển về bài toán hồi quy

8.1 Giới thiệu

Trong bài toán phân loại ảnh ngôn ngữ ký hiệu ASL, mục tiêu truyền thống là dự đoán một nhãn rời rạc tương ứng với chữ cái trong bảng chữ cái tiếng Anh. Mô hình Logistic Regression đa lớp (còn gọi là Softmax Regression) thường được sử dụng trong bối cảnh này, với đầu ra là một phân phối xác suất trên các lớp, thu được thông qua hàm softmax. Nhưng trong phần này chúng ta sẽ đưa bài toán phân loại ảnh ngôn ngữ ký hiệu ASL về dạng hồi quy với hai mô hình là hồi quy tuyến tính và cây quyết định.

8.2 Ý tưởng

Ta chọn ngẫu nhiên một lớp sau đó lấy kết quả của hàm softmax của các mẫu cho lớp đó(xác suất một mẫu rơi vào lớp đang xét). Ta lấy kết quả này làm nhãn cho mô hình hồi quy của chúng ta như hồi quy tuyến tính và cây quyết định.

8.3 Kết quả thực nghiệm

Mô hình	PCA	Train:Test	Accuracy	Precision	Recall
Linear Regression	no	8:2	0.9884908	0.9895288	0.7590361
		7:3	0.987005	0.9786477	0.7352941
		6:4	0.988398	0.9675	0.7755511
	yes	8:2	0.9927603	0.9962584	0.8433735
		7:3	0.9525743	0.9968354	0.8422464
		6:4	0.5924819	0.9552607	0.8416834
Decision Tree	no	8:2	0.9819397	0.8304348	0.7670678
		7:3	0.9836634	0.8666667	0.7647059
		6:4	0.9814636	0.8300221	0.7535070
	yes	8:2	0.9905328	0.8897638	0.9075630
		7:3	0.9899752	0.8927614	0.8930473
		6:4	0.9904703	0.8900000	0.8983957

Bảng 2: So sánh các mô hình Linear Regression và Decision Tree với và không sử dụng PCA

8.4 So sánh và đánh giá kết quả thực nghiệm

Bảng dưới trình bày kết quả đánh giá hiệu năng của hai mô hình: **Linear Regression** và **Decision Tree**, với ba tỷ lệ huấn luyện/kiểm tra khác nhau (8:2, 7:3, 6:4), và với hai trường hợp: có hoặc không sử dụng PCA. Các chỉ số được so sánh gồm **Accuracy**, **Precision** và **Recall**.

8.4.1 So sánh giữa Linear Regression và Decision Tree

Trong hầu hết các trường hợp, mô hình **Decision Tree** cho kết quả **Accuracy** và **Recall** cao hơn so với **Linear Regression**, đặc biệt khi có sử dụng PCA.

Linear Regression là mô hình tuyến tính, do đó khó nắm bắt được các đặc trưng phi tuyến tính phức tạp từ ảnh ASL. Ngược lại, **Decision Tree** có khả năng phân tách phi tuyến tốt hơn, nên hiệu quả hơn trong việc khai thác các đặc trưng phân biệt trong dữ liệu gốc hoặc dữ liệu đã giảm chiều bằng PCA.

8.4.2 Ảnh hưởng của PCA

- Với **Linear Regression**, việc **sử dụng PCA giúp tăng Recall rõ rệt** (từ khoảng 0.75 lên đến 0.84), nhưng đồng thời làm **giảm Accuracy** đáng kể trong một số trường hợp (ví dụ, ở tỷ lệ train:test là 6:4, Accuracy chỉ còn khoảng 0.59).

  Điều này là do PCA có thể loại bỏ nhiễu và giảm chiều dữ liệu, nhưng cũng có nguy cơ làm mất các thành phần mang thông tin quan trọng đối với mô hình tuyến tính. Tuy nhiên, PCA lại giúp phân cụm dữ liệu tốt hơn, từ đó tăng khả năng phát hiện đúng nhãn (Recall).

- Với **Decision Tree**, việc sử dụng PCA đem lại **cải thiện rõ ràng cả về Accuracy, Precision lẫn Recall** ở mọi tỷ lệ chia dữ liệu.

  Lý do là vì Decision Tree dễ bị overfitting khi làm việc với dữ liệu có chiều cao như ảnh. PCA giúp loại bỏ nhiễu và giảm số chiều đầu vào, khiến mô hình khái quát hóa tốt hơn, từ đó tăng độ chính xác và ổn định của kết quả.

8.4.3 Ảnh hưởng của tỷ lệ huấn luyện/kiểm tra

Khi tăng tỷ lệ kiểm tra (nghĩa là giảm số lượng mẫu huấn luyện) từ 8:2 đến 6:4, ta quan sát thấy:

- **Accuracy** có xu hướng giảm nhẹ trong hầu hết các trường hợp (trừ Decision Tree có PCA, kết quả gần như không thay đổi).
- **Recall** lại có thể tăng, phản ánh rằng mô hình có khả năng phát hiện đúng nhiều hơn trên tập kiểm tra đa dạng hơn.

Nguyên nhân là do việc giảm số lượng mẫu huấn luyện khiến mô hình có ít dữ liệu để học, dẫn đến hiệu năng có thể giảm. Tuy nhiên, nếu mô hình đủ đơn giản hoặc dữ liệu đầu vào đã được PCA xử lý tốt, mô hình vẫn có thể hoạt động ổn định.

8.4.4 Kết luận

- **Decision Tree với PCA** là lựa chọn tốt nhất cho bài toán phân loại chữ cái ASL trong thử nghiệm này.
- PCA tỏ ra hữu ích với các mô hình phi tuyến như Decision Tree, nhưng cần thận trọng khi sử dụng với mô hình tuyến tính như Linear Regression.
- Việc chọn tỷ lệ train:test ảnh hưởng đến kết quả, nhưng có thể được giảm nhẹ thông qua tiền xử lý phù hợp và lựa chọn mô hình thích hợp.

9 Thực nghiệm

9.1 Phân cụm

9.1.1 KMeans

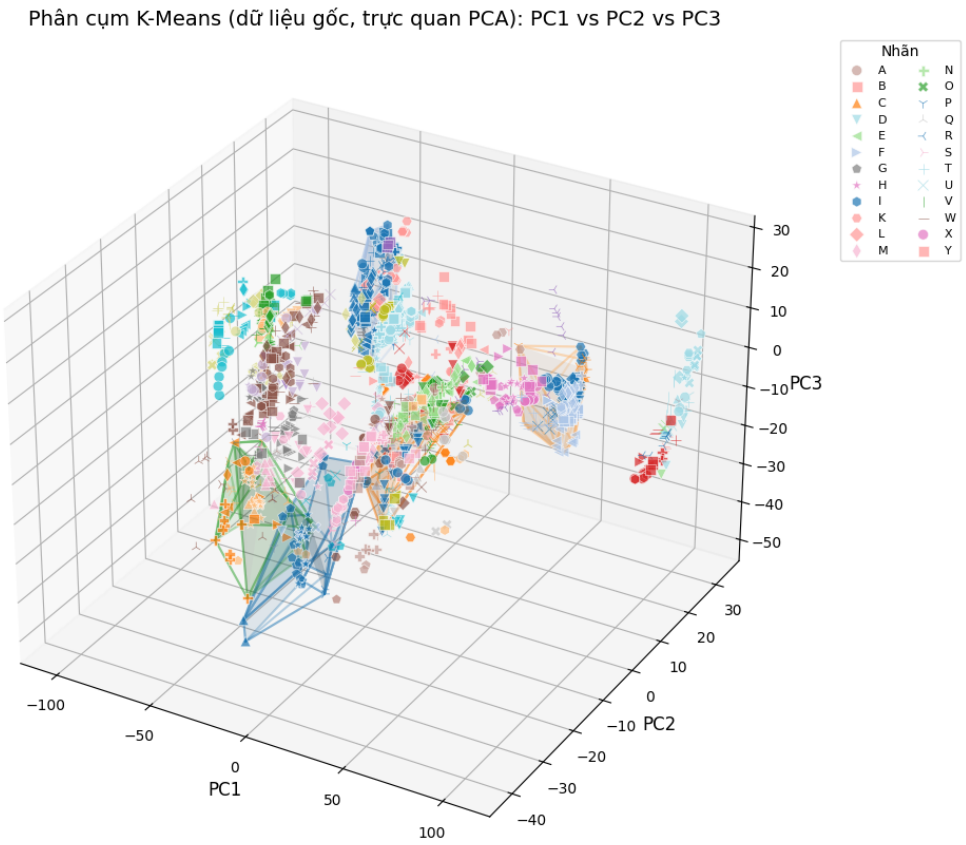
Thuật toán KMeans được áp dụng trên dữ liệu gốc (sau chuẩn hóa, chưa PCA) với số cụm $k = 24$ (bằng số lượng nhãn thực tế). Các bước thực nghiệm và kết quả thu được như sau:

- **Silhouette Score:** 0.1654
- **Adjusted Rand Index (ARI):** 0.0426
- **Normalized Mutual Information (NMI):** 0.2048

Nhận xét:

- **Silhouette Score = 0.1654** Giá trị này khá thấp (gần 0), cho thấy các điểm dữ liệu trong cùng một cụm không thực sự gần nhau, và các cụm bị chồng lấn đáng kể. Điều này phản ánh ranh giới giữa các cụm không rõ ràng, nhiều điểm nằm gần biên cụm hoặc bị phân loại sai cụm.
- **ARI = 0.0426** ARI gần bằng 0, nghĩa là mức độ tương đồng giữa nhãn phân cụm và nhãn thực tế là rất thấp. Kết quả này cho thấy KMeans gần như không tái hiện được cấu trúc phân lớp thực sự của dữ liệu, việc gán nhãn cụm gần như ngẫu nhiên so với nhãn gốc.
- **NMI = 0.2048** NMI cũng thấp, cho thấy lượng thông tin chung giữa nhãn cụm và nhãn thật là hạn chế. Điều này đồng nghĩa với việc các cụm được tạo ra không đại diện tốt cho các lớp thực tế trong dữ liệu.

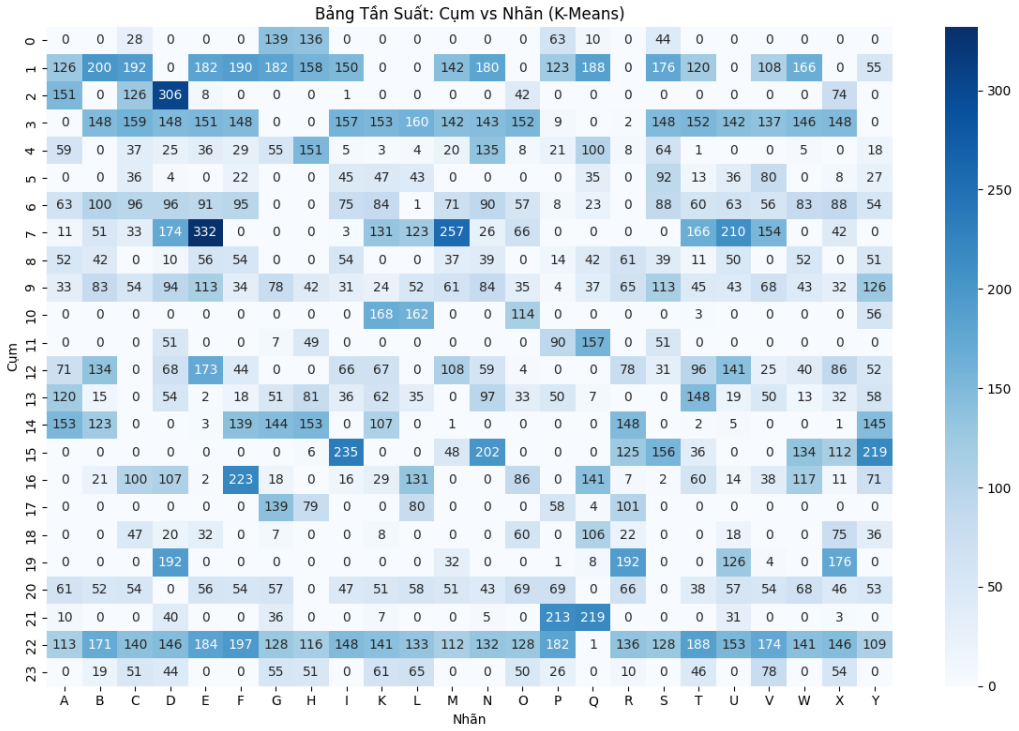
Tóm lại, tất cả các chỉ số đều cho thấy KMeans không phù hợp để phân cụm dữ liệu ký hiệu tay ASL trên không gian đặc trưng gốc. Các cụm bị pha trộn mạnh, không tách biệt rõ ràng, và không phản ánh đúng các nhãn thực tế. Điều này có thể do dữ liệu có cấu trúc phức tạp, các lớp bị chồng lấn hoặc không phù hợp với giả định phân vùng của KMeans.



Hình 4: Phân cụm KMeans trên dữ liệu gốc, trực quan hóa bằng PCA 3D (PC1, PC2, PC3). Mỗi điểm biểu diễn một ảnh, màu sắc theo cụm KMeans, hình dạng theo nhãn gốc. Đường bao lồi minh họa ranh giới một số cụm.

Nhận xét: Hình ảnh thể hiện kết quả phân cụm KMeans trên dữ liệu gốc, sau khi giảm chiều bằng PCA xuống 3 thành phần chính để trực quan hóa. Mỗi điểm biểu diễn một ảnh, màu sắc theo cụm KMeans, hình dạng marker theo nhãn gốc.

Quan sát cho thấy các cụm bị chồng lấn đáng kể, không tách biệt rõ ràng theo từng nhãn. Nhiều cụm chứa các điểm thuộc nhiều nhãn khác nhau, thể hiện sự pha trộn mạnh giữa các lớp. Một số cụm có hình dạng chặt chẽ, nhưng phần lớn các cụm phân tán rộng, ranh giới không rõ ràng. Điều này phù hợp với các chỉ số định lượng thấp (Silhouette Score, ARI, NMI), xác nhận rằng KMeans chưa thể phân tách hiệu quả các lớp ký hiệu tay trên không gian đặc trưng gốc.



Hình 5: Bảng tần suất giữa các cụm KMeans và các nhãn gốc. Hàng biểu diễn các cụm KMeans, cột biểu diễn các nhãn gốc. Mỗi ô cho biết số lượng điểm dữ liệu thuộc nhãn gốc tương ứng trong từng cụm.

Nhận xét: Bảng tần suất thể hiện rõ ràng mối quan hệ giữa các cụm Kmeans và các nhãn gốc. Ta có thể thấy:

- **Sự pha trộn nhãn trong từng cụm:** Nhiều cụm (ví dụ: cụm 0, 1, 2, 3, 4, 5, 6, 7...) chứa đồng thời nhiều nhãn khác nhau với số lượng lớn, thể hiện qua các ô màu đậm phân tán trên nhiều cột. Điều này cho thấy các cụm không thuần nhất về nhãn, mà bị pha trộn mạnh giữa các lớp.
- **Không có đường chéo nổi bật:** Nếu KMeans phân cụm tốt, bảng tần suất sẽ xuất hiện các ô giá trị lớn tập trung theo đường chéo (mỗi cụm tương ứng với một nhãn). Tuy nhiên, ở đây các ô giá trị lớn phân tán rộng, không tạo thành đường chéo rõ ràng, xác nhận sự pha trộn giữa các nhãn trong từng cụm.

Tổng thể,kết quả này cho thấy KMeans chưa thể tách biệt hiệu quả các lớp ký hiệu tay trên không gian đặc trưng gốc. Các cụm vừa bị pha trộn nhãn, vừa có sự xuất hiện của các cụm nhỏ, rời rạc, không đại diện cho cấu trúc phân lớp thực tế của dữ liệu.

9.1.2 DBScan

Thuật toán DBSCAN được áp dụng với các tham số:

- **Eps:** 38.44
- **MinPts:** 45

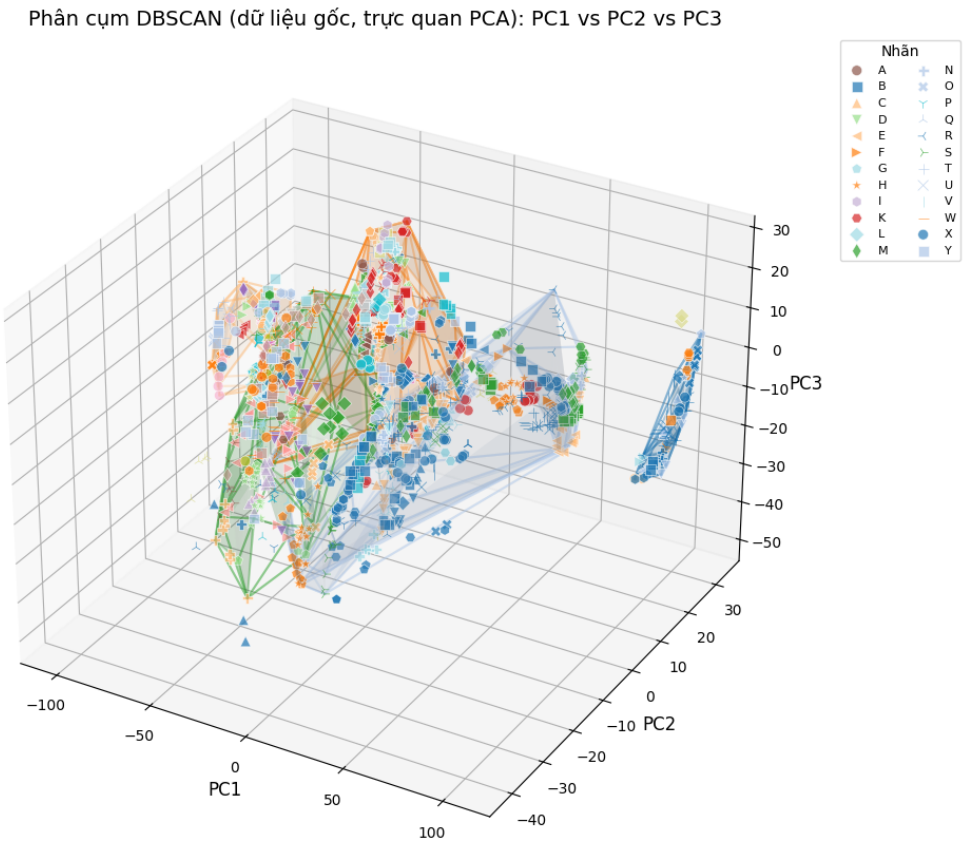
Kết quả thu được như sau:

- **Số cụm được phát hiện:** 55
- **Tỷ lệ nhiễu (noise):** 8.56%
- **Silhouette Score:** 0.1714
- **Adjusted Rand Index (ARI):** 0.0157
- **Normalized Mutual Information (NMI):** 0.2121

Nhận xét:

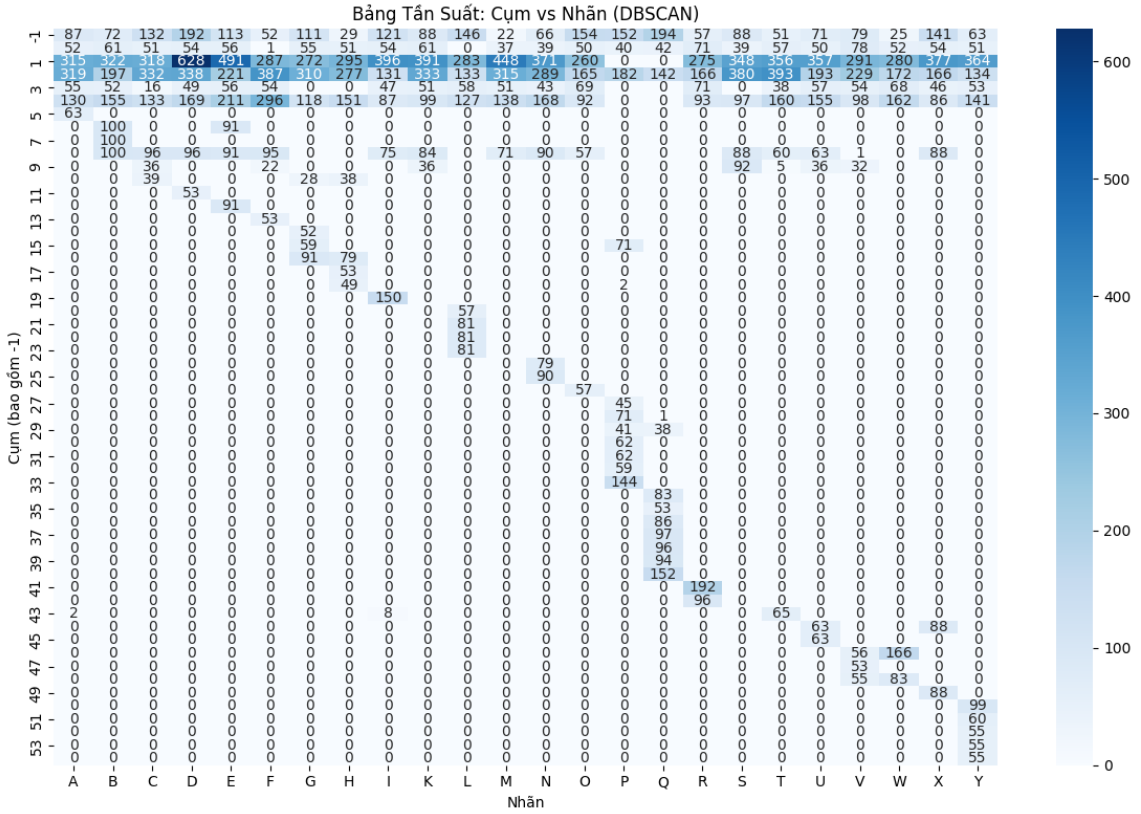
- Mặc dù DBSCAN phát hiện được 55 cụm, nhưng hầu hết các cụm có kích thước nhỏ và phân mảnh, thể hiện khả năng phân mảnh quá mức (over-segmentation) trên dữ liệu này.
- **Tỷ lệ nhiễu (8.56%)** là một con số tương đối hợp lý, cho thấy một phần nhỏ dữ liệu bị loại bỏ do không thuộc cụm nào.
- **Silhouette Score = 0.1714** là khá thấp, cho thấy độ phân cách giữa các cụm là yếu — điều này ngụ ý rằng nhiều điểm dữ liệu gần với biên cụm hoặc cụm bị chồng lấn.
- **ARI = 0.0157** gần bằng 0, phản ánh mức độ tương quan rất thấp giữa nhãn phân cụm và nhãn gốc — nghĩa là phân cụm gần như không tái hiện được nhãn thực.
- **NMI = 0.2121** cũng tương đối thấp, cho thấy thông tin chung giữa cụm và nhãn gốc là hạn chế.

Tổng thể, DBSCAN không cho kết quả phân cụm phù hợp với cấu trúc phân lớp của dữ liệu gốc trong trường hợp này. Nguyên nhân có thể đến từ việc dữ liệu có mật độ phân bố không đồng đều, cấu trúc phức tạp, và các lớp bị chồng lấn khiến DBSCAN khó phát hiện ranh giới rõ ràng.



Hình 6: Phân cụm DBSCAN trên dữ liệu gốc (trực quan bằng PCA 3D: PC1 vs PC2 vs PC3). Mỗi điểm dữ liệu được tô màu theo nhãn gốc để đánh giá chất lượng phân cụm. Đường bao lồi (Convex Hull) thể hiện hình dạng và ranh giới của từng cụm DBSCAN.

Nhận xét: Dữ liệu sau khi được phân cụm bằng DBSCAN và chiếu xuống không gian 3 chiều bằng PCA cho thấy sự hình thành các cụm có cấu trúc tương đối rõ ràng. Một số cụm (đặc biệt là cụm bên phải) có hình dạng chặt chẽ và không bị chồng lấn nhiều, trong khi các cụm khác có sự phân tán cao và trùng lặp về nhãn gốc, cho thấy DBSCAN chưa tách biệt hoàn toàn các lớp. Điều này có thể xuất phát từ việc các nhãn gốc không hoàn toàn phân chia tuyến tính hoặc các cụm có mật độ không đồng đều. Nhìn chung, DBSCAN đã phát hiện được cấu trúc tiềm ẩn trong dữ liệu mà không cần biết trước số lượng cụm.



Hình 7: Bảng tần suất giữa các cụm DBSCAN và các nhãn gốc. Hàng biểu diễn các cụm được tạo bởi thuật toán DBSCAN, cột biểu diễn các nhãn gốc. Mỗi ô cho biết số lượng điểm dữ liệu trong nhãn gốc tương ứng thuộc về cụm tương ứng.

Nhận xét: Bảng tần suất thể hiện rõ ràng mối quan hệ giữa các cụm DBSCAN và các nhãn gốc. Ta có thể thấy:

- Một số cụm lớn (như cụm 0, 1, 2, 3) bao gồm nhiều nhãn khác nhau, cho thấy sự chồng lấn giữa các lớp trong không gian dữ liệu — điều này cho thấy DBSCAN có xu hướng gộp các điểm có mật độ gần nhau dù khác nhãn.
- Một số cụm nhỏ hơn (như cụm 6, 7, 8, 9, ...) gần như thuần nhất, chỉ chứa một nhãn duy nhất với tần suất cao, điều này chứng tỏ DBSCAN phát hiện tốt các cụm rõ ràng và tách biệt về mật độ.
- Cụm '-1' (noise) chứa khá nhiều điểm từ các nhãn khác nhau, điều này cho thấy một phần dữ liệu không thuộc về bất kỳ cụm mật độ cao nào và có thể là outlier hoặc vùng biên.

Tổng thể, DBSCAN thể hiện hiệu quả tốt với các cụm có mật độ rõ ràng, tuy nhiên với những lớp bị chồng lấn hoặc phân bố không đồng đều, mô hình chưa thể phân tách hoàn toàn các nhãn một cách tối ưu.

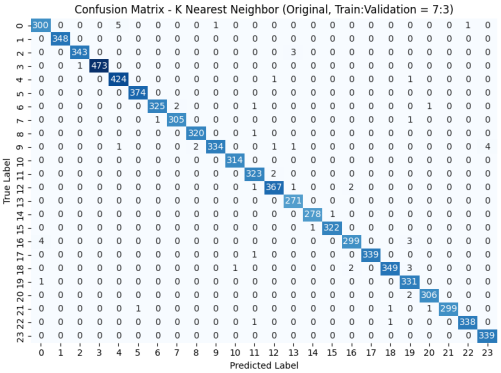
9.2 Kết quả các chỉ số của ba mô hình: Naive Bayes, Multinomial Logistic Regression, KNN

Ta sẽ so sánh độ chính xác các mô hình thông qua các chỉ số *accuracy*, *precision*, *recall*, *confusion matrix*.

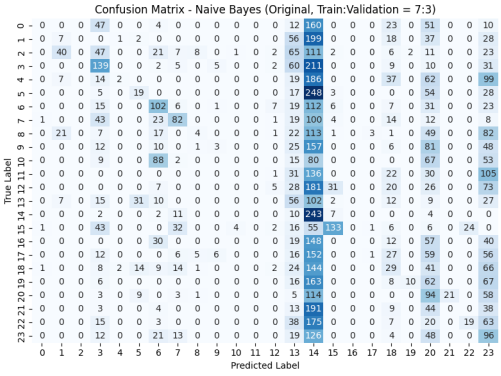
Bảng 3: So sánh hiệu suất mô hình - dữ liệu gốc

Mô hình	Train size	Train Acc	Val Acc	Precision	Recall	Time (s)
KNN	0.8	0.9967	0.9942	0.9943	0.9942	0.300
KNN	0.7	0.9963	0.9927	0.9927	0.9927	0.264
KNN	0.6	0.9947	0.9896	0.9897	0.9896	0.247
Naive Bayes	0.8	0.1385	0.1355	0.2604	0.1355	2.342
Naive Bayes	0.7	0.1314	0.1292	0.2164	0.1292	2.843
Naive Bayes	0.6	0.1266	0.1279	0.2753	0.1279	2.665
Logistic Reg.	0.8	0.9960	0.9792	0.9793	0.9792	57.480
Logistic Reg.	0.7	0.9969	0.9812	0.9812	0.9812	43.693
Logistic Reg.	0.6	0.9981	0.9790	0.9791	0.9790	35.765

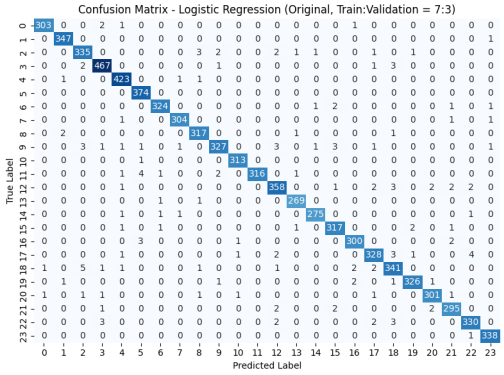
Hình 8: Ma trận nhầm lẫn của các mô hình - dữ liệu gốc - tỷ lệ 7:3



(a) KNN



(b) Naive Bayes

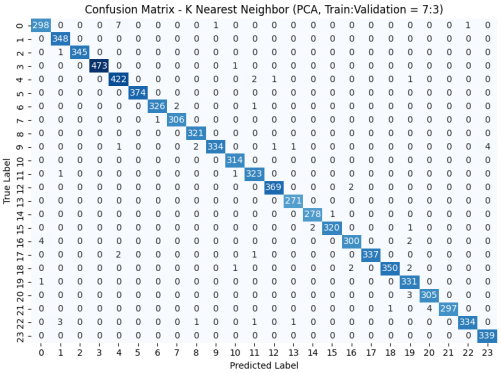


(c) Logistic Regression

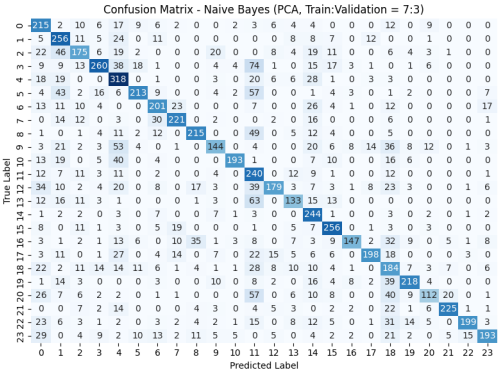
Bảng 4: So sánh hiệu suất mô hình - dữ liệu giảm chiều

Mô hình	**Train size**	**Train Acc**	**Val Acc**	**Precision**	**Recall**	**Time (s)**
KNN	0.8	0.9963	0.9935	0.9936	0.9935	0.050
KNN	0.7	0.9961	0.9920	0.9920	0.9920	0.008
KNN	0.6	0.9949	0.9893	0.9894	0.9893	0.008
Naive Bayes	0.8	0.6250	0.6143	0.6707	0.6143	0.069
Naive Bayes	0.7	0.6200	0.6113	0.6675	0.6113	0.048
Naive Bayes	0.6	0.6234	0.6065	0.6640	0.6065	0.042
Logistic Reg.	0.8	0.8051	0.7763	0.7771	0.7763	1.685
Logistic Reg.	0.7	0.8030	0.7865	0.7871	0.7865	1.467
Logistic Reg.	0.6	0.8023	0.7853	0.7858	0.7853	1.273

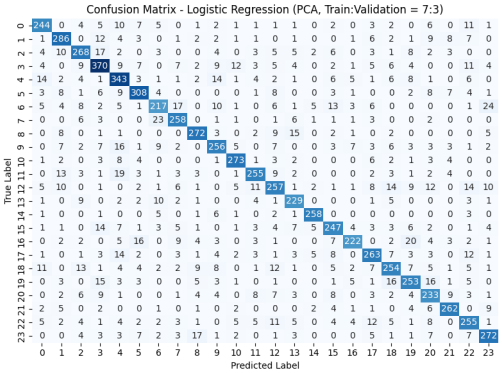
Hình 9: Ma trận nhầm lẫn của các mô hình - dữ liệu giảm chiều - tỷ lệ 7:3



(a) KNN



(b) Naive Bayes



(c) Logistic Regression

Nhận xét:

1. Hiệu suất tổng thể của các mô hình trên dữ liệu gốc

   - **KNN**: Đạt độ chính xác (**Accuracy**), **Precision** và **Recall** rất cao ở cả tập huấn luyện và tập kiểm tra, dao động khoảng **99.4%**. Thời gian huấn luyện khá ngắn ($\sim 0.2 - 0.3$s).
   - **Naive Bayes**: Có hiệu suất rất thấp trên dữ liệu gốc, chỉ đạt khoảng **13-14%** độ chính xác, cho thấy mô hình không phù hợp với dữ liệu gốc (có thể do dữ liệu có nhiều chiều và phân phối không phù hợp với giả định của Naive Bayes). Thời gian huấn luyện lại cao nhất trong ba mô hình ($\sim 2.3 - 2.8$s).
   - **Logistic Regression**: Có độ chính xác cao (gần **98-98.1%**) nhưng thời gian huấn luyện rất lớn, từ **35 đến 57 giây**, phản ánh độ phức tạp tính toán của mô hình với dữ liệu nhiều chiều.

2. Hiệu suất sau khi giảm chiều bằng PCA

   - **KNN**: Vẫn duy trì độ chính xác rất cao ($\sim 99.3\%$) với thời gian huấn luyện giảm mạnh, chỉ còn $\sim 0.008 - 0.05$s. Điều này cho thấy **KNN** hưởng lợi rõ rệt từ **PCA** cả về hiệu suất lẫn tốc độ.
   - **Naive Bayes**: Cải thiện đáng kể hiệu suất ($\sim 61 - 62\%$ accuracy) và cũng tăng **Precision** (lên đến **67%**) so với khi dùng dữ liệu gốc. Thời gian huấn luyện cũng giảm đáng kể (< 0.07s). **PCA** giúp dữ liệu phù hợp hơn với giả định phân phối của **NB**.
   - **Logistic Regression**: Bị giảm mạnh độ chính xác còn khoảng **77-78%**, nhưng bù lại thời gian huấn luyện giảm cực mạnh (xuống còn **1.2-1.6**s). Điều này phản ánh sự đánh đổi giữa độ chính xác và thời gian khi sử dụng **PCA**.

3. Kết luận

   - **KNN** là mô hình ổn định nhất cả trước và sau **PCA**, với độ chính xác rất cao và thời gian huấn luyện nhanh sau **PCA**.
   - **Naive Bayes** hoạt động rất kém trên dữ liệu gốc, nhưng cải thiện rõ rệt sau khi giảm chiều bằng **PCA**.
   - **Logistic Regression** hiệu suất cao với dữ liệu gốc nhưng bị giảm mạnh sau **PCA**; tuy nhiên, việc giảm chiều giúp tiết kiệm rất nhiều tài nguyên tính toán.

10 Kết luận

Nhìn chung, trong bài toán nhận diện ngôn ngữ ký hiệu, mô hình **KNN** nổi bật là lựa chọn tối ưu nhất nhờ vào độ chính xác vượt trội, khả năng dự đoán cân bằng và thời gian huấn luyện nhanh chóng, đáp ứng tốt yêu cầu của bài toán. So sánh với **Multinomial Logistic Regression**, mô hình này đạt hiệu suất khá, phù hợp cho các tình huống cần cân bằng giữa độ chính xác và độ phức tạp tính toán, nhưng vẫn không thể sánh bằng KNN về hiệu quả tổng thể. Trong khi đó, **Gaussian Naive Bayes** thể hiện hiệu suất kém nhất, cho thấy không phù hợp với đặc điểm của dữ liệu hình ảnh đã qua tiền xử lý trong bài toán này.

Để nâng cao hiệu quả của các mô hình, một số cải tiến có thể được áp dụng:

- Đối với KNN, nên tập trung vào tối ưu hóa bằng cách sử dụng các kỹ thuật giảm chiều dữ liệu tiên tiến như t-SNE hoặc UMAP, đồng thời áp dụng các cấu trúc dữ liệu như KD-Tree để tăng tốc độ tìm kiếm láng giềng, đặc biệt khi xử lý dữ liệu lớn.
- Với Multinomial Logistic Regression, việc tinh chỉnh siêu tham số thông qua Grid Search và bổ sung các đặc trưng bổ trợ (như đặc trưng hình học) có thể cải thiện đáng kể hiệu suất.
- Đối với Gaussian Naive Bayes, thử nghiệm các biến thể khác như Multinomial Naive Bayes hoặc cải thiện quy trình tiền xử lý dữ liệu, chẳng hạn như cân bằng lớp, sẽ giúp tăng hiệu quả.
- Các cải tiến chung như sử dụng kỹ thuật tăng cường dữ liệu (*data augmentation*), thử nghiệm các mô hình học sâu như Convolutional Neural Networks (CNN) nếu có đủ tài nguyên, và kiểm tra hiệu suất trên dữ liệu thực tế (như video ký hiệu) sẽ đảm bảo hệ thống hoạt động ổn định và hiệu quả hơn trong các ứng dụng thực tiễn, từ đó nâng cao chất lượng nhận diện ngôn ngữ ký hiệu.

Tài liệu

[1] 15.1. phương pháp phân cụm dựa trên mật độ (density-based clustering) — phamdinhkhanh.github.io.

[2] Tiep Vu. Bài 32: Naive bayes classifier — machinelearningcoban.com.

[3] Tiep Vu. Bài 4: K-means clustering — machinelearningcoban.com.

[4] Tiep Vu. Bài 6: K-nearest neighbors — machinelearningcoban.com.