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

Báo cáo đề tài

Sử dụng mô hình học máy để phân loại giai đoạn xơ gan

Nhóm 12:
Đào Mạnh Đức - 22000085
Tạ Đăng Đức - 22000088
Nguyễn Huy Hoàng - 22000095

Giảng viên hướng dẫn: TS. Cao Văn Chung

Hà Nội, tháng 4 năm 2025

Mục lục

1 Giới thiệu

1.1 Lý do chọn đề tài

Xơ gan là hậu quả của tổn thương gan kéo dài, dẫn đến sự hình thành mô sẹo nghiêm trọng và làm suy giảm chức năng gan. Các nguyên nhân phổ biến dẫn đến xơ gan bao gồm viêm gan mạn tính (đặc biệt là viêm gan B và C), lạm dụng rượu trong thời gian dài, và một số bệnh lý tự miễn như viêm đường mật nguyên phát (Primary Biliary Cirrhosis - PBC). Việc phát hiện và phân loại chính xác các giai đoạn xơ gan là yếu tố then chốt trong việc đưa ra phác đồ điều trị phù hợp, từ đó cải thiện tiên lượng cho người bệnh.

Trong thực hành lâm sàng, việc đánh giá mức độ tiến triển của xơ gan thường dựa trên các chỉ số lâm sàng và xét nghiệm sinh hóa. Tuy nhiên, quá trình này có thể chịu ảnh hưởng bởi nhiều yếu tố chủ quan hoặc thiếu dữ liệu. Do đó, việc khai thác dữ liệu y tế sẵn có để tìm hiểu mối quan hệ giữa các đặc điểm lâm sàng và giai đoạn bệnh là một hướng tiếp cận có tiềm năng cao trong hỗ trợ chẩn đoán và ra quyết định điều trị.

Tập dữ liệu được sử dụng trong đề tài này là kết quả của một nghiên cứu lâm sàng do Mayo Clinic thực hiện trong giai đoạn 1974–1984, với đối tượng là các bệnh nhân mắc bệnh viêm đường mật nguyên phát (Primary Biliary Cirrhosis - PBC). Dữ liệu đã được xử lý và mở rộng nhằm tăng số lượng mẫu, đồng thời giữ lại các đặc trưng lâm sàng quan trọng phục vụ phân tích.

Thông qua việc khai thác và phân tích tập dữ liệu đã nói trên, nhóm chúng em kỳ vọng làm rõ hơn mối liên hệ giữa các chỉ số y học và mức độ tiến triển của bệnh xơ gan, từ đó tạo nền tảng cho các ứng dụng phân tích dữ liệu trong lĩnh vực y tế.

1.2 Mục tiêu

Mục tiêu chính của đề tài này là:

- Xây dựng và đánh giá các mô hình học máy nhằm phân loại chính xác các giai đoạn tiến triển của bệnh xơ gan, dựa trên các đặc trưng lâm sàng và chỉ số xét nghiệm từ dữ liệu bệnh nhân.
- Thực hiện các bước tiền xử lý dữ liệu bao gồm xử lý outliers, chuẩn hóa, và biến đổi dữ liệu phù hợp với từng loại mô hình.

- Đánh giá và so sánh hiệu quả của các mô hình học máy khác nhau thông qua các chỉ số đánh giá.
- Phân tích kết quả thu được để rút ra nhận xét về mức độ phù hợp của từng mô hình trong bài toán chẩn đoán y tế, từ đó đề xuất mô hình có tiềm năng ứng dụng thực tiễn.

1.3 Phạm vi

- **Dữ liệu:** Đề tài sử dụng bộ dữ liệu liên quan đến bệnh nhân mắc viêm đường mật nguyên phát (Primary Biliary Cirrhosis - PBC), được công bố bởi Mayo Clinic.
- **Phương pháp:** Đề tài sử dụng các phương pháp học máy cơ bản để thực hiện phân cụm và phân loại.
- **Giới hạn:**
  - Không sử dụng các mô hình học sâu (Deep Learning) do hạn chế về tài nguyên và mục tiêu đơn giản hóa mô hình.
  - Không thực hiện xử lý ảnh y khoa hay dữ liệu thời gian thực.
  - Việc đánh giá mô hình chỉ dựa trên tập dữ liệu sẵn có, không triển khai thực nghiệm trên hệ thống thực tế.

1.4 Cấu trúc báo cáo

Báo cáo gồm 7 phần chính. Phần đầu trình bày lý do chọn đề tài, mục tiêu và phạm vi nghiên cứu. Tiếp theo, phần 2 và 3 giới thiệu tập dữ liệu, các bước tiền xử lý và phân tích sơ bộ thông qua trực quan hóa. Phần 4 trình bày tổng quan về các phương pháp học máy được sử dụng trong đề tài. Phần 5 mô tả quá trình xây dựng và đánh giá mô hình, bao gồm ba hướng tiếp cận chính: phân cụm, phân loại và hồi quy. Cuối cùng, phần 6 đưa ra kết luận và định hướng phát triển trong tương lai.

2 Tiền xử lý dữ liệu

2.1 Mô tả dữ liệu

Bộ dữ liệu được sử dụng trong đề tài là dữ liệu lâm sàng của 418 bệnh nhân mắc viêm đường mật nguyên phát (Primary Biliary Cirrhosis - PBC), được thu thập bởi Mayo Clinic trong giai đoạn 1974–1984. Sau quá trình mở rộng dữ liệu, tập dữ liệu bao gồm 25,000 bản ghi với 19 thuộc tính — bao gồm cả biến định lượng (số học) và định tính (phân loại).

Biến mục tiêu trong bài toán là `Stage`, biểu thị giai đoạn mô học của bệnh với 3 mức: Giai đoạn 1, 2 hoặc 3.

Các đặc trưng trong tập dữ liệu có thể chia thành hai nhóm chính: (1) Thông tin bệnh nhân và biểu hiện lâm sàng; (2) Các chỉ số sinh hóa.

STT	**Tên trường**	**Kiểu dữ liệu**	**Mô tả**
1	N_Days	Numeric	Số ngày từ lúc đăng ký đến khi tử vong, ghép gan hoặc kết thúc nghiên cứu năm 1986
2	Status	Nominal	Tình trạng bệnh nhân: C (còn sống), CL (ghép gan), D (tử vong)
3	Drug	Nominal	Loại thuốc điều trị: D-penicillamine hoặc placebo
4	Age	Numeric	Tuổi bệnh nhân (tính theo ngày)
5	Sex	Nominal	Giới tính: M (nam), F (nữ)
6	Ascites	Nominal	Có báng bụng: N (Không), Y (Có)
7	Hepatomegaly	Nominal	Có gan to: N (Không), Y (Có)
8	Spiders	Nominal	Có giãn mao mạch hình nhện: N (Không), Y (Có)
9	Edema	Nominal	Phù: N (không phù), S (phù nhẹ), Y (phù không đáp ứng thuốc)

Bảng 1: Mô tả các biến về thông tin bệnh nhân và biểu hiện lâm sàng

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	Bilirubin	Numeric	Nồng độ bilirubin huyết thanh [mg/dl]
2	Cholesterol	Numeric	Nồng độ cholesterol huyết thanh [mg/dl]
3	Albumin	Numeric	Nồng độ albumin [gm/dl]
4	Copper	Numeric	Nồng độ đồng trong nước tiểu [μg/ngày]
5	Alk_Phos	Numeric	Alkaline phosphatase [U/lít]
6	SGOT	Numeric	SGOT [U/ml]
7	Tryglicerides	Numeric	Nồng độ tryglicerides [mg/dl]
8	Platelets	Numeric	Số lượng tiểu cầu [nghìn tế bào/ml]
9	Prothrombin	Numeric	Thời gian prothrombin [giây]
10	Stage	Ordinal	Giai đoạn mô học của bệnh (1, 2 hoặc 3)

Bảng 2: Mô tả các biến sinh hóa và giai đoạn bệnh

2.2 Xử lý dữ liệu

2.2.1 Đọc dữ liệu

	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	2221	C	Placebo	18499	F	N	Y	N	N	0.5	149.000000	4.04	227.0	598.0	52.70	57.000000	256.0	9.9	1
1	1230	C	Placebo	19724	M	Y	N	Y	N	0.5	219.000000	3.93	22.0	663.0	45.00	75.000000	220.0	10.8	2
2	4184	C	Placebo	11839	F	N	N	N	N	0.5	320.000000	3.54	51.0	1243.0	122.45	80.000000	225.0	10.0	2
3	2090	D	Placebo	16467	F	N	N	N	N	0.7	255.000000	3.74	23.0	1024.0	77.50	58.000000	151.0	10.2	2
4	2105	D	Placebo	21699	F	N	Y	N	N	1.9	486.000000	3.54	74.0	1052.0	108.50	109.000000	151.0	11.5	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
24995	3584	D	D-penicillamine	23612	F	N	N	N	N	0.8	231.000000	3.87	173.0	9009.8	127.71	96.000000	295.0	11.0	2
24996	3584	D	D-penicillamine	23612	F	N	N	N	N	0.8	231.000000	3.87	173.0	9009.8	127.71	96.000000	295.0	11.0	2
24997	971	D	D-penicillamine	16736	F	N	Y	Y	Y	5.1	369.510563	3.23	18.0	790.0	179.80	124.702128	104.0	13.0	3
24998	3707	C	D-penicillamine	16990	F	N	Y	N	N	0.8	315.000000	4.24	13.0	1637.0	170.50	70.000000	426.0	10.9	2
24999	3707	C	D-penicillamine	16990	F	N	Y	N	N	0.8	315.000000	4.24	13.0	1637.0	170.50	70.000000	426.0	10.9	2

25000 rows × 19 columns

Hình 1: Mẫu dữ liệu

2.2.2 Data Cleaning

Kiểm tra giá trị thiếu

Mẫu dữ liệu không có giá trị null, nếu thiếu thì có thể loại bỏ, thay thế bằng trung bình, trung vị

Kiểm tra trùng lặp dữ liệu

Bộ dữ liệu gồm có 25,000 mẫu dữ liệu và sau khi kiểm tra thì có 15,361 mẫu bị trùng lặp. Sau khi thực hiện loại bỏ trùng lặp thì bộ dữ liệu còn 9,639 mẫu.

Chuyển đổi dữ liệu về dạng phù hợp

Trong quá trình tiền xử lý dữ liệu, một số biến định tính cần được chuyển đổi về dạng số để thuận tiện cho việc sử dụng trong các mô hình học máy. Cụ thể:

- Các biến nhị phân như "Ascites", "Hepatomegaly", và "Spiders"có giá trị ban đầu là "N"(No) và "Y"(Yes) được ánh xạ về dạng số: "0"và "1".
- Biến "Sex"được chuyển đổi từ "F"(Female) và "M"(Male) sang "0"và "1".
- Biến "Drug"với hai giá trị "Placebo"và "D-penicillamine"cũng được ánh xạ tương ứng sang "0"và "1".
- Các biến phân loại nhiều mức như "Edema"và "Status"được chuyển đổi bằng kỹ thuật One-hot Encoding, tạo ra các cột mới tương ứng với từng mức giá trị.
- Cuối cùng, toàn bộ các cột dạng "boolean"được chuyển đổi về kiểu số nguyên "int"để đảm bảo tính thống nhất trong kiểu dữ liệu.

Sau khi thực hiện ta có được kết quả như sau:

	N_Days	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Bilirubin	Cholesterol	Albumin	...	Tryglicerides	Platelets	Prothrombin	Stage	Edema_N	Edema_S	Edema_Y	Status_C	Status_CL	Status_D
0	2221	0	50.68	0	0	1	0	0.5	149.000000	4.04	...	57.000000	256.0	9.9	1	1	0	0	1	0	0
1	1230	0	54.04	1	1	0	1	0.5	219.000000	3.93	...	75.000000	220.0	10.8	2	1	0	0	1	0	0
2	4184	0	32.44	0	0	0	0	0.5	320.000000	3.54	...	80.000000	225.0	10.0	2	1	0	0	1	0	0
3	2090	0	45.12	0	0	0	0	0.7	255.000000	3.74	...	58.000000	151.0	10.2	2	1	0	0	0	0	1
4	2105	0	59.45	0	0	1	0	1.9	486.000000	3.54	...	109.000000	151.0	11.5	1	1	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
24963	3577	0	49.03	0	1	0	1	0.7	369.510563	3.49	...	124.702128	243.0	9.7	1	1	0	0	1	0	0
24971	4795	0	64.04	0	1	0	1	1.8	369.510563	3.24	...	124.702128	139.0	10.5	1	1	0	0	1	0	0
24972	3358	1	67.36	0	0	1	0	2.1	262.000000	3.48	...	84.000000	412.0	11.8	3	1	0	0	0	0	1
24991	4365	1	58.42	0	0	0	0	0.9	346.000000	3.40	...	90.000000	228.0	10.3	2	1	0	0	1	0	0
24992	694	1	78.49	1	1	1	1	0.8	300.000000	2.94	...	99.000000	97.0	11.2	3	1	0	0	0	0	1

9639 rows × 23 columns

Hình 2: Dữ liệu sau chuyển đổi

3 Phân tích và trực quan hóa dữ liệu

3.1 Phân tích tham số thống kê

	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
count	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000	9639.000000
mean	1910.982571	18429.717606	3.228571	371.706706	3.496118	97.027569	1973.572709	122.317487	123.587337	253.787605	10.713328	2.029152
std	1093.620373	3693.953156	4.512278	197.824339	0.382319	73.108854	1827.063380	47.653515	55.206301	95.740700	0.922026	0.809956
min	41.000000	9598.000000	0.300000	120.000000	1.960000	4.000000	289.000000	26.350000	33.000000	62.000000	9.000000	1.000000
25%	1103.000000	15628.000000	0.800000	271.000000	3.290000	51.000000	1031.000000	89.900000	93.000000	188.000000	10.000000	1.000000
50%	1690.000000	18628.000000	1.300000	369.510563	3.520000	97.648387	1713.000000	122.556346	124.702128	249.000000	10.600000	2.000000
75%	2598.000000	20819.000000	3.300000	369.510563	3.760000	102.000000	1982.655769	134.850000	125.000000	307.000000	11.100000	3.000000
max	4795.000000	28650.000000	28.000000	1775.000000	4.640000	588.000000	13862.400000	457.250000	598.000000	721.000000	18.000000	3.000000

Hình 3: Thống kê cơ bản

Từ bảng thống kê cơ bản ta có thể thấy một số bất thường như sau:

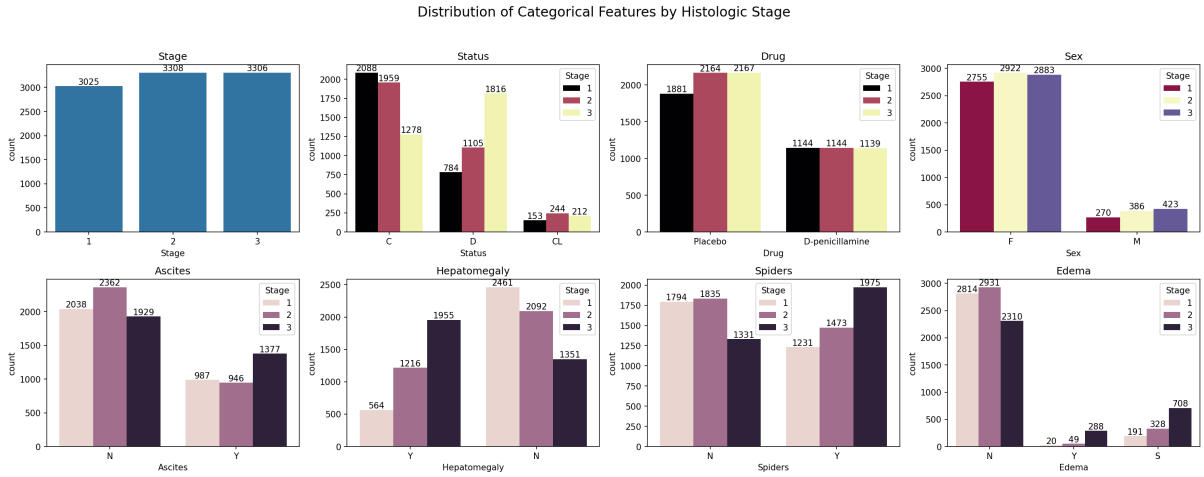
- Cột Cholesterol có max là quá lớn . Nó bất thường về mặt y tế và có thể là nhập liệu sai
- Cột Copper là nồng độ Đồng trong máu. Với giá trị tối thiểu 4 µg/dL là bất thường
- Cột Alk_Phos là Alkaline Phosphatase. có giá trị max bất thường do ngay cả trong các bệnh lý nghiêm trọng (như tắc mật hoặc ung thư gan), ALK hiếm khi vượt quá 2000–3000 U/L

3.2 Phân tích khám phá dữ liệu

Sau khi thực hiện phân tích, dựa vào hình 4 và 5 kết quả thu được như sau:
Với các biến phân loại

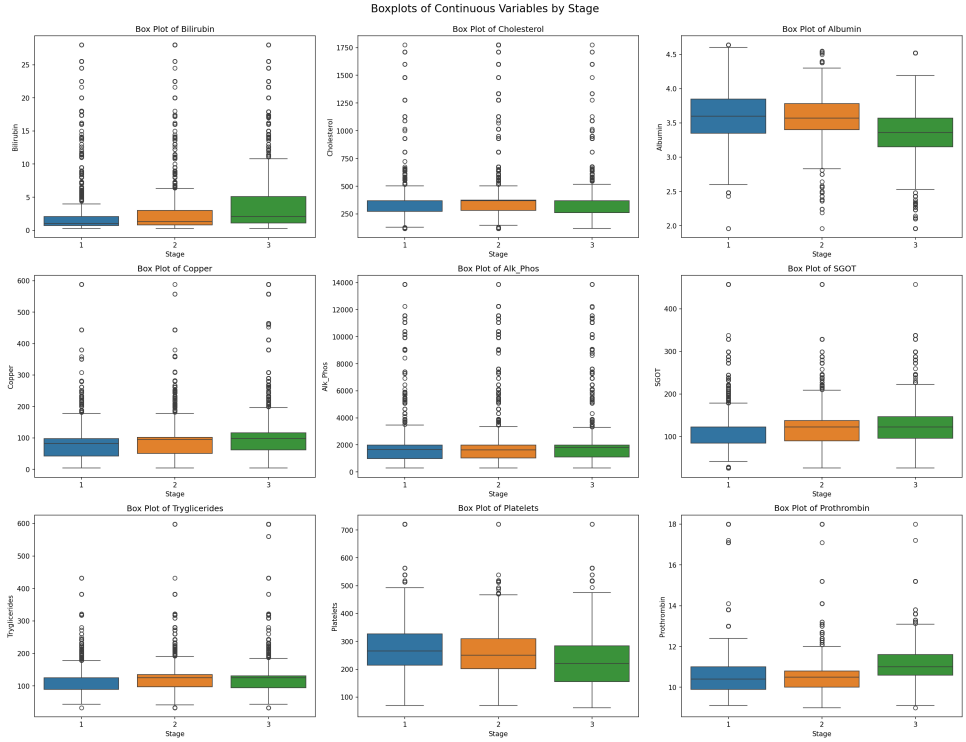
- Stage: Phân bố đồng đều giữa ba giai đoạn, tập dữ liệu không bị mất cân bằng nghiêm trọng.
- Status: Đa số ca tử vong thuộc giai đoạn 3, phản ánh mức độ nghiêm trọng.
- Drug: Số lượng bệnh nhân dùng giả dược (placebo) và D-penicillamine cho thấy phương pháp điều trị.
- Sex: Bệnh nhân nữ chiếm đa số.

- Ascites: Phần lớn bệnh nhân có cổ trướng thuộc giai đoạn 3, liên quan đến giai đoạn nặng.
- Hepatomegaly: Đa số bệnh nhân gan to ở giai đoạn 1, có thể là dấu hiệu sớm.
- Spiders: Hầu hết xuất hiện ở giai đoạn 3, là triệu chứng muộn.
- Edema: Đa số không bị phù, nhóm có phù tập trung ở giai đoạn nặng.



Hình 4: Quan hệ giữa các biến phân loại và biến mục tiêu

Với các biến liên tục

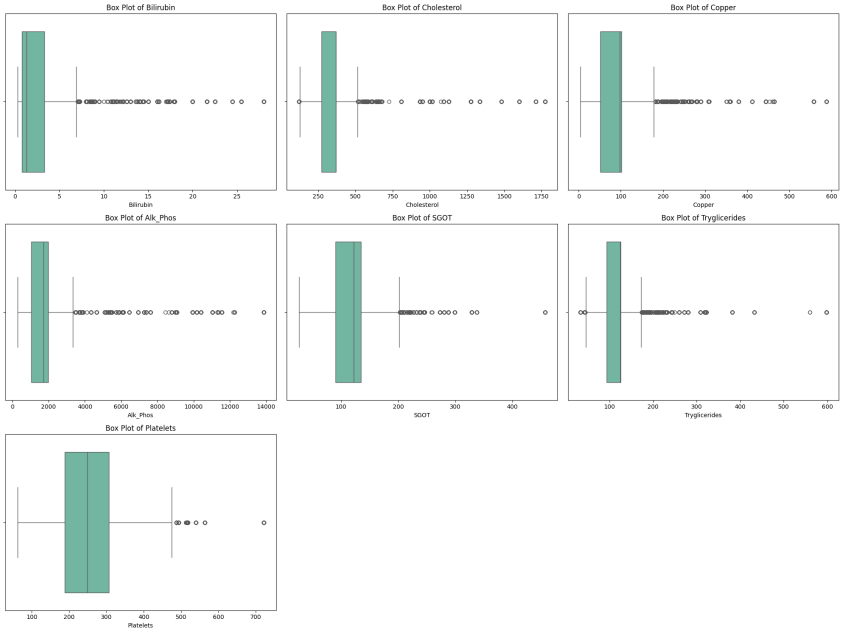


Hình 5: Quan hệ giữa các biến liên tục và biến mục tiêu

- Bilirubin: Giai đoạn 3 dao động 0–11 mg/dl, giai đoạn 2 dưới 5 mg/dl.
- Cholesterol: Tương tự giữa các giai đoạn, trong khoảng 100–500 mg/dl.
- Albumin: Giai đoạn 3 thấp hơn (2.5–4.25 gm/dl), giai đoạn 1 ít ngoại lai.
- Platelets: Dưới 500, giai đoạn 3 có xu hướng thấp hơn.
- Spiders: Hầu hết xuất hiện ở giai đoạn 3, là triệu chứng muộn.
- Prothrombin: Giai đoạn 3 cao hơn đáng kể, phản ánh đông máu nặng.

3.3 Phát hiện outliers

Sử dụng biểu đồ hộp (Boxplot) để thể hiện tóm tắt phân phối dữ liệu thông qua các thông số thống kê như giá trị tối thiểu, giá trị tối đa, trung vị (median), và các tứ phân vị (quartiles).



Hình 6: Biểu đồ Box plot

Qua biểu đồ boxplot của các thuộc tính liên tục, có thể rút ra một số nhận xét như sau:

- Bilirubin: Nhiều outliers từ 5 đến 25, cho thấy nồng độ bất thường ở một số bệnh nhân.
- Cholesterol: Outliers từ 750 đến 1750, phản ánh nhóm nhỏ bệnh nhân có mức cao.

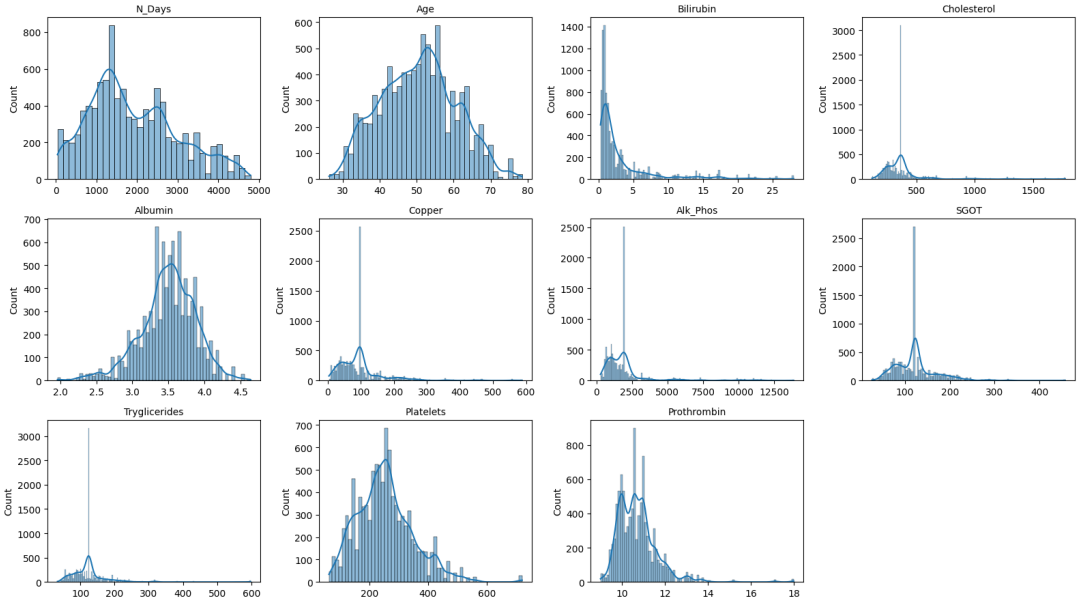
- Copper: Outliers từ 200 đến 600, chỉ ra nồng độ đồng bất thường.
- Alk_Phos: Outliers từ 4000 đến 14000, thể hiện biến thiên lớn.
- SGOT: Ít outliers từ 250 đến 400, một số giá trị cao hơn bình thường.
- Triglycerides: Outliers từ 300 đến 600, ít trường hợp cao.
- Platelets: Ít outliers trên 500, phần lớn trong phạm vi bình thường.

Tổng kết: Các biến Bilirubin, Cholesterol, Copper, và Alk_Phos có nhiều outliers, phản ánh biến thiên lớn và chỉ số sinh hóa bất thường. Trong tập dữ liệu y tế, outliers có thể ảnh hưởng đến hiệu suất mô hình học máy, đặc biệt với các thuật toán nhạy cảm như KNN. Nhóm sử dụng phương pháp IQR để xử lý.

Các bước IQR:

1. Tính: $Q1$ (25%), $Q3$ (75%), $IQR = Q3 - Q1$.
2. Khoảng hợp lý: $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$.
3. Phát hiện: Giá trị ngoài khoảng trên là outlier.
4. Xử lý: Cắt (Đưa giá trị về ngưỡng giới hạn) hoặc Thay trung vị (Thay outlier bằng median).

3.4 Kiểm tra phân phối



Hình 7: Biểu đồ histogram và đường KDE mô tả phân bố của các thuộc tính

Xét sự phân phối của các thuộc tính liên tục dựa trên biểu đồ Histogram với đường mật độ hạt nhân KDE. Biểu đồ histogram cho thấy tần số của các giá trị trong các khoảng nhất định, trong khi đường KDE ước lượng mật độ xác suất của biến liên tục.

Hầu hết các thuộc tính liên tục (trừ Albumin và Platelets) đều có phân bố lệch phải, với đa số giá trị tập trung ở mức thấp và đuôi dài bên phải, cho thấy sự hiện diện của một số ít giá trị cao bất thường, phù hợp với đặc điểm của dữ liệu y khoa.

3.5 Chuẩn hóa dữ liệu

Để đảm bảo các thuộc tính liên tục có cùng thang đo và không bị ảnh hưởng bởi sự khác biệt về đơn vị hoặc độ lớn, dữ liệu đã được chuẩn hóa bằng phương pháp **StandardScaler**. Phương pháp này biến đổi các giá trị của các thuộc tính sao cho chúng có trung bình bằng 0 và độ lệch chuẩn bằng 1, theo công thức:

$$z = \frac{x - \mu}{\sigma}$$

trong đó x là giá trị gốc, μ là trung bình, và σ là độ lệch chuẩn của thuộc tính.

Các thuộc tính được chuẩn hóa bao gồm: "N_Days", "Age", "Bilirubin", "Cholesterol", "Albumin", "Copper", "Alk_Phos", "SGOT", "Tryglicerides", "Platelets", và "Prothrombin". Việc chuẩn hóa giúp giảm tác động của các giá trị ngoại lai (outliers) đã được xử lý trước đó và đảm bảo rằng các thuật toán học máy (nếu áp dụng sau này) hoạt động hiệu quả hơn, đặc biệt với các phương pháp nhạy cảm với thang đo như hồi quy hoặc SVM.

Sau khi chuẩn hóa, các thuộc tính này có phân bố với trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp cân bằng đóng góp của từng thuộc tính trong quá trình phân tích hoặc huấn luyện mô hình.

4 Giảm chiều dữ liệu

Để hiểu rõ hơn về cấu trúc dữ liệu và mối liên hệ giữa các chiều với đầu ra `Stage`, hai phương pháp giảm chiều phổ biến là PCA (Principal Component Analysis) và LDA (Linear Discriminant Analysis) đã được áp dụng trong bài báo cáo này.

Nguyên lý hoạt động của PCA (Principal Component Analysis)

- PCA là kỹ thuật giảm chiều không có giám sát, mục tiêu là tìm các trục mới (gọi là *thành phần chính*) sao cho dữ liệu được chiếu lên các trục đó giữ lại tối đa phương sai.
- Cho tập dữ liệu $X \in \mathbb{R}^{n \times d}$ (với n mẫu và d đặc trưng), PCA tìm ma trận chiếu $W \in \mathbb{R}^{d \times k}$ sao cho:

  $$Z = XW$$

  với Z là dữ liệu đã giảm chiều còn k chiều, sao cho phương sai của Z là lớn nhất.
- Quá trình thực hiện PCA:

  1. Chuẩn hóa dữ liệu về trung bình 0.
  2. Tính ma trận hiệp phương sai:

     $$\Sigma = \frac{1}{n} X^T X$$

  3. Tính các trị riêng (eigenvalues) và vector riêng (eigenvectors) của Σ.
  4. Chọn k vector riêng ứng với k trị riêng lớn nhất để tạo thành ma trận chiếu W.
- Các thành phần chính không có ý nghĩa phân loại trực tiếp, vì PCA không sử dụng thông tin nhãn. Thay vào đó, nó giữ lại càng nhiều phương sai (tức thông tin tổng quát) càng tốt.

Nguyên lý hoạt động của LDA (Linear Discriminant Analysis)

- LDA là kỹ thuật giảm chiều có giám sát, mục tiêu là tìm các trục mới (gọi là *thành phần phân biệt*) sao cho khả năng phân biệt giữa các lớp là lớn nhất.

- LDA sử dụng thông tin nhãn lớp để tối đa hóa khoảng cách giữa các lớp (tăng *phương sai giữa các lớp*) và đồng thời giảm sự phân tán trong cùng một lớp (giảm *phương sai trong lớp*).
- Cho tập dữ liệu có K lớp, LDA tìm ma trận chiếu $W \in \mathbb{R}^{d\times(K-1)}$ để ánh xạ dữ liệu X về không gian $(K-1)$ chiều:

$$Z = XW$$

- Quá trình thực hiện:
  1. Tính ma trận phương sai trong lớp S_W và giữa các lớp S_B:

$$S_W = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu_k)(x_i - \mu_k)^T$$

$$S_B = \sum_{k=1}^{K} n_k(\mu_k - \mu)(\mu_k - \mu)^T$$

     trong đó μ_k là trung bình lớp k, μ là trung bình toàn bộ, n_k là số mẫu thuộc lớp k.
  2. Giải bài toán tổng quát:

$$\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

  3. Nghiệm của bài toán là các vector riêng tương ứng với trị riêng lớn nhất của:

$$S_W^{-1} S_B$$

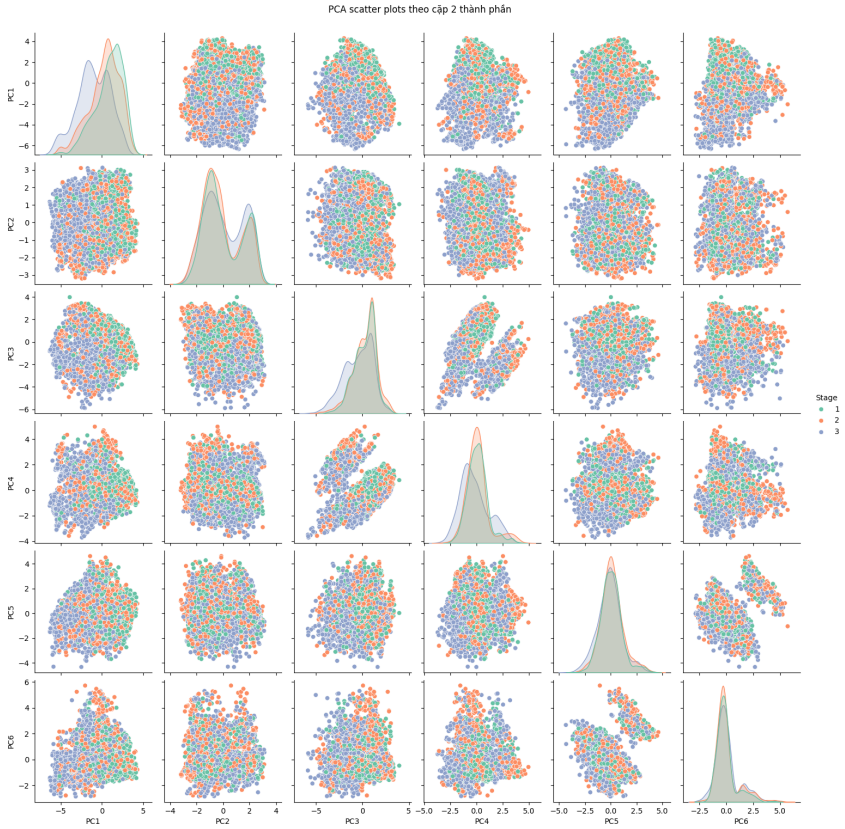
- Số lượng thành phần phân biệt tối đa là $K-1$, tương ứng với số lớp trừ đi một.

4.1 Trực quan dữ liệu theo thành phần chính

4.1.1 PCA - Principal Component Analysis

- Áp dụng PCA để trích xuất các thành phần chính nhằm rút gọn chiều dữ liệu đầu vào.

- Lựa chọn 6 thành phần đầu tiên (PC1 đến PC6) dựa trên tỷ lệ phương sai giữ lại.
- Trực quan hóa dữ liệu bằng scatter plot cho từng cặp thành phần nhằm phân tích cấu trúc và khả năng phân tách giữa các nhóm dữ liệu.

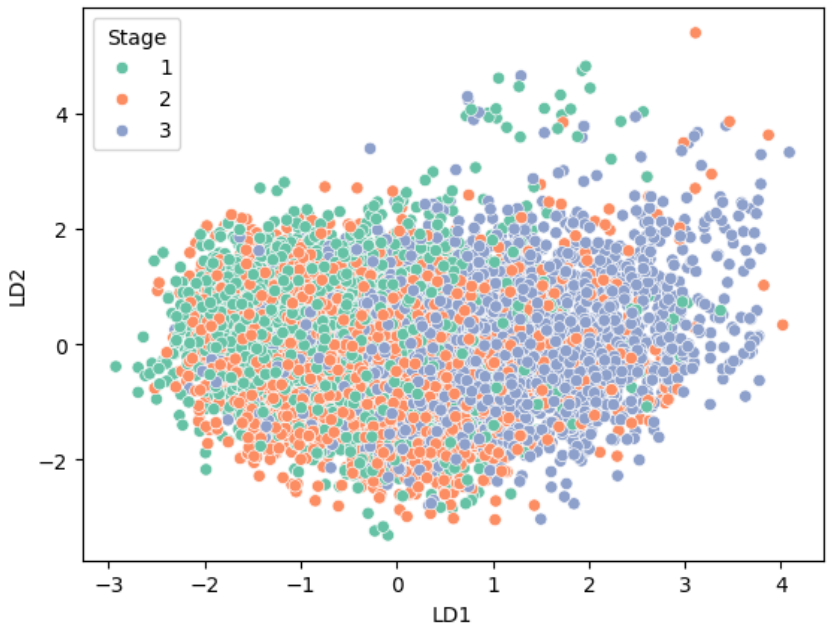


Hình 8: Biểu đồ scatter giữa các cặp thành phần PCA

- Nhận xét:
  - Các scatter plot cho thấy sự chồng lấn đáng kể giữa các giai đoạn, biểu hiện qua việc các cụm dữ liệu không được tách biệt rõ ràng.
  - Phân phối dữ liệu trên các thành phần chính có xu hướng gần chuẩn, điều này cho thấy PCA đã rút trích được các chiều chính phản ánh phương sai tổng thể.
  - Tuy nhiên, các thành phần PCA không làm nổi bật được sự khác biệt giữa các nhãn lớp (giai đoạn), phản ánh hạn chế của PCA trong việc phục vụ cho bài toán phân loại.

4.1.2 LDA - Linear Discriminant Analysis

- Sử dụng LDA để giảm chiều dữ liệu về 2 chiều (LD1, LD2) nhằm tối ưu hóa khả năng phân biệt giữa các lớp thuộc biến `Stage`.
- Trực quan hóa dữ liệu trong không gian LD1–LD2 để đánh giá khả năng phân tách giữa các nhóm.



Hình 9: Biểu đồ LDA với hai thành phần LD1 và LD2

- Nhận xét:
  - Biểu đồ LDA thể hiện một mức độ phân tách nhất định giữa các giai đoạn, dù vẫn còn sự chồng lấn — đặc biệt là ở vùng trung tâm.
  - Các điểm dữ liệu có xu hướng tập trung ở khu vực trung tâm với một số điểm ngoại biên, phản ánh sự tương đồng cao giữa các nhóm.
  - Thành phần LD1 đóng vai trò chính trong việc phân biệt giữa các lớp, trong khi LD2 chủ yếu hỗ trợ điều chỉnh nhẹ về không gian.

4.2 Đánh giá định lượng

PCA – Tỷ lệ phương sai giữ lại

Bảng 3: Tỷ lệ phương sai giữ lại bởi các thành phần chính của PCA

Thành phần chính	**Tỷ lệ phương sai (%)**
PC1	18.67
PC2	9.74
PC3	8.94
PC4	6.54
PC5	5.70
PC6	5.28
Tổng	**54.87**

Nhận xét

- PC1 mang nhiều thông tin nhấtvới 18.67% phương sai, nhưng mức độ này vẫn còn tương đối thấp, cho thấy dữ liệu không bị chi phối mạnh bởi một thành phần chính duy nhất.
- Các thành phần còn lại có tỷ lệ phương sai khá đồng đều và giảm dần, cho thấy thông tin trong dữ liệu được phân tán đều qua nhiều chiều.
- Tổng cộng 6 thành phần đầu giữ lại 54.87% phương sai, tức là hơn một nửa thông tin ban đầu được bảo toàn. Hiệu suất ở mức trung bình nhưng vẫn đủ để giảm chiều trong khi vẫn giữ được cấu trúc tổng thể của dữ liệu.

LDA – Tỷ lệ phương sai giữa các lớp

Bảng 4: Tỷ lệ phương sai giữa các lớp của LDA

Thành phần phân biệt	**Tỷ lệ phương sai (%)**
LD1	91.26
LD2	8.74
Tổng	**100**

Nhận xét

- LD1 gần như là đủ để phân biệt các lớp với tỷ lệ phương sai lớn (91,26%), góp phần giải thích khả năng phân biệt giữa các lớp của thành phần phân biệt này
- Với tỷ lệ nhỏ (8.74%), LD2 không đóng góp nhiều vào khả năng phân biệt. Tuy nhiên, nó vẫn có thể hỗ trợ thêm trong trường hợp các lớp bị chồng lấp

5 Phương pháp

5.1 Phân cụm dữ liệu (Clustering)

5.1.1 K-Means Clustering

Tổng quan về phương pháp

K-Means là một thuật toán phân cụm không giám sát (unsupervised learning) phổ biến, được sử dụng để nhóm các điểm dữ liệu thành K cụm sao cho các điểm trong cùng một cụm có độ tương đồng cao, trong khi các cụm khác nhau có sự khác biệt lớn.

Mỗi cụm được đại diện bởi một điểm trung tâm gọi là *centroid* (tâm cụm), được tính bằng trung bình cộng của tất cả các điểm dữ liệu thuộc cụm đó.

Nguyên lý hoạt động của thuật toán

1. Khởi tạo

- Chọn trước số cụm K.
- Khởi tạo ngẫu nhiên K điểm centroid $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ từ tập dữ liệu hoặc không gian đặc trưng.

2. Gán nhãn cho từng điểm dữ liệu

- Với mỗi điểm dữ liệu $\mathbf{x}_i$, tính khoảng cách đến từng centroid $\mathbf{c}_j$ bằng khoảng cách Euclidean:

$$d(\mathbf{x}_i, \mathbf{c}_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - c_{jk})^2}$$

  trong đó:

  - $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ là vector đặc trưng của điểm i,
  - $\mathbf{c}_j = (c_{j1}, c_{j2}, \ldots, c_{jn})$ là centroid của cụm j,
  - n là số chiều của dữ liệu.

- Gán điểm $\mathbf{x}_i$ vào cụm có centroid gần nhất, tức là chọn j^* sao cho:

$$j^* = \arg\min_{j=1,\ldots,K} d(\mathbf{x}_i, \mathbf{c}_j)$$

3. Cập nhật centroid

- Sau khi gán nhãn cho tất cả các điểm, cập nhật lại centroid của mỗi cụm bằng trung bình cộng các điểm thuộc cụm đó:

$$\mathbf{c}_j = \frac{1}{|S_j|} \sum_{\mathbf{x}_i \in S_j} \mathbf{x}_i$$

  trong đó S_j là tập các điểm thuộc cụm j, và $|S_j|$ là số lượng điểm trong cụm.

4. Lặp lại

Tiếp tục lặp lại hai bước 2 và 3 cho đến khi:

- Các centroid không còn thay đổi (đạt được hội tụ), hoặc
- Số vòng lặp đạt giới hạn đã định trước.

5.1.2 Phương pháp Elbow

Một trong những cách phổ biến để chọn số cụm tối ưu K trong K-Means là phương pháp *Elbow*.

Nguyên lý và lý thuyết của phương pháp Elbow

- Khi chạy thuật toán K-Means với các giá trị K khác nhau, ta tính tổng bình phương khoảng cách giữa các điểm dữ liệu và centroid của cụm mà chúng thuộc về. Tổng này được gọi là *Within-Cluster Sum of Squares* (WCSS):

$$\text{WCSS}(K) = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in S_j} \|\mathbf{x}_i - \mathbf{c}_j\|^2$$

- WCSS biểu diễn mức độ phân tán trong các cụm; khi K tăng, cụm nhỏ hơn nên các điểm gần centroid hơn, khiến WCSS giảm.
- Tuy nhiên, khi K vượt qua một giá trị nhất định, sự cải thiện về WCSS là không đáng kể.
- Phương pháp *Elbow* dựa vào việc vẽ đồ thị WCSS theo K và chọn giá trị K tại điểm mà độ dốc của đồ thị giảm mạnh – tức điểm tạo thành "khuỷu tay" (elbow point).

5.2 Phân loại (Classification)

5.2.1 KNN - K-nearest neighbors

Tổng quan về phương pháp

KNN là thuật toán học có giám sát, không huấn luyện mô hình mà dựa vào k láng giềng gần nhất (dựa trên khoảng cách) để gán nhãn. Dễ hiểu, hiệu quả, nhưng tốn tài nguyên khi dữ liệu lớn.

Nguyên lý hoạt động của thuật toán KNN

KNN giả định rằng các điểm gần nhau trong không gian đặc trưng thường thuộc cùng lớp. Dưới đây là các bước thực hiện thuật toán:

1. Chuẩn bị dữ liệu

KNN yêu cầu tập dữ liệu huấn luyện $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, trong đó:

- $\mathbf{x}_i \in \mathbb{R}^d$: Vector đặc trưng d chiều của điểm huấn luyện thứ i.
- $y_i \in \{1, 2, \ldots, C\}$: Nhãn của $\mathbf{x}_i$, thuộc một trong C lớp.

Điểm dữ liệu mới $\mathbf{x} \in \mathbb{R}^d$ cần được dự đoán nhãn $\hat{y}$. Trước khi áp dụng KNN, dữ liệu thường được chuẩn hóa (normalization) để đảm bảo các đặc trưng có cùng thang đo, tránh đặc trưng có giá trị lớn chi phối khoảng cách.

2. Tính khoảng cách đến các điểm huấn luyện

Để tìm láng giềng gần nhất, KNN tính khoảng cách từ $\mathbf{x}$ đến mỗi $\mathbf{x}_i \in \mathcal{D}$ bằng một hàm đo. Các hàm khoảng cách phổ biến bao gồm:

- **Khoảng cách Euclidean (chuẩn L^2):**

$$d_{\text{Euc}}(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{d} (x_j - x_{i,j})^2}$$

  Đo độ dài đường thẳng giữa hai điểm, phù hợp với dữ liệu liên tục.

- **Khoảng cách Manhattan (chuẩn L^1):**

$$d_{\text{Man}}(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{d} |x_j - x_{i,j}|$$

  Phù hợp với dữ liệu rời rạc hoặc đặc trưng không liên tục.

- **Khoảng cách Minkowski (chuẩn L^p):**

$$d_{\text{Mink}}(\mathbf{x}, \mathbf{x}_i) = \left(\sum_{j=1}^{d} |x_j - x_{i,j}|^p \right)^{1/p}$$

  Tổng quát hóa với $p = 1$ (Manhattan) và $p = 2$ (Euclidean).

Trong đó, x_j và $x_{i,j}$ lần lượt là giá trị đặc trưng thứ j của $\mathbf{x}$ và $\mathbf{x}_i$.

3. Tìm k láng giềng gần nhất

Sắp xếp các điểm $\mathbf{x}_i \in \mathcal{D}$ theo khoảng cách $d(\mathbf{x}, \mathbf{x}_i)$ tăng dần và chọn k điểm có khoảng cách nhỏ nhất, tạo tập láng giềng:

$$\mathcal{N}_k(\mathbf{x}) = \{\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \ldots, \mathbf{x}_{(k)}\}$$

Tập nhãn tương ứng là:

$$\mathcal{Y}_k(\mathbf{x}) = \{y_{(1)}, y_{(2)}, \ldots, y_{(k)}\}$$

Giá trị k được chọn dựa trên thử nghiệm, thường là số lẻ để tránh hòa phiếu khi biểu quyết.

4. Biểu quyết để dự đoán nhãn

Trong bài toán phân loại, nhãn dự đoán $\hat{y}$ được xác định bằng biểu quyết đa số:

$$\hat{y} = \arg \max_{c \in \{1,2,\ldots,C\}} \sum_{j=1}^{k} \mathbb{I}(y_{(j)} = c)$$

Trong đó:

- $\mathbb{I}(y_{(j)} = c) = 1$ nếu $y_{(j)} = c$, và 0 nếu không.
- $\sum_{j=1}^{k} \mathbb{I}(y_{(j)} = c)$ đếm số láng giềng thuộc lớp c.

Nếu dùng trọng số (weighted KNN), khoảng cách $d(\mathbf{x}, \mathbf{x}_{(j)})$ có thể được sử dụng để ưu tiên láng giềng gần hơn, ví dụ: trọng số tỷ lệ nghịch $w_j = 1/d(\mathbf{x}, \mathbf{x}_{(j)})$.

5. Gán nhãn cho điểm mới

Nhãn $\hat{y}$ từ bước biểu quyết được gán cho $\mathbf{x}$, hoàn tất quá trình dự đoán.

5.2.2 Multi Layers Perceptron

Tổng quan về phương pháp

Multi-Layer Perceptron (MLP) là một kiến trúc mạng neuron truyền thẳng (feedforward neural network) gồm nhiều lớp liên kết liên tiếp. Là phần mở rộng của Perceptron đơn lớp, MLP khắc phục giới hạn tuyến tính và có khả năng mô hình hóa các quan hệ phi tuyến giữa đầu vào và đầu ra. MLP thường được sử dụng trong các bài toán học máy có giám sát.

Cấu trúc của MLP

Một mạng MLP gồm ba thành phần chính:

1. **Lớp đầu vào (Input Layer):** Nhận dữ liệu đầu vào thô, với số neuron bằng số đặc trưng của dữ liệu. Lớp này không tính toán mà chỉ truyền dữ liệu vào lớp ẩn.
2. **Lớp ẩn (Hidden Layers):** Một hoặc nhiều lớp, nơi thực hiện các biến đổi phi tuyến nhờ vào kết nối đầy đủ giữa các neuron. Mỗi neuron có trọng số và bias riêng. Số lượng lớp và neuron là các siêu tham số cần điều chỉnh phù hợp với bài toán.
3. **Lớp đầu ra (Output Layer):** Tạo ra kết quả cuối cùng. Số neuron phụ thuộc vào loại bài toán:
   - *Hồi quy:* 1 neuron (dự đoán giá trị liên tục).
   - *Phân loại nhị phân:* 1 neuron (sigmoid) hoặc 2 neuron (softmax).
   - *Phân loại đa lớp:* N neuron với hàm softmax, N là số lớp.

Nguyên lý hoạt động

Quá trình hoạt động và học của MLP diễn ra qua hai pha chính lặp đi lặp lại: **Pha truyền xuôi** (Forward Propagation) và **Pha lan truyền ngược** (Backpropagation).

Pha Truyền Xuôi (Forward Propagation)

Quá trình này diễn ra theo các bước sau:

1. **Nhận đầu vào:** Dữ liệu X được đưa vào lớp đầu vào.
2. **Tính toán tại các lớp ẩn:** Với mỗi lớp l (từ lớp đầu tiên sau lớp input đến lớp cuối cùng trước lớp output):

- Đối với mỗi neuron j trong lớp l, tính *tổng trọng số đầu vào (weighted sum)* $z_j^{(l)}$:

$$z_j^{(l)} = \left(\sum_i w_{ji}^{(l)} a_i^{(l-1)} \right) + b_j^{(l)}$$

  Trong đó: $a_i^{(l-1)}$ là đầu ra (activation) của neuron i ở lớp trước đó $(l-1)$, $w_{ji}^{(l)}$ là trọng số kết nối từ neuron i (lớp $l-1$) đến neuron j (lớp l), và $b_j^{(l)}$ là bias của neuron j (lớp l).

- Đưa tổng trọng số $z_j^{(l)}$ qua một *hàm kích hoạt (activation function)* ϕ phi tuyến tính (ví dụ: Sigmoid, Tanh, ReLU) để tính đầu ra $a_j^{(l)}$ của neuron:

$$a_j^{(l)} = \phi(z_j^{(l)})$$

  Hàm kích hoạt giúp mạng học được các mối quan hệ phức tạp, phi tuyến.

- Đầu ra $a^{(l)}$ của lớp l sẽ trở thành đầu vào cho lớp tiếp theo $l+1$.

3. **Tính toán tại lớp đầu ra (Output Layer):** Quá trình tương tự như lớp ẩn, nhưng hàm kích hoạt ϕ_{out} ở lớp đầu ra thường được chọn phù hợp với bài toán:

   - *Hồi quy:* Thường dùng hàm tuyến tính ($\phi_{\text{out}}(z) = z$) hoặc không dùng hàm kích hoạt.
   - *Phân loại nhị phân:* Thường dùng hàm Sigmoid.
   - *Phân loại đa lớp:* Thường dùng hàm Softmax.

4. **Kết quả dự đoán:** Đầu ra của lớp đầu ra, ký hiệu là $\hat{y}$, là kết quả dự đoán của mạng MLP cho mẫu dữ liệu đầu vào X.

Tính toán Lỗi (Calculating Error)

- Sau khi có kết quả dự đoán $\hat{y}$ từ pha truyền xuôi, chúng ta so sánh nó với giá trị thực tế (nhãn) y của dữ liệu huấn luyện.
- Một *hàm mất mát (loss function)* $L(\hat{y}, y)$ (còn gọi là hàm chi phí - cost function) được sử dụng để đo lường mức độ sai khác giữa dự đoán và giá trị thực. Ví dụ:
  - *Hồi quy:* Trung bình sai số bình phương - MSE: $L = \frac{1}{N} \sum (y_i - \hat{y}_i)^2$.

– *Phân loại:* Cross-Entropy.

Pha Lan Truyền Ngược (Backpropagation)

Đây là thuật toán cốt lõi để huấn luyện MLP, nhằm điều chỉnh các *trọng số (weights)* W và *biases* b của mạng sao cho giảm thiểu hàm mất mát L.

- **Lan truyền lỗi ngược:** Lỗi tính toán được ở lớp đầu ra sẽ được lan truyền ngược lại qua mạng, từ lớp cuối cùng về các lớp ẩn cho đến lớp đầu tiên.
- **Tính toán Gradient:** Thuật toán tính toán *gradient* (đạo hàm riêng) của hàm mất mát L theo từng trọng số $w_{ji}^{(l)}$ và bias $b_j^{(l)}$ trong mạng, tức là tính $\frac{\partial L}{\partial w_{ji}^{(l)}}$ và $\frac{\partial L}{\partial b_j^{(l)}}$. Gradient cho biết "mức độ đóng góp"của mỗi tham số vào lỗi tổng thể và hướng cần điều chỉnh tham số để giảm lỗi nhanh nhất. Việc tính toán này dựa trên quy tắc chuỗi (chain rule) trong giải tích.
- **Cập nhật Trọng số và Bias:** Sử dụng gradient đã tính toán, một thuật toán tối ưu hóa (optimization algorithm) như *Gradient Descent* (hoặc các biến thể như SGD, Adam, RMSprop) được dùng để cập nhật các tham số:

$$w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} - \eta \frac{\partial L}{\partial w_{ji}^{(l)}}$$

$$b_j^{(l)} \leftarrow b_j^{(l)} - \eta \frac{\partial L}{\partial b_j^{(l)}}$$

Trong đó η là *tốc độ học (Learning rate)*, một siêu tham số kiểm soát độ lớn của bước cập nhật. Việc chọn giá trị η phù hợp là rất quan trọng (thường $\eta \in (0, 1)$) .

Quá trình Lặp (Iteration)

- Toàn bộ quy trình: **Truyền Xuôi** → **Tính Lỗi** → **Lan Truyền Ngược** → **Cập nhật Tham số** được lặp đi lặp lại trên nhiều mẫu dữ liệu (thường là các lô nhỏ - *mini-batches*) hoặc toàn bộ tập dữ liệu huấn luyện (gọi là một *epoch*).
- Quá trình lặp này tiếp tục cho đến khi mô hình đạt được hiệu suất mong muốn (ví dụ: hàm mất mát trên tập validation ngừng cải thiện) hoặc đạt đến số lượng epochs tối đa đã định trước.

5.2.3 SVM - Support Vector Machine

Tổng quan về phương pháp

SVM là một thuật toán học có giám sát, chủ yếu được sử dụng cho bài toán phân loại, mặc dù cũng có biến thể cho hồi quy (Support Vector Regression - SVR). Nguyên lý cốt lõi của SVM là tìm ra một siêu phẳng (hyperplane) trong không gian đặc trưng N-chiều (N là số lượng đặc trưng) phân tách tối ưu các điểm dữ liệu của các lớp khác nhau.

Siêu phẳng tối ưu là siêu phẳng có khoảng cách (margin) lớn nhất đến các điểm dữ liệu gần nhất của bất kỳ lớp nào. Những điểm dữ liệu gần nhất này nằm trên các siêu phẳng song song xác định lề và được gọi là véc-tơ hỗ trợ (support vectors). Việc tối đa hóa lề này được chứng minh là giúp cải thiện khả năng tổng quát hóa của mô hình trên dữ liệu chưa thấy.

Nguyên lý hoạt động

Linear SVM (Trường hợp Phân tách Tuyến tính)

- Giả sử dữ liệu có thể phân tách tuyến tính (linearly separable) bằng một siêu phẳng. Siêu phẳng được định nghĩa bởi phương trình:

$$w \cdot x + b = 0$$

  trong đó w là vector pháp tuyến (vector trọng số) và b là hệ số chặn (bias term).

- Việc này có thể được tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có tọa độ $\mathbf{x}_0$ tới siêu phẳng có phương trình $\mathbf{w}^T\mathbf{x} + b = 0$ được xác định bởi:

$$\frac{|\mathbf{w}^T\mathbf{x}_0 + b|}{\|\mathbf{w}\|_2}$$

  Với $\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$ với d là số chiều của không gian.

- Ta quan sát thấy một điểm quan trọng sau đây, với cặp dữ liệu $(\mathbf{x}_n, y_n)$ bất kỳ, khoảng cách từ điểm tới mặt phân chia là:

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{\|\mathbf{w}\|_2}$$

Điều này có thể để nhận thấy vì theo giá trị ở trên, $y_n \in \{1, -1\}$ - nhãn của điểm dữ liệu luôn cùng dấu với phía của $\mathbf{x}_n$. Từ đó suy ra y_n cùng dấu với $(\mathbf{w}^T\mathbf{x}_n + b)$, và tử số luôn là một số không âm.

- Với mặt phân chia như trên, *margin* được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hai lớp):

$$margin = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{\|\mathbf{w}\|_2}$$

- Bài toán tối ưu trong SVM chính là bài toán tìm $\mathbf{w}$ và b sao cho *margin* này đạt giá trị lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{\|\mathbf{w}\|_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}$$

- Nếu ta thay vector hệ số $\mathbf{w}$ bởi $k\mathbf{w}$ và b bởi kb trong đó k là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từ điểm đến mặt phân chia không đổi, tức *margin* không đổi. Dựa trên tính chất này, ta có thể giả sử:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

Như vậy, với mỗi n, ta có:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \tag{1}$$

Vậy bài toán tối ưu (1) có thể được dự về bài toán tối ưu có ràng buộc sau đây:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2}$$

$$\text{subject to: } y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \forall n = 1, 2, \ldots, N$$

Bằng 1 biến đổi đơn giản, ta có thể đưa bài toán này về bài toán dưới đây:

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{subject to: } 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N$$

Soft Margin SVM (Trường hợp Không Phân tách Tuyến tính)

- Để xử lý dữ liệu không hoàn toàn phân tách tuyến tính (do nhiễu hoặc bản chất chồng chéo của dữ liệu), SVM giới thiệu các *biến bù (slack variables)* $\xi_i \geq 0$. Các biến này đo lường mức độ vi phạm lề của mỗi điểm dữ liệu x_i.
- $\xi_i = 0$ nếu điểm x_i nằm đúng phía và ngoài lề; $0 < \xi_i \leq 1$ nếu điểm nằm trong lề nhưng đúng phía; và $\xi_i > 1$ nếu điểm nằm sai phía của siêu phẳng.
- Bài toán tối ưu được điều chỉnh như sau:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

  Thỏa mãn ràng buộc:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

- Tham số $C > 0$ là *tham số điều chuẩn (regularization parameter)* hay tham số phạt. Nó kiểm soát sự đánh đổi giữa việc tối đa hóa margin (đảm bảo phân biệt rõ ràng giữa các lớp - C nhỏ) và việc phân loại chính xác nhiều điểm huấn luyện hơn (dễ bị overfitting - C lớn).

Non-Linear SVM và Kỹ thuật Kernel (Kernel Trick)

- Với dữ liệu có mối quan hệ phi tuyến, SVM sử dụng kỹ thuật *kernel trick* để ánh xạ dữ liệu sang không gian đặc trưng có chiều cao hơn, nơi dữ liệu có thể phân tách tuyến tính.
- Không cần biểu diễn tường minh phép ánh xạ $\Phi(x)$, mà chỉ cần tính tích vô hướng $\Phi(x_i) \cdot \Phi(x_j)$ thông qua hàm kernel $K(x_i, x_j) = \langle \Phi(x_i)^T, \Phi(x_j) \rangle$.
- Một số hàm kernel phổ biến:
  - **Linear Kernel:** $K(x_i, x_j) = x_i^T x_j$.
    .
  - **Polynomial Kernel:** $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$.
    Các tham số:
    * $\gamma > 0$: Hệ số điều chỉnh độ ảnh hưởng của tích vô hướng $x_i^T x_j$.
    * r: Hằng số dịch (bias) giúp tăng tính linh hoạt của mô hình.

    - d: Bậc của đa thức. Bậc cao cho phép mô hình học được ranh giới quyết định phức tạp hơn, nhưng dễ dẫn tới overfitting.
  - **RBF (Gaussian) Kernel:** $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$.
    Tham số:
    - $\gamma > 0$: Kiểm soát độ rộng của "vùng ảnh hưởng" xung quanh mỗi điểm huấn luyện. Giá trị γ lớn → vùng ảnh hưởng hẹp → mô hình có xu hướng overfit. Ngược lại, γ nhỏ → vùng ảnh hưởng rộng → dễ underfit.
    - (Lưu ý: đôi khi $\gamma = \frac{1}{2\sigma^2}$, với σ là độ lệch chuẩn của phân bố Gaussian.)
  - **Sigmoid Kernel:** $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$.
    Các tham số:
    - γ: Tương tự như trong kernel đa thức, điều chỉnh độ nhạy của hàm kích hoạt sigmoid.
    - r: Hằng số dịch bias. Giá trị của γ và r cần được chọn cẩn thận để đảm bảo hàm kernel là dương bán xác định (positive semi-definite), nếu không mô hình SVM có thể không hội tụ.

- Việc chọn kernel phù hợp và tinh chỉnh các tham số (γ, d, r) cùng với C là rất quan trọng để đạt hiệu suất cao.

Multi-class SVM (SVM với nhiều lớp phân loại)

- SVM ban đầu được thiết kế cho bài toán phân loại nhị phân. Để mở rộng cho bài toán phân loại nhiều lớp ($K > 2$), hai phương pháp phổ biến được sử dụng:

- **One-vs-Rest (OvR):** Huấn luyện K mô hình SVM, mỗi mô hình phân biệt một lớp với phần còn lại. Khi dự đoán, mỗi mô hình trả về một giá trị, lớp tương ứng với giá trị lớn nhất sẽ được chọn.

- **One-vs-One (OvO):** Huấn luyện $K(K-1)/2$ mô hình SVM, mỗi mô hình phân biệt giữa một cặp lớp. Khi dự đoán, sử dụng nguyên tắc biểu quyết (voting): lớp nào thắng nhiều nhất sẽ được chọn.

5.3 Hồi quy (Regression)

5.3.1 Linear regression

Hồi quy tuyến tính là một thuật toán học có giám sát, mô hình hóa mối quan hệ tuyến tính giữa biến phụ thuộc y và một hoặc nhiều biến độc lập.

Hồi quy tuyến tính đơn

Với một biến đầu vào x, mô hình có dạng:

$$\hat{y} = w_0 + w_1 x$$

Trong đó:

- $\hat{y}$ là giá trị dự đoán của biến phụ thuộc.
- x là giá trị của biến độc lập.
- w_0 là hệ số chặn (intercept) hay bias, là giá trị của $\hat{y}$ khi $x = 0$.
- w_1 là hệ số góc (slope), biểu thị mức độ thay đổi của $\hat{y}$ khi x thay đổi một đơn vị.

Nhiệm vụ là tìm ra các giá trị w_0 và w_1 sao cho đường thẳng này "khớp"tốt nhất với dữ liệu huấn luyện.

Hồi quy tuyến tính bội

Khi số biến độc lập là D $(x_1, \ldots, x_D)$, mô hình mở rộng thành:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

Để biểu diễn gọn hơn, ta có thể sử dụng ký hiệu vector. Đặt vector trọng số $\boldsymbol{w} = [w_0, w_1, \ldots, w_D]^T$ và vector đầu vào mở rộng $\bar{\boldsymbol{x}} = [1, x_1, \ldots, x_D]^T$ (thêm $x_0 = 1$ để tương ứng với w_0). Khi đó, phương trình hồi quy có dạng:

$$\hat{y} = \boldsymbol{w}^T \bar{\boldsymbol{x}}$$

Đối với toàn bộ tập dữ liệu huấn luyện gồm N điểm dữ liệu, ta có thể biểu diễn dưới dạng ma trận:

$$\boldsymbol{Y} \approx \boldsymbol{X}\boldsymbol{w}$$

Trong đó:

- $\boldsymbol{Y}$ là vector cột chứa các giá trị thực tế của biến phụ thuộc $[y_1, y_2, \ldots, y_N]^T$.
- $\boldsymbol{X}$ là ma trận dữ liệu kích thước $N \times (D+1)$, mỗi hàng là một điểm dữ liệu $\bar{\boldsymbol{x}}_i^T = [1, x_{i1}, \ldots, x_{iD}]$.
- $\boldsymbol{w}$ là vector cột chứa các hệ số hồi quy $[w_0, w_1, \ldots, w_D]^T$.

Ước lượng tham số mô hình

Để tìm ra bộ tham số $\boldsymbol{w}$ tối ưu, chúng ta cần định nghĩa một hàm mất mát (loss function) để đo lường mức độ sai khác giữa giá trị dự đoán và giá trị thực tế. Hàm mất mát phổ biến nhất trong Hồi quy tuyến tính là tổng bình phương sai số:

$$L(\boldsymbol{w}) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\bar{\boldsymbol{x}}_i)^2$$

Hoặc dưới dạng ma trận:

$$L(\boldsymbol{w}) = ||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}||_2^2$$

Mục tiêu là tìm $\boldsymbol{w}$ để cực tiểu hóa $L(\boldsymbol{w})$. Do L là hàm lồi, nghiệm tối ưu toàn cục được xác định bằng cách giải:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{Y}) = \boldsymbol{0}$$

Từ đó, nghiệm dạng đóng là:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

với điều kiện $\boldsymbol{X}^T\boldsymbol{X}$ khả nghịch.

5.3.2 Random Forest Regressor

Mô hình Random Forest (RF) là một phương pháp học máy thuộc nhóm ensemble learning. Trong bài toán hồi quy, Random Forest Regression sử dụng nhiều cây quyết định (decision trees) để dự đoán giá trị liên tục.

Nguyên lý hoạt động

Random Forest kết hợp hai kỹ thuật chính:

- **Bagging (Bootstrap Aggregating)**: Tạo ra nhiều tập dữ liệu con bằng cách lấy mẫu có thay thế từ tập huấn luyện gốc.
- **Ngẫu nhiên hóa đặc trưng (Feature Randomization)**: Tại mỗi nút phân tách của cây, chỉ xét một tập con ngẫu nhiên các đặc trưng.

Mỗi cây quyết định được huấn luyện trên một tập dữ liệu bootstrap và đưa ra một dự đoán. Dự đoán cuối cùng của mô hình là trung bình của tất cả dự đoán từ các cây:

$$\hat{f}(x) = \frac{1}{M} \sum_{m=1}^{M} \hat{f}_m(x),$$

trong đó $\hat{f}_m(x)$ là dự đoán từ cây thứ m.

Lấy mẫu và cấu trúc cây

Với tập huấn luyện $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, Random Forest tạo B tập bootstrap $\mathcal{B}_b$ bằng cách lấy mẫu có thay thế kích thước n, dẫn đến các quan sát lặp lại. Mỗi cây T_b được xây dựng từ $\mathcal{B}_b$ qua các bước:

1. Chọn ngẫu nhiên m biến từ p biến (thường $m = \sqrt{p}$).
2. Tìm biến và điểm phân tách tốt nhất trong m biến bằng cách tối thiểu hóa:

$$\text{MSE} = \frac{1}{n_{\text{node}}} \sum_{i \in \text{node}} (Y_i - \bar{Y}_{\text{node}})^2.$$

3. Tách node thành hai node con, lặp lại đến khi số mẫu trong node nhỏ hơn $n_{\min}$ (tương đương `min_samples_leaf`).

Khi B tăng, phương sai mô hình giảm, xấp xỉ:

$$\text{Var}(\hat{f}(x)) \approx \frac{\sigma^2}{B},$$

với σ^2 là phương sai của mỗi cây.

Đánh giá với mẫu Out-of-Bag (OOB)

Các quan sát không dùng để huấn luyện một cây, gọi là mẫu *out-of-bag* (OOB), được dùng để ước lượng lỗi:

$$\text{OOB Error} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{f}_{\text{OOB}}(X_i))^2,$$

trong đó $\hat{f}_{\text{OOB}}(X_i)$ là trung bình dự đoán từ các cây không sử dụng X_i.

Tham số quan trọng trong huấn luyện

Các tham số chính:

- `n_estimators`: Số cây.
- `max_depth`: Độ sâu tối đa.
- `min_samples_split`, `min_samples_leaf`: Số mẫu tối thiểu tại nút/lá.
- `max_features`: Số đặc trưng ngẫu nhiên tại mỗi nút (thường $\sqrt{d}$).
- `bootstrap`: Có dùng bootstrap hay không.

Tầm quan trọng của biến

Random Forest đánh giá đặc trưng qua:

- **MDI (Mean Decrease in Impurity)**: Tổng giảm MSE khi phân tách.
- **MDA (Mean Decrease in Accuracy)**: Giảm độ chính xác khi hoán vị đặc trưng trong OOB.

6 Thực nghiệm và kết quả

6.1 Các chỉ số đánh giá mô hình

6.1.1 Chỉ số đánh giá hiệu suất mô hình phân loại

Trong các bài toán phân loại, đánh giá mô hình là bước quan trọng để kiểm chứng khả năng khái quát hóa trên dữ liệu chưa từng thấy. Các chỉ số phổ biến như độ chính xác, độ bao phủ và độ tin cậy thường được dùng để đo lường hiệu quả phân biệt giữa các lớp.

Ma trận nhầm lẫn (Confusion Matrix)

Ma trận nhầm lẫn là một công cụ cơ bản giúp tổng hợp và minh họa kết quả phân loại của mô hình so với giá trị thực tế. Trong bài toán phân loại k lớp, đây là một ma trận vuông kích thước $k \times k$, trong đó phần tử ở hàng i, cột j biểu thị số lượng mẫu thực tế thuộc lớp i nhưng được dự đoán là lớp j.

Đối với trường hợp đặc biệt là phân loại nhị phân, ma trận nhầm lẫn có cấu trúc đơn giản hơn, bao gồm bốn thành phần chính:

- **True Positive (TP)**: Số lượng mẫu thuộc lớp dương và được mô hình dự đoán đúng là dương.
- **True Negative (TN)**: Số lượng mẫu thuộc lớp âm và được mô hình dự đoán đúng là âm.
- **False Positive (FP)**: Số lượng mẫu thuộc lớp âm nhưng bị mô hình dự đoán nhầm là dương.
- **False Negative (FN)**: Số lượng mẫu thuộc lớp dương nhưng bị mô hình dự đoán nhầm là âm.

Dựa trên các thành phần này, nhiều chỉ số đánh giá hiệu suất mô hình phân loại được xây dựng nhằm phản ánh các khía cạnh khác nhau của chất lượng dự đoán.

Accuracy

Accuracy (độ chính xác tổng thể) là chỉ số đo lường tỷ lệ dự đoán đúng trên toàn bộ tập dữ liệu kiểm thử:

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

Chỉ số này phản ánh khả năng dự đoán chính xác tổng thể của mô hình. Tuy nhiên, trong các trường hợp dữ liệu không cân bằng (một lớp chiếm ưu thế), accuracy có thể bị thiên lệch và không phản ánh đúng hiệu suất thực sự. Do đó, chỉ số này thường được sử dụng kết hợp với các chỉ số khác để có cái nhìn toàn diện hơn.

Precision

Precision (độ chính xác với lớp dương) đo lường tỷ lệ dự đoán đúng dương tính trên tổng số dự đoán là dương tính:

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

Precision đặc biệt quan trọng trong các bài toán mà sai sót trong việc dự đoán nhầm dương tính (FP) gây ra hậu quả lớn, ví dụ như trong chẩn đoán bệnh.

Recall

Recall (độ bao phủ lớp dương) đo lường tỷ lệ các mẫu thực sự dương tính được mô hình phát hiện chính xác:

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

Chỉ số này phù hợp với các bài toán yêu cầu không bỏ sót các mẫu dương, chẳng hạn như phát hiện gian lận hoặc bệnh lý nghiêm trọng. Trong trường hợp dữ liệu cân bằng, precision và recall thường được coi trọng như nhau.

F1-score

F1-score là trung bình điều hòa giữa precision và recall, phản ánh sự cân bằng giữa hai chỉ số này:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chỉ số này đặc biệt hữu ích trong các bài toán mà cả precision và recall đều đóng vai trò quan trọng, và thường được sử dụng trong đánh giá mô hình khi dữ liệu không hoàn toàn cân bằng.

6.1.2 Chỉ số đánh giá hồi quy

Trong bài toán hồi quy, các chỉ số đánh giá giúp phản ánh sai lệch giữa giá trị dự đoán và thực tế, hỗ trợ so sánh và tối ưu mô hình một cách khách quan.

Mean Squared Error (MSE)

Sai số bình phương trung bình (MSE) đo lường giá trị trung bình của bình phương sai số giữa giá trị thực tế và giá trị dự đoán:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

trong đó y_i là giá trị thực, $\hat{y}_i$ là giá trị dự đoán và n là số lượng mẫu. Chỉ số MSE nhấn mạnh các sai số lớn hơn do việc bình phương hóa, do đó rất nhạy cảm với các giá trị ngoại lai. Giá trị MSE càng nhỏ phản ánh mô hình có độ chính xác càng cao.

Mean Absolute Error (MAE)

Sai số tuyệt đối trung bình (MAE) đo lường giá trị trung bình của sai số tuyệt đối giữa giá trị thực tế và dự đoán:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Khác với MSE, MAE không bình phương sai số nên ít bị ảnh hưởng bởi các điểm ngoại lai. MAE có cùng đơn vị với biến mục tiêu, giúp việc diễn giải kết quả trực quan hơn. Mô hình có MAE càng nhỏ càng được xem là hiệu quả.

R-squared (R^2)

Hệ số xác định (R^2) phản ánh tỷ lệ phương sai của biến phụ thuộc được giải thích bởi mô hình. Đối với hồi quy tuyến tính đơn có hệ số chặn, R^2 tương ứng với bình phương hệ số tương quan Pearson giữa giá trị thực và dự đoán. Công thức tính R^2 như sau:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

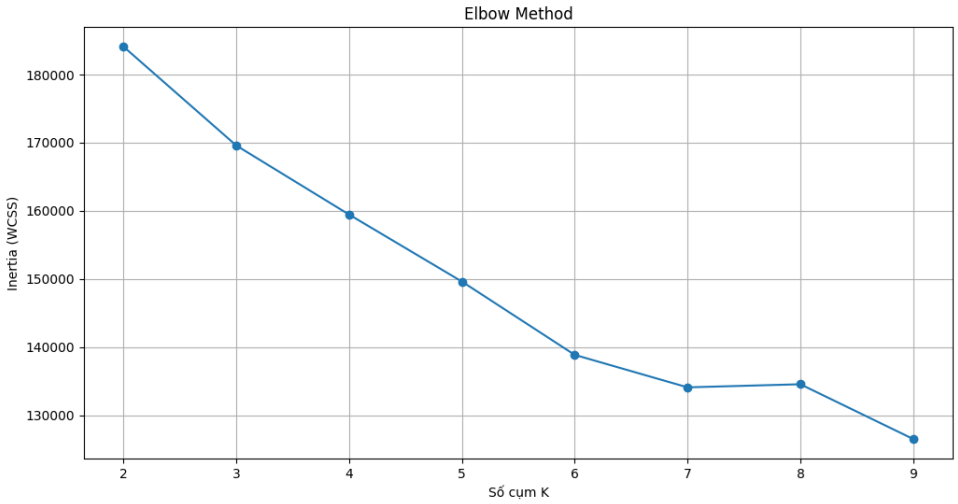
trong đó $\overline{y}$ là giá trị trung bình của các y_i. Giá trị R^2 nằm trong khoảng $[0, 1]$, với giá trị gần 1 cho thấy mô hình có khả năng giải thích dữ liệu tốt, trong khi giá trị gần 0 cho thấy mô hình kém hiệu quả.

6.2 Kết quả thực nghiệm

6.2.1 Kết quả phân cụm

Xác định số lượng cụm tối ưu (K):

Để xác định số lượng cụm tối ưu cho bài toán phân cụm, phương pháp *Elbow* đã được áp dụng lên tập dữ liệu. Hình 10 minh họa kết quả thu được từ phương pháp này.

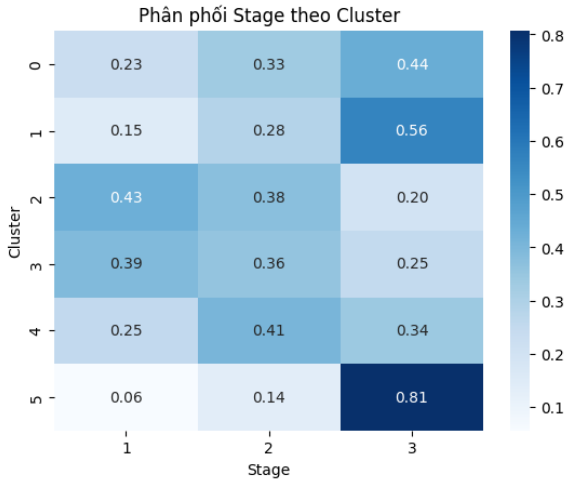


Hình 10: Biểu đồ phương pháp Elbow

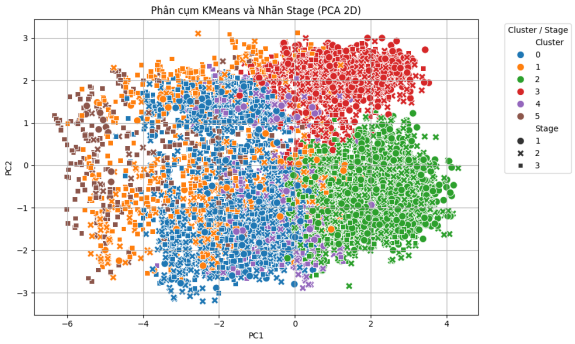
Từ biểu đồ, có thể quan sát thấy rằng điểm gãy (elbow point) xuất hiện rõ rệt tại $K = 6$, tức là sau ngưỡng này, mức giảm của quán tính nội cụm (within-cluster inertia) không còn đáng kể. Do đó, giá trị $K = 6$ được lựa chọn là số lượng cụm tối ưu để triển khai thuật toán KMeans trong các bước tiếp theo.

Kết quả phân cụm với $K = 6$:

Thuật toán KMeans với số cụm $K = 6$ được áp dụng nhằm khảo sát khả năng phân tách dữ liệu dựa trên các đặc trưng đã trích xuất. Hình 11 thể hiện biểu đồ nhiệt mô tả phân phối tương đối của nhãn `Stage` trong từng cụm, qua đó phản ánh mức độ đồng nhất nội tại của mỗi nhóm. Trong khi đó, Hình 12 minh họa kết quả phân cụm trên không gian hai chiều sau khi giảm chiều bằng PCA, giúp trực quan hóa mối liên hệ giữa cụm được gán và nhãn thực tế.



Hình 11: Biểu đồ nhiệt phân phối Stage theo Cluster



Hình 12: Biểu đồ phân tán với phân cụm KMeans và nhãn Stage

Để đánh giá chất lượng phân cụm, chỉ số Silhouette Score được tính toán và đạt giá trị 0.1299. Đây là mức thấp, cho thấy các cụm chưa thực sự tách biệt rõ ràng và có sự chồng lấn đáng kể giữa các nhóm.

Phân tích chi tiết kết quả phân cụm cho thấy:

- Cụm thứ 6 thể hiện tính thuần nhất cao, với 81% số mẫu thuộc về `Stage` 3, phản ánh khả năng nhận diện tương đối tốt nhóm dữ liệu này.
- Một số cụm khác, như cụm thứ 2, cũng cho thấy sự nghiêng về một nhãn `Stage` nhất định, tuy chưa đạt mức độ phân biệt tuyệt đối.
- Ngược lại, nhiều cụm vẫn cho thấy sự chồng lấn đáng kể, đặc biệt giữa `Stage` 1 và `Stage` 2, cho thấy các đặc trưng hiện tại chưa đủ để phân tách rõ ràng giữa các giai đoạn đầu. Điều này có thể xuất phát từ bản chất dữ liệu hoặc do đặc trưng trích xuất chưa mang tính phân biệt cao.

Nhận định:

Mặc dù phương pháp KMeans Clustering đã được triển khai và lựa chọn số lượng cụm tối ưu một cách có hệ thống, kết quả cho thấy dữ liệu không có xu hướng phân cụm rõ ràng theo các giai đoạn của bệnh (`Stage`). Điều này cho thấy rằng các đặc trưng hiện tại trong tập dữ liệu chưa đủ mạnh để phân tách các nhóm một cách tự nhiên bằng phương pháp phân cụm không giám sát. Do đó, việc áp dụng các phương pháp học có giám sát có thể là hướng tiếp cận phù hợp hơn cho bài toán phân loại giai đoạn xơ gan.

6.2.2 Kết quả mô hình KNN

Để tìm bộ tham số tối ưu cho mô hình K-Nearest Neighbors (KNN), chúng tôi sử dụng kỹ thuật GridSearchCV với 5-fold cross-validation. Lưới tham số được thiết lập để thử nghiệm các giá trị k (số láng giềng) từ tập $\{3, 5, 7, 9, 11, 13, 15, 17, 19\}$, cố định khoảng cách Manhattan và trọng số láng giềng theo khoảng cách 'distance'. Tổng cộng có 9 tổ hợp tham số, với 45 lần huấn luyện (9 tổ hợp $\times$ 5 fold).

Bảng 5 trình bày kết quả GridSearchCV, bao gồm độ chính xác trung bình, độ lệch chuẩn, và thứ hạng của từng tổ hợp.

Bảng 5: Kết quả GridSearchCV cho mô hình KNN

k	Thứ hạng	Train Accuracy	Test Accuracy
7	1	0.9912	0.7894
5	2	0.9912	0.7925
9	3	0.9912	0.7930
11	4	0.9912	0.7920
3	5	0.9912	0.7915
13	6	0.9912	0.7884
15	7	0.9912	0.7801
17	8	0.9912	0.7853
19	9	0.9912	0.7827

Theo kết quả, bộ tham số được chọn là $k = 7$, weights='distance', metric='manhattan'.

Bảng 6: Kết quả hiệu suất mô hình KNN trên dữ liệu gốc và dữ liệu giảm chiều

Tỷ lệ train:test	**Train Accuracy**	**Test Accuracy**	**Recall**	**F1-score**
Dữ liệu gốc				
8:2	0.9912	0.7894	0.79	0.79
7:3	0.9918	0.7994	0.80	0.80
6:4	0.9939	0.7884	0.79	0.79
Dữ liệu giảm chiều (PCA)				
8:2	0.9912	0.7173	0.72	0.72
7:3	0.9918	0.7230	0.72	0.72
6:4	0.9939	0.7041	0.70	0.70

Mô hình KNN được đánh giá trên dữ liệu gốc và dữ liệu giảm chiều (PCA) với tỷ lệ huấn luyện/kiểm tra 6:4, 7:3, và 8:2. Bảng 6 trình bày các chỉ số: Test Accuracy, Recall macro, và F1-score macro.

1. So sánh hiệu suất

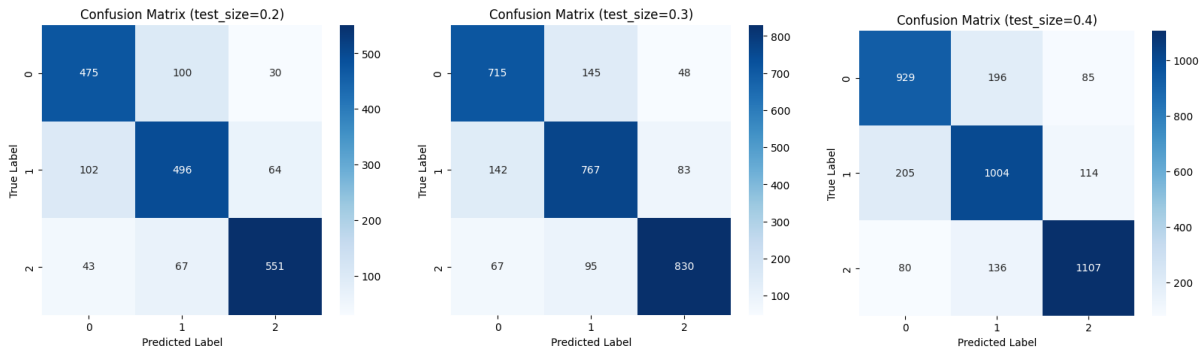
- So sánh dữ liệu gốc và PCA:
  - Dữ liệu gốc: Test Accuracy từ 78.84% (6:4) đến 79.94% (7:3), Train Accuracy >99%, Recall và F1-score 0.79–0.80, hiệu suất ổn định.
  - Dữ liệu PCA: Test Accuracy giảm còn 70.41%–72.30%, Train Accuracy >99%, Recall và F1-score 0.65–0.78.
  - Nhận xét: Dữ liệu gốc vượt PCA 7–9% do PCA làm mất đặc trưng quan trọng cho KNN.
- Ảnh hưởng tỷ lệ test size:
  - Dữ liệu gốc: Tối ưu ở 7:3 (79.94%), giảm nhẹ ở 6:4 (78.84%), chênh lệch 1%.
  - Dữ liệu PCA: Tăng từ 70.41% (6:4) lên 72.30% (8:2), nhưng vẫn thấp hơn gốc.
  - Nhận xét: 7:3 và 8:2 hiệu quả hơn nhờ dữ liệu huấn luyện lớn, nhưng PCA hạn chế cải thiện do mất thông tin.

2. Đánh giá và cải thiện overfitting

Ta thấy Train Accuracy >99%, Test Accuracy 70–80%, chênh lệch 20–25%, cho thấy nhạy cảm với nhiễu khi k nhỏ. Nguyên nhân là KNN phụ thuộc vào khoảng cách, dễ bị ảnh hưởng bởi ngoại lai khi k thấp. Ta có thể tăng k để giảm nhiễu, nhưng không hiệu quả do Train Accuracy không thay đổi (xem 5).

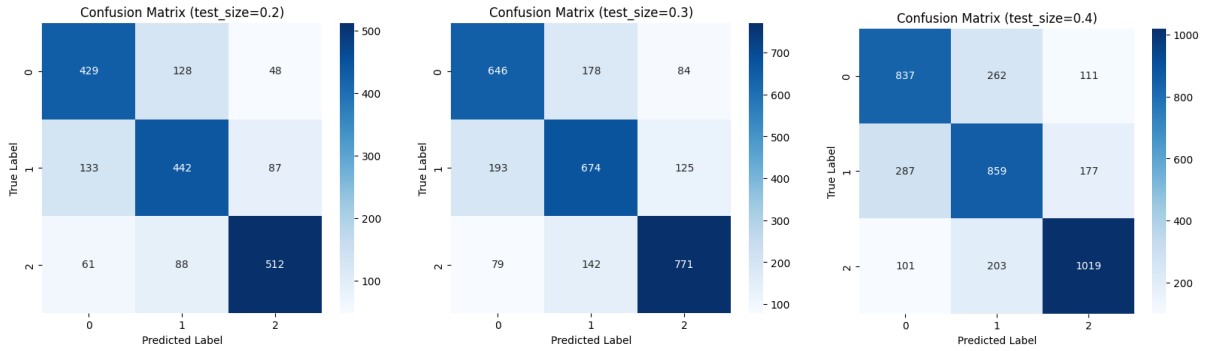
3. Trực quan hóa và đánh giá (Confusion Matrix)

Các ma trận nhầm lẫn được trình bày trong Hình 13 và 14



(a) Tỉ lệ 8:2 **(b)** Tỉ lệ 7:3 **(c)** Tỉ lệ 6:4

Hình 13: Các ma trận nhầm lẫn cho dữ liệu gốc với ba tỷ lệ train/test khác nhau



(a) Tỉ lệ 8:2 **(b)** Tỉ lệ 7:3 **(c)** Tỉ lệ 6:4

Hình 14: Các ma trận nhầm lẫn cho dữ liệu giảm chiều với ba tỷ lệ train/test khác nhau

Đánh giá:

- *Dữ liệu gốc:* Lớp 3 có Recall cao (0.83–0.84), nhưng lớp 1 và 2 nhầm lẫn nhiều (Recall 0.75–0.79) do chồng lấn đặc trưng, đặc biệt ở tỷ lệ 6:4.
- *Dữ liệu PCA:* Hiệu suất giảm, với lớp 2 thấp nhất (Recall 0.65–0.68), nhầm lẫn giữa lớp 1 và 2 tăng do mất thông tin quan trọng.

4. Đánh giá tổng quan và nhận định

- *Phù hợp:* KNN hoạt động tốt trên dữ liệu gốc (Test Accuracy 78.84–79.94%) nhờ giữ nguyên đặc trưng, nhưng kém hiệu quả trên dữ liệu PCA (70.41–72.30%) do mất thông tin. Lớp 3 dự đoán tốt (F1-score 0.77–0.85), nhưng lớp 1 và 2 dễ nhầm lẫn (F1-score 0.69–0.78) do chồng lấn đặc trưng.
- *Không phù hợp:* KNN trở nên nhạy cảm với nhiễu và ngoại lai khi k nhỏ, dẫn đến hiệu suất thấp trên tập kiểm tra (chênh lệch Train-Test 20–25%), đặc biệt khi PCA làm tăng nhiễu do giảm chiều.
- *Nhận xét:* KNN phù hợp với dữ liệu gốc có đặc trưng rõ ràng, nhưng cần điều chỉnh k và xử lý nhiễu để cải thiện tổng quát hóa. Tỷ lệ 7:3 cho kết quả tốt nhất, nhưng ảnh hưởng của test size không lớn.

6.2.3 Kết quả mô hình MLP

Với mô hình Multi-Layer Perceptron (MLP), quá trình lựa chọn siêu tham số được thực hiện bằng kỹ thuật tìm kiếm trên lưới (GridSearchCV) nhằm xác định

tổ hợp tham số tối ưu cho mô hình. Trong quá trình tìm kiếm, một số tham số được giữ cố định như sau:

- `activation` = 'relu'
- `solver` = 'adam'
- `max_iter` = 1000

Lưới tìm kiếm được thiết lập để thử nghiệm các giá trị khác nhau cho ba siêu tham số quan trọng:

- **Cấu trúc tầng ẩn (hidden_layer_sizes)**: (50,), (100,), (200,), (32,), (64,), (128,), (32, 32), (64, 64), (128, 128)
- **Hệ số điều chuẩn (alpha)**: $1e^{-5}$, $1e^{-4}$, $1e^{-3}$, $1e^{-2}$, $1e^{-1}$
- **Tốc độ học ban đầu (learning_rate_init)**: 0.001, 0.01, 0.1

Bảng 7 trình bày 5 cấu hình siêu tham số có độ chính xác trên tập kiểm tra cao nhất, được sắp xếp theo thứ tự giảm dần theo *Accuracy Test*.

Bảng 7: Kết quả huấn luyện MLP với các tổ hợp tham số khác nhau

Alpha	HLSs	LRI	Accuracy Train	Accuracy Test
0.10000	(128, 128)	0.001	0.9756	0.7552
0.00001	(128, 128)	0.010	0.9778	0.7427
0.00001	(128, 128)	0.001	0.9840	0.7422
0.00010	(128, 128)	0.010	0.9681	0.7407
0.00100	(200,)	0.001	0.9524	0.7381

Trong số các cấu hình trên, tổ hợp tham số gồm *alpha = 0.1*, *hidden_layer_sizes = (128, 128)* và *learning_rate_init = 0.001* đạt độ chính xác kiểm tra cao nhất. Tuy nhiên, sự chênh lệch đáng kể giữa độ chính xác train và test cho thấy mô hình có dấu hiệu bị overfitting.

Để giảm bớt tình trạng này, bộ tham số được chọn là:

- `hidden_layer_sizes` = (100,)
- `alpha` = 0.0001

- `learning_rate_init` = 0.001

Cấu hình này giúp giảm độ phức tạp của mô hình bằng cách sử dụng một tầng ẩn duy nhất và hệ số điều chuẩn nhỏ, từ đó hạn chế phần nào hiện tượng quá khớp.

Bảng 8 trình bày hiệu suất mô hình trên các tỷ lệ chia tập huấn luyện – kiểm tra khác nhau, được đánh giá trên cả dữ liệu gốc và dữ liệu đã giảm chiều bằng PCA.

Bảng 8: Kết quả hiệu suất mô hình MLP trên dữ liệu gốc và dữ liệu giảm chiều

Train:test	**Test Accu.**	**Train Accu.**	**Precision**	**Recall**	**F1-score**
Dữ liệu gốc					
8:2	0.7007	0.8688	0.7	0.7	0.7
7:3	0.7168	0.8711	0.72	0.72	0.72
6:4	0.6948	0.9092	0.69	0.69	0.69
Dữ liệu giảm chiều (PCA)					
8:2	0.6955	0.8248	0.69	0.69	0.69
7:3	0.6974	0.8359	0.70	0.70	0.70
6:4	0.6914	0.8439	0.69	0.69	0.69

1. So sánh hiệu suất giữa các trường hợp

Dữ liệu gốc

Khi huấn luyện mô hình MLP trên tập dữ liệu gốc, với các cách chia khác nhau, ta thu được kết quả sau:

- *Hiệu suất:* Test Accuraccy giao động từ 69,48% đến 71.68% trong khi Training Accuracy luôn cao trên 84.39%.
- *Nhận xét:*
  - Mô hình có xu hướng overfitting khi tập huấn luyện giảm (tỷ lệ 6:4): độ chính xác huấn luyện rất cao, nhưng độ chính xác kiểm thử không tăng theo mà thậm chí giảm.
  - Sự chênh lệch lớn giữa độ chính xác huấn luyện và kiểm thử (đặc biệt ở tỷ lệ 6:4 với chênh lệch hơn 21%)

Dữ liệu giảm chiều

Sau khi áp dụng phương pháp PCA để giảm chiều dữ liệu, các kết quả thu được như sau:

- *Hiệu suất:* Test accuracy có xu hướng giảm, giao động trong khoảng 69.14% - 69.74%, tương tự training accuracy cũng cho thấy mức độ giảm đáng kể khi nằm trong khoảng 82.48% - 84.39%.
- *Nhận xét:*
  - Mặc dù hiệu suất tổng thể có phần giảm nhẹ so với mô hình huấn luyện trên dữ liệu gốc, sự khác biệt giữa độ chính xác huấn luyện và kiểm thử được thu hẹp đáng kể.
  - Điều này cho thấy việc áp dụng PCA giúp mô hình giảm overfitting và cải thiện tính tổng quát hóa, dù đánh đổi một phần nhỏ về độ chính xác.

2. Đánh giá tổng quát

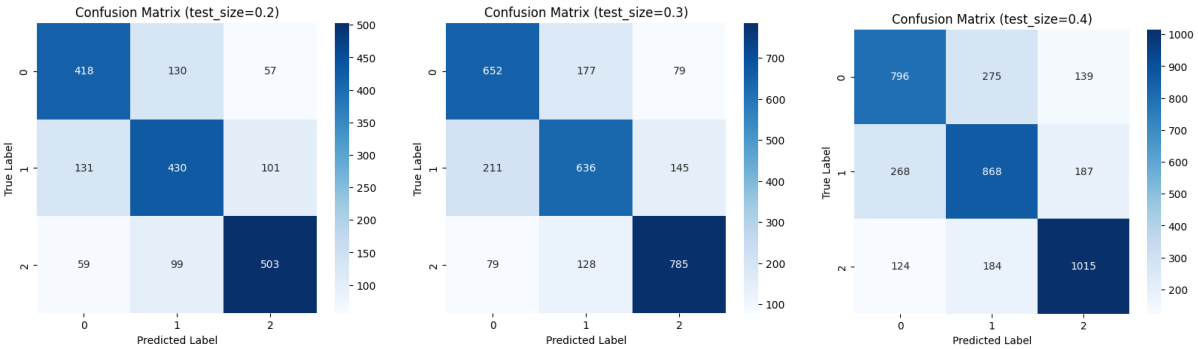
Dữ liệu	Test accuracy cao nhất	Overfitting
Data gốc	0.7168 (tỷ lệ 7:3)	Có, rõ rệt ở tỷ lệ 6:4
Data giảm chiều (PCA)	0.6974 (tỷ lệ 7:3)	Ít hơn

Bảng 9: Đánh giá tổng quát hiệu suất mô hình MLP

Nhận xét: Dữ liệu gốc mang lại hiệu suất cao hơn, nhưng dễ dẫn đến overfitting, đặc biệt khi tỷ lệ train giảm. PCA giúp mô hình ổn định hơn và hạn chế overfitting, dù phải đánh đổi một phần nhỏ về độ chính xác.

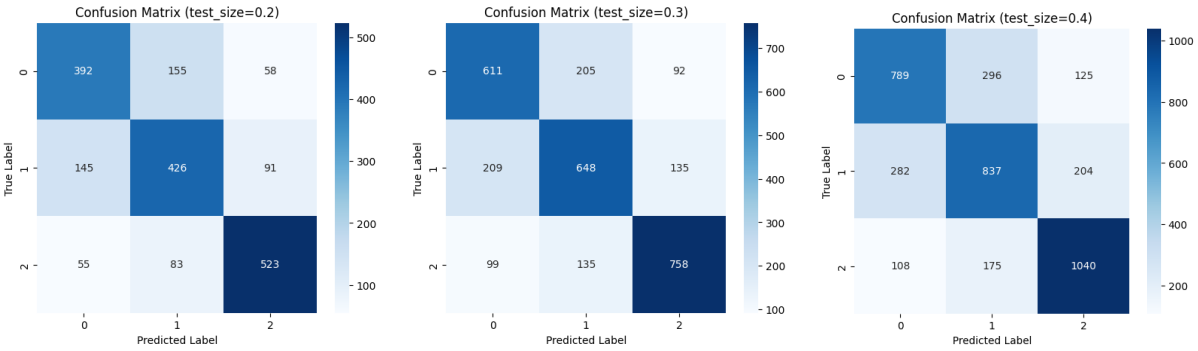
3. Trực quan hóa và đánh giá

Các hình 15 và 16 minh họa phân bố dự đoán trên ba lớp:



(a) Tỉ lệ 8:2 **(b)** Tỉ lệ 7:3 **(c)** Tỉ lệ 6:4

Hình 15: Ma trận nhầm lẫn cho dữ liệu gốc với tỷ lệ train/test khác nhau



(a) Tỉ lệ 8:2 **(b)** Tỉ lệ 7:3 **(c)** Tỉ lệ 6:4

Hình 16: Ma trận nhầm lẫn cho dữ liệu giảm chiều với tỷ lệ train/test khác nhau

Đánh giá:

- Lớp thứ ba luôn có tỷ lệ dự đoán đúng cao nhất trên cả hai loại dữ liệu, cho thấy đặc trưng phân biệt tốt hơn.
- Nhầm lẫn chủ yếu xảy ra giữa lớp 0 và lớp 1, đặc biệt rõ khi giảm tỷ lệ huấn luyện (6:4).
- Trên dữ liệu gốc, tuy độ chính xác cao hơn, nhưng tần suất nhầm lẫn cũng lớn hơn khi test size tăng.
- Dữ liệu PCA cho dự đoán ổn định hơn, tỉ lệ nhầm lẫn không biến động nhiều giữa các tỷ lệ train/test.

6.2.4 Kết quả mô hình SVM

Tương tự như hai phương pháp KNN và MLP, quá trình lựa chọn tham số cũng được thực hiện bằng kỹ thuật tìm kiếm trên lưới (GridSearchCV) nhằm xác định tập hợp các tham số mang lại kết quả tối ưu nhất cho mô hình. Trong quá trình huấn luyện mô hình, một số tham số được giữ cố định như sau:

- `max_iter` = 10000
- `cv` = 5
- `scoring` = accuracy
- `njobs` = -1

Giải thích ý nghĩa từng tham số và giá trị đi kèm:

- `max_iter`: Số lần lặp tối đa của thuật toán. 10000 là con số đủ lớn giúp đảm bảo thuật toán có đủ thời gian để hội tụ tìm ra tham số tối ưu.
- `cv` : Số lượng fold (k-fold cross-validation) được sử dụng trong GridSearchCV.

  GridSearchCV chia tập dữ liệu huấn luyện thành 5 phần (fold). Trong mỗi lần thử, 4 phần được dùng để huấn luyện và 1 phần để kiểm tra. Quá trình này lặp lại 5 lần, mỗi lần một phần khác nhau làm tập kiểm tra.

  Kết quả trung bình từ 5 lần thử này được dùng để đánh giá hiệu suất của mỗi tổ hợp tham số trong param_grid giúp đảm bảo rằng mô hình không bị overfitting và đánh giá được hiệu suất một cách công bằng trên toàn bộ dữ liệu.
- `scoring`: Được sử dụng để đánh giá hiệu suất của mô hình thông quan chỉ số accuracy
- `n_jobs`: là tham số kiểm soát số lượng luồng (threads) hoặc tiến trình (processes) được sử dụng để chạy song song trong GridSearchCV. Giá trị -1 ở đây tức nghĩa là sử dụng tất cả các CPU core có sẵn trên máy tính để thực hiện tính toán song song nhằm tăng tốc độ tính toán lên nhanh hơn.

GridSearchCV được thiết lập các giá trị khác nhau cho các tham số quan trọng trong từng loại SVM, cụ thể như sau:

- **param_grid_linear (Hard/Soft Margin SVM)**: 'C': [0.01, 0.1, 1, 10, 100, 1000]
- **param_grid_linsvc (Multi-Class SVM)**: 'C': [0.01, 0.1, 1, 10, 100]
- **param_grid_poly (Polynomial kernel SVM)**: 'C': [0.1, 1, 10]; 'degree': [2, 3, 4]; 'gamma': ['scale', 0.01, 0.1, 1]; 'coef0': [0.0, 0.5, 1.0]
- **param_grid_sigmoid (Sigmoid kernel SVM)**: 'C': [0.1, 1, 10]; 'gamma': [0.001, 0.01, 0.1]; 'coef0': [0.0, 0.5, 1.0]
- **param_grid_rbf (RBF kernel SVM)**: 'C': [0.1, 1, 10, 100]; 'gamma': [0.001, 0.01, 0.1, 1]

Dưới đây là kết quả thu được sau khi huấn luyện mô hình SVM trên dữ liệu gốc và dữ liệu giảm chiều (sử dụng PCA) với các tỷ lệ chia tập huấn luyện/kiểm

tra là 6:4, 7:3, và 8:2. Bảng trình bày bao gồm các chỉ số hiệu suất quan trọng: độ chính xác trên tập kiểm tra (Test Accuracy), độ nhạy trung bình (Recall), và F1-score trung bình (F1-score).

Bảng 10: Kết quả mô hình SVM với tỉ lệ chia 4:1 trên dữ liệu gốc và dữ liệu PCA

Model	**Train Acc**	**Test Acc**	**Precision**	**Recall**	**F1-score**
Dữ liệu gốc					
Hard/Soft Margin	0.5653	0.5555	0.5500	0.5572	0.5489
Multi-class	0.5660	0.5493	0.5412	0.5499	0.5401
Polynomial Kernel	0.8611	0.7158	0.7178	0.7147	0.7158
RBF Kernel	0.9803	0.7422	0.7426	0.7409	0.7412
Sigmoid Kernel	0.5636	0.5539	0.5487	0.5558	0.5472
Dữ liệu giảm chiều (PCA)					
Hard/Soft Margin	0.5575	0.5420	0.5389	0.5436	0.5375
Multi-class	0.5506	0.5405	0.5330	0.5413	0.5311
Polynomial Kernel	0.8690	0.6883	0.6911	0.6868	0.6883
RBF Kernel	0.8937	0.7272	0.7283	0.7261	0.7260
Sigmoid Kernel	0.5570	0.5420	0.5389	0.5436	0.5376

Bảng 11: Kết quả mô hình SVM với tỉ lệ chia 7:3 trên dữ liệu gốc và dữ liệu PCA

Model	**Train Acc**	**Test Acc**	**Precision**	**Recall**	**F1-score**
Dữ liệu gốc					
Hard/Soft Margin	0.5610	0.5533	0.5498	0.5551	0.5463
Multi-class	0.5598	0.5558	0.5477	0.5557	0.5450
Polynomial Kernel	0.8724	0.7133	0.7127	0.7116	0.7120
RBF Kernel	0.9803	0.7317	0.7310	0.7307	0.7308
Sigmoid Kernel	0.5611	0.5536	0.5502	0.5554	0.5467
Dữ liệu giảm chiều (PCA)					
Hard/Soft Margin	0.5539	0.5415	0.5366	0.5431	0.5348
Multi-class	0.5514	0.5450	0.5373	0.5455	0.5350
Polynomial Kernel	0.7992	0.6822	0.6841	0.6806	0.6819
RBF Kernel	0.8963	0.7244	0.7251	0.7236	0.7242
Sigmoid Kernel	0.5514	0.5432	0.5392	0.5447	0.5373

Bảng 12: Kết quả mô hình SVM với tỉ lệ chia 3:2 trên dữ liệu gốc và dữ liệu PCA

Model	**Train Acc**	**Test Acc**	**Precision**	**Recall**	**F1-score**
Dữ liệu gốc					
Hard/Soft Margin	0.5629	0.5511	0.5483	0.5532	0.5436
Multi-class	0.5570	0.5609	0.5544	0.5618	0.5514
Polynomial Kernel	0.9321	0.6994	0.7007	0.6985	0.6992
RBF Kernel	0.9435	0.7326	0.7321	0.7317	0.7319
Sigmoid Kernel	0.5629	0.5511	0.5484	0.5533	0.5435
Dữ liệu giảm chiều (PCA)					
Hard/Soft Margin	0.5513	0.5423	0.5369	0.5437	0.5353
Multi-class	0.5535	0.5458	0.5401	0.5487	0.5358
Polynomial Kernel	0.8793	0.6712	0.6745	0.6702	0.6716
RBF Kernel	0.9030	0.7098	0.7107	0.7091	0.7096
Sigmoid Kernel	0.5514	0.5384	0.5344	0.5402	0.5325

Dưới đây là phân tích hiệu suất của các mô hình SVM trên cả dữ liệu gốc và dữ liệu sau khi giảm chiều bằng PCA, ứng với các tỉ lệ chia tập dữ liệu khác nhau:

1. Trường hợp tỉ lệ chia 4:1

- Trên dữ liệu gốc, hai mô hình *Polynomial Kernel SVM* và *RBF Kernel SVM* đạt hiệu suất cao nhất với F1-score lần lượt là 0.7158 và 0.7412.
- Các mô hình đơn giản như *Hard/Soft Margin SVM* và *Sigmoid Kernel SVM* cho kết quả khá thấp (F1-score $\sim$ 0.54).
- Sau khi giảm chiều bằng PCA, hiệu suất các mô hình chủ yếu giảm nhẹ, đặc biệt là *Polynomial Kernel* và *RBF Kernel*.
- Mô hình *RBF Kernel* vẫn giữ được hiệu suất tốt nhất với F1-score = 0.7260.

2. Trường hợp tỉ lệ chia 7:3

- Trên dữ liệu gốc, *RBF Kernel* và *Polynomial Kernel* tiếp tục vượt trội với F1-score lần lượt là 0.7308 và 0.7120.
- Sau khi áp dụng PCA, hiệu suất của các mô hình phức tạp này giảm nhẹ nhưng vẫn tốt hơn các mô hình còn lại.
- *RBF Kernel* sau PCA vẫn đạt F1-score cao nhất (0.7242), cho thấy tính ổn định cao.

3. Trường hợp tỉ lệ chia 3:2

- *Polynomial Kernel* và *RBF Kernel* vẫn duy trì hiệu suất cao trên dữ liệu gốc với F1-score lần lượt là 0.6992 và 0.7319.
- Sau PCA, hiệu suất của *Polynomial Kernel* giảm nhiều hơn (còn 0.6712), trong khi *RBF Kernel* giữ được F1-score ở mức cao (0.7096).
- Các mô hình đơn giản vẫn cho kết quả thấp và không có cải thiện rõ rệt sau PCA.

Kết luận chung:

- Hiệu suất tổng thể: *RBF Kernel SVM* và *Polynomial Kernel SVM* có xu hướng đạt độ chính xác cao nhất trên tập huấn luyện, nhưng bị overfitting nghiêm trọng trên tập kiểm tra, đặc biệt với tỷ lệ 3:2. *Sigmoid Kernel SVM* cho kết quả thấp nhất, cho thấy nó không phù hợp với dữ liệu này. *Hard/Soft Margin SVM* và *Multi-class SVM* ổn định hơn nhưng hiệu suất thấp, phù hợp với dữ liệu tuyến tính hơn.
- Tác động của PCA: Giảm chiều bằng PCA không cải thiện hiệu suất mà còn làm giảm độ chính xác trên tập kiểm tra, đặc biệt với kernel SVM. Điều này cho thấy dữ liệu có thể phi tuyến mạnh, và PCA (tuyến tính) làm mất thông tin quan trọng.
- Tỷ lệ dữ liệu: Tăng tỷ lệ huấn luyện (70-30) cải thiện nhẹ hiệu suất trên tập kiểm tra so với 60-40, nhưng overfitting vẫn xảy ra với kernel SVM. Tỷ lệ 3:2 (60-40) không cho kết quả tốt hơn, cho thấy lượng dữ liệu huấn luyện từng đó là chưa đủ để mô hình có thể học và cho ra kết quả tốt.

6.2.5 So sánh các mô hình phân loại

Để đánh giá hiệu quả của các mô hình phân loại (KNN, MLP, và SVM), chúng tôi so sánh hiệu suất dựa trên Test Accuracy, Recall, và F1-score trên dữ liệu gốc và dữ liệu giảm chiều (PCA) với các tỷ lệ train:test khác nhau (6:4, 7:3, 8:2). Bảng 13 tóm tắt kết quả tốt nhất của từng mô hình trên các tập dữ liệu.

Bảng 13: So sánh hiệu suất tốt nhất của các mô hình phân loại

Mô hình	**Tỷ lệ**	**Dữ liệu**	**Test Accuracy**	**Recall**	**F1-score**
KNN					
KNN	7:3	Gốc	0.7994	0.80	0.80
KNN	8:2	PCA	0.7230	0.72	0.72
MLP					
MLP	7:3	Gốc	0.7168	0.72	0.72
MLP	7:3	PCA	0.6974	0.70	0.70
SVM (RBF Kernel)					
SVM (RBF)	8:2	Gốc	0.7422	0.7415	0.7412
SVM (RBF)	7:3	PCA	0.7244	0.7236	0.7242

Phân tích hiệu suất:

- **KNN**: Đạt hiệu suất cao nhất trên dữ liệu gốc (Test Accuracy 0.7994 với tỷ lệ 7:3), nhưng giảm đáng kể trên PCA (0.7230), cho thấy sự phụ thuộc vào đặc trưng gốc. KNN dễ bị nhiễu khi k nhỏ, dẫn đến chênh lệch Train-Test lớn (20–25%).
- **MLP**: Hiệu suất trên dữ liệu gốc đạt 0.7168 (tỷ lệ 7:3). Accuracy không giảm quá nhiều khi chuyển sang dữ liệu giảm chiều.
- **SVM (RBF Kernel)**: Vượt trội với Test Accuracy 0.7422 trên dữ liệu gốc (tỷ lệ 8:2) và 0.7244 trên PCA (tỷ lệ 7:3). RBF Kernel xử lý tốt dữ liệu phi tuyến, nhưng vẫn bị overfitting (Train Accuracy $\sim$0.98), đặc biệt trên dữ liệu gốc.

So sánh tổng quan:

- KNN cho hiệu suất cao nhất trên dữ liệu gốc (0.7994), phù hợp khi giữ nguyên đặc trưng, nhưng nhạy với nhiễu.

- SVM (RBF) đạt cân bằng tốt giữa hiệu suất (0.7422) và ổn định, dù bị ảnh hưởng bởi overfitting.
- MLP có hiệu suất trung bình (0.7168), nhưng PCA giúp giảm overfitting, dù mất độ chính xác.
- Dữ liệu gốc luôn vượt trội hơn PCA (7–9%), do PCA làm mất thông tin quan trọng, đặc biệt với KNN và SVM.

Khuyến nghị:

- Ưu tiên KNN trên dữ liệu gốc với tỷ lệ 7:3 để đạt hiệu suất cao nhất.
- Điều chỉnh tham số (tăng k cho KNN, giảm độ phức tạp cho MLP/SVM) để giảm overfitting.
- Sử dụng SVM (RBF) nếu cần mô hình ổn định trên cả dữ liệu gốc và PCA.

6.2.6 Kết quả mô hình hồi quy

1. Phân lớp và mô hình hồi quy

Nhóm lựa chọn phân lớp `Stage 3` từ bài toán phân loại ban đầu với nhãn `Stage` (gồm các lớp `Stage 1`, `Stage 2`, `Stage 3`). Mô hình phân loại được chọn là `MLPClassifier` (Multilayer Perceptron).

Sau khi huấn luyện `MLPClassifier` trên tập dữ liệu gốc, ta sử dụng hàm `predict_proba` để lấy xác suất dự đoán cho lớp *3* (cột thứ 3 trong ma trận xác suất), chuyển sang bài toán hồi quy với đầu ra là giá trị xác suất liên tục trong khoảng $[0, 1]$.

Hai mô hình hồi quy được áp dụng là Linear Regression và Random Forest Regressor (với `n_estimators=300`, `max_depth=20`, `max_features='sqrt'`, `min_samples_split=2`, `min_samples_leaf=1`, `random_state=42`) và thực hiện trên cả dữ liệu gốc và dữ liệu giảm chiều (PCA, giữ lại 1/3 số chiều ban đầu).

2. Kết quả thực nghiệm

Bảng 14 trình bày kết quả của các mô hình hồi quy trên dữ liệu gốc và dữ liệu giảm chiều (PCA) với các tỷ lệ train:test khác nhau. Các chỉ số đánh giá bao gồm Mean Squared Error (MSE), Mean Absolute Error (MAE), và hệ số xác định R^2 trên tập huấn luyện (Train) và tập kiểm tra (Test).

Bảng 14: Kết quả thực nghiệm của các mô hình hồi quy

Dữ liệu	**Test Size**	**Train MSE**	**Test MSE**	**Train MAE**	**Test MAE**	**Train R^2**	**Test R^2**
Linear Regression							
Dữ liệu gốc	0.2	0.099325	0.099687	0.254190	0.254259	0.381988	0.368721
	0.3	0.100664	0.098502	0.256175	0.252656	0.377970	0.379871
	0.4	0.108222	0.108945	0.265176	0.267342	0.360465	0.356603
Dữ liệu PCA	0.2	0.111194	0.110018	0.273337	0.271526	0.308136	0.303299
	0.3	0.112637	0.108531	0.276745	0.270225	0.303990	0.316729
	0.4	0.120141	0.120770	0.284986	0.286775	0.290027	0.286765
Random Forest Regressor							
Dữ liệu gốc	0.2	0.003616	0.035061	0.042156	0.125862	0.977500	0.777973
	0.3	0.004057	0.035984	0.045247	0.128902	0.974933	0.773456
	0.4	0.004662	0.045954	0.048574	0.146723	0.972452	0.728609
Dữ liệu PCA	0.2	0.010532	0.065828	0.076396	0.190889	0.934468	0.583137
	0.3	0.011547	0.065751	0.080593	0.191811	0.928646	0.586054
	0.4	0.012640	0.077417	0.084568	0.210035	0.925303	0.542799

3. Đánh giá và so sánh

- **So sánh giữa Linear Regression và Random Forest:** Random Forest vượt trội hơn Linear Regression (Test R^2 0.777973 vs 0.368721 ở test size 0.2), nhờ khả năng xử lý dữ liệu phi tuyến tính. Tuy nhiên, Random Forest vẫn dễ overfitting, với Train R^2 gần 0.98, trong khi Linear Regression ổn định hơn nhưng hiệu suất thấp do giả định tuyến tính.
- **So sánh dữ liệu gốc và PCA:** Dữ liệu gốc cho hiệu suất tốt hơn PCA với Random Forest, với chênh lệch Test R^2 khoảng 0.194–0.185 (từ 0.777973 đến 0.583137 ở test size 0.2). Với Linear Regression, chênh lệch nhỏ hơn (0.065–0.029), cho thấy PCA ảnh hưởng mạnh hơn đến Random Forest.
- **Ảnh hưởng của test size:** Khi test size tăng từ 0.2 đến 0.4, Test R^2 của Random Forest giảm từ 0.777973 xuống 0.728609 (dữ liệu gốc), do tập huấn luyện nhỏ hơn, làm giảm khả năng tổng quát hóa.

4. Giải thích nguyên nhân:

- Random Forest phù hợp hơn với dữ liệu phi tuyến tính, nhưng dễ overfitting do học quá kỹ dữ liệu huấn luyện.
- PCA làm mất đặc trưng quan trọng, ảnh hưởng mạnh đến Random Forest hơn Linear Regression.
- Linear Regression ổn định nhưng không nắm bắt được mối quan hệ phức tạp trong dữ liệu, dẫn đến hiệu suất thấp.

5. Kết luận:

Random Forest Regressor vượt trội Linear Regression trên dữ liệu gốc, nhưng bị ảnh hưởng mạnh bởi PCA và dễ overfitting. Cần điều chỉnh tham số (như giảm `max_depth` hoặc tăng `min_samples_leaf`) để giảm overfitting. Nên ưu tiên dữ liệu gốc để đạt hiệu suất tối ưu.

A Tài liệu tham khảo

- Liver Cirrhosis Stage Dataset (Kaggle)
- Website Machine Learning cơ bản
- Deep AI KhanhBlog - Random Forest
- Erwan Scornet, Giles Hooker, *Theory of Random Forests: A Review*, 2025
- New York University Slide, Lecture 11 - Decision trees
- Slide bài giảng "Decision Trees and Random Forests", Università degli Studi di Padova, Machine Learning 2021, P. Zanuttigh.