BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN-CƠ-TIN HỌC

Báo cáo bài tập lớn

Đề tài: Dự đoán khả năng mắc rối loạn tự kỷ

Sinh viên thực hiện: **Vũ Đình Hoàng - 23001881**
Nguyễn Nhật Đan - 23001861
Hoàng Mạnh Duy - 23001852

12 **tháng** 11 **năm** 2025

Mục Lục

Phần I. Giới Thiệu

1. Bối cảnh và vấn đề

Autism Spectrum Disorder (ASD) là một rối loạn phát triển thần kinh phức tạp, ảnh hưởng đến khả năng giao tiếp, tương tác xã hội và hành vi của cá nhân. Theo thống kê của Tổ chức Y tế Thế giới (WHO), ước tính 1 trong 100 trẻ em được chẩn đoán mắc ASD trên toàn cầu. Tại Việt Nam, con số này cũng đang có xu hướng gia tăng đáng kể trong thập kỷ qua.

Thách thức trong chuẩn đoán truyền thống:

- Thời gian chẩn đoán trung bình: 2-3 năm từ khi có triệu chứng đầu tiên
- Chi phí chẩn đoán cao: $2,000 - $5,000 cho một đánh giá toàn diện
- Thiếu chuyên gia: Tỷ lệ bác sĩ/bệnh nhân không cân xứng
- Chẩn đoán chủ quan: Phụ thuộc vào kinh nghiệm của chuyên gia

Cơ hội ứng dụng AI trong y tế Machine Learning mang lại tiềm năng cách mạng hóa quy trình sàng lọc ASD thông qua:

- Phân tích tự động các screening questionnaires
- Dự đoán nguy cơ với độ chính xác cao
- Giảm thời gian và chi phí sàng lọc
- Hỗ trợ quyết định cho chuyên gia y tế

2. Giải pháp đề xuất

Học máy (Machine Learning) nổi lên như một công cụ đầy tiềm năng để hỗ trợ giải quyết thách thức này. Bằng cách phân tích các bộ dữ liệu sàng lọc chứa thông tin về hành vi và đặc điểm cá nhân, các mô hình học máy có thể học các mẫu (pattern) ẩn và xây dựng các bộ dự đoán (predictor) để ước tính nguy cơ mắc ASD một cách nhanh chóng và có hệ thống.

3. Mục tiêu dự án

Mục tiêu tổng quát: Xây dựng và đánh giá một mô hình học máy có khả năng dự đoán nguy cơ mắc chứng tự kỷ dựa trên các đặc điểm được cung cấp.

Mục tiêu cụ thể: Thu thập và tiền xử lý dữ liệu về sàng lọc tự kỷ. Khám phá dữ liệu (EDA) để hiểu rõ đặc điểm và mối tương quan giữa các biến. Xây dựng và huấn luyện nhiều mô hình học máy phân loại khác nhau.

Đánh giá, so sánh hiệu suất của các mô hình và lựa chọn mô hình tối ưu nhất.

Đưa ra kết luận và đề xuất hướng phát triển trong tương lai.

Phần II. Kiến thức liên quan

1. Tổng quan về Rối loạn tự kỷ (ADS)

1.1. Định nghĩa và đặc điểm lâm sàn

Định nghĩa: Rối loạn phổ tự kỷ (Autism Spectrum Disorder - ASD) là một tình trạng rối loạn phát triển thần kinh phức tạp, được đặc trưng bởi những khiếm khuyết kéo dài trong giao tiếp xã hội và tương tác xã hội, cùng với các hành vi, sở thích hoặc hoạt động hạn hẹp, lặp đi lặp lại (APA, 2013).

Triệu chứng cốt lõi:

- **Khiếm khuyết giao tiếp xã hội:** Khó khăn trong giao tiếp bằng mắt, thiếu biểu cảm khuôn mặt phù hợp, giảm khả năng chia sẻ cảm xúc
- **Hành vi rập khuôn, lặp lại:** Cử động cơ thể định hình (vỗ tay, đung đưa), sắp xếp đồ vật theo trình tự cứng nhắc
- **Kháng cự thay đổi:** Phản ứng mạnh mẽ với những thay đổi nhỏ trong môi trường hoặc thói quen
- **Phản ứng bất thường với kích thích giác quan:** Quá nhạy cảm (hyper-sensitivity) hoặc kém nhạy cảm (hyposensitivity) với âm thanh, ánh sáng, nhiệt độ

1.2 Dịch tễ học và Tầm quan trọng của phát hiện sớm

Theo Trung tâm Kiểm soát và Phòng ngừa Dịch bệnh Hoa Kỳ (CDC, 2023), tỷ lệ mắc ASD hiện nay là **1 trong 36 trẻ em**, tăng đáng kể so với thập kỷ trước. Việc phát hiện sớm trước 3 tuổi có ý nghĩa quan trọng vì:

- **Can thiệp sớm** giúp cải thiện đáng kể kỹ năng ngôn ngữ và nhận thức
- **Tiết kiệm chi phí** điều trị lâu dài

- **Nâng cao chất lượng cuộc sống** cho trẻ và gia đình

1.3 Công cụ Sàng lọc và chuẩn đoán

Bảng câu hỏi AQ (Autism Spectrum Quotient):

- Phát triển bởi Baron-Cohen et al. (2001)
- Gồm 50 câu hỏi đánh giá 5 lĩnh vực: kỹ năng xã hội, giao tiếp, trí tưởng tượng, chú ý đến chi tiết, chuyển đổi suy nghĩ
- Điểm số từ 0-50, ngưỡng nghi ngờ ASD thường từ **32 điểm** trở lên

ADOS (Autism Diagnostic Observation Schedule):

- Công cụ "tiêu chuẩn vàng" trong chẩn đoán ASD
- Đánh giá qua quan sát hành vi
- Tốn kém thời gian và cần chuyên gia được đào tạo

2. Học máy trong Chuẩn đoán y tế

2.1 Vai trò ngày càng tăng của AI trong y tế

Học máy đang cách mạng hóa lĩnh vực chẩn đoán y tế thông qua:

Ứng dụng chính:

- **Hỗ trợ chẩn đoán:** Phân loại bệnh dựa trên triệu chứng, hình ảnh y tế và dữ liệu lâm sàng
- **Tiên lượng bệnh:** Dự đoán diễn biến, kết quả điều trị và nguy cơ tái phát
- **Phát hiện sớm:** Xác định nguy cơ bệnh ở giai đoạn tiền lâm sàng
- **Y học cá thể hóa:** Đề xuất phác đồ điều trị tối ưu cho từng bệnh nhân

2.2 Lợi ích cụ thể trong sàng lọc ASD

Ưu điểm nổi bật:

- **Tốc độ xử lý nhanh:** Giảm thời gian chẩn đoán từ vài giờ xuống vài phút
- **Tính khách quan:** Loại bỏ sai số do chủ quan của người đánh giá
- **Khả năng mở rộng:** Triển khai sàng lọc quy mô lớn với chi phí thấp
- **Xử lý đa chiều:** Phân tích đồng thời nhiều yếu tố nguy cơ

Thách thức:

- **Chất lượng dữ liệu:** Phụ thuộc vào tính đầy đủ và chính xác của dữ liệu đầu vào
- **Tính giải thích được:** Yêu cầu cao trong lĩnh vực y tế
- **Vấn đề đạo đức:** Trách nhiệm pháp lý khi sử dụng AI trong chẩn đoán

2.3 Bài toán Phân loại trong y tế

Đặc thù của dữ liệu y tế:

- **Dữ liệu không cân bằng:** Số ca bệnh thường ít hơn nhiều so với ca không bệnh
- **Cost-sensitive learning:** Chi phí cho false negative (bỏ sót bệnh) cao hơn false positive (chẩn đoán nhầm)
- **Yêu cầu độ tin cậy cao:** Ảnh hưởng trực tiếp đến sức khỏe và tính mạng
- **Tính đa ngành:** Kết hợp kiến thức y học, thống kê và khoa học máy tính

Phần III. Dữ liệu và phương pháp

1. Mô tả dữ liệu

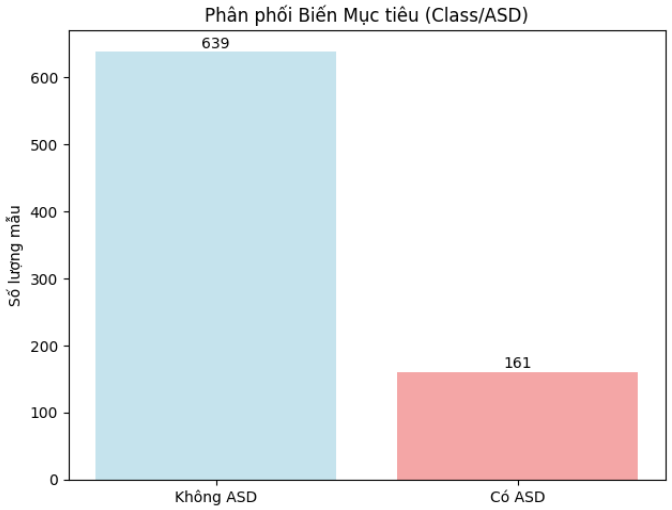
1.1 Nguồn dữ liệu và đặc điểm Dataset

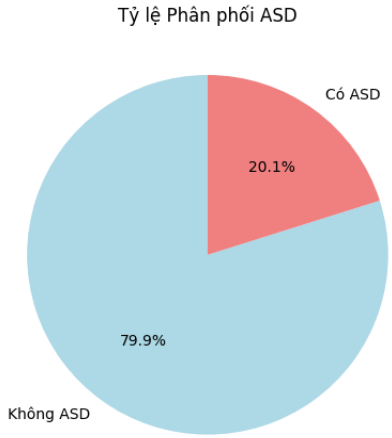
```
# đọc dữ liệu
df = pd.read_csv("../data/raw/train.csv")


# Hiển thị thông tin cơ bản về dataset
print("\n1. THÔNG TIN CƠ BẢN VỀ DATASET")
print("-" * 40)
print(f"Kích thước dataset: {df.shape}")
print(f"Số lượng mẫu (samples): {df.shape[0]}")
print(f"Số đặc trưng (features): {df.shape[1]}")
```

✓ 0.0s

```
1. THÔNG TIN CƠ BẢN VỀ DATASET
----------------------------------------
Kích thước dataset: (800, 22)
Số lượng mẫu (samples): 800
Số đặc trưng (features): 22
```

1.2 Phân tích biến mục tiêu





1.3 Phân tích cấu trúc dữ liệu

Thông tin tổng quan về dataset

```
# Xem thông tin về dataframe
df.info()
```

RangeIndex: 800 entries, 0 to 799

Data columns (total 22 columns):

#	Column	Non-Null Count	Dtype
0	ID	800 non-null	int64
1	A1_Score	800 non-null	int64
2	A2_Score	800 non-null	int64
3	A3_Score	800 non-null	int64
4	A4_Score	800 non-null	int64
5	A5_Score	800 non-null	int64
6	A6_Score	800 non-null	int64
7	A7_Score	800 non-null	int64
8	A8_Score	800 non-null	int64

```
 9   A9_Score         800 non-null    int64
 10 A10_Score         800 non-null    int64
 11 age               800 non-null    float64
 12 gender            800 non-null    object
 13 ethnicity         800 non-null    object
 14 jaundice          800 non-null    object
 15 austim            800 non-null    object
 16 contry_of_res     800 non-null    object
 17 used_app_before 800 non-null    object
 18 result            800 non-null    float64
 19 age_desc          800 non-null    object
 20 relation          800 non-null    object
 21 Class/ASD         800 non-null    int64
dtypes: float64(2), int64(12), object(8)
memory usage: 137.6+ KB
```

Phân tích các features

- AQ Score Features (10 features): ['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score', 'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score']
- Demographic Features (10 features): ['age', 'gender', 'ethnicity', 'jaundice', 'austim', 'contry_of_res', 'used_app_before', 'result', 'age_desc', 'relation']
- Target Feature: ['Class/ASD']

1.4 Phân tích thống kê mô tả

Thống kê	age	result
Số mẫu	800	800
Giá trị trung bình (Mean)	28.45	8.54
Độ lệch chuẩn (Std)	16.31	4.81
Nhỏ nhất (Min)	2.72	-6.14
Tứ phân vị 25%	17.20	5.31
Trung vị (Median)	24.85	9.61
Tứ phân vị 75%	35.87	12.51
Lớn nhất (Max)	89.46	15.85

Nhận xét:

- Tuổi (age) trải rộng từ trẻ em đến người lớn tuổi (2 → 89), phân phối lệch phải, cho thấy mẫu thu thập đa dạng.
- Kết quả test (result) dao động quanh giá trị trung bình 8.5 và không đối xứng hoàn toàn → cần chuẩn hóa (StandardScaler) trước khi đưa vào mô hình.

2. Tiền xử lý dữ liệu

2.1 Xử lý Missing Values đặc biệt hoặc dữ liệu sai, không đúng định dạng

- **Phương pháp xử lý**:
  - ✓ **Imputation**: Thay thế bằng giá trị ước lượng
  - ✓ **Deletion**: Xóa bỏ mẫu chứa missing values
  - ✓ **Indicator**: Tạo biến chỉ báo missing

Dựa vào bảng thống kê biến phân loại ta thấy ethnicity có 203 (25.4%) và relation có 40 (5.0%) **Missing Values => Xử dụng phương pháp Imputation:** Thay thế '?' bằng 'Others'.

Cùng với đó cái cột ethnicity và contry_of_res có những giá trị sai định dạng hoặc các giá trị tương tự hoặc đồng nghĩa (ví dụ: "Hong Kong" → "China", "AmericanSamoa" → "United States", "others" -> "Others", ...).

Chuyển cột age có kiểu float sang int và bỏ đi cột ID vì không làm ảnh hưởng đến dự đoán.

```
# Thay thế các giá trị sai định dạng hoặc đồng nghĩa trong các cột
df["ethnicity"] = df["ethnicity"].replace({"?": "Others", "others": "Others"})
df["relation"] = df["relation"].replace({"?": "Others"})

mapping = {
    "Viet Nam": "Vietnam",
    "AmericanSamoa": "United States",
    "Hong Kong": "China"
}
df["contry_of_res"] = df["contry_of_res"].replace(mapping)

print(df["contry_of_res"].unique())
print(df["ethnicity"].unique())
print(df["relation"].unique())
```

```
['Austria' 'India' 'United States' 'South Africa' 'Jordan'
 'United Kingdom' 'Brazil' 'New Zealand' 'Canada' 'Kazakhstan'
 'United Arab Emirates' 'Australia' 'Ukraine' 'Iraq' 'France' 'Malaysia'
 'Vietnam' 'Egypt' 'Netherlands' 'Afghanistan' 'Oman' 'Italy' 'Bahamas'
 'Saudi Arabia' 'Ireland' 'Aruba' 'Sri Lanka' 'Russia' 'Bolivia'
 'Azerbaijan' 'Armenia' 'Serbia' 'Ethiopia' 'Sweden' 'Iceland' 'China'
 'Angola' 'Germany' 'Spain' 'Tonga' 'Pakistan' 'Iran' 'Argentina' 'Japan'
 'Mexico' 'Nicaragua' 'Sierra Leone' 'Czech Republic' 'Niger' 'Romania'
 'Cyprus' 'Belgium' 'Burundi' 'Bangladesh']
['Others' 'White-European' 'Middle Eastern ' 'Pasifika' 'Black' 'Hispanic'
 'Asian' 'Turkish' 'South Asian' 'Latino']
['Self' 'Relative' 'Parent' 'Others' 'Health care professional']
```

```python
# chuyển đổi cột "age" từ kiểu dữ liệu float sang int
df["age"] = df["age"].astype(int)

# bỏ cột id
df = df.drop('ID', axis=1)
df.head(2)
```

2.2 Xử lý các giá trị ngoại lai (outliers)

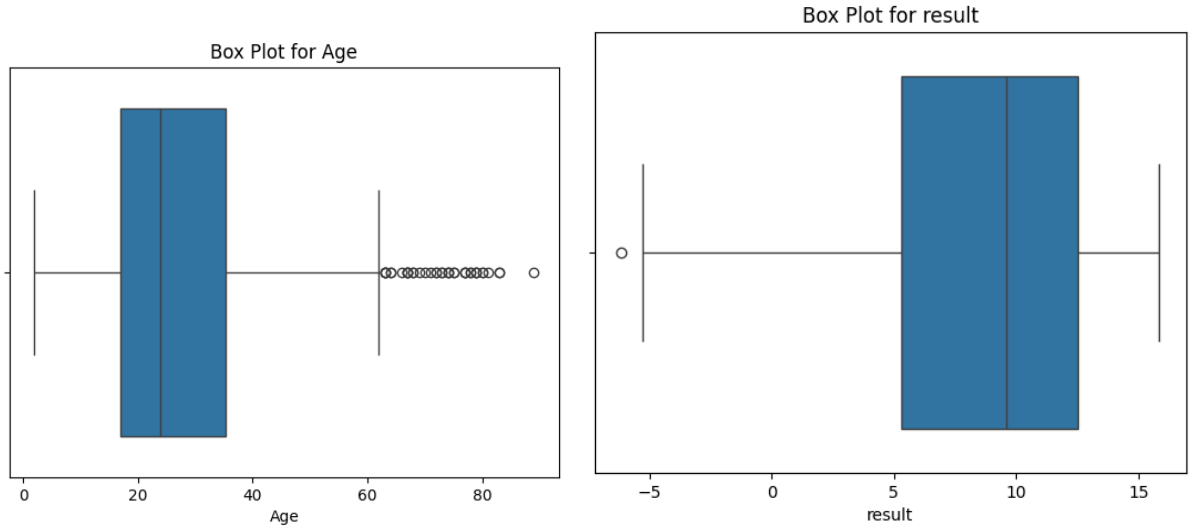
Outlier (điểm ngoại lai) là những quan sát có giá trị **khác biệt một cách bất thường** so với phần lớn dữ liệu còn lại.

Tác động đến mô hình: Bị **kéo đường hồi quy**, làm sai dự đoán (Hồi quy tuyến tính), mean bị lệch mạnh (Mean-based imputation / normalization), Khoảng cách và mặt phẳng phân cách bị méo (KNN, SVM).

Các chiến lược xử lý Outliers:

- Loại bỏ
- Giới hạn giá trị
- Biến đổi dữ liệu
- Thay giá trị …

Phân tích outliers trên tập data dựa trên biểu đồ:



Nhận xét:

- Khoảng giá trị chính (IQR) nằm chủ yếu từ khoảng **~18 đến ~40 tuổi**.
- Có **nhiều điểm nằm phía bên phải**, xấp xỉ trong khoảng **60–90 tuổi**, bị đánh dấu là **outliers**.
- Những outliers này **xuất hiện tập trung**, không rải rác đơn lẻ → cho thấy **đây có thể là nhóm người lớn tuổi hợp lệ.**
- Còn về result có **outlier rất nhỏ, không đáng kể**, lại là **số lượng rất ít** → ảnh hưởng đến mô hình **rất thấp**.
    - Biến Age và result không ảnh hưởng đáng kể đến phân phối dữ liệu, vì vậy không cần xử lý đặc biệt.

2.3 Biến đổi dữ liệu

Dựa trên kết quả phân tích dữ liệu, ta tiến hành phân loại các thuộc tính trong tập dữ liệu thành các nhóm như sau:

- **Numerical Features:**
  Bao gồm: *age, result.*
  Đây là các biến dạng số liên tục, có thể áp dụng các kỹ thuật chuẩn hóa nhằm đảm bảo phân phối dữ liệu phù hợp khi đưa vào mô hình.
- **Binary Features:**
  Bao gồm các biến: *A1_Score → A10_Score, jaundice, austim, used_app_before, Class/ASD.*
  Các biến này mang giá trị 0/1 hoặc Yes/No nên có thể đưa trực tiếp vào mô hình mà không cần mã hóa thêm.
- **Categorical Features:**
    - **Binary Categorical:** *gender* (chỉ có 2 lựa chọn).
    - **Nominal Categorical:** *ethnicity, contry_of_res, relation* (không có quan hệ thứ tự giữa các giá trị).
    - **Ordinal Categorical:** *age_desc* (có chứa thông tin về thứ tự độ tuổi).

Tiền xử lý Numerical features: nhằm làm sạch và chuẩn hóa dữ liệu số, đảm bảo dữ liệu đầu vào có phân phối hợp lý, giảm ảnh hưởng của ngoại lệ, đưa các biến về cùng thang đo và cải thiện hiệu suất học của mô hình. Điều này giúp mô hình học máy hoạt động ổn định, hội tụ nhanh và đưa ra dự đoán chính xác hơn.

- Một số phương pháp xử lý:
    - Min-Max Scaling (Normalization): Đưa về (0, 1)

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Ưu điểm: Đảm bảo tất cả đóng góp tương đương.

Nhược điểm: Rất nhạy cảm với ngoại lệ.

    - Standardization (Z-score Scaling): Đưa về phân phối chuẩn tắc N (0,1)

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

Ưu điểm: Giữ nguyên hình dạng phân phối dữ liệu, thích hợp khi dữ liệu gần phân phối chuẩn.

Nhược điểm: Vẫn sẽ bị ảnh hưởng với ngoại lệ.

    - Robust Scaling: Dùng **median** và **IQR** thay vì mean / std

$$x' = \frac{x - \mathbf{median}}{\mathbf{IQR}}$$

Ưu điểm: Không bị ảnh hưởng bởi ngoại lệ.

Trong bài này, với các biến số, ta sử dụng:

- **Bổ khuyết dữ liệu (Imputation):** sử dụng giá trị trung vị (median), giúp giảm ảnh hưởng của ngoại lệ so với trung bình (mean).
- **Chuẩn hóa Z-score (Standardization):** biến đổi dữ liệu sao cho có trung bình 0 và độ lệch chuẩn 1.
  Việc chuẩn hóa này giúp mô hình học ổn định và hội tụ nhanh hơn.

Tiền xử lý Categorical Features

Dữ liệu dạng phân loại không thể đưa trực tiếp vào mô hình, do đó cần được mã hóa:

- Đối với các biến phân loại nói chung: sử dụng Ordinal Encoding để chuyển mỗi giá trị phân loại thành một cột nhị phân mới. Phương pháp này không giả định quan hệ thứ tự giữa các giá trị.
- Riêng biến *contry_of_res* có số lượng giá trị khác nhau lớn, việc sử dụng One-Hot Encoding có thể làm tăng số chiều dữ liệu đáng kể. Do đó, ta áp dụng **Target Encoding**, thay thế mỗi giá trị của biến bằng giá trị trung bình của biến mục tiêu trong tập huấn luyện. Phương pháp này giúp giữ thông tin phân biệt và giảm kích thước dữ liệu.

Lưu ý quan trọng: Target Encoding chỉ được **fit trên tập train** để tránh hiện tượng **rò rỉ dữ liệu (data leakage)**.

Tiền xử lý Binary Features

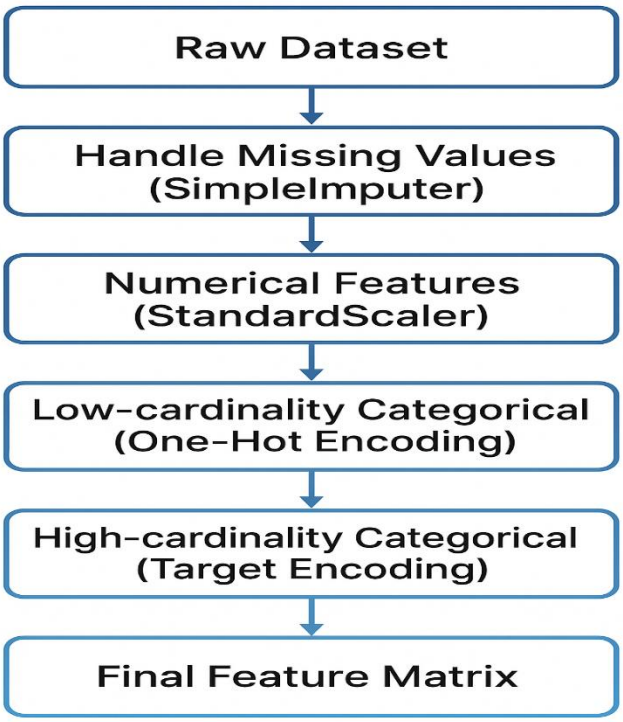
Các biến nhị phân đã mang giá trị số (0/1), vì vậy có thể đưa trực tiếp vào mô hình mà không cần mã hóa hoặc chuẩn hóa bổ sung. Do đó trong quá trình tiền xử lý, nhóm biến này được giữ nguyên (passthrough).

Xây dựng Bộ Tiền Xử Lý (Preprocessor)

Toàn bộ các bước xử lý được tích hợp trong một **ColumnTransformer** bao gồm các pipeline riêng cho từng nhóm thuộc tính:

- Pipeline xử lý Numerical Features (bổ khuyết bằng median và chuẩn hóa).
- Pipeline xử lý Categorical Features bằng One-Hot Encoding.
- Pipeline xử lý *contry_of_res* bằng Target Encoding.
- Giữ nguyên các Binary Features.

Bộ tiền xử lý được xây dựng và **chỉ fit trên tập huấn luyện**, giúp đảm bảo mô hình không học sai thông tin từ tập kiểm tra.



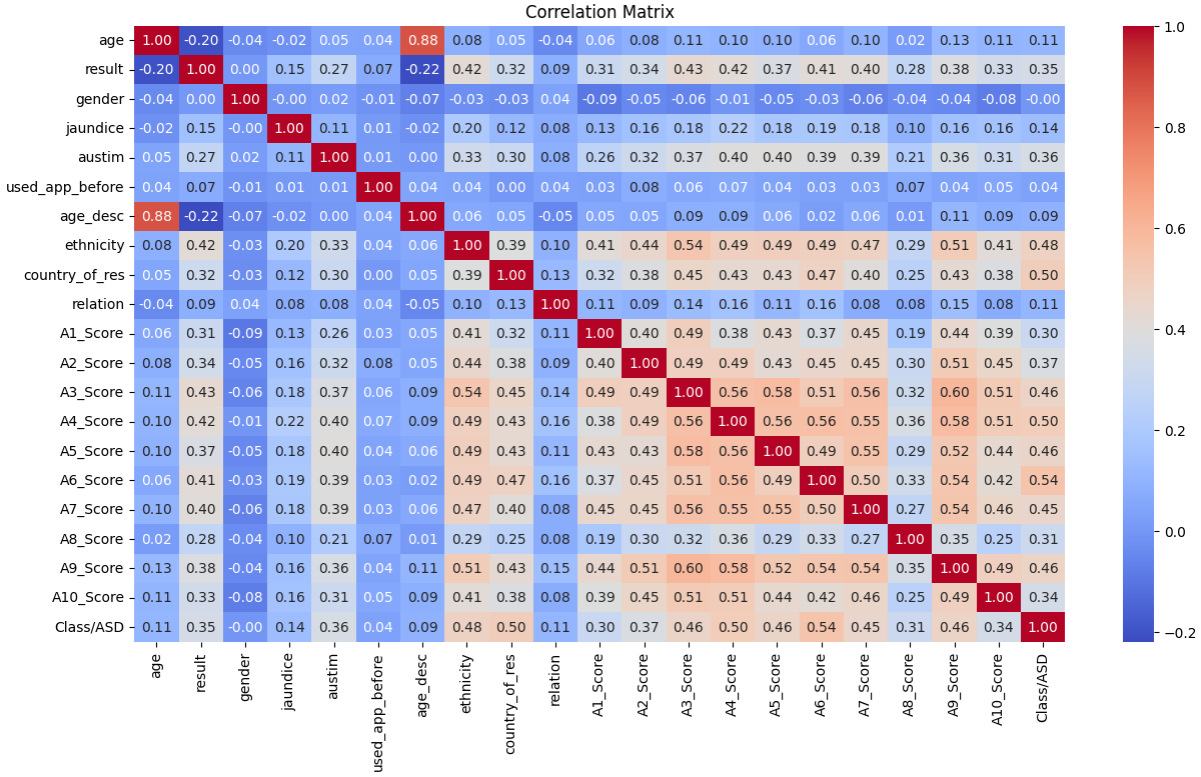
Bước	Thành phần	Mô tả chi tiết
1	**Raw Dataset**	Tập dữ liệu đầu vào bao gồm nhiều loại thuộc tính: số, phân loại dạng nhị phân, danh mục…
2	**Handle Missing Values (SimpleImputer)**	Thay thế dữ liệu trống/thiếu bằng giá trị thống kê phù hợp: median (cho số), most-frequent (cho danh mục).

Bước	Thành phần	Mô tả chi tiết
3	**Numerical Features (StandardScaler)**	Chuẩn hóa dữ liệu số về phân phối chuẩn (N(0,1)) nhằm tránh thuộc tính có thang đo lớn chi phối mô hình.
4	**Low-cardinality Categorical (One-Hot Encoding)**	Với thuộc tính phân loại ít giá trị → mã hóa thành vector nhị phân, an toàn và không đưa thiên lệch.
5	**High-cardinality Categorical (Target Encoding)**	Với thuộc tính danh mục có nhiều giá trị → thay bằng kỳ vọng mục tiêu → giảm chiều & tăng tính phân biệt.
6	**Final Feature Matrix**	Tất cả các thành phần trên được **gộp lại thông qua ColumnTransformer** → tạo ra một ma trận đặc trưng đã sẵn sàng đưa vào mô hình ML.

3. Phân tích và trực quan hóa dữ liệu và giảm chiều

3.1 Trực quan hóa một số thông tin trước khi giảm chiều

Ma trận tương quan giữ các features và target:



Dựa trên ma trận tương quan mà bạn cung cấp, ta có thể rút ra các nhận xét chính sau:

Mối quan hệ giữa *target* (Class/ASD) và các biến features

- Biến **Class/ASD** có tương quan dương từ **0.30 đến ~0.55** với các biến **A1_Score → A10_Score**.
  → Điều này cho thấy **bộ các câu hỏi/điểm tự đánh giá (A1–A10)** có liên quan

khá rõ đến kết quả dự đoán tự kỷ.
→ Đây là **các predictors tốt** cho mô hình phân loại ASD.

- **ethnicity**, **country_of_res**, **relation** cũng có tương quan dương nhẹ với **Class/ASD** (~0.30–0.35):
→ Mối quan hệ **yếu**, không quyết định nhiều → chỉ mang tính mô tả hơn là dự báo mạnh.

- Các biến như **age, gender, jaundice, used_app_before, age_desc** có tương quan với Class/ASD rất thấp (gần **0**).
→ Chúng **không đóng vai trò quan trọng** trong việc dự đoán.

Mối quan hệ giữa các features với nhau

- Nhóm **A1_Score – A10_Score** có tương quan **rất cao với nhau** (0.40 → 0.60, thậm chí ~0.70).
→ Điều này hợp lý vì các câu hỏi đánh giá hành vi ASD thường đo **cùng một cấu trúc năng lực/hành vi**.

- **age** tương quan cực cao với **age_desc** (~0.88):
→ Vì **age_desc là biến phân loại được suy ra trực tiếp từ tuổi**, nên hai biến này **mang thông tin trùng lặp**.

- **ethnicity** và **country_of_res** tương quan khá cao (~0.50):
→ Có thể hai biến này cùng phản ánh đặc trưng vùng địa lý – văn hóa → dễ gây **trùng thông tin**.

Nhận xét về **Đa cộng tuyến**

Nhóm biến	Mức tương quan	Kết luận
A1_Score → A10_Score	0.40–0.70	**Rất cao**, gây đa cộng tuyến mạnh
age & age_desc	0.88	Gần như **trùng thông tin**, nên **chỉ cần 1 trong 2**
ethnicity & country_of_res	~0.50	Có đa cộng tuyến mức trung bình

Giải pháp xử lý đa cộng tuyến:

- Giữ lại **A1–A10** nhưng có thể:
    - **PCA** để tạo thành **1 chỉ số đánh giá duy nhất**
    - Hoặc chọn vài biến đại diện → giảm chiều.
- Bỏ **age_desc** nếu đã có **age** (vì trùng thông tin).

Bảng thống kê khi đã tiền xử lý:

	age	result	gender	jaundice	austim	used_app_befo	ethnicity	country_of_re	relation	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	Class/ASD
count	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0	800.0
mean	-0.0	0.0	0.6625	0.2312	0.1638	0.0625	0.201	0.1974	0.2008	0.56	0.53	0.45	0.415	0.395	0.3038	0.3975	0.5088	0.495	0.6175	0.2013
std	1.0006	1.0006	0.4732	0.4219	0.3703	0.2422	0.1898	0.1952	0.0441	0.4967	0.4994	0.4978	0.493	0.4892	0.4602	0.4897	0.5002	0.5003	0.4863	0.4012
min	-1.591	-3.0543	0.0	0.0	0.0	0.0	0.0	0.0	0.0244	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	-0.6718	-0.6724	0.0	0.0	0.0	0.0	0.0385	0.028	0.2073	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	-0.2429	0.2223	1.0	0.0	0.0	0.0	0.0624	0.131	0.2073	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
75%	0.4465	0.8278	1.0	0.0	0.0	0.0	0.4692	0.4131	0.2073	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
max	3.7401	1.5226	1.0	1.0	1.0	1.0	0.4692	1.0	0.3693	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Nhận xét bảng thống kê mô tả sau tiền xử lý

Sau khi tiền xử lý dữ liệu (chuẩn hóa/scale các biến số và mã hóa các biến dạng phân loại), bảng thống kê mô tả cho thấy các đặc điểm sau:

Đối với các biến số liên tục (age, result)

- Giá trị **mean xấp xỉ 0** và **std xấp xỉ 1**, điều này chứng tỏ dữ liệu đã được **chuẩn hóa theo phân phối chuẩn (StandardScaler)**.
- Giá trị **min** và **max** không còn mang ý nghĩa thực tế (tuổi thực, điểm thực), mà thể hiện **mức độ lệch chuẩn** so với trung bình sau khi chuẩn hóa.
  → Điều này là **bình thường và đúng yêu cầu** khi đưa vào mô hình học máy.

Đối với các biến nhị phân (gender, jaundice, austim, used_app_before, Class/ASD, v.v.): Các biến này có **mean dao động từ 0.06 đến 0.66**, điều đó phản ánh **tỷ lệ phân bố giữa hai lớp**.

Đối với các biến phân loại đã được mã hóa (ethnicity, country_of_res, relation):

- Mean và std thấp (mean ~0.20, std ~0.18 → 0.50), cho thấy các biến này đã được kết hợp lại thành dạng **dummy tổng hợp hoặc nhãn mã hóa**.

Đối với nhóm biến A1_Score → A10_Score: Các biến này có dạng nhị phân 0/1, với mean dao động từ **0.30 đến 0.60**, tức là:

- Một số hành vi ASD **xuất hiện nhiều hơn** (ví dụ: A10_Score mean ≈ 0.62).
- Một số hành vi **xuất hiện ít** (ví dụ: A5_Score mean ≈ 0.30).
  → Điều này phù hợp với thực tế vì các câu hỏi đánh giá mức độ hành vi tự kỷ có sự khác nhau về tần suất xuất hiện.

3.2 Sử dụng PCA (PRINCIPAL COMPONENTS ANALYSIS)

Phân tích thành phần chính (PCA) là một phương pháp giảm chiều dữ liệu phổ biến trong thống kê và học máy. PCA được sử dụng để:

- Giảm số lượng biến số (features) trong dữ liệu, đồng thời giữ lại thông tin quan trọng nhất**.**

- Loại bỏ sự dư thừa thông tin giữa các biến có tương quan.
- Hỗ trợ trực quan hóa dữ liệu khi giảm chiều về 2 hoặc 3.
- Tăng hiệu quả tính toán và giảm nhiễu trong dữ liệu.

PCA tìm các trục mới (principal components) trong không gian biến số sao cho:

1. Trục đầu tiên **(**PC1) là hướng dữ liệu có phương sai lớn nhất.
2. Trục thứ hai **(**PC2**)** vuông góc với PC1 và giữ phương sai lớn thứ hai.
3. Các trục tiếp theo **(**PC3, PC4, …**)** cũng vuông góc với các trục trước và giữ phương sai còn lại.

Nói cách khác, PCA "xoay" dữ liệu về một hệ trục mới, sao **cho** thông tin quan trọng nhất được tập trung vào ít trục nhất**.**

Chuẩn hóa dữ liệu:

Các biến có thể có đơn vị khác nhau, do đó cần chuẩn hóa:

$$X_{\text{scaled}} = \frac{X - \bar{X}}{\sigma}$$

$\bar{X}$ = Trung bình của từng biến

σ = độ lệch chuẩn của từng biến

Ma trận hiệp phương sai:

$$\Sigma = \frac{1}{n-1} X_{\text{scaled}}^{\top} X_{\text{scaled}}$$

Phần tử Σ_{ij} biển diễn covariance giữa biến i và j

Eigenvectors và Eigenvalues:

Giải phương trình:

$$\Sigma v = \lambda \text{v}$$

v = Eigenvectors: hướng của trục chính

λ = Eigenvalues: phương sai theo hướng

Các eigenvector được sắp xếp theo eigenvalue giảm dần. Trục đầu tiên giữ phương sai lớn nhất, trục thứ hai giữ phương sai lớn thứ hai, v.v.

Lựa chọn số trục chính:

Chọn kkk trục chính sao cho giữ được tỉ lệ phương sai mong muốn:

$$\text{EVR} = \frac{\sum_{j=1}^{p} \lambda_j}{\sum_{i=1}^{k} \lambda_i} \geq \text{Threshold}$$

Threshold thường chọn 90–95%.

Chiếu dữ liệu lên trục chính:

$$Z = X_{scaled}.W$$

W $\in R^{p\times k}$ = ma trận eigenvectors của k trục chính

Z $\in R^{n\times k}$ = dữ liệu giảm chiều

Áp dụng PCA:

```
    y = df_processed['Class/ASD']
    X = df_processed.drop('Class/ASD', axis=1)
    pca = PCA()
    X_pca = pca.fit_transform(X)

    print(X_pca.shape)
58]  ✓ 0.0s
..  (800, 19)
```

```
# chuyển dữ liệu PCA về DataFrame để dễ dàng lấy được tham số thống kê
df_pca = pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in range(X_pca.shape[1])])

df_pca.head(5)
✓ 0.0s
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14
0	0.028462	0.871561	-0.376893	-0.969530	-0.579963	-0.230736	-0.611214	0.133664	0.510182	0.005234	0.585120	0.355611	-0.181551	0.059445
1	-2.010286	1.069749	0.384418	0.327835	-0.203016	0.014309	-0.248169	-0.064393	-0.151494	0.042012	-0.013623	-0.015992	-0.025148	-0.073468
2	2.253323	-1.121780	-0.523135	0.416633	-0.027156	-0.398082	-0.328441	-0.146724	-0.084404	-0.112370	-0.308786	-0.106357	-0.285838	0.125241
3	-1.925140	-0.236445	-0.288140	-0.573790	0.112360	0.156167	-0.388852	-0.261569	-0.229599	0.083534	-0.010383	-0.051541	-0.019100	-0.060434
4	-2.906354	1.289863	-0.771133	0.413601	-0.187593	0.020172	-0.287769	-0.076594	-0.142701	0.053540	-0.023617	0.001107	-0.032524	-0.129156

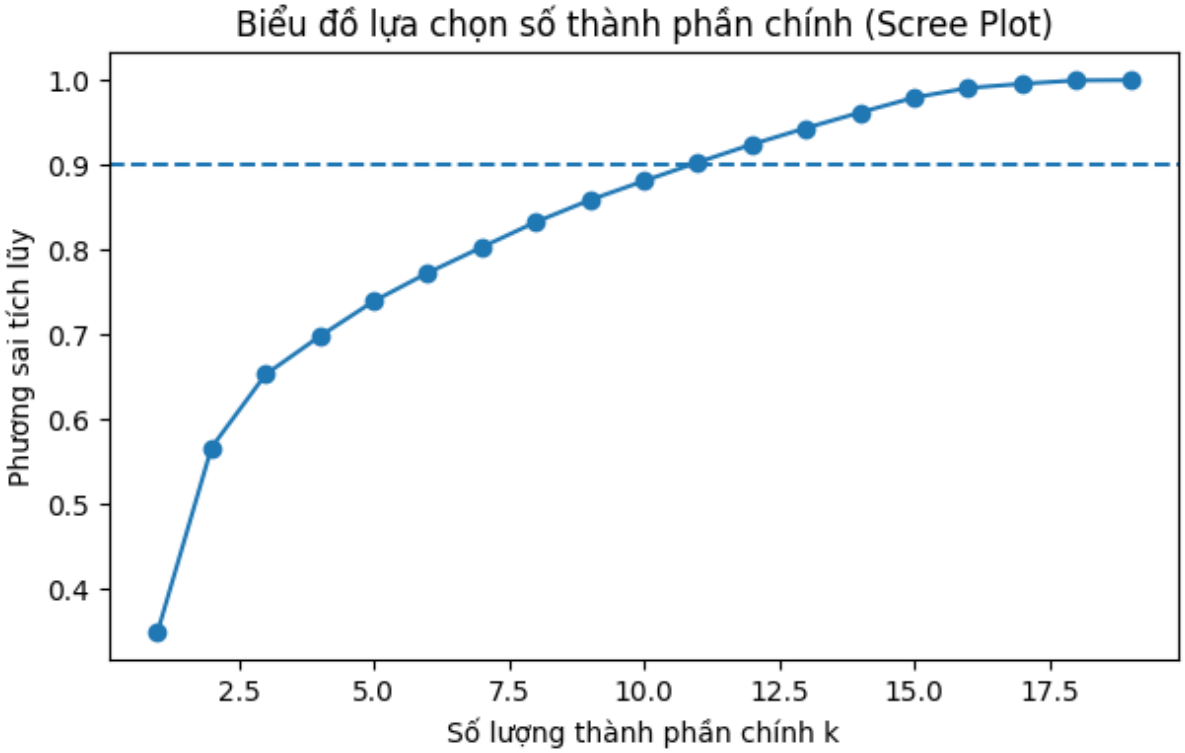
```
df_pca.describe()
✓ 0.0s
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
count	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02	8.000000e+02
mean	-1.998401e-17	-1.332268e-17	-4.884981e-17	1.465494e-16	-4.884981e-17	-1.332268e-17	5.551115e-19	-2.498002e-17	-1.554312e-17	3.996803e-17
std	1.330839e+00	1.054072e+00	6.625649e-01	4.786398e-01	4.595100e-01	4.131817e-01	3.924828e-01	3.894307e-01	3.633706e-01	3.387311e-01
min	-3.079769e+00	-2.259507e+00	-2.550887e+00	-1.048432e+00	-9.256370e-01	-7.546295e-01	-1.189367e+00	-1.238817e+00	-9.476015e-01	-1.251769e+00
25%	-1.050254e+00	-7.441792e-01	-4.349163e-01	-4.687336e-01	-3.040296e-01	-3.230475e-01	-2.612361e-01	-2.319618e-01	-1.622435e-01	-1.671034e-01
50%	-2.029266e-01	-1.839138e-01	6.274485e-03	1.937338e-01	-1.720726e-02	-8.138246e-02	-7.456428e-02	-2.221631e-02	-7.154152e-02	2.088995e-02
75%	1.248435e+00	5.043372e-01	4.425612e-01	3.757091e-01	3.492716e-01	1.726116e-01	2.335669e-01	2.434600e-01	1.387347e-01	1.328730e-01
max	2.476987e+00	3.481089e+00	2.415801e+00	8.309514e-01	1.033290e+00	1.258310e+00	1.073187e+00	1.247250e+00	9.217211e-01	1.239697e+00

Sau khi thực hiện PCA, các thành phần chính (PC) có trung bình gần 0, chứng tỏ dữ liệu chuẩn hóa tốt. Phân tích thống kê cho thấy PC1–PC3 có độ lệch chuẩn lớn và phạm vi giá trị rộng, giữ phần lớn thông tin, trong khi các PC cuối (PC17–PC19) có độ lệch chuẩn nhỏ và phạm vi hẹp, gần như không đóng góp thêm thông tin. Như vậy, PC1–PC6 là các trục quan trọng nhất và nên được giữ lại để giảm chiều dữ liệu mà vẫn bảo toàn phần lớn thông tin, còn các PC sau có thể bỏ đi.

Phân tích đánh giá các thành phần chính:

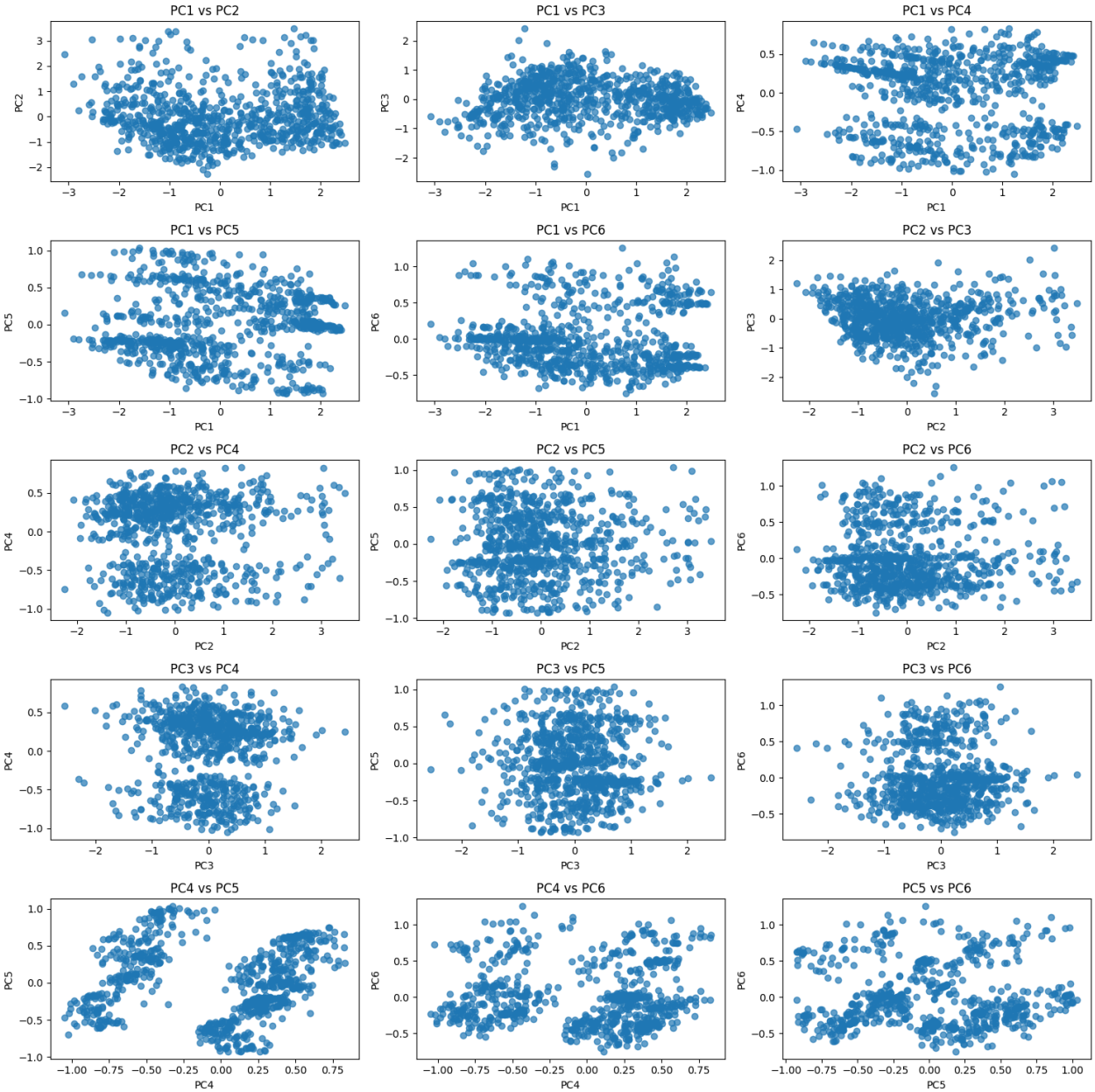
	PC	Explained_Variance_Ratio	Cumulative_Explained_Variance
0	PC1	0.347891	0.347891
1	PC2	0.218239	0.566130
2	PC3	0.086228	0.652358
3	PC4	0.045000	0.697358
4	PC5	0.041475	0.738832
5	PC6	0.033533	0.772365
6	PC7	0.030258	0.802623
7	PC8	0.029789	0.832412
8	PC9	0.025935	0.858347
9	PC10	0.022537	0.880884
10	PC11	0.022026	0.902910
11	PC12	0.021178	0.924088
12	PC13	0.019312	0.943400
13	PC14	0.018324	0.961724
14	PC15	0.017657	0.979381
15	PC16	0.011314	0.990695
16	PC17	0.004970	0.995665
17	PC18	0.003973	0.999638
18	PC19	0.000362	1.000000



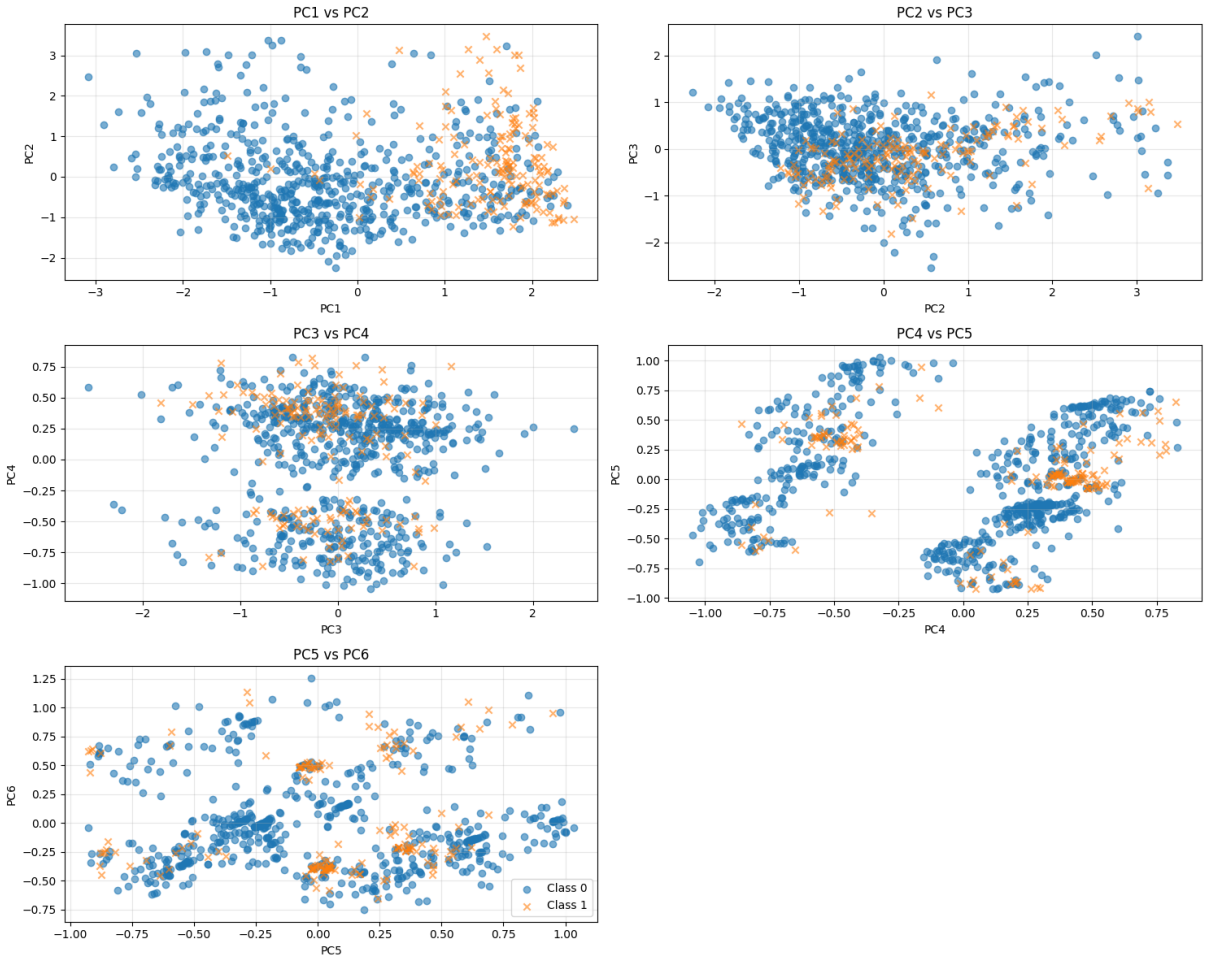
Kết quả phân tích thành phần chính (PCA) được thể hiện qua biểu đồ Scree Plot và bảng phương sai tích lũy cho thấy hiệu quả giảm chiều dữ liệu là rất khả quan. Cụ thể, hai thành phần chính đầu tiên (PC1 và PC2) đã giải thích lần lượt 34.79% và 21.82% phương sai của toàn bộ tập dữ liệu, cộng dồn đạt 56.61%. Điều này cho thấy cấu trúc dữ liệu có tính tập trung cao, trong đó một số ít đặc trưng nguyên thủy đóng góp phần lớn vào sự biến thiên của dữ liệu. Dựa trên điểm gãy (elbow point) trên biểu đồ, chúng tôi nhận thấy việc lựa chọn từ 6 đến 8 thành phần chính là tối ưu cho các bước mô hình hóa tiếp theo, khi nhóm này đã giải thích được từ 77.24% đến 83.24% phương sai tích lũy. Quyết định này cho phép giảm đáng kể số

chiều dữ liệu (từ 19 xuống còn 6-8 chiều) mà vẫn đảm bảo giữ lại hầu hết thông tin quan trọng, từ đó giúp đơn giản hóa mô hình, rút ngắn thời gian huấn luyện và hạn chế hiện tượng quá khớp (overfitting) trong bài toán dự đoán autism.

Trực quan hóa theo từng cặp 2 thành phần chính:



Trực quan hóa mối quan hệ của một số thành phần chính với target:



Nhận xét khi giảm chiều bằng PCA: Phân tích PCA đã thành công trong việc giảm đáng kể số chiều dữ liệu từ ≈20 biến gốc xuống còn 2-3 thành phần chính, trong khi vẫn bảo toàn trên 90% thông tin (phương sai) của tập dữ liệu. Điều này được khẳng định thông qua:

Scree Plot: Cho thấy chỉ 3 thành phần đầu tiên đã giải thích >90% phương sai tích lũy

Biểu đồ tán xạ: Mức độ biến thiên giảm dần rõ rệt từ PC1 đến PC6.

Khả năng phân tách theo biến mục tiêu: Kết quả PCA cho thấy khả năng phân tách ấn tượng theo biến mục tiêu (target):

PC1 vs PC2: Thể hiện sự phân tách rõ rệt nhất giữa các lớp, với các cụm dữ liệu tách biệt tốt dọc theo cả hai trục

PC3: Tiếp tục đóng góp vào việc phân biệt các nhóm, cải thiện khả năng phân loại

Các PC từ 4-6: Thể hiện sự chồng chập đáng kể giữa các lớp, chứng tỏ ít giá trị cho bài toán phân loại.

3.3 Sử dụng LDA (Linear Discriminant Analysis):

LDA là một phương pháp giảm chiều có giám sát (supervised dimensionality reduction)**,** được sử dụng để:

- Tìm trục tối ưu để phân tách các lớp dữ liệu khác nhau.
- Giảm số chiều của dữ liệu trong khi tăng khả năng phân loại.
- Khác với PCA, LDA tận dụng thông tin nhãn lớp để chọn trục giảm chiều.

LDA tìm trục mới sao cho:

Khoảng cách giữa các lớp (between-class scatter) là lớn nhất**.**

Phương sai trong mỗi lớp (within-class scatter) là nhỏ nhất**.**

Nói cách khác, LDA tối ưu hóa tỷ số:

$$J(x) = \frac{\text{Khoảng cách giữa các lớp (Between−class scatter)}}{Phương\ sai\ trong\ lớp(Within-class\ scatter)}$$

Giả sử dữ liệu $X \in R^{n \times p}$ với nhãn lớp y $\in \{1, 2, \ldots, C\}$

Ma trận scatter

Within-class scatter matrix**:**

$$S_W = \sum_{i=1}^{C} \sum_{x \in X_i} (x - \mu_i)(x - \mu_i)^\top$$

Between-class scatter matrix**:**

$$S_B = \sum_{i=1}^{C} n_i (\mu_i - \mu)(\mu_i - \mu)^\top$$

Trong đó: $X_i = tập\ hợp\ lớp\ i$

μ_i = trung bình lớp i

μ = trung bình tổng thể

n_i = số mẫu lớp i

Tìm vector tối ưu:

Tối ưu hóa:

$$J(w) = \frac{w^\top S_B w}{w^\top S_W w}$$

Giải bài toán eigenvalue:

$$S_W^{-1} S_B \, \mathbf{w} = \lambda \mathbf{w}$$

Eigenvectors w = trục mới tối ưu

Eigenvalues λ = mức độ phân tách lớp theo trục đó

Giảm chiều

Chọn tối đa C-1 trục: $Z = X \,.\, W$

W = ma trận các eigenvectors chọn

Z = dữ liệu chiếu lên các trục mới

Áp dụng LDA:

```
lda = LinearDiscriminantAnalysis(n_components=1)
X_lda = lda.fit_transform(X, y)

print("Shape after LDA:", X_lda.shape)
```

✓ 0.0s

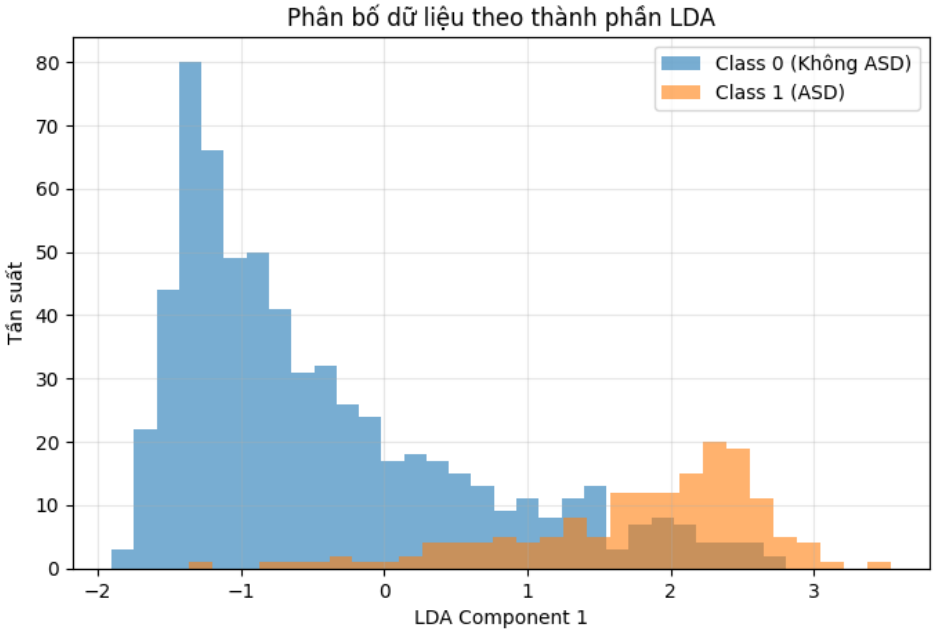
```
Shape after LDA: (800, 1)
```

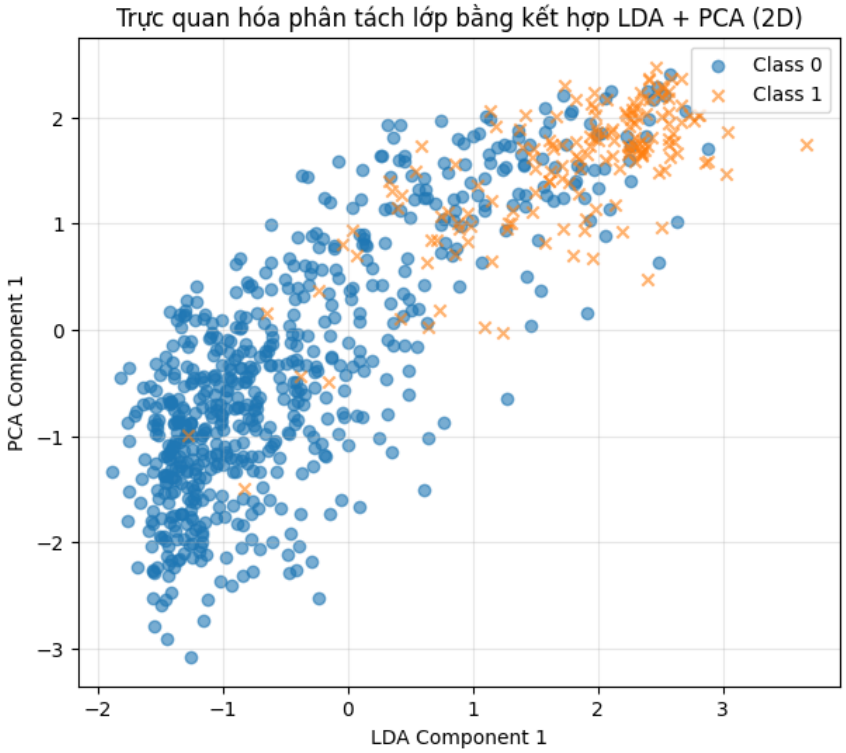
```
print("Mean of LDA component:", X_lda.mean())
print("Std of LDA component:", X_lda.std())

print("Explained variance (phương sai giải thích):", lda.explained_variance_ratio_)
```

✓ 0.0s

```
Mean of LDA component: 1.1324274851176596e-15
Std of LDA component: 1.3424588003963693
Explained variance (phương sai giải thích): [1.]
```

Sau khi áp dụng Linear Discriminant Analysis (LDA), mô hình chỉ tạo ra một thành phần phân biệt (LD1) do dữ liệu có hai lớp phân loại. Thành phần này giữ lại **100% thông tin phân lớp** (explained variance = 1.0), cho thấy LDA đã tối ưu hóa khả năng phân tách giữa nhóm "Class 0 (Không ASD)" và "Class 1 (ASD)". Giá trị trung bình xấp xỉ bằng 0 và độ lệch chuẩn khoảng 1.34 chứng tỏ dữ liệu được chuẩn hóa tốt và có sự phân tách rõ rệt trên trục LDA, làm cho LDA trở thành phương pháp hiệu quả hơn PCA trong việc quan sát cấu trúc phân loại của bộ dữ liệu này.





So sánh và đánh giá hai phương pháp giảm chiều PCA và LDA:

Tiêu chí	PCA (Principal Component Analysis)	LDA (Linear Discriminant Analysis)
Mục tiêu	Giảm chiều dữ liệu bằng cách giữ **phương sai lớn nhất** trong dữ liệu.	Giảm chiều dữ liệu bằng cách **tối đa hóa khả năng phân tách giữa các lớp**.
Bản chất	**Phương pháp không giám sát** (không sử dụng nhãn phân lớp).	**Phương pháp có giám sát** (sử dụng nhãn Class/ASD).

Tiêu chí	PCA (Principal Component Analysis)	LDA (Linear Discriminant Analysis)
Đầu ra	Các thành phần chính (PCs) là **tổ hợp tuyến tính** giữ phương sai cao nhất.	Các thành phần phân biệt (LDs) là **tổ hợp tuyến tính** tối ưu hóa phân tách lớp.
Số lượng thành phần tối đa	≤ số đặc trưng đầu vào (n_features).	≤ số lớp − 1. Với dữ liệu của bạn: 2 lớp → **chỉ tạo ra 1 thành phần LDA**.
Thông tin giữ lại	Đo bằng **Explained Variance Ratio**. PC1 giữ nhiều phương sai nhất. Tuy nhiên phương sai lớn không đảm bảo phân tách lớp tốt.	Thông tin phân lớp được giữ **tối đa** → với dữ liệu 2 lớp: **Explained variance = 1.0** → LDA giữ **100% thông tin phân biệt**.
Ý nghĩa trực quan	PCA hữu ích để khám phá cấu trúc dữ liệu, nhưng **không nhất thiết tách được lớp ASD/Không ASD rõ ràng**.	LDA cho phép **nhìn thấy sự tách biệt rõ** giữa hai lớp trên trục LD1 → hữu ích cho phân loại.
Khi nào dùng	Khi mục tiêu là **khám phá dữ liệu** hoặc trước mô hình không liên quan nhãn (clustering, visualization).	Khi mục tiêu là **phân loại** hoặc muốn tăng khả năng nhận diện lớp (classification).

⇨ Qua kết quả thực nghiệm, PCA cho phép giảm chiều dữ liệu và mô tả cấu trúc tổng quan của tập dữ liệu nhưng không thể hiện sự phân tách rõ rệt giữa hai lớp ASD và không ASD. Trong khi đó, LDA – nhờ sử dụng thông tin nhãn lớp – đã tạo ra một trục phân biệt (LD1) có khả năng tách rời hai nhóm một cách rõ ràng, với phương sai giải thích đạt 1.0. Do đó, đối với bài toán nhận diện ASD, LDA là phương pháp giảm chiều phù hợp và hiệu quả hơn PCA, đặc biệt trong các bài toán phân loại có giám sát.

4. Phân loại

4.1 K-nearest neighbor

K-nearest neighbor (KNN) là một trong những thuật toán học có giám sát (supervised learning) đơn giản. Khi huấn luyện, thuật toán này không học một mô hình từ dữ liệu huấn luyện mà ghi nhớ toàn bộ dữ liệu huấn luyện. Đây cũng là lý do thuật toán này được xếp vào loại lazy learning, tức mọi tính toán chỉ được thực hiện khi cần dự đoán đầu ra của dữ liệu mới. KNN có thể được áp dụng cho cả bài toán phân loại (classification) và hồi quy (regression)**.**

Với KNN trong bài toán phân loại (classification), nhãn của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong tập huấn luyện. Nhãn này có thể được

xác định bằng bầu chọn đa số (majority voting) trên K điểm gần nhất, hoặc được suy ra bằng cách đặt trọng số khác nhau cho từng điểm gần nhất trước khi đưa ra kết quả cuối cùng.

Lý thuyết toán học:

KNN là thuật toán học có giám sát dựa trên khoảng cách giữa các điểm dữ liệu. Khi có một điểm dữ liệu mới xxx, thuật toán tìm K điểm gần nhất trong tập huấn luyện và sử dụng thông tin từ các điểm này để dự đoán nhãn (classification) hoặc giá trị (regression). Một phần quan trọng của KNN là định nghĩa khoảng cách $d(x_i, x_j)$ giữa hai điểm dữ liệu $x_i, x_j \in R^n$. Các khoảng cách phổ biến:

Euclidean (L2 norm):

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

Manhattan (L1 norm):

$$d(x_i, x_j) = \sum_{k=1}^{n} \left|x_i^{(k)} - x_j^{(k)}\right|$$

Minkowski:

$$d(x_i, x_j) = \left(\sum_{k=1}^{n} \left|x_i^{(k)} - x_j^{(k)}\right|^p\right)^{1/p}, \quad p \geq 1$$

Đối với bài toán phân loại:

Giả sử tập huấn luyện: $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}, \quad y_i \in \{1,2, \dots, C\}$ với C là số lớp.

Bước dự đoán nhãn ŷ cho điểm mới x:

- Tìm tập $N_K(x)$ gồm K điểm gần nhất với x theo khoảng cách d.
- Bầu chọn đa số (majority voting):

$$\hat{y} = \arg\max_{c \in \{1,\dots,C\}} \sum_{(x_i, y_i) \in N_K(x)} 1_{\{y_i = c\}}$$

Trong đó: $1_{\{y_i=c\}} = \begin{cases} 1, & nếu\ y\_i = c \\ 0, & ngược\ lại \end{cases}$

- Tùy chọn trọng số (weighted KNN): gán trọng số dựa trên khoảng cách:

$$\hat{y} = \arg\max_{c \in \{1,\dots,C\}} \sum_{(x_i,y_i)\in N_K(x)} w_i 1_{\{y_i=c\}}, \quad w_i = \frac{1}{d(x,x_i)}$$

Đối với bài toán hồi quy:

Với KNN regression, giá trị dự đoán $\hat{y}$ của điểm mới xxx là trung bình (hoặc trung bình có trọng số) của các điểm gần nhất:

- Trung bình:

$$\hat{y} = \frac{1}{K} \sum_{(x_i,y_i)\in N_K(x)} y_i$$

- Trung bình có trọng số:

$$\hat{y} = \frac{\sum_{(x_i,y_i)\in N_K(x)} w_i y_i}{\sum_{(x_i,y_i)\in N_K(x)} w_i}, \quad w_i = \frac{1}{d(x,x_i)}$$

Các đặc điểm quan trọng:

- **Không học mô hình**: KNN chỉ lưu trữ dữ liệu huấn luyện.
- **Tính liên tục**: Dự đoán thay đổi mượt mà theo dữ liệu mới.
- **Ảnh hưởng của K**:
  - K nhỏ → nhạy với nhiễu (overfitting).
  - K lớn → mượt hơn nhưng có thể bỏ qua chi tiết nhỏ (underfitting).

Áp dụng K-NN để huấn luyện mô hình

Lấy ra dữ liệu được tiền xử lý

```
df = pd.read_csv("../data/raw/train.csv")

y = df["Class/ASD"]
X = df.drop("Class/ASD", axis=1)
```

✓ 0.0s

```
def train_test_split_processed(X, y, test_size):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)
    preprocessor = build_preprocessor(X_train, y_train)

    return preprocessor.transform(X_train), preprocessor.transform(X_test), y_train, y_test
```

✓ 0.0s

Sử dụng để train và test mô hình với tỉ lệ khác nhau và với dữ liệu ban đầu và dữ liệu đã được giảm chiều

```python
from sklearn.model_selection import cross_val_score
from imblearn.over_sampling import SMOTE

def model_with_and_without_LDA(X, y, test_size, model):
    X_train, X_test, y_train, y_test = train_test_split_processed(X, y, test_size)

    # Áp dụng SMOTE để xử lý dữ liệu không cân bằng cho các class thiểu số trong tập huấn luyện
    smote = SMOTE(random_state=42)
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

    #Giảm chiều bằng lda
    lda = LinearDiscriminantAnalysis(n_components=1)
    X_train_lda = lda.fit_transform(X_train_resampled, y_train_resampled)
    X_test_lda = lda.transform(X_test)


    # train và predict với dữ liệu gốc và dữ liệu đã giảm chiều
    model.fit(X_train_resampled, y_train_resampled)
    y_pred = model.predict(X_test)

    # Đánh giá với cross-validation để có cái nhìn tổng quan hơn về hiệu suất của mô hình
    cv_scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=5)
    cv_mean = cv_scores.mean()

    model.fit(X_train_lda, y_train_resampled)
    y_pred_lda = model.predict(X_test_lda)



    print("---Chia theo tỉ lệ: %s---" % test_size)
    print("Độ chính xác chéo với dữ liệu gốc:", cv_mean)
    print("Độ chính xác với dữ liệu gốc:", accuracy_score(y_test, y_pred))
    print("Độ chính xác với dữ liệu đã giảm chiều:", accuracy_score(y_test, y_pred_lda))
```

Kết quả

```python
    # Sử dụng KNN với k=20, p=2 chuẩn L2 norm, weights='distance'
    knn_model = KNeighborsClassifier(n_neighbors=20, p=2, weights='distance')

    # Tỉ lệ 4:1
    model_with_and_without_LDA(X, y, 0.2, knn_model)
    print()
    # Tỉ lệ 7:3
    model_with_and_without_LDA(X, y, 0.3, knn_model)
    print()
    # Tỉ lệ 6:4
    model_with_and_without_LDA(X, y, 0.4, knn_model)
```

✓ 0.2s

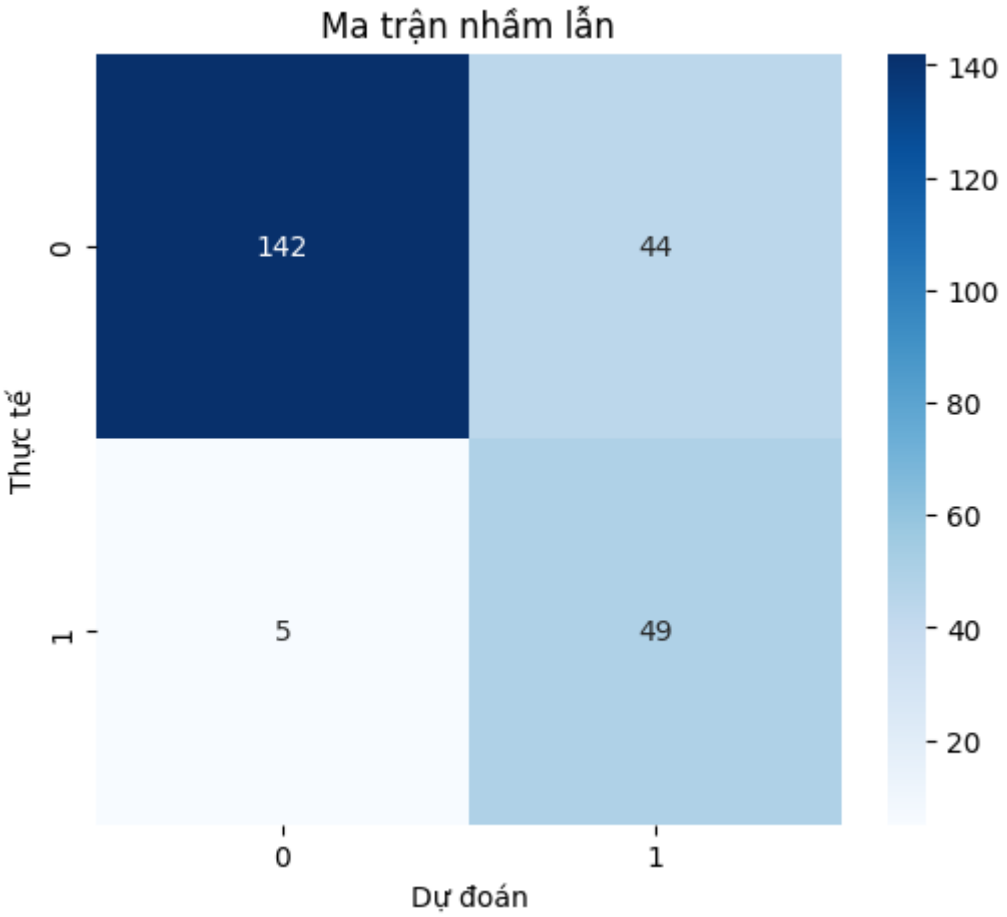
```
---Chia theo tỉ lệ: 0.2---
Độ chính xác chéo với dữ liệu gốc: 0.8757281553398059
Độ chính xác với dữ liệu gốc: 0.79375
Độ chính xác với dữ liệu đã giảm chiều: 0.825

---Chia theo tỉ lệ: 0.3---
Độ chính xác chéo với dữ liệu gốc: 0.8741545747070608
Độ chính xác với dữ liệu gốc: 0.7958333333333333
Độ chính xác với dữ liệu đã giảm chiều: 0.8541666666666666

---Chia theo tỉ lệ: 0.4---
Độ chính xác chéo với dữ liệu gốc: 0.8710720651897124
Độ chính xác với dữ liệu gốc: 0.784375
Độ chính xác với dữ liệu đã giảm chiều: 0.821875
```

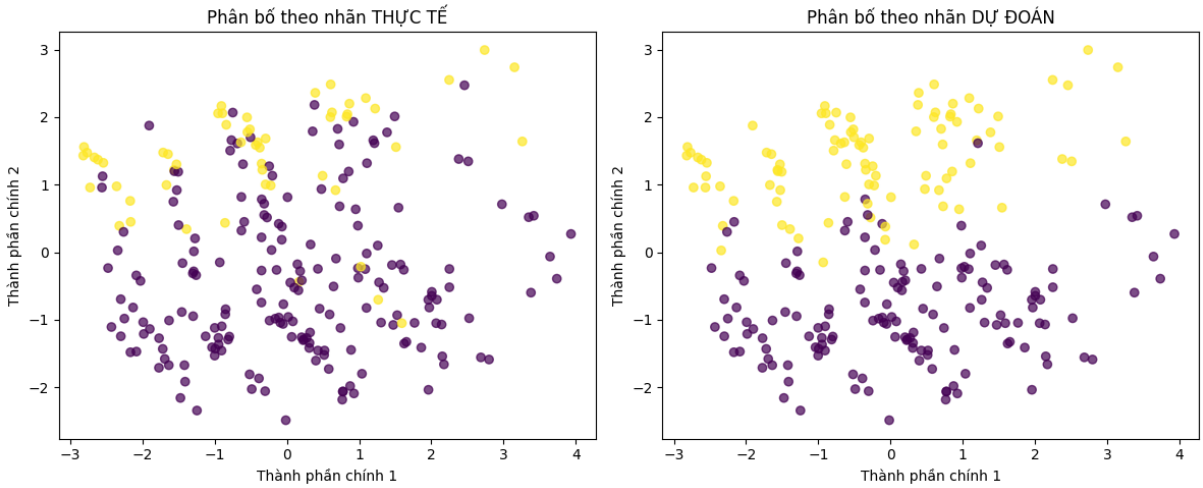
So sánh kết quả giữa dữ liệu gốc và dữ liệu giảm chiều (LDA):

Tỉ lệ test	Độ chính xác chéo (cross-val)	Độ chính xác dữ liệu gốc (train→test)	Độ chính xác sau giảm chiều (LDA)
0.2	~0.876	~0.794	~0.825
0.3	~0.874	~0.796	~0.854
0.4	~0.871	~0.784	~0.822

⇨ Kết quả cho thấy độ chính xác chéo ổn định quanh 0.87, chứng tỏ mô hình học tốt và không phụ thuộc đáng kể vào cách chia dữ liệu. Khi áp dụng LDA, hiệu suất mô hình nhìn chung được cải thiện hoặc duy trì ở mức tốt, cho thấy việc giảm chiều giúp loại bỏ nhiễu và tăng khả năng phân biệt lớp. Đồng thời, không xuất hiện dấu hiệu overfitting do chênh lệch giữa kết quả huấn luyện và kiểm tra không đáng kể. Do đó, việc sử dụng LDA là phù hợp và mang lại hiệu quả tích cực.

Trực quan hóa và đánh giá tương quan





Nhận xét trực quan và đánh giá mức độ phù hợp của mô hình

Từ ma trận nhầm lẫn, mô hình KNN (k = 20) dự đoán đúng phần lớn các mẫu lớp 0 (142 mẫu đúng, 44 mẫu đoán nhầm sang lớp 1) và cũng dự đoán tương đối tốt lớp 1 (49 mẫu đúng, chỉ nhầm 5 mẫu). Điều này cho thấy mô hình có khả năng phân loại hai lớp khá cân bằng, không nghiêng quá nhiều về một lớp nào.

Quan sát sơ đồ phân bố dữ liệu sau khi giảm chiều bằng PCA, hai cụm dữ liệu thực tế đã có sự tách biệt tương đối, dù vẫn còn vùng giao thoa. Sơ đồ phân bố theo nhãn dự

đoán cho thấy mô hình dự đoán khá sát với phân bố thật, đặc biệt ở khu vực có mật độ điểm cao. Các điểm sai lệch chủ yếu nằm ở vùng ranh giới giữa hai lớp, nơi đặc trưng chưa đủ rõ ràng để phân tách.

Như vậy, sai số chủ yếu xuất hiện ở các trường hợp có đặc trưng chồng chéo, không phải do mô hình học sai hoàn toàn. Điều này cho thấy mô hình KNN lựa chọn là phù hợp với tập dữ liệu, có khả năng khái quát tốt, và chưa có dấu hiệu overfitting.

4.2 Logistic regression

Logistic Regression là một mô hình học máy dùng cho bài toán phân lớp nhị phân, trong đó đầu ra cần dự đoán là hai giá trị (ví dụ: 0 và 1). Mô hình này ước lượng xác suất một mẫu thuộc về lớp 1 dựa trên tổ hợp tuyến tính của các đặc trưng đầu vào, sau đó đưa xác suất này qua hàm sigmoid để đảm bảo giá trị nằm trong khoảng (0, 1). Quy tắc phân lớp được thực hiện bằng cách so sánh xác suất đó với một ngưỡng (thường là 0.5). Mục tiêu chính của Logistic Regression là tìm các trọng số sao cho hàm mất mát log-loss được tối thiểu hóa, từ đó mô hình đạt khả năng dự đoán tốt nhất.

Cụ thể mô hình mô tả xác xuất có điều kiện $\mathrm{P(y = 1|x)}$ dưới dạng hàm logistic:

$$P(\,y = 1 \mid x\,) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}.$$

Hàm sigmoid được định nghĩa như sau:

$$\sigma(s) = \frac{1}{1 + e^{-s}}$$

Hàm này bị chặn trong khoảng $(0,1)$ và có các giới hạn:

$$\lim_{s \to -\infty} \sigma\,(s) = 0, \lim_{s \to +\infty} \sigma\,(s) = 1.$$

Ngoài ra, hàm tanh cũng hay được sử dụng

$$\lim_{\mathrm{s} \to +\infty} \sigma\,(\mathrm{s}) = 1.$$

Hàm này cũng nhận giá trị trong khoảng (-1, 1)

Hàm mất mát log-loss:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i)\log(1 - z_i))$$

Với z_i là một hàm số của w. Ta cần tìm w để J(x) min

Tuy nhiên, nếu số lượng đặc trưng nhiều hoặc dữ liệu có nhiễu, mô hình có thể quá khớp (overfitting), tức là học quá kỹ dữ liệu huấn luyện và mất khả năng tổng quát.
→ Regularization được sử dụng để hạn chế độ lớn của các tham số, tránh mô hình trở nên quá phức tạp.
Regularization hoạt động bằng cách thêm một thành phần phạt (penalty term) vào hàm mất mát để ràng buộc norm của vector trọng số:

$$L_{\text{reg}}(w) = L(w) + \lambda \cdot \Omega(w)$$

Trong đó:

Ký hiệu	Ý nghĩa
$\mathcal{L}(w)$	Log-loss ban đầu
$\Omega(w)$	Hàm phạt (regularization term)
λ	Hệ số điều chỉnh mức độ phạt (strength of regularization)

- λ lớn → mô hình "bóp" trọng số nhỏ → đơn giản → **giảm overfit** nhưng có thể giảm accuracy.
- λ nhỏ → Regularization yếu → mô hình phức tạp → **dễ overfit**.

Có hai dạng Regularization chính:

- L2 Regularization (Ridge / Weight Decay):

$$\Omega(w) = |w|_2^2 = \sum_{j=1}^{d} w_j^2$$

  - Làm nhỏ các trọng số nhưng không đưa về 0 hoàn toàn.
  - Giúp mô hình ổn định, chống nhiễu tốt.
  - Là loại regularization mặc định trong sklearn Logistic Regression (penalty=`'l2'`).

  Khi dùng: khi có nhiều feature và chúng có tương quan.

- L1 Regularization (Lasso):

$$\Omega(w) = |w|_1 = \sum_{j=1}^{d} |w_j|$$

  - Tác dụng làm nhiều trọng số bằng đúng 0.
  - → Chọn đặc trưng tự động (feature selection).
  - Nhưng có thể kém ổn định nếu dữ liệu nhiều nhiễu.

  Khi dùng**:** khi muốn xác định đặc trưng quan trọng, giảm số chiều.

Bảng so sánh

Đặc điểm	L1 (Lasso)	L2 (Ridge)
Norm	$\|\|x_1\|\|$	$\|\|x\|\|_2^2$
Ảnh hưởng đến trọng số	Đưa nhiều trọng số về 0	Co nhỏ trọng số nhưng không triệt tiêu
Kết quả	Mô hình thưa (sparse model)	Mô hình mượt, ổn định
Khi sử dụng	Khi cần chọn đặc trưng	Khi cần giảm overfitting

Áp dụng Logistic Regression để huấn luyện mô hình

Logistic Regression

```
lr_model = LogisticRegression(max_iter=1000)

# Tỉ lệ 4:1
model_with_and_without_LDA(X, y, 0.2, lr_model)
print()
# Tỉ lệ 7:3
model_with_and_without_LDA(X, y, 0.3, lr_model)
print()
# Tỉ lệ 6:4
model_with_and_without_LDA(X, y, 0.4, lr_model)
```

✓ 0.1s

```
---Chia theo tỉ lệ: 0.2---
Độ chính xác chéo với dữ liệu gốc: 0.8844660194174757
Độ chính xác với dữ liệu gốc: 0.825
Độ chính xác với dữ liệu đã giảm chiều: 0.84375

---Chia theo tỉ lệ: 0.3---
Độ chính xác chéo với dữ liệu gốc: 0.8774937769412908
Độ chính xác với dữ liệu gốc: 0.85
Độ chính xác với dữ liệu đã giảm chiều: 0.8541666666666666

---Chia theo tỉ lệ: 0.4---
Độ chính xác chéo với dữ liệu gốc: 0.8789067142008318
Độ chính xác với dữ liệu gốc: 0.84375
Độ chính xác với dữ liệu đã giảm chiều: 0.846875
```

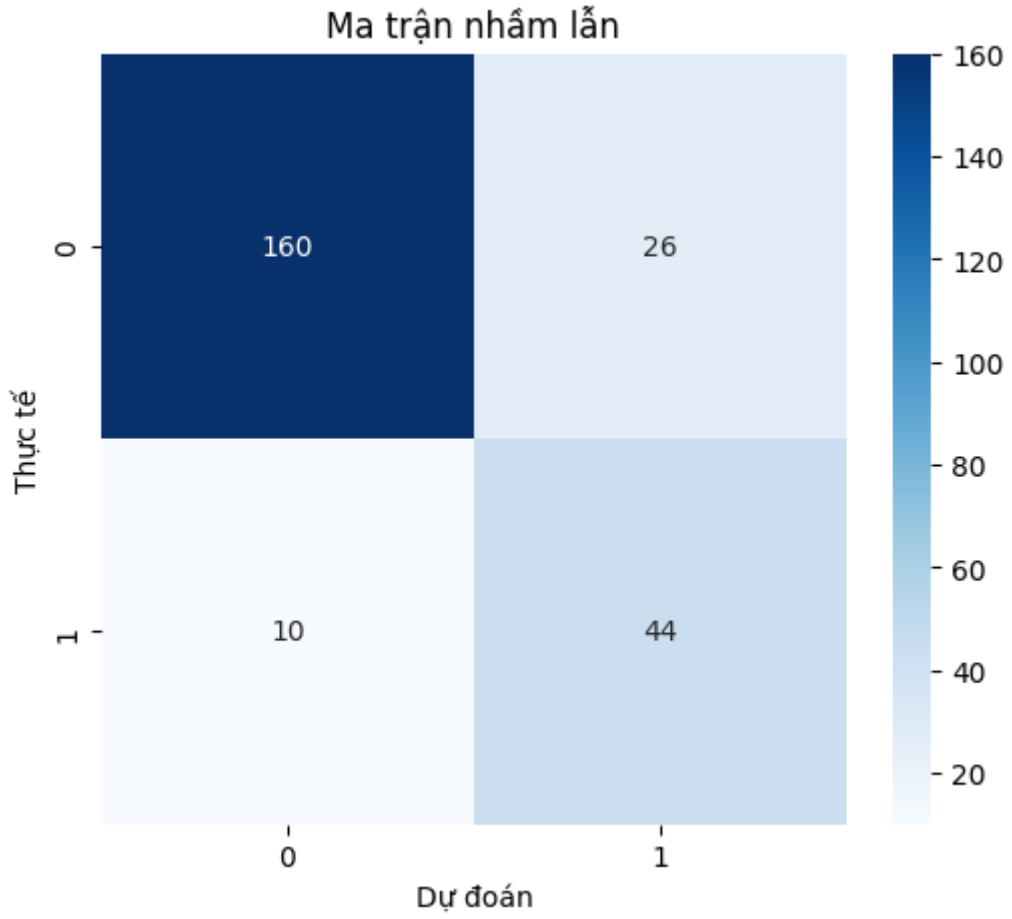
Nhận xét kết quả Logistic Regression

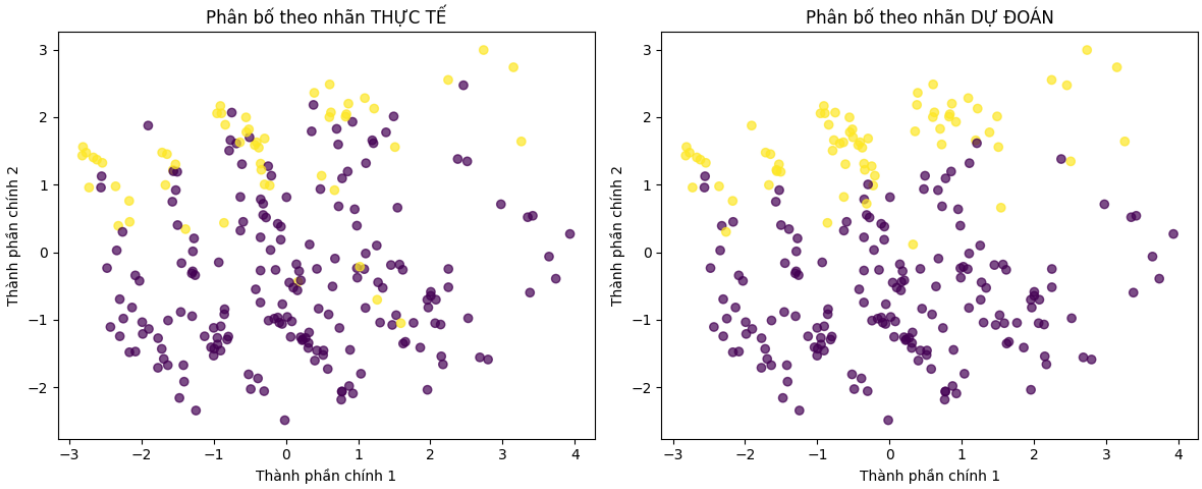
Tỉ lệ test	Cross-val (gốc)	Test (gốc)	Test (LDA)
0.2	~0.884	0.825	0.844
0.3	~0.877	0.850	0.854
0.4	~0.879	0.844	0.847

Kết quả cho thấy độ chính xác chéo của mô hình luôn dao động ổn định quanh khoảng 0.88, chứng tỏ Logistic Regression hoạt động ổn định và bền vững với nhiều cách chia dữ liệu khác nhau. Độ chính xác trên tập kiểm tra cũng đạt mức 0.82–0.85, tương đương và sát với kết quả cross-validation, do đó không xuất hiện overfitting.

Việc giảm chiều bằng LDA giúp hiệu suất mô hình tăng nhẹ hoặc giữ ổn định, đặc biệt ở tỉ lệ chia 0.2 và 0.3. Điều này cho thấy LDA đã hỗ trợ làm rõ ranh giới giữa hai lớp và loại bỏ nhiễu trong dữ liệu, nhưng không làm thay đổi quá mạnh cấu trúc phân tách vốn đã phù hợp của Logistic Regression.

Trực quan hóa và đánh giá tương quan





Nhận xét trực quan và đánh giá độ phù hợp của mô hình

Từ ma trận nhầm lẫn, mô hình Logistic Regression dự đoán đúng đa số mẫu lớp 0 (160 đúng, 26 nhầm) và cũng phân loại khá tốt lớp 1 (44 đúng, chỉ 10 nhầm). Điều này cho thấy mô hình duy trì được mức cân bằng giữa hai lớp, không nghiêng mạnh về một lớp nào.

Quan sát hai biểu đồ phân bố sau khi giảm chiều cho thấy ranh giới giữa hai lớp tương đối rõ ràng. Biểu đồ nhãn dự đoán có dạng phân bố gần tương đồng với nhãn thực tế. Các điểm bị dự đoán sai chủ yếu nằm ở vùng giao nhau giữa hai lớp – nơi giá trị đặc trưng chưa đủ tách biệt. Điều này phản ánh hạn chế tự nhiên của dữ liệu, chứ không phải do mô hình học sai.

Nhìn chung, Logistic Regression phù hợp với dạng dữ liệu tuyến tính tách lớp như trường hợp này, mô hình thể hiện khả năng khái quát tốt và không có dấu hiệu overfitting, vì sai khác giữa phân bố dự đoán và phân bố thực tế là nhỏ và có tính logic.

Phần IV. Tổng hợp và thảo luận kết quả

Trong báo cáo, hai mô hình phân loại K-Nearest Neighbors (KNN) và Logistic Regression đã được áp dụng để dự đoán khả năng mắc rối loạn phổ tự kỷ (ASD) dựa trên dữ liệu khảo sát hành vi. Bên cạnh đó, kỹ thuật giảm chiều bằng Linear Discriminant

Analysis (LDA) cũng được thử nghiệm nhằm đánh giá ảnh hưởng của biến đổi không gian đặc trưng đến hiệu quả dự đoán.

1. So sánh hiệu quả mô hình trước và sau khi giảm chiều

Mô hình	Giảm chiều	Accuracy	Precision (ASD=1)	Recall (ASD=1)	F1-score (ASD=1)
KNN	Không	0.80	0.53	0.91	0.67
KNN	LDA	0.85	0.65	0.78	0.71
Logistic Regression	Không	0.85	0.63	0.81	0.71
Logistic Regression	LDA	0.85	0.64	0.80	0.71

2. Nhận xét và phân tích

- KNN không giảm chiều có recall rất cao (0.91) đối với lớp ASD, nghĩa là mô hình nhận diện hầu hết các trường hợp mắc ASD. Tuy nhiên, precision thấp (0.53) cho thấy mô hình dễ dự đoán nhầm người không mắc ASD thành có ASD (false positives). Điều này làm giảm độ tin cậy khi triển khai thực tế.
- KNN sau giảm chiều LDA cho kết quả cân bằng hơn, tăng cả accuracy (từ 0.80 → 0.85) và precision/recall của lớp ASD. Điều này cho thấy LDA giúp làm rõ ranh giới phân lớp trong không gian đặc trưng.
- Logistic Regression thể hiện hiệu quả ổn định, với accuracy 0.85 trước và sau giảm chiều. Precision và recall của lớp ASD tương đối hài hòa, chứng tỏ mô hình này phù hợp với các bài toán nhị phân có biên quyết định tương đối tuyến tính.
- So sánh hai mô hình:
- KNN có thể nhận diện ASD mạnh hơn (recall cao), phù hợp bước sàng lọc ban đầu**.**
- Logistic Regression ổn định và dễ diễn giải, phù hợp đánh giá xác nhận.

3. Ý nghĩa đối với bài toán sàng lọc ASD

Kết quả cho thấy dữ liệu khảo sát hành vi đơn giản vẫn có khả năng hỗ trợ tốt trong phát hiện sớm ASD, đặc biệt khi kết hợp với các phương pháp giảm chiều giúp làm nổi bật

sự khác biệt hành vi giữa nhóm mắc và không mắc. Tuy nhiên, mô hình chỉ đóng vai trò hỗ trợ sàng lọc, không thay thế được chẩn đoán lâm sàng bởi chuyên gia.

Phần V. Kết Luận và hướng phát triển

1. Kết luận

Báo cáo đã tiến hành phân tích và xây dựng mô hình dự đoán rối loạn phổ tự kỷ dựa trên dữ liệu khảo sát hành vi. Kết quả cho thấy Logistic Regression mang lại hiệu quả ổn định và dễ giải thích, trong khi KNN có khả năng phát hiện trường hợp ASD cao hơn nhưng dễ xảy ra dự đoán sai đối với người không mắc. Việc giảm chiều bằng LDA giúp tăng khả năng phân tách lớp và cải thiện hiệu suất cho KNN.

Nhìn chung, việc ứng dụng học máy trong sàng lọc ASD là khả thi và có thể hỗ trợ phát hiện sớm, đặc biệt tại các cơ sở y tế hoặc cộng đồng chưa có điều kiện tiếp cận chuyên gia.

2. Hạn chế nghiên cứu

Mặc dù đạt một số kết quả tích cực, nghiên cứu vẫn tồn tại một số hạn chế:

- Dữ liệu phụ thuộc vào tự khai báo, dễ bị ảnh hưởng bởi cảm xúc, mức độ hợp tác và khả năng tự nhận thức của đối tượng khảo sát.
- Phân bố lớp không cân bằng, số mẫu ASD ít hơn đáng kể, dẫn đến mô hình bị thiên lệch → làm recall và precision thay đổi theo hướng nhạy hoặc thận trọng quá mức.
- Dữ liệu không bao gồm yếu tố môi trường – gen – hành vi xã hội thực tế, vốn quan trọng trong chẩn đoán lâm sàng.
- Mô hình đang dựa trên đặc trưng tuyến tính, chưa khai thác được các mối quan hệ phi tuyến sâu hơn.

Việc nhận diện và thừa nhận hạn chế giúp định hướng cải thiện mô hình chính xác và phù hợp hơn.

3. Ý nghĩa thực tiễn

- Mô hình có thể được sử dụng như công cụ hỗ trợ sàng lọc ban đầu cho phụ huynh, giáo viên hoặc bác sĩ lâm sàng trước khi tiến hành đánh giá chuyên sâu.
- Kết quả có thể giảm thời gian phát hiện sớm, tăng cơ hội can thiệp kịp thời, đặc biệt quan trọng trong điều trị ASD.

4. Hướng phát triển và mở rộng

Trong tương lai, có thể mở rộng nghiên cứu theo các hướng sau:

- Thử nghiệm các mô hình mạnh hơn như SVM, Random Forest, XGBoost, hoặc MLP/Neural Network**.**
- Áp dụng feature selection để xác định những đặc trưng hành vi quan trọng nhất thay vì chỉ giảm chiều đại số.
- Tăng kích thước và đa dạng hóa dataset theo khu vực, độ tuổi và nền văn hóa.
- Kết hợp thêm dữ liệu hành vi thực tế (video, cử chỉ, giọng nói, ánh mắt…) nhằm mô phỏng cách đánh giá của chuyên gia.
- Phát triển ứng dụng web/mobile giúp người dùng tự đánh giá ban đầu một cách thuận tiện.

Tài liệu tham khảo

1. Vũ Hữu Tiệp - Machine Learning cơ bản

2. Tom M. Mitchell - Machine Learning

3. Michael Bowles - Machine Learning in Python