

**ĐẠI HỌC HUẾ**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

**🙢**

**BÁO CÁO ĐỒ ÁN**

**Học kỳ I, năm học 2023 - 2024**

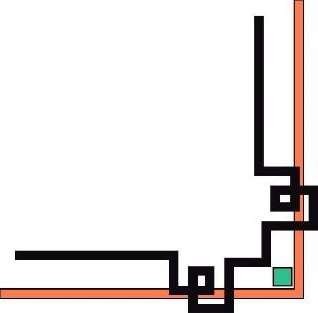
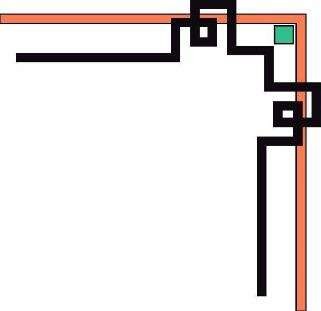
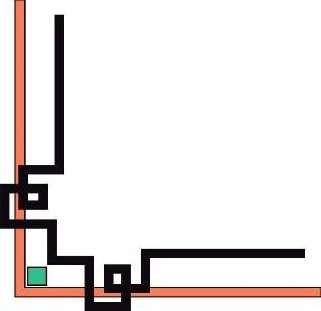
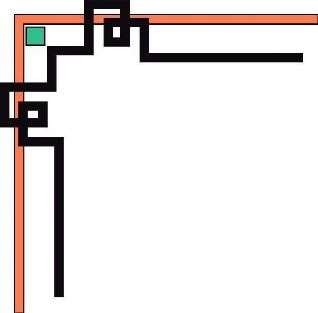
**Học phần:**

**HỌC MÁY 1**

**Số phách**

*(Do hội đồng chấm thi ghi)*

**Thừa Thiên Huế, tháng 01 năm 2024**



**ĐẠI HỌC HUẾ**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

**🙢**

**BÁO CÁO ĐỒ ÁN**

**Học kỳ I, năm học 2023 – 2024**

**Học phần:**

**HỌC MÁY 1**

**Giảng viên hướng dẫn:** TS. Hoàng Hữu Trung

**Lớp:** Khoa học dữ liệu và trí tuệ nhân tạo - K3

**Sinh viên thực hiện:** Nguyễn Văn Minh Khánh

*(ký và ghi rõ họ tên)*

**Số phách**

*(Do hội đồng chấm thi ghi)*

**Thừa Thiên Huế, tháng 01 năm 2022**

# LỜI CẢM ƠN

“Em đã rất cố gắng và nỗ lực trong bài báo cáo đồ án này. Tuy nhiên, sẽ không thể thực hiện được nếu không có sự hỗ trợ, giúp đỡ ân cần của giảng viên bộ môn - Ngôn ngữ lập trình Python cũng như Ban giám hiệu Khoa Kỹ thuật và Công nghệ - Đại học Huế vì đã tạo điều kiện về cơ sở vật chất, như môi trường học tập thân thiện, giúp em phát huy hết khả năng học tập và rèn luyện nhân cách một cách hiệu quả.

Em muốn bày tỏ lòng biết ơn chân thành đối với giảng viên bộ môn - Ngôn ngữ lập trình Python và toàn thể giáo viên của Khoa Kỹ thuật và Công nghệ - Đại học Huế.

Em muốn bày tỏ lòng biết ơn đến gia đình và bạn bè vì đã luôn đồng hành, động viên và quan tâm em trên con đường học tập và trong cuộc sống.

Và lời cảm ơn đặc biệt cuối cùng em xin dành tặng cho bản thân chính mình vì đã không bỏ cuộc vào những lúc bản thân suy sụp, mệt mỏi nhất, cảm ơn bản thân đã luôn cố gắng để vượt qua những khó khăn tưởng chừng không thể bước tiếp, cảm ơn vì tất cả.”

# DANH MỤC HÌNH ẢNH

# DANH MỤC BẢNG BIỂU

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc155636822)

[DANH MỤC HÌNH ẢNH ii](#_Toc155636823)

[DANH MỤC BẢNG BIỂU iii](#_Toc155636824)

[MỤC LỤC iv](#_Toc155636825)

[CHƯƠNG 1: HỌC CÓ GIÁM SÁT (SUPERVISED LEARNING) 6](#_Toc155636826)

[1.1 Giới thiệu tập dữ liệu 6](#_Toc155636827)

[1.1.1 Tập dữ liệu 6](#_Toc155636828)

[1.1.2 Thành phần tập dữ liệu 6](#_Toc155636829)

[1.2 Đọc dữ liệu 7](#_Toc155636830)

[1.2.1 Nhập các thư viện cần thiết 7](#_Toc155636831)

[1.2.2 Nhập tập dữ liệu 8](#_Toc155636832)

[1.3 Đọc hiểu tập dữ liệu 8](#_Toc155636833)

[1.3.1 Thông tin dữ liệu 8](#_Toc155636834)

[1.3.2 Phân tách tập dữ liệu 10](#_Toc155636835)

[1.3.3 Kiểm tra giá trị ngoại lệ trong biến liên tục 15](#_Toc155636836)

[1.4 Tiền xử lý dữ liệu 16](#_Toc155636837)

[1.4.1 Kiểm tra và xử lý giá trị trùng lặp 16](#_Toc155636838)

[1.4.2 Xử lý giá trị bị thiếu 17](#_Toc155636839)

[1.4.3 Chuyển đổi dữ liệu 17](#_Toc155636840)

[1.4.4 Xử lý các giá trị ngoại lệ 18](#_Toc155636841)

[1.4.5 Lưu tập dữ liệu sau khi đã tiền xử lý 20](#_Toc155636842)

[1.5 Mối tương quan của tập dữ liệu 21](#_Toc155636843)

[1.5.1 Biểu đồ phân tán của cột age và thalachh 21](#_Toc155636844)

[1.5.2 Biểu đồ phân phối tần suất 21](#_Toc155636845)

[1.5.3 Biểu đồ tròn thể tỉ lệt của cột output so với cột sex 22](#_Toc155636846)

[1.5.4 Biểu đồ mật độ (KDE) của các trường dữ liệu so với output 23](#_Toc155636847)

[1.5.5 Biểu đồ thể hiện mức độ tương quan của toàn bộ tập dữ liệu 25](#_Toc155636848)

[1.6 Mô hình 26](#_Toc155636849)

[1.6.1 Chuẩn bị mô hình 26](#_Toc155636850)

[1.6.2 Linear Regression 27](#_Toc155636851)

[1.6.3 Logistic Regression 28](#_Toc155636852)

[1.6.4 Support Vector Machine 30](#_Toc155636853)

[1.6.5 K-Nearest Neighbours 33](#_Toc155636854)

[1.6.6 Bernoulli Naive Bayes 36](#_Toc155636855)

[1.7 cc 39](#_Toc155636856)

[CHƯƠNG 2: HỌC KHÔNG GIÁM SÁT (UNSUPERVISED LEARNING) 41](#_Toc155636857)

[TÀI LIỆU THAM KHẢO 43](#_Toc155636858)

[KẾT QUẢ KIỂM TRA ĐẠO VĂN 44](#_Toc155636859)

# CHƯƠNG 1: HỌC CÓ GIÁM SÁT (SUPERVISED LEARNING)

## Giới thiệu tập dữ liệu

### 1.1.1 Tập dữ liệu

Nguồn: [Heart Disease - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/45/heart+disease)

Link thực hiện các thuật toán được tải lên Github dưới đây:

[machine-learning-1/supervised\_learning at main · minhkhanh-coder/machine-learning-1 (github.com)](https://github.com/minhkhanh-coder/machine-learning-1/tree/main/supervised_learning)

Tập dữ liệu ban đầu: [heart.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/supervised_learning/heart.csv)

A screenshot of a computer

Description automatically generated

Hình : Ảnh tập dữ liệu trước tiền xử lý

Đây là một bộ dữ liệu đa biến, được sử dụng để phân tích dữ liệu số đa biến. Bộ dữ liệu này bao gồm 14 cột, bao gồm tuổi, giới tính, loại đau ngực, huyết áp nghỉ, cholesterol huyết thanh, đường huyết nhanh, kết quả điện tâm đồ nghỉ, nhịp tim tối đa đạt được, angina do tập thể dục gây ra, oldpeak - giảm độ chênh lệch ST do tập thể dục so với nghỉ, độ dốc của đoạn ST cao điểm tập thể dục, số mạch máu chính và Thalassemia.

### 1.1.2 Thành phần tập dữ liệu

Một trong những nhiệm vụ chính trên bộ dữ liệu này là dự đoán dựa trên các thuộc tính cung cấp của một bệnh nhân để xác định liệu người đó có bệnh đau tim hay không, và nhiệm vụ thí nghiệm khác là chẩn đoán và tìm ra các thông tin chi tiết từ bộ dữ liệu này để giúp hiểu vấn đề sâu hơn. Dưới đây là mô tả chi tiết từng cột:

* **age:** Tuổi của bệnh nhân.
* **sex:** Giới tính (male hoặc female).
* **cp:** Loại đau ngực mà bệnh nhân có thể trải qua (typical angina, non-anginal pain, atypical angina, asymptomatic).
* **trtbps:** Huyết áp nghỉ của bệnh nhân.
* **chol:** Lượng cholesterol trong huyết thanh.
* **fbs:** Mức đường huyết nhanh (TRUE/FALSE), cho biết liệu có đường huyết nhanh cao hay không.
* **restecg:** Kết quả các xét nghiệm điện tâm đồ khi bệnh nhân đang nghỉ (normal, hypertrophy, having ST-T wave abnormality).
* **thalachh:** Nhịp tim tối đa đạt được trong quá trình kiểm tra.
* **exng:** Cho biết liệu có gặp đau tim do tập thể dục hay không (yes/no).
* **oldpeak:** Mức giảm đau tim do tập thể dục so với lúc nghỉ.
* **slp:** Độ dốc của đoạn ST segment đạt tới đỉnh cao nhất trong quá trình tập thể dục (flat, downsloping, upsloping).
* **caa:** Số mạch mạch máu chính (number of major vessels).
* **thall:** Một chứng rối loạn về máu có tên là thalassemia (normal, fixed detect, reversible defect).
* **output:** Dự đoán khả năng mắc bệnh đau tim (less chance of heart attack /more chance of heart attack).

Dữ liệu này có thể giúp phân loại và đánh giá nguy cơ mắc bệnh tim mạch dựa trên các chỉ số y tế của bệnh nhân.

## 1.2 Đọc dữ liệu

### 1.2.1 Nhập các thư viện cần thiết

Để giải quyết các bài toán Supervise Learing ta cần nhập các thư viện cần thiết như dưới đây.

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.preprocessing import RobustScaler, StandardScaler  from lazypredict.Supervised import LazyClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LogisticRegression  from sklearn.naive\_bayes import BernoulliNB  from sklearn.linear\_model import LinearRegression  from sklearn.neighbors import KNeighborsClassifier  from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report  from sklearn.metrics import mean\_squared\_error, r2\_score  from sklearn.svm import SVC, LinearSVC  from scipy import stats  from scipy.stats import zscore  from scipy.stats.mstats import winsorize  from sklearn.preprocessing import MinMaxScaler  from matplotlib.colors import ListedColormap  import matplotlib.pyplot as plt  import seaborn as sns  import time  import warnings  warnings.filterwarnings("ignore") |

### 1.2.2 Nhập tập dữ liệu

Truyền tập dữ liệu [heart.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/supervised_learning/heart.csv) vô code để thức hiện bài toán.

|  |
| --- |
| df = pd.read\_csv('heart.csv') |

## 1.3 Đọc hiểu tập dữ liệu

### 1.3.1 Thông tin dữ liệu

Để hiểu hơn về dữ liệu ta kiểm tra 5 dòng đầu của tập dữ liệu.

|  |
| --- |
| df.head() |

Kết quả:

A screenshot of a computer

Description automatically generated

Hình : Kết quả in ra 5 dòng đầu tập dữ liệu

* Kết quả hiển thị 5 dòng đầu của dữ liệu về thông tin của các biến về tuổi, giới tính, ... .

In ra thông tin tổng quan tập dữ liệu:

|  |
| --- |
| df.info() |

Kết quả:

A screenshot of a computer

Description automatically generated

Hình : Kết quả in ra thông tin tổng quan tập dữ liệu

* Kết quả cho ta biết được tập dữ liệu có 303 dòng và 14 cột.
* Có 5 kiểu dữ liệu 5 số nguyên (int64) như cột age, trtbps, chol, thalachh, caa.
* 1 kiểu dữ liệu là float64 là cột odlpeak.
* Còn lại là các cột kiểu dữ liệu object.

Kiểm tra số lượng giá trị duy nhất trong mỗi cột.

|  |
| --- |
| dict = {}  for i in list(df.columns):      dict[i] = df[i].value\_counts().shape[0]  pd.DataFrame(dict,index=["unique count"]).transpose() |

Kết quả:

A screenshot of a phone

Description automatically generated

Hình : Kết quả kiểm tra số lượng duy nhất trong mỗi cột

* Hàm này được thực hiện để có thể in ra các các giá trị duy nhất xuất hiện trong cột đó, mục đích để xem đâu là cột phân loại đâu là cột liên tục.

Tổng quan một số thông tin cơ bản của tập dữ liệu.

|  |
| --- |
| df.describe() |

Kết quả:

A screenshot of a computer screen

Description automatically generated

Hình : Kết quả các thông tin cơ bản của dữ liệu

* Kết quả này cho ta biết được một số thông tin cơ bản của các trường dự liệu như mean, min, max... .

Kiểm tra giá trị bị thiếu.

|  |
| --- |
| # Kiểm tra giá trị NaN  df.isna().sum() |

Kết quả:

A list of black text

Description automatically generated

Hình : Kết quả in ra giá trị bị thiếu

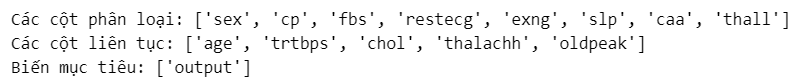
* Kết quả cho ta thấy được là có 2 giá trị bị thiếu trong cột thall và tất cả các cột còn lại không có giá trị nào bị thiếu.

### 1.3.2 Phân tách tập dữ liệu

Để dễ cho quá trình thực hiện thuật toán ta chia các cột dữ liệu thành các biến phân loại, biến liên tục và biến mục tiêu.

|  |
| --- |
| cat\_cols = ['sex','cp','fbs','restecg','exng', 'slp', 'caa','thall']  con\_cols = ["age","trtbps","chol","thalachh","oldpeak"]  target\_col = ["output"]  print("Các cột phân loại:", cat\_cols)  print("Các cột liên tục:", con\_cols)  print("Biến mục tiêu:", target\_col) |

Kết quả:



Hình : Kết quả phân tách dữ liệu

* Các biến phân loại được đặt trong biến cat\_cols, phân loại nằm trong biến con\_cols và biến mục tiêu output nằm trong biến target\_col

Để hiểu hơn về các biến mục tiêu, biến phân loại và biến liên tục ta trực quan hóa các thông tin về các trường dữ liệu đó như sau:

Trực quan các biến mục tiêu.

|  |
| --- |
| x = df[target\_col].value\_counts()  # Set seaborn style with grid  sns.set(style="whitegrid")  # Plot the countplot  plt.figure(figsize=(8, 6))  ax = sns.countplot(data=df, x="output", palette="viridis")  # Add text labels on top of each bar  for p in ax.patches:      ax.annotate(f'{p.get\_height()}', (p.get\_x() + p.get\_width() / 2., p.get\_height()),                  ha='center', va='baseline', fontsize=12, color='black', xytext=(0, 5),                  textcoords='offset points')  # Add labels and title  plt.xlabel("output")  plt.ylabel("Count")  plt.title("BarChart: Target variables in a Dataset")  # Show the plot  plt.show() |

Kết quả:

A graph of a bar chart

Description automatically generated

Hình : Kết quả trực quan biến mục tiêu

* Biểu đồ cột này cung cấp cho ta thấy biến mục tiêu output có 165 more chance of heart attack(nhiều nguy cơ mắc bệnh tim) và less chance of heart attack (ít nguy cơ mặt bệnh tim).

Trực quan các biến phân loại.

|  |
| --- |
| # Tạo subplots  fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(15, 30))  # Duyệt qua các cột và vẽ countplot  for i, ax in zip(df[cat\_cols].columns, axes.flatten()):      sns.countplot(x=i, data=df[cat\_cols], ax=ax)      ax.set\_title(f'Countplot for {i}')      for p in ax.patches:          height = p.get\_height()          ax.annotate(f'{height}', (p.get\_x() + p.get\_width() / 2., height),                      ha='center', va='center', xytext=(0, 10), textcoords='offset points')        # Thêm lưới ngang      for row in axes:          for ax in row:              ax.yaxis.grid(True, linestyle='--', alpha=0.5)  plt.suptitle("BarChart: Categorical variables in a Dataset", y=0.9, fontsize=16)  # Thêm tiêu đề chung  plt.show() |

Kết quả:

A screenshot of a graph

Description automatically generated

Hình : Kết quả trực quan các biến phân loại

* Kết quả này cho ta thấy thấy được các biểu đồ của các biến phân loại như cột sex(giới tính) có 207 số lượng là male(nam) 96 số lượng là female(nữ), cột cp(loại đau ngực) có 23 loại typical angina(đau ngực điển hình) 87 loại non-anginal pain(đau không phải là angina) 50 loại atypical angina(đau ngực không điển hình) 143 loại asymptomatic(không có triệu chứng) và các cột khác.

Trực quan các biến liên tục.

|  |
| --- |
| # Tạo subplots  fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 15))  # Duyệt qua các cột và vẽ histogram  for col, ax in zip(con\_cols, axes.flatten()):      sns.histplot(df[col], bins=20, kde=True, ax=ax)      ax.set\_title(f'Histogram for {col}')      for p in ax.patches:          height = p.get\_height()          ax.annotate(f'{height}', (p.get\_x() + p.get\_width() / 2., height),                      ha='center', va='center', xytext=(0, 10), textcoords='offset points')        # Thêm lưới ngang      for row in axes:          for ax in row:              ax.yaxis.grid(True, linestyle='--', alpha=0.3)  # Tự chỉnh layout  plt.tight\_layout()  plt.suptitle("Histogram: Continuous variables in Dataset", y=1.02, fontsize=16)  # Thêm tiêu đề chung  plt.show() |

Kết quả:

A graph of a number of data

Description automatically generated with medium confidence

Hình : Kết quả trực quan các biến liên tục

* Cũng như biểu đồ của các biến phân loại biểu đồ này cũng cung cấp cái nhìn về số lượng của các thành phần trong biến liên tục.

### 1.3.3 Kiểm tra giá trị ngoại lệ trong biến liên tục

Để hiểu hơn về dữ liệu ta đi tìm các giá trị ngoại lệ trong dữ liệu và và trực quan nó để tiện tiền xử lý sau này.

Biểu đồ hiển thị các giá trị ngoại lệ trong tập dữ liệu.

|  |
| --- |
| plt.figure(figsize=(8, 10))  for i, column in enumerate(df[con\_cols], 1):      plt.subplot(2, 3, i)      sns.boxplot(y=df[column])      plt.title(f'Boxplot for {column}')  plt.tight\_layout()  plt.suptitle("Outliers of the Dataset ", y=1.02, fontsize=16)  # Thêm tiêu đề chung  plt.show() |

Kết quả:

A group of blue boxes with black lines

Description automatically generated

Hình : Kết quả trực quan các giá trị ngoại lệ

* Từ biểu đồ boxplot này ta thấy được các giá trị ngoại lệ tồn tại trong các trường dữ liệu của tập dữ liệu như cột age ta thấy không có giá trị ngoại lệ, và các cột còn lại đều có giá trị ngoại lệ được thể hiện bằng những chấm trên biểu đồ.

## 1.4 Tiền xử lý dữ liệu

Có lẽ từ các phần trên đã cung cấp cái nhìn rằng tập dữ liệu của chúng ta chưa tốt và để có thể thực hiện các bài toán Supervise Learing ta cần phải tiền xử lý dữ liệu để xử lý các giá trị bị thiếu, giá trị ngoại lệ. Dưới đây cách ta thực hiện tiền xử lý dữ liệu của chúng ta.

### 1.4.1 Kiểm tra và xử lý giá trị trùng lặp

Đầu tiên ta kiểm tra xem có giá trị nào trùng lặp trong tập dữ liệu hay không?

Kiểm tra giá trị trùng lặp.

|  |
| --- |
| df.duplicated().sum() |

Kết quả:

A close up of a word

Description automatically generated

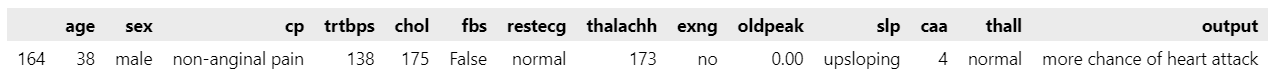
* Như ta thấy thấy trong tập dữ liệu có 1 dòng có các giá trị trùng lặp.

Bước tiếp theo là ta tìm ra xem ra xem giá trị trùng lặp đó ở đâu và dòng đó như thế nào.

In ra dòng chứa giá trị trùng lặp.

|  |
| --- |
| df[df.duplicated()] |

Kết quả:



* Như vậy ta thấy được dòng chứa các giá trị trùng lập là dòng số 164.

Để có thể tối ưu cho các thuật toán sau này ta quyết định loại bỏ dòng trùng lặp đó khỏi tập dữ liệu [heart.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/supervised_learning/heart.csv)

Loại bỏ giá trị trùng lặp.

|  |
| --- |
| # Loại bỏ các dòng chứa giá trị trùng lặp  df = df.drop\_duplicates()  # Kiểm tra lại  df.duplicated().sum() |

Kết quả:

A close up of a word

Description automatically generated

* Kết quả cho thấy ta đã loại bỏ dòng chứa các giá trị trùng lặp ra khỏi tập dự liệu, bởi khi chúng ta kiểm tra lại xem còn giá trị nào trùng lặp nữa không thì kết quả cho ra là: 0

### 1.4.2 Xử lý giá trị bị thiếu

Bước tiếp theo ta cần làm là xử lý các giá trị bị thiếu. Ta cũng đã biết ở phần thông tin về dữ liệu là có 2 giá trị bị thiếu ở cột thall. Với 2 giá trị bị thiếu này ta xử lý bằng cách xóa dòng chứa 2 giá trị đó khỏi tập dữ liệu.

Xử lý giá trị bị thiếu.

|  |
| --- |
| # Loại bỏ giá trị NaN  df = df.dropna()  # Kiểm tra lại  df.isna().sum() |

Kết quả sau khi xử lý:

A black text on a white background

Description automatically generated

* Vậy là chúng ta đã loại bỏ 2 dòng chứa 2 giá trị đó. Sau khi kiểm tra lại thì không còn giá trị nào bị thiếu trong tập dữ liệu nữa.

### 1.4.3 Chuyển đổi dữ liệu

Với tập dữ liệu này của chúng ta nó có các cột chứa kiểu dữ liệu là object nhưng là kiểu phân loại nên ta có thể chuyển về dạng kiểu dữ liệu là số.

Chuyển đổi dữ liệu từ văn bản sang dạng số.

|  |
| --- |
| df['sex'].replace({'male': 1, 'female': 0}, inplace=True)  df['cp'].replace({'asymptomatic': 0, 'atypical angina': 1, 'non-anginal pain': 2, 'typical angina': 3}, inplace=True)  df['restecg'].replace({'hypertrophy': 0, 'normal': 1, 'having ST-T wave abnormality': 2}, inplace=True)  df['fbs'].replace({True: 1, False: 0}, inplace=True)  df['exng'].replace({'yes': 1, 'no': 0}, inplace=True)  df['slp'].replace({'downsloping': 0, 'flat': 1, 'upsloping': 2}, inplace=True)  df['thall'].replace({'fixed defect': 1, 'normal': 2, 'reversible defect': 3}, inplace=True)  df['output'].replace({'less chance of heart attack': 0, 'more chance of heart attack': 1}, inplace=True) |

Lần lượt ta chuyển đổi các cột sex, cp, restecg, fbs, exng, slp, thall, output về dạng kiểu số phù hợp.

* Cột sex ta chuyển 'male' = 1 và 'female' = 0.
* Cột cp ta chuyển 'asymptomatic' = 0, 'atypical angina' = 1, 'non-anginal pain' = 2 và 'typical angina' = 3.
* Cột fbs ta chuyển True = 1 và False = 0
* Cột exng ta chuyển 'yes' = 1 và 'no' = 0
* Cột slp ta chuyển 'downsloping' = 0, 'flat' = 1 và 'upsloping' = 2
* Cột thall ta chuyển 'fixed defect' = 1, 'normal' = 2 và 'reversible defect' = 3
* Cuối cùng ta chuyển cột output ta chuyển 'less chance of heart attack' = 0 và 'more chance of heart attack' = 1

Kết quả:

A screenshot of a table

Description automatically generated

Sau khi chuyển đổi ta có được một tập dữ liệu với tất cả là các kiểu dữ liệu số như phần kết quả ở trên.

### 1.4.4 Xử lý các giá trị ngoại lệ

Một vấn đề cuối mà mà ta cần xử lý đó là giá trị ngoại lệ vì giá trị ngoại lệ ảnh hưởng rất nhiều tới hiệu quả của các mô hình thuật toán. Với dữ liệu này của chúng ta có rất nhiều giá trị ngoại lệ trong các cột như ta đã thấy ở phần trực quan các giá trị ngoại lệ.

Để giải quyết các ngoại lệ này ta xây dựng một hàm để xử lý các giá trị ngoại lệ hàm này áp dụng cho tất cả các cột cần xử lý.

Hàm xử lý giá trị ngoại lệ.

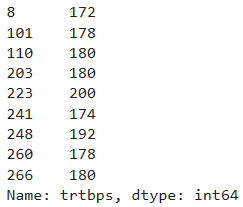
|  |
| --- |
| def iqr(df, var):      q1 = np.quantile(df[var], 0.25)      q3 = np.quantile(df[var], 0.75)      diff = q3 - q1      lower\_v = q1 - (1.5 \* diff)      upper\_v = q3 + (1.5 \* diff)      return df[(df[var] < lower\_v) | (df[var] > upper\_v)] |

Đầu tiên ta bắt đầu xử lý với cột trtbds cột có giá trị ngoại lệ nằm trên Q3(giá trị tối đa của dữ liệu) mà ta thấy được qua biểu đồ boxplot ở trên.

Ta in ra các giá trị ngoại lệ trong cột.

|  |
| --- |
| # In ra các giá trị ngoại lệ của cột  iqr(df,'trtbps')['trtbps'] |

Kết quả:



* Kết quả cho ta thấy có có 9 giá trị ngoại lệ xuất hiện trong cột trtbps.

Sau đó ta in ra giá trị ngoại lệ nhỏ nhất trong của cột. Điều này cho ta được con số gần nhất với điểm Q3 (giá trị tối đa).

|  |
| --- |
| # In ra giá trị ngoại lệ nhỏ nhất của cột  iqr(df, 'trtbps')['trtbps'].min() |

Kết quả:



Kết quả là 172 đây là giá trị gần Q3 nhất.

Dựa vào con số 172 ta lại tìm giá trị giá trị lớn nhất trong cột trtbps mà nhỏ hơn giá trị 172(là giá trị nằm tối đa nằm trong biểu đồ boxplot

|  |
| --- |
| # Tìm giá trị lớn nhất của cột (<172)  df[df['trtbps'] < 172].trtbps.max() |

Kết quả:



Vậy là ta đã có giá trị lớn nhất trong cột ở ngưỡng cho phép.

Tiếp theo nữa với con số 170 ta tính toán phần trăm phân vị của 170, và lưu nó vô biến.

|  |
| --- |
| # Tính phần trăm winsorize cho giá trị 170 trong cột "trtbps"  winsorize\_percentile\_trtbps = (stats.percentileofscore(df['trtbps'], 170)) / 100  # In ra phần trăm winsorize  print(winsorize\_percentile\_trtbps)  trtbps\_winsorize = winsorize(df.trtbps, (0, (1 - winsorize\_percentile\_trtbps)))  # Lưu giá trị ngoại lệ đã được xử lý vào cột  df['trtbps'] = trtbps\_winsorize |

Kết quả:



Xử lý cột chol.

|  |
| --- |
| # In ra giá trị ngoại lệ nhỏ nhất của cột  iqr(df, 'chol')['chol'].min()  # Tìm giá trị lớn nhất của cột (<394)  df[df['chol'] < 394].chol.max()  # Tính phần trăm winsorize cho giá trị 360 trong cột "chol"  winsorize\_percentile\_chol = (stats.percentileofscore(df['chol'], 360)) / 100  # In ra phần trăm winsorize  print(winsorize\_percentile\_chol)  chol\_winsorize = winsorize(df.chol, (0, (1 - winsorize\_percentile\_chol)))  # Lưu giá trị ngoại lệ đã được xử lý vào cột  df['chol'] = chol\_winsorize |

Xử lý cột oldpeak.

|  |
| --- |
| iqr(df, 'oldpeak')['oldpeak']  # In ra giá trị ngoại lệ nhỏ nhất của cột  iqr(df, 'oldpeak')['oldpeak'].min()  # Tìm giá trị lớn nhất của cột (<4.2)  df[df['oldpeak'] < 4.2].oldpeak.max()  # Tính phần trăm winsorize cho giá trị 4 trong cột "oldpeak"  winsorize\_percentile\_oldpeak = (stats.percentileofscore(df['oldpeak'], 4)) / 100  # In ra phần trăm winsorize  print(winsorize\_percentile\_oldpeak)  oldpeak\_winsorize = winsorize(df.oldpeak, (0, (1 - winsorize\_percentile\_oldpeak)))  # Lưu giá trị ngoại lệ đã được xử lý vào cột  df['oldpeak'] = oldpeak\_winsorize |

Bắt đầu bằng xử lý giá trị ngoại lệ cột thalachh, cột duy nhất có giá trị ngoại lệ nằm dưới khoảng Q1 (giá trị tối thiếu của dữ liệu).

In ra các dòng chứa giá trị ngoại lệ.

|  |
| --- |
| # Giá trị ngoại lệ trong cột  iqr(df, 'thalachh')['thalachh'] |

Kết quả:

A close-up of a word

Description automatically generated

* Với hàm trên ta truy tìm được trong cột thalachh có một giá trị ngoại lệ là 71 nằm ở dòng 272.

Có được kết quả này tiếp theo ta đi tìm giá nhỏ nhất lớn hơn 71(giá trị ngoại lên trong cột thalachh.

Tìm giá trị lớn nhất của tập chứa giá trị ngoại lệ.

|  |
| --- |
| # Tìm giá trị bé nhất của cột  df[df['thalachh'] > 71].thalachh.max() |

Kết quả:



Kết quả cho thấy 202 là giá trị đó.

Tiếp theo ta xử lý giá trị ngoại lệ đó bằng cách thay thế giá trị trung bình đó vào trong cột thalachh, dòng chứa giá trị ngoại lệ đó dòng 272.

Xử lý giá trị ngoại lệ.

|  |
| --- |
| # Thay thế giá trị trung bình vào giá trị ngoại lệ  df.at[272, 'thalachh'] = 88 |

Kết quả sau khi xử lý giá trị ngoại lệ của tập dữ liệu:

A group of blue boxes with white text

Description automatically generated

### 1.4.5 Lưu tập dữ liệu sau khi đã tiền xử lý

Lưu tập dữ liệu.

|  |
| --- |
| df.to\_csv('heart\_processed.csv', index=False) |

Kết quả sau khi tiền xử lý: [heart\_preprocessed.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/supervised_learning/heart_processed.csv).

## 1.5 Mối tương quan của tập dữ liệu

### 1.5.1 Biểu đồ phân tán của cột age và thalachh

Vẽ biểu đồ phân tán.

|  |
| --- |
| plt.scatter(df['age'], df['thalachh'], color='b', label='thalachh')  plt.title('Scatter plot: age và thalachh')  plt.xlabel('age')  plt.ylabel('thalachh')  plt.legend()  plt.grid(True)  plt.show() |

Kết quả:

A graph with blue dots

Description automatically generated

### 1.5.2 Biểu đồ phân phối tần suất

Vẽ biểu đồ phân phối tần suất.

|  |
| --- |
| plt.hist(df['age'], bins=10, color='skyblue', edgecolor='black')  plt.title('Histogram: Age')  plt.xlabel('Age')  plt.ylabel('Frequency')  plt.grid(True)  plt.show() |

Kết quả:

A graph of age and age

Description automatically generated

### 1.5.3 Biểu đồ tròn thể tỉ lệt của cột output so với cột sex

Vẽ biểu đồ tròn.

|  |
| --- |
| female\_output\_counts = df[df['sex'] == 0]['output'].value\_counts()  male\_output\_counts = df[df['sex'] == 1]['output'].value\_counts()  labels = ['No Heart Attack', 'Heart Attack']  # Tạo figure và trục  fig, axs = plt.subplots(1, 2, figsize=(12, 6))  # Biểu đồ tròn cho biến phân loại 'female'  axs[0].pie(female\_output\_counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=['lightcoral', 'lightskyblue'])  axs[0].set\_title('Female - Output Distribution')  # Biểu đồ tròn cho biến phân loại 'male'  axs[1].pie(male\_output\_counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=['lightcoral', 'lightskyblue'])  axs[1].set\_title('Male - Output Distribution')  plt.show() |

Kết quả:

A graph of a person and person

Description automatically generated with medium confidence

### 1.5.4 Biểu đồ mật độ (KDE) của các trường dữ liệu so với output

Vẽ biểu đồ mật độ (KDE).

|  |
| --- |
| num\_columns = len(df.columns) - 1  num\_rows = (num\_columns + 1) // 2  plt.figure(figsize=(15, 5\*num\_rows))  for i, column in enumerate(df.columns.drop('output')):      plt.subplot(num\_rows, 2, i+1)      sns.kdeplot(data=df[df['output'] == 0][column], label='0', fill=True)      sns.kdeplot(data=df[df['output'] == 1][column], label='1', fill=True)      plt.title(f'KDE plot of {column} grouped by output')      plt.legend(title="output").set\_alpha(None)  plt.tight\_layout()  plt.show() |

Kết quả:

A screenshot of a graph

Description automatically generated

### 1.5.5 Biểu đồ thể hiện mức độ tương quan của toàn bộ tập dữ liệu

Vẽ biểu đồ tương quan.

|  |
| --- |
| corr = df.corr()  plt.figure(figsize=(10,10))  ax = sns.heatmap(corr, vmin=-1, vmax=1, cmap="coolwarm", linewidths=.1, square=True, annot=True ,fmt=".2f")  plt.yticks(rotation=0)  plt.suptitle("Heatmap", y=0.9, fontsize=16)  # Thêm tiêu đề chung  plt.show() |

Kết quả:

A diagram of a heatmap

Description automatically generated

## 1.6 Mô hình

### 1.6.1 Chuẩn bị mô hình

Chuẩn bị mô hình và in ra các thuật toán phù hợp với tập dữ liệu bằng lazypredict.

|  |
| --- |
| X = df.drop('output', axis=1)  y = df['output']  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=65)  MM\_scaler = MinMaxScaler()  X\_train = MM\_scaler.fit\_transform(X\_train)  X\_test = MM\_scaler.fit\_transform(X\_test)  clf = LazyClassifier(verbose=0,ignore\_warnings=True, custom\_metric=None)  models, predictions = clf.fit(X\_train, X\_test, y\_train, y\_test)  print(models) |

Kết quả:

A screenshot of a computer

Description automatically generated

### 1.6.2 Linear Regression

Độ chính xác và thời gian xử lý của thuật toán.

|  |
| --- |
| start\_time = time.time()  model\_lnr = LinearRegression()  model\_lnr.fit(X\_train, y\_train)  y\_pred\_lnr = model\_lnr.predict(X\_test)  end\_time = time.time()  model\_lnr\_time = end\_time - start\_time  model\_lnr\_mse = mean\_squared\_error(y\_test, y\_pred\_lnr)  model\_lnr\_r2 = r2\_score(y\_test, y\_pred\_lnr)  print(f"MSE - Độ Lỗi Trung Bình Bình Phương: {round(model\_lnr\_mse, 4)}")  print(f"R² - Hệ số xác định: {round(model\_lnr\_r2, 4)}")  print(f"Thời gian xử lý mô hình: {round(model\_lnr\_time, 5)} giây") |

Kết quả:

A black text on a white background

Description automatically generated

Biểu đồ trực quan hóa của thuật toán.

|  |
| --- |
| # Sắp xếp DataFrame theo cột 'age'  df\_sorted = df.sort\_values(by='age')  # Lấy dữ liệu  X\_age\_lnr = df\_sorted['age'].values.reshape(-1, 1)  y\_thalachh\_lnr = df\_sorted['thalachh'].values  # Huấn luyện mô hình Linear Regression  model\_linear = LinearRegression()  model\_linear.fit(X\_age\_lnr, y\_thalachh\_lnr)  # Dự đoán giá trị  y\_pred\_thalachh = model\_linear.predict(X\_age\_lnr)  # Vẽ biểu đồ  plt.figure(figsize=(8, 5))  sns.scatterplot(x=X\_age\_lnr.flatten(), y=y\_thalachh\_lnr, label='Actual')  sns.lineplot(x=X\_age\_lnr.flatten(), y=y\_pred\_thalachh, color='red', label='Linear Regression')  plt.title('Linear Regression: age vs thalachh', fontsize=16)  plt.xlabel('Age')  plt.ylabel('Thalachh')  plt.legend(title='Annotation')  plt.show() |

Kết quả:

A graph with a red line and blue dots

Description automatically generated

### 1.6.3 Logistic Regression

Độ chính xác và thời gian xử lý của thuật toán.

|  |
| --- |
| start\_time = time.time()  model\_lgt = LogisticRegression()  model\_lgt.fit(X\_train, y\_train)  y\_pred\_lgt = model\_lgt.predict(X\_test)  end\_time = time.time()  model\_lgt\_time = end\_time - start\_time  model\_lgt\_accuracy = accuracy\_score(y\_test, y\_pred\_lgt)  classification\_rep\_lgt = classification\_report(y\_test, y\_pred\_lgt)  print(f'Độ chính xác: {model\_lgt\_accuracy}')  print(f"Thời gian xử lý mô hình: {round(model\_lgt\_time, 5)} giây")  print(f'Báo cáo phân loại:\n{classification\_rep\_lgt}') |

Kết quả:

A screenshot of a computer

Description automatically generated

Biểu đồ trực quan hóa của thuật toán.

|  |
| --- |
| plt.figure(figsize=(12,6))  plt.scatter(range(len(y\_pred\_lgt)),y\_pred\_lgt,color="black",label="predict")  plt.scatter(range(len(y\_test)),y\_test,color="red", marker="x",label="actual")  plt.title("Logistic Regression: Chance of heart attack", fontsize=16)  plt.xlabel("number of values")  plt.ylabel("""          output             0: less chance       1: more chance""")  plt.legend(title="ouput result")  plt.show() |

Kết quả:

A graph with red dots and black dots

Description automatically generated

### 1.6.4 Support Vector Machine

Độ chính xác và thời gian xử lý của thuật toán.

|  |
| --- |
| start\_time = time.time()  model\_svm = LinearSVC()  model\_svm.fit(X\_train, y\_train)  y\_pred\_svm = model\_svm.predict(X\_test)  end\_time = time.time()  model\_svm\_time = end\_time - start\_time  model\_svm\_accuracy = accuracy\_score(y\_test, y\_pred\_svm)  classification\_rep\_svm = classification\_report(y\_test, y\_pred\_svm)  print(f'Độ chính xác: {model\_svm\_accuracy}')  print(f"Thời gian xử lý mô hình: {round(model\_svm\_time, 5)} giây")  print(f'Báo cáo phân loại:\n{classification\_rep\_svm}') |

Kết quả:

A screenshot of a computer

Description automatically generated

Biểu đồ trực quan hóa của thuật toán.

|  |
| --- |
| # Chọn các biến đầu vào  X\_svm = df[['age', 'thalachh']]  y\_svm = df['output']  # Chia thành tập huấn luyện và tập kiểm thử  X\_train\_svm, X\_test\_svm, y\_train\_svm, y\_test\_svm = train\_test\_split(X\_svm, y\_svm, test\_size=0.2, random\_state=42)  # Chuẩn hóa dữ liệu  sd\_scaler = StandardScaler()  X\_train\_scaled\_svm = sd\_scaler.fit\_transform(X\_train\_svm)  X\_test\_scaled\_svm = sd\_scaler.transform(X\_test\_svm)  # Sử dụng transform thay vì fit\_transform cho tập kiểm thử  # Huấn luyện mô hình SVM  model\_svm = SVC(kernel='linear', C=1)  model\_svm.fit(X\_train\_scaled\_svm, y\_train\_svm)  # Dự đoán giá trị trên tập kiểm thử  y\_pred\_svm = model\_svm.predict(X\_test\_scaled\_svm)  # Thêm cột dự đoán vào tập kiểm thử  X\_test\_svm['predicted\_output'] = y\_pred\_svm  # Tạo ma trận dự đoán cho toàn bộ không gian 2D  h = 0.01  # Kích thước của mỗi ô lưới trong không gian đặc trưng  x\_min, x\_max = X\_test\_svm['age'].min() - 1, X\_test\_svm['age'].max() + 1  y\_min, y\_max = X\_test\_svm['thalachh'].min() - 1, X\_test\_svm['thalachh'].max() + 1  xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  # Dự đoán giá trị cho từng điểm trên không gian 2D  Z = model\_svm.predict(np.c\_[xx.ravel(), yy.ravel()])  Z = Z.reshape(xx.shape)  # Trực quan hóa thuật toán SVM bằng các điểm phân cụm và đường chia cụm  plt.figure(figsize=(8, 6))  # Vẽ đường chia cụm  plt.contour(xx, yy, Z, colors='k', levels=[0], alpha=0.5, linestyles=['-'])  # Vẽ các điểm dữ liệu phân cụm  sns.scatterplot(x='age', y='thalachh', hue='predicted\_output', data=X\_test\_svm, palette='Set1', s=100)  plt.title('SVM: age vs thalachh')  plt.xlabel('PC1 (age)')  plt.ylabel('PC2 (thalachh)')  plt.legend(title='Predicted Output')  plt.show() |

Kết quả:

A graph with red and blue dots

Description automatically generated

### 1.6.5 K-Nearest Neighbours

Độ chính xác và thời gian xử lý của thuật toán.

|  |
| --- |
| best\_k = None  best\_accuracy = 0  for k in range(2, 21):      model\_knn = KNeighborsClassifier(n\_neighbors=k)      model\_knn.fit(X\_train, y\_train)      y\_pred\_knn = model\_knn.predict(X\_test)      model\_knn\_accuracy = accuracy\_score(y\_test, y\_pred\_knn)      if model\_knn\_accuracy > best\_accuracy:          best\_accuracy = model\_knn\_accuracy          best\_k = k  print(f'K tốt nhất là {best\_k} với độ chính xác {best\_accuracy}')  model\_knn\_best = KNeighborsClassifier(n\_neighbors=best\_k)  model\_knn\_best.fit(X\_train, y\_train)  y\_pred\_knn\_best = model\_knn\_best.predict(X\_test)  classification\_rep\_knn\_best = classification\_report(y\_test, y\_pred\_knn\_best)  print(f'Báo cáo phân loại với k tốt nhất:\n{classification\_rep\_knn\_best}') |

Kết quả:

A screenshot of a computer

Description automatically generated

Biểu đồ trực quan hóa của thuật toán.

|  |
| --- |
| # Chọn các biến độc lập và biến phụ thuộc  numeric\_features = ["age", "thalachh"]  target\_column = "output"  # Chuẩn hóa dữ liệu  scaler = StandardScaler()  df[numeric\_features] = scaler.fit\_transform(df[numeric\_features])  # Xác định các biến độc lập và biến phụ thuộc  X\_knn = df[numeric\_features]  y\_knn = df[target\_column]  # Khởi tạo và huấn luyện mô hình KNN  knn\_model = KNeighborsClassifier(n\_neighbors=15)  knn\_model.fit(X\_knn, y\_knn)  # Dự đoán giá trị trên toàn bộ tập dữ liệu  df["predicted\_output"] = knn\_model.predict(X\_knn)  # Tạo meshgrid để vẽ đường biên quyết định  h = 0.02  x\_min, x\_max = df["age"].min() - 1, df["age"].max() + 1  y\_min, y\_max = df["thalachh"].min() - 1, df["thalachh"].max() + 1  xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  # Dự đoán trên meshgrid  Z = knn\_model.predict(np.c\_[xx.ravel(), yy.ravel()])  Z = Z.reshape(xx.shape)  # Trực quan hóa dữ liệu với màu sắc tương ứng với giá trị dự đoán  plt.figure(figsize=(8, 8))  colors = df["predicted\_output"].map({0: 'blue', 1: 'red'})  scatter = plt.scatter(df["age"], df["thalachh"], c=colors, s=50, alpha=0.5)  # Vẽ đường biên quyết định  plt.contourf(xx, yy, Z, cmap=ListedColormap(['#AAAAFF', '#FFAAAA']), alpha=0.3)  plt.xlabel('PC1 (age)')  plt.ylabel('PC2 (thalachh)')  plt.title('KNN: predictions ouput (n\_neighbors=15)')  legend\_labels = {0: '0', 1: '1'}  plt.legend(title='Predicted output', handles=[      plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='blue', markersize=10),      plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='red', markersize=10)],      labels=[legend\_labels[0], legend\_labels[1]])  plt.show() |

Kết quả:

A diagram of a graph

Description automatically generated with medium confidence

### 1.6.6 Bernoulli Naive Bayes

Độ chính xác và thời gian xử lý của thuật toán.

|  |
| --- |
| start\_time = time.time()  model\_bnb = BernoulliNB()  model\_bnb.fit(X\_train, y\_train)  y\_pred\_bnb = model\_bnb.predict(X\_test)  end\_time = time.time()  model\_bnb\_time = end\_time - start\_time  model\_bnb\_accuracy = accuracy\_score(y\_test, y\_pred\_bnb)  classification\_rep\_bnb = classification\_report(y\_test, y\_pred\_bnb)  print(f'Độ chính xác: {model\_bnb\_accuracy}')  print(f"Thời gian xử lý mô hình: {round(model\_bnb\_time, 5)} giây")  print(f'Báo cáo phân loại:\n{classification\_rep\_bnb}') |

Kết quả:

A screenshot of a computer

Description automatically generated

Biểu đồ trực quan hóa của thuật toán.

|  |
| --- |
| # Chọn các biến độc lập (X) và biến phụ thuộc (y)  X\_bnb = df[['age', 'thalachh']]  y\_bnb = df['output']  # Chia dữ liệu thành tập huấn luyện và tập kiểm tra  X\_train\_bnb, X\_test\_bnb, y\_train\_bnb, y\_test\_bnb = train\_test\_split(X\_bnb, y\_bnb, test\_size=0.2, random\_state=42)  # Khởi tạo và huấn luyện mô hình Bernoulli Naive Bayes  model\_bnb = BernoulliNB()  model\_bnb.fit(X\_train\_bnb, y\_train\_bnb)  # Dự đoán giá trị trên toàn bộ tập dữ liệu  df['predicted\_output'] = model\_bnb.predict(X\_bnb)  # Tạo ma trận dự đoán cho toàn bộ không gian 2D  h = 0.01  # Kích thước của mỗi ô lưới trong không gian đặc trưng  x\_min, x\_max = X\_bnb['age'].min() - 1, X\_bnb['age'].max() + 1  y\_min, y\_max = X\_bnb['thalachh'].min() - 1, X\_bnb['thalachh'].max() + 1  xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  # Dự đoán giá trị cho từng điểm trên không gian 2D  Z = model\_bnb.predict(np.c\_[xx.ravel(), yy.ravel()])  Z = Z.reshape(xx.shape)  # Trực quan hóa thuật toán Bernoulli Naive Bayes bằng các điểm phân cụm và đường chia cụm  plt.figure(figsize=(8, 6))  # Vẽ đường chia cụm  plt.contour(xx, yy, Z, colors='k', levels=[0], alpha=0.5, linestyles=['-'])  # Vẽ các điểm dữ liệu phân cụm  sns.scatterplot(x='age', y='thalachh', hue='predicted\_output', data=df, palette='Set1', s=100)  plt.title('Bernoulli Naive Bayes Clustering: age vs thalachh')  plt.xlabel('PC1 (age)')  plt.ylabel('PC2 (thalachh)')  plt.legend(title='Predicted Output')  plt.show() |

Kết quả:

A graph with red and blue dots

Description automatically generated

## 1.7 Phân tích, đánh giá và kết luận

### 1.7.1 Phân tích, đánh giá các thuật toán

|  |  |
| --- | --- |
|  | Linear Regression |
| MSE - Độ Lỗi Trung Bình Bình Phương | 0.1223 |
| R² - Hệ số xác định | 0.4903 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Logistic Regression | SVM | Knearest Neighbors | Bernoulli NB |
| Độ chính xác | 0.82 | 0.82 | 0.85 | 0.88 |

Linear Regression có MSE khá thấp (0.1223), đồng thời là 0.4903. Điều này có nghĩa là mô hình có khả năng giải thích được 49.03% biến thiên của dữ liệu, tức là nó tốt hơn so với một mô hình đoán trung bình.

Logistic Regression và SVM: cả hai đều có độ chính xác giống nhau, đạt 82%. Điều này cho thấy mô hình khá hiệu quả trong việc dự đoán nhóm.

KNN: có độ chính xác cao nhất, đạt 85% khi sử dụng k = 11.

Bernoulli Naive Bayes: có độ chính xác cao nhất, đạt 88.33%.

Phân tích và Đánh Giá Chung:

* Độ chính xác: Bernoulli Naive Bayes có độ chính xác tốt nhất trên tập dữ liệu.
* Thời gian xử lý: Linear Regression là nhanh nhất, nhưng Bernoulli Naive Bayes cũng có thời gian xử lý chấp nhận được.
* Phân tích mô hình: Linear Regression cho phép hiểu rõ tác động của từng biến độc lập lên biến phụ thuộc. Trong khi đó, các mô hình khác như Naive Bayes và k-NN cung cấp độ chính xác tốt mà không cần nhiều giả định về dữ liệu.

### 1.7.2 Kết luận

1. Độ Chính Xác: Bernoulli Naive Bayes là sự lựa chọn tốt nhất trong trường hợp này, với độ chính xác đạt 88.33%.
2. Các mối quan hệ giữa các trường dữ liệu: Nếu bạn muốn hiểu rõ hơn về mối quan hệ tuyến tính giữa các biến độc lập và biến phụ thuộc, Linear Regression có thể cung cấp thông tin chi tiết về tác động của từng biến.
3. Độ Phức Tạp Của Mô Hình: Nếu ưu tiên độ phức tạp thấp và tính dễ diễn giải, Linear Regression và Logistic Regression là lựa chọn tốt. K-NN và Naive Bayes đơn giản và dễ triển khai.
4. Thời Gian Xử Lý: Nếu tốc độ xử lý là một ưu tiên, Linear Regression và SVM là lựa chọn tối ưu. K-NN và Naive Bayes cũng có thời gian xử lý tốt, nhưng có thể tăng khi có nhiều dữ liệu.
5. Tập dữ liệu: tập dữ liệu này thuộc tập phân loại nên ưu tiên chọn thuật toán BernoulliNB, tiếp đến là k-NN, Logistic Regression, SVM và cuối cùng là Linear Regression.

# CHƯƠNG 2: HỌC KHÔNG GIÁM SÁT (UNSUPERVISED LEARNING)

## 2.1 Giới thiệu tập dữ liệu

### 2.1.1 Tập dữ liệu

Nguồn: [country Data (kaggle.com)](https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data)

Link Github: [machine-learning-1/unsupervised learning at main · minhkhanh-coder/machine-learning-1 (github.com)](https://github.com/minhkhanh-coder/machine-learning-1/tree/main/unsupervised%20learning)

Tập dữ liệu ban đầu: [country\_data.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/unsupervised%20learning/country_data.csv).



Tập dữ liệu đánh giá sức khỏe kinh tế và xã hội của một quốc gia. Ví dụ, tỷ lệ tử vong trẻ em, thu nhập, và sản phẩm quốc nội có thể cung cấp cái nhìn về mức độ phát triển và chất lượng cuộc sống trong một quốc gia. Tỷ lệ lạm phát thường là một chỉ số về tình trạng kinh tế hiện tại, trong khi tuổi thọ trung bình thường là một độ đo của chất lượng cuộc sống.

### 2.1.2 Thành phần tập dữ liệu

Dưới đây là mô tả chi tiết về các trường của tập dữ liệu:

* **country:** Tên quốc gia.
* **child\_mort:** Tỷ lệ tử vong trẻ em dưới 5 tuổi trên 1000 ca sinh sống.
* **exports:** Xuất khẩu hàng hóa và dịch vụ bình quân đầu người. Tính bằng %age trong GDP bình quân đầu người.
* **health:** Tổng chi tiêu y tế bình quân đầu người. Tính bằng % age trong GDP bình quân đầu người.
* **imports:** Nhập khẩu hàng hóa và dịch vụ bình quân đầu người. Tính bằng %age trong GDP bình quân đầu người.
* **income:** Thu nhập ròng mỗi người.
* **inflation:** Đo lường tốc độ tăng trưởng hàng năm của Tổng GDP.
* **life\_expec:** Số năm trung bình mà một đứa trẻ mới sinh có thể sống nếu tình trạng tử vong hiện nay được giữ nguyên.
* **total\_fear:** Số con mà mỗi phụ nữ sẽ sinh ra nếu tỷ suất sinh theo độ tuổi hiện tại không đổi.
* **gdpp:** GDP bình quân đầu người. Được tính bằng Tổng GDP chia cho tổng dân số.

## 2.2 Đọc dữ liệu

### 2.2.1 Nhập các thư viện cần thiết

Nhập các thư viện để giải quyết các bài toán.

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.preprocessing import  StandardScaler  from sklearn.model\_selection import train\_test\_split  from scipy import stats  from scipy.stats.mstats import winsorize  import scipy.cluster.hierarchy as hcluster  from sklearn.cluster import AgglomerativeClustering  import matplotlib.pyplot as plt  import seaborn as sns  import time  import warnings  from yellowbrick.cluster import KElbowVisualizer  from sklearn.metrics import silhouette\_score  import matplotlib.patches as ptch  from sklearn.decomposition import PCA  from sklearn.cluster import DBSCAN  from sklearn.cluster import KMeans  warnings.filterwarnings("ignore") |

### 2.2.2 Nhập tập dữ liệu

Nhập tập dữ liệu [country\_data.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/unsupervised%20learning/country_data.csv).

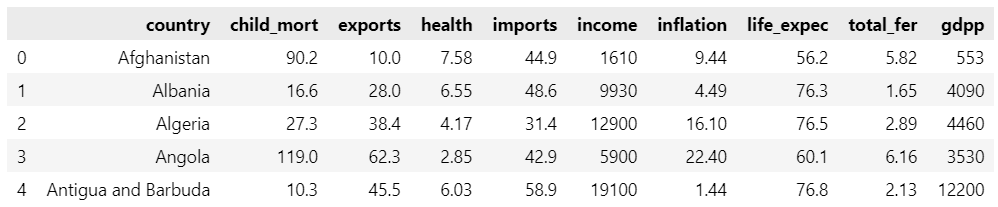
|  |
| --- |
| df = pd.read\_csv('country\_data.csv') |

## 2.3 Đọc hiểu tập dữ liệu

Kiểm tra 5 dòng đầu của tập dữ liệu.

|  |
| --- |
| df.head() |

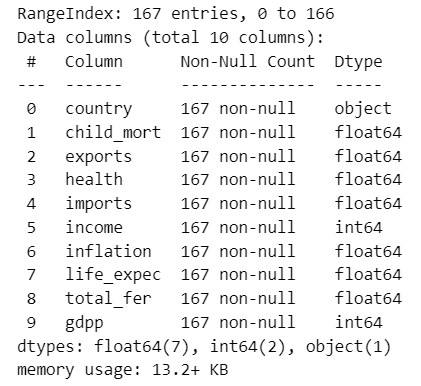
Kết quả:



In ra thông tin tổng quan tập dữ liệu:

|  |
| --- |
| df.info() |

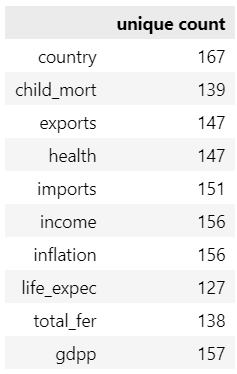
Kết quả:



Kiểm tra số lượng giá trị duy nhất trong mỗi cột.

|  |
| --- |
| dict = {}  for i in list(df.columns):      dict[i] = df[i].value\_counts().shape[0]  pd.DataFrame(dict,index=["unique count"]).transpose() |

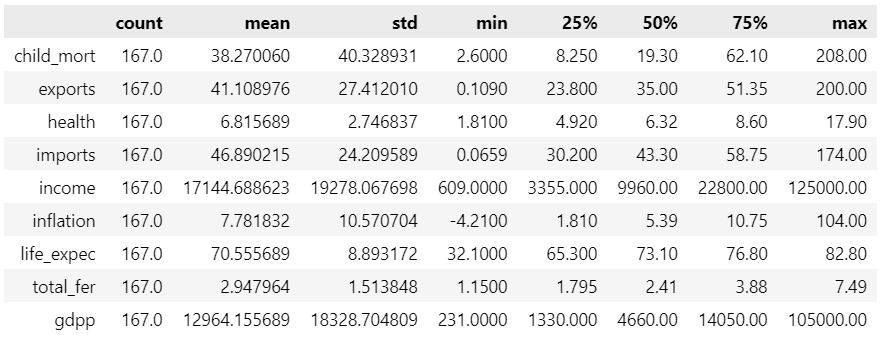
Kết quả:



Tổng quan một số thông tin cơ bản của tập dữ liệu.

|  |
| --- |
| df.describe().transpose |

Kết quả:



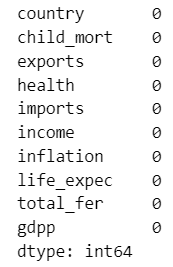
## 2.4 Tiền xử lý dữ liệu

### 2.4.1 Kiểm tra giá trị thiếu

Kiểm tra giá trị thiếu.

|  |
| --- |
| df.isna().sum() |

Kết quả:



* Hình ảnh trên cho ta thấy tập dữ liệu không chứa giá trị bị thiếu nào.

### 2.4.2 Kiểm tra và xử lý những giá trị trùng lặp

Kiểm tra giá trị trùng lặp.

|  |
| --- |
| # Kiểm tra các giá trị trùng lặp  df.duplicated().sum() |

Kết quả:

A close up of a word

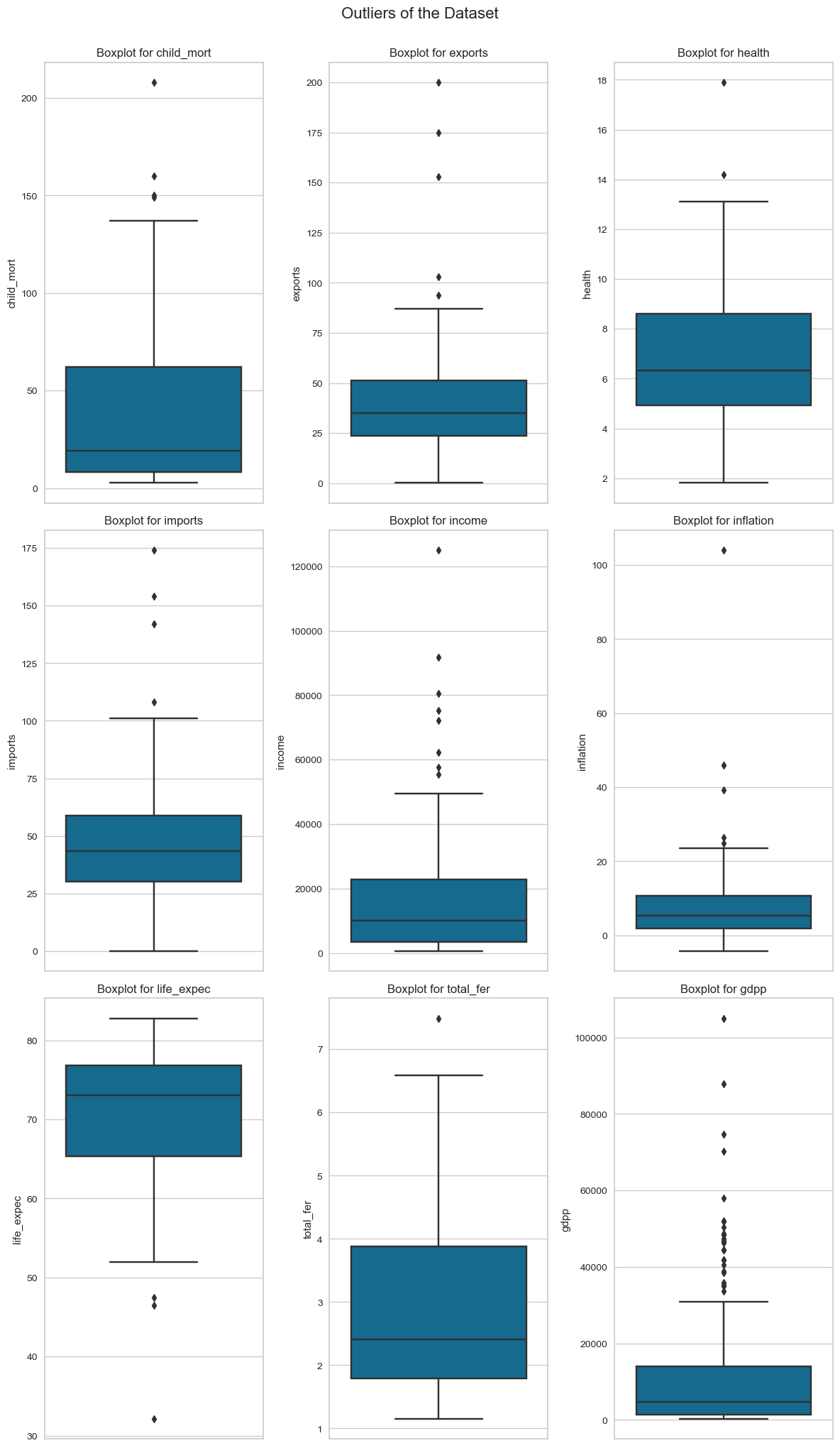
Description automatically generated

### 2.4.3 Kiểm tra giá trị ngoại lệ

Kiểm tra các giá trị ngoại lệ bằng biểu đồ trực quan hóa.

|  |
| --- |
| # Kích thước của figure  plt.figure(figsize=(12, 20))  # Vẽ boxplot cho tất cả các cột  for i, column in enumerate(df.columns[1:], 1):      plt.subplot(3, 3, i)      sns.boxplot(y=df[column])      plt.title(f'Boxplot for {column}')  plt.tight\_layout()  plt.suptitle("Outliers of the Dataset", y=1.02, fontsize=16)  plt.show() |

Kết quả:



* Trước khi xử lý các giá trị ngoại lệ, thì ở mỗi biểu đồ các cột đều có những chấm nhỏ, đó là giá trị ngoại lệ.

Hàm xử lý giá trị ngoại lệ.

|  |
| --- |
| def iqr(df, var):      q1 = np.quantile(df[var], 0.25)      q3 = np.quantile(df[var], 0.75)      diff = q3 - q1      lower\_v = q1 - (1.5 \* diff)      upper\_v = q3 + (1.5 \* diff)      return df[(df[var] < lower\_v) | (df[var] > upper\_v)] |

Xử lý giá trị ngoại lệ cột child\_mort.

|  |
| --- |
| # Giá trị trung bình  iqr(df, 'child\_mort')['child\_mort']  # In ra giá trị ngoại lệ nhỏ nhất của cột  iqr(df, 'child\_mort')['child\_mort'].min()  # Tìm giá trị lớn nhất của cột (<172)  df[df['child\_mort'] < 149].child\_mort.max()  # Tính phần trăm winsorize cho giá trị 137 trong cột "child\_mort"  winsorize\_percentile\_child\_mort = (stats.percentileofscore(df['child\_mort'], 137)) / 100  # In ra phần trăm winsorize  print(winsorize\_percentile\_child\_mort)  child\_mort\_winsorize = winsorize(df.child\_mort, (0, (1 - winsorize\_percentile\_child\_mort)))  # Lưu giá trị ngoại lệ đã được xử lý vào cột  df['child\_mort'] = child\_mort\_winsorize |

Xử lý giá trị ngoại lệ cột exports.

|  |
| --- |
| outlier\_min\_exports = iqr(df, 'exports')['exports'].min()  df[df['exports'] < outlier\_min\_exports].exports.max()  winsorize\_percentile\_exports = (stats.percentileofscore(df['exports'], 86.9)) / 100  exports\_winsorize = winsorize(df.exports, (0, (1 - winsorize\_percentile\_exports)))  df['exports'] = exports\_winsorize |

Xử lý giá trị ngoại lệ cột health.

|  |
| --- |
| outlier\_min\_health = iqr(df, 'health')['health'].min()  df[df['health'] < outlier\_min\_health].health.max()  winsorize\_percentile\_health = (stats.percentileofscore(df['health'], 13.1)) / 100  print(winsorize\_percentile\_health)  health\_winsorize = winsorize(df.health, (0, (1 - winsorize\_percentile\_health)))  df['health'] = health\_winsorize |

Xử lý giá trị ngoại lệ cột imports.

|  |
| --- |
| outlier\_min\_imports = iqr(df, 'imports')['imports'].min()  df[df['imports'] < outlier\_min\_imports].imports.max()  winsorize\_percentile\_imports = (stats.percentileofscore(df['imports'], 101)) / 100  print(winsorize\_percentile\_imports)  imports\_winsorize = winsorize(df.imports, (0, (1 - winsorize\_percentile\_imports)))  df['imports'] = imports\_winsorize |

Xử lý giá trị ngoại lệ cột income.

|  |
| --- |
| outlier\_min\_income = iqr(df, 'income')['income'].min()  df[df['income'] < outlier\_min\_income].income.max()  winsorize\_percentile\_income = (stats.percentileofscore(df['income'], 49400)) / 100  print(winsorize\_percentile\_income)  income\_winsorize = winsorize(df.income, (0, (1 - winsorize\_percentile\_income)))  df['income'] = income\_winsorize |

Xử lý giá trị ngoại lệ cột inflation.

|  |
| --- |
| outlier\_min\_inflation = iqr(df, 'inflation')['inflation'].min()  df[df['inflation'] < outlier\_min\_inflation].inflation.max()  winsorize\_percentile\_inflation = (stats.percentileofscore(df['inflation'], 23.6)) / 100  print(winsorize\_percentile\_inflation)  inflation\_winsorize = winsorize(df.inflation, (0, (1 - winsorize\_percentile\_inflation)))  df['inflation'] = income\_winsorize |

Xử lý giá trị ngoại lệ cột life\_expec.

|  |
| --- |
| iqr(df, 'life\_expec')['life\_expec']  outlier\_min\_life\_expec = iqr(df, 'life\_expec')['life\_expec'].max()  df[df['life\_expec'] > outlier\_min\_life\_expec].life\_expec.min()  df.at[31, 'life\_expec'] = 52  df.at[66, 'life\_expec'] = 52  df.at[87, 'life\_expec'] = 52 |

Xử lý giá trị ngoại lệ cột total\_fer.

|  |
| --- |
| outlier\_min\_total\_fer = iqr(df, 'total\_fer')['total\_fer'].min()  df[df['total\_fer'] < outlier\_min\_total\_fer].total\_fer.max()  winsorize\_percentile\_total\_fer = (stats.percentileofscore(df['total\_fer'], 6.59)) / 100  print(winsorize\_percentile\_total\_fer)  total\_fer\_winsorize = winsorize(df.total\_fer, (0, (1 - winsorize\_percentile\_total\_fer)))  df['total\_fer'] = total\_fer\_winsorize |

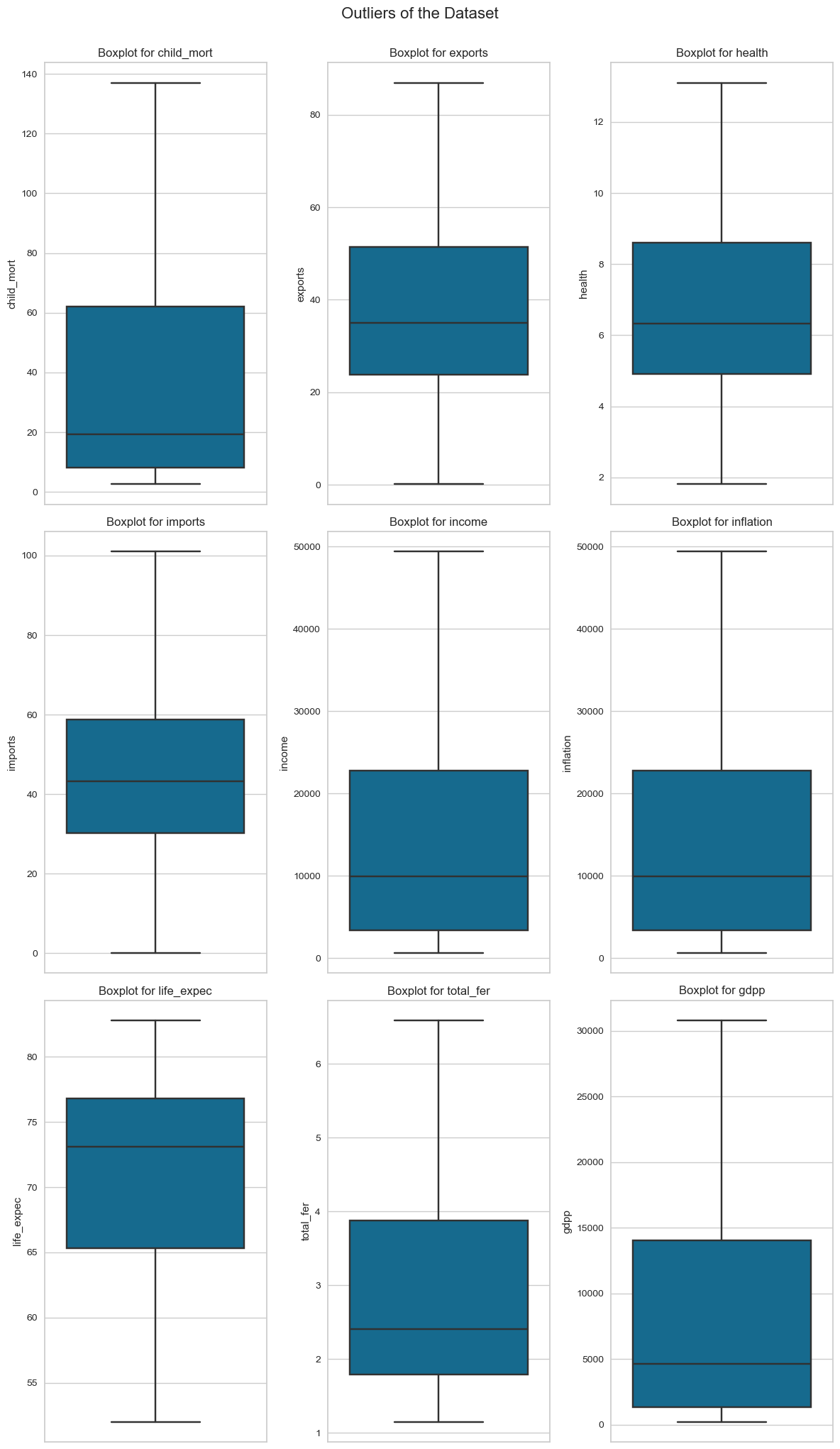
Xử lý giá trị ngoại lệ cột gdpp.

|  |
| --- |
| outlier\_min\_gdpp = iqr(df, 'gdpp')['gdpp'].min()  df[df['gdpp'] < outlier\_min\_gdpp].gdpp.max()  winsorize\_percentile\_gdpp = (stats.percentileofscore(df['gdpp'], 30800)) / 100  print(winsorize\_percentile\_total\_fer)  gdpp\_winsorize = winsorize(df.gdpp, (0, (1 - winsorize\_percentile\_gdpp)))  df['gdpp'] = gdpp\_winsorize |

Vẽ biểu đồ sau khi xử lý giá trị ngoại lệ.

|  |
| --- |
| # Kích thước của figure  plt.figure(figsize=(12, 20))  # Vẽ boxplot cho tất cả các cột  for i, column in enumerate(df.columns[1:], 1):      plt.subplot(3, 3, i)      sns.boxplot(y=df[column])      plt.title(f'Boxplot for {column}')  plt.tight\_layout()  plt.suptitle("Outliers of the Dataset", y=1.02, fontsize=16)  plt.show() |

Kết quả:



Sau khi xử lý giá trị ngoại lệ của các cột thì ở biểu đồ các cột, giá trị ngoại lệ đó đã bị mất đi.

2.4.4 Lưu tập dữ liệu sau khi đã tiền xử lý

Lưu tập dữ liệu.

|  |
| --- |
| df.to\_csv('country\_data\_processed.csv', index=False) |

Tập dữ liệu đã được lưu sau khi đã tiền xử lý: [country\_data\_processed.csv](https://github.com/minhkhanh-coder/machine-learning-1/blob/main/unsupervised%20learning/country_data_processed.csv).

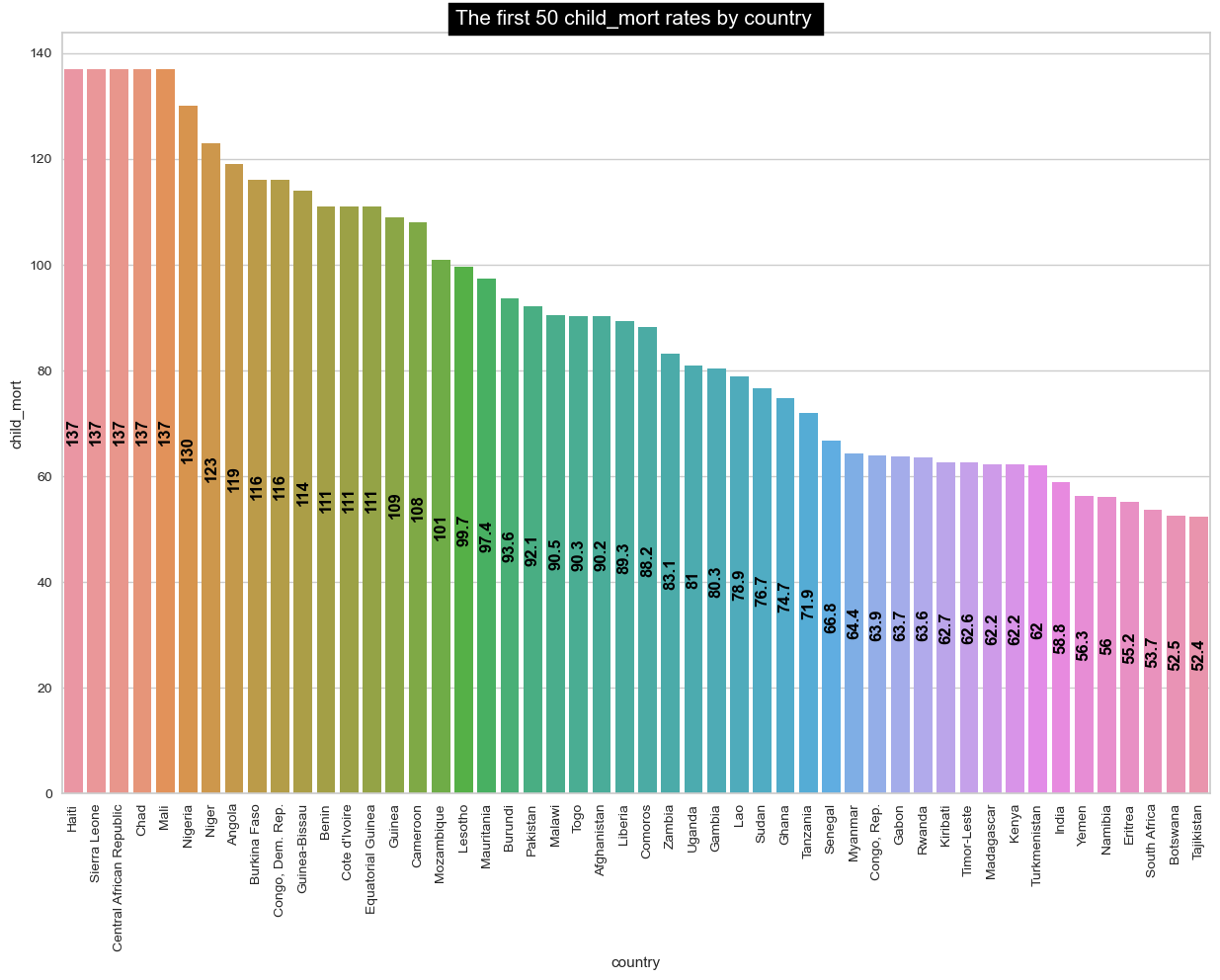
## 2.5 Mối tương quan của tập dữ liệu

### 2.5.1 Biểu đồ thể hiện 50 quốc gia có tỉ lệ child\_mort cao nhất theo thứ tự giảm dần

Vẽ biểu đồ.

|  |
| --- |
| plt.figure(figsize=(15,10)) #size  plt.title('The first 50 child\_mort rates by country ',backgroundcolor = 'Black', color='white',fontsize=15) #title  ax = sns.barplot(x = 'country',y = 'child\_mort', data = df.sort\_values(ascending = False,by = 'child\_mort').iloc[:50]);  plt.xticks(rotation=90) # ticks  for i in ax.containers:      i.datavalues      ax.bar\_label(i,fontweight='bold',color='black',label\_type='center', rotation=90)  plt.show() |

Kết quả:



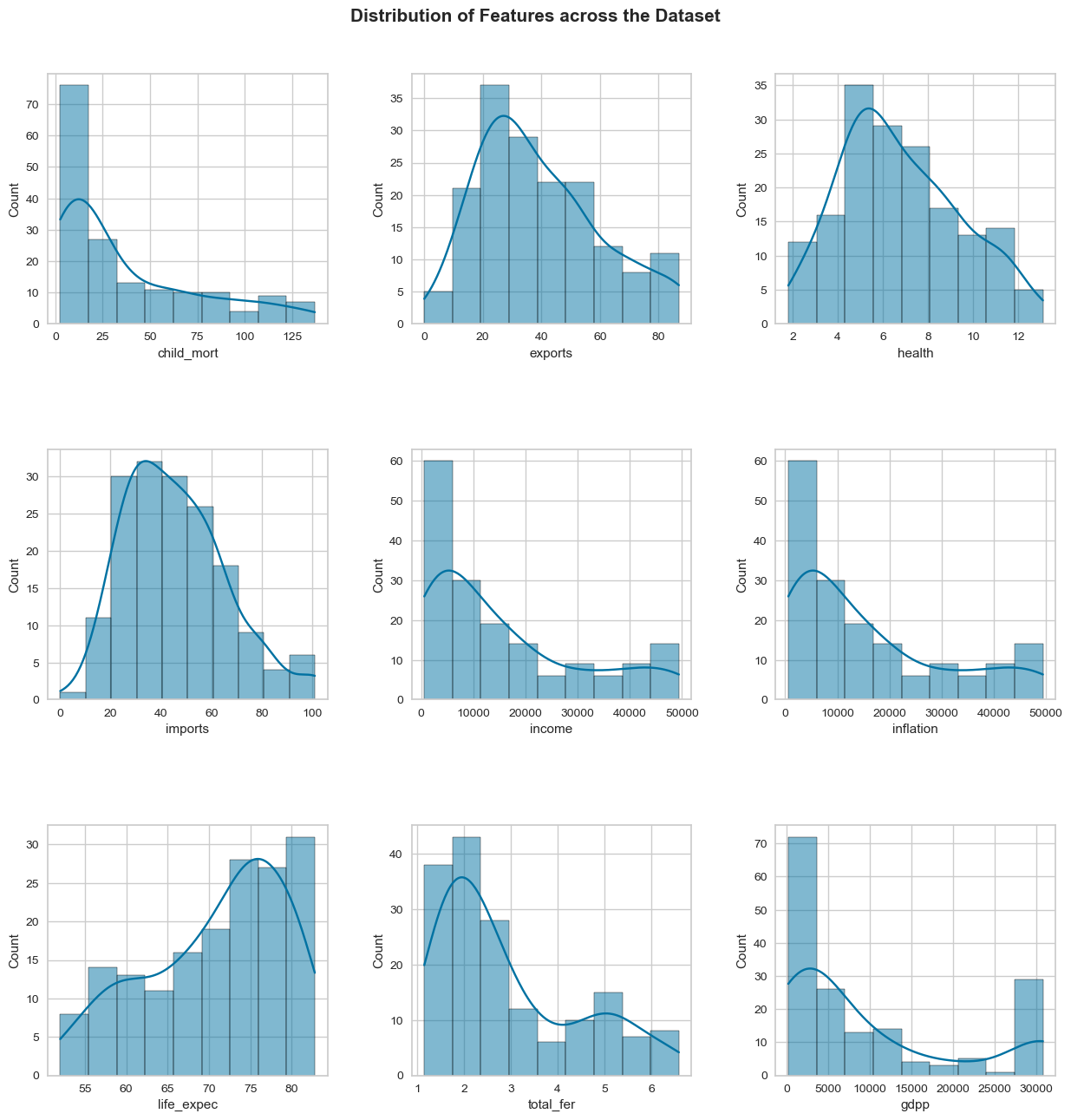
Biểu đồ trên thể hiện tỉ lệ tử vong trẻ em của 50 quốc gia theo chiều giảm dần và cho thấy sự chênh lệch lớn về tỷ lệ tử vong trẻ em giữa các quốc gia.

### 2.5.2 Biểu đồ phân phối các trường trên tập dữ liệu

Vẽ biểu đồ.

|  |
| --- |
| df2 = df.copy()  fig, ax = plt.subplots(3, 3, figsize=(15, 15), gridspec\_kw={"hspace": 0.5, "wspace": 0.3})  for i, col in enumerate(df2.drop(columns='country')):      sns.histplot(df[col], kde=True, ax=ax[i//3, i%3])    fig.suptitle('Distribution of Features across the Dataset', fontsize=15, fontweight='bold', y=0.93)  plt.show() |

Kết quả:



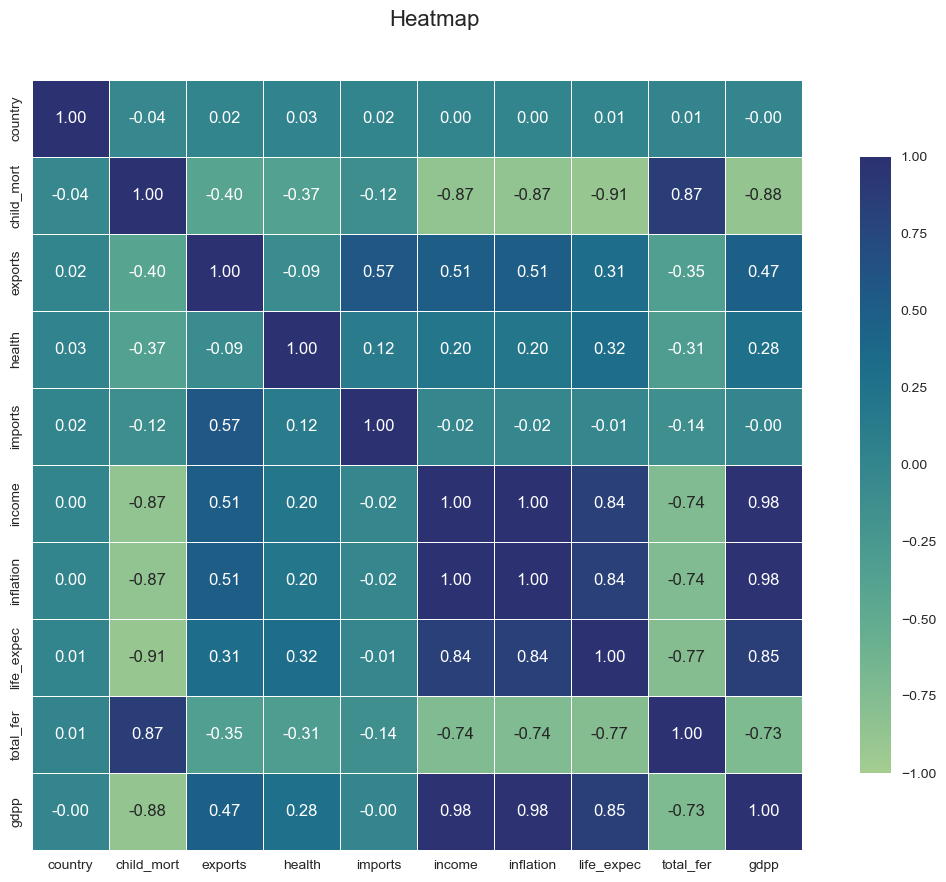
Biểu đồ thể hiện sự phân phối tần suất của các trường trong tập dữ liệu.

### 2.5.3 Biểu đồ thể hiện mức độ tương quan của toàn bộ tập dữ liệu

Vẽ biểu đồ.

|  |
| --- |
| from sklearn.preprocessing import LabelEncoder  df\_encoded = df.copy()  # Sử dụng LabelEncoder để chuyển đổi cột "country" thành dữ liệu số  label\_encoder = LabelEncoder()  df\_encoded['country'] = label\_encoder.fit\_transform(df['country'])  # Tạo heatmap cho ma trận tương quan  plt.figure(figsize=(15, 10))  sns.heatmap(df\_encoded.corr(method='spearman', min\_periods=1), annot=True, cmap="crest", linewidths=.5, fmt=".2f", vmin=-1, vmax=1, center=0, square=True, cbar\_kws={"shrink": 0.8})  plt.suptitle("Heatmap", y=0.95, fontsize=16)  # Thêm tiêu đề chung  plt.show() |

Kết quả:



Biểu đồ này thể hiện sự tương quan của các trường dữ liệu dựa trên điểm đánh giá giữa các cột với nhau. Càng gần giá trị -1 hoặc 1 thì 2 cột đó tương quang mạnh mẽ với nhau.

## 2.6 Mô hình

### 2.6.1 K-Means

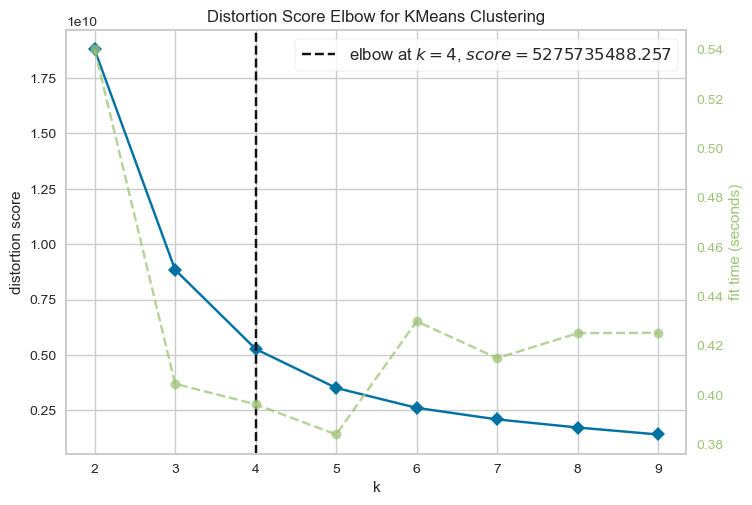
Chuẩn bị dữ liệu.

|  |
| --- |
| df\_km = df.drop('country', axis=1) |

Dùng phương pháp k-ellbow để xác định cụm k để phân loại cho thuật toán K-Means.

|  |
| --- |
| model = KMeans(random\_state=1)  visualizer = KElbowVisualizer(model, k=(2,10))  visualizer.fit(df\_km)  visualizer.show()  plt.show() |

Kết quả:



Ở biểu đồ trên, nó đã gợi ý cho ta xác định số cụm k là 4 bằng phương pháp k-ellbow để giải các bài toán Kmeans. Đường màu xanh lục là điểm đánh giá để chọn ra được k, còn đường màu xanh lục là thời gian xác định.

Phân cụm cho thuật toán.

|  |
| --- |
| number\_of\_clusters = 4  kmeans = KMeans(n\_clusters=number\_of\_clusters, init='k-means++', random\_state=42)  y\_kmeans\_pred = kmeans.fit\_predict(df\_km) |

Tính thời gian chạy và độ chính xác của thuật toán.

|  |
| --- |
| # Dữ liệu và thuật toán KMeans  kmeans = KMeans(n\_clusters=number\_of\_clusters, init='k-means++')  start\_time = time.time()  # Thực hiện clustering và lấy nhãn  labels = kmeans.fit\_predict(df\_km)  end\_time = time.time()  # Đánh giá chất lượng mô hình bằng Silhouette Score  silhouette\_avg = silhouette\_score(df\_km, labels)  # In thông tin  print(f'Thời gian chạy: {end\_time - start\_time:.4f} giây')  print(f'Độ chính xác: {silhouette\_avg:.4f}') |

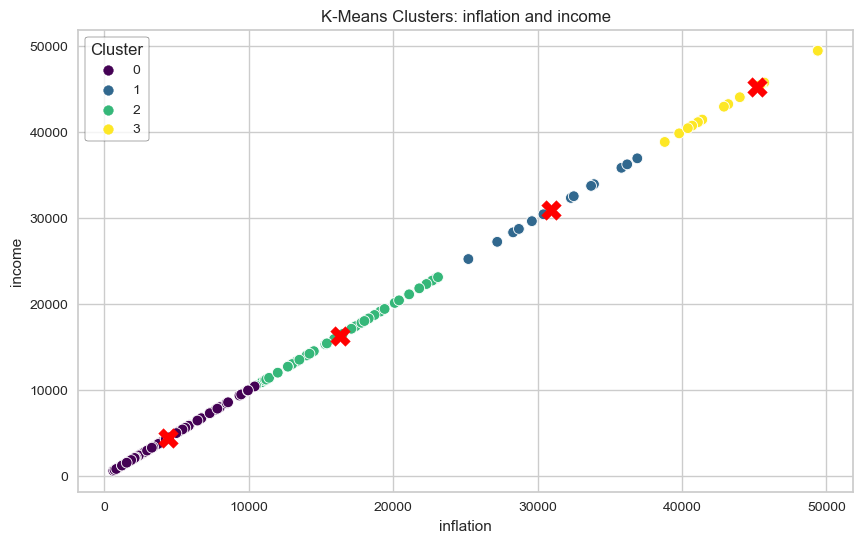
Kết quả:



Vẽ biểu đồ trực quan hóa cho thuật toán (biểu đồ 1: inflation – income, 1 điểm).

|  |
| --- |
| df\_km = df.drop(['country'], axis =1)  selected\_components = df\_km.select\_dtypes(include=['int64', 'float64'])  kmeans = KMeans(n\_clusters=4, random\_state=42)  clusters = kmeans.fit\_predict(selected\_components)  # Thêm nhãn cụm vào dataset  df\_km['Cluster'] = clusters  plt.figure(figsize=(10, 6))  sns.scatterplot(x=df\_km["inflation"], y=df\_km["income"], hue=clusters, palette='viridis', s=60)  for cluster\_label in df\_km['Cluster'].unique():      cluster\_center = df\_km[df\_km['Cluster'] == cluster\_label][['inflation', 'income']].mean()      plt.scatter(cluster\_center['inflation'], cluster\_center['income'], s=200, c='red', marker='X')  plt.title('K-Means Clusters: inflation and income')  plt.xlabel('inflation')  plt.ylabel('income')  plt.legend(title='Cluster', frameon=True, edgecolor='black')  plt.show() |

Kết quả:



Hình ảnh mô tả mối quan hệ giữa tốc độ tăng trưởng GDP và thu nhập của một nhóm người. Đường biểu đồ thể hiện rằng tốc độ tăng trưởng GDP tăng lên thì thu nhập cũng tăng lên, nhưng mức độ tăng lên của thu nhập không đồng đều giữa các nhóm người.

Theo biểu đồ, có thể chia nhóm người thành 4 nhóm dựa trên mức độ tăng trưởng GDP và thu nhập. Nhóm 0 có tốc độ tăng trưởng GDP và thu nhập thấp nhất. Nhóm 1 có tốc độ tăng trưởng GDP và thu nhập trung bình. Nhóm 2 có tốc độ tăng trưởng GDP và thu nhập khá cao. Nhóm 3 có tốc độ tăng trưởng GDP và thu nhập cao nhất.

Vẽ biểu đồ trực quan hóa cho thuật toán (biểu đồ 2: gdpp – income, 0.98 điểm).

|  |
| --- |
| plt.figure(figsize=(10, 6))  sns.scatterplot(x=df\_km["gdpp"], y=df\_km["income"], hue=clusters, palette='viridis', s=60)  for cluster\_label in df\_km['Cluster'].unique():      cluster\_center = df\_km[df\_km['Cluster'] == cluster\_label][['gdpp', 'income']].mean()      plt.scatter(cluster\_center['gdpp'], cluster\_center['income'], s=200, c='red', marker='X')  plt.title('K-Means Clusters: gdpp and income')  plt.xlabel('gdpp')  plt.ylabel('income')  plt.legend(title='Cluster', frameon=True, edgecolor='black')  plt.show() |

Kết quả:



Hình ảnh mô tả mối quan hệ giữa GDP bình quân đầu người và thu nhập của một nhóm người. Đường biểu đồ thể hiện GDP bình quân đầu người tăng lên thì thu nhập cũng tăng lên, nhưng mức độ tăng lên của thu nhập không đồng đều giữa các nhóm người.

Theo biểu đồ, có thể chia nhóm người thành 4 nhóm dựa trên GDP bình quân đầu người và thu nhập. Nhóm 0 có GDP bình quân đầu người và thu nhập thấp nhất. Nhóm 1 có GDP bình quân đầu người và thu nhập trung bình. Nhóm 2 có GDP bình quân đầu người và thu nhập khá cao. Nhóm 3 có GDP bình quân đầu người và thu nhập cao nhất.

### 2.6.2 Hierarchical Clustering

Giới thiệu về thuật toán Hierarrchical Clustering:

* Thuật toán phân cụm K-means cho thấy cần phải cấu hình trước số lượng cụm cần phân chia. Ngược lại, phương pháp phân cụm phân cấp (Hierachical Clustering) không yêu cầu khai báo trước số lượng cụm. Thay vào đó, thuật toán chỉ yêu cầu xác định trước thước đo về sự khác biệt giữa các cụm (không giao nhau), dựa trên sự khác biệt từng cặp giữa các quan sát trong hai cụm.
* Theo phương pháp này, chúng tạo ra những biểu diễn phân cấp trong đó các cụm ở mỗi cấp của hệ thống phân cấp được tạo bằng cách hợp nhất các cụm ở cấp độ thấp hơn bên dưới. Ở cấp thấp nhất, mỗi cụm chứa một quan sát. Ở cấp cao nhất, chỉ có một cụm chứa tất cả dữ liệu. Các cấp của biểu diễn phân cụm được thể hiện trong đồ thị dendrogram bên dưới.

Chuẩn hóa StanderScaler.

|  |
| --- |
| #scaling data  scaling=StandardScaler()  scaled=scaling.fit\_transform(df\_km)  scaled\_df=pd.DataFrame(scaled,columns=df\_km.columns) |

Vẽ biểu đồ trực quan thuật toán Hierarchical Clustering.

|  |
| --- |
| plt.figure(figsize=(50, 12))  dend=hcluster.dendrogram(hcluster.linkage(scaled\_df,method='ward'))  plt.title("Hierarchical Clustering", fontsize=70) |

Kết quả:

A graph with lines and text

Description automatically generated with medium confidence

Từ biểu đồ dendogram ở trên, chúng ta có thể lấy số cụm tối thiểu là 2 và số cụm tối đa là 5. Như chúng ta có thể thấy sơ đồ dendogram 4 ở bên phải số cụm, vì vậy chúng ta sẽ lấy 3 số cụm

Tính thời gian chạy và độ chính xác của thuật toán.

|  |
| --- |
| # Timing the clustering process  start\_time = time.time()  # Hierarchical clustering  hcluster = AgglomerativeClustering(n\_clusters=4, affinity='euclidean', linkage='ward')  hcluster\_label = hcluster.fit\_predict(scaled\_df)  # Calculate silhouette score  silhouette\_avg = silhouette\_score(scaled\_df, hcluster\_label)  # Timing information  end\_time = time.time()  # Adding cluster labels to the original dataframe  hcluster\_df = pd.DataFrame(df\_km)  hcluster\_df['hcluster'] = hcluster\_label  # Print information  print(f'Thời gian chạy: {end\_time - start\_time:.4f} giây')  print(f'Silhouette Score: {silhouette\_avg:.4f}') |

Kết quả:



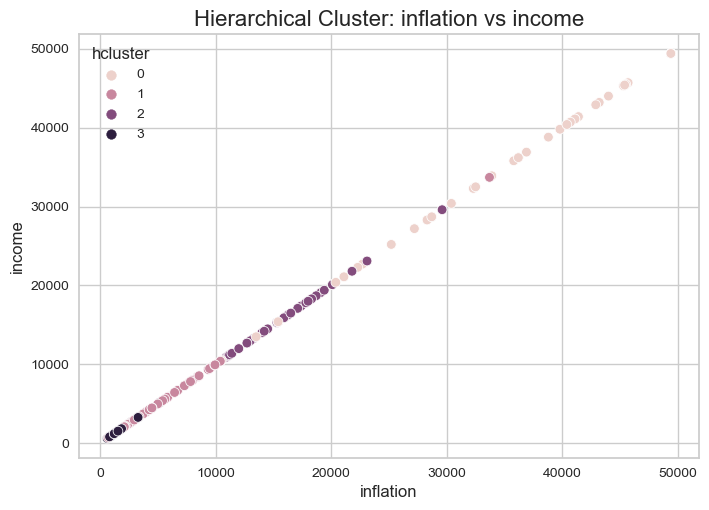
Phân cụm các label.

|  |
| --- |
| hcluster\_df = pd.DataFrame(df\_km)  # thêm nhãn hcluster trong hcluster\_df  hcluster\_df['hcluster'] = hcluster\_label |

Vẽ biểu đồ trực quan hóa của cột inflation và income (biểu đồ 1).

|  |
| --- |
| sns.scatterplot(x=hcluster\_df['inflation'], y=hcluster\_df['income'], hue='hcluster', data=hcluster\_df)  plt.title("Hierarchical Cluster: inflation vs income", fontsize=16)  plt.xlabel("inflation", fontsize=12)  plt.ylabel("income", fontsize=12)  plt.show() |

Kết quả:



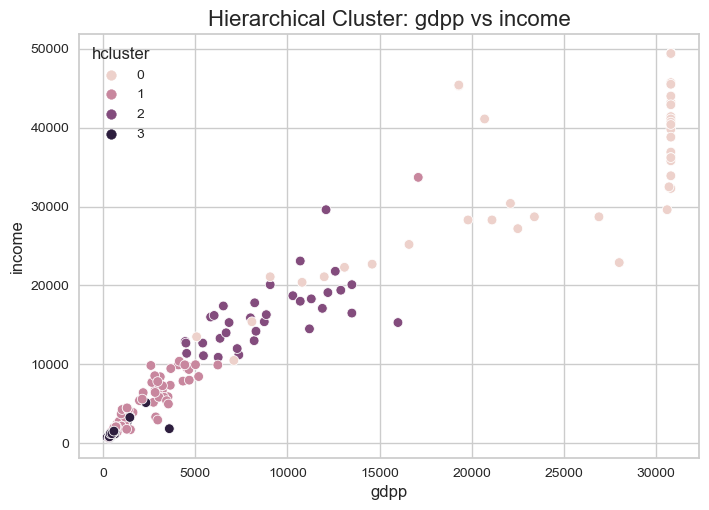
Biểu đồ trên một biểu đồ phân cụm tốc độ tăng trưởng GDP và thu nhập của một nhóm người. Được phân thành 4 cụm.

Theo biểu đồ, có thể chia nhóm người thành 4 nhóm dựa trên mức độ tăng trưởng GDP và thu nhập. Nhóm 0 có tốc độ tăng trưởng GDP và thu nhập cao nhất. Nhóm 1 có tốc độ tăng trưởng GDP và thu nhập khá cao. Nhóm 2 có tốc độ tăng trưởng GDP và thu nhập trung bình. Nhóm 3 có tốc độ tăng trưởng GDP và thu nhập thấp nhất.

Vẽ biểu đồ trực quan hóa của cột gdpp và income (biểu đồ 2).

|  |
| --- |
| sns.scatterplot(x=hcluster\_df['gdpp'],y=hcluster\_df['income'],hue='hcluster',data=hcluster\_df)  plt.title("Hierarchical Cluster: gdpp vs income", fontsize=16)  plt.xlabel("gdpp", fontsize=12)  plt.ylabel("income", fontsize=12)  plt.show() |

Kết quả:



Hình ảnh mô tả mối quan hệ giữa GDP bình quân đầu người và thu nhập của một nhóm người. Đường biểu đồ thể hiện GDP bình quân đầu người tăng lên thì thu nhập cũng tăng lên, nhưng mức độ tăng lên của thu nhập không đồng đều giữa các nhóm người.

Theo biểu đồ, có thể chia nhóm người thành 4 nhóm dựa trên GDP bình quân đầu người và thu nhập. Nhóm 0 có GDP bình quân đầu người và thu nhập cao nhất. Nhóm 1 có GDP bình quân đầu người và thu nhập khá cao. Nhóm 2 có GDP bình quân đầu người và thu nhập trung bình. Nhóm 3 có GDP bình quân đầu người và thu nhập thấp nhất.

2.6.3 DBScan

Tính thời gian và độ chính xác của thuật toán.

|  |
| --- |
| # Chuẩn hóa dữ liệu  scaler = StandardScaler()  df\_km\_scaled = scaler.fit\_transform(df\_km)  # Sử dụng Nearest Neighbors để tính toán khoảng cách  neighbors = NearestNeighbors(n\_neighbors=2)  # Bắt đầu đo thời gian  start\_time = time.time()  # Fit dữ liệu  neighbors\_fit = neighbors.fit(df\_km\_scaled)  # Tính toán khoảng cách và chỉ số của hàng gần nhất  distances, indices = neighbors\_fit.kneighbors(df\_km\_scaled)  # Kết thúc đo thời gian  end\_time = time.time()  # In thời gian chạy  print("Thời gian chạy:", end\_time - start\_time, "giây")  # Sử dụng chỉ số hàng gần nhất để thực hiện DBSCAN  dbscan = DBSCAN(eps=2.0, min\_samples=10)  dbscan\_labels = dbscan.fit\_predict(df\_km\_scaled)  # Tính toán điểm Silhouette  silhouette = silhouette\_score(df\_km\_scaled, dbscan\_labels)  # In điểm Silhouette  print(f"Điểm Silhouette: {silhouette:.4f}") |

Kết quả:

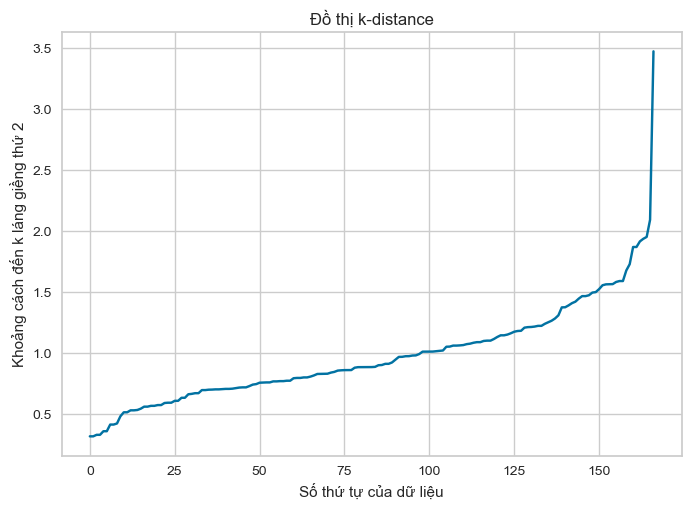
A number with numbers and circles

Description automatically generated with medium confidence

Vẽ đồ thị k-distance.

|  |
| --- |
| # Sắp xếp và vẽ đồ thị để xem độ dốc của đồ thị  distances = np.sort(distances, axis=0)  distances = distances[:, 1]  plt.plot(distances)  plt.xlabel('Số thứ tự của dữ liệu')  plt.ylabel('Khoảng cách đến k láng giềng thứ 2')  plt.title('Đồ thị k-distance')  plt.show() |

Kết quả:

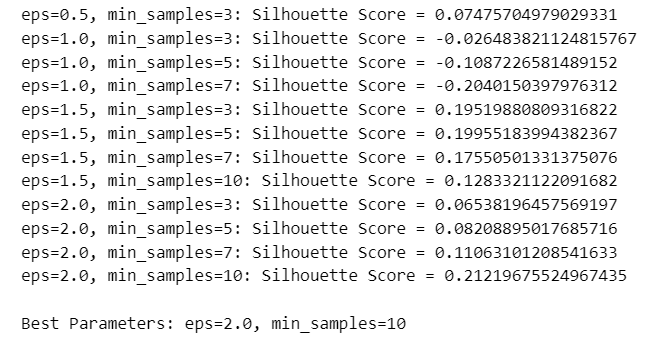


Đồ thị k-distance là một đồ thị biểu diễn khoảng cách giữa một điểm dữ liệu và k láng giềng gần nhất của nó. Trong đồ thị này, mỗi điểm dữ liệu được biểu diễn bằng một chấm trên đồ thị, và k láng giềng gần nhất của điểm dữ liệu đó được biểu diễn bằng k đường thẳng nối từ điểm dữ liệu đó đến k láng giềng.

In ra các thông số tốt nhất để vẽ biểu đồ trực quan cho thuật toán.

|  |
| --- |
| # Import thư viện cần thiết  from sklearn.cluster import DBSCAN  from sklearn.metrics import silhouette\_score  import matplotlib.pyplot as plt  # Thiết lập các giá trị eps và min\_samples cần thử nghiệm  eps\_values = [0.5, 1.0, 1.5, 2.0]  min\_samples\_values = [3, 5, 7, 10]  # Lưu trữ kết quả tốt nhất  best\_eps = None  best\_min\_samples = None  best\_silhouette\_score = -1  # Thử nghiệm với từng giá trị eps và min\_samples  for eps in eps\_values:      for min\_samples in min\_samples\_values:          # Áp dụng thuật toán DBSCAN          dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)          labels = dbscan.fit\_predict(scaled\_df)          # Kiểm tra số lượng cụm để tránh lỗi          unique\_labels = np.unique(labels)          if len(unique\_labels) > 1:              # Tính silhouette score              silhouette\_avg = silhouette\_score(scaled\_df, labels)              # In kết quả và kiểm tra xem có phải là tốt nhất không              print(f"eps={eps}, min\_samples={min\_samples}: Silhouette Score = {silhouette\_avg}")              if silhouette\_avg > best\_silhouette\_score:                  best\_silhouette\_score = silhouette\_avg                  best\_eps = eps                  best\_min\_samples = min\_samples  # In ra thông tin về các tham số tốt nhất  print(f"\nBest Parameters: eps={best\_eps}, min\_samples={best\_min\_samples}") |

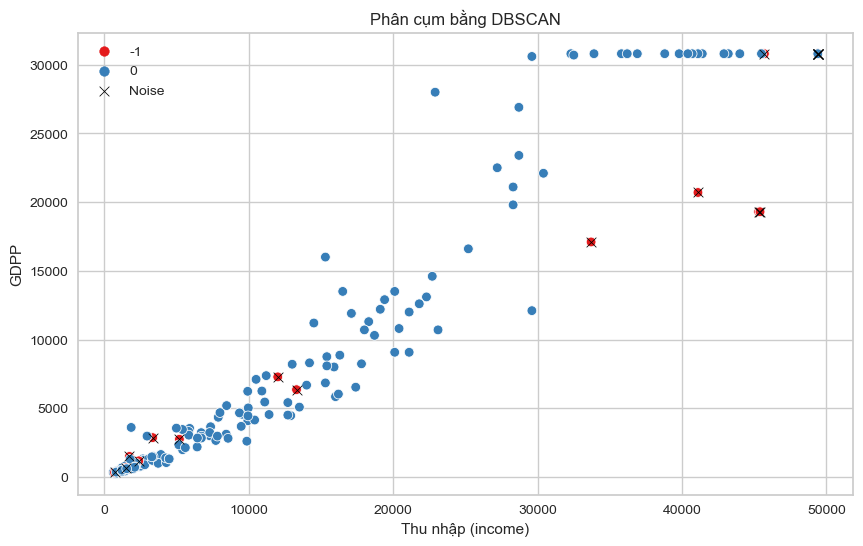
Kết quả:



Vẽ biểu đồ trực quan cho thuật toán.

|  |
| --- |
| # Áp dụng DBSCAN với các tham số tốt nhất  best\_dbscan = DBSCAN(eps=best\_eps, min\_samples=best\_min\_samples)  best\_labels = best\_dbscan.fit\_predict(scaled\_df)  noise\_points = df\_km[labels == -1]  plt.figure(figsize=(10, 6))  sns.scatterplot(x=df\_km['income'], y=df\_km['gdpp'], hue=labels, palette='Set1')  sns.scatterplot(x=noise\_points['income'], y=noise\_points['gdpp'], color='black', marker='x', label='Noise')  plt.xlabel('Thu nhập (income)')  plt.ylabel('GDPP')  plt.title('Phân cụm bằng DBSCAN')  plt.legend()  plt.show() |

Kết quả:



Đồ thị trên biểu thị kết quả phân cụm dữ liệu theo thuật toán DBSCAN. Dữ liệu được biểu diễn trên hai chiều: thu nhập và GDP.

Trong thuật toán DBSCAN, dữ liệu được chia thành hai loại: điểm dữ liệu core và điểm dữ liệu noise.

Điểm dữ liệu core là điểm dữ liệu có ít nhất k điểm dữ liệu lân cận trong phạm vi epsilon. Điểm dữ liệu noise là điểm dữ liệu không phải là điểm dữ liệu core.

Trong đồ thị trên, các điểm dữ liệu được biểu thị bằng màu xanh lam, và các điểm dữ liệu noise được biểu thị bằng màu đỏ.

Có thể thấy rằng, dữ liệu được phân thành ba cụm:

* Cụm 1: Các điểm dữ liệu có thu nhập cao và GDP cao.
* Cụm 2: Các điểm dữ liệu có thu nhập thấp và GDP thấp.
* Cụm 3: Các điểm dữ liệu có thu nhập và GDP trung bình.

Các điểm dữ liệu noise được biểu thị bằng màu đỏ. Các điểm dữ liệu này không thuộc bất kỳ cụm nào.

## 1.7 Phân tích, đánh giá và kết luận

### 1.7.1 Phân tích, đánh giá các thuật toán

|  |  |  |  |
| --- | --- | --- | --- |
|  | Kmeans | Hierarchical Clustering | DBScan |
| Độ chính xác | 0.6044 | 0.2007 | 0.2677 |
| Thời gian (s) | 0.2576 | 0.007 | 0.002 |

1. **Kmeans:** 
   1. Phân tích:
      1. Độ chính xác tương đối nhưng không cao.
      2. Thời gian chạy ổn định nhưng không phải là nhanh nhất.
   2. Đánh giá:
      1. K-Means là một phương pháp tốt cho việc phân cụm, nhưng có thể cần điều chỉnh số lượng cụm để cải thiện độ chính xác.
2. **Hierarchical Clustering:**
   1. Phân tích:
      1. Độ chính xác thấp, có thể không phù hợp cho dữ liệu có cấu trúc phức tạp.
      2. Thời gian chạy nhanh.
   2. Đánh giá:
      1. Hierarchical Clustering không đạt được độ chính xác cao, có thể không phù hợp cho các tập dữ liệu phức tạp.
3. **DBSCAN:**
   1. Phân tích:
      1. Độ chính xác tương đối cao, đặc biệt là so với Hierarchical Clustering.
      2. Thời gian chạy rất nhanh.
   2. Đánh giá:
      1. DBSCAN có thể là lựa chọn tốt với dữ liệu có cấu trúc phức tạp, với độ chính xác khá và thời gian chạy ngắn.

### 1.7.2 Kết luận

* + - Nếu ưu tiên độ chính xác:
      * + Chọn DBSCAN, với độ chính xác tương đối cao.
    - Nếu ưu tiên thời gian chạy:
      * + Chọn DBSCAN, với thời gian chạy ngắn nhất.
    - Nếu cần cả hai:
      * + DBSCAN vẫn là lựa chọn tốt nhất với sự cân bằc giữa độ chính xác và thời gian chạy.

=> Đối với tập dữ liệu này nên chọn thuật toán DBSCAN vì:

* DBSCAN có thời gian chạy rất nhanh, làm cho nó là lựa chọn hữu ích cho việc xử lý dữ liệu lớn.
* DBSCAN thường hiệu quả với dữ liệu có các cụm có hình dạng và kích thước khác nhau.
* DBSCAN là một thuật toán linh hoạt và thời gian chạy nhanh, đặc biệt là khi sử dụng cùng với Nearest Neighbors để tính khoảng cách

DBSCAN là một lựa chọn mạnh mẽ đối với dữ liệu có cấu trúc phức tạp và không đồng đều.

# TÀI LIỆU THAM KHẢO

# KẾT QUẢ KIỂM TRA ĐẠO VĂN

**ĐẠI HỌC HUẾ**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

🙢

**PHIẾU ĐÁNH GIÁ ĐỒ ÁN/TIỂU LUẬN/BÀI TẬP LỚN**

**Học kỳ I, năm học 2021 – 2022**

|  |  |
| --- | --- |
| **Cán bộ chấm thi 1** | **Cán bộ chấm thi 2** |
| **Nhận xét:**  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  Điểm đánh giá của CBCT1:  Bằng số: ..............................................  Bằng chữ: ............................................ | **Nhận xét:**  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  ..............................................................  Điểm đánh giá của CBCT2:  Bằng số: ..............................................  Bằng chữ: ............................................ |

Điểm kết luận: ..................................................................................................................

Bằng số: ............................................................................................................................

Bằng chữ: ..........................................................................................................................

*Thừa Thiên Huế, ngày 15 tháng 01 năm 2022*

**Cán bộ chấm thi 1**

*(Ký và ghi rõ họ tên)*

**Cán bộ chấm thi 2**

*(Ký và ghi rõ họ tên)*