

**ĐẠI HỌC HUẾ**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

**🙢**

**BÁO CÁO ĐỒ ÁN**

**Học kỳ I, năm học 2023 - 2024**

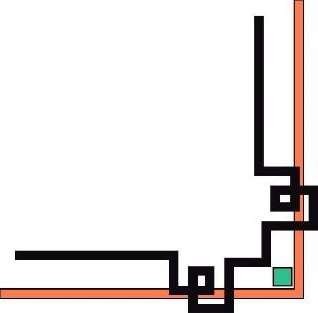
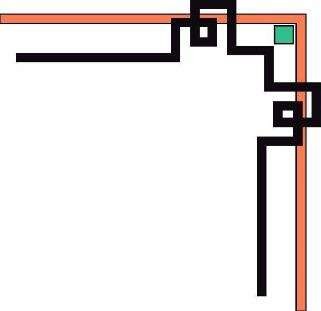
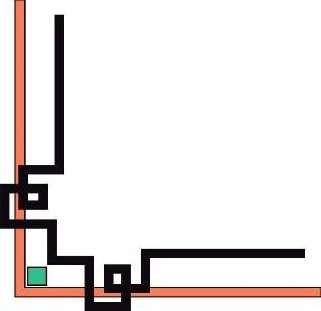
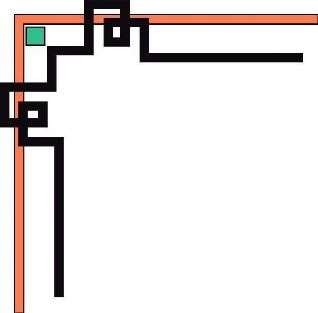
**Học phần:**

**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 1: 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 2: 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 3: 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 4: 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 5: 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 6: Kết quả in ra giá trị bị thiếu

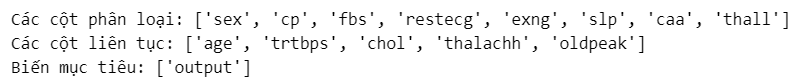
* Kết quả cho ta thấy được là có 2 giá trị bị thiếu trong cột thall và tất cả các cột còn lại không có giá trị nào bị thiếu.

### 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 7: 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 8: 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 9: 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 10: 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 11: 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

Kiểm tra giá trị trùng lặp.

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

Kết quả:

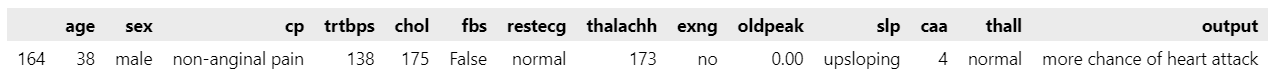
A close up of a word

Description automatically generated

In ra dòng chứa giá trị trùng lặp.

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

Kết quả:



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

### 1.4.2 Xử lý giá trị bị thiếu

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

### 1.4.3 Chuyển đổi dữ liệu

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) |

Kết quả:

A screenshot of a table

Description automatically generated

### 1.4.4 Xử lý các 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 thalachh.

* In ra các dòng chứa giá trị ngoại lệ.

|  |
| --- |
| # Giá trị trung bình  iqr(df, 'thalachh')['thalachh'] |

* Kết quả:

A close-up of a word

Description automatically generated

* 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:



* Xử lý giá trị ngoại lệ.

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

Xử lý cột trtbps.

|  |
| --- |
| # In ra các giá trị ngoại lệ của cột  iqr(df,'trtbps')['trtbps']  # In ra giá trị ngoại lệ nhỏ nhất của cột  iqr(df, 'trtbps')['trtbps'].min()  # Tìm giá trị lớn nhất của cột (<172)  df[df['trtbps'] < 172].trtbps.max()  # 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 |

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 |

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 cc

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

# TÀI LIỆU THAM KHẢO

# KẾT QUẢ KIỂM TRA ĐẠO VĂN

Text

Description automatically generated with low confidence

**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)*