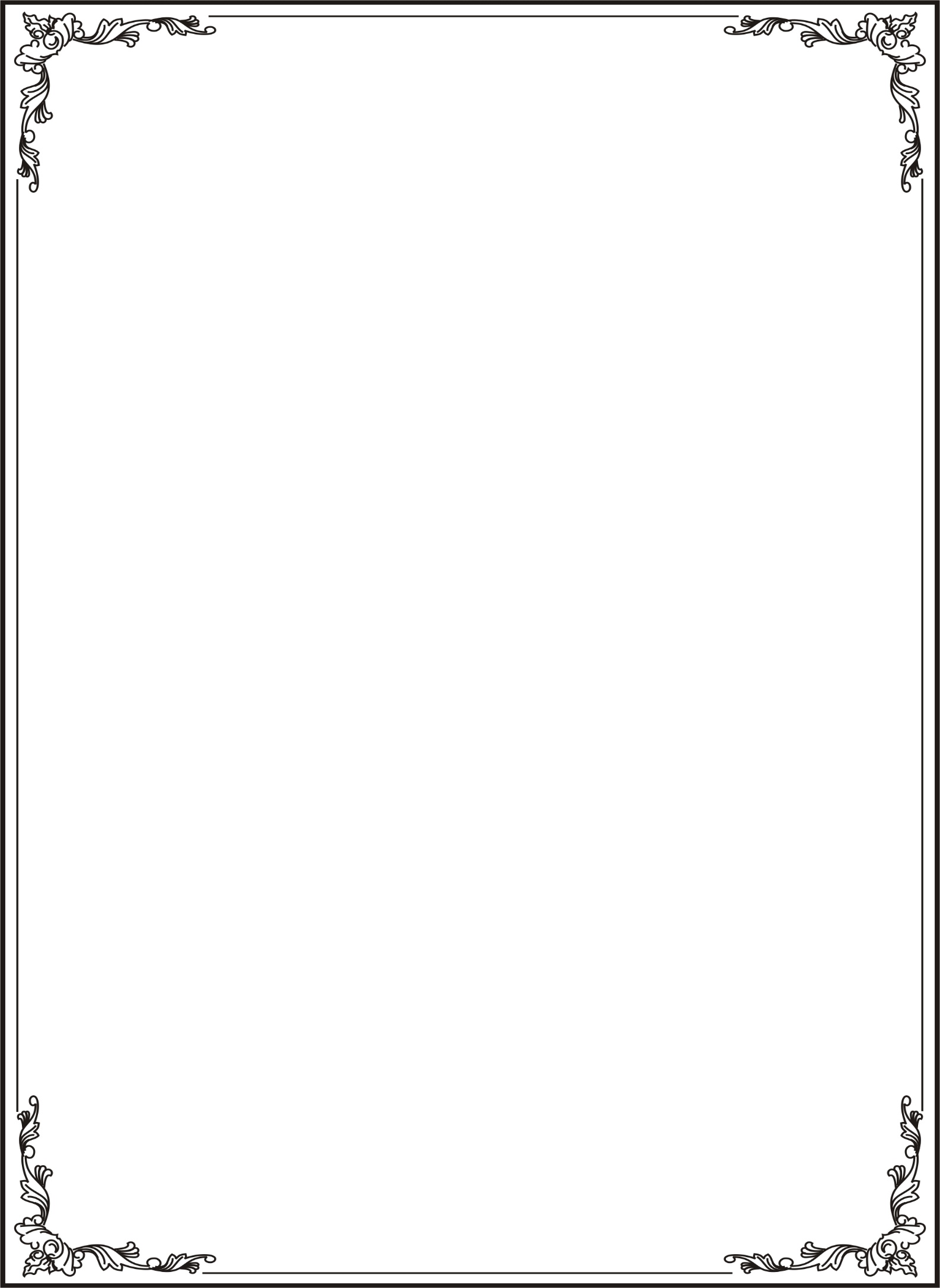
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**Đề tài: Khai phá dữ liệu và**

**dự đoán nguy cơ mắc bệnh tim ở các bệnh nhân**

**SVTH: Đào Quang Duy - 21110398**

**Nguyễn Đình Liệu - 21110528**

**Nguyễn Anh Dũng - 21110873**

**Trầm Dũ Phong - 21110913**

**GVHD: GV, Ths.Trần Trọng Bình**

**Tp. Hồ Chí Minh, tháng 05 năm 2024**

Mục lục

[**1.** **Giới thiệu đề tài** 1](#_Toc167398052)

[**2.** **Dữ liệu (dataset)** 1](#_Toc167398053)

[**3.** **Câu hỏi phân tích** 4](#_Toc167398054)

[**4.** **EDA và tiền xử lý.** 4](#_Toc167398055)

[**4.1.** **Kiểm tra phân bố.** 4](#_Toc167398056)

[**4.2.** **Xử lý ngoại biên.** 7](#_Toc167398057)

[**4.3.** **Chuẩn hóa dữ liệu.** 7](#_Toc167398058)

[**4.4.** **Đánh giá tương quan.** 8](#_Toc167398059)

[**4.5.** **Xử lý mất cân bằng dữ liệu.** 9](#_Toc167398060)

[**5.** **Mô hình hóa dữ liệu (data modeling)** 10](#_Toc167398061)

[**5.1.** **Random Forest** 10](#_Toc167398062)

[**5.2.** **Logistic Regression** 10](#_Toc167398063)

[**5.3.** **Gradient Descent** 10](#_Toc167398064)

[**5.4.** **SVC** 10](#_Toc167398065)

[**5.5.** **Kết quả của các mô hình** 10](#_Toc167398066)

[**6.** **Bài toán phân cụm và khai phá luật kết hợp** 15](#_Toc167398067)

[**TÀI LIỆU THAM KHẢO**](#_Toc167398068)

|  |  |  |
| --- | --- | --- |
| **Số điểm giáo viên đã chấm** | **Nội dung chỉnh sửa** | **Số điểm lần 2** |
| 8.3 | - Thêm chỉnh sửa lại file báo cáo thêm các trường hợp mà có thể rút trích ra được từ việc phân cụm (trang 19)  - Thêm phần triển khai mô hình phân loại bệnh tim bằng ANN |  |

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Số điểm lần 1** | **Số điểm lần 2** |
| **Đào Quang Duy** | **8.1** |  |
| **Nguyễn Đình Liệu** | **8.0** |  |
| **Nguyễn Anh Dũng** | **8.0** |  |
| **Trầm Dũ Phong** | **8.0** |  |

**Số điểm được tính như bên dưới: 8.3 \* TB/10**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tiêu chí** | **Đào Quang Duy** | **Trầm Dũ Phong** | **Nguyễn Đình Liệu** | **Nguyễn Anh Dũng** |
| 1. Tham gia các cuộc họp thường xuyên, đúng giờ | 10 | 10 | 10 | 10 |
| 2. Hoàn thành công việc được giao đúng hạn | 10 | 10 | 10 | 10 |
| 3. Chuẩn bị công việc một cách chu đáo, chất lượng | 10 | 10 | 9 | 10 |
| 4. Có thái độ hợp tác, phối hợp với các thành viên trong nhóm | 10 | 9 | 10 | 9 |
| 5. Khả năng đóng góp ý kiến của thành viên đối với công việc của nhóm | 9 | 9 | 9 | 9 |
| Trung bình | 9.8 | 9.6 | 9.6 | 9.6 |
| Quy đổi điểm | 8.134 | 7.968 | 7.968 | 7.968 |

1. **Giới thiệu đề tài**

Theo CDC, bệnh tim là nguyên nhân hàng đầu gây tử vong cho người dân thuộc hầu hết các chủng tộc ở Hoa Kỳ (người Mỹ gốc Phi, người Mỹ da đỏ, thổ dân Alaska và người da trắng). Khoảng một nửa số người Mỹ (47%) có ít nhất 1 trong 3 yếu tố nguy cơ chính gây bệnh tim: huyết áp cao, cholesterol cao và hút thuốc. Các chỉ số quan trọng khác bao gồm tình trạng bệnh tiểu đường, béo phì (chỉ số BMI cao), không hoạt động thể chất đầy đủ hoặc uống quá nhiều rượu. Việc xác định và ngăn ngừa các yếu tố có tác động lớn nhất đến bệnh tim là rất quan trọng trong chăm sóc sức khỏe. Đổi lại, sự phát triển trong điện toán cho phép ứng dụng các phương pháp học máy để phát hiện các “mẫu” trong dữ liệu có thể dự đoán tình trạng của bệnh nhân.

Ngoài EDA cổ điển, có thể được sử dụng để áp dụng một số phương pháp học máy, đặc biệt là các mô hình phân loại (hồi quy logistic, SVM, rừng ngẫu nhiên, v.v.). Trong lĩnh vực học máy và khai phá dữ liệu, việc hiểu và áp dụng các kỹ thuật phân loại và phân cụm là một kỹ năng quan trọng. Đề tài "Preprocessing & PCA" và các phương pháp phân loại và phân cụm bao gồm Random Forest, Logistic Regression, SVC, K-Means, DBScan và Hierarchical Clustering cung cấp một cái nhìn toàn diện về cách xử lý và phân tích dữ liệu. Với bộ dữ liệu khảo sát CDC hàng năm năm 2022 của hơn 60 nghìn người lớn liên quan đến tình trạng sức khỏe của họ chúng ta sẽ rút kết ra được các thông tin bổ ích cho các công việc liên quan đến bệnh tim.

1. **Dữ liệu (dataset)**

**Nguồn dữ liệu:** <https://www.kaggle.com/code/yogitasingh05/heartdiseaseprediction-2020/>

Bộ dữ liệu HeartDiseasePrediction chứa thông tin về bệnh nhân tim mạch, bao gồm các thuộc tính liên quan đến tình trạng sức khỏe chung, hoạt động thể chất, giấc ngủ, tiền sử bệnh lý và các yếu tố nguy cơ như hút thuốc và uống rượu. Bằng cách phân tích dữ liệu này, chúng ta có thể hiểu rõ hơn về các yếu tố gây bệnh tim mạch và phát triển các mô hình dự đoán để hỗ trợ phát hiện sớm và điều trị hiệu quả.

Bộ dữ liệu gồm 40 thuộc tính và hơn 400.000 dòng dữ liệu, các thuộc tính bao gồm:

1. State: Tiểu bang nơi người tham gia sống.

2. Sex: Giới tính của người tham gia (nam hoặc nữ).

3. GeneralHealth: Đánh giá sức khỏe tổng quát của người tham gia (rất tốt, tốt, trung bình, kém, rất kém).

4. PhysicalHealthDays: Số ngày trong tháng qua mà sức khỏe thể chất không tốt.

5. MentalHealthDays: Số ngày trong tháng qua mà sức khỏe tinh thần không tốt.

6. LastCheckupTime: Thời gian kể từ lần khám sức khỏe gần nhất.

7. PhysicalActivities: Người tham gia có thực hiện các hoạt động thể chất thường xuyên không.

8. SleepHours: Số giờ ngủ trung bình mỗi đêm.

9. RemovedTeeth: Số răng đã bị nhổ (do bệnh hoặc lý do khác).

10. HadAngina: Người tham gia có từng bị đau thắt ngực (angina) không.

11. HadStroke: Người tham gia có từng bị đột quỵ không.

12. HadAsthma: Người tham gia có từng bị hen suyễn không.

13. HadSkinCancer: Người tham gia có từng bị ung thư da không.

14. HadCOPD: Người tham gia có từng bị bệnh phổi tắc nghẽn mạn tính (COPD) không.

15. HadDepressiveDisorder: Người tham gia có từng bị rối loạn trầm cảm không.

16. HadKidneyDisease: Người tham gia có từng bị bệnh thận không.

17. HadArthritis: Người tham gia có từng bị viêm khớp không.

18. HadDiabetes: Người tham gia có từng bị bệnh tiểu đường không.

19. DeafOrHardOfHearing: Người tham gia có bị điếc hoặc khó nghe không.

20. BlindOrVisionDifficulty: Người tham gia có bị mù hoặc gặp khó khăn về thị lực không.

21. DifficultyConcentrating: Người tham gia có gặp khó khăn trong việc tập trung không.

22. DifficultyWalking: Người tham gia có gặp khó khăn trong việc đi bộ không.

23. DifficultyDressingBathing: Người tham gia có gặp khó khăn trong việc mặc quần áo hoặc tắm rửa không.

24. DifficultyErrands: Người tham gia có gặp khó khăn trong việc thực hiện các công việc vặt không.

25. SmokerStatus: Tình trạng hút thuốc của người tham gia (hiện tại, trước đây, chưa bao giờ).

26. ECigaretteUsage: Tình trạng sử dụng thuốc lá điện tử của người tham gia.

27. ChestScan: Người tham gia có từng chụp CT scan hoặc X-ray ngực không.

28. RaceEthnicityCategory: Chủng tộc hoặc dân tộc của người tham gia.

29. AgeCategory: Nhóm tuổi của người tham gia.

30. HeightInMeters: Chiều cao của người tham gia (tính bằng mét).

31. WeightInKilograms: Cân nặng của người tham gia (tính bằng kilôgam).

32. BMI: Chỉ số khối cơ thể (Body Mass Index) của người tham gia.

33. AlcoholDrinkers: Tình trạng uống rượu của người tham gia.

34. HIVTesting: Người tham gia có từng làm xét nghiệm HIV không.

35. FluVaxLast12: Người tham gia có tiêm vắc-xin cúm trong 12 tháng qua không.

36. PneumoVaxEver: Người tham gia có từng tiêm vắc-xin phế cầu (pneumococcal vaccine) không.

37. TetanusLast10Tdap: Người tham gia có tiêm vắc-xin uốn ván trong 10 năm qua không.

38. HighRiskLastYear: Người tham gia có thuộc nhóm nguy cơ cao trong năm qua không.

39. CovidPos: Người tham gia có từng dương tính với COVID-19 không.

40. HadHeartAttack: Người tham gia có từng bị đau tim (heart attack) không

Những thuộc tính này cung cấp thông tin về tình trạng sức khỏe, thói quen và các yếu tố rủi ro của người tham gia, có thể được sử dụng để phân tích dữ liệu và xây dựng các mô hình dự đoán liên quan đến sức khỏe.

Các thư viện chính được sử dụng trong phân tích như : matplotlib, seaborn, pandas, numpy, plotly, scipy, sklearn

1. **Câu hỏi phân tích**

- Các yếu tố nào ảnh hưởng nhiều nhất đến tình trạng sức khỏe tổng quát của người tham gia?

- Có mối quan hệ nào giữa số ngày sức khỏe tinh thần không tốt và các yếu tố khác như số ngày sức khỏe thể chất không tốt và thời gian ngủ?

- Các mô hình học máy nào dự đoán tốt nhất về bệnh tim?

- Những nhóm người nào có nguy cơ cao hơn bị các bệnh mạn tính như COPD, hen suyễn, và viêm khớp?

- Có sự chênh lệch về số lượng người bệnh tim giữa giới tính hoặc các yếu tố khác?

- Mối liên hệ giữa việc tiêm phòng và tình trạng sức khỏe tổng quát của người tham gia như thế nào?

- Tình trạng sức khỏe của người tham gia thay đổi như thế nào theo độ tuổi?

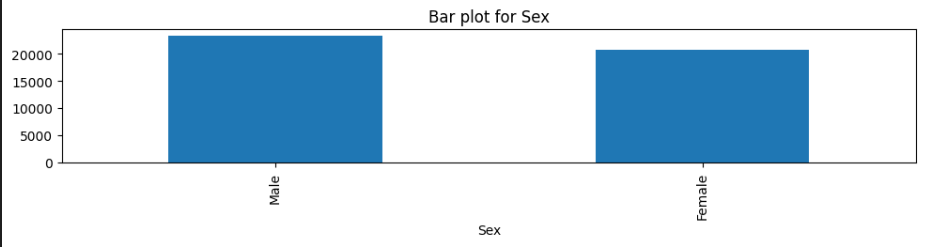
- Có mối quan hệ nào giữa tình trạng hút thuốc và các vấn đề sức khỏe khác nhau không?

- Có thể dự đoán tình trạng sức khỏe tổng quát của người tham gia dựa trên các yếu tố khác không?

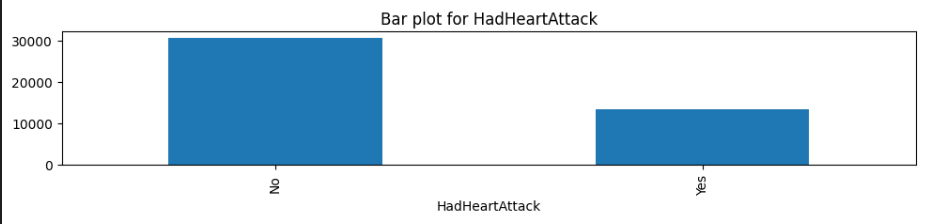
- Các triệu chứng nào thường xảy ra ở người bị bệnh tim?

1. **EDA và tiền xử lý.**
   1. **Kiểm tra phân bố.**

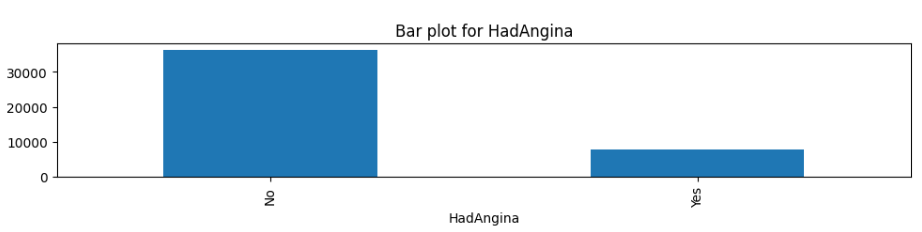
* **Sex:**

****

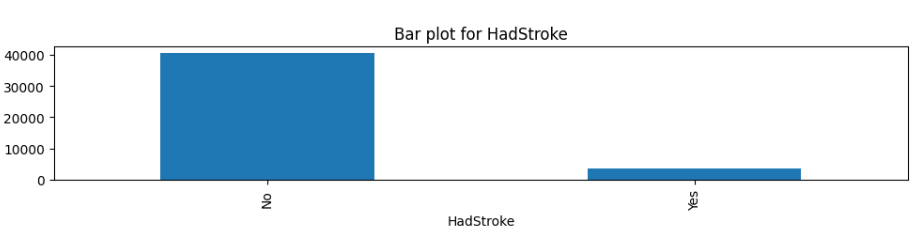
* **HadHeartAttack**

****

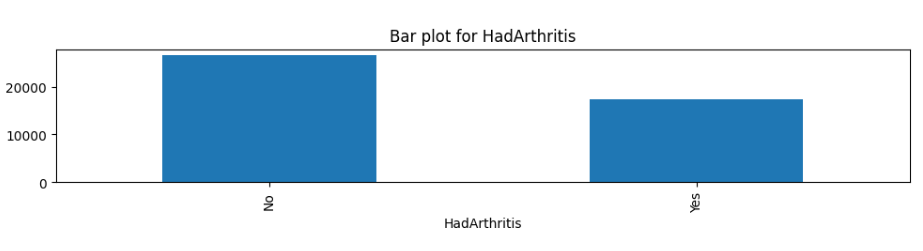
* **HadAgina:**

****

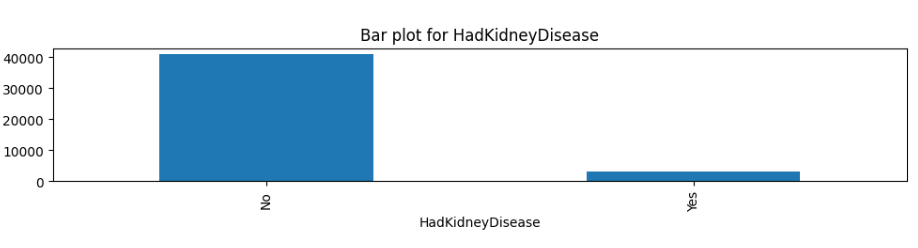
* **HadStroke:**

****

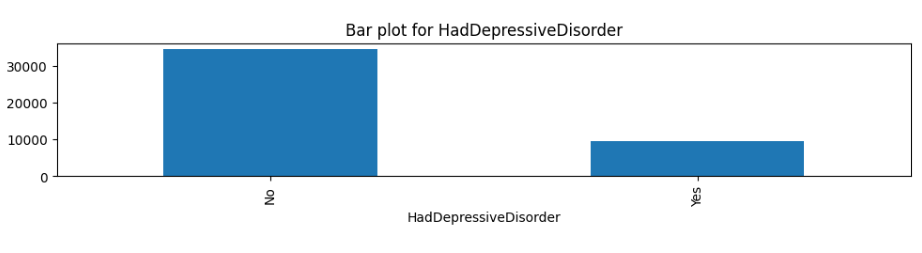
* **HadAthritis:**

****

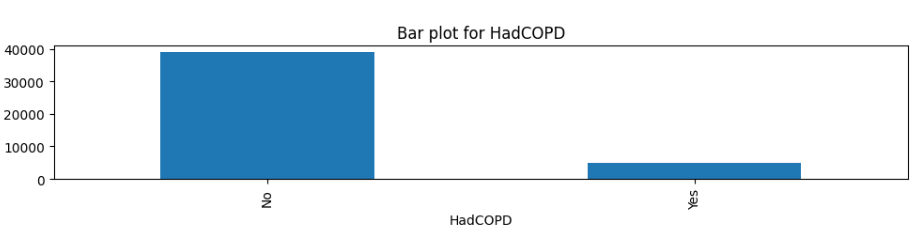
* **HadKidneyDisease**

****

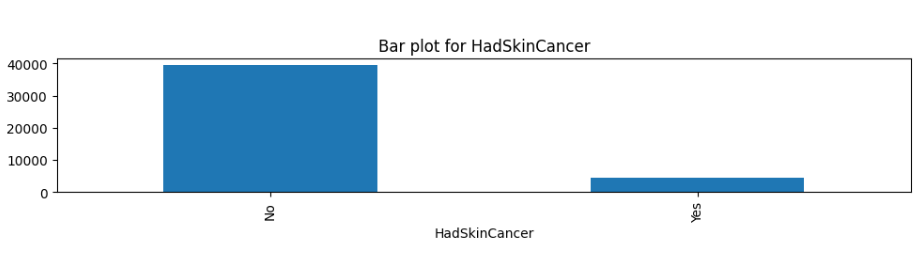
* **HadDepressiveDisorder**

****

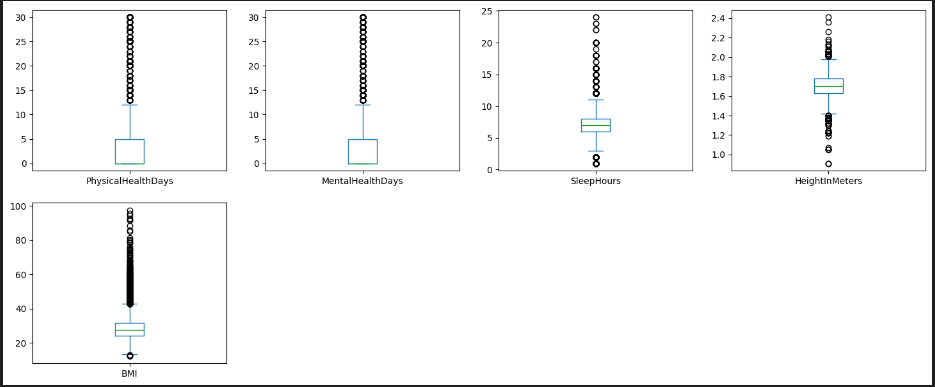
* **HadCOPD**

****

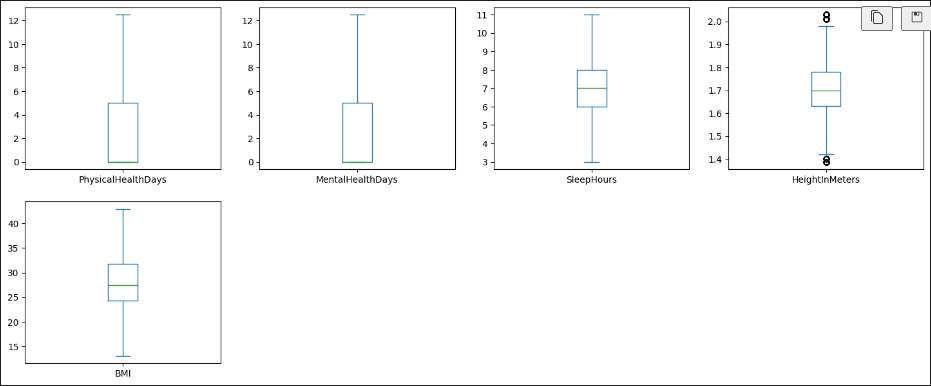
* **HadSkinCancer**

****

* 1. **Xử lý ngoại biên.**
* **Trước khi xử lý:**

****

* **Xóa các giá trị ngoại biên:**

****

* 1. **Chuẩn hóa dữ liệu.**
* **Chuẩn hóa các giá trị categories:**

**categories = [**

**['Poor', 'Fair', 'Good', 'Very good', 'Excellent'],**

**['Age 80 or older', 'Age 55 to 59', 'Age 70 to 74', 'Age 40 to 44',**

**'Age 75 to 79', 'Age 65 to 69', 'Age 60 to 64', 'Age 50 to 54',**

**'Age 45 to 49', 'Age 35 to 39', 'Age 25 to 29', 'Age 30 to 34',**

**'Age 18 to 24']**

**]**

**encoder = OrdinalEncoder(categories=categories)**

**encoded\_data = encoder.fit\_transform(df\_preprocess[['GeneralHealth', 'AgeCategory']])**

* 1. **Đánh giá tương quan.**
* **Đánh giá tương quan về các chỉ số cơ thể:**

**#Các thuộc tính đo lường chỉ số cơ thể**

**['PhysicalHealthDays','BMI' , 'SleepHours', 'HeightInMeters', 'MentalHealthDays', 'HadHeartAttack']**

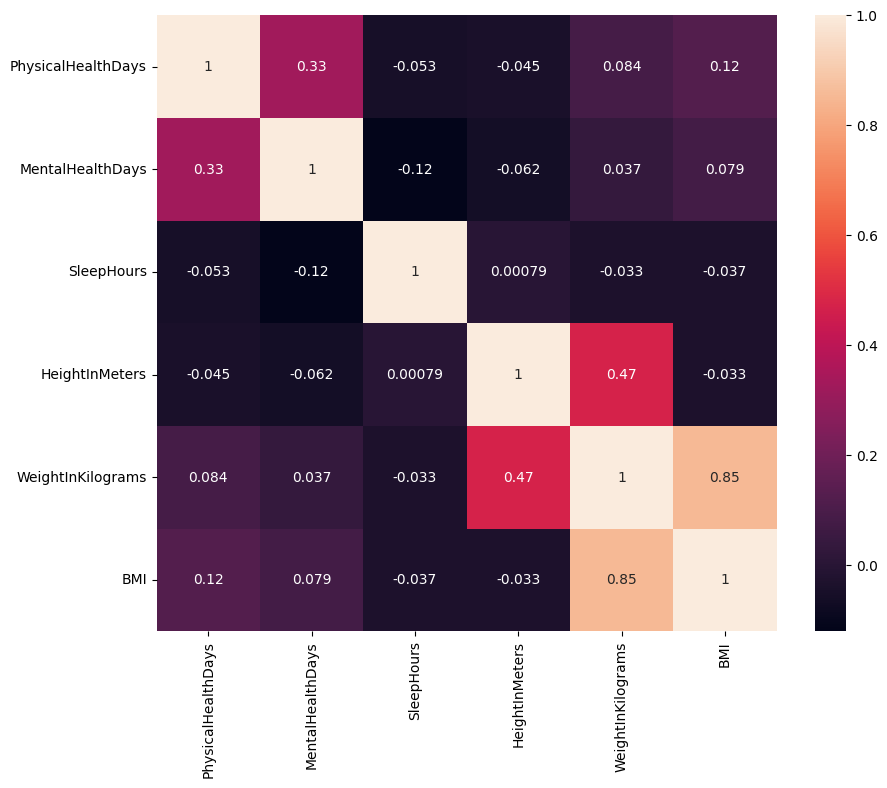
**df\_con = df\_preprocess.select\_dtypes(include='number')**

**correlation\_matrix = df\_con.corr()**

**plt.figure(figsize=(10,8))**

**sns.heatmap(correlation\_matrix, annot=True)**

**plt.show()**

****

* 1. **Xử lý mất cân bằng dữ liệu.**

RandomOverSampler giải quyết vấn đề này bằng cách sao chép ngẫu nhiên các mẫu từ các lớp thiểu số để tăng số lượng mẫu trong các lớp này, làm cho các lớp trở nên cân bằng hơn. Điều này giúp các mô hình học máy học được nhiều hơn từ các lớp thiểu số, cải thiện hiệu suất tổng thể của mô hình.

**from imblearn.over\_sampling import RandomOverSampler**

**X = df\_preprocess.drop(['HadHeartAttack'],axis=1)**

**y = df\_preprocess['HadHeartAttack']**

**sampler = RandomOverSampler(sampling\_strategy='minority')**

**X\_resampled, y\_resampled = sampler.fit\_resample(X, y)**

1. **Mô hình hóa dữ liệu (data modeling)**

* Sử dụng GridSearchCV để tìm ra siêu tham số tốt nhất cho từng mô hình

GridSearchCV: là một công cụ trong thư viện scikit-learn của Python, được sử dụng để tự động hóa quá trình điều chỉnh siêu tham số của các mô hình học máy. GridSearchCV thực hiện một tìm kiếm toàn diện trên một lưới các giá trị siêu tham số đã được xác định trước.

Cách hoạt động của GridSearchCV

- Khởi tạo: Bạn cung cấp mô hình học máy, một lưới các giá trị siêu tham số cần thử nghiệm, và các tham số khác như số lượng folds cho cross-validation.

- Cross-Validation: GridSearchCV chia dữ liệu huấn luyện thành các fold (mặc định là 5) và thực hiện cross-validation cho mỗi tổ hợp của các giá trị siêu tham số.

- Đánh giá: Với mỗi tổ hợp siêu tham số, GridSearchCV tính toán một điểm số (như độ chính xác) dựa trên cross-validation.

- Chọn lựa: GridSearchCV chọn tổ hợp siêu tham số nào cho điểm số tốt nhất và sử dụng tổ hợp này để huấn luyện mô hình cuối cùng trên toàn bộ dữ liệu huấn luyện.

* Sau đó lưu lại mô hình: phục vụ cho việc tổng hợp và đánh giá
  1. **Random Forest**

Random Forest là một phương pháp học máy thuộc nhóm học có giám sát, sử dụng nhiều cây quyết định để dự đoán và phân loại. Mỗi cây trong rừng ngẫu nhiên được xây dựng từ một mẫu ngẫu nhiên của dữ liệu huấn luyện và chỉ sử dụng một tập con ngẫu nhiên của các đặc trưng để chia nhánh. Kết quả cuối cùng là trung bình (đối với dự đoán liên tục) hoặc bỏ phiếu số đông (đối với phân loại) từ tất cả các cây trong rừng.

* 1. **Logistic Regression**

Logistic Regression là một kỹ thuật thống kê dùng để mô hình hóa xác suất của một biến nhị phân dựa trên một hoặc nhiều biến giải thích. Mô hình này sử dụng hàm logistic để chuyển đổi đầu ra của một hàm tuyến tính thành giá trị xác suất từ 0 đến 1.

from sklearn.linear\_model import LogisticRegression

# Khởi tạo mô hình logistic regression

model = LogisticRegression()

# Định nghĩa lưới các giá trị tham số

param\_grid = {

    'penalty': ['l1', 'l2', 'elasticnet', 'none'],

    'C': [0.01, 0.1, 1, 10, 100],

    'solver': ['lbfgs', 'liblinear', 'saga']

}

# Khởi tạo GridSearchCV

grid\_search = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=5, scoring='precision')

# Thực hiện tìm kiếm trên tập huấn luyện

grid\_search.fit(X\_train, y\_train)

# Sử dụng mô hình với các tham số tối ưu để dự đoán trên tập kiểm tra

best\_model = grid\_search.best\_estimator\_

# Lưu mô hình tốt nhất vào một file

joblib\_file\_LR = "../Model/best\_model\_LR.pkl"

joblib.dump(best\_model, joblib\_file\_LR)

* 1. **Gradient Descent**

Gradient Descent là một thuật toán tối ưu hóa được sử dụng để tìm giá trị tối thiểu của hàm mất mát trong học máy. Bằng cách di chuyển ngược chiều gradient của hàm mất mát, thuật toán giảm dần giá trị của hàm này, giúp tìm ra các tham số tối ưu cho mô hình. Có nhiều biến thể của Gradient Descent như Batch Gradient Descent, Stochastic Gradient Descent (SGD), và Mini-batch Gradient Descent.

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

# Define the grid of hyperparameters to search

param\_grid = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [10, 30],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

# Initialize GridSearchCV with the Random Forest model

grid\_search = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=3, scoring='precision')

# Perform the grid search on the training data

grid\_search.fit(X\_train, y\_train)

# Use the best model found by GridSearchCV to make predictions on the test set

best\_model = grid\_search.best\_estimator\_

# Lưu mô hình tốt nhất vào một file

joblib\_file = "../Model/best\_rondom.pkl"

joblib.dump(best\_model, joblib\_file)

print(f'Model saved to {joblib\_file}')

* 1. **SVC**

SVC là một thuật toán học có giám sát dùng để phân loại dữ liệu. SVC tìm ra một siêu phẳng (hyperplane) tốt nhất để phân chia các điểm dữ liệu của các lớp khác nhau sao cho khoảng cách giữa các điểm dữ liệu gần nhất của hai lớp (margin) là lớn nhất. Kernel trick được sử dụng để xử lý dữ liệu không tuyến tính bằng cách ánh xạ dữ liệu vào không gian cao hơn.

from sklearn.svm import SVC

para = {

    'C': [0.1, 10, 100],

    'gamma': [0.01, 0.1, 1],

    'kernel': ['linear', 'sigmoid', None]

}

#Huấn luyện mô hình

grid = GridSearchCV(SVC(), para, cv = 2, scoring = 'precision')

grid.fit(X\_train, y\_train)

grid.best\_estimator\_

best\_model = grid.best\_estimator\_

#Lưu lại mô hình

joblib\_file = "../Model/svc.pkl"

joblib.dump(best\_model, joblib\_file)

* 1. **Naive Bayes**

from sklearn.naive\_bayes import GaussianNB

import joblib

# Khởi tạo mô hình Gaussian Naive Bayes

model = GaussianNB()

# Thiết lập lưới các siêu tham số để tìm kiếm

param\_grid = {

    'var\_smoothing': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 1e-03, 1e-02, 1e-01, 1.0]

}

# Sử dụng GridSearchCV để tìm kiếm siêu tham số tốt nhất

grid\_search = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=5, n\_jobs=-1, scoring='precision')

grid\_search.fit(X\_train, y\_train)

print("Best parameters found: ", grid\_search.best\_params\_)

# Đánh giá mô hình với tập kiểm tra

best\_model = grid\_search.best\_estimator\_

joblib\_file = "../Model/NaiveBayes\_model.pkl"

joblib.dump(best\_model, joblib\_file)

* 1. **SGDClassifier**

from sklearn.linear\_model import SGDClassifier

# Khởi tạo mô hình logistic regression

model\_GD = SGDClassifier()

# Định nghĩa lưới các giá trị tham số

param\_grid\_GD = {

    'learning\_rate': ['optimal', 'invscaling'],

    'eta0': [0.1, 0.01, 0.001, 0.0001],

    'max\_iter': [500, 1000, 1500, 2000]

}

# Khởi tạo GridSearchCV

grid\_search\_GD = GridSearchCV(estimator=model\_GD, param\_grid=param\_grid\_GD, cv=5, scoring='precision')

# Thực hiện tìm kiếm trên tập huấn luyện

grid\_search\_GD.fit(X\_train, y\_train)

# Sử dụng mô hình với các tham số tối ưu để dự đoán trên tập kiểm tra

best\_model\_GD = grid\_search\_GD.best\_estimator\_

# Lưu mô hình tốt nhất vào một file

joblib\_file\_GD = "../Model/best\_model\_GD.pkl"

joblib.dump(best\_model\_GD, joblib\_file\_GD)

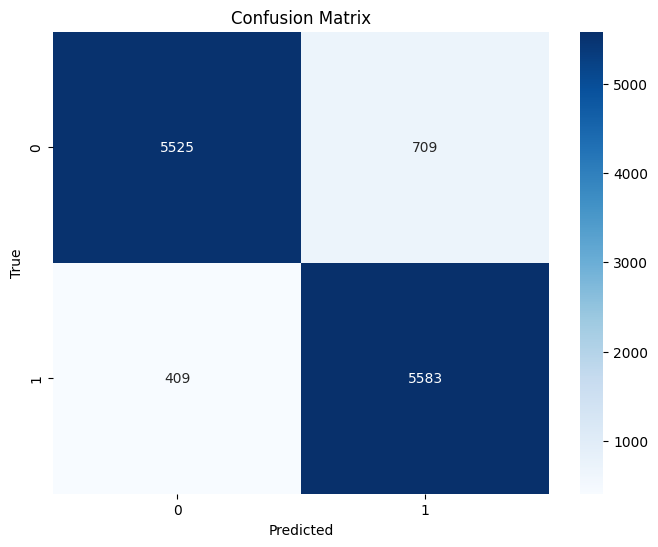
* 1. **Kết quả của các mô hình**
* RandomForest

Precision: 0.8873172282263191

Recall: 0.9317423230974633

Accuracy: 0.9085555373793555

F1 Score: 0.9089873005535656



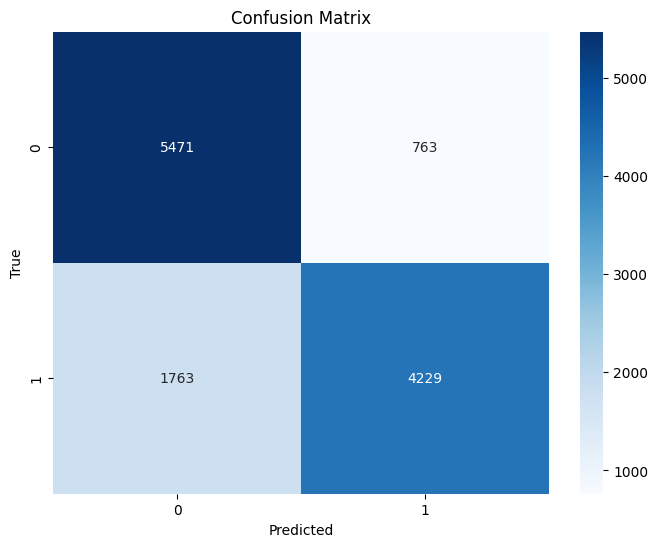
- SGDClassifier:

Precision: 0.8471554487179487

Recall: 0.7057743658210948

Accuracy: 0.7933911336495992

F1 Score: 0.7700291332847778



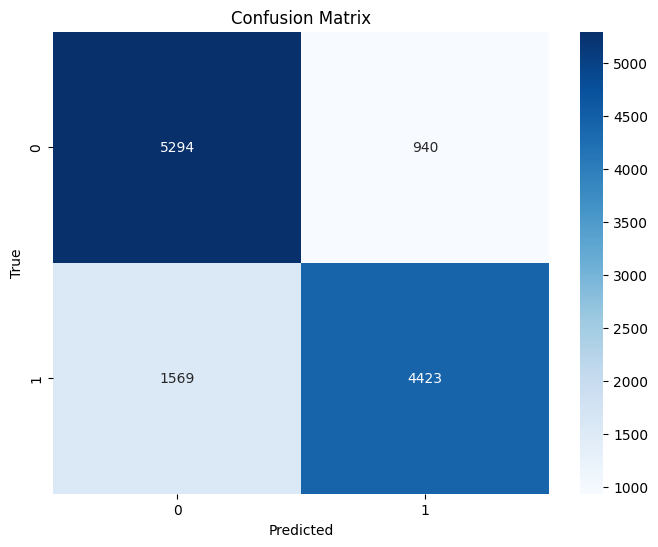
- Logistic regression:

Precision: 0.8247249673690099

Recall: 0.738150867823765

Accuracy: 0.7947816129559954

F1 Score: 0.7790400704535447



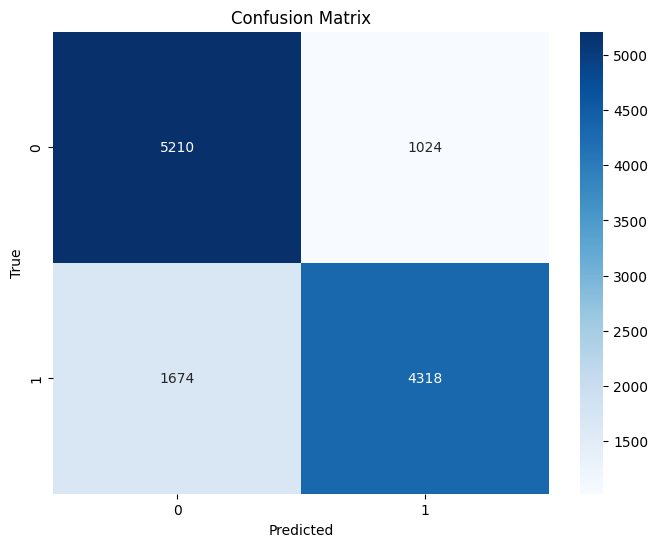
- Naive Bayes

Precision: 0.8083114938225384

Recall: 0.7206275033377837

Accuracy: 0.7793227547848847

F1 Score: 0.7619551791071113



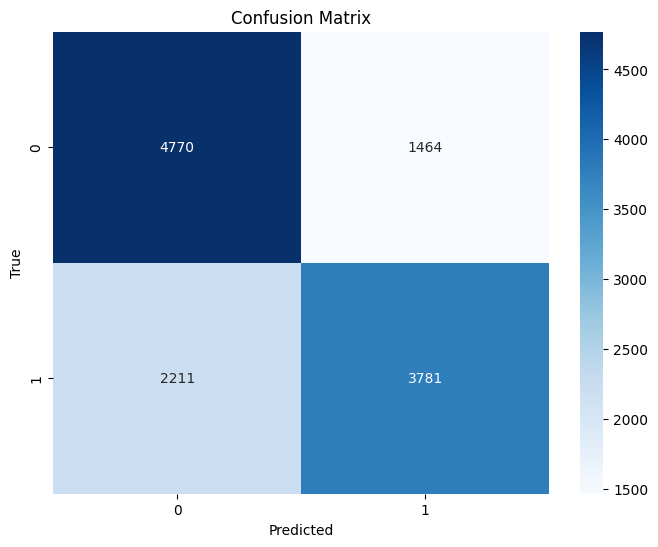
- SVC

Precision: 0.7208770257387989

Recall: 0.6310080106809078

Accuracy: 0.699411091117291

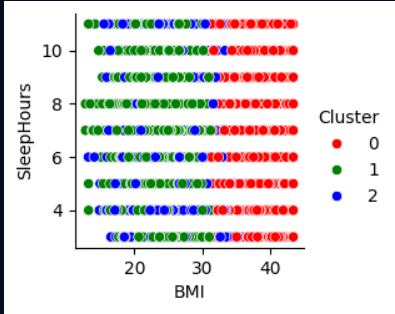
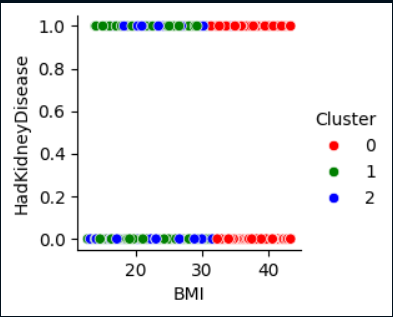
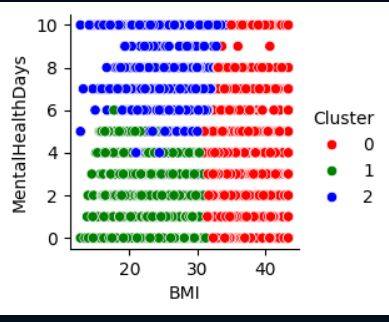
F1 Score: 0.6729554151463913



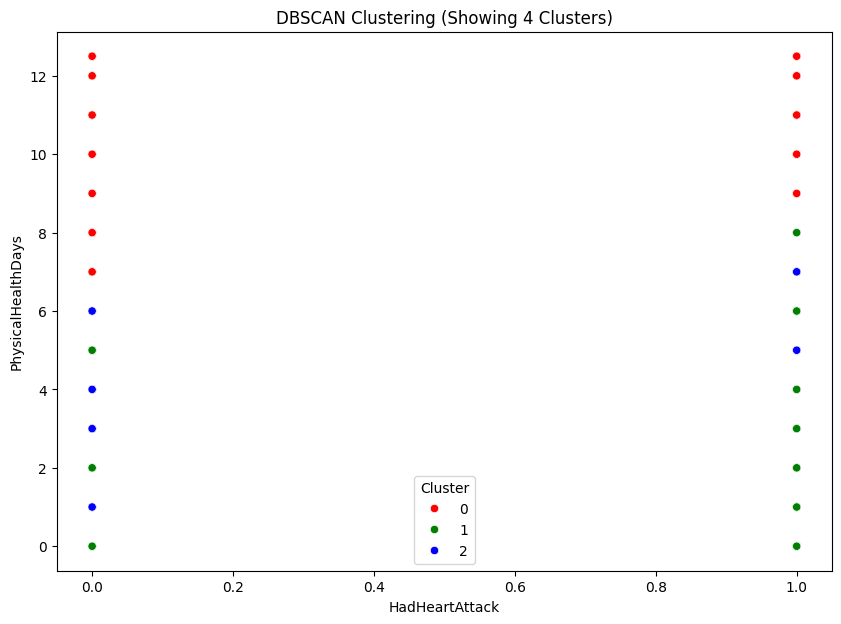
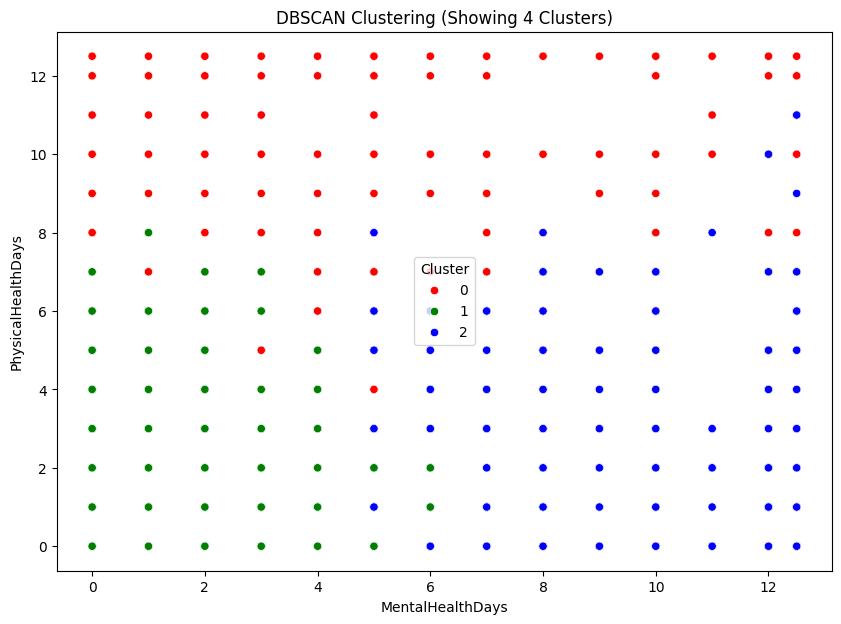
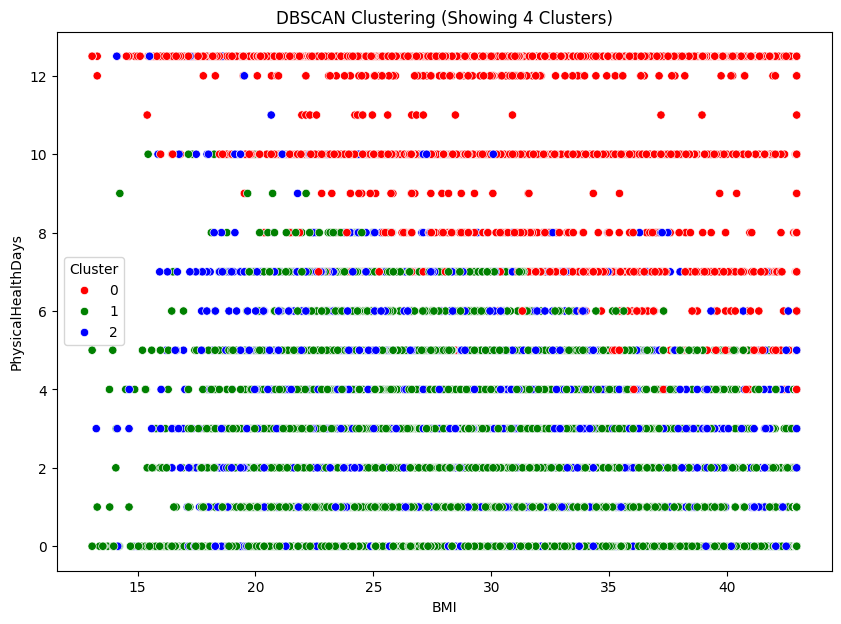
Dựa trên chỉ số hiệu suất phân tích được từ các mô hình trên, ta nhận thấy Random Forest đạt hiệu suất cao nhất với hơn 80%, cho thấy khả năng phân loại chính xác các mẫu thuộc lớp dương tính tốt hơn so với các mô hình khác. Trong khi đó, Logistic Regression, Gradient Descent và SVC có tỷ lệ tương đối tương đương nhau với hơn 79%, cho thấy cả ba đều có khả năng phân loại khá tốt nhưng không vượt trội như Random Forest.

1. **Bài toán phân cụm và khai phá luật kết hợp**

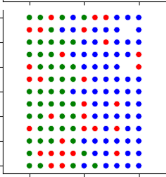
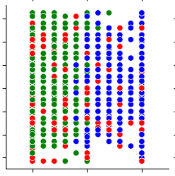
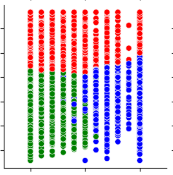
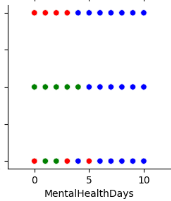
Để giải quyết bài toán phân nhóm bệnh nhân dựa trên các triệu chứng bệnh hoặc thói quen sống của bệnh nhân, bạn có thể sử dụng ba thuật toán phân cụm phổ biến: DBSCAN, KMeans và Agglomerative Clustering.

* Xem các biểu đồ vẽ dựa trên BMI thì ta quan sát được là những những người trong cluster 0 sẽ là những người có chỉ số BMI cao vượt qua định mức cân đối là (18,5-24,9) cho thấy những người này có nguy cơ béo phì

* Những người có lối sức khỏe thể chất không ổn thường nằm ở cluster 0 điều đó càng chứng minh rõ hơn chúng ta cần quan tâm nhiều hơn đến những người này(chúng ta vừa xem họ ở trên là chỉ sô BMI khá cao).

* Những người có số ngày sức khỏe tinh thần không tốt sẽ chủ yếu ở cluster 3, thì cho ta thấy những người này chúng ta cần đặt biệt quan tâm đến tinh thần của họ

Thuật toán Apriori là một thuật toán khai phá luật kết hợp được sử dụng rộng rãi để phân tích các mẫu xuất hiện đồng thời trong tập dữ liệu. Trong bài toán phát hiện mối liên hệ giữa các triệu chứng và các căn bệnh về tim, Apriori có thể được áp dụng để tìm ra các luật kết hợp giữa các triệu chứng và phát hiện ra bệnh tim.



Trong kết quả của thuật toán khai phá mẫu, các cột thường được hiển thị như sau:

1. antecedents: Là tập hợp các yếu tố (hoặc mục) trong phần điều kiện của luật kết hợp. Đây là các yếu tố mà nếu chúng xuất hiện cùng nhau trong dữ liệu, chúng ta sẽ quan sát một sự xuất hiện của consequents (phần kết quả của luật kết hợp).
2. consequents: Là mục hoặc tập hợp các mục mà chúng ta mong muốn xem xuất hiện khi các yếu tố trong antecedents đồng thời xuất hiện.
3. antecedent support: Là tỷ lệ (hoặc số lượng) các lần mà tập hợp antecedents xuất hiện trong tất cả các mẫu trong dữ liệu.
4. consequent support: Là tỷ lệ (hoặc số lượng) các lần mà consequents xuất hiện trong tất cả các mẫu trong dữ liệu.
5. support: Là tỷ lệ (hoặc số lượng) các lần mà antecedents và consequents xuất hiện cùng nhau trong tất cả các mẫu trong dữ liệu.
6. confidence: Là xác suất có điều kiện của consequents dựa trên antecedents. Nói cách khác, đây là xác suất có điều kiện của consequents khi antecedents đã được quan sát.
7. lift: Là một chỉ số cho biết mức độ tăng của xác suất của consequents khi antecedents xuất hiện. Lift càng cao, mối quan hệ giữa antecedents và consequents càng mạnh. Lift lớn hơn 1 chỉ ra rằng antecedents và consequents xuất hiện cùng nhau thường xuyên hơn so với việc chúng xuất hiện ngẫu nhiên.
8. leverage: Là một chỉ số đo lường mức độ mà sự xuất hiện đồng thời của antecedents và consequents trong dữ liệu nhiều hơn (hoặc ít hơn) so với những gì được mong đợi nếu chúng không liên quan với nhau. Nó cũng cung cấp thông tin về mức độ tăng hoặc giảm của sự xuất hiện của consequents khi antecedents xuất hiện.
9. conviction: Là một chỉ số mà mức độ độc lập của consequents với antecedents. Conviction càng cao, mối quan hệ giữa antecedents và consequents càng mạnh. Conviction bằng 1 chỉ ra rằng antecedents và consequents không liên quan.
10. zhangs\_metric: Là một chỉ số thay thế cho lift, được sử dụng trong một số trường hợp. Nó cũng đo lường mức độ tăng của xác suất của consequents khi antecedents xuất hiện.

* Hiển thị các các cặp chiệu chứng hay xuất hiện nhiều nhất:

frequent\_itemsets\_2\_items = frequent\_itemsets[frequent\_itemsets['itemsets'].apply(lambda x: len(x) == 2)]

print(frequent\_itemsets\_2\_items)

* Bên dưới là các cặp triệu chứng đi kèm nhiều nhất

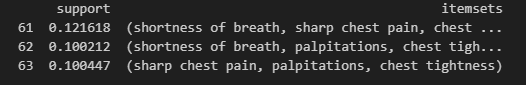


* 3 triệu chứng thường gặp nhất:

frequent\_itemsets\_3\_items = frequent\_itemsets[frequent\_itemsets['itemsets'].apply(lambda x: len(x) == 3)]

# Hiển thị các itemsets chứa đúng 3 mục

print(frequent\_itemsets\_3\_items)

****

# **TÀI LIỆU THAM KHẢO**

1. <https://mindx.edu.vn/blog/eda-la-gi>
2. <https://www.kaggle.com/code/prathamjainzee/eda-disease-symptom-dataset-773>
3. <https://www.kaggle.com/code/yogitasingh05/heartdiseaseprediction-2020#Using-Random-Oversampling-and-PCA>
4. <https://www.python.org/>