

# BÁO CÁO ĐỒ ÁN CUỐI KỲ

## HEART DISEASE PREDICTION

NHÓM 13:

21520473 - Lê Quốc Thuận

21520578 - Nguyễn Thị Vân Anh

Hệ hỗ trợ quyết định - IS254.021

GVHD: ThS. Nguyễn Hồ Duy Trí

# OVERVIEW

- Lý do chọn đề tài
- Mô tả dữ liệu
- Mô tả bài toán
- Các mô hình được lựa chọn
- Quy trình ra quyết định
- Kết quả đạt được
- Kết luận

# LÝ DO CHỌN ĐỀ TÀI

- **Bệnh tim mạch ngày càng phổ biến và gia tăng nhanh chóng**
  - Là nguyên nhân hàng đầu gây tử vong toàn cầu, cướp đi 19,5 triệu sinh mạng mỗi năm.
  - Nguyên nhân chủ yếu do di truyền, tuổi tác, lối sống, thói quen sinh hoạt, chế độ ăn uống, ...
- **Cần tìm kiếm và ngăn ngừa các yếu tố ảnh hưởng đến bệnh tim mạch.**
- **Ứng dụng mô hình học máy:**
  - Xác định bệnh nhân có mắc bệnh tim mạch hay không.
  - Đẩy nhanh quá trình chẩn đoán và kiểm soát bệnh tim mạch.

# MÔ TẢ DỮ LIỆU

- Tập dữ liệu Heart Disease.xlsx được thu thập từ Kaggle. Tập dữ liệu gồm 319795 dòng và 18 trường dữ liệu.
- Tập dữ liệu chứa các thông tin y tế về bệnh nhân và kết quả chẩn đoán xem bệnh nhân có mắc các bệnh tim mạch hay không.



# MÔ TẢ DỮ LIỆU

- 1. BMI (#)
- 2. Smoking: 1= Yes, 0= No (Binary)
- 3. AlcoholDrinking: 1= Yes, 0= No (Binary)
- 4. Stroke: 1= Yes, 0= No (Binary)
- 5. PhysicalHealth (#)
- 6. MentalHealth(#)
- 7. DiffWalking: 1= Yes, 0= No (Binary)
- 8. Sex: 1= Male, 0= Female (Binary)
- 9. AgeCategory: (categorical)
- 10. Race: White/ Hispanic/ Others (categorical)
- 11. Diabetic: Yes/ No/ Others
- 12. PhysicalActivity: 1= Yes, 0= No (Binary)
- 13. GenHealth: Very good/ Good/ Fair/ Poor/Excellent... (Ordinal)
- 14. SleepTime (#)
- 15. Asthma : 1= Yes, 0= No (Binary)
- 16. KidneyDisease: 1= Yes, 0= No (Binary)
- 17. SkinCancer: 1= Yes, 0= No (Binary)
- 18. HeartDisease: 1= Yes, 0= No (Binary)

# MÔ TẢ BÀI TOÁN

## Mục tiêu

dự đoán một cá nhân có mắc bệnh tim mạch không dựa trên các yếu tố nguy cơ của họ.

## Dữ liệu

Tập dữ liệu chứa các thông tin y tế về bệnh nhân và tình trạng mắc bệnh tim mạch

## Thử nghiệm

Thử nghiệm với nhiều Mô hình phân loại khác nhau và xem mô hình nào mang lại độ chính xác cao nhất.

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

```

log = LogisticRegression()
log.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
log_pred = log.predict(X_test)
log_score = metrics.accuracy_score(y_test, log_pred)
print("Logistic Regression Accuracy on Test Data:", log_score)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_log_pred = log.predict(X_train)
train_log_score = metrics.accuracy_score(y_train, train_log_pred)
print("Logistic Regression Accuracy on Train Data:", train_log_score)

print("Logistic Regression Report:", metrics.classification_report(y_test, log_pred))

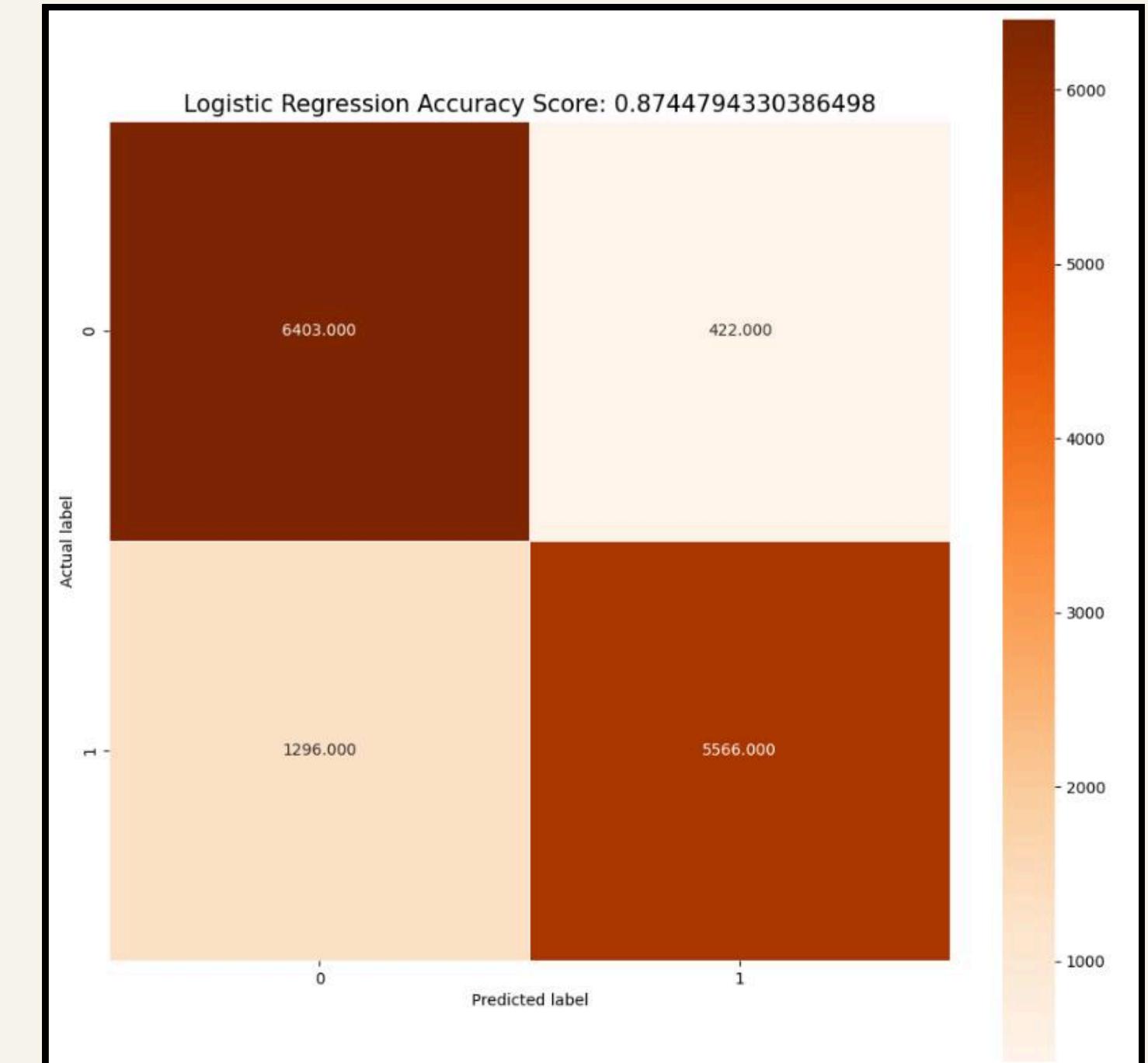
```

Logistic Regression Accuracy on Test Data: 0.8744794330386498  
 Logistic Regression Accuracy on Train Data: 0.8723544168148274  
 Logistic Regression Report:

	precision	recall	f1-score	support
0	0.83	0.94	0.88	6825
1	0.93	0.81	0.87	6862
accuracy			0.87	13687
macro avg	0.88	0.87	0.87	13687
weighted avg	0.88	0.87	0.87	13687

Report of Logistic Regression

Logistic Regression



Confusion Matrix for Logistic Regression

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

7

## Decision Tree

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
clf_pred = clf.predict(X_test)
clf_score = metrics.accuracy_score(y_test, clf_pred)
print("Decision Tree Classifier Accuracy on Test Data:", clf_score)

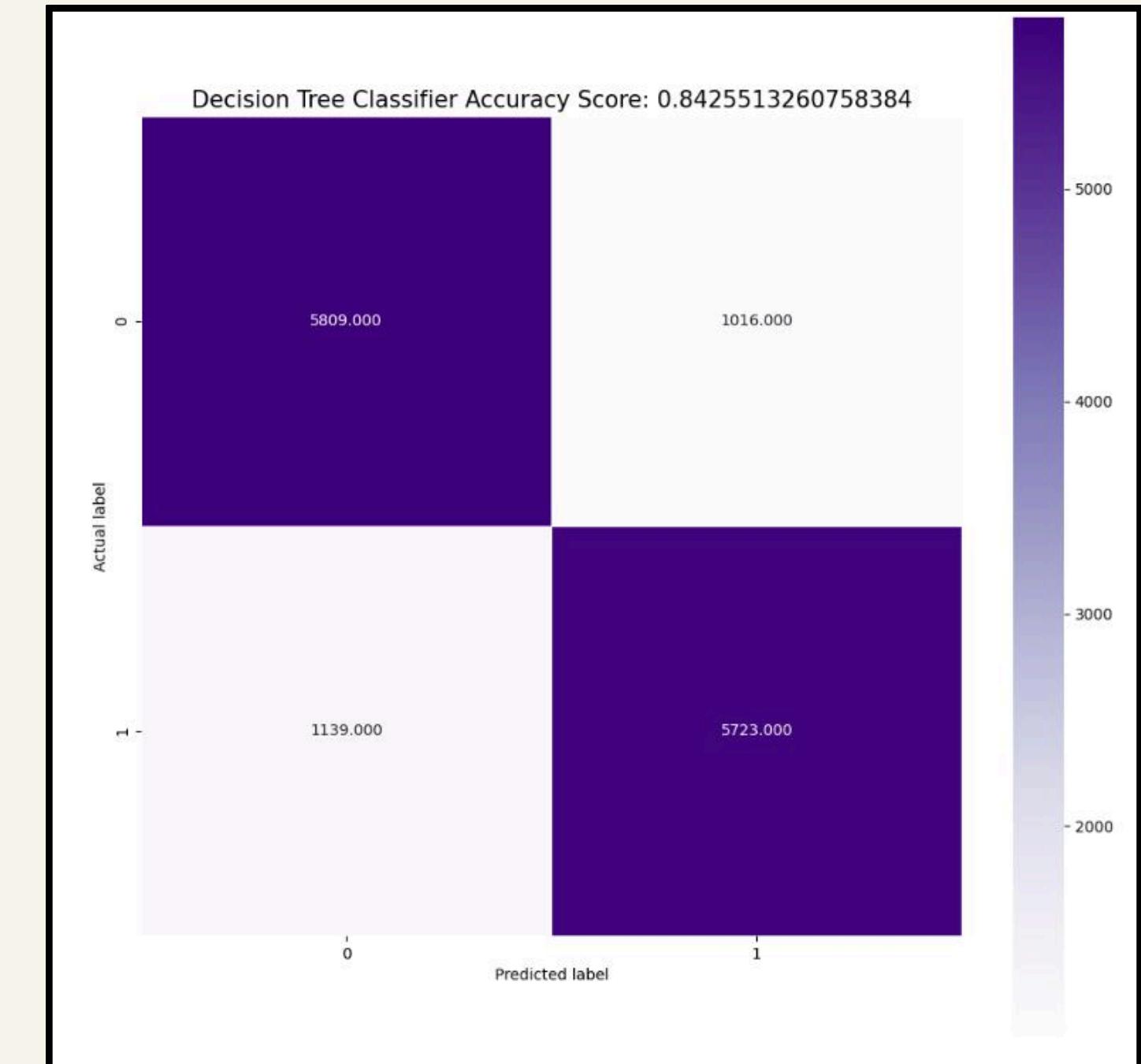
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_clf_pred = clf.predict(X_train)
train_clf_score = metrics.accuracy_score(y_train, train_clf_pred)
print("Decision Tree Classifier Accuracy on Train Data:", train_clf_score)

print("Decision Tree Classifier Report:", metrics.classification_report(y_test, clf_pred))

Decision Tree Classifier Accuracy on Test Data: 0.8417476437495434
Decision Tree Classifier Accuracy on Train Data: 0.9830731386541318
Decision Tree Classifier Report:
          precision    recall  f1-score   support
0       0.83      0.85      0.84     6825
1       0.85      0.83      0.84     6862

   accuracy                           0.84      13687
  macro avg       0.84      0.84      0.84     13687
weighted avg       0.84      0.84      0.84     13687
```

Report of Decision Tree



Confusion Matrix for Decision Tree

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

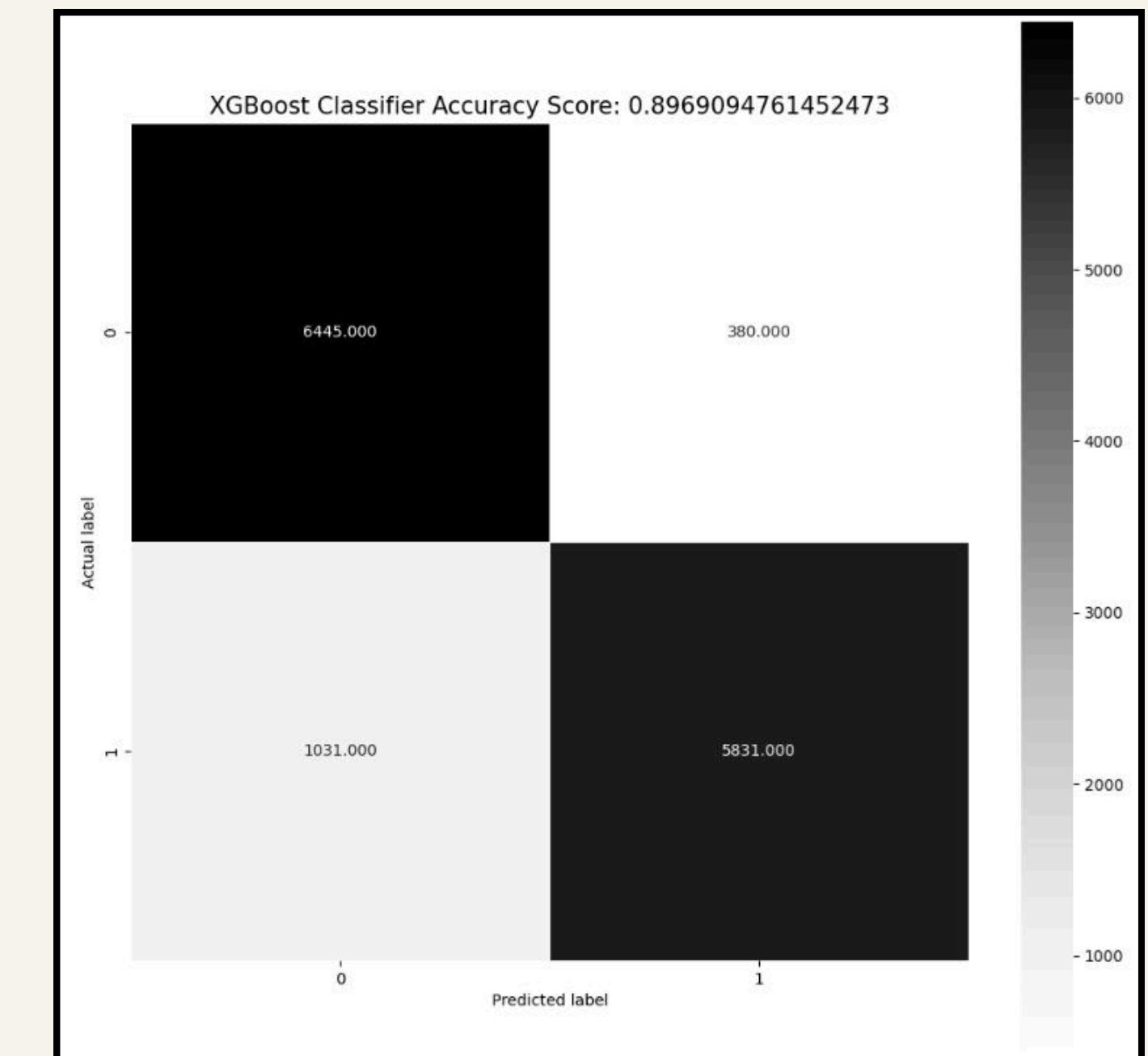
8

```
xgb = XGBClassifier()  
xgb.fit(X_train, y_train)  
  
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra  
xgb_pred = xgb.predict(X_test)  
xgb_score = metrics.accuracy_score(y_test, xgb_pred)  
print("XGBoost Classifier Accuracy on Test Data:", xgb_score)  
  
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện  
train_xgb_pred = xgb.predict(X_train)  
train_xgb_score = metrics.accuracy_score(y_train, train_xgb_pred)  
print("XGBoost Classifier Accuracy on Train Data:", train_xgb_score)  
print("XGBoost Classifier Report:", metrics.classification_report(y_test, xgb_pred))
```

XGBoost Classifier Accuracy on Test Data: 0.8969094761452473  
XGBoost Classifier Accuracy on Train Data: 0.9099831949146351  
XGBoost Classifier Report:

	precision	recall	f1-score	support
0	0.86	0.94	0.90	6825
1	0.94	0.85	0.89	6862
accuracy			0.90	13687
macro avg	0.90	0.90	0.90	13687
weighted avg	0.90	0.90	0.90	13687

Report of XGboost



XGBoost

Confusion Matrix for XGboost

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

```

ada = AdaBoostClassifier()
ada.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
ada_pred = ada.predict(X_test)
ada_score = metrics.accuracy_score(y_test, ada_pred)
print("Adaboost Classifier Accuracy on Test Data:", ada_score)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_ada_pred = ada.predict(X_train)
train_ada_score = metrics.accuracy_score(y_train, train_ada_pred)
print("Adaboost Classifier Accuracy on Train Data:", train_ada_score)

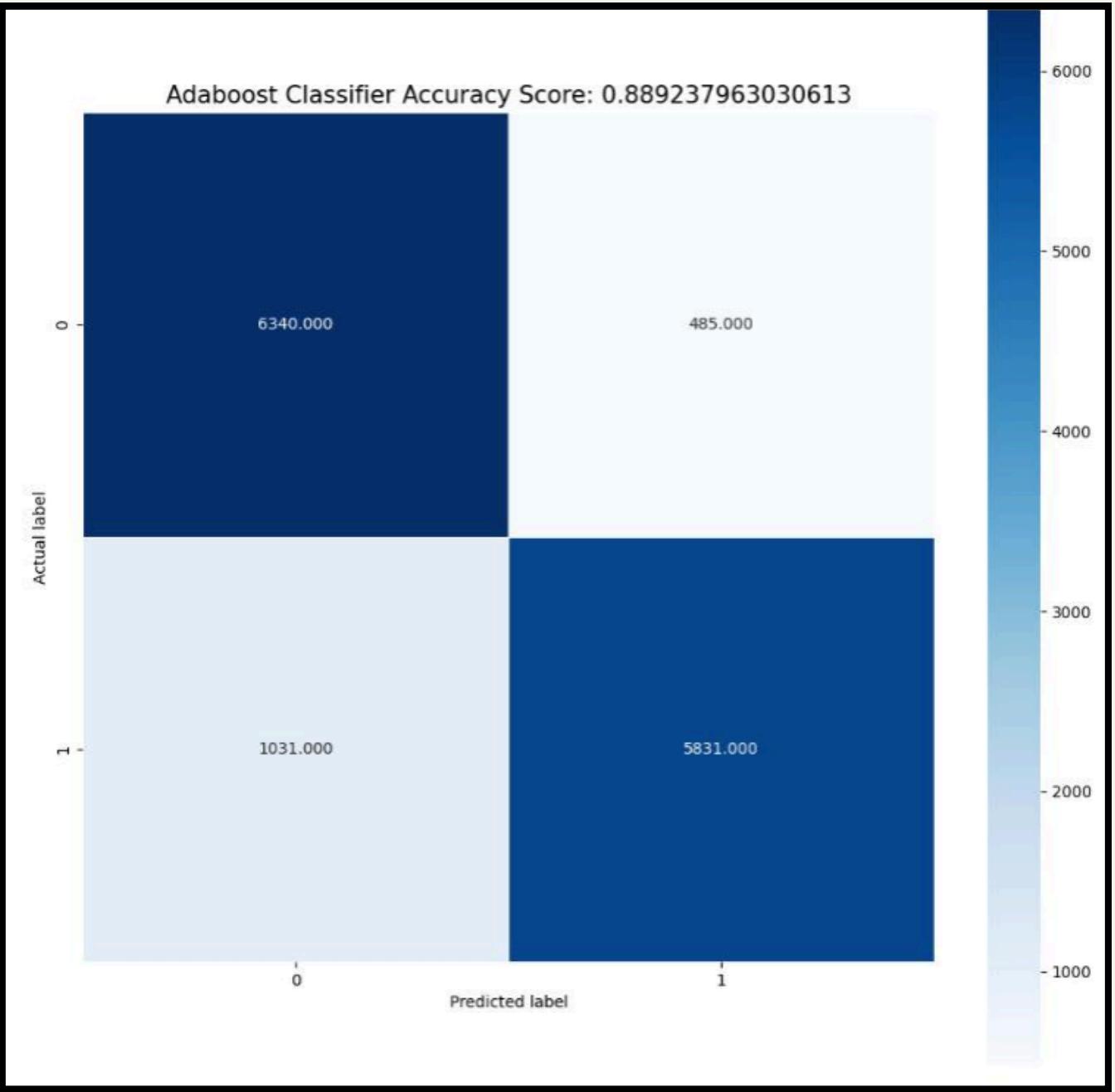
print("Adaboost Classifier Report:", metrics.classification_report(y_test, ada_pred))

Adaboost Classifier Accuracy on Test Data: 0.889237963030613
Adaboost Classifier Accuracy on Train Data: 0.8889403054141601
Adaboost Classifier Report:
          precision    recall  f1-score   support
0           0.86     0.93      0.89     6825
1           0.92     0.85      0.88     6862

accuracy                           0.89      0.89    0.89    13687
macro avg       0.89     0.89      0.89     13687
weighted avg    0.89     0.89      0.89     13687

```

Report of Adaboost



Confusion Matrix for Adaboost

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

10

## K-Nearest Neighbour (KNN)

```
knc = KNeighborsClassifier()
knc.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
knc_pred = knc.predict(X_test)
knc_score = metrics.accuracy_score(y_test, knc_pred)
print("KNearest Classifier Accuracy on Test Data:", knc_score)

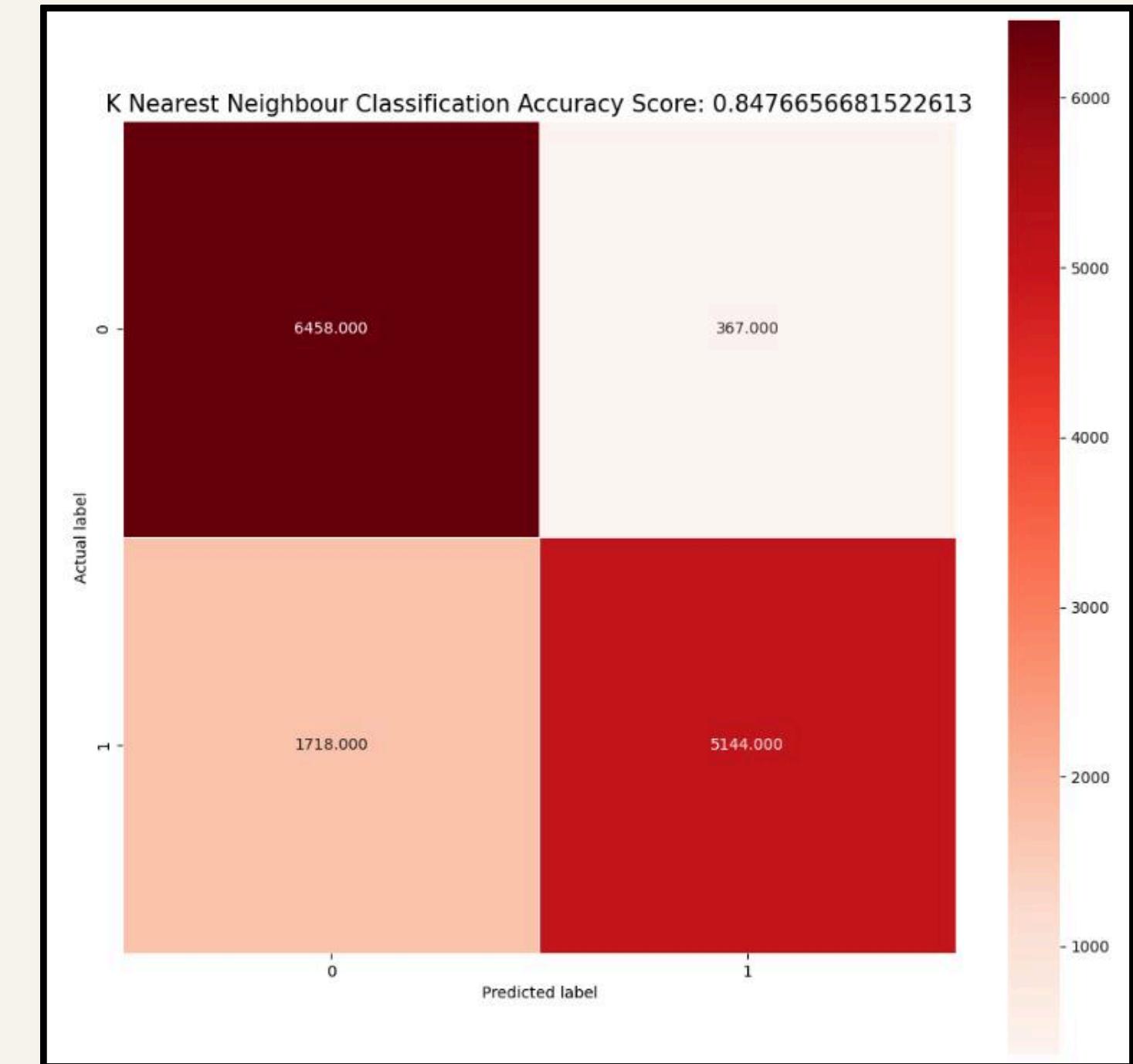
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_knc_pred = knc.predict(X_train)
train_knc_score = metrics.accuracy_score(y_train, train_knc_pred)
print("KNearest Classifier Accuracy on Train Data:", train_knc_score)

print("KNearest Classifier Report:", metrics.classification_report(y_test, knc_pred))

KNearest Classifier Accuracy on Test Data: 0.8476656681522613
KNearest Classifier Accuracy on Train Data: 0.8790520957646314
KNearest Classifier Report:
             precision    recall  f1-score   support
          0       0.79      0.95      0.86     6825
          1       0.93      0.75      0.83     6862

   accuracy                           0.86      13687
  macro avg       0.86      0.85      0.85     13687
weighted avg       0.86      0.85      0.85     13687
```

Report of Knearest Neighbour



Confusion Matrix for Knearest Neighbour

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

11

## Support Vector Machine (SVM)

```
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
svm_pred = svm.predict(X_test)
svm_score = metrics.accuracy_score(y_test, svm_pred)
print("Support Vector Machine Classification Accuracy on Test Data:", svm_score)

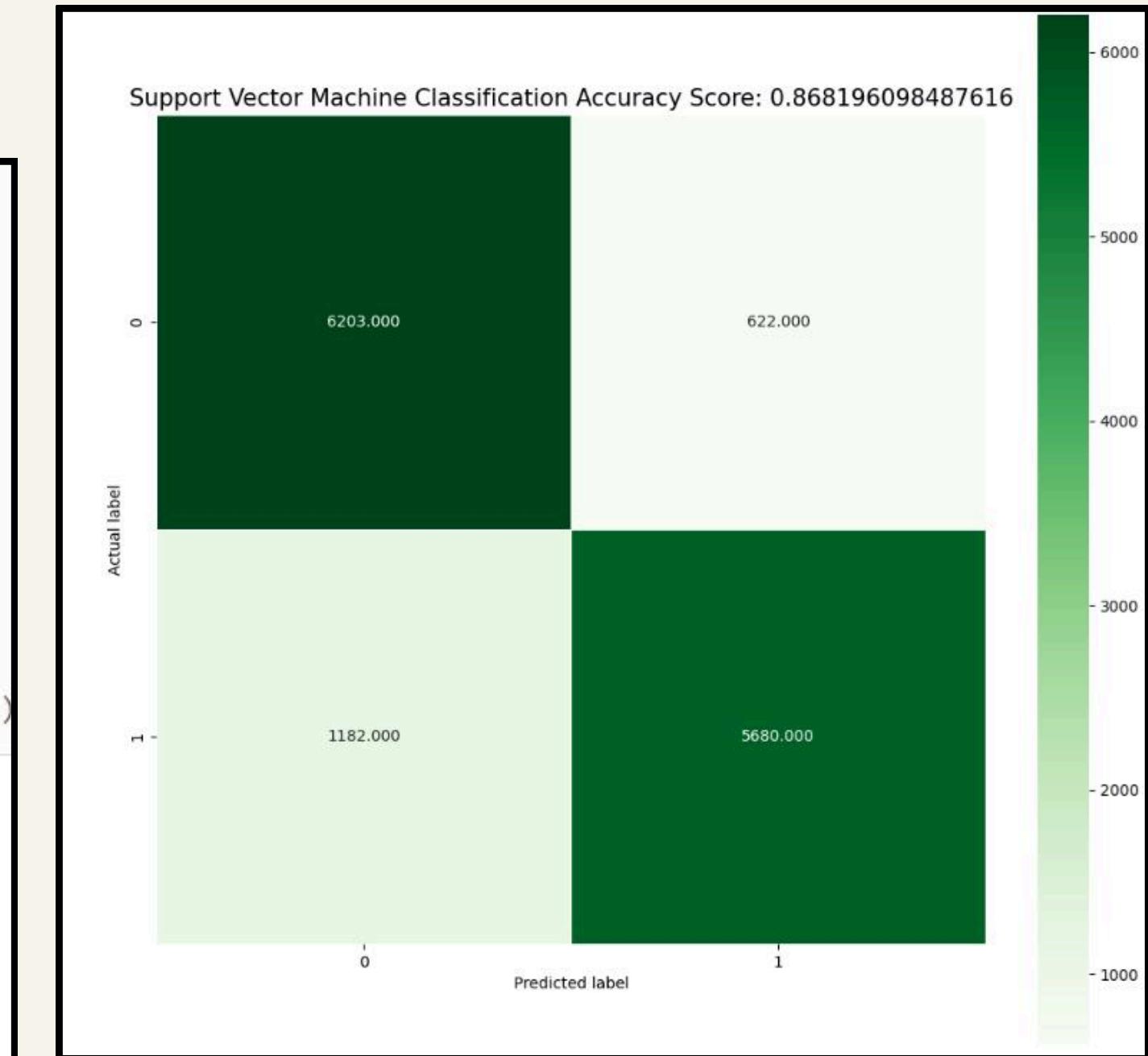
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_svm_pred = svm.predict(X_train)
train_svm_score = metrics.accuracy_score(y_train, train_svm_pred)
print("Support Vector Machine Classification Accuracy on Train Data:", train_svm_score)

print("Support Vector Machine Classification Report:", metrics.classification_report(y_test, svm_pred))

Support Vector Machine Classification Accuracy on Test Data: 0.868196098487616
Support Vector Machine Classification Accuracy on Train Data: 0.8668257872817166
Support Vector Machine Classification Report:
             precision    recall   f1-score   support
0            0.84     0.91     0.87     6825
1            0.90     0.83     0.86     6862

accuracy                           0.87      13687
macro avg                           0.87      0.87      0.87     13687
weighted avg                          0.87      0.87      0.87     13687
```

Report of Support Vector Machine



Confusion Matrix for Support

Vector Machine

# CÁC MÔ HÌNH ĐƯỢC LỰA CHỌN

12

## Random Forest

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu kiểm tra
rfc_pred = rfc.predict(X_test)
rfc_score = metrics.accuracy_score(y_test, rfc_pred)
print("Random Forest Classification Accuracy:", rfc_score)

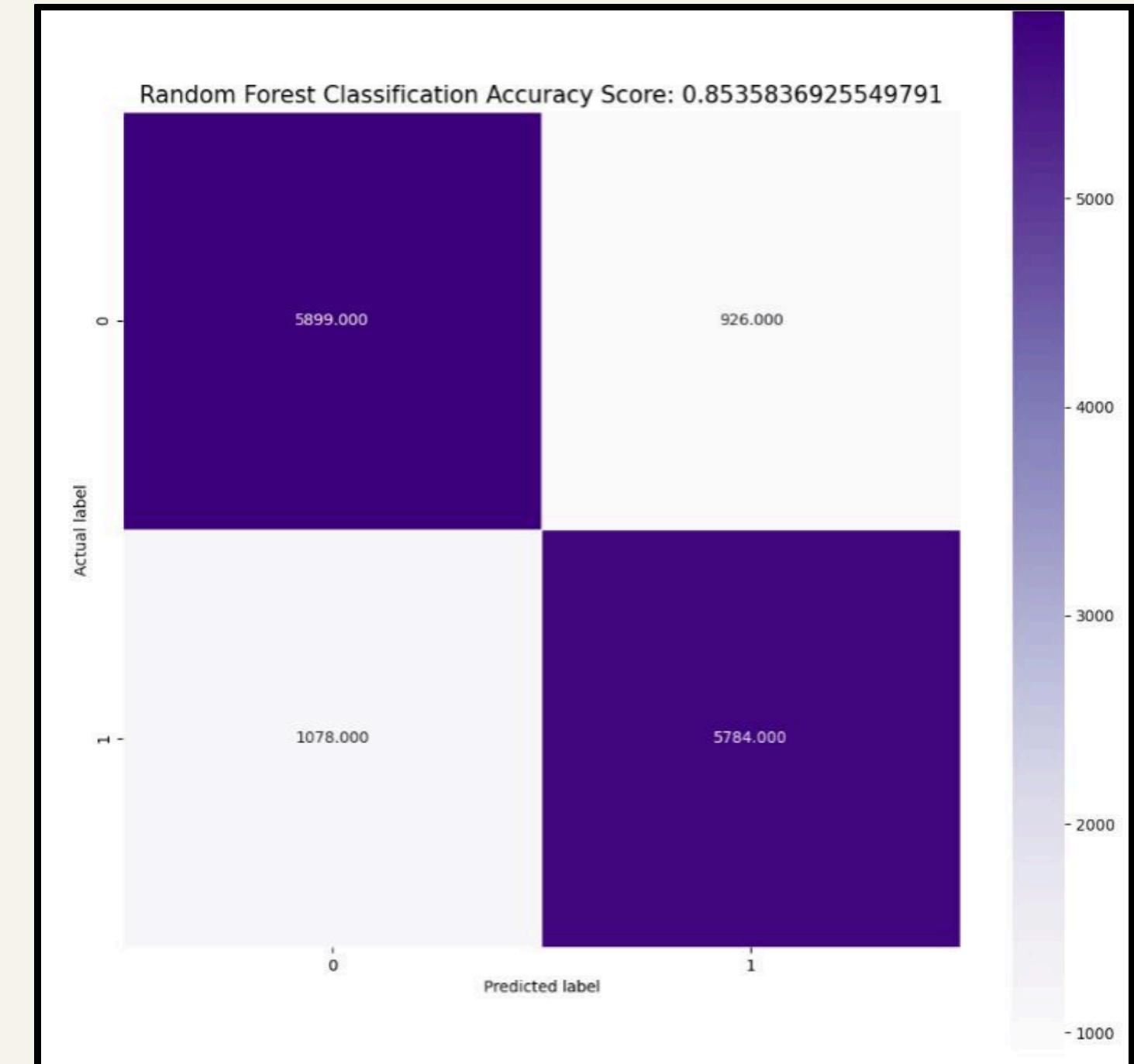
# Tính toán độ chính xác của mô hình dự đoán trên dữ liệu huấn luyện
train_rfc_pred = rfc.predict(X_train)
train_rfc_score = metrics.accuracy_score(y_train, train_rfc_pred)
print("Random Forest Classification Accuracy on Train Data:", train_rfc_score)

print("Random Forest Classification Report:", metrics.classification_report(y_test, rfc_pred))

Random Forest Classification Accuracy: 0.8540951267626214
Random Forest Classification Accuracy on Train Data: 0.9829757178694074
Random Forest Classification Report:
          precision    recall  f1-score   support
0           0.85     0.86     0.86      6825
1           0.86     0.84     0.85      6862

accuracy                           0.85      13687
macro avg       0.85     0.85     0.85      13687
weighted avg    0.85     0.85     0.85      13687
```

Report of Random Forest

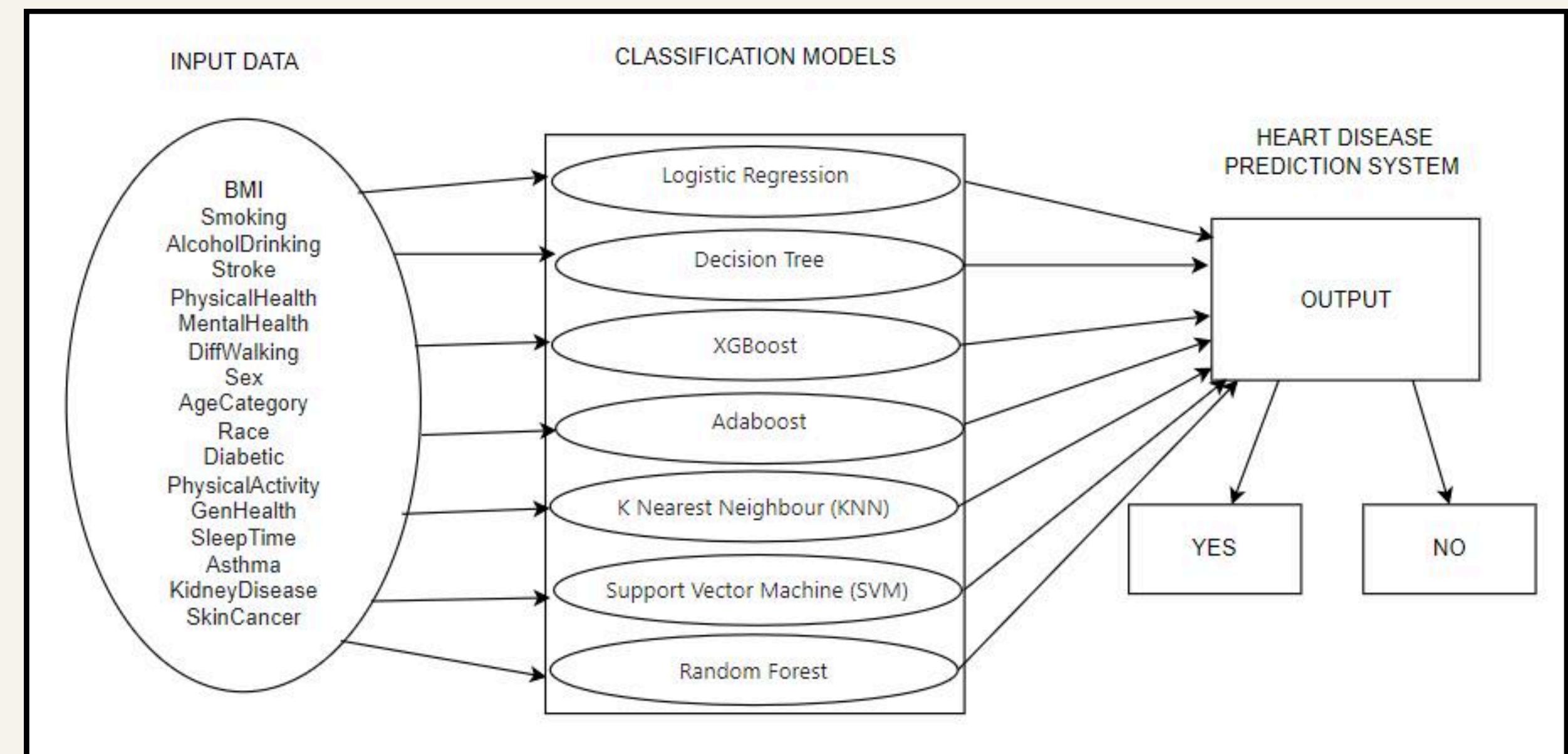


Confusion Matrix for Random Forest

# QUY TRÌNH RA QUYẾT ĐỊNH

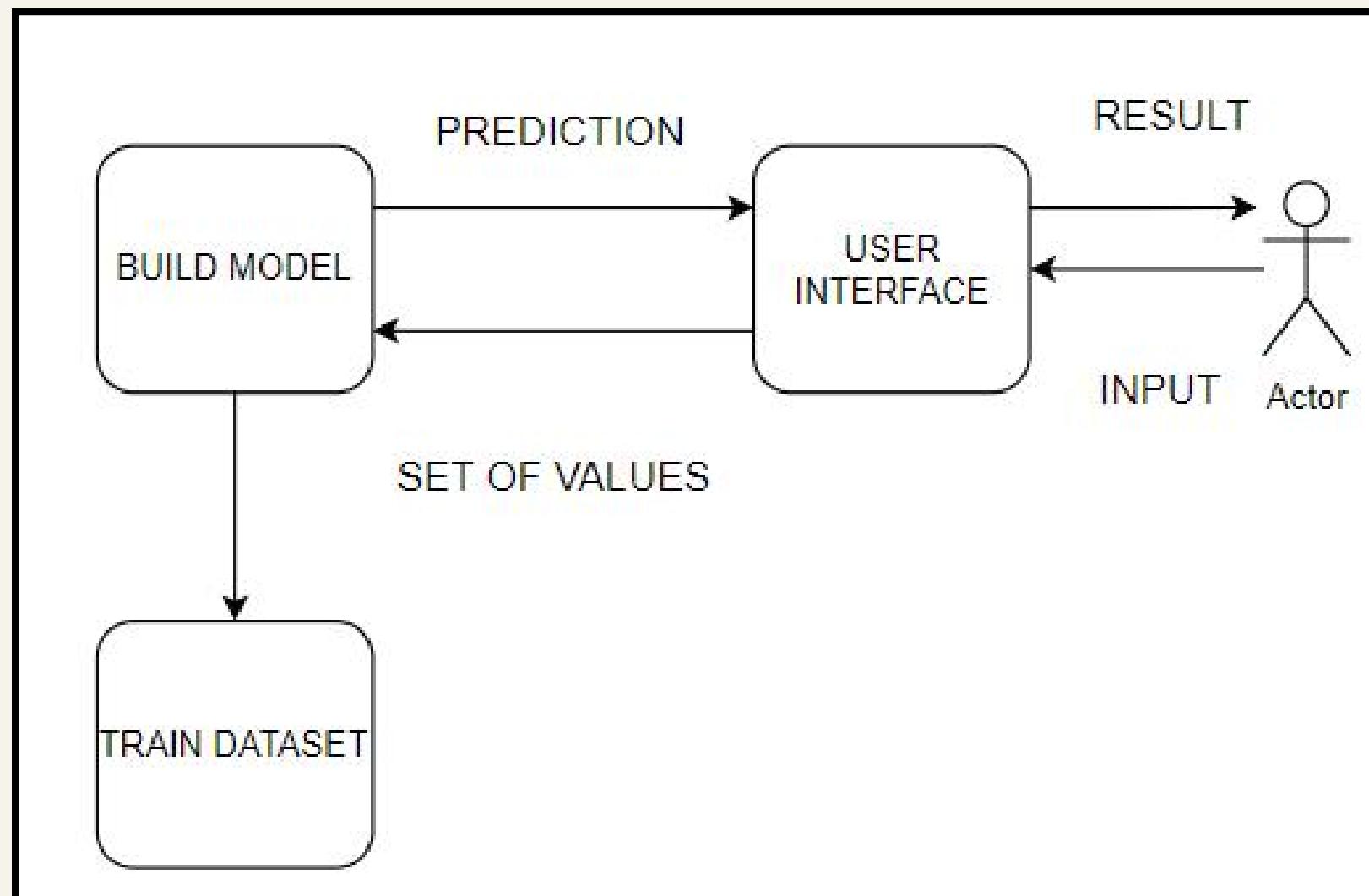
13

- Cách tiếp cận thủ công:  
Bất kỳ người dùng nào  
cũng có thể biết được  
tình trạng tim mạch của  
cá nhân bằng hệ thống  
chẩn đoán thủ công dựa  
trên các thông tin y tế  
nhập vào hệ thống



# QUY TRÌNH RA QUYẾT ĐỊNH

- Sơ đồ kiến trúc



- Quy trình thực hiện:

1. **Bắt đầu:** Người dùng truy cập vào ứng dụng thông qua giao diện web
2. **Nhập thông tin**
3. **Xử lý dữ liệu:** Dữ liệu được nhập sẽ được hệ thống kiểm tra định dạng và xử lý để chuẩn bị cho quá trình dự đoán.
4. **Dự đoán:** Các thuật toán học máy được sử dụng để phân tích dữ liệu đã được xử lý và dự đoán nguy cơ mắc bệnh tim mạch của người dùng.
5. **Hiển thị kết quả:** Kết quả dự đoán được hiển thị cho người dùng mức độ nguy cơ mắc bệnh tim mạch.

# QUY TRÌNH RA QUYẾT ĐỊNH

Ứng dụng trên nền tảng web:

BMI	26.54
Smoking	0
AlcoholDrinking	0
Stroke	0
PhysicalHealth	0
MentalHealth	0

DiffWalking	0
Sex	1
AgeCategory	6
Race	5
Diabetic	0
PhysicalActivity	1

GenHealth	0
SleepTime	7
Asthma	0
KidneyDisease	0
SkinCancer	0

**Clear** **Submit**

output

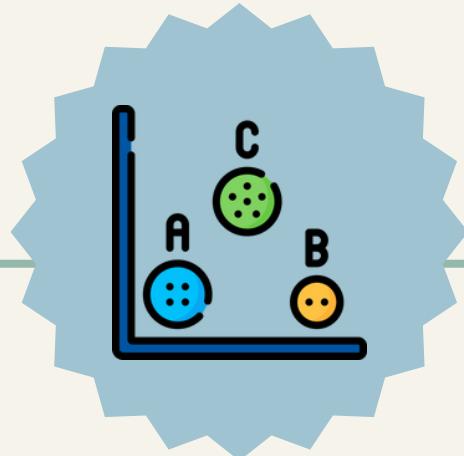
Bạn không bị bệnh tim

**Flag**

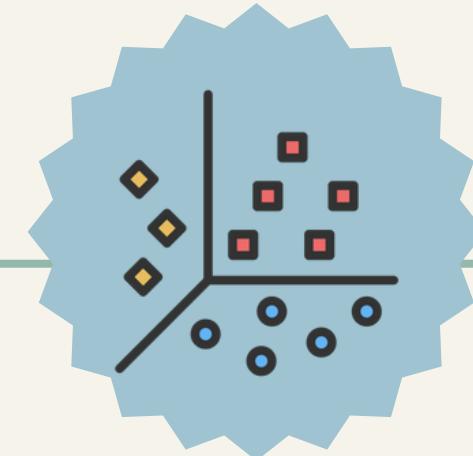
# KẾT QUẢ ĐẠT ĐƯỢC

- Nhóm đã xây dựng mô hình máy học giúp dự đoán nguy cơ mắc bệnh tim mạch với giao diện web thân thiện với người dùng
- Mô hình sử dụng thuật toán Random Forest Classification để phân loại bệnh nhân mắc bệnh tim mạch và không mắc bệnh tim mạch dựa trên các yếu tố nguy cơ
- Mô hình đạt được độ chính xác 98% trong việc dự đoán bệnh tim mạch.
- Ý nghĩa, áp dụng thực tế:
  - Hỗ trợ bác sĩ trong việc chẩn đoán bệnh tim mạch, giúp đưa ra quyết định điều trị chính xác và kịp thời hơn.
  - Screening bệnh tim mạch cho người dân, giúp phát hiện sớm các trường hợp nguy cơ cao mắc bệnh tim mạch để có biện pháp phòng ngừa kịp thời.

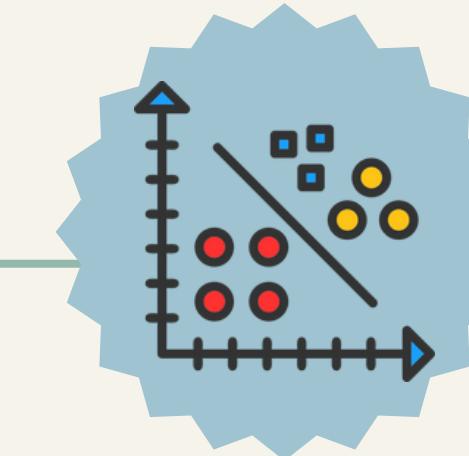
# KẾT LUẬN



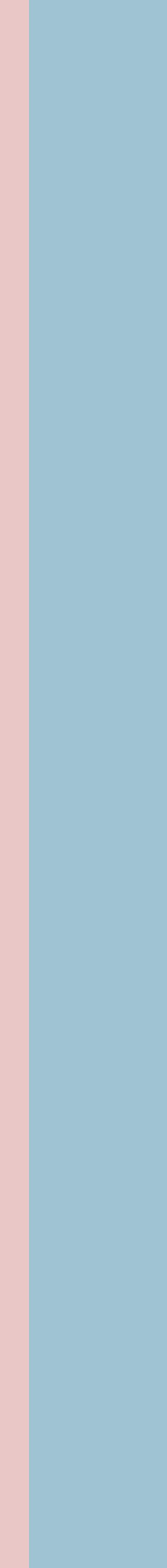
Ưu điểm



Hạn chế



Hướng phát triển



# THANK YOU

