TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO GIỮA KÌ MÔN

HỌC MÁY

Người hướng dẫn: **TS. TRẦN LƯƠNG QUỐC ĐẠI**

Người thực hiện: **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

NGUYỄN QUỐC MẠNH – 52200085

PHẠM THỊ THANH BÌNH – 52200104

Lớp **: 22050201**

Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO GIỮA KỲ MÔN

HỌC MÁY

Người hướng dẫn: **TS. TRẦN LƯƠNG QUỐC ĐẠI**
Người thực hiện: **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**
NGUYỄN QUỐC MẠNH – 52200085
PHẠM THỊ THANH BÌNH – 52200104
Lớp **: 22050201**
Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Để hoàn thành Bài báo cáo giữa kỳ I năm học 2024 - 2025 môn Học máy lần này.

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành của mình đến Ban giám hiệu trường Đại học Tôn Đức Thắng, quý thầy cô giáo giảng viên trong khoa Công nghệ thông tin vì đã tạo điều kiện tốt nhất để có những kiến thức mà hoàn thành bài báo cáo cuối kỳ I lần này. Đây thực sự là một cơ hội tuyệt vời giúp cho nghề nghiệp của em trong tương lai rộng mở hơn khi được tiếp xúc với sự hiện đại, nhiều kiến thức.

Bên cạnh đó, nhóm chúng em cũng nhận được rất nhiều sự giúp đỡ tận tình của quý thầy cô. Thầy cô giảng viên là những người hướng dẫn và truyền cảm hứng cho chúng em trong học tập trong suốt thời gian qua ở môi trường đại học.

Với lòng biết ơn sâu sắc và vô cùng đặc biệt của mình, nhóm chúng em xin gửi lời cảm ơn của mình đến thầy Trần Lương Quốc Đại – Giảng viên lý thuyết Học máy – người đã luôn đồng hành, dẫn dắt và giúp đỡ chúng em trong việc hoàn thành bài báo cáo cuối kỳ. Từ những kiến thức thầy đã giảng dạy trên những giờ học để em có thể áp dụng những kiến thức đó vào bài tập lần này. Một lần nữa nhóm chúng em xin chân thành cảm ơn thầy vì sự hỗ trợ của thầy ạ.

Vì kiến thức của tụi em vẫn còn hạn chế nên trong quá trình giải những quyết vấn đề nên khi hoàn thành bài báo cáo cuối kỳ lần này không tránh khỏi những sai sót, nhóm chúng em kính mong nhận được những lời nhận xét, đóng góp ý kiến từ thầy ạ.

Lời cuối cùng, nhóm chúng em xin gửi lời cảm ơn chân thành và gửi ngàn lời chúc tốt đẹp đến với quý thầy cô khi đã tạo cơ hội cho chúng em nâng cấp kiến thức trong môn học này.

Chúng em xin chân thành cảm ơn ạ!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Trần Lương Quốc Đại;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 14 tháng 11 năm 2024

Tác giả

(ký tên và ghi rõ họ tên)

Huỳnh Hoàng Tiến Đạt

Nguyễn Quốc Mạnh

Phạm Thị Thanh Bình

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Trong thời đại công nghệ 4.0, học máy (Machine Learning) đang trở thành một trong những công cụ quan trọng nhất để phân tích dữ liệu và tạo ra các giải pháp dự đoán thông minh. Báo cáo này sẽ khám phá các thuật toán học máy cơ bản thuộc hai lớp chính là phân loại (Classification) và hồi quy (Regression).

Bài báo cáo Giữa kỳ này bao gồm 04 chương:

Chương 1: Giới thiệu chung

Chương 2: Giải quyết vấn đề thứ nhất

Chương 3: Giải quyết vấn đề thứ hai

Chương 4: Giải quyết vấn đề thứ ba

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

CHƯƠNG 1 – GIỚI THIỆU CHUNG

1.1 Giới thiệu chung về học máy

Học máy (Machine Learning) là một lĩnh vực nghiên cứu của trí tuệ nhân tạo (Artificial Intelligence - AI) chuyên về phát triển các thuật toán và mô hình cho phép máy tính có thể tự động học hỏi và đưa ra dự đoán dựa trên dữ liệu, thay vì phải lập trình chi tiết từng quy tắc.

Ứng dụng của học máy hiện nay rất rộng lớn, trải dài từ các lĩnh vực kinh tế, y tế, giáo dục đến công nghệ và khoa học xã hội. Với những ứng dụng đa dạng và phong phú như vậy, học máy không chỉ là một công cụ hữu ích mà còn trở thành một yếu tố quan trọng trong phát triển công nghệ và thay đổi xã hội.

Các phương pháp học máy phổ biến bao gồm:

Học có giám sát (Supervised Learning): Dựa vào các dữ liệu có nhãn (labeled data), với mục tiêu tìm ra mối quan hệ giữa các biến đầu vào và đầu ra. Các bài toán phân loại và hồi quy thường thuộc loại này.

Học không giám sát (Unsupervised Learning): Áp dụng cho các dữ liệu chưa có nhãn, với mục tiêu tìm ra các cấu trúc tiềm ẩn hoặc nhóm trong dữ liệu, chẳng hạn như phân cụm khách hàng (customer segmentation).

Học tăng cường (Reinforcement Learning): Dựa trên việc học từ môi trường thông qua cơ chế phần thưởng/phạt để tối ưu hóa một hành động nào đó.

1.2 Giới thiệu về bài toán Phân loại, Hồi quy

1.2.1 Bài toán Phân loại (Classification)

Phân loại là kỹ thuật được sử dụng khi bài toán yêu cầu chia dữ liệu vào các nhóm hoặc lớp nhất định, tức là đầu ra là các giá trị rời rạc (discrete values). Ví dụ điển hình là phân loại email thành "spam" và "không spam", hoặc phân loại hình ảnh thành các nhóm như "mèo", "chó", và "con người."

Một số thuật toán phân loại phổ biến bao gồm:

K-Nearest Neighbors (KNN): Dựa trên khoảng cách giữa các điểm dữ liệu để dự đoán lớp cho một điểm mới bằng cách xem xét lớp của các điểm lân cận.

Decision Trees: Xây dựng các cây quyết định để phân loại dựa trên các điều kiện khác nhau trong dữ liệu.

Naive Bayes: Sử dụng lý thuyết xác suất Bayes để phân loại dựa trên các giả định độc lập giữa các thuộc tính.

Phân loại thường áp dụng cho các bài toán nhận dạng mẫu, như nhận dạng khuôn mặt, nhận diện giọng nói, và phát hiện gian lận. Các kết quả phân loại có thể được đo lường bằng các chỉ số như độ chính xác (accuracy), F1-score, và mức độ nhạy (recall).

1.2.2 Bài toán Hồi quy (Regression)

Hồi quy được sử dụng cho các bài toán dự đoán mà đầu ra là một giá trị liên tục (continuous value). Thay vì chia dữ liệu vào các nhóm, hồi quy dự đoán một giá trị số cụ thể dựa trên các thuộc tính đầu vào. Chẳng hạn, dự báo giá nhà dựa trên diện tích, số lượng phòng, vị trí là một bài toán hồi quy.

Các thuật toán hồi quy phổ biến bao gồm:

Linear Regression (Hồi quy tuyến tính): Xây dựng một mô hình tuyến tính để dự đoán giá trị đầu ra dựa trên một hoặc nhiều biến đầu vào.

Polynomial Regression (Hồi quy đa thức): Mở rộng hồi quy tuyến tính thành mô hình đa thức để mô hình hóa các mối quan hệ phi tuyến tính.

Support Vector Regression (SVR): Một biến thể của thuật toán SVM, được sử dụng cho dự đoán giá trị liên tục.

Hồi quy cung cấp các mô hình để dự đoán giá trị liên tục và thường được đánh giá bằng các chỉ số như Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), và R-squared (R^2).

CHƯƠNG 2 – VẤN ĐỀ THỨ NHẤT

Tổng quan: Bệnh tim mạch là một trong những nguyên nhân hàng đầu gây tử vong trên toàn thế giới, việc phát hiện sớm và điều trị kịp thời bệnh tim là vô cùng quan trọng. Bộ dữ liệu Heart Disease Dataset cung cấp thông tin chi tiết về nhiều bệnh nhân, bao gồm các đặc điểm sinh lý và lâm sàng liên quan đến tình trạng sức khỏe của họ.

Trong bài báo cáo này, chúng em sẽ sử dụng Heart Disease Dataset để thực hiện hai loại bài toán chính: phân loại và hồi quy. Cụ thể, chúng em sẽ phân loại khả năng mắc bệnh tim dựa trên các yếu tố nguy cơ và thực hiện hồi quy để dự đoán các chỉ số sức khỏe tổng quát của bệnh nhân;

2.1 Thống kê sơ lược về Dataset

Đường dẫn của Dataset: https://archive.ics.uci.edu/dataset/45/heart+disease

Dataset Name: Heart Disease

Dataset Characteristics: Multivariate.

Subject Area: Health and Medicine

Associated Tasks: Classification

Feature Type: Categorical, Integer, Real

Instances: 303

Features: 13

Biến mục tiêu: là một giá trị nhị phân cho biết sự hiện diện (1) hoặc không (0) của bệnh tim.

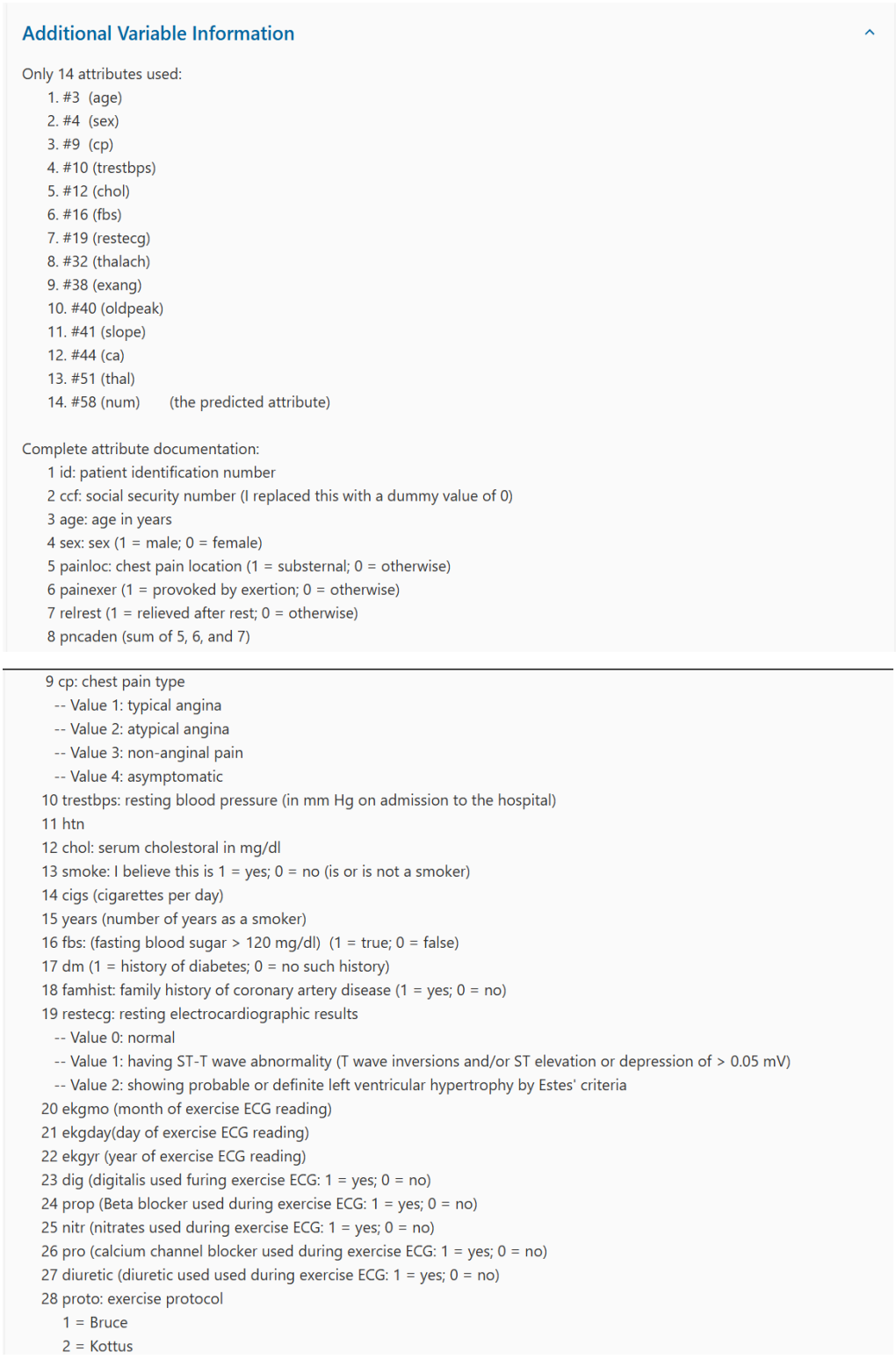
Variables Table

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
age	Feature	Integer	Age		years	no
sex	Feature	Categorical	Sex			no
cp	Feature	Categorical				no
trestbps	Feature	Integer		resting blood pressure (on admission to the hospital)	mm Hg	no
chol	Feature	Integer		serum cholestoral	mg/dl	no
fbs	Feature	Categorical		fasting blood sugar > 120 mg/dl		no
restecg	Feature	Categorical				no
thalach	Feature	Integer		maximum heart rate achieved		no
exang	Feature	Categorical		exercise induced angina		no
oldpeak	Feature	Integer		ST depression induced by exercise relative to rest		no

Variables Table

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
slope	Feature	Categorical				no
ca	Feature	Integer		number of major vessels (0-3) colored by flourosopy		yes
thal	Feature	Categorical				yes
num	Target	Integer		diagnosis of heart disease		no

Hình 2. 1 Bảng biến của tập dữ liệu.



Additional Variable Information

Only 14 attributes used:

1. #3 (age)
2. #4 (sex)
3. #9 (cp)
4. #10 (trestbps)
5. #12 (chol)
6. #16 (fbs)
7. #19 (restecg)
8. #32 (thalach)
9. #38 (exang)
10. #40 (oldpeak)
11. #41 (slope)
12. #44 (ca)
13. #51 (thal)
14. #58 (num) (the predicted attribute)

Complete attribute documentation:

1 id: patient identification number
2 ccf: social security number (I replaced this with a dummy value of 0)
3 age: age in years
4 sex: sex (1 = male; 0 = female)
5 painloc: chest pain location (1 = substernal; 0 = otherwise)
6 painexer (1 = provoked by exertion; 0 = otherwise)
7 relrest (1 = relieved after rest; 0 = otherwise)
8 pncaden (sum of 5, 6, and 7)

---

9 cp: chest pain type
-- Value 1: typical angina
-- Value 2: atypical angina
-- Value 3: non-anginal pain
-- Value 4: asymptomatic
10 trestbps: resting blood pressure (in mm Hg on admission to the hospital)
11 htn
12 chol: serum cholestoral in mg/dl
13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)
14 cigs (cigarettes per day)
15 years (number of years as a smoker)
16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
17 dm (1 = history of diabetes; 0 = no such history)
18 famhist: family history of coronary artery disease (1 = yes; 0 = no)
19 restecg: resting electrocardiographic results
-- Value 0: normal
-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
-- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
20 ekgmo (month of exercise ECG reading)
21 ekgday(day of exercise ECG reading)
22 ekgyr (year of exercise ECG reading)
23 dig (digitalis used furing exercise ECG: 1 = yes; 0 = no)
24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)
25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)
26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)
27 diuretic (diuretic used used during exercise ECG: 1 = yes; 0 = no)
28 proto: exercise protocol
1 = Bruce
2 = Kottus

Hình 2. 2 Thông tin một số biến bổ sung

2.2 Thống kê đặc điểm của dữ liệu bằng Data visualization

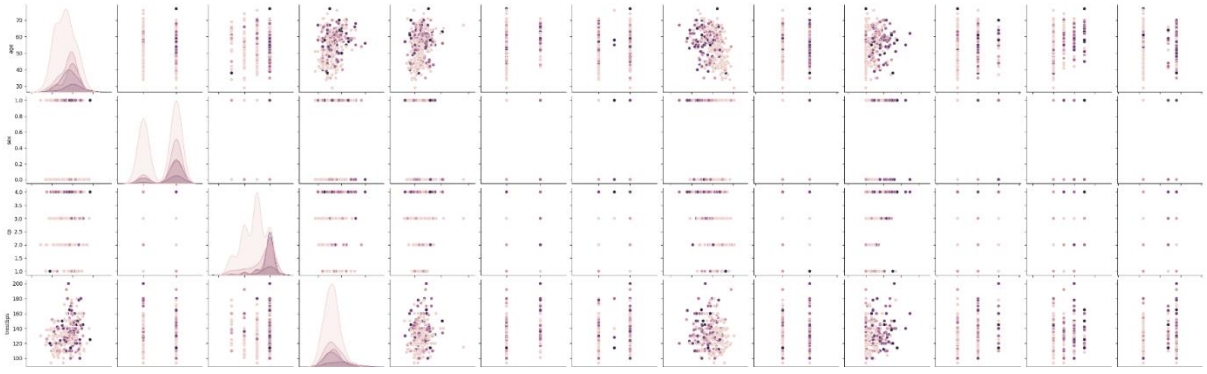
Trực quan hóa dữ liệu (là các variables của tập dữ liệu) bằng biểu đồ cột có thể hiểu sâu hơn về các đặc điểm, sự phân bố và xu hướng trong dữ liệu.

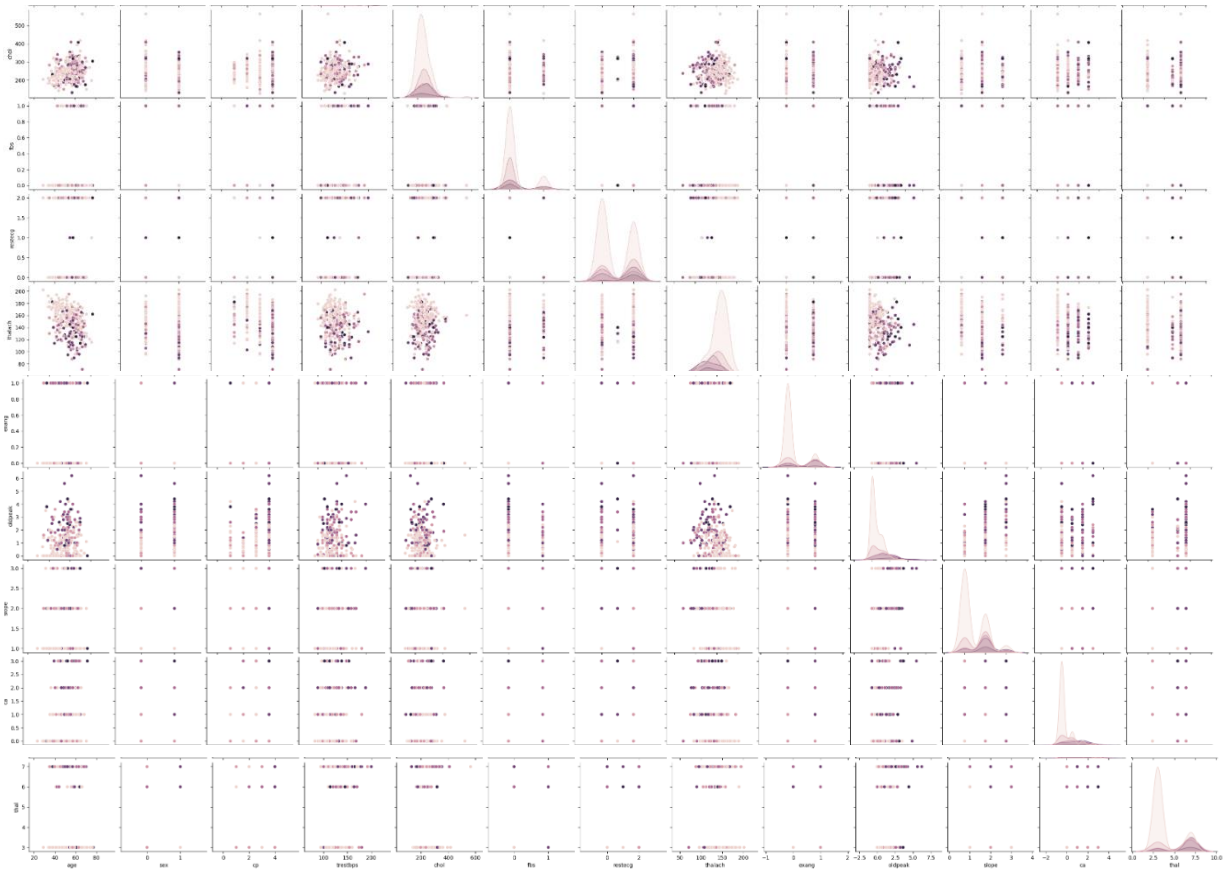
```
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        299 non-null    float64
 12  thal      301 non-null    float64
 13  num       303 non-null    int64
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
None
```

	age	sex	cp	trestbps	chol	fbs
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000

	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	299.000000
mean	0.990099	149.607261	0.326733	1.039604	1.600660	0.672241
std	0.994971	22.875003	0.469794	1.161075	0.616226	0.937438
min	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000
75%	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000

	thal	num
count	301.000000	303.000000
mean	4.734219	0.937294
std	1.939706	1.228536
min	3.000000	0.000000
25%	3.000000	0.000000
50%	3.000000	0.000000
75%	7.000000	2.000000
max	7.000000	4.000000





Hình 2. 3 Thống kê đặc điểm của dữ liệu bằng Data visualization

2.3 Tiền xử lý dữ liệu: type conversation và data normalization

Đọc dữ liệu UCI

```
url = 'https://archive.ics.uci.edu/static/public/45/data.csv'
df = pd.read_csv(url)
```

2.3.1 Định nghĩa biến Mục tiêu

Biến mục tiêu là thông tin cần dự đoán. Ở đây, ta gán cột `num` từ DataFrame vào biến `y` làm biến mục tiêu. Với giá trị 1 thể hiện sự hiện diện của bệnh tim và 0 là không có bệnh tim.

```
# Bước 2: Định nghĩa Biến Mục Tiêu
y = df['num']
```

2.3.2 Tạo tập đặc trưng

Để tạo tập đặc trưng `X`, ta loại bỏ cột `num` khỏi DataFrame gốc. Điều này giúp đảm bảo rằng dữ liệu huấn luyện không chứa biến mục tiêu, tránh ảnh hưởng đến tính chính xác của mô hình.

```
# Bước 3: Tạo Tập Đặc Trưng
X = df.drop(columns=['num'])
```

2.3.3 Dọn dẹp dữ liệu

Xử lý bị thiếu (NaN) trong DataFrame: `X.mean()` tính trung bình cho mỗi cột của `X`, sau đó `X.fillna()` sẽ điền những giá trị thiếu trong từng cột bằng giá trị trung bình của cột đó.

```
X = X.fillna(X.mean())
y = y[X.index]
```

2.3.4 Chuyển đổi biến

Định nghĩa danh sách các cột nhị phân và cột nhãn (categorical columns): Các cột nhị phân là những cột có giá trị 0 và 1 (như 'fbs', 'exang', 'sex'). Các cột nhãn là các biến phân loại có nhiều loại (như 'cp', 'slope', 'thal', 'restecg').

Mã hóa các biến nhị phân bằng LabelEncoder nhằm chuyển đổi các biến nhị phân này thành các giá trị số (0 hoặc 1).

```
# Bước 6: Chuyển Đổi Biến
binary_cols = ['fbs', 'exang', 'sex']
le = LabelEncoder()
for col in binary_cols:
    X[col] = le.fit_transform(X[col])
```

Mã hóa các biến phân loại nhiều loại bằng one-hot encoding: Áp dụng one-hot encoding cho các cột nhãn để tạo các cột nhị phân tương ứng với từng giá trị trong các biến phân loại. Phương pháp này giúp giảm thiểu thông tin bị mất khi chuyển đổi các biến phân loại.

```
categorical_cols = ['cp', 'slope', 'thal', 'restecg']
X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)
```

2.3.5 Chuẩn hóa các biến số

Để đảm bảo các biến số được chuẩn hóa, ta định nghĩa danh sách các cột số cần chuẩn hóa (như 'age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca'). Sau đó, sử dụng StandardScaler để đưa các cột này về dạng có trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp tăng độ chính xác của các mô hình.

```
numeric_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca']
scaler = StandardScaler()
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])
```

2.4 Phân chia dữ liệu thành tập train và tập evaluation

Sau khi tiền xử lý, dữ liệu được chia thành tập huấn luyện và tập kiểm tra với tỷ lệ 80-20, nghĩa là 20% dữ liệu sẽ được dùng cho tập kiểm tra, và phần còn lại sẽ dùng để huấn luyện mô hình.

```
# Bước 7: Chia Dữ Liệu Thành Tập Huấn Luyện và Kiểm Tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Việc sử dụng `train_test_split` đảm bảo rằng mô hình được kiểm tra trên dữ liệu chưa từng thấy, tăng độ tin cậy trong việc đánh giá hiệu suất mô hình.

In ra màn hình:

```
Kích thước tập huấn luyện: (237, 18)
Kích thước tập kiểm tra: (60, 18)
```

2.5 Thực hiện phân loại (và hồi quy)

2.5.1 Phân loại

Các chỉ số đánh giá:

Accuracy: Tỷ lệ dự đoán đúng.

Precision: Độ chính xác (trung bình macro cho các lớp).

Recall: Khả năng phát hiện đúng các mẫu dương tính.

F1-Score: Trung bình điều hòa giữa precision và recall.

2.5.1.1 KNN (K-Nearest Neighbors)

Mô hình k-NN là phương pháp phân loại dựa trên việc tìm kiếm k điểm dữ liệu gần nhất với điểm cần phân loại. Dựa trên nhãn của các điểm gần nhất, mô hình xác định nhãn cho điểm cần phân loại.

```
# 1. K-Nearest Neighbors (KNN)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_predictions = knn_model.predict(X_test)

knn_accuracy = accuracy_score(y_test, knn_predictions)
knn_precision = precision_score(y_test, knn_predictions, average='macro', zero_division=0)
knn_recall = recall_score(y_test, knn_predictions, average='macro')
knn_f1 = f1_score(y_test, knn_predictions, average='macro')
```

Đầu tiên, ta tạo một mô hình K-NN với 5 láng giềng gần nhất và huấn luyện mô hình trên tập huấn luyện `X_train` và `y_train`. Sau đó, dự đoán trên tập kiểm tra `X_test` và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
KNN Results:
Accuracy: 0.45901639344262296
Precision: 0.20794573643410855
Recall: 0.24622331691297208
F1-Score: 0.22042735042735045
```

2.5.1.2 Support Vector Machine (SVM)

SVM là một mô hình phân loại tìm kiếm hyperplane tối ưu phân chia các lớp trong không gian đặc trưng, nhằm tối đa hóa khoảng cách giữa các lớp. Mô hình này hiệu quả trong các trường hợp không gian đặc trưng có chiều cao.

```
# 2. Support Vector Machine (SVM)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)

svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_precision = precision_score(y_test, svm_predictions, average='macro', zero_division=0)
svm_recall = recall_score(y_test, svm_predictions, average='macro')
svm_f1 = f1_score(y_test, svm_predictions, average='macro')
```

Đầu tiên, ta tạo mô hình SVM với hàm nhân tuyến tính và huấn luyện mô hình trên dữ liệu `X_train` và `y_train`. Sau đó, dự đoán trên tập kiểm tra `X_test` và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
SVM Results:
Accuracy: 0.47540983606557374
Precision: 0.2346236559139785
Recall: 0.28035030103995623
F1-Score: 0.24787878787878787
```

2.5.1.3 Naive Bayes

Mô hình Naive Bayes sử dụng định lý Bayes, với giả định các đặc trưng là độc lập nhau. Đây là mô hình phân loại có khả năng làm việc tốt với các bài toán có số lượng lớn các đặc trưng rời rạc.

```
# 3. Naive Bayes
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)
naive_bayes_predictions = naive_bayes_model.predict(X_test)

naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions)
naive_bayes_precision = precision_score(y_test, naive_bayes_predictions, average='macro', zero_division=0)
naive_bayes_recall = recall_score(y_test, naive_bayes_predictions, average='macro')
naive_bayes_f1 = f1_score(y_test, naive_bayes_predictions, average='macro')
```

Đầu tiên, ta tạo mô hình Gaussian Naive Bayes và huấn luyện mô hình trên dữ liệu `X_train` và `y_train`. Sau đó, dự đoán trên tập kiểm tra `X_test` và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
Naive Bayes Results:
Accuracy: 0.11475409836065574
Precision: 0.2868253968253968
Recall: 0.13814997263273124
F1-Score: 0.12047619047619047
```

2.5.2 Hồi quy

Các chỉ số được tính toán trong hàm `evaluate_regression()`

```
def evaluate_regression(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, mae, r2
```

MSE (Mean Squared Error): Sai số bình phương trung bình giữa giá trị dự đoán và giá trị thực.

MAE (Mean Absolute Error): Sai số tuyệt đối trung bình giữa giá trị dự đoán và giá trị thực.

R^2 Score: Đo lường mức độ mô hình giải thích được biến phụ thuộc, chỉ số càng gần 1 thì mô hình càng tốt.

2.5.2.1 Linear Regression (Hồi quy tuyến tính)

Linear Regression là một mô hình học máy loại hồi quy, được sử dụng để dự đoán giá trị liên tục dựa trên một hoặc nhiều đặc trưng. Mô hình này tìm kiếm mối quan hệ tuyến tính giữa biến độc lập (input) và biến phụ thuộc (output), giúp dự đoán giá trị mục tiêu.

```
# 1. Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
linear_predictions = linear_model.predict(X_test)

linear_mse, linear_mae, linear_r2 = evaluate_regression(y_test, linear_predictions)
```

Ta tạo mô hình hồi quy tuyến tính và huấn luyện mô hình trên dữ liệu `X_train` và `y_train`. Sau đó, dự đoán trên tập kiểm tra `X_test` và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
Linear Regression Results:
MSE: 0.8315287536743805
MAE: 0.6444303389752595
R^2 Score: 0.4744551416383832
```

2.5.2.2 Random Forest Regressor (Hồi quy rừng ngẫu nhiên)

Random Forest Regressor là một thuật toán hồi quy sử dụng tập hợp các cây quyết định (ensemble method). Bằng cách kết hợp nhiều cây quyết định, mô hình này giúp giảm thiểu hiện tượng overfitting, cải thiện độ chính xác và tính ổn định của dự đoán.

```
# 3. Random Forest Regressor
random_forest_model = RandomForestRegressor(random_state=42)
random_forest_model.fit(X_train, y_train)
random_forest_predictions = random_forest_model.predict(X_test)

rf_mse, rf_mae, rf_r2 = evaluate_regression(y_test, random_forest_predictions)
```

Ta tạo mô hình hồi quy tuyến tính rừng ngẫu nhiên và cố định random_state để đảm bảo kết quả tái lập được, huấn luyện mô hình trên dữ liệu X_train và y_train. Sau đó, dự đoán trên tập kiểm tra X_test và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
Random Forest Regressor Results:
MSE: 0.8901966666666666
MAE: 0.657
R^2 Score: 0.43737570224719113
```

2.5.2.3 K-Nearest Neighbors Regressor (Hồi quy K-Nearest Neighbors)

K-Nearest Neighbors (KNN) Regressor dựa trên việc tìm kiếm k điểm dữ liệu gần nhất với điểm cần dự đoán. Giá trị dự đoán cho điểm này được xác định bằng cách tính

```
# 4. K-Neares
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_predictions = knn_model.predict(X_test)

knn_mse, knn_mae, knn_r2 = evaluate_regression(y_test, knn_predictions)
```

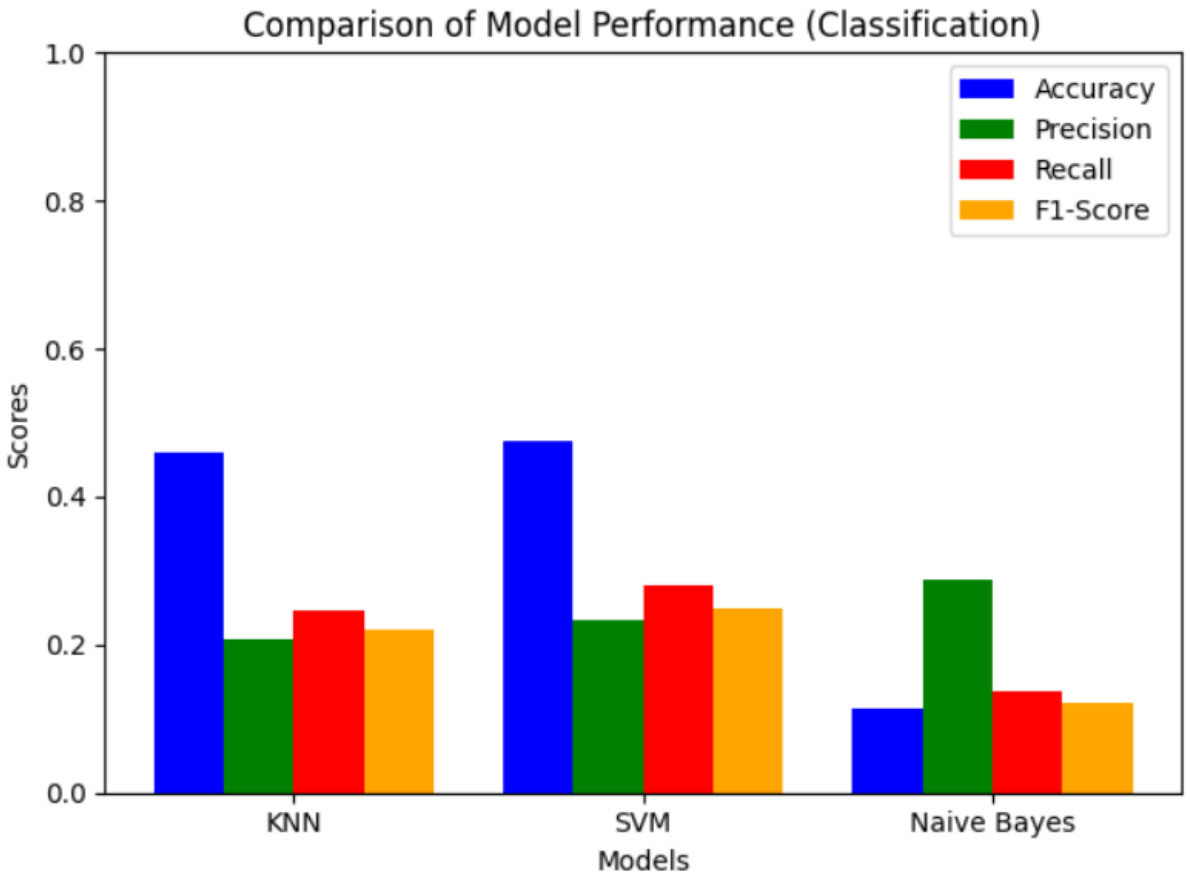
trung bình của các giá trị mục tiêu của các điểm gần nhất. KNN không yêu cầu quá trình huấn luyện, nhưng độ chính xác phụ thuộc vào lựa chọn k và không gian đặc trưng.

Ta tạo mô hình hồi quy K-láng giềng gần nhất với 5 láng giềng và huấn luyện mô hình trên dữ liệu `X_train` và `y_train`. Sau đó, dự đoán trên tập kiểm tra `X_test` và tính toán các chỉ số, ta được kết quả in ra màn hình:

```
K-Nearest Neighbors Regressor Results:
MSE: 1.204666666666667
MAE: 0.7300000000000001
R^2 Score: 0.23862359550561796
```

2.6 Đánh giá kết quả và so sánh các phương pháp

2.6.1 Giữa các thuật toán Phân loại



Hình 2. 4 Biểu đồ so sánh giữa các thuật toán Phân loại

Độ chính xác tổng thể (Accuracy): Mô hình KNN và Support Vector Machine (SVM) đạt được độ chính xác tương đối cao, cho thấy cả hai mô hình này có khả năng dự đoán khá tốt trên tổng thể. Trong khi đó, mô hình Naive Bayes có độ chính xác thấp hơn rất nhiều, cho thấy nó không phân loại tổng thể tốt bằng các mô hình khác.

Độ chính xác của dự đoán (Precision): SVM và KNN ở mức thấp hơn Naïve Bayes, cho thấy cả hai mô hình có khả năng phân loại đúng các trường hợp dương, nhưng không quá chắc chắn. Precision của Naive Bayes cao hơn hẳn so với các chỉ số khác của

nó, nghĩa là khi Naive Bayes dự đoán một trường hợp là đúng, thì khả năng cao nó sẽ chính xác.

Độ nhạy (Recall): SVM và KNN có chỉ số Recall cao hơn cho thấy khả năng phát hiện các trường hợp đúng cao. Naive Bayes có Recall thấp nhất, cho thấy khả năng phát hiện các trường hợp đúng thấp hơn, dẫn đến bỏ sót một số trường hợp.

F1-Score: Điểm F1 cho tất cả các mô hình cho thấy sự đánh đổi giữa độ chính xác và độ nhạy. KNN và SVM có F1-Score trung bình, phù hợp với mức Precision và Recall của chúng. Naive Bayes có F1-Score thấp nhất, cho thấy sự không đồng đều giữa Precision và Recall, có thể không phải là lựa chọn tốt nếu cần một sự cân bằng.

Kết luận: Các mô hình có hiệu suất khác nhau, với KNN và Support Vector Machine cho thấy khả năng dự đoán tương đối tốt, trong khi Naive Bayes cần cải thiện về độ chính xác và độ nhạy.

2.6.2 Giữa các thuật toán Hồi quy

MSE (Mean Squared Error):

Hồi quy tuyến tính có MSE là 0.8315 thấp nhất, cho thấy độ lệch giữa giá trị dự đoán và giá trị thực tế trung bình là khá thấp, phản ánh mức độ chính xác tương đối trong việc dự đoán.

Hình 2. 5 Biểu đồ so sánh giữa các thuật toán Hồi quy

Rừng ngẫu nhiên có MSE là 0.8902, xấp xỉ với hồi quy tuyến tính.

K-Nearest Neighbors có MSE cao nhất là 1.2047, cho thấy sai số bình phương trung bình của nó lớn..

MAE (Mean Absolute Error):

Hồi quy tuyến tính: MAE là 0.6444, độ lệch tuyệt đối trung bình giữa giá trị dự đoán và giá trị thực tế khá nhỏ, cho thấy mô hình có độ chính xác chấp nhận được.

Rừng ngẫu nhiên: MAE là 0.657, khá gần với hồi quy tuyến tính, cho thấy mô hình này có độ chính xác tốt và ổn định.

K-Nearest Neighbors (KNN) có MAE cũng khá cao là 0.7300, cao hơn hồi quy tuyến tính, cho thấy các sai số dự đoán của mô hình khá lớn nên có thể cần phải cải thiện.

R^2 Score (Coefficient of Determination):

Hồi quy tuyến tính: R^2 là 0.4745, cho thấy mô hình có khả năng giải thích được phần lớn biến thiên trong dữ liệu..

Rừng ngẫu nhiên (Random Forest Regressor): R^2 là 0.4374, gần tương tự như hồi quy tuyến tính, cho thấy mô hình có thể giải thích khá tốt biến thiên trong dữ liệu

K-Nearest Neighbors (KNN): R^2 là 0.2386, chỉ ra rằng mô hình giải thích được một phần nhỏ sự biến thiên của dữ liệu, phản ánh độ chính xác chưa cao.

Kết luận: Các mô hình hồi quy có mức độ chính xác khác nhau trong việc dự đoán, với hồi quy tuyến tính và rừng ngẫu nhiên cho thấy độ chính xác tương đối tốt, trong khi KNN có hiệu quả thấp hơn một chút so với các mô hình còn lại.

CHƯƠNG 3 – VẤN ĐỀ THỨ HAI

3.1 Overfitting

3.1.1 Tìm hiểu chung

Trong học máy, mục tiêu là tạo ra một mô hình có khả năng dự đoán những dữ liệu mới, chưa được tiếp cận trước đó, nhưng sẽ có các đặc tính tương tự như tập dữ liệu huấn luyện. Khả năng này được gọi là tổng quát hóa. Một mô hình được coi là có khả năng tổng quát hóa khi nó có thể dự đoán chính xác các kết quả trên dữ liệu mà nó chưa từng gặp.

Tuy nhiên, trong quá trình huấn luyện mô hình có thể có nguy cơ xảy ra quá khớp (overfitting). Quá khớp xảy ra khi mô hình học quá sâu vào dữ liệu huấn luyện, dẫn đến việc không thể tổng quát hóa cho dữ liệu mới. Điều này thường xảy ra khi mô hình quá phức tạp so với số lượng dữ liệu có sẵn, hoặc khi dữ liệu huấn luyện không đại diện cho thực tế.

Ví dụ, ta đang dạy một chiếc máy tính nhận diện các loại động vật khác nhau. Ta cho máy tính xem một bức ảnh của một con mèo, và nó học cách nhận diện con mèo. Sau đó, tiếp tục máy tính xem một bức ảnh của một con chó, và nó học cách nhận diện con chó. Tuy nhiên, nếu bạn chỉ cho máy tính xem những bức ảnh của các động vật đang ngồi, nó sẽ chỉ có khả năng nhận diện các động vật trong tư thế ngồi. Nếu bạn sau đó cho nó xem một bức ảnh của một con vật đứng, nó sẽ không thể nhận diện được. Điều này xảy ra vì máy tính đã quá khớp với dữ liệu huấn luyện và không thể tổng quát hóa cho dữ liệu mới.

Việc nhận thức về quá khớp là rất quan trọng khi huấn luyện các mô hình, bởi vậy. có nhiều kỹ thuật có thể được áp dụng để ngăn chặn hiện tượng này, như regularization và phương pháp kiểm tra chéo (cross-validation).

3.1.2 Giải pháp chung để ngăn ngừa Overfitting

K-fold Cross-Validation

Một trong những thách thức chính của quá khớp là khả năng không tổng quát hóa với các tập dữ liệu mới. Để khắc phục điều này, ta tiến hành phân chia tập dữ liệu nhằm phân tích hiệu suất của mô hình trên từng tập dữ liệu và xác định các trường hợp quá khớp trong quá trình huấn luyện.

Trong phương pháp này, tập dữ liệu được chia thành k phần bằng nhau, gọi là "folds." Thuật toán được huấn luyện lặp đi lặp lại trên k-1 folds trong khi sử dụng fold còn lại làm tập kiểm tra. Sau tất cả các lần lặp, điểm số sẽ được trung bình hóa để đánh giá hiệu suất tổng thể của mô hình.

Tăng kích thước tập dữ liệu: Với một tập dữ liệu lớn hơn, mô hình có thể học được nhiều mẫu và biến thể khác nhau trong hành vi của khách hàng, từ đó làm giảm nguy cơ quá khớp.

Chọn đặc trưng: Giúp đảm bảo rằng chỉ các đặc trưng thông tin và liên quan nhất được sử dụng trong mô hình, tránh đặc trưng thừa thãi hoặc không cần thiết dẫn đến mô hình phức tạp và quá khớp.

Đơn giản hóa dữ liệu: Các phương pháp đơn giản hóa dữ liệu đặc biệt hữu ích cho những mô hình phức tạp hơn, chẳng hạn như cây quyết định và mạng nơ-ron nhằm giảm độ phức tạp của mô hình bằng cách loại bỏ các thành phần không cần thiết, giúp ngăn ngừa quá khớp.

Dừng sớm: Một biện pháp chủ động để ngăn ngừa hiện tượng quá khớp trong các thuật toán học lặp, cho phép ta theo dõi hiệu suất của mô hình trong quá trình huấn

luyện và dừng lại khi hiệu suất bắt đầu giảm. Điều này giúp ngăn mô hình trở nên quá chuyên biệt đối với dữ liệu huấn luyện và cho phép nó tổng quát tốt hơn cho dữ liệu mới.

3.2 Giải pháp overfitting vào các phương pháp ở Câu 1

Hàm để tính toán và in kết quả

```
# Hàm để tính toán và in kết quả
def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions, average='macro')
    recall = recall_score(y_test, predictions, average='macro')
    f1 = f1_score(y_test, predictions, average='macro')

    return accuracy, precision, recall, f1
```

3.2.1 Với Decision Tree (Cây quyết định)

Phương pháp giải quyết Overfitting: Tỉa (Pruning)

Tỉa là quá trình loại bỏ các phần của cây không cung cấp sức mạnh dự đoán đáng kể cho biến mục tiêu, giúp giảm kích thước của cây, loại bỏ nhiễu ghi nhận từ dữ liệu huấn luyện. Ta có thể thực hiện Tỉa trước (Pre-pruning) bằng cách thiết lập các điều kiện để dừng cây, không cho cây phát triển quá sâu (như đặt độ sâu tối đa hoặc số lượng mẫu tối thiểu cho mỗi lá) hoặc Tỉa sau (Post-pruning) cho phép cây phát triển hoàn toàn và sau đó cắt bớt bằng cách loại bỏ các nút có ít tầm quan trọng.

Đầu tiên, tạo một cây quyết định không giới hạn độ sâu, sau đó đánh giá mô hình với các tập dữ liệu huấn luyện và kiểm tra.

```
# Decision Tree không có giải pháp overfitting
decision_tree_model = DecisionTreeClassifier()
dt_results_no_overfitting = evaluate_model(decision_tree_model, X_train, y_train, X_test, y_test)
```

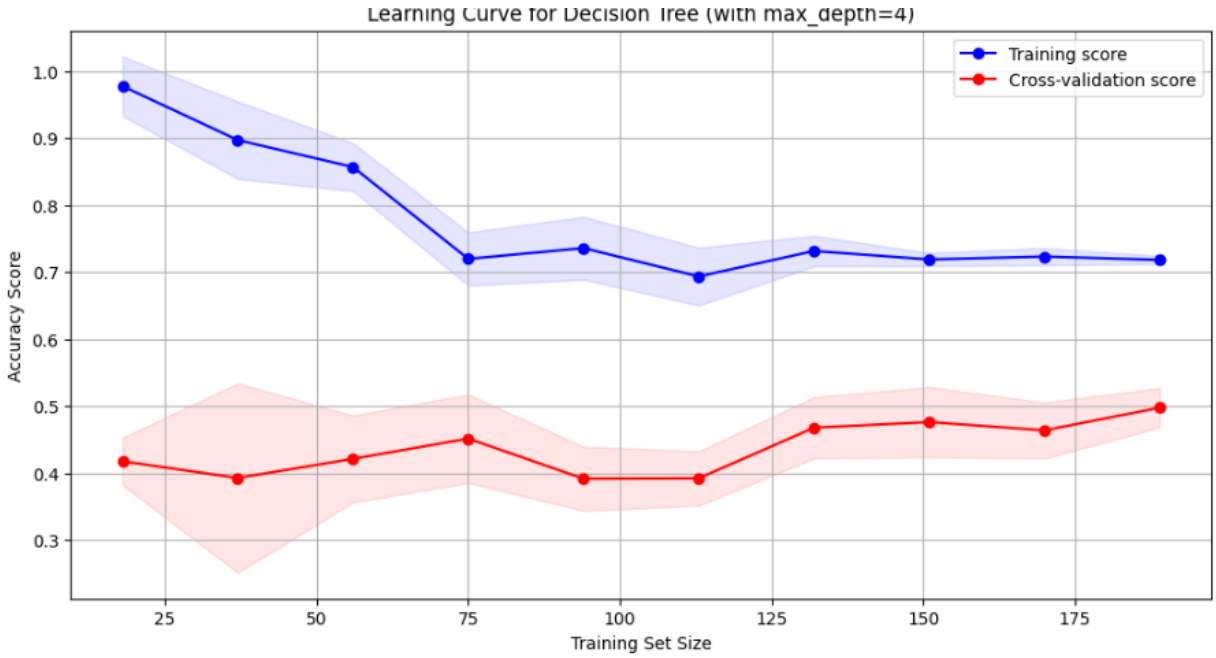
Tạo một cây quyết định có độ sâu tối đa là 4, tiếp tục đánh giá mô hình thứ hai để so sánh hiệu quả sau khi giảm độ phức tạp của mô hình.

```
# Decision Tree với giải pháp overfitting (Tỉa)
decision_tree_model_overfitting = DecisionTreeClassifier(max_depth=4)  # Giới hạn độ sâu
dt_results_with_overfitting = evaluate_model(decision_tree_model_overfitting, X_train, y_train, X_test, y_test)
```

Nhận xét: Trước khi có giải pháp cho quá khớp, mô hình đạt điểm cao ở các chỉ số. Điều này đôi khi có hạn chế vì khi đó mô hình có thể đã học quá chi tiết các mẫu trong tập huấn luyện, khiến cho nó khó thích nghi với dữ liệu mới, khó tổng quát hóa. Sau khi chống quá khớp bằng cách giới hạn độ sâu của cây (tỉa trước), Accuracy của mô hình đã tăng đáng kể. cho thấy việc giảm thiểu overfitting giúp mô hình tổng quát hóa tốt hơn trên dữ liệu mới, các chỉ số còn lại giảm xuống, cho thấy mô hình ít nguy cơ quá khớp hơn, nhưng có thể thiếu độ chính xác do mất đi một số đặc điểm. Tuy nhiên kết

quả vẫn cho thấy rằng độ sâu tối đa đóng vai trò quan trọng trong việc giảm thiểu overfitting.

Nhận xét: Training Score ban đầu rất cao, gần 1.0, nhưng giảm dần khi kích thước tập huấn luyện tăng lên, việc giới hạn độ sâu cây giúp ngăn mô hình học quá kỹ các đặc trưng của tập huấn luyện, làm cho điểm số huấn luyện giảm và trở nên ổn định hơn khi kích thước tập tăng. Trong khi đó, Cross-Validation Score thấp hơn điểm số huấn luyện,, nhưng trở nên ổn định khi kích thước tập tăng.



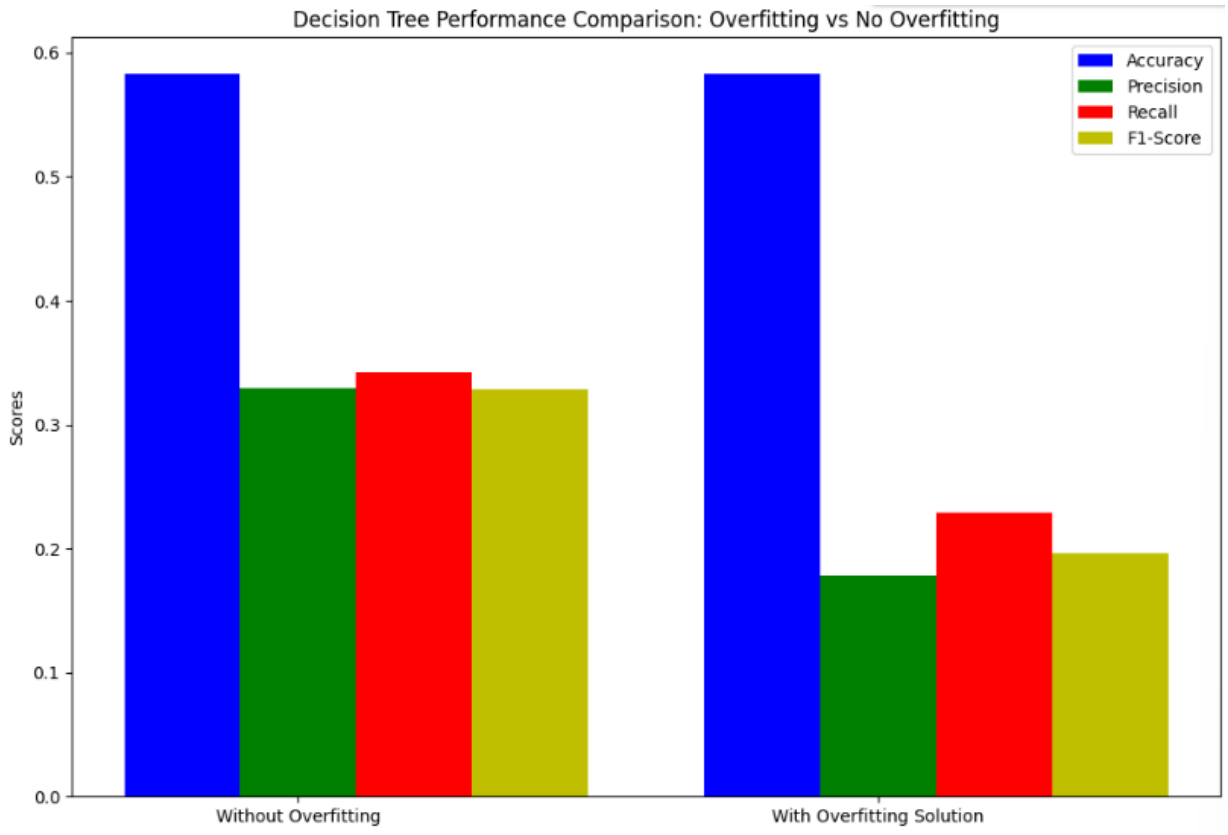
3.2.2 Với K-Nearest Neighbors (KNN)

Phương pháp giải quyết Overfitting: Điều chỉnh số lượng láng giềng (k)

Sự lựa chọn số lượng láng giềng gần nhất cần xem xét k rất quan trọng. Một k nhỏ có thể dẫn đến overfitting, trong khi một k lớn hơn sẽ làm mượt đường quyết định.

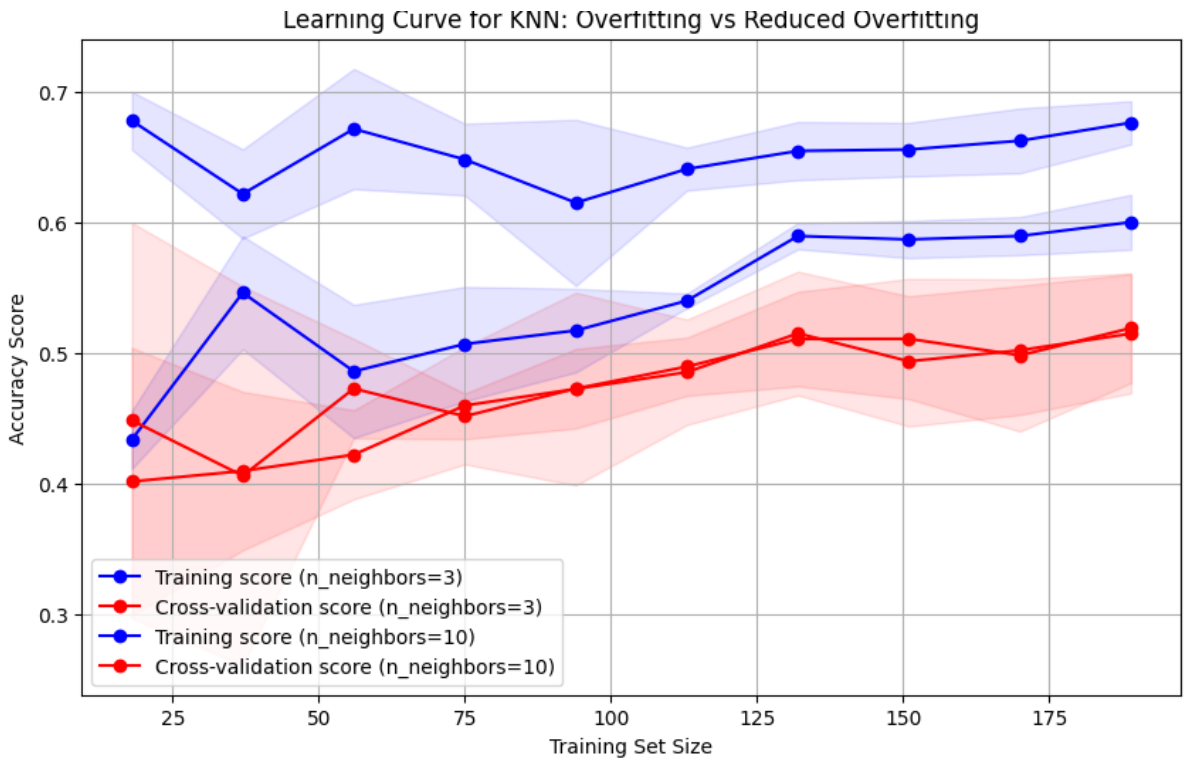
```
# 1. K-Nearest Neighbors (KNN)
# KNN không có giải pháp overfitting
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_results_no_overfitting = evaluate_model(knn_model, X_train, y_train, X_test, y_test)

# KNN với giải pháp overfitting (tăng số lượng láng giềng)
knn_model_overfitting = KNeighborsClassifier(n_neighbors=10)  # Tăng số lượng láng giềng
knn_results_with_overfitting = evaluate_model(knn_model_overfitting, X_train, y_train, X_test, y_test)
```

Nhận xét: Khi không áp dụng giải pháp overfitting, mô hình có thể nhạy cảm với dữ liệu nhiễu và có thể dẫn đến độ chính xác và các chỉ số khác thấp hơn, đặc biệt khi số lượng láng giềng (k) quá nhỏ. Bằng cách tăng số lượng láng giềng (k), mô hình trở nên mượt mà hơn và ít nhạy cảm hơn với dữ liệu nhiễu, từ đó cải thiện các chỉ số như độ chính xác, độ chính xác (precision), độ nhạy (recall), và F1-score. Kết quả cho thấy rằng việc điều chỉnh tham số k là rất quan trọng để duy trì sự cân bằng giữa việc đạt được độ chính xác cao và tránh overfitting.



Nhận xét: Trường hợp n_neighbors=3, training score của tập huấn luyện cao hơn hẳn, dao động khoảng 0.6 đến 0.7, cho thấy mô hình đã học tốt trên tập huấn luyện. Trong khi đó, Cross-Validation Score thấp hơn nhiều, khoảng từ 0.4 đến 0.55, cho thấy mô hình có khả năng bị overfitting, vì nó học quá kỹ các đặc trưng của tập huấn luyện nhưng không tổng quát tốt trên dữ liệu mới.

Trường hợp n_neighbors=10, Training Score huấn luyện thấp hơn so với trường hợp n_neighbors=3, dao động khoảng 0.45 đến 0.55, cho thấy mô hình ít bị overfitting hơn và tổng quát hơn khi tăng giá trị n_neighbors. Đồng thời, Cross-Validation Score gần như tương đương, ít thay đổi so với điểm số huấn luyện, cho thấy mô hình có khả năng tổng quát tốt hơn, giúp giảm hiện tượng overfitting.



3.2.3 Với Random Forest

Rừng Ngẫu Nhiên về bản chất ít bị overfitting hơn so với một cây quyết định đơn lẻ vì nó trung bình hóa nhiều cây. Tuy nhiên, ta vẫn có thể áp dụng thêm các kỹ thuật khác:

Cách thực hiện:

Độ sâu tối đa: Giới hạn độ sâu tối đa của từng cây để giảm độ phức tạp.

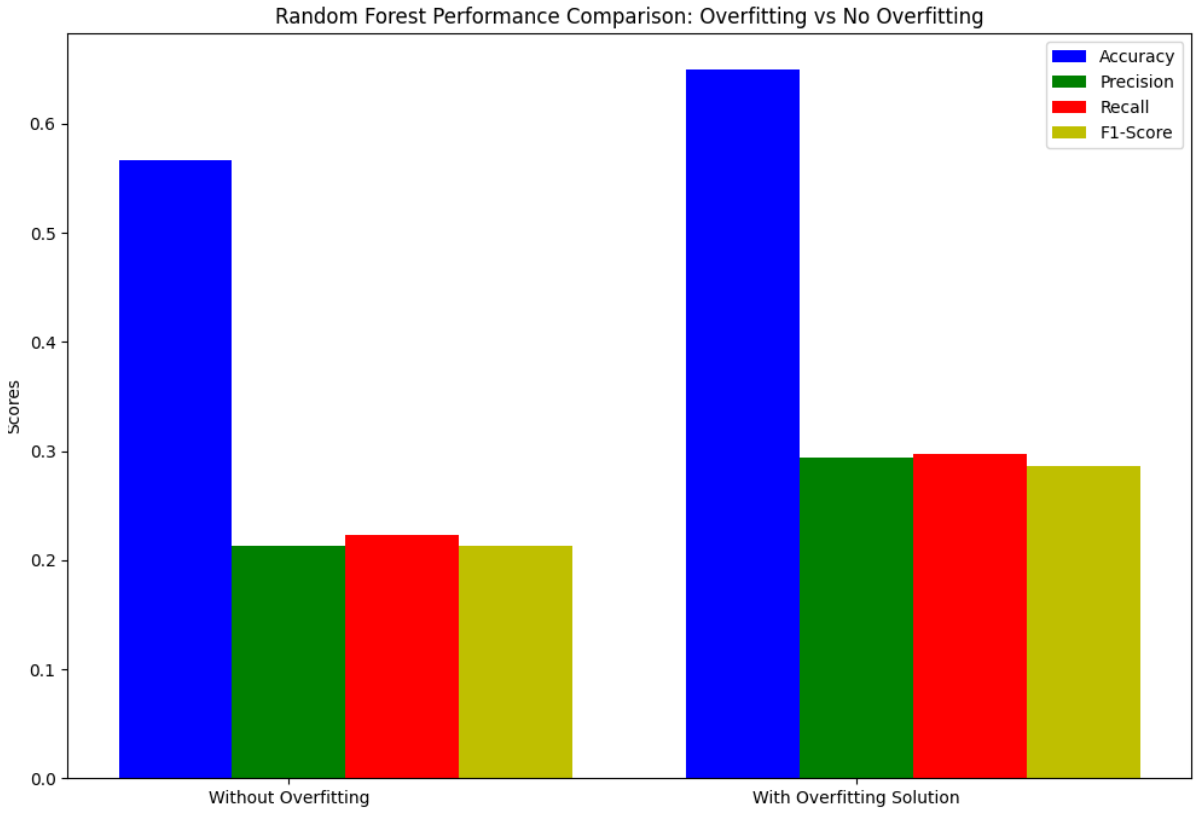
Số mẫu tối thiểu để tách: Tăng số mẫu tối thiểu cần có để tách một nút bên trong.

Số mẫu tối thiểu cho lá: Chỉ định số mẫu tối thiểu phải có trong một nút lá.

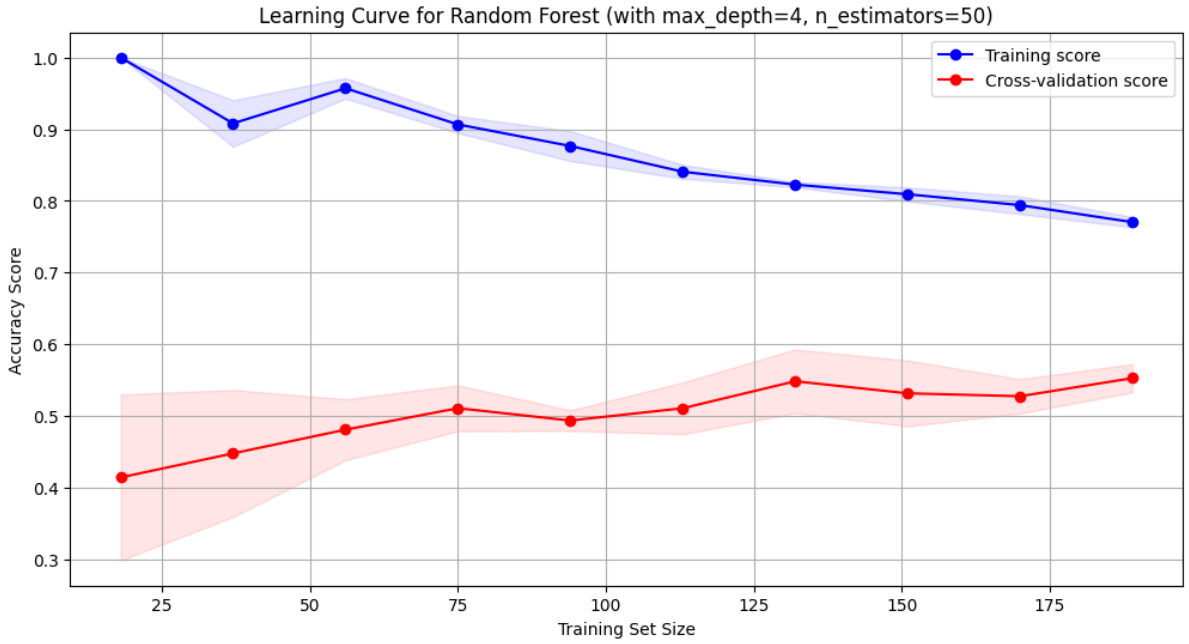
```
# 1. Random Forest không giảm overfitting
random_forest_model = RandomForestClassifier(random_state=42)
rf_results_no_overfitting = evaluate_model(random_forest_model, X_train, y_train, X_test, y_test)

# 2. Random Forest với giải pháp overfitting (Giới hạn độ sâu, số lượng mẫu tối thiểu)
random_forest_model_overfitting = RandomForestClassifier(n_estimators=50, max_depth=4, min_samples_split=5, random_state=42)
rf_results_with_overfitting = evaluate_model(random_forest_model_overfitting, X_train, y_train, X_test, y_test)
```

Nhận xét: Việc giới hạn độ sâu cây và số mẫu tối thiểu để tách (min_samples_split) cho thấy rõ ràng hiệu quả trong việc kiểm soát độ phức tạp của mô hình. Những thay đổi này đã cải thiện độ chính xác, độ chính xác, độ nhạy và F1-score, cho thấy mô hình có thể tổng quát tốt hơn trên dữ liệu mới. Kết quả nhấn mạnh tầm quan trọng của việc điều chỉnh tham số trong rừng ngẫu nhiên để duy trì hiệu suất và tính khả thi trong ứng dụng thực tế

.



Nhận xét: Training Score ban đầu rất cao, gần 1.0, nhưng giảm dần khi kích thước tập huấn luyện tăng lên, cho thấy mô hình ban đầu học rất tốt trên tập huấn luyện nhỏ, nhưng khi tăng kích thước tập huấn luyện, độ chính xác huấn luyện giảm đi, giúp mô hình trở nên tổng quát hơn và ít bị overfitting. Trong khi đó, Cross-Validation Score thấp hơn đáng kể so với điểm số huấn luyện. Mặc dù điểm số thấp hơn, nhưng biểu đồ cho thấy sự ổn định khi kích thước tập huấn luyện tăng lên.

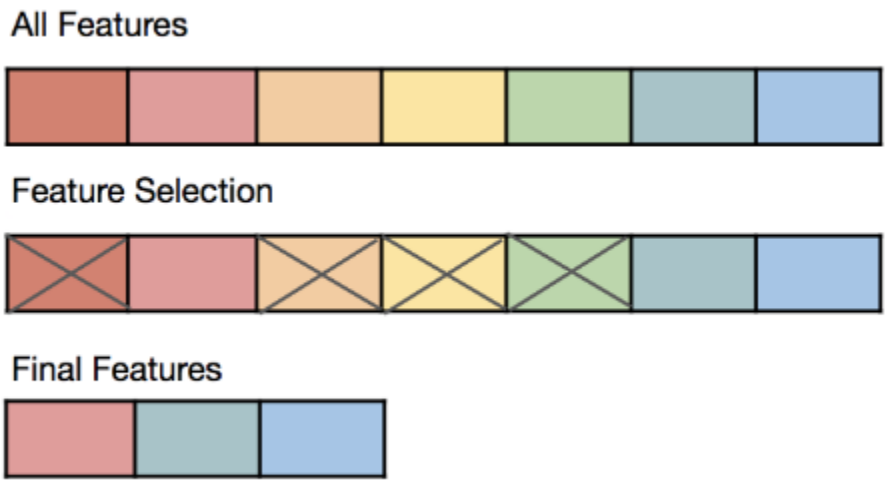

Learning Curve for Random Forest (with max_depth=4, n_estimators=50)
Training score
Cross-validation score
Accuracy Score
Training Set Size
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
25
50
75
100
125
150
175

CHƯƠNG 4: VẤN ĐỀ THỨ BA

4.1 Tìm hiểu chung về Feature selection

Feature Selection là một trong những kỹ thuật quan trọng nhằm mục tiêu giảm số lượng biến đầu vào xuống đến mức tối thiểu nhưng vẫn hiệu quả nhất để đưa vào các thuật toán học máy.

"Feature selection is primarily focused on removing non-informative or redundant predictors from the model." (Page 488, Applied Predictive Modeling, 2013.)



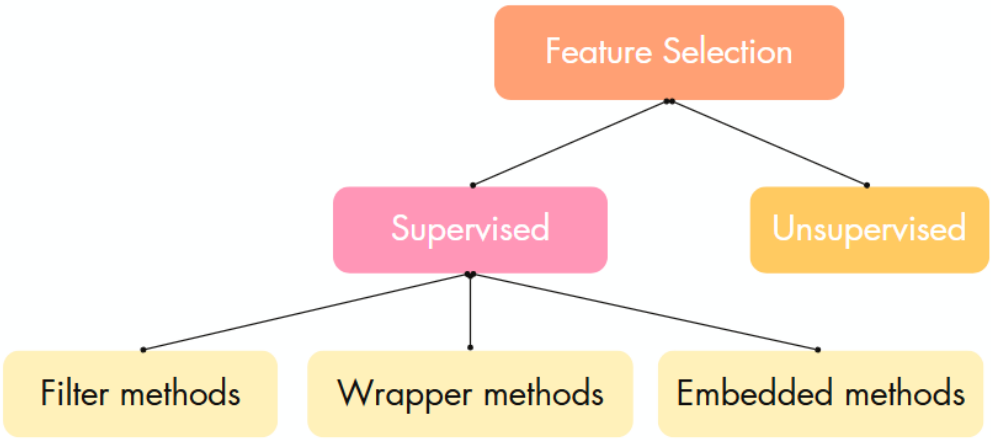
Hình 4. 1 Minh họa Feature selection

Lợi ích của Feature Selection:

Mô hình trở nên đơn giản hơn: Việc giảm thiểu số lượng biến đầu vào đồng nghĩa với việc thuật toán học máy sẽ làm việc với ít biến đầu vào hơn, từ đó mô hình có thể đơn giản và hạn chế ảnh hưởng bởi các thuộc tính nhiễu.

Thời gian huấn luyện ngắn hơn: Quá nhiều dữ liệu đầu vào có thể làm chậm quá trình đào tạo, vì vậy một tập hợp chính xác các đặc trưng sẽ giảm thiểu thời gian cần thiết để huấn luyện mô hình.

Cải thiện độ chính xác: Để đào tạo một mô hình, chúng ta cần thu thập lượng lớn dữ liệu để máy tính có thể học hỏi, tuy nhiên thông thường một phần trong dữ liệu trong đó sẽ không đóng góp quá nhiều vào hiệu suất hoạt động của mô hình, điều này có thể khiến cho mô hình phải làm việc với các dữ liệu rác dẫn đến kết quả trả về không chính xác. Vì vậy, việc chắt lọc dữ liệu đầu vào thực sự cần thiết để cải thiện độ chính xác cho dữ liệu đầu ra.



Hình 4. 2 Một số phương pháp lựa chọn tính năng

4.1.1 Phương pháp Lọc (Filter Methods - Lựa chọn Tính năng Đơn biến)

Phương pháp Lọc thường được sử dụng như một bước tiền xử lý, bằng cách chọn các tính năng dựa trên một số tiêu chí thống kê nhất định trước khi xây dựng mô hình. Nhờ vậy, nó việc lựa chọn các tính năng không phụ thuộc vào bất kỳ thuật toán học máy nào.

Trong phương pháp này, các tính năng được xếp hạng dựa trên tác động của chúng lên biến mục tiêu, từ đó các biến có mức độ quan trọng hàng đầu sẽ được chọn để xây dựng mô hình. Điều này làm cho phương pháp lọc trở thành bước đầu tiên hiệu quả để thu hẹp nhóm tính năng xuống chỉ còn những tính năng có liên quan nhất, có khả năng dự đoán nhất.

Hình 4. 3 Minh họa phương pháp Lọc

Các kỹ thuật phổ biến:

SelectKBest: Chọn ra K tính năng hàng đầu dựa trên một hàm điểm số (ví dụ: F-statistic, chi-squared, thông tin tương hỗ).

Phân tích Phương sai (Analysis of Variance - ANOVA) xác định các tính năng liên quan nhất đến biến mục tiêu bằng cách sử dụng điểm F.

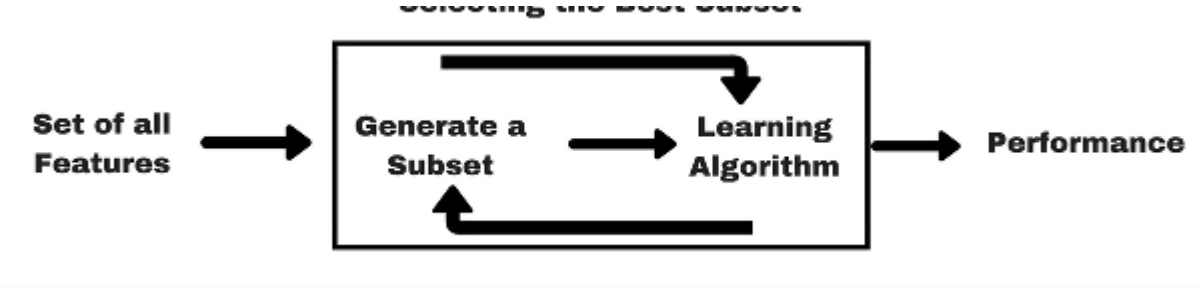
Kiểm định Chi - Square (Chi - Square Test) đo lường mức độ phụ thuộc giữa các tính năng và biến mục tiêu. Phương pháp này được sử dụng trong các bài toán phân loại.

Thông tin tương hỗ (Mutual Information): đo lường mức độ thông tin một tính năng mang lại về biến mục tiêu.

Tuy nhiên, phương pháp Lọc có nhược điểm đó là không giải quyết được vấn đề đa cộng tuyến (multicollinearity) trong tập dữ liệu, nghĩa là không thể tự động phát hiện hoặc loại bỏ những đặc trưng có sự tương quan mạnh mẽ với nhau. Vì vậy, trước khi bắt đầu huấn luyện mô hình, ta có thể thêm bước xử lý multicollinearity.

4.1.2 Phương pháp Bọc (Wrapper Methods)

Phương pháp Bọc chia dữ liệu thành các tập hợp con và dùng các tập hợp này để huấn luyện mô hình. Sau khi thu được kết quả, dựa trên những đánh giá và suy luận rút ra, ta sẽ quyết định thêm hay xóa một/một vài tính năng khỏi tập hợp con. Cuối cùng, ta sẽ đánh giá kết quả của tất cả các mô hình và chọn ra mô hình tốt nhất.



Hình 4. 4 Minh họa phương pháp Bọc

Phương pháp Bọc có nhược điểm đó là tốn kém về mặt tính toán, bởi vì để tìm ra một tổ hợp các tính năng hiệu quả nhất ta có thể sẽ phải thử rất nhiều tổ hợp con. Ví dụ, đối với một tập dữ liệu có 3 tính năng, số lượng các tổ hợp là 7. Thế nhưng, trong thực tế, tập dữ liệu ta phải giải quyết lớn hơn nhiều, chỉ với một tập dữ liệu có 200 tính năng, các tổ hợp con có thể chạm đến khoảng 1,1259 nghìn tỷ.

Các kỹ thuật phổ biến:

Loại bỏ Tính năng Đệ quy (RFE) liên tục tạo ra các mô hình, sau đó giữ lại tính năng có hiệu suất tốt nhất hoặc kém nhất ở mỗi lần lặp. Tiếp tục xây dựng mô hình với các tính năng còn lại.

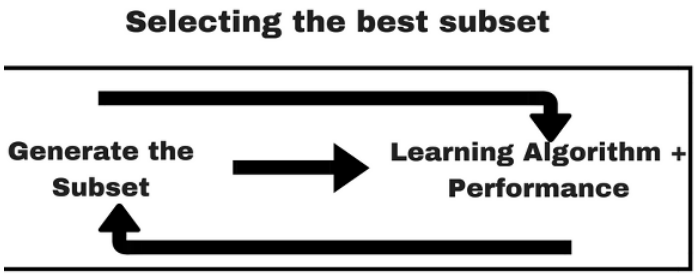
Lựa chọn Tiến dần: Dần dần thêm vào các tính năng cải thiện hiệu suất mô hình tốt nhất, và dừng lại khi việc thêm mới không cải thiện hiệu suất của mô hình nữa.

Loại bỏ tính năng ngược: Phương pháp này bắt đầu với toàn bộ tính năng, lưu lại hiệu suất của nó. Sau đó ta mới tiến hành tạo tất cả các tập hợp tính năng, tiếp tục xác định hiệu suất của từng mô hình, tiếp tục loại bỏ tính năng tạo ra mô hình có hiệu suất thấp nhất, và lặp lại quy trình cho đến khi đạt được tiêu chí.

4.1.3 Phương pháp Nhúng (Embedded Methods)

Phương pháp nhúng là sự kết hợp các đặc điểm của phương pháp lọc và phương pháp bọc. Phương pháp này tích hợp việc tìm kiếm một tập hợp con tối ưu của các tính

năng vào quá trình đào tạo bộ phân loại hoặc thuật toán hồi quy. Độ quan trọng của tính năng được xác định trong mô hình và các tính năng không liên quan hoặc dư thừa sẽ tự động bị loại bỏ.



Hình 4. 5 Minh họa phương pháp Nhúng

Hai kỹ thuật phổ biến của các phương pháp này là hồi quy LASSO và RIDGE

Hồi quy Lasso thực hiện chính quy hóa L1, các hệ số được thu nhỏ theo một hằng số nhất định. Lasso có khả năng đặt thu nhỏ các hệ số về 0 khi tham số điều chỉnh (regularization parameter) λ lớn, nghĩa là LASSO có thể loại bỏ hoàn toàn các tính năng không quan trọng.

Công thức của hàm mất mát (Loss Function) trong hồi quy Lasso:

$$\text{Loss Function} = \text{MSE} + \lambda \sum_{i=1}^{n} |\beta_i|$$

λ là tham số điều chỉnh, quyết định mức độ phạt.

$\sum_{i=1}^{n} |\beta_i|$ là L1 regularization, giúp loại bỏ các biến không quan trọng bằng cách giảm giá trị của các hệ số β_i về 0.

$$\text{Loss Function} = \text{MSE} + \lambda \sum_{i=1}^{n} \beta_i^2$$

Ridge cũng giúp thu nhỏ các hệ số về 0 nhưng lại không thể loại bỏ hoàn toàn các đặc trưng, nghĩa là tất cả các đặc trưng vẫn được giữ lại trong mô hình. Hồi quy Ridge thực hiện chính quy hóa L2 bằng cách thêm hình phạt tương đương với bình phương độ

lớn của các hệ số. Mục đích của việc thu nhỏ các hệ số là để giảm độ lệch và ngăn ngừa overfitting.

4.1.4 Phương pháp Kết hợp (Hybrid Methods)

Phương pháp kết hợp tận dụng sức mạnh của nhiều kỹ thuật lựa chọn tính năng. Ví dụ, chúng có thể sử dụng một phương pháp lọc ban đầu để giảm không gian tính năng và sau đó áp dụng một phương pháp bọc hoặc nhúng để tinh chỉnh hơn nữa.

4.2 Feature selection using correlation analysis

Lựa chọn đặc trưng bằng phân tích tương quan (Feature selection using correlation analysis) là phương pháp nhận diện và loại bỏ những đặc trưng có sự tương quan mạnh mẽ với đặc trưng khác.

Các bước cơ bản thực hiện Feature selection using correlation analysis:

Bước 1: Tiền xử lý dữ liệu và phân chia dữ liệu thành các features và biến mục tiêu (target).

Bước 2: Tính toán ma trận tương quan. Mức độ tương quan của hai đặc trưng thường được đo bằng chỉ số Pearson correlation coefficient.

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

Trong đó:

$X_{i,}\ Y_i$ các giá trị của hai biến X, Y.

$\bar{X}, \bar{Y}$ giá trị trung bình của biến X, Y

$\sum(Xi - \bar{X})(Yi - \bar{Y})$ tổng tích của các độ lệch của X và Y so với giá trị trung bình của chúng.

$\sqrt{\sum(X_i - \bar{X})^2 \ \sum(Y_i - \bar{Y})^2}$ căn bậc hai của tích tổng bình phương của các độ lệch của X, Y.

Bước 3: Xác định các ngưỡng tương quan, thông thường nằm trong khoảng từ 0.7 đến 0.9. Từ đó ta sẽ lựa chọn đặc trưng, có thể loại bỏ một trong hai biến trong các cặp biến có hệ số tương quan cao hơn ngưỡng, vì chúng có thể chứa thông tin tương tự nhau và không mang lại nhiều giá trị bổ sung cho mô hình.

Bước 4: Tính tương quan với biến mục tiêu để chọn những đặc trưng có tương quan cao với biến mục tiêu. Các đặc trưng có tương quan thấp với biến mục tiêu có thể bị loại bỏ.

Bước 5: Xây dựng và đánh giá mô hình

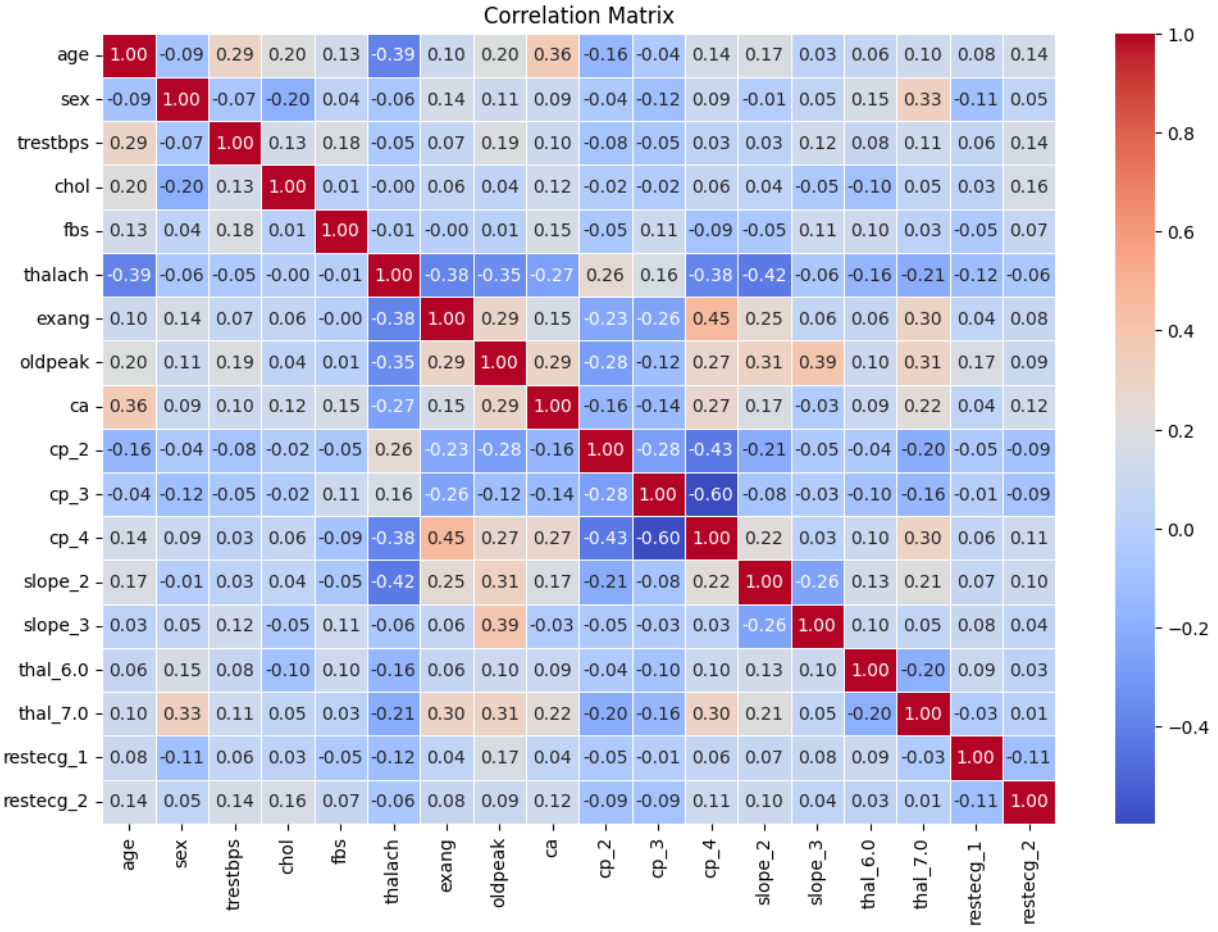
4.3 Áp dụng cho toàn bộ thuật toán ở Câu 1

Các bước thực hiện gồm:

Bước 1: Thu thập dữ liệu và tiền xử lý dữ liệu.

Bước 2: Tính ma trận tương quan giữa các biến sử dụng hệ số tương quan Pearson. Mỗi giá trị trong ma trận biểu thị mối quan hệ tương quan giữa hai biến.

```python
# Tính toán ma trận tương quan giữa các biến
corr_matrix = X.corr()

# Hiển thị ma trận tương quan dưới dạng heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```



Bước 3,4: Chọn các thuộc tính có tương quan cao với biến mục tiêu và loại bỏ thuộc tính có tương quan cao với nhau để tránh trùng lặp

```
# Lọc các cặp biến có tương quan cao (ngưỡng 0.9)
threshold = 0.9
high_corr_vars = np.where(abs(corr_matrix) > threshold)

# Tạo danh sách các cặp cột có tương quan cao
high_corr_pairs = [(X.columns[x], X.columns[y]) for x, y in zip(*high_corr_vars) if x != y and x < y]
```

```
# Loại bỏ các cột có mối tương quan cao
columns_to_drop = ['trestbps', 'chol']  # Các cột có thể được loại bỏ
X_selected = X.drop(columns=columns_to_drop)
```

4.3.1 Với thuật toán KNN

```
# K-Nearest Neighbors Classifier trên dữ liệu gốc
knn_before = KNeighborsClassifier()
knn_before.fit(X_train_class_original, y_train_class_original)
knn_pred_before = knn_before.predict(X_test_class_original)
accuracy_knn_before = accuracy_score(y_test_class_original, knn_pred_before)
print(f"Accuracy của K-Nearest Neighbors trước khi chọn lọc tính năng: {accuracy_knn_before}")
```

```
# K-Nearest Neighbors Classifier sau khi chọn lọc tính năng
knn_after = KNeighborsClassifier()
knn_after.fit(X_train, y_train)
knn_pred_after = knn_after.predict(X_test)
accuracy_knn_after = accuracy_score(y_test, knn_pred_after)
print(f"Accuracy của K-Nearest Neighbors sau khi chọn lọc tính năng: {accuracy_knn_after}")
```

4.3.2 Với thuật toán SVM

```
# SVM trên dữ liệu gốc
svm_before = SVC(random_state=42)
svm_before.fit(X_train_class_original, y_train_class_original)
svm_pred_before = svm_before.predict(X_test_class_original)
accuracy_svm_before = accuracy_score(y_test_class_original, svm_pred_before)
print(f"Accuracy của SVM trước khi chọn lọc tính năng: {accuracy_svm_before}")
```

```
# SVM sau khi chọn lọc tính năng
svm_after = SVC(random_state=42)
svm_after.fit(X_train, y_train)
svm_pred_after = svm_after.predict(X_test)
accuracy_svm_after = accuracy_score(y_test, svm_pred_after)
```

4.3.3 Với thuật toán Naïve Bayes

```
# Naive Bayes trên dữ liệu gốc
nb_before = GaussianNB()
nb_before.fit(X_train_class_original, y_train_class_original)
nb_pred_before = nb_before.predict(X_test_class_original)
accuracy_nb_before = accuracy_score(y_test_class_original, nb_pred_before)
print(f"Accuracy của Naive Bayes trước khi chọn lọc tính năng: {accuracy_nb_before}")
```

```
# Naive Bayes sau khi chọn lọc tính năng
nb_after = GaussianNB()
nb_after.fit(X_train, y_train)
nb_pred_after = nb_after.predict(X_test)
accuracy_nb_after = accuracy_score(y_test, nb_pred_after)
```

4.3.4 Với thuật toán Hồi quy tuyến tính

```
# Linear Regression trên dữ liệu gốc
linreg_before = LinearRegression()
linreg_before.fit(X_train_reg_original, y_train_reg_original)
linreg_pred_before = linreg_before.predict(X_test_reg_original)
mse_linreg_before = mean_squared_error(y_test_reg_original, linreg_pred_before)
r2_linreg_before = r2_score(y_test_reg_original, linreg_pred_before)
print(f"MSE của Linear Regression trước khi chọn lọc tính năng: {mse_linreg_before}")
print(f"R2 của Linear Regression trước khi chọn lọc tính năng: {r2_linreg_before}")
```

```
# Linear Regression sau khi chọn lọc tính năng
linreg_after = LinearRegression()
linreg_after.fit(X_train, y_train)
linreg_pred_after = linreg_after.predict(X_test)
mse_linreg_after = mean_squared_error(y_test, linreg_pred_after)
r2_linreg_after = r2_score(y_test, linreg_pred_after)
```

4.3.5 Với thuật toán Random Forest

```
# Random Forest Regressor trên dữ liệu gốc
rf_regressor_before = RandomForestRegressor(random_state=42)
rf_regressor_before.fit(X_train_reg_original, y_train_reg_original)
rf_pred_before = rf_regressor_before.predict(X_test_reg_original)
mse_rf_before = mean_squared_error(y_test_reg_original, rf_pred_before)
r2_rf_before = r2_score(y_test_reg_original, rf_pred_before)
print(f"MSE của Random Forest Regressor trước khi chọn lọc tính năng: {mse_rf_before}")
print(f"R2 của Random Forest Regressor trước khi chọn lọc tính năng: {r2_rf_before}")
```

```
# Random Forest Regressor sau khi chọn lọc tính năng
rf_regressor_after = RandomForestRegressor(random_state=42)
rf_regressor_after.fit(X_train, y_train)
rf_pred_after = rf_regressor_after.predict(X_test)
mse_rf_after = mean_squared_error(y_test, rf_pred_after)
r2_rf_after = r2_score(y_test, rf_pred_after)
```

4.3.6 Với thuật toán K-Nearest Neighbors

```
# K-Nearest Neighbors Classifier trên dữ liệu gốc
knn_before = KNeighborsClassifier()
knn_before.fit(X_train_class_original, y_train_class_original)
knn_pred_before = knn_before.predict(X_test_class_original)
accuracy_knn_before = accuracy_score(y_test_class_original, knn_pred_before)
print(f"Accuracy của K-Nearest Neighbors trước khi chọn lọc tính năng: {accuracy_knn_before}")
```

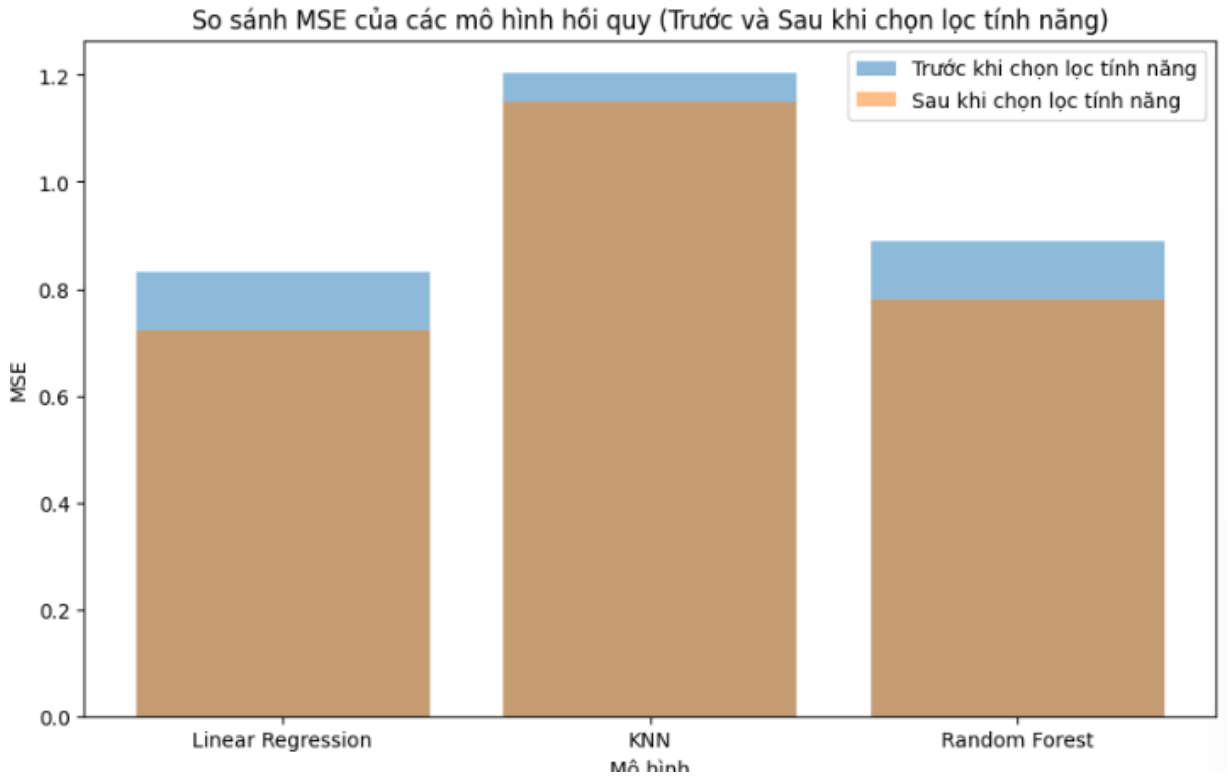
```
# K-Nearest Neighbors Classifier sau khi chọn lọc tính năng
knn_after = KNeighborsClassifier()
knn_after.fit(X_train, y_train)
knn_pred_after = knn_after.predict(X_test)
accuracy_knn_after = accuracy_score(y_test, knn_pred_after)
```

4.3.7 Kết quả

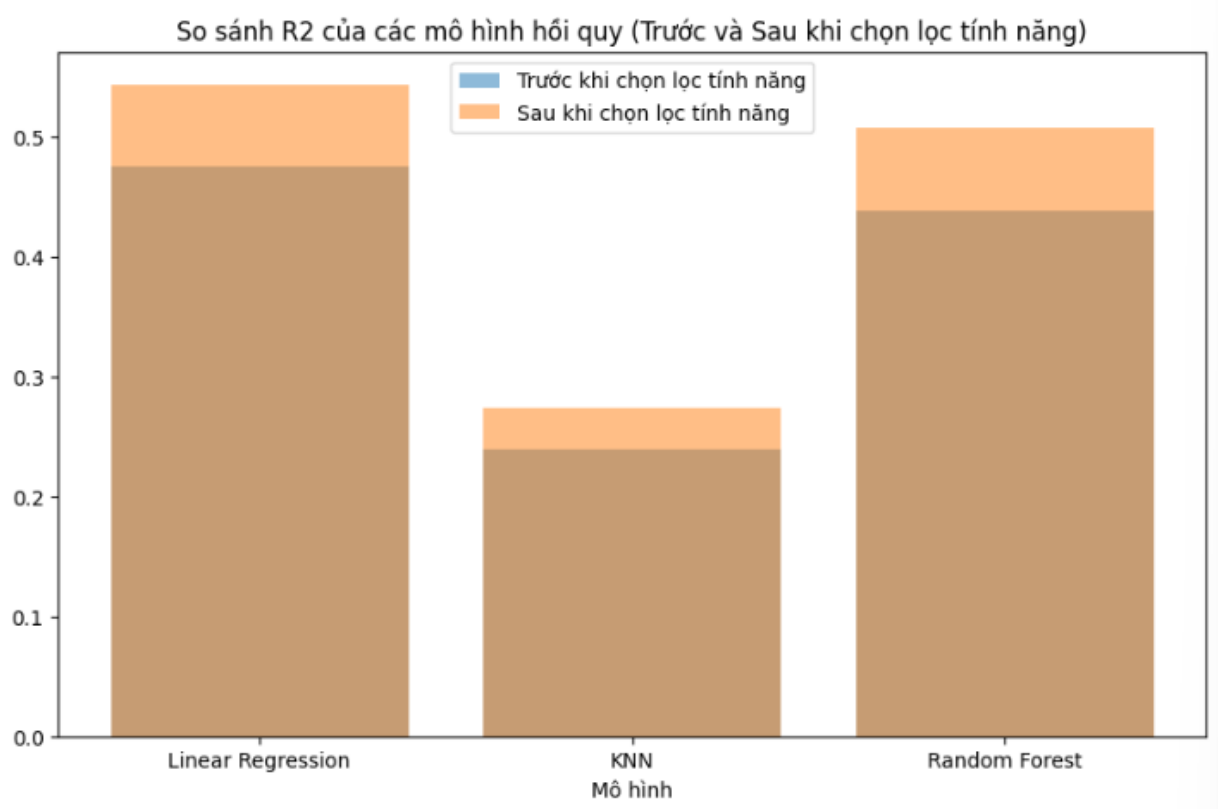
```
High correlation pairs (greater than 0.9):
        age  sex  fbs  thalach  exang   oldpeak        ca   cp_2   cp_3  \
0  0.936181    1    1      150      0  1.068965 -0.721976  False  False
1  1.378929    1    0      108      1  0.381773  2.478425  False  False
2  1.378929    1    0      129      1  1.326662  1.411625  False  False
3 -1.941680    1    0      187      0  2.099753 -0.721976  False   True
4 -1.498933    0    0      172      0  0.295874 -0.721976   True  False

    cp_4  slope_2  slope_3  thal_6.0  thal_7.0  restecg_1  restecg_2
0  False    False     True      True     False      False       True
1   True     True    False     False     False      False       True
2   True     True    False     False      True      False       True
3  False    False     True     False     False      False      False
4  False    False    False     False     False      False       True
Accuracy của K-Nearest Neighbors trước khi chọn lọc tính năng: 0.5666666666666667
Accuracy của SVM trước khi chọn lọc tính năng: 0.6
Accuracy của Naive Bayes trước khi chọn lọc tính năng: 0.25
Accuracy của K-Nearest Neighbors sau khi chọn lọc tính năng: 0.5833333333333334
Accuracy của SVM sau khi chọn lọc tính năng: 0.6
Accuracy của Naive Bayes sau khi chọn lọc tính năng: 0.23333333333333334
MSE của Linear Regression trước khi chọn lọc tính năng: 0.8315287536743805
R2 của Linear Regression trước khi chọn lọc tính năng: 0.4744551416383832
MSE của Random Forest Regressor trước khi chọn lọc tính năng: 0.8901966666666666
R2 của Random Forest Regressor trước khi chọn lọc tính năng: 0.43737570224719113
MSE của KNN trước khi chọn lọc tính năng: 1.204666666666667
R2 của KNN trước khi chọn lọc tính năng: 0.23862359550561796
```

Nhận xét: Trước khi có Feature selection using correlation analysis thì MSE của các mô hình cao hơn, bởi vì các mô hình có thể gặp hiện tượng overfitting do sử dụng quá nhiều đặc trưng bao gồm những đặc trưng có tính tương quan cao với nhau. Khi có nhiều đặc trưng dư thừa, các mô hình sẽ học quá nhiều chi tiết không quan trọng từ dữ liệu huấn luyện dẫn đến MSE lớn trên tập kiểm tra.

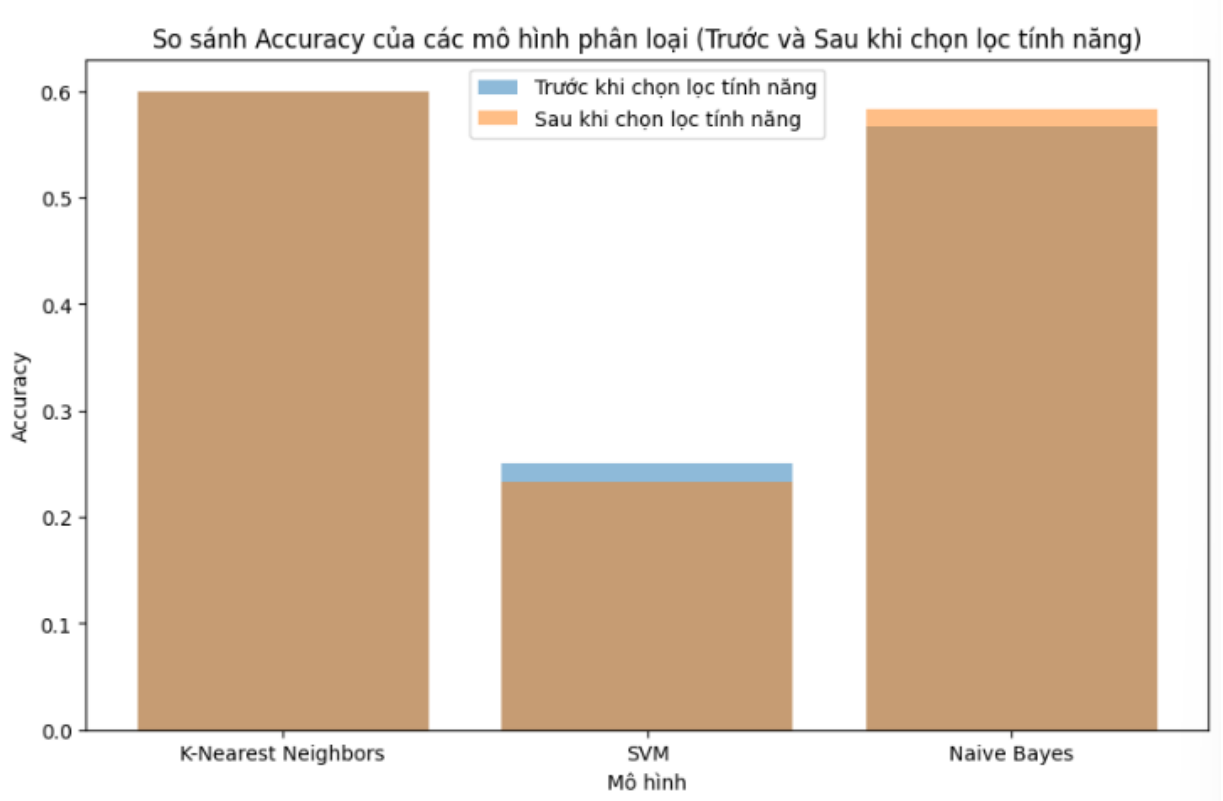
Sau khi Feature Selection thì MSE đã giảm, bởi vì việc loại bỏ các đặc trưng có độ tương quan cao giúp giảm overfitting và làm cho mô hình trở nên đơn giản hơn, tập trung vào những đặc trưng quan trọng.



Nhận xét: Trước khi có Feature Selection using correlation analysis thì chỉ số $R^2 - Score$ thấp hơn. Sau khi chọn lọc đặc trưng thì chỉ số này đều tăng so với trước đó, cho thấy việc loại bỏ các đặc trưng không cần thiết giúp cải thiện khả năng dự đoán của mô hình.



Nhận xét: Trước khi áp dụng feature selection, các mô hình có thể sử dụng nhiều đặc trưng, bao gồm những đặc trưng có tính tương quan cao, dẫn đến hiện tượng overfitting. Sau khi dùng feature selection using correlation analysis, Accuracy có xu hướng tang nhẹ hoặc ổn định, vì việc loại bỏ các đặc trưng có tương quan cao giúp giảm overfitting và nhiễu. Mô hình sẽ tập trung vào những đặc trưng quan trọng hơn và có thể tổng quát hóa tốt hơn.


So sánh Accuracy của các mô hình phân loại (Trước và Sau khi chọn lọc tính năng)
Trước khi chọn lọc tính năng
Sau khi chọn lọc tính năng
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
K-Nearest Neighbors
SVM
Naive Bayes
Mô hình

TÀI LIỆU THAM KHẢO

1. https://scikit-learn.org/stable/supervised_learning.html
2. https://machinelearningcoban.com/2016/12/28/linearregression/
3. https://www.codecademy.com/article/fe-filter-methods
4. https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/#3
5. Slide Môn Nhập Môn Học Máy Khoa Công Nghệ Thông Tin Trường Đại Học Tôn Đức Thắng

PHỤ LỤC