

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN HỌC MÁY VÀ ỨNG DỤNG

**HỌC MÁY: K-Nearest Neighbor, Ứng dụng thực tế và Hướng
nghiên cứu**

Giảng viên giảng dạy: TS. Võ Thị Hồng Thắm

Sinh viên thực hiện : Chu Doãn Đức

MSSV : 2000003917

Môn học : Học máy và Ứng dụng

Khóa : 2020

Tp.HCM, tháng 8 năm 2023

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH

KỲ THI KẾT THÚC HỌC PHẦN

TRUNG TÂM KHẢO THÍ

HỌC KỲ ...III... NĂM HỌC ...2022... - ...2023...

PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN

Môn thi: Học máy và Ứng dụng..... Lớp học phần: 20DTH1D

Nhóm sinh viên thực hiện: 1

1. Chu Doãn Đức Tham gia đóng góp: 100%

2..... Tham gia đóng góp:

3..... Tham gia đóng góp:

4. Tham gia đóng góp:

5..... Tham gia đóng góp:

6..... Tham gia đóng góp:

Ngày thi: 13/09/2023 Phòng thi: L.401

Đề tài tiểu luận/báo cáo của sinh viên: Mô hình dự đoán bệnh ung thư tuyến tiền liệt
bằng phương pháp K-NN

Phản đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Tiêu chí (theo CDR HP)	Đánh giá của GV	Điểm tối đa	Điểm đạt được
Cấu trúc của báo cáo		
Nội dung			
- Các nội dung thành phần		
- Lập luận		
- Kết luận		
Trình bày		
TỔNG ĐIỂM			

Giảng viên chấm thi

(ký, ghi rõ họ tên)

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Trường Đại học Nguyễn Tất Thành đã đưa môn học “Học máy và Ứng dụng” vào trương trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – cô Võ Thị Hồng Thắm trực tiếp hướng dẫn, dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc và đã cho em chắc chắn được hoạch định trong tương lai của mình.

“Học máy và Ứng dụng” là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên nói chung và riêng bản thân em nói riêng. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều ngỡ ngàng và hạn hẹp. Mặc dù em đã cố gắng hết sức nhưng chắc chắn bài báo cáo của em khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong các thầy/cô chấm bài xem xét và góp ý để bài tiểu luận của em được hoàn thiện hơn.

Kính chúc thầy có nhiều sức khỏe, hạnh phúc, thành công trên con đường giảng dạy.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Chu Doãn Đức

LỜI MỞ ĐẦU

Trong thời đại Cách mạng công nghiệp lần thứ tư (4 IR hay Industry 4.0) hiện nay, thế giới số có vô số dữ liệu, chẳng hạn như dữ liệu Internet kết nối vạn vật (IoT), dữ liệu an ninh mạng, dữ liệu di động, dữ liệu kinh doanh, dữ liệu truyền thông xã hội, sức khỏe dữ liệu, v.v. Để phân tích những dữ liệu này một cách thông minh và phát triển các ứng dụng thông minh và tự động tương ứng, kiến thức về trí tuệ nhân tạo (AI), đặc biệt là học máy (ML) chính là chìa khóa để mở ra các tri thức mới. Nhiều loại thuật toán học máy như học có giám sát, không giám sát, bán giám sát và học tăng cường. Bên cạnh đó, học sâu, là một phần của nhóm phương pháp học máy rộng lớn hơn, có thể phân tích dữ liệu một cách thông minh trên quy mô lớn. Trong bài báo này, tôi trình bày một cái nhìn toàn diện về thuật toán Hồi quy tuyến tính (Linear Regression), học máy này là một phương pháp trong lĩnh vực máy học, thường được sử dụng trong học sâu và nhiều lĩnh vực khác để dự đoán giá trị đầu ra dựa trên các biến đầu vào. Có thể được áp dụng để nâng cao trí thông minh và khả năng của một ứng dụng. Do đó, đóng góp quan trọng của nghiên cứu này là giải thích các nguyên tắc của các kỹ thuật trong thuật toán Hồi quy tuyến tính và khả năng ứng dụng của phân tích và dự đoán dữ liệu trong các lĩnh vực ứng dụng trong thế giới thực khác nhau, chẳng hạn như kinh tế, y học, xử lý ngôn ngữ tự nhiên, môi trường, quảng cáo tiếp thị, v.v. Tôi cũng nhấn mạnh những thách thức và hướng phát triển tiềm năng dựa trên nghiên cứu của bài báo cáo này. Nhìn chung, bài báo cáo này nhằm mục đích phục vụ như một bài tìm hiểu về tổng quan của Học máy: Hồi quy tuyến tính, ứng dụng thực tế và hướng phát triển của thuật toán.

A series of horizontal dotted lines for writing.

Điểm giảng viên chấm vòng 2:

Giáo viên hướng dẫn

MỤC LỤC

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	1
1.1 – Tổng quan về học máy (Machine Learning):	1
1.1.1 – Giới thiệu học máy:	1
1.1.2 – Các thuật toán của học máy:	1
1.2 – Tổng quan về thuật toán K-Nearest Neighbors (K-NN):	5
1.2.1 – Giới thiệu K – NN:	5
1.2.2 – Thuật toán K-NN:.....	5
1.2.3 – Ưu và nhược điểm của thuật toán:.....	6
CHƯƠNG 2: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM.....	7
2.1 – Khai báo thư viện và dữ liệu:.....	7
2.2 – Khám phá dữ liệu:	8
2.2.1 – Tổng quát về dữ liệu:	8
2.2.2 – Tên tính năng và ý nghĩa:.....	9
2.2.3 – Làm sạch và sắp xếp lại dữ liệu:	10
2.3 – Trực quan hóa dữ liệu:.....	11
2.4 – Khởi tạo mô hình:.....	16
2.5 – Khởi tạo mô hình:.....	18
2.6 – Mô hình K-NN:	22
2.7 – Thực nghiệm:	23
2.3 – Kết luận và hướng phát triển.....	24
2.3.1 – Kết luận:.....	24
2.3.2 – Hướng phát triển:.....	25
CHƯƠNG 3: CHUYÊN ĐỀ NGHIÊN CỨU	27
3.1 – Giới thiệu:	27
3.2 – Các loại dữ liệu và kỹ thuật học máy:	29
3.2.1 – Các loại dữ liệu:.....	29
3.2.2 – Các kỹ thuật học máy:	30
3.3 – Nhiệm vụ và thuật toán của học máy:	34
3.3.1 – Phân tích phân loại:	34

3.3.2 – Phân tích phân cụm:.....	39
3.3.3 – Giảm kích thước và học tính năng.....	40
3.3.4 – Khai phá luật kết hợp:.....	42
3.3.5 – Học tăng cường	44
3.3.6 – Mạng Nơ-ron nhân tạo và Học sâu:.....	45
3.4 – Thách thức và hướng nghiên cứu.....	48
3.5 – Kết luận:.....	49
Tài liệu tham khảo	51

DANH MỤC HÌNH

Hình 1. 1: Tổng quan cách huấn luyện	5
Hình 3. 1: Các kỹ thuật trong máy học	30
Hình 3. 2: Mô hình học có giám sát.....	31
Hình 3. 3: Mô hình học không giám sát.....	32
Hình 3. 4: Sự khác biệt giữa 2 mô hình	32
Hình 3. 5: Cấu trúc chung của mô hình dự đoán dựa trên học máy	34
Hình 3. 6: Cấu trúc cây quyết định	36
Hình 3. 7: Cấu trúc rừng ngẫu nhiên khi xem xét nhiều cây quyết định	37
Hình 3. 8: Ví dụ về phân tích thành phần chính (PCA).....	42
Hình 3. 9: Hiệu suất học máy và học sâu theo sự gia tăng của dữ liệu	46
Hình 3. 10: Ví dụ về mạng thần kinh tích chập (CNN hoặc ConvNet).....	47
Hình 3. 11: Cấu trúc mô hình ANN với nhiều lớp xử lý	48

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 – Tổng quan về học máy (Machine Learning):

1.1.1 – Giới thiệu học máy:

Thuật toán học máy (Machine Learning) là các chương trình máy tính có khả năng học hỏi và hoàn thành các nhiệm vụ, đồng thời là cách để cải thiện hiệu suất theo thời gian vô cùng hiệu quả.

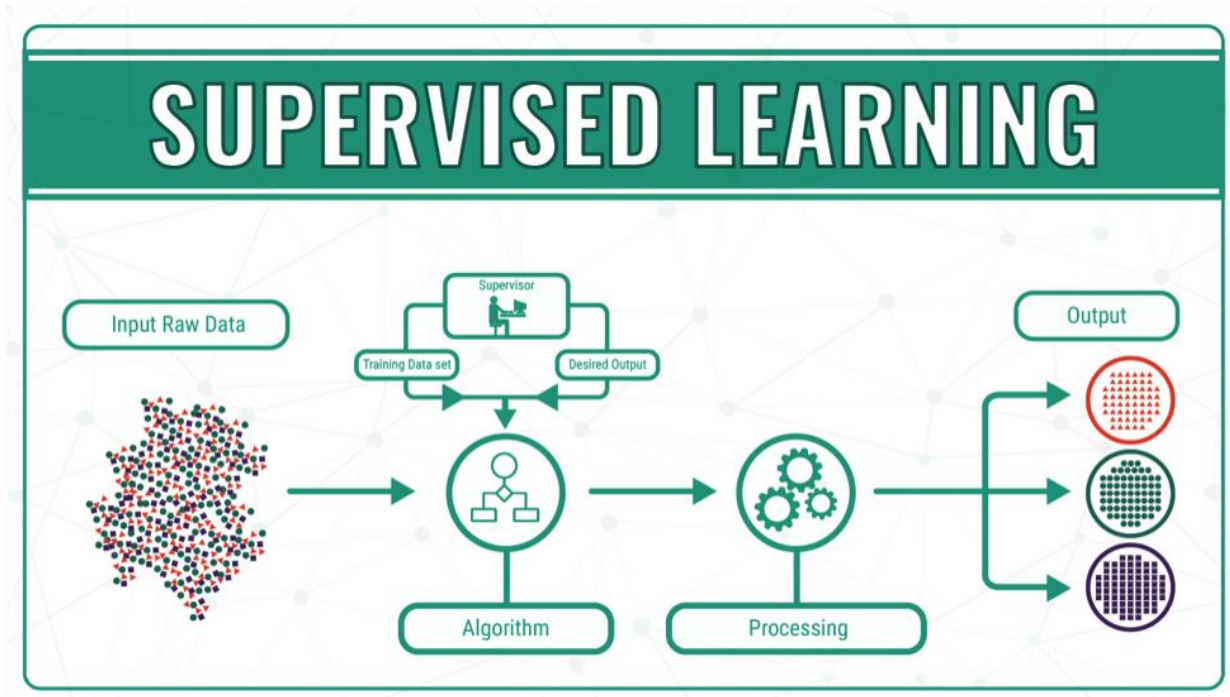
Ngoài ra, Machine Learning còn được biết là còn là công nghệ phát triển từ lĩnh vực trí tuệ nhân tạo tiên tiến nhất hiện nay. Để đảm bảo không có sai lệch và không xuất hiện dữ liệu giả, Machine Learning vẫn cần quá trình tìm hiểu và lựa chọn kỹ thuật phân tích dữ liệu từ con người.

Học máy ngày càng mang tính phổ biến trên toàn thế giới. Sự tăng trưởng vượt bậc của dữ liệu lớn (Big Data) và các thuật toán Machine Learning đã cải thiện độ chính xác của những mô hình và dự đoán tương lai.

1.1.2 – Các thuật toán của học máy:

a. Học có giám sát (Supervised Learning):

Học có giám sát là phương pháp sử dụng những dữ liệu được gán nhãn sẵn để suy luận ra quan hệ giữa đầu vào và đầu ra. Sau khi tìm hiểu cách tốt nhất để mô hình hóa các mối quan hệ cho dữ liệu được gán nhãn, thuật toán huấn luyện sẽ được sử dụng cho các bộ dữ liệu mới. Ứng dụng của học có giám sát chính là giúp xác định tín hiệu tốt nhất để dự báo xu hướng, lợi nhuận trong tương lai trong lĩnh vực cổ phiếu, chứng khoán.

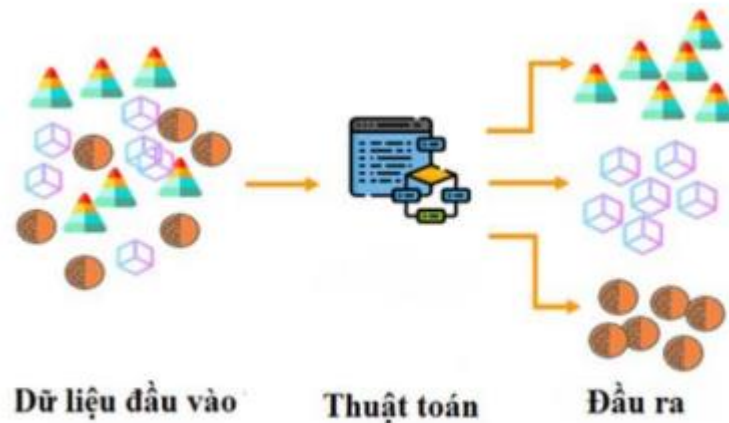


Hình 1. 1: Mô hình học có giám sát

Supervised Learning Là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là data, label tức dữ liệu, nhãn. Supervised Learning Là nhóm phổ biến nhất trong các thuật toán Machine learning

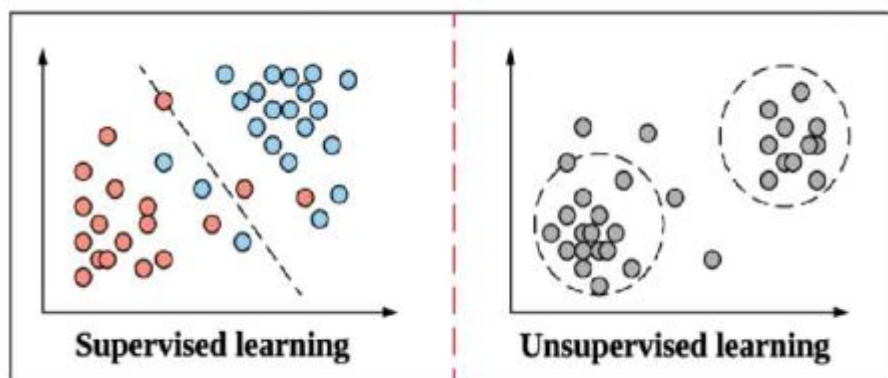
b. Học không giám sát (Unsupervised Learning)

Học không giám sát sử dụng những dữ liệu chưa được gán nhãn sẵn để suy luận và tìm cách để mô tả dữ liệu cùng cấu trúc của chúng. Ứng dụng của học không giám sát đó là hỗ trợ phân loại thành các nhóm có đặc điểm tương đồng.



Hình 1. 2: Mô hình học không giám sát

Trong thuật toán này, chúng ta không biết được dữ liệu đầu ra hay nhãn mà chỉ có dữ liệu đầu vào. Thuật toán Học không giám sát dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm hoặc giảm số chiều của dữ liệu để thuận tiện trong việc lưu trữ và tính toán. Một cách toán học, Học không giám sát là khi chúng ta chỉ có dữ liệu vào X mà không biết nhãn Y tương ứng. Sự khác nhau giữa học có giám sát và học không giám sát:



Hình 1. 3: Sự khác biệt giữa 2 mô hình

Học có giám sát: Là cách huấn luyện một mô hình trong đó dữ liệu học có đầu vào và đầu ra tương ứng đầu vào đó. Mô hình được huấn luyện bằng cách giảm thiểu sai số lỗi (loss) của các dự đoán tại các vòng lặp huấn luyện. Sau quá trình huấn luyện, mô hình sẽ có khả năng đưa ra dự đoán về đầu ra với một đầu vào mới gặp (không có trong

dữ liệu học). Nếu không gian đầu ra được biểu diễn dưới dạng rời rạc, ta gọi đó là bài toán phân loại (classification). Nếu không gian đầu ra được biểu diễn dưới dạng liên tục, ta gọi đó là bài toán hồi quy (regression). Học không giám sát: Là cách huấn luyện một mô hình trong đó dữ liệu học chỉ bao gồm đầu vào mà không có đầu ra. Mô hình sẽ được huấn luyện cách để tìm cấu trúc hoặc mối quan hệ giữa các đầu vào. Một trong những phương pháp học không giám sát quan trọng nhất là phân cụm (clustering): Tạo các cụm khác nhau ở mỗi cụm biểu diễn một đặc trưng nào đó của dữ liệu và phân các đầu vào mới vào các cụm theo các đặc trưng của đầu vào đó. Các phương pháp học không giám sát khác có thể kể đến như: phát hiện điểm bất thường (anomaly detection), Singular-value decomposition, ...

c. Học tăng cường (Reinforcement Learning)

Phương pháp học tăng cường tập trung vào việc làm sao để cho 1 tác tử trong môi trường có thể hành động sao cho lấy được phần thưởng nhiều nhất có thể. Khác với học có giám sát nó không có cặp dữ liệu gán nhãn trước làm đầu vào và cũng không có đánh giá các hành động là đúng hay sai. Trong đó, thuật toán học một chính sách hành động tùy theo các quan sát về thế giới. Mỗi hành động đều có tác động tới môi trường, và môi trường cung cấp thông tin phản hồi để hướng dẫn cho thuật toán của quá trình học.

d. Học bán giám sát (Semi-supervised learning)

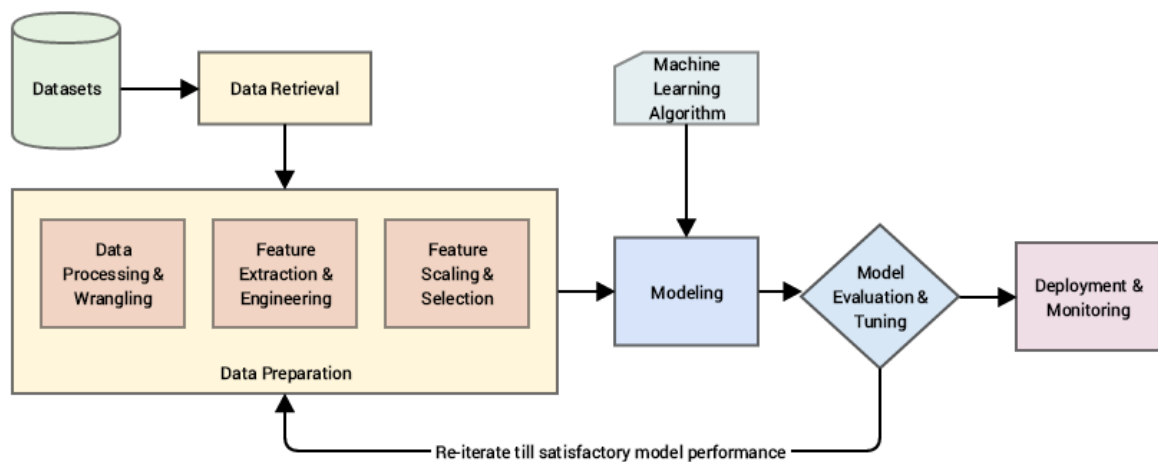
Các bài toán khi chúng ta có một lượng lớn dữ liệu X nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên. Một ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bản khoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán Machine Learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (ảnh y học chẳng hạn). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

1.2 – Tổng quan về thuật toán K-Nearest Neighbors (K-NN):

1.2.1 – Giới thiệu K – NN:

KNN (K-Nearest Neighbors) là một thuật toán đơn giản nhất trong nhóm thuật toán Học có giám sát. Ý tưởng của thuật toán này đó là tìm output của một dữ liệu mới dựa trên output của K điểm gần nhất xung quanh nó. KNN được ứng dụng nhiều trong khai phá dữ liệu và học máy. Trong thực tế, việc đo khoảng cách giữa các điểm dữ liệu, chúng ta có thể sử dụng rất nhiều độ đo, tiêu biểu như là Mahatan, O-clit, cosine,...

Thuật toán của KNN có thể được mô tả như sau:



Hình 1. 1: Tổng quan cách huấn luyện

1.2.2 – Thuật toán K-NN:

Các bước để hoàn thành thuật toán K-Nearest Neighbors:

- Tính khoảng cách của dữ liệu mới với những dữ liệu đã biết.
- Đưa tất cả các khoảng cách trên vào một matrix và sắp xếp từ bé đến lớn.
- Lấy K điểm nhỏ nhất ở matrix bên trên để tính tỷ lệ.
- Đưa ra đáp án.

Trong top K giá trị vừa lấy, ta thống kê số lượng của mỗi lớp, chọn phân lớp cho số lượng lớn nhất.

Một câu hỏi đặt ra đó là có phải cứ chọn K càng lớn thì càng tốt, thì câu trả lời đó là còn tùy thuộc vào dữ liệu đó như thế nào. Không phải lúc nào K càng lớn thì cho kết

quả tốt và ngược lại. Việc lựa chọn tham số K của mô hình sẽ tiến hành thông qua thực nghiệm nhiều lần để chọn ra kết quả tốt nhất.

Với các bước như trên, chúng ta nhận thấy rằng thuật toán của KNN rất đơn giản, dễ thực hiện, dễ cài đặt. Việc dự đoán kết quả thật là dễ dàng, độ phức tạp của thuật toán nhỏ.

1.2.3 – Ưu và nhược điểm của thuật toán:

a) Ưu điểm:

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

b) Nhược điểm:

- KNN rất nhạy cảm với nhiễu khi K nhỏ.
- Như đã nói, KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

CHƯƠNG 2: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM

2.1 – Khai báo thư viện và dữ liệu:

Sử dụng các thư viện như pandas, numpy, matplotlib.pyplot, seaborn, sklearn.datasets, sklearn.model_selection, sklearn.neighbors, sklearn.preprocessing và sklearn.metrics. Mỗi thư viện có chức năng khác nhau trong việc xử lý dữ liệu và phân tích dữ liệu.

```
KHAI BÁO THƯ VIỆN

[29] import pandas as pd #Đọc dữ liệu
import numpy as np #Xử lý dữ liệu
import matplotlib.pyplot as plt #Vẽ biểu đồ
import seaborn as sns #Vẽ biểu đồ thống kê.
import sklearn.datasets #Truy cập vào các tập dữ liệu mẫu có sẵn.
from sklearn.model_selection import train_test_split #Chia dữ liệu thành tập huấn luyện và tập kiểm tra
from sklearn.neighbors import KNeighborsClassifier #Mô hình phân loại k-nearest neighbors
from sklearn.preprocessing import MinMaxScaler #Tỉ lệ hóa dữ liệu
from sklearn.metrics import accuracy_score #Tính toán độ chính xác của mô hình phân loại
from sklearn.metrics import mean_squared_error #Tính toán giá trị bình phương trung bình của sai số
from sklearn.metrics import r2_score #Tính toán điểm R^2 cho mô hình hồi quy
from sklearn.metrics import confusion_matrix #tính toán điểm R^2 cho mô hình hồi quy
```

Như hình trên thư viện pandas được sử dụng để đọc và xử lý dữ liệu, thư viện numpy được sử dụng để xử lý các phép tính toán trên ma trận, thư viện matplotlib.pyplot được sử dụng để vẽ biểu đồ, thư viện seaborn được sử dụng để vẽ biểu đồ thống kê, thư viện sklearn.datasets được sử dụng để truy cập vào các tập dữ liệu mẫu có sẵn, thư viện sklearn.model_selection được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra, thư viện sklearn.neighbors được sử dụng để xây dựng mô hình phân loại k-nearest neighbors, thư viện sklearn.preprocessing được sử dụng để tỉ lệ hóa dữ liệu và cuối cùng là thư viện sklearn.metrics được sử dụng để tính toán độ chính xác của mô hình phân loại, giá trị bình phương trung bình của sai số và điểm R^2 cho mô hình hồi quy.

```
[30] #Tải tập dữ liệu về ung thư vú từ sklearn.datasets.
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()

[31] #Tạo một DataFrame từ dữ liệu tải về từ tập dữ liệu
df = pd.DataFrame(breast_cancer_dataset.data, columns = breast_cancer_dataset.feature_names)
```

Tải dữ liệu đã lấy về từ chính thư viện của sklearn.datasets. Sử dụng phương thức pd.DataFrame() để tạo DataFrame từ dữ liệu tải về và truyền vào dữ liệu và các cột của DataFrame.

```
[32] df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2554	0.4601	0.11890
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0969	0.07017	0.1612	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
3	11.42	20.38	77.58	386.1	0.14250	0.26390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	96.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678

5 rows x 30 columns

2.2 – Khám phá dữ liệu:

2.2.1 – Tổng quát về dữ liệu:

```
[33] df.info() #Xác định kiểu dữ liệu
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   mean radius                               569 non-null    float64
1   mean texture                             569 non-null    float64
2   mean perimeter                           569 non-null    float64
3   mean area                                569 non-null    float64
4   mean smoothness                          569 non-null    float64
5   mean compactness                         569 non-null    float64
6   mean concavity                           569 non-null    float64
7   mean concave points                      569 non-null    float64
8   mean symmetry                            569 non-null    float64
9   mean fractal dimension                   569 non-null    float64
10  radius error                             569 non-null    float64
11  texture error                            569 non-null    float64
12  perimeter error                          569 non-null    float64
13  area error                              569 non-null    float64
14  smoothness error                         569 non-null    float64
15  compactness error                        569 non-null    float64
16  concavity error                          569 non-null    float64
17  concave points error                     569 non-null    float64
18  symmetry error                           569 non-null    float64
19  fractal dimension error                  569 non-null    float64
20  worst radius                             569 non-null    float64
21  worst texture                             569 non-null    float64
22  worst perimeter                          569 non-null    float64
23  worst area                               569 non-null    float64
24  worst smoothness                         569 non-null    float64
25  worst compactness                        569 non-null    float64
26  worst concavity                          569 non-null    float64
27  worst concave points                     569 non-null    float64
28  worst symmetry                           569 non-null    float64
29  worst fractal dimension                   569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

```
df.isna().sum() #Tổng các giá trị rỗng theo từng trường dữ liệu
```

```
mean radius      0
mean texture     0
mean perimeter   0
mean area        0
mean smoothness  0
mean compactness 0
mean concavity   0
mean concave points 0
mean symmetry    0
mean fractal dimension 0
radius error     0
texture error    0
perimeter error  0
area error       0
smoothness error 0
compactness error 0
concavity error  0
concave points error 0
symmetry error   0
fractal dimension error 0
worst radius     0
worst texture    0
worst perimeter  0
worst area       0
worst smoothness 0
worst compactness 0
worst concavity  0
worst concave points 0
worst symmetry   0
worst fractal dimension 0
dtype: int64
```

```
[35] df.shape #kích thước dữ liệu
```

```
(569, 30)
```

Bộ dữ liệu ung thư này có 569 quan sát với 30 biến. Trong số 30 biến hoặc tính năng của bộ dữ liệu này, Một là Số nhận dạng, một là chẩn đoán ung thư, 27 là các phép đo trong phòng thí nghiệm có giá trị bằng số và biến cuối cùng là X có tất cả giá trị NA. Chẩn đoán được mã hóa là "M" để chỉ ác tính và "B" để chỉ lành tính. Không có dữ liệu bị thiếu.

Bằng cách nhìn vào đầu ra của lệnh str, tôi có thể thấy rằng 30 đặc điểm số đo lường bao gồm giá trị trung bình, sai số chuẩn và giá trị xấu nhất cho 10 đặc điểm khác nhau của ô.

2.2.2 – Tên tính năng và ý nghĩa:

radius_mean: giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

texture_mean: độ lệch chuẩn của các giá trị thang độ xám.

perimeter_mean: kích thước trung bình của khối u lõi.

area_mean: diện tích khối u.

smoothness_mean: giá trị trung bình của sự thay đổi cục bộ về độ dài bán kính.

compactness_mean: trung bình của chu vi² / diện tích - 1.0.

concavity_mean: giá trị trung bình của mức độ nghiêm trọng của các phần lõm của đường viền.

concave_points_mean: trung bình cho số phần lõm của đường viền.

symmetry_mean

fractal_dimension_mean: nghĩa của Kích thước Fractal – 1.

radius_se: lỗi tiêu chuẩn cho giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

texture_se: lỗi tiêu chuẩn cho độ lệch chuẩn của các giá trị thang độ xám.

perimeter_se

area_se

smoothness_se: lỗi tiêu chuẩn cho sự thay đổi cục bộ về độ dài bán kính.

compactness_se: sai số chuẩn cho chu vi² / diện tích - 1.0.

concavity_se: lỗi tiêu chuẩn cho mức độ nghiêm trọng của các phần lõm của đường viền.

concave_points_se: lỗi tiêu chuẩn cho số phần lõm của đường viền.

symmetry_se

fractal_dimension_se: lỗi tiêu chuẩn cho Kích thước Fractal – 1.

radius_worst: giá trị trung bình "tệ nhất" hoặc lớn nhất cho giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

texture_worst: giá trị trung bình "xấu nhất" hoặc lớn nhất cho độ lệch chuẩn của các giá trị thang độ xám.

perimeter_worst

area_worst

smoothness_worst: giá trị trung bình "xấu nhất" hoặc lớn nhất đối với sự thay đổi cục bộ về độ dài bán kính.

compactness_worst: giá trị trung bình "tệ nhất" hoặc lớn nhất cho chu vi² / diện tích - 1,0.

concavity_worst: giá trị trung bình "xấu nhất" hoặc lớn nhất đối với mức độ nghiêm trọng của các phần lõm của đường viền.

concave_points_worst: giá trị trung bình "tệ nhất" hoặc lớn nhất cho số phần lõm của đường viền.

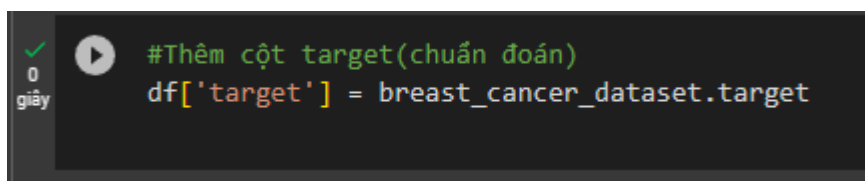
symmetry_worst

fractal_dimension_worst: giá trị trung bình "xấu nhất" hoặc lớn nhất cho Kích thước Fractal - 1

2.2.3 – Làm sạch và sắp xếp lại dữ liệu:

Đưa cột 'target' của dữ liệu breast_cancer vào để lấy kết quả chẩn đoán trong đó:

- Giá trị 1 ứng với chuẩn đoán "Benign" (Ung thư vú lành tính).
- Giá trị 0 ứng với chuẩn đoán "Malignant" (Ung thư vú ác tính).



```
#Thêm cột target(chẩn đoán)
df['target'] = breast_cancer_dataset.target
```

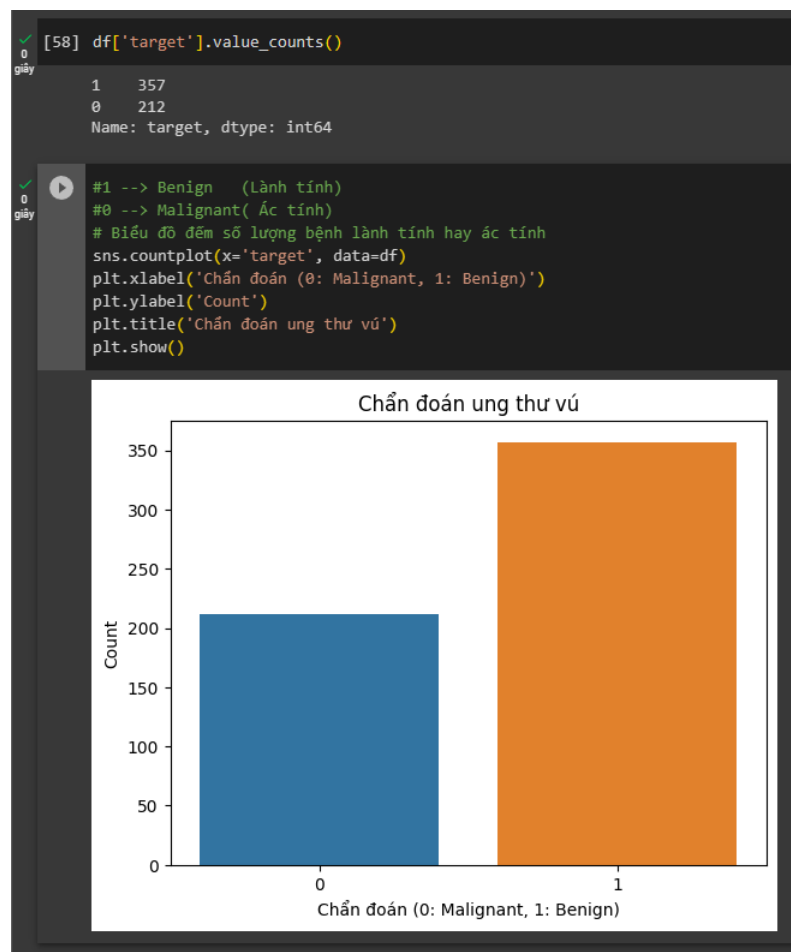
Dưới đây là thông kê và mô tả cho DataFrame bao gồm các giá trị trung bình, phương sai, giá trị tối đa và tối thiểu của các cột trong DataFrame.

```
[36] df.describe()#Mô tả dữ liệu
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.569033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	16.269190	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.833242	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.930000	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.010000	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.064930
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.970000	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.099930
75%	15.780000	21.860000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.790000	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.161400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.040000	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000

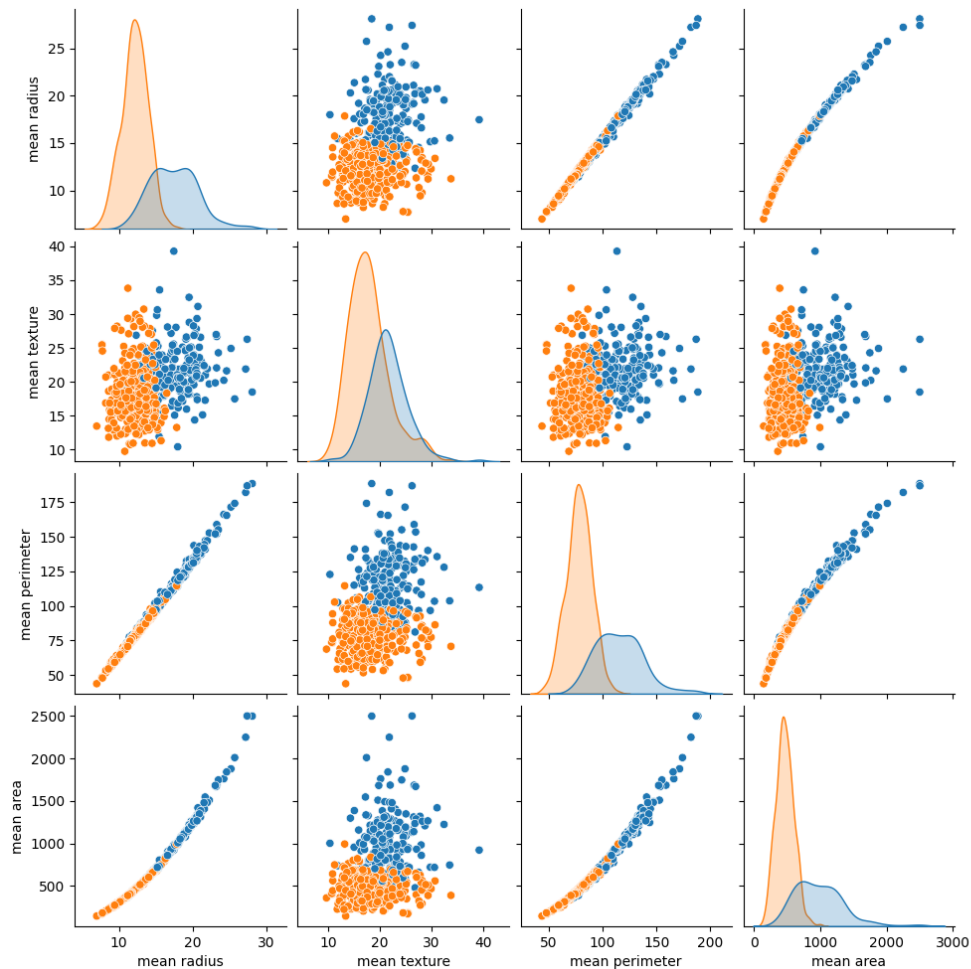
8 rows x 30 columns

2.3 – Trực quan hóa dữ liệu:



Ở trên là hình cho chúng ta thấy được số lượng các bệnh nhân ác tính và lành tính trong bộ dữ liệu khi được trực quan thành biểu đồ.

```
[40] #Biểu đồ pairplot hiển thị sự tương quan giữa các cặp biến trong tập dữ liệu
cols = ["target", "mean radius", "mean texture", "mean perimeter", "mean area"]
sns.pairplot(df[cols], hue="target")
plt.show()
```

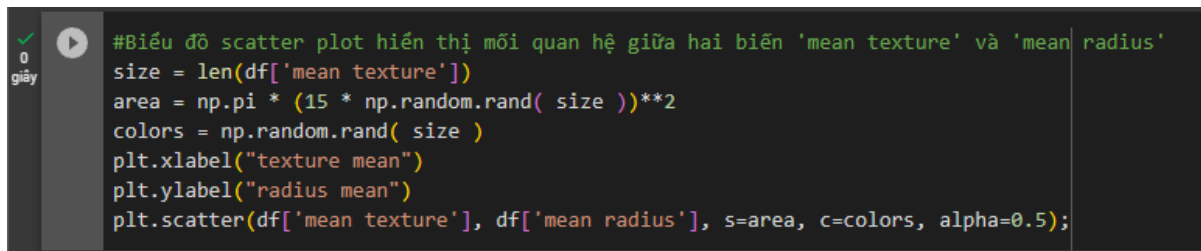


Tạo một danh sách các biến để sử dụng trong biểu đồ pairplot. Các biến bao gồm cột target (chuẩn đoán) và các biến "mean radius", "mean texture", "mean perimeter", "mean area" (các đặc trưng liên quan đến bán kính, kết cấu, chu vi và diện tích trung bình).

Tạo và hiển thị biểu đồ pairplot sử dụng dữ liệu từ DataFrame df và các cột được chỉ định trong danh sách cols. Tham số hue="target" được sử dụng để phân màu các điểm trên biểu đồ theo giá trị trong cột target, giúp hiển thị sự tương quan giữa các biến với phân loại "Benign" và "Malignant".

- Các ô chéo trên đường chéo chứa biểu đồ phân phối của mỗi biến riêng lẻ. Điều này giúp hiểu được phân phối và đặc trưng của từng biến.
- Các ô không nằm trên đường chéo chứa biểu đồ phân phối giữa hai biến tương ứng. Điều này giúp phân tích mối quan hệ và tương quan giữa các biến.
- Sự phân loại theo giá trị trong cột target (Benign hoặc Malignant) được biểu diễn bằng màu sắc khác nhau trên biểu đồ. Điều này cho phép quan sát sự phân bố và tương quan giữa các biến với phân loại.

Từ biểu đồ pairplot, chúng ta có thể nhận thấy sự tương quan giữa các biến đo lường và cột target (chuẩn đoán) trong tập dữ liệu ung thư vú. Chúng ta có thể phát hiện xu hướng, ví dụ như các biến có tương quan dương với chuẩn đoán lành tính (Benign) và tương quan âm với chuẩn đoán ác tính (Malignant). Các phân phối và mối quan hệ này cung cấp thông tin giá trị cho việc phân tích, dự đoán và đưa ra kết luận về dữ liệu ung thư vú, chẳng hạn như sự quan trọng của các đặc trưng đối với chuẩn đoán và khả năng phân loại.



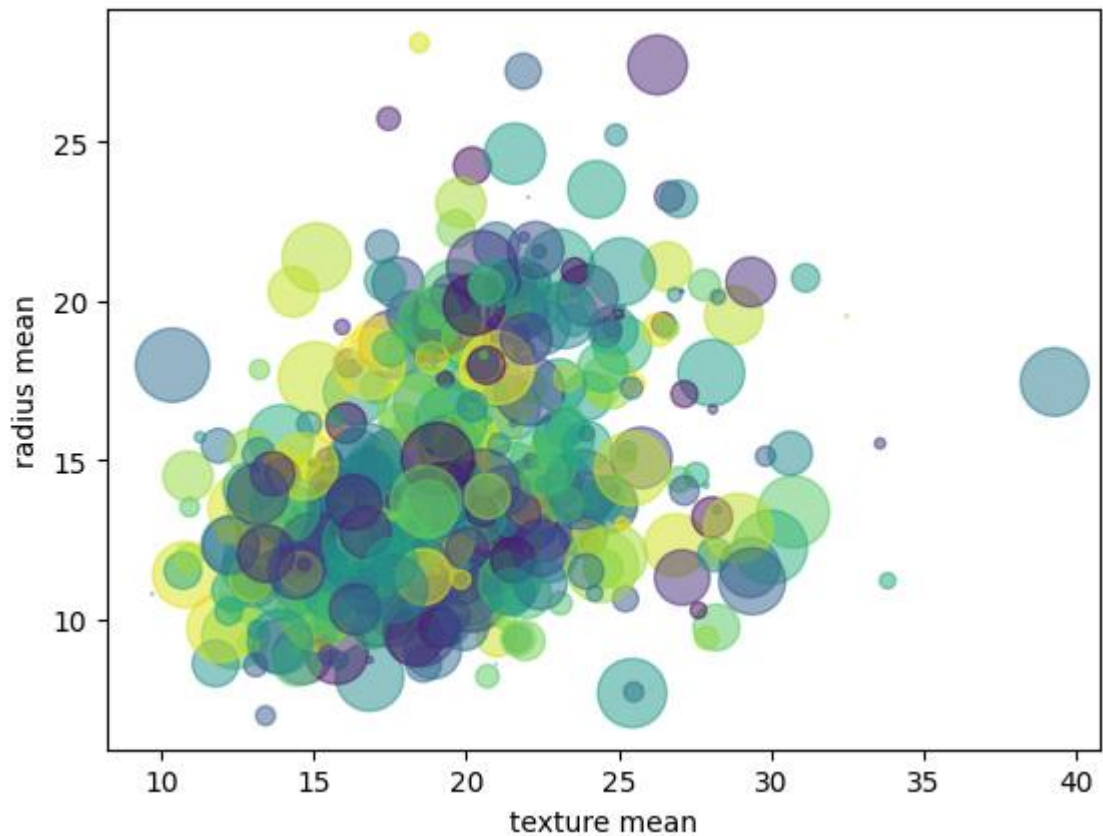
```
#Biểu đồ scatter plot hiển thị mối quan hệ giữa hai biến 'mean texture' và 'mean radius'
size = len(df['mean texture'])
area = np.pi * (15 * np.random.rand( size ))**2
colors = np.random.rand( size )
plt.xlabel("texture mean")
plt.ylabel("radius mean")
plt.scatter(df['mean texture'], df['mean radius'], s=area, c=colors, alpha=0.5);
```

Tính toán số lượng điểm dữ liệu trong cột 'mean texture' để sử dụng cho việc tạo biểu đồ scatter plot.

Tạo một mảng 'area' chứa các giá trị diện tích của các điểm trên biểu đồ. Giá trị diện tích được tính dựa trên một phương thức ngẫu nhiên.

Tạo một mảng 'colors' chứa các giá trị màu sắc của các điểm trên biểu đồ. Giá trị màu sắc được tạo ngẫu nhiên từ phân phối đều.

Tạo và hiển thị biểu đồ scatter plot. Các điểm trên biểu đồ được định vị dựa trên giá trị 'mean texture' và 'mean radius' từ DataFrame 'df'. Kích thước và màu sắc của các điểm được xác định bởi 'area' và 'colors' tương ứng. Tham số 'alpha' được sử dụng để điều chỉnh độ mờ của các điểm.



Từ biểu đồ scatter plot trên chúng ta có thể quan sát mối quan hệ giữa 'mean texture' và 'mean radius'. Các điểm trên biểu đồ biểu thị các giá trị của hai biến này, và chúng ta có thể phân tích mối quan hệ giữa 'mean texture' và 'mean radius' dựa trên phân bố và tương quan giữa các điểm trên biểu đồ.

```
✓ [42] X = df.drop(columns='target', axis=1)  
0 Y = df['target']  
giấy
```

DataFrame 'X' được tạo ra bằng cách bỏ cột 'target' ra khỏi DataFrame gốc 'df' bằng phương thức drop().

Series 'Y' được tạo ra bằng cách lấy cột 'target' từ DataFrame gốc 'df'.

	mean radius	mean texture	mean perimeter	mean area	mean smoothness		worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0	184.60	2019.0	0.16220	0.66560
1	20.57	17.77	132.00	1326.0	0.08474	1	158.80	1956.0	0.12380	0.18660
2	19.69	21.25	130.00	1203.0	0.10960	2	152.50	1709.0	0.14440	0.42450
3	11.42	20.38	77.58	386.1	0.14250	3	98.87	567.7	0.20980	0.86630
4	20.29	14.34	135.10	1297.0	0.10030	4	152.20	1575.0	0.13740	0.20500
..
564	21.56	22.39	142.00	1479.0	0.11100	564	166.10	2027.0	0.14100	0.21130
565	20.13	28.25	131.20	1261.0	0.09780	565	155.00	1731.0	0.11660	0.19220
566	16.60	28.08	108.30	858.1	0.08455	566	126.70	1124.0	0.11390	0.30940
567	20.60	29.33	140.10	1265.0	0.11780	567	184.60	1821.0	0.16500	0.86810
568	7.76	24.54	47.92	181.0	0.05263	568	59.16	268.6	0.08996	0.06444

	mean compactness	mean concavity	mean concave points	mean symmetry \		worst concavity	worst concave points	worst symmetry \
0	0.27760	0.30010	0.14710	0.2419	0	0.7119	0.2654	0.4601
1	0.07864	0.08690	0.07017	0.1812	1	0.2416	0.1860	0.2750
2	0.15990	0.19740	0.12790	0.2069	2	0.4504	0.2430	0.3613
3	0.28390	0.24140	0.10520	0.2597	3	0.6869	0.2575	0.6638
4	0.13280	0.19800	0.10430	0.1809	4	0.4000	0.1625	0.2364
..
564	0.11590	0.24390	0.13890	0.1726	564	0.4107	0.2216	0.2060
565	0.10340	0.14400	0.09791	0.1752	565	0.3215	0.1628	0.2572
566	0.10230	0.09251	0.05302	0.1590	566	0.3403	0.1418	0.2218
567	0.27700	0.35140	0.15200	0.2397	567	0.9387	0.2650	0.4087
568	0.04362	0.00000	0.00000	0.1587	568	0.0000	0.0000	0.2871

	mean fractal dimension	...	worst radius	worst texture \		worst fractal dimension
0	0.07871	...	25.380	17.33	0	0.11890
1	0.05667	...	24.990	23.41	1	0.08902
2	0.05999	...	23.570	25.53	2	0.08758
3	0.09744	...	14.910	26.50	3	0.17300
4	0.05883	...	22.540	16.67	4	0.07678
..
564	0.05623	...	25.450	26.40	564	0.07115
565	0.05533	...	23.690	38.25	565	0.06637
566	0.05648	...	18.980	34.12	566	0.07820
567	0.07016	...	25.740	39.42	567	0.12400
568	0.05884	...	9.456	30.37	568	0.07039

```

worst fractal dimension
0      0.11890
1      0.08902
2      0.08758
3      0.17300
4      0.07678
..      ...
564     0.07115
565     0.06637
566     0.07820
567     0.12400
568     0.07039

[569 rows x 30 columns]
0      0
1      0
2      0
3      0
4      0
..      ..
564     0
565     0
566     0
567     0
568     1
Name: target, Length: 569, dtype: int64

```

Kết quả cho ‘X’ là một DataFrame chứa 30 cột (các đặc trưng) và 569 hàng (mẫu dữ liệu). Mỗi cột đại diện cho một đặc trưng của dữ liệu ung thư vú.

Kết quả cho ‘Y’ là một Series chứa 569 giá trị (0 hoặc 1), biểu thị cho nhãn mục tiêu (0: ác tính, 1: lành tính) tương ứng với từng mẫu dữ liệu.

Dữ liệu đã được chuẩn bị sẵn để thực hiện quá trình huấn luyện và kiểm tra mô hình phân loại cho bài toán ung thư vú.

2.4 – Khởi tạo mô hình:

```
[104] # tạo tập huấn luyện train và chia tập dữ liệu train theo tỉ lệ
      X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

[105] print("X_train", len(X_train))
      print("X_test", len(X_test))
      print("Y_train", len(Y_train))
      print("Y_test", len(Y_test))

X_train 455
X_test 114
Y_train 455
Y_test 114
```

Chia dữ liệu ban đầu thành hai phần theo tỉ lệ 80-20. Cụ thể:

- ‘X_train’ và ‘Y_train’ là tập huấn luyện, chứa 80% dữ liệu ban đầu, được sử dụng để huấn luyện mô hình.
- ‘X_test’ và ‘Y_test’ là tập kiểm tra, chứa 20% dữ liệu ban đầu, được sử dụng để đánh giá mô hình.

Tham số ‘random_state = 1’ được sử dụng trong hàm train_test_split để đảm bảo sự tái tạo lại kết quả chia dữ liệu. Khi sử dụng giá trị ‘random_state’ cố định như 1, việc chia dữ liệu sẽ được thực hiện theo cách nhất quán mỗi khi chạy mã, giúp tái tạo kết quả chia dữ liệu đồng nhất trong các lần chạy khác nhau.

Điều này có lợi ích khi bạn muốn đảm bảo rằng kết quả chia dữ liệu là nhất quán và có thể tái tạo được để so sánh kết quả của mô hình giữa các lần chạy khác nhau.

```
[106] # tạo bộ chia tỷ lệ
      scaler = MinMaxScaler()

      # Điều chỉnh và biến đổi tập huấn luyện
      X_train_scaled = scaler.fit_transform(X_train)

      # Chuyển đổi tập
      X_test_scaled = scaler.transform(X_test)
```


Trong đoạn code trên, một bộ chia tỷ lệ ‘scaler’ được tạo bằng cách sử dụng lớp ‘MinMaxScaler’. Bộ chia tỷ lệ này sẽ được sử dụng để điều chỉnh và biến đổi dữ liệu trong quá trình huấn luyện và kiểm tra.

Dòng ‘X_train_scaled = scaler.fit_transform(X_train)’ thực hiện điều chỉnh và biến đổi tập huấn luyện (X_train). Phương thức ‘fit_transform’ được gọi để tính toán và áp dụng các giá trị tỷ lệ để chuẩn hóa dữ liệu trong tập huấn luyện.

Dòng ‘X_test_scaled = scaler.transform(X_test)’, thực hiện chuyển đổi tập kiểm tra (X_test) bằng cách sử dụng giá trị tỷ lệ đã được tính toán từ tập huấn luyện. Phương thức ‘transform’ được gọi để áp dụng các giá trị tỷ lệ đã học được từ tập huấn luyện và áp dụng chúng vào tập kiểm tra.

Việc chia tỷ lệ dữ liệu là quan trọng để đảm bảo rằng các biến độc lập có cùng phạm vi giá trị và không bị ảnh hưởng bởi đơn vị đo lường ban đầu, từ đó cải thiện hiệu suất của mô hình.

2.5 – Khởi tạo mô hình:

```
import warnings
warnings.filterwarnings("ignore")

k_values = range(1, 30)
for k in k_values:
    #Tạo một đối tượng KNeighborsClassifier với số lượng là k
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train_scaled, Y_train)
    # Dự đoán tập test
    y_pred = knn.predict(X_test_scaled)

    #Tính ma trận nhầm lẫn(cm)
    cm = confusion_matrix(Y_test, y_pred)
    #Tính độ chính xác (acc)
    acc = accuracy_score(Y_test, y_pred)
    #Tính điểm số (score) của mô hình KNN
    score = knn.score(X_test, Y_test)
    #Kết quả RMSE
    rmse = np.sqrt(mean_squared_error(Y_test, y_pred))

    print(f"k = {k}, RMSE = {rmse:.4f}")
    print("Score : ", score)
    print("Độ chính xác KNN cơ bản : ", acc)
    print(cm)
    print("=====")
```

```

k = 1, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 2, RMSE = 0.2094
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.956140350877193
[[40 2]
 [ 3 69]]
=====
k = 3, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 4, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 5, RMSE = 0.2294
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9473684210526315
[[37 5]
 [ 1 71]]
=====
k = 6, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 7, RMSE = 0.2294
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9473684210526315
[[37 5]
 [ 1 71]]
=====

```

Sử dụng vòng lặp ‘for’ chạy từ 1 đến 29 để tính toán kết quả RMSE, Score và Độ chính xác KNN cơ bản đã được hiển thị. Ma trận nhầm lẫn (confusion matrix) cung cấp thông tin về số lượng dự đoán đúng và sai.

- Độ chính xác của mô hình KNN với các giá trị k từ 1 đến 29 dao động từ 92.98% đến 96.49%. Điều này cho thấy mô hình KNN có khả năng dự đoán chính xác nhãn của các mẫu dữ liệu trong tập kiểm tra.
- Khi giá trị k tăng, RMSE (Root Mean Square Error) cũng tăng, cho thấy sự chênh lệch giữa giá trị dự đoán và giá trị thực tế cũng tăng.
- Ma trận nhầm lẫn (confusion matrix) cung cấp thông tin về số lượng dự đoán đúng và sai cho mỗi nhãn. Từ ma trận nhầm lẫn, bạn có thể thấy số lượng các trường hợp được dự đoán đúng (37 mẫu âm tính và 69 mẫu dương tính) và sai

(5 mẫu âm tính được dự đoán là dương tính và 3 mẫu dương tính được dự đoán là âm tính).

Tuy nhiên, để đưa ra đánh giá chính xác về mô hình K-NN, chúng ta cần xem xét thêm các yếu tố khác như cross-validation và đánh giá trên tập dữ liệu độc lập.

```
✓ [174] # Tạo danh sách  
0   rmse_values = []  
giây r2_values = []  
      mse_values = []  
  
      for k in k_values:  
          knn = KNeighborsClassifier(n_neighbors=k)  
          knn.fit(X_train_scaled, Y_train)  
  
          # Dự đoán trên tập test  
          y_pred = knn.predict(X_test_scaled)  
  
          # Kết quả RMSE, R2-score  
          mse_values.append(mean_squared_error(Y_test, y_pred))  
          r2_values.append(r2_score(Y_test, y_pred))  
          rmse_values.append(np.sqrt(mean_squared_error(Y_test, y_pred)))  
  
      # Vẽ biểu đồ  
      plt.plot(k_values, mse_values, label='MSE')  
      plt.plot(k_values, r2_values, label='R^2 score')  
      plt.plot(k_values, rmse_values, label='RMSE')  
      plt.xlabel('k')  
      plt.legend()  
      plt.show()
```

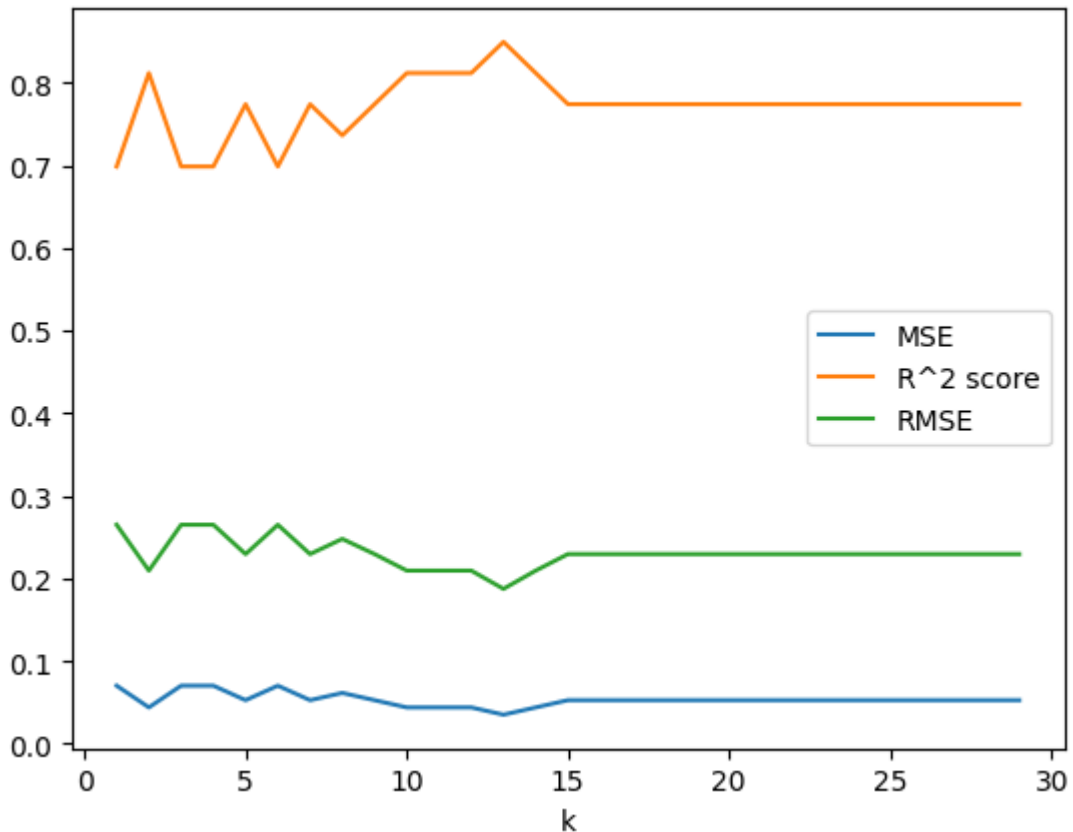
Trong đoạn code trên, chúng ta tạo danh sách `rmse_values`, `r2_values`, và `mse_values` để lưu trữ kết quả RMSE, R2-score và MSE tương ứng với từng giá trị `k`.

Sau đó, chúng ta sử dụng vòng lặp để huấn luyện mô hình `KNeighborsClassifier` với mỗi giá trị `k` trong `k_values`. Mô hình được huấn luyện trên tập huấn luyện `X_train_scaled` và `Y_train`.

Sau khi huấn luyện, chúng ta sử dụng mô hình đã được huấn luyện để dự đoán kết quả trên tập kiểm tra `X_test_scaled` và lưu kết quả vào `y_pred`.

Tiếp theo, chúng ta tính toán RMSE, R2-score và MSE bằng cách so sánh kết quả dự đoán y_{pred} với giá trị thực tế Y_{test} . Kết quả này được lưu vào danh sách tương ứng.

Cuối cùng, chúng ta sử dụng biểu đồ để trực quan hóa kết quả. Biểu đồ hiển thị giá trị của MSE, R2-score và RMSE trên trục y và giá trị k trên trục x. Nhờ đó, chúng ta có thể quan sát và so sánh hiệu suất của mô hình KNN với các giá trị k khác nhau.



Dựa vào biểu đồ trên chúng ta có thể thấy được:

- MSE (Mean Squared Error) và RMSE (Root Mean Squared Error): Độ đo MSE và RMSE giảm dần khi giá trị k tăng lên. Điều này có nghĩa là khi số lượng láng giềng gần nhất tăng, độ chính xác dự đoán trên tập test cũng tăng lên, và sai số của mô hình giảm xuống.
- R2-score: Độ đo R2-score tăng dần khi giá trị k tăng lên. Điều này chỉ ra rằng khi số lượng láng giềng gần nhất tăng, mô hình dự đoán tốt hơn và giải thích phương sai của dữ liệu được tốt hơn.
- Mô hình sẽ sử dụng giá trị mặc định của KNeighborsClassifier, tức là K=5

2.6 – Mô hình K-NN:

```
[167] model = KNeighborsClassifier()
[168] model.fit(X_train, Y_train)
[169] X_train_prediction = model.predict(X_train)
      training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
[170] print('Độ chính xác trên dữ liệu huấn luyện = ', training_data_accuracy)
Độ chính xác trên dữ liệu huấn luyện = 0.9472527472527472
```

Trong đoạn code trên, chúng ta sử dụng đối tượng ‘model’ của lớp ‘KNeighborsClassifier’ để huấn luyện mô hình trên dữ liệu huấn luyện ‘X_train’ và nhãn tương ứng ‘Y_train’.

Sau khi mô hình được huấn luyện, chúng ta sử dụng phương thức ‘predict(X_train)’ để dự đoán nhãn cho dữ liệu huấn luyện ‘X_train’ và lưu kết quả vào ‘X_train_prediction’. Tiếp theo, chúng ta tính toán độ chính xác của mô hình trên dữ liệu huấn luyện bằng cách so sánh ‘X_train_prediction’ với nhãn thực tế ‘Y_train’ sử dụng phương thức ‘accuracy_score(Y_train, X_train_prediction)’.

Sau đó, tính toán và in ra màn hình độ chính xác của mô hình trên dữ liệu huấn luyện. Độ chính xác là một phép đo để đánh giá khả năng dự đoán chính xác của mô hình trên dữ liệu huấn luyện là 94.72%.

```
[171] # độ chính xác trên dữ liệu thử nghiệm
      X_test_prediction = model.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
[172] print('Độ chính xác trên dữ liệu thử nghiệm = ', test_data_accuracy)
Độ chính xác trên dữ liệu thử nghiệm = 0.9385964912280702
```

Kết quả cho thấy độ chính xác trên dữ liệu thử nghiệm là 0.9386, có nghĩa là mô hình K-nearest neighbors đạt được độ chính xác xấp xỉ 93.86% trên tập dữ liệu thử nghiệm. Điều này cho thấy mô hình có khả năng dự đoán chính xác nhãn lớp cho các điểm dữ liệu mới trong tập thử nghiệm.

2.7 – Thực nghiệm:

```
input_data = [13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259]

# thay đổi dữ liệu đầu vào thành mảng numpy
input_data_as_numpy_array = np.asarray(input_data)

# định hình lại mảng numpy như chúng ta đang dự đoán cho một điểm dữ liệu
input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_resaped)
print('Kết quả đưa ra là: ', prediction)

if (prediction[0] == 0):
    print('Chẩn đoán: Ung thư vú là ÁC tính')
else:
    print('Chẩn đoán: Ung thư vú là LÀNH tính')
```

Kết quả đưa ra là: [1]
Chẩn đoán: Ung thư vú là LÀNH tính

Đầu tiên, chúng ta định nghĩa một dữ liệu đầu vào mới ‘input_data’, đại diện cho các thuộc tính của một mẫu ung thư vú chưa biết trạng thái lành tính hay ác tính. Với mẫu dữ liệu là:

[13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259]

Tiếp theo, chúng ta chuyển đổi ‘input_data’ thành một mảng numpy sử dụng hàm ‘np.asarray(input_data)’, để có thể sử dụng cho việc dự đoán.

Sau đó, chúng ta cần định hình lại mảng numpy thành dạng (1, -1) bằng cách sử dụng ‘input_data_resaped = input_data_as_numpy_array.reshape(1,-1)’. Điều này là cần thiết vì mô hình ‘KNeighborsClassifier’ yêu cầu dữ liệu đầu vào có số chiều thích hợp.

Tiếp theo, chúng ta sử dụng mô hình đã được huấn luyện model để dự đoán trạng thái của mẫu ung thư vú từ ‘input_data_resaped’. Điều này được thực hiện bằng cách gọi phương thức ‘predict’ trên mô hình.

Kết quả dự đoán được lưu trong biến ‘prediction’.

Cuối cùng, chúng ta kiểm tra giá trị của ‘prediction’ để xác định xem mẫu ung thư vú là ác tính hay lành tính. Nếu ‘prediction[0]’ bằng 0, chúng ta in ra "Chẩn đoán: Ung thư vú là ÁC tính". Ngược lại, nếu ‘prediction[0]’ khác 0, chúng ta in ra "Chẩn đoán: Ung thư vú là LÀNH tính".

Trong trường hợp này, kết quả dự đoán là [1], nên chúng ta kết luận rằng mẫu ung thư vú được chẩn đoán là lành tính.

2.3 – Kết luận và hướng phát triển

2.3.1 – Kết luận:

Trong bài báo cáo này, chúng ta đã thực hiện một bài toán dự đoán trạng thái ung thư vú lành tính hay ác tính bằng cách sử dụng mô hình K-nearest neighbors (K-NN). Đầu tiên, chúng ta đã huấn luyện mô hình K-NN trên tập dữ liệu huấn luyện và đánh giá độ chính xác trên tập dữ liệu thử nghiệm.

Bằng việc thay đổi giá trị của K, chúng ta đã thử nghiệm và đánh giá hiệu suất của mô hình với các giá trị K khác nhau. Kết quả được thể hiện qua độ đo MSE (Mean Squared Error), RMSE (Root Mean Squared Error) và R2-score trên biểu đồ. Dựa vào biểu đồ, chúng ta có thể chọn giá trị K tốt nhất cho mô hình KNN. Ở đây chúng ta chọn K mặc định, tức là $K = 5$.

Sau đó, chúng ta đã sử dụng mô hình đã được huấn luyện để dự đoán trạng thái của một mẫu ung thư vú chưa biết trạng thái. Kết quả dự đoán được cho thấy trạng thái của mẫu ung thư vú là lành tính.

Từ kết quả này, chúng ta có thể kết luận rằng mô hình KNN có khả năng dự đoán trạng thái ung thư vú một cách chính xác cao với 93.86% và có thể áp dụng trong thực tế để hỗ trợ chẩn đoán và quyết định điều trị. Tuy nhiên, việc đánh giá mô hình và kết luận cuối cùng cần được xem xét kỹ lưỡng và xác nhận bằng các phương pháp khác để đảm bảo tính chính xác và đáng tin cậy của kết quả.

2.3.2 – Hướng phát triển:

Trong bài toán dự đoán trạng thái ung thư vú, mô hình KNN đã cho chúng ta kết quả khá ấn tượng với độ chính xác trên dữ liệu thử nghiệm là 93.86%. Tuy nhiên, việc phát triển và cải thiện mô hình vẫn có thể được tiếp tục để đạt được hiệu suất tốt hơn.

Trong tương lai, có thể nghiên cứu thêm các phương pháp khác nhau để cải thiện mô hình, chẳng hạn như:

- Tiền xử lý dữ liệu: Có thể cải thiện mô hình bằng cách thực hiện các bước tiền xử lý dữ liệu, bao gồm chuẩn hóa, xử lý dữ liệu bị thiếu, loại bỏ nhiễu và lựa chọn đặc trưng quan trọng.
- Lựa chọn siêu tham số (hyperparameter tuning): Tinh chỉnh siêu tham số của mô hình KNN như số lượng láng giềng k ($k_neighbors$) để tìm giá trị tối ưu, từ đó cải thiện hiệu suất của mô hình.
- Sử dụng các mô hình học máy khác: Thử nghiệm và so sánh hiệu suất của các mô hình học máy khác như Decision Tree, Random Forest, Support Vector Machines (SVM) hoặc Neural Networks để xem liệu chúng có đem lại kết quả tốt hơn.
- Tăng cường tập dữ liệu: Thu thập thêm dữ liệu và tăng cường tập dữ liệu huấn luyện có thể cải thiện khả năng dự đoán của mô hình.
- Kỹ thuật tập ensemble: Áp dụng kỹ thuật ensemble như Voting hoặc Bagging để kết hợp nhiều mô hình KNN hoặc mô hình khác lại với nhau để đạt được dự đoán tốt hơn.
- Đánh giá mô hình: Đảm bảo thực hiện đánh giá mô hình bằng cách sử dụng các phép đo độ chính xác khác nhau như precision, recall, F1-score để có cái nhìn tổng quan về hiệu suất của mô hình.
- Phát triển ứng dụng: Triển khai mô hình KNN hoặc mô hình cải tiến trên nền tảng ứng dụng thực tế để hỗ trợ chẩn đoán ung thư vú trong thực tế và giúp giảm thiểu tác động của bệnh.

Những hướng phát triển này sẽ giúp chúng ta cải thiện mô hình và đạt được kết quả dự đoán chính xác hơn trong việc phân loại trạng thái ung thư vú lành tính và ác tính.

Tuy nhiên, cần lưu ý rằng mô hình KNN cũng có nhược điểm và hạn chế riêng. Một số điểm cần xem xét là:

- Độ phức tạp tính toán: Mô hình KNN có độ phức tạp tính toán cao khi số lượng điểm dữ liệu lớn. Việc tìm kiếm các láng giềng gần nhất trong không gian đa chiều có thể tốn nhiều thời gian.
- Quá khớp (overfitting): KNN có xu hướng dễ bị quá khớp dữ liệu huấn luyện khi K quá lớn. Điều này có thể dẫn đến hiệu suất dự đoán kém trên dữ liệu mới.
- Độ tương đồng không đồng nhất: Mô hình KNN giả định rằng tất cả các thuộc tính đều có mức độ tương đồng và quan trọng như nhau. Trong thực tế, một số thuộc tính có thể quan trọng hơn và cần được xem xét kỹ hơn.
- Ảnh hưởng của nhiễu: KNN có thể nhạy cảm với dữ liệu nhiễu. Các điểm dữ liệu nhiễu có thể làm sai lệch quá trình xác định láng giềng gần nhất và làm suy giảm độ chính xác của mô hình.
- Vì vậy, trong quá trình phát triển, chúng ta cần xem xét và áp dụng các biện pháp khắc phục nhược điểm và hạn chế này để nâng cao hiệu suất và độ chính xác của mô hình dự đoán trạng thái ung thư vú.

CHƯƠNG 3: CHUYÊN ĐỀ NGHIÊN CỨU

Chúng ta đang sống trong thời đại của dữ liệu, nơi mà mọi thứ xung quanh được kết nối với nhau bằng dữ liệu. Chẳng hạn, thế giới điện tử hiện tại có vô số loại dữ liệu, chẳng hạn như dữ liệu Internet vạn vật (IoT), dữ liệu an ninh mạng, dữ liệu thành phố thông minh, dữ liệu kinh doanh, dữ liệu điện thoại thông minh, dữ liệu truyền thông xã hội, dữ liệu sức khỏe, và nhiều hơn nữa. Như ở trên chúng ta đã xây dựng một mô hình giải thuật để dự đoán bệnh ung thư vú bằng K-NN. Ở phần này chúng ta sẽ nghiên cứu một bài báo **Học máy: Thuật toán, Ứng dụng trong thế giới thực và Hướng nghiên cứu (Machine Learning: Algorithms, Real-World Applications and Research Directions)** của tác giả **“Iqbal H. Sarker”** nhấn mạnh những thách thức và hướng nghiên cứu tiềm năng dựa trên nghiên cứu của tác giả. Nhìn chung, bài viết này nhằm mục đích phục vụ như một điểm tham chiếu cho cả giới học thuật và chuyên gia trong ngành cũng như cho những người ra quyết định trong các tình huống và lĩnh vực ứng dụng trong thế giới thực khác nhau, đặc biệt là từ quan điểm kỹ thuật.

3.1 – Giới thiệu:

Trong thời gian gần đây, Trí tuệ nhân tạo (AI), đặc biệt là lĩnh vực học máy (ML), đã trải qua một sự phát triển vượt bậc. Điều này diễn ra trong bối cảnh mà phân tích dữ liệu và tính toán đã tạo điều kiện cho các ứng dụng hoạt động một cách thông minh. Lĩnh vực học máy thường cung cấp khả năng học hỏi tự động và nâng cao kinh nghiệm của hệ thống, mà không cần phải viết mã cụ thể. Chúng thường được gọi là công nghệ phổ biến nhất và mới nhất trong cuộc cách mạng công nghiệp thứ tư (4 IR hoặc Công nghiệp 4.0). Công nghiệp 4.0 liên quan đến việc tự động hóa liên tục các hoạt động sản xuất và công nghiệp, bao gồm việc xử lý dữ liệu phức tạp và sử dụng các công nghệ thông minh mới như tự động hóa học máy. Do đó, thuật toán học máy chính là yếu tố quan trọng để phân tích thông minh dữ liệu và phát triển ứng dụng phù hợp trong thế giới thực.

Trong lĩnh vực học máy, các thuật toán có thể được chia thành bốn loại chính: học có giám sát, học không giám sát, học bán giám sát và học tăng cường. Chúng được thảo

luận một cách ngắn gọn trong phần "Các loại dữ liệu trong thế giới thực và kỹ thuật học máy".

Hiệu quả và hiệu suất của giải pháp học máy phụ thuộc vào tính chất của dữ liệu và hiệu suất của các thuật toán học. Trong lĩnh vực học máy, các phân tích phân loại, hồi quy, phân cụm dữ liệu, kỹ thuật tính năng và giảm kích thước, học quy tắc kết hợp hoặc học tăng cường tồn tại để xây dựng các hệ thống dựa trên dữ liệu. Học sâu, phát triển từ mạng nơ-ron nhân tạo, có khả năng phân tích dữ liệu thông minh và là một phần quan trọng của lĩnh vực học máy rộng lớn hơn.

Vì vậy, việc lựa chọn thuật toán học phù hợp với ứng dụng mục tiêu trong một lĩnh vực cụ thể là một thách thức. Mục tiêu của các thuật toán học khác nhau có sự khác biệt, và kết quả của chúng có thể thay đổi tùy thuộc vào tính chất của dữ liệu. Hiểu rõ nguyên tắc và khả năng ứng dụng của các thuật toán là quan trọng để áp dụng chúng trong các lĩnh vực khác nhau như hệ thống IoT, an ninh mạng, đề xuất kinh doanh, thành phố thông minh, chăm sóc sức khỏe, COVID-19, hệ thống nhận thức, nông nghiệp bền vững và nhiều vấn đề khác.

Với sự quan trọng và tiềm năng của "Machine Learning" trong việc phân tích dữ liệu, bài báo này cung cấp một cái nhìn toàn diện về các loại thuật toán học máy khác nhau có thể được áp dụng để nâng cao khả năng thông minh của ứng dụng. Mục tiêu của bài báo là giải thích nguyên tắc và tiềm năng của các kỹ thuật học máy khác nhau cũng như khả năng ứng dụng của chúng trong các lĩnh vực thực tế khác nhau.

Các đóng góp chính của bài báo bao gồm:

Xác định phạm vi nghiên cứu dựa trên tính chất của loại dữ liệu trong thế giới thực và khả năng của các kỹ thuật học máy khác nhau.

Cung cấp cái nhìn toàn diện về các thuật toán học máy có thể cải thiện khả năng thông minh của ứng dụng dựa trên dữ liệu.

Thảo luận về khả năng ứng dụng các giải pháp học máy trong các lĩnh vực thực tế khác nhau.

Đề cập và tóm tắt các hướng nghiên cứu tiềm năng trong phạm vi của dịch vụ và phân tích thông minh dữ liệu.

Phần còn lại của bài báo được tổ chức như sau: Phần tiếp theo trình bày các loại dữ liệu và các thuật toán học máy rộng hơn, đồng thời xác định phạm vi nghiên cứu. Sau đó, bài báo trình bày một cách ngắn gọn và giải thích các thuật toán học máy khác nhau. Cuối cùng, bài báo tập trung vào thảo luận và tóm tắt các lĩnh vực ứng dụng thực tế khác nhau dựa trên các thuật toán học máy. Để kết thúc, các hướng nghiên cứu tiềm năng và tóm tắt của bài báo được nêu ra, cùng với mục đích chính của bài báo trong việc cung cấp hướng dẫn cơ bản cho những người trong ngành học thuật và ngành muốn nghiên cứu và phát triển hệ thống thông minh tự động dựa trên dữ liệu trong các lĩnh vực liên quan bằng cách sử dụng các kỹ thuật học máy.

3.2 – Các loại dữ liệu và kỹ thuật học máy:

3.2.1 – Các loại dữ liệu:

Thường thì, tính chất của dữ liệu được coi là khóa để xây dựng mô hình học máy hoặc các hệ thống thực tế dựa trên dữ liệu. Dữ liệu có thể tồn tại ở nhiều dạng khác nhau, bao gồm dữ liệu có cấu trúc, bán cấu trúc và không có cấu trúc. Ngoài ra, khái niệm "siêu dữ liệu" cũng là một loại dữ liệu đặc biệt về dữ liệu.

a. Dữ liệu có cấu trúc

Dữ liệu có cấu trúc đã được xác định một cách rõ ràng và tuân theo một trật tự tiêu chuẩn. Chúng thường được tổ chức một cách cụ thể, dễ dàng truy cập và sử dụng bởi máy tính hoặc thực thể. Dữ liệu có cấu trúc thường được lưu trữ dưới dạng bảng trong các hệ cơ sở dữ liệu quan hệ. Ví dụ về dữ liệu có cấu trúc bao gồm tên, ngày tháng, địa chỉ, số thẻ tín dụng, thông tin chứng khoán và vị trí địa lý.

b. Dữ liệu không có cấu trúc

Dữ liệu không có định dạng hoặc tổ chức cố định, làm cho việc nắm bắt, xử lý và phân tích chúng trở nên khó khăn hơn. Loại dữ liệu này thường chứa văn bản và tài liệu đa phương tiện, ví dụ như dữ liệu cảm biến, email, mục blog, tài liệu xử lý văn bản, tệp PDF, tệp âm thanh, video và hình ảnh.

c. Dữ liệu bán cấu trúc

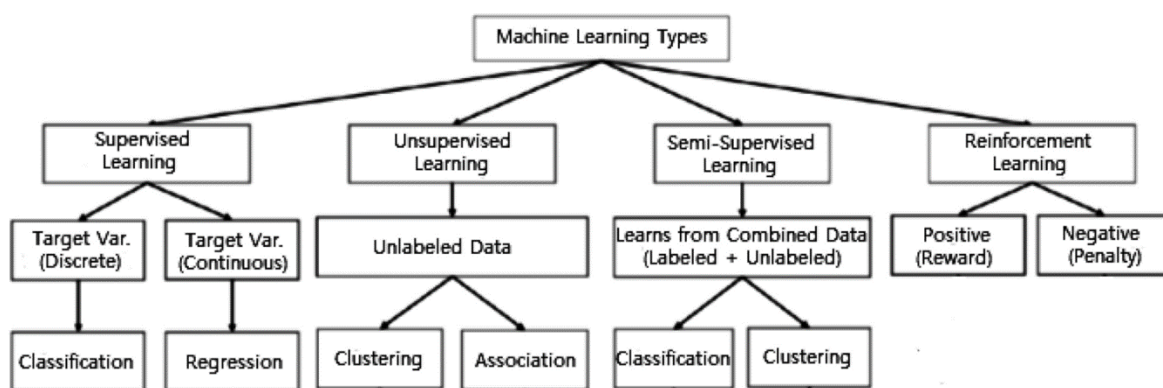
Dữ liệu bán cấu trúc không được lưu trữ trong các cơ sở dữ liệu quan hệ, nhưng vẫn có tổ chức cụ thể giúp việc phân tích trở nên thuận tiện hơn. Ví dụ về dữ liệu bán cấu trúc bao gồm tài liệu HTML, XML, JSON và cơ sở dữ liệu NoSQL.

d. Siêu dữ liệu

Khái niệm "siêu dữ liệu" không chỉ đơn giản là dữ liệu, mà là "dữ liệu về dữ liệu". Sự khác biệt chính giữa dữ liệu và siêu dữ liệu là dữ liệu thường chỉ là các tài liệu có thể phân loại, đo lường hoặc ghi lại điều gì đó về thuộc tính của chúng. Ngược lại, siêu dữ liệu mô tả thông tin có liên quan đến dữ liệu, mang lại nhiều ý nghĩa hơn cho người sử dụng. Ví dụ về siêu dữ liệu có thể bao gồm tên tác giả, kích thước tệp, ngày tạo và từ khóa liên quan đến tài liệu.

3.2.2 – Các kỹ thuật học máy:

Các thuật toán Học Máy thường được phân thành bốn loại chính: Học có giám sát, Học không giám sát, Học bán giám sát và Học tăng cường, như được biểu diễn trong hình. Dưới đây, chúng ta sẽ tóm tắt ngắn gọn về từng loại kỹ thuật học cùng với các lĩnh vực ứng dụng của chúng để giải quyết các vấn đề trong thế giới thực.

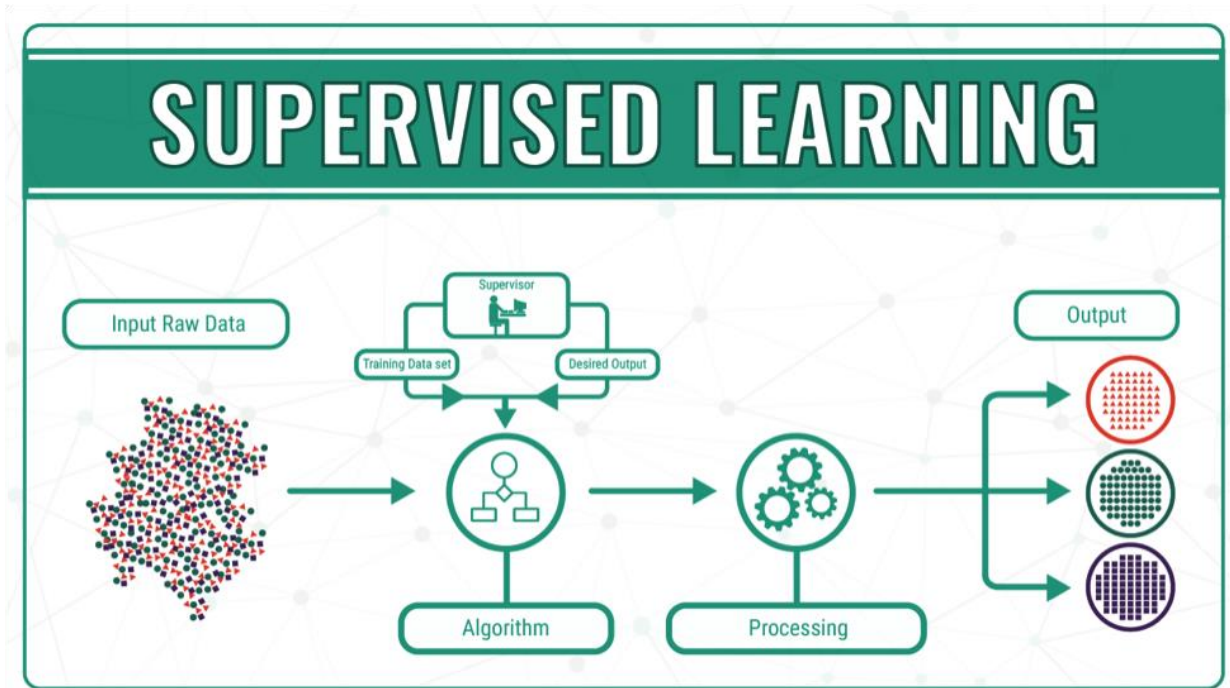


Hình 3. 1: Các kỹ thuật trong máy học

a. Học có giám sát (Supervised Learning)

Học có giám sát là phương pháp sử dụng những dữ liệu được gán nhãn sẵn để suy luận ra quan hệ giữa đầu vào và đầu ra. Sau khi tìm hiểu cách tốt nhất để mô hình hóa các

mối quan hệ cho dữ liệu được gán nhãn, thuật toán huấn luyện sẽ được sử dụng cho các bộ dữ liệu mới. Ứng dụng của học có giám sát chính là giúp xác định tín hiệu tốt nhất để dự báo xu hướng, lợi nhuận trong tương lai trong lĩnh vực cổ phiếu, chứng khoán.

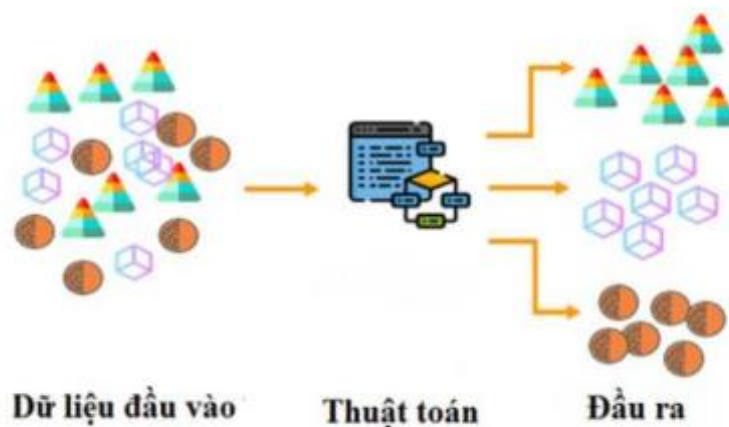


Hình 3. 2: Mô hình học có giám sát

Supervised Learning là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là data, label tức dữ liệu, nhãn. Supervised Learning Là nhóm phổ biến nhất trong các thuật toán Machine learning

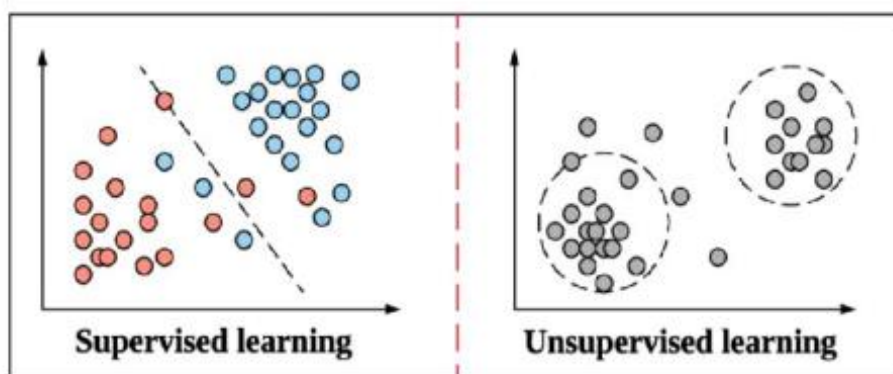
b. Học không giám sát (Unsupervised Learning)

Học không giám sát sử dụng những dữ liệu chưa được gán nhãn sẵn để suy luận và tìm cách để mô tả dữ liệu cùng cấu trúc của chúng. Ứng dụng của học không giám sát đó là hỗ trợ phân loại thành các nhóm có đặc điểm tương đồng.



Hình 3. 3: Mô hình học không giám sát

Trong thuật toán này, chúng ta không biết được dữ liệu đầu ra hay nhãn mà chỉ có dữ liệu đầu vào. Thuật toán Học không giám sát dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm hoặc giảm số chiều của dữ liệu để thuận tiện trong việc lưu trữ và tính toán. Một cách toán học, Học không giám sát là khi chúng ta chỉ có dữ liệu vào X mà không biết nhãn Y tương ứng. Sự khác nhau giữa học có giám sát và học không giám sát:



Hình 3. 4: Sự khác biệt giữa 2 mô hình

Học có giám sát: Là cách huấn luyện một mô hình trong đó dữ liệu học có đầu vào và đầu ra tương ứng đầu vào đó. Mô hình được huấn luyện bằng cách giảm thiểu sai số lỗi (loss) của các dự đoán tại các vòng lặp huấn luyện. Sau quá trình huấn luyện, mô hình sẽ có khả năng đưa ra dự đoán về đầu ra với một đầu vào mới gặp (không có trong dữ

liệu học). Nếu không gian đầu ra được biểu diễn dưới dạng rời rạc, ta gọi đó là bài toán phân loại (classification). Nếu không gian đầu ra được biểu diễn dưới dạng liên tục, ta gọi đó là bài toán hồi quy (regression). Học không giám sát: Là cách huấn luyện một mô hình trong đó dữ liệu học chỉ bao gồm đầu vào mà không có đầu ra. Mô hình sẽ được huấn luyện cách để tìm cấu trúc hoặc mối quan hệ giữa các đầu vào. Một trong những phương pháp học không giám sát quan trọng nhất là phân cụm (clustering): Tạo các cụm khác nhau với mỗi cụm biểu diễn một đặc trưng nào đó của dữ liệu và phân các đầu vào mới vào các cụm theo các đặc trưng của đầu vào đó. Các phương pháp học không giám sát khác có thể kể đến như: phát hiện điểm bất thường (anomaly detection), Singular-value decomposition, ...

c. Học tăng cường (Reinforcement Learning)

Phương pháp học tăng cường tập trung vào việc làm sao để cho 1 tác tử trong môi trường có thể hành động sao cho lấy được phần thưởng nhiều nhất có thể. Khác với học có giám sát nó không có cặp dữ liệu gán nhãn trước làm đầu vào và cũng không có đánh giá các hành động là đúng hay sai. Trong đó, thuật toán học một chính sách hành động tùy theo các quan sát về thế giới. Mỗi hành động đều có tác động tới môi trường, và môi trường cung cấp thông tin phản hồi để hướng dẫn cho thuật toán của quá trình học.

d. Học bán giám sát (Semi-supervised learning)

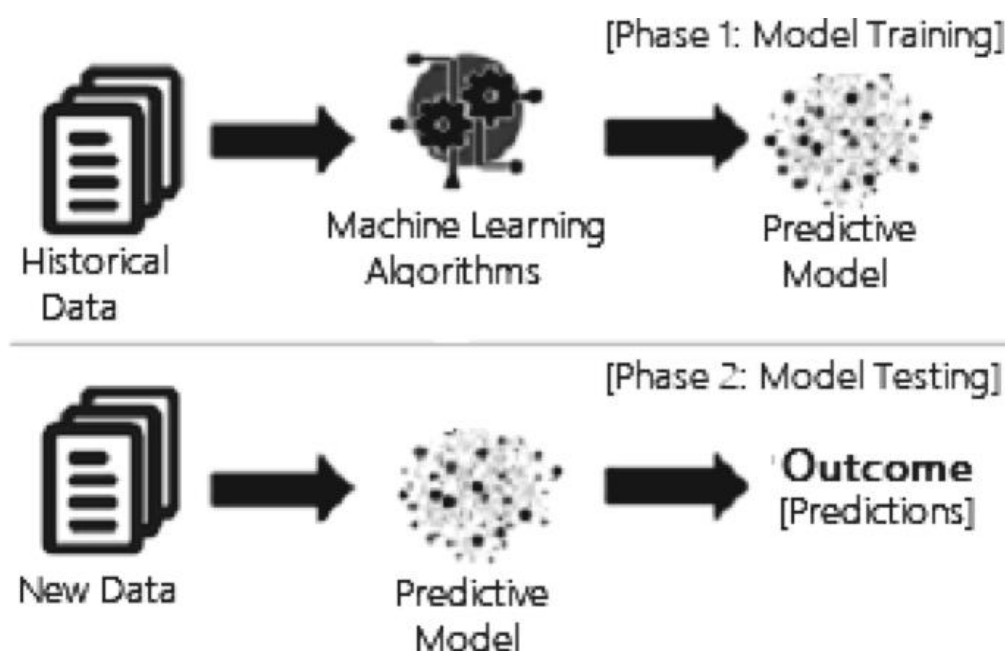
Các bài toán khi chúng ta có một lượng lớn dữ liệu X nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên. Một ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bản khoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán Machine Learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (ảnh y học chẳng hạn). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

Vì vậy, để xây dựng các mô hình hiệu quả trong các lĩnh vực ứng dụng khác nhau, các loại kỹ thuật học máy khác nhau có thể đóng một vai trò quan trọng tùy theo khả năng học tập của chúng và bản chất của dữ liệu đã được thảo luận trước đó, cũng như kết quả mục tiêu. Trong hình trên, chúng ta đã tổng hợp các loại kỹ thuật học máy khác nhau kèm theo ví dụ. Trong phần tiếp theo, chúng ta sẽ nghiên cứu cái nhìn toàn diện về các thuật toán học máy có thể được áp dụng để nâng cao trí thông minh và khả năng của ứng dụng dựa trên dữ liệu.

3.3 – Nhiệm vụ và thuật toán của học máy:

3.3.1 – Phân tích phân loại:

Phân loại là một phương pháp học có giám sát trong lĩnh vực học máy, có thể áp dụng để dự đoán nhãn lớp cho các ví dụ cụ thể. Nó tạo ra một ánh xạ từ biến đầu vào sang biến đầu ra, nhãn hoặc danh mục. Ví dụ, phân loại có thể được sử dụng để phát hiện thư rác và không phải thư rác trong email.



Hình 3. 5: Cấu trúc chung của mô hình dự đoán dựa trên học máy

Trong phân loại, có các loại chính sau:

a. Phân loại nhị phân:

Đây liên quan đến việc phân loại với hai nhãn lớp như "đúng" và "sai" hoặc "có" và "không". Một ví dụ cụ thể là phân loại giữa "phát hiện ung thư" và "không phát hiện ung thư".

Phân loại nhiều lớp: Trong trường hợp này, có nhiều hơn hai nhãn lớp. Không có nguyên tắc về trạng thái bình thường và bất thường như phân loại nhị phân. Thay vào đó, mỗi ví dụ được gán vào một lớp cụ thể trong tập hợp các lớp đã cho. Ví dụ, phân loại các loại tấn công mạng khác nhau có thể là một nhiệm vụ phân loại nhiều lớp.

b. Phân loại nhiều nhãn:

Trong trường hợp này, mỗi ví dụ có thể liên kết với nhiều lớp hoặc nhãn. Ví dụ, tin tức có thể được phân loại vào nhiều danh mục như "tên thành phố", "công nghệ" hoặc "tin tức mới nhất".

Một số thuật toán phân loại phổ biến bao gồm:

a. Naive Bayes (NB):

Dựa trên định lý Bayes với giả định về tính độc lập giữa các tính năng. Thường sử dụng cho phân loại tài liệu, văn bản và lọc thư rác.

Phân tích phân biệt tuyến tính (LDA): Tạo bộ phân loại ranh giới quyết định tuyến tính bằng cách áp dụng quy tắc Bayes sau khi khớp mật độ có điều kiện của các lớp. Thường được sử dụng để giảm chiều dữ liệu và phân loại trong các lĩnh vực khác nhau.

b. Hồi quy logistic (LR):

Là một phương pháp phân loại dựa trên xác suất, sử dụng hàm logistic (hàm sigmoid) để ước tính xác suất. LR thường hoạt động tốt khi dữ liệu có thể phân tách tuyến tính và có thể bị overfit trong trường hợp dữ liệu nhiều chiều. Kỹ thuật chuẩn hóa (L1 và L2) có thể được sử dụng để tránh overfitting. LR cũng có khuyết điểm khi giả định về tính tuyến tính giữa các biến phụ thuộc và độc lập không được thỏa mãn. Nó thường được sử dụng cho phân loại.

c. K-hàng xóm gần nhất (KNN):

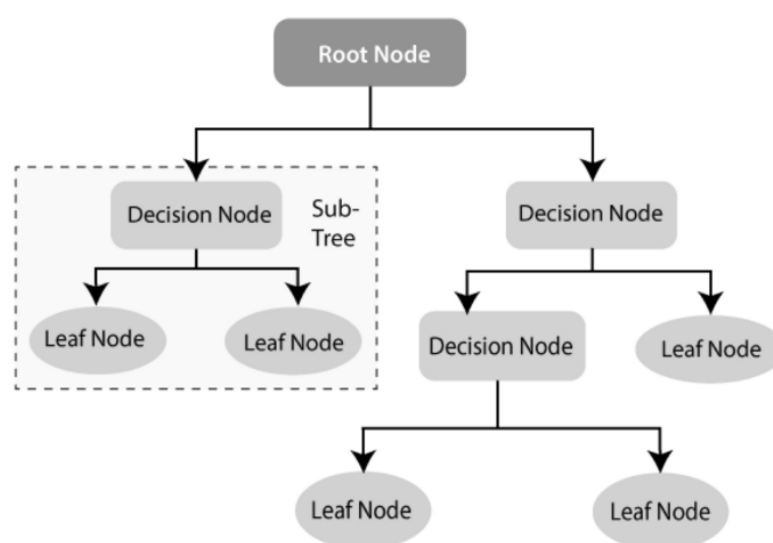
Là một thuật toán "học tập dựa trên cá thể" hoặc "lười biếng", không tạo ra một mô hình chung mà lưu trữ dữ liệu huấn luyện trong không gian n chiều. KNN phân loại dựa trên các phép đo tương tự như khoảng cách Euclide và bỏ phiếu đa số từ các hàng xóm gần nhất của mỗi điểm để quyết định phân loại. Nó phù hợp cho dữ liệu nhiều và độ chính xác phụ thuộc vào chất lượng dữ liệu. Tuy nhiên, việc chọn số lượng hàng xóm tối ưu là một vấn đề.

d. Máy vector hỗ trợ (SVM)

Là một kỹ thuật phân loại, hồi quy hoặc tác vụ khác. SVM xây dựng siêu phẳng hoặc tập hợp các siêu phẳng để tạo khoảng cách lớn nhất giữa các điểm dữ liệu huấn luyện gần nhất trong các lớp. Nó hoạt động tốt trong không gian nhiều chiều và sử dụng các hàm nhân như tuyến tính, đa thức, RBF, sigmoid. Tuy nhiên, SVM không hoạt động tốt khi dữ liệu chứa nhiễu hoặc các lớp mục tiêu chồng chéo.

e. Cây quyết định (DT):

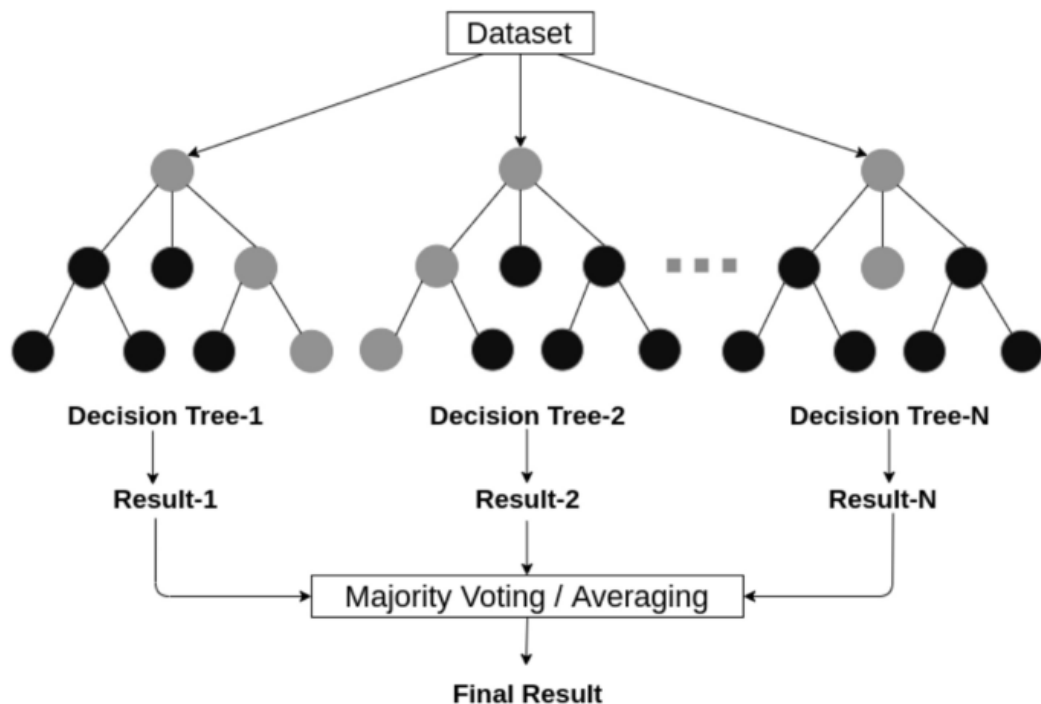
Là phương pháp học có giám sát phi tham số, sử dụng cây quyết định để phân loại dựa trên thuộc tính của các nút trong cây. Thuộc tính tại mỗi nút quyết định việc điều hướng sang nhánh cây tương ứng. Tiêu chí phổ biến là "gini" và "entropy" để phân tách các thể hiện. DT được sử dụng cho cả nhiệm vụ phân loại và hồi quy.



Hình 3. 6: Cấu trúc cây quyết định

f. Rừng ngẫu nhiên (Random Forest - RF):

Một bộ phân loại Random Forest được biết đến là một kỹ thuật phân loại tổ hợp được sử dụng trong lĩnh vực học máy và khoa học dữ liệu trong nhiều lĩnh vực ứng dụng khác nhau. Phương pháp này sử dụng "tổ hợp song song" bằng cách đặt nhiều bộ phân loại cây quyết định song song, như hình 5, trên các tập con dữ liệu khác nhau và sử dụng đa số phiếu bỏ phiếu hoặc trung bình cho kết quả cuối cùng. Điều này giúp giảm vấn đề overfitting, tăng độ chính xác và kiểm soát. Do đó, mô hình học RF với nhiều cây quyết định thường chính xác hơn so với mô hình dựa trên cây quyết định đơn lẻ. Để xây dựng một loạt các cây quyết định với sự biến đổi kiểm soát, nó kết hợp hợp chất bao (bagging) và lựa chọn đặc trưng ngẫu nhiên. Nó thích nghi cả với các vấn đề phân loại và hồi quy, cũng như phù hợp với cả giá trị phân loại và liên tục.



Hình 3. 7: Cấu trúc rừng ngẫu nhiên khi xem xét nhiều cây quyết định

g. AdaBoost (Adaptive Boosting):

AdaBoost là một quá trình học tổ hợp sử dụng phương pháp lặp để cải thiện các bộ phân loại kém bằng cách học từ sai lầm của chúng. Phương pháp này được phát triển bởi Yoav Freund và cộng sự và còn được gọi là "meta-learning". Khác với Random

Forest sử dụng tổ hợp song song, AdaBoost sử dụng "tổ hợp tuần tự". Nó tạo ra một bộ phân loại mạnh bằng cách kết hợp nhiều bộ phân loại hoạt động kém để thu được một bộ phân loại tốt với độ chính xác cao. Theo cách này, AdaBoost được gọi là một bộ phân loại thích ứng bởi việc cải thiện đáng kể hiệu suất của bộ phân loại, nhưng trong một số trường hợp, nó có thể gây ra overfitting. AdaBoost thường được sử dụng để tăng hiệu suất của cây quyết định, bộ ước tính cơ bản, trong các vấn đề phân loại nhị phân, tuy nhiên, nó nhạy cảm với dữ liệu nhiễu và ngoại lệ.

h. Extreme Gradient Boosting (XGBoost):

Gradient Boosting, tương tự như Random Forest [19] ở trên, là một thuật toán học tổ hợp tạo ra một mô hình cuối cùng dựa trên một loạt các mô hình cá nhân, thường là cây quyết định. Đạo hàm được sử dụng để giảm thiểu hàm mất mát, tương tự cách mạng nơ-ron sử dụng gradient descent để tối ưu hóa trọng số. Extreme Gradient Boosting (XGBoost) là một biến thể của gradient boosting có các xấp xỉ chi tiết hơn khi xác định mô hình tốt nhất. Nó tính toán đạo hàm cấp hai của hàm mất mát để giảm thiểu mất mát và sử dụng chế độ hóa nâng cao (L1 và L2), giảm overfitting và cải thiện tổng quát hóa và hiệu suất của mô hình. XGBoost dễ dàng hiểu và có thể xử lý tốt các tập dữ liệu có kích thước lớn.

i. Stochastic Gradient Descent (SGD):

Stochastic Gradient Descent (SGD) là một phương pháp lặp để tối ưu hóa hàm mục tiêu với tính chất mịn phù hợp, trong đó từ "ngẫu nhiên" ám chỉ xác suất ngẫu nhiên. Điều này giúp giảm gánh nặng tính toán, đặc biệt trong các vấn đề tối ưu hóa chiều cao, cho phép các lặp nhanh hơn nhưng với tốc độ hội tụ thấp hơn. Đạo hàm là độ dốc của một hàm tính toán mức độ thay đổi của biến dưới sự thay đổi của biến khác. Toán học, Gradient Descent là một hàm lồi mà đầu ra là đạo hàm riêng của một tập hợp các tham số đầu vào. Trong học máy quy mô lớn và thưa thớt, SGD đã được áp dụng thành công vào các vấn đề thường gặp trong phân loại văn bản và xử lý ngôn ngữ tự nhiên. Tuy nhiên, SGD nhạy cảm với việc tỷ lệ tỷ lệ và cần một loạt siêu tham số, chẳng hạn như tham số chính quy hóa và số vòng lặp.

j. Phân loại dựa trên luật:

Thuật ngữ phân loại dựa trên luật có thể được sử dụng để chỉ bất kỳ phương pháp phân loại nào sử dụng luật IF-THEN để dự đoán lớp. Một số thuật toán phân loại như Zero-R, One-R, cây quyết định, DTNB, Ripple Down Rule learner (RIDOR), Repeated Incremental Pruning to Produce Error Reduction (RIPPER) tồn tại với khả năng tạo ra luật. Cây quyết định là một trong những thuật toán phân loại dựa trên luật phổ biến nhất trong các kỹ thuật này vì nó có nhiều ưu điểm như dễ hiểu hơn, khả năng xử lý dữ liệu nhiều chiều, đơn giản và nhanh chóng, độ chính xác tốt và khả năng tạo ra luật cho phân loại dễ hiểu bởi con người. Luật dựa trên cây quyết định cũng cung cấp độ chính xác đáng kể trong mô hình dự đoán cho các trường hợp kiểm tra chưa được thấy. Vì luật dễ dàng hiểu, các bộ phân loại dựa trên luật này thường được sử dụng để tạo ra các mô hình mô tả có thể mô tả một hệ thống bao gồm các thực thể và mối quan hệ của chúng.

3.3.2 – Phân tích phân cụm:

Phân tích phân cụm, còn được gọi là phân cụm, là một phương pháp không giám sát trong máy học để nhóm các điểm dữ liệu tương tự trong một tập dữ liệu lớn mà không cần biết kết quả cụ thể trước. Nó hoạt động bằng cách gom nhóm các đối tượng lại với nhau theo cách sao cho các đối tượng trong cùng một nhóm, hay cụm, có sự tương đồng nhất định hơn so với các đối tượng trong các nhóm khác. Phương pháp này thường được sử dụng để khám phá các mẫu hay xu hướng trong dữ liệu, ví dụ như việc nhóm người tiêu dùng dựa trên hành vi của họ. Các lĩnh vực như an ninh mạng, thương mại điện tử, xử lý dữ liệu di động và phân tích hành vi có thể sử dụng phân cụm để tìm hiểu và phân loại dữ liệu.

Có nhiều loại phương pháp phân cụm khác nhau:

a. Phương pháp phân vùng:

Dựa vào đặc trưng và tương đồng của dữ liệu, phương pháp này chia dữ liệu thành nhiều nhóm hoặc cụm. Số lượng cụm có thể được xác định tùy theo ứng dụng, và một số thuật toán phổ biến trong loại này là K-means, K-Medoids, CLARA, v.v.

b. Phương pháp dựa trên mật độ:

Sử dụng khái niệm về mật độ dữ liệu để xác định các cụm riêng biệt. Các thuật toán dựa trên mật độ tìm các vùng có mật độ cao và cách xa các vùng có mật độ thấp. Ví dụ: DBSCAN, OPTICS.

c. Phương pháp dựa trên phân cấp:

Xây dựng một cấu trúc cây từ các cụm dựa trên cách chúng tương đồng nhau. Phân cụm theo phân cấp có thể được thực hiện từ dưới lên hoặc từ trên xuống. Kỹ thuật BOTS là một ví dụ về phân cụm theo thứ bậc từ dưới lên.

d. Phương pháp dựa trên lưới:

Sử dụng biểu diễn lưới để xử lý dữ liệu lớn. Các phương pháp như STING, CLIQUE sử dụng cách tiếp cận này.

e. Phương pháp dựa trên mô hình:

Sử dụng các mô hình xác suất như mô hình hỗn hợp Gaussian (GMM) để phân cụm dữ liệu. GMM sử dụng các phân phối Gaussian để mô tả các cụm.

f. Phương pháp dựa trên ràng buộc:

Sử dụng các ràng buộc để nhóm dữ liệu dựa trên kiến thức miền. Các thuật toán như COP K-means, CMWK-Means sử dụng cách tiếp cận này.

Nhiều thuật toán phân cụm đã được phát triển cho các lĩnh vực khác nhau. Các ví dụ gồm K-means, DBSCAN, GMM, và nhiều phương pháp khác. Chúng tôi đã tóm tắt các loại phương pháp phân cụm và cung cấp một cái nhìn tổng quan về sự đa dạng và ứng dụng của chúng.

3.3.3 – Giảm kích thước và học tính năng

a. Giảm kích thước:

Trong lĩnh vực học máy và khoa học dữ liệu, việc xử lý dữ liệu nhiều chiều đang là thách thức lớn cho các nhà nghiên cứu và nhà phát triển ứng dụng. Để giải quyết vấn đề này, giảm kích thước dữ liệu đã trở thành một kỹ thuật học máy không giám sát

quan trọng. Kỹ thuật này đem lại nhiều lợi ích như khả năng hiểu rõ hơn từ phía người đọc, giảm chi phí tính toán và tránh sự dư thừa bằng cách đơn giản hóa mô hình. Cả việc lựa chọn và trích xuất tính năng đều có thể dùng để giảm kích thước.

- **Lựa chọn tính năng:**

Còn gọi là lựa chọn biến hoặc thuộc tính, kỹ thuật này tập trung vào việc chọn ra một tập con các tính năng duy nhất từ dữ liệu để xây dựng mô hình học máy. Quá trình này giúp giảm phức tạp của mô hình bằng cách loại bỏ những tính năng không quan trọng và giúp thuật toán học máy đào tạo nhanh hơn. Sự lựa chọn đúng tính năng có thể giúp giảm thiểu vấn đề overfitting, tăng khả năng tổng quát và chính xác của mô hình.

- **Trích xuất tính năng:**

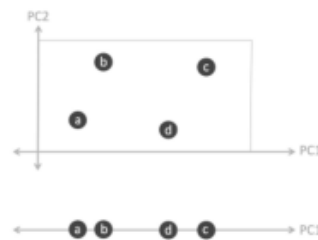
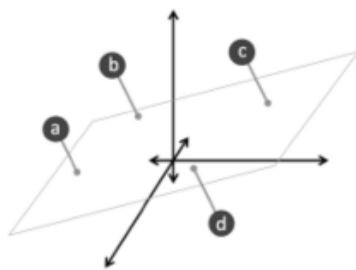
Trong các mô hình máy học, kỹ thuật trích xuất tính năng giúp chúng ta hiểu sâu hơn về dữ liệu, cải thiện khả năng dự đoán và giảm tải chi phí tính toán. Mục tiêu của trích xuất tính năng là giảm số lượng tính năng bằng cách tạo ra những tính năng mới từ những tính năng gốc, sau đó loại bỏ các tính năng gốc đó. Ví dụ, phân tích thành phần chính (PCA) thường được dùng để tạo ra các thành phần tương ứng với không gian chiều thấp hơn từ dữ liệu gốc.

b. Học tính năng/đặc trưng:

Có nhiều thuật toán khác nhau đã được đề xuất để giảm kích thước dữ liệu trong lĩnh vực học máy và khoa học dữ liệu. Dưới đây là một số phương pháp phổ biến được sử dụng trong các lĩnh vực ứng dụng khác nhau.

- **Ngưỡng phương sai:** Phương pháp đơn giản là loại bỏ các tính năng có phương sai thấp, tức là có ít biến đổi. Điều này giúp loại bỏ các tính năng không mang lại nhiều thông tin và giữ lại những tính năng có sự biến đổi lớn.
- **Tương quan Pearson:** Phương pháp này đo lường mối quan hệ tuyến tính giữa các tính năng và biến phản hồi. Nó cũng có thể được sử dụng để lựa chọn tính năng, với giá trị kết quả nằm trong khoảng từ -1 đến 1, thể hiện mức độ tương quan.

- **ANOVA:** Phân tích phương sai kiểm tra sự khác biệt giữa giá trị trung bình của các nhóm khác nhau. Nó dựa vào giả định về mối quan hệ tuyến tính và phân phối chuẩn của các biến.
- **The chi-square:** Loại bỏ tính năng đệ quy (RFE): RFE loại bỏ lần lượt các tính năng yếu nhất cho đến khi đạt được số lượng tính năng cần giữ lại.
- **Lựa chọn dựa trên mô hình:** Sử dụng mô hình tuyến tính bị phạt L1 như Lasso giúp loại bỏ các tính năng không quan trọng. Các công cụ phân loại cây cũng có thể sử dụng để ước tính tầm quan trọng của các tính năng.
- **Phân tích thành phần chính (PCA):** Phương pháp này biến đổi tập hợp các biến tương quan thành tập hợp các biến không tương quan, giúp giảm kích thước bộ dữ liệu và cải thiện mô hình học máy.



(a) An example of the original features in a 3D space.

(b) Principal components in 2D and 1D space.

Hình 3. 8: Ví dụ về phân tích thành phần chính (PCA)

3.3.4 – Khai phá luật kết hợp:

Học luật kết hợp là một phương pháp trong học máy nhằm khám phá mối quan hệ có ý nghĩa giữa các biến trong các bộ dữ liệu lớn. Những mối quan hệ này được biểu thị dưới dạng câu "If - Then". Ví dụ, nếu một khách hàng mua một máy tính hoặc laptop, thì họ có khả năng mua phần mềm chống vi-rút cùng lúc. Phương pháp này được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm dịch vụ IoT, chẩn đoán y học, phân tích hành vi sử dụng, khai thác sử dụng web, ứng dụng điện thoại thông minh, an ninh mạng và sinh học tin học. Khác với việc khai thác chuỗi, học luật kết hợp không xem xét thứ tự của các sự kiện trong hoặc qua các giao dịch. Hiệu suất của các luật kết hợp thường được đo bằng các tham số như 'hỗ trợ' và 'độ tin cậy'.

Nhiều phương pháp học luật kết hợp đã được đề xuất trong tài liệu khai thác dữ liệu, bao gồm các phương pháp dựa vào logic phụ thuộc, dựa trên mẫu thường xuyên và dựa trên cây. Một số thuật toán phổ biến trong lĩnh vực này bao gồm:

- **AIS và SETM:** Những thuật toán này được giới thiệu bởi Agrawal et al. AIS tạo ra nhiều tập hợp ứng viên, dẫn đến việc sử dụng tài nguyên cao hơn. SETM thể hiện hành vi ổn định nhưng cũng tạo ra một số lượng lớn tập hợp ứng viên.
- **Apriori:** Agrawal et al. đề xuất các thuật toán Apriori, giúp tạo ra các luật kết hợp cho các bộ dữ liệu. Những thuật toán này vượt trội hơn AIS và SETM do tính chất Apriori. Apriori tiến hành từ dưới lên, tạo ra các tập hợp ứng viên trong khi sử dụng nguyên tắc "tất cả các tập con của tập hợp phổ biến phải cũng phổ biến" để giảm không gian tìm kiếm.
- **ECLAT:** Được đề xuất bởi Zaki et al., ECLAT sử dụng tìm kiếm theo chiều sâu để tìm tập hợp phổ biến. Khác với Apriori, nó biểu thị dữ liệu theo chiều dọc, làm cho nó hiệu quả cho các bộ dữ liệu nhỏ và trung bình.
- **FP-Growth:** Frequent Pattern Growth của Han et al. sử dụng cây mẫu thường xuyên (FP-tree). Khác với Apriori, nó ngăn chặn việc tạo ra tập hợp ứng viên và thay vào đó tạo ra một cây thông qua chiến lược "chia để trị". Tuy nhiên, việc sử dụng FP-Tree khá phức tạp và không phù hợp cho môi trường khai thác tương tác.
- **ABC-RuleMiner:** Được đề xuất gần đây bởi Sarker et al., thuật toán này tập trung vào việc khám phá các luật kết hợp không lặp lại bằng cách xem xét các đặc trưng ngữ cảnh liên quan. Nó xây dựng một cây tạo ra luật kết hợp thông qua việc đi qua cây. Do đó, ABC-RuleMiner mạnh mẽ hơn so với các phương pháp dựa trên luật truyền thống từ cả hai khía cạnh là tạo ra luật không lặp lại và ra quyết định thông minh, đặc biệt trong môi trường tính cách thông minh dựa trên ngữ cảnh, nơi có sự tham gia của người dùng hoặc người dùng.

Trong số những kỹ thuật học luật kết hợp được thảo luận ở trên, Apriori là thuật toán phổ biến nhất để khám phá luật kết hợp từ một bộ dữ liệu cho trước. Điểm mạnh chính của kỹ thuật học luật kết hợp là tính toàn diện, khi nó tạo ra tất cả các mối quan hệ thỏa

mãn các ràng buộc được chỉ định bởi người dùng, như hỗ trợ tối thiểu và giá trị độ tin cậy. Phương pháp ABC-RuleMiner đã thảo luận ở trên cũng có thể mang lại kết quả đáng kể trong việc tạo ra luật không lặp lại và ra quyết định thông minh cho các lĩnh vực ứng dụng liên quan trong thế giới thực.

3.3.5 – Học tăng cường

Học tăng cường là một phương pháp học máy cho phép một "đại diện" học thông qua thử và sai trong môi trường tương tác bằng cách sử dụng thông tin từ hành động và trải nghiệm của nó. Khác với học có giám sát, dựa trên dữ liệu mẫu hoặc ví dụ đã có, phương pháp học tăng cường dựa vào việc tương tác với môi trường. Bài toán cần giải quyết trong học tăng cường được xác định là MDP (quyết định Markov), nghĩa là liên quan đến việc đưa ra quyết định theo chuỗi. Một bài toán học tăng cường thường bao gồm bốn yếu tố: Đại diện, Môi trường, Phần thưởng và Chính sách.

Học tăng cường có thể được chia thành hai phần: Kỹ thuật dựa trên mô hình và Kỹ thuật không dựa trên mô hình. Kỹ thuật dựa trên mô hình là quá trình suy luận hành vi tối ưu từ mô hình môi trường bằng cách thực hiện hành động và quan sát kết quả, bao gồm trạng thái tiếp theo và phần thưởng ngay lập tức. AlphaZero, AlphaGo là ví dụ về các phương pháp dựa trên mô hình. Mặt khác, phương pháp không dựa trên mô hình không sử dụng phân phối xác suất chuyển tiếp và hàm phần thưởng liên quan đến MDP. Q-learning, Mạng Q Sâu, Kiểm soát Monte Carlo, SARSA (State–Action–Reward–State–Action) và nhiều thuật toán khác là ví dụ về các thuật toán không dựa trên mô hình. Mạng chính sách, cần thiết cho học dựa trên mô hình nhưng không cần cho học không dựa trên mô hình, là sự khác biệt chính giữa học không dựa trên mô hình và học dựa trên mô hình. Dưới đây, chúng ta sẽ thảo luận về các thuật toán học tăng cường phổ biến.

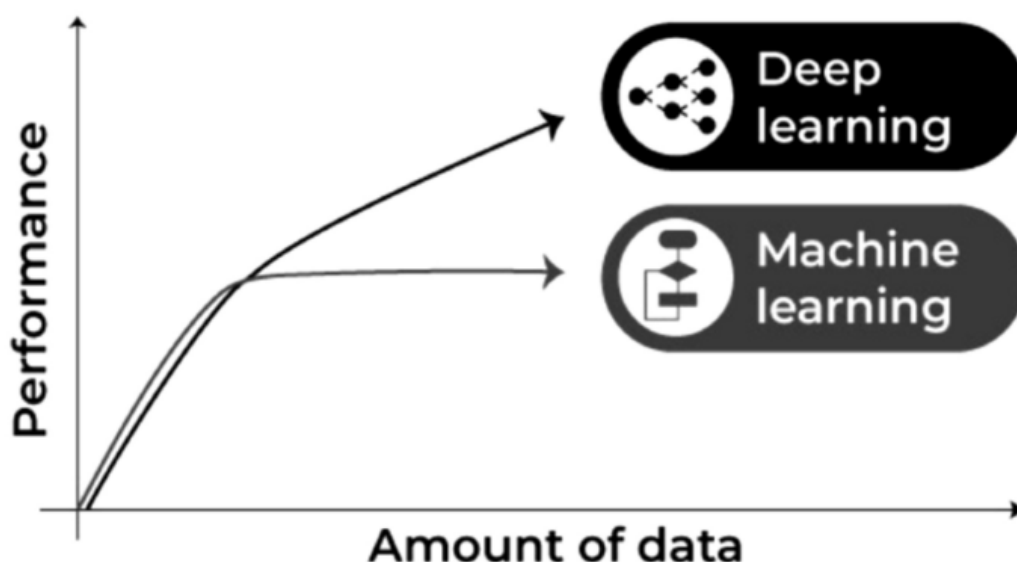
- **Phương pháp Monte Carlo:** Phương pháp Monte Carlo là một loại rất rộng của thuật toán tính toán dựa trên việc lặp lại lấy mẫu ngẫu nhiên để thu được kết quả số. Ý tưởng cơ bản là sử dụng ngẫu nhiên để giải quyết các vấn đề ban đầu là xác định. Tối ưu hóa, tích phân số và việc tạo ra các giá trị từ phân phối xác suất là ba loại vấn đề mà phương pháp Monte Carlo thường được sử dụng.

- **Q-learning:** Q-learning là một thuật toán học tăng cường không dựa trên mô hình để học chất lượng các hành vi mà nói cho một đại diện biết hành động nào được thực hiện dưới điều kiện nào. Nó không cần một mô hình về môi trường (vì vậy còn được gọi là "không dựa trên mô hình") và có thể xử lý các chuyển tiếp và phần thưởng ngẫu nhiên mà không cần sự điều chỉnh. 'Q' trong Q-learning thường đại diện cho chất lượng, vì thuật toán tính toán phần thưởng kỳ vọng tối đa cho một hành vi cụ thể trong một trạng thái cụ thể.
- **Deep Q-learning:** Bước làm cơ bản trong Deep Q-Learning là đưa trạng thái ban đầu vào mạng thần kinh, trả lại giá trị Q của tất cả các hành động có thể như một đầu ra. Khi ta có môi trường tương đối đơn giản, Q-learning hoạt động tốt. Tuy nhiên, khi số trạng thái và hành động trở nên phức tạp hơn, học sâu có thể được sử dụng như một bộ xấp xỉ hàm.

Học tăng cường, cùng với học có giám sát và không có giám sát, là một trong những hình thức cơ bản của học máy. Học tăng cường có thể được sử dụng để giải quyết nhiều vấn đề thế giới thực trong các lĩnh vực khác nhau, như lý thuyết trò chơi, lý thuyết kiểm soát, phân tích hoạt động.

3.3.6 – Mạng Nơ-ron nhân tạo và Học sâu:

Học sâu là một phần của hệ thống rộng hơn, bao gồm các phương pháp học máy dựa trên mạng nơ-ron nhân tạo (ANN) với khả năng học biểu diễn. Học sâu đem đến kiến trúc tính toán bằng cách kết hợp nhiều lớp xử lý như lớp đầu vào, lớp ẩn và lớp đầu ra để học từ dữ liệu [41]. Một ưu điểm chính của học sâu so với các phương pháp học máy truyền thống là hiệu suất tốt hơn trong nhiều tình huống, đặc biệt khi xử lý dữ liệu lớn [105, 129]. Hình 9 thể hiện so sánh hiệu suất tổng thể giữa học sâu và học máy theo sự gia tăng của dữ liệu. Tuy nhiên, kết quả này có thể biến đổi tùy thuộc vào tính chất của dữ liệu và bối cảnh thực nghiệm.

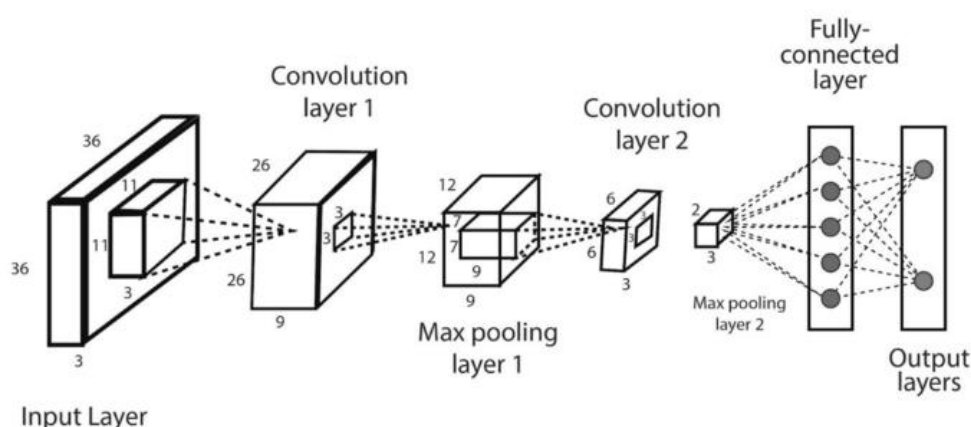


Hình 3. 9: Hiệu suất học máy và học sâu theo sự gia tăng của dữ liệu

Các thuật toán học sâu phổ biến bao gồm:

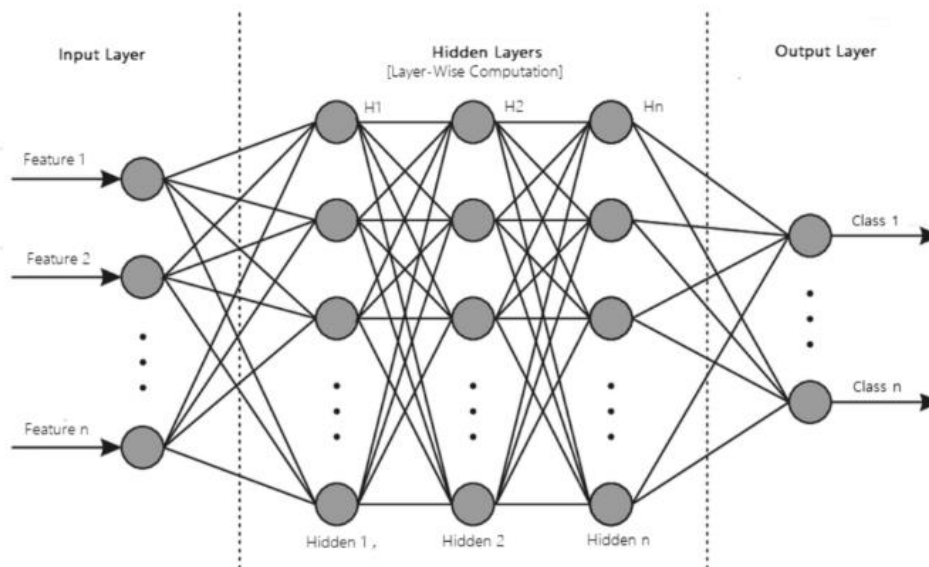
- **Mạng Nơ-ron Nhiều Lớp (MLP):** Còn gọi là perceptron nhiều lớp, MLP là cơ sở của học sâu. Nó bao gồm một lớp đầu vào, nhiều lớp ẩn và một lớp đầu ra, như hình 10. MLP sử dụng "Backpropagation" để điều chỉnh trọng số và thường được sử dụng khi xây dựng mô hình dự đoán.
- **Mạng Nơ-ron Tích Chập (CNN):** CNN cải tiến thiết kế của mạng nơ-ron thông thường, với các lớp tích chập và gộp, thường được dùng cho xử lý ảnh và ngôn ngữ tự nhiên.
- **Mạng Nơ-ron Tái Phát Nhân Tạo Bộ Nhớ Ngắn Hạn Dài (LSTM-RNN):** LSTM được sử dụng cho dữ liệu tuần tự, chẳng hạn như chuỗi thời gian và ngôn ngữ tự nhiên.
- **Mạng Nơ-ron Nhiều Lớp (MLP):** MLP, hay còn gọi là perceptron đa lớp, là kiến trúc cơ bản trong học sâu. Nó bao gồm một lớp đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra, như hình 10. Mỗi nút trong lớp này kết nối với từng nút trong lớp tiếp theo với trọng số cụ thể. MLP sử dụng kỹ thuật "Backpropagation" để điều chỉnh trọng số trong quá trình xây dựng mô hình. Khả năng tùy chỉnh nhiều siêu tham số, như số lớp ẩn, số nơ-ron và số vòng lặp, giúp MLP phù hợp với nhiều tình huống, tuy nhiên có thể dẫn đến tốn kém về tính toán.

- **Mạng Nơ-ron Tích Chập (CNN):** CNN là phiên bản cải tiến của mạng nơ-ron thông thường, bao gồm các lớp tích chập, lớp gộp và các lớp kết nối đầy đủ, như hình 11. Nhờ tận dụng cấu trúc hai chiều của dữ liệu đầu vào, CNN phổ biến trong việc xử lý hình ảnh và ngôn ngữ tự nhiên. Một số mô hình dựa trên CNN, như AlexNet, Xception, Inception, VGG và ResNet, thường được áp dụng trong nhận dạng hình ảnh, xử lý hình ảnh y tế, xử lý ngôn ngữ tự nhiên và nhiều lĩnh vực khác.



Hình 3. 10: Ví dụ về mạng thần kinh tích chập (CNN hoặc ConvNet)

- **LSTM-RNN:** Bộ nhớ ngắn hạn dài (LSTM) là kiến trúc trong học sâu, thuộc loại mạng nơ-ron tái phát nhân tạo (RNN). LSTM có khả năng lưu trữ thông tin qua các liên kết phản hồi, khác với mạng nơ-ron thông thường. Điều này giúp LSTM phù hợp cho việc phân tích và dự đoán dữ liệu tuần tự như chuỗi thời gian, xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói. Đặc điểm này phân biệt LSTM với các mạng thông thường và thường được áp dụng trong lĩnh vực phân tích chuỗi thời gian, xử lý ngôn ngữ tự nhiên và nhiều ứng dụng khác.



Hình 3. 11: Cấu trúc mô hình ANN với nhiều lớp xử lý

Ngoài các phương pháp đã nêu, còn tồn tại nhiều phương pháp học sâu khác cho các mục đích khác nhau. Ví dụ, bản đồ tự tổ chức (SOM) sử dụng học không giám sát để biểu diễn dữ liệu nhiều chiều bằng bản đồ lưới 2D. Bộ mã hóa tự động (AE) giúp giảm kích thước và trích xuất đặc trưng trong học không giám sát.

Nhìn chung, các phương pháp học sâu đa dạng có thể được áp dụng cho nhiều mục đích trong nhiều lĩnh vực khác nhau.

3.4 – Thách thức và hướng nghiên cứu

Nghiên cứu của bài báo này tập trung vào các thuật toán học máy để phân tích thông tin từ dữ liệu và ứng dụng của chúng trong nhiều ngữ cảnh. Trong phần này, chúng tôi tóm tắt và thảo luận về những thách thức và cơ hội nghiên cứu trong tương lai.

Tổng thể, hiệu quả và hiệu suất của giải pháp học máy phụ thuộc vào bản chất của dữ liệu cũng như hiệu suất của các thuật toán. Việc thu thập dữ liệu đúng kiểu trong các lĩnh vực như an ninh mạng, IoT, chăm sóc sức khỏe và nông nghiệp, được thảo luận ở phần "Ứng dụng của Machine Learning", không đơn giản. Mặc dù khả năng tạo ra lượng dữ liệu lớn đang tăng lên, việc thu thập dữ liệu hữu ích cho các ứng dụng học máy vẫn còn là thách thức quan trọng. Dữ liệu thường chứa nhiễu, giá trị bị thiếu và dữ liệu không có ý nghĩa rõ ràng, điều này cần xử lý cẩn thận.

Việc chọn thuật toán học máy phù hợp với mục tiêu ứng dụng cũng là một thách thức quan trọng. Một số thuật toán có thể cho kết quả khác nhau dựa trên đặc điểm của dữ liệu. Việc chọn sai thuật toán có thể dẫn đến kết quả không chính xác. Về mặt xây dựng mô hình, các kỹ thuật trong phần "Nhiệm vụ và thuật toán học máy" có thể giúp giải quyết nhiều vấn đề thực tế như an ninh mạng, chăm sóc sức khỏe và thành phố thông minh.

Tuy nhiên, việc kết hợp các phương pháp, tinh chỉnh hoặc thiết kế các phương pháp học máy mới có thể đóng một vai trò quan trọng trong tương lai. Sự thành công cuối cùng của giải pháp dựa trên học máy phụ thuộc vào cả dữ liệu và thuật toán học. Việc xử lý hiệu quả dữ liệu và sử dụng các thuật toán học máy đa dạng sẽ đóng góp quan trọng vào xây dựng các ứng dụng thông minh.

3.5 – Kết luận:

Nghiên cứu của bài báo tập trung vào việc sử dụng các thuật toán học máy để phân tích thông tin từ dữ liệu và áp dụng chúng trong nhiều ngữ cảnh khác nhau. Trong phần này, chúng tôi tổng kết và thảo luận về những thách thức và cơ hội mà chúng tôi đã gặp trong quá trình nghiên cứu và cách chúng có thể ảnh hưởng đến tương lai của lĩnh vực này.

Tổng thể, hiệu suất và hiệu quả của giải pháp dựa trên học máy phụ thuộc vào cả tính chất của dữ liệu và khả năng hoạt động của các thuật toán học. Việc thu thập dữ liệu phù hợp trong các lĩnh vực như an ninh mạng, IoT, chăm sóc sức khỏe và nông nghiệp, đã được thảo luận tại phần "Ứng dụng của Machine Learning", đôi khi không đơn giản. Mặc dù khả năng sản xuất dữ liệu lớn đang gia tăng, việc thu thập dữ liệu có giá trị cho ứng dụng học máy vẫn là thách thức quan trọng. Thường xuyên, dữ liệu chứa nhiều, thiếu thông tin và không rõ ràng, điều này yêu cầu quá trình xử lý cẩn thận.

Việc lựa chọn thuật toán học máy thích hợp cho mục tiêu ứng dụng cũng đứng trước thách thức quan trọng. Hiệu quả của các thuật toán có thể thay đổi tùy thuộc vào đặc điểm dữ liệu. Lựa chọn sai thuật toán có thể dẫn đến kết quả không chính xác. Về việc xây dựng mô hình, những kỹ thuật được thảo luận tại phần "Nhiệm vụ và thuật toán

học máy" có thể giải quyết nhiều vấn đề thực tế như an ninh mạng, chăm sóc sức khỏe và đô thị thông minh.

Tuy nhiên, việc kết hợp các phương pháp, điều chỉnh hoặc thiết kế các phương pháp học máy mới có thể đóng một vai trò quan trọng trong tương lai. Thành công cuối cùng của giải pháp học máy và các ứng dụng liên quan chủ yếu phụ thuộc vào cả dữ liệu và thuật toán. Việc xử lý dữ liệu hiệu quả và tận dụng đa dạng thuật toán học máy sẽ đóng góp quan trọng vào việc phát triển các ứng dụng thông minh.

TÀI LIỆU THAM KHẢO

- [Breast Cancer Classification using KNN - Coding Ninjas](#)
- [Breast Cancer Classification using SVM and KNN | IEEE Conference Publication | IEEE Xplore](#)
- [Breast Cancer Prediction by KNN Classification | Kaggle](#)
- [Implementing KNN Algorithm for detecting Breast Cancer - YouTube](#)
- [Paper for projects - Google Drive](#)
- Slide bài tập và các Lab của cô Võ Thị Hồng Thắm

LINK SOURCE CODE:

<https://drive.google.com/drive/folders/1nIiPZwwZv9V91xf07S2BbvyUX8HzmofU?usp=sharing>