TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP HỒ CHÍ MINH

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

BÁO CÁO PROJECT CUỐI KÌ

Môn học: MACHINE LEARNING

Nội dung: PREDICTING LUNG DISEASE WITH ML

GVHD: TS. Vũ Quang Huy

SVTH:

Bùi Trung Kiên	21146115
Võ Thành Đạt	21146450

Mã lớp học: MALE337029_01CLC

Thành phố Hồ Chí Minh, tháng 5 năm 2024

MỤC LỤC

HỌC KÌ II – NĂM HỌC 2023-2024

MÃ LỚP: MALE337029_01CLC

Tên đề tài: Predicting lung disease with machine learning

STT	Họ và tên	MSSV	Tỉ lệ % hoàn thành công việc
1	Bùi Trung Kiên	21146115	100%
2	Võ Thành Đạt	21145450	100%

Ghi chú:

Tỷ lệ % =100%: Mức độ phần tram của từng sinh viên tham gia

Nhận xét của giảng viên:

……..
……..
…….
……..
…….
……..

…….
……...……
…….
…….

…….
…….

…….

…….

Ngày tháng năm 2024

GVHD

Lời cảm ơn

Lời nói đầu tiên, nhóm chúng em bày tỏ long biết ơn sâu sắc và lời cảm ơn chân thành nhất đến giảng viên môn Machine Learning là thầy Vũ Quang Huy đã dành nhiều thời gian và công sức để truyền đạt những kiến thức sâu rộng về Machine Learning .Các bài tập và hướng dẫn của thầy đã giúp chúng em nắm vững các khái niệm , mô hình , thuật toán về học máy từ đó có thể ứng dụng nó vào thực tế ví dụ như chúng em có thể có kiến thức để hoàn thành các vấn đề còn nan giải trong báo cáo cuối kì của nhóm.

Tuy nhiên, do kiến thức và kỹ năng của bản thân còn nhiều hạn chế, nên chắc chắn đồ án của chúng em không thể tránh khỏi những thiếu sót. Vì vậy, chúng em rất mong nhận được những ý kiến đóng góp quý báu từ thầy cùng các bạn, để đồ án có thể được hoàn thiện một cách tốt nhất.

Để kết thúc, một lần nữa chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới thầy - người đã dày công truyền đạt kiến thức, động viên và khích lệ tinh thần chúng em không ngừng phấn đấu, rèn luyện bản thân. Chúng em kính chúc thầy luôn dồi dào sức khỏe, thành công hơn nữa trên con đường hoàn thành sứ mệnh cao cả là trồng người, và tiếp tục đồng hành cùng nhiều thế hệ sinh viên, đóng góp nhiều hơn nữa cho sự nghiệp giáo dục đào tạo của đất nước.

Chương I. Giới thiệu tổng quan:

1. Giới thiệu về đề tài và lý do chọn đề tài:

Đề tài nhóm chúng em chọn cho cuối kì là dự đoán tỷ lên người mắc bệnh phổi

Như chúng ta biết sau quãng thời gian 2 năm covid-19 đã để lại cho chúng ta một hậu quả vô cùng nghiêm trọng đặc biệt là vấn đề về sức khỏe, càng ngày có nhiều người bị bệnh phổi nhiều hơn. Và việc dự đoán và dự phòng các bệnh phổi trở nên hết sức cấp thiết, nhằm giúp các cơ quan y tế có thể quản lý và ứng phó với tình hình.

Đề tài này cho phép chúng em có thể vẫn dụng các kỹ thuật Machine Learning như phân loại, hồi quy, clustering để xây dựng mô hình dự đoán.

Việc thực hành và áp dụng các kiến thức Machine Learning vào một bài toán thực tế sẽ giúp chúng tôi nắm vững hơn các khái niệm và kỹ thuật đã học.

Với những lý do trên, chúng em tin rằng việc chọn đề tài dự đoán đoán tỷ lệ người mắc bệnh phổi là một lựa chọn phù hợp và đem lại tính thực tiễn cao, giúp chúng em vừa áp dụng kiến thức ML vừa góp phần giải quyết được vấn đề thiết yếu hiện nay.

2. Nội dung thực hiện:

- Chương 2: Cở sở lý thuyết

  + Trên tiền đề cơ sở lý thuyết mà tụi em đã được học chúng em sẽ trình bày về các thuật toán Machine Learning được sử dụng trong đề tài bao gồm ưu - nhược điểm và các ứng dụng thực tế của từng thuật toán.

  + Các thuật toán dự kiến được đề cập gồm: Regression, Classification, Clustering.... Và cũng trình bày cơ sở lý thuyết về các kỹ thuật tiền xử lý dữ liệu, feature engineering, và phương pháp đánh giá mô hình.

- Chương 3: Đặc trưng dữ liệu

  + Trong chương này, nhóm sẽ mô tả chi tiết về các đặc trưng (features) của tập dữ liệu đã chuẩn bị để huấn luyện mô hình.

  + Nhóm sẽ phân tích các đặc trưng về mặt thống kê, bao gồm phân phối, tương quan, ... Từ đó, nhóm sẽ lựa chọn các đặc trưng phù hợp để đưa vào mô hình.

  + Nhóm cũng sẽ trình bày về các kỹ thuật xử lý dữ liệu như xử lý giá trị bị khuyết, xử lý ngoại lệ, chuẩn hóa dữ liệu, ...

- Chương 4: Lưu mô đồ để huấn luyện mô hình

+ Trong chương này, nhóm sẽ trình bày lưu đồ giải thuật chi tiết của các bước trong quá trình xây dựng mô hình.

+ Lưu đồ sẽ bao gồm các bước chính như: tiền xử lý dữ liệu, lựa chọn mô hình, huấn luyện mô hình, đánh giá mô hình.

+ Việc trình bày lưu đồ giải thuật sẽ giúp người đọc hiểu rõ hơn về các bước triển khai mà nhóm thực hiện.

- Chương 5: Triển khai code

+ Dựa trên lưu đồ giải thuật ở chương trước, nhóm sẽ trình bày chi tiết về việc triển khai code thực hiện các bước xây dựng mô hình.

+ Nhóm sẽ sử dụng các thư viện machine learning phổ biến như Sklearn, ... để cài đặt các thuật toán.

+ Trong phần này, nhóm sẽ chia sẻ các đoạn code quan trọng và giải thích chức năng của chúng.

+ Ở chương này, nhóm sẽ trình bày kết quả của mô hình sau khi huấn luyện và đánh giá.

Nhóm sẽ phân tích các chỉ số đánh giá mô hình như accuracy, precision, recall, F1-score, ... Từ đó, nhóm sẽ đưa ra đánh giá về hiệu suất của mô hình.

+ Ngoài ra, nhóm cũng sẽ thảo luận về những hạn chế và thách thức trong quá trình xây dựng mô hình.

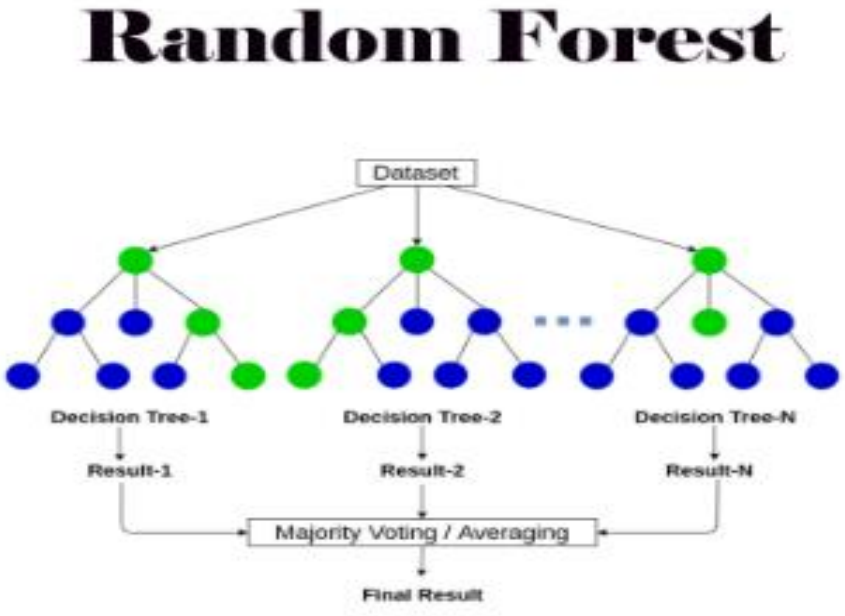
Chương II: Cơ sở lý thuyết:

1. Các thuật toán sẽ thực hiện trong đề tài:

1.1 Random Forest:

1.1.1 Khái niệm

Là một thuật toán học máy được sử dụng để phân loại và hồi quy. Nó xây dựng một số cây quyết định (decision trees) và kết hợp chúng để tạo ra một mô hình có độ chính xác cao hơn so với từng cây đơn lẻ. Mỗi cây trong Random Forest được xây dựng trên mọi tập dữ liệu ngẫu nhiên được chọn từ tập dữ liệu ban đầu.



1.1.2 Ưu và nhược điểm của thuật toán:

- Ưu điểm:
  - Có độ chính xác cao, đặc biệt với các bộ dữ liệu lớn và phức tạp.
  - Có khả năng xử lý dữ liệu nhiễu và dữ liệu nhiều chiều.
  - Kháng với hiện tượng overfitting.
  - Dễ triển khai và có thể sử dụng để giải quyết nhiều bài toán khác nhau.
  - Cung cấp thông tin về tầm quan trọng của biến dự báo.

- Nhược điểm:
  - Mô hình Random Forest khá phức tạp và không dễ hiểu như các mô hình đơn giản hơn.
  - Yêu cầu nhiều tài nguyên tính toán, đặc biệt là với các bộ dữ liệu lớn.
  - Không thể giải thích chi tiết các hoạt động của mô hình như các cây quyết định đơn lẻ.
  - Khó kiểm xóa và hiệu chỉnh mô hình so với các mô hình tuyến tính

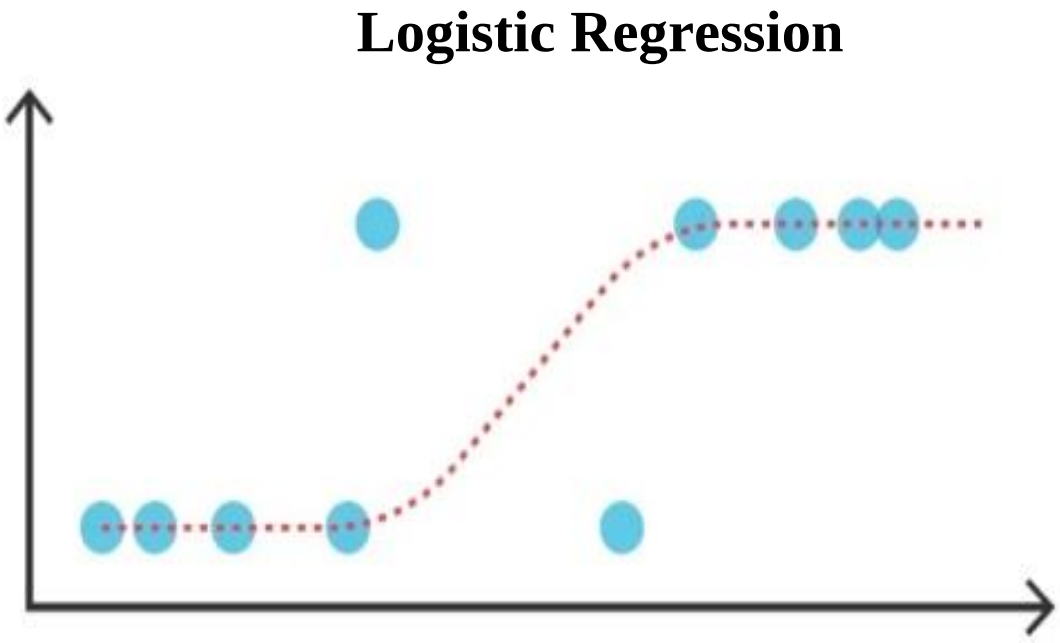
1.1.3 Ứng dụng của Random Forest:

- Phân loại và hồi quy (classification và regression).
- Chẩn đoán y tế (medical diagnosis).

1.2 Logistic Regression:

1.2.1 Khái niệm

Logistic Regression là một thuật toán học máy được sử dụng để dự đoán biến phụ thuộc định tính (categorical) dựa trên một hoặc nhiều biến độc lập. Nó sử dụng hàm Logistic để biến đổi giá trị dự đoán thành xác suất nằm trong khoảng 0 và 1, từ đó có thể phân loại đối tượng vào một trong các nhóm.



1.2.2 Ưu và nhược điểm của thuật toán:

- Ưu điểm**:**
  - Dễ hiểu, dễ triển khai và diễn giải kết quả.
  - Có thể làm việc với nhiều biến độc lập.
  - Cho kết quả dự đoán dưới dạng xác suất, hữu ích cho việc ra quyết định.
  - Không nhạy cảm với phân phối của dữ liệu như hồi quy tuyến tính.
  - Có thể được sử dụng để phân loại các vấn đề hai lớp hoặc đa lớp.
- Nhược điểm:
  - Giả định về tính tuyến tính giữa biến độc lập và biến phụ thuộc (thông qua hàm Logistic).
  - Nhạy cảm với multicollinearity (tương quan đa cộng tuyến) giữa các biến độc lập.
  - Không phù hợp với các vấn đề phi tuyến tính phức tạp.
  - Yêu cầu số lượng dữ liệu huấn luyện lớn để đạt độ chính xác cao.

- Kém hiệu quả với các tập dữ liệu có nhiều biến độc lập hoặc tương tác phức tạp giữa chúng.

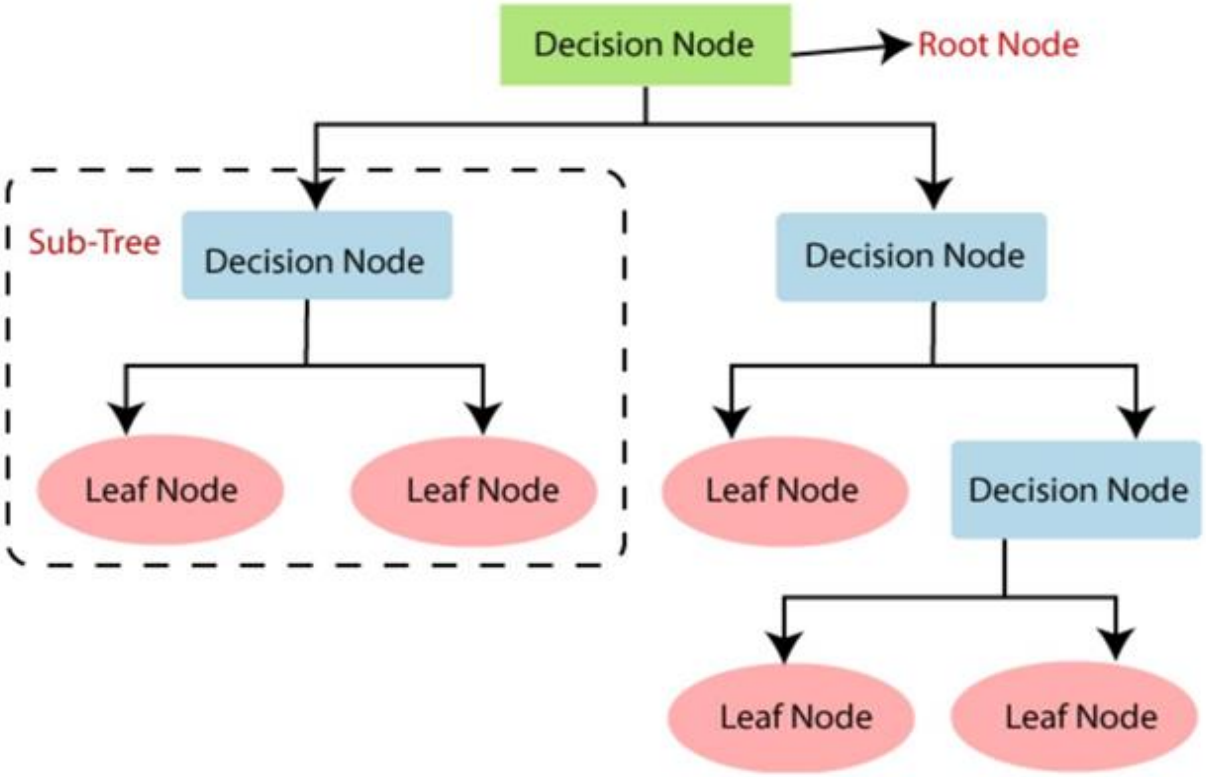
1.2.3 Ứng dụng của Logistic Regression:

- Dự đoán khả năng khách hàng sẽ mua một sản phẩm hoặc dịch vụ
- Phân loại khách hàng tiềm năng và lập chiến lược tiếp thị phù hợp.
- Dự đoán khả năng một bệnh nhân mắc một bệnh cụ thể dựa trên triệu chứng và lịch sử bệnh.
- Hỗ trợ bác sĩ trong việc đưa ra chẩn đoán và lựa chọn phương pháp điều trị phù hợp.

1.3 Decision Tree:

1.3.1 Khái niệm:

- Là một thuật toán học máy phổ biến được sử dụng cho cả bài toán phân loại và hồi quy.
- Ý tưởng cơ bản của Decision Tree là xây dựng một cây quyết định bằng cách phân chia tập dữ liệu thành các nhóm con, sao cho mỗi nhóm con chứa các điểm dữ liệu có tính chất tương tự nhau.
- Quá trình phân chia này tiếp tục được lặp lại cho đến khi mỗi nhóm con chứa các điểm dữ liệu thuộc cùng một lớp (trong trường hợp phân loại) hoặc có giá trị dự đoán gần nhau (trong trường hợp hồi quy).Mỗi nút trong cây quyết định đại diện cho một biến đầu vào và mỗi cạnh đại diện cho một quyết định (phân chia) dựa trên giá trị của biến đó. Các lá của cây đại diện cho các nhóm con cuối cùng.
- Khi dự đoán, dữ liệu mới được đưa vào cây và đi từ nút gốc xuống tới lá tương ứng với quyết định cuối cùng. Đối với bài toán hồi quy, giá trị dự đoán được tính dựa trên giá trị trung bình của các điểm dữ liệu trong lá tương ứng.



1.3.2 Ưu và nhược điểm của thuật toán:

- Ưu điểm:
  - Bằng cách kết hợp nhiều cây quyết định nên mô hình có khả năng giảm overfitting từ đó đưa ra những dự đoán có tính chính xác cao hơn.
  - Xử lí tốt những dữ liệu bị thiếu, nhiễu hoặc những dữ liệu phi tuyến.
  - Khả năng xử lí dữ liệu lớn và phức tạp một cách hiệu quả.
  - Ít bị ảnh hưởng bởi nhiễu vì thuật toán trung bình hóa các dự đoán từ nhiều cây.
- Nhược điểm:
  - Tính toán phức tạp nên yêu cầu nhiều tài nguyên tính toán và thời gian huấn luyên.
  - Việc chọn và điều chỉnh các siêu tham số có thể phức tạp yêu cầu kinh nghiệm và nhiều lần thử.
  - Cần phải sử dụng thêm một số các kỹ thuật giảm chiều của dữ liệu khi dữ liệu đưa vào là dữ liệu đa chiều

1.3.3 Ứng dụng của Decision Tree:

- Dự đoán khả năng khách hàng sẽ mua một sản phẩm hoặc dịch vụ
- Phân loại khách hàng tiềm năng và lập chiến lược tiếp thị phù hợp.
- Dự đoán khả năng một bệnh nhân mắc một bệnh cụ thể dựa trên triệu chứng và lịch sử bệnh.

- Hỗ trợ bác sĩ trong việc đưa ra chẩn đoán và lựa chọn phương pháp điều trị phù hợp.

Chương III. Dữ liệu huấn luyện:

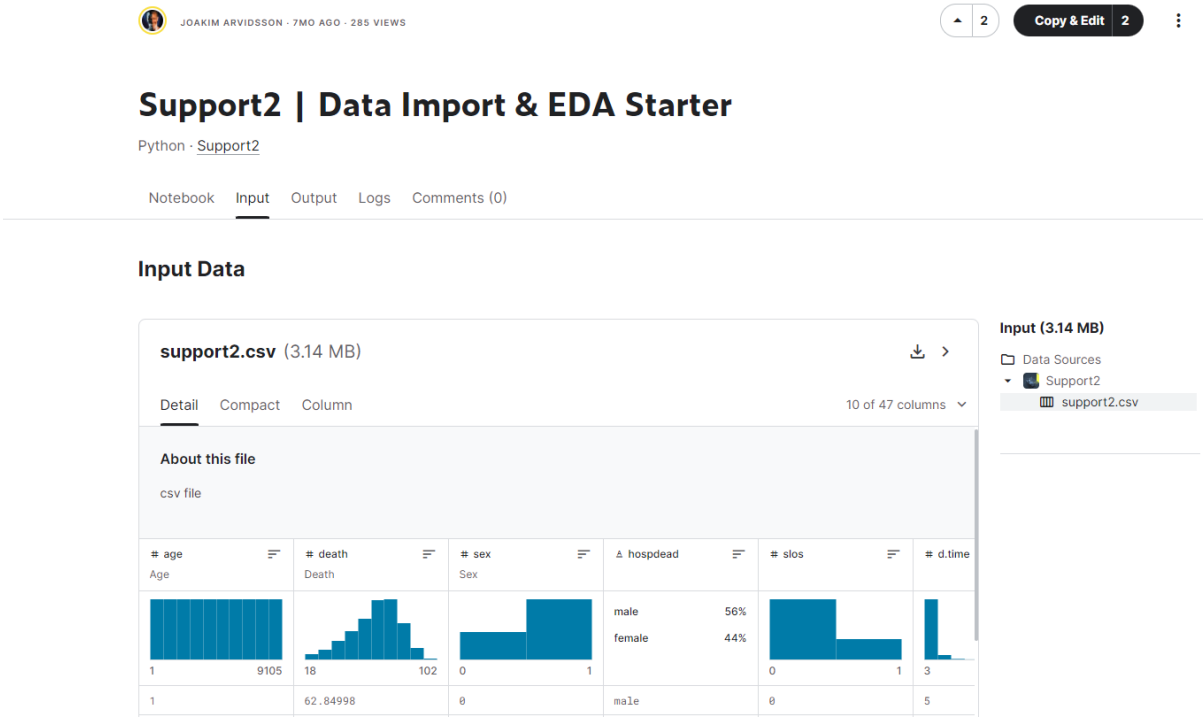
1. Cách huấn luyện:

- **Chuẩn bị dữ liệu:** Thu thập, làm sạch và chuẩn bị tập dữ liệu huấn luyện và kiểm tra. Việc này rất quan trọng vì chất lượng dữ liệu ảnh hưởng trực tiếp đến chất lượng của mô hình.
- **Chọn mô hình:** Chọn kiến trúc mô hình phù hợp với bài toán, ví dụ như mạng neural, hồi quy, phân loại, v.v. Điều này yêu cầu hiểu biết về các loại mô hình khác nhau và các ứng dụng của chúng.
- **Chọn siêu tham số:** Tùy chỉnh các siêu tham số bao của mô hình để tối ưu hóa hiệu suất. Điều nay có thể thực hiện thông qua chỉnh siêu tham số bằng các kỹ thuật như cross-validation.
- **Huấn luyện mô hình:** Chia dữ liệu thành để huấn luyện mô hình
- **Đánh giá mô hình:** Sử dụng các siêu tham số để đánh giá độ hiệu suất của mô hình
- **Tinh chỉnh mô hình:** Thực hiện tinh chỉnh mô hình bằng cách điều chỉnh siêu tham số hoặc sử dụng kỹ thuật tăng cường dữ liệu để cải thiện hiệu suất.
- **Dự đoán và đánh giá dữ liệu mới:** Sau khi chọn được các siêu tham số tối ưu, đánh giá hiệu suất của mô hình trên tập kiểm tra để đảm bảo mô hình hoạt động tốt trên dữ liệu mới.

2. Các đặc trưng đầu vào, đầu ra:

2.1 Các đặc trưng đầu vào:

- Tệp dữ liệu: https://www.kaggle.com/code/joebeachcapital/support2-data-import-eda-starter/input



```
df_clas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9105 entries, 1 to 9105
Data columns (total 45 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        9105 non-null   float64
 1   death      9105 non-null   int64
 2   sex        9105 non-null   object
 3   slos       9105 non-null   int64
 4   d.time     9105 non-null   int64
 5   dzgroup    9105 non-null   object
 6   dzclass    9105 non-null   object
 7   num.co     9105 non-null   int64
 8   edu        7471 non-null   float64
 9   income     6123 non-null   object
 10  scoma      9104 non-null   float64
 11  charges    8933 non-null   float64
 12  totcst     8217 non-null   float64
 13  totmcst    5630 non-null   float64
 14  avtisst    9023 non-null   float64
 15  race       9063 non-null   object
 16  sps        9104 non-null   float64
 17  aps        9104 non-null   float64
 18  surv2m     9104 non-null   float64
 19  surv6m     9104 non-null   float64
 20  hday       9105 non-null   int64
 21  diabetes   9105 non-null   int64
 22  dementia   9105 non-null   int64
 23  ca         9105 non-null   object
 24  prg2m      7456 non-null   float64
 25  prg6m      7472 non-null   float64
 26  dnr        9075 non-null   object
 27  dnrday     9075 non-null   float64
 28  meanbp     9104 non-null   float64
 29  wblc       8893 non-null   float64
 30  hrt        9104 non-null   float64
 31  resp       9104 non-null   float64
 32  temp       9104 non-null   float64
 33  pafi       6780 non-null   float64
 34  alb        5733 non-null   float64
 35  bili       6504 non-null   float64
 36  crea       9038 non-null   float64
 37  sod        9104 non-null   float64
 38  ph         6821 non-null   float64
 39  glucose    4605 non-null   float64
 40  bun        4753 non-null   float64
 41  urine      4243 non-null   float64
 42  adlp       3464 non-null   float64
 43  adls       6238 non-null   float64
 44  adlsc      9105 non-null   float64
dtypes: float64(31), int64(7), object(7)
memory usage: 3.2+ MB
```

- Đầu vào bao gồm 45 cột dữ liệu: age, death, sex, slos, d.time, dzgroup, dzclass, num.co, edu, income, scoma, charges, totcst, totmcst, avtisst, race, sps, aps, surv2m, surv6m, hday, diabetes, dementia, ca, prg2m, prg6m, dnr, dnrday, meanbp, wblc, hrt, resp, temp, pafi, alb, bili, crea, sod, ph, glucose, bun, urine, adlp, adls, adlsc
- Tập dữ liệu bao gồm kiểu float64(31), kiểu int64(7), object(7) có kích thước 3.2+ MB.
- Index: 9105 entries, 1 to 9105

2.2 Các đặc trưng đầu ra:

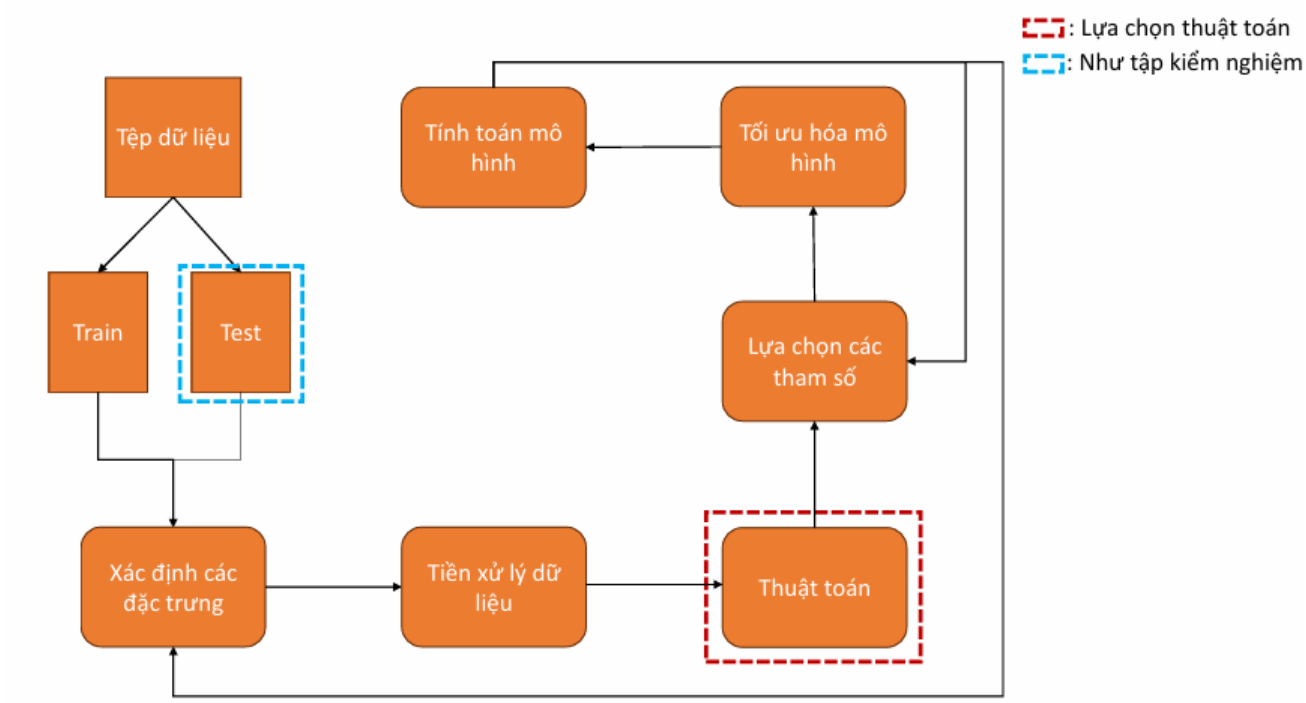
- Dự đoán tỷ lệ tử vong của người mắc bệnh phổi.

Chương IV. Lưu đồ thuật toán để huấn luyện mô hình:

1. Lưu đồ thuật toán:

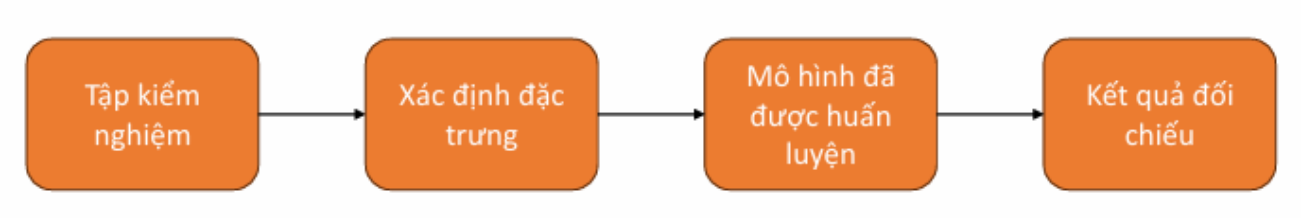
1.1 Các khối của lưu đồ thuật toán:

- Giai đoạn huấn luyện mô hình:



: bao gồm các thuật toán **Decision Tree, Logistic Regression, Random Forest.**

- Giai đoạn kiểm nghiệm:



1.2 Giải thích lưu đồ:

- Giai đoạn huấn luyện:
  - Theo như lưu đồ giải thuật ở trên đầu tiên ta sẽ phải chia tệp dữ liệu thành 2 tệp đó là tập train và tập test. Cần xác định các tính năng (features) của dữ liệu đầu vào và biến mục tiêu (target) của bài toán. Điều này sẽ ảnh hưởng lớn đến hiệu suất của mô hình. Sau khi chia dữ liệu và xác định đặc trưng đầu vào/đầu ra, cần tiến hành kiểm tra và tiền xử lý dữ liệu để đảm bảo dữ liệu được sạch và phù hợp với mô hình. Ngay sau đó là bước thử sai những siêu tham số nhằm chọn được những siêu tham số tăng hiệu suất của mô hình. Giai đoạn này là một vòng lặp nhằm chọn ra một mô hình phù hợp và hiệu suất cao nhất từ nhưng siêu tham số đã chọn.
- Giai đoạn kiểm nghiệm:
  - Ta sẽ áp dụng mô hình vào tập kiểm nghiệm để đánh giá độ chính xác và trích xuất đặc trưng để từ đó đưa vào mô hình để dự đoán và sẽ nhận được những kết quả dự đoán sau khi được mô hình xử lí hay còn gọi là kết quả mục tiêu.

Chương V. Code và chương trình huấn luyện.

1. Code của mô hình:

```
[1]  from google.colab import drive
     drive.mount('/content/drive')

Mounted at /content/drive
```

Ta kết nối với gg drive của mình

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
import seaborn as sns
```

- Ta khai báo thư viện
  - Thư viện numpy: Cung cấp các chức năng thao tác mảng hiệu quả.
  - Thư viện pandas: Dùng cho phân tích và thao tác dữ liệu ở định dạng bảng (dataframes).
  - Thư viện matplotlib: Cho phép trực quan hóa dữ liệu để khám phá và phân tích.
  - Thư viện sklearn:
    - train_test_split: Chia dữ liệu thành tập huấn luyện và tập thử nghiệm để đánh giá mô hình.
    - RandomForestClassifier: Triển khai thuật toán phân loại Random Forest.
    - DecisionTreeClassifier: Triển khai thuật toán phân loại Decision Tree.
    - LogisticRegression: Triển khai thuật toán phân loại Logistic Regression.
    - StandardScaler: Chuẩn hóa các tính năng bằng cách loại bỏ giá trị trung bình và tỷ lệ cho đơn vị phương sai.
    - SelectKBest: Chọn ra K tính năng hàng đầu dựa trên bài kiểm tra chi bình phương (f_classif).
    - f_classif: Tính toán giá trị F-ANOVA giữa nhãn và tính năng để chọn tính năng.
    - accuracy_score: Tính độ chính xác của dự đoán mô hình.
    - confusion_matrix: Tạo ma trận nhầm lẫn để đánh giá hiệu suất mô hình trên một nhiệm vụ phân loại.
    - SVC: Triển khai thuật toán phân loại Support Vector Machine (SVM).

- Thư viện Seaborn: Được xây dựng trên nền matplotlib để tạo đồ họa thống kê thông tin.

```
[ ]  df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/support2.csv')
```

- Ta đọc tệp csv bằng hàm **pd.read_csv():**Tham số này chỉ định đường dẫn đến tệp CSV bạn muốn đọc. Trong trường hợp này, tệp nằm trong Google Drive của bạn trong thư mục "Colab Notebooks".
- Đoạn mã này đọc dữ liệu từ tệp CSV có tên "support2.csv" nằm trong thư mục Colab Notebooks trên Google Drive của bạn và lưu trữ nó trong Pandas DataFrame có tên df. Sau đó, bạn có thể sử dụng biến df này để truy cập, phân tích và thao tác dữ liệu để sử dụng thêm trong các tác vụ học máy của mình.

```
[ ]  num_cols = [x for x in df.select_dtypes(include=np.number)]
     cat_cols = [x for x in df.select_dtypes(exclude=np.number)]
     print("There are", len(num_cols), "numerical columns and", len(cat_cols), "categorical columns in the dataset\n")
     print("Numerical columns:", num_cols)
     print("Categorical columns:", cat_cols)
```

- **num_cols = [x for x in df.select_dtypes(include=np.number)]:** Dòng này tạo ra một danh sách mới có tên *num_cols*. Nó duyệt qua tất cả các cột trong dataframe df bằng cách sử dụng *df.select_dtypes()*. Hàm này trả về một dataframe mới chỉ chứa các cột có kiểu dữ liệu là số (kiểu numerical). Biểu thức *include=np.number* chỉ định chỉ chọn các cột có dtype là kiểu số theo thư viện numpy (ký hiệu bằng np).
- **cat_cols = [x for x in df.select_dtypes(exclude=np.number)]**: Dòng này tương tự như dòng trước, nhưng nó sử dụng exclude=np.number để chọn các cột không phải là kiểu số. Do đó, danh sách cat_cols sẽ chứa các cột theo nhóm (categorical).
- **print("There are", len(num_cols), "numerical columns and", len(cat_cols), "categorical columns in the dataset\n")**: Dòng này in ra tổng số cột theo nhóm và cột số trong dataframe.
- **print("Numerical columns:", num_cols)**: In ra danh sách các tên cột theo thứ tự mà chúng xuất hiện trong dataframe df.
- **print("Categorical columns:", cat_cols)**: In ra danh sách các tên cột theo nhóm.

```
df_clas = df.drop(['hospdead', 'sfdm2'], axis = 1)
df_clas
```

- Ta đặt df_clas là biến mới có giá trị được lấy các dataframe df sử dụng hàm drop() để loại bỏ 2 cột là ['hospdead', 'sfdm2'] trong dataframe và tham số này được thực hiện trên axis =1 tức thực hiện trên cột.

```
[ ]  df_clas.info()
```

- Nó cho chúng ta biết được thông tin về loại dữ liệu , số lượng cột, giá trị không null, giá trị null, sử dụng bộ nhớ. Mục đích để xác định loại dữ liệu , kiểm tra giá trị thiếu và đánh giá cấu trúc dữ liệu.

```
# Target values
print('Real data:', df_clas['death'].value_counts())
print('Percentage values:', df_clas['death'].value_counts(normalize = True)*100) # Calculating relative frequencies

# Distribution of classes: Death (1) or Not Death (0)
unique, counts = np.unique(df_clas[['death']], return_counts=True)
plt.title('Distribution of classes')
sns.barplot(x=unique, y=counts)
```

- **df_clas['death']**: Lấy chọn cột 'death' từ DataFrame df_clas. Giả định rằng 'death' là biến mục tiêu nhị phân cho biết liệu một điểm dữ liệu thuộc về lớp tử vong (1) hay không tử vong (0).
- **print('Real data:', df_clas['death'].value_counts()):**
  - Dòng này sử dụng phương thức value_counts() để đếm số lần xuất hiện của mỗi giá trị duy nhất trong cột 'death'.
  - Kết quả (print) hiển thị số lượng thực tế các điểm dữ liệu thuộc về mỗi lớp (tử vong hoặc không tử vong).
- **print('Percentage values:', df_clas['death'].value_counts(normalize = True)*100):**
  - Dòng này cũng sử dụng value_counts() nhưng với đối số normalize=True.
  - Chuẩn hóa tính toán tần suất tương đối (tỷ lệ phần trăm) của mỗi lớp.
  - Kết quả (print) hiển thị tỷ lệ phần trăm các điểm dữ liệu trong mỗi lớp. Bằng cách nhân với 100, nó trình bày các giá trị dưới dạng tỷ lệ phần trăm.
- **np.unique(df_clas[['death']], return_counts=True):**
  - Dòng này sử dụng hàm np.unique từ thư viện NumPy (np).
  - Nó trích xuất các giá trị duy nhất (0 và 1) từ cột 'death' và đếm số lần xuất hiện của chúng.
  - Đối số return_counts=True đảm bảo rằng cả giá trị duy nhất và số đếm tương ứng của chúng được trả về dưới dạng tuple (unique, counts).
- **plt.title('Distribution of classes'):**
  - Dòng này đặt tiêu đề cho biểu đồ thanh sẽ được tạo.
- **sns.barplot(x=unique, y=counts):**
  - Dòng này nhập thư viện seaborn (sns) để trực quan hóa (giả sử nó được nhập).
  - Nó tạo ra biểu đồ thanh bằng cách sử dụng sns.barplot().
  - Đối số x chỉ định các giá trị duy nhất (0 và 1) trên trục x đại diện cho các lớp.
  - Đối số y chỉ định số đếm tương ứng trên trục y đại diện cho số lượng điểm dữ liệu trong mỗi lớp.

```
features = df_clas.drop('death', axis = 1)
targets = df_clas['death']

# Encode our targets using one-hot encoding
targets_onehot = pd.get_dummies(targets)
targets_onehot.head()
```

- **features = df_clas.drop('death', axis = 1):** Dòng này tạo một DataFrame mới có tên features chứa tất cả các cột từ df_clas ngoại trừ cột 'death'. Phương thức drop() được sử dụng để loại trừ cột được chỉ định (axis = 1 biểu thị cột). Thao tác này phân tách các đặc trưng (biến độc lập) khỏi biến mục tiêu (biến phụ thuộc, 'death' trong trường hợp này).
- **targets = df_clas['death']:** Dòng này tạo một biến mới có tên targets chứa cột 'death' từ df_clas. Nó chỉ chọn biến mục tiêu để xử lý tiếp theo.
- **targets_onehot = pd.get_dummies(targets):** Dòng này áp dụng mã hóa one-hot cho biến targets.
- **targets_onehot.head()**: Dòng này hiển thị một vài hàng đầu tiên của DataFrame targets_onehot. Điều này cho phép bạn xem cách mã hóa one-hot biến đổi biến mục tiêu.

```
[ ] features_onehot = pd.get_dummies(features)
    features_onehot.shape
```

- Với đầu vào là một dataframe là features sử dụng hàm pd.get_dummies để mã hóa các one-hot các biến phân loại được gán vào biến features_onehot
- Sau đó ta shape nó về dạng kích thước.

```
features_onehot = features_onehot.astype('float32')
targets_onehot = targets_onehot.astype('float32')
#features_onehot.info()
```

- **features_onehot = features_onehot.astype('float32')** và **targets_onehot = targets_onehot.astype('float32'):** sử dụng phương thức .astype của DataFrame trong Pandas, nó cho phép thay đổi kiểu dữ liệu toàn bộ datafram của features_onehot thành kiểu dữ liệu float32.

```
# Checking for NaN values in targets_onehot
nan_check_targets = targets_onehot.isna().sum()
#print("\nNaN values in targets_onehot:\n", nan_check_targets)

# Checking for NaN values in features_onehot
nan_check_features = features_onehot.isna().sum()

# Displaying features with NaN values
features_with_nan = nan_check_features[nan_check_features > 0]
#print("\nFeatures with NaN values:\n", features_with_nan)
```

- **nan_check_targets = targets_onehot.isna().sum():**
  - Dòng lệnh này áp dụng .isna() cho targets_onehot để xác định các giá trị thiếu.
  - Sau đó, nó sử dụng phương thức .sum() để tính tổng số lượng giá trị thiếu (giá trị True) trên tất cả các cột trong DataFrame kết quả (từ .isna()).
  - Kết quả (nan_check_targets) là một Series chứa số lượng giá trị thiếu cho mỗi cột.
- **nan_check_features = features_onehot.isna().sum():**

- Dòng lệnh này thực hiện thao tác tương tự trên features_onehot để đếm các giá trị thiếu trong mỗi cột và lưu trữ kết quả trong nan_check_features.

- **features_with_nan = nan_check_features[nan_check_features > 0]:**
  - Dòng lệnh này sử dụng lọc boolean trên nan_check_features.
  - Nó chỉ chọn những tính năng (cột) mà số lượng giá trị thiếu (nan_check_features) lớn hơn 0 (nghĩa là có ít nhất một số giá trị thiếu trong cột đó).
  - Kết quả (features_with_nan) là một Series chứa tên các tính năng có giá trị thiếu.

```
# Impute NaN values in numerical columns with mean
features_onehot = features_onehot.apply(lambda col: col.fillna(col.mean()) if col.dtype == 'float32' else col)

# Impute NaN values in categorical columns
features_onehot = features_onehot.fillna('Unknown')
```

- **features_onehot = features_onehot.apply(lambda col: col.fillna(col.mean()) if col.dtype == 'float32' else col):**
  - Dòng lệnh này áp dụng một hàm tùy chỉnh (được xác định bằng cú pháp lambda) cho mỗi cột trong features_onehot.
  - Tham số axis=0 chỉ định rằng hàm được áp dụng theo cột.
- **Phân tích hàm Lambda:**
  - Hàm lambda nhận một đối số duy nhất col đại diện cho một cột từ features_onehot.
  - Hàm đầu tiên kiểm tra kiểu dữ liệu của cột bằng cách sử dụng col.dtype.
- **Cột số (float32):**
  - if col.dtype == 'float32': Điều kiện này kiểm tra xem kiểu dữ liệu của cột có phải là float32 hay không (mà bạn có thể đã xác nhận trước đó).
  - Nếu đúng, hàm điền giá trị thiếu (NaN) trong cột đó bằng giá trị trung bình của cột (col.mean ()). Thao tác này thay thế NaN bằng giá trị trung bình của dữ liệu số trong cột.
- **Cột phân loại:**
  - else col: Phần này được áp dụng nếu kiểu dữ liệu không phải là float32 (giả định dữ liệu phân loại).
  - Trong trường hợp này, nó thay thế các giá trị thiếu (NaN) bằng chuỗi ký tự 'Unknown'. Đây là một chiến lược phổ biến cho việc xử lý giá trị thiếu dữ liệu phân loại, gán nhãn danh mục không xác định cho các giá trị thiếu.

```
train_features, test_features, train_targets, test_targets = train_test_split(features_onehot, targets_onehot, test_size=0.2, stratify=targets)
```

- **train_test_split():** Đây là một hàm trong thư viện sklearn.model_selection dùng để chia dữ liệu thành 2 tập - tập huấn luyện và tập kiểm tra.
- **features_onehot:** Đây là tập dữ liệu đặc trưng (features) đã được mã hóa one-hot, nghĩa là các biến phân loại được biến đổi thành biến nhị phân.
- **targets_onehot:** Đây là tập dữ liệu nhãn (targets) đã được mã hóa one-hot, ví dụ với bài toán phân loại đa lớp.
- **test_size=0.2**: Tỉ lệ dữ liệu sẽ được đưa vào tập kiểm tra, ở đây là 20%.

- **stratify=targets**: Việc chia dữ liệu dựa trên tỉ lệ của các nhãn (targets) để đảm bảo tập kiểm tra và tập huấn luyện có cùng phân bố nhãn.

```
[ ] print('Number of samples in the test set:', len(test_targets))
    print('Number of classes in the classification problem:', len(set(targets)))
```

- Cho ta thấy được số mẫu trong tập kiểm trai và số lớp trong bài toán phân loại.

```
[ ] rf_model = RandomForestClassifier(random_state=42,max_depth=2)
    D_tree = DecisionTreeClassifier(random_state=42)
    LR = LogisticRegression(max_iter=1000,random_state=42)
```

- Cho ta thấy định nghĩa của ba mô hình: Random Forest Classifier, Decision Tree Classifier, Logistic Regression
  - Random Forest Classifier:
    - Sử dụng RandomForestClassifier từ thư viện sklearn.ensemble.
    - Đặt random_state=42 để đảm bảo tính toán lặp lại.
    - Thiết lập max_depth=2 để giới hạn độ sâu tối đa của các cây quyết định trong Random Forest.
  - Decision Tree Classifier:
    - Sử dụng DecisionTreeClassifier từ thư viện sklearn.tree.
    - Cũng đặt random_state=42 để đảm bảo tính toán lặp lại.
  - LogisticRegression:
    - Sử dụng LogisticRegression từ thư viện sklearn.linear_model.
    - Thiết lập max_iter=1000 để tăng số lần lặp tối đa trong quá trình huấn luyện.
    - Đặt random_state=42 để đảm bảo tính toán lặp lại.

```
rf_model.fit(train_features, train_targets)
pred = rf_model.predict(test_features)
acc = accuracy_score(test_targets,pred)
print('Độ chính xác bằng phương pháp RandomForest :', acc*100,'%')
```

```
Độ chính xác bằng phương pháp RandomForest : 78.14387699066447 %
```

- Sử dụng phương thức fit() của đối tượng rf_model để huấn luyện mô hình trên tập dữ liệu huấn luyện train_features và train_targets.
- Sử dụng phương thức predict() của mô hình rf_model để dự đoán nhãn của các mẫu trong tập kiểm tra test_features, lưu kết quả vào biến pred.
- Sử dụng hàm accuracy_score() từ thư viện sklearn.metrics để tính độ chính xác của mô hình trên tập kiểm tra, so sánh pred với test_targets.
- Lưu kết quả vào biến acc.
- In ra độ chính xác của mô hình Random Forest Classifier, với kết quả được nhân 100 để hiển thị dưới dạng phần trăm.

```
[ ] D_tree.fit(train_features, train_targets)
    pred1 = D_tree.predict(test_features)
    acc1 = accuracy_score(test_targets,pred1)
    print('Độ chính xác phương pháp DecisionTreeClassifier:', acc1*100,'%')
```

```
Độ chính xác phương pháp DecisionTreeClassifier: 90.82921471718835 %
```

- Sử dụng phương thức fit() của đối tượng D_tree để huấn luyện mô hình trên tập dữ liệu huấn luyện train_features và train_targets.
- Sử dụng phương thức predict() của mô hình D_tree để dự đoán nhãn của các mẫu trong tập kiểm tra test_features, lưu kết quả vào biến pred1.
- Sử dụng hàm accuracy_score() từ thư viện sklearn.metrics để tính độ chính xác của mô hình trên tập kiểm tra, so sánh pred1 với test_targets.
- Lưu kết quả vào biến acc1.
- In ra độ chính xác của mô hình Decision Tree Classifier, với kết quả được nhân 100 để hiển thị dưới dạng phần trăm.

```
scaler = StandardScaler()
train_features_scaled = scaler.fit_transform(train_features)
test_features_scaled = scaler.transform(test_features)

LR.fit(train_features_scaled, train_targets[1])
pred2 = LR.predict(test_features_scaled)
acc2 = accuracy_score(test_targets[1], pred2)
print('Độ chính xác phương pháp LogisticRegression:', acc2*100,'%')
```

```
Độ chính xác phương pháp LogisticRegression: 85.11806699615596 %
```

- Tạo một đối tượng scaler của lớp StandardScaler từ thư viện sklearn.preprocessing.
- Sử dụng phương thức fit_transform() để chuẩn hóa tập dữ liệu huấn luyện train_features, lưu kết quả vào train_features_scaled.
- Sử dụng phương thức transform() để chuẩn hóa tập dữ liệu kiểm tra test_features, lưu kết quả vào test_features_scaled.
- Sử dụng phương thức fit() của đối tượng LR để huấn luyện mô hình Logistic Regression trên tập dữ liệu huấn luyện train_features_scaled và train_targets[1].
- Sử dụng phương thức predict() của mô hình LR để dự đoán nhãn của các mẫu trong tập kiểm tra test_features_scaled, lưu kết quả vào biến pred2.
- Sử dụng hàm accuracy_score() từ thư viện sklearn.metrics để tính độ chính xác của mô hình trên tập kiểm tra, so sánh pred2 với test_targets[1].
- Lưu kết quả vào biến acc2.
- In ra độ chính xác của mô hình Logistic Regression, với kết quả được nhân 100 để hiển thị dưới dạng phần trăm.

Chương VI. Tài liệu tham khảo.

- Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning
- https://lms.simplilearn.com/courses/2789/Machine-Learning/syllabus
- https://scikit-learn.org/stable/index.html
- https://www.kaggle.com/code/joebeachcapital/support2-data-import-eda-starter/input
- https://ndquy.github.io/posts/cac-phuong-phap-scaling/
- https://www.geeksforgeeks.org/pandas-filling-nan-in-categorical-data/
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html