BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

BÀI TẬP LỚN

HỌC MÁY CƠ BẢN

DỰ ĐOÁN TÌNH TRẠNG SỨC KHỎE TOÀN DIỆN SỬ DỤNG SVM VÀ RANDOM FOREST

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **NGUYỄN QUỐC BẢO**

LỚP: **12423TN**

NGƯỜI HƯỚNG DẪN: **PGS.TS. NGUYỄN VĂN HẬU**

HƯNG YÊN – 2025

NHẬN XÉT

Nhận xét của giảng viên hướng dẫn:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIÁO VIÊN HƯỚNG DẪN

Nguyễn Văn Hậu

LỜI CAM ĐOAN

Em xin cam đoan bài tập lớn " Dự đoán tình trạng sức khỏe toàn diện sử dụng SVM và Random Forest " là kết quả thực hiện của bản thân em dưới sự hướng dẫn của thầy Nguyễn Văn Hậu.

Những phần sử dụng tài liệu tham khảo trong bài tập lớn đã được nêu rõ trong phần tài liệu tham khảo. Các kết quả trình bày trong đồ án và chương trình xây dựng được hoàn toàn là kết quả do bản thân em thực hiện.

Nếu vi phạm lời cam đoan này, em xin chịu hoàn toàn trách nhiệm trước khoa và nhà trường.

Hưng Yên, ngày ... tháng 01 năm 2026

Sinh viên

NGUYỄN QUỐC BẢO

LỜI CẢM ƠN

Để có thể hoàn thành bài tập lớn này, lời đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Học máy cơ bản, Khoa Công nghệ thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện bài tập lớn này.

Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Văn Hậu đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

Em cũng xin chân thành cảm ơn tất cả các Thầy, các Cô trong Trường đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết, quý báu để giúp em thực hiện được bài tập lớn này.

Mặc dù em đã có cố gắng, nhưng với trình độ còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hy vọng sẽ nhận được những ý kiến nhận xét, góp ý của các Thầy cô về những kết quả triển khai trong bài tập lớn.

Em xin trân trọng cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN BÀI TOÁN

1.1 Bài toán

Ngày nay, dưới tác động của nhịp sống hiện đại và công nghiệp hóa, con người ngày càng đối mặt với nhiều nguy cơ sức khỏe tiềm ẩn do lối sống thiếu cân bằng gây ra. Các bệnh lý không lây nhiễm (như béo phì, tim mạch, rối loạn giấc ngủ) và các vấn đề về sức khỏe tinh thần đang gia tăng nhanh chóng. Tình trạng sức khỏe suy giảm không chỉ xuất phát từ các yếu tố sinh học mà còn chịu ảnh hưởng sâu sắc bởi các thói quen sinh hoạt hàng ngày như chế độ dinh dưỡng, mức độ vận động, giấc ngủ, và trạng thái tâm lý. Trong nhiều trường hợp, sự suy giảm sức khỏe diễn ra âm thầm thông qua sự tích tụ của những thói quen xấu, và các triệu chứng chỉ trở nên rõ rệt khi cơ thể đã rơi vào tình trạng báo động hoặc mắc bệnh mãn tính, khiến việc điều trị trở nên tốn kém và khó khăn.

Mặc dù các tổ chức y tế và chuyên gia sức khỏe luôn khuyến cáo về lối sống lành mạnh, nhưng việc đánh giá chính xác tác động tổng hòa của các yếu tố lối sống lên sức khỏe cá nhân vẫn gặp nhiều trở ngại. Thực tế, các đợt khám sức khỏe định kỳ thường diễn ra thưa thớt, không phản ánh được bức tranh toàn diện về thói quen hàng ngày. Bản thân mỗi người cũng gặp khó khăn trong việc tự theo dõi và lượng hóa xem việc thiếu ngủ, uống ít nước, hay mức độ căng thẳng (stress) của ngày hôm nay sẽ ảnh hưởng cụ thể như thế nào đến điểm số sức khỏe tổng thể của họ. Việc đánh giá thủ công hoặc dựa trên cảm tính thường dẫn đến sự chủ quan, thiếu chính xác và không đưa ra được cảnh báo kịp thời.

Để giải quyết vấn đề này, tôi có ý tưởng ứng dụng máy học (Machine Learning) để dự đoán và đánh giá điểm số sức khỏe toàn diện (Overall Health Score), dựa trên dữ liệu về lối sống và chỉ số cơ thể được thu thập hàng ngày. Mô hình sẽ phân tích mối tương quan phức tạp giữa các yếu tố đầu vào như: mức độ hoạt động thể chất, chỉ số khối cơ thể (BMI), chất lượng giấc ngủ, mức độ căng thẳng (Stress Level), thói quen thiền định (Mindfulness), lượng nước tiêu thụ và thói quen ăn uống.

Mô hình học máy sẽ học từ dữ liệu lịch sử của nhiều cá nhân để nhận diện các mẫu hành vi dẫn đến điểm số sức khỏe thấp, từ đó đưa ra đánh giá khách quan về tình trạng sức khỏe hiện tại của người dùng. Nhờ vậy, hệ thống có thể cung cấp các cảnh báo sớm và gợi ý điều chỉnh lối sống được cá nhân hóa (ví dụ: cần ưu tiên cải thiện giấc ngủ hay tăng cường vận động), giúp người dùng chủ động ngăn ngừa bệnh tật và duy trì trạng thái sức khỏe tối ưu trước khi cần đến sự can thiệp y tế chuyên sâu.

1.2 Dữ liệu bài toán

Dữ liệu bài toán được lấy từ website:

https://www.kaggle.com/datasets/miadul/holistic-health-and-lifestyle-score-dataset

	Physical_Activity	Nutrition_Score	Stress_Level	Mindfulness	Sleep_Hours	Hydration	BMI	Alcohol	Smoking	Overall_Health_Score	Health_Status
0	54.934283	5.643011	5.696572	0.000000	6.292214	2.578565	24.275932	4.280610	8.984006	36.950187	Poor
1	42.234714	6.389001	5.566647	4.450144	8.519054	2.448713	25.970141	7.461846	3.223304	55.167774	Average
2	57.953771	5.805238	3.126960	9.129716	6.702720	3.261433	25.193857	0.000000	4.600482	78.304426	Good
3	75.460597	7.220836	6.159168	16.496689	7.135854	3.726265	19.527300	9.958423	3.947706	94.018274	Good
4	40.316933	9.394357	2.019835	25.241623	8.076086	3.049478	23.348229	4.320347	8.084322	100.000000	Good
5	40.317261	5.457916	3.691631	21.941359	6.911555	3.309235	26.284728	3.438906	6.716783	88.758120	Good

Hình 1.1: Tổng quan dữ liệu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Physical_Activity     10000 non-null  float64
 1   Nutrition_Score       10000 non-null  float64
 2   Stress_Level          10000 non-null  float64
 3   Mindfulness           10000 non-null  float64
 4   Sleep_Hours           10000 non-null  float64
 5   Hydration             10000 non-null  float64
 6   BMI                   10000 non-null  float64
 7   Alcohol               10000 non-null  float64
 8   Smoking               10000 non-null  float64
 9   Overall_Health_Score  10000 non-null  float64
 10  Health_Status         10000 non-null  object
dtypes: float64(10), object(1)
memory usage: 859.5+ KB
```

Hình 1.2: Thông tin dữ liệu

- Dữ liệu bài toán gồm các cột:

- Physical_Activity: Số phút hoạt động thể chất
- Nutrition_Score: Điểm dinh dưỡng
- Strees_Level: Mức độ căng thẳng
- Mindfulness: Thời gian dành chánh niệm
- Sleep_Hours: Số giờ ngủ trung bình mỗi đêm
- Hydration: Lít nước tiêu thụ mỗi ngày (0,5–5,0)
- BMI: Chỉ số khối cơ thể (18–40)
- Alcohol: Đơn vị rượu mỗi tuần (0–20)
- Smoking: Thuốc lá mỗi ngày (0–30)
- Overall_Health_Score: Điểm số sức khỏe
- Health_Status: Trạng thái sức khỏe

- Dữ liệu bài toán là 1 file csv gồm 10000 rows x 11 columns
- Sau khi mô tả dữ liệu, ta có:

```
df.describe()
```

	Physical_Activity	Nutrition_Score	Stress_Level	Mindfulness	Sleep_Hours	Hydration	BMI	Alcohol	Smoking	Overall_Health_Score
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	45.047069	6.966599	4.987202	15.224636	7.000194	2.503302	24.095086	3.523663	5.706911	78.227945
std	19.832871	1.883295	1.938195	9.454891	1.468580	0.801660	3.356663	3.270784	5.000260	19.697853
min	0.000000	0.000000	1.000000	0.000000	3.000000	0.500000	18.000000	0.000000	0.000000	2.217088
25%	31.548189	5.675978	3.599696	8.053871	6.003898	1.958461	21.653393	0.298894	1.065818	64.627060
50%	44.948100	7.031693	4.988464	14.896178	7.014341	2.506579	24.072122	2.980658	4.954994	81.118118
75%	58.421618	8.387730	6.327795	21.790305	8.025752	3.052666	26.380536	5.706382	8.991626	97.972163
max	120.000000	10.000000	10.000000	52.278333	10.000000	5.000000	36.376168	18.040621	27.978693	100.000000

Hình 1.3: Mô tả dữ liệu

1.3 Tiền xử lý

- Kiểm tra dữ liệu khuyết thiếu

```
df.isnull().sum()
```

	0
Physical_Activity	0
Nutrition_Score	0
Stress_Level	0
Mindfulness	0
Sleep_Hours	0
Hydration	0
BMI	0
Alcohol	0
Smoking	0
Overall_Health_Score	0
Health_Status	0

dtype: int64

Hình 1.4: Kiểm tra dữ liệu

- Chuẩn hóa dữ liệu liên tục

```
X = df[['Physical_Activity','Nutrition_Score','Stress_Level','Mindfulness','Sleep_Hours','Hydration','BMI','Alcohol','Smoking']]
y = df['Health_Status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Hình 1.5: Chuẩn hóa dữ liệu

- Xử lý dữ liệu mất cân bằng

```
smote = SMOTE()
X_res_scaled, y_res = smote.fit_resample(X_train_scaled, y_train)
print(y_res.value_counts())
```

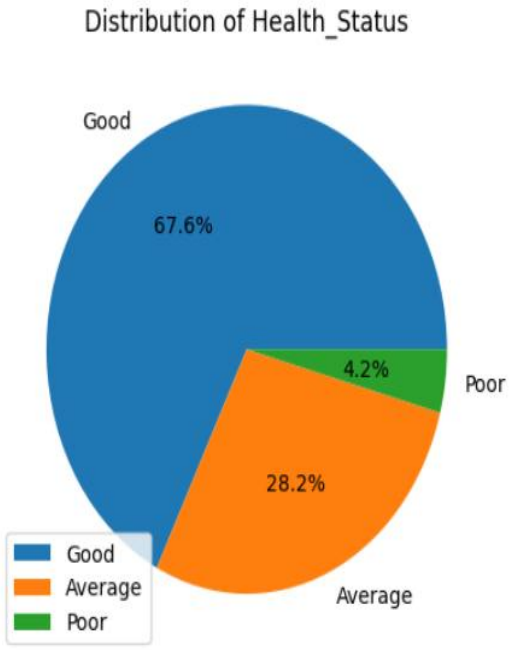
```
Health_Status
Average    5386
Good       5386
Poor       5386
Name: count, dtype: int64
```

Hình 1.6: Xử lý dữ liệu mất cân bằng

1.4 Thống kê và trực quan hóa dữ liệu

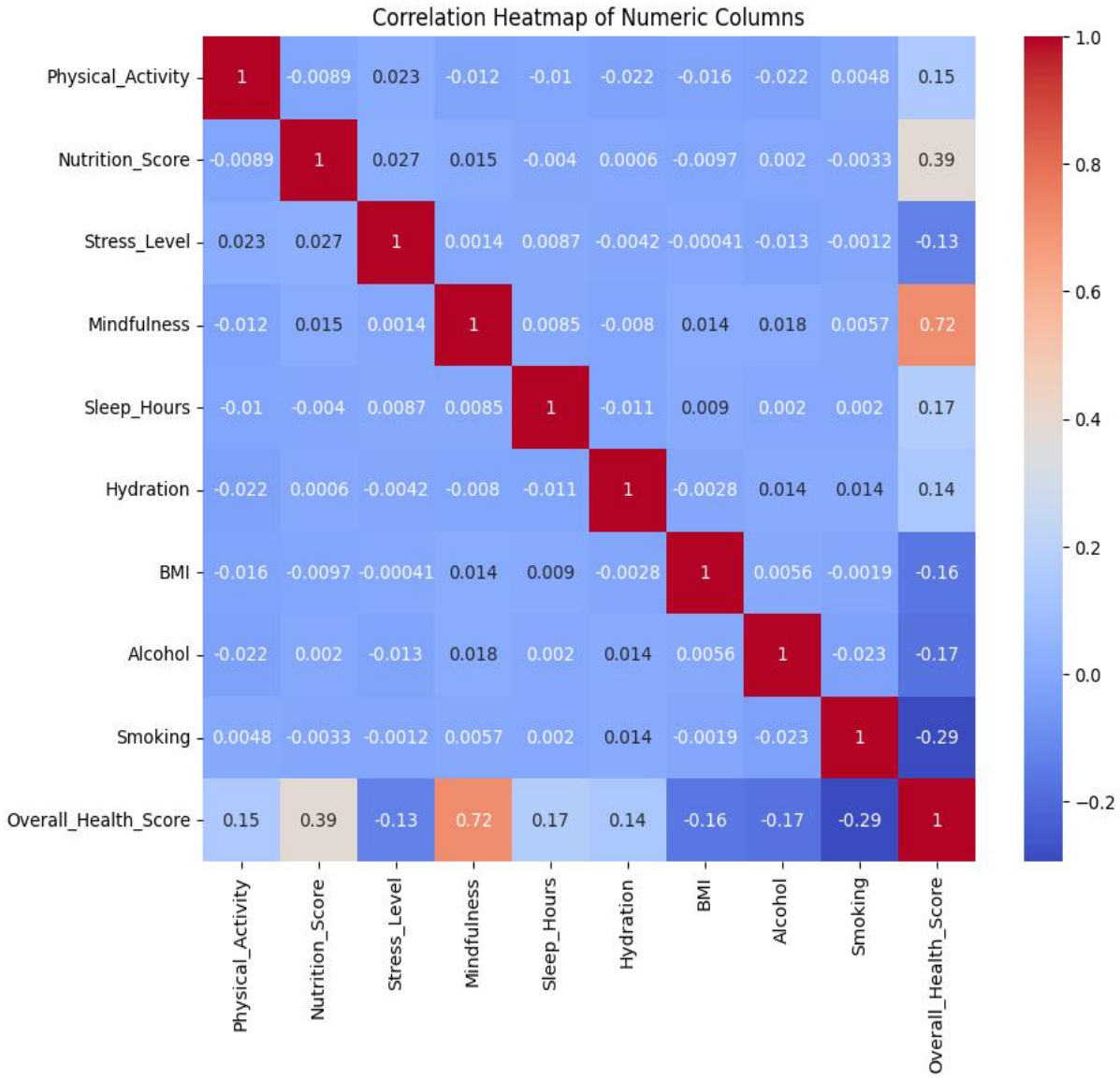
a) Tỉ lệ phần trăm của mỗi loại sức khỏe

```
plt.pie(df['Health_Status'].value_counts(), labels=df['Health_Status'].value_counts().index, autopct='%1.1f%%')
plt.title('Distribution of Health_Status')
plt.legend()
plt.show()
```



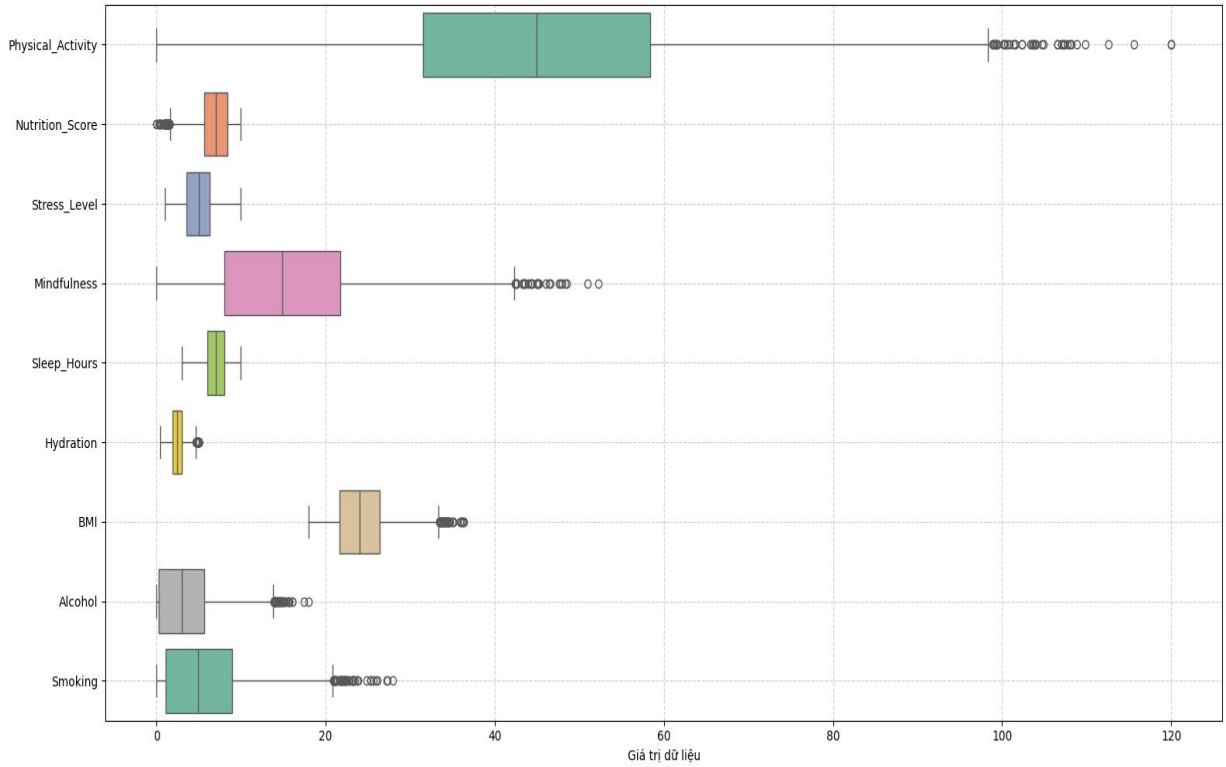
Hình 1.7: Tỉ lệ phần trăm các lớp

b) Ma trận tương quan giữa các thuộc tính



Hình 1.8: Ma trận tương quan

c) Biểu đồ Boxplot phát hiện ngoại lai



Hình 1.9: Biểu đồ boxplot

CHƯƠNG 2: THƯ VIỆN

2.1 Pandas

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh chóng, mạnh mẽ, linh hoạt và giàu tính diễn đạt. Tên của thư viện này bắt nguồn từ thuật ngữ "panel data" (dữ liệu bảng). Pandas được thiết kế để làm việc một cách dễ dàng và trực quan với các dữ liệu có cấu trúc (dạng bảng, đa chiều, có thể không đồng nhất) và dữ liệu chuỗi thời gian. Thư viện này thường được sử dụng để thao tác, phân tích và làm sạch dữ liệu. Pandas cung cấp đa dạng các cấu trúc dữ liệu và các phép toán hỗ trợ việc xử lý dữ liệu số và dữ liệu thời gian. Đây là một công cụ toàn diện để xử lý dữ liệu trong Python, phù hợp với nhiều loại tác vụ phân tích và quản lý dữ liệu. Thư viện Pandas trong Python là một thư viện mã nguồn mở, cung cấp sự hỗ trợ mạnh mẽ cho việc thao tác dữ liệu. Nó cũng là một bộ công cụ phân tích và xử lý dữ liệu đầy quyền năng của ngôn ngữ lập trình Python. Thư viện này được sử dụng rộng rãi trong cả lĩnh vực nghiên cứu và phát triển các ứng dụng khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu độc đáo được gọi là DataFrame. Pandas cung cấp rất nhiều hàm để thao tác và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã giúp Pandas trở nên phổ biến rộng rãi.

Tại sao chọn Pandas?

- Pandas rất phù hợp với nhiều loại dữ liệu khác nhau
- Dữ liệu dạng bảng với các cột có kiểu dữ liệu khác nhau (không đồng nhất), chẳng hạn như trong các bảng SQL hoặc bảng tính Excel.
- Dữ liệu chuỗi thời gian có trật tự và không có trật tự (không nhất thiết phải có tần suất cố định)
- Dữ liệu ma trận bất kỳ (đồng nhất hoặc không đồng nhất về kiểu dữ liệu) với các nhãn hàng và cột
- Bất kỳ dạng bộ dữ liệu quan sát/thống kê nào khác. Dữ liệu thực tế không bắt buộc phải được gán nhãn sẵn để có thể đưa vào cấu trúc dữ liệu của pandas

- Pandas được xây dựng trên nền tảng NumPy. Hai cấu trúc dữ liệu chính của pandas là Series (1 chiều) và DataFrame (2 chiều) có thể xử lý hầu hết các trường hợp sử dụng điển hình trong tài chính, thống kê, khoa học xã hội và nhiều lĩnh vực kỹ thuật

Ưu điểm của Pandas:

- Dễ dàng xử lý dữ liệu bị thiếu: Dữ liệu này được biểu diễn dưới dạng NaN (Not a Number), áp dụng cho cả dữ liệu dấu phẩy động (floating-point) lẫn các dạng dữ liệu khác tùy theo mong muốn của người dùng (có thể chọn bỏ qua hoặc gán về giá trị 0)
- Có thể thay đổi kích thước (Resizable): Các cột có thể được thêm vào hoặc xóa bỏ khỏi DataFrame và các đối tượng nhiều chiều khác một cách linh hoạt
- Căn chỉnh dữ liệu tự động và chủ động: Các đối tượng có thể được căn chỉnh theo chủ đích dựa trên một tập hợp nhãn (labels), hoặc người dùng có thể bỏ qua các nhãn và để cho Series, DataFrame, v.v., tự động căn chỉnh dữ liệu khi thực hiện các phép tính
- Linh hoạt trong tái cấu trúc dữ liệu: Hỗ trợ mạnh mẽ việc tái định hình (reshaping) và xoay trục (pivoting) các bộ dữ liệu
- Truy xuất dữ liệu thông minh: Hỗ trợ cắt lát (slicing) dựa trên nhãn, đánh chỉ mục nâng cao (fancy indexing) và trích xuất tập con (subsetting) từ các bộ dữ liệu lớn
- Hiệu suất cao

Cài đặt thư viện Pandas:

- Sử dụng pip và nhập câu lệnh: pip install pandas
- Với Anaconda, sử dụng lệnh: conda install pandas
- Khai báo thư viện Pandas: import pandas as pd

2.1.1 Series

Series([data, index, dtype, name, copy, . . .])

Series là một mảng một chiều tương tự như mảng Numpy, hoặc giống như một cột của bảng, nhưng nó bao gồm thêm một nhãn bổ sung để đánh dấu bảng (được gọi là chỉ

mục). Series có thể được khởi tạo thông qua NumPy, các kiểu Dict (từ điển), hoặc các kiểu dữ liệu vô hướng thông thường. Series có nhiều thuộc tính như index (chỉ mục), array (mảng), value (giá trị), dtype (kiểu dữ liệu), v.v. Bạn có thể thực hiện chuyển đổi Series sang một kiểu dữ liệu (dtype) cụ thể, tạo một bản sao của bảng, trả về giá trị boolean của một phần tử, chuyển đổi Series từ dạng DatetimeIndex sang PeriodIndex.

2.1.2 DataFrame

DataFrame([data, index, columns, dtype, copy])
DataFrame là một cấu trúc dữ liệu hai chiều có gắn nhãn, bao gồm các cột và các hàng giống như một bảng tính hoặc một bảng. Giống như Series, DataFrame có thể chứa bất kỳ loại dữ liệu nào. Một điều quan trọng cần nhấn mạnh là tất cả các cột trong một dataframe thực chất đều là các Pandas Series. Vì vậy, một DataFrame là sự kết hợp của nhiều Series đóng vai trò như các cột! DataFrame được sử dụng rộng rãi và là một trong những cấu trúc dữ liệu quan trọng nhất.

2.2 Matplotlib

Matplotlib là một thư viện vẽ đồ thị dành cho ngôn ngữ lập trình Python và là phần mở rộng của thư viện toán học NumPy. Nó cung cấp một API hướng đối tượng để nhúng các biểu đồ vào các ứng dụng sử dụng các bộ công cụ giao diện đồ họa (GUI) đa năng như Tkinter, wxPython, Qt, hoặc GTK. Matplotlib cho phép bạn tạo và hiển thị các đồ thị, hình ảnh và các dạng đồ họa khác, rất hữu ích cho việc trực quan hóa dữ liệu trong Python.

Một thành phần quan trọng của Matplotlib là Pyplot, một mô-đun cung cấp các hàm đơn giản để thêm các yếu tố biểu đồ như đường kẻ, hình ảnh, văn bản và nhiều thứ khác vào khung vẽ (hay còn gọi là axes).

Matplotlib là một trong những thư viện Python phổ biến nhất được sử dụng để trực quan hóa dữ liệu. Đây là một thư viện đa nền tảng dùng để tạo các đồ thị 2D từ dữ liệu dạng mảng. Matplotlib được viết bằng Python và sử dụng NumPy, phần mở rộng toán học của Python. Nó cung cấp một API hướng đối tượng để nhúng các biểu đồ vào ứng dụng và sử dụng các bộ công cụ GUI của Python như PyQt, wxPython và Tkinter. Nó

cũng có thể được sử dụng trong các trình bao (shell) Python và IPython, Jupyter Notebooks và các máy chủ web.

Matplotlib có một giao diện tên là Pylab, được thiết kế để có nét tương đồng với MATLAB - một ngôn ngữ lập trình độc quyền được phát triển bởi MathWorks. Matplotlib, kết hợp cùng với NumPy, có thể được coi là một phiên bản mã nguồn mở tương đương của MATLAB.

Matplotlib ban đầu được viết bởi John D. Hunter vào năm 2003. Phiên bản ổn định hiện tại (theo văn bản gốc) là 2.2.0, được phát hành vào tháng 1 năm 2018

2.2.1 Matplotlib được sử dụng để làm gì ?

Để thực hiện các suy luận thống kê cần thiết, việc trực quan hóa dữ liệu của bạn là điều cần thiết và Matplotlib là một giải pháp như vậy dành cho người dùng Python. Đây là một thư viện vẽ đồ thị rất mạnh mẽ, hữu ích cho những người làm việc với Python và NumPy. Mô-đun được sử dụng nhiều nhất của Matplotlib là Pyplot, cung cấp một giao diện giống như MATLAB, nhưng thay vào đó nó sử dụng Python và là mã nguồn mở.

Để cài đặt Matplotlib, nếu bạn có Anaconda, chỉ cần gõ lệnh conda install matplotlib hoặc sử dụng công cụ pip với lệnh pip install matplotlib.

2.2.2 General Concept

Một hình vẽ Matplotlib có thể được phân loại thành nhiều phần như sau :
Figure: Nó giống như một cửa sổ chứa tất cả mọi thứ bạn sẽ vẽ lên đó.
Axes: Thành phần chính của một figure là các axes (các khung nhỏ hơn để vẽ lên). Một figure có thể chứa một hoặc nhiều axes. Nói cách khác, figure chỉ là vật chứa (container), còn axes là nơi các bản vẽ thực tế được thực hiện.
Axis: Chúng là các đối tượng giống như trục số và chịu trách nhiệm tạo ra các giới hạn cho đồ thị.
Artist: Mọi thứ bạn có thể nhìn thấy trên figure đều là một artist (phần tử đồ họa), chẳng hạn như các đối tượng Văn bản (Text), đối tượng Đường 2D (Line2D), hay các đối tượng tập hợp (collection). Hầu hết các Artist đều được gắn vào các Axes.

2.2.3 Ưu điểm :

Matplotlib là một thư viện giống như GNUplot. Ưu điểm chính so với GNUplot là Matplotlib là một mô-đun của Python. Do sự phổ biến ngày càng tăng của Python, Matplotlib cũng nhận được sự quan tâm tương tự.

Một lý do khác tạo nên sức hấp dẫn của Matplotlib là nó được coi là giải pháp thay thế hoàn hảo cho MATLAB, nếu được sử dụng kết hợp với Numpy và Scipy. Trong khi MATLAB đắt đỏ và mã nguồn đóng, Matplotlib lại miễn phí và là mã nguồn mở. Nó cũng mang tính hướng đối tượng. Hơn nữa, nó có thể được sử dụng với các bộ công cụ giao diện đồ họa (GUI) đa năng như wxPython, Qt và GTK+. Ngoài ra còn có một thủ tục "pylab", được thiết kế để giống với MATLAB. Điều này có thể giúp những người đã quen thuộc với MATLAB chuyển sang Matplotlib dễ dàng hơn.

Matplotlib có thể được sử dụng để tạo ra các hình ảnh chất lượng cao cho nhiều định dạng bản cứng (in ấn) và các môi trường tương tác trên đa nền tảng.

Một đặc điểm khác của Matplotlib là đường cong học tập của nó, có nghĩa là người dùng thường tiến bộ nhanh chóng ngay sau khi bắt đầu. Trang web chính thức có nói như sau: "Matplotlib cố gắng làm cho những việc khó khăn, phức tạp trở nên dễ dàng nhất có thể. Bạn có thể tạo các hình vẽ, biểu đồ histogram, biểu đồ quang phổ, biểu đồ cột, biểu đồ sai số, biểu đồ phân tán, v.v., chỉ với một vài dòng mã."

Matplotlib có một số giao diện để tương tác với thư viện Matplotlib: API Hướng đối tượng, Giao diện Kịch bản (pyplot), Giao diện MATLAB (pylab). Pyplot và pylab đều là các giao diện nhẹ, nhưng Pyplot cung cấp một giao diện thủ tục cho các thư viện vẽ đồ thị hướng đối tượng trong Matplotlib. Các lệnh vẽ của nó được thiết kế tương tự như của Matlab về cả cách đặt tên và ý nghĩa tham số. Thiết kế này đã làm cho việc sử dụng pyplot trở nên dễ dàng và dễ hiểu hơn, vì vậy trong các bài viết về Matplotlib, tôi sẽ sử dụng giao diện pyplot thay vì hai giao diện kia. Nếu chúng ta muốn can thiệp sâu hơn, với nhiều tùy chỉnh hơn, thì API Hướng đối tượng sẽ là sự lựa chọn đúng đắn.

Để cài đặt Matplotlib, nếu bạn có Anaconda, chỉ cần gõ conda install matplotlib hoặc sử dụng công cụ pip (pip install matplotlib) .

2.3 Numpy

NumPy (Numeric Python) là một thư viện Python mã nguồn mở được sử dụng trong hầu hết các lĩnh vực khoa học và kỹ thuật. Nó là tiêu chuẩn mặc định để làm việc với dữ liệu số trong Python và là tiêu chuẩn cốt lõi của hệ sinh thái Python và PyData.

Numpy hỗ trợ mạnh mẽ cho việc tính toán với các ma trận, véc-tơ và các hàm đại số tuyến tính cơ bản. Điều này biến nó trở thành một công cụ quan trọng trong việc triển khai các thuật toán Học máy (Machine Learning). Nó cho phép làm việc hiệu quả với các ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn, với tốc độ xử lý nhanh hơn nhiều lần so với việc chỉ sử dụng "Python thuần" (core Python).

2.3.1 Cài đặt

- pip install numpy

2.3.2 Các phép toán với Numpy

- Khai báo thư viện: import numpy as np
- Khởi tạo mảng:
  + Khởi tạo một mảng một chiều với kiểu dữ liệu của các phần tử là Số nguyên (Integer):

```
arr = np.array([1,3,4,5,6], dtype = int)
```

  + Khởi tạo một mảng một chiều với kiểu dữ liệu của các phần tử là mặc định:

```
arr = np.array([1,3,4,5,6])
```

  + Khởi tạo một mảng hai chiều:

```
arr1 = np.array([(4,5,6), (1,2,3)])
```

  + Khởi tạo một mảng ba chiều:

```
arr2 = np.array(([(2,4,0,6), (4,7,5,6)],
                 [(0,3,2,1), (9,4,5,6)],
                 [(5,8,6,4), (1,4,6,8)]))
```

- Khởi tạo mảng với built-in function:

+ np.zeros
+ np.ones((2,3,4), dtype = int): Mảng 3 chiều với các phần tử là 1 và kích thước là 2x3x4
+ np.arange(1,7,2): Mảng có các phần tử từ 1 đến 6 và bước nhảy là 2.
+ np.full((2,3),5): Một mảng 2 chiều với các phần tử là 5 và kích thước là 2x3.
+ np.eye(4, dtype=int): Một ma trận đơn vị với kích thước 4x4.
+ np.random.random((2,3)): Một ma trận ngẫu nhiên với kích thước là 2x3.

2.3.3 Tính toán mảng

- dtype: Kiểu dữ liệu của phần tử trong mảng.
- shape: Kích thước (hình dạng) của mảng (ví dụ: số hàng, số cột).
- size: Tổng số lượng phần tử trong mảng.
- ndim: Số chiều của mảng.

- Truy cập các phần tử trong mảng: Các phần tử trong mảng được đánh chỉ mục bắt đầu từ 0
  + arr[i]: Truy cập tới phần tử thứ i của mảng 1 chiều.
  + arr1[i,j]: Truy cập tới hàng i và cột j của mảng 2 chiều.
  + arr2[n,i,j]: Truy cập tới chiều n, hàng i và cột j của mảng 3 chiều.
  + arr[a:b]: Truy cập các phần tử từ chỉ mục a đến b-1 trong mảng 1 chiều.
  + arr1[:,:i]: Truy cập các phần tử từ cột 0 đến cột i-1 (lấy tất cả các hàng).

- Các hàm thống kê:
  + arr.max() or np.max(arr): Lấy giá trị lớn nhất trong mảng.
  + arr.min() or np.min(arr): Lấy giá trị nhỏ nhất trong mảng.
  + arr.sum() or np.sum(arr): Tính tổng các phần tử trong mảng.
  + arr.mean() or np.mean(arr): Tính giá trị trung bình cộng của mảng.
  + np.median(arr): Lấy giá trị trung vị của mảng.

2.4 Scikit-learn

Scikit-learn là một thư viện học máy mã nguồn mở hỗ trợ cả học có giám sát và học không giám sát. Nó cũng cung cấp đa dạng các công cụ để khớp mô hình (model fitting), tiền xử lý dữ liệu, lựa chọn mô hình, đánh giá mô hình và nhiều tiện ích khác.

- Các công cụ đơn giản và hiệu quả để phân tích dữ liệu dự báo.

- Dễ tiếp cận với mọi người và có thể tái sử dụng trong nhiều ngữ cảnh khác nhau.
- Được xây dựng trên nền tảng NumPy, SciPy và matplotlib.
- Mã nguồn mở, có thể sử dụng cho mục đích thương mại - Giấy phép BSD.

2.4.1 Cài đặt

- Pip install -U scikit-learn

2.4.2 Làm việc với scikit-learn

- Bài toán :
  + Phân loại (Classification):
    - Định nghĩa: Xác định xem một đối tượng thuộc về nhóm/loại nào.
    - Phát hiện thư rác (spam detection), nhận dạng hình ảnh.
    - Thuật toán: Gradient boosting, láng giềng gần nhất (nearest neighbors), rừng ngẫu nhiên (random forest), hồi quy logistic, và nhiều hơn nữa...
  + Hồi quy (Regression):
    - Định nghĩa: Dự đoán một thuộc tính có giá trị liên tục gắn liền với một đối tượng.
    - Ứng dụng: Dự đoán phản ứng thuốc, dự đoán giá cổ phiếu.
    - Thuật toán: Gradient boosting, láng giềng gần nhất, rừng ngẫu nhiên, hồi quy Ridge, và nhiều hơn nữa....
  + Phân cụm (Clustering):
    - Định nghĩa: Tự động nhóm các đối tượng tương tự nhau vào các tập hợp.
    - Ứng dụng: Phân khúc khách hàng, nhóm các kết quả thí nghiệm.
    - Thuật toán: k-Means, HDBSCAN, phân cụm phân cấp (hierarchical clustering), và nhiều hơn nữa...

- Các hàm có sẵn hỗ trợ mô hình và tiền xử lý dữ liệu:

+ Giảm chiều dữ liệu (Dimensionality reduction):
    - Định nghĩa: Giảm số lượng các biến ngẫu nhiên cần xem xét.
    - Ứng dụng: Trực quan hóa dữ liệu, tăng hiệu quả xử lý.
    - Thuật toán: PCA (Phân tích thành phần chính), lựa chọn đặc trưng (feature selection), phân rã ma trận không âm, và nhiều hơn nữa…

+ Lựa chọn mô hình (Model selection):
    - Định nghĩa: So sánh, kiểm định và lựa chọn các tham số và mô hình.
    - Ứng dụng: Cải thiện độ chính xác thông qua việc tinh chỉnh tham số
    - Thuật toán: Tìm kiếm lưới (Grid search), kiểm chứng chéo (cross validation), các chỉ số đánh giá (metrics), và nhiều hơn nữa...

+ Tiền xử lý (Preprocessing):
    - Định nghĩa: Trích xuất đặc trưng và chuẩn hóa.
    - Ứng dụng: Biến đổi dữ liệu đầu vào (ví dụ như văn bản) để sử dụng với các thuật toán học máy.
    - Thuật toán: Tiền xử lý, trích xuất đặc trưng, và nhiều hơn nữa...

CHƯƠNG 3: MÔ HÌNH

3.1. Lý thuyết

3.1.1 Mô hình Random Forest

- Rừng ngẫu nhiên (Random Forest) là một thuật toán học máy có giám sát linh hoạt, được sử dụng rộng rãi cho cả bài toán phân loại và hồi quy. Nó là một phương pháp "tổ hợp" (ensemble method), hoạt động bằng cách xây dựng nhiều "cây quyết định" (decision trees) trong quá trình huấn luyện và gộp kết quả của chúng lại để đưa ra dự đoán chính xác và ổn định hơn.
- Các loại bài toán:
  + Phân loại (Classification): Đối với biến phụ thuộc là nhãn rời rạc (ví dụ: Có bệnh/Không bệnh, Chó/Mèo/Gà). Kết quả đầu ra là lớp được chọn bởi phiếu bầu đa số (majority vote) từ các cây thành phần.
  + Hồi quy (Regression): Đối với biến phụ thuộc là giá trị liên tục (ví dụ: Giá nhà, Nhiệt độ). Kết quả đầu ra là giá trị trung bình (mean prediction) của các cây thành phần.
- Khai báo mô hình:
  + Bước 1: khai báo từ thư viện

    ```
    from sklearn.ensemble import RandomForestClassifier
    ```

  + Step 2: gán vào biến

    ```
    rf = RandomForestClassifier()
    ```

3.1.2 Mô hình SVM

- (Support Vector Machine - SVM) là một thuật toán học máy mạnh mẽ được sử dụng rộng rãi cho cả phân loại tuyến tính và phi tuyến tính, cũng như các tác vụ hồi quy và phát hiện ngoại lai. SVM có khả năng thích ứng cao, làm cho chúng phù hợp với nhiều ứng dụng khác nhau như phân loại văn bản, phân loại hình ảnh, phát hiện thư rác, nhận dạng chữ viết tay, phân tích biểu hiện gen, phát hiện khuôn mặt và phát hiện bất thường.

- SVM đặc biệt hiệu quả vì chúng tập trung vào việc tìm ra siêu phẳng phân cách tối đa (maximum separating hyperplane) giữa các lớp khác nhau trong đặc trưng mục tiêu, giúp chúng trở nên mạnh mẽ đối với cả phân loại nhị phân và đa lớp. Trong phần tóm tắt này, chúng ta sẽ khám phá thuật toán SVM, các ứng dụng của nó và cách nó xử lý hiệu quả cả phân loại tuyến tính và phi tuyến tính, cũng như các tác vụ hồi quy và phát hiện ngoại lai.
- Khai báo mô hình:
  + Step 1: khai báo từ thư viện

```
from sklearn.svm import SVC
```

  + Step 2: gán biến

```
svm_model = SVC()
```

3.1.3 Metrics đánh giá

3.1.3.1 Điểm Accuracy

Hàm accuracy_score tính toán độ chính xác, dưới dạng tỷ lệ (mặc định) hoặc số lượng (khi normalize=False) các dự đoán đúng.

Trong phân loại đa nhãn (multilabel classification), hàm trả về độ chính xác tập con (subset accuracy). Nếu toàn bộ tập hợp các nhãn được dự đoán cho một mẫu khớp hoàn toàn với tập hợp nhãn thực tế, thì độ chính xác tập con là 1.0; ngược lại là 0.0.

Nếu y^i là giá trị dự đoán của mẫu thứ i và yi là giá trị thực tế tương ứng, thì tỷ lệ các dự đoán đúng trên tổng số mẫu nsamples được định nghĩa là

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Hình 3.1: Công thức Accuracy

3.1.3.2 Điểm Precision

Precision (Độ chính xác của dự báo dương tính) là tỷ lệ tp / (tp + fp), trong đó tp là số lượng các trường hợp dương tính thật (true positives) và fp là số lượng các trường hợp dương tính giả (false positives).

Về mặt trực quan, precision thể hiện khả năng của bộ phân loại trong việc không gán nhãn dương tính cho một mẫu thực chất là âm tính.

Giá trị tốt nhất là 1 và giá trị tệ nhất là 0.

3.1.3.3 Điểm Recall

Recall (Độ thu hồi) là tỷ lệ tp / (tp + fn), trong đó tp là số lượng các trường hợp dương tính thật (true positives) và fn là số lượng các trường hợp âm tính giả (false negatives).

Về mặt trực quan, recall thể hiện khả năng của bộ phân loại trong việc tìm ra tất cả các mẫu dương tính.

Giá trị tốt nhất là 1 và giá trị tệ nhất là 0

3.2. Code

- Dữ liệu cho mô hình:

```
X = df[['Physical_Activity',
        'Nutrition_Score',
        'Stress_Level',
        'Mindfulness',
        'Sleep_Hours',
        'Hydration',
        'BMI',
        'Alcohol','Smoking']]
```

Hình 3.2: Thuộc tính cho mô hình

+ Chia dữ liệu cho 2 tập (Train 80%, Test 20%):

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Hình 3.3: Chia dữ liệu

- Training model:
  + RandomForest (chưa áp dụng Smote):

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)
y_pred_class = rf.predict(X_test_scaled)
```

Hình 3.4: Huấn luyện RandomForest (chưa Smote)

```
print("Classification Report:\n", classification_report(y_test, y_pred_class))
```

```
Classification Report:
               precision    recall  f1-score   support

     Average       0.84      0.84      0.84       551
        Good       0.94      0.97      0.95      1373
        Poor       0.94      0.43      0.59        76

    accuracy                           0.91      2000
   macro avg       0.91      0.75      0.80      2000
weighted avg       0.91      0.91      0.91      2000
```

Hình 3.5: Điểm RandomForest (chưa Smote)

  + RandomForest (áp dụng Smote):

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_res_scaled, y_res)
y_pred_class = rf.predict(X_test_scaled)
```

Hình 3.6: Huấn luyện RandomForest (áp dụng Smote)

```
print("Classification Report:\n", classification_report(y_test, y_pred_class))
```

```
Classification Report:
               precision    recall  f1-score   support

     Average       0.83      0.88      0.86       551
        Good       0.96      0.95      0.96      1373
        Poor       0.75      0.71      0.73        76

    accuracy                           0.92      2000
   macro avg       0.85      0.85      0.85      2000
weighted avg       0.92      0.92      0.92      2000
```

Hình 3.7: Điểm RandomForest (áp dụng Smote)

+ SVM (áp dụng Smote)

```
from sklearn.model_selection import GridSearchCV
model = SVC()
param_grid = {
    'kernel': ['linear', 'rbf']
}
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=5,
    scoring='recall',
    verbose=3, # để in ra tiến trình
    n_jobs=-1 # sử dụng tất cả các nhân CPU
)
```

Hình 3.8: Khai báo và gán tham số cho GridSearchCV

```
grid_search.fit(X_res_scaled, y_res)
print(grid_search.best_params_)
best_model = grid_search.best_estimator_
```

Hình 3.9: Huấn luyện SVM (áp dụng Smote)

```
y_pred_class = best_model.predict(X_test_scaled)
print("Classification Report:\n", classification_report(y_test, y_pred_class))
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
/usr/local/lib/python3.12/dist-packages/sklearn/model_selection/_search.py:1108:
  warnings.warn(
{'kernel': 'linear'}
Classification Report:
              precision    recall  f1-score   support

     Average       0.97      0.99      0.98       551
        Good       1.00      0.99      0.99      1373
        Poor       0.95      1.00      0.97        76

    accuracy                           0.99      2000
   macro avg       0.97      0.99      0.98      2000
weighted avg       0.99      0.99      0.99      2000
```

Hình 3.10: Điểm SVM (áp dụng Smote)

+ SVM (chưa áp dụng Smote)

```
from sklearn.model_selection import GridSearchCV
model = SVC()
param_grid = {
    'kernel': ['linear', 'rbf']
}
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=5,
    scoring='recall',
    verbose=3,
    n_jobs=-1
)
grid_search.fit(X_train_scaled, y_train)
print(grid_search.best_params_)
best_model = grid_search.best_estimator_
```

Hình 3.11: Huấn luyện SVM (chưa áp dụng Smote)

```
y_pred_class = best_model.predict(X_test_scaled)
print("Classification Report:\n", classification_report(y_test, y_pred_class))
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
/usr/local/lib/python3.12/dist-packages/sklearn/model_selection/_search.py:1108
  warnings.warn(
{'kernel': 'linear'}
Classification Report:
              precision    recall  f1-score   support

     Average       0.99      1.00      0.99       551
        Good       1.00      1.00      1.00      1373
        Poor       1.00      0.97      0.99        76

    accuracy                           1.00      2000
   macro avg       1.00      0.99      0.99      2000
weighted avg       1.00      1.00      1.00      2000
```

Hình 3.12: Điểm SVM (chưa áp dụng Smote)

+ Lưu lại mô hình

```
model_demo = SVC(kernel='linear')
model_demo.fit(X_res_scaled, y_res)
joblib.dump(model_demo, 'svm.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

Hình 3.13: Lưu mô hình

TÀI LIỆU THAM KHẢO

1. Dữ liệu

https://www.kaggle.com/datasets/miadul/holistic-health-and-lifestyle-score-dataset

2. Tài liệu
    - Slides and documents of Mr Hoang Quoc Viet.
    - Machine learning basic of Mr Nguyen Van Hau, Mr Pham Minh Chuan, Mr Nguyen Van Quyet.
    - Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython 2nd Edition.
    - Machine learning of Vũ Hữu Tiệp.