

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH (UEH)
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ



ĐỒ ÁN MÔN HỌC
LẬP TRÌNH PHÂN TÍCH DỮ LIỆU

**Đề tài: Ứng dụng lập trình phân tích để phân tích dữ liệu
về giấc ngủ từ bộ Sleep Health and Lifestyle Dataset**

Mã lớp HP: 23C1INF50907002
Giảng viên: TS. Nguyễn An Tế
Khóa: K47

Nhóm sinh viên:

Huỳnh Nguyễn Anh Cường	-	31211024275
Mã Thành An	-	31211026893
Trịnh Thị Hải Âu	-	31211027629
Trịnh Uyên Chi	-	31211020738
Võ Tuấn Cường	-	31211027631

Tp. Hồ Chí Minh, Ngày 3 tháng 12 năm 2023

LỜI MỞ ĐẦU

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn An Tế đã tận tình giảng dạy và truyền đạt những kiến thức bổ ích cho chúng em trong suốt thời gian học tập môn Lập trình phân tích dữ liệu để chúng em có thể hoàn thành bài báo cáo một cách tốt nhất. Vì chúng em vẫn chỉ là sinh viên, trong quá trình làm bài cũng không thể tránh khỏi những thiếu sót nên chúng em rất mong nhận được sự góp ý, nhận xét của thầy để có thể rút được kinh nghiệm cho những bài báo cáo sau này.

Cuối cùng, kính chúc thầy luôn tràn đầy sức khỏe, nhiệt huyết, hạnh phúc và thành công trong công việc và cuộc sống để tiếp tục truyền đạt kiến thức cho những thế hệ mai sau.

Chúng em xin chân thành cảm ơn thầy.

MỤC LỤC

LỜI MỞ ĐẦU	2
MỤC LỤC	3
Chương 1: Tổng quan đề tài.....	1
I. Giới thiệu đề tài	1
II. Mục tiêu	1
III. Phương pháp nghiên cứu	1
IV. Dữ liệu đồ án	2
Chương 2: Tiền xử lý và khám phá dữ liệu (EDA).....	3
I. Tổng quát dữ liệu.....	3
II. Xử lý dữ liệu bị khuyết.....	4
1. Kiểm tra dữ liệu bị thiếu	4
2. Biến numerical	5
3. Biến categorical	5
III. Xử lý dữ liệu nhiễu.....	5
1. Kiểm tra dữ liệu nhiễu	5
2. Hiển thị dữ liệu nhiễu.....	7
3. Xử lý Outliers.....	8
IV. Xử lý chi tiết	9
1. Gender.....	9
2. Age.....	10
3. Occupation	12
4. Sleep Duration	13
5. Quality of Sleep	14
6. Physical Activity Level.....	15
7. Stress Level.....	17
8. BMI Category	18
9. Blood Pressure	19
10. Heart Rate	21
11. Daily Steps	22
12. Sleep Disorder.....	23
V. Phân phối dữ liệu	25

VI. Phân tích tính tương quan giữa các biến	27
1. Quan sát sự tương quan giữa các biến số.....	27
2. Quan sát sự ảnh hưởng giữa các biến phân loại.....	29
3. Quan sát sự ảnh hưởng giữa các biến phân loại và biến số.....	30
VII. Tổng quan dữ liệu sau xử lý	32
Chương 3: Phân tích và khai thác dữ liệu	34
I. Phân cụm	34
1. Định nghĩa phân cụm.....	34
2. Mục đích phân cụm.....	34
3. Tiền xử lý trước phân cụm.....	34
4. Phân cụm phân hoạch	42
5. Phân cụm phân cấp	44
6. Biểu diễn kết quả phân cụm.....	47
7. Đánh giá mô hình.....	49
8. Quan sát các biến mục tiêu	50
II. Phân lớp	53
1. Định nghĩa phân lớp.....	53
2. Mục đích phân lớp	53
3. Tiền xử lý trước phân lớp.....	53
4. Xây dựng hàm đánh giá mô hình	56
5. Xây dựng mô hình phân lớp.....	57
6. Đánh giá mô hình.....	61
7. Kết luận.....	63
Chương 4: Kết luận và đề xuất	64
I. Kết luận.....	64
II. Đề xuất.....	64
III. Ưu nhược điểm	65
Mã nguồn.....	66
Tài liệu tham khảo.....	66
Bảng phân công	67

Chương 1: Tổng quan đề tài

I. Giới thiệu đề tài

Trong thế giới hiện đại, chất lượng giấc ngủ đã nổi lên như một vấn đề ngày càng quan trọng và tập trung trong tâm điểm của quan tâm đối với sức khỏe và chất lượng cuộc sống. Khả năng có giấc ngủ đủ và chất lượng là một khía cạnh cực kỳ quan trọng, không chỉ ảnh hưởng đến sức khỏe và khả năng phục hồi của cơ thể mà còn đối với hiệu suất làm việc, trí tuệ và sự cân bằng tâm lý của con người.

Tuy nhiên, một số lớn người gặp khó khăn trong việc duy trì một giấc ngủ đủ để và sâu. Có nhiều yếu tố có thể góp phần ảnh hưởng đến chất lượng giấc ngủ, bao gồm tuổi tác, lối sống hàng ngày, tình trạng tâm lý và môi trường xung quanh.

Nhóm đã quyết định khám phá và làm rõ những yếu tố ảnh hưởng đến chất lượng giấc ngủ bằng cách áp dụng các phương pháp và kỹ thuật phân tích dữ liệu từ bộ dữ liệu "Sleep Health and Lifestyle Dataset" mà nhóm thu thập từ Kaggle.

II. Mục tiêu

Mục tiêu của nghiên cứu là tìm hiểu các yếu tố có tác động đến thời gian ngủ và các chứng rối loạn giấc ngủ, đồng thời xác định rõ những yếu tố nào có ảnh hưởng mạnh mẽ và quan trọng đối với sự cải thiện hoặc giảm thiểu chất lượng giấc ngủ. Cuối cùng, dựa trên kết quả nghiên cứu, nhóm sẽ đề xuất các khuyến nghị và biện pháp cải thiện để hỗ trợ người dùng trong việc tối ưu hóa chất lượng giấc ngủ của họ và giải quyết các vấn đề về giấc ngủ một cách hiệu quả.

III. Phương pháp nghiên cứu

Nghiên cứu này tập trung vào việc áp dụng các phương pháp nghiên cứu để khám phá và hiểu sâu hơn về bộ dữ liệu "Sleep Health and Lifestyle Dataset". Nhóm đặt ra một loạt mục tiêu nghiên cứu quan trọng trong việc tiếp cận và phân tích dữ liệu này.

Trước hết, nhóm sẽ sử dụng phương pháp thống kê để mô tả các biến và tính các thước đo cơ bản liên quan đến giấc ngủ và lối sống. Nhóm cũng tiến hành chuẩn hóa và mã hóa dữ liệu để đảm bảo tính nhất quán và phù hợp cho các phân tích tiếp theo.

Tiếp theo, nhóm sẽ tiến hành kiểm định mối quan hệ tương quan giữa các biến trong bộ dữ liệu với các biến mục tiêu để xác định các mối quan hệ quan trọng. Điều này sẽ giúp nhóm xác định các biến ảnh hưởng và có thể là yếu tố quan trọng trong sức khỏe và giấc ngủ.

Sử dụng các thuật toán máy học Machine Learning, phương pháp phân cụm - phân lớp cho các giá trị thu được cho bộ dữ liệu Sleep Health and Lifestyle Dataset và biểu diễn trực quan kết quả theo các biến mục tiêu đặt ra.

Cuối cùng, nhóm sẽ đánh giá và dựa trên các kết quả thu được để đưa ra các biện pháp, khuyến nghị để cải thiện chất lượng giấc ngủ. Mục tiêu cuối cùng là mang lại cái nhìn sâu hơn và chi tiết hơn về mối quan hệ giữa giấc ngủ và lối sống, từ đó cung cấp thông tin quan trọng cho lĩnh vực sức khỏe và nghiên cứu sức khỏe.

IV. Dữ liệu đồ án

Bộ dữ liệu về sức khỏe giấc ngủ và lối sống bao gồm 374 hàng và 13 cột, bao gồm nhiều biến số liên quan đến giấc ngủ và thói quen hàng ngày. Nó bao gồm các chi tiết như giới tính, tuổi tác, nghề nghiệp, thời gian ngủ, chất lượng giấc ngủ, mức độ hoạt động thể chất, mức độ căng thẳng, chỉ số BMI, huyết áp, nhịp tim, số bước đi hàng ngày và sự hiện diện hay vắng mặt của rối loạn giấc ngủ.

Ý nghĩa của các thuộc tính:

STT	Tên thuộc tính	Mô tả
1	Person ID	Mã định danh cho mỗi cá nhân
2	Gender	Giới tính của người đó (Nam/Nữ)
3	Age	Tuổi của người tính bằng năm
4	Occupation	Nghề nghiệp của một người
5	Sleep Duration	Số giờ người đó ngủ mỗi ngày
6	Quality of Sleep	Đánh giá chủ quan về chất lượng giấc ngủ, từ 1 đến 10
7	Physical Activity Level	Số phút một người dành cho hoạt động thể chất hàng ngày
8	Stress Level	Đánh giá chủ quan về mức độ căng thẳng của một người, từ 1 đến 10
9	BMI Category	Danh mục BMI của một người
10	Blood Pressure (systolic/diastolic)	Số đo huyết áp của một người, được biểu thị bằng huyết áp tâm thu so với huyết áp tâm trương
11	Heart Rate (bpm)	Nhịp tim lúc nghỉ ngơi của một người tính bằng nhịp mỗi phút
12	Daily Steps	Số bước mà người đó thực hiện mỗi ngày
13	Sleep Disorder	Sự hiện diện hay vắng mặt của chứng rối loạn giấc ngủ ở người (Không có, Mất ngủ, Ngưng thở khi ngủ)

Chương 2: Tiền xử lý và khám phá dữ liệu (EDA)

I. Tổng quát dữ liệu

Bộ dữ liệu "Sleep Health and Lifestyle" đã trải qua một quy trình tổng hợp và xử lý cẩn thận để đảm bảo rằng dữ liệu đầu vào của chúng ta là đáng tin cậy và chính xác. Để làm điều này, nhóm sẽ thực hiện một loạt bước kiểm tra dữ liệu đầu vào. Trong quá trình này, nhóm kiểm tra số lượng quan sát, xử lý dữ liệu thiếu, và loại bỏ dữ liệu không cần thiết. Mục tiêu cuối cùng của quá trình này là tạo ra một tập dữ liệu sạch, thống nhất và tin cậy, sẵn sàng cho các phân tích tiếp theo.

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27.0	Software Engineer	6.1	6.0	42.0	6	Overweight	126/83	77.0	4200.0	None
1	2	Male	28.0	Doctor	6.2	6.0	60.0	8	Normal	125/80	75.0	10000.0	None
2	3	Male	28.0	Doctor	6.2	6.0	60.0	8	Normal	125/80	75.0	10000.0	None
3	4	Male	28.0	Sales Representative	5.9	4.0	30.0	8	Obese	140/90	85.0	3000.0	Sleep Apnea
4	5	Male	28.0	Sales Representative	5.9	4.0	30.0	8	Obese	140/90	85.0	3000.0	Sleep Apnea
...
369	370	Female	59.0	Nurse	8.1	9.0	75.0	3	Overweight	140/95	68.0	7000.0	Sleep Apnea
370	371	Female	59.0	Nurse	8.0	9.0	75.0	3	Overweight	140/95	68.0	7000.0	Sleep Apnea
371	372	Female	59.0	Nurse	8.1	9.0	75.0	3	Overweight	140/95	68.0	7000.0	Sleep Apnea
372	373	Female	59.0	Nurse	8.1	9.0	75.0	3	Overweight	140/95	68.0	7000.0	Sleep Apnea
373	374	Female	59.0	Nurse	8.1	9.0	75.0	3	Overweight	140/95	68.0	7000.0	Sleep Apnea

```
## Thông tin chi tiết bộ dữ liệu
print('=== THÔNG TIN BỘ DỮ LIỆU ===')
print('\n>> Số lượng phần tử trong bộ dữ liệu:', data.size)
print('>> Kích thước bộ dữ liệu (số dòng, số cột):', data.shape)
data.info()
```

Sau khi quan sát tổng quan bộ dữ liệu và metadata ta thấy được:

- Bộ dữ liệu chứa tổng cộng 374 hàng, 13 cột và 4862 phần tử. Các cột dữ liệu thuộc vào ba kiểu dữ liệu chính: Float (6 thuộc tính), Int (2 thuộc tính), và Object (5 thuộc tính).
- Biến Numerical:
 - + Continuous: Sleep Duration, Physical Activity Level
 - + Discrete: Age, Heart Rate, Daily Steps
- Biến Categorical:
 - + Categorical: Person ID, Gender, Occupation, BMI Category, Blood Pressure, Sleep Disorder
 - + Ordinal: Quality of Sleep, Stress Level

```

=== THÔNG TIN BỘ DỮ LIỆU ===

>> Số lượng phần tử trong bộ dữ liệu: 4862
>> Kích thước bộ dữ liệu (số dòng, số cột): (374, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Person ID                            374 non-null    int64
1   Gender                               374 non-null    object
2   Age                                   373 non-null    float64
3   Occupation                           373 non-null    object
4   Sleep Duration                       372 non-null    float64
5   Quality of Sleep                     373 non-null    float64
6   Physical Activity Level               371 non-null    float64
7   Stress Level                         374 non-null    int64
8   BMI Category                         373 non-null    object
9   Blood Pressure                       373 non-null    object
10  Heart Rate                           372 non-null    float64
11  Daily Steps                          373 non-null    float64
12  Sleep Disorder                       373 non-null    object
dtypes: float64(6), int64(2), object(5)
memory usage: 38.1+ KB

```

II. Xử lý dữ liệu bị khuyết

1. Kiểm tra dữ liệu bị thiếu

Dùng cú pháp `isnull().sum()` để kiểm tra và đếm các giá trị thiếu (missing data) trong cột, sau đó trả về tổng số giá trị thiếu của cột tương ứng.

```

## Phân tích dữ liệu thiếu
print('Kiểm tra giá trị thiếu trong bộ dữ liệu:')
missing_values = data.isnull().sum()
print(missing_values)

```

```

Kiểm tra giá trị thiếu trong bộ dữ liệu:
Person ID                0
Gender                   0
Age                      1
Occupation               1
Sleep Duration           2
Quality of Sleep         1
Physical Activity Level   3
Stress Level             0
BMI Category             1
Blood Pressure           1
Heart Rate               2
Daily Steps              1
Sleep Disorder           1
dtype: int64

```

Kết quả kiểm tra giá trị thiếu trong bộ dữ liệu cho thấy rằng một số cột có sự thiếu sót dữ liệu. Cụ thể, các cột 'Age', 'Occupation', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps', và 'Sleep Disorder' có ít nhất một giá trị bị thiếu. Điều này đồng nghĩa rằng các cột này không hoàn toàn hoàn chỉnh về thông tin. Tuy nhiên, các cột còn lại trong bộ dữ liệu không có sự thiếu sót dữ liệu nào và đều chứa đầy đủ thông tin.

Việc thiếu sót dữ liệu có thể do quá trình thu nhập dữ liệu không hoàn chỉnh, có thể do các lỗi kỹ thuật, thiết bị hoặc đơn giản là do không thu nhập đủ thông tin từ nguồn

cung cấp. Cũng có thể do người thực hiện thu nhập, xử lý dữ liệu nhập sai, hiểu lầm hoặc bỏ sót.

Do kích thước của tập dữ liệu khá nhỏ và số lượng dữ liệu thiếu không nhiều nên ta sẽ tiến hành xử lý bằng phương pháp thay thế các dữ liệu khuyết bằng giá trị phù hợp với tập dữ liệu.

2. Biến numerical

Trong trường hợp dữ liệu định lượng bị thiếu, việc sử dụng giá trị trung vị có thể giúp duy trì tính chính xác của dữ liệu. Trung vị là giá trị ở giữa khi dữ liệu được sắp xếp theo thứ tự. Bằng cách gán trung vị cho các giá trị thiếu, ta có thể giữ được phân phối ban đầu của dữ liệu mà không bị ảnh hưởng bởi giá trị ngoại lệ

Tiến hành gán trung vị cho những cột biến numerical.

```
data.fillna(data.median(numeric_only=True).round(1), inplace=True)
```

3. Biến categorical

Đối với dữ liệu định danh, việc sử dụng mode là một cách phổ biến để điền giá trị cho dữ liệu thiếu. Điều này sẽ giữ nguyên tính chất phân loại của dữ liệu mà không ảnh hưởng quá nhiều đến cấu trúc của tập dữ liệu

Tiến hành gán mode cho những cột biến categorical.

```
string_columns = data.select_dtypes(include=['object']).columns
data[string_columns] = data[string_columns].fillna(data[string_columns].mode().iloc[0])
```

Kiểm tra lại số giá trị còn thiếu ở các cột trong tập dữ liệu.

```
print('Kiểm tra giá trị thiếu trong bộ dữ liệu:')
missing_values = data.isnull().sum()
print(missing_values)
```

```
Kiểm tra giá trị thiếu trong bộ dữ liệu:
Person ID      0
Gender          0
Age            0
Occupation     0
Sleep Duration 0
Quality of Sleep 0
Physical Activity Level 0
Stress Level   0
BMI Category   0
Blood Pressure 0
Heart Rate     0
Daily Steps    0
Sleep Disorder 0
dtype: int64
```

III. Xử lý dữ liệu nhiễu

1. Kiểm tra dữ liệu nhiễu

Chúng ta sử dụng biểu đồ hộp (boxplot) để kiểm tra dữ liệu nhiễu. Biểu đồ này là một công cụ phân tích dữ liệu thường được sử dụng để hiển thị trực quan sự phân phối của dữ liệu số và độ lệch thông qua việc hiển thị các giá trị quan trọng như giá trị

tối thiểu, phần tư thứ nhất (Q1), trung vị, phần tư thứ ba (Q3), và giá trị tối đa. Boxplot giúp nhanh chóng xác định giá trị trung bình, độ phân tán của dữ liệu, và các ngoại lệ.

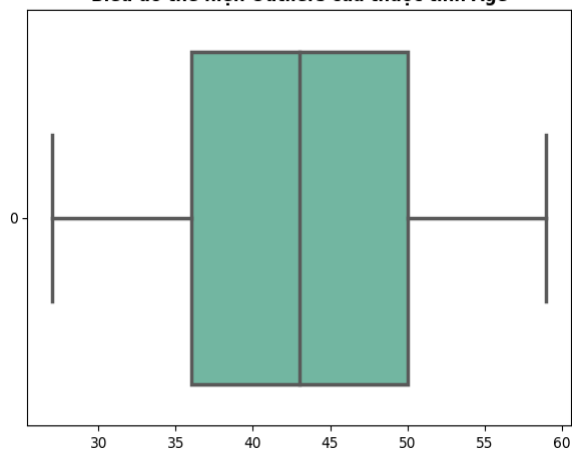
Chúng ta sẽ áp dụng boxplot cho các thuộc tính dữ liệu numerical bao gồm: 'Age', 'Sleep Duration', 'Physical Activity Level', 'Heart Rate' và 'Daily Steps'

```
# Chọn các thuộc tính để phát hiện Outliers
selected_columns = ['Age', 'Sleep Duration', 'Physical Activity Level', 'Heart Rate', 'Daily Steps']

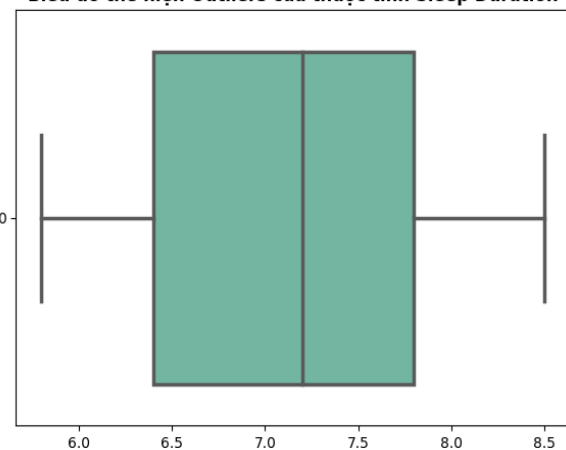
for column in selected_columns:
    # Vẽ biểu đồ boxplot cho các thuộc tính được chọn
    plt.figure(figsize=(7, 5))
    sns.boxplot(data=data[column], orient="h", palette="Set2", linewidth=2.5)
    plt.title(f"Biểu đồ thể hiện Outliers của thuộc tính {column}", fontweight='bold')
    # Hiển thị biểu đồ
    plt.show()
```

Kết quả thu được:

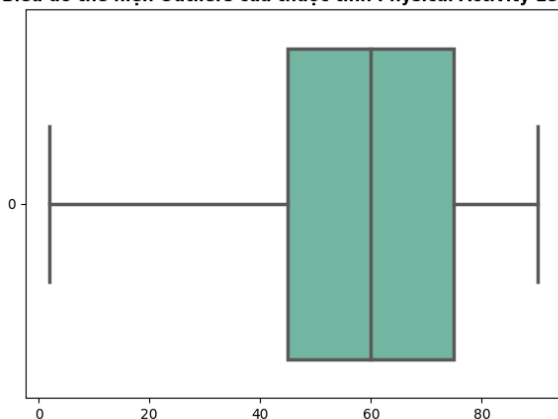
Biểu đồ thể hiện Outliers của thuộc tính Age



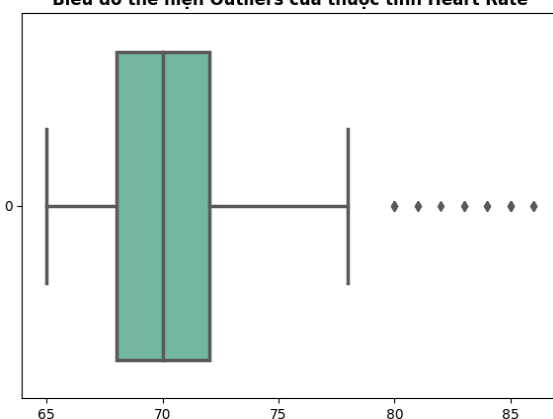
Biểu đồ thể hiện Outliers của thuộc tính Sleep Duration

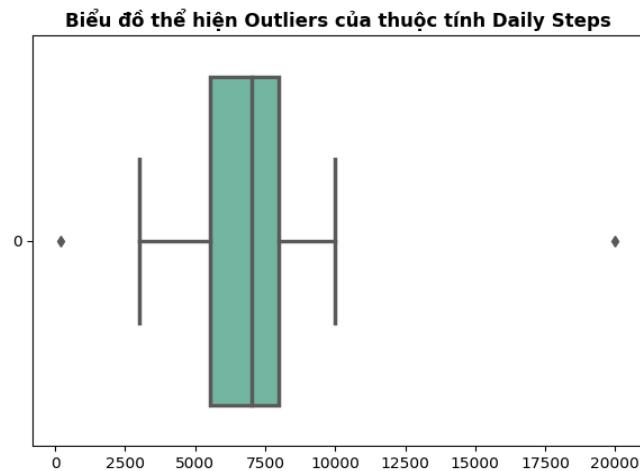


Biểu đồ thể hiện Outliers của thuộc tính Physical Activity Level



Biểu đồ thể hiện Outliers của thuộc tính Heart Rate





Kết quả quan sát từ boxplot cho ta thấy bộ dữ liệu xuất hiện các Outliers ở hai thuộc tính Heart Rate và Daily Steps.

2. Hiển thị dữ liệu nhiễu

Sau khi nhóm xem xét dữ liệu bị nhiễu, trước khi bắt đầu quá trình xử lý dữ liệu, nhóm đã thực hiện một kiểm tra để xác định giá trị ngoại lệ (Outliers) trong hai biến Heart Rate và Daily Steps. Mục đích của việc này là để hiểu rõ tại sao những giá trị ngoại lệ xuất hiện trong dữ liệu.

```
# Hàm để in ra các giá trị outliers của một thuộc tính cụ thể
def print_outliers(column_name, data1):
    column = data1[column_name]
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = column[(column < lower_bound) | (column > upper_bound)]

    if not outliers.empty:
        print(f"Outliers for '{column_name}':")
        print(outliers)

print_outliers('Heart Rate', data)
print_outliers('Daily Steps', data)
```

```

Outliers for 'Heart Rate':
3      85
4      85
5      85
6      82
16     80
18     80
80     81
81     81
93     84
145    84
147    80
264    83
266    83
276    86
277    86
Name: Heart Rate, dtype: int64
Outliers for 'Daily Steps':
10      200
265    20000
Name: Daily Steps, dtype: int64

```

Dựa trên kết quả của kiểm tra này, nhóm đã đưa ra quyết định về việc xử lý các giá trị ngoại lệ. Cụ thể, khi xem xét thuộc tính "Heart Rate," nhóm đã phát hiện rằng các giá trị ngoại lệ nằm trong khoảng từ 80 đến 86 nhịp/phút. Đây là các giá trị tương đối bình thường cho một người trưởng thành (thường nằm trong khoảng từ 60 đến 100 nhịp/phút). Vì vậy, nhóm đã quyết định giữ lại các giá trị ngoại lệ này cho việc phân tích.

Tuy nhiên, đối với thuộc tính "Daily Steps," nhóm đã xác định hai giá trị ngoại lệ với các giá trị là 200 và 20,000 bước/ngày. Các giá trị này không hợp lý so với thực tế (thường nằm trong khoảng từ 4,000 đến 10,000 bước/ngày). Vì vậy, nhóm đã quyết định thay thế các giá trị ngoại lệ này bằng giá trị trung vị của thuộc tính "Daily Steps."

3. Xử lý Outliers

```

# IQR
# Calculate the upper and lower limits
Q1 = data["Daily Steps"].quantile(0.25)
Q3 = data["Daily Steps"].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

# Create arrays of Boolean values indicating the outlier rows
upper_array = np.where(data["Daily Steps"]>=upper)[0]
lower_array = np.where(data["Daily Steps"]<=lower)[0]

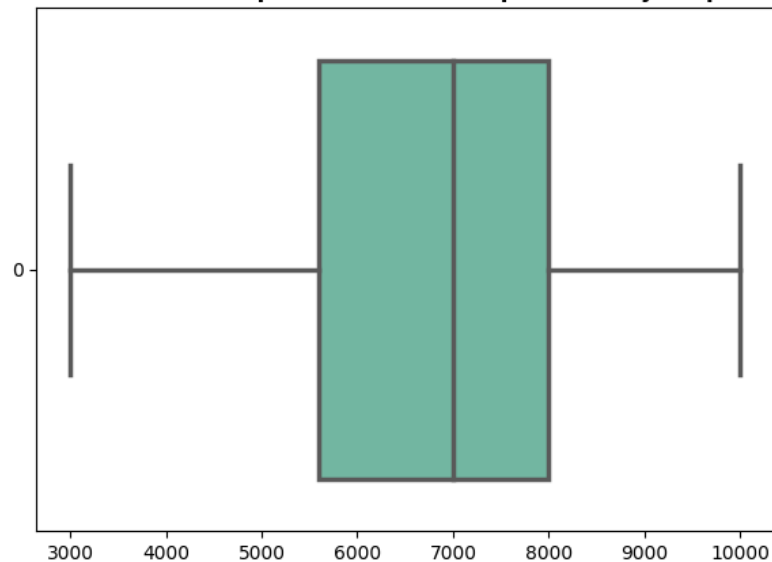
data["Daily Steps"][lower_array] = data["Daily Steps"].median()
data["Daily Steps"][upper_array] = data["Daily Steps"].median()

```

```
# Chọn các thuộc tính để phát hiện Outliers
selected_columns = ['Daily Steps']

for column in selected_columns:
    # Vẽ biểu đồ boxplot cho các thuộc tính được chọn
    plt.figure(figsize=(7, 5))
    sns.boxplot(data=data[column], orient="h", palette="Set2", linewidth=2.5)
    plt.title(f"Biểu đồ thể hiện Outliers của thuộc tính {column}", fontweight='bold')
    # Hiển thị biểu đồ
    plt.show()
```

Biểu đồ thể hiện Outliers của thuộc tính Daily Steps



IV. Xử lý chi tiết

1. Gender

Thực hiện các thống kê cơ bản với biến categorical - Gender (Giới tính):

```
gender_count=data['Gender'].value_counts().reset_index()
gender_count
```

index	Gender	
0	Male	189
1	Female	185

```
data['Gender'].describe()
```

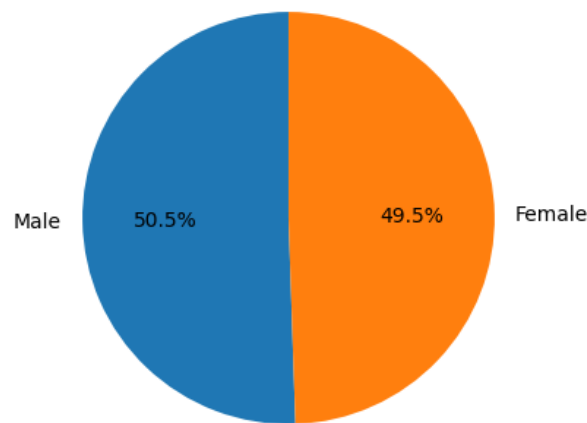
```
count      374
unique       2
top        Male
freq       189
Name: Gender, dtype: object
```

```
# Tính toán số lượng mẫu cho từng giá trị của thuộc tính "Gender"
gender_counts = data['Gender'].value_counts()

# Tạo biểu đồ Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Biểu đồ Pie Chart cho Gender')
plt.axis('equal') # Đảm bảo biểu đồ tròn

# Hiển thị biểu đồ
plt.show()
```

Biểu đồ Pie Chart cho Gender



Nhận xét: Tỷ lệ gần như cân bằng giữa các giới tính trong bộ dữ liệu mang theo một số ý nghĩa quan trọng:

- Sự cân bằng giữa nam và nữ trong dữ liệu giúp đảm bảo rằng không có sự thiên vị theo một giới tính cụ thể. Điều này tạo điều kiện cho một cái nhìn tổng quan và đa dạng hơn về các yếu tố liên quan đến cả nam và nữ.
- Tỷ lệ gần bằng nhau giữa nam và nữ có thể hỗ trợ trong quá trình phân tích dữ liệu, đặc biệt trong các trường hợp mà giới tính đóng vai trò quan trọng trong kết quả. Điều này cung cấp cơ hội cho việc nghiên cứu và hiểu sâu hơn về ảnh hưởng của giới tính đối với các biến liên quan.
- Một tỷ lệ gần bằng nhau giữa nam và nữ cũng tạo nền tảng vững chắc cho các phân tích so sánh chi tiết giữa các nhóm giới tính. Điều này tạo điều kiện cho sự phân tích rõ ràng và đáng tin cậy về sự khác biệt và tương đồng giữa nam và nữ trong bộ dữ liệu.

2. Age

Thực hiện các thống kê cơ bản với biến numerical - Age (Tuổi):

```
data['Age'].describe()

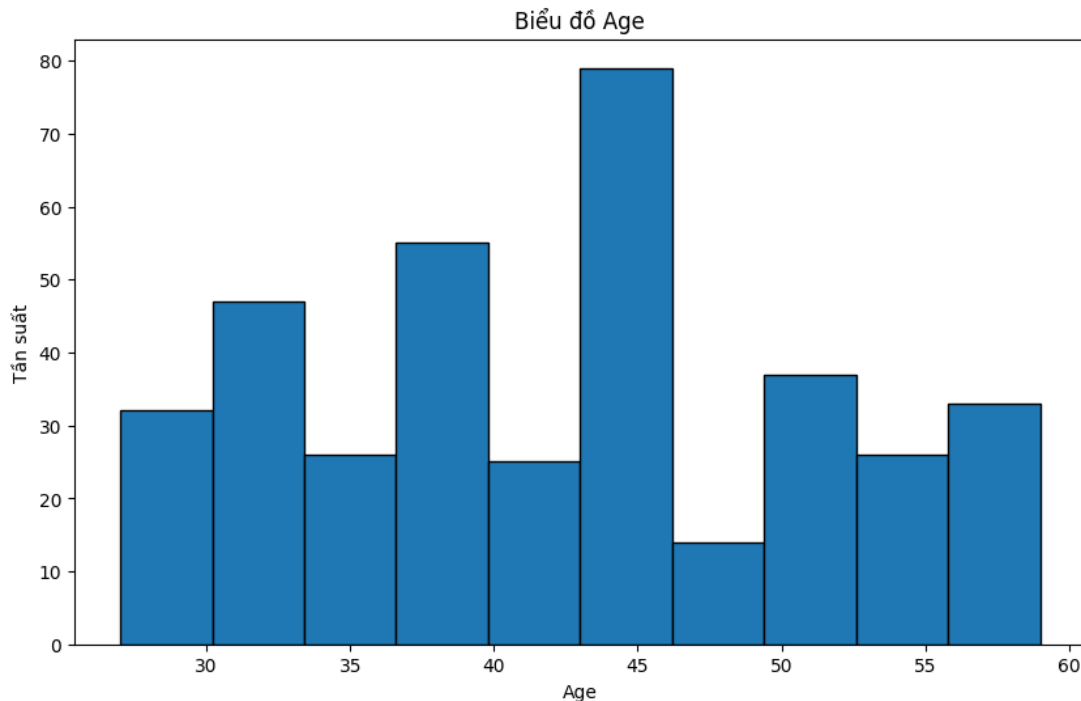
count    374.000000
mean     42.213904
std       8.657140
min       27.000000
25%      36.000000
50%      43.000000
75%      50.000000
max       59.000000
Name: Age, dtype: float64
```

```
data['Age'].unique()

array([27., 28., 29., 30., 31., 32., 33., 34., 35., 36., 37., 38.,
       39., 40., 41., 42., 43., 44., 45., 46., 47., 48., 49., 50., 51., 52., 53., 54.,
       55., 56., 57., 58., 59.])
```

```
# Tạo biểu đồ Histogram cho thuộc tính "Age"
plt.figure(figsize=(10, 6))
plt.hist(data['Age'], bins=10, edgecolor='black')
plt.title('Biểu đồ Age')
plt.xlabel('Age')
plt.ylabel('Tần suất')

# Hiển thị biểu đồ
plt.show()
```



Ta thấy dữ liệu về độ tuổi trải đều từ 27 đến 59 tuổi, và số lượng nhiều nhất rơi vào khoảng độ tuổi 43, ngoài ra độ tuổi còn chia ra thành từng khoảng trên biểu đồ, để hiểu rõ hơn ta sẽ tiến hành sử dụng bách phân vị để phân khoảng cho độ tuổi:

```
# Giả sử 'ages' là danh sách hoặc mảng chứa dữ liệu về tuổi
ages = data['Age']
ages = ages.dropna()

# Xác định các phân vị (percentiles) bạn muốn sử dụng
percentiles = [25, 50, 75]

# Tính toán các phân vị của dữ liệu
age_percentiles = np.percentile(ages, percentiles)

# Hiển thị kết quả
for p, value in zip(percentiles, age_percentiles):
    print(f"{p}th Percentile: {value}")

25th Percentile: 36.0
50th Percentile: 43.0
75th Percentile: 50.0
```

Sau khi sử dụng bách phân vị với P(25), P(50), P(75): ta thu được sự phân bố của độ tuổi như sau:

- Dưới 35 tuổi
- 36 đến 42 tuổi
- 43 đến 49 tuổi
- Trên 50 tuổi

Việc sử dụng bách phân vị để phân chia độ tuổi thành 4 nhóm cụ thể cung cấp cái nhìn tổng quan và rõ ràng về sự phân bố của dữ liệu theo độ tuổi, mặc dù nó có thể không cung cấp đủ thông tin chi tiết cho mọi mục đích phân tích. Nên nhóm quyết định chỉ sử dụng các phân nhóm này trong việc biểu diễn trực quan dữ liệu chứ không đưa vào quá trình xử lý, phân tích.

Ngoài ra kiểu dữ liệu hiện tại của biến là 'float64' dù dữ liệu là những giá trị số nguyên được tính theo năm, nhóm quyết định chuyển kiểu dữ liệu của biến Age từ 'float' thành 'int'

```
# Chuyển kiểu dữ liệu của cột "Age" từ float thành int
data['Age'] = data['Age'].astype('int')
```

3. Occupation

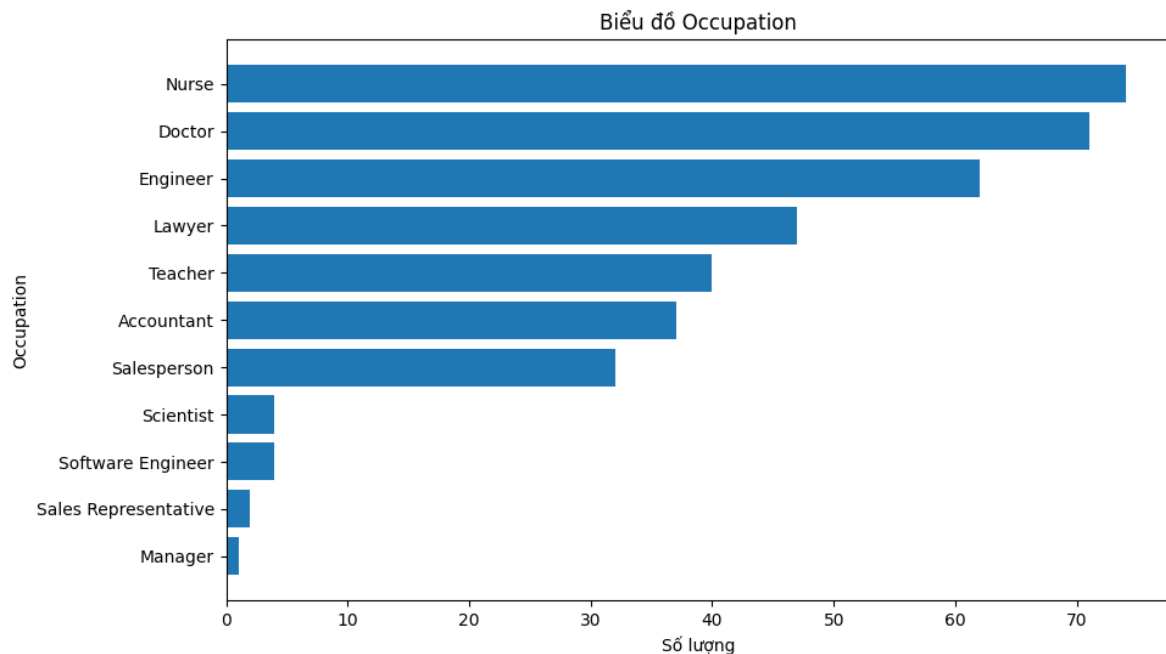
Thực hiện các thống kê cơ bản với biến categorical - Occupation (Nghề nghiệp):

```
data['Occupation'].describe()

count      374
unique       11
top        Nurse
freq         74
Name: Occupation, dtype: object
```

```
# Tạo biểu đồ Bar Chart cho thuộc tính "Occupation"
occupation_counts = data['Occupation'].value_counts().sort_values(ascending=True)
plt.figure(figsize=(10, 6))
plt.barh(occupation_counts.index, occupation_counts.values)
plt.xlabel('Số lượng')
plt.ylabel('Occupation')
plt.title('Biểu đồ Occupation')

# Hiển thị biểu đồ
plt.show()
```

Nhận xét: Biến categorical này bao gồm 11 giá trị độc lập, trong đó số lượng lớn nhất tập trung vào lĩnh vực y tế với các vai trò như Nurse và Doctor. Bên cạnh đó, dữ liệu còn thể hiện sự đa dạng về các ngành nghề và lĩnh vực khác như kế toán, giáo viên, kỹ thuật, luật sư, tạo nên sự phong phú, đa dạng trong bộ dữ liệu. Đáng chú ý, bên cạnh các ngành nghề phổ biến, còn xuất hiện các vai trò đặc biệt hiếm như Manager và Sales Representative, làm nổi bật sự đặc biệt và đa dạng hơn trong tập dữ liệu.

4. Sleep Duration

Thực hiện các thống kê cơ bản với biến numerical - Sleep Duration (Thời gian ngủ):

```
data['Sleep Duration'].describe()

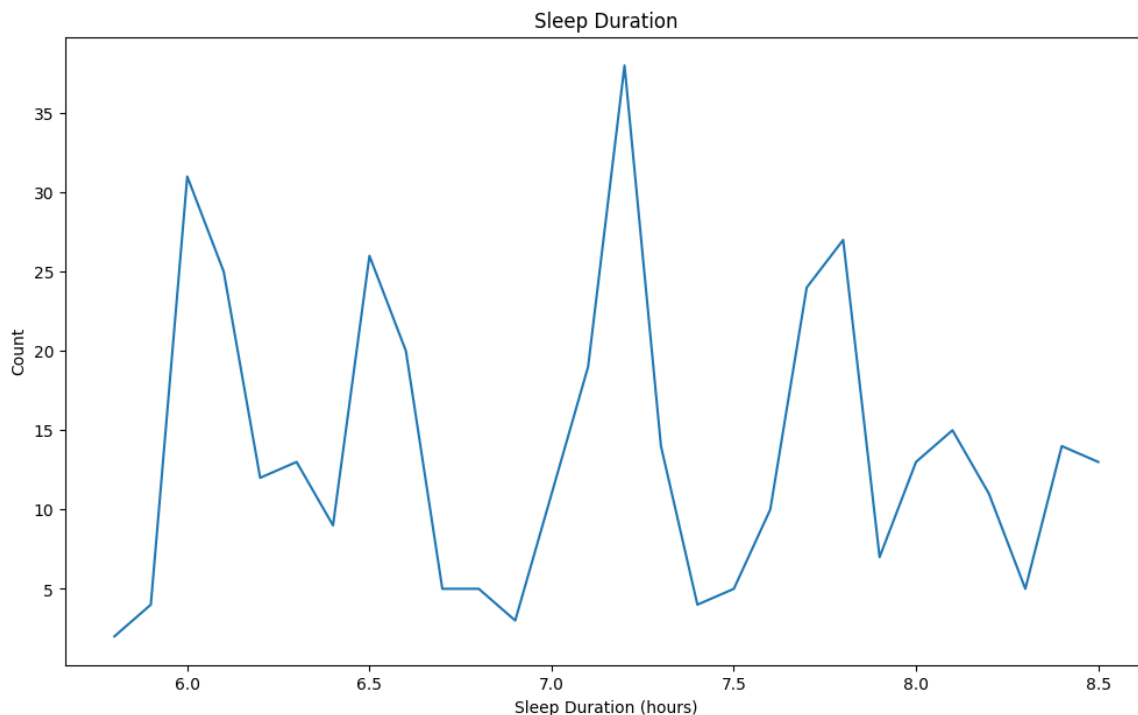
count    374.000000
mean      7.129947
std       0.794796
min       5.800000
25%       6.400000
50%       7.200000
75%       7.800000
max       8.500000
Name: Sleep Duration, dtype: float64
```

```
data['Sleep Duration'].unique()

array([6.1, 6.2, 5.9, 6.3, 7.8, 6. , 6.5, 7.6, 7.7, 7.9, 6.4, 7.2, 7.5,
       5.8, 6.7, 7.3, 7.4, 7.1, 6.6, 6.9, 8. , 6.8, 8.1, 8.3, 8.5, 8.4,
       8.2])
```

```
plt.figure(figsize=(12, 7))
sns.lineplot(x = 'index',
             y = 'Sleep Duration',
             data = Sleep_Duration_count)
plt.xlabel('Sleep Duration (hours)')
plt.ylabel('Count')
plt.title('Sleep Duration')

plt.show()
```



Nhận xét: Phân tích về thời gian ngủ trong dữ liệu cho thấy một phạm vi rộng từ 5.8 đến 8.5 giờ, với số lượng lớn nhất tập trung vào khoảng 7.2 giờ. Điều đáng chú ý, phân bố thời gian ngủ biểu hiện dưới dạng nhiều đỉnh trên biểu đồ, cho thấy sự đa dạng và sự chênh lệch về thời gian ngủ giữa các quan sát trong dữ liệu. Sự phân chia đa dạng này tạo ra cơ sở phong phú cho việc phân tích các yếu tố ảnh hưởng đến thời gian ngủ, từ mức độ áp lực đến thói quen sinh hoạt.

5. Quality of Sleep

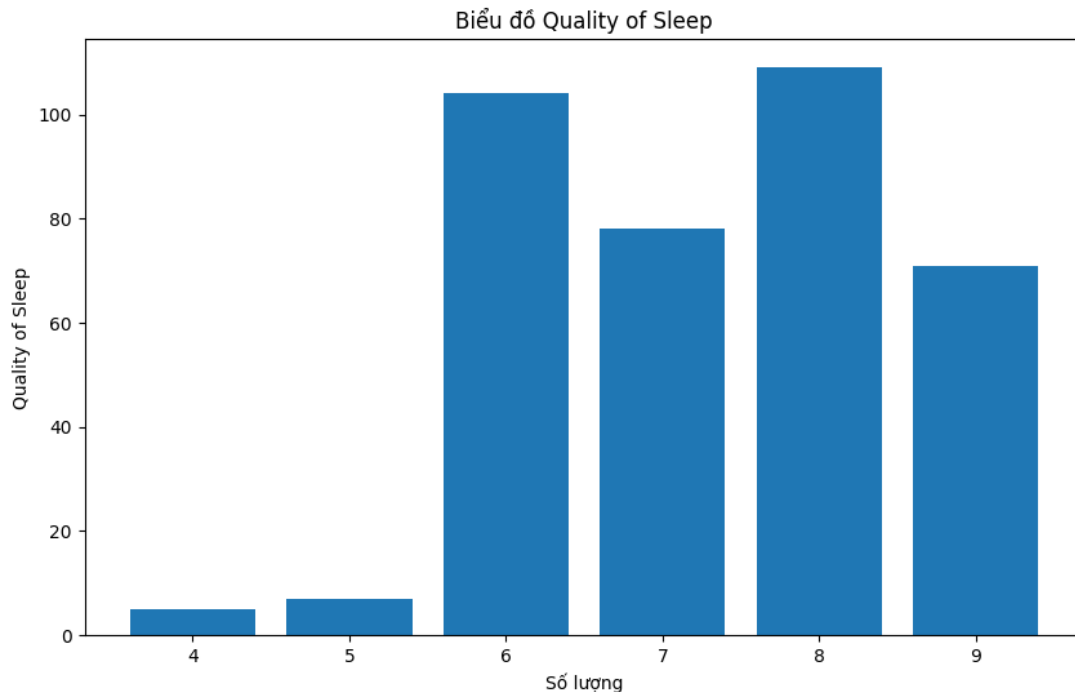
Thực hiện các thống kê cơ bản với biến numerical - Quality of Sleep (Chất lượng giấc ngủ):

```
data['Quality of Sleep'].describe()
```

count	374.000000
mean	7.315508
std	1.195131
min	4.000000
25%	6.000000
50%	7.000000
75%	8.000000
max	9.000000
Name: Quality of Sleep, dtype: float64	

```
# Tạo biểu đồ Bar Chart cho thuộc tính "Quality of Sleep"
quality_of_sleep_counts = data['Quality of Sleep'].value_counts()
plt.figure(figsize=(10, 6))
plt.bar(quality_of_sleep_counts.index, quality_of_sleep_counts.values)
plt.xlabel('Số lượng')
plt.ylabel('Quality of Sleep')
plt.title('Biểu đồ Quality of Sleep')

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Phân tích về chất lượng giấc ngủ trong dữ liệu cho thấy sự phân bố của các giá trị từ 4 đến 9, trong đó tập trung nhiều nhất ở mức 6 trở lên, ngụ ý rằng hầu hết các quan sát cho thấy chất lượng giấc ngủ ở mức trung bình trở lên. Tuy nhiên, cần lưu ý rằng dữ liệu này được thu thập dựa trên ý kiến chủ quan và tự đánh giá cá nhân, không dựa trên các phương pháp kiểm tra chính xác. Sự đa dạng trong quan niệm về chất lượng giấc ngủ có thể khiến phân tích trở nên không đáng tin cậy, do mỗi người có quan niệm khác nhau về mức độ tốt/xấu của giấc ngủ. Mặc dù đây là biến mang tính chủ quan, ta vẫn có thể phân tích các yếu tố ảnh hưởng đến quan niệm về chất lượng giấc ngủ. Tổng quát, nhóm cần xem xét cẩn thận và suy xét kỹ về các phân tích liên quan đến biến này để hiểu rõ hơn về ảnh hưởng của các yếu tố đến chất lượng giấc ngủ theo quan niệm cá nhân.

Thêm vào đó, mặc dù kiểu dữ liệu hiện tại của biến là 'float64', tuy nhiên, biến này thực tế thuộc dạng định tính thứ tự, được lấy dữ liệu dựa trên phương pháp tự đánh giá. Để phản ánh chính xác loại dữ liệu và sự chủ quan trong việc đánh giá, ta quyết định chuyển đổi kiểu dữ liệu của biến này thành 'object'.

```
# Chuyển dạng dữ liệu của 'Quality of Sleep' từ float64 thành object
data['Quality of Sleep'] = data['Quality of Sleep'].astype('object')
```

6. Physical Activity Level

Thực hiện các thống kê cơ bản với biến numerical - Physical Activity Level (Mức độ hoạt động thể chất):

```
data['Physical Activity Level'].describe()

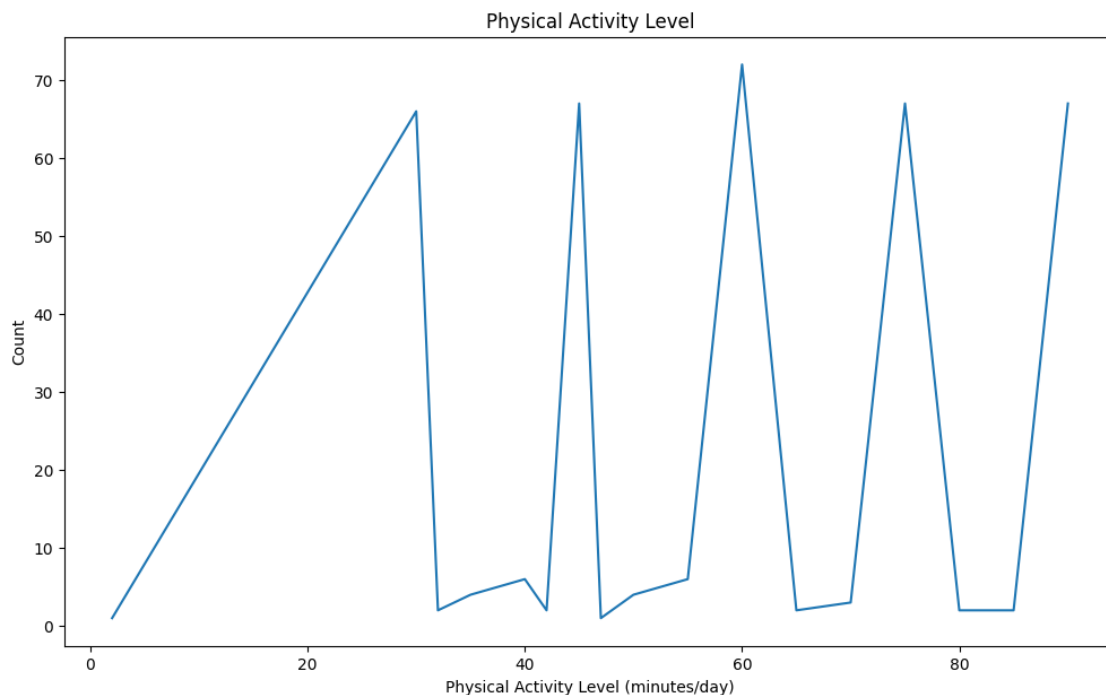
count    374.000000
mean      59.216578
std       20.918549
min        2.000000
25%       45.000000
50%       60.000000
75%       75.000000
max       90.000000
Name: Physical Activity Level, dtype: float64
```

```
plt.figure(figsize=(12, 7))
sns.lineplot(x = 'index',
              y = 'Physical Activity Level',
              data = Physical_Activity_Level)
plt.xlabel('Physical Activity Level (minutes/day)')
plt.ylabel('Count')
plt.title('Physical Activity Level')

plt.show()
```

```
data['Physical Activity Level'].unique()

array([42., 60., 30., 40., 75.,  2., 35., 45., 50., 32., 70., 80., 55.,
       90., 47., 65., 85.] )
```



Phân tích về mức độ hoạt động thể chất trong dữ liệu phản ánh một phạm vi rộng từ 2 đến 90 phút. Điều đáng chú ý, phân bố của mức độ hoạt động thể chất qua nhiều đỉnh trên biểu đồ, ngụ ý về sự đa dạng và chênh lệch về mức độ hoạt động thể chất giữa các quan sát trong dữ liệu. Sự phân bố thành 5 đỉnh rõ ràng có thể hiểu rằng có sự chênh lệch về cường độ hoạt động giữa các quan sát trong tập dữ liệu, tạo ra nền tảng để xác định các cấp độ hoạt động vận động khác nhau của mỗi cá nhân.

7. Stress Level

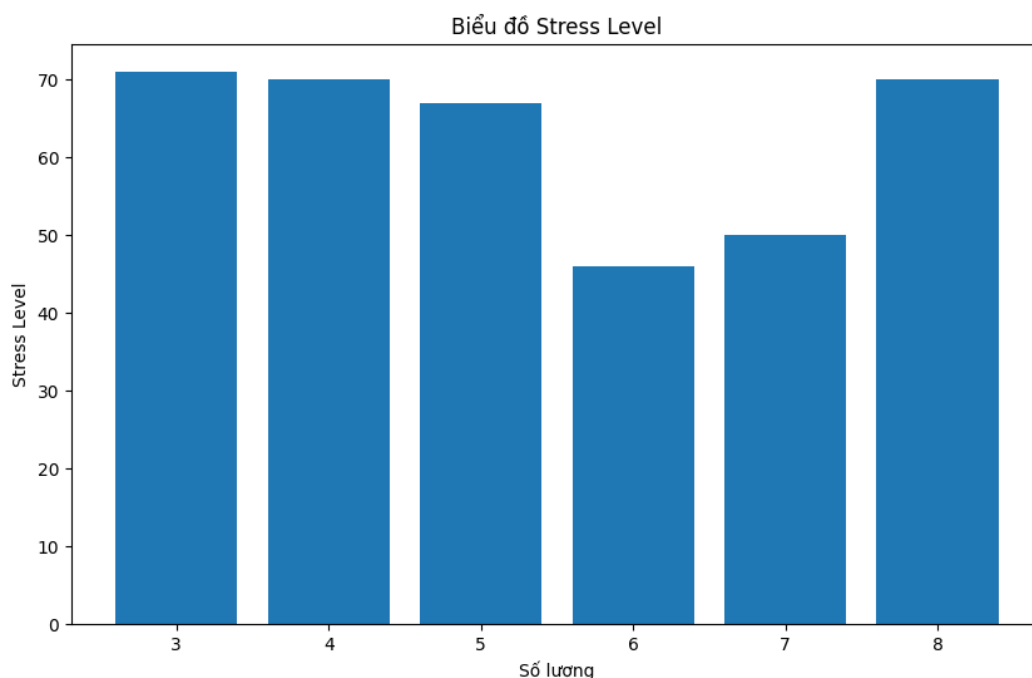
Thực hiện các thống kê cơ bản với biến numerical - Stress Level (Mức độ Stress):

```
# Tạo biểu đồ Bar Chart cho thuộc tính "Stress Level"
stress_Level_count = data['Stress Level'].value_counts()
plt.figure(figsize=(10, 6))
plt.bar(stress_Level_count.index, stress_Level_count.values)
plt.xlabel('Số lượng')
plt.ylabel('Stress Level')
plt.title('Biểu đồ Stress Level')

# Hiển thị biểu đồ
plt.show()
```

```
data['Stress Level'].describe()
```

```
count    374.000000
mean      5.385027
std       1.774526
min       3.000000
25%       4.000000
50%       5.000000
75%       7.000000
max       8.000000
Name: Stress Level, dtype: float64
```



Phân tích về mức độ áp lực trong dữ liệu cho thấy sự phân bố của các giá trị từ 3 đến 8, trong đó có sự chênh lệch không đáng kể. Điều này ngụ ý rằng bộ dữ liệu chứa các quan sát với các mức độ áp lực trải đều, tạo điều kiện thuận lợi cho việc phân tích so sánh các yếu tố tác động đến áp lực. Tuy nhiên, cần lưu ý rằng dữ liệu này dựa trên ý kiến chủ quan và tự đánh giá cá nhân, không dựa trên các phương pháp kiểm tra chính xác. Sự đa dạng trong quan niệm về mức độ áp lực có thể khiến phân tích trở nên không đáng tin cậy, do mỗi người có quan niệm khác nhau về mức độ ít/nhiều của áp lực. Nhóm cần tiếp tục xem xét cẩn thận và suy xét kỹ về các phân tích liên quan đến biến này để hiểu rõ hơn về ảnh hưởng của các yếu tố đến mức độ áp lực cá nhân.

Thêm vào đó, mặc dù kiểu dữ liệu hiện tại của biến là 'float64', tuy nhiên, biến này thực tế thuộc dạng định tính thứ tự, được lấy dữ liệu dựa trên phương pháp tự đánh giá. Để phản ánh chính xác loại dữ liệu và sự chủ quan trong việc đánh giá, ta quyết định chuyển đổi kiểu dữ liệu của biến này thành 'object'.

```
# Chuyển dạng dữ liệu của 'Stress Level' từ float64 thành object
data['Stress Level'] = data['Stress Level'].astype('object')
```

8. BMI Category

Thực hiện các thống kê cơ bản với biến categorical - BMI Category (Chỉ số BMI):

```
data['BMI Category'].describe()

count      374
unique       4
top      Normal
freq       195
Name: BMI Category, dtype: object
```

```
data['BMI Category'].unique()

array(['Overweight', 'Normal', 'Obese', 'Normal Weight'], dtype=object)
```

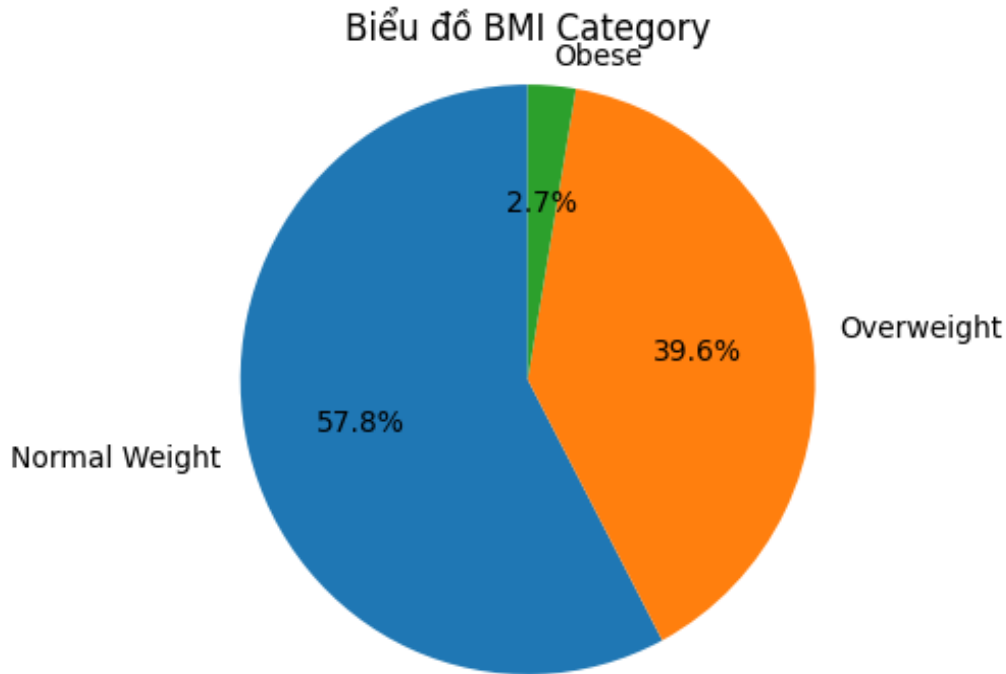
Sau khi khám phá các giá trị của biến chỉ số BMI, nhóm nhận thấy có 2 giá trị tương đồng nhau là 'Normal' và 'Normal Weight'. Do đó, nhóm quyết định thực hiện việc gộp chung lại thành một giá trị duy nhất là 'Normal Weight' để tạo sự thống nhất và làm dữ liệu trở nên dễ hiểu và tiện lợi hơn cho quá trình phân tích và diễn giải.

```
# Thay thế các giá trị Normal thành Normal Weight
data['BMI Category'] = data['BMI Category'].replace({'Normal': 'Normal Weight'})
```

```
# Tính toán số lượng mẫu cho từng giá trị của thuộc tính "BMI Category"
BMI_category_counts = data['BMI Category'].value_counts()

# Tạo biểu đồ Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(BMI_category_counts, labels=BMI_category_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Biểu đồ BMI Category')
plt.axis('equal') # Đảm bảo biểu đồ tròn

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Phân tích dữ liệu của biến categorical này cho thấy có 3 giá trị độc lập, trong đó giá trị phổ biến nhất là 'Normal' - đại diện cho nhóm người có chỉ số BMI bình thường. Ngoài ra, dữ liệu còn ghi nhận 2 giá trị BMI cho người thừa cân và béo phì. Mặc dù tỷ lệ tổng thể của nhóm này chỉ chiếm khoảng 40%, nhưng nó phản ánh khá gần tỷ lệ trên toàn cầu, với khoảng 37.5% dân số thế giới được ghi nhận là thừa cân hoặc béo phì. Điều này cho thấy sự tương quan của dữ liệu trong tập dữ liệu với tình trạng thực tế về chỉ số BMI của người dân trên toàn cầu.

9. Blood Pressure

Thực hiện các thống kê cơ bản với biến categorical - Blood Pressure (Huyết áp):

```
data['Blood Pressure'].describe()

count      374
unique      25
top        130/85
freq        100
Name: Blood Pressure, dtype: object
```

```
data['Blood Pressure'].unique()

array(['126/83', '125/80', '140/90', '120/80', '132/87', '130/86',
       '117/76', '118/76', '128/85', '131/86', '128/84', '115/75',
       '130/85', '135/88', '129/84', '115/78', '119/77', '121/79',
       '125/82', '135/90', '122/80', '142/92', '140/95', '139/91',
       '118/75'], dtype=object)
```

Sau khi xem xét các giá trị của huyết áp, nhóm đã nhận thấy có thể phân loại các chỉ số huyết áp thành các nhóm tương đồng nhau. Nhóm quyết định sử dụng phương pháp Rời rạc hóa để gộp các feature có ý nghĩa tương đồng lại với nhau. Nhóm quyết định phân thành 4 nhóm: Huyết áp bình thường, Huyết áp cao, Tăng huyết áp giai đoạn

1 và Tăng huyết áp giai đoạn 2. Việc này sẽ giúp tạo ra một cách phân loại rõ ràng và tiện lợi, đồng thời cung cấp cái nhìn tổng quan về tình trạng huyết áp của các quan sát trong tập dữ liệu.

```
# Mã hóa các giá trị của Blood Pressure thành 4 nhóm
data['Blood Pressure'].replace(['120/80','117/76','118/76','115/75',
                                '115/78','119/77','118/75'],
                                'HA Bình thường', inplace=True)

data['Blood Pressure'].replace(['125/80','121/79','122/80'],
                                'HA Cao', inplace=True)

data['Blood Pressure'].replace(['132/87','130/86','126/83','128/85','131/86',
                                '128/84','130/85','135/88','129/84','125/82'],
                                'Tăng HA giai đoạn 1', inplace=True)

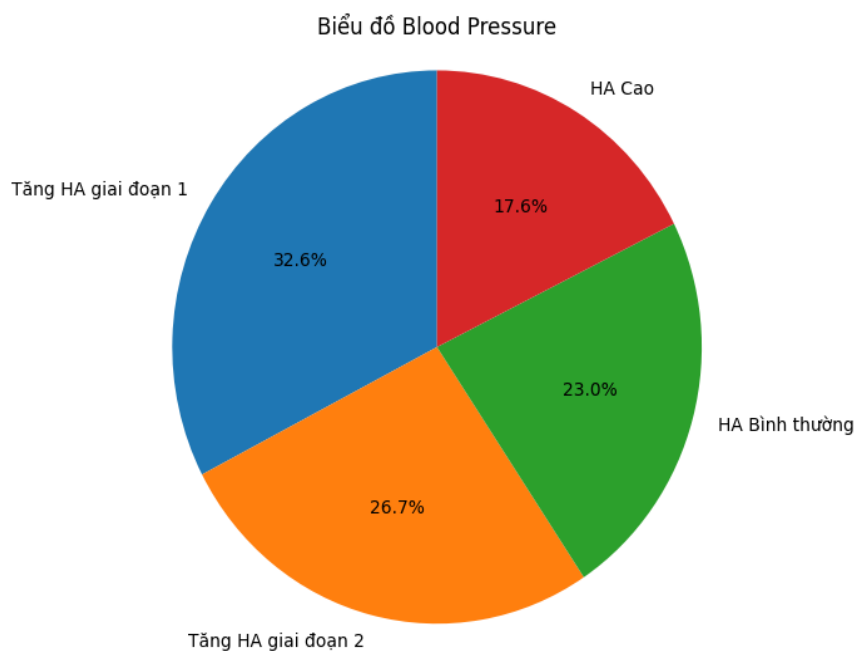
data['Blood Pressure'].replace(['140/90','135/90','142/92','140/95','139/91'],
                                'Tăng HA giai đoạn 2', inplace=True)

# 0 = Huyết áp bình thường (<120 và <80)
# 1 = Huyết áp cao (120-129 và <80)
# 2 = Tăng huyết áp giai đoạn 1 (130-139 hoặc 80-89)
# 3 = Tăng huyết áp giai đoạn 2 (>=140 hoặc >=90)
```

```
# Tính toán số lượng mẫu cho từng giá trị của thuộc tính "Blood Pressure"
blood_bressure_counts = data['Blood Pressure'].value_counts()

# Tạo biểu đồ Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(blood_bressure_counts, labels=blood_bressure_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Biểu đồ Blood Pressure')
plt.axis('equal') # Đảm bảo biểu đồ tròn

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Khi nhóm áp dụng phương pháp rời rạc hóa để phân nhóm các giá trị, nhóm nhận thấy rằng bộ dữ liệu đã được chia đều thành các nhóm với số lượng giá trị tương đồng. Điều này sẽ hữu ích trong việc phân tích tác động của huyết áp đến chất lượng giấc ngủ và các yếu tố khác, vì việc có các nhóm có số lượng gần bằng nhau giúp tạo điều kiện thuận lợi cho việc so sánh và phân tích tác động của biến huyết áp đối với các biến khác trong tập dữ liệu.

10. Heart Rate

Thực hiện các thống kê cơ bản với biến numerical - Heart Rate (Nhịp tim):

```
data['Heart Rate'].unique()

array([77., 75., 85., 82., 70., 80., 78., 69., 72., 68., 76., 81., 65.,
       84., 74., 67., 73., 83., 86.]
```

```
data['Heart Rate'].describe()

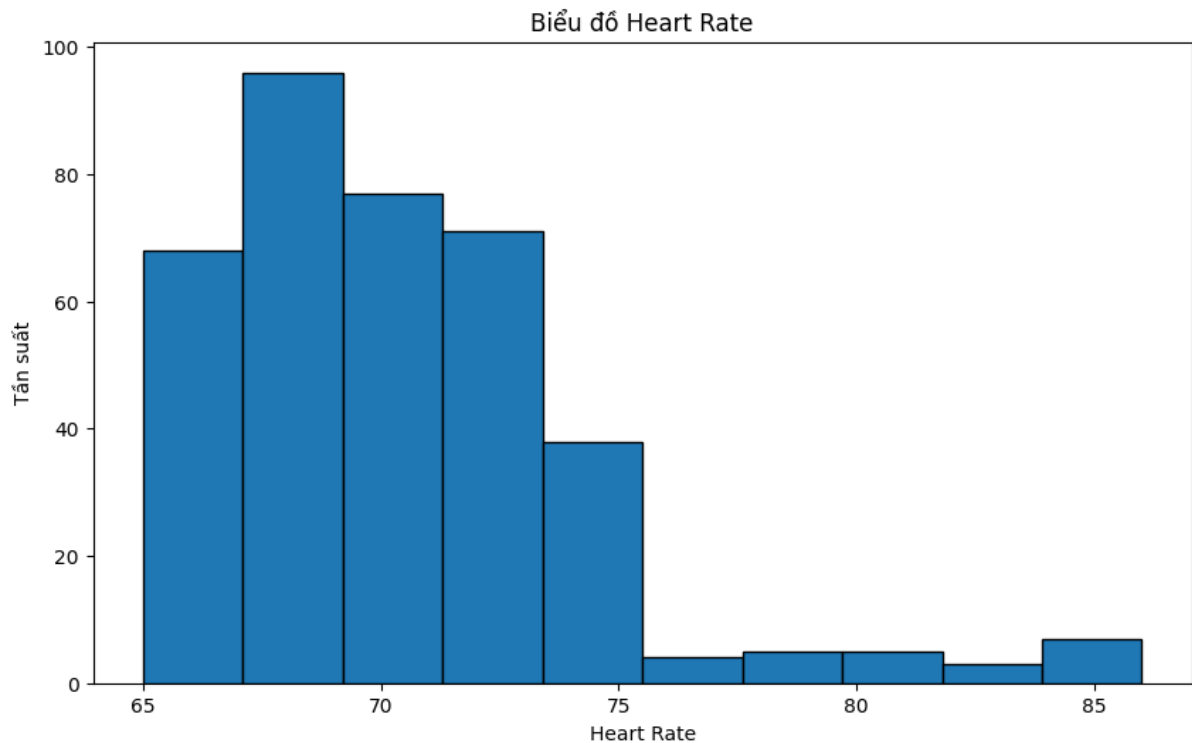
count    374.000000
mean      70.179144
std        4.127004
min       65.000000
25%       68.000000
50%       70.000000
75%       72.000000
max       86.000000
Name: Heart Rate, dtype: float64
```

Sau khi xem xét dữ liệu của biến nhịp tim, nhóm nhận thấy rằng giá trị của biến nằm trong khoảng từ 65 đến 86 nhịp một phút, đây là khoảng nhịp tim bình thường ở người trưởng thành. Hiện tại, kiểu dữ liệu của biến đang là float, do đó nhóm quyết định thực hiện việc chuyển đổi sang kiểu dữ liệu int vì đây là biến định lượng rời rạc, được đo bằng cách đếm.

```
# Chuyển kiểu dữ liệu của cột "Heart Rate" từ float thành int
data['Heart Rate'] = data['Heart Rate'].astype('int')
```

```
# Tạo biểu đồ Histogram cho thuộc tính "Heart Rate"
plt.figure(figsize=(10, 6))
plt.hist(data['Heart Rate'], bins=10, edgecolor='black')
plt.title('Biểu đồ Heart Rate')
plt.xlabel('Heart Rate')
plt.ylabel('Tần suất')

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Qua việc quan sát và phân tích biến, nhóm nhận thấy rằng biến nhịp tim không có tác động mạnh đối với quá trình phân tích. Lưu ý rằng dữ liệu về nhịp tim được ghi nhận khi cơ thể trong tình trạng nghỉ ngơi chứ không phải trong giấc ngủ, và hầu hết các giá trị nhịp tim trong bộ dữ liệu đều nằm trong ngưỡng bình thường của người trưởng thành. Do đó, nhóm quyết định loại bỏ biến này trong quá trình phân tích, để tập trung vào các yếu tố có ảnh hưởng lớn hơn đối với quá trình nghiên cứu.

11. Daily Steps

Thực hiện các thống kê cơ bản với biến numerical - Daily Steps (Số bước chân hàng ngày):

```
data['Daily Steps'].unique()
```

```
array([ 4200., 10000., 3000., 3500., 8000., 200., 4000., 4100.,
        6800., 5000., 7000., 5500., 5200., 5600., 3300., 4800.,
        7500., 7300., 6200., 6000., 20000., 3700.])
```

```
data['Daily Steps'].describe()
```

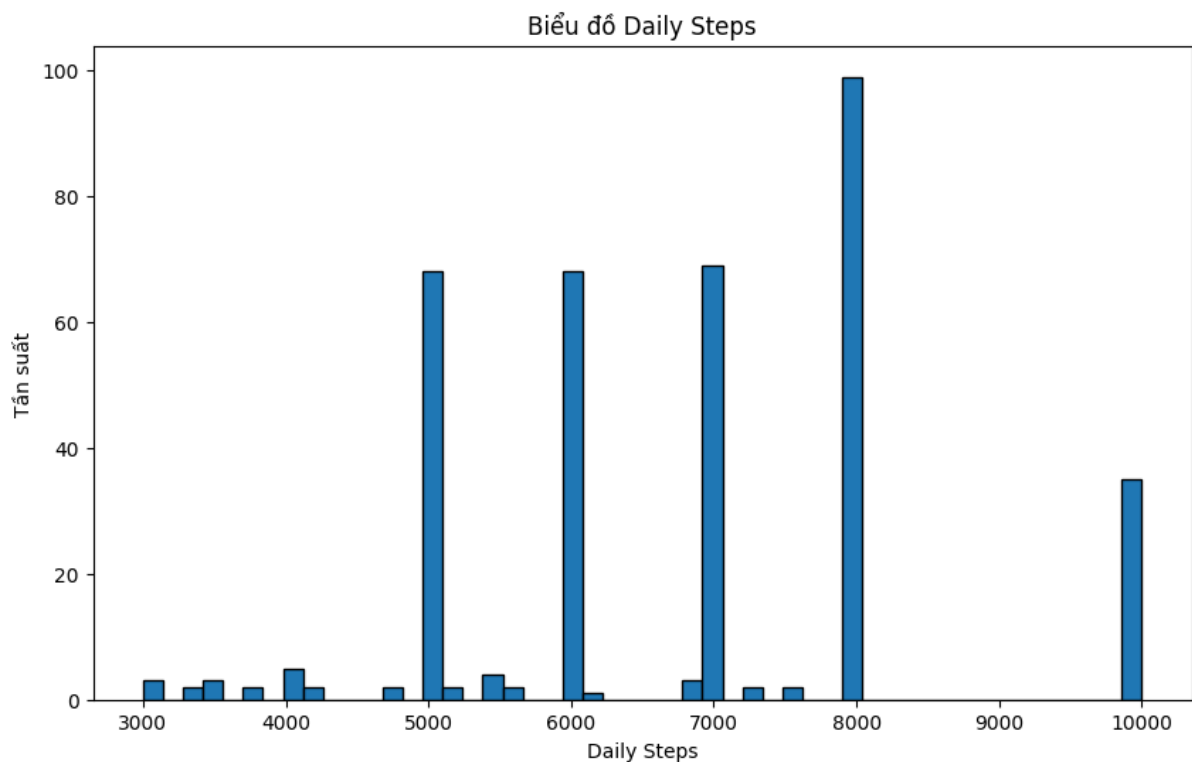
```
count      374.000000
mean       6820.053476
std        1779.432388
min         200.000000
25%        5525.000000
50%        7000.000000
75%        8000.000000
max       20000.000000
Name: Daily Steps, dtype: float64
```

Hiện tại, kiểu dữ liệu của biến đang là float, do đó nhóm quyết định thực hiện việc chuyển đổi sang kiểu dữ liệu int vì đây là biến định lượng rời rạc, được đo bằng cách đếm.

```
# Chuyển kiểu dữ liệu của cột "Daily Steps" từ float thành int
data['Daily Steps'] = data['Daily Steps'].astype('int')
```

```
# Tạo biểu đồ Histogram cho thuộc tính "Daily Steps"
plt.figure(figsize=(10, 6))
plt.hist(data['Daily Steps'], bins=50, edgecolor='black')
plt.title('Biểu đồ Daily Steps')
plt.xlabel('Daily Steps')
plt.ylabel('Tần suất')

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Sau khi tiến hành phân tích biến này, nhóm quan sát thấy dữ liệu đã được phân chia rõ ràng thành các nhóm: 5000, 6000, 7000, 8000 và 10000. Nhưng biến này không nói lên được nhiều sự khác nhau giữa các quan sát, ví dụ như một người tập gym có thể có số bước chân thấp nhưng cường độ hoạt động một ngày cao.

12. Sleep Disorder

Thực hiện các thống kê cơ bản với biến categorical - Sleep Disorder (Triệu chứng khi ngủ):

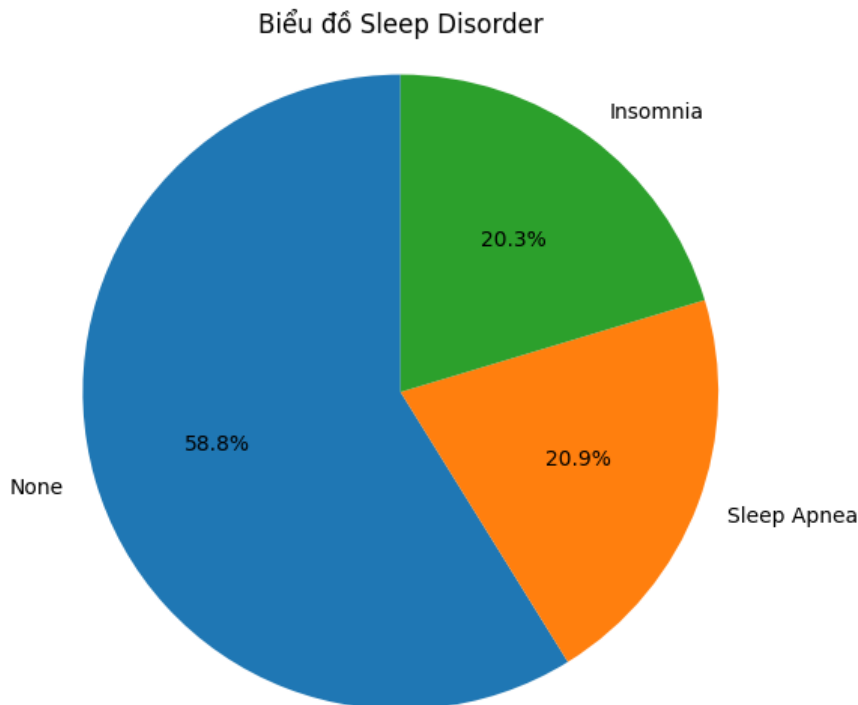
```
data['Sleep Disorder'].unique()
array(['None', 'Sleep Apnea', 'Insomnia'], dtype=object)
```

```
data['Sleep Disorder'].describe()
count      374
unique      3
top         None
freq       220
Name: Sleep Disorder, dtype: object
```

```
# Tính toán số lượng mẫu cho từng giá trị của thuộc tính "Sleep Disorder"
Sleep_Disorder_counts = data['Sleep Disorder'].value_counts()

# Tạo biểu đồ Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(Sleep_Disorder_counts, labels=Sleep_Disorder_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Biểu đồ Sleep Disorder')
plt.axis('equal') # Đảm bảo biểu đồ tròn

# Hiển thị biểu đồ
plt.show()
```



Nhận xét: Dựa trên kết quả phân tích, nhóm nhận thấy biến này có 3 giá trị: 'Không', 'Mất ngủ' và 'Ngưng thở khi ngủ'. Sự phân bố của chúng trong bộ dữ liệu được phân thành 1 'Mất ngủ' : 1 'Ngưng thở' : 3 'Không'. Điều này tạo ra một tỷ lệ phân bố đủ tốt để thực hiện phân tích và xử lý dữ liệu. Sự cân đối trong phân phối giữa các nhóm cho phép nhóm dễ dàng so sánh và rút ra kết luận ý nghĩa về tần suất của các hiện tượng này, cung cấp nền tảng vững chắc cho việc nghiên cứu và đánh giá tác động của chúng đối với các yếu tố khác trong tập dữ liệu.

Nhưng để thuận tiện cho việc thực hiện phân cụm thì nhóm quyết định giảm giá trị của biến Sleep Disorder thành 2 giá trị mới là “Yes” là những người mắc các triệu chứng khi ngủ và “No” là những người không mắc các triệu chứng khi ngủ.

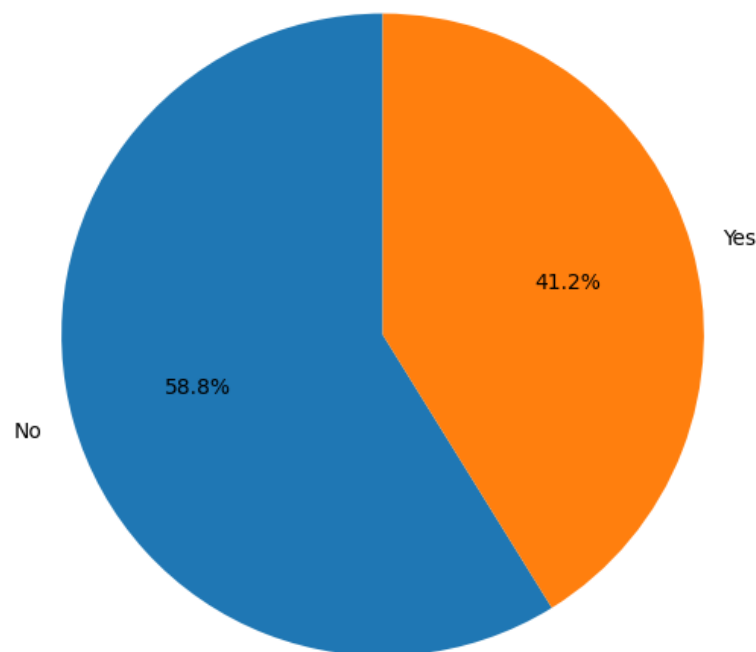
```
data['Sleep Disorder'].replace(['Insomnia', 'Sleep Apnea'], 'Yes', inplace=True)
data['Sleep Disorder'].replace(['None'], 'No', inplace=True)
```

```
# Tính toán số lượng mẫu cho từng giá trị của thuộc tính "Sleep Disorder"
Sleep_Disorder_counts = data['Sleep Disorder'].value_counts()

# Tạo biểu đồ Pie Chart
plt.figure(figsize=(6, 6))
plt.pie(Sleep_Disorder_counts, labels=Sleep_Disorder_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Biểu đồ Sleep Disorder')
plt.axis('equal') # Đảm bảo biểu đồ tròn

# Hiển thị biểu đồ
plt.show()
```

Biểu đồ Sleep Disorder



V. Phân phối dữ liệu

Nhóm đã thực hiện một loạt phân tích đối với các biến định lượng liên tục trong bộ dữ liệu. Đầu tiên, nhóm sử dụng biểu đồ phân phối để trực quan hóa hình dạng của phân phối dữ liệu và hiển thị giá trị trung bình và trung vị của phân phối. Sau đó, nhóm đã áp dụng kiểm định Shapiro-Wilk để đánh giá dữ liệu giúp nhóm hiểu xem dữ liệu có phân phối chuẩn hay không.

Nhóm đã sử dụng kiểm định Shapiro-Wilk để kiểm tra tính chuẩn của dữ liệu. Quy trình này bắt đầu với hai giả thuyết:

- Giả thuyết null (H_0) giả định rằng dữ liệu tuân theo một phân phối chuẩn.
- Giả thuyết thay thế (H_1) giả định rằng dữ liệu không tuân theo phân phối chuẩn.

Nhóm sử dụng hàm kiểm định 'scipy.stats.shapiro()' trong Python, để tính giá trị kiểm định và giá trị p. Nếu giá trị p lớn hơn một ngưỡng đã chọn trước (thông thường là 0.05), có thể kết luận rằng dữ liệu có thể được coi là tuân theo phân phối chuẩn.

Ngược lại, nếu giá trị p nhỏ hơn alpha, có đủ bằng chứng để kết luận rằng dữ liệu không tuân theo phân phối chuẩn.

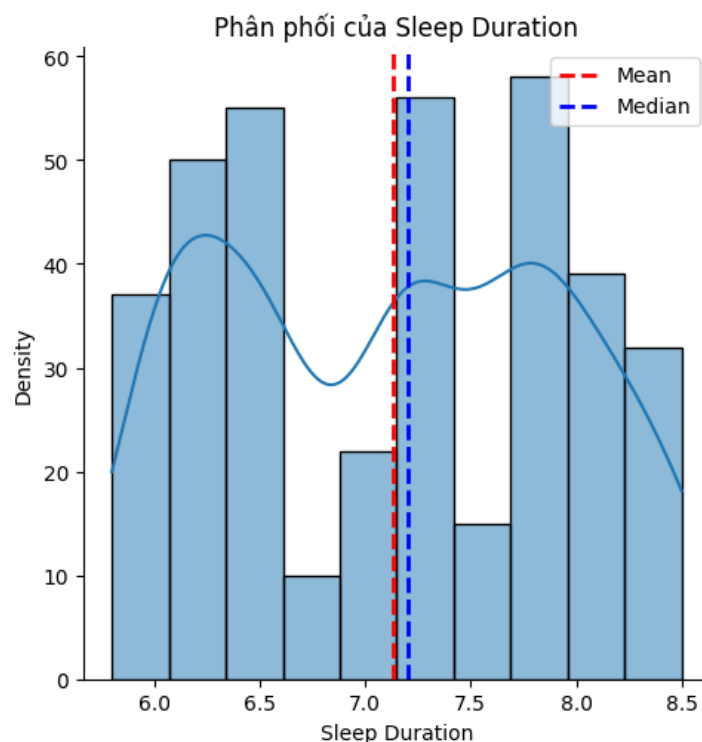
```
numerical_columns = ['Sleep Duration', 'Physical Activity Level']
for column in numerical_columns:
    sns.displot(data[column], kde=True)

    # Tính giá trị trung bình của dữ liệu
    mean_value = np.mean(data[column])
    median_value = np.median(data[column])

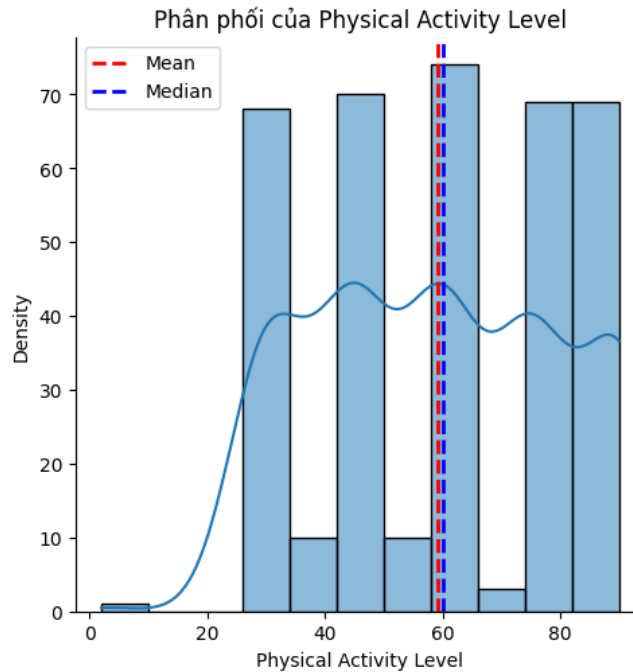
    # Vẽ đường thể hiện giá trị trung bình
    plt.axvline(mean_value, color='r', linestyle='dashed', linewidth=2, label='Mean')
    plt.axvline(median_value, color='b', linestyle='dashed', linewidth=2, label='Median')

    plt.title(f"Phân phối của {column}")
    plt.xlabel(column)
    plt.ylabel("Density")
    plt.legend()
    plt.show()

    # Kiểm định Shapiro-Wilk
    stat, p = stats.shapiro(data[column])
    print("statistic = ", stat)
    print("pvalue = ", p)
    if p > 0.05:
        print(f"Dữ liệu biến {column} có thể tuân theo phân phối chuẩn.", end="\n\n")
    else:
        print(f"Dữ liệu biến {column} không tuân theo phân phối chuẩn.", end="\n\n")
```



```
statistic = 0.9362137913703918
pvalue = 1.4187778556162822e-11
Dữ liệu biến Sleep Duration không tuân theo phân phối chuẩn.
```



```

statistic = 0.9086865782737732
pvalue = 3.0538234701421585e-14
Dữ liệu biến Physical Activity Level không tuân theo phân phối chuẩn.

```

Dựa trên kết quả kiểm định, chúng ta có thể thấy rằng tất cả các biến liên tục trong dữ liệu không tuân theo phân phối chuẩn. Điều này có nghĩa là chúng ta cần thay đổi cách tiếp cận khi sử dụng các kiểm định thống kê, vì dữ liệu không đáp ứng yêu cầu về tính chuẩn của chúng, nên ta có thể sử dụng các phương pháp thống kê phi tham số.

VI. Phân tích tính tương quan giữa các biến

1. Quan sát sự tương quan giữa các biến số

Hệ số tương quan Spearman là thước đo thống kê về độ mạnh của mối quan hệ đơn điệu giữa dữ liệu được ghép nối. Trong một mẫu, nó được ký hiệu và tính như sau:

$$r_s = \frac{\sum (R_i - \bar{R})(S_i - \bar{S})}{(\sum (R_i - \bar{R})^2 \sum (S_i - \bar{S})^2)^{0.5}}$$

Trong đó, R_i là giá trị thứ hạng của x ở vị trí i và S_i là giá trị thứ hạng của y ở vị trí i . \bar{R} và \bar{S} trung bình là các trung bình của các giá trị thứ hạng. Thông qua công thức trên, ta có thể thấy hệ số tương quan Spearman bị ràng buộc như sau:

$$-1 \leq r_s \leq 1$$

Khi hệ số tương quan càng gần ± 1 thì 2 biến tương quan càng mạnh. Chúng ta có thể mô tả bằng lời cường độ của mối tương quan như sau:

0.00 - 0.19: “rất yếu”

0.20 - 0.39: “yếu”

0.40 - 0.59: “vừa phải”

0.60 - 0.79: “mạnh”

0.80 - 1.00: “rất mạnh”

Thêm vào đó, việc tính toán hệ số tương quan của Spearman yêu cầu dữ liệu phải thỏa mãn các điều kiện như sau:

- Dữ liệu đo bằng thang đo thứ bậc, thang đo khoảng và thang đo tỷ lệ;
- Dữ liệu tương quan một cách đơn điệu.

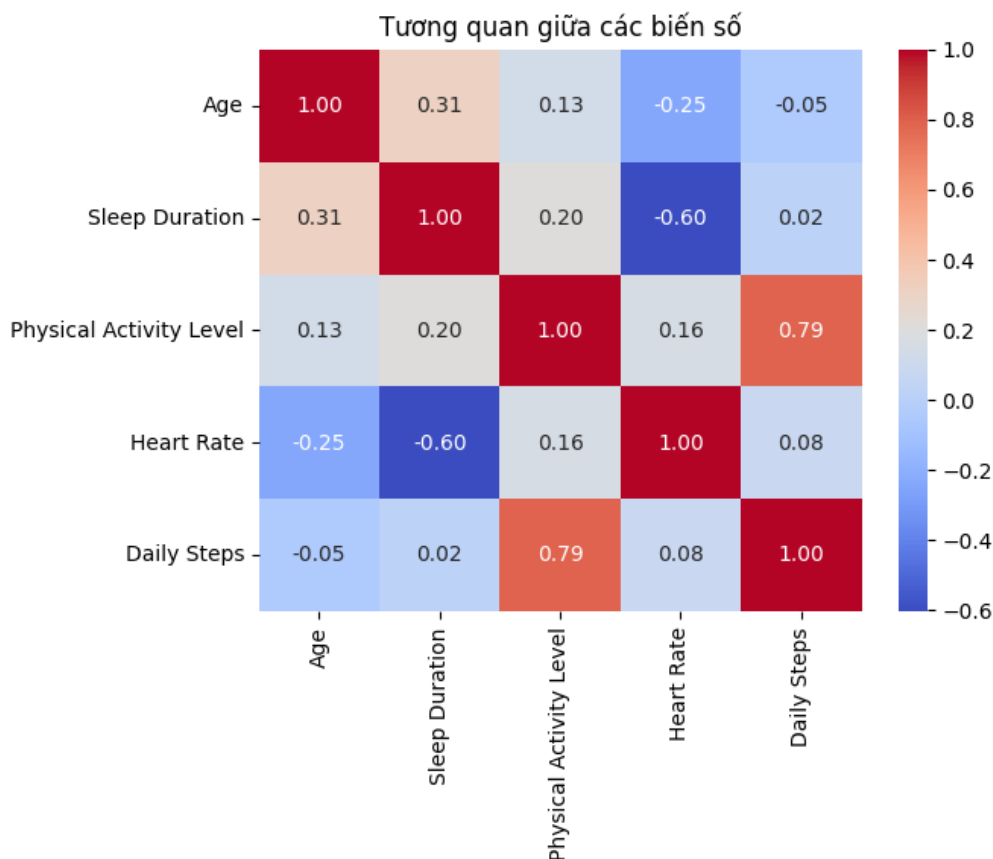
Không giống như hệ số tương quan Pearson, hệ số tương quan Spearman không có yêu cầu về phân phối chuẩn và do đó nó là một thống kê phi tham số.

Sau đây, nhóm thực hiện kiểm định cho các biến số:

```
# Tính ma trận tương quan sử dụng Spearman
correlation_matrix_spearman = data.corr(method="spearman")

# Vẽ biểu đồ Heatmap
sns.heatmap(correlation_matrix_spearman, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Tương quan giữa các biến số')

# Hiển thị biểu đồ
plt.show()
```



Dựa vào kết quả thu được ta có thể đưa ra các nhận xét sau:

- Biến Daily Steps và biến Physical Activity Level tương quan mạnh với nhau.
- Biến Heart Rate có mối tương quan mạnh với biến Sleep Duration.

2. Quan sát sự ảnh hưởng giữa các biến phân loại

Kiểm định Chi bình phương là một kiểm định thống kê được sử dụng để tìm ra sự khác biệt giữa dữ liệu được quan sát và dữ liệu dự kiến. Chúng ta có thể sử dụng kiểm định này để tìm ra mối tương quan giữa các biến phân loại trong bộ dữ liệu. Mục đích của kiểm định này là để xác định xem sự khác biệt giữa 2 biến phân loại là do ngẫu nhiên hay do tồn tại mối quan hệ giữa chúng.

Đối với kiểm định này, dữ liệu phải đáp ứng các yêu cầu sau:

- Hai biến là biến phân loại
- Cỡ mẫu tương đối lớn
- Danh mục của các biến là hai hoặc nhiều hơn
- Các quan sát độc lập với nhau

Trong lúc tiến hành kiểm định Chi bình phương, đầu tiên chúng ta phải xem xét 2 giả thuyết như sau:

- H_0 (Giả thuyết rỗng) = 2 biến độc lập với nhau.
- H_1 (Giả thuyết thay thế) = 2 biến ảnh hưởng nhau.

Nếu giá trị p thu được sau khi tiến hành kiểm định nhỏ hơn 0.05 thì chúng ta bác bỏ giả thuyết rỗng và chấp nhận giả thuyết thay thế. Nếu giá trị p lớn hơn 0.05 thì chúng ta chấp nhận giả thuyết rỗng và bác bỏ giả thuyết thay thế.

Thực hiện kiểm định cho các biến phân loại:

```
# Tạo heatmap cho từng cặp biến
variables = ['Gender', 'Occupation', 'Quality of Sleep', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Sleep Disorder']
# Tạo ma trận chứa giá trị kiểm định
significant_pairs = {}

plt.figure(figsize=(12, 8))

alpha = 0.05
for i in range(len(variables)):
    x = variables[i]
    significant_pairs[x] = []
    for j in range(len(variables)):
        if i == j:
            significant_pairs[x].append(0)
            continue

        var1 = variables[i]
        var2 = variables[j]

        # Tạo bảng tần số (contingency table)
        contingency_table = pd.crosstab(data[var1], data[var2])

        # Kiểm định chi-square
        chi2, p_value, _, _ = chi2_contingency(contingency_table)

        if p_value < alpha:
            significant_pairs[x].append(chi2)
        else:
            significant_pairs[x].append(0)
print(significant_pairs)
```

Tương quan giữa các biến phân loại

Gender	0.00	271.28	84.95	184.37	50.71	123.47	30.60
Occupation	271.28	0.00	719.81	853.54	361.98	616.90	200.74
Quality of Sleep	84.95	719.81	0.00	844.37	180.79	208.16	87.48
Stress Level	184.37	853.54	844.37	0.00	102.57	460.63	99.51
BMI Category	50.71	361.98	180.79	102.57	0.00	247.46	241.42
Blood Pressure	123.47	616.90	208.16	460.63	247.46	0.00	198.67
Sleep Disorder	30.60	200.74	87.48	99.51	241.42	198.67	0.00
	Gender	Occupation	Quality of Sleep	Stress Level	BMI Category	Blood Pressure	Sleep Disorder

Dựa vào kết quả thu được ta có thể đưa ra các nhận xét là tất cả các biến phân loại đều có sự ảnh hưởng với nhau.

3. Quan sát sự ảnh hưởng giữa các biến phân loại và biến số

Tiếp đến, nhóm tìm hiểu về kiểm định Kruskal-Wallis để đưa ra sự ảnh hưởng giữa các biến phân loại và biến định lượng liên tục. Ở đây nhóm không sử dụng kiểm định ANOVA vì kiểm định này yêu cầu bộ dữ liệu phải tuân theo phân phối chuẩn, và bộ dữ liệu ở đồ án này không đáp ứng được yêu cầu trên.

Kiểm định Kruskal-Wallis, đôi khi được gọi là “Kiểm định ANOVA một chiều theo xếp hạng”, là một kiểm tra phi tham số dựa trên xếp hạng, được sử dụng để kiểm tra xem có hay không sự khác biệt có ý nghĩa thống kê giữa 2 hoặc nhiều nhóm. Để có được một kết quả hợp lệ, một bộ dữ liệu phải thỏa mãn các giả định như sau:

- Dữ liệu có thể là định tính hoặc định lượng.
- Các mẫu thuộc các nhóm phải độc lập với nhau và được lấy mẫu ngẫu nhiên độc lập, không được lấy theo cặp hoặc không được ghép cặp, mỗi mẫu chỉ thuộc về một nhóm.
- Phân phối của các nhóm đối với biến độc lập đối với từng nhóm phải không chuẩn, các nhóm không phải có cùng phân phối, không có sự khác biệt lớn giữa phương sai của các nhóm.
- Không có sự tương quan giữa các biến.

Kiểm định Kruskal-Wallis được ký hiệu và tính toán như sau:

$$H = \left[\frac{12}{N(N+1)} \sum_{j=1}^k n_j \bar{R}_j^2 \right] - 3(N+1)$$

Trong đó k là số nhóm (mẫu độc lập), n_j là số quan sát trong nhóm thứ j, N là tổng số quan sát trong mẫu kết hợp và \bar{R} trung bình là giá trị trung bình của các cấp trong nhóm thứ j.

Trong lúc tiến hành kiểm định Kruskal-Wallis, đầu tiên chúng ta phải xem xét 2 giả thuyết như sau:

- H_0 (Giả thuyết rỗng) = các trung vị của mỗi nhóm đều giống nhau, nghĩa là tất cả các nhóm đều có cùng một phân bố.
- H_1 (Giả thuyết thay thế) = ít nhất một trong các nhóm có trung vị khác, nghĩa là ít nhất một nhóm đến từ một phân phối khác với các nhóm khác.

Nếu giá trị p thu được sau khi tiến hành kiểm định nhỏ hơn 0.05 thì chúng ta bác bỏ giả thuyết rỗng và chấp nhận giả thuyết thay thế. Nếu giá trị p lớn hơn 0.05 thì chúng ta chấp nhận giả thuyết rỗng và bác bỏ giả thuyết thay thế.

Thực hiện kiểm định cho các biến phân loại:

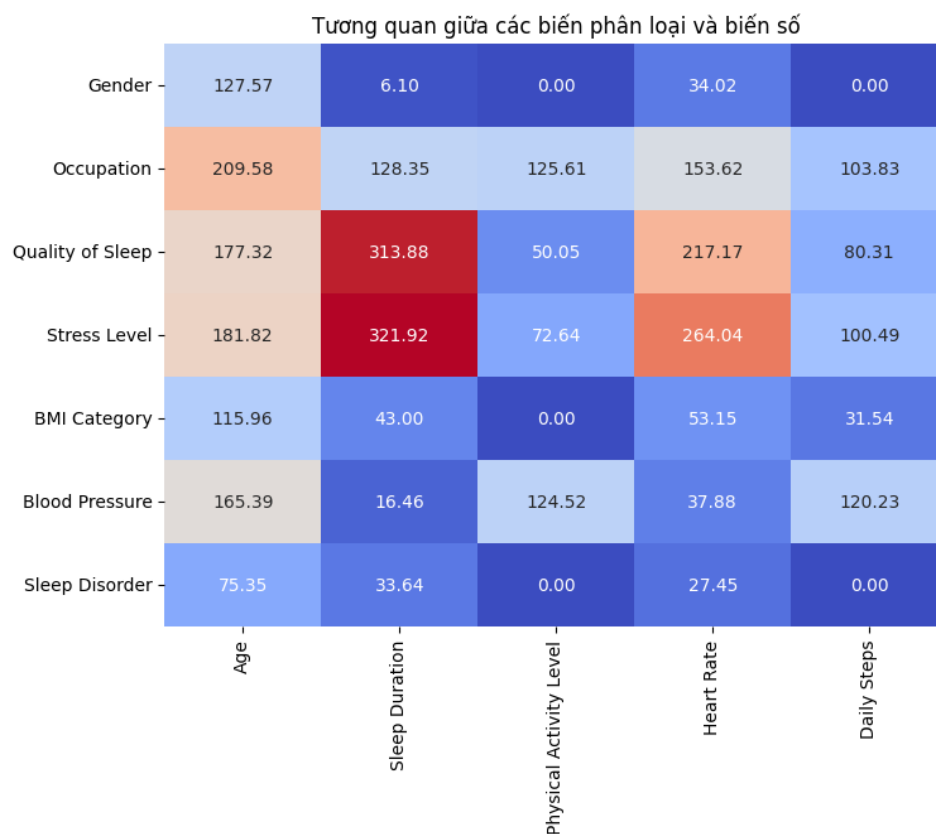
```
# Tạo heatmap cho từng cặp biến
numericals = ['Age', 'Sleep Duration', 'Physical Activity Level', 'Heart Rate', 'Daily Steps']
categoricals = ['Gender', 'Occupation', 'Quality of Sleep', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Sleep Disorder']
# Tạo ma trận chứa giá trị kiểm định
significant_pairs = {}

plt.figure(figsize=(12, 8))

alpha = 0.05
for i in range(len(numericals)):
    x = numericals[i]
    significant_pairs[x] = []
    for j in range(len(categoricals)):
        gb = data.groupby(categoricals[j])[x]
        datagb = [group for name, group in gb]

        stat, p_value = stats.kruskal(*datagb)

        if p_value < alpha:
            significant_pairs[x].append(stat)
        else:
            significant_pairs[x].append(0)
print(significant_pairs)
```



Dựa vào kết quả thu được ta có thể đưa ra các nhận xét sau, tất cả các cặp biến đều ảnh hưởng nhau ngoại trừ các biến sau:

- Biến Gender không ảnh hưởng đến các biến Physical Activity Level và Daily Steps.
- Biến BMI Category không ảnh hưởng đến biến Physical Activity Level.
- Biến Sleep Disorder không ảnh hưởng đến biến Physical Activity Level và Daily Steps.

VII. Tổng quan dữ liệu sau xử lý

	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	Male	27	Software Engineer	6.1	6.0	42.0	6	Overweight	Tăng HA giai đoạn 1	77	4200	No
1	Male	28	Doctor	6.2	6.0	60.0	8	Normal Weight	HA Cao	75	10000	No
2	Male	28	Doctor	6.2	6.0	60.0	8	Normal Weight	HA Cao	75	10000	No
3	Male	28	Sales Representative	5.9	4.0	30.0	8	Obese	Tăng HA giai đoạn 2	85	3000	Yes
4	Male	28	Sales Representative	5.9	4.0	30.0	8	Obese	Tăng HA giai đoạn 2	85	3000	Yes
...
369	Female	59	Nurse	8.1	9.0	75.0	3	Overweight	Tăng HA giai đoạn 2	68	7000	Yes
370	Female	59	Nurse	8.0	9.0	75.0	3	Overweight	Tăng HA giai đoạn 2	68	7000	Yes
371	Female	59	Nurse	8.1	9.0	75.0	3	Overweight	Tăng HA giai đoạn 2	68	7000	Yes
372	Female	59	Nurse	8.1	9.0	75.0	3	Overweight	Tăng HA giai đoạn 2	68	7000	Yes
373	Female	59	Nurse	8.1	9.0	75.0	3	Overweight	Tăng HA giai đoạn 2	68	7000	Yes

```
## Thông tin chi tiết bộ dữ liệu sau xử lý
print('=== THÔNG TIN BỘ DỮ LIỆU SAU XỬ LÝ===')
print('\n>> Số lượng phần tử trong bộ dữ liệu:', data.size)
print('>> Kích thước bộ dữ liệu (số dòng, số cột):', data.shape)
data.info()
```

```
=== THÔNG TIN BỘ DỮ LIỆU SAU XỬ LÝ===

>> Số lượng phần tử trong bộ dữ liệu: 4488
>> Kích thước bộ dữ liệu (số dòng, số cột): (374, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 12 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Gender                                374 non-null    object
1   Age                                    374 non-null    int64
2   Occupation                            374 non-null    object
3   Sleep Duration                        374 non-null    float64
4   Quality of Sleep                      374 non-null    object
5   Physical Activity Level               374 non-null    float64
6   Stress Level                          374 non-null    object
7   BMI Category                          374 non-null    object
8   Blood Pressure                        374 non-null    object
9   Heart Rate                           374 non-null    int64
10  Daily Steps                           374 non-null    int64
11  Sleep Disorder                        374 non-null    object
dtypes: float64(2), int64(3), object(7)
memory usage: 35.2+ KB
```

Chương 3: Phân tích và khai thác dữ liệu

I. Phân cụm

1. Định nghĩa phân cụm

Phân cụm là quá trình tổ chức các đối tượng hoặc mẫu dữ liệu vào các nhóm hoặc cụm dựa trên sự tương đồng giữa chúng. Mục tiêu chính của phân cụm là tạo ra các nhóm có tính chất tương tự bên trong nhóm và khác biệt với nhóm khác. Quá trình này giúp chúng ta hiểu và phân tích dữ liệu, khám phá các mẫu tiềm ẩn và tạo ra những thông tin hữu ích từ dữ liệu không được gán nhãn hay không có cấu trúc. Hai phương pháp phân cụm sẽ được áp dụng đối với bộ dữ liệu trên bao gồm: phân cụm phân cấp và phân cụm phân hoạch

2. Mục đích phân cụm

Mục đích phân cụm cho bộ dữ liệu bao gồm:

- Lựa chọn mô hình tốt nhất để phân cụm cho bộ dữ liệu: Nhằm tìm ra mô hình phân cụm phù hợp nhất với bộ dữ liệu cụ thể. Qua việc so sánh và đánh giá các phương pháp phân cụm khác nhau, chúng ta có thể chọn mô hình tốt nhất để áp dụng cho bộ dữ liệu cụ thể.
- Khám phá các cấu trúc ẩn trong dữ liệu: Khám phá và hiểu các cấu trúc ẩn, mối quan hệ và mẫu trong dữ liệu. Qua việc phân cụm, chúng ta có thể tìm ra các nhóm dữ liệu có tính chất tương tự và phân biệt với nhóm khác. Điều này giúp chúng ta hiểu sâu hơn về dữ liệu và tạo ra những thông tin mới từ việc phân tích cấu trúc.
- Quan sát kết quả phân cụm thông qua các biến mục tiêu: Quan sát và phân tích kết quả phân cụm đối với các biến mục tiêu là Quality of Sleep, Stress Level, Sleep Duration và Sleep Disorder. Đánh giá và so sánh các nhóm dữ liệu được phân cụm. Điều này giúp chúng ta hiểu rõ hơn về mối liên hệ giữa các biến mục tiêu và các nhóm dữ liệu.

3. Tiền xử lý trước phân cụm

a. Mã hóa dữ liệu

Mã hóa dữ liệu là một bước quan trọng trong quá trình tiền xử lý dữ liệu, giúp chuyển đổi thông tin từ dạng không thể sử dụng trực tiếp vào mô hình thành dạng có thể áp dụng các thuật toán phân tích. Trong đề án này, nhóm quyết định sử dụng chủ yếu hai phương pháp mã hóa phổ biến: Label Encoding và One-Hot Encoding.

Label Encoding: Label Encoding là một kỹ thuật chuyển đổi các giá trị của biến có giá trị định tính thứ tự thành các giá trị số nguyên tăng dần và mỗi giá trị độc lập của biến được ánh xạ sang một số nguyên duy nhất. Trong đề án này, những biến có thể áp dụng Label Encoding gồm các biến là Quality of Sleep, Stress Level, BMI Category và Blood Pressure.

- Quality of Sleep: Đánh giá chủ quan về chất lượng giấc ngủ với mức thấp nhất là 4 và mức cao nhất là 9.

- Stress Level: Đánh giá chủ quan về mức độ căng thẳng của một người, dao động từ 3 đến 8. Tương tự như Quality of Sleep, Stress Level có thể được mã hóa thành các giá trị số nguyên dựa trên mức độ căng thẳng.
- BMI Category: các loại BMI như "Underweight" được mã hóa thành 0, "Normal" mã hóa thành 1, "Overweight" mã hóa thành 2.
- Blood Pressure: Số đo huyết áp của một người, được biểu thị bằng huyết áp tâm thu so với huyết áp tâm trương, gồm các giá trị 'HA Bình thường' được mã hóa thành 0, 'HA Cao' được mã hóa thành 1, 'Tăng HA giai đoạn 1' mã hóa thành 2 và 'Tăng HA giai đoạn 2' được mã hóa thành 3.

```
# Quality of Sleep
data_cluster['Quality of Sleep'] = data_cluster['Quality of Sleep'].astype('int64')
# Stress Level
data_cluster['Stress Level'] = data_cluster['Stress Level'].astype('int64')
# BMI Category
data_cluster['BMI Category'] = data_cluster['BMI Category'].replace({'Normal Weight': 0})
data_cluster['BMI Category'] = data_cluster['BMI Category'].replace({'Overweight': 1})
data_cluster['BMI Category'] = data_cluster['BMI Category'].replace({'Obese': 2})
# Blood Pressure
data_cluster['Blood Pressure'] = data_cluster['Blood Pressure'].replace({'HA Bình thường': 0})
data_cluster['Blood Pressure'] = data_cluster['Blood Pressure'].replace({'HA Cao': 1})
data_cluster['Blood Pressure'] = data_cluster['Blood Pressure'].replace({'Tăng HA giai đoạn 1': 2})
data_cluster['Blood Pressure'] = data_cluster['Blood Pressure'].replace({'Tăng HA giai đoạn 2': 3})
```

One-Hot Encoding: One-Hot Encoding là một phương pháp mã hóa dữ liệu dạng định danh (categorical) thành dạng số bằng cách tạo ra các biến giả (dummy variables) đối với mỗi giá trị của biến gốc. Mỗi biến giả chỉ mang giá trị 0 hoặc 1, thể hiện sự có mặt hoặc không có mặt của giá trị đó. Trong đồ án của chúng ta, có một số biến như Gender, Occupation, và Sleep Disorder có thể được mã hóa bằng phương pháp này.

- Gender: Gồm hai giá trị là "Male" và "Female" ta tiến hành tạo ra hai cột mới, mỗi cột đại diện cho một giá trị. Mỗi hàng trong cột tương ứng sẽ được đặt là 1 nếu giá trị của hàng đó trùng khớp với giới tính tương ứng và 0 nếu ngược lại.
- Occupation: Tương tự, với 10 loại nghề nghiệp khác nhau gồm 'Software Engineer', 'Doctor', 'Sales Representative', 'Teacher', 'Nurse', 'Scientist', 'Lawyer', 'Salesperson' và 'Manager', ta tạo ra một cột cho mỗi loại và ánh xạ giá trị tương ứng.
- Sleep Disorder: Gồm hai giá trị 'Yes' và 'No' nên ta tạo ra một cột mới với 1 nghĩa là có triệu chứng khi ngủ, nếu không có thì trả giá trị là 0.

```
# Gender
# Occupation
# Sleep Disorder
df_encoded = pd.get_dummies(data_cluster, columns=['Gender', 'Occupation', 'Sleep Disorder'], prefix=['Gender', 'Occupation', 'Sleep Disorder'], drop_first=True)
data_cluster = pd.concat([data_cluster, df_encoded], axis=1)
```

Sau khi đã thực hiện One-Hot Encoding cho các biến Gender, Occupation và Sleep Disorder, ta tiến hành xóa bỏ các cột gốc đã được mã hóa thành các biến giả mới

nhằm làm giảm kích thước của DataFrame cũng như lược bỏ các cột trùng nhau có thể sẽ xuất hiện trong bộ data nếu có sự trùng lặp trong dữ liệu hoặc do quá trình mã hóa.

```
# Xóa các cột ban đầu và bỏ bớt các cột nhị phân
selected_columns = ['Gender', 'Occupation', 'Sleep Disorder']
data_cluster.drop(selected_columns, axis=1, inplace=True)
# Xóa các cột trùng nhau
data_cluster = data_cluster.T.drop_duplicates().T
```

Cấu trúc của bộ dữ liệu sau khi mã hóa dữ liệu như sau:

```
data_cluster.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 21 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Age                                       374 non-null    float64
1   Sleep Duration                         374 non-null    float64
2   Quality of Sleep                       374 non-null    float64
3   Physical Activity Level                 374 non-null    float64
4   Stress Level                           374 non-null    float64
5   BMI Category                           374 non-null    float64
6   Blood Pressure                         374 non-null    float64
7   Heart Rate                             374 non-null    float64
8   Daily Steps                            374 non-null    float64
9   Gender_Male                            374 non-null    float64
10  Occupation_Doctor                       374 non-null    float64
11  Occupation_Engineer                     374 non-null    float64
12  Occupation_Lawyer                       374 non-null    float64
13  Occupation_Manager                      374 non-null    float64
14  Occupation_Nurse                        374 non-null    float64
15  Occupation_Sales Representative          374 non-null    float64
16  Occupation_Salesperson                  374 non-null    float64
17  Occupation_Scientist                    374 non-null    float64
18  Occupation_Software Engineer            374 non-null    float64
19  Occupation_Teacher                      374 non-null    float64
20  Sleep Disorder_Yes                      374 non-null    float64
dtypes: float64(21)
memory usage: 61.5 KB
```

b. Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là quá trình biến đổi dữ liệu về một khoảng giá trị cụ thể để đảm bảo rằng tất cả các biến đều có cùng một thang đo, mục tiêu là để giữ cho dữ liệu ổn định và thống nhất, đồng thời loại bỏ ảnh hưởng của đơn vị đo lường và đơn vị đo khác nhau. Trong đồ án này, nhóm quyết định chuẩn hóa dữ liệu trước khi tiến hành phân cụm nhằm:

- **Đảm bảo các biến có cùng đơn vị:** Khi phân cụm, các thuật toán thường dựa vào khoảng cách giữa các điểm dữ liệu. Nếu các biến có đơn vị đo lường khác nhau, sự khác biệt này có thể tạo ra ảnh hưởng không mong muốn trong việc đánh giá khoảng cách.

- **Giảm ảnh hưởng của biến lớn:** Nếu có biến có giá trị lớn hơn đáng kể so với các biến khác, nó có thể chiếm ưu thế trong quá trình phân cụm, làm giảm tác động của các biến khác.
- **Hỗ trợ thuật toán:** Nhiều thuật toán phân cụm (như K-Means) dựa vào độ đo khoảng cách. Việc chuẩn hóa giúp thuật toán hội tụ nhanh chóng hơn và làm tăng hiệu suất của mô hình.
- **Đảm bảo phân phối đối với một số thuật toán:** Một số thuật toán, đặc biệt là các thuật toán dựa trên phân phối, yêu cầu dữ liệu của chúng tuân theo một phân phối chuẩn. Chuẩn hóa giúp làm giảm độ biến động và đưa dữ liệu về gần với phân phối chuẩn.
- **Tăng độ nhạy của mô hình:** Chuẩn hóa có thể giúp tăng độ nhạy của mô hình đối với biến và thông tin quan trọng hơn, đặc biệt là đối với các thuật toán như Support Vector Machines (SVM).
- **Giảm đối tượng nhiễu (Outliers):** Khi chuẩn hóa, tác động của các giá trị nhiễu có thể được giảm, giúp mô hình phân cụm trở nên ổn định hơn và ít bị ảnh hưởng bởi các điểm dữ liệu nhiễu.

Tóm lại, chuẩn hóa dữ liệu trước khi phân cụm giúp đảm bảo rằng mô hình sẽ hoạt động hiệu quả hơn và có khả năng tổng quát hóa tốt hơn trên dữ liệu mới. Trong đồ án này, nhóm sử dụng phương pháp chuẩn hóa Min-Max để đưa dữ liệu về miền giá trị nằm trong khoảng $[0,1]$ như sau:

```
# Chuẩn hóa dữ liệu
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data_cluster)

# Tạo DataFrame mới từ dữ liệu đã chuẩn hóa
data_cluster = pd.DataFrame(scaled_data, columns=data_cluster.columns)
```

c. Elbow phương sai

Elbow phương sai là một phương pháp quan trọng trong quá trình phân tích dữ liệu nhằm xác định số lượng cụm trong phân cụm bằng giá trị tối ưu dựa trên sự giảm đột ngột của phương sai khi số lượng cụm tăng. Phương pháp này bao gồm việc vẽ biểu đồ giải thích phương sai dựa trên số lượng cụm và chọn điểm gãy của đường cong làm số lượng cụm cần sử dụng. Việc sử dụng "điểm gãy" hoặc "đầu gối của đường cong" như một điểm cắt là một phương pháp trong tối ưu hóa toán học để chọn một điểm nơi lợi ích giảm đáng kể và không còn xứng đáng với chi phí bổ sung. Trong phân cụm, điều này có nghĩa là người ta nên chọn một số lượng cụm sao cho việc thêm cụm không cung cấp mô hình hóa tốt hơn cho dữ liệu.

Phân tích thành phần chính (PCA) giúp chúng ta tìm ra các chiều mới (thành phần chính) sao cho chúng giữ lại lượng lớn thông tin trong dữ liệu gốc. Chính vì vậy để tìm ra tham số K phù hợp là rất quan trọng trong việc giảm chiều dữ liệu. Trong đồ

án này, nhóm quyết định áp dụng phương pháp Elbow phương sai vào việc chọn số lượng thành phần chính K cho PCA nhằm:

- **Tối ưu hóa số lượng thành phần chính:** Chọn K dựa trên Elbow phương sai giúp tối ưu hóa số lượng thành phần chính sao cho chúng giữ lại độ lớn thông tin mong muốn và giảm chiều dữ liệu một cách hiệu quả.
- **Hiểu rõ hơn về cấu trúc dữ liệu:** Elbow phương sai giúp hiểu rõ về cấu trúc ẩn trong dữ liệu, đồng thời giúp xác định cách giữ lại phương sai đủ lớn để đảm bảo tính đại diện.
- **Quản lý overfitting:** Chọn K tại điểm Elbow giúp giảm nguy cơ overfitting và tăng tính tổng quát của mô hình.

Trước hết, chúng ta thực hiện PCA khi chưa xác định được K và giữ nguyên số chiều ban đầu:

```
## Áp dụng PCA (chưa xác định k --> giữ nguyên số chiều)
pca = PCA().fit(data_cluster)
```

Tiến hành vẽ biểu đồ biểu diễn phần trăm phương sai tích lũy theo số features

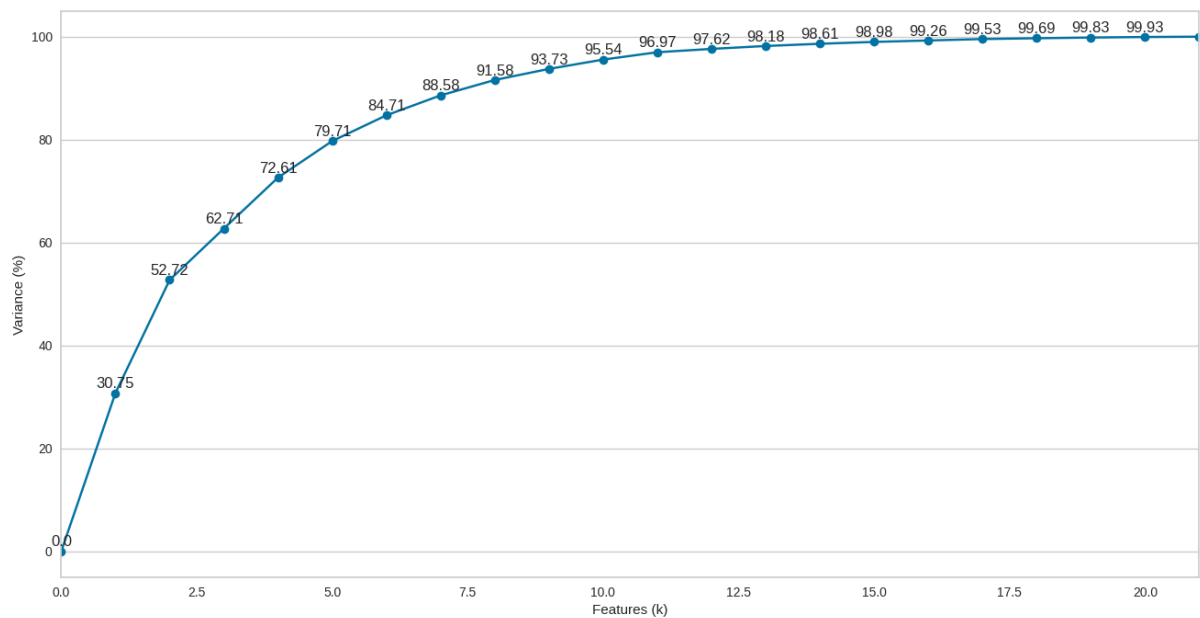
```
## Vẽ đồ thị biểu diễn % phương sai tích lũy theo số features --> chọn k theo điểm "gãy"
nb_features = data_cluster.shape[1] - 1
features = data_cluster.columns[:nb_features]
print('* Số lượng features = %2d' % nb_features)
print(' Các features:', ', '.join(features))

points = np.cumsum(pca.explained_variance_ratio_) * 100 # Các điểm dữ liệu
points = np.insert(points, 0, 0) # Thêm điểm k = 0, variance = 0
x_i = np.arange(0, nb_features + 1)
y_i = (points[-22:]) / 0.01 / 100

plt.figure(figsize = (16, 8))
plt.plot(points, marker = 'o')
plt.xlabel('Số features (k)')
plt.ylabel('Variance (%)')
plt.title('Đồ thị biểu diễn % phương sai tích lũy theo số features (k)')
plt.xlim([0, nb_features + 1])
plt.grid(axis = 'x')
for i in x_i:
    plt.text(i, y_i[i] + 1, y_i[i], ha = 'center', va = 'baseline') # tung độ của text cao hơn point 1 đơn vị

plt.show()
```

Kết quả hiển thị biểu đồ




Sau khi vẽ đồ thị Elbow phương sai, chúng ta quan sát điểm giảm đột ngột của phương sai (góc "gãy"). Quan sát từ biểu đồ có thể dễ dàng xác định giá trị K tại điểm gãy nằm ở khoảng [10,12.5]. Vậy K=11 là giá trị tối ưu cho số lượng thành phần chính trong PCA.

Ta tiến hành kiểm chứng lại bằng cách tính phương sai tích lũy theo các giá trị của K như sau:

```
## Kiểm chứng: Tính phương sai tích lũy theo giá trị của k
var = 0.0
for k in range(1, nb_features + 1):
    pca = PCA(k)
    pca.fit(data_cluster)

    newVar = pca.explained_variance_ratio_.sum() * 100
    print(' * k = %2d' %k, ': phương sai tích lũy ~ %.2f%%' %newVar,
          '--> tăng ~ %.2f%%' %(newVar - var))
    var = newVar
```

Kết quả kiểm chứng



* k = 1	: phương sai tích lũy ~ 30.75%	--> tăng ~ 30.75%
* k = 2	: phương sai tích lũy ~ 52.73%	--> tăng ~ 21.98%
* k = 3	: phương sai tích lũy ~ 62.72%	--> tăng ~ 9.99%
* k = 4	: phương sai tích lũy ~ 72.62%	--> tăng ~ 9.90%
* k = 5	: phương sai tích lũy ~ 79.72%	--> tăng ~ 7.10%
* k = 6	: phương sai tích lũy ~ 84.72%	--> tăng ~ 5.00%
* k = 7	: phương sai tích lũy ~ 88.59%	--> tăng ~ 3.87%
* k = 8	: phương sai tích lũy ~ 91.58%	--> tăng ~ 2.99%
* k = 9	: phương sai tích lũy ~ 93.74%	--> tăng ~ 2.16%
* k = 10	: phương sai tích lũy ~ 95.55%	--> tăng ~ 1.81%
* k = 11	: phương sai tích lũy ~ 96.97%	--> tăng ~ 1.43%
* k = 12	: phương sai tích lũy ~ 97.62%	--> tăng ~ 0.65%
* k = 13	: phương sai tích lũy ~ 98.19%	--> tăng ~ 0.57%
* k = 14	: phương sai tích lũy ~ 98.62%	--> tăng ~ 0.43%
* k = 15	: phương sai tích lũy ~ 98.98%	--> tăng ~ 0.36%
* k = 16	: phương sai tích lũy ~ 99.26%	--> tăng ~ 0.28%
* k = 17	: phương sai tích lũy ~ 99.54%	--> tăng ~ 0.28%
* k = 18	: phương sai tích lũy ~ 99.69%	--> tăng ~ 0.16%
* k = 19	: phương sai tích lũy ~ 99.83%	--> tăng ~ 0.14%
* k = 20	: phương sai tích lũy ~ 99.94%	--> tăng ~ 0.11%

Dựa trên đồ thị thì có thể chọn k trong khoảng [10 , 12.5] => Chọn k bằng 11

➔ Elbow phương sai không chỉ là một công cụ mạnh mẽ trong việc xác định số lượng cụm, mà còn là một hướng tiếp cận hữu ích khi chọn số lượng thành phần chính trong PCA. Việc kết hợp hai phương pháp này giúp tối ưu hóa tiền xử lý dữ liệu và chuẩn bị cho quá trình phân cụm hiệu quả.

d. Giảm chiều dữ liệu

Trong quá trình tiền xử lý trước khi thực hiện phân cụm, thường xuyên chúng ta đối mặt với vấn đề về chiều dữ liệu khi số chiều tăng lên do quá trình mã hóa và chuẩn hóa dữ liệu. Điều này có thể dẫn đến độ phức tạp cao cho mô hình và ảnh hưởng đến hiệu suất của thuật toán phân cụm. Trong bối cảnh này, phân tích thành phần chính (PCA) trở thành một công cụ mạnh mẽ để giảm chiều dữ liệu và giữ lại thông tin quan trọng.

PCA là một phương pháp thống kê được sử dụng để giảm chiều dữ liệu bằng cách chuyển đổi dữ liệu từ không gian biến ban đầu sang không gian mới có số chiều ít hơn (thường là 2 hoặc 3 chiều) sao cho thông tin chủ yếu của dữ liệu được bảo toàn. Nói một cách đơn giản, PCA giúp chúng ta tìm ra các hướng (thành phần chính) trong dữ liệu để tối ưu hóa việc thể hiện sự biến thiên từ dữ liệu đó.

Ý nghĩa của PCA trong phân tích dữ liệu:

- **Giảm độ phức tạp cho mô hình:** PCA không chỉ giảm số chiều dữ liệu mà còn giúp giảm độ phức tạp của mô hình. Bằng cách chọn các thành phần chính quan trọng nhất, chúng ta tạo ra một biểu diễn mới của dữ liệu với ít chiều hơn mà vẫn giữ lại sự đa dạng và tính đại diện.
- **Bảo toàn thông tin quan trọng:** PCA không chỉ giúp giảm chiều dữ liệu mà còn đảm bảo rằng sự biến động lớn nhất của dữ liệu được bảo toàn. Các thành

phần chính được chọn giữ lại phần lớn thông tin quan trọng và sự biểu diễn tốt nhất về tính chất của dữ liệu.

- **Tăng hiệu suất phân cụm:** Việc giảm chiều dữ liệu bằng PCA cũng làm tăng hiệu suất của thuật toán phân cụm. Dữ liệu có chiều thấp hơn giúp thuật toán hoạt động nhanh chóng hơn và giảm nguy cơ overfitting.
- **Khám phá liên kết tiềm ẩn:** Trong không gian mới, PCA tạo điều kiện để các liên kết tiềm ẩn của dữ liệu có thể được khám phá một cách hiệu quả, giúp hiểu rõ hơn về mối quan hệ giữa các biến.

Ta tiến hành thực hiện PCA với k tối ưu là 11 đã tìm được nhờ phương pháp Elbow phương sai ở trên:

```
X = data_cluster.copy()
## Thực hiện PCA với k = 11
k = 11
pca = PCA(k)
pca.fit(X)
```

PCA
PCA(n_components=11)

Để tối ưu hóa biểu diễn, làm nổi bật mối quan hệ và giảm độ phức tạp, giúp người phân tích hiểu rõ hơn về cấu trúc và tính chất của dữ liệu, ta tiến hành chuyển chiều dữ liệu vào không gian mới trong PCA:

```
## Chuyển dữ liệu vào không gian mới (Transform data)
PC_name = ['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5', 'PC 6', 'PC 7', 'PC 8', 'PC 9', 'PC 10', 'PC 11']
P = pca.transform(X)
data_cluster_pca = pd.DataFrame(data = P, columns = PC_name)
```

Kết quả phân tích thành phần chính PCA với k = 11

data_cluster_pca											
	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8	PC 9	PC 10	PC 11
0	-0.422211	0.573815	0.215426	0.193616	0.075993	-0.355139	0.181793	0.371527	-0.177672	0.799084	0.141124
1	-0.947152	0.845036	-0.200390	-0.552328	-0.005800	-0.011789	0.228975	-0.203221	-0.060624	-0.116887	-0.163672
2	-0.947152	0.845036	-0.200390	-0.552328	-0.005800	-0.011789	0.228975	-0.203221	-0.060624	-0.116887	-0.163672
3	0.379916	1.387983	0.507518	0.417890	-0.175932	-0.214634	0.416574	0.593048	0.333668	0.716902	-0.044609
4	0.379916	1.387983	0.507518	0.417890	-0.175932	-0.214634	0.416574	0.593048	0.333668	0.716902	-0.044609
...
369	1.264271	-0.534413	-0.427407	-0.078800	-0.177268	0.488772	-0.134592	0.148723	-0.083808	0.016248	0.132989
370	1.266631	-0.521055	-0.422370	-0.082332	-0.172421	0.474868	-0.130973	0.147390	-0.087756	0.011790	0.134421
371	1.264271	-0.534413	-0.427407	-0.078800	-0.177268	0.488772	-0.134592	0.148723	-0.083808	0.016248	0.132989
372	1.264271	-0.534413	-0.427407	-0.078800	-0.177268	0.488772	-0.134592	0.148723	-0.083808	0.016248	0.132989
373	1.264271	-0.534413	-0.427407	-0.078800	-0.177268	0.488772	-0.134592	0.148723	-0.083808	0.016248	0.132989

374 rows x 11 columns

➔ Trong bước tiền xử lý trước khi phân cụm, PCA là một công cụ hiệu quả để giảm chiều dữ liệu và giảm độ phức tạp của mô hình. Việc này không chỉ giữ lại thông tin quan trọng mà còn tăng hiệu suất và tính tổng quát của quá trình phân cụm. Sự kết hợp linh hoạt của PCA trong quy trình tiền xử lý là một chiến lược quan trọng đối với việc xử lý dữ liệu phức tạp và đa dạng.

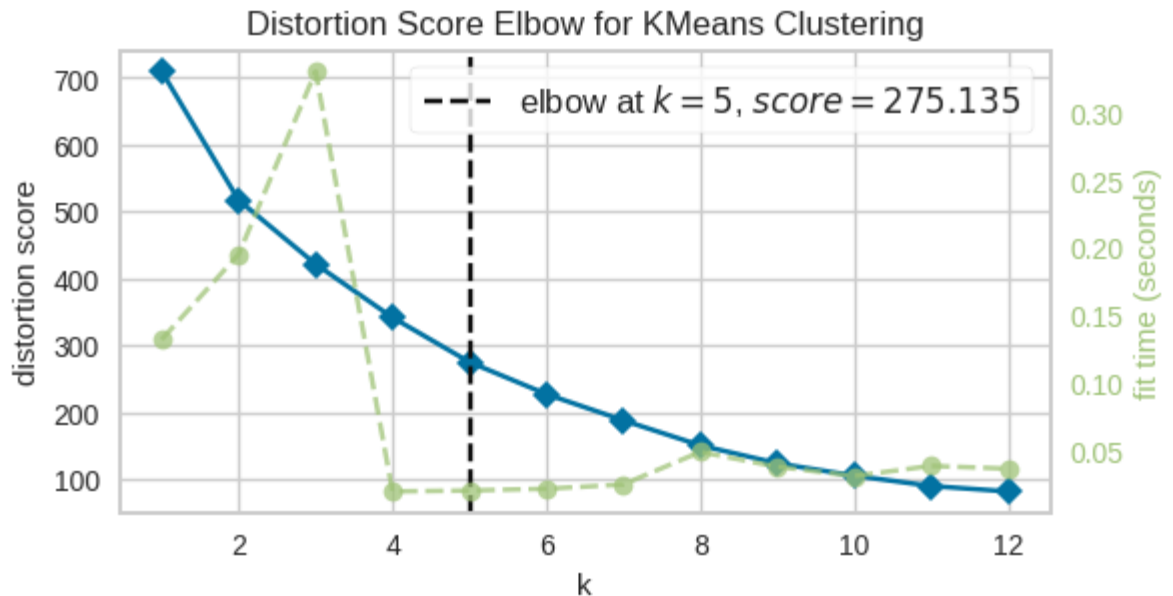
4. Phân cụm phân hoạch

Loại phân cụm này sẽ phân tập dữ liệu có n phần tử cho trước thành k tập con ($k \leq n$), mỗi tập con biểu diễn một cụm. Số các cụm được thiết lập là các được trung được lựa chọn trước, sao cho mỗi đối tượng thuộc duy nhất 1 cụm, các phần tử trong cụm có sự tương tự nhau. Mỗi cụm có ít nhất 1 phần tử.

a. Phương pháp Elbow

Phương pháp Elbow là một cách giúp ta lựa chọn được số lượng các cụm phù hợp dựa vào đồ thị trực quan hoá bằng cách nhìn vào sự suy giảm của hàm biến dạng và lựa chọn ra điểm khuỷu tay hay nói cách khác là “điểm gãy” (elbow point). Để thực hiện phương pháp Elbow trong Python ta sử dụng hàm `KelbowVisualizer` của thư viện `yellowbrick.cluster` để thực hiện phương pháp elbow để giúp chọn số lượng cụm tối ưu bằng cách lắp mô hình với một loạt các giá trị cho K (ở đây ta chạy K từ 1 → 12). Trong visualizer "elbow" sẽ được chú thích bằng một đường nét đứt thẳng đứng.

```
plt.figure(figsize = (6, 3))
Elbow_M = KelbowVisualizer(KMeans(), k=(1,13))
Elbow_M.fit(data_cluster_kmeans)
Elbow_M.show()
```



Điểm khuỷu tay (điểm gãy) là điểm mà ở đó tốc độ suy giảm của hàm biến dạng sẽ thay đổi nhiều nhất. Tức là kể từ sau vị trí này thì gia tăng thêm số lượng cụm cũng không giúp hàm biến dạng giảm đáng kể. Nếu thuật toán phân chia theo số lượng cụm tại vị trí này sẽ đạt được tính chất phân cụm một cách tổng quát nhất mà không gặp các hiện tượng vị khớp (overfitting). Trong hình trên thì ta thấy vị trí của điểm khuỷu tay chính là $k = 5$ vì khi số lượng cụm lớn hơn 5 thì tốc độ suy giảm của hàm biến dạng dường như không đáng kể so với trước đó => Chọn số cụm bằng 5.

b. Thực hiện phân cụm

Phương pháp K-means là một thuật toán phân cụm (clustering) phổ biến được sử dụng trong machine learning để chia dữ liệu thành các nhóm có ý nghĩa. Dưới đây là cách phương pháp K-means hoạt động:

1. **Chọn số lượng cụm (clusters):** Trước khi bắt đầu, cần xác định số lượng cụm muốn tạo. Điều này thường được xác định bằng phương pháp elbow $k=5$.
2. **Khởi tạo các điểm trọng tâm ban đầu:** Mỗi cụm được đại diện bởi một điểm trọng tâm. Các điểm trọng tâm ban đầu có thể được chọn ngẫu nhiên từ dữ liệu hoặc dựa trên phương pháp khác.
3. **Gán điểm dữ liệu vào cụm gần nhất:** Mỗi điểm dữ liệu được gán vào cụm có điểm trọng tâm gần nhất, thường dựa trên khoảng cách Euclidean giữa điểm dữ liệu và các điểm trọng tâm.
4. **Cập nhật điểm trọng tâm:** Sau khi mỗi điểm dữ liệu đã được gán vào một cụm, điểm trọng tâm của cụm được cập nhật bằng cách tính trung bình của tất cả các điểm dữ liệu trong cụm đó.

5. **Lặp lại quá trình:** Bước 3 và 4 được lặp lại cho đến khi sự thay đổi giữa các điểm trọng tâm của hai vòng lặp liên tiếp là nhỏ (hoặc thỏa mãn một điều kiện dừng khác).

Thuật toán này cố gắng tối ưu hóa tổng khoảng cách giữa mỗi điểm dữ liệu và điểm trọng tâm của cụm nó thuộc về. Kết quả cuối cùng là một tập hợp các cụm mà mỗi cụm chứa các điểm dữ liệu có sự tương đồng cao với điểm trọng tâm của nó.

```
## Thực hiện clustering bằng phương pháp KMeans
k = 5
model_kmeans = KMeans(n_clusters = k)
model_kmeans.fit_predict(data_cluster_kmeans)

## Các trọng tâm
print(f'*** {k} trọng tâm:')
print(model_kmeans.cluster_centers_)

*** 5 trọng tâm:
[[-4.98763213e-01 -1.36301557e-01 -4.95830859e-01  5.71500700e-01
  1.84125337e-01 -8.48453750e-02  1.22461827e-01 -4.82792544e-02
  3.83810552e-02 -2.37216761e-02  6.16992010e-03]
 [ 1.16412584e+00  4.81435786e-03 -4.48193153e-01 -2.89309006e-01
 -1.45173540e-01 -1.92669498e-03  3.53999317e-02  2.45480302e-02
 -4.80006604e-02  4.03009048e-03  6.19387535e-02]
 [-1.00037291e+00  5.47185053e-01 -4.18180908e-03 -4.75223926e-01
 -5.76967093e-02  2.25444095e-01  6.42120868e-02  2.76998583e-02
 -2.05038832e-02 -7.56442625e-02  2.36498325e-02]
 [-2.96198476e-01 -8.53515397e-01  2.47569373e-01 -1.49676168e-01
 -6.28573431e-02 -1.69101727e-01 -1.22082066e-01 -4.55941932e-03
  5.38889575e-02  2.58767061e-02 -9.36971002e-03]
 [ 4.53660223e-01  4.85902767e-01  4.91892821e-01  3.05407808e-01
  8.27451315e-02  5.40210990e-02 -5.14449459e-02 -6.12555629e-04
 -2.47893132e-02  4.60733398e-02 -6.27179699e-02]]
```

Kết quả sau khi phân cụm:

```
## Kết quả gom cụm
data_cluster_kmeans['cluster'] = model_kmeans.labels_

print('Số phần tử của mỗi cluster:')
print(data_cluster_kmeans['cluster'].value_counts())

Số phần tử của mỗi cluster:
4      89
3      83
1      71
2      66
0      65
Name: cluster, dtype: int64
```

5. Phân cụm phân cấp

Hierarchical Agglomerative Clustering (HAC) là một phương pháp phân cụm phân cấp được sử dụng để tổ chức dữ liệu thành một cấu trúc cây phân cấp, hay còn gọi

là dendrogram. Phương pháp này bắt đầu với việc coi mỗi điểm dữ liệu là một cụm đơn lẻ và sau đó liên tục gộp các cụm gần nhau để tạo thành các cụm lớn hơn. Dưới đây là quy trình hoạt động cơ bản của HAC:

1. **Bắt đầu với các cụm đơn lẻ:** Mỗi điểm dữ liệu được coi là một cụm đơn lẻ ban đầu.
2. **Tính ma trận khoảng cách:** Tính toán ma trận khoảng cách giữa tất cả các cụm. Các phương pháp thường dùng để đo khoảng cách bao gồm khoảng cách Euclidean, khoảng cách Manhattan, hay khoảng cách tương quan.
3. **Gộp hai cụm gần nhau:** Chọn cặp cụm có khoảng cách nhỏ nhất và gộp chúng thành một cụm mới. Điều này tạo ra một cấp độ mới trong cây phân cấp.
4. **Cập nhật ma trận khoảng cách:** Tính toán lại ma trận khoảng cách giữa cụm mới và các cụm còn lại.
5. **Lặp lại quá trình:** Lặp lại bước 3 và 4 cho đến khi chỉ còn lại một cụm duy nhất, hoặc số lượng cụm mong muốn.

Kết quả cuối cùng của HAC là một cây phân cấp, hay dendrogram, mô tả sự tương đồng giữa các cụm. Khi cây phân cấp được tạo ra, có thể chọn một cấp độ cụm cụ thể để chia dữ liệu thành các nhóm.

Có hai loại chính của HAC: phân cụm từ trên xuống (top-down), được gọi là "divisive", và phân cụm từ dưới lên (bottom-up), được gọi là "agglomerative". Phương pháp agglomerative thường được ưa chuộng vì tính đơn giản và khả năng mở rộng tốt cho dữ liệu lớn.

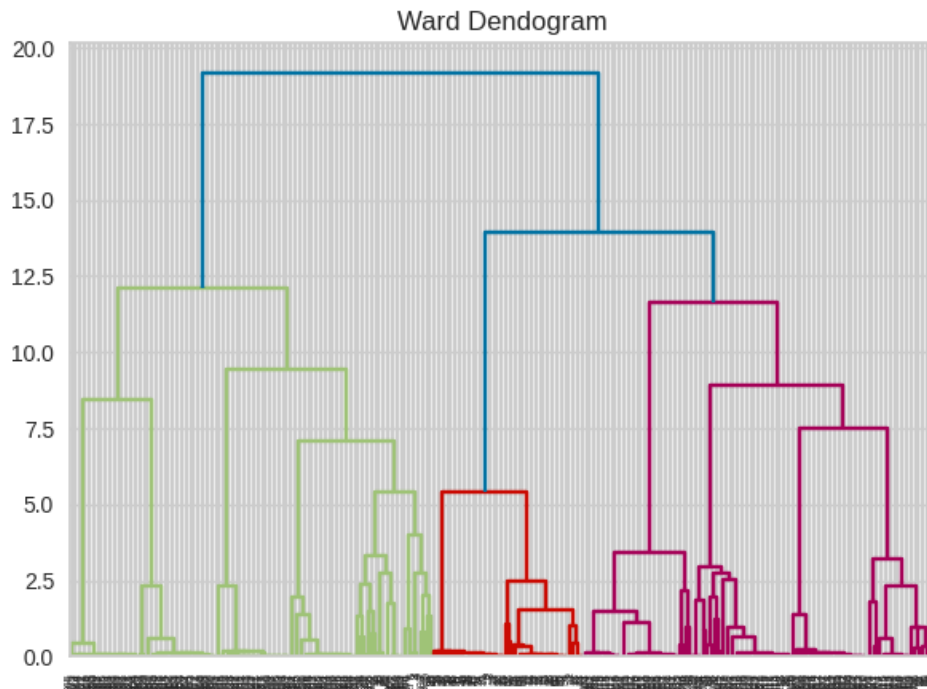
Thuật toán phân cấp mà nhóm sử dụng sẽ là loại agglomerative và 2 phương pháp phân cụm ở đây là Ward và Complete.

a. Sử dụng HAC với ward

Đầu tiên, ta xây dựng Dendrogram để chọn số cụm tốt nhất

```
## Xây dựng Dendrogram
plt.figure(figsize = (7, 5))
plt.title("Ward Dendrogram")
dg = hierarchy.dendrogram(hierarchy.linkage(data_cluster_hac_ward, method = 'ward'))

plt.axhline(y = 200, color = 'r', linestyle = '--')
plt.show()
```



Tiếp theo, ta xây dựng mô hình HAC với phương pháp Ward với số cụm là 3

```
## Xây dựng mô hình HAC
k = 3
model_hac_ward = AgglomerativeClustering(n_clusters = k, metric = 'euclidean', linkage = 'ward')
model_hac_ward.fit_predict(data_cluster_hac_ward)
```

Thu được kết quả phân cụm như sau:

```
## Kết quả gom cụm khách hàng
data_cluster_hac_ward['cluster'] = model_hac_ward.labels_

print('Số phần tử của mỗi cluster:')
print(data_cluster_hac_ward['cluster'].value_counts())
```

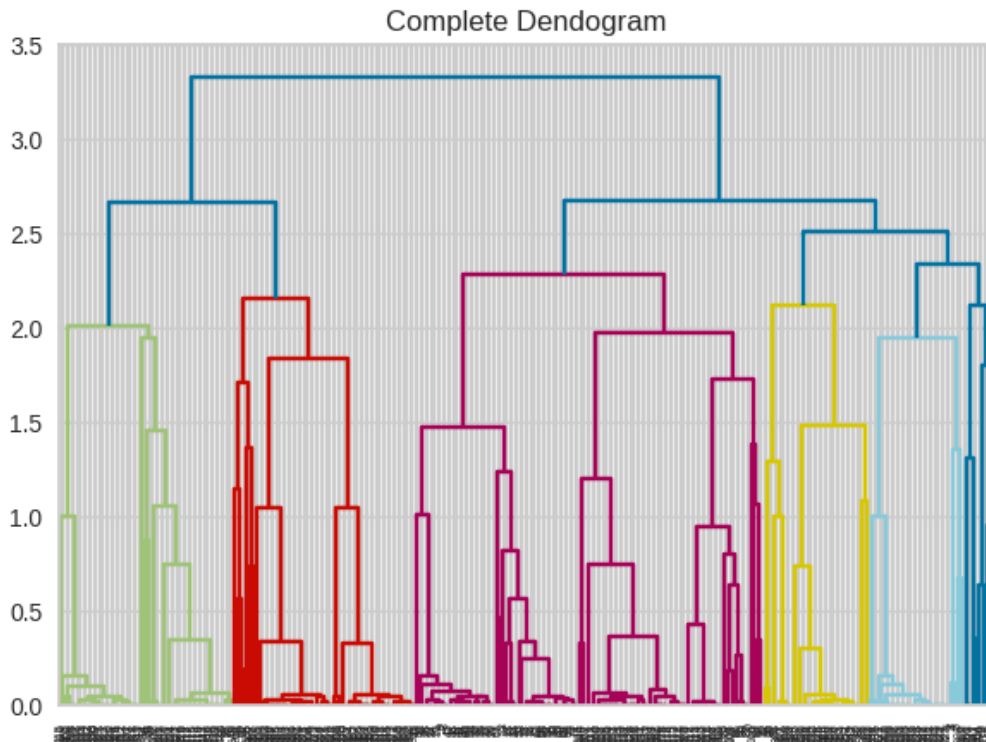
```
Số phần tử của mỗi cluster:
0    157
1    152
2     65
Name: cluster, dtype: int64
```

b. Sử dụng HAC với complete

Đầu tiên, xây dựng Dendrogram để chọn số cụm tốt nhất

```
## Xây dựng Dendrogram
plt.figure(figsize = (7, 5))
plt.title("Complete Dendrogram")
dg = hierarchy.dendrogram(hierarchy.linkage(data_cluster_hac_comp, method = 'complete'))

plt.axhline(y = 200, color = 'r', linestyle = '--')
plt.show()
```



Tiếp theo, ta xây dựng mô hình HAC phương pháp Complete với số cụm là 6

```
## Xây dựng mô hình HAC
k = 6
model_hac_comp = AgglomerativeClustering(n_clusters = k, metric = 'euclidean', linkage = 'complete')
model_hac_comp.fit_predict(data_cluster_hac_comp)
```

Thu được kết quả phân cụm như sau:

```
## Kết quả gom cụm khách hàng
data_cluster_hac_comp['cluster'] = model_hac_comp.labels_

print('Số phần tử của mỗi cluster:')
print(data_cluster_hac_comp['cluster'].value_counts())

Số phần tử của mỗi cluster:
0    139
1     73
3     70
4     43
5     38
2     11
Name: cluster, dtype: int64
```

6. Biểu diễn kết quả phân cụm

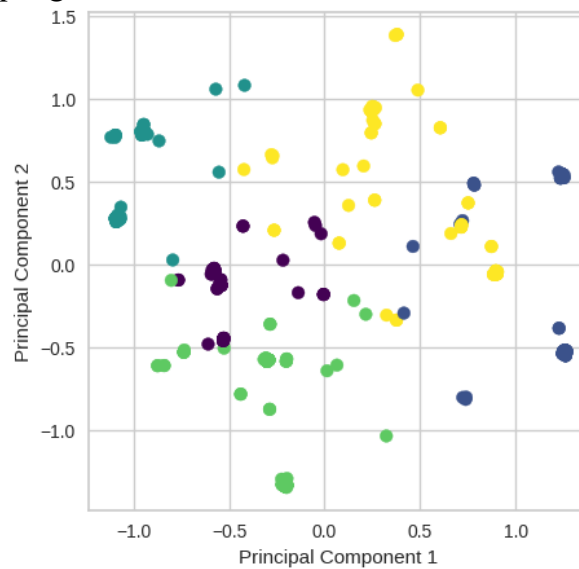
Để biểu diễn kết quả phân cụm của các phương pháp, nhóm sử dụng PCA để giảm chiều dữ liệu về 2 chiều, nhằm thuận tiện cho việc quan sát và biểu diễn bằng biểu đồ scatter plot.

```
# Giảm chiều dữ liệu về 2 chiều bằng PCA
pca = PCA(n_components=2)
data_2d = pca.fit_transform(data_cluster_pca)
# Tạo DataFrame mới sau khi giảm chiều
df_2d = pd.DataFrame(data_2d, columns=['PC1', 'PC2'])

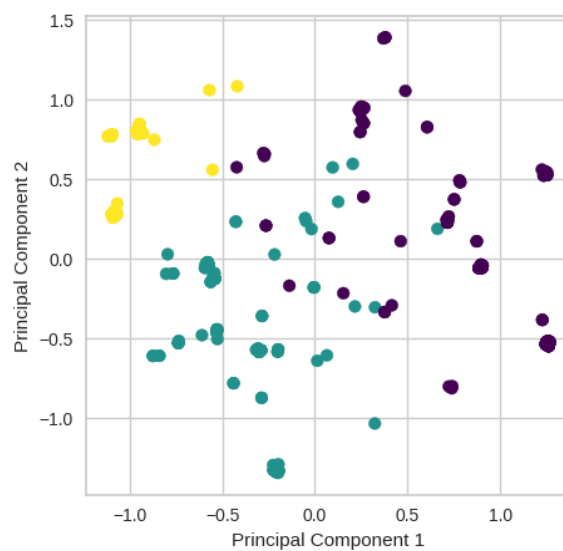
# Thêm cột cụm vào DataFrame
df_2d['cluster'] = data_cluster_kmeans['cluster'] # labels là kết quả của quá trình phân cụm

# Biểu đồ Scatter Plot
plt.figure(figsize = (5, 5))
plt.scatter(df_2d['PC1'], df_2d['PC2'], c=df_2d['cluster'], cmap='viridis')
plt.title('Phân cụm dữ liệu trong không gian 2D (PCA)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

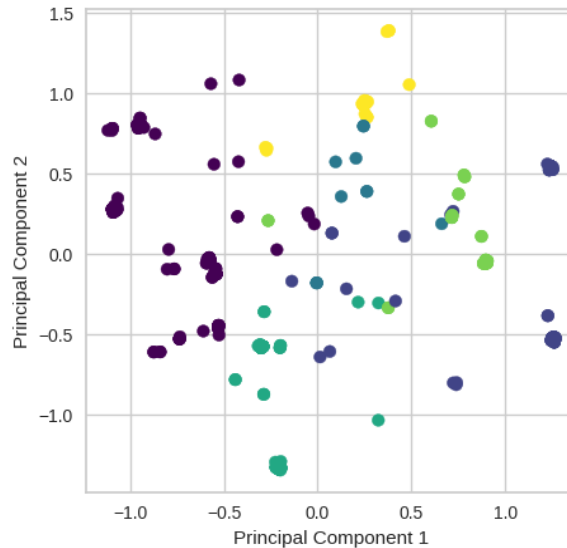
Biểu diễn kết quả gom cụm



Biểu đồ 1: K-Means



Biểu đồ 2: Ward



Biểu đồ 3: Complete

7. Đánh giá mô hình

Chỉ số Silhouette là một phương pháp đánh giá mô hình phân cụm được sử dụng để đo lường độ tách biệt (separation) và đồng nhất (cohesion) giữa các cụm. Nó cung cấp một con số từ -1 đến 1, trong đó:

- Giá trị gần 1 cho biết các điểm dữ liệu trong cùng một cụm gần nhau và tách biệt tốt với các cụm khác.
- Giá trị gần 0 cho biết các cụm có sự trộn lẫn, hay các điểm dữ liệu có thể nằm giữa hai cụm.
- Giá trị gần -1 cho biết các điểm dữ liệu trong cùng một cụm không tách biệt tốt hoặc được gán vào sai cụm.

Quy trình tính toán chỉ số Silhouette cho một phân cụm:

1. **Tính khoảng cách giữa các điểm dữ liệu trong cùng một cụm:** Đối với mỗi điểm dữ liệu trong cụm, tính khoảng cách trung bình của nó đến tất cả các điểm dữ liệu khác trong cùng một cụm.
2. **Tính khoảng cách trung bình đến cụm lân cận gần nhất:** Tính khoảng cách trung bình từ mỗi điểm dữ liệu trong cụm đó đến tất cả các điểm dữ liệu trong cụm lân cận gần nhất (cụm khác).
3. **Tính giá trị Silhouette cho từng điểm dữ liệu:** Sử dụng các giá trị tính được ở bước 1 và 2 để tính giá trị Silhouette cho mỗi điểm dữ liệu theo công thức:

$$\text{Silhouette}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Trong đó, $a(i)$ là khoảng cách trung bình từ điểm i đến các điểm cùng cụm và $b(i)$ là khoảng cách trung bình từ điểm i đến cụm lân cận gần nhất.

4. **Tính giá trị Silhouette trung bình cho toàn bộ phân cụm:** Lấy giá trị Silhouette cho tất cả các điểm dữ liệu trong cụm và tính giá trị trung bình.

5. **Đánh giá mô hình phân cụm:** Giá trị Silhouette càng gần 1 thì mô hình phân cụm càng tốt, vì nó chỉ ra rằng các cụm tách biệt tốt và các điểm dữ liệu trong cùng một cụm gần nhau.

Tính toán điểm Silhouette cho từng phương pháp

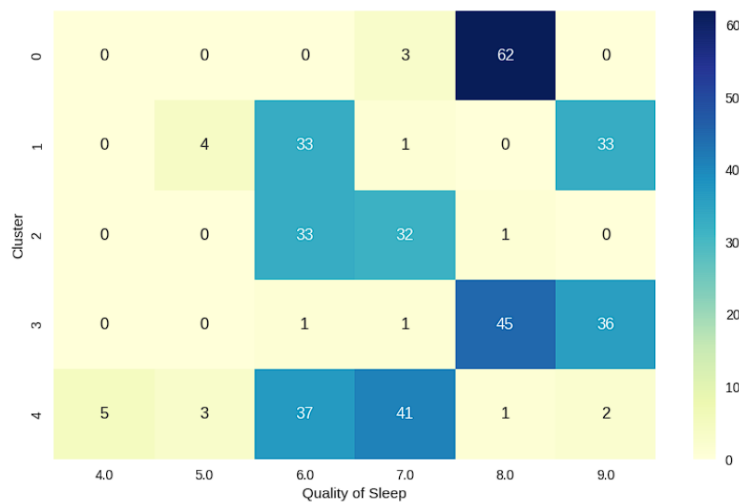
```
# Tính toán Silhouette Score để đánh giá hiệu năng của việc phân cụm
score1 = silhouette_score(data_cluster_hac_ward, model_hac_ward.labels_, metric='euclidean')
# Silhouette score nằm gần 1 nghĩa là ví dụ đang được phân cụm chính xác, xa các cụm khác
print('Silhouette Score: %.3f' % score1)
```

- Phân cụm với K-means có điểm Silhouette: 0.558
 - Phân cụm với HAC bằng phương pháp Ward có điểm Silhouette: 0.396
 - Phân cụm với HAC bằng phương pháp Complete có điểm Silhouette: 0.545
- Kết quả cho thấy phương pháp K-means cho tỷ lệ chính xác cao hơn và có số điểm sát với thực tế hơn dù số cụm ($k = 5$) ít hơn so với HAC Complete ($k = 6$)

8. Quan sát các biến mục tiêu

Thực hiện quan sát mối quan hệ giữa các biến mục tiêu liên quan đến giấc ngủ và các cụm

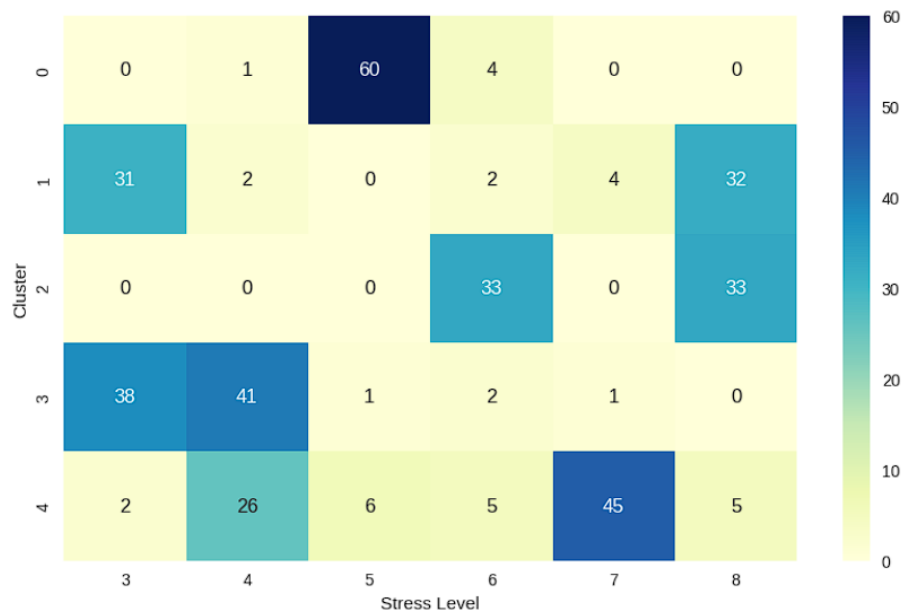
a. Quality of Sleep



Đối với biến Quality of sleep, có thể thấy được rằng kết quả phân cụm khá tốt:

- Cụm 0 và cụm 3 là 2 cụm bao gồm những người có chất lượng giấc ngủ tốt. (từ 8 đến 9)
- Cụm 2 và cụm 4 là 2 cụm bao gồm những người có chất lượng giấc ngủ trung bình. (từ 6 đến 7)
- Cụm 1 có số người có chất lượng giấc ngủ được phân bố đều ở mức trung bình và tốt.

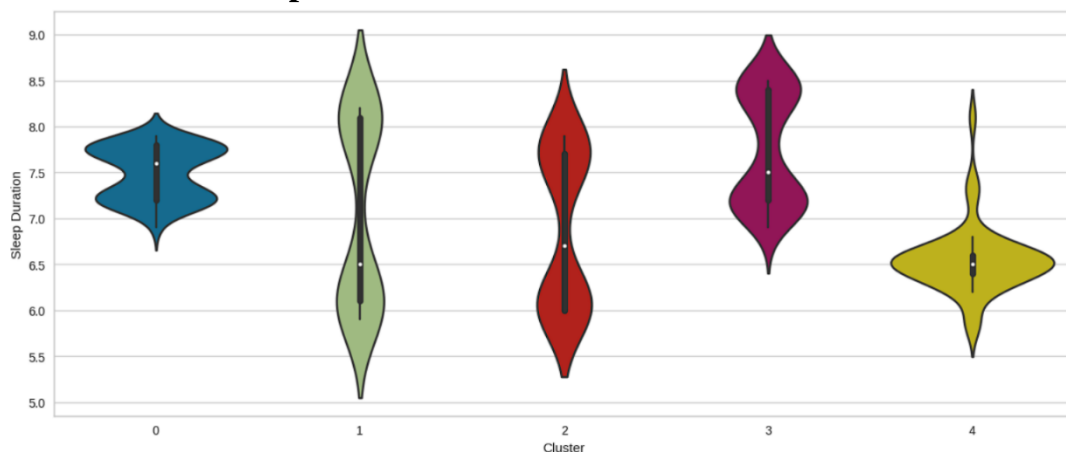
b. Stress Level



Đối với biến Stress Level, có thể thấy được rằng kết quả phân cụm chưa rõ ràng:

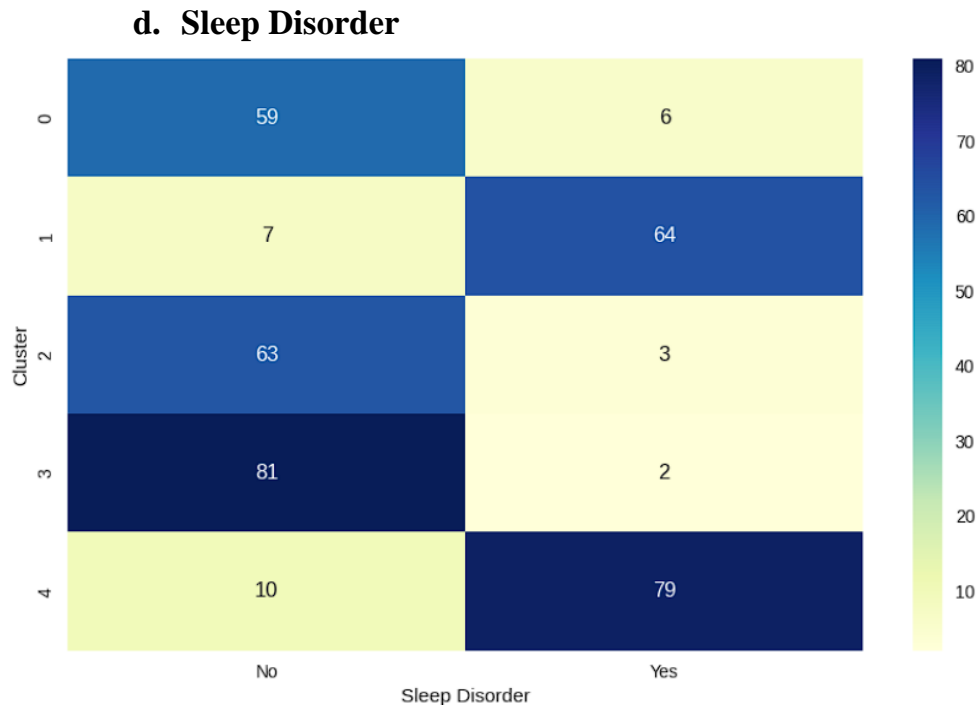
- Cụm 0 bao gồm những người có mức độ áp lực trung bình (từ 5 đến 6) và cụm 3 bao gồm những người có mức độ áp lực thấp. (từ 3 đến 4)
- Cụm 1 phân bố đồng đều ở 2 mức độ áp lực thấp và cao. Và đối với cụm 2 là trung bình và cao.
- Cụm 4 phân bố khá đồng đều ở cả 3 mức độ áp lực.

c. Sleep Duration



Đối với biến Sleep Duration, có thể thấy được rằng kết quả phân cụm khá tốt:

- Cụm 0 và cụm 3 là những cụm có số giờ ngủ trung bình vào khoảng 7.5 tiếng. Cụm 0 bao gồm những người có số giờ ngủ vào khoảng 6.5 đến 8 tiếng. Cụm 3 bao gồm những người có số giờ ngủ vào khoảng từ 6.5 đến 9 tiếng.
- Cụm 1, cụm 2 và cụm 4 là những cụm có số giờ ngủ trung bình vào khoảng 6.5 tiếng. Cụm 1 bao gồm những người có số giờ ngủ từ 5 đến 9 tiếng, cụm 2 bao gồm những người từ 5.5 đến 8 tiếng, cụm 4 bao gồm những người có số giờ ngủ từ 5.5 đến 8 tiếng.



Đối với biến Sleep Disorder, có thể thấy được rằng kết quả phân cụm khá tốt:

- Cụm 0, cụm 2 và cụm 3 là những cụm bao gồm những người không mắc triệu chứng khi ngủ.
- Cụm 1 và cụm 4 là những cụm bao gồm những người có mắc triệu chứng khi ngủ.

Kết luận và đánh giá: Dựa vào kết quả phân cụm của 4 biến mục tiêu là Quality of Sleep, Stress Level, Sleep Duration và Sleep Disorder, ta có thể kết luận được rằng:

- Cụm 0 và cụm 3 là những người có chất lượng giấc ngủ rất tốt, mức độ căng thẳng thấp, thời gian ngủ hợp lý và không mắc triệu chứng khi ngủ
- Cụm 1 và cụm 4 là những người có chất lượng giấc ngủ trung bình, mức độ căng thẳng cao, thời gian ngủ khá thấp và có mắc triệu chứng khi ngủ
- Cụm 2 là những người có chất lượng giấc ngủ trung bình, mức độ căng thẳng khá cao, thời gian ngủ khá thấp nhưng không mắc triệu chứng khi ngủ

Với kết luận trên, ta có thể thấy được ý nghĩa của việc phân cụm đối với bộ dữ liệu. Có thể thấy được sự tương đồng giữa các tính chất và dễ dàng tập trung phân tích chi tiết cho từng nhóm, điều này có thể tiết kiệm được thời gian trong việc xử lý và tìm hiểu dữ liệu. Từ đó có thể điều chỉnh và điều chỉnh các chỉ số hay thói quen để cải thiện chất lượng giấc ngủ và sức khỏe.

Mức độ căng thẳng thấp và thời gian ngủ hợp lý sẽ ảnh hưởng tốt tới chất lượng giấc ngủ và không mắc các triệu chứng khi ngủ. Còn nếu có mức độ căng thẳng cao và thời gian ngủ thấp sẽ ảnh hưởng xấu đến chất lượng giấc ngủ và có nguy cơ mắc các triệu chứng khi ngủ.

II. Phân lớp

1. Định nghĩa phân lớp

Phân lớp (classification) là một phương pháp trong lĩnh vực học máy (machine learning) và khai phá dữ liệu (data mining) nhằm dự đoán lớp hoặc nhãn của một đối tượng dựa trên các đặc trưng hoặc thuộc tính của nó. Mục tiêu của phân lớp là xây dựng một mô hình có khả năng tự động học từ dữ liệu đào tạo (training data) để phân loại các đối tượng mới vào các lớp đã biết trước. Phân lớp được tiến hành trong 2 bước là xây dựng mô hình xác định tập các lớp dữ liệu và sử dụng mô hình phân lớp trong bước 1 để kiểm tra và đánh giá

2. Mục đích phân lớp

Mục đích của việc phân lớp cho bộ dữ liệu là xây dựng mô hình phân lớp để dự đoán một người có mắc các chứng rối loạn giấc ngủ hay không và đánh giá tính thích hợp của bộ dữ liệu vào việc ứng dụng xây dựng mô hình dự đoán. Nhóm sẽ tiến hành xây dựng mô hình phân lớp cho biến Sleep Disorder. Để dự đoán xem một người trưởng thành có mắc các triệu chứng khi ngủ không.

3. Tiền xử lý trước phân lớp

a. Loại bỏ các đặc trưng

Hai biến có tương quan cao, ảnh hưởng mạnh với nhau: Sự ảnh hưởng mạnh giữa hai biến có thể tạo ra hiện tượng cộng tuyến trong mô hình phân loại, khiến cho chúng cung cấp thông tin tương đồng cho mô hình. Điều này có thể gây khó khăn trong việc phân biệt ảnh hưởng của từng biến đối với biến phụ thuộc. Mô hình với nhiều biến độc lập có tương quan cao sẽ trở nên phức tạp và khó giải thích, đồng thời có độ chính xác thấp và khả năng dự đoán kém. Nhằm khắc phục vấn đề này, quyết định loại bỏ một trong hai biến có ảnh hưởng mạnh với nhau được coi là giải pháp đơn giản và hiệu quả nhất. Việc này giúp giữ lại thông tin quan trọng từ cả hai biến, đồng thời loại bỏ sự trùng lặp thông tin giữa chúng.

- **Biến Stress Level và biến Quality of Sleep:** Cả hai biến Stress Level và Quality of Sleep đều được đánh giá theo yếu tố chủ quan. Tuy nhiên, theo nghiên cứu thực tế, việc đánh giá mức độ áp lực thường dễ thực hiện hơn so với việc đánh giá chất lượng giấc ngủ. Không những vậy biến Stress Level còn liên quan đến các biến khác như Heart Rate, Blood Pressure. Vì vậy, nhóm quyết định giữ lại biến Stress Level trong mô hình của nhóm, trong khi loại bỏ biến Quality of Sleep.
- **Biến Physical Activity Level và biến Daily Steps:** Biến Daily Steps không phản ánh đầy đủ về hoạt động hàng ngày của một người so với biến Physical Activity Level. Do sự tương quan mạnh giữa hai biến này, nhóm quyết định giữ lại biến Physical Activity Level và loại bỏ biến Daily Steps.

Biến ít quan trọng, ít đóng góp cho mô hình: Nếu một biến không có đóng góp lớn vào khả năng dự đoán của mô hình, việc giữ lại nó có thể tạo ra nhiễu và chỉ làm gia tăng độ phức tạp của dữ liệu mà không đem lại giá trị thêm vào mô hình. Loại

bỏ các biến ít quan trọng không chỉ giúp giảm chiều của dữ liệu mà còn nâng cao khả năng tập trung của mô hình vào những đặc trưng quan trọng hơn, từ đó cải thiện hiệu suất và khả năng giải thích của mô hình. Nên nhóm quyết định loại bỏ biến Heart Rate vì đây là biến chứa các giá trị bình thường đối với người trưởng thành và không tương quan mạnh với biến phụ thuộc.

Biến không ảnh hưởng đặc biệt đến biến phụ thuộc: Nếu có các biến không có ảnh hưởng đặc biệt đến biến phụ thuộc, giữ lại chúng có thể chỉ tạo ra nhiễu và làm mất thời gian tính toán. Loại bỏ những biến này có thể giúp mô hình tập trung vào các biến quan trọng hơn và làm cho quá trình đào tạo và dự đoán nhanh chóng và hiệu quả hơn.

- Từ kết quả kiểm định ở trên, biến Physical Activity Level khi kiểm tra với biến Sleep Disorder có hệ số kiểm định là 0 (Kruskal-Wallis) nghĩa là không tương quan. Với hệ số kiểm định thấp như vậy, nhóm quyết định loại bỏ biến này khỏi mô hình.
- Khi quan sát dữ liệu ta thấy được rằng biến giới tính không phản ánh được một người có bị mắc các triệu chứng khi ngủ hay không, cụ thể là nam dưới 27 tuổi lại nhiều hơn so với nữ và nữ trên 50 tuổi lại nhiều hơn với nam, mà đa phần những người mắc các triệu chứng khi ngủ lại xuất hiện nhiều ở độ tuổi trên 50 nên ta thấy dữ liệu có phần lệch hơn về phía nữ. Nên nhóm quyết định loại bỏ biến Gender ra khỏi mô hình phân lớp

b. Xử lý cho các mô hình không cần chuẩn hóa

Đối với mô hình Decision Tree: mô hình hoạt động dựa trên việc phân chia dữ liệu dựa trên giá trị của các biến độc lập. Khi chuẩn hóa dữ liệu, các giá trị của các biến độc lập sẽ được chuyển về cùng một thang đo, làm mất đi sự khác biệt giữa chúng. Điều này có thể gây ảnh hưởng đến khả năng hiệu quả của mô hình Decision Tree trong việc phân chia dữ liệu. Mô hình Decision Tree sử dụng các hàm như entropy hoặc information gain để đánh giá mức độ phân chia của dữ liệu. Đặc điểm quan trọng là các hàm này không phụ thuộc vào thang đo của các biến độc lập. Do đó, mô hình không cần chuẩn hóa dữ liệu trước khi đưa vào vì việc này không ảnh hưởng đến hiệu quả của mô hình Decision Tree.

➔ Vì vậy, mô hình này, nhóm chỉ cần thực hiện quá trình mã hóa dữ liệu trước khi tiến hành quá trình phân lớp.

```
# Stress Level
data_classify_no_standard['Stress Level'] = data_classify_no_standard['Stress Level'].astype('int64')
# BMI Category
data_classify_no_standard['BMI Category'] = data_classify_no_standard['BMI Category'].replace({'Normal Weight': 0})
data_classify_no_standard['BMI Category'] = data_classify_no_standard['BMI Category'].replace({'Overweight': 1})
data_classify_no_standard['BMI Category'] = data_classify_no_standard['BMI Category'].replace({'Obese': 2})
# Blood Pressure
data_classify_no_standard['Blood Pressure'] = data_classify_no_standard['Blood Pressure'].replace({'HA Bình thường': 0})
data_classify_no_standard['Blood Pressure'] = data_classify_no_standard['Blood Pressure'].replace({'HA Cao': 1})
data_classify_no_standard['Blood Pressure'] = data_classify_no_standard['Blood Pressure'].replace({'Tăng HA giai đoạn 1': 2})
data_classify_no_standard['Blood Pressure'] = data_classify_no_standard['Blood Pressure'].replace({'Tăng HA giai đoạn 2': 3})
# Occupation
# Sleep Disorder
df_encoded = pd.get_dummies(data_classify_no_standard, columns=['Occupation', 'Sleep Disorder'], prefix=['Occupation', 'Sleep Disorder'], drop_first=True)
data_classify_no_standard = pd.concat([data_classify_no_standard, df_encoded], axis=1)

# Xóa các cột ban đầu và bỏ bớt các cột nhị phân
selected_columns = ['Occupation', 'Sleep Disorder']
data_classify_no_standard.drop(selected_columns, axis=1, inplace=True)
# Xóa các cột trùng
# Xóa các cột trùng nhau
data_classify_no_standard = data_classify_no_standard.T.drop_duplicates().T
```

Kết quả sau khi xử lý:

	Age	Sleep Duration	Stress Level	BMI Category	Blood Pressure	Occupation_Doctor	Occupation_Engineer	Occupation_Lawyer	Occupation_Other
0	27.0	6.1	6.0	1.0	2.0	0.0	0.0	0.0	0.0
1	28.0	6.2	8.0	0.0	1.0	1.0	0.0	0.0	0.0
2	28.0	6.2	8.0	0.0	1.0	1.0	0.0	0.0	0.0
3	28.0	5.9	8.0	2.0	3.0	0.0	0.0	0.0	0.0
4	28.0	5.9	8.0	2.0	3.0	0.0	0.0	0.0	0.0
...
369	59.0	8.1	3.0	1.0	3.0	0.0	0.0	0.0	0.0
370	59.0	8.0	3.0	1.0	3.0	0.0	0.0	0.0	0.0
371	59.0	8.1	3.0	1.0	3.0	0.0	0.0	0.0	0.0
372	59.0	8.1	3.0	1.0	3.0	0.0	0.0	0.0	0.0
373	59.0	8.1	3.0	1.0	3.0	0.0	0.0	0.0	0.0

c. Xử lý cho các mô hình cần chuẩn hóa

Đối với mô hình K-NN: Thuật toán K-NN hoạt động dựa trên việc đo khoảng cách giữa các điểm dữ liệu, thường sử dụng các hàm toán học như khoảng cách Euclid, khoảng cách Manhattan, hoặc khoảng cách Minkowski. Khi các thuộc tính dữ liệu có đơn vị đo khác nhau, thì khoảng cách giữa các điểm dữ liệu sẽ bị ảnh hưởng bởi đơn vị đo của các thuộc tính đó. Chuẩn hóa dữ liệu giúp đưa các thuộc tính về cùng một khoảng giá trị. Điều này đảm bảo rằng khoảng cách giữa các điểm dữ liệu không bị ảnh hưởng quá mức bởi đơn vị đo khác nhau của các thuộc tính.

Đối với mô hình SVM: Mô hình SVM tìm kiếm một siêu mặt phân cách tối ưu để phân chia các điểm dữ liệu thuộc các lớp khác nhau. Siêu mặt phân cách này được xác định bởi một tập hợp các tham số mà các tham số này được tính toán dựa trên các giá trị của các thuộc tính dữ liệu. Vì vậy, nếu các thuộc tính dữ liệu có phạm vi giá trị lớn, các tham số của siêu mặt phân cách có thể bị ảnh hưởng nhiều, dẫn đến hiệu suất giảm. Chuẩn hóa dữ liệu giúp đưa các giá trị về cùng một phạm vi, giảm thiểu ảnh hưởng này.

→ Do đó, nhóm quyết định thực hiện chuẩn hóa dữ liệu sau khi đã tiến hành quá trình mã hóa, như được thực hiện trong bước xử lý cho cả mô hình Decision Tree.

```
# Chuẩn hóa dữ liệu
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data_classify_no_standard)

# Tạo DataFrame mới từ dữ liệu đã chuẩn hóa
data_classify_standard = pd.DataFrame(scaled_data, columns=data_classify_no_standard.columns)
```

Kết quả sau khi xử lý:

	Age	Sleep Duration	Stress Level	BMI Category	Blood Pressure	Occupation_Doctor	Occupation_Engineer	Occupation_Lawyer
0	0.00000	0.111111	0.6	0.5	0.666667	0.0	0.0	0.0
1	0.03125	0.148148	1.0	0.0	0.333333	1.0	0.0	0.0
2	0.03125	0.148148	1.0	0.0	0.333333	1.0	0.0	0.0
3	0.03125	0.037037	1.0	1.0	1.000000	0.0	0.0	0.0
4	0.03125	0.037037	1.0	1.0	1.000000	0.0	0.0	0.0
...
369	1.00000	0.851852	0.0	0.5	1.000000	0.0	0.0	0.0
370	1.00000	0.814815	0.0	0.5	1.000000	0.0	0.0	0.0
371	1.00000	0.851852	0.0	0.5	1.000000	0.0	0.0	0.0
372	1.00000	0.851852	0.0	0.5	1.000000	0.0	0.0	0.0
373	1.00000	0.851852	0.0	0.5	1.000000	0.0	0.0	0.0

4. Xây dựng hàm đánh giá mô hình

Tiến hành xây dựng hàm đánh giá cho các mô hình như sau:

```
# Hàm đánh giá
def classification_eval(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    return accuracy, precision, recall, f1
```

Đoạn code trên định nghĩa một hàm có tên là `classification_eval`, được sử dụng để đánh giá hiệu suất của mô hình phân loại. Hàm này nhận vào hai tham số là `y_test` và `y_pred`, tương ứng là nhãn thực tế và nhãn dự đoán của mô hình trên tập kiểm tra. Các chỉ số đánh giá hiệu suất trả về bởi hàm này bao gồm:

- **Accuracy (Độ chính xác)** là tỷ lệ số lượng dự đoán đúng trên tổng số mẫu. Đây là một chỉ số tổng quát về hiệu suất của mô hình.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Trong đó:

- TP – True Positive: giá trị thực tế và dự đoán đều là positive
- FP – False Positive: giá trị thực tế là negative nhưng dự đoán là positive
- TN – True Negative: giá trị thực tế và dự đoán đều là negative

- FN – False Negative: giá trị thực tế là positive nhưng dự đoán là negative
- **Precision (Độ chính xác của dự đoán tích cực)** là tỷ lệ số lượng dự đoán tích cực đúng trên tổng số dự đoán tích cực. Precision là quan trọng khi chúng ta muốn giảm số lượng dự đoán tích cực giả mạo.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Tỷ lệ dự đoán tích cực thu được)** là tỷ lệ số lượng dự đoán tích cực đúng trên tổng số mẫu thực tế tích cực. Recall quan trọng khi chúng ta muốn đảm bảo rằng mô hình không bỏ sót các mẫu tích cực.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-score:** Là một số hợp nhất giữa precision và recall. F1-score thường được sử dụng khi chúng ta muốn cân nhắc cả precision và recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. Xây dựng mô hình phân lớp

a. Decision Tree

Mô hình phân lớp Decision Tree (cây quyết định) là một phương pháp máy học được sử dụng để phân loại và dự đoán. Nó biểu diễn quyết định thông qua cấu trúc cây, trong đó mỗi nút đại diện cho một quyết định dựa trên một thuộc tính nào đó, và mỗi lá của cây đại diện cho một phân lớp hoặc kết quả.

Đối với mô hình này, dữ liệu của chúng ta không cần chuẩn hóa, vì thế dữ liệu ta sẽ dùng là 'data_classify_no_standard'.

Sau đó, ta tách biến target “Sleep Disorder” với các biến feature còn lại bằng phương thức drop(). Biến độc lập X sẽ được gán với các cột feature và biến phụ thuộc y là cột “Sleep Disorder_Yes” để mô hình dự đoán thông qua tác động của các biến độc lập đến biến phụ thuộc.

```
data_classify_tree = data_classify_no_standard.copy()

X = data_classify_tree.drop('Sleep Disorder_Yes', axis = 1)
y = data_classify_tree['Sleep Disorder_Yes']
```

Tiếp theo, ta tiến hành tách tập dữ liệu với phương thức `train_test_split()` thành tập train và tập test theo tỷ lệ 80:20 với tham số `random_state = 1` để áp dụng và xây dựng mô hình cây quyết định như sau:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2, random_state = 1)

# Tạo mô hình cây quyết định
decision_tree_model = DecisionTreeClassifier(random_state=42)

# Thiết lập các tham số cần thử nghiệm
param_grid = {'max_depth': [3, 5, 7, None],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4],
              'max_features': [None, 'sqrt', 'log2'],
              'criterion': ['gini', 'entropy']}

# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold trong cross-validation (cv)
grid_search_tree = GridSearchCV(estimator=decision_tree_model,
                                param_grid=param_grid,
                                cv=5, scoring='accuracy')
grid_search_tree.fit(X_train, y_train)

# In ra siêu tham số tốt nhất
print("Best Parameters:", grid_search_tree.best_params_)
```

Đoạn code trên thực hiện việc tạo và tinh chỉnh mô hình Decision Tree để đạt được hiệu suất tối ưu trên dữ liệu đào tạo. Đầu tiên, lưới tham số (`param_grid`) là một dictionary chứa các tham số và các giá trị thử nghiệm tương ứng của chúng, được xác định với các giá trị khác nhau cho các tham số của cây quyết định, như độ sâu tối đa (`max_depth`), số lượng mẫu tối thiểu để chia một nút (`min_samples_split`), số lượng mẫu tối thiểu trong mỗi lá (`min_samples_leaf`), loại độ quan trọng tối đa (`max_features`), và tiêu chí chia (`criterion`).

Sau đó, ta sử dụng phương thức `GridSearchCV()`, một quy trình tìm kiếm tham số tốt nhất tự động để thử nghiệm các tổ hợp khác nhau của tham số từ `param_grid` và sử dụng cross-validation để đánh giá hiệu suất của từng tổ hợp. Trong đó, `estimator = decision_tree_model` có nghĩa là mô hình cây quyết định được sử dụng trong quá trình tìm kiếm, `param_grid = param_grid` là lưới các giá trị tham số cần thử nghiệm ở trên, `cv = 5` là số lượng fold trong quá trình cross-validation và cuối cùng `scoring = 'accuracy'` là phương pháp đánh giá hiệu suất của mô hình, đo lường khả năng dự đoán đúng.

Sau khi quá trình tìm kiếm kết thúc, thông tin về tham số tốt nhất được in ra, giúp cho việc đánh giá được tối ưu để sử dụng trong mô hình và đảm bảo rằng mô hình được điều chỉnh để đạt được hiệu suất tốt nhất trên dữ liệu kiểm thử. Sau đây là kết quả thu được cho tham số tốt nhất:

Best Parameters:

- 'criterion': 'entropy'

- 'max depth': 5
- 'max features': None
- 'min samples leaf': 1
- 'min samples split': 2

Cuối cùng, ta tiến hành kiểm thử mô hình và gán vào y_pred để thực hiện đánh giá mô hình ở phần sau.

```
y_pred = grid_search_tree.best_estimator_.predict(X_test)
```

b. K-NN

Mô hình phân loại k-Nearest Neighbors (k-NN) là một thuật toán máy học được sử dụng để phân loại, trong đó việc phân loại của một mẫu mới được xác định dựa trên việc quan sát các mẫu huấn luyện gần nhất trong không gian đặc trưng. Ý tưởng cơ bản của thuật toán k-NN là xác định lớp của một điểm dữ liệu dựa trên đa số (majority vote) của k điểm dữ liệu gần nhất với nó trong không gian đặc trưng. Số lượng k được chọn là một tham số quan trọng của thuật toán và có thể được điều chỉnh theo yêu cầu của bài toán cụ thể.

Khác với 2 mô hình trên, dữ liệu của chúng ta cần phải được chuẩn hóa đối với mô hình này, vì thế dữ liệu được dùng sẽ là 'data_classify_standard'. Tương tự như các mô hình trên, ta tiến hành tách tập dữ liệu để áp dụng và xây dựng mô hình phân lớp k-NN như sau:

```
data_classify_knn = data_classify_standard.copy()
```

```
X = data_classify_knn.drop('Sleep Disorder_Yes', axis = 1)
y = data_classify_knn['Sleep Disorder_Yes']
```

```
# Chia tập dữ liệu thành training, test sets theo tỷ lệ 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2, random_state = 1)
```

```
# Xây dựng mô hình kNN Classification
k = int(pow(X_train.shape[0], 1/2) / 2)
```

```
# Khởi tạo mô hình K-NN
knn_model = KNeighborsClassifier(n_neighbors=k)
```

```
# Thiết lập các tham số cần tìm kiếm
param_grid = {'weights': ['uniform', 'distance'],
              'p': [1, 2],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}
```

```
# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold trong cross-validation (cv)
grid_search_knn = GridSearchCV(estimator=knn_model, param_grid=param_grid,
                               cv=5, scoring='accuracy')
```

```
# Huấn luyện GridSearchCV trên dữ liệu
grid_search_knn.fit(X_train, y_train)
```

```
# In ra các tham số tốt nhất
print("Best Parameters:", grid_search_knn.best_params_)
```

Đầu tiên chúng ta phải xác định giá trị của k trước. Thông thường, ta tính k bằng cách lấy căn bậc hai của số lượng mẫu trong tập huấn luyện, sau đó chia cho 2, đây sẽ là kích thước cụm dữ liệu gần nhất được sử dụng trong thuật toán k -NN. Sau đó, mô hình k -NN sẽ được khởi tạo với giá trị k đã được xác định trước đó.

Đối với mô hình k -NN, các tham số cần tìm kiếm sẽ là trọng số của các điểm láng giềng (weights), tham số p dùng để xác định loại khoảng cách được tính toán trong mô hình và thuật toán được dùng để tìm kiếm các điểm láng giềng (algorithm). Sau đó, ta cũng sử dụng phương thức `GridSearchCV()` với `estimator = knn_model` với các tham số tương tự như mô hình trước. Kết quả thu được cho tham số tốt nhất là:

Best Parameters:

- 'algorithm': 'auto'
- 'p': 1
- 'weights': 'uniform'

Cuối cùng, ta tiến hành kiểm thử mô hình và gán kết quả dự đoán vào biến `y_pred` như sau để chuẩn bị cho việc đánh giá:

```
# Kiểm thử mô hình
y_pred = grid_search_knn.best_estimator_.predict(X_test)
```

c. SVM

Mô hình phân loại Support Vector Machine (SVM) là một thuật toán máy học được sử dụng chủ yếu cho bài toán phân loại. SVM tìm ra siêu phẳng tốt nhất để phân tách giữa các lớp dữ liệu, với các điểm gần siêu phẳng được gọi là vector hỗ trợ để tối đa hóa biên lựa chọn giữa các lớp, và kiểm soát sự chấp nhận của mô hình đối với các điểm dữ liệu bị lỗi với tham số C . SVM còn hiệu quả trong các không gian đặc trưng cao chiều mà không cần tính toán trực tiếp các chiều đó. Đây là một trong những công cụ quan trọng và mạnh mẽ trong lĩnh vực máy học.

Tương tự với mô hình k -NN, dữ liệu được dùng cho mô hình SVM cần được chuẩn hóa, vì thế chúng ta chọn dữ liệu 'data_classify_standard' để tiến hành tách tập dữ liệu và xây dựng mô hình.

```
data_classify_svm = data_classify_standard.copy()

X = data_classify_svm.drop('Sleep Disorder_Yes', axis = 1)
y = data_classify_svm['Sleep Disorder_Yes']

print(pd.concat([X, y], axis = 1).head())
```



```
# Chia tập dữ liệu thành training, test sets theo tỷ lệ 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2, random_state = 1)

# Khởi tạo mô hình SVM
svm_model = SVC()

# Thiết lập các tham số cần tìm kiếm
param_grid = {'C': [0.1, 1, 10],
              'kernel': ['linear', 'poly', 'rbf'],
              'gamma': [0.1, 1, 'scale', 'auto'],
              'degree': [2, 3, 4],
              'coef0': [0.0, 1.0, 2.0],
              'shrinking': [True, False]}

# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold trong cross-validation (cv)
grid_search_svm = GridSearchCV(estimator=svm_model, param_grid=param_grid,
                               cv=5, scoring='accuracy')

# Huấn luyện GridSearchCV trên dữ liệu
grid_search_svm.fit(X_train, y_train)

# In ra các tham số tốt nhất
print("Best Parameters:", grid_search_svm.best_params_)
```

Mô hình SVM sẽ được khởi tạo với phương thức SVC(). Đối với mô hình này, các tham số cần tìm kiếm sẽ là tham số C kiểm soát độ chấp nhận của mô hình đối với các điểm dữ liệu bị lỗi phân loại, loại hàm kernel được sử dụng, tham số gamma liên quan đến hình dạng siêu phẳng, hệ số độc lập trong các hàm kernel tuyến tính (coef0) và xác định có hay không việc thuật toán kỹ thuật giảm kích thước của tập dữ liệu hỗ trợ trong lúc đào tạo (shrinking). Phương thức GridSearchCV() được sử dụng tương tự các mô hình với estimator = svm_model. Kết quả thu được tham số tốt nhất là:

Best Parameters:

- 'C': 0.1
- 'coef0': 0.0
- 'degree': 4
- 'gamma': 1
- 'kernel': 'poly'
- 'shrinking': True

Cuối cùng, ta tiến hành kiểm thử mô hình và gán kết quả dự đoán vào biến y_pred như sau để chuẩn bị cho việc đánh giá:

```
# Kiểm thử mô hình
y_pred = grid_search_svm.best_estimator_.predict(X_test)
```

6. Đánh giá mô hình

Sau khi kiểm thử từng mô hình, nhóm tiến hành đánh giá mô hình với các chỉ số của hàm đánh giá classification_eval đã được xây dựng ở phần 4, bổ sung thêm chỉ số AUC và confusion matrix.

```

scores = classification_eval(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
auc = metrics.auc(fpr, tpr)

## Hiển thị giá trị các chỉ số
print(f'Accuracy = {(scores[0] * 100):.1f}%')
print(f'Precision = {(scores[1] * 100):.1f}%')
print(f'Recall = {(scores[2] * 100):.1f}%')
print(f'F1 = {(scores[3] * 100):.1f}%')
print(f'AUC = {(auc * 100):.1f}%')

```

```

# Ma trận nhầm lẫn
print("* Sleep Disorder_Yes = 0 -->", y_test[y_test == 0].size, 'trường hợp')
print("* Sleep Disorder_Yes = 1 -->", y_test[y_test == 1].size, 'trường hợp')
print(confusion_matrix(y_test, y_pred))

```

Hơn nữa, vì tập dữ liệu không lớn, nhóm quyết định sử dụng thêm k-Fold Cross Validation để có thể đánh giá từng mô hình một cách chính xác hơn. Từng mô hình với các tham số khác nhau đã được xác định ở phần trên sẽ được gán vào biến model để cho ra kết quả độ chính xác và độ lệch chuẩn với phương thức cross_val_score() như sau (đoạn code sử dụng cho mô hình cây quyết định và cũng áp dụng tương tự cho các mô hình khác):

```

## Đánh giá mô hình bằng k-Fold Cross Validation
model = DecisionTreeClassifier(random_state=42, criterion = 'entropy',
                               max_depth = 5, max_features = None,
                               min_samples_leaf = 1, min_samples_split = 2)

kfold = KFold(n_splits = 10)
results = model_selection.cross_val_score(model, X, y, cv = kfold)
print(f'Accuracy = {results.mean()*100.0:.2f}%; std = {results.std()*100.0:.2f}%')

```

Kết quả đánh giá mô hình được xác định và tổng hợp trong bảng sau:

	Decision Tree	K-NN	SVM
Accuracy	89.3%	89.3%	90.7%
Precision	89.7%	87.1%	90.0%
Recall	83.9%	87.1%	87.1%
F1	86.7%	87.1%	88.5%
AUC	88.5%	89.0%	90.1%
Confusion matrix	[41 3] [5 26]	[40 4] [4 27]	[41 3] [4 27]

Accuracy (Cross Validation)	91.70%	90.14%	83.07%
Standard deviation (Cross Validation)	3.70%	4.08%	21.75%

7. Kết luận

Khi so sánh kết quả đánh giá của các mô hình, ta thấy các chỉ số của mô hình SVM có giá trị cao nhất trong 3 mô hình phân lớp với Accuracy = 90.7%, Precision = 90.0%, Recall = 87.1%, F1-score = 88.5% và AUC = 90.1%. Điều này chứng tỏ rằng SVM có khả năng dự đoán chính xác nhất và có thể phân loại các điểm dữ liệu tốt. Nếu xem xét Confusion Matrix, ta cũng có thể thấy rằng SVM có ít false positives và false negatives hơn so với các mô hình khác. Tuy nhiên, nếu xem xét k-Fold Cross Validation, kết quả lại cho ra mô hình.

Decision Tree có độ chính xác cao nhất và độ lệch chuẩn thấp nhất trong 3 mô hình, lần lượt là 91.7% và 3.70%. Điều này cho thấy mô hình cây quyết định hoạt động tốt trên tập dữ liệu, chỉ ra rằng mô hình hiệu quả trong việc đưa ra dự đoán chính xác cũng như hiệu suất của mô hình ổn định qua các lần chia dữ liệu.

Ngược lại, mô hình SVM được đánh giá là tốt nhất khi so sánh các chỉ số thông thường thì lại cho ra kết quả không tốt bằng mô hình Decision Tree. Nếu quan sát kỹ Confusion Matrix, ta có thể thấy mô hình Decision Tree chỉ dự đoán sai 1 trường hợp so với mô hình SVM nhưng các chỉ số khác lại thấp hơn rõ ràng. Nguyên nhân có thể là vì bộ dữ liệu chứa ít quan sát, do đó khi đánh giá mô hình bằng các chỉ số thông thường, dự đoán sai chênh lệch một chút cũng có thể dẫn đến sự khác biệt rõ rệt ở kết quả.

Tóm lại, ở đây ta đánh giá mô hình Decision Tree tốt nhất với k-Fold Cross Validation vì khi bộ dữ liệu có ít quan sát, các chỉ số đánh giá thông thường có thể cho ra kết quả có sự chênh lệch lớn, ảnh hưởng đến việc đánh giá mô hình một cách chính xác và hiệu quả. Hơn nữa, SVM và k-NN thích hợp với các biến định lượng, còn Decision Tree thì làm việc hiệu quả hơn trong việc dự đoán cho các biến phân loại, phù hợp hơn với bộ dữ liệu của nhóm.

Chương 4: Kết luận và đề xuất

I. Kết luận

Kết luận của nhóm dựa trên các phương pháp kiểm định, phân cụm, và phân lớp đã cho thấy rằng vấn đề của giấc ngủ không chỉ đơn giản là một khía cạnh độc lập, mà thực sự là một hệ thống phức tạp, bao gồm nhiều yếu tố từ các biến cá nhân đến môi trường làm việc và lối sống. Để cải thiện chất lượng giấc ngủ, sự chú ý đặc biệt cần được đặt vào việc nhận biết và điều chỉnh những yếu tố này.

Mặc dù bộ dữ liệu mà nhóm sử dụng còn hạn chế, nhưng tổng thể, kết quả thu được mang lại ảnh hưởng tích cực. Điều này cho thấy rằng bộ dữ liệu này có tiềm năng để được áp dụng trong thực tế để dự đoán khả năng xuất hiện các triệu chứng khi ngủ của một người. Đồng thời, việc này hỗ trợ trong việc xác định các biện pháp điều chỉnh phù hợp để cải thiện chất lượng giấc ngủ và ứng phó với các vấn đề liên quan.

II. Đề xuất

Dựa trên kết luận rút ra từ nghiên cứu về chất lượng giấc ngủ, rối loạn giấc ngủ và thời lượng giấc ngủ, dưới đây là một số khuyến nghị:

1. Quản lý giấc ngủ cá nhân: Nhận thức được tầm quan trọng của việc quản lý cá nhân đối với chất lượng giấc ngủ, chúng ta cần phải điều chỉnh chiến lược giấc ngủ theo nhu cầu cá nhân, xem xét các yếu tố như tuổi tác, nghề nghiệp và mức độ căng thẳng.
2. Kiểm soát căng thẳng: Xem xét những tác động đáng kể của áp lực công việc đến chất lượng giấc ngủ, ta thực hiện các kỹ thuật quản lý căng thẳng trong cả cuộc sống cá nhân và nghề nghiệp để cải thiện giấc ngủ. Điều này có thể bao gồm thực hành chánh niệm, kỹ thuật thư giãn hoặc tìm kiếm sự hỗ trợ khi cần thiết.
3. Những cân nhắc về nghề nghiệp: Có thể thấy rõ ràng các nghề nghiệp cụ thể, chẳng hạn như giáo viên, luật sư, nhà khoa học và nhân viên bán hàng, có mối quan hệ chặt chẽ với thời gian ngủ. Những người làm những nghề này nên đặc biệt chú ý đến thói quen ngủ của mình và cân nhắc điều chỉnh thói quen của mình nếu cần.
4. Nhận thức về rối loạn giấc ngủ: Hiểu rằng rối loạn giấc ngủ có thể chỉ phụ thuộc một phần vào các triệu chứng, chúng ta cần nâng cao nhận thức về chứng rối loạn giấc ngủ và khuyến khích những người gặp phải các triệu chứng tìm kiếm các biện pháp thích hợp và phát triển thói quen ngủ lành mạnh.
5. Chiến lược ngủ phù hợp với lứa tuổi: Người lớn tuổi thường có xu hướng thời gian ngủ lâu hơn. Điều chỉnh các khuyến nghị về giấc ngủ dựa trên các nhóm

tuổi và xem xét các yếu tố cụ thể theo độ tuổi khi giải quyết các vấn đề liên quan đến giấc ngủ.

6. Phương pháp tiếp cận dành riêng cho giới tính đối với chứng ngưng thở khi ngủ: Có thể thấy được sự khác biệt về giới tính khi nói đến chứng ngưng thở khi ngủ. Do đó, ta cần phát triển các phương pháp tiếp cận dành riêng cho giới tính để chẩn đoán và quản lý chứng ngưng thở khi ngủ, xem xét các yếu tố riêng biệt có thể góp phần vào sự xuất hiện của nó ở các giới tính khác nhau.
7. Hoạt động thể chất và thời gian ngủ: Mặc dù mức độ hoạt động thể chất có thể không có ảnh hưởng rõ ràng đến thời gian ngủ trong dân số nói chung, nhưng những cá nhân có nghề nghiệp hoặc lối sống cụ thể có thể cần đặc biệt chú ý đến việc duy trì sự cân bằng giữa hoạt động thể chất và giấc ngủ.
8. Quản lý chỉ số BMI và huyết áp: Nhận thấy mối tương quan chặt chẽ giữa chỉ số BMI, huyết áp và các triệu chứng giấc ngủ, nhóm khuyến khích những người có chỉ số BMI hoặc huyết áp cao hơn quản lý các yếu tố sức khỏe này vì nó có thể tác động tiêu cực đến giấc ngủ.
9. Điều chỉnh lối sống: Ta thấy được rằng các yếu tố lối sống đóng một vai trò quan trọng trong giấc ngủ. Vì thế, nhóm khuyến khích các cá nhân điều chỉnh lối sống của họ, bao gồm duy trì lịch trình ngủ đều đặn, tạo môi trường thuận lợi cho giấc ngủ và áp dụng các thực hành vệ sinh giấc ngủ lành mạnh.
10. Phương pháp tiếp cận toàn diện: Nhấn mạnh sự phức tạp của giấc ngủ như một hệ thống liên quan đến các yếu tố cá nhân, môi trường làm việc và lối sống. Chúng ta nên có một cách tiếp cận toàn diện để cải thiện giấc ngủ có tính đến tính chất nhiều mặt của ảnh hưởng đến giấc ngủ.

Tóm lại, cách tiếp cận cá nhân hóa và toàn diện, xem xét sự khác biệt của từng cá nhân và các yếu tố đa dạng ảnh hưởng đến giấc ngủ, là chìa khóa để cải thiện chất lượng giấc ngủ tổng thể.

III. Ưu nhược điểm

Báo cáo về tác động đến giấc ngủ thông qua việc sử dụng lập trình mang đến nhiều ưu điểm và nhược điểm quan trọng. Đầu tiên, việc áp dụng phân tích đơn biến là một chìa khóa để hiểu rõ hơn về bộ dữ liệu. Qua đó, chúng ta có thể đưa ra quyết định về việc loại bỏ những điểm dữ liệu nhiễu hoặc áp dụng các phân tích cụ thể sẽ được sử dụng trong bài nghiên cứu. Mặt khác, tiền xử lý và quan sát phân phối là công đoạn quan trọng để lựa chọn kiểm định phù hợp, đồng thời giúp tăng tính chính xác của kết quả.

Phân tích mối quan hệ giữa các biến thông qua biểu đồ trực quan là một trong những ưu điểm nổi bật của nghiên cứu này. Sự đa dạng của các biểu đồ không chỉ làm cho thông tin trở nên dễ hiểu mà còn giúp làm rõ hơn về mối liên kết giữa các yếu tố ảnh hưởng đến giấc ngủ. Kết hợp với việc mô hình hóa phân cụm cung cấp cơ sở để rút ra mối quan hệ, đồng thời giúp hiểu rõ hơn về bối cảnh lý thuyết của nghiên cứu. Phương pháp phân lớp cho thấy bộ dữ liệu thích hợp để đưa vào thực nghiệm.

Tuy nhiên, nghiên cứu cũng gặp một số thách thức quan trọng. Bộ dữ liệu có ít quan sát và xuất hiện một số trường hợp lỗi thu nhập dữ liệu, chẳng hạn như sự chênh lệch giữa số quan sát nam trên 50 tuổi và số quan sát nữ. Điều này có thể ảnh hưởng đến tính đại diện của kết quả và đưa ra những phân tích không chính xác. Hơn nữa, bộ dữ liệu khi phân cụm chưa được tối ưu, làm giảm khả năng phát hiện cấu trúc ẩn bên trong dữ liệu và đặt ra thách thức trong việc đưa ra những kết luận có tính hệ quả.

Để khắc phục nhược điểm của bài báo cáo, nhóm nghiên cứu có thể thực hiện các biện pháp sau:

- Tăng số lượng quan sát trong bộ dữ liệu bằng cách thu thập thêm dữ liệu từ các nguồn khác nhau.
- Sửa lỗi thu nhập dữ liệu bằng cách kiểm tra lại các dữ liệu đã thu thập và loại bỏ các dữ liệu không hợp lệ.
- Thử nghiệm các phương pháp phân cụm khác nhau để tìm ra phương pháp phù hợp chính xác hơn với bộ dữ liệu.

Mã nguồn

[Link Colab](#)

[Link Github](#)

[Link Dataset](#)

Tài liệu tham khảo

- [1] Các bài giảng học phần Lập trình phân tích dữ liệu, TS. Nguyễn An Tế, khoa Công nghệ thông tin trong kinh doanh, trường Công nghệ và Thiết kế UEH, 2023.
- [2] [Body mass index and sleep disturbances: a systematic review and meta-analysis](#)
- [3] [Relationship between Sleep and Hypertension: Findings from the NHANES \(2007–2014\)](#)
- [4] [Sleep-disordered breathing, sleep apnea, and other obesity-related sleep disorders: An Obesity Medicine Association \(OMA\) Clinical Practice Statement \(CPS\) 2022](#)
- [5] [Neurological Sleep Disorders and Blood Pressure](#)
- [6] [Data Visualization with Seaborn](#)

Bảng phân công

Thành viên	Phân công	Đánh giá
Huỳnh Nguyễn Anh Cường	Phân cụm phân cấp, Biểu diễn kết quả phân cụm, Đánh giá mô hình phân cụm, Định nghĩa, mục tiêu phân lớp.	100%
Mã Thành An	Phân phối dữ liệu, Phân tích tương quan giữa các biến, Kết luận và đề xuất.	100%
Thịnh Thị Hải Âu	Định nghĩa, mục tiêu phân cụm, Tiền xử lý trước phân cụm, Phân cụm phân hoạch.	100%
Trịnh Uyên Chi	Tiền xử lý trước phân lớp, Xây dựng hàm đánh giá, Xây dựng mô hình phân lớp, Đánh giá mô hình phân lớp	100%
Võ Tuấn Cường	Chương 1, Xử lý các dữ liệu bị nhiễu, Quan sát các biến mục tiêu, Kết luận và đề xuất.	100%