|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



BÁO CÁO BÀI TẬP LỚN HỌC PHẦN

HỌC MÁY

**ĐỀ TÀI: PHÂN LOẠI VÀ DỰ BÁO CHẤT LƯỢNG**

**GIẤC NGỦ SINH VIÊN**

**Giảng viên hướng dẫn :** TS. Tạ Quang Chiểu

**Lớp :** 64TTNT1

**Nhóm sinh viên thực hiện :** Phạm Tiến Thành : 2251262645

Chu Đức Hoàng : 2251262604

Mai Hoàng Tùng : 2251262656

Lê Hồng Nhật : 2251262623

Hà Nội, năm 2025

**Lời cảm ơn**

Chúng em xin bày tỏ lòng biết ơn chân thành đến thầy TS. Tạ Quang Chiểu, người đã tận tình hướng dẫn và dành nhiều tâm huyết để chỉ bảo chúng em trong suốt quá trình học tập. Nhờ có sự hỗ trợ quý báu của thầy, chúng em đã có cơ hội áp dụng các kiến thức từ môn học "Học máy" vào thực tế và đạt được những kết quả đáng khích lệ. Quá trình thực hiện dự án đã giúp chúng em hiểu sâu hơn về quy trình xây dựng một mô hình học máy, cũng như cách áp dụng kiến thức liên ngành để giải quyết những bài toán thực tiễn.

Dù đã cố gắng nỗ lực hết mình, nhưng với thời gian và kinh nghiệm còn hạn chế, nhóm chúng em nhận thấy đề tài mình xây dựng vẫn còn nhiều điểm cần hoàn thiện. Chúng em rất mong nhận được thêm những góp ý, chỉ bảo từ thầy để có thể nâng cao chất lượng dự án và cải thiện các kỹ năng cần thiết cho các dự án trong tương lai.

Một lần nữa, chúng em xin chân thành cảm ơn thầy vì đã luôn đồng hành, động viên và tạo điều kiện cho chúng em trong suốt quá trình học tập và thực hiện đề tài này. Những kiến thức và kinh nghiệm tích lũy được sẽ là nền tảng vững chắc, giúp chúng em tự tin hơn khi bước vào các dự án lớn tiếp theo.

Mục lục

[Chương 1: Giới thiệu 5](#_Toc568704938)

[1.1 Giới thiệu bài toán 5](#_Toc1028371692)

[1.2 Giới thiệu Machine Learning 5](#_Toc1535412545)

[Chương 2: Nội dung 6](#_Toc2028989425)

[2.1 Chuẩn bị dữ liệu 6](#_Toc143666831)

[2.1.1 Thu thập dữ liệu 6](#_Toc1952527225)

[2.1.2 Tiền xử lí dữ liệu 7](#_Toc516420397)

[2.1.3. Trực quan hóa dữ liệu 8](#_Toc3950135)

[2.2 Phương pháp học máy được chọn để giải quyết bài toán 10](#_Toc542644701)

[2.2.1. Phương pháp K Means 10](#_Toc1881432429)

[2.2.2. Phương pháp Naive Bayes 12](#_Toc706367070)

[2.2.3. Phương pháp K-Nearest Neighbors 14](#_Toc2118633052)

[2.2.4. Phương pháp Linear Regression 16](#_Toc387052302)

[Chương 3: Thực nghiệm 18](#_Toc43154792)

[3.1. Xây dựng K Means 18](#_Toc2080282495)

[3.1.1 Thuật toán K Means sử dụng thư viện Sklearn 18](#_Toc884169411)

[3.1.2 Thuật toán K Means không sử dụng thư viện Sklearn 22](#_Toc493218691)

[3.1.3. So sánh độ chính xác của thuật toán K Means khi sử dụng thư viện Sklearn và không sử dụng thư viện Sklearn 27](#_Toc1565317797)

[3.1.4. Giao diện nhập dữ liệu 28](#_Toc1236171902)

[3.2. Xây dựng Thuật toán Naive Bayes 29](#_Toc1148169191)

[3.3.1. Thuật toán Naive Bayes dùng thư viện 29](#_Toc1160198818)

[3.2.2.Thuật toán Naive Bayes không dùng thư viện 32](#_Toc1250182456)

[3.2.3. Đánh giá thuật toán Naive Bayes không dùng thư viện và thuật toán Naive Bayes dùng thư viện 39](#_Toc441508169)

[3.3. Xây dựng KNN 43](#_Toc107229980)

[3.3.1. Tìm kiếm giá trị K tốt nhất cho thuật toán KNN 43](#_Toc2051875421)

[3.3.2. KNN sử dụng thư viện sklearn 44](#_Toc2022818851)

[3.3.3. KNN không sử dụng thư viện sklearn 48](#_Toc537700413)

[3.4. Xây dựng Linear Regression 51](#_Toc2038224761)

[3.4.1. Linear Regression sử dụng thư viện sklearn 51](#_Toc1276877900)

[3.4.2. Áp dụng Polynomial Features để cải thiện độ chính xác của thuật toán 52](#_Toc2065197788)

[3.4.3. Linear Regression không sử dụng thư viện sklearn 55](#_Toc423556515)

[3.4.4 So sánh độ chính xác giữa Linear Regression trước khi dùng Polynomial Features và sau khi sử dụng 59](#_Toc978730688)

[3.4.5. So sánh độ chính xác giữa Linear Regression khi dùng thư viện Sklearn và không sử dụng thư viện Sklearn 60](#_Toc1649069446)

[3.4.6. Giao diện dự báo 60](#_Toc1210166648)

[3.5. So sánh độ chính xác của KNN và Linear Regression 64](#_Toc828193987)

[Chương 4: Kết luận và hướng phát triển 65](#_Toc311882680)

[4.1. Kết Luận 65](#_Toc988590453)

[4.2. Hướng phát triển 65](#_Toc614915701)

[TÀI LIỆU THAM KHẢO 66](#_Toc1340345167)

# Chương 1: Giới thiệu

## Giới thiệu bài toán

* Trong bối cảnh hiện đại, khi áp lực học tập và lối sống không lành mạnh trở nên phổ biến, chất lượng giấc ngủ của sinh viên ngày càng trở thành một mối quan tâm lớn. Các vấn đề như thiếu ngủ, rối loạn nhịp sinh học hay ngủ không sâu giấc không chỉ ảnh hưởng đến sức khỏe thể chất mà còn gây suy giảm khả năng tập trung và hiệu suất học tập. Việc phân loại và dự báo chất lượng giấc ngủ đóng vai trò quan trọng trong việc hiểu rõ tình trạng sức khỏe tổng thể của sinh viên, từ đó đưa ra các giải pháp cải thiện phù hợp.

Trước đây, chất lượng giấc ngủ thường được đánh giá dựa trên các tiêu chuẩn chung hoặc bảng câu hỏi tự đánh giá, thiếu tính cá nhân hóa và phụ thuộc nhiều vào cảm nhận chủ quan. Tuy nhiên, giấc ngủ là một quá trình phức tạp, chịu ảnh hưởng từ nhiều yếu tố như tuổi, giới tính, thời gian ngủ, hoạt động thể chất, mức độ căng thẳng và thói quen sinh hoạt hàng ngày.

Với sự phát triển của Machine Learning, việc phân loại và dự báo chất lượng giấc ngủ có thể được thực hiện chính xác hơn thông qua việc phân tích dữ liệu đa chiều. Công nghệ này có khả năng xử lý thông tin từ các cảm biến giấc ngủ, thiết bị đeo thông minh, hoặc các dữ liệu sinh lý học như nhịp tim, chu kỳ ngủ, và tần suất thức giấc. Nhờ đó, các mô hình Machine Learning có thể xây dựng các dự đoán cá nhân hóa, giúp sinh viên hiểu rõ hơn về giấc ngủ của mình và áp dụng các giải pháp phù hợp để nâng cao chất lượng cuộc sống.

## Giới thiệu Machine Learning

* Học máy (Machine Learning) là lĩnh vực nghiên cứu và phát triển các phương pháp, kỹ thuật cho phép máy tính hoặc hệ thống thông minh học từ dữ liệu .Các hệ thống này tự động phát hiện ra các mẫu, xu hướng, và mối quan hệ trong dữ liệu để từ đó đưa ra các dự đoán, phân loại, hoặc quyết định, giúp giải quyết các vấn đề cụ thể trong thực tế.
* Phân nhóm dựa trên phương thức học:
* Supervised learning
* Unsupervised learning
* Semi-supervised learning
* Reinforcement learning
* Phân nhóm dựa trên chức năng của các thuật toán:
* Regression Algorithms
* Classification Algorithms
* Clustering Algorithms
* Bayesian Algorithms

# Chương 2: Nội dung

## 2.1 Chuẩn bị dữ liệu

### 2.1.1 Thu thập dữ liệu

* Nguồn dữ liệu sử dụng trong bài toán dự đoán lượng calo tiêu hao được lấy từ nguồn Kaggel ([student lifestyle dataset](https://www.kaggle.com/datasets/steve1215rogg/student-lifestyle-dataset))
* Bộ dữ liệu gồm file: student\_lifestyle\_dataset.csv với tổng cộng 2000 mẫu, bao gồm các thông tin về đặc trưng đầu vào và nhãn mục tiêu.
* Cấu trúc dữ liệu:
* File student\_lifestyle\_dataset.csv: Chứa các đặc trưng (features) đầu vào và cột biến mục tiêu (Sleep\_Quality) gồm:
* Student\_ID: Mã định danh duy nhất cho mỗi sinh viên (dạng số nguyên).
* Age: Tuổi của sinh viên (dạng số nguyên).
* Gender: Giới tính của sinh viên (dạng chuỗi), có thể là "Male", "Female", hoặc "Other".
* Sleep\_Hours: Số giờ ngủ trung bình hàng ngày của sinh viên (dạng số thập phân).
* Social\_Hours: Số giờ dành cho hoạt động xã hội trung bình mỗi ngày (dạng số thập phân).
* Physical\_Activity: Số giờ tham gia các hoạt động thể chất trung bình mỗi ngày (dạng số thập phân).
* Extracurricular\_Hours: Số giờ tham gia các hoạt động ngoại khóa trung bình mỗi ngày (dạng số thập phân).
* Study\_Hours: Số giờ học tập trung bình mỗi ngày (dạng số thập phân)
* Stress\_Level: Mức độ căng thẳng của sinh viên (dạng chuỗi), ví dụ: "Low", "Moderate", hoặc "High".
* Sleep\_Quality: Chất lượng giấc ngủ được đánh giá theo thang điểm (dạng số nguyên).

### 2.1.2 Tiền xử lí dữ liệu

* Tạo một bản sao dữ liệu và xác định các cột phân loại (categorical variables)



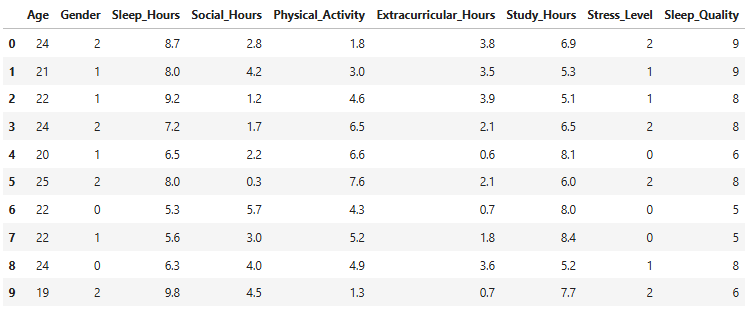
* Áp dụng Label Encoding cho từng cột phân loại



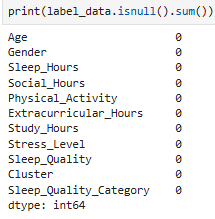
* Xóa cột không liên quan



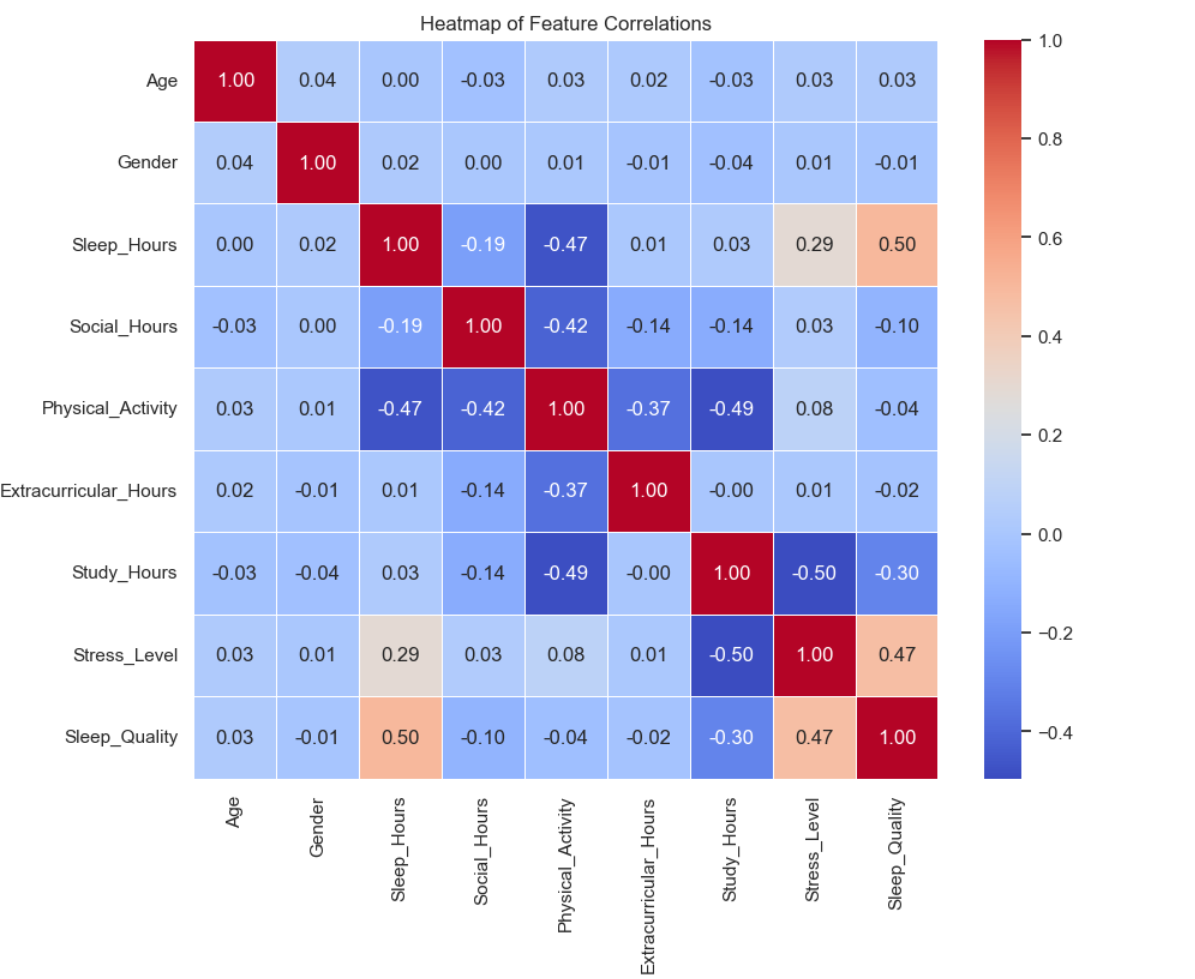
* Sau khi chạy ta được bảng dữ liệu như sau



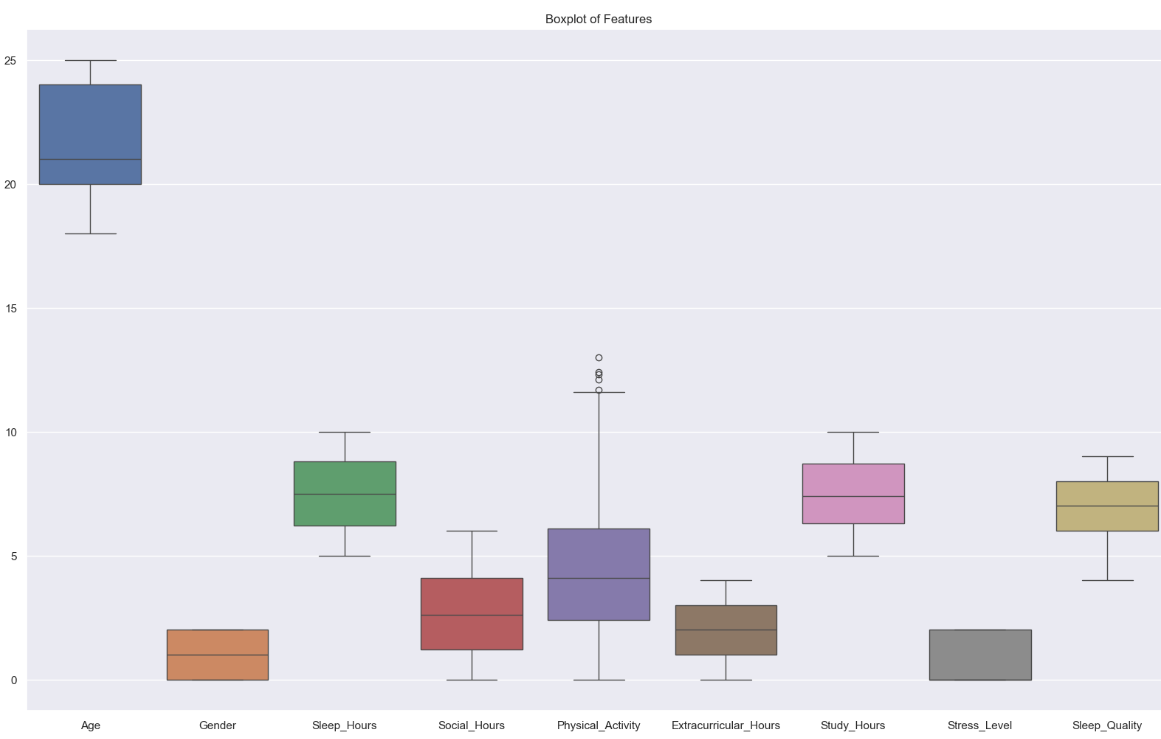
* Gender: Các giá trị được mã hóa thành số (Female = 0, Male = 1, Other = 2).
* Stress\_Level: Các mức độ căng thẳng được mã hóa (Low = 0, Moderate = 1, High = 2).
* Hiển thị dữ liệu thiếu nếu có

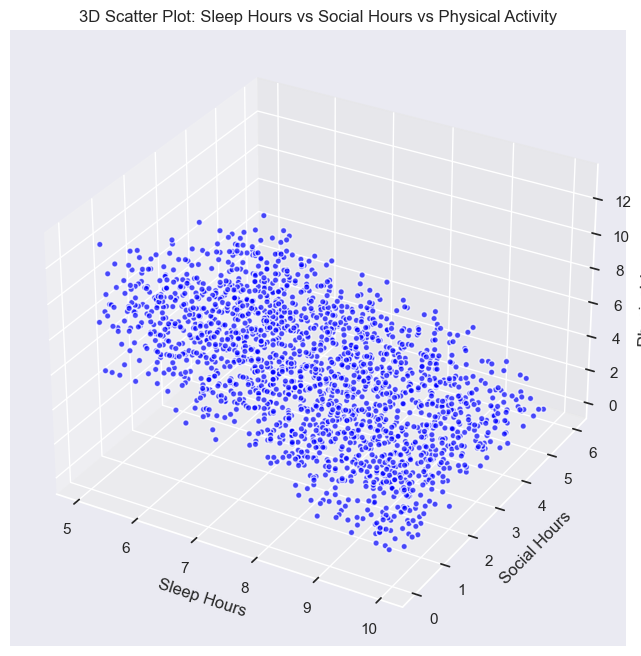


### 2.1.3. Trực quan hóa dữ liệu



Biểu đổ HeatMap thể hiện mức độ tương quan giữa các thuộc tính trong bộ dữ liệu

Biểu đồ Boxplot thể hiện sự phân bố của dữ liệu trong bộ dữ liệu



Biểu đồ Scatter 3D thể hiện sự phân bố của dữ liệu trong bộ dữ liệu theo 3 thuộc tính Sleep\_Hours, Social\_Hours và Physical\_Activity

## 2.2 Phương pháp học máy được chọn để giải quyết bài toán

### 2.2.1. Phương pháp K Means

- K-Means là một thuật toán phân cụm (clustering) không giám sát, được sử dụng phổ biến trong Machine Learning để phân chia dữ liệu thành **K cụm** (clusters) dựa trên tính tương đồng. Thuật toán này tìm cách tối ưu hóa khoảng cách trong cụm (intra-cluster) và giảm khoảng cách giữa các cụm khác nhau (inter-cluster).

- K-Means hoạt động theo các bước sau:

1. Khởi tạo số cụm (K):
2. Người dùng chọn số lượng cụm mong muốn (K).
3. Khởi tạo tâm cụm (centroids):

Lựa chọn ngẫu nhiên K điểm làm tâm cụm ban đầu.

1. Phân cụm dữ liệu:

Mỗi điểm dữ liệu sẽ được gán vào cụm có tâm gần nhất dựa trên khoảng cách (thường sử dụng khoảng cách Euclid).

Trong đó:

* và là các thành phần của điểm dữ liệu và tâm cụm
* d là số chiều trong không gian

1. Cập nhật tâm cụm:

Tính lại tâm cụm mới bằng cách lấy trung bình tất cả các điểm dữ liệu trong cùng cụm.

1. Lặp lại:

Lặp lại bước 3 và 4 cho đến khi:

* 1. Không có sự thay đổi về cụm của các điểm dữ liệu.
  2. Hoặc khi đạt đến số lần lặp tối đa.

1. Kết quả cuối cùng:

Trả về K cụm với các điểm dữ liệu được phân nhóm và tâm cụm tương ứng.

Hàm mục tiêu :

K-Means cố gắng tối thiểu hóa tổng bình phương khoảng cách giữa các điểm dữ liệu và trung tâm cụm của chúng:

Trong đó :

* X : là điểm dữ liệu
* : trung tâm của cụm
* : khoảng cách Euclid giữa x và

**Ưu điểm**

* Đơn giản, dễ hiểu và nhanh chóng với dữ liệu lớn.
* Hiệu quả với dữ liệu có cấu trúc phân cụm rõ ràng.

**Nhược điểm**

1. Phụ thuộc vào K:
2. Số lượng cụm *￼*K cần được xác định trước, và việc chọn *￼*K không chính xác có thể làm giảm hiệu quả phân cụm.
3. Nhạy cảm với tâm cụm khởi tạo:Khởi tạo ngẫu nhiên có thể dẫn đến kết quả không tối ưu.
4. Không phù hợp với dữ liệu phi cầu hoặc có kích thước không đồng đều:

K-Means giả định các cụm có dạng cầu và kích thước tương tự.

1. Dễ bị ảnh hưởng bởi outliers:

Các giá trị bất thường có thể làm lệch tâm cụm.

**Các cải tiến phổ biến**

1. K-Means++:
2. Cải thiện việc chọn tâm cụm ban đầu để giảm thiểu khả năng hội tụ vào cực trị cục bộ.
3. Hierarchical K-Means:

Kết hợp K-Means với phương pháp phân cụm phân cấp.

1. Mini-Batch K-Means:

Tăng tốc K-Means bằng cách sử dụng một tập con dữ liệu (mini-batch) để cập nhật tâm cụm.

### 2.2.2. Phương pháp Naive Bayes

**a.Khái niệm**

**-** Thuật toán Naive Bayes là một nhóm các thuật toán phân loại dựa trên định lý Bayes, được gọi là "Naive" vì nó giả định rằng các đặc trưng (features) trong dữ liệu độc lập với nhau. Đây là một thuật toán học máy đơn giản nhưng rất hiệu quả, đặc biệt trong các bài toán phân loại văn bản như phân loại email (spam hoặc không spam), phân tích cảm xúc, và nhận diện chủ đề.

**b. Công thức tổng quát**

Naive Bayes dựa trên định lý Bayes, cho phép tính xác suất có điều kiện. Định lý Bayes cho phép tính xác suất của một lớp Ckhi biết các đặc trưng :

Trong đó:

=

* là xác suất có điều kiện của lớp cho các đặc trưng
* là xác suất của lớp C (xác suất a priori).
* là xác suất của các đặc trưng cho lớp C .
* là xác suất của các đặc trưng, có thể coi là một hằng số trong quá trình phân loại

**c. Chi tiết các bước**

* Dữ liệu cần có:

D : Tập dữ liệu huấn luyện , được vector hóa dưới dạng

: tập các tài liệu của D thuộc lớp với i ={1,2,3...}

Các thuộc tính độc lập xác suất đôi một với nhau

* Thuật toán Naive Bayes cơ bản :

Bước 1: Huấn luyện Naive Bayes (dựa vào tập dữ liệu)

Tính xác suất

Tính xác suất

Bước 2 : Phân lớp

Tính

được gán vào lớp sao cho

**d.Đánh giá**

**Ưu điểm**:

* Dễ dàng cài đặt và hiệu quả đối với nhiều loại dữ liệu.
* Tốc độ huấn luyện và dự đoán nhanh, phù hợp với các bài toán có kích thước dữ liệu lớn.
* Có thể xử lý tốt trong các bài toán phân loại văn bản.

**Nhược điểm**:

* Giả định độc lập giữa các đặc trưng thường không đúng trong thực tế.
* Không thể mô hình hóa các mối quan hệ phức tạp giữa các đặc trưng.

### 2.2.3. Phương pháp K-Nearest Neighbors

**a. Khái niệm**

- Thuật toán **K-Nearest Neighbors (KNN)** là một thuật toán học máy thuộc nhóm **học có giám sát (supervised learning)**, được sử dụng cho cả bài toán phân loại (classification) và hồi quy (regression). Là thuật đơn giản được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.  
 Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.

**b. Các bước trong KNN**

- Ta có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.

- Đo khoảng cách (Euclidian, Manhattan, Minkowski, Minkowski hoặc Trọng số) từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D. Trong bài toán này nhóm em sử dụng đo khoảng cách bằng Euclidian.

- Chọn K (K là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.

- Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.

- Lấy đúng lớp (lớp xuất hiện nhiều lần nhất).

- Lớp của dữ liệu mới là lớp mà bạn đã nhận được ở bước 5.

Ưu điểm của KNN

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

Nhược điểm của KNN

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

**c. Các bước tính toán chi tiết**

+ Tính khoảng cách

Các công thức phổ biến để tính khoảng cách giữa hai điểm ,

* Khoảng cách Euclid (Euclidean distance):
* Khoảng cách Manhattan:
* Khoảng cách Minkowski:
* Khoảng cách Hamming (cho dữ liệu phân loại):  
   d(A|B) = Số lượng thuộc tính khác nhau giữa A và B

+ Chọn K điểm gần nhất

* Sắp xếp tập dữ liệu theo thứ tự tăng dần của khoảng cách.
* Chọn K điểm đầu tiên.

+ Dự đoán kết quả

* Phân loại: Sử dụng quy tắc đa số:
* Hồi quy: Sử dụng trung bình cộng:

**d. Các yếu tố quan trọng trong KNN**

* **Giá trị K:**
  + K nhỏ (ví dụ: 1 hoặc 3): Mô hình nhạy với nhiễu.
  + K lớn: Mô hình ổn định hơn nhưng có thể mất chi tiết cục bộ.
* **Chuẩn hóa dữ liệu:** Nếu các đặc trưng có đơn vị khác nhau, cần chuẩn hóa dữ liệu để đảm bảo khoảng cách không bị chi phối bởi một số đặc trưng.
* **Đo khoảng cách:** Lựa chọn phù hợp với dữ liệu.
* **Độ phức tạp:** Thuật toán KNN khá chậm với tập dữ liệu lớn do cần tính khoảng cách đến tất cả điểm.

### 2.2.4. Phương pháp Linear Regression

**a. Khái niệm**

* Hồi quy tuyến tính là một loại thuật toán [học máy có giám sát](https://www.geeksforgeeks.org/supervised-machine-learning/) dùng để tính toán mối quan hệ tuyến tính giữa biến phụ thuộc và một hoặc nhiều biến độc lập bằng cách áp dụng phương trình tuyến tính vào dữ liệu quan sát.

**b. Các bước thực hiện**

* Hàm số f(x) mô tả mối quan hệ giữa nhãn và các đại lượng đầu vào:

y≈=f(x) = w1\*x1+w2\*x2+….+wn\*xn

* Vector đặc trưng x = [x1,x2,…,xn]
* Vector trọng số w = [w1,w2,…,wn] – các hệ số tối ưu cần tìm
* y: giá trị thực tế
* : giá trị dự đoán
* khi đó = w1\*x1+w2\*x2+….+wn\*xn =
* Dựa vào hàm mất mát để tìm hệ số w để hàm số đạt giá trị nhỏ nhất
* Giải phương trình đạo hàm bằng 0:
* Nếu ma trận khả nghịch => phương trình có nghiệm duy nhất:
* Nếu ma trận không khả nghịch thì phương trình vô nghiệm hoặc có vô số nghiệm thì ta tìm nghiệm đặc biệt duy nhất bằng cách dựa vào giả nghịch đảo

w = y

*- Bias trick:* Thêm hệ số b vào sẽ khiến mô hình linh hoạt hơn:

Đặt = 1, ,

* =++…+ + b

**c. Thuật toán Hồi Quy Tuyến Tính với Polynomial Features**

* Mô hình hồi quy tuyến tính được mở rộng bằng cách thêm các lũy thừa của biến đầu vào:

Trong đó:

* là các Polynomial Features.
* là các tham số của mô hình cần tìm.
* k là bậc đa thức.
* **Các Thành Phần Mở Rộng của Polynomial Regression**
  + Polynomial Features không chỉ nâng các biến lên lũy thừa mà còn thêm các sản phẩm chéo giữa các biến, như:
* **Tính Chất và Ảnh Hưởng của Polynomial Features**
  + Tăng số lượng đặc trưng (features):
  + Với mỗi biến đầu vào, bậc đa thức càng cao sẽ tạo ra càng nhiều đặc trưng, dẫn đến sự phức tạp trong mô hình. Ví dụ:
  + Với một biến đầu vào *X=[x1,x2]X = [x\_1, x\_2]*X=[x1 ,x2 ], nếu chọn bậc 3, các đặc trưng sẽ là
* Giúp mô hình hóa mối quan hệ phi tuyến:
* Nếu mối quan hệ giữa các biến không tuyến tính, việc thêm Polynomial Features sẽ cải thiện độ chính xác.
* **Ưu Điểm của Polynomial Regression**
  + Mở rộng mô hình tuyến tính để xử lý phi tuyến tính:
  + Sử dụng Polynomial Features giúp mô hình tuyến tính có thể học được các mối quan hệ phi tuyến.
  + Dễ dàng triển khai với LinearRegression:
  + Không cần thay đổi thuật toán, chỉ cần thêm Polynomial Features vào đầu vào.
* **Nhược Điểm**
  + Nếu bậc đa thức quá cao, mô hình sẽ khớp rất tốt với dữ liệu huấn luyện nhưng không tổng quát tốt cho dữ liệu mới.
  + Khi số lượng biến tăng, số lượng Polynomial Features cũng tăng đáng kể, dẫn đến mô hình phức tạp hơn và tốn nhiều tài nguyên hơn
  + Ví dụ: Nếu có 3 biến đầu vào và chọn bậc 3, số lượng đặc trưng sẽ là:

Số lượng đặc trưng = với n = số lượng biến , d = bậc

* **Lưu Ý Khi Sử Dụng Polynomial Features trong Hồi Quy Tuyến Tính**
  + Chọn bậc đa thức phù hợp:
  + Nên thử nghiệm nhiều bậc khác nhau và sử dụng cross-validation để chọn bậc tốt nhất.
  + Chuẩn hóa dữ liệu (Standardization):
  + Polynomial Features có thể tạo ra các giá trị rất lớn, dễ dẫn đến vấn đề về số học, nên việc chuẩn hóa hoặc chuẩn hóa đặc trưng (standardization/scaling) là cần thiết.

# Chương 3: Thực nghiệm

## 3.1. Xây dựng K Means

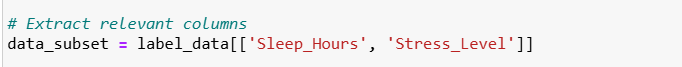
### 3.1.1 Thuật toán K Means sử dụng thư viện Sklearn

- Import các thư viện cần thiết

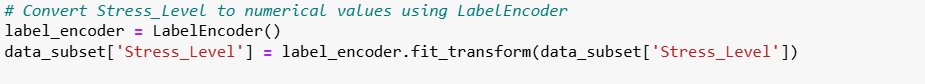
## 

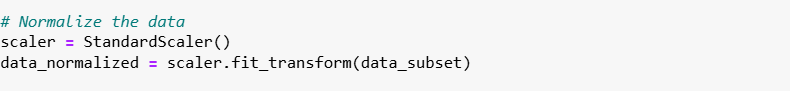
-Sử dụng phương pháp Elbow để tìm ra K tối ưu

+) Trích xuất các cột liên quan

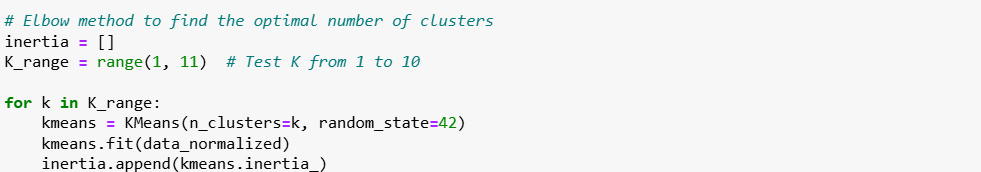


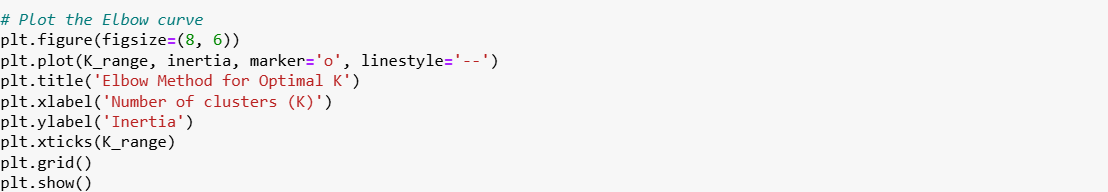
+) Mã hóa thuộc tính phân loại

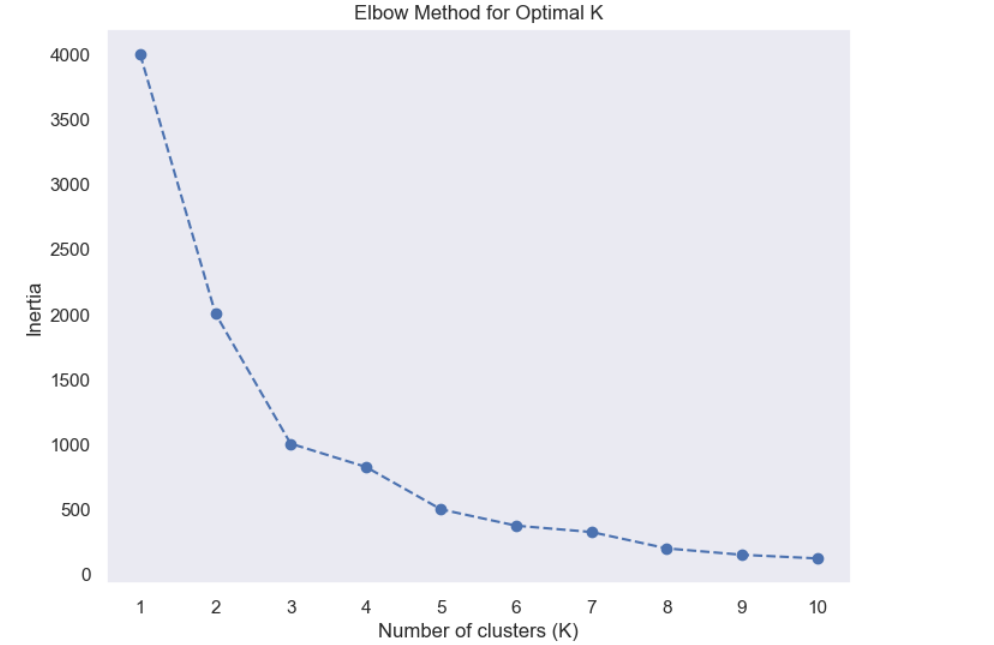
+) Chuẩn hóa dữ liệu



+) Xác định số cụm tối ưu bằng phương pháp Elbow

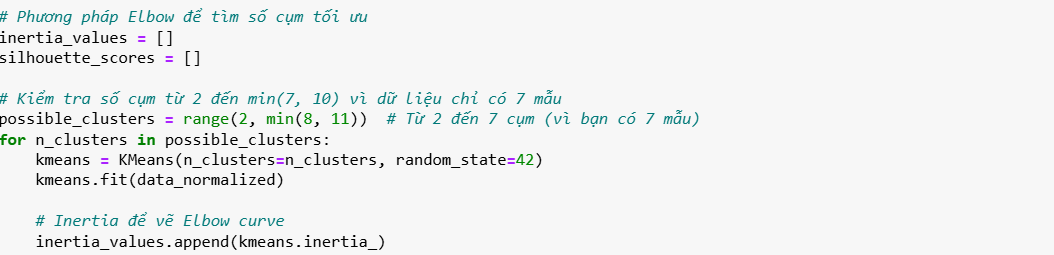
+) Vẽ đồ thị Elbow để tìm số cụm tối ưu

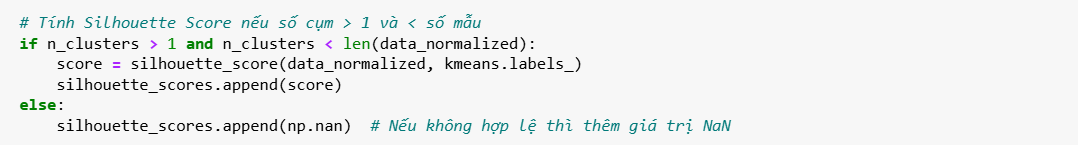
+) Kết quả in ra :

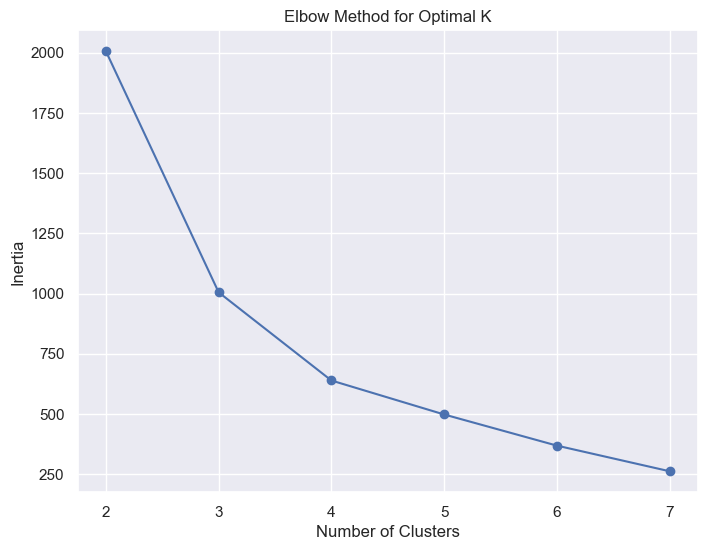
- Code sử dụng thư viện sklearn K means

+) Xác định số cụm tối ưu

Elbow Method (Inertia)

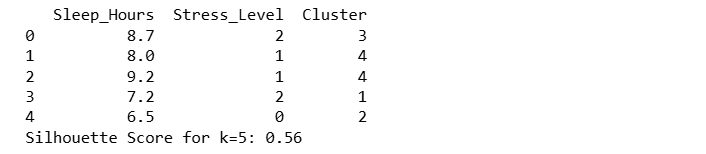
Silhouette Score

+) Trực quan hóa kết quả

+) Thực hiện phân cụm với *k* tối ưu

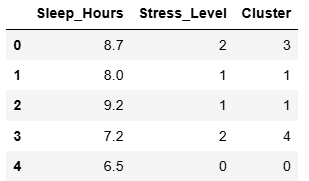


-Kết quả in ra màn hình :



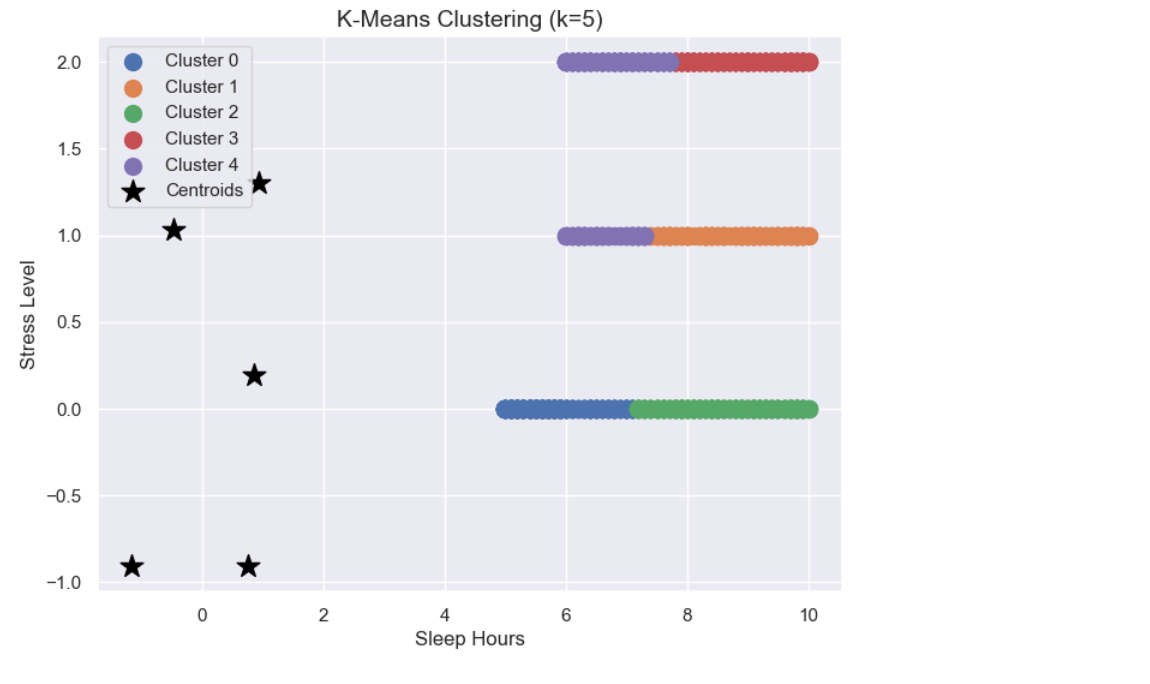
- Thử với mức K = 5 :

- Kết quả đầu ra :



-Đánh giá độ chính xác của thuật toán :

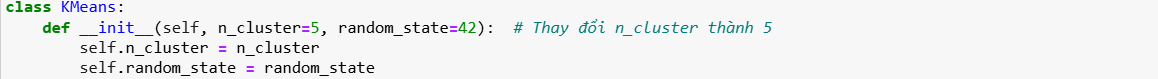
- Trực quan hóa dữ liệu sau khi đã được phân cụm :



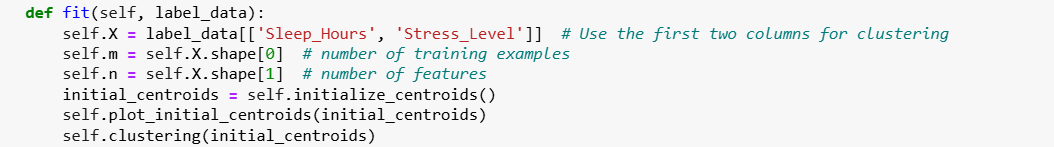
### 3.1.2 Thuật toán K Means không sử dụng thư viện Sklearn

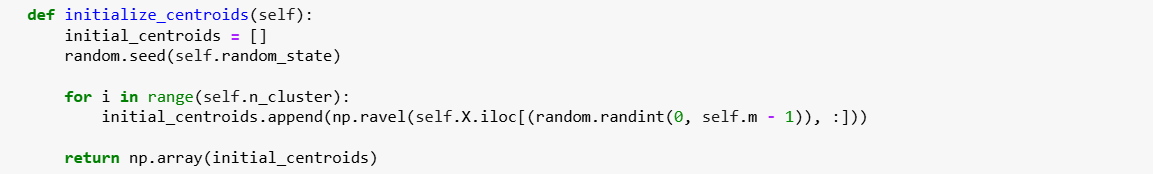
\* Code :

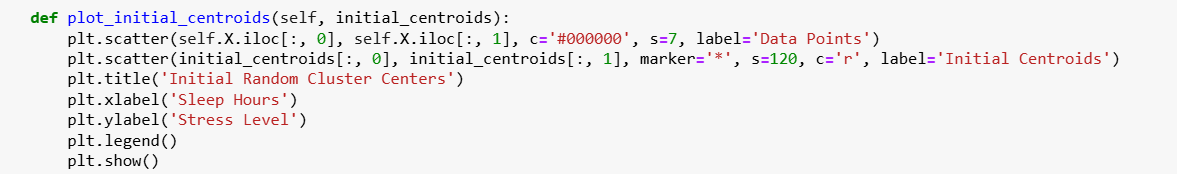
- Lớp KMeans :

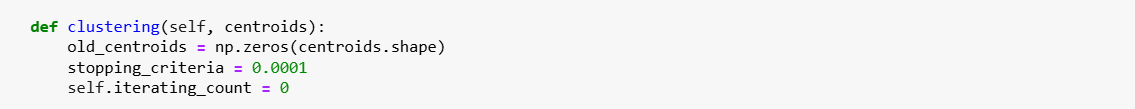


- Phương thức fit:

- Phương thức initialize\_centroids:

- Vẽ tâm cụm ban đầu:

- Phân cụm (Clustering) :

- Vòng lặp :

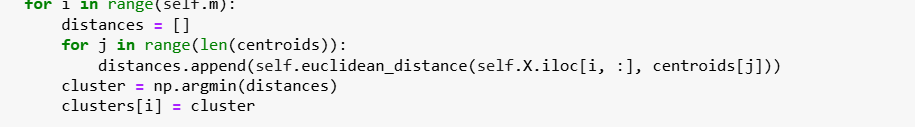
+) Điều kiện dừng của vòng lặp :

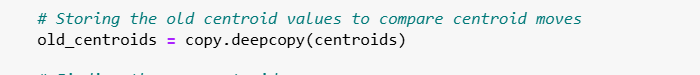


+) Khởi tạo các cụm :

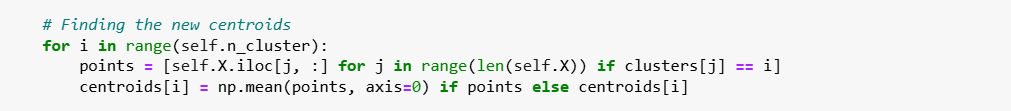


+) Gán điểm dữ liệu vào cụm gần nhất:

+) Lưu trữ giá trị của tâm cụm trước đó :



+) Cập nhật tâm cụm mới :

+) Trực quan hóa trạng thái phân cụm



+) Đếm số lần lặp



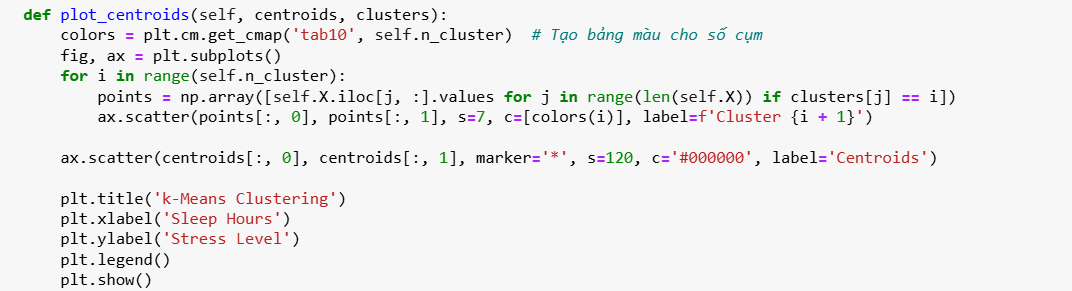
+) Kết thúc vòng lặp



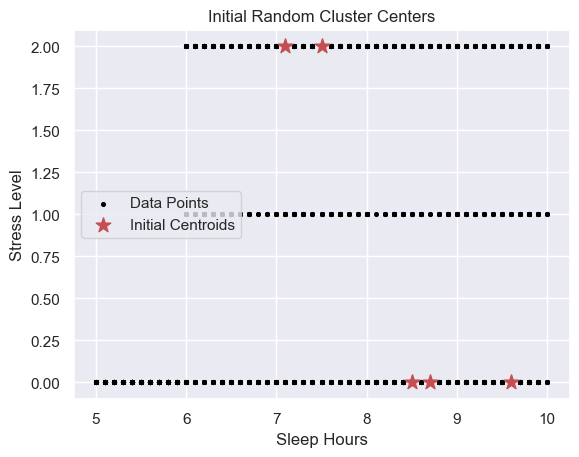
- Tính khoảng cách Euclidean:

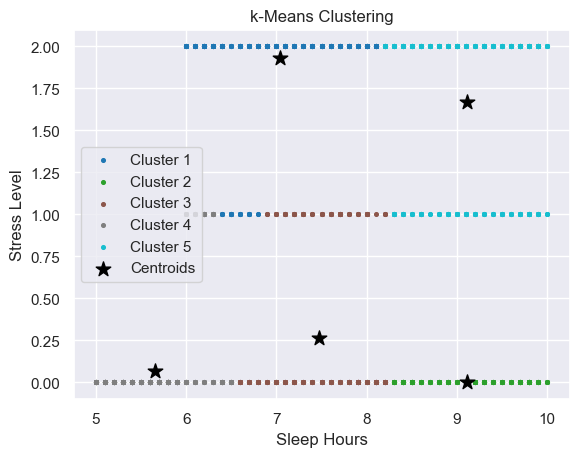


- Vẽ phân cụm:

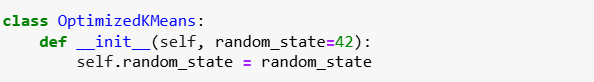
- Áp dụng K-Means và Gán nhãn cụm

- Kết quả đầu ra :

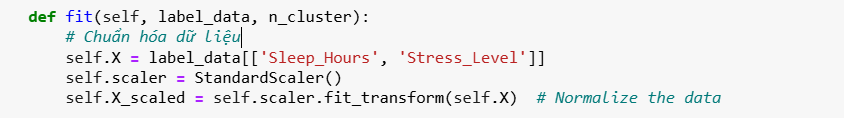




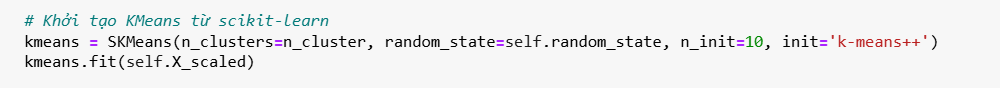
- Tối ưu hóa độ chính xác của thuật toán:

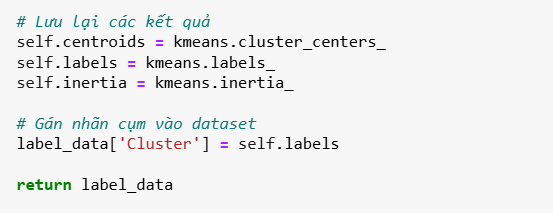
- Khởi tạo lớp   


- Phương thức fit

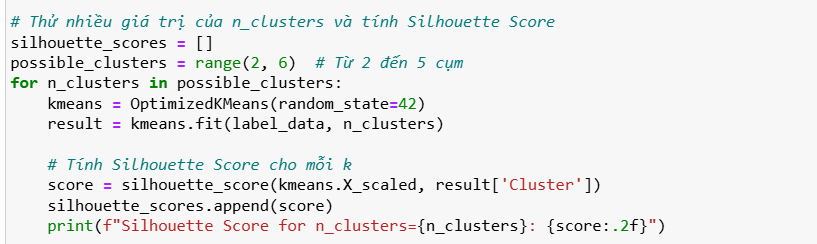


- Áp dụng K-Means

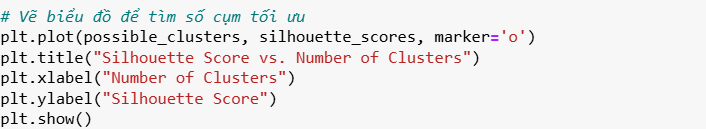
- Lưu kết quả và Gán nhãn cụm vào dữ liệu :



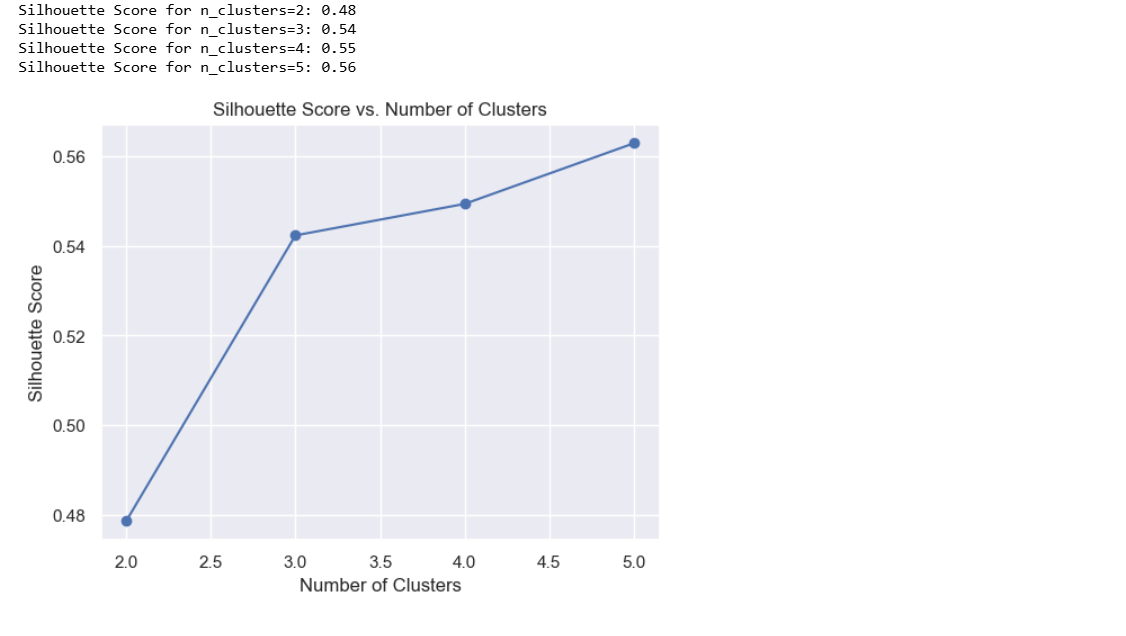
- Tính Silhouette Score:



- Vẽ biểu đồ Silhouette Score:

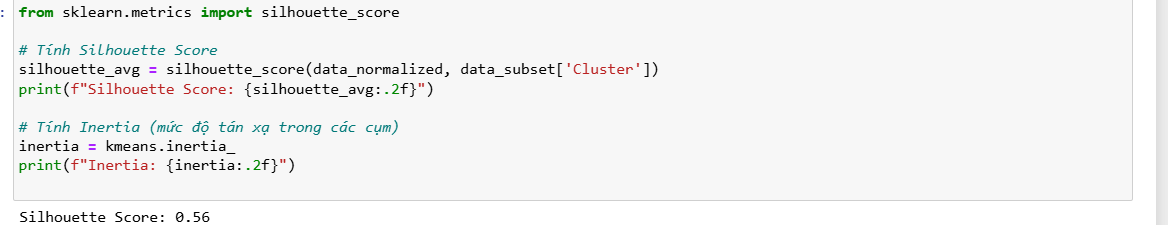


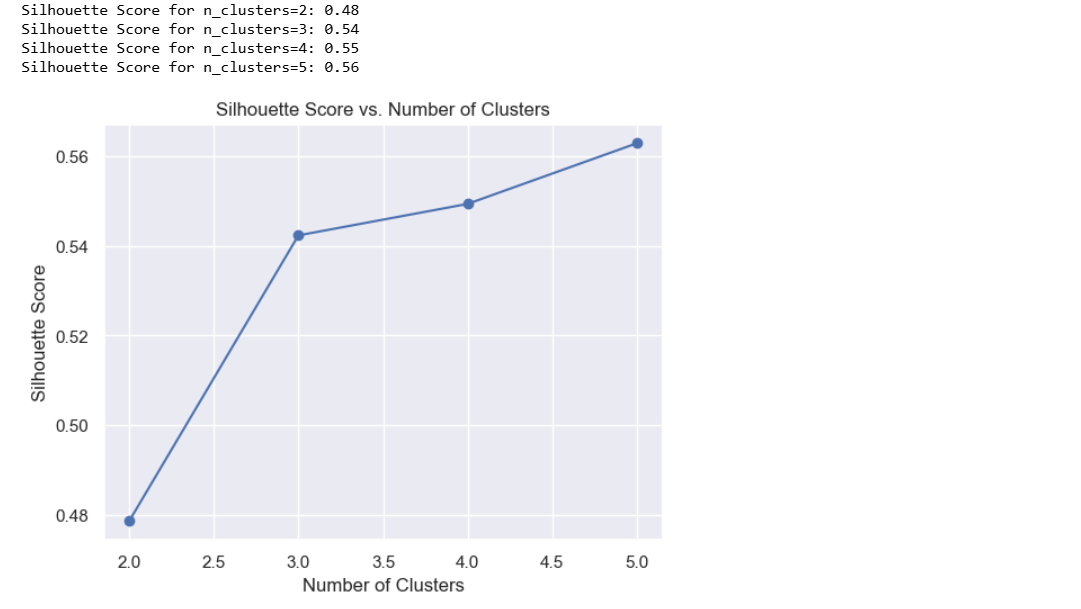
- Kết quả đầu ra và đánh giá độ chính xác của thuât toán :



### 3.1.3. So sánh độ chính xác của thuật toán K Means khi sử dụng thư viện Sklearn và không sử dụng thư viện Sklearn

- Độ chính xác bằng chỉ số Silhouette Score khi sử dụng thư viện Skleearn với mức k = 5

- Độ chính xác bằng chỉ số Silhouette Score khi không sử dụng thư viện Skleearn với mức k = 5

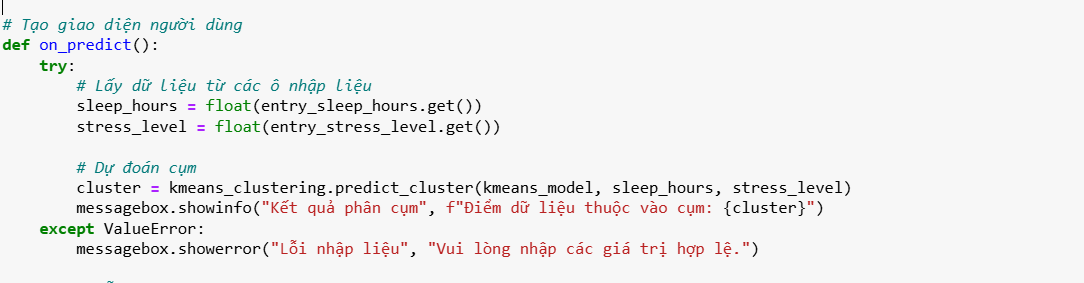


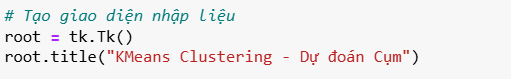
**=> Độ chính xác của thuật toán K Means khi sử dụng thư viện Sklearn và không sử dụng thư viện Sklearn là bằng nhau**

### 3.1.4. Giao diện nhập dữ liệu

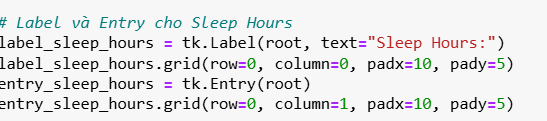
- Code :

- Hàm xử lý sự kiện on\_predict:

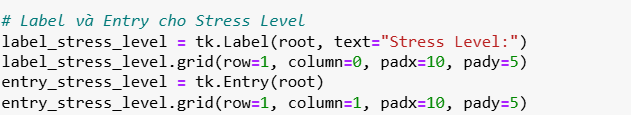
- Tạo giao diện



- Nhãn và ô nhập liệu cho Sleep Hours:



- Nhãn và ô nhập liệu cho Stress Level:



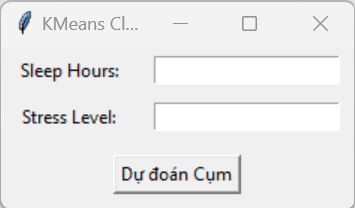
- Nút dự đoán :



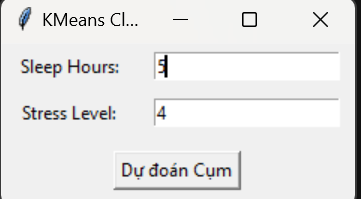
- Chạy giao diện :



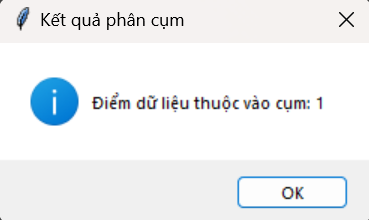
- Màn hình giao diện :



- Thử nhập dữ liệu :

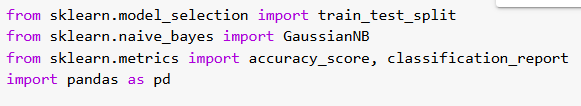


- Kết quả :



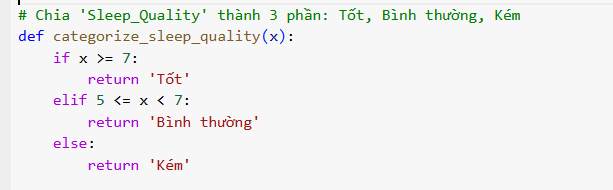
## 3.2. Xây dựng Thuật toán Naive Bayes

### 3.3.1. Thuật toán Naive Bayes dùng thư viện



Import các thư viện cần thiết :

* **sklearn.model\_selection.train\_test\_split**: Được sử dụng để chia bộ dữ liệu thành hai phần: tập huấn luyện (train) và tập kiểm tra (test).
* **sklearn.naive\_bayes.GaussianNB**: Là thuật toán Naive Bayes dành cho dữ liệu có phân phối chuẩn (Gaussian).
* **sklearn.metrics.accuracy\_score**: Tính toán độ chính xác (accuracy) của mô hình.
* **sklearn.metrics.classification\_report**: Tạo báo cáo chi tiết các chỉ số đánh giá mô hình như precision, recall, F1-score.
* **pandas**: Thư viện xử lý dữ liệu dưới dạng bảng (DataFrame).

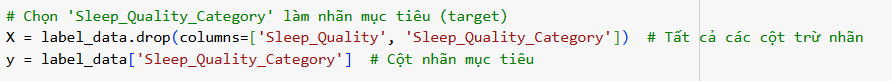


Chia thuộc tính Sleep\_Quality thành 3 loại:

* **Tốt**: Nếu số giờ ngủ lớn hơn hoặc bằng 7.
* **Bình thường**: Nếu số giờ ngủ nằm trong khoảng từ 5 đến dưới 7.
* **Kém**: Nếu số giờ ngủ nhỏ hơn 5.



Kết quả được lưu vào một cột mới trong DataFrame: **Sleep\_Quality\_Category**.



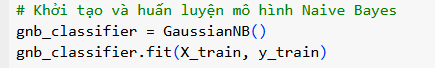
Tách dữ liệu thành đăc trưng (feature) và nhãn mục tiêu (target)

* **X**: Tập dữ liệu đặc trưng (features). Chứa tất cả các cột trong label\_data, trừ:
* Sleep\_Quality: Cột gốc dùng để tính nhãn mục tiêu.
* Sleep\_Quality\_Category: Cột nhãn mục tiêu.
* **y**: Là cột nhãn mục tiêu (Sleep\_Quality\_Category), biểu thị chất lượng giấc ngủ.

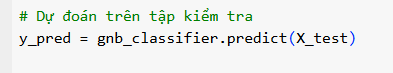


Chia bộ dữ liệu thành tập train và tập test

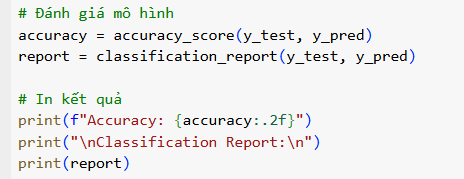
* **train\_test\_split**: Chia dữ liệu thành hai phần:
* **X\_train, y\_train**: Tập huấn luyện (80% dữ liệu).
* **X\_test, y\_test**: Tập kiểm tra (20% dữ liệu).
* **test\_size=0.2**: Chỉ định 20% dữ liệu làm tập kiểm tra.
* **random\_state=42**: Đảm bảo kết quả phân chia giống nhau mỗi khi chạy lại.
* **stratify=y**: Đảm bảo tỷ lệ phân phối các lớp trong tập huấn luyện và kiểm tra giống nhau.



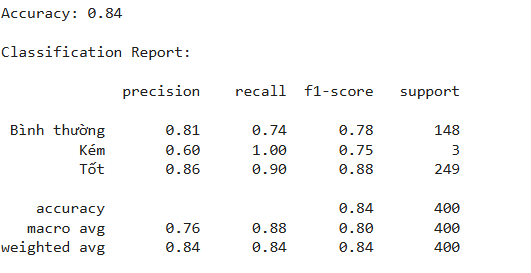
Khợi tạo mô hình Naive Bayes



Dự đoán trên tập kiểm tra



Đánh giá mô hình



* Độ chính xác của mô hình là **0.84 (84%)**, nghĩa là mô hình đã dự đoán đúng 84% trường hợp trên tập kiểm tra.
* Đây là một kết quả khá tốt, nhưng cần phân tích thêm chi tiết hiệu suất trên từng lớp (class).

### 3.2.2.Thuật toán Naive Bayes không dùng thư viện

#### 3.2.2.1.Thuật toán Naive Bayes tổng quát



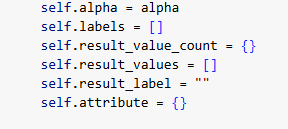
Import các thư viện cần thiết

* pandas: Thư viện xử lý và phân tích dữ liệu dạng bảng (DataFrame).
* train\_test\_split: Hàm từ thư viện scikit-learn dùng để chia dữ liệu thành tập huấn luyện và kiểm tra.

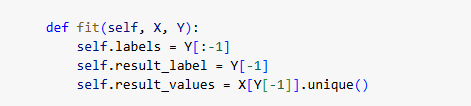


Định nghĩa lớp Naive Bayes

* **Mục đích:** Khởi tạo lớp Naive Bayes với tham số alpha (Laplace Smoothing).
* alpha=1: Giảm xác suất bằng 0 khi không có dữ liệu (giá trị mặc định là 1).



* **Các biến khởi tạo:**
  + self.alpha: Giá trị Laplace Smoothing.
  + self.labels: Danh sách các cột đặc trưng (features).
  + self.result\_value\_count: Tần suất xuất hiện của từng giá trị trong cột kết quả (target).
  + self.result\_values: Danh sách các giá trị duy nhất trong cột kết quả.
  + self.result\_label: Tên cột kết quả.
  + self.attribute: Tần suất kết hợp giữa giá trị của các đặc trưng và giá trị kết quả.



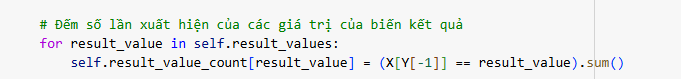
Phương pháp fit

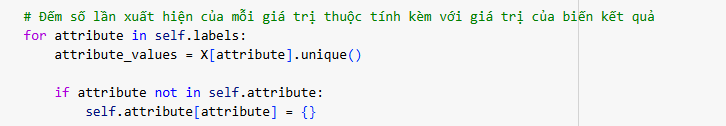
* X: Dữ liệu đầu vào (DataFrame).
* Y: Danh sách các cột (bao gồm đặc trưng và cột kết quả).
* **Các bước:**

1.Lấy danh sách cột đặc trưng từ Y[:-1].

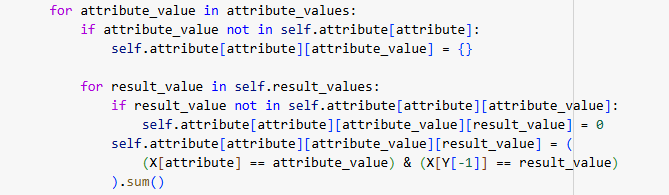
2.Lấy tên cột kết quả từ Y[-1].

3.Lưu các giá trị duy nhất trong cột kết quả vào self.result\_values.

- Đếm số lần xuất hiện của mỗi giá trị trong cột kết quả và lưu vào self.result\_value\_count.

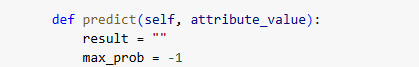


* Lấy danh sách các giá trị duy nhất của từng đặc trưng.
* Khởi tạo từ điển cho từng đặc trưng trong self.attribute.



Đếm tần suất của từng giá trị đặc trưng (attribute\_value) kết hợp với từng giá trị kết quả (result\_value).

Lưu tần suất vào self.attribute.

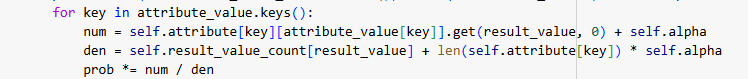


Phương pháp predict

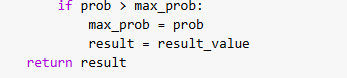
* attribute\_value: Từ điển chứa các giá trị của từng đặc trưng để dự đoán.
* result: Kết quả dự đoán (ban đầu là chuỗi rỗng).
* max\_prob: Xác suất lớn nhất (khởi tạo là -1).



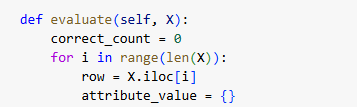
Tính xác suất tiên nghiệm *P(Y)* với Laplace Smoothing.



* Tính xác suất có điều kiện *P(X∣Y)*  với Laplace Smoothing.
* num: Số lần xuất hiện của giá trị đặc trưng key ứng với result\_value.
* den: Tổng số lần xuất hiện của result\_value (đã làm mịn).

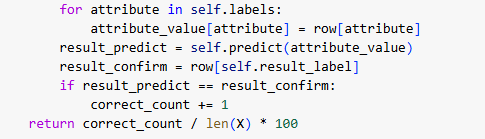


Nếu xác suất lớn hơn max\_prob, cập nhật giá trị dự đoán.



Phương pháp evaluate

* X: Tập dữ liệu để đánh giá.
* correct\_count: Số lượng dự đoán đúng.

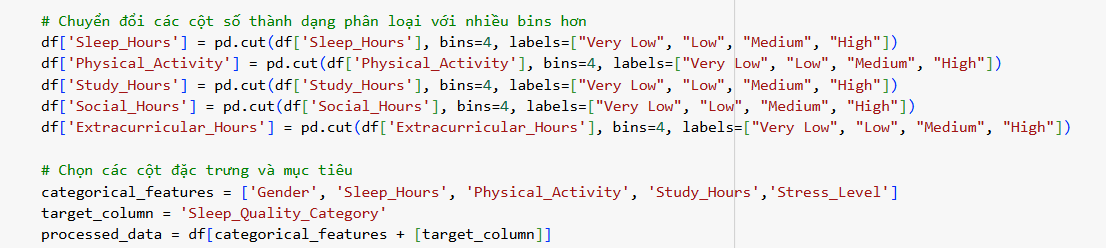


So sánh kết quả dự đoán với kết quả thực tế, tính độ chính xác (*accuracy*).



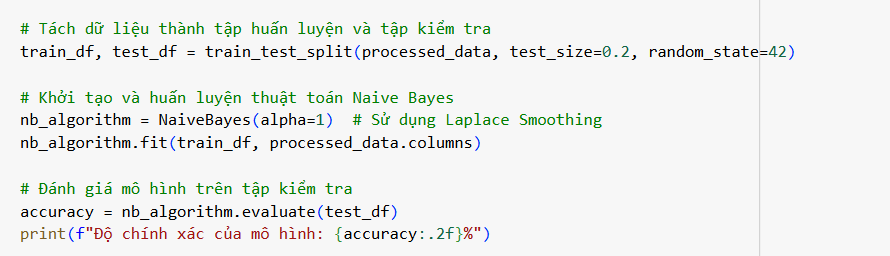
Hàm tiền xử lí

* Tạo cột Sleep\_Quality\_Category từ Sleep\_Quality bằng hàm categorize\_sleep\_quality.
* Chuyển đổi giá trị số trong cột Sleep\_Quality thành các nhãn phân loại



Chia dữ liệu liên tục thành các nhóm phân loại (bins) với 4 mức độ.

Chọn các cột đặc trưng và cột mục tiêu để huấn luyện mô hình.



Chia và đánh giá mô hình

#### 3.2.2.2. Thuật toán Gaussion Naive Bayes



Import thư viện cần thiết

* **LabelEncoder**: Công cụ để mã hóa (encode) các biến phân loại (categorical variables) thành số nguyên.
* **numpy (viết tắt là np)**: Thư viện xử lý mảng số học (arrays), giúp tính toán nhanh chóng.



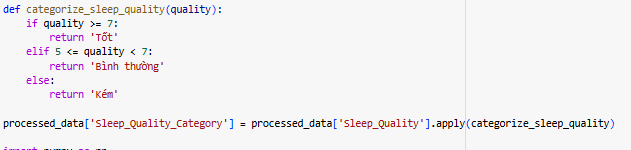
Mã hóa các biến phân loại

* **Mục tiêu**: Mã hóa các cột 'Gender' và 'Stress\_Level' thành số nguyên để máy học dễ xử lý.
* **Cách làm**:
* Tạo một từ điển label\_encoders, trong đó mỗi cột được xử lý bằng LabelEncoder().
* **fit(processed\_data[col])**: Tìm tất cả các giá trị duy nhất trong cột.



**transform**: Chuyển giá trị từ dạng chuỗi thành số nguyên dựa trên bộ mã hóa vừa tạo.

* 'Gender': {Male: 0, Female: 1, Other: 2}
* 'Stress\_Level': {Low: 0, Moderate: 1, High: 2}



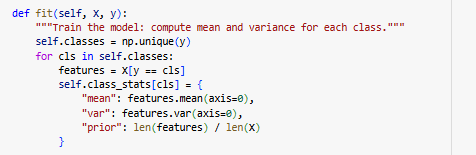
Phân loại Sleep\_Quality

**apply(categorize\_sleep\_quality)**: Áp dụng hàm phân loại cho từng giá trị của cột Sleep\_Quality.



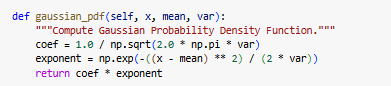
Tạo lớp Gaussian Naive Bayes

**\_\_init\_\_**: Hàm khởi tạo lớp, tạo từ điển class\_stats để lưu thống kê từng lớp (mean, variance, prior).

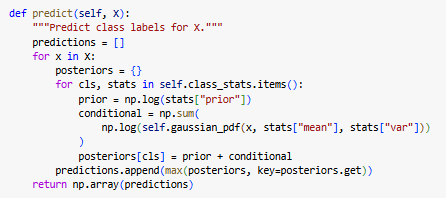


Hàm fit :

* **X**: Dữ liệu đặc trưng (features).
* **y**: Nhãn lớp (labels).
* **np.unique(y)**: Tìm tất cả các lớp duy nhất.
* Với mỗi lớp:
* **features = X[y == cls]**: Lấy các mẫu thuộc lớp cls.
* Tính **mean**, **variance**, và **prior** (xác suất tiên nghiệm của lớp).

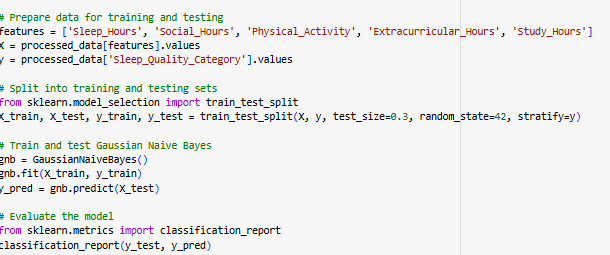


Hàm Gaussian\_pdf: Tính toán mật độ xác suất theo phân phối Gaussian.



Hàm predict:

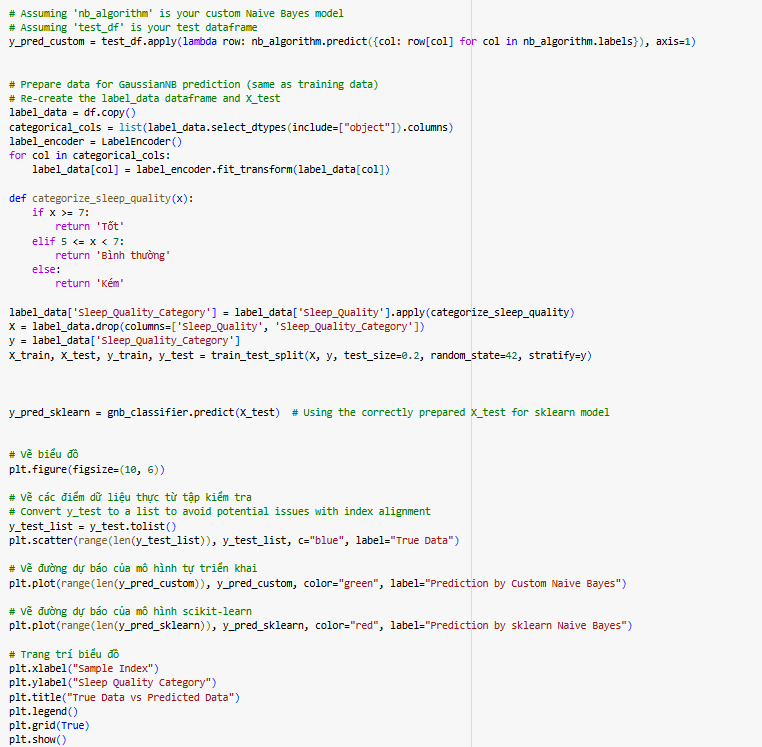
* Với mỗi mẫu đầu vào x, tính toán xác suất hậu nghiệm cho từng lớp:
* **prior**: Log của xác suất tiên nghiệm.
* **conditional**: Tổng log của xác suất có điều kiện.
* Trả về lớp có xác suất cao nhất.

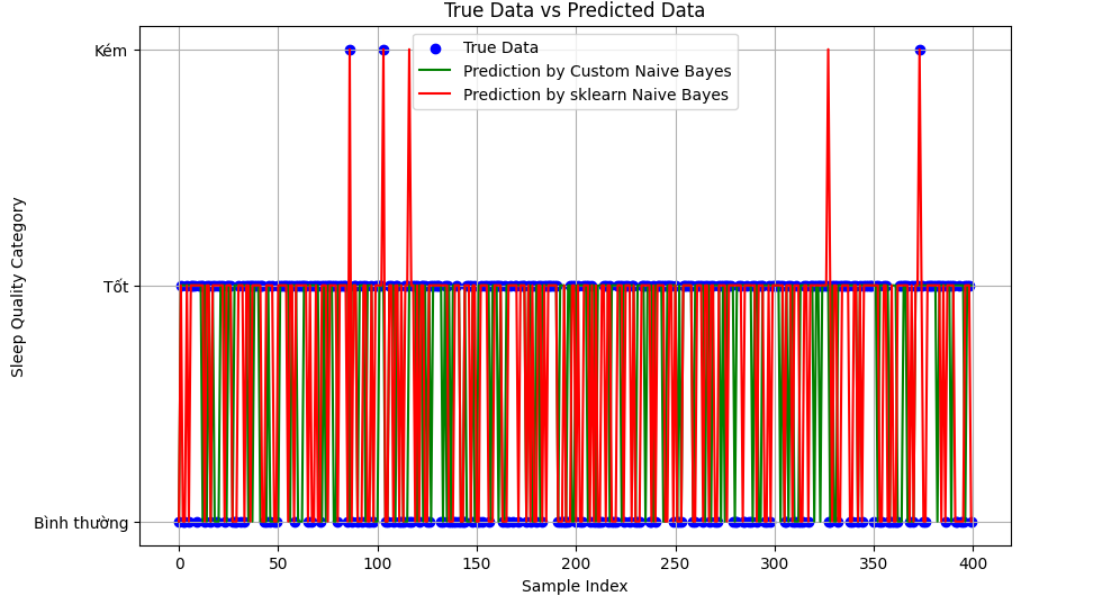


* Chia dữ liệu thành tập huyến luyện và kiểm tra
* Huấn luyện mô hình
* Đánh gia mô hình



### 3.2.3. Đánh giá thuật toán Naive Bayes không dùng thư viện và thuật toán Naive Bayes dùng thư viện





**Nhận xét về biểu đồ:**

* **Dữ liệu thực (True Data - chấm xanh dương):** Các chấm đại diện cho các giá trị thực từ tập kiểm tra. Chúng nằm ở ba cấp độ:
  + "Kém"
  + "Bình thường"
  + "Tốt".
* **Dự báo từ scikit-learn (đường màu đỏ):** Dự báo của mô hình scikit-learn khá gần với dữ liệu thực, nhưng có một số điểm lệch (đường màu đỏ khác với các chấm xanh dương).
* **Dự báo từ mô hình tùy chỉnh (đường màu xanh lá cây):** Mô hình tự triển khai cũng có xu hướng khớp với dữ liệu thực, nhưng có vẻ kém chính xác hơn so với scikit-learn.

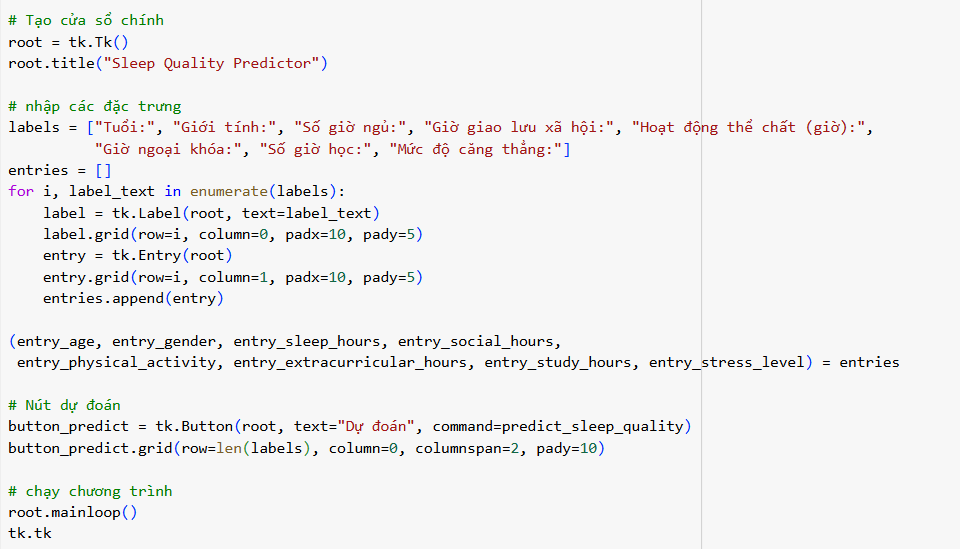
### 3.2.4. Xây dựng giao diện



### **Hàm predict\_sleep\_quality()**

Đây là hàm được gọi khi nhấn nút "Dự đoán". Nó thực hiện các bước:

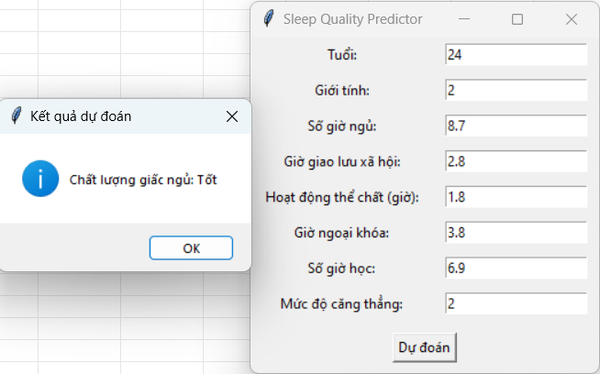
* **Thu thập dữ liệu đầu vào từ người dùng:**
  + Các giá trị nhập từ giao diện được truy cập qua các widget entry (entry\_age, entry\_gender,...).
  + Chuyển đổi dữ liệu nhập sang dạng phù hợp (int cho tuổi, float cho số giờ, mã hóa nhãn cho giới tính và mức độ căng thẳng).
* **Tạo DataFrame:**
  + Dữ liệu thu thập được gói vào một DataFrame của pandas để đưa vào mô hình dự đoán.
* **Dự đoán:**
  + Sử dụng mô hình gnb\_classifier (một mô hình phân loại Naive Bayes) để đưa ra kết quả dự đoán chất lượng giấc ngủ.
  + Hiển thị kết quả dự đoán thông qua hộp thoại messagebox.



Tạo cửa sổ chính chạy và hiện ra các ô nhập các biến đặc trưng

* **Danh sách labels**: Các nhãn hiển thị tương ứng với từng trường thông tin cần nhập.
* **Lặp qua các nhãn (for i, label\_text in enumerate(labels)):**
* Tạo một nhãn (tk.Label) hiển thị tên trường nhập liệu.
* Tạo một ô nhập liệu (tk.Entry) để người dùng điền thông tin.
* Lưu các widget nhập liệu vào danh sách entries để dễ dàng truy cập sau này.

Gán các widget trong entries cho các biến cụ thể để sử dụng trong hàm predict\_sleep\_quality.

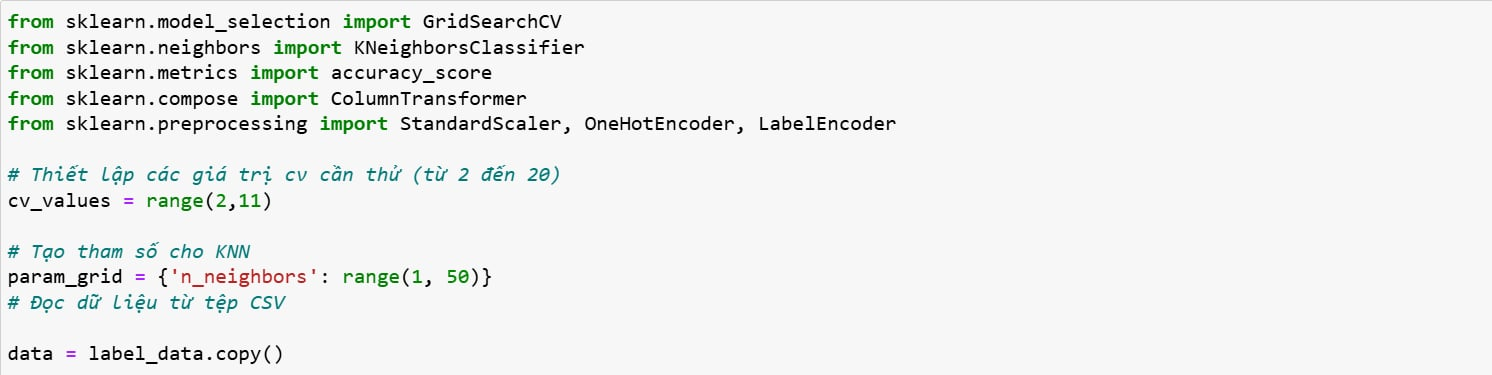


Kết quả chương trình

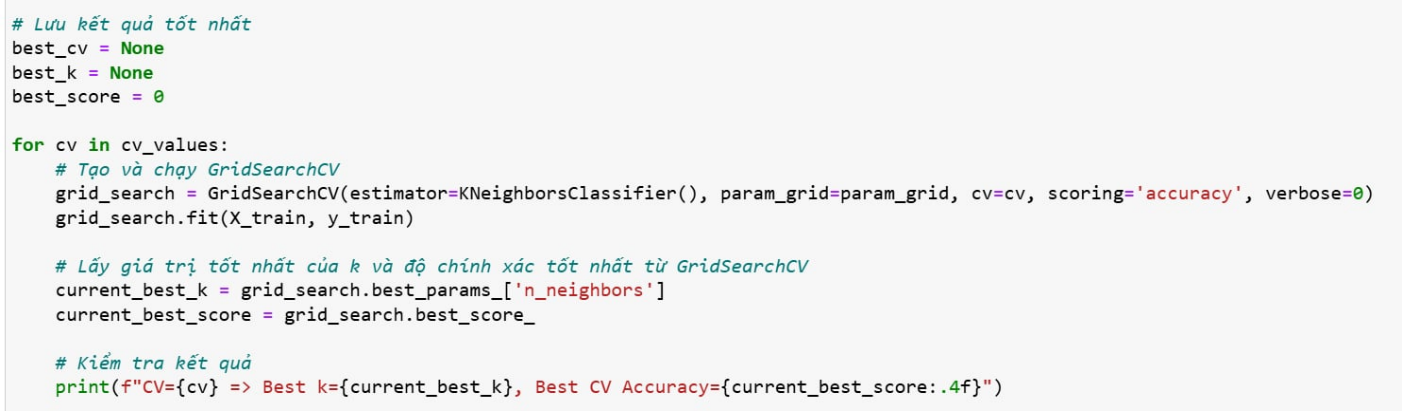
## 3.3. Xây dựng KNN

### 3.3.1. Tìm kiếm giá trị K tốt nhất cho thuật toán KNN

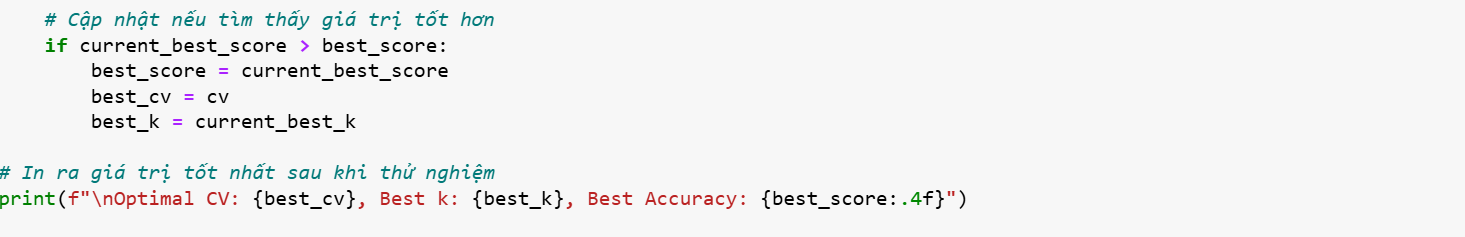
- Thiết lập giá trị cv và tạo tham số cho KNN



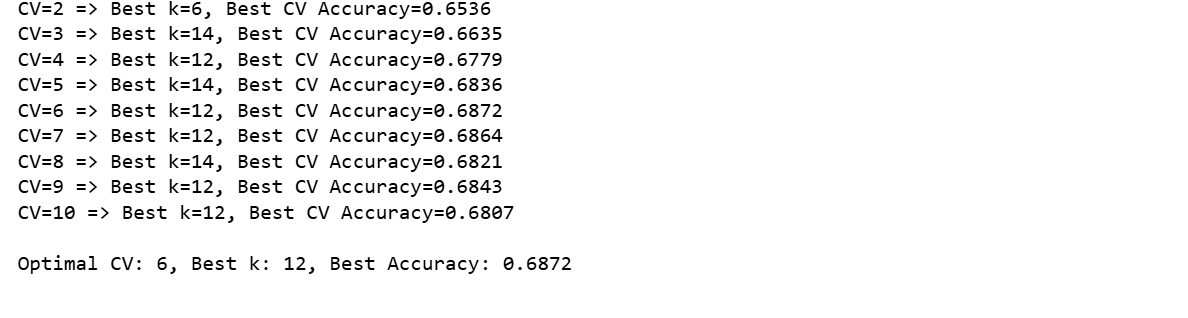
* Tạo và chạy GridsearchCV để tìm ra giá trị K tốt nhất với mỗi giá trị cv



- Cập nhật khi có giá trị tốt hơn sau đó in ra giá trị tốt nhất cuối cùng

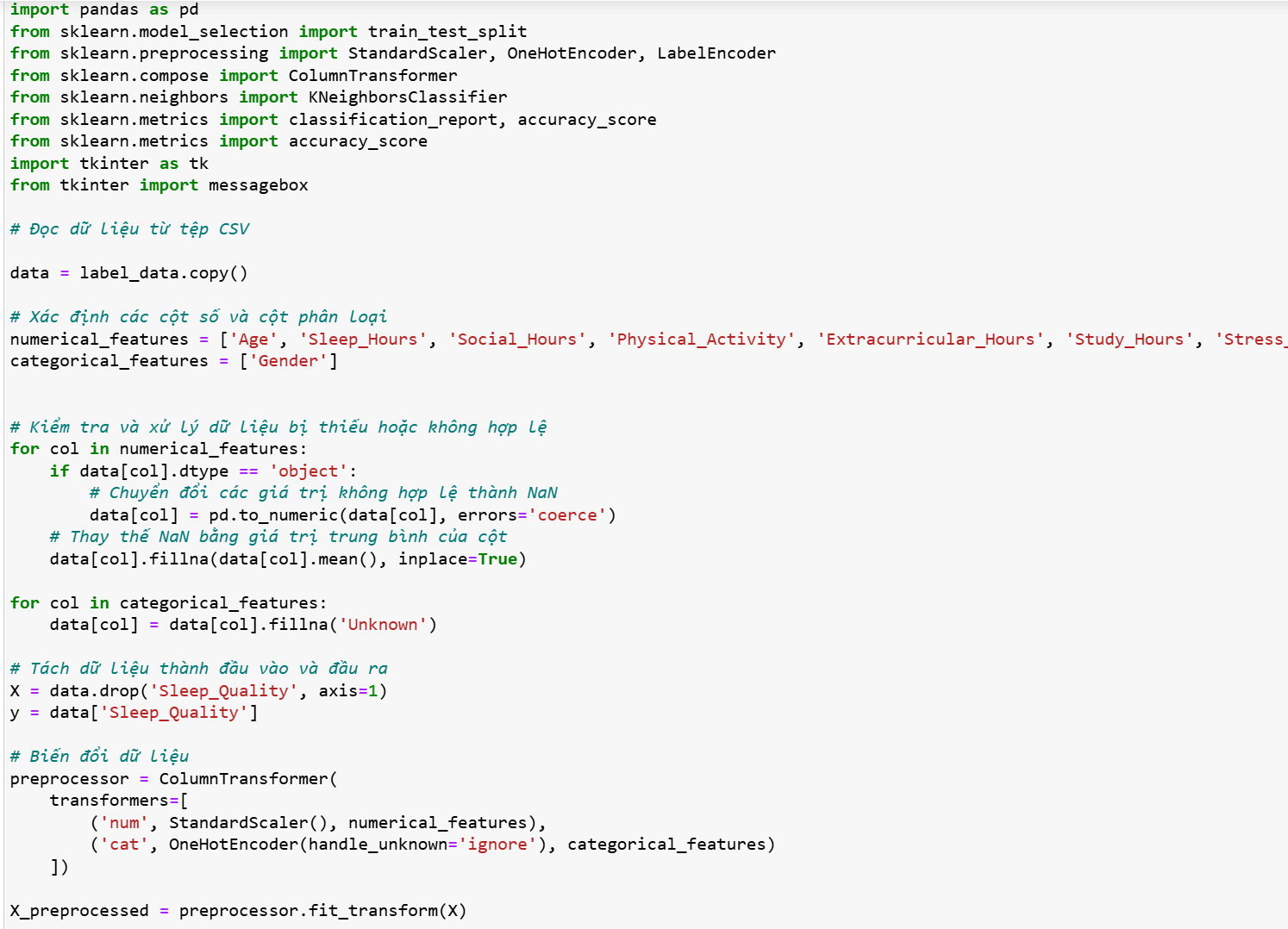


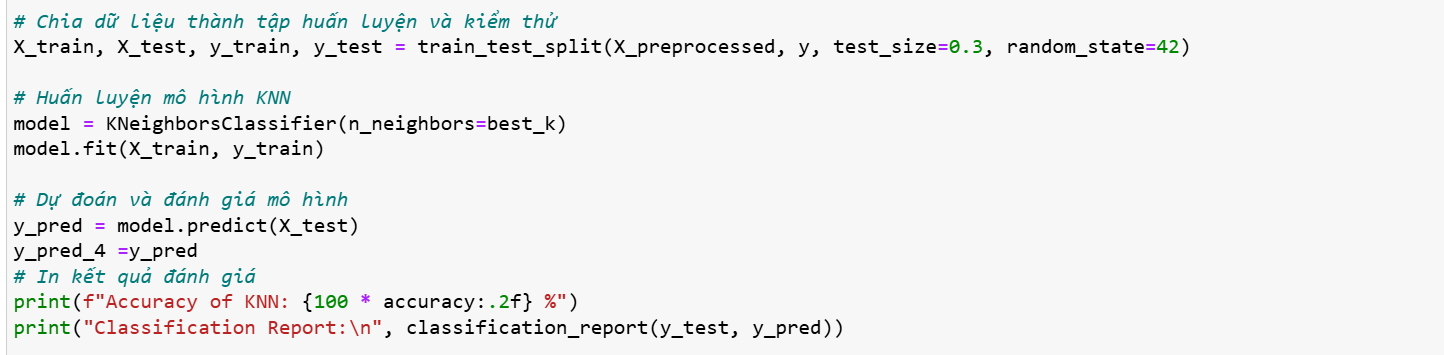
- Cuối cùng in ra giá trị tốt nhất k=12 với cv=6



### 3.3.2. KNN sử dụng thư viện sklearn

- Đọc và xử lí dữ liệu trước khi chạy thuật toán

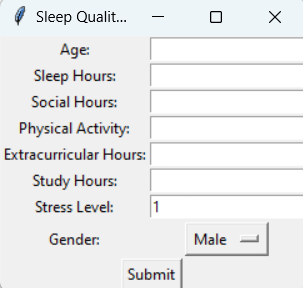
- Huấn luyện mô hình KNN

- Hàm dự đoán chất lượng giấc ngủ khi người dùng nhập vào dữ liệu

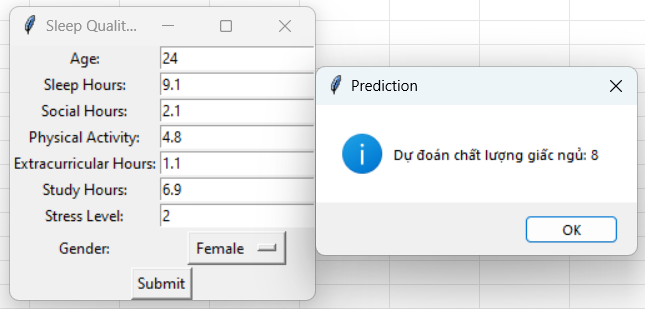
- Tạo giao diện nhập bằng thư viện tkinter



- Giao diện nhập



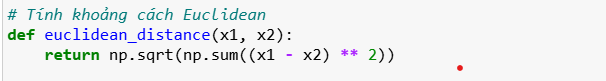
* Giao diện sau khi nhập và in ra kết quả dự đoán:

=> Ta thấy độ chính xác của mô hình dừng lại ở mức khá thấp (xấp xỉ 69%) qua việc kết quả dự đoán có sự chênh lệch so với giá trị thực tế.

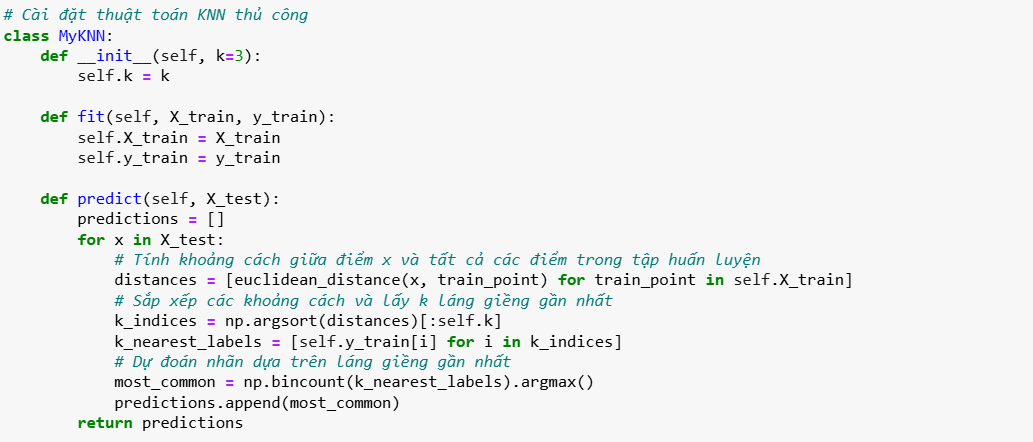


### 3.3.3. KNN không sử dụng thư viện sklearn

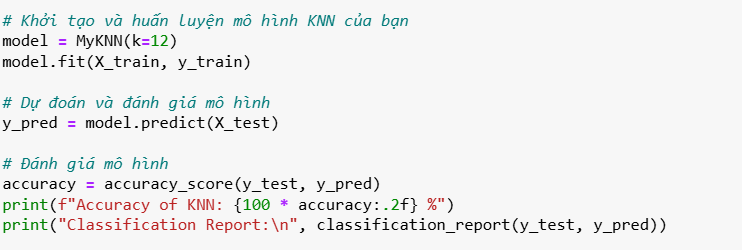
- Cũng tương tự như khi sử dụng thư viện sklearn, ta sẽ đọc dữ liệu, xử lí dữ liệu, chia dữ liệu thành tập train và test. Sau đó ta sẽ xử lí thuật toán bằng các phương pháp toán học. Đầu tiên ta sẽ sử dụng Euclidean để tính khoảng cách giữa điểm **x** (Điểm dữ liệu nhập vào) với các điểm dữ liệu trong tập dữ liệu để lấy ra **k** điểm gần với **x** nhất



- Sau đó ta sẽ cài đặt thủ công thuật toán KNN

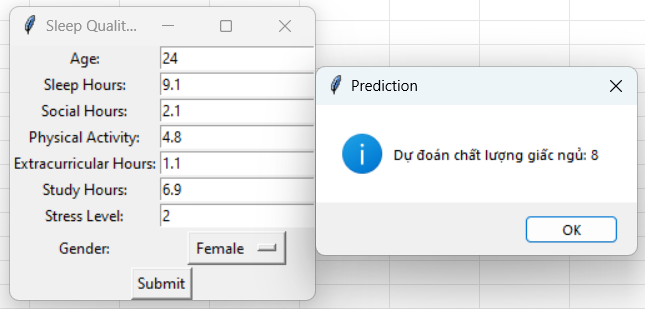


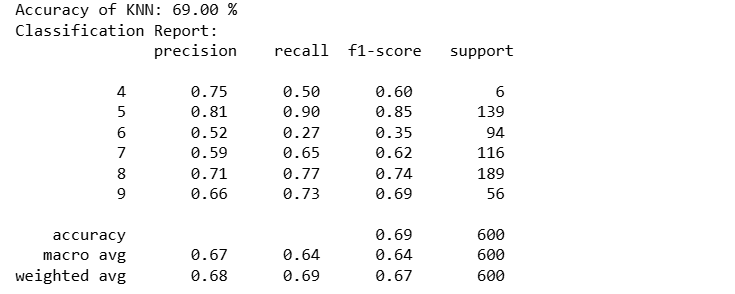
* =>Công đoạn này nhằm mục đích tính toàn bộ khoảng cách từ điểm **x** nhập vào tới tất cả các điểm dữ liệu trong bộ. Sau đó sắp xếp các điểm theo khoảng cách tăng dần (np.argsort(distances)) tiếp đến trả về một mảng chứa các chỉ số của các phần tử trong **distances** theo thứ tự đã sắp xếp rồi lấy **k** = 12 láng giềng gần nhất
* Hàm khởi tạo và huấn luyện mô hình KNN

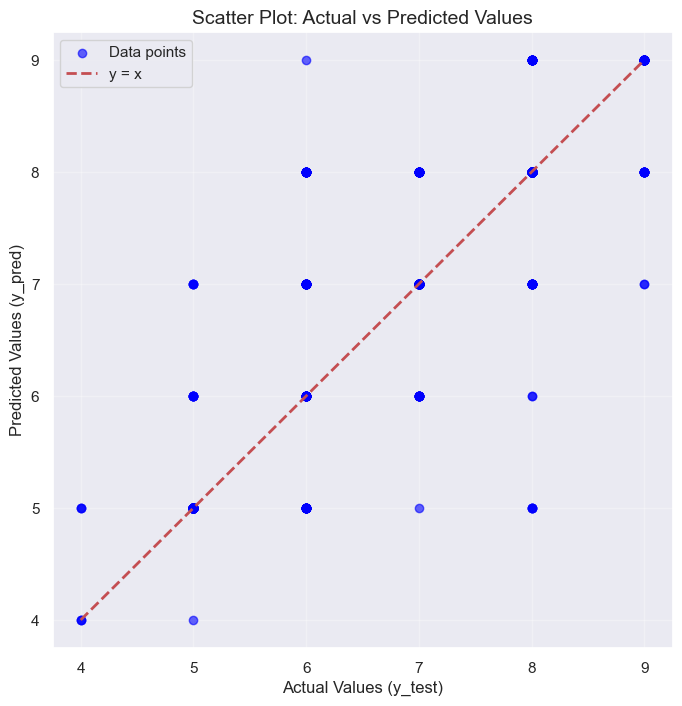


Về phần dự đoán, đánh giá mô hình và tạp giao diện tương tự như mô hình sử dụng sklearn

=> Kêt quả thu được hoàn toàn khớp với mô hình có sử dụng sklearn

=>Kết luận: Mô hình sử dụng KNN tương đối đơn giản, không cần huấn luyện lại mô hình khi thêm mới dữ liệu. Tuy nhiên mô hình mất nhiều thời gian để thực thi, tốn nhiều tài nguyên. Ngoài ra mô hình còn nhạy cảm với nhiễu (Noisy Data) vì thế mà chất lượng mô hình ở mức thấp.

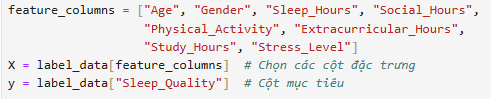


- Những điểm nằm trên đường thẳng là những điểm dự đoán khớp với giá trị thực tế. Những điểm nằm bên ngoài phân bố đều ở các mức tọa độ khác nhau cho thấy mô hình KNN không phù hợp với bộ dữ liệu nhiễu.

## 3.4. Xây dựng Linear Regression

### 3.4.1. Linear Regression sử dụng thư viện sklearn

* Xác định đặc trưng (X) và nhãn (y)



* Chia dữ liệu thành tập huấn luyện và kiểm tra
* Huấn luyện mô hình Linear Regression



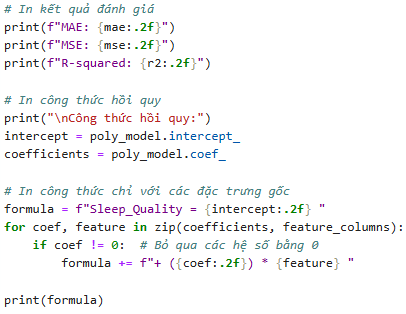
* Dự đoán trên tập kiểm tra



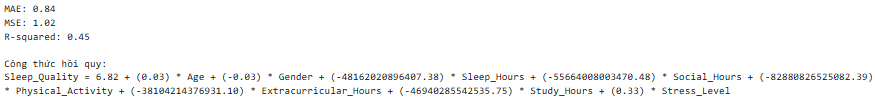
* Đánh giá mô hình

[[1]](#endnote-1)

* In kết quả



* Kết quả sau khi run

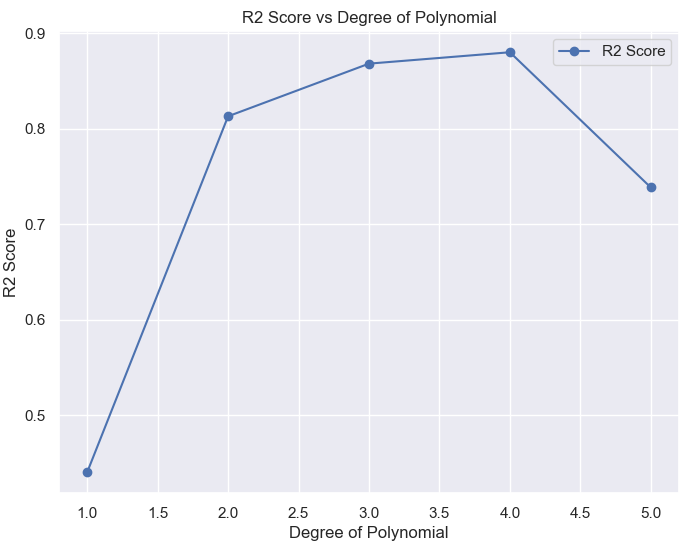


* Nhận thấy sự thiếu hiệu quả nên áp dụng thêm Polynomial Features để cải thiện chất lượng

### 3.4.2. Áp dụng Polynomial Features để cải thiện độ chính xác của thuật toán

* Thử các giá trị degree để có thể cải thiện độ chính xác của thuật toán





Ta có thể thấy được tại Degree = 4 thì độ chính xác của bài toán đạt mức tốt nhất là 88%. Và khi tiếp tục tăng giá trị của Degree thì bắt đầu có dấu hiệu overfiting và chất lượng bắt đầu giảm.

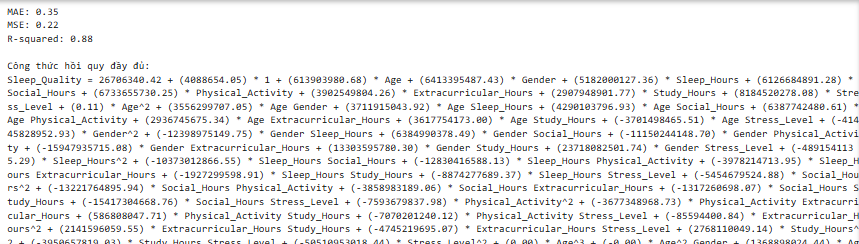
* Sử dụng Polynomial Features với degree = 4 vào bài toán các bước tương tự như trên và chỉ thêm



* Cùng với thêm đoạn tạo tên cho các Features mới



* Sau khi áp dụng thêm Polynomial Features với degree = 4 vào bài toán ta được kết quả sau:

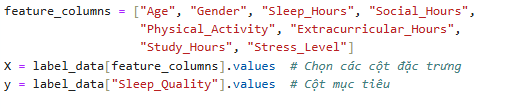


Sau khi áp dụng Polynomial Features thì chất lượng dự báo được cải thiện rõ rệt

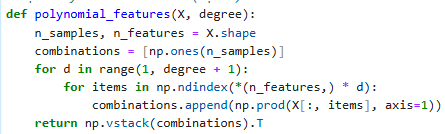
* Có thể thấy được sau khi sử dụng Polynomial Features thì số lượng Features ban đầu từ 8 được tăng lên rất nhiều lên đến 495. Điều này cho thấy rằng thuật toán đã học được các quan hệ phi tuyến tính của các Features ban đầu.

### 3.4.3. Linear Regression không sử dụng thư viện sklearn

* Xác định đặc trưng (X) và nhãn (y) như trên



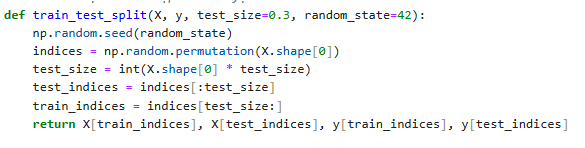
* Tạo Polynomial Features (bậc 4): Hàm polynomial\_features trong đoạn mã trên có nhiệm vụ tạo các **đặc trưng đa thức (polynomial features)** từ một tập dữ liệu đầu vào X, nhằm mục đích mở rộng không gian đặc trưng để mô hình học máy có thể tìm hiểu tốt hơn các mối quan hệ phi tuyến giữa các biến đầu vào.



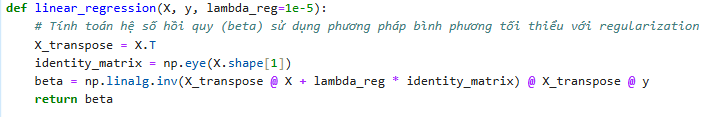
* Thêm cột bias (cột 1s) vào X\_poly



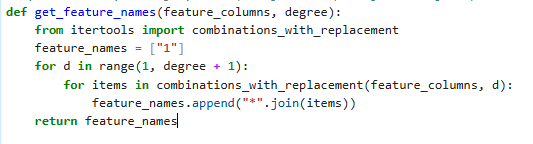
* Hàm train\_test\_split trong đoạn mã trên được sử dụng để chia tập dữ liệu thành hai phần: **tập huấn luyện (training set)** và **tập kiểm tra (test set)**. Đây là một bước phổ biến trong quy trình xây dựng mô hình học máy, nhằm đánh giá hiệu suất của mô hình trên dữ liệu chưa từng được sử dụng để huấn luyện.



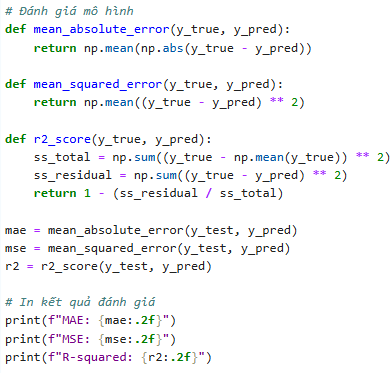
* Hàm linear\_regression trong đoạn mã trên triển khai thuật toán **hồi quy tuyến tính (linear regression)**, với **regularization** (điều chuẩn) nhằm tránh overfitting. Đây là một phương pháp mở rộng của **hồi quy bình phương tối thiểu (ordinary least squares)**, thường được gọi là **Ridge Regression** khi sử dụng điều chuẩn .



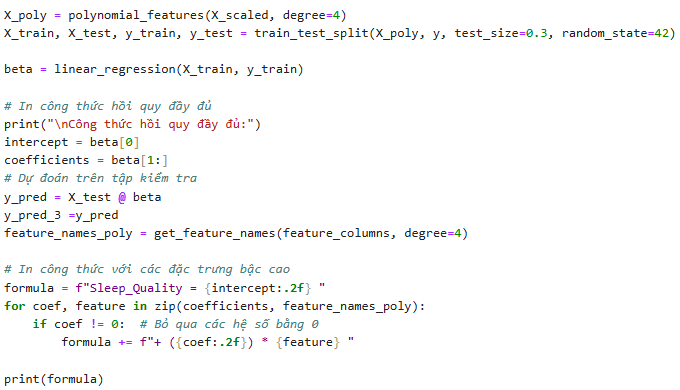
* Hàm get\_feature\_names trong đoạn mã trên được sử dụng để tạo danh sách tên của các đặc trưng đa thức tương ứng với dữ liệu đầu vào khi mở rộng không gian đặc trưng (polynomial feature expansion)

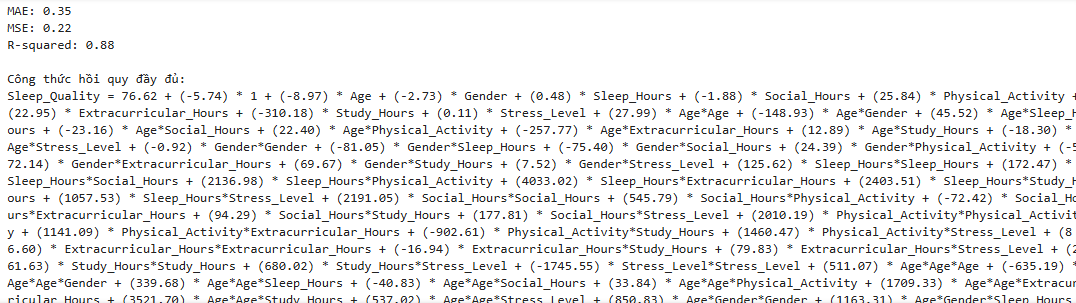


* Đánh giá mô hình



* Hàm mse tính Mean Squared Error (MSE), tức là trung bình của bình phương sai số giữa giá trị thực và giá trị dự đoán. MSE là một chỉ số đánh giá độ chính xác của mô hình hồi quy.
* Công thức: MSE =
* Hàm mae tính Mean Absolute Error (MAE), tức là trung bình của sai số tuyệt đối giữa giá trị thực và giá trị dự đoán. MAE đo lường khoảng cách trung bình giữa các giá trị thực và dự đoán, không tính đến dấu
* Công thức: MAE =
* Hàm r2\_score tính hệ số xác định (R-squared), cho biết phần trăm biến thiên của biến mục tiêu có thể được giải thích bởi mô hình. Chỉ số này dao động từ 0 đến 1, với giá trị càng gần 1 cho thấy mô hình dự đoán càng chính xác.
* Công thức:
  + Trong đó:
    - : Giá trị thực tế
    - : Giá trị dự đoán
    - : Giá trị trung bình của các giá trị thực tế
    - Nếu nghĩa là mô hình dự đoán chính xác toàn bộ
* Gọi hàm chạy chương trình và in ra kết quả

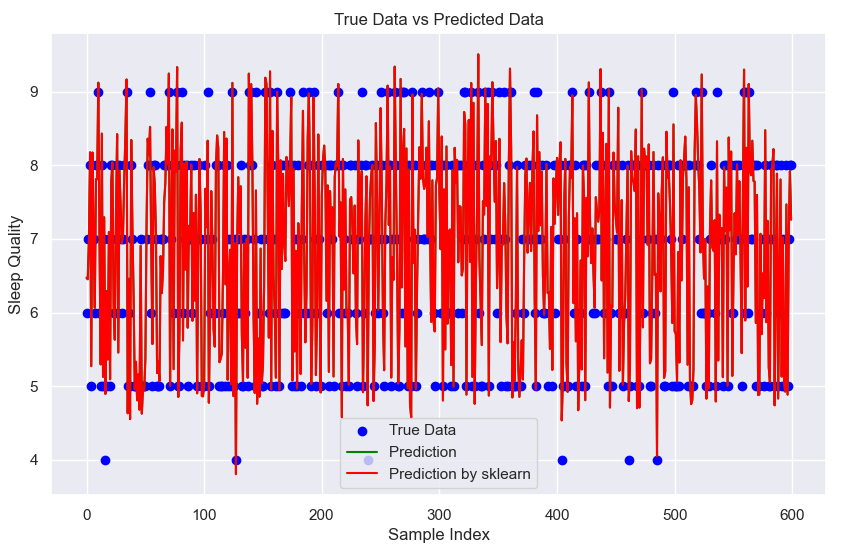


* Sau khi chạy chương trình ra được kết quả
* 

### 3.4.4 So sánh độ chính xác giữa Linear Regression trước khi dùng Polynomial Features và sau khi sử dụng

Chúng ta có thể thấy được sự khác nhau rõ rệt trước là sau khi dùng Polynomial Features. Việc sử dụng cho thấy sự hiệu quả đáng kể tuy nhiên mới chỉ được ở mức tốt chứ chưa được xuất sắc.

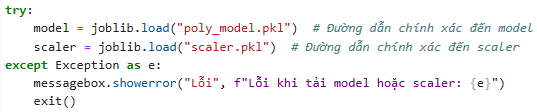
### 3.4.5. So sánh độ chính xác giữa Linear Regression khi dùng thư viện Sklearn và không sử dụng thư viện Sklearn

Có thể thấy được rằng các điểm dự báo giữa việc sử dụng thư viện và không sử dụng gần như là bằng nhau.

Ta có thể dễ hiẻu được điều này bởi vì bản chất là chúng ta để cùng sử dụng 1 công thức nên việc kết quả ra gần bằng nhau là hợp lý.

### 3.4.6. Giao diện dự báo

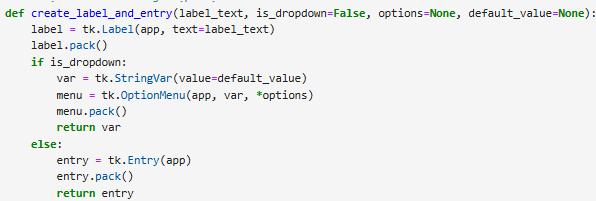
* Tải mô hình và scaler



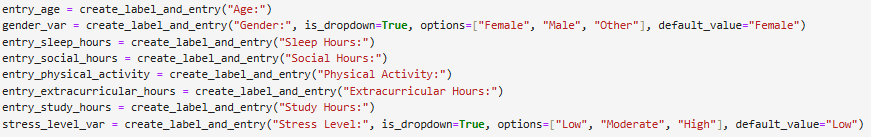
* Tạo cửa sổ ứng dụng



* Hàm tạo nhãn và trường nhập liệu



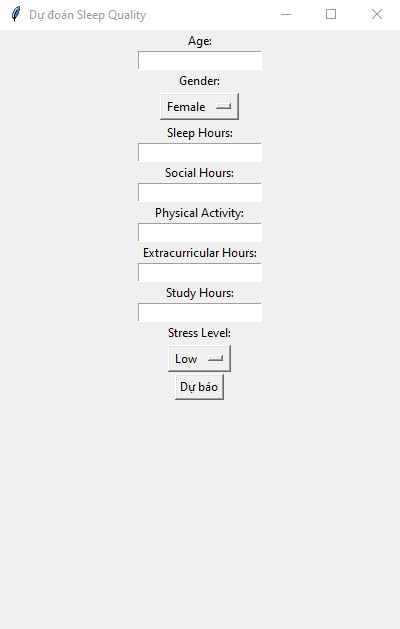
* Tạo các trường nhập liệu và menu



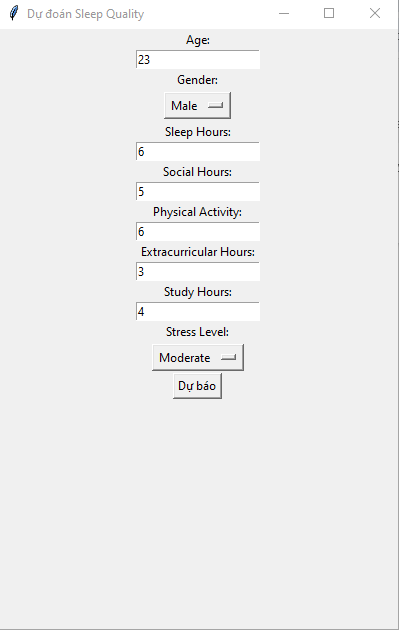
* Tạo nút dự báo



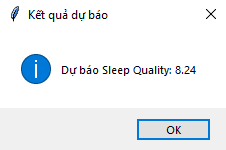
* Khi chạy chương trình giao diện hiện lên được như sau



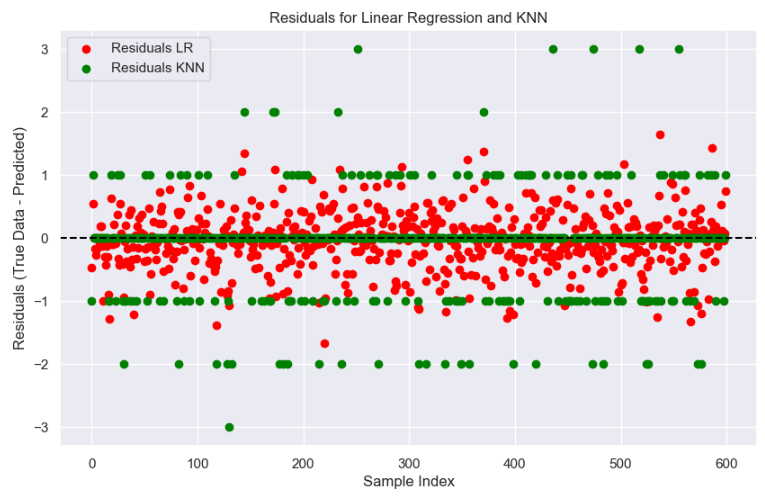
* Nhập dữ liệu vào sẽ ra kết quả dự báo như sau



* Kết quả sẽ hiện ra



### 3.5. So sánh độ chính xác của KNN và Linear Regression



* **Linear Regression (LR)** cho kết quả dự đoán ổn định hơn với residuals phân bố đều quanh trục zero, mặc dù có một số sai số lớn. Điều này cho thấy LR không mắc sai số có tính hệ thống và có thể phù hợp hơn với dữ liệu nếu mối quan hệ giữa các biến là tuyến tính hoặc gần tuyến tính.
* **K-Nearest Neighbors (KNN)** có nhiều residuals lặp lại với giá trị sai số cố định (-1 hoặc 1), cho thấy dự đoán của mô hình này bị ảnh hưởng bởi việc chọn số lượng láng giềng và có thể gặp khó khăn trong việc phản ánh chính xác dữ liệu có sự biến thiên lớn.

Linear Regression thể hiện hiệu suất tốt hơn KNN trong việc dự đoán chất lượng giấc ngủ trên tập dữ liệu hiện tại

# Chương 4: Kết luận và hướng phát triển

## 4.1. Kết Luận

* **Phân loại chất lượng giấc ngủ**: Qua việc áp dụng các thuật toán học máy như Naive Bayes có thể phân loại giấc ngủ sinh viên thành các mức độ như tốt, bình thường và kém dựa trên các yếu tố như thời gian ngủ, mức độ tỉnh táo khi thức dậy, và các yếu tố khác như hoạt động thể chất hoặc tâm lý.
* **Dự báo chất lượng giấc ngủ**: Bằng cách sử dụng mô hình hồi quy (như Linear Regression, KNN có thể dự báo chất lượng giấc ngủ của sinh viên trong tương lai dựa trên các dữ liệu lịch sử về giấc ngủ và các yếu tố tác động.
* **Tác động của yếu tố ngoài**: Qua các phân tích, có thể thấy được ảnh hưởng của các yếu tố bên ngoài như chế độ ăn uống, thói quen sinh hoạt, mức độ stress, và thời gian sử dụng thiết bị điện tử trước khi ngủ đến chất lượng giấc ngủ của sinh viên.

## 4.2. Hướng phát triển

* **Tối ưu hóa mô hình**: Tiến hành tối ưu hóa các mô hình học máy bằng cách sử dụng các kỹ thuật như Grid Search hoặc Random Search để tìm ra các siêu tham số tối ưu. Ngoài ra, có thể thử nghiệm với các mô hình tiên tiến hơn như deep learning (LSTM, CNN).
* **Sử dụng dữ liệu đa dạng**: Tăng cường chất lượng dự báo bằng cách sử dụng thêm các nguồn dữ liệu khác như cảm biến đeo tay, ứng dụng theo dõi giấc ngủ hoặc dữ liệu từ các khảo sát tâm lý.
* **Chuyển sang dự báo thời gian thực**: Phát triển các ứng dụng hoặc hệ thống có thể theo dõi và dự báo chất lượng giấc ngủ của sinh viên trong thời gian thực, từ đó đưa ra khuyến nghị cải thiện chất lượng giấc ngủ.
* **Tích hợp các yếu tố bên ngoài**: Xem xét tích hợp các yếu tố ngoài như thói quen ăn uống, thời gian làm việc, và các yếu tố xã hội để làm phong phú mô hình dự báo, từ đó cải thiện độ chính xác và khả năng áp dụng thực tế của mô hình.
* **Tạo công cụ hỗ trợ sinh viên**: Xây dựng một ứng dụng giúp sinh viên theo dõi chất lượng giấc ngủ, nhận cảnh báo về các yếu tố ảnh hưởng đến giấc ngủ và nhận được các đề xuất cải thiện giấc ngủ dựa trên dữ liệu của họ.

# TÀI LIỆU THAM KHẢO

1. Slide bài giảng của thầy Tạ Quang Chiểu
2. [Blog: Machine learning cơ bản](https://machinelearningcoban.com/)
3. [Kaggle](https://www.kaggle.com/datasets/steve1215rogg/student-lifestyle-dataset)

1. [↑](#endnote-ref-1)