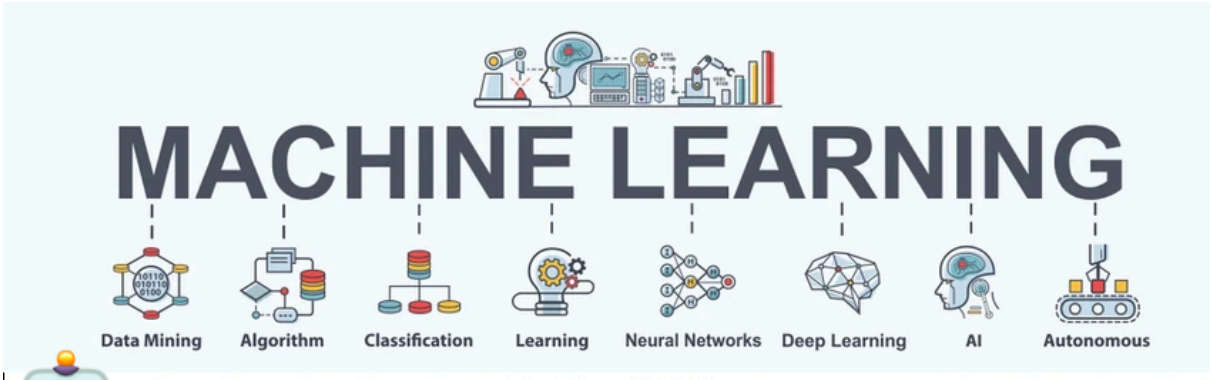


Báo cáo dự án học máy

Đề bài: Thực hành khả năng cài đặt và thử nghiệm mô hình học máy để giải quyết bài toán cụ thể

Đề tài lựa chọn: Dự đoán lượng calo đốt khi tập luyện dựa trên thể trạng và thời gian tập luyện

▼ Phần I: Mở đầu và tóm tắt báo cáo

1. Mở đầu:

Trong thời đại hiện nay, với sự phát triển vượt bậc của công nghệ và trí tuệ nhân tạo, việc ứng dụng các mô hình học máy để dự đoán và phân tích dữ liệu đã trở thành một xu hướng phổ biến và cần thiết trong nhiều lĩnh vực, từ kinh doanh, y tế đến thể thao. Một trong những ứng dụng quan trọng của học máy là trong việc theo dõi và tối ưu hóa sức khỏe và thể chất của con người.

Một trong những vấn đề quan trọng trong việc rèn luyện sức khỏe là hiểu rõ lượng calo được đốt cháy trong quá trình tập luyện. Việc này không chỉ giúp người tập có thể điều chỉnh chế độ tập luyện hợp lý mà còn hỗ trợ trong việc xây dựng một chế độ dinh dưỡng phù hợp, từ đó đạt được hiệu quả tối đa trong việc cải thiện sức khỏe.

Trong bài báo cáo này nhóm em sẽ trình bày về việc thiết kế một bài toán học máy nhằm dự đoán lượng calo bị đốt cháy trong quá trình tập luyện dựa trên thể trạng và thời gian tập luyện của người dùng.

2. Tóm tắt báo cáo:

1. Thu thập và đánh giá dữ liệu:

- Hai bộ dữ liệu chính được sử dụng là dữ liệu về lượng calo đốt cháy và dữ liệu về thông tin thể trạng, thời gian tập của người dùng.
- Sơ bộ và đánh giá sơ lược về dữ liệu thu thập được.

2. Làm sạch và tiền xử lý dữ liệu:

- Dữ liệu liệu thu thập được sẽ được kiểm tra và loại bỏ các lỗi, khuyết thiếu.
- Chuẩn hóa, tổ chức lại dữ liệu để sử dụng cho quá trình huấn luyện mô hình.

3. Phân tích, trực quan hóa dữ liệu và nhận xét:

- Phân tích về các mối tương quan giữa các trường dữ liệu và đánh giá mức độ ảnh hưởng của trường dữ liệu đề kết quả đầu ra của mô hình.
- Trực quan hóa các môi tương quan bằng các biểu đồ.
- Rút ra nhận xét về các đặc điểm của dữ liệu.

4. Huấn Luyện và đánh giá các mô hình:

- Huấn luyện và đánh giá 3 mô hình: LinearRegression, RandomForestRegressor và XGBRegressor sử dụng Pipeline và đánh giá dựa trên các phương pháp đánh giá mô hình: R-squared score, Mean Absolute Error và Cross-validation R-squared score

5. Lựa chọn mô hình và tối ưu hóa:

- Lựa chọn mô hình tối ưu nhất và tối ưu hóa mô hình

6. Thử nghiệm và thiết kế kế giao diện đơn giản cho mô hình:

- Thử nghiệm mô hình sau khi tối ưu hóa trên sample mới
- Thiết kế giao diện đơn giản cho mô hình

▼ Phần II: Thu thập và đánh giá dữ liệu

1. Thu thập dữ liệu:

Dữ liệu được tìm kiếm và lựa chọn từ trang web: https://www.kaggle.com/

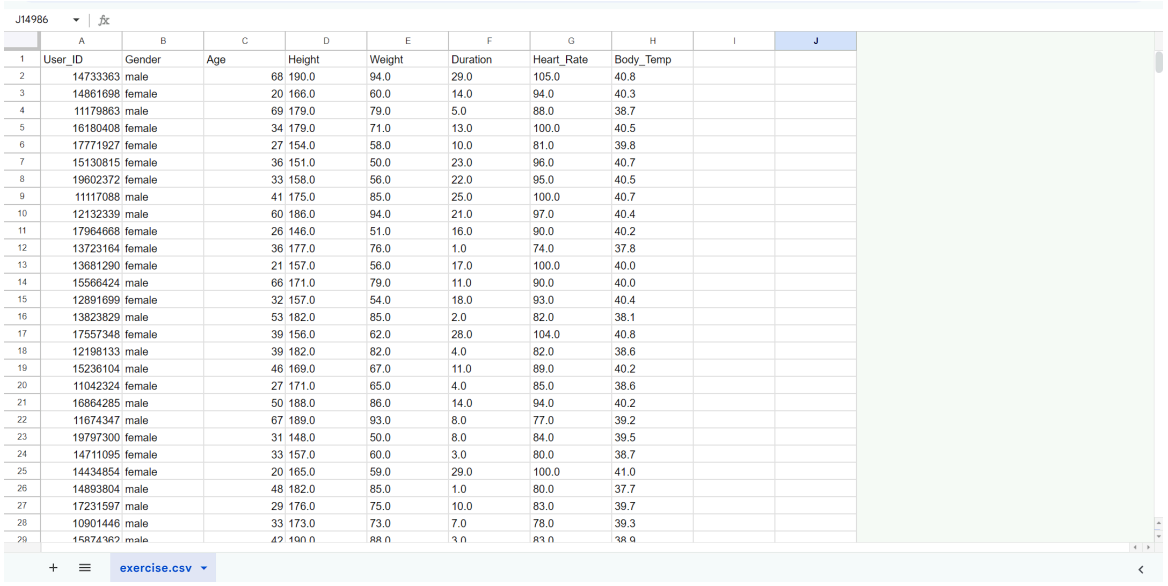
Lý do lựa chọn: **Kaggle l**à một nền tảng trực tuyến cho cộng đồng Machine Learning (ML) và Khoa học dữ liệu. **Kaggle** cho phép người dùng chia sẻ, tìm kiếm các bộ dữ liệu; tìm hiểu và xây dựng các model và chứa đa dạng các bộ dữ liệu chất lượng cao.

2. Sơ lược về dữ liệu thu thập được:

Dữ liệu thu thập được gồm 2 file chính là: calories.csv và exercise.csv.

Tổng quan về file exercise.csv:

- Gồm các trường thông tin:
  + User_ID: Id của người luyện tập
  + Age: Tuổi của người luyện tập
  + Height: Chiều cao của người luyện tập (cm)
  + Weight: Cân nặng của người luyện tập (kg)
  + Duration: Thời gian luyện tập (min)
  + Heart_Rate: Nhịp tim trong quá trình luyện tập (bpm**)**
  + Body_Temp: Nhiệt độ cơ thể trong quá trình luyện tập (**°**C)
- Số lượng: dataset chứa dữ liệu của 15000 người
- Hình ảnh về dữ liệu:



J14986

	A	B	C	D	E	F	G	H	I	J
1	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp		
2	14733363	male	68	190.0	94.0	29.0	105.0	40.8		
3	14861698	female	20	166.0	60.0	14.0	94.0	40.3		
4	11179863	male	69	179.0	79.0	5.0	88.0	38.7		
5	16180408	female	34	179.0	71.0	13.0	100.0	40.5		
6	17771927	female	27	154.0	58.0	10.0	81.0	39.8		
7	15130815	female	36	151.0	50.0	23.0	96.0	40.7		
8	19602372	female	33	158.0	56.0	22.0	95.0	40.5		
9	11117088	male	41	175.0	85.0	25.0	100.0	40.7		
10	12132339	male	60	186.0	94.0	21.0	97.0	40.4		
11	17964668	female	26	146.0	51.0	16.0	90.0	40.2		
12	13723164	female	36	177.0	76.0	1.0	74.0	37.8		
13	13681290	female	21	157.0	56.0	17.0	100.0	40.0		
14	15566424	male	66	171.0	79.0	11.0	90.0	40.0		
15	12891699	female	32	157.0	54.0	18.0	93.0	40.4		
16	13823829	male	53	182.0	85.0	2.0	82.0	38.1		
17	17557348	female	39	156.0	62.0	28.0	104.0	40.8		
18	12198133	male	39	182.0	82.0	4.0	82.0	38.6		
19	15236104	male	46	169.0	67.0	11.0	89.0	40.2		
20	11042324	female	27	171.0	65.0	4.0	85.0	38.6		
21	16864285	male	50	188.0	86.0	14.0	94.0	40.2		
22	11674347	male	67	189.0	93.0	8.0	77.0	39.2		
23	19797300	female	31	148.0	50.0	8.0	84.0	39.5		
24	14711095	female	33	157.0	60.0	3.0	80.0	38.7		
25	14434854	female	20	165.0	59.0	29.0	100.0	41.0		
26	14893804	male	48	182.0	85.0	1.0	80.0	37.7		
27	17231597	male	29	176.0	75.0	10.0	83.0	39.7		
28	10901446	male	33	173.0	73.0	7.0	78.0	39.3		
29	15874362	male	42	190.0	88.0	3.0	83.0	38.9		

exercise.csv

Tổng quan về file calories.csv:

- Gồm các trường thông tin:

  + User_ID: Id của người luyện tập

  + Calories: Số calo đốt được (Kcal)

- Số lượng: dataset chứa dữ liệu về số calo đốt được của 15000 người phía trên
- Hình ảnh về dữ liệu:

	A	B
1	User_ID	Calories
2	14733363	231
3	14861698	66
4	11179863	26
5	16180408	71
6	17771927	35
7	15130815	123
8	19602372	112
9	11117088	143
10	12132339	134
11	17964668	72
12	13723164	3
13	13681290	92
14	15566424	58
15	12891699	88
16	13823829	7
17	17557348	170
18	12198133	11
19	15236104	43

▼ Phần III: Làm sạch và tiền xử lý dữ liệu

1. Các thư viện sử dụng:

Các thư viện sử dụng:

- Thư viện xử lý dữ liệu: Pandas và Numpy
- Thư viện trực quan hóa dữ liệu: Seaborn và Matplotlib

2. Làm sạch dữ liệu và tiền xử lý dữ liệu:

1. Gộp 2 file dữ liệu:

Đầu tiên ta cần merge dữ liệu của 2 file dữ liệu thành một dựa theo id của người luyện tập:

```
In [5]: calories = pd.read_csv('calories.csv')
        exercise = pd.read_csv('exercise.csv')

In [6]: calories.head(1)
```

Out[6]:

	User_ID	Calories
0	14733363	231.0

```
In [7]: exercise.head(1)
```

Out[7]:

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	14733363	male	68	190.0	94.0	29.0	105.0	40.8

```
In [8]: data = pd.merge(calories,exercise,on='User_ID')
```

Sau khi gộp 2 file ta có dữ liệu mới như sau:

```
In [10]: data.head()
```

Out[10]:

	User_ID	Calories	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	14733363	231.0	male	68	190.0	94.0	29.0	105.0	40.8
1	14861698	66.0	female	20	166.0	60.0	14.0	94.0	40.3
2	11179863	26.0	male	69	179.0	79.0	5.0	88.0	38.7
3	16180408	71.0	female	34	179.0	71.0	13.0	100.0	40.5
4	17771927	35.0	female	27	154.0	58.0	10.0	81.0	39.8

2. Kiểm tra tra các khuyết thiếu và lỗi dữ liệu:

Trước tiên kiểm tra các trường dữ liệu rỗng của data:

```
In [38]: data.isnull().any()

Out[38]: User_ID       False
         Calories      False
         Gender        False
         Age           False
         Height        False
         Weight        False
         Duration      False
         Heart_Rate    False
         Body_Temp     False
         dtype: bool
```

🔎 Thông tin trả về cho thấy dữ liệu không có giá trị null nào.

Tiếp theo ta sẽ loại bỏ các hàng và các cột dữ liệu bị trùng lặp để đảm bảo dữ liệu không bị trùng nhau:

In [26]:
```
data = data.drop_duplicates()
data
```

Out[26]:

	User_ID	Calories	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	14733363	231.0	male	68	190.0	94.0	29.0	105.0	40.8
1	14861698	66.0	female	20	166.0	60.0	14.0	94.0	40.3
2	11179863	26.0	male	69	179.0	79.0	5.0	88.0	38.7
3	16180408	71.0	female	34	179.0	71.0	13.0	100.0	40.5
4	17771927	35.0	female	27	154.0	58.0	10.0	81.0	39.8
...	...	...	...	...	...	...	...	...	...
14995	15644082	45.0	female	20	193.0	86.0	11.0	92.0	40.4
14996	17212577	23.0	female	27	165.0	65.0	6.0	85.0	39.2
14997	17271188	75.0	female	43	159.0	58.0	16.0	90.0	40.1
14998	18643037	11.0	male	78	193.0	97.0	2.0	84.0	38.3
14999	11751526	98.0	male	63	173.0	79.0	18.0	92.0	40.5

15000 rows × 9 columns

🔎 Sau khi thực hiện loại bỏ các hàng và cột trùng lặp, cuối cùng còn lại 15000 hàng và 9 cột (tương tự ban đầu)
⇒ Dữ liệu không có hàng, cột trùng lặp

3. Loại bỏ trường thông tin không cần thiết:

Trong bộ dữ liệu có trường “User_ID”, trường này tồn tại để đối chiếu người người luyện tập trên 2 file dữ liệu. Sau khi đã gộp 2 file dữ liệu, trường này không còn giá trị sử dụng nên ta sẽ loại bỏ:

In [44]:
```
data = data.drop(columns=['User_ID'], axis=1)
data
```

Out[44]:

	Calories	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	231.0	male	68	190.0	94.0	29.0	105.0	40.8
1	66.0	female	20	166.0	60.0	14.0	94.0	40.3
2	26.0	male	69	179.0	79.0	5.0	88.0	38.7
3	71.0	female	34	179.0	71.0	13.0	100.0	40.5
4	35.0	female	27	154.0	58.0	10.0	81.0	39.8
...	...	...	...	...	...	...	...	...
14995	45.0	female	20	193.0	86.0	11.0	92.0	40.4
14996	23.0	female	27	165.0	65.0	6.0	85.0	39.2
14997	75.0	female	43	159.0	58.0	16.0	90.0	40.1
14998	11.0	male	78	193.0	97.0	2.0	84.0	38.3
14999	98.0	male	63	173.0	79.0	18.0	92.0	40.5

15000 rows × 8 columns

4. Tiền xử lý dữ liệu:

Đầu tiên với cột “Gender” trong dữ liệu đang có dạng object và nhận 2 giá trị “male” và “female”, ta cần chuẩn hóa giá trị “male” thành “0” và “female” thành “1”

```
In [153]: data['Gender'].replace({'male': 0, 'female': 1}, inplace=True)
          data
```

Out[153]:

	User_ID	Calories	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	14733363	231.0	0	68	190.0	94.0	29.0	105.0	40.8
1	14861698	66.0	1	20	166.0	60.0	14.0	94.0	40.3
2	11179863	26.0	0	69	179.0	79.0	5.0	88.0	38.7
3	16180408	71.0	1	34	179.0	71.0	13.0	100.0	40.5
4	17771927	35.0	1	27	154.0	58.0	10.0	81.0	39.8
...	...	...	...	...	...	...	...	...	...
14995	15644082	45.0	1	20	193.0	86.0	11.0	92.0	40.4
14996	17212577	23.0	1	27	165.0	65.0	6.0	85.0	39.2
14997	17271188	75.0	1	43	159.0	58.0	16.0	90.0	40.1
14998	18643037	11.0	0	78	193.0	97.0	2.0	84.0	38.3
14999	11751526	98.0	0	63	173.0	79.0	18.0	92.0	40.5

15000 rows × 9 columns

Để huấn luyện mô hình, ta sẽ chia tập dữ liệu thành 2 phần X và Y tương ứng với đầu vào và đầu ra của mô hình.

```
In [156]: X = data.data = data.drop(columns=['Calories'], axis=1)
          Y = data['Calories']
```

```
In [157]: X.head()
```

Out[157]:

	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	0	68	190.0	94.0	29.0	105.0	40.8
1	1	20	166.0	60.0	14.0	94.0	40.3
2	0	69	179.0	79.0	5.0	88.0	38.7
3	1	34	179.0	71.0	13.0	100.0	40.5
4	1	27	154.0	58.0	10.0	81.0	39.8

```
In [158]: Y.head()
```

```
Out[158]: 0    231.0
          1     66.0
          2     26.0
          3     71.0
          4     35.0
          Name: Calories, dtype: float64
```

Tiếp theo ta sẽ chia dữ liệu thành 2 phần gồm:

- X_train và Y_train: Đây các tập dữ liệu huấn liệu
- X_test và Y_test: Đây các tập dữ liệu kiểm tra

Để chia dữ liệu ta có thể sử dụng hàm **train_test_split** từ **module sklearn.model_selection**, đặt giá trị test_size=0.2 để chia thành 80% dữ liệu huấn luyện và 20% dữ liệu kiểm tra, đặt giá trị là random_state=42 để đảm bảo cho việc phân chia ngẫu nhiên, đảm bảo kết quả có thể tái lặp được.

```
In [52]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

         # Hiển thị kích thước của các tập dữ liệu
         print("Size of datasets:")
         print("X_train:", X_train.shape)
         print("Y_train:", Y_train.shape)
         print("X_test:", X_test.shape)
         print("Y_test:", Y_test.shape)
```

```
Size of datasets:
X_train: (12000, 7)
Y_train: (12000,)
X_test: (3000, 7)
Y_test: (3000,)
```

🔎 Sau khi chia kích thước của các tập dữ liệu như sau:
- X_train: 12000 hàng và 7 cột
- Y_train: 12000 hàng và 1 cột
- X_test: 3000 hàng và 7 cột
- Y_train: 3000 hàng và 1 cột

▼ Phần IV: Phân tích, trực quan hóa dữ liệu và nhận xét

Để phân tích được chính xác nhất ta sẽ đặt data_analyst là dữ liệu khi chưa loại bỏ trường “User_ID”

1. Các thông tin và thống kê cơ bản của dữ liệu:

Các thông tin cơ bản của dữ liệu:

```
In [160]: # Display general information about the dataset
          print("Dataset information:")
          print(data_analyst.info())
          print("\n")

Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   User_ID     15000 non-null  int64
 1   Calories    15000 non-null  float64
 2   Gender      15000 non-null  int64
 3   Age         15000 non-null  int64
 4   Height      15000 non-null  float64
 5   Weight      15000 non-null  float64
 6   Duration    15000 non-null  float64
 7   Heart_Rate  15000 non-null  float64
 8   Body_Temp   15000 non-null  float64
dtypes: float64(6), int64(3)
memory usage: 1.0 MB
None
```

🔎 Nhận xét: Có thể thấy dữ liệu của các cột đều đủ thông tin của 15000 người, không có dữ liệu null và các thông tin về kiểu dữ liệu của mỗi trường

Các thống kê cơ bản về dữ liệu:

In [161]:
```
# Display basic statiastics for the columns
print("Basic Statistics for columns:")
print(data_analyst.describe())
print("\n")
```

```
Basic Statistics for columns:
            User_ID      Calories        Gender           Age        Height  \
count  1.500000e+04  15000.000000  15000.000000  15000.000000  15000.000000
mean   1.497736e+07     89.539533      0.503533     42.789800    174.465133
std    2.872851e+06     62.456978      0.500004     16.980264     14.258114
min    1.000116e+07      1.000000      0.000000     20.000000    123.000000
25%    1.247419e+07     35.000000      0.000000     28.000000    164.000000
50%    1.499728e+07     79.000000      1.000000     39.000000    175.000000
75%    1.744928e+07    138.000000      1.000000     56.000000    185.000000
max    1.999965e+07    314.000000      1.000000     79.000000    222.000000

             Weight      Duration    Heart_Rate     Body_Temp
count  15000.000000  15000.000000  15000.000000  15000.000000
mean      74.966867     15.530600     95.518533     40.025453
std       15.035657      8.319203      9.583328      0.779230
min       36.000000      1.000000     67.000000     37.100000
25%       63.000000      8.000000     88.000000     39.600000
50%       74.000000     16.000000     96.000000     40.200000
75%       87.000000     23.000000    103.000000     40.600000
max      132.000000     30.000000    128.000000     41.500000
```

Đoạn trên thực hiện việc hiển thị các thống kê cơ bản cho các cột trong dữ liệu và đưa ra kết quả mô tả các đặc trưng của tập dữ liệu. Cụ thể, cung cấp các thông tin về số lượng, giá trị trung bình, độ lệch chuẩn, giá trị nhỏ nhất, phần trăm (25%, 50%, 75%) và giá trị lớn nhất cho từng cột trong tập dữ liệu.

🔎 Nhận xét:

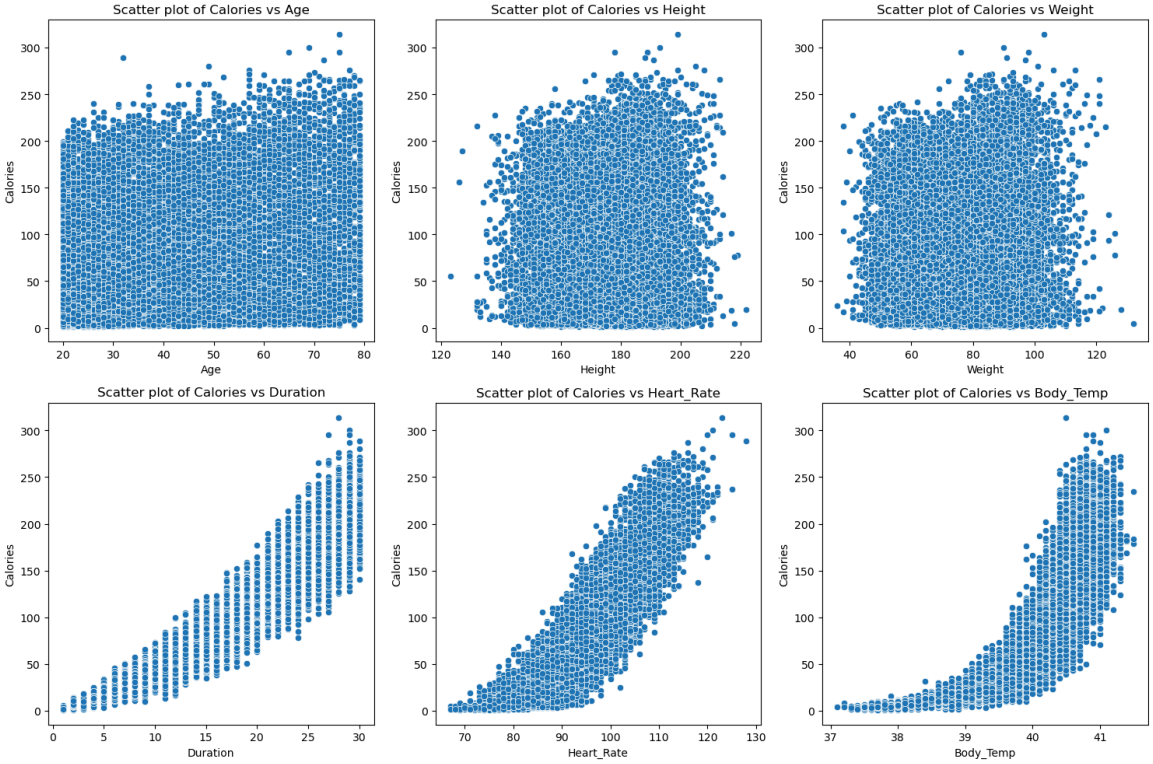
- **User_ID**:
  - mean và std thể hiện giá trị trung bình và độ lệch chuẩn của ID người dùng.
  - ID người dùng là giá trị duy nhất và không ảnh hưởng đến việc phân tích dữ liệu.
- **Calories**:
  - Lượng calo đốt cháy dao động từ 1 đến 314.
  - Giá trị trung bình là 89.54 và độ lệch chuẩn là 62.46, cho thấy sự phân tán lớn trong lượng calo đốt cháy.
- **Gender (Giới tính):**
  - Glới tính khá cân bằng về số lượng nam và nữ.
- **Age (Tuổi)**:
  - Tuổi người dùng dao động từ 20 đến 79, với trung bình là 42.79 và độ lệch chuẩn là 16.98.
  - Phân phối tuổi khá đa dạng, bao gồm cả người trẻ và người già.
- **Height (Chiều cao)**:
  - Chiều cao dao động từ 123 cm đến 222 cm, với trung bình là 174.47 cm.
  - Độ lệch chuẩn là 14.26, cho thấy chiều cao người dùng có sự phân tán khá rộng.
- **Weight (Cân nặng)**:
  - Cân nặng dao động từ 36 kg đến 132 kg, với trung bình là 74.97 kg.
  - Độ lệch chuẩn là 15.04, cho thấy sự đa dạng trong cân nặng của người dùng.
- **Duration (Thời gian tập luyện)**:
  - Thời gian tập luyện dao động từ 1 phút đến 30 phút, với trung bình là 15.53 phút.

    - Độ lệch chuẩn là 8.32, cho thấy có sự khác biệt lớn trong thời gian tập luyện của người dùng.
- **Heart_Rate (Nhịp tim)**:
    - Nhịp tim dao động từ 67 bpm đến 128 bpm, với trung bình là 95.52 bpm.
    - Độ lệch chuẩn là 9.58, cho thấy sự phân tán nhịp tim trong quá trình tập luyện.
- **Body_Temp (Nhiệt độ cơ thể)**:
    - Nhiệt độ cơ thể dao động từ 37.1°C đến 41.5°C, với trung bình là 40.03°C.
    - Độ lệch chuẩn là 0.78, cho thấy sự phân tán nhỏ hơn so với các thuộc tính khác.

2. Phân tích về mức độ ảnh hưởng của các đặc trưng với mức calo đốt:

Ta sẽ sử dụng các biểu đồ sau để phân tích các mối liên hệ giữa các đặc trưng và đặc biệt là mức độ ảnh hưởng của các đặc trưng thể trạng với mức độ calo đốt được.

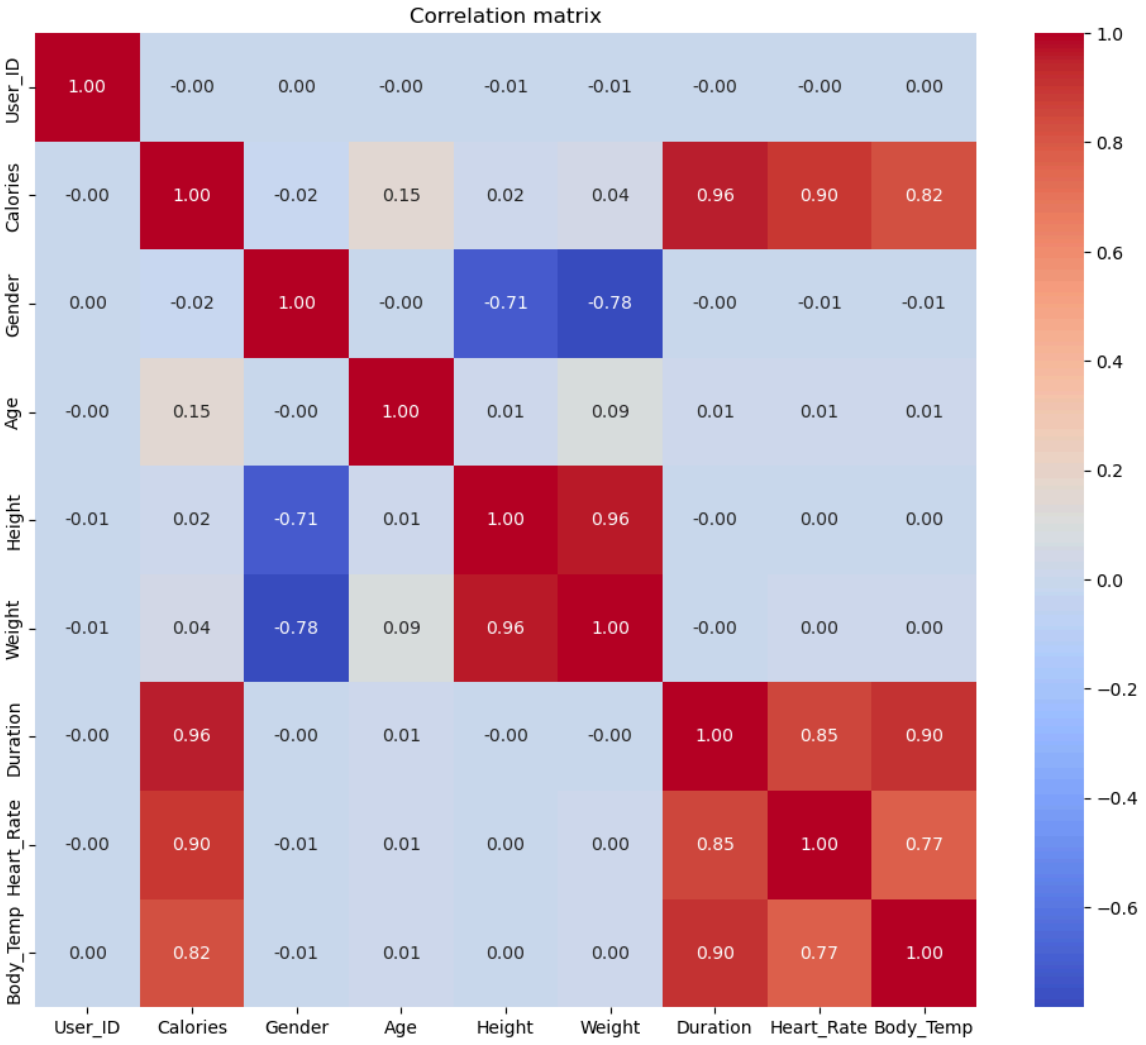
1. **Biểu đồ phân tán (scatter plot) giữa các đặc trưng số và "Calories"**: Biểu đồ này sẽ giúp ta thấy được mối quan hệ tuyến tính hoặc phi tuyến tính giữa các đặc trưng với "Calories".


Scatter plot of Calories vs Age
Calories
Age
Scatter plot of Calories vs Height
Calories
Height
Scatter plot of Calories vs Weight
Calories
Weight
Scatter plot of Calories vs Duration
Calories
Duration
Scatter plot of Calories vs Heart_Rate
Calories
Heart_Rate
Scatter plot of Calories vs Body_Temp
Calories
Body_Temp

🔎 Nhận xét:

1. **Scatter plot of Calories vs Age**:
    - Mối quan hệ giữa tuổi và lượng calo tiêu thụ không rõ ràng và có vẻ như phân bố đều.
    - Không có xu hướng tăng hay giảm rõ ràng khi tuổi tăng.
2. **Scatter plot of Calories vs Height**:
    - Tương tự như Age, chiều cao cũng không có mối quan hệ rõ ràng với lượng calo tiêu thụ.
    - Dữ liệu phân tán khá rộng và không có xu hướng cụ thể.
3. **Scatter plot of Calories vs Weight**:
    - Mối quan hệ giữa cân nặng và lượng calo tiêu thụ cũng không rõ ràng.
    - Dữ liệu phân tán đều mà không có xu hướng rõ rệt.
4. **Scatter plot of Calories vs Duration**:
    - Có một mối quan hệ rõ ràng và tuyến tính giữa thời gian hoạt động và lượng calo tiêu thụ.
    - Khi thời gian hoạt động tăng, lượng calo tiêu thụ cũng tăng lên.
5. **Scatter plot of Calories vs Heart_Rate**:
    - Mối quan hệ mạnh mẽ giữa nhịp tim và lượng calo tiêu thụ. Khi nhịp tim tăng, lượng calo tiêu thụ cũng tăng theo.
    - Mối quan hệ này dường như là phi tuyến tính, với một xu hướng tăng dần rõ ràng.
6. **Scatter plot of Calories vs Body_Temp**:
    - Mối quan hệ giữa nhiệt độ cơ thể và lượng calo tiêu thụ cũng rõ ràng.
    - Khi nhiệt độ cơ thể tăng, lượng calo tiêu thụ cũng tăng theo, có vẻ như đây cũng là một mối quan hệ phi tuyến tính.

2. **Biểu đồ nhiệt (heatmap):** Biểu đồ này sẽ giúp ta thấy được mối tương quan giữa các đặc trưng với nhau:



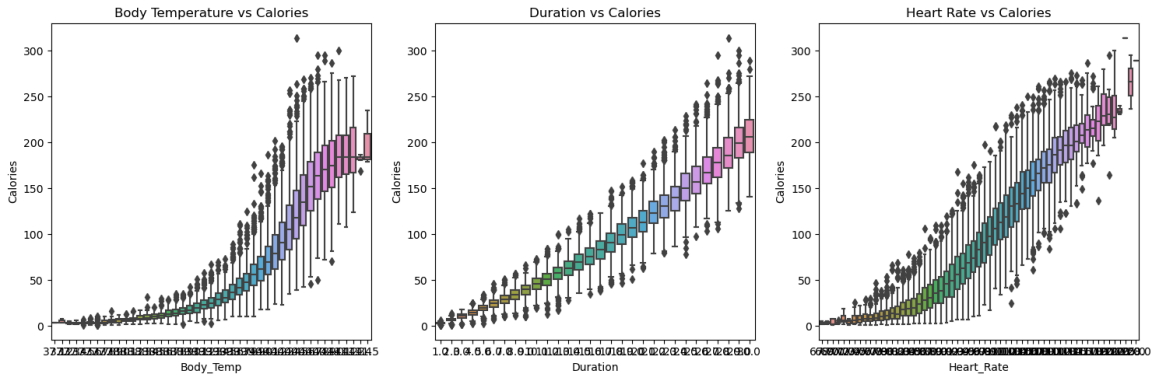
🔎 Nhận xét:

1. **Calories**:
    - Có tương quan mạnh và dương với **Duration** (0.96), **Heart_Rate** (0.90), và **Body_Temp** (0.82).
    - Điều này xác nhận những quan sát từ biểu đồ phân tán rằng thời gian hoạt động, nhịp tim, và nhiệt độ cơ thể có mối quan hệ mạnh mẽ với lượng calo tiêu thụ.
2. **Gender**:
    - Có tương quan âm và khá mạnh với **Height** (-0.71) và **Weight** (-0.78).
    - Điều này cho thấy có sự khác biệt rõ rệt về chiều cao và cân nặng giữa hai giới tính trong dữ liệu.
3. **Age**:
    - Không có mối tương quan mạnh với các đặc trưng khác, bao gồm cả **Calories** (0.15).
    - Điều này cũng phù hợp với nhận xét từ biểu đồ phân tán.
4. **Height** và **Weight**:
    - Có tương quan rất mạnh với nhau (0.96), cho thấy người cao thường cũng có cân nặng lớn hơn.
    - Tương quan mạnh với **Gender**, như đã nhận xét ở trên.
5. **Duration**:
    - Có tương quan mạnh với **Heart_Rate** (0.85) và **Body_Temp** (0.90), điều này hợp lý vì thời gian hoạt động dài hơn thường dẫn đến tăng nhịp tim và nhiệt độ cơ thể.
6. **Heart_Rate**:
    - Có tương quan mạnh với **Duration** (0.85) và **Body_Temp** (0.77).
    - Nhịp tim tăng thường đi kèm với nhiệt độ cơ thể cao hơn khi hoạt động thể chất.

> ⇒ **Kết luận:**
>
> - Các đặc trưng **Duration**, **Heart_Rate**, và **Body_Temp** có mối tương quan mạnh mẽ và tích cực với lượng calo tiêu thụ (**Calories**), làm rõ rằng các yếu tố liên quan đến hoạt động thể chất là những yếu tố chính ảnh hưởng đến lượng calo tiêu thụ.
> - **Gender** có mối tương quan âm rõ rệt với **Height** và **Weight**, phản ánh sự khác biệt giữa các giới tính về chiều cao và cân nặng.
> - **Age**, **Height**,**Gender** và **Weight** không có mối tương quan mạnh với lượng calo tiêu thụ, cho thấy rằng các đặc trưng nhân khẩu học này ít ảnh hưởng đến lượng calo tiêu thụ so với các đặc trưng liên quan đến hoạt động thể chất.

3. Nhận xét về 3 đặc trưng quan trọng nhất Duration, Heart_Rate, và Body_Temp:

Ta sẽ sử dụng biểu đồ hộp (box plots) để nhận xét mức độ ảnh hưởng giữa "Calories" và các đặc trưng **Body_Temp**, **Duration**, và **Heart_Rate:**



🔎 Nhận xét:

1. **Body Temperature vs Calories**:
    - Có một mối quan hệ rõ ràng giữa nhiệt độ cơ thể và lượng calo tiêu thụ. Khi nhiệt độ cơ thể tăng, lượng calo tiêu thụ cũng tăng lên.
    - Mối quan hệ này dường như là phi tuyến tính, với một sự tăng trưởng mạnh ở các giá trị cao của nhiệt độ cơ thể (khoảng từ 40 độ trở lên).
    - Sự phân bố của dữ liệu khá chặt chẽ, với các hộp hộp càng về sau càng dãn ra, cho thấy rằng sự biến thiên lượng calo tiêu thụ tăng dần khi nhiệt độ cơ thể tăng.
2. **Duration vs Calories**:
    - Có một mối quan hệ tuyến tính mạnh mẽ giữa thời gian hoạt động và lượng calo tiêu thụ. Khi thời gian hoạt động tăng, lượng calo tiêu thụ cũng tăng lên.
    - Các hộp hộp biểu diễn dữ liệu cũng cho thấy rằng sự biến thiên lượng calo tiêu thụ tăng dần khi thời gian hoạt động tăng, đặc biệt rõ rệt ở các khoảng thời gian dài hơn.
    - Sự phân bố của dữ liệu khá đồng đều và chặt chẽ, đặc biệt ở các khoảng thời gian hoạt động ngắn hơn.
3. **Heart Rate vs Calories**:
    - Mối quan hệ giữa nhịp tim và lượng calo tiêu thụ cũng rất rõ ràng và có dạng phi tuyến tính. Khi nhịp tim tăng, lượng calo tiêu thụ cũng tăng lên.
    - Các hộp cho thấy sự phân bố dữ liệu tương đối chặt chẽ ở các giá trị nhịp tim thấp, nhưng biến thiên ngày càng lớn ở các giá trị nhịp tim cao hơn.
    - Mối quan hệ phi tuyến tính này thể hiện rằng nhịp tim cao hơn dẫn đến sự gia tăng mạnh mẽ hơn trong lượng calo tiêu thụ.

> ## ⇒ Kết luận:

> - Cả ba đặc trưng **Body_Temp**, **Duration**, và **Heart_Rate** đều có mối quan hệ mạnh mẽ và dương với lượng calo tiêu thụ, nhưng mỗi đặc trưng biểu hiện mối quan hệ này theo cách riêng.
> - **Duration** có mối quan hệ gần như tuyến tính với "Calories", trong khi **Body_Temp** và **Heart_Rate** có mối quan hệ phi tuyến tính, với sự gia tăng mạnh mẽ hơn ở các giá trị cao.
> - Các biểu đồ hộp cho thấy rằng sự biến thiên lượng calo tiêu thụ không đồng đều, với sự biến thiên lớn hơn khi các giá trị của các đặc trưng tăng lên, đặc biệt rõ rệt ở **Heart_Rate** và **Body_Temp**.
> - Những nhận xét này nhấn mạnh rằng các yếu tố liên quan đến hoạt động thể chất (thời gian hoạt động, nhịp tim, và nhiệt độ cơ thể) có tác động đáng kể đến lượng calo tiêu thụ.

▼ Phần V: Huấn luyện và đánh giá các mô hình

I. Huấn luyện các mô hình

Để bắt đầu huấn luyện các mô hình ta sẽ sử dụng **Pipeline** một công cụ trong Scikit-learn để tự động thực hiện các bước tiền xử lý dữ liệu và huấn luyện mô hình theo một trình tự nhất định.

- Quy trình bao gồm hai bước:
    1. Bước đầu tiên là **preprocessor**, được định nghĩa là một **ColumnTransformer**. Bước này sẽ thực hiện tiền xử lý dữ liệu, bao gồm chuẩn hóa các cột số và giữ nguyên các cột khác.

2. Bước thứ hai là chọn **model.** Bước này sẽ sử dụng mô hình hồi quy tuyến tính để học từ dữ liệu đã được xử lý.

Đầu tiên ta sẽ định nghĩa **preprocessor:**

```
preprocessor = ColumnTransformer(transformers=[
    ('num',StandardScaler(),['Age','Height','Weight','Duration','Heart_Rate','Bo
],remainder='passthrough')
```

- Các cột 'Age', 'Height', 'Weight', 'Duration', 'Heart_Rate', và 'Body_Temp' được chỉ định để được chuẩn hóa bằng cách sử dụng **StandardScaler()**, giúp đưa các giá trị trong các cột này về cùng một phạm vi và giúp mô hình học tốt hơn.
- Đối với các cột khác, chúng ta muốn giữ nguyên giá trị của chúng, nên sử dụng **remainder='passthrough'.** Điều này đảm bảo rằng các cột không được chỉ định sẽ được chuyển tiếp mà không thay đổi.

Để biểu diễn quá trình huấn luyện bằng biểu đồ ta sử dụng:

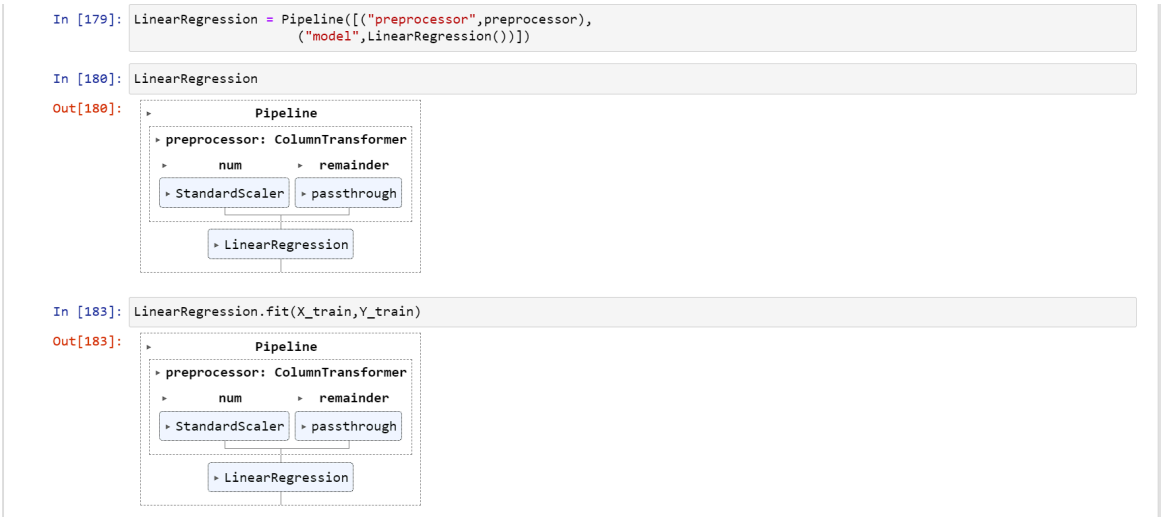
```
set_config(display='diagram')
```

1. LinearRegression:

Lý do lựa chọn:

- **Đơn giản và hiệu quả:** Linear Regression là một trong những mô hình học máy đơn giản nhất và dễ hiểu. Trong trường hợp dữ liệu không có sự phức tạp đặc biệt và không có yếu tố phi tuyến, Linear Regression có thể cho kết quả tốt.
- **Tính diễn giải cao:** Linear Regression cho phép hiểu được mức độ ảnh hưởng của mỗi biến độc lập đối với biến phụ thuộc. Điều này có ý nghĩa quan trọng trong việc hiểu các yếu tố quan trọng đóng vai trò như thế nào trong việc dự đoán lượng calo đốt khi tập luyện.
- **Khả năng ứng dụng linh hoạt:** Linear Regression có thể được sử dụng cho các bài toán với số lượng lớn các biến độc lập, với điều chỉnh các yếu tố được dự đoán. Trong trường hợp này, có nhiều biến độc lập như thời gian tập luyện, tuổi, chiều cao, cân nặng và nhịp tim.

Quá trình huấn luyện mô hình:



2. RandomForestRegressor:

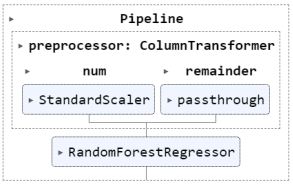
Lý do lựa chọn:

- **Xử lý tốt với dữ liệu không tuyến tính và tương tác phức tạp:** RandomForestRegressor có khả năng mô hình hóa các mối quan hệ phức tạp giữa các đặc trưng đầu vào và biến mục tiêu một cách linh hoạt. Điều này phù hợp với bài toán của chúng ta khi muốn dự đoán lượng calo đốt dựa trên nhiều yếu tố như thời gian tập luyện, cân nặng, chiều cao và tuổi.
- **Khả năng xử lý đặc trưng có ý nghĩa và không có ý nghĩa:** RandomForestRegressor có khả năng xử lý cả các đặc trưng có ý nghĩa và không có ý nghĩa trong việc dự đoán biến mục tiêu. Điều này có ích khi tập dữ liệu có chứa các đặc trưng không quan trọng hoặc không ảnh hưởng đến kết quả cuối cùng.
- **Giảm nguy cơ overfitting:** RandomForestRegressor có khả năng giảm nguy cơ overfitting trong khi vẫn giữ được khả năng mô hình hóa các mối quan hệ phức tạp. Điều này giúp cải thiện khả năng tổng quát hóa của mô hình và giảm thiểu hiện tượng mô hình hoá quá mức.
- **Tính linh hoạt và hiệu suất cao:** RandomForestRegressor là một trong những mô hình học máy phổ biến và hiệu quả, có thể áp dụng cho nhiều loại dữ liệu và cho các bài toán dự đoán khác nhau. Điều này làm cho nó trở thành lựa chọn phù hợp cho nhiều bài toán dự đoán, bao gồm cả bài toán của chúng ta về dự đoán lượng calo đốt khi tập luyện.

Quá trình huấn luyện mô hình:

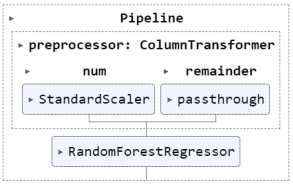
```
In [189]: RandomForestRegressor = Pipeline([("preprocessor",preprocessor),
                                            ("model",RandomForestRegressor())])

In [190]: RandomForestRegressor
```

Out[190]:




```
In [191]: RandomForestRegressor.fit(X_train,Y_train)
```

Out[191]:



3. XGBRegressor:

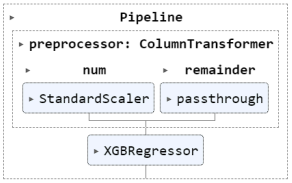
Lý do lựa chọn:

- **Hiệu suất cao:** XGBoost (Extreme Gradient Boosting) là một thuật toán Gradient Boosting Tree được tối ưu hóa mạnh mẽ, có hiệu suất cao và được sử dụng rộng rãi trong các cuộc thi máy học và các ứng dụng thực tế. Nó thường cho kết quả tốt trên nhiều loại dữ liệu và bài toán dự đoán.
- **Tính linh hoạt và khả năng tinh chỉnh:** XGBoost cung cấp nhiều siêu tham số (hyperparameters) để tinh chỉnh, giúp tối ưu hóa hiệu suất của mô hình trên dữ liệu cụ thể của bạn. Bạn có thể điều chỉnh các tham số như tỷ lệ học (learning rate), độ sâu cây (tree depth), số lượng cây (number of trees) và các siêu tham số khác để cải thiện hiệu suất của mô hình.
- **Xử lý tốt với dữ liệu lớn:** XGBoost có khả năng xử lý tốt với các tập dữ liệu lớn và có thể được tối ưu hóa để tận dụng tài nguyên máy tính hiệu quả. Điều này làm cho nó phù hợp với các bài toán có dữ liệu lớn và phức tạp.
- **Khả năng xử lý các loại dữ liệu:** XGBoost có thể xử lý các loại dữ liệu khác nhau, bao gồm cả dữ liệu dạng số và dạng category. Điều này làm cho nó phù hợp với việc xử lý các tập dữ liệu đa dạng và có các đặc trưng đa dạng như trong bài toán của chúng ta.

Quá trình huấn luyện mô hình:

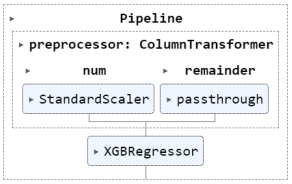
```
In [200]: XGBRegressor = Pipeline([("preprocessor", preprocessor),
                                   ("model", XGBReg())])

In [201]: XGBRegressor
```

Out[201]:




```
In [203]: XGBRegressor.fit(X_train,Y_train)
```

Out[203]:



II. Đánh giá các mô hình

Xây dựng hàm **model_scorer** để đánh giá điểm của mô hình:

```
In [28]:
def model_scorer(model_name, model):
    output = []

    output.append(model_name)

    pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('model', model)
    ])

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

    pipeline.fit(X_train, y_train)

    y_pred = pipeline.predict(X_test)

    # R-squared score
    output.append(r2_score(y_test, y_pred))

    # Mean absolute error
    output.append(mean_absolute_error(y_test,y_pred))

    # Cross-validation R-squared score
    kfold = KFold(n_splits=5, shuffle=True, random_state=42)
    cv_results = cross_val_score(pipeline, X, y, cv=kfold, scoring='r2')
    output.append(cv_results.mean())

    return output
```

Các mô hình sau khi huấn luyện sẽ được đánh giá dựa trên các phương pháp sau:

1.R-squared score (Điểm R bình phương):

Điểm R-squared (R^2) là một thước đo thống kê thể hiện tỷ lệ phần trăm biến động của biến phụ thuộc (biến mục tiêu) có thể được giải thích bởi biến độc lập (biến đầu vào) trong mô hình. Nó có giá trị từ 0 đến 1, với giá trị 1 biểu thị rằng mô hình hoàn toàn giải thích được biến động của dữ liệu.

Trong hàm **model_scorer**, bước này thực hiện như sau:

- Dữ liệu được chia thành tập huấn luyện và tập kiểm tra bằng **train_test_split**.
- Mô hình được huấn luyện trên tập huấn luyện.
- Dự đoán được thực hiện trên tập kiểm tra.
- Điểm R-squared được tính toán trên tập kiểm tra bằng cách so sánh giá trị thực tế với giá trị dự đoán.

2. Mean Absolute Error (Sai số trung bình tuyệt đối):

Mean Absolute Error (MAE) là một phép đo lỗi phổ biến trong việc đánh giá hiệu suất của mô hình dự đoán. Nó được sử dụng để đo lường sự chênh lệch trung bình giữa giá trị dự đoán của mô hình và giá trị thực tế của biến mục tiêu.

Trong hàm **model_scorer**, phương pháp tính Mean Absolute Error (MAE) được thực hiện như sau:

- Sau khi mô hình được huấn luyện và dự đoán trên tập kiểm tra, chúng ta tính toán sai số tuyệt đối giữa giá trị thực tế và giá trị dự đoán cho mỗi mẫu trong tập kiểm tra.
- Sau đó, chúng ta tính toán giá trị trung bình của các sai số tuyệt đối này, được gọi là Mean Absolute Error.
- Giá trị MAE được thêm vào danh sách đầu ra để đánh giá hiệu suất của mô hình.

3. Cross-validation R-squared score (Điểm R bình phương của cross-validation):

Cross-validation là kỹ thuật đánh giá mô hình bằng cách chia dữ liệu thành nhiều phần (folds), huấn luyện mô hình trên một số phần nhất định và kiểm tra trên phần còn lại. Quá trình này được lặp lại nhiều lần để đảm bảo rằng mô hình không bị overfitting vào một phần cụ thể của dữ liệu.

Trong hàm **model_scorer**, cross-validation được thực hiện như sau:

- Sử dụng KFold để chia dữ liệu thành 5 phần (5-fold cross-validation).
- Mô hình được huấn luyện và đánh giá trên các phần khác nhau của dữ liệu.
- Điểm R-squared trung bình của các lần lặp được tính toán và lưu trữ.

Kết quả đánh giá các mô hình:

```
In [33]: model_dict = {
             'log': LinearRegression(),
             'RF': RandomForestRegressor(),
             'XGBR': XGBRegressor(),
         }

In [40]: model_output=[]
         for model_name,model in model_dict.items():
             model_output.append(model_scorer(model_name,model))

In [41]: model_output

Out[41]: [['log', 0.9672937151257295, 8.441513553849708, 0.9671402283675838],
          ['RF', 0.9982777625803916, 1.6847033333333334, 0.9979108452440327],
          ['XGBR', 0.9988680981634738, 1.4984578529596329, 0.9988499611672703]]
```

Các giá trị đầu ra sau khi đánh giá mô hình cho thấy hiệu suất tốt của các mô hình được đánh giá, được đo bằng ba phương pháp khác nhau:

1. **Linear Regression (log)**:
    - Điểm R bình phương (R-squared score) trên tập kiểm tra là khoảng 0.967, cho thấy mô hình có khả năng giải thích được khoảng 96.7% sự biến thiên của biến mục tiêu.
    - Sai số trung bình tuyệt đối (Mean Absolute Error) là khoảng 8.44, có nghĩa là giá trị dự đoán trung bình chênh lệch với giá trị thực tế khoảng 8.44 đơn vị, đây là sai số khá lớn so với 2 mô hình còn lại.
    - Điểm R bình phương của cross-validation là khoảng 0.967, cho thấy mô hình có hiệu suất ổn định trên các fold khác nhau của cross-validation nhưng vẫn thấp hơn khá nhiều so với 2 mô hình còn lại.
2. **Random Forest Regressor (RF)**:
    - Điểm R bình phương trên tập kiểm tra là gần 0.998, cho thấy mô hình có khả năng giải thích tốt sự biến thiên của biến mục tiêu.
    - Sai số trung bình tuyệt đối là khoảng 1.68, có nghĩa là giá trị dự đoán trung bình chênh lệch với giá trị thực tế khoảng 1.68 đơn vị.
    - Điểm R bình phương của cross-validation cũng gần bằng 0.998, cho thấy hiệu suất ổn định của mô hình trên các fold khác nhau.
3. **XGBoost Regressor (XGBR)**:
    - Điểm R bình phương trên tập kiểm tra là gần 0.999, cho thấy mô hình có khả năng giải thích rất tốt sự biến thiên của biến mục tiêu.
    - Sai số trung bình tuyệt đối là khoảng 1.50, có nghĩa là giá trị dự đoán trung bình chênh lệch với giá trị thực tế khoảng 1.50 đơn vị.

- Điểm R bình phương của cross-validation cũng gần bằng 0.999, cho thấy hiệu suất ổn định của mô hình trên các fold khác nhau.

🔎 Kết luận:

- Cả ba mô hình đều hoạt động rất tốt, nhưng Random Forest và XGBoost Regressors vượt trội hơn so với Linear Regression.
- Mặc dù chỉ số của các phương pháp đánh giá ở mức tốt nhưng so với 2 mô hình còn lại Linear Regression vẫn có sự chênh lệch khá lớn.
- XGBoost Regressor có hiệu suất nhỉnh hơn một chút so với Random Forest Regressor, nhưng cả hai đều có điểm R-squared rất cao.
- Với điểm R-squared cao như vậy trên tất cả các mô hình, điều này cho thấy các đặc trưng đã chọn có thể dự đoán chính xác biến mục tiêu (Calories).

⇒ Cả ba mô hình Linear Regression, Random Forest Regressor và XGBoost Regressor đều cho kết quả rất tốt trên tập dữ liệu được sử dụng. Tuy nhiên, mô hình XGBoost Regressor là có hiệu suất tốt nhất với điểm R-squared cao nhất và sai số trung bình tuyệt đối thấp nhất.

▼ Phần VI: Lựa chọn mô hình và tối ưu hóa:

Qua phần huấn luyện và đánh giá các mô hình, mô hình tốt nhất là: **XGBoost Regressor**

⇒ Đây sẽ là mô hình dùng để dự đoán

Tối ưu hóa mô hình XGBoost Regressor:

Để tối ưu hóa cho mô hình **XGBoost Regressor** ta cần xác định tham siêu tham số grid của mô hình. Các siêu tham số bao gồm số lượng cây

(**n_estimators**), độ sâu tối đa của cây (**max_depth**), và tỷ lệ học (**learning_rate**).

Ý nghĩa của mỗi tham số:

1. **n_estimators**:
    - Đây là số lượng cây quyết định trong mô hình XGBoost. Các giá trị lớn hơn có thể tăng độ phức tạp của mô hình, nhưng cũng có thể gây ra overfitting.
    - Trong grid search, chúng ta thử nghiệm các giá trị khác nhau như 50, 100, và 200 để xác định số lượng cây tối ưu cho mô hình.
2. **max_depth**:
    - Tham số này xác định độ sâu tối đa của mỗi cây trong ensemble. Độ sâu lớn hơn có thể cho phép mô hình học được các mối quan hệ phức tạp hơn trong dữ liệu, nhưng cũng có thể dẫn đến overfitting.
    - Chúng ta thử nghiệm các giá trị khác nhau như 3, 4 và 5 để xác định độ sâu tối ưu.
3. **learning_rate**:
    - Đây là tỷ lệ học, điều chỉnh mức độ cập nhật cho các trọng số của cây sau mỗi bước học. Tỉ lệ học thấp hơn có thể làm cho việc học diễn ra chậm hơn, nhưng có thể giúp tránh overfitting.
    - Chúng ta thử nghiệm các giá trị khác nhau như 0.01, 0.1 và 0.2 để xác định tỷ lệ học tốt nhất cho mô hình.

Bằng cách thử nghiệm các giá trị khác nhau của các tham số này thông qua grid search, chúng ta có thể tìm ra các giá trị tối ưu giúp cải thiện hiệu suất của mô hình XGBoost Regressor trên dữ liệu huấn luyện và dữ liệu kiểm tra.

Kết quả:

In [43]:
```
param_grid = {
    'model__n_estimators': [50, 100, 200],
    'model__max_depth': [3, 4, 5],
    'model__learning_rate': [0.01, 0.1, 0.2]
}

pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', XGBRegressor())
])

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

print("Best parameters found: ", grid_search.best_params_)
print("Best cross-validation score: ", grid_search.best_score_)

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print("R2 score: ", r2_score(y_test, y_pred))
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
```

```
Best parameters found:  {'model__learning_rate': 0.1, 'model__max_depth': 5, 'model__n_estimators': 200}
Best cross-validation score:  0.9990977010602972
R2 score:  0.9993140124206046
Mean Absolute Error:  1.18884205515186
```

⇒ Các tham số tối ưu là **learning_rate=0.1**, **max_depth=5**, và **n_estimators=200**.

🔎 Nhận xét:

- Điểm R-squared trên tập kiểm tra đã tăng lên gần 0.999, cho thấy mô hình có khả năng giải thích rất tốt sự biến thiên của biến mục tiêu.
- Sai số trung bình tuyệt đối đã giảm xuống còn khoảng 1.19, thể hiện sự cải thiện đáng kể so với trước khi tối ưu hóa.
- Điểm R-squared của cross-validation cũng đã tăng lên gần 0.999, cho thấy hiệu suất ổn định của mô hình trên các fold khác nhau.

▼ Phần VII: Thử nghiệm và thiết kế giao diện đơn giản cho mô hình:

Thử nghiệm mô hình:

Sau khi tối ưu hóa mô hình ta sẽ thử nghiệm trên một sample mới:

```
In [50]: sample = pd.DataFrame({
             'Gender':'0',
             'Age':20,
             'Height':177.0,
             'Weight':73.0,
             'Duration':60.0,
             'Heart_Rate':105.0,
             'Body_Temp':37.8,
         },index=[0])

In [51]: predicted_value = best_model.predict(sample)

In [52]: print("Predicted value:", predicted_value)

         Predicted value: [162.68658]
```

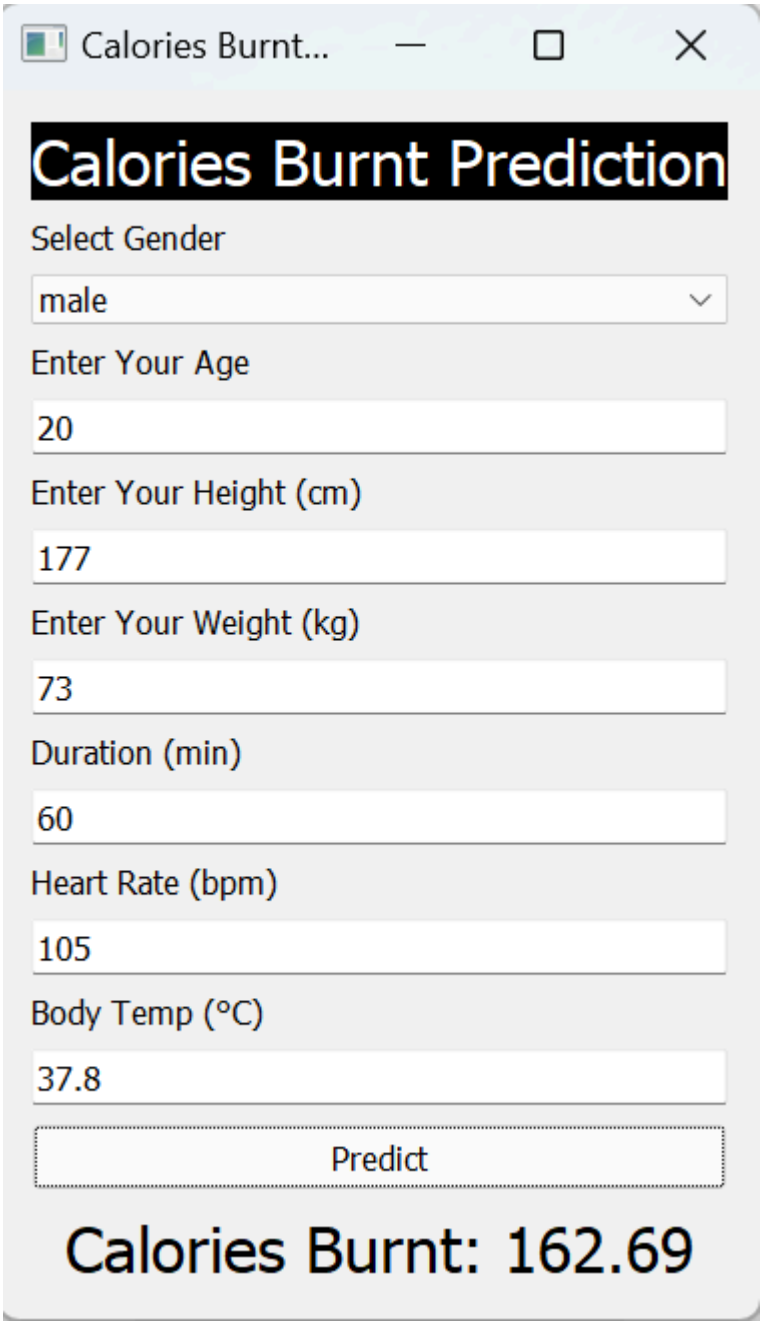
⇒ Mô hình đã dự đoạn được số calo đốt được dựa vào thông tin mới được cung cấp trung sample

Thiết kế giao diện đơn giản cho mô hình:

Sử dụng thư viện **pickle để** lưu mô hình vào file pipeline.pkl:

```
In [38]: import pickle
         from PyQt5 import QtWidgets, QtGui, QtCore

In [39]: with open('pipeline.pkl', 'wb') as file:
             pickle.dump(best_model, file)

         print("Model saved to pipeline.pkl")

         Model saved to pipeline.pkl
```

Sau đó thiết kế giao diện bằng thư viện **PyQt5,** giao diện mô hình:



⇒ Giao diện hoạt động tốt và có thể dự đoán chính xác

▼ Phần VIII: Kết luận:

Trong dự án này, nhóm em đã xây dựng một ứng dụng dự đoán lượng calo tiêu thụ dựa trên các thông tin cá nhân và hoạt động thể chất bằng cách sử dụng mô hình học máy gồm các quá trình sau:

1. Thiết kế bài toán và thu thập dữ liệu
2. Xử lý và phân tích dữ liệu
3. Huấn luyện, đánh giá và tối ưu hóa các mô hình

4. Thử nghiệm và thiết kế giao diện người dùng

Kết quả và Ý nghĩa

Ứng dụng dự đoán calo tiêu thụ này có thể được sử dụng bởi những người muốn theo dõi lượng calo tiêu thụ của mình dựa trên các hoạt động thể chất và các yếu tố thể trảng cá nhân. Việc có một công cụ dự đoán như vậy giúp người dùng có thể hiểu rõ hơn về quá trình tiêu hao năng lượng của mình và điều chỉnh chế độ tập luyện và dinh dưỡng một cách hợp lý.