BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

BÀI TẬP LỚN

HỌC MÁY CƠ BẢN

DỰ ĐOÁN NGUY CƠ BÉO PHÌ

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **HOÀNG CÔNG HOÀN**

LỚP: 12423TN

GIẢNG VIÊN: **PGS.TS.NGUYỄN VĂN HẬU**

HƯNG YÊN – 2025

NHẬN XÉT

Nhận xét của giảng viên:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIẢNG VIÊN

NGUYỄN VĂN HẬU

LỜI CAM ĐOAN

Em xin cam đoan bài tập lớp môn Học máy cơ bản "Dự đoán nguy cơ béo phì" là sản phẩm của bản thân dưới sự hướng dẫn của thầy Nguyễn Văn Hậu. Những phần sử dụng tài liệu tham khảo trong bài tập lớn đã được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong bài tập lớn là hoàn toàn trung thực, nếu sai em xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của bộ môn và nhà trường đề ra.

Hưng Yên ngày 04 tháng 12 năm 2025

SINH VIÊN

HOÀN

HOÀNG CÔNG HOÀN

LỜI CẢM ƠN

Để có thể hoàn thành bài tập lớn này, em gửi lời cảm ơn chân thành tới bộ môn Học máy cơ bản, Khoa học máy tính khoa Công nghệ thông tin của Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện bài tập lớp môn Học máy cơ bản. Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Văn Hậu đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

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

Em trân trọng cảm ơn!

NỘI DUNG

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN BÀI TOÁN

1.1 Bài toán

Trong Trong những năm gần đây, tỷ lệ thừa cân và béo phì trên toàn cầu đã tăng mạnh và được Tổ chức Y tế Thế giới (WHO) coi là một trong những vấn đề sức khỏe cộng đồng nghiêm trọng nhất thế kỷ 21. Theo các báo cáo gần đây, hơn 1,9 tỷ người trưởng thành bị thừa cân và hơn 650 triệu người bị béo phì (dữ liệu cập nhật đến năm 2024–2025). Tại Việt Nam, tình trạng thừa cân – béo phì cũng đang gia tăng nhanh chóng ở cả nhóm tuổi thiếu niên và người trưởng thành, đặc biệt tại các thành phố lớn, do sự thay đổi lối sống, chế độ ăn uống nhiều năng lượng và giảm hoạt động thể chất.

Béo phì không chỉ là vấn đề thẩm mỹ mà còn là yếu tố nguy cơ chính dẫn đến nhiều bệnh lý mãn tính nghiêm trọng như tiểu đường type 2, bệnh tim mạch, tăng huyết áp, đột quỵ, một số loại ung thư, rối loạn lipid máu và các vấn đề về xương khớp. Việc phát hiện sớm mức độ béo phì (hoặc nguy cơ thừa cân) ở mỗi cá nhân sẽ giúp can thiệp kịp thời thông qua thay đổi lối sống, chế độ dinh dưỡng và tập luyện, từ đó giảm thiểu nguy cơ mắc bệnh và cải thiện chất lượng cuộc sống.

Tuy nhiên, phương pháp đánh giá béo phì truyền thống chủ yếu dựa vào chỉ số khối cơ thể BMI = cân nặng / chiều cao² kết hợp với một số yếu tố lâm sàng khác. Cách tiếp cận này tuy đơn giản nhưng chưa phản ánh đầy đủ các yếu tố nguy cơ cá nhân hóa như: tiền sử gia đình, thói quen ăn uống, mức độ hoạt động thể chất, thời gian sử dụng thiết bị điện tử, thói quen tiêu thụ nước, hút thuốc, uống rượu bia, v.v. Những yếu tố này có mối liên hệ chặt chẽ đến sự tích tụ mỡ thừa và mức độ béo phì theo phân loại chi tiết như Insufficient Weight, Normal Weight, Overweight Level loại I/II, Obesity Type loại I/II/III.

Với sự phát triển mạnh mẽ của Machine Learning, đặc biệt là các mô hình phân loại đa lớp, chúng ta có khả năng xây dựng hệ thống dự đoán mức độ béo phì dựa trên tập hợp các đặc trưng sinh học, hành vi và lối sống. Mô hình học máy có thể học được các mối quan hệ phi tuyến tính phức tạp giữa các biến đầu vào và nhãn đầu ra là các mức độ béo phì, từ đó đưa ra dự đoán chính xác hơn so với các quy tắc heuristic truyền thống.

Bài toán được nghiên cứu trong báo cáo này là bài toán phân loại đa lớp có giám sát: Dựa trên 16 đặc trưng đầu vào bao gồm thông tin nhân khẩu học, chỉ số cơ thể, thói quen ăn uống, hoạt động thể chất, thói quen tiêu thụ chất kích thích và phương tiện di chuyển chính, dự đoán mức độ béo phì của một cá nhân thuộc vào một trong 7 lớp sau:

- Insufficient_Weight
- Normal_Weight (0rmal_Weight trong dữ liệu)
- Overweight_Level_I
- Overweight_Level_II
- Obesity_Type_I
- Obesity_Type_II
- Obesity_Type_III

Dữ liệu huấn luyện được lấy từ tập dữ liệu obesity_level.csv với 20.758 mẫu, đã được gán nhãn bởi các chuyên gia dựa trên các tiêu chí y khoa. Mục tiêu là xây dựng mô hình đạt độ chính xác cao trên tập kiểm tra

1.2 Tập dữ liệu

Dữ liệu được lấy từ Kaggle:

https://www.kaggle.com/datasets/jpkochar/obesity-risk-dataset

```
df.head()
```

Python

	id	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	C
0	0	Male	24.443011	1.699998	81.669950	1	1	2.000000	2.983297	Sometimes	0	2.76
1	1	Female	18.000000	1.560000	57.000000	1	1	2.000000	3.000000	Frequently	0	2.00
2	2	Female	18.000000	1.711460	50.165754	1	1	1.880534	1.411685	Sometimes	0	1.91
3	3	Female	20.952737	1.710730	131.274851	1	1	3.000000	3.000000	Sometimes	0	1.67
4	4	Male	31.641081	1.914186	93.798055	1	1	2.679664	1.971472	Sometimes	0	1.97

Hình 1.1 head

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20758 entries, 0 to 20757
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              20758 non-null  int64
 1   Gender                          20758 non-null  object
 2   Age                             20758 non-null  float64
 3   Height                          20758 non-null  float64
 4   Weight                          20758 non-null  float64
 5   family_history_with_overweight  20758 non-null  int64
 6   FAVC                            20758 non-null  int64
 7   FCVC                            20758 non-null  float64
 8   NCP                             20758 non-null  float64
 9   CAEC                            20758 non-null  object
 10  SMOKE                           20758 non-null  int64
 11  CH2O                            20758 non-null  float64
 12  SCC                             20758 non-null  int64
 13  FAF                             20758 non-null  float64
 14  TUE                             20758 non-null  float64
 15  CALC                            20758 non-null  object
 16  MTRANS                          20758 non-null  object
 17  0be1dad                         20758 non-null  object
dtypes: float64(8), int64(5), object(5)
memory usage: 2.9+ MB
```

Hình 1.2 info

Tập dữ liệu có các đặc trưng sau:

+ Gender: Giới tính
+ Age: Tuổi
+ Height: Chiều cao
+ Weight: Cân nặng
+ family_history_with_overweight: Tiền sử gia đình
+ FAVC: Ăn thực phẩm nhiều calo thường xuyên
+ FCVC: Mức độ ăn rau
+ NCP: Số bữa ăn chình mỗi ngày
+ CAEC: Ăn vặt giữa các bữa hay không
+ SMOKE: Có hút thuốc không
+ CH2O: Lượng nước uống mỗi ngày
+ SCC: Tiêu thụ đồ uống có calo
+ FAF: Tần suất hoạt động thể chất
+ TUE: Thời gian sử dụng thiết bị công nghệ mỗi ngày

+ CALC: Mức tiêu thụ rượu
+ MTRANS: Phương tiện di chuyển chính

- Tập dữ liệu ở định dạng csv bao gồm 20758 hàng × 18 cột - Mô tả:

```
df.describe().T
```
✓ 0.0s

	count	mean	std	min	25%	50%	75%	max
id	20758.0	10378.500000	5992.462780	0.00	5189.250000	10378.500000	15567.750000	20757.000000
Age	20758.0	23.841804	5.688072	14.00	20.000000	22.815416	26.000000	61.000000
Height	20758.0	1.700245	0.087312	1.45	1.631856	1.700000	1.762887	1.975663
Weight	20758.0	87.887768	26.379443	39.00	66.000000	84.064875	111.600553	165.057269
family_history_with_overweight	20758.0	0.819636	0.384500	0.00	1.000000	1.000000	1.000000	1.000000
FAVC	20758.0	0.914443	0.279716	0.00	1.000000	1.000000	1.000000	1.000000
FCVC	20758.0	2.445908	0.533218	1.00	2.000000	2.393837	3.000000	3.000000
NCP	20758.0	2.761332	0.705375	1.00	3.000000	3.000000	3.000000	4.000000
SMOKE	20758.0	0.011803	0.108000	0.00	0.000000	0.000000	0.000000	1.000000
CH2O	20758.0	2.029418	0.608467	1.00	1.792022	2.000000	2.549617	3.000000
SCC	20758.0	0.033096	0.178891	0.00	0.000000	0.000000	0.000000	1.000000
FAF	20758.0	0.981747	0.838302	0.00	0.008013	1.000000	1.587406	3.000000
TUE	20758.0	0.616756	0.602113	0.00	0.000000	0.573887	1.000000	2.000000

Hình 1.3 describe

1.3 Tiền xử lý dữ liệu

- Loại bỏ cột id không dùng đến

```
df = df.drop(['id'], axis=1)
df.head()
```
Python

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC
0	Male	24.443011	1.699998	81.669950	1	1	2.000000	2.983297	Sometimes	0	2.763573	0	0.000000	0.976473	Sometimes
1	Female	18.000000	1.560000	57.000000	1	1	2.000000	3.000000	Frequently	0	2.000000	0	1.000000	1.000000	0
2	Female	18.000000	1.711460	50.165754	1	1	1.880534	1.411685	Sometimes	0	1.910378	0	0.866045	1.673584	0
3	Female	20.952737	1.710730	131.274851	1	1	3.000000	3.000000	Sometimes	0	1.674061	0	1.467863	0.780199	Sometimes
4	Male	31.641081	1.914186	93.798055	1	1	2.679664	1.971472	Sometimes	0	1.979848	0	1.967973	0.931721	Sometimes

Hình 1.4 Loại bỏ cột id

- Đổi tên cột mục tiêu từ 0be1dad thành obesity_level

```
df = df.rename(columns={'0be1dad': 'obesity_level'})
df.head()
```
Python

Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	obesity_level
669950	1	1	2.000000	2.983297	Sometimes	0	2.763573	0	0.000000	0.976473	Sometimes	Public_Transportation	Overweight_Level_II
000000	1	1	2.000000	3.000000	Frequently	0	2.000000	0	1.000000	1.000000	0	Automobile	0rmal_Weight
165754	1	1	1.880534	1.411685	Sometimes	0	1.910378	0	0.866045	1.673584	0	Public_Transportation	Insufficient_Weight
274851	1	1	3.000000	3.000000	Sometimes	0	1.674061	0	1.467863	0.780199	Sometimes	Public_Transportation	Obesity_Type_III
798055	1	1	2.679664	1.971472	Sometimes	0	1.979848	0	1.967973	0.931721	Sometimes	Public_Transportation	Overweight_Level_II

Hình 1.5 Đổi tên cột mục tiêu thành obesity_level

- Kiểm tra giá trị bị thiếu

```
df.isna().sum()
```

```
id                                0
Gender                            0
Age                               0
Height                            0
Weight                            0
family_history_with_overweight    0
FAVC                              0
FCVC                              0
NCP                               0
CAEC                              0
SMOKE                             0
CH2O                              0
SCC                               0
FAF                               0
TUE                               0
CALC                              0
MTRANS                            0
obesity_level                     0
dtype: int64
```

Hình 1.6 Kiểm tra giá trị thiếu

- Kiểm tra giá trị bất thường trong các cột

```
if (df['Age'] > 100).any():
    print(f"Age > 100: {(df['Age'] > 100).sum()} giá trị")
else:
    print("Tất cả giá trị trong cột 'Age' đều <= 100.")
if (df['FCVC'] > 3).any() or (df['FCVC'] < 1).any():
    print(f"FCVC ngoài khoảng 1-3: {(df['FCVC'] < 1).sum() + (df['FCVC'] > 3).sum()} giá trị")
else:
    print("Tất cả giá trị trong cột 'FCVC' đều trong khoảng 1-3.")
if (df['NCP'] > 4).any() or (df['NCP'] < 1).any():
    print(f"NCP ngoài khoảng 1-4: {(df['NCP'] < 1).sum() + (df['NCP'] > 4).sum()} giá trị")
else:
    print("Tất cả giá trị trong cột 'NCP' đều trong khoảng 1-4.")
```

```
Tất cả giá trị trong cột 'Age' đều <= 100.
Tất cả giá trị trong cột 'FCVC' đều trong khoảng 1-3.
Tất cả giá trị trong cột 'NCP' đều trong khoảng 1-4.
```

Hình 1.7 Kiểm tra giá trị bất thường

- Sửa lỗi chính tả trong obesity_level

```
df['obesity_level'] = df['obesity_level'].replace('0rmal_Weight', 'Normal_Weight')
df['obesity_level'].unique()
```

```
array(['Overweight_Level_II', 'Normal_Weight', 'Insufficient_Weight',
       'Obesity_Type_III', 'Obesity_Type_II', 'Overweight_Level_I',
       'Obesity_Type_I'], dtype=object)
```

Hình 1.8 Sửa lỗi chính tả trong obesity_level

- Thay giá trị 0 thành No trong các cột phân loại CAEC và CALC

```
df['CAEC'] = df['CAEC'].replace('0', 'No')
df['CALC'] = df['CALC'].replace('0', 'No')
df.head()
```

✓ 0.0s Python

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC
0	Male	24.443011	1.699998	81.669950	1	1	2.000000	2.983297	Sometimes	0	2.763573	0	0.000000	0.976473	Sometimes
1	Female	18.000000	1.560000	57.000000	1	1	2.000000	3.000000	Frequently	0	2.000000	0	1.000000	1.000000	No
2	Female	18.000000	1.711460	50.165754	1	1	1.880534	1.411685	Sometimes	0	1.910378	0	0.866045	1.673584	No
3	Female	20.952737	1.710730	131.274851	1	1	3.000000	3.000000	Sometimes	0	1.674061	0	1.467863	0.780199	Sometimes
4	Male	31.641081	1.914186	93.798055	1	1	2.679664	1.971472	Sometimes	0	1.979848	0	1.967973	0.931721	Sometimes

Hình 1.9 Thay giá trị 0 thành No trong các cột phân loại

- Phân loại các cột theo kiểu dữ liệu

```
# Danh sách các cột theo loại
numerical_cols = ['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']
binary_cols    = ['Gender', 'family_history_with_overweight', 'FAVC', 'SMOKE', 'SCC']
ordinal_cols   = ['CAEC', 'CALC']
nominal_cols   = ['MTRANS']
```

✓ 0.0s

Hình 1.10 Phân loại các cột theo loại dữ liệu

- Định nghĩa các bước tiền xử lý dữ liệu trước khi đưa vào mô hình

```
ordinal_categories = {
    'CAEC': ['No', 'Sometimes', 'Frequently', 'Always'],
    'CALC': ['No', 'Sometimes', 'Frequently']
}


# xây dựng bộ tiền xử lý
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('binary', OneHotEncoder(drop='if_binary'), binary_cols),
        ('ord_caec', OrdinalEncoder(categories=[ordinal_categories['CAEC']]), ['CAEC']),
        ('ord_calc', OrdinalEncoder(categories=[ordinal_categories['CALC']]), ['CALC']),
        ('nom', OneHotEncoder(handle_unknown='ignore'), nominal_cols)
    ],
    remainder='drop'  # bỏ cột id
)
```

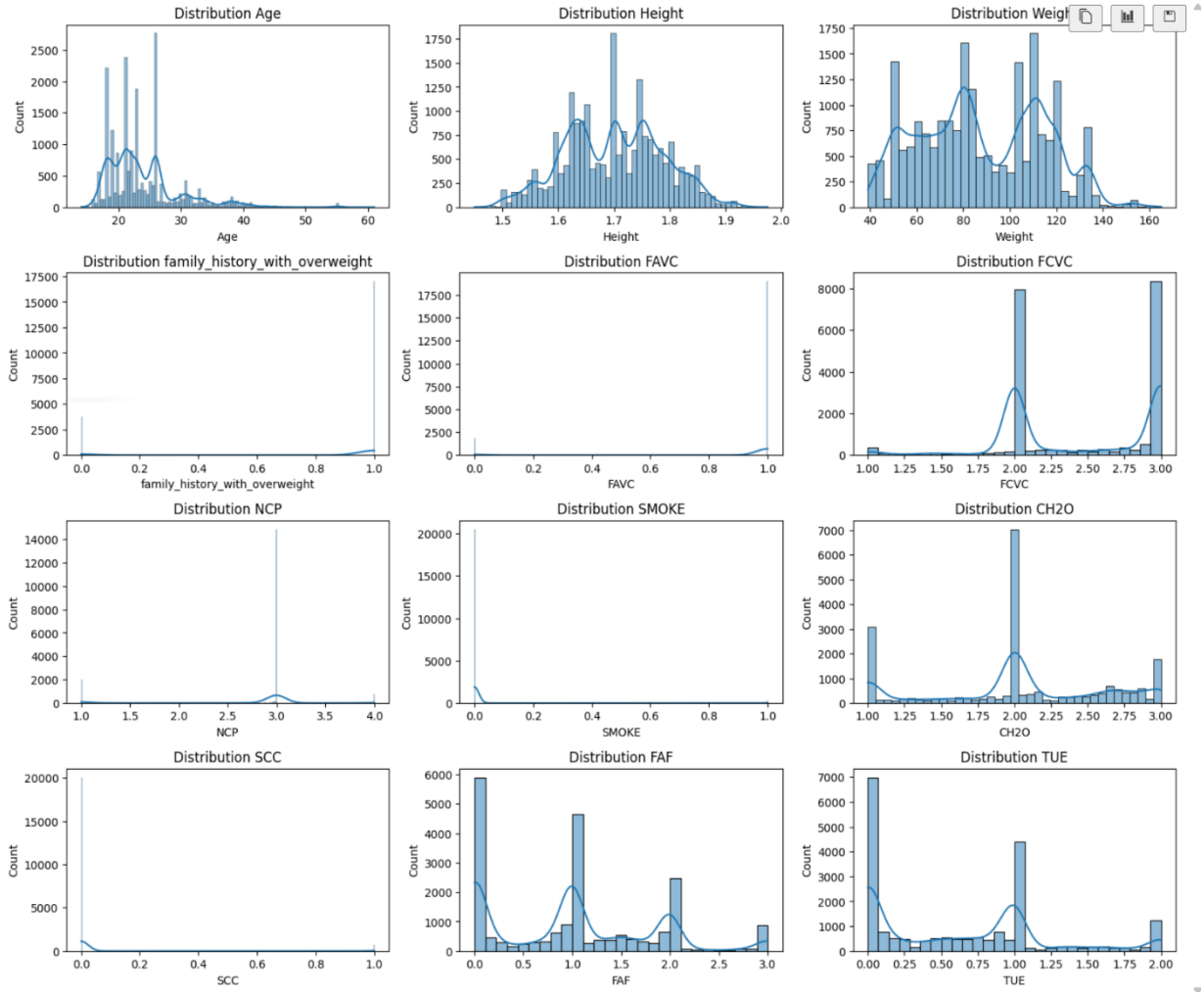
Hình 1.11 Định nghĩa các bước tiền xử lý

1.4 Phân tích và trực quan hoá dữ liệu

a) Biểu đồ phân bố các đặc điểm số

```
# Chỉ chọn các cột số
numeric_columns = df.select_dtypes(include='number').columns

# Thay thế inf và -inf bằng NaN
df[numeric_columns] = df[numeric_columns].replace([np.inf, -np.inf], np.nan)

# Phân bố cho tất cả các đặc điểm số
plt.figure(figsize=(15, 15))
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(len(numeric_columns) // 3 + 1, 3, i)
    sns.histplot(df[col].dropna(), kde=True)
    plt.title(f'Distribution {col}')
    plt.xlabel(col)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

✓ 4.1s Pytho



Hình 1.12 Biểu đồ phân bố các đặc điểm số

b) Biểu đồ hiển thị mối quan hệ đôi một giữa các đặc trưng chính

```
# Chọn các cột numerical quan trọng
num_cols = ['Age', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']

# Tạo DataFrame chỉ chứa các cột này + cột target để tô màu
eda_df = df[num_cols + ['obesity_level']]

# Vẽ pairplot, tô màu theo mức độ béo phì (rất hữu ích!)
sns.pairplot(eda_df, hue='obesity_level', diag_kind='kde', palette='husl')

plt.show()
```

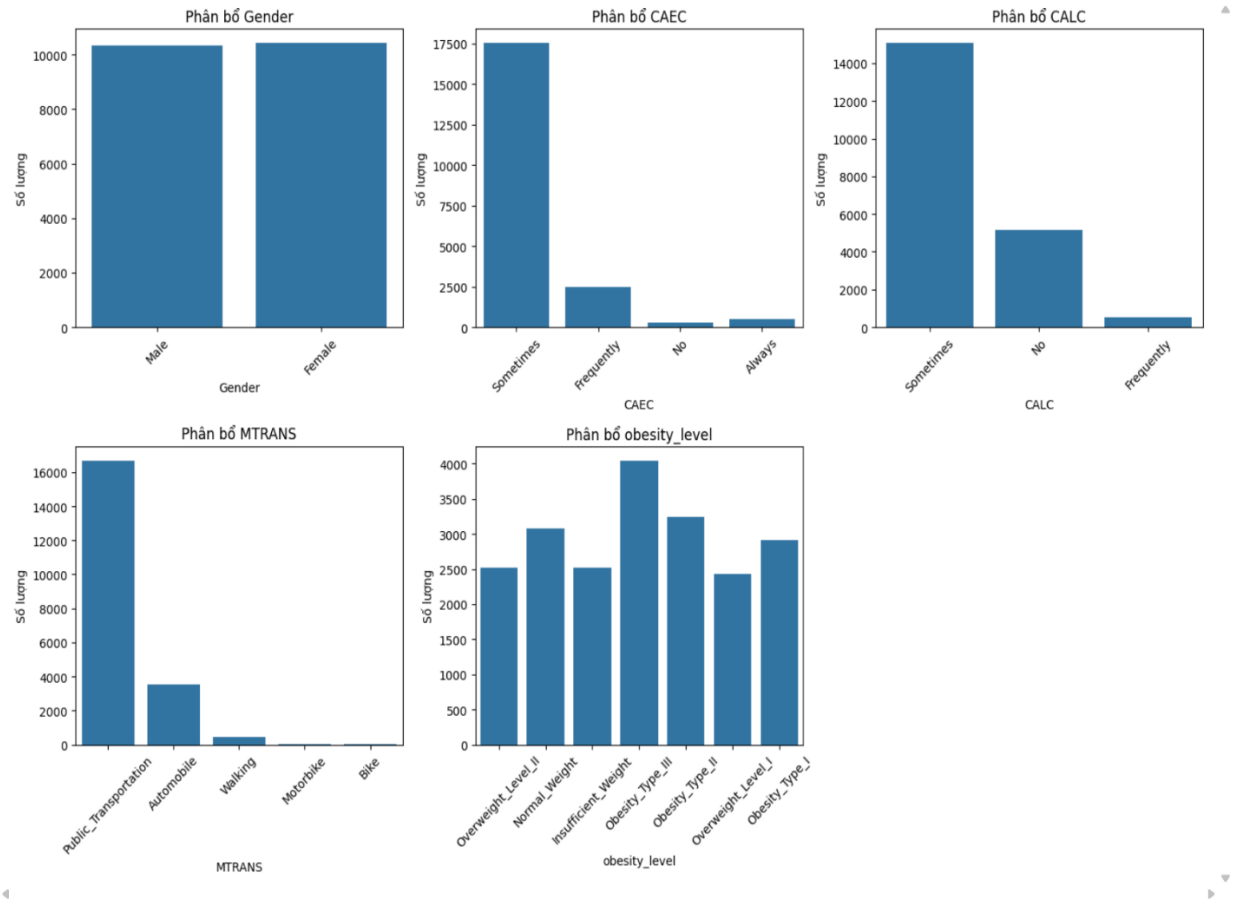
✓ 24.3s

Hình 1.13 Biểu đồ hiển thị mối quan hệ đôi một

c) Biểu đồ phân bố cho tất cả các đặc điểm phân loại

```python
# Chỉ chọn các cột phân loại
non_numeric_columns = df.select_dtypes(include=['object', 'category']).columns

# Kiểm tra các cột không phải là số
if len(non_numeric_columns) == 0:
    print("Tập dữ liệu không chứa cột nào không phải là số.")
else:
    # Biểu đồ phân bố cho tất cả các đặc điểm phân loại.
    plt.figure(figsize=(15, 10))
    for i, col in enumerate(non_numeric_columns, 1):
        plt.subplot(len(non_numeric_columns) // 3 + 1, 3, i)
        sns.countplot(x=col, data=df)
        plt.title(f'Phân bố {col}')
        plt.xlabel(col)
        plt.ylabel('Số lượng')
        plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()
```

0.4s Python

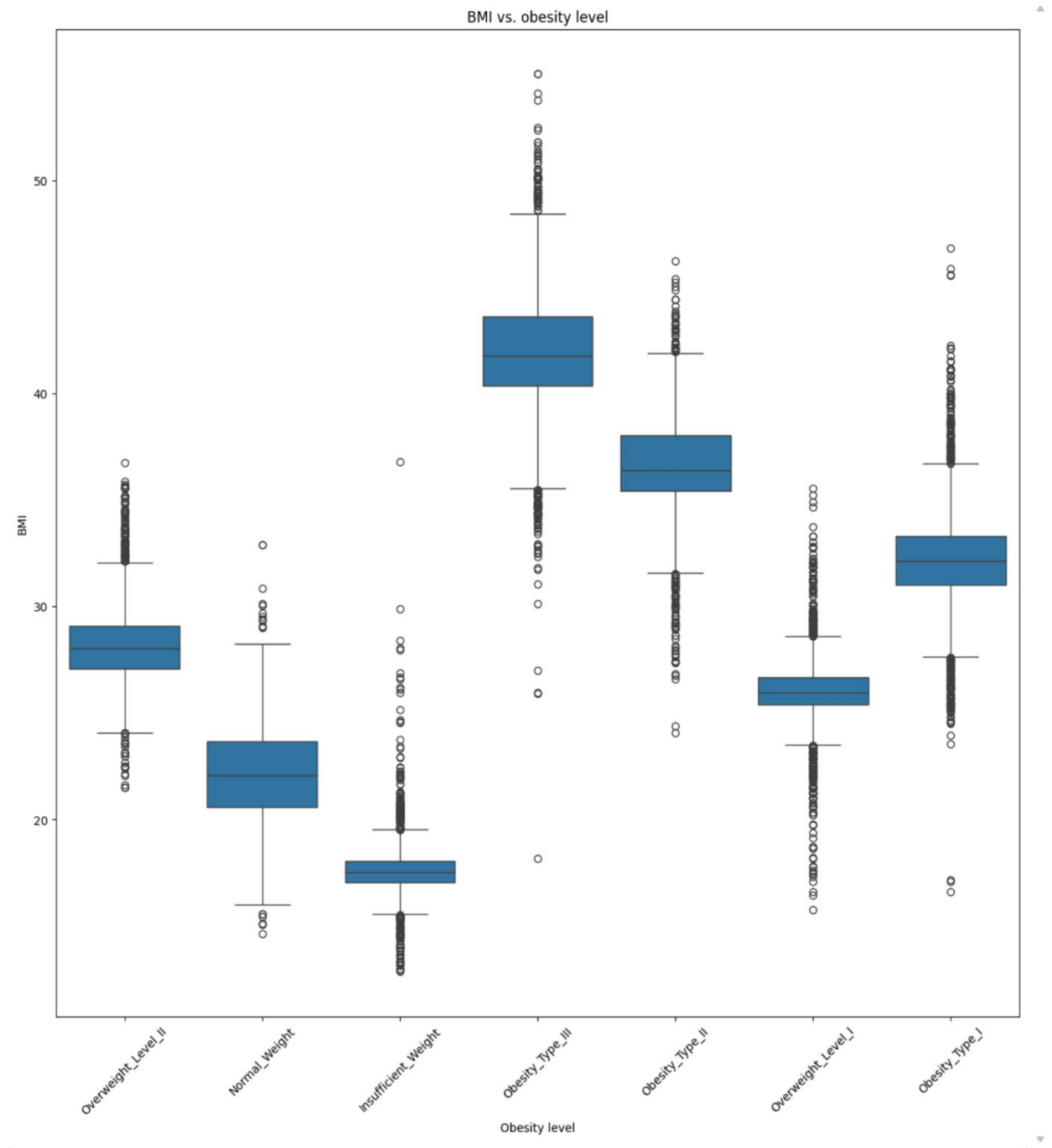


Hình 1.14 Biểu đồ phân bố các đặc điểm phân loại

d) Biểu đồ hộp thể hiện mối quan hệ giữa chỉ số BMI và mức độ béo phì

```python
# Thêm cột BMI
df['BMI'] = df['Weight'] / (df['Height'] ** 2)

# Biểu đồ hộp thể hiện mối quan hệ giữa chỉ số BMI và mức độ béo phì
plt.figure(figsize=(15, 15))
sns.boxplot(x='obesity_level', y='BMI', data=df)
plt.title('BMI vs. obesity level')
plt.xlabel('Obesity level')
plt.ylabel('BMI')
plt.xticks(rotation=45)
plt.show()
```

✓ 0.1s Python

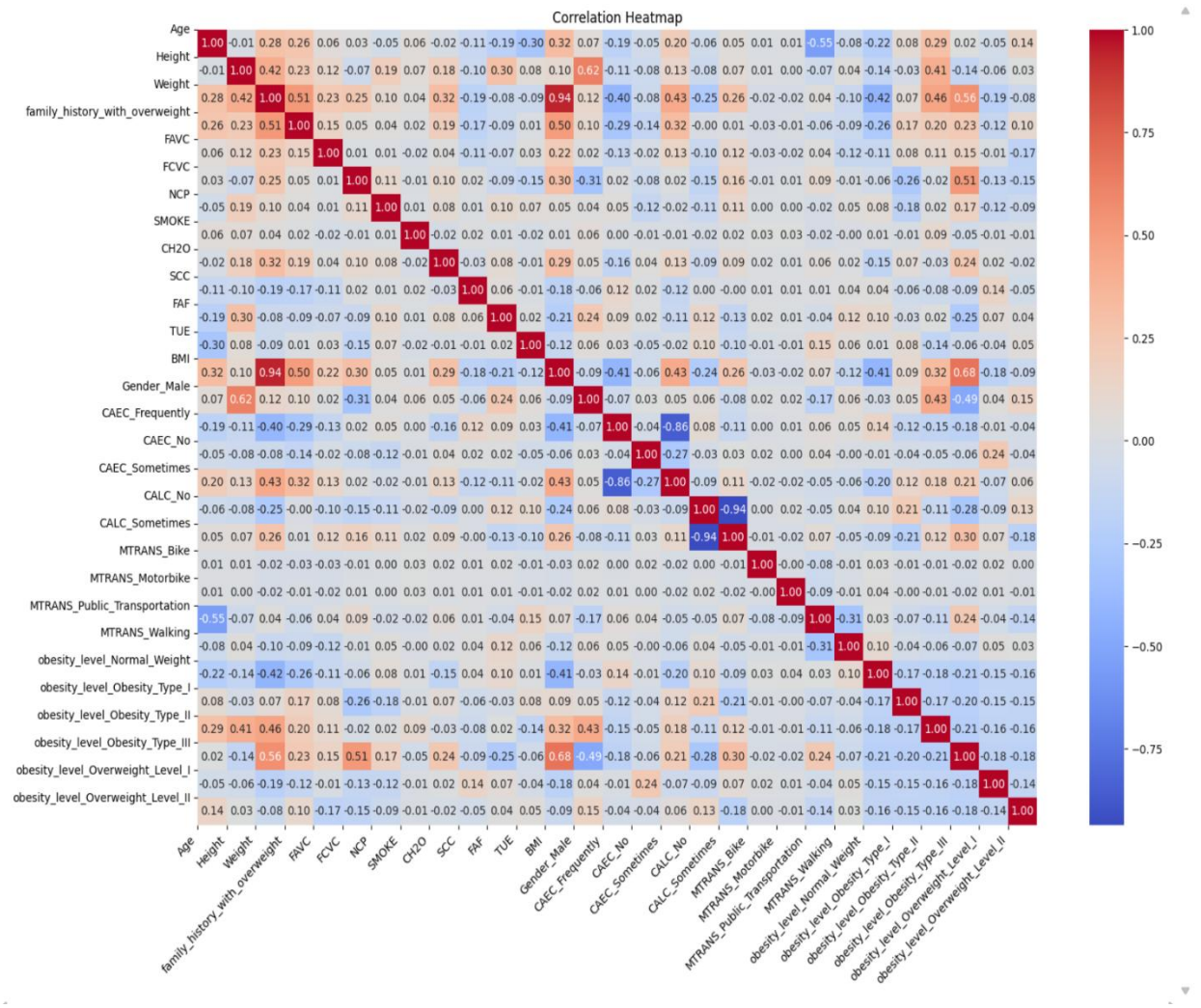


Hình 1.15 Biểu đồ thể hiện mối quan hệ giữa BMI và các mức độ béo phì

e) Biểu đồ ma trận tương quan

```python
# Kiểm tra các đặc điểm phân loại và áp dụng mã hóa one-hot.
categorical_features = df.select_dtypes(include=['object']).columns
data = pd.get_dummies(df, columns=categorical_features, drop_first=True)

# Tạo ma trận tương quan
correlation = data.corr()

# Hình ảnh trực quan của ma trận tương quan
fig, ax = plt.subplots(figsize=(18, 12))
sns.heatmap(correlation, annot=True, fmt='.2f', ax=ax, cmap='coolwarm')

ax.set_title('Correlation Heatmap')

# Đặt nhãn trục
ax.set_xticks(np.arange(len(correlation.columns)))
ax.set_yticks(np.arange(len(correlation.columns)))
ax.set_xticklabels(correlation.columns, rotation=45, ha='right')
ax.set_yticklabels(correlation.columns)

plt.show()
```

✓ 0.9s Python

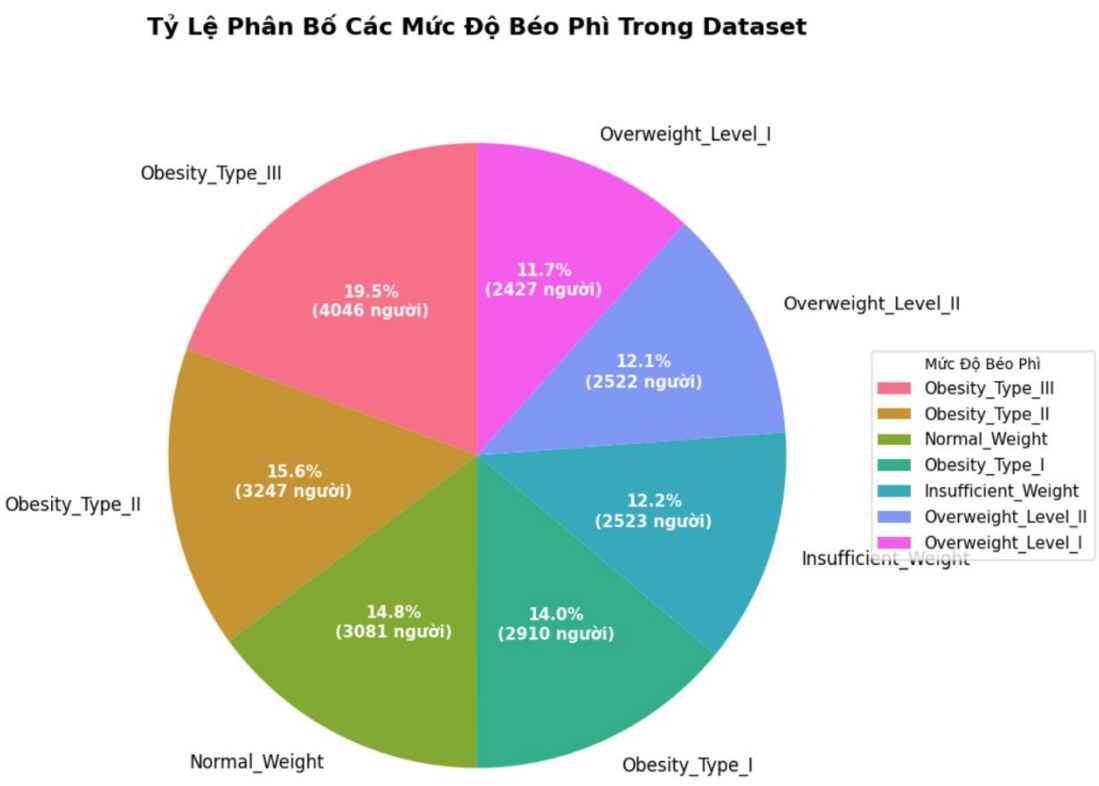


Hình 1.16 Ma trận tương quan

f) Tỷ Lệ Phân Bố Các Mức Độ Béo Phì

```python
plt.figure(figsize=(10, 8))

# Đếm số lượng từng lớp và tính tỷ lệ phần trăm
obesity_counts = df['obesity_level'].value_counts()
labels = obesity_counts.index
sizes = obesity_counts.values
percentages = 100 * sizes / sizes.sum()

# Vẽ pie chart
wedges, texts, autotexts = plt.pie(
    sizes,
    labels=labels,
    autopct=lambda pct: f'{pct:.1f}%\n({int(pct/100 * sizes.sum())} người)',
    startangle=90,
    colors=sns.color_palette("husl", len(labels)),
    textprops={'fontsize': 12}
)

# Cải thiện hiển thị phần trăm
for autotext in autotexts:
    autotext.set_color('white')
    autotext.set_fontweight('bold')
    autotext.set_fontsize(11)

# Tiêu đề
plt.title('Tỷ Lệ Phân Bố Các Mức Độ Béo Phì Trong Dataset', fontsize=16, fontweight='bold', pad=20)

# Thêm legend bên cạnh để dễ đọc tên đầy đủ
plt.legend(labels, title="Mức Độ Béo Phì", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1), fontsize=11)

# Hiển thị biểu đồ
plt.tight_layout()
plt.show()
```

✓ 0.1s Python



Hình 1.17 Biểu đồ tỷ lệ phân bố các mức độ béo phì

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Pandas

Pandas là một trong những thư viện mã nguồn mở phổ biến và mạnh mẽ nhất trong hệ sinh thái Python dành cho khoa học dữ liệu và học máy. Được phát triển bởi Wes McKinney vào năm 2008, Pandas cung cấp các cấu trúc dữ liệu và công cụ xử lý dữ liệu hiệu quả, giúp người dùng thao tác nhanh chóng với dữ liệu có cấu trúc (structured data) như bảng tính, cơ sở dữ liệu hoặc dữ liệu dạng chuỗi thời gian.

Trong môn Học máy cơ bản, Pandas đóng vai trò nền tảng trong giai đoạn tiền xử lý dữ liệu (data preprocessing) – bước quan trọng nhất trước khi áp dụng các thuật toán học máy. Gần như mọi dự án học máy thực tế đều bắt đầu bằng việc sử dụng Pandas để đọc, khám phá, làm sạch và biến đổi dữ liệu.

rãi.

2.1.1 Series

- Là mảng một chiều (1D) có nhãn (labeled array), tương tự như cột trong bảng tính Excel.
- Mỗi phần tử trong Series có một chỉ số (index) có thể là số nguyên, chuỗi ký tự hoặc ngày tháng.
- Ví dụ: Một cột "Age" trong tập dữ liệu là một đối tượng Series.

2.1.2 DataFrame

- Là cấu trúc dữ liệu hai chiều (2D), giống như bảng (table) với hàng và cột, mỗi cột là một Series.
- DataFrame hỗ trợ các cột có kiểu dữ liệu khác nhau (int, float, string, category, datetime, v.v.).
- Đây là cấu trúc chính được sử dụng để lưu trữ và thao tác toàn bộ tập dữ liệu trong dự án này (obesity_level.csv được đọc vào dưới dạng DataFrame).

2.2 Matplotlib

Matplotlib là một trong những thư viện mã nguồn mở phổ biến nhất trong Python dành cho việc vẽ đồ thị và trực quan hóa dữ liệu (data visualization). Được phát triển bởi John D. Hunter vào năm 2003, Matplotlib được thiết kế để tạo ra các biểu đồ tĩnh, động và tương tác chất lượng cao, phù hợp cho cả mục đích nghiên cứu khoa học, báo cáo học thuật lẫn trình bày kết quả trong học máy.

Matplotlib thường được coi là "công cụ vẽ đồ thị cơ bản" trong hệ sinh thái Python, tương tự như cách MATLAB vẽ đồ thị. Nó là nền tảng cho nhiều thư viện trực quan hóa cao cấp hơn như Seaborn, Plotly hay Pandas plotting interface. Trong môn Học máy cơ bản, Matplotlib đóng vai trò quan trọng trong giai đoạn khám phá dữ liệu EDA và đánh giá mô hình, giúp sinh viên dễ dàng hình dung phân bố dữ liệu, mối quan hệ giữa các biến, hiệu suất mô hình confusion matrix, ROC curve, feature importance, v.v..

2.2.1 Các đặc điểm chính của Matplotlib

- Đa dạng loại biểu đồ: Hỗ trợ gần như mọi loại đồ thị phổ biến: line plot, scatter plot, bar chart, histogram, pie chart, boxplot, heatmap, 3D plot, v.v.
- Tùy chỉnh linh hoạt cao: Có thể điều chỉnh mọi chi tiết như màu sắc, kích thước, nhãn, tiêu đề, legend, grid, axis, font, style, v.v. để tạo biểu đồ chuyên nghiệp, phù hợp xuất bản.
- Hỗ trợ nhiều backend: Có thể hiển thị đồ thị trong Jupyter Notebook, script Python thông thường, ứng dụng GUI (Tkinter, Qt), hoặc lưu file (PNG, PDF, SVG, JPG).
- Tích hợp tốt: Làm việc mượt mà với NumPy (dữ liệu mảng), Pandas (DataFrame), Scikit-learn (kết quả mô hình), Seaborn giao diện đẹp hơn.
- Miễn phí và mã nguồn mở: Cộng đồng lớn, tài liệu phong phú, cập nhật thường xuyên (phiên bản mới nhất vào năm 2025–2026 là 3.8+ hoặc cao hơn).

2.3 Numpy

NumPy viết tắt của Numerical Python là thư viện nền tảng cốt lõi cho tính toán khoa học và học máy trong Python. Được phát triển bởi Travis Oliphant vào năm 2005 dựa trên các dự án tiền thân như Numeric và Numarray, NumPy cung cấp hỗ trợ cho mảng đa chiều multidimensional arrays và các hàm toán học hiệu suất cao, giúp Python trở thành một ngôn ngữ mạnh mẽ ngang tầm với MATLAB, R hay Julia trong lĩnh vực tính toán số.

Trong môn Học máy cơ bản và hầu hết các dự án học máy thực tế, NumPy đóng vai trò như "xương sống" cho việc xử lý dữ liệu số. Nó là nền tảng mà hầu hết các thư viện học máy (Scikit-learn, TensorFlow, PyTorch, Pandas, Matplotlib, Seaborn, v.v.) đều xây dựng lên.

2.3.1 Các đặc điểm chính của Numpy

- Mảng ndarray (n-dimensional array): Cấu trúc dữ liệu chính, hỗ trợ mảng nhiều chiều với hiệu suất gần bằng ngôn ngữ C nhờ được viết bằng C và Fortran.
- Hiệu suất cao: Các phép toán vectorized không cần vòng lặp Python nhanh gấp hàng chục đến hàng trăm lần so với list Python thông thường.
- Broadcasting: Cho phép thực hiện phép toán giữa các mảng có kích thước khác nhau mà không cần sao chép dữ liệu.
- Hỗ trợ toán học phong phú: Hàng trăm hàm cho đại số tuyến tính, thống kê, xử lý ngẫu nhiên, biến đổi Fourier, v.v.
- Tích hợp chặt chẽ: Pandas DataFrame được xây dựng trên NumPy array; Scikit-learn yêu cầu dữ liệu đầu vào là NumPy array hoặc tương thích.

2.4 Scikit-learn

Scikit-learn thường được gọi tắt là sklearn là một trong những thư viện học máy mã nguồn mở phổ biến và mạnh mẽ nhất trong hệ sinh thái Python. Được phát triển từ năm 2007 bởi David Cournapeau và sau đó được cộng đồng duy trì, Scikit-learn cung cấp các công cụ đơn giản, hiệu quả và nhất quán để thực hiện hầu hết các nhiệm vụ học máy cơ bản: phân loại classification, hồi quy regression, gom cụm, giảm chiều, tiền xử lý dữ liệu, lựa chọn mô hình và đánh giá mô hình.

Scikit-learn được xây dựng hoàn toàn dựa trên NumPy, SciPy và Matplotlib, đảm bảo tính tương thích cao và hiệu suất tốt. Trong môn Học máy cơ bản, Scikit-learn là thư viện chính được sử dụng để xây dựng, huấn luyện và đánh giá các mô hình học máy, thay vì phải tự viết lại các thuật toán từ đầu.

2.4.1. Các đặc điểm nổi bật của Scikit-learn

- **API thống nhất và dễ sử dụng**: Tất cả các estimator đều tuân theo cùng một giao diện:
    + .fit(X, y): Huấn luyện mô hình
    + .predict(X): Dự đoán
    + .predict_proba(X): Xác suất lớp (nếu có)
    + .score(X, y): Đánh giá điểm số mặc định
    + .transform(X) / .fit_transform(X): Biến đổi dữ liệu
- **Hỗ trợ đa dạng thuật toán**: Hàng chục mô hình học máy cổ điển và hiện đại, bao gồm:
    + Phân loại: LogisticRegression, SVC, RandomForestClassifier, GradientBoostingClassifier, KNeighborsClassifier, v.v.
    + Hồi quy: LinearRegression, Ridge, Lasso, SVR, v.v.
    + Gom cụm: KMeans, DBSCAN, Hierarchical clustering
    + Giảm chiều: PCA, t-SNE, TruncatedSVD
    + Ensemble: Random Forest, AdaBoost, VotingClassifier, StackingClassifier
- **Công cụ tiền xử lý mạnh mẽ**:
    + StandardScaler, MinMaxScaler, RobustScaler
    + OneHotEncoder, OrdinalEncoder, LabelEncoder
    + SimpleImputer, KNNImputer
    + PolynomialFeatures, FeatureUnion
- **Pipeline và ColumnTransformer**: Cho phép xây dựng quy trình xử lý dữ liệu và mô hình chỉ trong một đối tượng duy nhất, tránh data leakage và dễ triển khai.
- **Đánh giá và lựa chọn mô hình**:

    + cross_val_score, GridSearchCV, RandomizedSearchCV
    + metrics: accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, classification_report
- **Hiệu suất và ổn định**: Được viết bằng Python + Cython, tối ưu cho dữ liệu vừa và nhỏ phù hợp với bài tập lớn và nghiên cứu học thuật.

CHƯƠNG 3: PHƯƠNG PHÁP

3.1 Lý thuyết

3.1.1 Mô hình Logistic Regression

Logistic Regression: Hồi quy Logistic là một trong những thuật toán phân loại cơ bản và phổ biến nhất trong học máy, thường được sử dụng làm baseline để so sánh với các mô hình phức tạp hơn. Mặc dù tên gọi có từ "Regression", đây thực chất là mô hình phân loại classification, đặc biệt phù hợp với bài toán phân loại nhị phân, nhưng có thể mở rộng cho phân loại đa lớp thông qua chiến lược One-vs-Rest (OvR) hoặc Softmax (Multinomial).

Nguyên lý hoạt động: Logistic Regression dự đoán xác suất một mẫu thuộc vào một lớp cụ thể bằng cách sử dụng hàm sigmoid đối với phân loại nhị phân hoặc hàm softmax đối với đa lớp. Công thức cơ bản cho phân loại nhị phân:

nhị phân:

$$P(y = 1 \mid X) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

trong đó:

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- w là vector trọng số (weights) cần học.
- z là tổng tuyến tính của các đặc trưng.
- Hàm sigmoid $\sigma(z)$ ánh xạ giá trị z vào khoảng (0, 1), đại diện cho xác suất.

Hình 2.1 Công thức cơ bản của phân loại nhị phân

Đối với bài toán đa lớp, Scikit-learn sử dụng Multinomial Logistic Regression (softmax regression):

$$P(y = k \mid X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Hình 2.2 Công thức dành cho bài toán đa lớp

Mô hình tối ưu hóa các trọng số bằng cách cực tiểu hóa hàm mất mát cross-entropy loss thông qua các phương pháp gradient descent thường là L-BFGS hoặc liblinear trong Scikit-learn.

Các tham số chính trong triển khai: Trong notebook ML.ipynb, mô hình Logistic Regression được cấu hình cơ bản:

- solver='lbfgs': Phương pháp tối ưu hóa.
- multi_class='multinomial': Chế độ đa lớp.
- max_iter=1000: Số vòng lặp tối đa.
- random_state=42: Đảm bảo tái lập.

3.1.2 Random Forest Classifier

Random Forest là một thuật toán học ensemble dựa trên phương pháp Bagging kết hợp với Random Subspace Method. Được phát triển bởi Leo Breiman năm 2001, Random Forest xây dựng một "rừng" gồm nhiều decision trees độc lập, sau đó lấy kết quả bỏ phiếu để đưa ra dự đoán cuối cùng.

Nguyên lý hoạt động: Mỗi cây quyết định trong Random Forest được xây dựng theo các bước sau:

1. **Bootstrap sampling**: Từ tập train, lấy mẫu ngẫu nhiên có hoàn lại tạo nhiều tập con khác nhau.
2. **Random feature selection**: Tại mỗi nút phân chia, chỉ chọn ngẫu nhiên một tập con nhỏ các đặc trưng tăng tính đa dạng giữa các cây.
3. **Xây dựng cây đầy đủ**: Không cắt tỉa, mỗi cây phát triển đến mức tối đa.
4. **Dự đoán**:
    - Phân loại: Lấy lớp có số phiếu bầu cao nhất từ tất cả các cây.
    - Xác suất: Trung bình xác suất từ các cây.

Công thức dự đoán xác suất cho lớp k:

$$\hat{P}(y = k \mid X) = \frac{1}{N} \sum_{i=1}^{N} I(t_i(X) = k)$$

trong đó N là số cây, $t_i(X)$ là dự đoán của cây thứ i.

Hình 2.3 Công thức dự đoán xác suất cho lớp

3.1.3 Phương pháp đánh giá

Để đánh giá hiệu suất của các mô hình học máy trong bài toán phân loại đa lớp 7 lớp mức độ béo phì, báo cáo sử dụng một tập hợp các chỉ số đánh giá phổ biến và phù hợp với đặc thù của bài toán: dữ liệu mất cân bằng nhẹ giữa các lớp và cần quan tâm đến khả năng dự đoán chính xác trên từng lớp cụ thể. Các chỉ số được tính toán chủ yếu trên tập kiểm tra sau khi huấn luyện mô hình, đồng thời so sánh với kết quả trên tập huấn luyện để phát hiện hiện tượng overfitting.

Các phương pháp và chỉ số đánh giá chính được áp dụng trong notebook ML.ipynb bao gồm:

3.1.3.1 Accuracy

- Công thức:

$$\text{Accuracy} = \frac{\text{Số mẫu dự đoán đúng}}{\text{Tổng số mẫu}}$$

Hình 2.4 Công thức tín chỉ số accuracy

- Ý nghĩa: Tỷ lệ phần trăm các mẫu được dự đoán đúng trên toàn bộ tập dữ liệu.
- Ưu điểm: Dễ hiểu, trực quan.
- Nhược điểm: Có thể bị đánh giá cao giả tạo nếu dữ liệu mất cân bằng.
- Trong dự án: Được sử dụng làm chỉ số chính để so sánh nhanh giữa Logistic Regression và Random Forest.
    - Logistic Regression: ~0.82–0.85.
    - Random Forest: ~0.8964 trên tập test.

3.1.3.2 Precision, Recall và F1-Score

Do bài toán đa lớp và mất cân bằng nhẹ, các chỉ số này được tính theo hai cách:

- **Per-class metrics**: Precision, Recall, F1 cho từng lớp riêng lẻ.
- **Macro-average**: Trung bình không trọng số của các lớp, đánh giá công bằng giữa các lớp.
- **Weighted-average**: Trung bình có trọng số theo số lượng mẫu mỗi lớp, phản ánh hiệu suất tổng thể gần với accuracy.

Công thức cơ bản:

- Precision = TP / (TP + FP)

- Recall = TP / (TP + FN)
- F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

3.1.3.3 Confusion Matrix

- Ma trận hiển thị số lượng mẫu thực tế so với dự đoán.
- Giúp phát hiện cụ thể các lớp nào bị nhầm lẫn với nhau.
- Sử dụng confusion_matrix kết hợp seaborn.heatmap để vẽ biểu đồ trực quan.

3.2 Code

- Chuẩn bị dữ liệu cho mô hình

```
# Chuẩn bị dữ liệu train/test
X = df.drop(['obesity_level'], axis=1)
y = df['obesity_level']

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

print(f"Train: {X_train.shape[0]} mẫu")
print(f"Test : {X_test.shape[0]} mẫu")
```

✓ 0.0s

```
Train: 16606 mẫu
Test : 4152 mẫu
```

Hình 2.5 Chuẩn bị dữ liệu cho mô hình

- Hàm huấn luyện và in ra kết quả đánh giá

```python
def evaluate_model(model, X_train, X_test, y_train, y_test, label_encoder=None, model_name="Model"):
    print(f"\nHuấn luyện và đánh giá: {model_name}")

    # Huấn luyện mô hình
    model.fit(X_train, y_train)
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Metrics cơ bản
    acc_train = accuracy_score(y_train, y_train_pred)
    acc_test = accuracy_score(y_test, y_test_pred)
    prec = precision_score(y_test, y_test_pred, average='macro')
    rec = recall_score(y_test, y_test_pred, average='macro')
    f1 = f1_score(y_test, y_test_pred, average='macro')

    print(f"\n{model_name} Evaluation:")
    print(f"Accuracy Train: {acc_train:.4f} | Test: {acc_test:.4f}")
    print(f"Precision (macro): {prec:.4f}")
    print(f"Recall (macro): {rec:.4f}")
    print(f"F1-score (macro): {f1:.4f}")

    # Classes
    classes = label_encoder.classes_ if label_encoder else model.classes_
    print("\nClassification Report:")
    print(classification_report(y_test, y_test_pred, target_names=classes))

    # Confusion matrix
    cm = confusion_matrix(y_test, y_test_pred)

    # PR & ROC macro
    y_test_bin = label_binarize(y_test, classes=classes)
    if hasattr(model, "predict_proba"):
        y_score = model.predict_proba(X_test)
    else:
        y_score = model.decision_function(X_test)

    n_classes = len(classes)
    all_recall, mean_precision = np.unique([]), np.zeros(0)
    all_fpr, mean_tpr = np.unique([]), np.zeros(0)

    precision_dict, recall_dict, pr_auc_dict = {}, {}, {}
    fpr_dict, tpr_dict, roc_auc_dict = {}, {}, {}

    for i in range(n_classes):
        precision_dict[i], recall_dict[i], _ = precision_recall_curve(y_test_bin[:, i], y_score[:, i])
        pr_auc_dict[i] = auc(recall_dict[i], precision_dict[i])
        fpr_dict[i], tpr_dict[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
        roc_auc_dict[i] = auc(fpr_dict[i], tpr_dict[i])

    # Vẽ biểu đồ
    fig, axes = plt.subplots(1, 3, figsize=(20, 6))

    # PR Curve
    all_recall = np.unique(np.concatenate([recall_dict[i] for i in range(n_classes)]))
    mean_precision = np.zeros_like(all_recall)
    for i in range(n_classes):
        mean_precision += np.interp(all_recall, recall_dict[i][::-1], precision_dict[i][::-1])
    mean_precision /= n_classes
    axes[0].plot(all_recall, mean_precision, color='deepskyblue', linewidth=2)
    axes[0].set_title('Precision-Recall Curve')
    axes[0].set_xlabel('Recall')
    axes[0].set_ylabel('Precision')

    # ROC Curve
    all_fpr = np.unique(np.concatenate([fpr_dict[i] for i in range(n_classes)]))
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr_dict[i], tpr_dict[i])
    mean_tpr /= n_classes
    axes[1].plot(all_fpr, mean_tpr, color='darkorange', linewidth=2)
    axes[1].plot([0, 1], [0, 1], 'k--')
    axes[1].set_title('ROC Curve')
    axes[1].set_xlabel('False Positive Rate')
    axes[1].set_ylabel('True Positive Rate')

    # Confusion matrix
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[2],
                xticklabels=classes, yticklabels=classes, cbar=False)
    axes[2].set_title('Confusion Matrix')
    axes[2].set_xlabel('Predicted')
    axes[2].set_ylabel('Actual')

    plt.tight_layout()
    plt.show()
```

0.0s Pytho

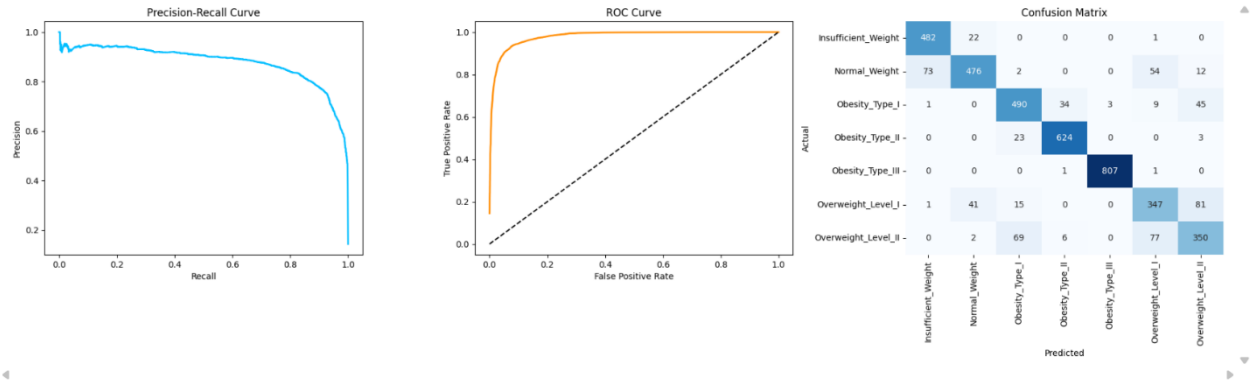
Hình 2.6 Hàm huấn luyện và hiển thị kết quả

- Huấn luyện mô hình LogisticRegression:

```python
# Logistic Regression

lr = LogisticRegression(
    solver='lbfgs',           # Tối ưu hóa cho hàm mất mát
    max_iter=1000,
    class_weight='balanced',  # Xử lý mất cân bằng lớp
    random_state=42,          # Đặt random_state để tái lập kết quả
    n_jobs=-1                 # Sử dụng đa lõi (chỉ ảnh hưởng đến một số solver)
)

pipeline_lr = Pipeline([
    ('preprocessor', preprocessor),  # ← Dùng định nghĩa ở trên
    ('classifier', lr)               # ← Mô hình Logistic Regression
])

# Huấn luyện mô hình
trained_lr = evaluate_model(
    model=pipeline_lr,
    X_train=X_train,
    X_test=X_test,
    y_train=y_train,
    y_test=y_test,
    model_name="Logistic Regression"
)
```

✓ 1.2s Python

```
Huấn luyện và đánh giá: Logistic Regression
c:\Users\84352\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:1184: FutureWarning: 'n_jobs' has no effect since 1.8 
  warnings.warn(msg, category=FutureWarning)

Logistic Regression Evaluation:
Accuracy Train: 0.8604 | Test: 0.8613
Precision (macro): 0.8458
Recall (macro): 0.8479
F1-score (macro): 0.8459

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.87      0.95      0.91       505
      Normal_Weight       0.88      0.77      0.82       617
     Obesity_Type_I       0.82      0.84      0.83       582
    Obesity_Type_II       0.94      0.96      0.95       650
   Obesity_Type_III       1.00      1.00      1.00       809
 Overweight_Level_I       0.71      0.72      0.71       485
Overweight_Level_II       0.71      0.69      0.70       504

           accuracy                           0.86      4152
          macro avg       0.85      0.85      0.85      4152
       weighted avg       0.86      0.86      0.86      4152
```



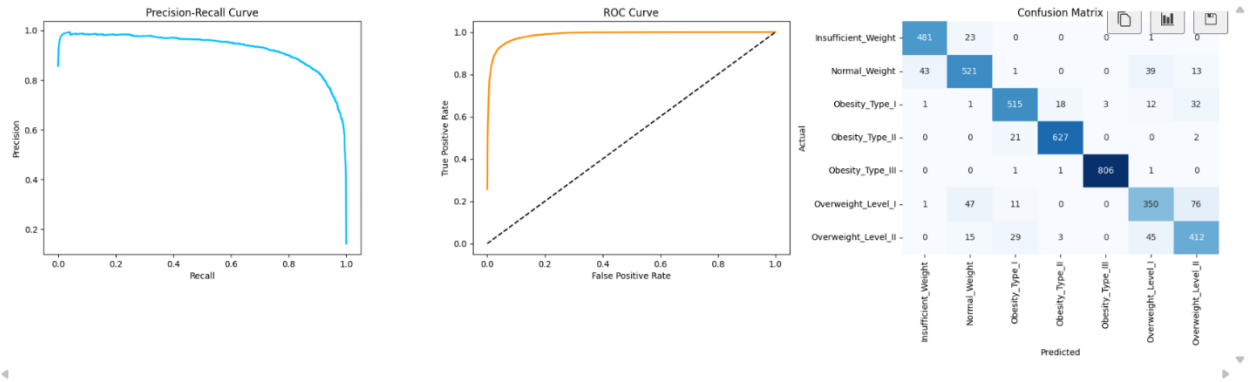
Hình 2.7 Kết quả huấn luyện mô hình logistic regression

- Huấn luyện mô hình Random Forest Classifier

```python
# Random Forest

rf = RandomForestClassifier(
    n_estimators=200,          # Số cây lớn → ổn định hơn
    max_depth=12,              # Giới hạn độ sâu để tránh overfitting
    min_samples_leaf=8,        # Tối thiểu 8 mẫu ở lá → giảm overfitting
    min_samples_split=10,      # Tối thiểu 10 mẫu để split → giảm overfitting
    class_weight='balanced',   # Xử lý mất cân bằng lớp (rất quan trọng!)
    random_state=42,           # Đặt random_state để tái lập kết quả
    n_jobs=-1                  # Sử dụng đa lõi để train nhanh hơn
)

pipeline_rf = Pipeline([
    ('preprocessor', preprocessor),  # ← Dùng định nghĩa ở trên
    ('classifier', rf)               # ← Mô hình Random Forest
])

# Huấn luyện mô hình
trained_rf = evaluate_model(
    model=pipeline_rf,
    X_train=X_train,
    X_test=X_test,
    y_train=y_train,
    y_test=y_test,
    model_name="Random Forest"
)
```

✓ 1.2s Python

```
Huấn luyện và đánh giá: Random Forest

Random Forest Evaluation:
Accuracy Train: 0.9185 | Test: 0.8940
Precision (macro): 0.8825
Recall (macro): 0.8831
F1-score (macro): 0.8825

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.91      0.95      0.93       505
      Normal_Weight       0.86      0.84      0.85       617
     Obesity_Type_I       0.89      0.88      0.89       582
    Obesity_Type_II       0.97      0.96      0.97       650
   Obesity_Type_III       1.00      1.00      1.00       809
 Overweight_Level_I       0.78      0.72      0.75       485
Overweight_Level_II       0.77      0.82      0.79       504

           accuracy                           0.89      4152
          macro avg       0.88      0.88      0.88      4152
       weighted avg       0.89      0.89      0.89      4152
```



Hình 2.8 Kết quả huấn luyện mô hình random forest classfier

TÀI LIỆU THAM KHẢO

1. Scikit-learn Developers 2025. Scikit-learn: Machine Learning in Python.
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.
3. McKinney, W 2010. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 56–61.
4. Hunter, J. D 2007. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering.
5. Waskom, M. L 2021. seaborn: statistical data visualization. Journal of Open Source Software.
6. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D & Oliphant, T. E 2020. Array programming with NumPy. Nature.
7. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X 2013. Applied Logistic Regression. John Wiley & Sons.
8. Streamlit Inc 2025. Streamlit Documentation. Truy cập ngày 04 tháng 01 năm 2026.
9. Palczewski, M 2020. Obesity Level Prediction Dataset. Kaggle.
10. Tổ chức Y tế Thế giới WHO 2024. Obesity and overweight. Fact sheet.
11. Nguyễn Văn Hậu. Giáo trình Học máy cơ bản dành cho ngành Khoa học Máy tính. Trường Đại học Sư phạm Kỹ thuật Hưng Yên.