

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP LỚN
MÔN : HỌC MÁY

ĐỀ TÀI:
XÂY DỰNG MÔ HÌNH DỰ ĐOÁN DOANH SỐ DỰA TRÊN
PHƯƠNG PHÁP QUẢNG CÁO

Giảng viên hướng dẫn: ThS.Trần Anh Đạt

Nhóm sinh viên thực hiện: Nhóm 10_64CNTT3

Họ và tên sinh viên	Mã sinh viên
Nguyễn Năng Anh	2251061705
Vũ Thị Thanh	2251061883
Ma Thị Tuyền	2251061919
Lưu Kim Thi	2251061889

MỤC LỤC

I. Giới thiệu bài toán	4
II. Thu thập và chuẩn bị dữ liệu	5
1. Khám phá dữ liệu	5
1.1 Thông Tin Chi Tiết về DataFrame	6
1.2 Kiểm tra số lượng giá trị duy nhất	7
1.3 Xác định các đặc trưng là giá trị phân loại hay giá trị số	8
1.4 Hiển thị các thông kê mô tả	9
2. Phân tích dữ liệu khám phá	11
2.1 Biểu đồ phân phối biên mục tiêu	11
2.2 Trực quan hóa các đặc điểm	13
3. Tiền xử lý dữ liệu	15
3.1 Xóa bất kỳ hàng nào trùng lặp	15
3.2 Kiểm tra giá trị null	17
3.3 Chia tệp dữ liệu/loại bỏ giá trị ngoại lai/chuẩn hóa	18
4. Lựa chọn và chiết suất tính năng	25
III. Tìm hiểu thuật toán	27
1. Tổng quan	27
2. Mô hình hồi quy tuyến tính	29
2.1 Lý do chọn mô hình	29
2.2 Tổng quan về mô hình hồi quy tuyến tính	29
2.3 Xây dựng mô hình	32
2.4 Đánh giá mô hình	33
3. Hồi quy Ridge	36
3.1 Lý do chọn mô hình	36
3.2 Tổng quan về mô hình Ridge	39
3.3 Huấn luyện mô hình	41
3.4 Đánh giá mô hình	47
4. Neural Network	49
4.1 Lý do chọn mô hình	49
4.2 Tổng quan về mô hình Neural network	49
4.3 Huấn luyện mô hình	57
4.4 Đánh giá mô hình	60
5. Stacking	64
5.1 Lý do chọn mô hình	64
5.2 Tổng quan về mô hình stacking	64
5.3 Huấn luyện mô hình	66
5.4 Đánh giá mô hình	74
IV. Chương trình demo	76

LỜI MỞ ĐẦU

Trong nền kinh tế hiện đại, sự phát triển không ngừng của công nghệ thông tin đã mang lại những thay đổi lớn trong cách thức doanh nghiệp tiếp cận và tương tác với khách hàng. Quảng cáo không chỉ còn là công cụ để giới thiệu sản phẩm hay dịch vụ mà đã trở thành một yếu tố quan trọng giúp doanh nghiệp đạt được doanh số và duy trì lợi thế cạnh tranh trên thị trường. Với sự đa dạng của các phương thức quảng cáo như truyền hình, báo chí, mạng xã hội, và quảng cáo trực tuyến, việc xác định kênh nào mang lại hiệu quả tốt nhất không phải là điều dễ dàng. Bên cạnh đó, tối ưu hóa ngân sách quảng cáo để đạt được doanh số cao nhất cũng là một thách thức lớn cho các nhà quản lý.

Sự phát triển của các mô hình học máy đã mở ra cơ hội mới cho việc dự đoán doanh số dựa trên các phương pháp quảng cáo. Các mô hình này giúp khai thác và phân tích dữ liệu quảng cáo, cho phép doanh nghiệp dự đoán doanh số một cách chính xác hơn dựa trên các yếu tố đầu vào cụ thể. Với khả năng này, các mô hình học máy không chỉ giúp tối ưu hóa các chiến dịch quảng cáo mà còn mang lại cái nhìn sâu sắc hơn về cách mà từng phương thức quảng cáo ảnh hưởng đến doanh thu của doanh nghiệp. Trong nghiên cứu này, chúng em tập trung vào việc xây dựng và đánh giá các mô hình dự đoán doanh số với hy vọng tìm ra phương pháp tối ưu nhất để hỗ trợ doanh nghiệp đưa ra các quyết định chiến lược.

Chính vì vậy, nhóm chúng em đã quyết định chọn đề tài “Xây dựng mô hình dự đoán doanh số dựa trên các phương pháp quảng cáo” cho bài tập lớn lần này. Với mong muốn áp dụng những kiến thức đã học trong lĩnh vực học máy và khai phá dữ liệu, nhóm em hy vọng có thể góp phần giải quyết những thách thức thực tiễn trong kinh doanh.

Là sinh viên ngành Công nghệ Thông tin, chúng em nhận thức rõ tầm quan trọng của việc áp dụng công nghệ vào các tình huống thực tế và tin rằng bài tập này là cơ hội quý báu để chúng em nâng cao kỹ năng trong lĩnh vực dự đoán và phân tích dữ liệu. Thông qua việc áp dụng các mô hình học máy, như Hồi quy tuyến tính, Ridge Regression, MLP Regression và phương pháp stacking, chúng em mong muốn đánh giá được hiệu quả của từng mô hình và tìm ra mô hình tối ưu nhất để dự đoán doanh số.

Tuy nhiên, do kiến thức còn hạn chế và thời gian thực hiện đề tài có hạn, bài làm của nhóm khó tránh khỏi những thiếu sót và hạn chế nhất định. Rất mong nhận được sự góp ý và hỗ

trợ từ thầy/cô để chúng em có thể hoàn thiện và nâng cao chất lượng của đề tài. Sự hướng dẫn và đóng góp ý kiến của thầy/cô không chỉ giúp nhóm em phát triển bài làm này mà còn là nguồn động lực quan trọng để chúng em hoàn thiện kỹ năng nghiên cứu và phân tích trong lĩnh vực học máy.

Nhóm em xin chân thành cảm ơn!

I. Giới thiệu bài toán

Mô tả bài toán

Trong bài toán này, mục tiêu là dự đoán doanh số bán hàng dựa trên các phương pháp quảng cáo khác nhau. Bằng cách phân tích dữ liệu quảng cáo và doanh số, chúng ta có thể ước tính doanh thu dự kiến từ mỗi chiến lược quảng cáo và đánh giá mức độ ảnh hưởng của từng phương pháp quảng cáo đến kết quả doanh số. Việc này có thể giúp các doanh nghiệp tối ưu hóa chi phí quảng cáo và tối đa hóa doanh thu.

Input bài toán

Bảng thống kê bao gồm các dữ liệu liên quan đến sản phẩm và các phương pháp quảng cáo được sử dụng. Cụ thể:

- **Các cột đặc trưng là:** TV Ad Budget, Radio Ad Budget, Newspaper Ad Budget
- **Biến mục tiêu:** Doanh số (sales), tức là lượng doanh thu đạt được từ các phương pháp quảng cáo đó.

TV_Ad_Budget	Radio_Ad_Budg	Newspaper_Ad	Sales_(\$)
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1	4.8
199.8	2.6	21.2	10.6
66.1	5.8	24.2	8.6
214.7	24	4	17.4
23.8	35.1	65.9	9.2
97.5	7.6	7.2	9.7
204.1	32.9	46	19
195.4	47.7	52.9	22.4
67.8	36.6	114	12.5
281.4	39.6	55.8	24.4
69.2	20.5	18.3	11.3
147.3	23.9	19.1	14.6

Output

Dự đoán doanh số dựa trên các phương pháp quảng cáo

Tóm tắt công việc thực hiện:

Sử dụng các phương pháp hồi quy để huấn luyện mô hình và đánh giá hiệu suất, cụ thể gồm 4 phương pháp chính:

- Hồi quy tuyến tính (Linear Regression): Một mô hình đơn giản nhưng hiệu quả trong việc dự đoán mối quan hệ tuyến tính giữa doanh số và các đặc trưng quảng cáo.
- Hồi quy Ridge (Ridge Regression): Giống với hồi quy tuyến tính nhưng thêm vào một hệ số phạt nhằm giảm thiểu tình trạng đa cộng tuyến, giúp mô hình có khả năng dự đoán ổn định hơn khi có các đặc trưng tương quan.
- Neural Network (MLP Regressor): Một mô hình mạng nơ-ron đa lớp, cho phép mô hình hóa các mối quan hệ phi tuyến phức tạp giữa doanh số và các đặc trưng quảng cáo.
- Stacking: Kết hợp dự đoán từ nhiều mô hình để tạo ra một dự đoán tổng hợp tốt hơn, giúp cải thiện hiệu suất mô hình thông qua việc kết hợp các mô hình mạnh.

II. Thu thập và chuẩn bị dữ liệu

1. Khám phá dữ liệu

Thông tin dữ liệu tại đây : [Advertisement Budget Prediction - \(Top ML Models\)](#)



Khám phá dữ liệu là bước khởi đầu không thể thiếu trong bất kỳ quy trình phân tích dữ liệu nào. Qua việc khai thác thông tin chi tiết từ DataFrame, chúng ta có thể hiểu rõ hơn về bản chất của dữ liệu mà mình đang làm việc. Quá trình này không chỉ giúp xác định cấu trúc và loại hình dữ liệu, mà còn phát hiện ra những mẫu, xu hướng, và sự bất thường có thể ảnh hưởng đến các phân tích sau này. Bằng cách sử dụng các phương pháp thống kê mô tả và trực quan hóa, chúng ta có thể vẽ ra bức tranh toàn cảnh về dữ liệu, từ đó đặt nền tảng cho

việc ra quyết định và xây dựng mô hình chính xác hơn. Khám phá dữ liệu không chỉ đơn thuần là tìm kiếm thông tin; đó còn là việc tìm ra những câu chuyện ẩn giấu trong dữ liệu, giúp ta đưa ra những hiểu biết sâu sắc và giá trị hơn trong các nghiên cứu và dự đoán.

Khám phá dữ liệu nhóm em sẽ làm gồm 4 bước:

- 1) Thông tin chi tiết về dataframe
- 2) Kiểm tra số lượng giá trị duy nhất
- 3) Xác định đặc trưng giá trị phân loại hay giá trị số
- 4) Tổng hợp và hiển thị các thống kê

1.1 Thông Tin Chi Tiết về DataFrame

Bằng cách chạy đoạn code sau, chúng ta có thể xem xét thông tin chi tiết về DataFrame, từ đó hiểu rõ hơn về cấu trúc dữ liệu đang làm việc

```
df = pd.read_csv('../Data/Advertising Budget and Sales.csv', index_col=0, names=['TV', 'Radio', 'Newspaper', 'Sales'], skiprows=1)
df.info()
```

Đọc dữ liệu từ file CSV:

1. **pd.read_csv**: Hàm này được sử dụng để đọc dữ liệu từ một file CSV và chuyển đổi nó thành một Data Frame của Pandas.
2. **'../Data/Advertising Budget and Sales.csv'**: Đây là đường dẫn đến file CSV mà bạn muốn đọc.
3. **index_col=0**: Điều này chỉ định rằng cột đầu tiên của file CSV (có chỉ số 0) sẽ được sử dụng làm chỉ số cho DataFrame, thay vì tạo một chỉ số mặc định.
4. **names=['TV', 'Radio', 'Newspaper', 'Sales']**: Tham số này cho phép chỉ định tên cho các cột trong DataFrame. Bằng cách này, các cột sẽ được đặt tên là 'TV', 'Radio', 'Newspaper', và 'Sales'.
5. **skiprows=1**: Tham số này cho phép bỏ qua dòng đầu tiên của file CSV khi đọc dữ liệu. Dòng này có thể chứa tiêu đề hoặc thông tin không cần thiết.

Hiển thị thông tin về DataFrame:

❖ **df.info()**: Hàm này được sử dụng để in ra thông tin chi tiết về DataFrame, bao gồm:

- Số lượng dòng và cột trong DataFrame.
- Tên của các cột và loại dữ liệu của chúng (dtype).
- Số lượng giá trị không null trong mỗi cột.
- Sử dụng bộ nhớ của DataFrame.

```
<class 'pandas.core.frame.DataFrame'>
Index: 200 entries, 1 to 200
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   TV           200 non-null    float64
1   Radio        200 non-null    float64
2   Newspaper    200 non-null    float64
3   Sales        200 non-null    float64
dtypes: float64(4)
memory usage: 7.8 KB
```

Từ hình kết quả trên ta có nhận xét như sau :

- Tập dữ liệu có 200 mẫu với 4 cột :
 1. **TV_Ad_Budget** : Ngân sách quảng cáo trên truyền hình
 2. **Radio_Ad_Budget** : Ngân sách quảng cáo trên đài phát thanh
 3. **Newspaper_Ad_Budget** : Ngân sách quảng cáo trên báo
 4. **Sales** : Doanh thu bán hàng
- Tất cả các cột đều có **200 giá trị không rỗng**, có nghĩa là không có giá trị nào bị thiếu.
- Tất cả 4 cột đều có kiểu dữ liệu là float

1.2 Kiểm tra số lượng giá trị duy nhất

Khi chạy đoạn code **df.nunique().sort_values()**, chúng ta sẽ nhận được số lượng các giá trị duy nhất trong từng cột của Data Frame và sắp xếp chúng theo thứ tự tăng dần.

```
Sales      121
Radio       167
Newspaper   172
TV          190
dtype: int64
```


Dựa vào hình trên, chúng ta có thể rút ra các nhận xét sau:

1. **Sales:** Cột này có 121 giá trị duy nhất, cho thấy rằng doanh số không phải là một giá trị liên tục và có thể có nhiều mức khác nhau, nhưng vẫn có sự lặp lại giữa các bản ghi.
2. **Radio:** 167: Cột ngân sách quảng cáo trên radio có 167 giá trị duy nhất, cho thấy sự đa dạng trong các mức ngân sách quảng cáo đã được sử dụng, tuy nhiên vẫn có nhiều mức ngân sách tương đồng.
3. **Newspaper:** Cột ngân sách quảng cáo trên báo có 172 giá trị duy nhất, cho thấy cũng có sự đa dạng trong ngân sách quảng cáo cho báo, tương tự như radio.
4. **TV:** Cột ngân sách quảng cáo trên truyền hình có số lượng giá trị duy nhất cao nhất (190), cho thấy rằng có nhiều mức ngân sách khác nhau đã được áp dụng cho quảng cáo trên truyền hình, gần như mỗi mức ngân sách đều là duy nhất.

1.3 Xác định các đặc trưng là giá trị phân loại hay giá trị số

Nhóm em xác định các đặc trưng là giá trị phân loại hay giá trị số để lựa chọn mô hình học máy phù hợp và thực hiện các bước tiền xử lý dữ liệu hiệu quả. Điều này rất quan trọng vì nó ảnh hưởng đến cách tính toán các chỉ số thống kê cũng như phương pháp trực quan hóa dữ liệu. Việc phân loại này giúp nhóm em tối ưu hóa quy trình phân tích và nâng cao hiệu quả của mô hình học máy.

```
nu = df[features].nunique().sort_values()
nf = [] # Danh sách lưu các đặc trưng số học
cf = [] # Danh sách lưu các đặc trưng phân loại
nnf = 0
ncf = 0

# Duyệt qua tất cả các cột đặc trưng để phân loại thành số học
hoặc phân loại
for i in range(df[features].shape[1]):
    # Nếu cột có ≤ 16 giá trị duy nhất, coi nó là đặc trưng phân
    loại (categorical)
    if nu.values[i] <= 16:
        cf.append(nu.index[i])
    # Nếu cột có > 16 giá trị duy nhất, coi nó là đặc trưng số
```

```
học (numerical)
else:
    nf.append(nu.index[i])
# In ra k ết quả, hiển thị số lượng đặc trưng số học và phân loại
print('\n\033[1mK ết luận:\033[0m Bộ dữ liệu có {} đặc trưng số
học và {} đặc trưng phân loại.'.format(len(nf), len(cf)))
```

Bước 1: Đếm giá trị duy nhất: Dùng `nunique()` và `sort_values()`.

Bước 2: Khởi tạo danh sách: Tạo danh sách để lưu các đặc trưng.

Bước 3: Phân loại: Duyệt qua cột; nếu ≤ 16 giá trị duy nhất, thêm vào đặc trưng phân loại; nếu > 16 , thêm vào đặc trưng số học.

Bước 4: In kết quả: Hiển thị số lượng đặc trưng số học và phân loại.

KẾT QUẢ: Bộ dữ liệu có 3 đặc trưng và 0 đặc trưng phân loại

1.4 Hiện thị các thông kê mô tả

Khi thực thi `display(df.describe())` chúng ta sẽ thấy được bảng thống kê tóm tắt các đặc trưng số học trong DataFrame, bao gồm số lượng giá trị (count), giá trị trung bình (mean), độ lệch chuẩn (std), cũng như các giá trị nhỏ nhất, lớn nhất và các phân vị quan trọng (25%, 50%, 75%). Thông qua bảng này, nhóm có thể hiểu rõ hơn về mức độ tập trung và phân tán của dữ liệu, cũng như nhanh chóng nhận diện các điểm bất thường và xu hướng tổng quan của từng đặc trưng. Bước này là tiền đề quan trọng để tiến hành các phân tích sâu hơn hoặc điều chỉnh dữ liệu phù hợp cho mô hình dự báo.

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

Từ hình trên ta có thể có nhận xét như sau

1. Số lượng quan sát (Count)

Tất cả các cột đều có 200.000000 giá trị không null. Điều này cho thấy rằng bộ dữ liệu đầy đủ và không có giá trị thiếu. Điều này là rất quan trọng trong phân tích dữ liệu, vì giá trị thiếu có thể ảnh hưởng đến tính chính xác của các mô hình phân tích sau này.

2. Giá trị trung bình (Mean)

- Ngân sách quảng cáo trên TV trung bình cao hơn nhiều so với ngân sách quảng cáo trên Radio và Newspaper. Điều này có thể cho thấy rằng các doanh nghiệp ưu tiên quảng cáo trên TV hơn để đạt được doanh thu cao hơn.

- Doanh số trung bình là 14.02, cho thấy rằng mặc dù ngân sách quảng cáo khác nhau, doanh thu từ sản phẩm này vẫn đạt một mức trung bình nhất định.

3. Độ lệch chuẩn (Standard Deviation)

- Độ lệch chuẩn cao nhất ở TV cho thấy sự biến động lớn trong ngân sách quảng cáo, tức là một số doanh nghiệp có thể chi tiêu rất nhiều cho quảng cáo trên TV, trong khi những doanh nghiệp khác có thể chi tiêu rất ít.

- Độ lệch chuẩn thấp hơn cho Radio và Newspaper có thể chỉ ra rằng ngân sách quảng cáo trên các kênh này có xu hướng ổn định hơn.

4. Giá trị nhỏ nhất và lớn nhất (Min/Max)

- Giá trị tối thiểu cho TV là rất thấp (0.7), nhưng giá trị tối đa lên đến 296.4 cho thấy sự chênh lệch lớn trong cách các doanh nghiệp phân bổ ngân sách quảng cáo.

- Radio có giá trị tối thiểu là 0, cho thấy một số doanh nghiệp không đầu tư vào quảng cáo qua kênh này. Tương tự, Newspaper cũng có giá trị tối thiểu không đáng kể.

5. Phân vị (Percentiles)

- Phân vị thứ 25, 50 và 75 cho thấy rằng 25% các doanh nghiệp chi tiêu dưới 74.38 cho quảng cáo TV, trong khi 75% chi tiêu dưới 218.83. Điều này cho thấy sự tập trung lớn vào ngân sách TV trong nhóm doanh nghiệp này.

- Các giá trị phân vị cho Radio và Newspaper cũng cho thấy rằng ngân sách chi tiêu có xu hướng thấp hơn so với TV, cho thấy TV là kênh quảng cáo chính.

6. Doanh số (Sales)

Doanh số có sự biến động, nhưng không quá lớn so với ngân sách quảng cáo. Điều này cho thấy rằng mặc dù ngân sách quảng cáo có sự khác biệt lớn, nhưng doanh thu từ sản phẩm có thể không thay đổi nhiều. Điều này có thể là do nhiều yếu tố khác nhau ảnh hưởng đến doanh số mà không chỉ dựa vào ngân sách quảng cáo.

=> Bảng thống kê mô tả cung cấp cái nhìn tổng quan về dữ liệu, cho thấy sự khác biệt lớn trong cách các doanh nghiệp đầu tư vào quảng cáo qua các kênh khác nhau và mối quan hệ giữa ngân sách quảng cáo và doanh thu. Phân tích kỹ lưỡng các thông số này là cần thiết để xây dựng các mô hình dự đoán doanh thu hiệu quả hơn trong tương lai. Sự hiểu biết về cách ngân sách quảng cáo ảnh hưởng đến doanh số có thể giúp các doanh nghiệp tối ưu hóa chiến lược quảng cáo của mình để đạt được hiệu quả cao nhất.

2. Phân tích dữ liệu khám phá

2.1 Biểu đồ phân phối biên mục tiêu

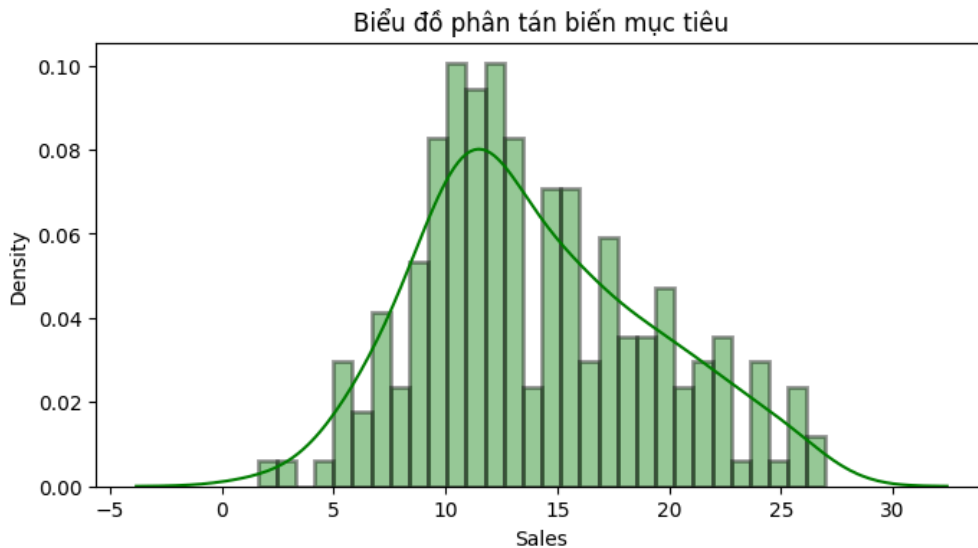
Vẽ biểu đồ phân phối biên mục tiêu để hiểu rõ hơn về cách mà biến này phân bố trong tập dữ liệu. Biểu đồ phân phối giúp chúng ta nhận diện các đặc điểm quan trọng như độ lệch, độ phân tán và sự tập trung của các giá trị. Qua việc phân tích biểu đồ này, nhóm chúng ta có thể phát hiện ra các giá trị bất thường (outliers) cũng như sự bất đối xứng trong phân phối, từ đó đánh giá ảnh hưởng của chúng đến việc xây dựng mô hình. Điều này rất quan trọng để đảm bảo rằng mô hình học máy mà nhóm em phát triển sẽ hoạt động hiệu quả và chính xác, góp phần nâng cao khả năng dự đoán trong các ứng dụng thực tế.

Chạy đoạn code sau chúng ta sẽ

```
plt.figure(figsize=[8,4])
sns.distplot(df[target],
color='g',hist_kws=dict(edgecolor="black", linewidth=2),
bins=30)
```

```
plt.title('Target Variable Distribution - Median Value of Homes  
($1Ms) ')  
plt.show()
```

Biểu đồ phân tán biến mục tiêu



Từ hình trên ta có nhận xét như sau:

1. **Phân phối:** Biến "Sales" có phân phối lệch phải (right-skewed) hoặc phân phối dương. Điều này có nghĩa là phần lớn các giá trị tập trung ở phía bên trái của biểu đồ (giá trị nhỏ hơn), và có một số ít các giá trị lớn kéo dài về phía bên phải.
2. **Giá trị trung bình:** Giá trị trung bình của biến "Sales" nằm trong khoảng từ 10 đến 15 (dựa trên đỉnh của đường cong mật độ). Điều này cho thấy doanh số trung bình của các đơn vị dữ liệu rơi vào khoảng này.
3. **Độ phân tán:** Dữ liệu có độ phân tán tương đối lớn, thể hiện qua các cột của biểu đồ có chiều cao khác nhau. Điều này cho thấy doanh số của các đơn vị dữ liệu khá khác biệt nhau, không đồng đều.
4. **Giá trị ngoại lệ:** Có thể có một số giá trị ngoại lệ ở phía bên phải của biểu đồ, tức là những đơn vị có doanh số rất cao so với phần còn lại.

2.2 Trực quan hóa các đặc điểm

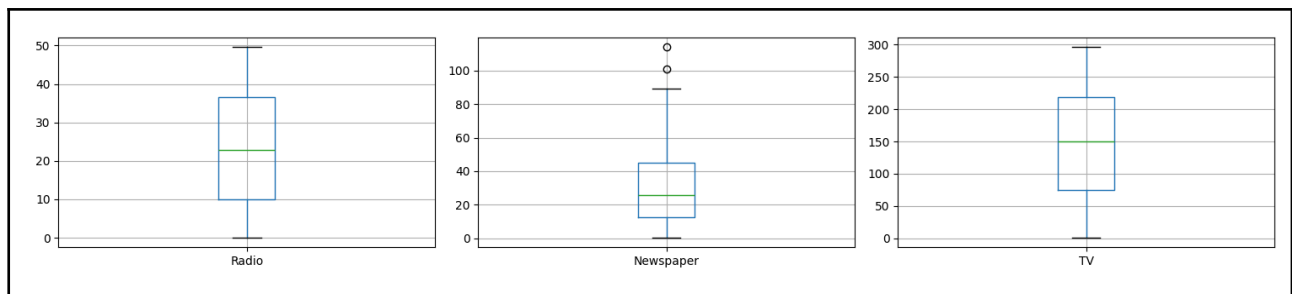
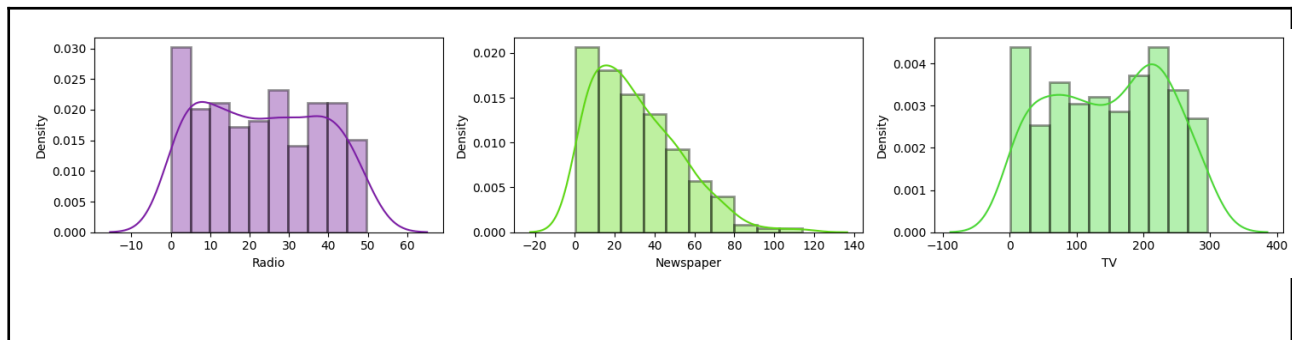
Đoạn code dưới đây giúp chúng ta trực quan hóa phân bố của các đặc điểm số bằng cách tạo các biểu đồ phân phối và biểu đồ hộp (boxplot). Những biểu đồ này hỗ trợ đánh giá tổng quan về phân bố giá trị của từng đặc điểm, phát hiện ra các xu hướng, phạm vi dữ liệu, cũng như các giá trị ngoại lai.

```
• # Trực quan hóa phân bố các đặc điểm số
• print('\033[1mPhân Bố Các Đặc Điểm Số'.center(130))
•
• n = 3 # Số lượng cột để hiển thị trong mỗi hàng
• plt.figure(figsize=[15, 3 * math.ceil(len(nf) / n)]) # Thiết lập kích thước hình
•
• # Vẽ biểu đồ phân bố cho từng đặc điểm số
• for i in range(len(nf)):
•     plt.subplot(math.ceil(len(nf) / 3), n, i + 1) # Tạo bố cục cho các biểu đồ
•     sns.distplot(df[nf[i]], hist_kws=dict(edgecolor="black", linewidth=2), bins=10, color=list(np.random.randint([255,255,255])/255))
•
• plt.tight_layout() # Điều chỉnh khoảng cách giữa các biểu đồ
• plt.show() # Hiển thị các biểu đồ
•
• # Trực quan hóa các giá trị bất thường (outliers) cho các đặc điểm số
• plt.figure(figsize=[15, 3 * math.ceil(len(nf) / n)]) # Thiết lập kích thước hình
• for i in range(len(nf)):
•     plt.subplot(math.ceil(len(nf) / 3), n, i + 1) # Tạo bố cục cho các boxplot
•     df.boxplot(nf[i]) # Vẽ boxplot cho từng đặc điểm số
•
```

- `plt.tight_layout()` # Điều chỉnh khoảng cách giữa các boxplot
- `plt.show()` # Hiển thị boxplots

Điều chỉnh khoảng cách giữa các biểu đồ: Hàm *plt.tight_layout()* được sử dụng để tự động điều chỉnh khoảng cách giữa các biểu đồ, nhằm đảm bảo rằng chúng không bị chồng chéo và dễ nhìn hơn.

Hiển thị biểu đồ: Cuối cùng, hàm *plt.show()* được gọi để hiển thị tất cả các biểu đồ đã được tạo ra. Nếu không có dòng này, các biểu đồ sẽ không được hiển thị trong môi trường tương tác.



Dựa các biểu đồ trên ta có nhận xét

Biểu đồ phân tích biểu đồ phân bố mật độ

1. **Radio:** Phân bố chi tiêu khá đồng đều ở các mức chi khác nhau, với đỉnh điểm nằm khoảng 0-10. Đường mật độ cho thấy phân phối tương đối bằng phẳng với sự dao động nhẹ trong chi tiêu, không có giá trị quá cao hoặc quá thấp.
2. **Báo chí:** Phân phối này bị lệch phải, với phần lớn chi tiêu tập trung ở mức thấp (0-30). Mật độ giảm dần khi chi tiêu tăng lên, cho thấy chi tiêu cao hơn trên báo chí là ít phổ biến hơn.

3. **TV:** Phân bố chi tiêu có vẻ là đa đỉnh, với nhiều đỉnh khác nhau, cho thấy có các mức chi tiêu cụ thể mà các nhà quảng cáo ưa chuộng. Đường mật độ cho thấy sự biến động trong chi tiêu, với các cụm tập trung khoảng 100-150 và 200-250.

Biểu đồ hộp:

Nhận xét chung:

- Doanh số (Sales): Biểu đồ cho thấy sự phân bố tương đối đồng đều của doanh số, không có quá nhiều điểm dữ liệu ngoại lệ. Trung vị của doanh số nằm ở mức khá thấp, cho thấy doanh số của các sản phẩm không quá cao.
- Ngân sách quảng cáo (TV, Radio, Newspaper): Ngân sách quảng cáo trên TV có sự phân tán rộng nhất, nghĩa là có những sản phẩm được đầu tư quảng cáo trên TV rất nhiều và cũng có những sản phẩm đầu tư rất ít. Ngân sách quảng cáo trên radio và báo cũng có sự phân tán nhất định, nhưng không rõ rệt bằng TV.

Kiểm tra ngoại lệ:

Phương pháp: Để xác định ngoại lệ, ta thường sử dụng quy tắc 1.5 lần khoảng tứ phân vị (IQR). Bất kỳ điểm dữ liệu nào nằm ngoài khoảng $[Q1 - 1.5IQR, Q3 + 1.5IQR]$ đều được coi là ngoại lệ.

=> Từ hình ảnh ta có thể thấy được có vài điểm ngoại lệ

3. Tiền xử lý dữ liệu

3.1 Xóa bất kỳ hàng nào trùng lặp

Đoạn code sau giúp kiểm tra và loại bỏ các hàng trùng lặp trong bộ dữ liệu. Trước tiên, chúng ta lưu kích thước ban đầu của Data Frame để làm cơ sở so sánh. Sau khi xóa các hàng trùng lặp (nếu có), chúng ta sẽ so sánh kích thước sau xử lý với kích thước ban đầu. Nếu không có thay đổi về kích thước, điều này cho thấy dữ liệu không chứa bản sao nào; ngược lại, nếu kích thước giảm đi, ta có thể xác định số lượng hàng trùng lặp đã bị loại bỏ hoặc điều chỉnh.


```
• counter = 0
• rs, cs = original_df.shape # Lưu kích thước ban đầu của DataFrame
• df.drop_duplicates(inplace=True) # Xóa các hàng trùng lặp
• # Kiểm tra xem kích thước của DataFrame có thay đổi hay không
• if df.shape == (rs, cs):
•     print('\n\033[1mInference:\033[0m Bộ dữ liệu không có bất kỳ bản sao nào')
• else:
•     print(f'\n\033[1mInference:\033[0m Số lượng bản sao đã loại bỏ/sửa chữa ---> {rs - df.shape[0]}')
```

Lưu kích thước ban đầu của DataFrame: Dòng mã `rs, cs = original_df.shape` được sử dụng để lưu kích thước ban đầu của Data Frame *original_df* vào hai biến: *rs* và *cs*. Ở đây, *rs* sẽ chứa số lượng hàng (rows) và *cs* sẽ chứa số lượng cột (columns) của DataFrame. Điều này rất hữu ích để kiểm tra xem có hàng nào bị xóa sau khi loại bỏ các bản sao.

Xóa các hàng trùng lặp: Hàm `df.drop_duplicates(inplace=True)` được gọi để xóa tất cả các hàng trùng lặp trong DataFrame *df*. Tham số *inplace=True* chỉ định rằng việc xóa sẽ được thực hiện trực tiếp trên DataFrame mà không cần tạo một bản sao mới. Điều này giúp giảm thiểu việc sử dụng bộ nhớ và đảm bảo rằng DataFrame *df* sẽ chỉ chứa các hàng duy nhất.

Kiểm tra kích thước của DataFrame: Đoạn mã tiếp theo `if df.shape == (rs, cs)` kiểm tra xem kích thước hiện tại của DataFrame *df* có giống như kích thước ban đầu (lưu trong *rs* và *cs*) hay không. Điều này giúp xác định liệu có bất kỳ hàng nào bị xóa do trùng lặp hay không.

In thông báo kết quả:

- Nếu kích thước không thay đổi (có nghĩa là không có hàng trùng lặp), câu lệnh `print('\n\033[1mInference:\033[0m Bộ dữ liệu không có bất kỳ bản sao nào')` sẽ được thực thi. Dòng thông báo này sẽ in ra rằng bộ dữ liệu không chứa bất kỳ bản sao nào, giúp người dùng biết rằng dữ liệu đã được làm sạch.

- Nếu kích thước đã thay đổi (có hàng trùng lặp bị xóa), câu lệnh print còn lại sẽ hiển thị số lượng bản sao đã bị loại bỏ. Câu lệnh này sử dụng định dạng chuỗi f-string để tính toán và in ra số lượng hàng đã bị xóa (tính bằng hiệu số giữa số hàng ban đầu và số hàng hiện tại).

KẾT QUẢ: Bộ dữ liệu không có bất kỳ bản sao nào

3.2 Kiểm tra giá trị null

Chúng ta kiểm tra giá trị null của tập dữ liệu vì nếu tập dữ liệu chứa các giá trị null, chúng có thể ảnh hưởng đến quá trình phân tích và xây dựng mô hình. Các giá trị null có thể làm giảm tính chính xác của mô hình, gây ra sai số trong dự đoán, hoặc thậm chí dẫn đến việc không thể thực hiện một số phép toán thống kê. Đoạn code dưới đây sẽ kiểm tra giá trị null

```
• # Kiểm tra các phần tử null
• nvc = pd.DataFrame(df.isnull().sum().sort_values(),
  columns=['Tổng số giá trị Null'])
• nvc['Tỷ lệ phần trăm'] = round(nvc['Tổng số giá trị Null'] /
  df.shape[0], 3) * 100
• print(nvc)
```

Kiểm tra các phần tử null:

Dòng mã `nvc = pd.DataFrame(df.isnull().sum().sort_values(), columns=['Tổng số giá trị Null'])` được sử dụng để kiểm tra và tổng hợp số lượng các giá trị null (hoặc NaN) trong DataFrame **df**. Cụ thể:

1. **df.isnull():** Hàm này sẽ trả về một Data Frame mới với các giá trị boolean, nơi mà giá trị **True** chỉ ra rằng phần tử tương ứng là null.
2. **sum():** Hàm này được áp dụng để tính tổng số lượng giá trị null cho mỗi cột trong DataFrame.
3. **sort_values():** Hàm này sắp xếp kết quả theo thứ tự tăng dần, giúp dễ dàng nhận diện các cột có nhiều giá trị null nhất.

4. **pd.DataFrame(..., columns=['Tổng số giá trị Null']):** Kết quả được chuyển đổi thành một Data Frame mới có tên **nvc**, trong đó có một cột duy nhất mang tên "Tổng số giá trị Null".

	Tổng số giá trị Null	Tỷ lệ phần trăm
TV	0	0.0
Radio	0	0.0
Newspaper	0	0.0
Sales	0	0.0

Từ hình hình ảnh trên ta có kết luận như sau:

Dữ liệu không có giá trị Null trong bất kỳ cột nào (TV, Radio, Newspaper, và Sales), cho thấy độ đầy đủ và toàn vẹn của dữ liệu. Điều này giúp đảm bảo quá trình phân tích và xây dựng mô hình không bị ảnh hưởng bởi dữ liệu thiếu, đồng thời giảm thiểu nhu cầu phải xử lý hoặc bổ sung dữ liệu.

3.3 Chia tập dữ liệu/loại bỏ giá trị ngoại lai/chuẩn hóa

Chia tập dữ liệu thành train và test

- **# Phân chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%)**
- **Train_X, Test_X, Train_Y, Test_Y = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_state=100)**

Phân chia dữ liệu thành tập huấn luyện và tập kiểm tra: được sử dụng để chia dữ liệu thành bốn phần:

- **Train_X** và **Train_Y** chứa 80% dữ liệu cho tập huấn luyện.
- **Test_X** và **Test_Y** chứa 20% dữ liệu cho tập kiểm tra.

Tham số:

1. **train_size=0.8:** Tỷ lệ dữ liệu cho tập huấn luyện.
2. **test_size=0.2:** Tỷ lệ dữ liệu cho tập kiểm tra.

3. **random_state=100**: Đảm bảo rằng phân chia là ngẫu nhiên nhưng có thể lặp lại

Loại bỏ các giá trị ngoại lai từ tập huấn luyện

```
• features1 = Train_X.columns
• outlier_indices = []
•
•
• for i in features1:
•     if Train_X[i].dtype in ['int64', 'float64']:
•         Q1 = Train_X[i].quantile(0.25)
•         Q3 = Train_X[i].quantile(0.75)
•         IQR = Q3 - Q1
•         outlier_condition = (Train_X[i] < (Q1 - 1.5 * IQR)) |
• (Train_X[i] > (Q3 + 1.5 * IQR))
•
• outlier_indices.extend(Train_X[outlier_condition].index.tolist())
•
•
• # Chỉ loại bỏ các chỉ số ngoại lai duy nhất
• outlier_indices = list(set(outlier_indices))
• Train_X = Train_X.drop(index=outlier_indices)
• Train_Y = Train_Y.drop(index=outlier_indices)
```

Xác định và loại bỏ các giá trị ngoại lai: Đoạn mã này được sử dụng để phát hiện và loại bỏ các giá trị ngoại lai (outliers) trong tập huấn luyện **Train_X**:

1. **Lấy danh sách các đặc điểm:** `features1 = Train_X.columns` lưu trữ tên của các cột (đặc điểm) trong **Train_X**.
2. **Khởi tạo danh sách chỉ số ngoại lai:** `outlier_indices = []` tạo một danh sách rỗng để lưu trữ các chỉ số của hàng có giá trị ngoại lai.
3. **Phát hiện giá trị ngoại lai:**
 - Vòng lặp `for i in features1`: lặp qua từng đặc điểm.
 - Nếu kiểu dữ liệu của đặc điểm là số (**int64** hoặc **float64**), mã tính toán các phần trăm Q1 (25%) và Q3 (75%), từ đó tính khoảng cách giữa chúng (IQR).
 - Điều kiện ngoại lai được xác định bằng công thức: một giá trị được coi là ngoại lai nếu nó nhỏ hơn $Q1 - 1.5 \times IQR$ hoặc lớn hơn $Q3 + 1.5 \times IQR$.

- Chỉ số của các hàng thỏa mãn điều kiện ngoại lai được thêm vào danh sách `outlier_indices`.

4. Loại bỏ giá trị ngoại lai duy nhất:

`outlier_indices = list(set(outlier_indices))` đảm bảo chỉ giữ lại các chỉ số duy nhất (tránh trùng lặp). Sau đó, các hàng có chỉ số ngoại lai được loại bỏ khỏi **Train_X** và **Train_Y** bằng các câu lệnh:

- `Train_X = Train_X.drop(index=outlier_indices)`
- `Train_Y = Train_Y.drop(index=outlier_indices)`

Chuẩn hóa dữ liệu

```
# Chuẩn hóa dữ liệu chỉ cho X
scaler_X = StandardScaler()
Train_X_std = scaler_X.fit_transform(Train_X)
Test_X_std = scaler_X.transform(Test_X)

# Không chuẩn hóa Y
Train_Y_std = Train_Y.values # Giữ nguyên giá trị
Test_Y_std = Test_Y.values

# Lưu scaler vào tệp .pkl
joblib.dump(scaler_X, '../savefile/scaler_X.pkl')
```

1. Khởi tạo bộ chuẩn hóa:

- `scaler_X = StandardScaler()` tạo m
- ột đối tượng **StandardScaler**, sẽ chuẩn hóa dữ liệu bằng cách biến đổi để có trung bình bằng 0 và độ lệch chuẩn bằng

2. Chuẩn hóa tập huấn luyện:

- `Train_X_std = scaler_X.fit_transform(Train_X)` sử dụng phương thức **fit_transform** để tính toán các tham số chuẩn hóa (trung bình và độ lệch chuẩn) từ **Train_X** và áp dụng chúng để chuẩn hóa dữ liệu.

3. Chuẩn hóa tập kiểm tra:

- `Test_X_std = scaler_X.transform(Test_X)` sử dụng phương thức transform để chuẩn hóa **Test_X** bằng các tham số đã tính toán từ **Train_X**, đảm bảo rằng tập kiểm tra không bị rò rỉ thông tin từ tập huấn luyện.

4. Giữ nguyên giá trị của Y:

- `Train_Y_std = Train_Y.values` và `Test_Y_std = Test_Y.values` giữ nguyên các giá trị trong **Train_Y** và **Test_Y** mà không chuẩn hóa, vì thường không cần thiết chuẩn hóa biến mục tiêu.

5. Lưu bộ chuẩn hóa:

- `joblib.dump(scaler_X, './savefile/scaler_X.pkl')` lưu đối tượng bộ chuẩn hóa vào tệp **.pkl** để sử dụng sau này, giúp tái sử dụng cùng một quy trình chuẩn hóa cho dữ liệu mới.

Chia tập huấn luyện thành 2 phần là train và validation

```
# Phân chia tập huấn luyện thành tập huấn luyện và tập validation
Train_X_std, Validation_X_std, Train_Y_std, Validation_Y_std =
train_test_split(Train_X_std, Train_Y_std, train_size=0.7,
test_size=0.3, random_state=100)
```

Đoạn mã trên được sử dụng để chia dữ liệu đã chuẩn hóa **Train_X_std** và **Train_Y_std** thành hai phần:

1. **Tập huấn luyện:** **Train_X_std** và **Train_Y_std** chứa 70% dữ liệu để huấn luyện mô hình.
2. **Tập validation:** **Validation_X_std** và **Validation_Y_std** chứa 30% dữ liệu để kiểm tra và điều chỉnh mô hình.

Tham số:

- **train_size=0.7:** Tỷ lệ dữ liệu cho tập huấn luyện.
- **test_size=0.3:** Tỷ lệ dữ liệu cho tập validation.

- **random_state=100:** Đảm bảo rằng phân chia dữ liệu là ngẫu nhiên nhưng có thể lặp lại.

Tập train_X

TV_Ad_Budget_(\$),Radio_Ad_Budget_(\$),Newspaper_Ad_Budget_(\$)
-1.412065949670761,1.3949718971988136,2.8362591352811
-0.7117120498920776,-1.5489786365906313,-1.0366628305616967
0.07715419109086291,1.2332904033522534,1.3440734900873041
1.5066285015396133,-0.6125733180626385,0.3058213421369286
-0.3842163874363267,-0.5856264024215452,0.049880114967766155
-1.1764018412144268,-1.2997196669105182,-0.8145251616978954
-0.0179965757037133,0.42488293411945416,-0.8724741187928
1.0574283699280091,-1.0571974261406782,-1.0608082293512404
-0.07331678895637392,1.5768635777761932,1.3682188888768478
1.4291802029858884,0.3979360184783607,1.4020224471822091
-1.4541093117427832,-1.522031720949538,-0.24469375026466592
-1.3821930345143243,-0.19489612562569206,-0.6599946094448162
-1.4087467368756015,1.5431799332248266,1.8656141039414462
1.641609821876105,0.35751564501672084,0.6052242871272696
0.3592872786794321,-0.9224628479352116,-1.171877063783141
-1.5913034406093816,1.118766011877607,-1.0608082293512404
-0.3742587490508477,1.1861333009803403,1.5710402387090143
0.6690804728943316,1.4892861019426402,-0.5344385357391893
-1.45300290747773,-0.47783873985717196,0.9142853916334278
-0.5988588148566498,-1.5287684498598113,-0.3605916644544754
0.3094990867520375,-0.32963070383115867,0.0015893173886789148
0.338265597643421,-0.033214631779132274,0.04022195545194874
-0.8334165190479309,-0.16794920998459886,-0.5972165725920028
-1.0646550104440524,1.6172839512378332,-1.0704663888670578

Tập train_Y

Sales_(\$)
8.7
8.8
18.5
16.1
12.4
7.6
15.0
13.2
19.2
20.8
5.3
7.6
9.3
21.4
12.8
1.6

16.0
22.6
5.6
8.7
14.4
14.9
11.3
11.6
10.6
13.4
20.1
11.8
23.2
9.9

Tập Validation X

TV_Ad_Budget_(\$),Radio_Ad_Budget_(\$),Newspaper_Ad_Budget_(\$)
1.2090057542402994,-0.11405537870241217,-0.03221424091668212
0.2962222355713991,1.1255027407878804,0.33962490044228977
-1.2826166506595353,-1.4479277029365314,0.1126581518205796
-1.1066983725160744,1.0985558251467868,0.6969768025275352
1.1536855409876388,0.2766748980934409,-0.3750789037282016
-0.4627710902551047,0.4450931208502742,-1.0752954686249665
1.3351358404563654,-1.3536134981927048,0.595566127611452
0.3449040232337402,-0.5115223844085386,-1.36504025409949
-1.3224472042014508,-0.8079384564605651,-0.046701480190408325
-0.7725642844700042,1.778965445084393,0.7259512810749876
-1.3213407999363977,0.1823606933496142,0.6100533668851781
1.3395614575165786,-0.2016328545359655,-1.4664509290155732
1.0541091571328496,-1.272772751269425,0.30099226237901977
0.8992125600253998,-0.9965668659482183,1.2474918949291298
-1.4010019070202289,0.9840314336721405,-0.4378569405810149
0.05834531858495842,-1.461401160757078,-0.30747178711747936
-0.035699043944564635,0.2564647113626209,0.7500966798645313
-1.3357240553820895,0.8156132109153073,1.7014253921725502
1.6172689280449346,-1.272772751269425,-1.0704663888670578
0.4798853435702323,-0.39026126402361877,-0.6165328916236378
1.3096885423601414,-1.3131931247310649,-0.5392676154970981
-1.1786146497445333,1.16592311424952,-0.9062776770981612
-1.5027910994051246,1.7452818005330264,2.140871650142244
1.6062048853944024,1.3006576924549866,0.9915506677599675
0.515290280051935,0.3844625606578141,-0.6020456523499116
-1.5038975036701778,-1.4075073294748912,-1.432647370710212
-0.8588638171441547,-0.7607813540886519,0.29616318262111097
0.7620184311588015,0.10825667533660772,-0.8483287200032564
-1.3844058430444306,-0.47110201094689863,-0.40405338227565385
0.2077098943671421,0.5798276990557407,1.0736450236444155

Tập Vadilation_Y

Sales_ (\$)

17.6
19.0
7.3
10.4
18.9
14.0
12.7
14.1
7.2
14.7
8.5
17.4
12.3
13.4
8.0
10.1
15.5
9.2
12.8
14.7
12.0
10.9

Tập Test_X

TV_Ad_Budget_ (\$) ,Radio_Ad_Budget_ (\$) ,Newspaper_Ad_Budget_ (\$)

-1.5127487377906033,1.0716089095056935,0.9625761892125151
1.036406688891998,0.7617193796331203,-1.224996941120137
-0.10318970411281085,1.2602373189933471,0.7356094405908049
0.8095938145560895,0.7078255483509339,1.3682188888768478
1.0751308381688605,1.0109783493132336,-0.3605916644544754
0.6889957496652895,-1.1986687332564183,-0.5440966952550069
-0.05893353351068235,-0.5856264024215452,-0.24469375026466592
0.5872065572803938,-1.3131931247310649,-1.1960224625726847
0.7952105591103978,1.2602373189933471,0.43137741584255546
0.7996361761706108,1.40844535501936,-0.16742847413812642
-0.30234247182238894,-0.558679486780452,-1.2201678613622282
0.9401495178323687,0.6269848014276538,2.1022390120789742
-0.9772490735048486,-1.1649850887050515,-0.046701480190408325
0.4787789393051792,-0.1275288365229588,-1.0221755912879706
-0.6342637513383527,-0.7540446251783784,-0.23020651099093986
-0.5568154527846279,-1.218878919987238,-1.0897827078986926
0.05723891431990496,0.8492968554666739,-1.1911933828147758
0.2840517886558135,-1.0235137815893116,0.21889790649457166
0.2099227028972484,0.9301376023899536,-1.1235862662040539
-0.34991785521967705,-0.3767878062030721,0.38308661826346824
0.776401686604493,0.06783630187496766,-1.2877749779729504
-0.16736115148589698,-1.1649850887050515,0.030563795936131326

Tập Test Y

Sales_ (\$)

6.6
20.7
17.2
19.4
21.8
12.2
12.2
11.7
22.6
22.3
11.9
19.7
8.7
15.6
10.6

Loại bỏ giá trị ngoại lai

```
Kích thước của các tập dữ liệu:  
Tập dữ liệu gốc ---> (200, 4)  
Tập huấn luyện ---> (110, 3) (110, 1)  
Tập xác thực ---> (48, 3) (48, 1)  
Tập kiểm tra ---> (40, 3) (40, 1)  
Các file dữ liệu đã được lưu thành công!
```

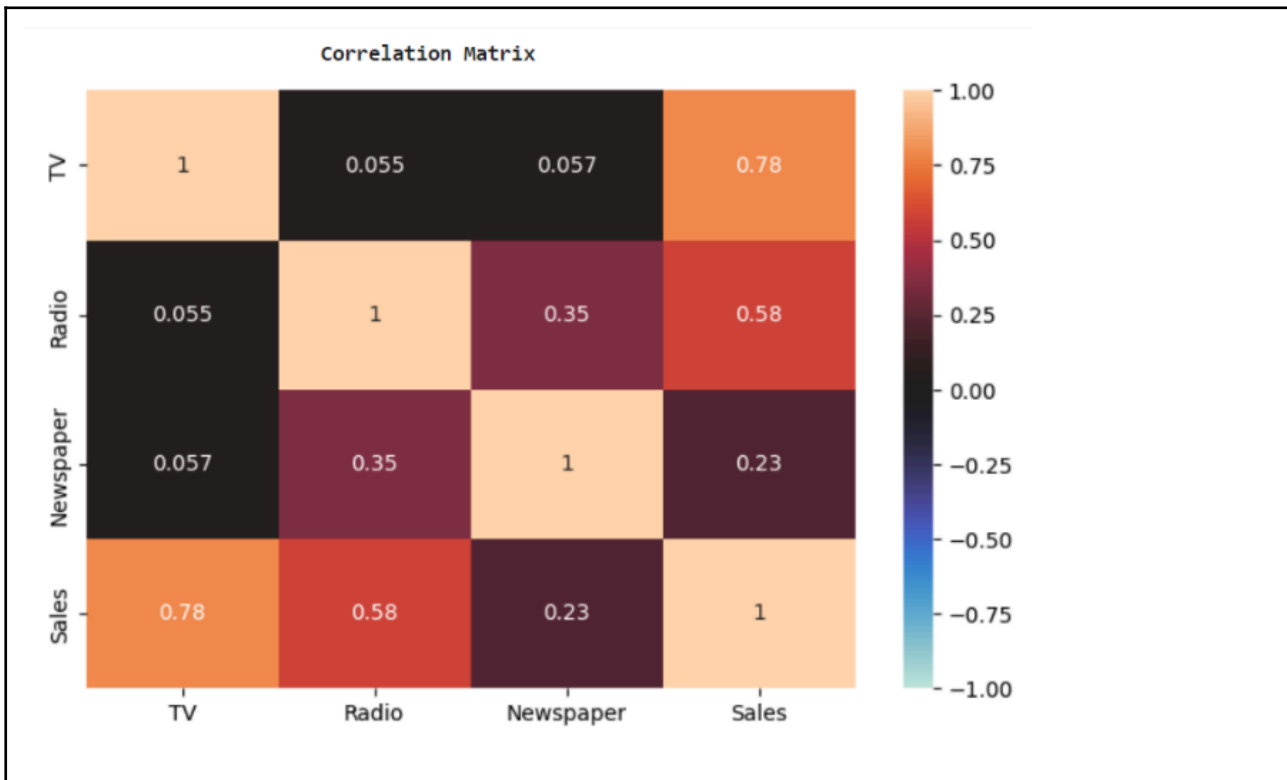
Nhận xét:

- Tổng dữ liệu trên 3 tập (Tập huấn luyện , tập xác thực , tập kiểm tra) là 198 => Chúng ta đã loại bỏ dữ liệu ngoại lai thành công
- Các dữ liệu trên tập X chúng ta đã chuẩn hóa thành

4. Lựa chọn và chiết suất tính năng

```
#Kiểm tra sự tương quan giữa các tính năng  
print('\033[1mMa trận tương quan'.center(70))
```

```
plt.figure(figsize=[8,5])  
sns.heatmap(df.corr(), annot=True, vmin=-1, vmax=1, center=0)  
#cmap='BuGn'  
plt.show()
```



Nhận xét

- TV_Ad_Budget và Sales: Có mối tương quan dương mạnh (0.78). Điều này cho thấy khi tăng ngân sách quảng cáo trên truyền hình, doanh số có xu hướng tăng đáng kể. Đây là một mối quan hệ tích cực và đáng chú ý.
- Radio_Ad_Budget và Sales: Cũng có mối tương quan dương (0.58), nhưng không mạnh bằng TV. Điều này cho thấy quảng cáo trên đài phát thanh cũng có tác động tích cực đến doanh số, nhưng mức độ ảnh hưởng không lớn bằng truyền hình.
- Newspaper_Ad_Budget và Sales: Mối tương quan khá yếu (0.23). Điều này cho thấy việc tăng ngân sách quảng cáo trên báo chí không có tác động đáng kể đến doanh số.
- Các biến quảng cáo với nhau: Các biến ngân sách quảng cáo trên truyền hình, đài phát thanh và báo chí có mối tương quan dương yếu hoặc rất yếu. Điều này có thể cho thấy các kênh quảng cáo này không cạnh tranh trực tiếp nhau mà bổ sung cho nhau.

=> Ma trận tương quan cho thấy có mối quan hệ tuyến tính đáng kể giữa chi tiêu quảng cáo trên TV, radio và doanh số nên ta **không loại bỏ đặc trưng nào**. Đây là điều kiện tiên quyết để áp dụng phân tích hồi quy tuyến tính.

III. Tìm hiểu thuật toán

1. Tổng quan

- Sau khi tiến hành xử lý dữ liệu xong, ta sẽ tiến hành xây dựng các mô hình học máy để đánh

giá hiệu suất của các mô hình

- Ở đây, nhóm sẽ tiến hành đánh giá 4 mô hình học máy đã lựa chọn 4 mô hình này:

(1) Linear Regression (Hồi quy tuyến tính)

(2) Ridge Regression (Hồi quy Ridge)

(3) Neural Network (Mạng Neural Nhân tạo)

(4) Stacking (Kết hợp 3 phương pháp trên)

- Để đánh giá hiệu năng, độ tin cậy (accuracy) của các mô hình, nhóm sẽ dùng 4 tham số để đánh giá, chi tiết như bảng dưới đây

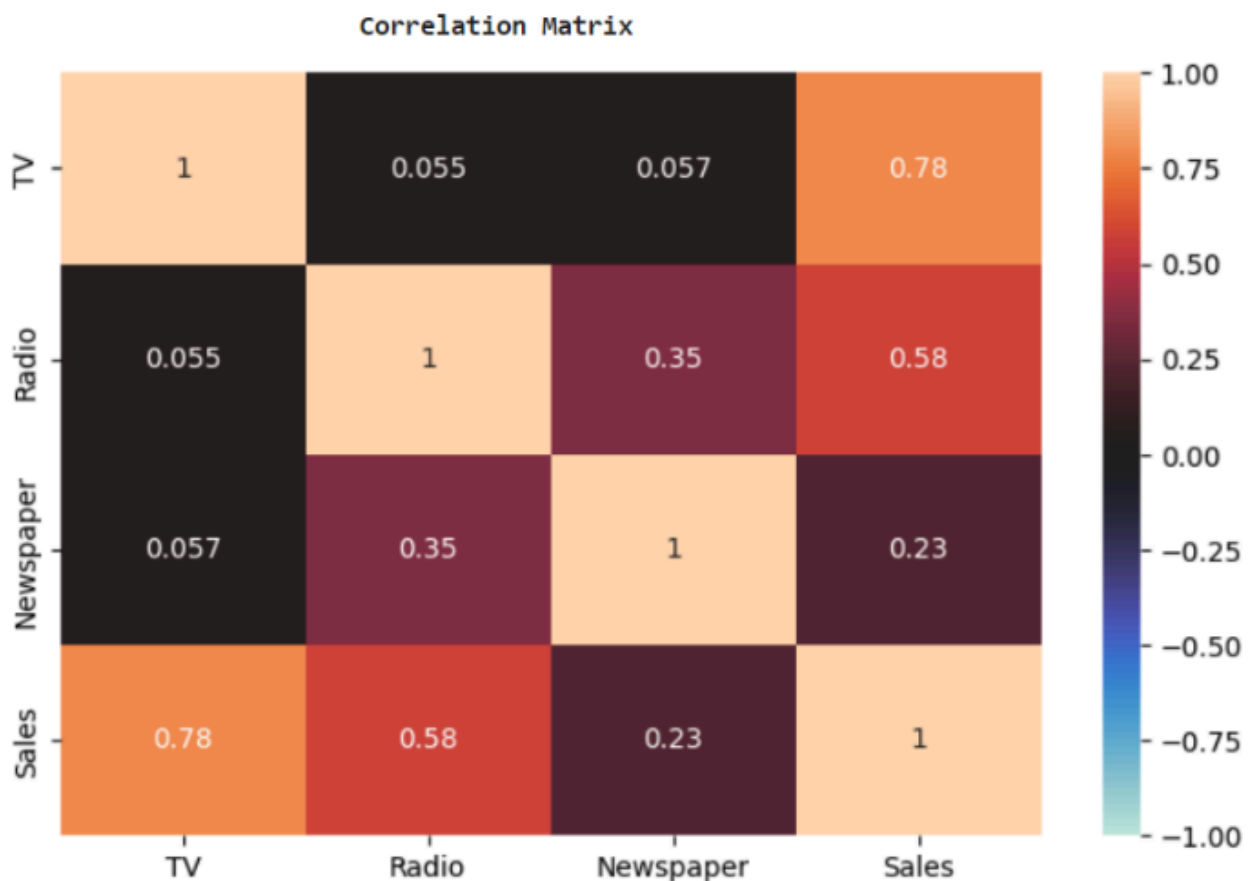
	RMSE	R2	MSE
Tên chỉ số	Căn trùng bình phương lỗi (Root Mean Square Error)	R-square(R bình phương)	Mean Squared Error
Mục đích	RMSE là căn bậc hai của MSE và mang cùng đơn vị với biến mục tiêu. Chỉ số này thường dễ hiểu hơn vì nó đưa sai số dự đoán về cùng thang đo với biến phụ thuộc.	Đo lường mức độ giải thích của mô hình sự biến thiên của dữ liệu	MSE là trung bình của các sai số bình phương giữa giá trị thực tế và giá trị dự đoán. Chỉ số này đo lường độ lớn của sai số dự đoán.
Giá trị	$[0; +\infty)$	$(-\infty; 1]$	$[0; +\infty)$

Ý nghĩa	RMSE càng nhỏ thì mô hình càng tốt	<p>R^2 nằm trong khoảng từ 0 đến 1.</p> <p>$R^2 = 1$: Mô hình dự đoán hoàn hảo (giải thích 100% phương sai).</p> <p>$R^2 = 0$: Mô hình không giải thích được gì, tức là không dự đoán tốt hơn một giá trị trung bình.</p> <p>$R^2 < 0$: Mô hình thậm chí còn tệ hơn việc chỉ sử dụng giá trị trung bình của biến mục tiêu để dự đoán.</p>	MSE luôn là số không âm và giá trị càng nhỏ thì mô hình càng tốt.
Công thức	$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ <p>y_i: giá trị thực tại điểm dữ liệu thứ i</p> <p>\hat{y}_i: giá trị dự đoán tại điểm thứ i</p> <p>n: số lượng các điểm dữ liệu</p>	$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$ <p>y_i: giá trị thực tại điểm dữ liệu thứ i</p> <p>\hat{y}_i: giá trị dự đoán tại điểm thứ i</p> <p>\bar{y}: giá trị trung bình của các giá trị thực</p> <p>n: là số lượng mẫu.</p>	$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ <p>y_i: giá trị thực tại điểm dữ liệu thứ i</p> <p>\hat{y}_i: giá trị dự đoán tại điểm thứ i</p> <p>n: số lượng các điểm dữ liệu</p>

2. Mô hình hồi quy tuyến tính

2.1 Lý do chọn mô hình

- Các biến độc lập có tương quan mạnh với biến mục tiêu.
- Mỗi quan hệ tuyến tính rõ ràng giữa biến độc lập và biến mục tiêu. Khi giá trị các biến độc lập tăng thì giá trị biến mục tiêu cũng tăng.
- Mô hình hồi quy tuyến tính trực quan, dễ diễn giải giúp hiểu rõ từng loại quảng cáo ảnh hưởng đến doanh thu.



2.2 Tổng quan về mô hình hồi quy tuyến tính

2.2.1 Định nghĩa

Hồi quy tuyến tính là một thuật toán supervised learning, ở đó quan hệ giữa đầu vào và đầu ra được mô tả bởi một hàm tuyến tính, là một phương pháp để dự đoán biến phụ thuộc (Y) dựa trên giá trị của biến độc lập (X).

2.2.2 Phân loại

- ❖ Hồi quy tuyến tính đơn biến (Simple Linear Regression): mô tả mối quan hệ giữa một biến độc lập và một biến phụ thuộc.

$$f(x) = w_0 + w_1 x_1$$

Trong đó :

- w_0 : còn được gọi là bias, là một hằng số, giúp mô hình có thể dịch chuyển đường hồi quy theo trục y, làm cho mô hình dự đoán chính xác hơn khi các biến đầu vào bằng 0
- w_1 : Trọng số của biến độc lập x_1
- ❖ Hồi quy tuyến tính đa biến (Multiple Linear Regression): mô tả mối quan hệ giữa nhiều biến độc lập và một biến phụ thuộc

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots + w_n x_n$$

2.2.3. Phân tích toán học

a. Dạng của Hồi quy tuyến tính

- Trong phương trình (1) như phía trên, nếu chúng ta đặt $w = [w_1 \ w_2 \ w_3 \ \dots \ w_m]$ là vector trọng số cần phải tối ưu và $x^T = [x_1, x_2, x_3, \dots, x_m]$ là vector đặc trưng. Khi đó, phương trình (1) có thể được viết lại dưới dạng:

$$f(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots + w_m x_m$$

$$\hat{y} = f(x) = x^T w$$

$$y \approx \hat{y} = x^T w$$

b. Sai số dự đoán

- ❖ Chúng ta mong muốn rằng sự sai khác e giữa giá trị thực y và giá trị dự đoán là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - x^T w)^2$$

- Hệ số $\frac{1}{2}$ là để thuận tiện cho việc tính toán (khi tính đạo hàm thì số $\frac{1}{2}$ sẽ bị triệt tiêu)
- Sự sai khác giữa y và \hat{y} không đồng nghĩa với e nhỏ nhất. Vì $e = y - \hat{y}$ có thể là một số âm, khi $e = -\infty$ là rất nhỏ nhưng sự sai lệch là rất lớn. Do đó chúng ta cần e^2

c. Hàm mất mát

- ❖ Điều tương tự xảy ra với tất cả các cặp *(input, outcome)* (x_i, y_i) , $i = 1, 2, \dots, N$, với N là số lượng dữ liệu quan sát được. Điều chúng ta muốn, tổng sai số là nhỏ nhất, tương đương với việc tìm w để hàm số sau đạt giá trị nhỏ nhất:

$$L(w) = \frac{1}{2N} \sum_{i=1}^N (y_i - x_i^T w)^2$$

- ❖ Hàm số $L(w)$ được gọi là **hàm mất mát** (loss function). Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số w sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của w làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là *điểm tối ưu* (optimal point), ký hiệu:

$$w^* = L(w)$$

2.2.4 Ưu và nhược điểm của mô hình

❖ Ưu điểm

- Dễ triển khai và diễn giải: Hồi quy tuyến tính là một mô hình đơn giản và dễ hiểu. Các hệ số của mô hình cho thấy mức độ và hướng tác động của các biến đầu vào đối với biến đầu ra, giúp việc diễn giải trở nên rõ ràng.
- Hiệu quả và tiết kiệm tài nguyên: Mô hình hồi quy tuyến tính có tốc độ huấn luyện nhanh và yêu cầu ít tài nguyên tính toán, nên rất phù hợp cho các tập dữ liệu nhỏ đến trung bình hoặc khi cần thử nghiệm nhanh các mô hình.
- Khả năng khái quát hóa tốt với dữ liệu tuyến tính: Nếu mối quan hệ giữa các biến là tuyến tính, mô hình hồi quy có thể khái quát hóa tốt và đưa ra dự đoán chính xác, ít xảy ra tình trạng overfitting.
- Hỗ trợ xác định mức độ tương quan: Mô hình hồi quy tuyến tính giúp xác định mức độ tương quan giữa các biến độc lập và biến phụ thuộc, nhờ đó có thể giúp phát hiện các yếu tố có ảnh hưởng quan trọng trong bộ dữ liệu.

❖ Nhược điểm

- Không phù hợp với quan hệ phi tuyến tính: Hồi quy tuyến tính không thể giải quyết tốt các quan hệ phi tuyến tính giữa biến độc lập và biến phụ thuộc. Trong các tình huống như vậy, các mô hình phi tuyến khác như hồi quy đa thức hoặc mạng nơ-ron sẽ hiệu quả hơn.
- Nhạy cảm với nhiễu và ngoại lệ: Hồi quy tuyến tính rất dễ bị ảnh hưởng bởi các giá trị nhiễu (noise) và giá trị ngoại lệ (outliers), làm giảm độ chính xác của mô hình.
- Khả năng biểu diễn hạn chế: Hồi quy tuyến tính chỉ có thể tạo ra một mặt phẳng tuyến tính trong không gian đa chiều, do đó khó nắm bắt được các quan hệ phức tạp trong dữ liệu.
- Không phù hợp với dữ liệu có tính phân loại: Hồi quy tuyến tính không lý tưởng cho các bài toán phân loại (classification). Đối với những bài toán này, hồi quy logistic hoặc các mô hình phân loại khác sẽ hiệu quả hơn.

2.3 Xây dựng mô hình

```
import time
# 1. Linear Regression
print("Linear Regression:")
start_time = time.time()
lin_reg = LinearRegression()
lin_reg.fit(Train_X_std, Train_Y)
end_time = time.time()
training_time = end_time - start_time
print(f"Thời gian huấn luyện mô hình: {training_time:.4f} giây")

evaluate_model(lin_reg, Train_X_std, Train_Y, Val_X_std, Val_Y,
               Test_X_std, Test_Y)
joblib.dump(lin_reg, r'../savefile/linear_regression_model.pkl')
```

- **import time:** Khai báo thư viện time của Python. Thư viện này cung cấp các hàm để làm việc với thời gian, bao gồm ghi lại thời điểm tính toán và khoảng thời gian.

- `start_time = time.time()` và `end_time = time.time()`: ghi lại thời điểm bắt đầu và kết thúc quá trình huấn luyện mô hình bằng cách gọi hàm `time.time()`.
- `lin_reg = LinearRegression()`: Tạo một đối tượng `LinearRegression` từ thư viện `scikit-learn`. Đối tượng này được sử dụng để huấn luyện và dự đoán.
- `lin_reg.fit(Train_X_std, Train_Y)`: Huấn luyện mô hình hồi quy tuyến tính (`lin_reg`) với dữ liệu huấn luyện. Trong đó:
 - **Train_X_std**: Là các đặc trưng (features) của dữ liệu đầu vào đã được chuẩn hóa.
 - **Train_Y**: Là biến mục tiêu (target variable) mà mô hình cần dự đoán.
- Phương thức `fit` sẽ tính toán các hệ số hồi quy (weights) và phương trình hồi quy dựa trên dữ liệu đầu vào.
- `evaluate_model(lin_reg, Train_X_std, Train_Y, Val_X_std, Val_Y, Test_X_std, Test_Y)`: Gọi hàm `evaluate_model` để đánh giá hiệu suất của mô hình hồi quy tuyến tính (`lin_reg`)
- `joblib.dump(lin_reg, r'../savefile/linear_regression_model.pkl')`: Lưu mô hình hồi quy tuyến tính đã huấn luyện (`lin_reg`) vào một tệp `.pkl`.

2.4 Đánh giá mô hình

Chỉ số đánh giá

Thời gian huấn luyện mô hình : 0.0150 giây

Dataset	R2 Score	MSE	RMSE
Training	0.8912	3.0963	1.7596
Validation	0.8809	3.2045	1.7901
Testing	0.9230	1.6368	1.2794

Thời gian huấn luyện: 0.0150 giây

Thời gian huấn luyện rất ngắn, chỉ 0.0150 giây, cho thấy mô hình hồi quy tuyến tính xử lý dữ liệu nhanh và hiệu quả. Thời gian này cho thấy mô hình không yêu cầu nhiều tài nguyên tính toán, dễ dàng mở rộng cho tập dữ liệu lớn hơn hoặc tái huấn luyện khi có dữ liệu mới.

R² Score

R² Score cho thấy mức độ mô hình giải thích được sự biến thiên của dữ liệu. Với giá trị R² cao trong cả ba tập dữ liệu:

- Training: 0.8912 cho thấy mô hình giải thích được 89.12% sự biến thiên trong tập huấn luyện.
- Validation: 0.8809 cho thấy mô hình giữ được tính khái quát hóa tốt, không có dấu hiệu overfitting hoặc underfitting.
- Testing: 0.9230 cho thấy mô hình duy trì độ chính xác cao trên tập dữ liệu mới, chứng minh tính khái quát hóa tốt của mô hình.

Mean Squared Error (MSE)

MSE là độ lệch trung bình bình phương của dự đoán so với giá trị thực. Trong trường hợp này:

- Training và Validation: Giá trị MSE trên tập huấn luyện (3.0963) và tập kiểm định (3.2045) là gần tương đương, cho thấy mô hình đã học vừa đủ mà không bị ảnh hưởng bởi nhiễu trong dữ liệu huấn luyện.
- Testing: Giá trị MSE thấp hơn đáng kể (1.6368) cho thấy mô hình hoạt động rất tốt trên dữ liệu mới, cho thấy khả năng tổng quát tốt khi áp dụng vào các tình huống thực tế.

Root Mean Squared Error (RMSE)

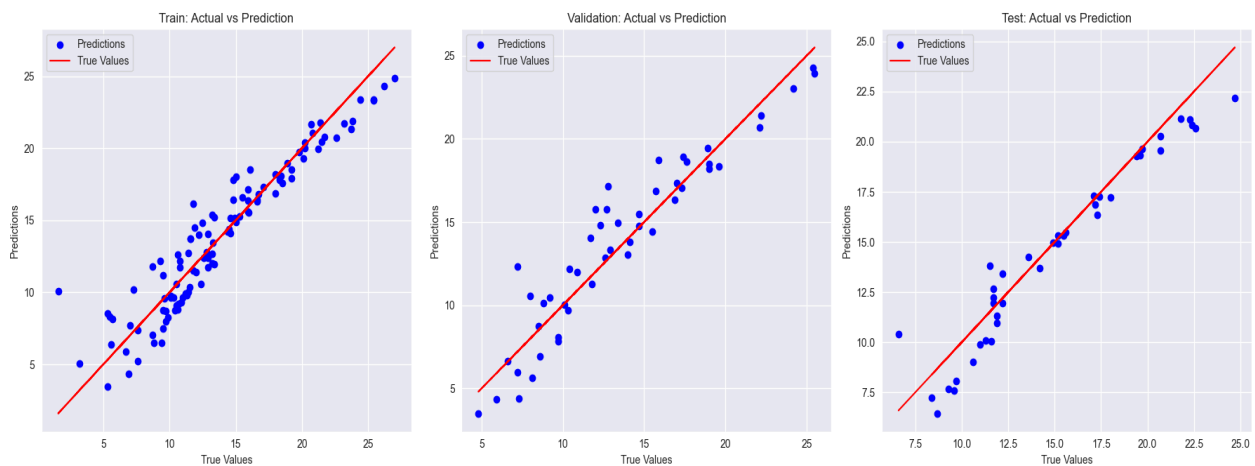
RMSE cho thấy mức sai lệch trung bình của các dự đoán.

- Training và Validation: Giá trị RMSE trong tập huấn luyện (1.7596) và kiểm định (1.7901) là gần như tương đương, xác nhận tính nhất quán của mô hình.
- Testing: RMSE trên tập kiểm tra (1.2794) nhỏ hơn cho thấy mô hình dự đoán chính xác hơn trên dữ liệu mới, củng cố thêm độ tin cậy và tính khái quát của mô hình.

Đánh giá chung:

Mô hình hồi quy tuyến tính này có độ chính xác cao với thời gian huấn luyện rất ngắn. Các chỉ số đánh giá cho thấy mô hình có khả năng khái quát hóa tốt, không bị overfitting hoặc underfitting và có thể áp dụng vào các dữ liệu mới một cách hiệu quả. Mô hình này phù hợp cho các ứng dụng yêu cầu tốc độ và độ chính xác ở mức độ cao, đặc biệt là các tình huống phải xử lý dữ liệu trong thời gian thực hoặc cần cập nhật mô hình thường xuyên.

Biểu đồ thể hiện mối quan hệ giữa giá trị thực và giá trị dự đoán



Nhận xét:

❖ Training Set:

- Dữ liệu dự đoán (các điểm màu xanh) khá gần với đường $y = x$ (đường màu đỏ), cho thấy mô hình có độ khớp tốt trên tập dữ liệu huấn luyện.
- Mô hình hoạt động tốt với tập huấn luyện khi các dự đoán gần với giá trị thực tế.

❖ Validation Set:

- Các điểm trên tập xác thực cũng khá gần với đường hồi quy, cho thấy mô hình có khả năng dự đoán khá tốt trên dữ liệu chưa được huấn luyện.
- Điều này là dấu hiệu tốt cho thấy mô hình có khả năng tổng quát hóa, không chỉ dự đoán tốt trên dữ liệu huấn luyện mà còn trên dữ liệu mới.

❖ Test Set:

- Tương tự như tập xác thực, các điểm trên tập kiểm tra cũng gần với đường chéo, cho thấy mô hình có hiệu suất tốt ngay cả với dữ liệu hoàn toàn mới.
- Điều này gợi ý rằng mô hình của bạn có độ chính xác cao và không có dấu hiệu bị lệch (bias) hay phương sai lớn (variance).

=> Đánh giá chung:

- Mô hình hồi quy tuyến tính này hoạt động tốt vì các giá trị dự đoán nằm gần đường hồi quy cho cả ba tập dữ liệu.
- Độ chính xác và khả năng tổng quát hóa: Mô hình có độ chính xác khá cao trên tất cả các tập dữ liệu (train, validation, test), cho thấy mô hình có thể áp dụng tốt cho dữ liệu chưa thấy trước và có khả năng tổng quát hóa tốt.
- Mô hình hồi quy tuyến tính này có thể dự đoán giá trị của biến mục tiêu với độ chính xác hợp lý dựa trên các đặc trưng đầu vào trong bộ dữ liệu quảng cáo này. Mô hình có thể hữu ích để dự đoán doanh số dựa trên các yếu tố quảng cáo, nếu đó là mục tiêu của dữ liệu.

3. Hồi quy Ridge

3.1 Lý do chọn mô hình

Tìm hiểu về **overfitting**, đa cộng tuyến và biểu đồ VIF

Overfitting là hiện tượng mô hình học máy học quá kỹ dữ liệu huấn luyện, dẫn đến hiệu suất cao trên dữ liệu này nhưng kém hiệu quả khi áp dụng cho dữ liệu mới.

Đa cộng tuyến xảy ra khi các biến độc lập trong mô hình có mối tương quan cao với nhau, làm giảm độ tin cậy và khả năng dự đoán của mô hình.

Variance Inflation Factor (VIF-hệ số phóng đại phương sai) là một chỉ số thống kê dùng để đánh giá mức độ nghiêm trọng của hiện tượng đa cộng tuyến trong phân tích hồi quy tuyến tính. VIF đo lường mức độ mà phương sai của hệ số hồi quy bị phóng đại lên do sự tương quan giữa các biến độc lập

- $VIF = 1$: Không có đa cộng tuyến.
- $1 < VIF < 5$: Đa cộng tuyến ở mức chấp nhận được.
- $VIF > 5$: Đa cộng tuyến cao, cần xem xét lại mô hình.

Biểu đồ VIF là một công cụ trực quan biểu thị giá trị VIF của từng biến độc lập, giúp dễ dàng nhận biết các biến có VIF cao, để có phương pháp điều chỉnh mô hình cho phù hợp.

Vẽ biểu đồ VIF

- Import các thư viện cần thiết

```
• import pandas as pd
• import matplotlib.pyplot as plt
• import seaborn as sns
• from statsmodels.stats.outliers_influence import
  variance_inflation_factor
```

- 1) **pandas**: thư viện cho việc thao tác và phân tích dữ liệu dạng bảng (DataFrame)
- 2) **matplotlib**: `matplotlib.pyplot` một module của thư viện matplotlib giúp tạo biểu đồ để trực quan hóa dữ liệu và tùy chỉnh giao diện đồ thị.
- 3) **seaborn**: là thư viện mở rộng của matplotlib để tạo các biểu đồ thống kê.
- 4) **Statsmodel**: `variance_inflation_factor` là một hàm của trong thư viện `statsmodels.stats.outliers_influence` dùng để đo lường mức độ đa cộng tuyến (multicollinearity) giữa các biến độc lập trong mô hình hồi quy.

Tính toán chỉ số đa cộng tuyến

```
• # Tính toán VIF cho dữ liệu huấn luyện
• vif_data = pd.DataFrame()
• vif_data['Feature'] = Train_X_std.columns
• # Tính toán chỉ số VIF cho từng biến
• vif_data['VIF'] = [variance_inflation_factor(Train_X_std.values, i)
  for i in range(Train_X_std.shape[1])]
```

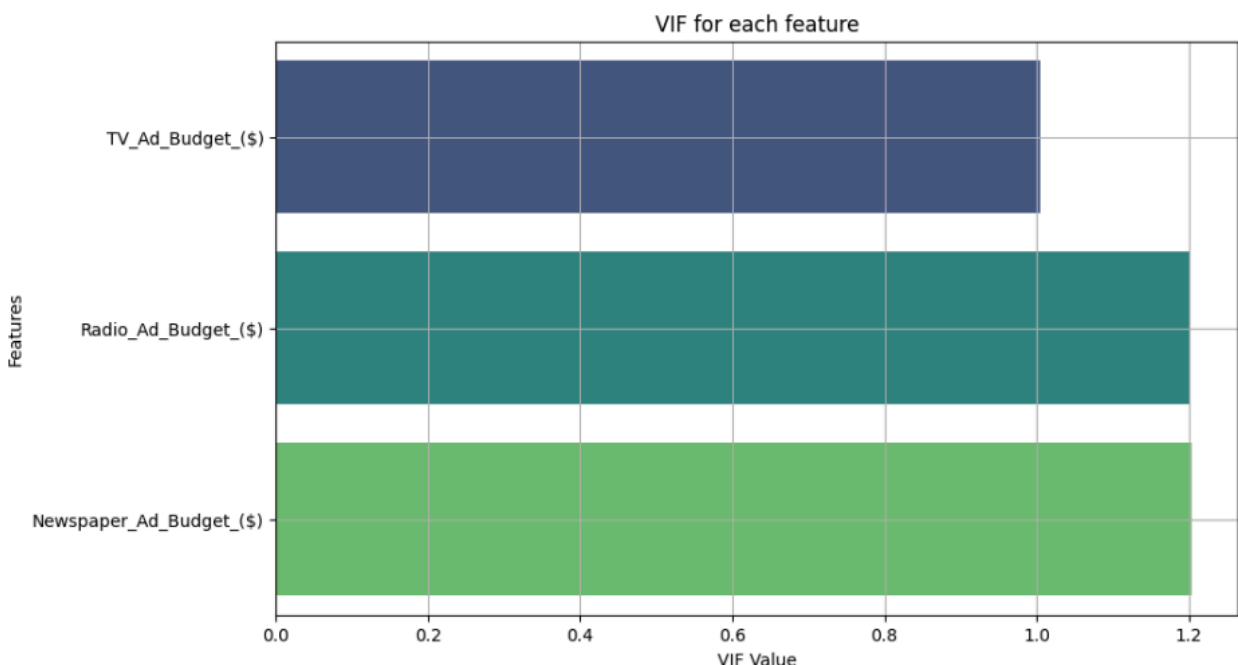
- Khởi tạo một DataFrame trống có tên là `vif_data`
- Sau đó, thêm một cột mới vào DataFrame `vif_data` có tên là “Feature”.
- Tính toán chỉ số VIF cho từng biến
- Kết quả: Ta thu được DataFrame `vif_data` trong đó “Feature” chứa tên các biến (các loại quảng cáo) và “VIF” chứa giá trị VIF tương ứng cho từng loại quảng cáo.

- Vẽ biểu đồ VIF

- # Vẽ biểu đồ thanh cho VIF
- # Tạo một đối tượng hình vẽ và quy định kích thước
- `plt.figure(figsize=(10, 6))`
- # Vẽ biểu đồ thanh
- `sns.barplot(x='VIF', y='Feature', data=vif_data, palette='viridis')`
- `plt.title('VIF for each feature')` #Định nghĩa tiêu đề
- `plt.xlabel('VIF Value')` #Đặt tên nhãn trục X
- `plt.ylabel('Features')` #Đặt tên nhãn trục Y
- `plt.grid(True)` #Thêm lưới cho biểu đồ
- `plt.show()` #Hiển thị biểu đồ

Vẽ biểu đồ thanh VIF với trục X là các phương thức quảng cáo và trục Y là giá trị VIF tương ứng.

Biểu đồ thanh VIF



Phân tích biểu đồ và lý do chọn mô hình

- Biểu đồ sau khi chạy mã

Phân tích

- Trục Y: chứa tên các phương pháp quảng cáo (TV_Ad_Budget (\$), Radio_Ad_Budget (\$), Newspaper_Ad_Budget (\$))
- Trục X: chứa chỉ số VIF tương ứng với từng loại quảng cáo

Nhận xét: chỉ số VIF của các biến đều nằm trong khoảng $1 < VIF \leq 1.2$ cho thấy giữa các biến có sự đa cộng tuyến nhưng không quá nghiêm trọng.

=> Lý do chọn mô hình:

- Giữa các kênh quảng cáo phần nào có mối tương quan với nhau, có thể dẫn đến đa cộng tuyến làm cho các hệ số hồi quy không ổn định và khó giải thích.
- Theo phân tích biểu đồ, giữa các biến có sự đa cộng tuyến dù không quá nghiêm trọng nhưng cũng sẽ có ảnh hưởng và mô hình ridge là lựa chọn tốt để giải quyết vấn đề này.
- Ngoài ra, mô hình Ridge hữu ích trong việc tối ưu hóa việc dự đoán doanh thu bằng cách giảm overfitting giúp mô hình có khả năng tổng quát hóa và dự đoán tốt hơn trên dữ liệu thực tế.

3.2 Tổng quan về mô hình Ridge

3.2.1 Định nghĩa

- Ridge Regression là một biến thể của hồi quy tuyến tính (Linear Regression), bổ sung thêm một thành phần điều chỉnh (regularization term) vào hàm mất mát.
- Phương pháp được phát triển nhằm giải quyết các vấn đề liên quan đến overfitting và đa cộng tuyến trong mô hình hồi quy.

3.2.2 Công thức

$$\underbrace{\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2}_{\text{RSS}} + \underbrace{\alpha \sum_{j=1}^p \beta_j^2}_{\text{L2 Regression}}$$

Giải thích:

- **RSS (Residual Sum of Squares):** là tổng bình phương sai số giữa giá trị thực tế y_i và giá trị dự đoán từ mô hình.
- **L2 Regression:** đây là phần phạt, phần này giúp giảm thiểu các trọng số của mô hình, giúp tránh overfitting, không cho mô hình quá phụ thuộc vào bất kỳ biến nào trong tập dữ liệu.
 - α là hệ số điều chỉnh, có tác dụng kiểm soát mức độ phạt.
 - Cần tìm được tham số alpha tốt nhất cho mô hình để mô hình có hiệu suất tốt nhất khi dự đoán.

3.2.3 Ưu và nhược điểm

❖ Ưu điểm

- **Tránh overfitting:** Hồi quy Ridge thêm thành phần regularization vào hàm mất mát làm giảm bớt độ phức tạp của mô hình. Điều này giúp mô hình không bị "quá khớp" (overfit) với dữ liệu huấn luyện, đặc biệt là khi có nhiều biến đầu vào hoặc khi dữ liệu huấn luyện có kích thước nhỏ.
- **Giảm đa cộng tuyến:** Trong các bài toán hồi quy tuyến tính có nhiều biến đầu vào tương quan mạnh với nhau (đa cộng tuyến), các hệ số hồi quy có thể trở nên không ổn định và dao động lớn. Hồi quy Ridge giúp giảm bớt tình trạng này bằng cách thêm hệ số phạt giúp các hệ số hồi quy trở nên ổn định và đáng tin cậy hơn, mang lại kết quả hồi quy phù hợp và không bị ảnh hưởng nhiều bởi đa cộng tuyến.
- **Ổn định hệ số hồi quy:** Hồi quy Ridge làm cho các hệ số hồi quy có xu hướng nhỏ hơn, làm giảm độ dao động của chúng. Điều này giúp mô hình không quá nhạy cảm với

những thay đổi nhỏ trong dữ liệu và trở nên ổn định hơn, ít biến động khi gặp các tập dữ liệu tương tự.

❖ Nhược điểm

Hệ số điều chỉnh cần phải được lựa chọn cẩn thận vì nó ảnh hưởng trực tiếp đến hiệu suất của mô hình:

- Nếu alpha quá lớn, tức là mô hình có thể bị điều chỉnh quá mức và dẫn đến underfitting (dưới khớp).
- Nếu alpha quá nhỏ không đủ để điều chỉnh mô hình dẫn đến việc mô hình có thể bị overfit dữ liệu huấn luyện giống như hồi quy tuyến tính thông thường.

3.3 Huấn luyện mô hình

Khởi tạo mô hình

- **# Khởi tạo mô hình Ridge Regression**
- `ridge = Ridge()`

Định nghĩa khoảng giá trị alpha

- **# Định nghĩa khoảng giá trị alpha cho mô hình**
- `alpha_range = np.logspace(0, 5, 100)`

- Sử dụng hàm `logspace(start, stop, num)` trong thư viện NumPy để tạo ra một mảng tên là ***alpha_range*** chứa 100 giá trị từ 10^0 đến 10^5 (từ 1 đến 100,000)
- Các giá trị này được phân bố đều theo logarit (nghĩa là khoảng cách giữa các giá trị sẽ tăng lên khi giá trị của alpha lớn hơn).
- Giải thích:
 1. **start:** giá trị đầu tiên của mũ (ở đây là 0 à 10^0)
 2. **stop:** giá trị cuối cùng của mũ (ở đây là 5 à 10^5)
 3. **num:** số lượng giá trị trong khoảng (start, stop) (ở đây là 100)

Tạo lưới tham số chứa các giá trị alpha vừa tạo

- **# Tạo lưới tham số cho RandomizedSearchCV**
- `param_grid = {'alpha': alpha_range}`

Tạo một từ điển tên `param_grid` có key là 'alpha' và value là `alpha_range` (danh sách các giá trị alpha vừa tạo ở bước trên)

Từ điển `param_grid` lưu trữ tập giá trị mà `RandomizedSearchCV` sẽ dùng để thử nghiệm các tham số của mô hình.

Khởi tạo RandomizedSearchCV

- **# Khởi tạo RandomizedSearchCV với các tham số đã định nghĩa**
- `random_search = RandomizedSearchCV(
• ridge,
• param_grid,
• n_iter=14,
• scoring='neg_mean_squared_error',
• cv=6,
• random_state=0
•)`

RandomizedSearchCV là một phương pháp dùng để tìm kiếm siêu tham số tốt nhất cho một mô hình bằng cách chọn ngẫu nhiên một số lượng cố định các kết hợp từ các phân phối tham số đã định trước.

Giải thích các tham số:

1. **ridge**: mô hình hồi quy ridge đã khai báo ở B1 (mô hình đang cần tối ưu)
2. **param_grid**: từ điển chứa tham số alpha muốn thử nghiệm
3. **n_iter = 14** : số lượng kết hợp tham số ngẫu nhiên sẽ được thử nghiệm (ở đây là 14 lần thử)
4. **scoring = 'neg_mean_squared_error'**: tiêu chí đánh giá hiệu suất của mô hình.
Dùng tiêu chí đánh giá là `neg_mean_squared_error` (nghịch đảo của MSE).

RandomizedSearchCV sẽ cố gắng tìm tổ hợp tham số có giá trị lỗi bình phương trung bình (MSE) thấp nhất.

5. **cv=6**: số lần phân tách dữ liệu để thực hiện cross-validation
6. **random_state = 0**: Kiểm soát tính ngẫu nhiên của quá trình tìm kiếm, giúp kết quả ổn định khi chạy lại.

Huấn luyện mô hình và tìm alpha tối ưu

- Huấn luyện mô hình bằng random_search

- # Huấn luyện mô hình với dữ liệu huấn luyện đã chuẩn hóa và tìm kiếm alpha tốt nhất
- `random_search.fit(Train_X_std, Train_Y)`

- Gọi đến phương thức fit để khởi động quá trình tìm kiếm ngẫu nhiên, thử nghiệm với 14 tổ hợp alpha và đánh giá hiệu suất qua cross-validation. Truyền vào 2 tham số là Train_X_std và Train_Y lần lượt là dữ liệu huấn luyện đã chuẩn hóa và nhãn tương ứng.

- Kết quả: chúng ta sẽ tìm được giá trị alpha tối ưu cho mô hình ridge.

- Huấn luyện mô hình ridge_best với alpha tốt nhất trên

- #Lấy mô hình và giá trị anpha tốt nhất
- `best_alpha = random_search.best_params_['alpha']`
- `ridge_best = Ridge(alpha=best_alpha)`
-
- # Huấn luyện lại mô hình với alpha tốt nhất
- `ridge_best.fit(Train_X_std, Train_Y)`

- Truy cập vào thuộc tính best_params_ của random_search để lấy ra giá trị alpha tối ưu qua quá trình tìm kiếm

- Tạo một mô hình Ridge mới (ridge_best) với giá trị alpha tốt nhất.

- Gọi đến phương thức fit để huấn luyện lại mô hình với giá trị alpha tối ưu vừa tìm được.
- **Đánh giá mô hình và in ra alpha tốt nhất**

```
• # Đánh giá mô hình trên các tập dữ liệu
• evaluate_model(ridge_best, Train_X_std, Train_Y, Val_X_std, Val_Y,
  Test_X_std, Test_Y)
• # In ra giá trị alpha tốt nhất đã tìm được
• print(f"Alpha tốt nhất: {best_alpha}")
```

- Gọi đến hàm evaluate_model để tiến hành đánh giá mô hình trên các tập dữ liệu: train, validation, test (với các chỉ số R2, MSE, RMSE), đồng thời in ra alpha tốt nhất đã tìm.

Phân tích tham số ảnh hưởng (alpha)

- **Khái niệm:** Alpha là tham số điều chỉnh trong hồi quy Ridge, đóng vai trò quan trọng trong việc điều chỉnh độ phức tạp của mô hình, giúp tránh hiện tượng overfitting. Nó được gọi là **tham số regularization** (tham số điều chỉnh)

-Lựa chọn tham số alpha

Tham số alpha trong hồi quy Ridge quyết định mức độ regularization.

- Nếu alpha = 0, thì không có regularization và mô hình sẽ có khả năng overfitting cao.
- Nếu alpha lớn, mô hình sẽ trở nên đơn giản hơn, nhưng có thể dẫn đến underfitting nếu giá trị quá lớn.

**** Giải thích lý do chọn khoảng từ 10^0 đến 10^5 (từ 1 đến 100,000)**

-Vẽ biểu đồ MSE theo alpha

+ Phân tích các bước thực hiện:

Ta thực hiện vẽ biểu đồ MSE theo alpha với một khoảng giá trị alpha đủ rộng để có cái nhìn bao quát về sự ảnh hưởng của tham số alpha đến hiệu suất của mô hình hồi quy.

Định nghĩa khoảng giá trị alpha

```
• alpha_range_test = np.logspace(-3, 8, 100)
```

- Đầu tiên ta định nghĩa một khoảng alpha rộng: lấy 100 giá trị alpha khác nhau trong khoảng từ 10^{-3} đến 10^8

+Dự đoán và tính toán MSE với alpha tối ưu

```
• val_predictions = ridge_best.predict(Val_X_std)  
• val_mse = mean_squared_error(Val_Y, val_predictions)
```

- Dùng mô hình đã huấn luyện với alpha tối ưu ridge_best để dự đoán trên bộ dữ liệu validation.

- Sau đó tính MSE bằng cách so sánh Val_Y (giá trị thực tế) với val_predictions (giá trị dự đoán). Lưu giá trị vào biến val_mse dùng để so sánh với MSE khi sử dụng các giá trị alpha khác trong dải alpha_range_test đã khởi tạo.

Tính MSE cho từng giá trị alpha

```
• mse_scores = []  
• for alpha in alpha_range_test:  
•     ridge = Ridge(alpha=alpha)  
•     ridge.fit(Train_X_std, Train_Y)  
•     val_predictions = ridge.predict(Val_X_std)  
•     mse_scores.append(mean_squared_error(Val_Y, val_predictions))
```

- Dùng vòng lặp for để lặp qua các giá trị alpha trong mảng đã khởi tạo và tính MSE cho từng giá trị alpha. Các giá trị này sẽ được lưu vào một mảng để vẽ biểu đồ trực quan.

1. Tạo một mô hình Ridge mới với giá trị alpha hiện tại (trong vòng lặp).
2. Huấn luyện mô hình đó với dữ liệu train (Train_X_std và Train_Y).
3. Dự đoán trên tập validation (Val_X_std).
4. Tính MSE giữa Val_Y và val_predictions, và thêm kết quả vào mảng mse_scores.

Vẽ biểu đồ

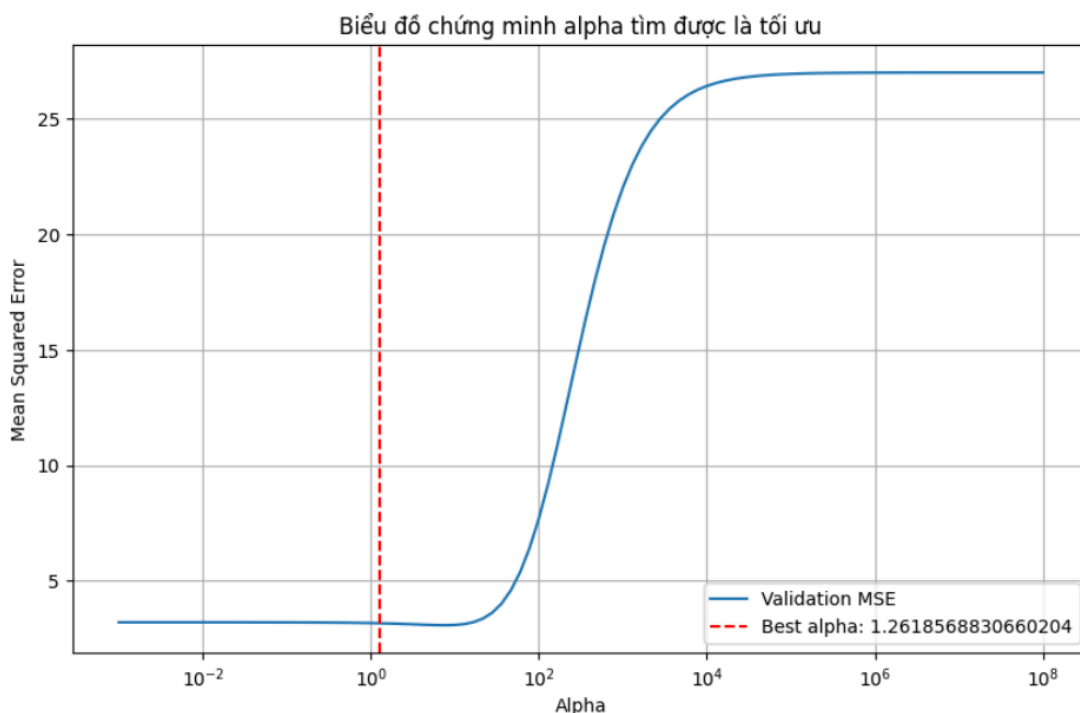
```
• plt.figure(figsize=(10, 6))
• plt.plot(alpha_range_test, mse_scores, label='Validation MSE')
• plt.axvline(best_alpha, color='red', linestyle='--', label=f'Best alpha: {best_alpha}')
• plt.xscale('log')
• plt.xlabel('Alpha')
• plt.ylabel('Mean Squared Error')
• plt.title('Biểu đồ chứng minh alpha tìm được là tối ưu')
• plt.legend()
• plt.grid(True)
• plt.show()
```

- Vẽ biểu đồ trực quan với trục X thể hiện các giá trị alpha và trục Y là giá trị MSE tương ứng.

- Sử dụng **plt.axvline** để vẽ thêm đường dọc màu đỏ biểu thị giá trị alpha tối ưu.

=> **Kết luận:** Biểu đồ sẽ cho thấy các điểm mà MSE đạt giá trị thấp và từ đó ta giúp chúng ta đánh giá xem giá trị `best_alpha` đã tìm được có thực sự là tối ưu hay không.

Biểu đồ chứng minh tìm alpha được tối ưu



Nhận xét:

Mối quan hệ giữa MSE và alpha:

- Nhìn biểu đồ ta có thể thấy, khi alpha nhỏ (từ 10^{-2} đến 10^0), MSE tương đối thấp và ổn định.
- Khi alpha bắt đầu tăng, MSE cũng bắt đầu tăng lên một cách nhanh chóng. Điều này cho thấy với một giá trị alpha quá cao khiến hiệu suất của mô hình bị giảm đáng kể.

Giá trị alpha tối ưu

Đường màu đỏ đánh dấu giá trị alpha đã tìm được từ quá trình huấn luyện.

Alpha tìm được là **1.2618568830660204** nằm trong khoảng từ 10^{-2} đến 10^0 tức là khoảng có MSE thấp. Từ đó cho thấy giá trị alpha tìm được là tối ưu và phù hợp với mô hình. Và điều này cũng chứng minh khoảng giá trị alpha đã định nghĩa trong quá trình huấn luyện mô hình (từ 10^0 đến 10^5) là hợp lý.

3.4 Đánh giá mô hình

Chỉ số đánh giá

Dataset	R2 Score	MSE	RMSE
Training	0.8911	3.0999	1.7607
Validation	0.8824	3.1639	1.7787
Testing	0.9221	1.6550	1.2865

Chỉ số alpha tốt nhất : 1.2618568830660204

Nhận xét:

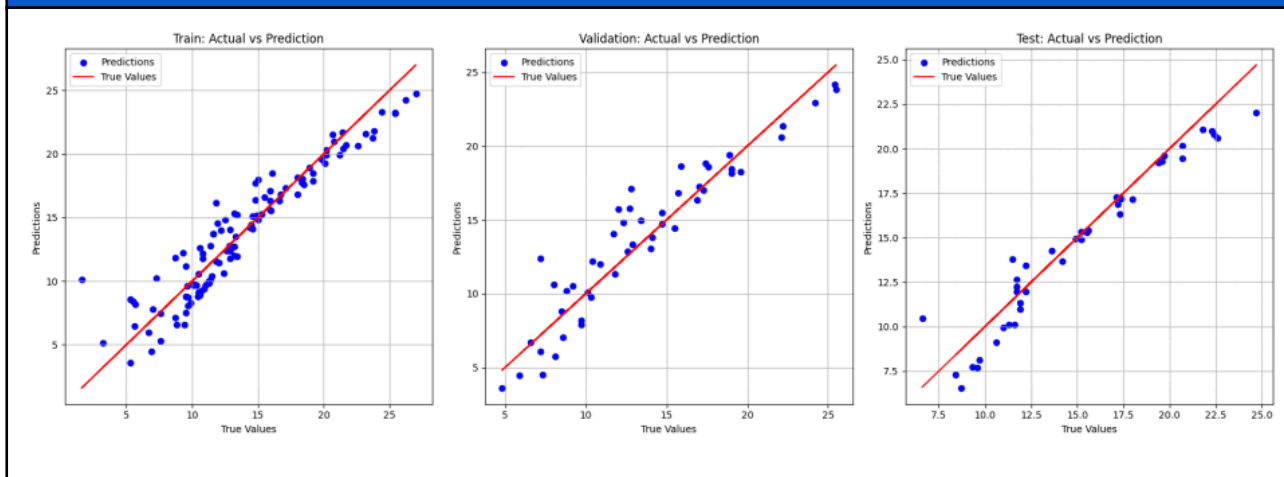
1. **Chỉ số R2:** Trên cả ba tập dữ liệu (training, validation, testing), các giá trị R2 đều ở mức cao (>0.88), điều này cho thấy mô hình phù hợp với dữ liệu. Đặc biệt, giá trị R² của tập testing là cao nhất (0.9221), cho thấy mô hình có khả năng tổng quát hóa tốt.
2. **Chỉ số MSE:** MSE trên tập testing thấp hơn nhiều so với tập training và validation (MSE testing = 1.6550), cho thấy mô hình có thể dự đoán tốt trên dữ liệu

mới (testing). Và sự chênh lệch MSE giữa các tập training, validation và testing không quá lớn, thể hiện rằng mô hình không bị overfitting và có thể áp dụng cho dữ liệu hoàn toàn mới.

3. **Chỉ số RMSE:** Sự khác biệt RMSE giữa các tập là nhỏ, cho thấy tính ổn định. Hơn nữa RMSE của tập testing (1.2865) thấp hơn so với tập training và validation, điều này cho thấy dự đoán của mô hình trên tập dữ liệu ngoài mẫu khá chính xác.
4. **Alpha tối ưu:** Giá trị alpha tìm được là **1.2618568830660204**. Điều này cho thấy mô hình đã được điều chỉnh một mức độ nhất định. Các chỉ số trên 3 tập đều cho kết quả khả quan có thể thấy alpha đã có tác động hiệu quả và tích cực đến mô hình.

=> **Nhìn chung**, mô hình *Ridge Regression* với *alpha tối ưu* đã mang lại một mô hình dự báo chính xác, ổn định và có khả năng tổng quát tốt trên dữ liệu mới.

Biểu đồ thể hiện mối quan hệ giữa giá trị thực và giá trị dự đoán



Nhận xét:

1. **Trên tập Train:** Các điểm dữ liệu phân bố khá đều xung quanh đường $y = x$ (đường màu đỏ) trên tập training. Điều này cho thấy mô hình có thể dự đoán với độ chính xác khá cao khi làm việc với dữ liệu nó đã được huấn luyện.
➤ *Kết luận:* mô hình đã học được quy luật từ dữ liệu training và không bị quá khớp.
2. **Trên tập Validation:** Mặc dù có một số điểm nằm xa đường $y = x$, tuy nhiên nhìn chung các điểm dữ liệu trên tập validation cũng khá gần với đường này. Với kết quả này, mô hình vẫn cho thấy được việc dự đoán tương đối chính xác trên tập validation và có khả năng tổng quát hóa tốt với dữ liệu chưa từng thấy.

- *Kết luận: Độ chính xác giảm hơn so với tập training nhưng không đáng lo ngại. Mô hình có tính ổn định và hiệu suất dự đoán tốt.*
- 3. **Trên tập Test:** Các điểm dự đoán vẫn bám khá gần đường $y = x$, đặc biệt là đối với các giá trị nhỏ và trung bình. Với các điểm lớn hơn thì có xu hướng ra xa đường màu đỏ
- *Kết luận: Mặc dù có một số điểm dự đoán sai lệch, nhưng độ chính xác dự đoán trên tập testing vẫn ở mức chấp nhận được. Mô hình vẫn duy trì được độ chính xác nhất định và thể hiện được khả năng tổng quát hóa*

Nhìn chung, các điểm dữ liệu đều bám sát khá sát đường $y = x$, cho thấy rằng mô hình dự đoán khá chính xác trên cả ba tập dữ liệu. Các biểu đồ cho thấy mô hình Ridge Regression đã học được quy luật của dữ liệu và dự đoán hiệu quả trên cả dữ liệu đã huấn luyện và dữ liệu mới.

4. Neural Network

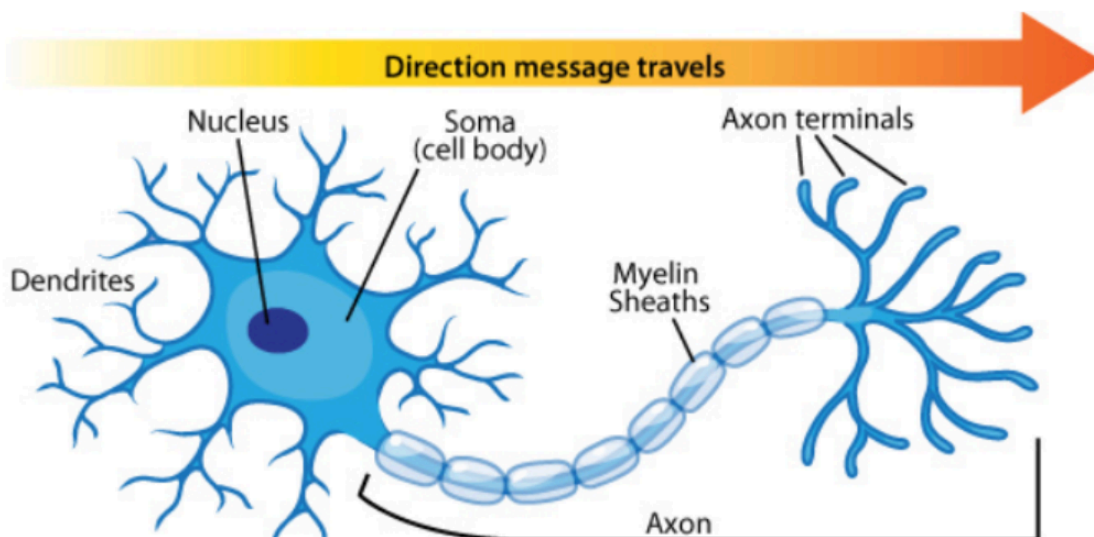
4.1 Lý do chọn mô hình

1. Mối quan hệ phi tuyến tính: Mặc dù ma trận tương quan cho thấy một số mối quan hệ tuyến tính giữa các biến (ví dụ: TV và Sales có hệ số tương quan 0.78), nhưng có thể còn tồn tại các mối quan hệ phi tuyến tính mà mô hình hồi quy tuyến tính không thể nắm bắt được.
2. Đa biến: Ma trận tương quan cho thấy nhiều biến đầu vào (TV, Radio, Newspaper) có ảnh hưởng đến biến đầu ra (Sales). Neural Network có thể xử lý tốt các bài toán có nhiều biến đầu vào và tìm ra các mẫu phức tạp trong dữ liệu.
3. Neural Network có thể học từ dữ liệu huấn luyện và tổng quát hóa tốt hơn cho các dữ liệu chưa từng thấy, giúp cải thiện độ chính xác của dự đoán.

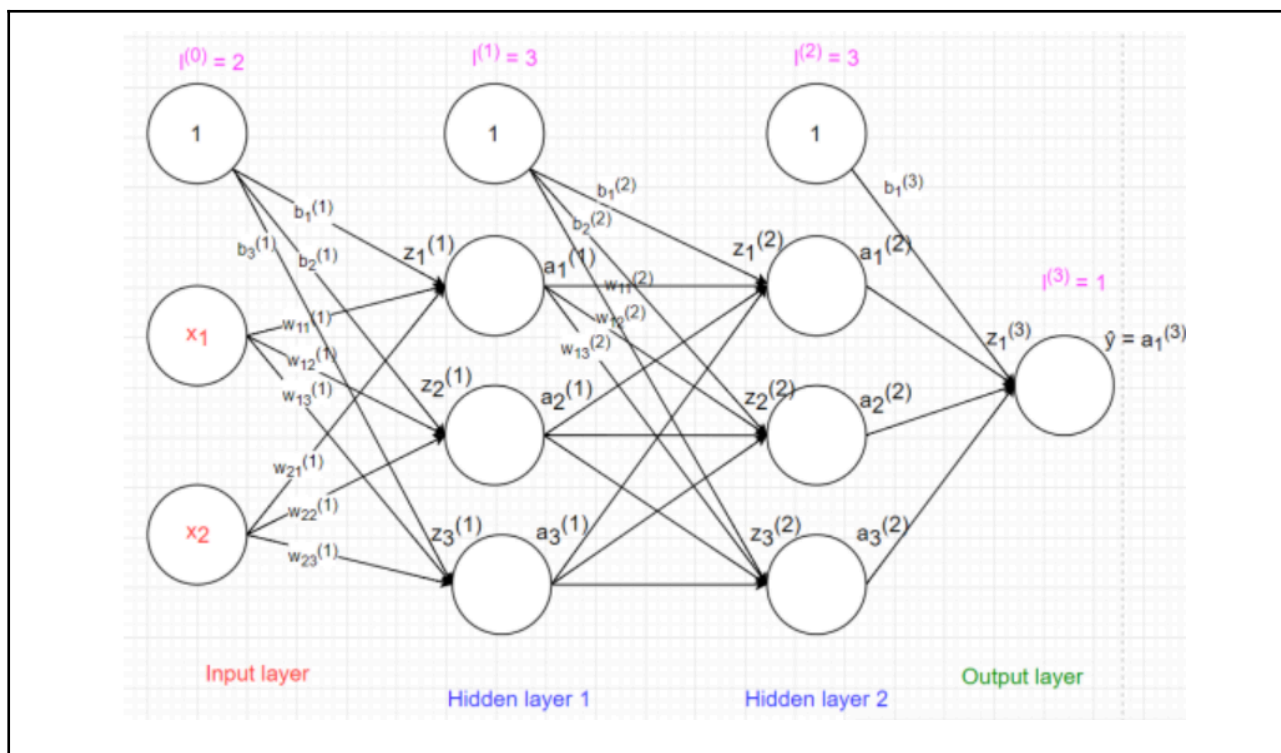
4.2 Tổng quan về mô hình Neural network

4.2.1 Định nghĩa

Neural Network là một mô hình học máy lấy cảm hứng từ cấu trúc và chức năng của não người. Nó bao gồm các lớp (layers) chứa các đơn vị tính toán gọi là nơ-ron (neurons). Mỗi nơ-ron nhận đầu vào, thực hiện một phép tính và truyền kết quả đến các nơ-ron ở lớp tiếp theo.



4.2.2 Cấu trúc mạng neural network



Lớp đầu vào (Input Layer):

- Số lượng neuron trong lớp đầu vào sẽ bằng với số lượng đặc trưng (features) của bộ dữ liệu. Mỗi neuron trong lớp đầu vào sẽ tương ứng với một đặc trưng, nhằm giúp mạng có thể nắm bắt được đầy đủ thông tin từ dữ liệu.
- Trước khi đưa vào lớp đầu vào, dữ liệu thường được chuẩn hóa hoặc tiêu chuẩn hóa. Điều này giúp giảm thiểu sự ảnh hưởng của các đặc trưng có giá trị lớn hơn lên mô hình và giúp mạng hội tụ nhanh hơn. Ví dụ, ta có thể chuẩn hóa dữ liệu bằng cách

đưa mỗi đặc trưng về khoảng $[0, 1]$ hoặc chuẩn hóa để có trung bình bằng 0 và độ lệch chuẩn bằng 1.

Các lớp ẩn (Hidden Layers):

- Dữ liệu từ lớp đầu vào sẽ được truyền qua các lớp ẩn. Số lượng lớp ẩn và số lượng nơron trong mỗi lớp là các tham số có thể điều chỉnh để tối ưu hóa hiệu suất của mô hình.
- Mỗi nơron trong lớp ẩn thực hiện phép tính tổng của đầu vào từ các nơron của lớp trước, nhân với trọng số tương ứng. Quá trình này có thể được mô tả bằng công thức:

$$Z = \sum(W_i.X_i) + b$$

Trong đó: W_i là trọng số liên kết, X_i là giá trị đầu vào từ lớp trước, b là bias.

- Sau khi thực hiện phép biến đổi tuyến tính, một hàm kích hoạt phi tuyến (như ReLU, sigmoid, tanh) sẽ được áp dụng cho kết quả để tạo ra đầu ra của nơron. Hàm kích hoạt giúp mạng học được các mối quan hệ phi tuyến trong dữ liệu, đồng thời tăng khả năng biểu diễn của mô hình.

Lớp đầu ra (Output Layer):

- Trong bài toán hồi quy, lớp đầu ra thường chỉ có **một nơron duy nhất**, đại diện cho giá trị dự đoán của mô hình.
- Giá trị đầu ra của nơron này chính là giá trị dự đoán cuối cùng mà mô hình đưa ra. Đối với các bài toán hồi quy, không cần áp dụng hàm kích hoạt trong lớp đầu ra để đảm bảo rằng giá trị đầu ra là một số thực, có thể nhận các giá trị liên tục trên toàn miền số thực.

4.2.3 Cơ chế hoạt động

Bước 1: Xây Dựng Cấu Trúc Mạng

Mạng MLP bao gồm các lớp nơron được kết nối theo tầng, với mục tiêu dự đoán hoặc phân loại đầu ra dựa trên dữ liệu đầu vào.

Chọn số lượng lớp ẩn (Hidden Layers) và nơron (Neurons):

- Việc lựa chọn số lớp ẩn và số nơ-ron trong mỗi lớp phụ thuộc vào độ phức tạp của dữ liệu và bài toán cụ thể.
- Mạng có quá ít lớp hoặc nơ-ron có thể không học được đủ đặc trưng, trong khi quá nhiều sẽ gây hiện tượng overfitting (quá khớp).
- **Quy tắc chung:** Bắt đầu với 1-3 lớp ẩn, với số lượng nơ-ron từ 64 đến 512 và điều chỉnh dựa trên kết quả huấn luyện.

Bước 2: Khởi Tạo Trọng Số

Khi khởi tạo MLP, các trọng số (weights) và độ lệch (bias) của nơ-ron được khởi tạo ngẫu nhiên để bắt đầu quá trình huấn luyện.

Phương pháp khởi tạo trọng số:

- **He Initialization:** Sử dụng cho các lớp ẩn với hàm kích hoạt ReLU. Phương pháp này giúp tránh hiện tượng gradient bị mất (vanishing gradient) trong các mô hình sâu.
- **Xavier Initialization:** Phù hợp với hàm kích hoạt sigmoid hoặc tanh, giúp duy trì độ ổn định của gradient khi lan truyền qua mạng.

Bước 3: Tiến Hành Tính Toán

Truyền Tiến (Forward Propagation)

Trong quá trình huấn luyện, mỗi mẫu dữ liệu sẽ đi qua mạng từ đầu vào đến đầu ra để tính toán dự đoán.

Bước truyền tiến (Forward Pass):

- **Nhận đầu vào từ lớp đầu vào:** Dữ liệu đầu vào được truyền vào nơ-ron của lớp đầu tiên.
- **Tính toán đầu ra cho mỗi lớp ẩn:** Tại mỗi nơ-ron, công thức tính toán là

$$Z = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b$$

Áp dụng hàm kích hoạt: Giá trị tính toán từ nơ-ron được đưa vào hàm kích hoạt (activation function) để tạo đầu ra phi tuyến. Một số hàm kích hoạt thông dụng:

1. **ReLU:** $\text{Max}(0, x)$, nhanh và tránh gradient mất dần ở mạng sâu.
2. **Sigmoid:** Cho các bài toán phân loại với đầu ra từ 0 đến 1.

3. **Tanh**: Giúp cân bằng các giá trị đầu ra ở phạm vi $[-1, 1]$.

Tính Toán Lỗi

Sau khi tính toán dự đoán, mô hình sẽ đánh giá độ chính xác của dự đoán so với giá trị thực tế dựa trên hàm mất mát (loss function).

- **Mean Squared Error (MSE)**: Là hàm mất mát phổ biến cho các bài toán hồi quy, tính toán bình phương sai lệch giữa dự đoán và giá trị thực tế. Công thức:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2$$

Trong đó : y_i : Giá trị thực tế, \hat{y} : Giá trị dự đoán của mô hình.

Mục tiêu: Tối thiểu hóa hàm mất mát để cải thiện độ chính xác của mô hình.

Bước 4: Cập Nhật Trọng Số và Bias

Lan Truyền Ngược (Backpropagation)

Lan truyền ngược được sử dụng để cập nhật trọng số và bias thông qua các gradient của hàm mất mát đối với mỗi tham số.

- **Gradient Descent**: Điều chỉnh trọng số theo gradient của hàm mất mát để giảm sai số.
- **Tính toán gradient**: Sử dụng đạo hàm để xác định cách hàm mất mát thay đổi theo từng tham số của mô hình.
- **Thuật toán tối ưu hóa**: Gradient Descent có nhiều biến thể cải tiến:
 1. **SGD (Stochastic Gradient Descent)**: Cập nhật dựa trên một mẫu ngẫu nhiên, nhanh hơn nhưng dao động lớn.
 2. **Adam**: Tối ưu hóa tốc độ và độ ổn định với tốc độ học thích ứng, thường được sử dụng phổ biến.
 3. **RMSprop**: Thích hợp cho mạng sâu, giúp điều chỉnh tốc độ học cho mỗi tham số.

Cập Nhật Trọng Số và Bias

Các trọng số và bias sẽ được điều chỉnh trong hướng tối ưu để giảm thiểu hàm mất mát sau mỗi lần lan truyền ngược.

Cập nhật trọng số:

$$w = w - \alpha \frac{\partial L}{\partial w}$$

Cập nhật bias:

$$b = b - \alpha \frac{\partial L}{\partial b}$$

Trong đó:

- L là hàm mất mát (loss function).
- $\frac{\partial L}{\partial w}$ là đạo hàm của hàm mất mát theo trọng số w .
- $\frac{\partial L}{\partial b}$ là đạo hàm của hàm mất mát theo bias b .
- α là tốc độ học (learning rate),

Bước 5: Lặp Lại Quá Trình

Các bước 3 và 4 sẽ được lặp lại cho đến khi mô hình đạt được độ chính xác mong muốn hoặc đến khi số lần lặp (epoch) đạt giới hạn. Có thể sử dụng kỹ thuật early stopping để tránh overfitting.

4.2.3 Ưu và nhược điểm

Ưu điểm

- ❖ Khả năng học phi tuyến tính

- Neural network có thể học và nắm bắt được các mối quan hệ phi tuyến giữa các đặc trưng trong dữ liệu. Đặc điểm này làm cho neural network trở nên phù hợp với những bài toán phức tạp mà các mô hình tuyến tính thông thường khó mô hình hóa.
- **Ví dụ:** Trong nhận diện hình ảnh, các đặc trưng như biên, góc, và họa tiết không phải là các hàm tuyến tính, do đó neural network rất hiệu quả trong việc phát hiện các đặc trưng này.
- ❖ **Khả năng tổng quát tốt**
- Neural network có khả năng tổng quát hóa rất tốt với dữ liệu mới khi được huấn luyện đầy đủ và tránh overfitting. Mạng học được đặc trưng phức tạp từ dữ liệu, giúp mô hình đưa ra các dự đoán có độ chính xác cao cho cả dữ liệu chưa thấy.
- **Ví dụ:** Mạng nơ-ron có thể dự đoán tốt với các dữ liệu hình ảnh chứa nhiều biến đổi về màu sắc, độ sáng, và góc nhìn nhờ khả năng trích xuất đặc trưng mạnh mẽ.
- ❖ **Xử lý dữ liệu lớn và nhiều chiều**
- Neural network có thể mở rộng quy mô để xử lý một lượng lớn dữ liệu. Khi dữ liệu huấn luyện đủ phong phú, neural network sẽ học và trích xuất đặc trưng một cách hiệu quả, dù dữ liệu đó có kích thước lớn và đa chiều.
- **Ví dụ:** Trong các ứng dụng như xử lý ngôn ngữ tự nhiên và nhận diện giọng nói, neural network xử lý hàng triệu từ và âm thanh khác nhau để phân tích ngữ nghĩa.

Nhược điểm

- ❖ **Yêu cầu tài nguyên tính toán lớn**
- Neural network, đặc biệt là các mạng sâu, đòi hỏi khả năng tính toán cao do số lượng lớn trọng số và phép tính toán ma trận cần thiết. Đặc biệt khi huấn luyện các mạng sâu hoặc với dữ liệu lớn, thời gian và tài nguyên cần thiết sẽ rất lớn.
- **Biện pháp khắc phục:**
 - **Sử dụng GPU/TPU:** Đẩy nhanh tốc độ tính toán bằng phần cứng chuyên dụng như GPU hoặc TPU.
 - **Mạng nhẹ hơn:** Áp dụng kiến trúc nhỏ gọn hơn hoặc các kỹ thuật giảm độ phức tạp như pruning (cắt tỉa mạng) và distillation (mô hình hoá tri thức).
- ❖ **Khó giải thích (Lack of Interpretability)**

- Neural network là một "hộp đen" khó hiểu, với hàng nghìn đến hàng triệu tham số khiến việc giải thích các kết quả trở nên khó khăn. Việc hiểu rõ cách một neural network đưa ra dự đoán có thể phức tạp.
- **Biện pháp khắc phục:**
 - **XAI (Explainable AI):** Sử dụng các phương pháp giải thích như SHAP (Shapley Additive Explanations) và LIME (Local Interpretable Model-Agnostic Explanations) để hiểu cách thức và lý do mô hình dự đoán.
 - **Mô hình đơn giản hóa:** Nếu có thể, sử dụng mô hình nông (shallow model) hoặc mô hình với ít lớp và nơ-ron hơn để dễ hiểu hơn.
- ❖ **Cần lượng dữ liệu lớn để huấn luyện hiệu quả**
 - Neural network cần nhiều dữ liệu để học đặc trưng một cách chính xác và giảm thiểu overfitting. Với dữ liệu nhỏ, mạng nơ-ron dễ dẫn đến hiện tượng quá khớp.
 - **Biện pháp khắc phục:**
 - **Data Augmentation:** Tăng kích thước dữ liệu bằng cách nhân bản dữ liệu hiện có với các biến đổi nhỏ (như xoay, thay đổi độ sáng cho ảnh) để tạo dữ liệu mới.
 - **Transfer Learning:** Sử dụng các mạng đã được huấn luyện trên các tập dữ liệu lớn khác để áp dụng cho bài toán mới. Điều này đặc biệt hiệu quả với các bài toán về thị giác máy tính và ngôn ngữ tự nhiên.
- ❖ **Tuning tham số khó và phức tạp**
 - Việc tìm kiếm các tham số tối ưu cho neural network, như số lớp ẩn, số nơ-ron, tốc độ học, và hàm kích hoạt, có thể tốn nhiều thời gian và cần kinh nghiệm.
 - **Biện pháp khắc phục:**
 - **Grid Search hoặc Random Search:** Tự động tìm kiếm các tham số tốt nhất qua thử nghiệm nhiều tổ hợp khác nhau.
 - **Bayesian Optimization:** Kỹ thuật tối ưu hoá thông minh giúp tăng tốc độ tìm kiếm các tham số tốt nhất.
 - **AutoML:** Các công cụ như Google AutoML hoặc Auto-Keras tự động hóa việc tối ưu hóa mạng nơ-ron.

4.3 Huấn luyện mô hình

Thiết lập GridSearchCV và tìm tham số tốt nhất trên 3 tập và nhận xét đưa ra chỉ số tối ưu

```
• # 1 Định nghĩa lưới tham số cho GridSearch
• param_grid = {
•     'hidden_layer_sizes': [ (50,), (100,), (50, 50)],
•     'activation': [ 'relu', 'tanh'],
•     'solver': [ 'adam', 'sgd', ],
•     'max_iter': [ 1000, 2000, ]
• }
•
• mlp = MLPRegressor(random_state=1)
•
• # 2 Tìm kiếm các tham số tốt nhất
• grid_search = GridSearchCV(mlp, param_grid, cv=5,
•     scoring='neg_mean_squared_error')
```

param_grid là từ điển chứa các tham số và các giá trị tiềm năng mà GridSearchCV sẽ thử nghiệm.

- **'hidden_layer_sizes'**: Đây là cấu hình số lượng nơ-ron trong các lớp ẩn của MLP.
 - (50,): Một lớp ẩn với 50 nơ-ron.
 - (100,): Một lớp ẩn với 100 nơ-ron.
 - (50, 50): Hai lớp ẩn, mỗi lớp có 50 nơ-ron.
- **'activation'**: Chọn hàm kích hoạt cho nơ-ron trong các lớp ẩn.
 - 'relu': Hàm kích hoạt ReLU (Rectified Linear Unit).
 - 'tanh': Hàm kích hoạt tanh.
- **'solver'**: Chọn thuật toán tối ưu hóa để cập nhật trọng số.
 - 'adam': Thuật toán tối ưu hóa Adam.
 - 'sgd': Thuật toán Gradient Descent ngẫu nhiên.
- **'max_iter'**: Số lần lặp tối đa trong quá trình huấn luyện của MLP.
 - 1000: 1000 epoch.
 - 2000: 2000 epoch.

```
mlp = MLPRegressor(random_state=1)
```

- **MLPRegressor** là mô hình hồi quy mạng nơ-ron đa lớp (MLP) được sử dụng cho bài toán hồi quy.
- **random_state=1**: Thiết lập seed để đảm bảo kết quả tái lập được qua mỗi lần chạy.

```
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='neg_mean_squared_error')
```

- **GridSearchCV** là một kỹ thuật tìm kiếm theo lưới (grid search) để thử tất cả các kết hợp tham số trong param_grid và chọn ra tham số tối ưu nhất.
- **cv=5**: Sử dụng 5-fold cross-validation, chia dữ liệu thành 5 tập con để đánh giá độ chính xác của mô hình trên từng tập.
- **scoring='neg_mean_squared_error'**: Điểm đánh giá là **negative MSE** (để tìm kiếm giá trị thấp nhất), giúp đánh giá sai số trung bình bình phương giữa dự đoán và giá trị thực tế.

Tìm chỉ số tốt nhất

```
• print("\n Trên tập train")
• grid_search.fit(Train_X_std, Train_Y)
• # Hiển thị tham số tốt nhất
• print("Best parameters found: ", grid_search.best_params_)
```

- `grid_search.fit()`: Huấn luyện grid_search, là đối tượng GridSearchCV, trên tập huấn luyện đã chuẩn hóa (Train_X_std) và giá trị mục tiêu (Train_Y).
- GridSearchCV sẽ thử tất cả các tổ hợp tham số có trong param_grid đã định nghĩa trước đó và chọn tổ hợp cho ra điểm số tốt nhất (ở đây là **negmean_squared_error**).
- Sau khi hoàn tất quá trình tìm kiếm, grid_search sẽ lưu trữ tham số tốt nhất trong thuộc tính best_params_.
- `grid_search.best_params_`: Thuộc tính này của GridSearchCV chứa tổ hợp tham số tối ưu được chọn dựa trên kết quả cross-validation.
- Lệnh này sẽ in ra các tham số (từ param_grid) mà giúp mô hình đạt được hiệu suất tốt nhất.

Kết quả: `Best parameters found: {'activation': 'tanh', 'hidden_layer_sizes': (50, 50), 'max_iter': 2000, 'solver': 'adam'}`

Huấn luyện mô hình với chỉ số tốt nhất:

```
• mlp_reg = MLPRegressor(  
•     activation='relu',  
•     hidden_layer_sizes=(50, 50),  
•     solver='adam',  
•     max_iter=2000,  
•     random_state=1,  
•     early_stopping=True,  
•     n_iter_no_change=10)  
•  
• mlp_reg.fit(Train_X_std, Train_Y)  
• # Lưu mô hình  
• joblib.dump(mlp_reg, '../savefile/mlp_regression_model1.pkl')
```

- **activation='relu'**: Sử dụng hàm kích hoạt ReLU (Rectified Linear Unit) cho các nơ-ron trong các lớp ẩn.
- **hidden_layer_sizes=(50, 50)**: Mô hình có hai lớp ẩn, mỗi lớp có 50 nơ-ron.
- **solver='adam'**: Dùng thuật toán tối ưu hóa Adam cho quá trình huấn luyện.
- **max_iter=2000**: Giới hạn số vòng lặp tối đa trong quá trình huấn luyện là 2000 epoch.
- **random_state=1**: Thiết lập seed cho quá trình ngẫu nhiên để tái lập kết quả.
- **early_stopping=True**: Kích hoạt cơ chế **early stopping**, ngừng huấn luyện sớm nếu mô hình không cải thiện sau một số vòng lặp.
- **n_iter_no_change=10**: Số vòng lặp không thay đổi trước khi dừng huấn luyện sớm là 10.
- **fit()**: Hàm này huấn luyện mlp_reg trên tập dữ liệu huấn luyện (Train_X_std, Train_Y), điều chỉnh các trọng số của mô hình dựa trên lỗi dự đoán để cải thiện hiệu suất.
- **joblib.dump()**: Lưu mô hình mlp_reg đã huấn luyện vào file

Đánh giá mô hình: `evaluate_model(mlp_reg, Train_X_std, Train_Y, Val_X_std, Val_Y, Test_X_std, Test_Y)`

evaluate_model(): Hàm đánh giá mô hình mlp_reg trên các tập dữ liệu huấn luyện, xác thực, và kiểm tra.

4.4 Đánh giá mô hình

Chỉ số đánh giá

Dataset	R2 Score	MSE	RMSE
Training	0.9734	0.7564	0.8697
Validation	0.9751	0.6695	0.8182
Testing	0.9871	0.2742	0.5236

Nhận xét:

1. Training Set

- R^2 Score = 0.9734: Điểm R^2 cao, cho thấy mô hình có khả năng giải thích 97.34% biến động của dữ liệu huấn luyện. Đây là dấu hiệu tốt, cho thấy mô hình đã học khá chính xác các đặc trưng trong tập huấn luyện.
- MSE = 0.7564 và RMSE = 0.8697: Mean Squared Error và Root Mean Squared Error đều nhỏ, cho thấy sai số trung bình giữa giá trị thực tế và giá trị dự đoán là khá thấp.

2. Validation Set

- R^2 Score = 0.9751: Điểm R^2 này tương đương với tập huấn luyện, điều này cho thấy mô hình có khả năng tổng quát tốt, không chỉ ghi nhớ mà còn hiểu được các mối quan hệ trong dữ liệu.
- MSE = 0.6695 và RMSE = 0.8182: Giá trị này nhỏ hơn một chút so với tập huấn luyện, cho thấy mô hình hoạt động tốt và duy trì tính tổng quát trên dữ liệu xác thực.

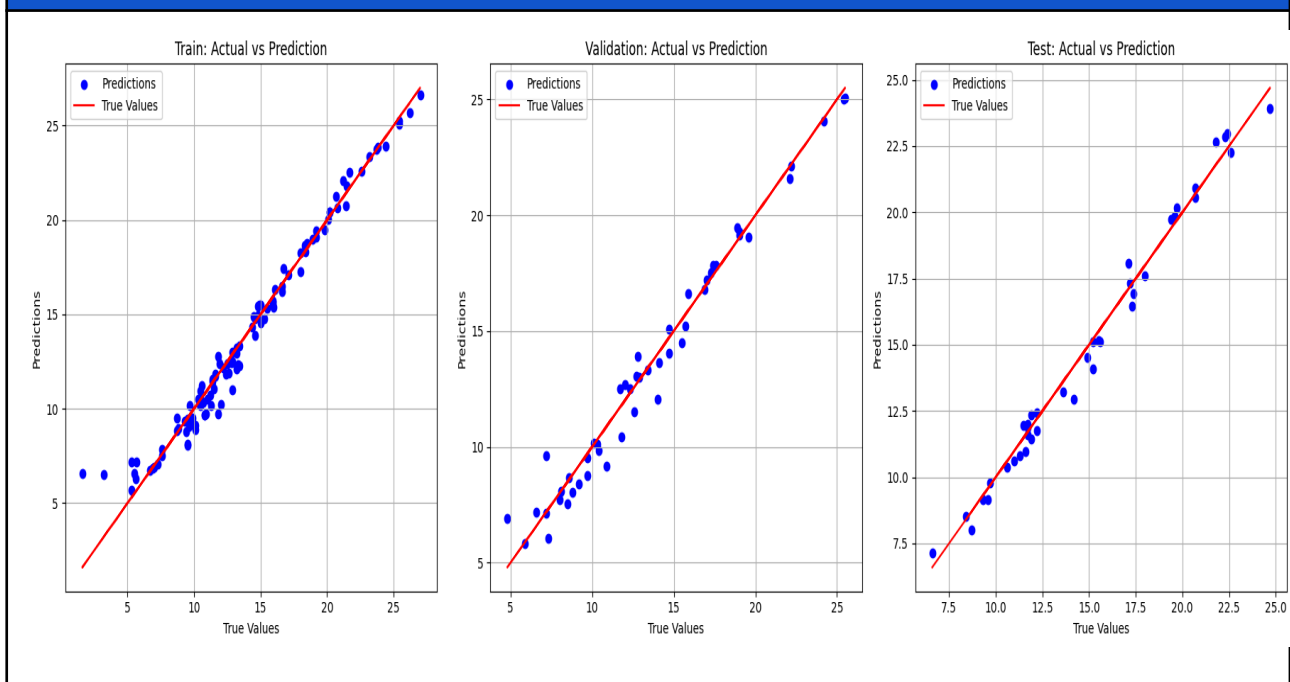
3. Testing Set

- R^2 Score = 0.9871: Điểm R^2 rất cao, cho thấy mô hình có thể giải thích được tới 98.71% biến động của dữ liệu kiểm tra, cao hơn cả hai tập trước. Điều này cho thấy mô hình có độ chính xác rất cao trên dữ liệu mới, chưa được thấy trong quá trình huấn luyện.
- MSE = 0.2742 và RMSE = 0.5236: Các giá trị lỗi trên tập kiểm tra này rất nhỏ, phản ánh sự chính xác cao của mô hình trong việc dự đoán dữ liệu chưa thấy.

Kết luận

- Mô hình hoạt động rất tốt, với độ chính xác cao trên cả ba tập dữ liệu (huấn luyện, xác thực và kiểm tra).
- R^2 Score trên tất cả các tập đều cao, cho thấy mô hình nắm bắt được các đặc trưng quan trọng của dữ liệu và có khả năng tổng quát hóa tốt.
- MSE và RMSE đều rất nhỏ, đặc biệt là trên tập kiểm tra, chứng tỏ mô hình dự đoán tốt mà không bị overfitting.

Biểu đồ thể hiện mối quan hệ giữa giá trị thực và giá trị đoán



Nhận xét:

1. Training Set

- **Đánh giá tổng quan:** Dữ liệu dự đoán (chấm màu xanh) nằm rất sát đường chéo (màu đỏ) – đường biểu diễn giá trị thực. Điều này cho thấy mô hình đã học rất tốt mối quan hệ giữa các biến đầu vào và biến mục tiêu trên tập huấn luyện.
 - **Phân bố điểm:** Hầu hết các điểm dữ liệu nằm gần đường chéo, nhưng có một vài điểm nằm hơi xa đường chéo, cho thấy vẫn có một chút sai lệch trong dự đoán, nhưng không đáng kể.
- **Kết luận:** Mô hình hoạt động tốt trên tập huấn luyện, chỉ có một số điểm có sai số nhỏ, cho thấy khả năng học tốt.

2. Validation Set

- **Đánh giá tổng quan:** Kết quả trên tập xác thực vẫn rất tốt với phần lớn các điểm dữ liệu dự đoán nằm sát đường chéo.
- **Phân bố điểm:** Mặc dù có một vài điểm nằm lệch xa khỏi đường chéo hơn so với tập huấn luyện, nhưng nhìn chung phân bố vẫn rất sát, cho thấy mô hình tổng quát hóa khá tốt.
- **Kết luận:** *Mô hình dự đoán chính xác trên tập xác thực, cho thấy khả năng không chỉ ghi nhớ mà còn hiểu được mối quan hệ trong dữ liệu, đảm bảo tính tổng quát hóa.*

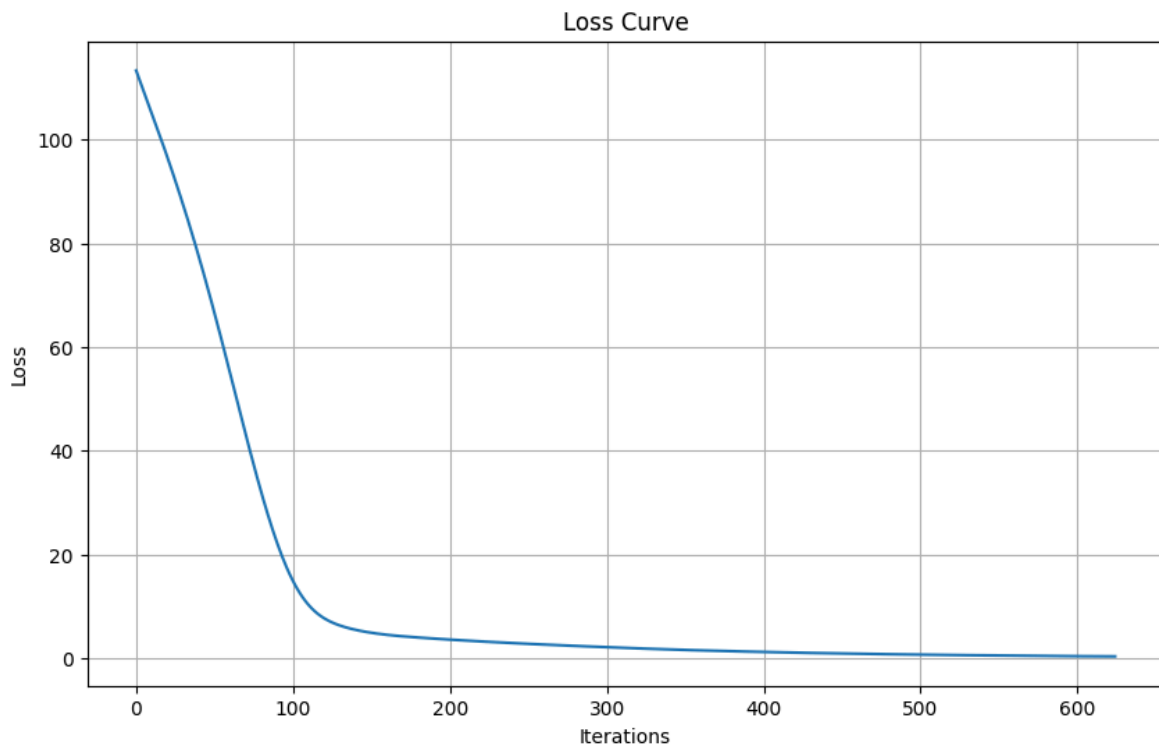
3. Testing Set

- **Đánh giá tổng quan:** Trên tập kiểm tra, các điểm dự đoán tiếp tục phân bố gần đường chéo, cho thấy độ chính xác của mô hình khi dự đoán trên dữ liệu mới, chưa từng thấy.
- **Phân bố điểm:** Hầu hết các điểm nằm sát đường chéo, với một số điểm vẫn lệch nhưng ở mức chấp nhận được, chứng tỏ mô hình vẫn có thể dự đoán tốt trên tập kiểm tra.
- **Kết luận:** *Với kết quả này, mô hình có khả năng dự đoán rất tốt trên dữ liệu mới, và các điểm sai lệch vẫn ở mức thấp*

Tổng kết

- Mô hình có độ chính xác cao và thể hiện tính tổng quát hóa tốt trên cả ba tập dữ liệu (train, validation, và test).
- Các biểu đồ này cho thấy mô hình không bị overfitting và có khả năng dự đoán chính xác cho các trường hợp mới.
- Đây là một dấu hiệu rất tốt về tính ổn định và hiệu quả của mô hình trong việc dự đoán trên tập dữ liệu chưa từng thấy.

Đồ thị hàm mất mát



Nhận xét

- **Giảm nhanh ban đầu:** Trong giai đoạn đầu của quá trình huấn luyện, Loss giảm rất nhanh. Điều này cho thấy mô hình dễ dàng học được những đặc trưng cơ bản từ dữ liệu.
- **Giảm chậm dần về sau:** Khi số lần lặp tăng lên, tốc độ giảm của Loss chậm lại. Điều này cho thấy mô hình đã bắt đầu học được những đặc trưng phức tạp hơn và cần nhiều lần lặp hơn để cải thiện thêm.
- **Ổn định:** Đến cuối quá trình, đường cong Loss gần như nằm ngang, cho thấy mô hình đã hội tụ và không còn cải thiện đáng kể nữa.

5. Stacking

5.1 Lý do chọn mô hình

Tận dụng điểm mạnh của các base model (ở đây là Linear Regression, Ridge, Neural Network) và giảm thiểu những điểm yếu của từng mô hình riêng lẻ:

1. Các mô hình Linear Regression và Ridge đều là các mô hình tuyến tính, còn Neural Network lại có khả năng học các mối quan hệ phi tuyến tính. Sự kết hợp của chúng giúp cải thiện độ chính xác, vì Stacking có thể “học” từ cả các xu hướng tuyến tính và phi tuyến tính trong dữ liệu. => **Đa dạng trong khả năng học dữ liệu**
 2. Mỗi mô hình có xu hướng tạo ra lỗi ở các điểm khác nhau trong dữ liệu. Stacking sẽ sử dụng một mô hình meta để tìm hiểu và điều chỉnh các lỗi này bằng cách kết hợp dự đoán từ các mô hình khác nhau. => **Giảm thiểu lỗi từng mô hình**
 3. Với Neural Network, mô hình có thể học được các mối quan hệ phức tạp hơn, trong khi Linear Regression và Ridge có lợi thế ở việc tính toán nhanh và đơn giản. Stacking sẽ giúp mô hình meta học cách cân bằng giữa tính phức tạp và khả năng dự đoán tổng quát. => **Tăng độ chính xác**
- *Khi dùng Stacking để kết hợp 3 mô hình kể trên sẽ cho mô hình học máy có hiệu suất tốt hơn nhờ vào việc tận dụng các đặc điểm riêng biệt của từng mô hình*

5.2 Tổng quan về mô hình stacking

5.2.1 Định nghĩa

- ❖ Là một thuật toán học máy kết hợp các dự đoán từ nhiều mô hình học máy trên cùng một tập dữ liệu
- ❖ Mô hình Stacking cơ bản thường được phân thành 2 cấp Base-Models và Meta-Model:
 - Base-Models: Mô hình cơ sở học trực tiếp từ bộ dữ liệu và đưa ra dự đoán cho mô hình meta.
 - Meta-Model: Mô hình học từ các dự đoán của mô hình cơ sở. Meta-model được huấn luyện dựa trên đầu ra dự đoán của các base-modes, các outputs này kết hợp với nhãn của bài toán tạo thành cặp dữ liệu đầu vào - đầu ra trong quá trình huấn luyện Model

5.2.2. Ý tưởng

Tư tưởng của học nhóm là khai thác sức mạnh của tập thể:

- Nếu không dùng học kết hợp: Ta có một model nhưng đầu ra của model đó không tốt nên ta phải thử các model khác.

- Sau khi tìm được model ưng ý, ta lại phải chỉnh chỉnh sửa sửa từ thuật toán đến hyperparameter để mô hình đạt độ chính xác cao nhất.

=> Hai việc này sẽ tốn thời gian bởi ta phải chạy từng model một

=> Để nhanh hơn, ta kết hợp những model "học yếu" này lại để tạo ra một model "học mạnh" hơn, không những thế kết quả thu được cũng tốt hơn so với từng model một

5.2.3. Các bước thực hiện

1. Sử dụng các base-models để học trên toàn bộ dữ liệu và đưa ra kết quả dự đoán ban đầu
2. Xây dựng bộ dữ liệu mới dựa trên outputs của các base-models
3. Huấn luyện Meta-model với bộ dữ liệu mới và đưa ra kết quả cuối cùng

5.2.4 Ưu và nhược điểm mô hình

Ưu điểm

- Cải thiện độ chính xác: Bằng cách kết hợp nhiều mô hình khác nhau (base models), stacking thường giúp cải thiện độ chính xác của dự đoán so với việc chỉ sử dụng một mô hình đơn lẻ. Đặc biệt hiệu quả khi các mô hình cơ sở có những ưu điểm và nhược điểm khác nhau, vì stacking giúp giảm thiểu các lỗi cố hữu của từng mô hình
- Tính linh hoạt cao: Có thể kết hợp các loại mô hình khác nhau (như các mô hình tuyến tính, cây quyết định, mạng neuron) để tạo ra một mô hình tổng hợp tốt hơn. Điều này giúp tận dụng tối đa khả năng của từng mô hình thành phần, tối ưu hóa sức mạnh của các mô hình khác nhau
- Giảm overfitting: Khi các mô hình thành phần không phụ thuộc mạnh mẽ vào nhau, stacking có thể giúp giảm overfitting do tính đa dạng của các mô hình được kết hợp
- Tận dụng được các mô hình yếu: Một số mô hình yếu nhưng lại có khả năng phát hiện tốt một số kiểu mẫu cụ thể trong dữ liệu. Khi kết hợp với các mô hình khác, stacking có thể cải thiện tổng thể dự đoán từ các mô hình này

Nhược điểm

- Tính phức tạp và tốn thời gian: Stacking yêu cầu việc huấn luyện và kiểm tra nhiều mô hình, do đó thời gian huấn luyện sẽ lâu hơn và tài nguyên tính toán cũng tăng lên. Kỹ

thuật này đòi hỏi việc tối ưu hóa không chỉ cho từng mô hình thành phần mà còn cho mô hình meta

- Dễ overfitting nếu không cẩn thận: Nếu không áp dụng các kỹ thuật như cross-validation đúng cách hoặc nếu mô hình meta quá phức tạp, stacking có thể dẫn đến overfitting, đặc biệt khi kích thước dữ liệu nhỏ

5.3 Huấn luyện mô hình

Load các model, các tập dữ liệu

- Sử dụng joblib để load các model

```
• linear_model = joblib.load('../savefile/linear_regression_model.pkl')  
• mlp_model = joblib.load('../savefile/mlp_regression_model1.pkl')  
• ridge_model = joblib.load('../savefile/ridge_regression_model.pkl')
```

- Sử dụng pandas để load các file dữ liệu từ tập train, test, validation

```
• Train_X_std = pd.read_csv('../savefile/Train_X.csv')  
• Train_Y = pd.read_csv('../savefile/Train_Y.csv').values.ravel()  
• Val_X_std = pd.read_csv('../savefile/Val_X.csv')  
• Val_Y = pd.read_csv('../savefile/Val_Y.csv').values.ravel()  
• Test_X_std = pd.read_csv('../savefile/Test_X.csv')  
• Test_Y = pd.read_csv('../savefile/Test_Y.csv').values.ravel()  
•
```

Tạo các dự đoán từ các mô hình cơ sở

- Dự đoán trên tập train

```
• kf = KFold(n_splits=5, shuffle=True, random_state=42)  
• # Tạo các dự đoán cho từng lớp mô hình cơ sở
```

```
• train_pred_linear_cv = cross_val_predict(linear_model, Train_X_std,
Train_Y, cv=kf)
• train_pred_mlp_cv = cross_val_predict(mlp_model, Train_X_std,
Train_Y, cv=kf)
• train_pred_ridge_cv = cross_val_predict(ridge_model, Train_X_std,
Train_Y, cv=kf)
• # Ghép các dự đoán thành ma trận đặc trưng cho mô hình meta
• train_meta_X_cv = np.column_stack((train_pred_linear_cv,
train_pred_mlp_cv, train_pred_ridge_cv))
```

- Ở đây dùng kỹ thuật cross-validation (k-fold) để chia tập huấn luyện thành nhiều phần nhỏ hơn (folds), giúp tối ưu hóa việc huấn luyện cho các mô hình cơ sở. Với các chỉ số :

1. `n_splits=5`: chia tập huấn luyện thành 5 tập con (fold) khác nhau
2. `shuffle=True`: giúp xáo trộn dữ liệu trước khi chia fold, giúp tăng tính ngẫu nhiên và tránh các vấn đề như dữ liệu không cân bằng theo thứ tự
3. `random_state=42`: đảm bảo tính tái lập (reproducibility) khi chạy lại mã

- Hàm `cross_val_predict`: huấn luyện các base model trên các fold khác nhau và tạo dự đoán cho từng mẫu trong dữ liệu, giúp tạo ra một mảng dự đoán dựa trên các mẫu chưa được thấy trong mỗi fold. Ở đây tạo ra 3 mảng dự đoán của 3 mô hình linear, mlp, ridge

- Hàm `np.column_stack`: lấy từng mảng `train_pred_linear_cv`, `train_pred_mlp_cv`, và `train_pred_ridge_cv` và xếp chúng lại thành các cột trong một ma trận mới

➤ *Kết quả cho ra `train_meta_X_cv` là tập dữ liệu đầu vào cho meta-model trong stacking*

Demo dữ liệu tập `train_pred_linear_cv`

```
Linear_Predictions
11.781005300074577
6.486634388478072
17.602153282338733
18.54370319347305
10.576511447148716
5.20882669074661
14.882020439169182
15.356128126589555
```

```
17.911273658983184
21.043965445438047
3.4386638539403513
7.387019936128068
12.165635640438046
21.803537843953716
12.757876173963766
10.105123236997859
15.563647215480518
20.74846543543289
6.367928153415421
7.048250580876406
14.227034879072727
15.168020536291182
9.794733882114691
13.71622453158545
8.822214392394352
15.221637980638132
19.301207779801047
```

Demo dữ liệu tập train_pred_mlp_cv

```
MLP_Predictions
9.518337994790112
8.971205510534151
18.77167521326983
16.347813515685104
11.859889203375323
7.8349727192874
14.569803109864319
13.24657568000905
19.432095978652587
20.635576621460253
5.6950585439290125
7.495172073610341
9.351628491953372
20.769336536235244
12.46772243536284
6.567678221804602
15.371242464253772
22.559798281863884
6.279422838991136
8.823778331316554
14.34834188500658
15.466914237538136
10.164198277273334
11.82179004230556
10.406337565915887
```

Demo dữ liệu tập train_pred_ridge_cv

```
Ridge_Predictions
11.853171482278226
6.563538886908831
17.577580203756924
18.496056611068706
10.620150029066968
5.304212475795955
14.853005750456918
15.323743571610775
17.88207436174297
20.982209730921205
3.5666404578429027
7.454653788730142
12.214900910535889
21.718263835755767
12.754267530635039
10.12756294788986
15.567970040731726
20.650733878964296
6.47763607019934
7.130623958409131
14.224784333974391
15.153973712625561
9.834378788786275
```

Demo dữ liệu tập train_meta_X_cv

```
Linear_Predictions,MLP_Predictions,Ridge_Predictions
11.781005300074577,9.518337994790112,11.853171482278226
6.486634388478072,8.971205510534151,6.563538886908831
17.602153282338733,18.77167521326983,17.577580203756924
18.54370319347305,16.347813515685104,18.496056611068706
10.576511447148716,11.859889203375323,10.620150029066968
5.20882669074661,7.8349727192874,5.304212475795955
14.882020439169182,14.569803109864319,14.853005750456918
15.356128126589555,13.24657568000905,15.323743571610775
17.911273658983184,19.432095978652587,17.88207436174297
21.043965445438047,20.635576621460253,20.982209730921205
3.4386638539403513,5.6950585439290125,3.5666404578429027
7.387019936128068,7.495172073610341,7.454653788730142
12.165635640438046,9.351628491953372,12.214900910535889
21.803537843953716,20.769336536235244,21.718263835755767
```

```
12.757876173963766,12.46772243536284,12.754267530635039
10.105123236997859,6.567678221804602,10.12756294788986
15.563647215480518,15.371242464253772,15.567970040731726
20.74846543543289,22.559798281863884,20.650733878964296
6.367928153415421,6.279422838991136,6.47763607019934
7.048250580876406,8.823778331316554,7.130623958409131
14.227034879072727,14.34834188500658,14.224784333974391
15.168020536291182,15.466914237538136,15.153973712625561
9.794733882114691,10.164198277273334,9.834378788786275
13.71622453158545,11.82179004230556,13.693008879705388
8.822214392394352,10.406337565915887,8.891678177984506
15.221637980638132,13.327703804636771,15.190533452385163
19.301207779801047,20.01593984113901,19.24015193488657
11.539426245927539,9.708812431301933,11.559382117676014
21.71781930842788,23.352616828878123,21.608378618698687
8.265746542983972,9.526654839304145,8.31554980983488
8.739842009370804,10.211983036661731,8.810258026681279
```

- Dự đoán trên tập test

```
• test_pred_linear = linear_model.predict(Test_X_std)
• test_pred_mlp = mlp_model.predict(Test_X_std)
• test_pred_ridge = ridge_model.predict(Test_X_std)
• test_meta_X = np.column_stack((test_pred_linear, test_pred_mlp,
    test_pred_ridge))
```

Hàm predict(): Các base model tạo dự đoán dựa trên các tập dữ liệu test X

Hàm np.column_stack(): lấy từng mảng test_pred_linear, test_pred_mlp, test_pred_ridge và xếp chúng lại thành các cột trong một ma trận mới tạo đầu vào cho meta model

Demo dữ liệu tập Test_Predictions

```
Test_Predictions
9.21780162422497
20.356567217285875
16.975612735630957
19.31822017574858
21.55307448037679
12.790213537636381
12.138404654509387
12.395528274843423
21.11612698658525
21.58463852318067
```

```
11.1672174876755
19.69767512926605
7.156089744466924
15.275595324990912
9.6057189382467
8.78978101915311
16.31644741442542
12.103213780916429
17.271256750131958
11.7323109261502
17.040447319086113
10.220051368506242
21.48408970171977
17.505943624527543
15.210804327982771
22.621544550619426
19.409001741112885
10.400404824606936
19.787137195504684
14.569857355371985
13.83062897513332
8.317175357290743
10.439840608861724
14.754197528715364
7.827899222592009
13.382866462462975
8.285314856451269
```

- Dự đoán trên tập validation

```
• val_pred_linear = linear_model.predict(Val_X_std)
• val_pred_mlp = mlp_model.predict(Val_X_std)
• val_pred_ridge = ridge_model.predict(Val_X_std)
• val_meta_X = np.column_stack((val_pred_linear, val_pred_mlp,
    val_pred_ridge))
```

Hàm predict(): Các base model tạo dự đoán dựa trên các tập dữ liệu validation X

Hàm np.column_stack(): lấy từng mảng val_pred_linear, val_pred_mlp, val_pred_ridge và xếp chúng lại thành các cột trong một ma trận mới tạo đầu vào cho meta model

Demo dữ liệu tập Validation_Predictions

```
Validation_Predictions
18.214374671039778
18.490389891312194
```



```
5.178702818581728
11.30993036774297
19.327454617234523
12.663165401236073
14.69570220314035
13.727446999520739
6.538226284394603
15.266111025738793
8.367947636482945
18.373214999961025
13.900082632587903
14.305716875693673
9.545860444801782
10.132470616864321
14.408241374734478
```

Thực hiện huấn luyện mô hình meta

- Tìm kiếm alpha cho meta model

```
• meta_model = Ridge()
• param_grid = {'alpha': np.logspace(-3, 3, 50)} # Các giá trị alpha
  từ 0.001 đến 1000
• # Sử dụng GridSearchCV để tìm alpha tốt nhất
• grid_search = GridSearchCV(meta_model, param_grid, cv=kf,
  scoring='r2')
• grid_search.fit(train_meta_X_cv, Train_Y)
• # Lấy ra alpha tốt nhất
• best_alpha = grid_search.best_params_['alpha']
```

1. Tạo mô hình Ridge Regression để làm meta-model trong stacking
2. **param_grid**: là một từ điển chứa các giá trị của alpha sẽ được thử nghiệm trong quá trình tìm kiếm tham số tối ưu
3. **np.logspace(-3, 3, 50)**: tạo ra 50 giá trị của alpha từ $10^{-3}=0.001$ đến $10^3=1000$, giúp tìm kiếm trên một phạm vi rộng từ nhỏ đến lớn để xem giá trị nào giúp mô hình hoạt động tốt nhất
4. Sử dụng GridSearchCV : **param_grid = {'alpha': np.logspace(-3, 3, 50)}**

- GridSearchCV tạo ra một mô hình Ridge Regression với các giá trị alpha khác nhau, chạy cross-validation để tìm giá trị alpha tối ưu dựa trên điểm số R^2 (mức độ phù hợp của mô hình)
- **cv=kf**: Sử dụng đối tượng kf (tạo từ KFold) để thực hiện cross-validation, chia dữ liệu thành các fold giúp mô hình được đánh giá trên các tập con khác nhau của dữ liệu huấn luyện
- **scoring='r2'**: Sử dụng R^2 làm tiêu chí đánh giá để chọn ra mô hình có độ phù hợp cao nhất

5. `param_grid = {'alpha': np.logspace(-3, 3, 50)}`

- Huấn luyện grid_search trên dữ liệu **train_meta_X_cv** (các dự đoán từ mô hình cơ sở) và **Train_Y** (giá trị thực).
- GridSearchCV sẽ thử tất cả các giá trị của alpha trong **param_grid** và chọn ra giá trị tối ưu bằng cách đánh giá các kết quả cross-validation trên từng giá trị. Sau đó lấy là giá trị alpha tốt nhất để tiến hành huấn luyện meta model bằng hàm **grid_search.best_params_['alpha']**

- Huấn luyện meta model

```
• meta_model_opt = Ridge(alpha=best_alpha)
• meta_model_opt.fit(train_meta_X_cv, Train_Y)
```

Tạo một mô hình Ridge Regression với giá trị alpha tối ưu (best_alpha) vừa tìm được từ quá trình GridSearchCV bằng **Ridge(alpha=best_alpha)**

Huấn luyện mô hình meta_model_opt trên **train_meta_X_cv**, là dữ liệu đầu ra từ các mô hình cơ sở (dự đoán của các mô hình cơ sở), và **Train_Y**, là nhãn thực tế

Thực hiện tính toán các chỉ số

Trên tập huấn luyện

```
• # Tính toán các chỉ số hiệu suất trên tập huấn luyện
• train_mse = mean_squared_error(Train_Y, train_meta_pred)
• train_rmse = np.sqrt(train_mse)
• train_r2 = r2_score(Train_Y, train_meta_pred)
```

Trên tập kiểm tra và xác thực

- # Tính toán các chỉ số hiệu suất trên tập xác thực và tập kiểm tra
- `val_mse = mean_squared_error(Val_Y, val_meta_pred)`
- `val_rmse = np.sqrt(val_mse)`
- `val_r2 = r2_score(Val_Y, val_meta_pred)`
-
- `test_mse = mean_squared_error(Test_Y, test_meta_pred)`
- `test_rmse = np.sqrt(test_mse)`
- `test_r2 = r2_score(Test_Y, test_meta_pred)`

Thực hiện lưu mô hình

- `joblib.dump(meta_model_opt, '../savefile/stacking_regressor_model.p')`

5.4 Đánh giá mô hình

Chỉ số đánh giá

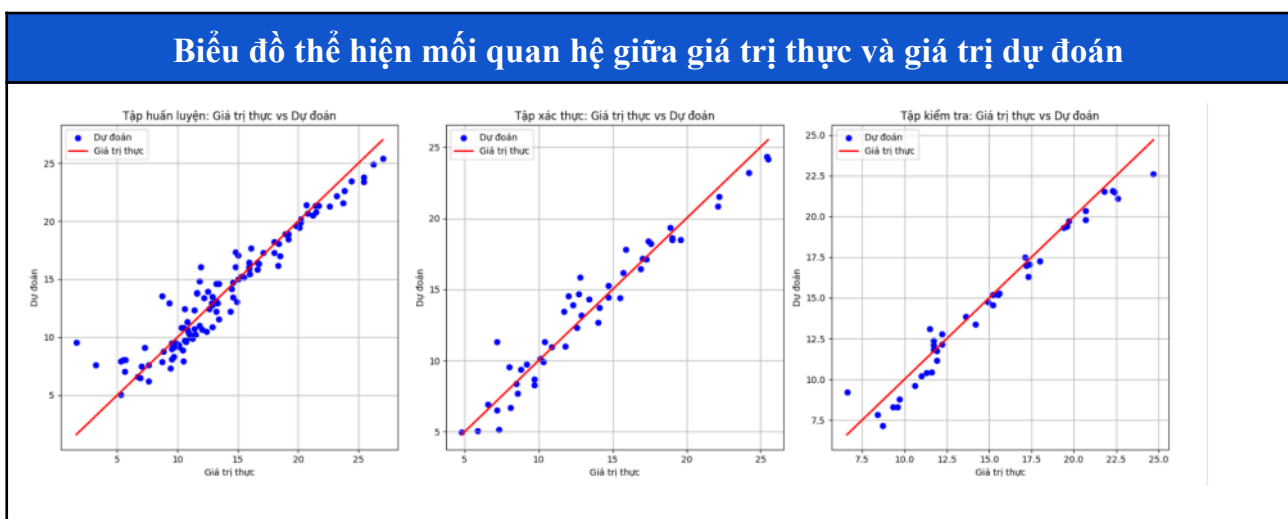
Dataset	R2 Score	MSE	RMSE
Training	0.9400	1.7073	1.3066
Validation	0.9409	1.5895	1.2607
Testing	0.9611	0.8263	0.9090

Nhận xét:

- ❖ Độ chính xác
 - Trên cả ba bộ dữ liệu (Training: 0.9400, Validation: 0.9409, Testing: 0.9611), R^2 đều khá cao (gần 1). Điều này cho thấy mô hình đã nắm bắt được phần lớn mối quan hệ giữa các đặc trưng và biến mục tiêu, tức là mô hình dự đoán khá chính xác
 - Giá trị R^2 cao trên cả bộ Training và Validation cho thấy mô hình đã học được quy luật chung từ dữ liệu huấn luyện và cũng có khả năng dự đoán tốt trên dữ liệu xác thực
 - Trên bộ dữ liệu Testing, MSE (0.8263) và RMSE (0.9090) đều nhỏ hơn so với các bộ Training và Validation. Đây là một dấu hiệu tích cực cho thấy mô hình đã có khả năng dự đoán chính xác và chính xác hơn nữa trên dữ liệu chưa từng thấy (Testing).

- Sự đồng nhất của MSE và RMSE giữa các bộ dữ liệu cũng cho thấy sai số của mô hình nhỏ trên cả ba tập, điều này chứng minh mô hình có độ chính xác cao
- ❖ Khả năng tổng quát hóa
- R^2 trên bộ Training (0.9400), Validation (0.9409) và Testing (0.9611) khá gần nhau, điều này cho thấy mô hình không bị lệch về bất kỳ bộ nào và có thể duy trì khả năng dự đoán ổn định giữa các bộ dữ liệu khác nhau
- Đặc biệt, R^2 trên bộ Testing cao hơn so với Training và Validation, cho thấy mô hình đã nắm bắt được các đặc trưng của dữ liệu chung mà không chỉ đơn thuần là học thuộc dữ liệu huấn luyện
- ❖ Hiệu suất
- Các chỉ số MSE và RMSE thấp trên cả ba tập dữ liệu cho thấy mô hình có hiệu suất cao khi xử lý và dự đoán các giá trị mục tiêu
- Với giá trị RMSE nhỏ (0.9090 trên bộ Testing), ta có thể kết luận mô hình dự đoán sát với giá trị thực tế, từ đó nâng cao hiệu suất của mô hình trong bối cảnh ứng dụng thực tế
- Do mô hình duy trì hiệu suất tốt trên cả ba tập, điều này cho thấy mô hình này có thể được triển khai và ứng dụng rộng rãi mà không cần quá nhiều điều chỉnh
- RMSE trên Testing thấp cho thấy hiệu suất mô hình sẽ duy trì được khi áp dụng lên dữ liệu thực tế, giúp tiết kiệm thời gian và nguồn lực cho việc hiệu chỉnh

Biểu đồ thể hiện mối quan hệ giữa giá trị thực và giá trị dự đoán



- **Độ chính xác:** Trên tập huấn luyện, các điểm xanh nằm sát đường hồi quy, cho thấy mô hình đạt độ chính xác cao trên dữ liệu mà nó đã học. Trên tập xác thực và kiểm tra, mặc dù

có một số điểm lệch xa đường hồi quy hơn, phần lớn các điểm vẫn nằm gần đường này, cho thấy mô hình duy trì độ chính xác tốt trên dữ liệu chưa từng thấy trong quá trình huấn luyện

- **Hiệu suất:** Trên cả ba tập dữ liệu, các điểm dữ liệu thực tế (màu xanh) phân bố khá sát đường hồi quy lý tưởng (màu đỏ). Điều này cho thấy mô hình có khả năng dự đoán tốt các giá trị mục tiêu. Trên tập xác thực và kiểm tra, mặc dù có một số điểm lệch xa đường hồi quy hơn, phần lớn các điểm vẫn nằm gần đường này, cho thấy mô hình duy trì độ chính xác tốt trên dữ liệu chưa từng thấy trong quá trình huấn luyện


- **Khả năng tổng quát hóa:** Mô hình tổng quát hóa tốt, vì các dự đoán trên tập xác thực và tập kiểm tra khá chính xác, không có dấu hiệu rõ ràng của hiện tượng overfitting (quá khớp) hoặc underfitting (chưa khớp). Có một vài điểm ở xa đường hồi quy trong tập kiểm tra, điều này có thể là do nhiễu trong dữ liệu hoặc giới hạn của mô hình

IV. Chương trình demo

Đây là chương trình giải quyết bài toán dự đoán doanh số bán hàng (Sales) dựa trên ngân sách quảng cáo từ các kênh khác nhau: Ngân sách quảng cáo trên TV (TV_Ad_Budget), Ngân sách quảng cáo trên Radio (Radio_Ad_Budget), và Ngân sách quảng cáo trên Báo (Newspaper_Ad_Budget) bằng các phương pháp học máy đã được nêu ở phần trên: Hồi quy tuyến tính (Linear Regression), Hồi quy Ridge (Ridge Regression), Mạng nơ-ron (Neural Network), và Stacking.

Ta sẽ nhập các trường thông tin đầu vào gồm TV_Ad_Budget, Radio_Ad_Budget, Newspaper_Ad_Budget và nhấn nút “Predict”. Sau khi bấm nút “Predict”, chương trình sẽ thực hiện dự đoán giá trị doanh số bán hàng (Sales) và các thông số đánh giá mô hình cùng với sơ đồ “Phân tán giá trị thực tế và giá trị dự đoán” tương ứng cho quá trình huấn luyện dữ liệu bằng mô hình được chọn.

Giao diện demo

TRƯỜNG ĐẠI HỌC THUY LỢI
THUYLOI UNIVERSITY

Predict Sales

Choose a model:

Linear Regression

TV Ad Budget:

Radio Ad Budget:

Newspaper Ad Budget:

Predict

Prediction: 9.729132726378424

Dưới đây là Source Code và Link Demo để trải nghiệm chương trình

[Link website](#)



[Link code](#)

