TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN

Nhóm sinh viên thực hiện:

1. Vũ Anh Xuân - 2151264697
2. Đào Thị Trang - 2151264688
3. Khổng Quốc Trung - 2151264690
4. Nguyễn Quang Việt - 2151264694

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN TIỀN XỬ LÝ DỮ LIỆU

TIỀN XỬ LÝ DỮ LIỆU DỰ ĐOÁN CƠ HỘI VIỆC LÀM TRONG LĨNH VỰC KHOA HỌC DỮ LIỆU

NGƯỜI HƯỚNG DẪN: TS. Tạ Quang Chiểu

Hà Nội, năm 2023

MỤC LỤC

PHẦN 1: GIỚI THIỆU BÀI TOÁN

a) Hiểu dữ liệu

- Nguồn dữ liệu: data_science_job.csv
- Dữ liệu gồm 19158 bản ghi và 13 cột thuộc tính trong đó cột target làm nhãn, các cột còn lại là đặc trưng

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	NaN	NaN	36.0	1
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	47.0	0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	83.0	0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	0.0	NaN	Pvt Ltd	52.0	1
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Startup	8.0	0
5	21651	city_176	0.764	NaN	Has relevent experience	Part time course	Graduate	STEM	11.0	NaN	NaN	24.0	1
6	28806	city_160	0.920	Male	Has relevent experience	no_enrollment	High School	NaN	5.0	50-99	Funded Startup	24.0	0
7	402	city_46	0.762	Male	Has relevent experience	no_enrollment	Graduate	STEM	13.0	<10	Pvt Ltd	18.0	1
8	27107	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	7.0	50-99	Pvt Ltd	46.0	1
9	699	city_103	0.920	NaN	Has relevent experience	no_enrollment	Graduate	STEM	17.0	10000+	Pvt Ltd	123.0	0
10	29452	city_21	0.624	NaN	No relevent experience	Full time course	High School	NaN	2.0	NaN	NaN	32.0	1
11	23853	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	5.0	5000-9999	Pvt Ltd	108.0	0
12	25619	city_61	0.913	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	1000-4999	Pvt Ltd	23.0	0
13	5826	city_21	0.624	Male	No relevent experience	NaN	NaN	NaN	2.0	NaN	NaN	24.0	0
14	8722	city_21	0.624	NaN	No relevent experience	Full time course	High School	NaN	5.0	NaN	NaN	26.0	0
15	6588	city_114	0.926	Male	Has relevent experience	no_enrollment	Graduate	STEM	16.0	Oct-49	Pvt Ltd	18.0	0
16	4167	city_103	0.920	NaN	Has relevent experience	no_enrollment	Graduate	STEM	1.0	50-99	Pvt Ltd	106.0	0
17	5764	city_21	0.624	NaN	Has relevent experience	no_enrollment	Graduate	STEM	2.0	5000-9999	Pvt Ltd	7.0	0
18	2156	city_21	0.624	NaN	Has relevent experience	no_enrollment	Graduate	STEM	7.0	10000+	Pvt Ltd	23.0	1
19	11399	city_13	0.827	Female	Has relevent experience	no_enrollment	Graduate	Arts	4.0	NaN	NaN	132.0	1

Minh họa dữ liệu data_science_job.csv

*) Các thuộc tính của dữ liệu

- **enrollee_id**: Mã số duy nhất của từng ứng viên.
- **city**: Thành phố nơi ứng viên đang sinh sống hoặc làm việc.
- **city_development_index**: Chỉ số phát triển thành phố, có thể là một đánh giá về mức độ phát triển kinh tế, xã hội của thành phố.
- **gender**: Giới tính của ứng viên.
- **relevent_experience**: Kinh nghiệm liên quan đến công việc, có thể là "Has relevent experience" hoặc "No relevent experience".
- **enrolled_university**: Trạng thái đăng ký của ứng viên tại trường đại học (full-time, part-time, không có).
- **education_level**: Trình độ học vấn của ứng viên (ví dụ: graduate, masters, phd).
- **major_discipline**: Ngành học chính của ứng viên.
- **experience**: Số năm kinh nghiệm làm việc của ứng viên.
- **company_size**: Kích thước của công ty mà ứng viên đang làm việc.
- **company_type**: Loại hình công ty mà ứng viên đang làm việc.
- **training_hours**: Số giờ đào tạo mà ứng viên đã tham gia.
- **target**: Mục tiêu, chính sách hoặc biến cố mà dữ liệu đang nghiên cứu hoặc dự đoán.

b) Tầm quan trọng của việc tiền xử lý dữ liệu

- Việc tiền xử lý dữ liệu là một phần quan trọng và không thể thiếu trong quy trình phân tích dữ liệu và xây dựng mô hình. Dưới đây là một số điểm quan trọng về tầm quan trọng của việc tiền xử lý dữ liệu:
- **Loại bỏ dữ liệu không chính xác hoặc thiếu sót**: Dữ liệu không chính xác hoặc thiếu sót có thể ảnh hưởng đến kết quả phân tích và dự đoán. Tiền xử lý giúp loại bỏ những dữ liệu này, từ việc điền giá trị thiếu đến loại bỏ các quan sát bị hỏng.
- **Chuẩn hóa dữ liệu**: Chuẩn hóa dữ liệu giúp đồng nhất các đơn vị đo, phạm vi và tỷ lệ của các biến, giúp các thuật toán máy học và phân tích dễ dàng hơn.
- **Xử lý ngoại lệ và nhiễu**: Tiền xử lý giúp phát hiện và xử lý các ngoại lệ và nhiễu trong dữ liệu, giúp mô hình học tập hiệu quả hơn và dự đoán chính xác hơn.
- **Giảm số chiều của dữ liệu**: Trong một số trường hợp, dữ liệu có thể chứa quá nhiều chiều (biến), điều này có thể dẫn đến hiện tượng overfitting hoặc làm tăng độ phức tạp của mô hình. Tiền xử lý giúp giảm số chiều của dữ liệu thông qua việc chọn lọc biến hoặc giảm chiều dữ liệu.
- **Tạo biến mới và rút trích đặc trưng**: Tiền xử lý cho phép tạo ra các biến mới từ dữ liệu gốc hoặc rút trích các đặc trưng quan trọng, giúp cải thiện khả năng dự đoán của mô hình.
- **Loại bỏ sự tương quan không cần thiết**: Trong trường hợp có sự tương quan cao giữa các biến, tiền xử lý có thể loại bỏ những biến không cần thiết để tránh tình trạng đa cộng tuyến và cải thiện hiệu suất của mô hình.
- **Tăng tốc độ xử lý**: Tiền xử lý có thể giúp tăng tốc độ xử lý dữ liệu, đặc biệt là khi xử lý dữ liệu lớn, bằng cách giảm kích thước dữ liệu hoặc loại bỏ các phần không cần thiết.

PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU

A. Xử lý dữ liệu thiếu

- Dữ liệu thiếu là những quan sát trong tập dữ liệu không chứa bất kỳ giá trị nào
- Xử lý dữ liệu thiếu là một phần quan trọng của quá trình phân tích dữ liệu và máy học vì dữ liệu thiếu có thể gây ra các vấn đề lớn trong việc xây dựng mô hình và phân tích dữ liệu. Dưới đây là một số lí do tại sao chúng ta cần phải xử lý dữ liệu thiếu:
    + Hiệu suất mô hình
    + Chính xác của kết quả
    + Rủi ro khi ra quyết định
    + Ảnh hưởng đến phân tích thống kê
    + Duy trì tính đồng nhất của dữ liệu
- Các kĩ thuật xử lý dữ liệu thiếu:
    + Đối với biến liên tục :
        + Mean or Median Imputation
        + End of Distribution Imputation
        + Arbitrary Value Imputation
    + Đối với biến rời rạc:
        + Frequent Category Imputation
        + Missing Category Imputation

a. Xử lý dữ liệu số (numerical data):

Mean or Median Imputation:

- Khi nào sử dụng: Sử dụng khi dữ liệu có ít giá trị thiếu và phân phối không bị lệch nhiều.
- Phương pháp: Thay thế các giá trị thiếu bằng giá trị trung bình hoặc trung vị của biến.
- Ưu điểm: Dễ triển khai, không làm thay đổi phân phối dữ liệu ban đầu.
- Nhược điểm: Có thể làm giảm phương sai, không phản ánh được sự biến động trong dữ liệu.
- Ví dụ: Trong một tập dữ liệu về điểm số của học sinh, giá trị trung bình của điểm có thể được sử dụng để thay thế cho các giá trị thiếu.

End of Distribution Imputation:

- Khi nào sử dụng: Phù hợp khi dữ liệu có giá trị thiếu và có độ lệch cao (skewed).
- Phương pháp: Thay thế các giá trị thiếu bằng các giá trị nằm ở cuối của phân phối (ví dụ: quantile thứ 95).
- Ưu điểm: Giữ được sự biến động trong dữ liệu.
- Nhược điểm: Có thể tạo ra các giá trị ngoại lệ (outliers).
- Ví dụ: Trong một tập dữ liệu về thu nhập, có thể sử dụng giá trị ở phân vị thứ 95 để thay thế các giá trị thu nhập thiếu.

Arbitrary Value Imputation:

- Khi nào sử dụng: Thường được sử dụng khi dữ liệu không thể xác định phân phối hoặc khi không muốn ảnh hưởng đến phân phối dữ liệu.
- Phương pháp: Thay thế các giá trị thiếu bằng một giá trị cố định được chọn trước.
- Ưu điểm: Dễ triển khai, không làm thay đổi phân phối dữ liệu ban đầu.
- Nhược điểm: Có thể ảnh hưởng đến tính khách quan của dữ liệu.
- Ví dụ: Trong một tập dữ liệu về chiều cao của người, có thể sử dụng giá trị -999 để thay thế cho các giá trị thiếu.

b. Xử lý dữ liệu phân loại (categorical data):

Frequent Category Imputation:

- Khi nào sử dụng: Sử dụng khi dữ liệu phân loại có giá trị thiếu.
- Phương pháp: Thay thế các giá trị thiếu bằng giá trị phổ biến nhất (mode) trong biến đó.
- Ưu điểm: Dễ triển khai, giữ được phân phối của dữ liệu.
- Nhược điểm: Có thể làm giảm độ biến động của dữ liệu.
- Ví dụ: Trong một tập dữ liệu về ngôn ngữ yêu thích của người dùng, giá trị phổ biến nhất (ví dụ: "English") có thể được sử dụng để thay thế cho các giá trị thiếu.

Missing Category Imputation:

- Khi nào sử dụng: Sử dụng khi giá trị thiếu được coi là một danh mục riêng biệt.
- Phương pháp: Tạo một danh mục mới hoặc sử dụng một giá trị đặc biệt để chỉ ra sự thiếu sót.
- Ưu điểm: Dễ hiểu, không làm thay đổi phân phối dữ liệu.
- Nhược điểm: Có thể tạo ra một biến mới không thể quan sát.
- Ví dụ: Trong một tập dữ liệu về tình trạng hôn nhân, có thể tạo ra một danh mục mới "Unknown" để đại diện cho các giá trị thiếu.

c. Cụ thể trong file data có những dữ liệu thiếu :

city_development_index có: 2.5 %
gender có: 23.531 %
enrolled_university có: 2.015 %
education_level có: 2.401 %
major_discipline có: 14.683 %
experience có: 0.339 %
company_size có: 30.995 %
company_type có: 32.049 %
training_hours có: 3.998 %

d. Các kĩ thuật được sử dụng để xử lý dữ liệu thiếu cho file dữ liệu trên:

- Mean or Median Imputation
- Frequent Category Imputation
- Missing Category Imputation

e. Code xử lý dữ liệu thiếu:

- **Dữ liệu đầu vào:** dữ liệu gồm 19158 mẫu, 13 thuộc tính

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	NaN	NaN	36.0	1
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	47.0	0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	83.0	0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	0.0	NaN	Pvt Ltd	52.0	1
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Startup	8.0	0
5	21651	city_176	0.764	NaN	Has relevent experience	Part time course	Graduate	STEM	11.0	NaN	NaN	24.0	1
6	28806	city_160	0.920	Male	Has relevent experience	no_enrollment	High School	NaN	5.0	50-99	Funded Startup	24.0	0
7	402	city_46	0.762	Male	Has relevent experience	no_enrollment	Graduate	STEM	13.0	<10	Pvt Ltd	18.0	1
8	27107	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	7.0	50-99	Pvt Ltd	46.0	1
9	699	city_103	0.920	NaN	Has relevent experience	no_enrollment	Graduate	STEM	17.0	10000+	Pvt Ltd	123.0	0
10	29452	city_21	0.624	NaN	No relevent experience	Full time course	High School	NaN	2.0	NaN	NaN	32.0	1
11	23853	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	5.0	5000-9999	Pvt Ltd	108.0	0
12	25619	city_61	0.913	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	1000-4999	Pvt Ltd	23.0	0
13	5826	city_21	0.624	Male	No relevent experience	NaN	NaN	NaN	2.0	NaN	NaN	24.0	0
14	8722	city_21	0.624	NaN	No relevent experience	Full time course	High School	NaN	5.0	NaN	NaN	26.0	0
15	6588	city_114	0.926	Male	Has relevent experience	no_enrollment	Graduate	STEM	16.0	Oct-49	Pvt Ltd	18.0	0
16	4167	city_103	0.920	NaN	Has relevent experience	no_enrollment	Graduate	STEM	1.0	50-99	Pvt Ltd	106.0	0
17	5764	city_21	0.624	NaN	Has relevent experience	no_enrollment	Graduate	STEM	2.0	5000-9999	Pvt Ltd	7.0	0
18	2156	city_21	0.624	NaN	Has relevent experience	no_enrollment	Graduate	STEM	7.0	10000+	Pvt Ltd	23.0	1
19	11399	city_13	0.827	Female	Has relevent experience	no_enrollment	Graduate	Arts	4.0	NaN	NaN	132.0	1

Dữ liệu minh họa

- **Dữ liệu đầu ra sau khi xử lý dữ liệu thiếu:**

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_ty
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	missing	missi
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt L
2	11561	city_21	0.624	Female	No relevent experience	Full time course	Graduate	STEM	5.0	missing	missi
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Start
5	21651	city_176	0.764	Female	Has relevent experience	Part time course	Graduate	STEM	11.0	missing	missi
6	28806	city_160	0.920	Male	Has relevent experience	no_enrollment	High School	No Major	5.0	50-99	Funded Start
7	402	city_46	0.762	Male	Has relevent experience	no_enrollment	Graduate	STEM	13.0	<10	Pvt L
8	27107	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	7.0	50-99	Pvt L
9	699	city_103	0.920	Female	Has relevent experience	no_enrollment	Graduate	STEM	17.0	10000+	Pvt L
10	29452	city_21	0.624	Female	No relevent experience	Full time course	High School	No Major	2.0	missing	missi

- **Thống kê tỷ lệ dữ liệu thiếu ở các thuộc tính**

Code[0]:

```
df.isna().head(10)
```

- Hàm isna() thực hiện kiểm tra các giá trị trong tập dữ liệu df, các phần tử rỗng gán True, các phần tử khác rỗng gán False

Đầu ra:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target
0	False	False	False	False	False	False	False	False	False	True	True	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	True	False	False	False	False	False	True	True	False	False
3	False	False	False	True	False	True	False	False	False	True	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
5	False	False	False	True	False	False	False	False	False	True	True	False	False
6	False	False	False	False	False	False	False	True	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	True	False	False	False	False	False	False	False	False	False

Code[1]:

```
print("Tỷ lệ phần trăm dữ liệu thiếu của các thuộc tính:")
print(df.isna().mean())
```

- Hàm isna().mean(): thực hiện tính tỷ lệ trung bình dữ liệu thiếu bằng cách lấy số lượng phân tử True của mỗi cột chia cho số lượng mẫu dữ liệu

Đầu ra:

```
Tỷ lệ phần trăm dữ liệu thiếu của các thuộc tính:
enrollee_id             0.000000
city                    0.000000
city_development_index  0.025003
gender                  0.235306
relevent_experience     0.000000
enrolled_university     0.020148
education_level         0.024011
major_discipline        0.146832
experience              0.003393
company_size            0.309949
company_type            0.320493
training_hours          0.039983
target                  0.000000
dtype: float64
```

***) Xét thuộc tính city_development_index (2.5003%)**

- **Thử nghiệm điền giá trị thiếu bằng giá trị mean (giá trị trung bình)**

Code [2]:

```
mean = df['city_development_index'].mean()
mean
```

- Hàm mean(): lấy ra giá trị mean (giá trị trung bình) của cột city_development_index
- Sau đó thực hiện in ra giá trị trung bình

Đầu ra:

```
0.8289509074361583
```

Code[3]

```
df['city_mean'] = df['city_development_index'].fillna(mean)
df.head()
```

- Thực hiện tạo ra cột mới ‘city_mean’, gán lại giá trị của cột ‘city_development_index’ sau khi điền các giá trị mean vào các giá trị thiếu
- Thực hiện in ra dữ liệu sau khi taoj thêm cột ‘city_mean’

Đầu ra:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target	city_mean
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	NaN	NaN	36.0	1	0.920
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	47.0	0	0.776
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	83.0	0	0.624
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	0.0	NaN	Pvt Ltd	52.0	1	0.789
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Startup	8.0	0	0.767

- **Thử nghiệm điền giá trị thiếu bằng giá trị median (giá trị trung vị)**

Code[4]:

```
median = df['city_development_index'].median()
median
```

- Hàm median(): lấy ra giá trị median (giá trị trung vị) của cột city_development_index
- Sau đó thực hiện in ra giá trị trung vị

Đầu ra:

```
0.903
```

Code[5]:

```
df['city_median'] = df['city_development_index'].fillna(median)
df.head()
```

- Thực hiện tạo ra cột mới 'city_median', gán lại giá trị của cột 'city_development_index' sau khi điền các giá trị median vào các giá trị thiếu
- Thực hiện in ra dữ liệu sau khi tạo thêm cột 'city_median'

Đầu ra:

enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target	city_mean	city_median
8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	NaN	NaN	36.0	1	0.920	0.920
29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	47.0	0	0.776	0.776
11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	83.0	0	0.624	0.624
33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	0.0	NaN	Pvt Ltd	52.0	1	0.789	0.789
666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Startup	8.0	0	0.767	0.767

- **Vẽ biểu đồ so sánh việc điền dữ liệu bằng mean và median**

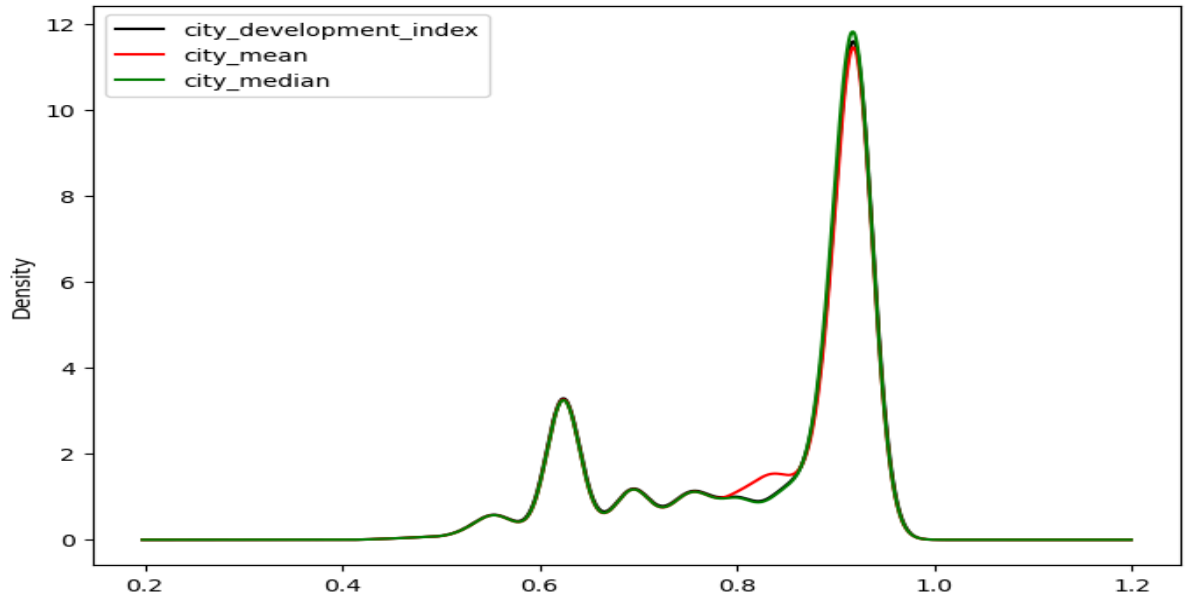
Code [6]

```
plt.rcParams["figure.figsize"] = [8,6]

fig = plt.figure()
ax = fig.add_subplot(111)
df['city_development_index'] .plot(kind='kde', ax=ax, color='black')
df['city_mean'] .plot(kind='kde', ax=ax, color='red')
df['city_median'] .plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

- Vẽ biểu đồ phân phối KDE để so sánh phân phối của dữ liệu gốc và dữ liệu sau khi điền giá trị mean và median

Đầu ra:



- Nhận xét: Dễ thấy phân phối của dữ liệu khi điền giá trị thiếu bằng median ít bị thay đổi hơn so với việc điền dữ liệu thiếu bằng giá trị mean, lên chúng ta sẽ sử dụng kỹ thuật điền giá trị mean cho những dữ liệu thiếu này

Code[7]

```
df['city_development_index'].fillna(median, inplace = True)
```

- Thực hiện điền những giá trị thiếu của cột city_development_index bằng giá trị median, tham số inplace = True để đảm bảo dữ liệu được thay đổi vĩnh viễn

Code[8]

```
df['city_development_index'].isna().mean()
```

- Thực hiện in ra giá trị thiếu sau khi thực hiện xử lý giá trị thiếu bằng median

Đầu ra:

```
0.0
```

Code[9]

```
df.drop(columns=['city_median', 'city_mean'], axis=1, inplace=True)
df.head()
```

- Thực hiện xóa đi cột ‘city_mean’, ‘city_median’ đã dùng để thực nghiệm trước đó
- In ra dữ liệu sau khi xóa cột

Đầu ra:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	NaN	NaN	36.0	1
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	47.0	0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	83.0	0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	0.0	NaN	Pvt Ltd	52.0	1
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Startup	8.0	0

***) Xét thuộc tính experience (0.339%)**

Code[10]

```
df.dropna(subset=['experience'], inplace=True)
```

- Đối với dữ liệu thiếu rất ít, ta thực hiện xóa bỏ những dòng chứa dữ liệu thiếu

Code[11]

```
df.shape()
```

- In ra kích thước dữ liệu sau khi xử lý

Đầu ra:

```
(19093, 13)
```

***) Xét thuộc tính training_hours (3.998%)**

- **Thử nghiệm điền giá trị thiếu bằng giá trị mean (giá trị trung bình)**

Code[11]:

```
mean = df['training_hours'].mean()
mean
```

- Hàm mean(): lấy ra giá trị mean (giá trị trung bình) của cột training_hours (3.998%)
- Sau đó thực hiện in ra giá trị trung bình

Đầu ra:

```
65.1657937806874
```

Code[12]:

```
df['mean_hours'] = df['training_hours'].fillna(mean)
```

- Thực hiện tạo ra cột mới ‘mean_hours’, gán lại giá trị của cột ‘traning_hours’ sau khi điền các giá trị mean vào các giá trị thiếu
- Thực hiện in ra dữ liệu sau khi tạo thêm cột ‘city_mean’

Code[13]:

```
median = df['training_hours'].median()
median
```

- Hàm median(): lấy ra giá trị median (giá trị trung vị) của cột training_hours (3.998%)
- Sau đó thực hiện in ra giá trị trung vị

Code[14]

```
df['median_hours'] = df['training_hours'].fillna(median)
```

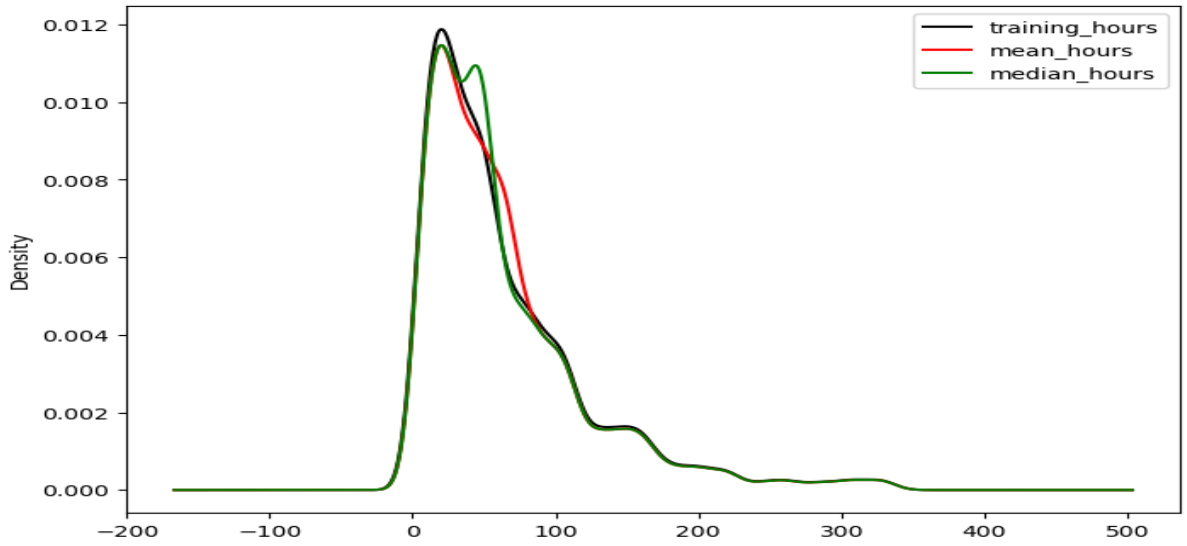
- Thực hiện tạo ra cột mới ‘median_hours’, gán lại giá trị của cột ‘traning_hours’ sau khi điền các giá trị mean vào các giá trị thiếu
- Thực hiện in ra dữ liệu sau khi tạo thêm cột ‘city_median’
- **Vẽ biểu đồ so sánh việc điền dữ liệu bằng mean và median**

Code[15]

```
plt.rcParams["figure.figsize"] = [8,6]

fig = plt.figure()
ax = fig.add_subplot(111)
df['training_hours'] .plot(kind='kde', ax=ax, color='black')
df['mean_hours'] .plot(kind='kde', ax=ax, color='red')
df['median_hours'] .plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

- Vẽ biểu đồ phân phối KDE để so sánh phân phối của dữ liệu gốc và dữ liệu sau khi điền giá trị mean và median



Nhận xét: Dễ thấy phân phối của dữ liệu khi điền giá trị thiếu bằng median hay mean đều bị thay đổi phân phối tương đối, ta đi thử nghiệm thêm các kỹ thuật khác

Code[16]

```
df.drop(columns=['mean_hours', 'median_hours'], inplace=True)
```

- Thực hiện xóa đi cột ‘city_mean’, ‘city_median’ đã dùng để thực nghiệm.

***) Dùng kỹ thuật điền giá trị đuôi để xử lý giá trị thiếu**

Code [17]

```
eod_value = df['training_hours'].mean() + 3*df['training_hours'].std()
df['training_hours'].fillna(eod_value, inplace=True)
```

- Giá trị đuôi được tính bằng cách lấy giá trị mean + 3*std
- Thực hiến thay thế các giá trị thiếu của cột training_hours bằng giá trị đuôi

Code[18]

```
df['training_hours'].isna().mean()
```

- In ra tỷ lệ thiếu của cột training_hours

B. Mã hóa dữ liệu

- Mã hóa dữ liệu là quá trình chuyển đổi dữ liệu từ một dạng hoặc biểu diễn ban đầu sang một dạng hoặc biểu diễn khác, thích hợp để sử dụng cho một mục đích cụ thể.
- Mục đích của việc mã hóa dữ liệu có thể bao gồm làm cho dữ liệu phù hợp với các mô hình học máy, giảm kích thước dữ liệu, bảo mật dữ liệu, hoặc chỉ đơn giản là để làm cho dữ liệu dễ dàng hiểu và xử lý hơn.
- **Các kĩ thuật thực hiện mã hóa là:**
    - One Hot Encoding
    - Label Encoding
    - Frequency Encoding
    - Ordinal Encoding
    - Mean Encoding

a) One Hot Encoding

Phương pháp:

Phương pháp: One Hot Encoding chuyển đổi các biến phân loại thành các biến nhị phân, trong đó mỗi giá trị của biến phân loại sẽ trở thành một biến nhị phân mới.

Ưu điểm:

Dễ sử dụng: One Hot Encoding dễ triển khai và sử dụng, đặc biệt là với các thư viện phổ biến như pandas hoặc scikit-learn.

Bảo toàn thông tin: Kỹ thuật này giữ lại tất cả các giá trị trong biến phân loại và không làm mất thông tin.

Phù hợp với các mô hình học máy: Đa số các mô hình học máy yêu cầu dữ liệu đầu vào là dạng số, nên One Hot Encoding là một lựa chọn phù hợp để biểu diễn biến phân loại.

Nhược điểm:

Tăng kích thước dữ liệu: One Hot Encoding tạo ra một số lượng biến nhị phân mới bằng số lượng giá trị khác nhau của biến phân loại, làm tăng kích thước của dữ liệu.

Không phù hợp với biến có nhiều giá trị: Khi một biến phân loại có quá nhiều giá trị, One Hot Encoding có thể tạo ra quá nhiều biến nhị phân, làm cho việc xử lý và huấn luyện mô hình trở nên khó khăn và tốn kém.

Ví dụ:

Giả sử chúng ta có một bộ dữ liệu về các loại hoa với biến phân loại là 'loại hoa' có các giá trị 'hoa đỏ', 'hoa xanh', và 'hoa vàng'. Bằng cách sử dụng One Hot Encoding, chúng ta có thể chuyển đổi biến phân loại này thành ba biến nhị phân 'is_red', 'is_green', và 'is_yellow'. Với mỗi quan sát, một trong ba biến nhị phân sẽ có giá trị 1 và các biến khác sẽ có giá trị 0, phản ánh loại hoa của quan sát đó.

b) Label Encoding

Phương pháp:

- Phương pháp: Label Encoding chuyển đổi các biến phân loại thành các giá trị số nguyên duy nhất, trong đó mỗi giá trị của biến phân loại sẽ được ánh xạ sang một giá trị số.

Ưu điểm:

Dễ sử dụng: Label Encoding dễ triển khai và sử dụng, đặc biệt là với các thư viện như scikit-learn.

Giữ lại thông tin về thứ tự: Kỹ thuật này giữ lại thông tin về thứ tự của các giá trị trong biến phân loại, nếu có.

Giảm kích thước dữ liệu: So với One Hot Encoding, Label Encoding không tạo ra thêm biến mới, giúp giảm kích thước của dữ liệu.

Nhược điểm:

Không phù hợp với các biến không có thứ tự: Label Encoding không phù hợp với các biến phân loại không có thứ tự hoặc không có mối quan hệ giữa các giá trị.

Nguy cơ hiểu nhầm: Mô hình có thể hiểu nhầm rằng các giá trị được mã hóa bằng Label Encoding có thứ tự hoặc có mối quan hệ với nhau, điều này có thể dẫn đến kết quả không chính xác.

Ví dụ:

Giả sử chúng ta có một bộ dữ liệu về các loại xe gồm 'xe hơi', 'xe máy', 'xe đạp'. Bằng cách sử dụng Label Encoding, chúng ta có thể chuyển đổi biến phân loại này thành các giá trị số nguyên 0, 1, 2. Cụ thể, 'xe hơi' sẽ được mã hóa thành 0, 'xe máy' sẽ được mã hóa thành 1, và 'xe đạp' sẽ được mã hóa thành 2. Điều này giúp mô hình có thể hiểu được loại xe một cách dễ dàng hơn.

c) Frequency Encoding

Phương pháp:

- Đếm tần suất xuất hiện của mỗi giá trị trong biến phân loại.
- Thay thế mỗi giá trị bằng tần suất xuất hiện của nó**.**

Ưu điểm:

Đơn giản: Kỹ thuật này dễ triển khai và không yêu cầu nhiều xử lý so với các phương pháp mã hóa khác.

Bảo toàn thông tin: Giữ lại thông tin về tần suất xuất hiện của các giá trị phân loại, không làm mất thông tin.

Nhược điểm:

Không phù hợp với các giá trị có tần suất xuất hiện cân bằng: Nếu các giá trị có tần suất xuất hiện gần như bằng nhau, Frequency Encoding có thể không cung cấp nhiều thông tin hữu ích cho mô hình học máy.

Dễ bị ảnh hưởng bởi nhiễu: Nếu có nhiễu trong dữ liệu, tần suất xuất hiện của các giá trị có thể bị méo mó và ảnh hưởng đến hiệu suất của mô hình.

Ví dụ:

Giả sử chúng ta có một bộ dữ liệu về các quốc gia và mỗi quốc gia được phân loại bằng tên của nó. Bằng cách sử dụng Frequency Encoding, chúng ta có thể chuyển đổi biến phân loại này thành các giá trị tương ứng với tần suất xuất hiện của từng quốc gia trong tập dữ liệu. Ví dụ, nếu quốc gia "Mỹ" xuất hiện 100 lần trong tập dữ liệu, thì nó sẽ được mã hóa thành giá trị 100. Điều này giúp mô hình có thể hiểu được phân phối của các quốc gia trong dữ liệu và có thể cải thiện hiệu suất dự đoán.

d) Ordinal Encoding

Phương pháp:

- Xác định thứ tự tương đối của các giá trị phân loại.
- Gán giá trị số nguyên cho mỗi giá trị phân loại dựa trên thứ tự tương đối đó.

Ưu điểm:

Giữ lại thông tin về thứ tự: Giữ lại thông tin về thứ tự của các giá trị phân loại, nếu có.

Giảm kích thước dữ liệu: So với One Hot Encoding, Ordinal Encoding không tạo ra thêm biến mới, giúp giảm kích thước của dữ liệu.

Nhược điểm:

Không phù hợp với các biến không có thứ tự: Ordinal Encoding không phù hợp với các biến phân loại không có thứ tự hoặc không có mối quan hệ giữa các giá trị.

Nguy cơ hiểu nhầm: Mô hình có thể hiểu nhầm rằng các giá trị được mã hóa bằng Ordinal Encoding có thứ tự hoặc có mối quan hệ với nhau, điều này có thể dẫn đến kết quả không chính xác.

Ví dụ:

Giả sử chúng ta có một bộ dữ liệu về mức độ hạnh phúc của người dân trong một quốc gia, được phân loại thành "Không hạnh phúc", "Bình thường", "Hạnh phúc", "Rất hạnh phúc". Bằng cách sử dụng Ordinal Encoding, chúng ta có thể chuyển đổi biến phân loại này thành các giá trị số nguyên 1, 2, 3, 4 tương ứng với thứ tự từ "Không hạnh phúc" đến "Rất hạnh phúc". Điều này giúp mô hình có thể hiểu được mức độ hạnh phúc của người dân một cách dễ dàng hơn.

e) Mean Encoding

Phương pháp:

Tính giá trị trung bình của mục tiêu (target) cho mỗi giá trị phân loại.

Thay thế mỗi giá trị phân loại bằng giá trị trung bình tương ứng.

Ưu điểm:

Bảo toàn thông tin: Mean Encoding giữ lại thông tin về mối quan hệ giữa biến phân loại và mục tiêu.

Giảm kích thước dữ liệu: So với One Hot Encoding, Mean Encoding không tạo ra thêm biến mới, giúp giảm kích thước của dữ liệu.

Nhược điểm:

Nguy cơ overfitting: Nếu số lượng quan sát trong mỗi nhóm của biến phân loại là nhỏ, Mean Encoding có thể dẫn đến overfitting. Không xử lý được giá trị mới: Mean Encoding không xử lý được giá trị phân loại mới không xuất hiện trong tập huấn luyện, do đó cần cẩn thận khi áp dụng trên dữ liệu mới.

Ví dụ:

Giả sử chúng ta có một bộ dữ liệu về các loại sản phẩm và mục tiêu là xác định xem một sản phẩm có được mua hay không. Bằng cách sử dụng Mean Encoding, chúng ta có thể chuyển đổi biến phân loại này thành các giá trị số, thể hiện xác suất mua của từng loại sản phẩm. Ví dụ, nếu sản phẩm A có xác suất mua là 0.8, sản phẩm B có xác suất mua là 0.6, thì chúng ta sẽ mã hóa sản phẩm A thành giá trị 0.8 và sản phẩm B thành giá trị 0.6. Điều này giúp mô hình có thể hiểu được mức độ quan trọng của từng loại sản phẩm đối với quá trình mua hàng.

- **Cụ thể trong file dữ liệu trên các cột mã hóa :**

city, gender, relevent_experience, enrolled_university, education_level, major_discipline, company_size, company_type.

- **Áp dụng các kĩ thuật:**

+ **Frequency Encoding** cho biến "city" do có số lương lớn giá trị, thay thế bằng tần xuất xuất hiện

+ Dùng **Ordinal Encoding** cho các cột gender, relevent_experience, enrolled_university, education_level, major_discipline, company_size, company_type. Ordinal Encoding: Sử dụng khi biến phân loại có thứ tự và các giá trị có mối quan hệ với nhau.

+ Dùng **One-Hot Encoding** cho cột major_discipline, gender. One-Hot Encoding: Sử dụng khi biến phân loại không có mối quan hệ thứ tự giữa các giá trị

Code[0]

```
print("Những cột có dữ liệu cần được mã hóa là: ")
for columns in df.columns:
        if df[columns].dtype == 'object':
                print("   +)",columns, ":", df[columns].unique())
```

In ra các giá trị phân biệt của thuộc tính có kiểu dữ liệu object

Đầu ra:

```
Những cột có dữ liệu cần được mã hóa là:
   +) city : ['city_103' 'city_40' 'city_21' 'city_115' 'city_162' 'city_176'
 'city_160' 'city_46' 'city_61' 'city_114' 'city_13' 'city_159' 'city_102'
 'city_67' 'city_100' 'city_16' 'city_71' 'city_104' 'city_64' 'city_101'
 'city_83' 'city_105' 'city_73' 'city_75' 'city_41' 'city_11' 'city_93'
 'city_90' 'city_36' 'city_20' 'city_57' 'city_152' 'city_19' 'city_65'
 'city_74' 'city_173' 'city_136' 'city_98' 'city_97' 'city_50' 'city_138'
 'city_82' 'city_157' 'city_89' 'city_150' 'city_70' 'city_175' 'city_94'
 'city_28' 'city_59' 'city_165' 'city_145' 'city_142' 'city_26' 'city_12'
 'city_37' 'city_43' 'city_116' 'city_23' 'city_99' 'city_149' 'city_10'
 'city_45' 'city_80' 'city_128' 'city_158' 'city_123' 'city_7' 'city_72'
 'city_106' 'city_143' 'city_78' 'city_109' 'city_24' 'city_134' 'city_48'
 'city_144' 'city_91' 'city_146' 'city_133' 'city_126' 'city_118' 'city_9'
 'city_167' 'city_27' 'city_84' 'city_54' 'city_39' 'city_79' 'city_76'
 'city_77' 'city_81' 'city_131' 'city_44' 'city_117' 'city_155' 'city_33'
 'city_141' 'city_127' 'city_62' 'city_53' 'city_25' 'city_2' 'city_69'
 'city_120' 'city_111' 'city_30' 'city_1' 'city_140' 'city_179' 'city_55'
 'city_14' 'city_42' 'city_107' 'city_18' 'city_139' 'city_180' 'city_166'
 'city_121' 'city_129' 'city_8' 'city_31' 'city_171']
   +) gender : ['Male' 'missing' 'Female' 'Other']
   +) relevent_experience : ['Has relevent experience' 'No relevent experience']
   +) enrolled_university : ['no_enrollment' 'Full time course' 'missing' 'Part time course']
   +) education_level : ['Graduate' 'Masters' 'High School' 'missing' 'Phd' 'Primary School']
   +) major_discipline : ['STEM' 'Business Degree' 'missing' 'Arts' 'Humanities' 'No Major' 'Other']
   +) company_size : ['missing' '50-99' '<10' '10000+' '5000-9999' '1000-4999' 'Oct-49'
 '100-500' '500-999']
   +) company_type : ['missing' 'Pvt Ltd' 'Funded Startup' 'Early Stage Startup' 'Other'
 'Public Sector' 'NGO']
```

Code[1]

```
df['company_size'] = df['company_size'].replace('Oct-49', '10-49')
df['company_size'].unique()
```

Dùng replace để thay thế giá trị Oct-49 thành giá trị 10-49 do lỗi nhập liệu trong cột company_size.

Đầu ra:

```
array(['missing', '50-99', '<10', '10000+', '5000-9999', '1000-4999',
       '10-49', '100-500', '500-999'], dtype=object)
```

Code[2]

```
df['city'].value_counts()
```

Đếm tần suất xuất hiện của các giá trị phân biệt trong cột city.

Đầu ra:

```
city
city_103    4337
city_21     2685
city_16     1530
city_114    1334
city_160     845
           ...
city_129       3
city_111       3
city_121       3
city_140       1
city_171       1
Name: count, Length: 123, dtype: int64
```

Code[3]

```
frequency_count = (df['city'].value_counts() / len(df['city'])
).to_dict()
print(frequency_count)
```

Thực hiện kĩ thuật mã hóa Frequency bằng cách thay thế giá trị hiện tại bằng xác suất xuất hiện của giá trị đó trong cột city.
Lấy tuần suất xuất hiện của các giá trị phân biệt chia cho số giá trị của cột city, sau đó chuyển thành từ điển với 1 cặp Key- Value trong đó Key là giá trị, Value là xác suất xuất hiện trong cột city
Đầu ra:

```
{'city_103': 0.227151311999162, 'city_21': 0.14062745508825225, 'city_16': 0.08013408055308228, 'city_114': 0.06986853820772011, 'city_160': 0.04425705756036244
```

Code[4]

```
df['city'] = df['city'].map(frequency_count)
```

Hàm map thực hiện ánh xạ thay thế các giá trị của cột city bằng các Key-Value tương ứng.

Code[5]

```
df['city'].dtype
```

In ra kiểu dữ liệu của cột city sau khi mã hóa.
Đầu ra:

```
dtype('float64')
```

Code[6]

```
df.head()
```

In ra 5 dòng đầu tiên của dữ liệu sau khi mã hóa cột city

Đầu ra:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_ty
0	8949	0.227151	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	20.0	missing	missi
1	29725	0.003562	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt L
2	11561	0.140627	0.624	missing	No relevent experience	Full time course	Graduate	STEM	5.0	missing	missi
3	33241	0.002828	0.789	missing	No relevent experience	missing	Graduate	Business Degree	0.0	missing	Pvt L
4	666	0.006704	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	20.0	50-99	Funded Start

*) Thực hiện kĩ thuật mã hóa Ordinal Encoding cho các cột gender, relevent_experience, enrolled_university, education_level, major_discipline, company_size, company_type.

Code[7]

```
label_enrolled_university = {'no_enrollment':0, 'Full time course':2,
'Part time course':1, 'missing':3}
label_education_level = {'Graduate':2, 'Masters':3, 'High School':1,
'Phd':4, 'Primary School':0, 'missing':5}
label_company_size = {'50-99':2, '<10':0, '10000+':7, '5000-9999':6,
'1000-4999':5, '10-49':1, '100-500':3, '500-999':4, 'missing':8}
label_company_type = {'Pvt Ltd':2, 'Funded Startup':3, 'Early Stage
Startup':4, 'Other':5, 'Public Sector':0,'NGO':1, 'missing':6}
label_relevent_experience = {'Has relevent experience':1, 'No relevent
experience':0}
```

Thực hiện tạo các từ điển từng giá trị phân biệt của các cột cần mã hóa.

Code[8]

```
df['enrolled_university'] =
df['enrolled_university'].map(label_enrolled_university)
df['education_level'] =
df['education_level'].map(label_education_level)
df['company_size'] = df['company_size'].map(label_company_size)
df['company_type'] = df['company_type'].map(label_company_type)
df['relevent_experience'] =
df['relevent_experience'].map(label_relevent_experience)
```

Hàm map thực hiện ánh xạ thay thế các giá trị của cột city bằng các Key-Value tương ứng cho các thuộc tính trên.

Code[9]

```
df.head()
```

In ra 5 dòng đầu tiên của dữ liệu sau khi mã hóa.

Đầu ra:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_t
0	8949	0.227151	0.920	Male	1	0	2	STEM	20.0	8	
1	29725	0.003562	0.776	Male	0	0	2	STEM	15.0	2	
2	11561	0.140627	0.624	missing	0	2	2	STEM	5.0	8	
3	33241	0.002828	0.789	missing	0	3	2	Business Degree	0.0	8	
4	666	0.006704	0.767	Male	1	0	3	STEM	20.0	2	

*) **Dùng One-Hot Encoding cho cột major_discipline, gender**

Code[10]

```
data_major = pd.get_dummies(df['major_discipline'], prefix='major')
data_gender = pd.get_dummies(df['gender'], prefix='gender')
```

Thực hiện tạo ra các cột tương ứng với số giá trị phân biệt của thuộc, tên cột = tên cột gốc + giá trị phân biệt. Tương với giá trị ở cột gốc thì cột mã hóa tương ứng nhận giá trị True, ngược lại nhận giá trị False.

Code[11]

```
data_gender.head()
```

In ra 5 dòng đầu tiên của dữ liệu của cột gender sau khi mã hóa.

Đầu ra:

	gender_Female	gender_Male	gender_Other	gender_missing
0	False	True	False	False
1	False	True	False	False
2	False	False	False	True
3	False	False	False	True
4	False	True	False	False

Code[12]

```
data_major.head()
```

In ra 5 dòng đầu tiên của dữ liệu của cột major sau khi mã hóa.

Đầu ra:

	major_Arts	major_Business Degree	major_Humanities	major_No Major	major_Other	major_STEM	major_missing
0	False	False	False	False	False	True	False
1	False	False	False	False	False	True	False
2	False	False	False	False	False	True	False
3	False	True	False	False	False	False	False
4	False	False	False	False	False	True	False

Code[13]

```
df = pd.concat([df, data_major, data_gender], axis=1)
df = df.drop(columns=['major_discipline', 'gender'], axis=1)
```

Thực hiện nối các cột mã hóa với dữ liệu gốc, xóa đi cột gender và cột major.

Code[14]

```
df.replace([True, False], [1, 0], inplace=True)
```

Thay thế giá trị True thành 1, False thành 0.

Code[15]

```
df.head()
```

In ra 5 dòng đầu tiên của dữ liệu sau khi mã hóa.

	enrollee_id	city	city_development_index	relevent_experience	enrolled_university	education_level	experience	company_size	company_type	training_hours
0	8949	0.227151	0.920	1	0	2	20.0	8	6	36.0
1	29725	0.003562	0.776	0	0	2	15.0	2	2	47.0
2	11561	0.140627	0.624	0	2	2	5.0	8	6	83.0
3	33241	0.002828	0.789	0	3	2	0.0	8	2	52.0
4	666	0.006704	0.767	1	0	3	20.0	2	3	8.0

5 rows × 22 columns

Code[16]

```
df.shape
```

In ra kích thước của dữ liệu sau khi mã hóa.
Đầu ra:

```
(19093, 22)
```

Code[17]

```
df.dtypes
```

In ra kiểu dữ liệu của các thuộc tính trong df.
Đầu ra:

```
enrollee_id                int64
city                     float64
city_development_index   float64
relevent_experience        int64
enrolled_university        int64
education_level            int64
experience               float64
company_size               int64
company_type               int64
training_hours           float64
target                     int64
major_Arts                 int64
major_Business Degree      int64
major_Humanities           int64
major_No Major             int64
major_Other                int64
major_STEM                 int64
major_missing              int64
gender_Female              int64
gender_Male                int64
gender_Other               int64
gender_missing             int64
dtype: object
```

Code[18]

```
pd.set_option("display.max_columns", None)
```

Thực hiện cài đặt hiển đầy đủ các cột dữ liệu của df.

C.Xử lý dữ liệu ngoại lệ

- Giá trị ngoại lệ là những giá trị nằm quá xa so với các giá trị còn lại trong cột. Với dạng số, dữ liệu ngoại lệ có thể là một giá trị phi thực tế như số tuổi âm, hoặc một giá trị khác xa với phần còn lại của các giá trị trong trường đó. Với dạng hạng mục, dữ liệu ngoại lệ có thể là một giá trị phi thực tế như một hạng mục nằm ngoài những khả năng có thể xảy ra như một địa danh không có trên bản đồ.
- Cách tốt nhất để phát hiện dữ liệu ngoại lệ là trực quan hóa dữ liệu cột đó lên biểu đồ Boxplot. Box plot vừa giúp xác định xem dữ liệu có điểm ngoại lệ không, vừa giúp tìm ra ngưỡng lớn nhất và nhỏ nhất để làm điểm cắt.

- Vì các giá trị ngoại lệ có ảnh hưởng lớn đến chất lượng mô hình machine learning.
- Đôi khi, dữ liệu ngoại lệ không phải là lỗi mà là kết quả của một hiện tượng thực sự và có thể mang lại thông tin có giá trị. Việc phân tích chúng có thể cung cấp hiểu biết sâu sắc về dữ liệu.

- **Có bốn kỹ thuật chính để xử lý các ngoại lệ:**
  Outlier Trimming
  Outlier Capping Using IQR
  Outlier Capping Using Quantiles

a, Outlier Trimming

Phương pháp: Outlier trimming là quá trình loại bỏ hoặc điều chỉnh các giá trị ngoại lệ từ tập dữ liệu để tạo ra một tập dữ liệu mới mà không còn sự hiện diện của các giá trị ngoại lệ.

Ưu điểm:

- Cải thiện tính nhất quán và độ chính xác: Loại bỏ các giá trị ngoại lệ giúp cải thiện tính nhất quán của dữ liệu và tăng độ chính xác của phân tích hoặc mô hình hóa dữ liệu.
- Giảm thiểu ảnh hưởng của nhiễu: Outlier trimming giúp giảm thiểu ảnh hưởng của các giá trị ngoại lệ lên kết quả phân tích, giảm thiểu các đội ngũ giá trị gây nhiễu.

Nhược điểm:

- Mất mát thông tin: Việc loại bỏ hoặc điều chỉnh các giá trị ngoại lệ có thể dẫn đến mất mát thông tin, đặc biệt là nếu các giá trị này có ý nghĩa quan trọng trong dữ liệu.
- Sự chủ quan trong lựa chọn: Quá trình quyết định loại bỏ hoặc điều chỉnh outliers có thể dễ dàng bị ảnh hưởng bởi sự chủ quan của người phân tích.

Ví dụ: Giả sử bạn đang phân tích dữ liệu về thời gian phản hồi của một trang web. Một số lượt truy cập có thể gặp phải lỗi kỹ thuật hoặc các vấn đề mạng, dẫn đến thời gian phản hồi rất lớn, không phản ánh thực tế. Trong trường hợp này, outlier trimming có thể được sử dụng để loại bỏ các giá trị thời gian phản hồi cực đoan, như thời gian phản hồi rất cao, để tạo ra một tập dữ liệu mới với thời gian phản hồi ổn định hơn và độ chính xác cao hơn.

b, Phương pháp Capping Outlier Sử Dụng IQR:

Phương pháp: Trong phương pháp này, các outliers được xác định dựa trên khoảng cách so với phạm vi của Interquartile Range (IQR). Cụ thể, outliers được xác định như là các giá trị nằm ngoài khoảng từ Q1 - 1.5 * IQR đến Q3 + 1.5 * IQR.

Ưu điểm:

- Dễ dàng triển khai và hiểu.
- Linh hoạt trong việc điều chỉnh ngưỡng capping bằng cách thay đổi hệ số 1.5.

Nhược điểm:

- Không phản ánh chính xác phân phối của dữ liệu nếu phân phối không phải là chuẩn.
- Không hiệu quả đối với dữ liệu có phân phối đuôi dài hoặc nhiều outliers.

Ví dụ: Giả sử bạn đang phân tích dữ liệu về số lần mua hàng trực tuyến trong một tháng từ một nhóm khách hàng. Nếu một số khách hàng có thói quen mua hàng trực tuyến rất thường xuyên, gây ra các giá trị ngoại lai cao, thì capping outlier sử dụng IQR có thể giúp loại bỏ những giá trị này để tạo ra một phân phối mua hàng trực tuyến ổn định hơn.

c, Phương pháp Capping Outlier Sử Dụng Quantiles:

Phương pháp: Trong phương pháp này, quantiles của phân phối dữ liệu được sử dụng để xác định ngưỡng cho việc capping outliers. Cụ thể, các quantiles như 5th percentile và 95th percentile thường được sử dụng để xác định giới hạn cho các outliers.

Ưu điểm:

- Có thể tùy chỉnh ngưỡng capping dựa trên phân phối cụ thể của dữ liệu.
- Phản ánh chính xác hơn phân phối của dữ liệu, đặc biệt là trong trường hợp dữ liệu không tuân theo phân phối chuẩn.

Nhược điểm:

- Đòi hỏi sự chủ quan trong việc xác định các ngưỡng quantiles phù hợp.
- Có thể làm mất mát thông tin nếu capping quá nhiều giá trị, đặc biệt là đối với dữ liệu có nhiều outliers.

Ví dụ: Nếu bạn đang nghiên cứu về số lượng lượt xem của các video trên một trang web chia sẻ video, việc sử dụng capping outlier dựa trên quantiles có thể giúp loại bỏ các giá trị lượt xem cực đoan, như lượt xem rất cao, để tạo ra một phân phối lượt xem ổn định hơn cho các video.

- Cụ thể trong file dữ liệu trên những thuộc tính có dữ liệu ngoại lệ:
  city_development_index, experience, training_hours
- Các kĩ thuật được sử dụng:
  Outlier Trimming
  Outlier Capping Using IQR

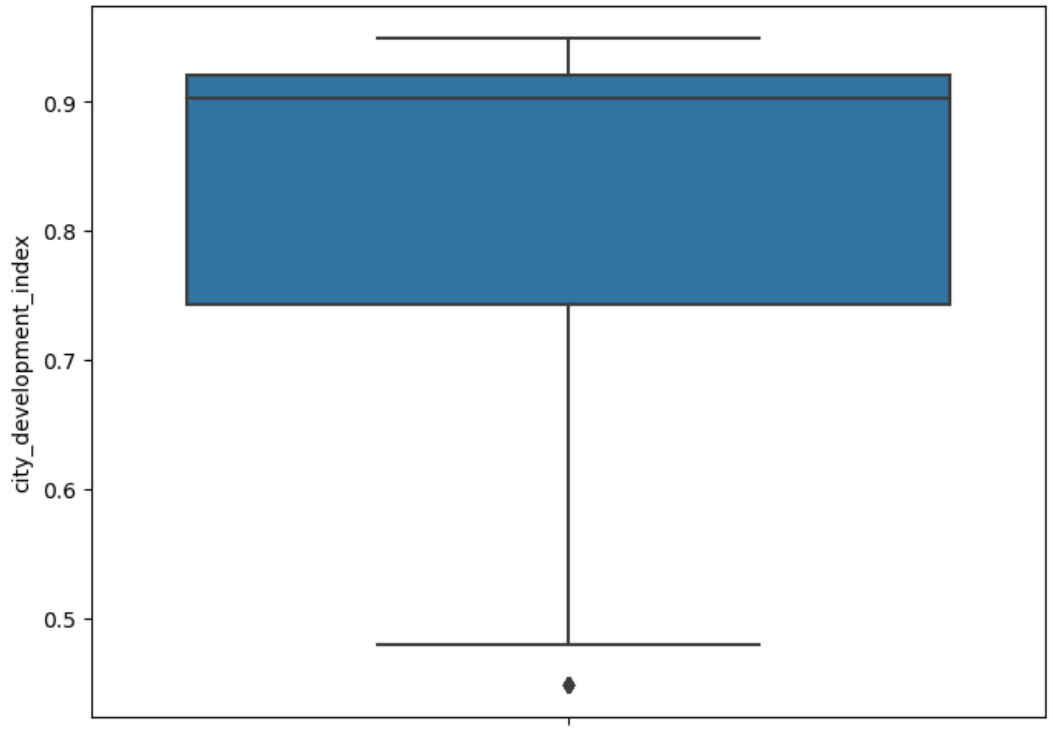
***) Thực hiện xử lí ngoại lên thuộc tính city_development_index**

Code[0]

```
sns.boxplot( y='city_development_index', data=df)
```

- Vẽ biểu đồ boxplot cho cột ‘city_development_index’ để quan sát Min, Q1, Q2, Q3, Max và các điểm dữ liệu ngoại lệ

Đầu ra:



Code[1]

```
IQR = df["city_development_index"].quantile(0.75) -
df["city_development_index"].quantile(0.25)

lower_city_limit = df["city_development_index"].quantile(0.25) - (IQR * 1.5)

upper_city_limit = df["city_development_index"].quantile(0.75) + (IQR * 1.5)

print(lower_city_limit)
print(upper_city_limit)
```

- IQR được tính bằng Q3(75%) - Q1(25%)
- Giới hạn dưới được tính bằng Q1 - 1,5*IQR
- Giới hạn trên được tính bằng Q3 + 1,5*IQR
- Thực hiện in giới hạn trên và giới hạn dưới

Đầu ra:

```
0.4774999999999999
1.1855000000000002
```

Code[3]

```
outlier_city_development_index= np.where(df["city_development_index"] > upper_city_limit, True,
        np.where(df["city_development_index"] < lower_city_limit, True, False))
```

Tạo ra biến mới outlier_city_development_index, np.where để tạo ra 1 mảng mới với giá trị cột city_development_index, Nếu giá trị lớn hơn giới hạn trên hoặc nhỏ hơn giới hạn dưới thì được gán bằng True, các giá trị còn lại gán bằng False.

Code[4]

```
df = df.loc[~outlier_city_development_index]
```

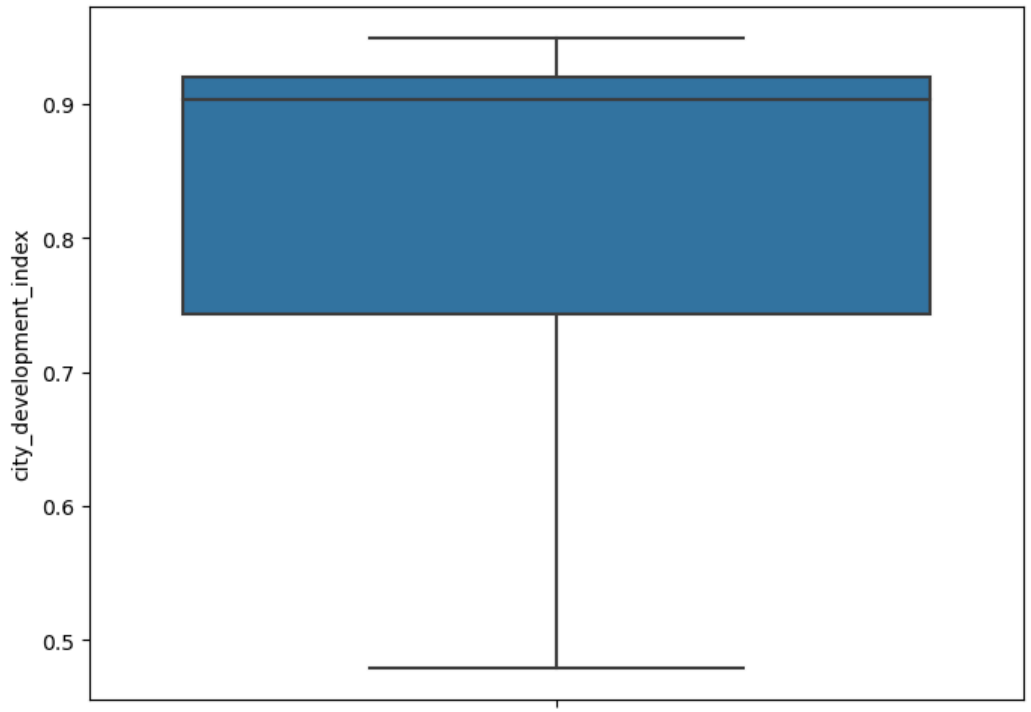
~outlier_city_development_index thực hiện phủ định, False thành True, True thành False. Sau đó lọc trong dữ liệu df với điều kiện trên, những dòng tương ứng với giá trị True sẽ được lấy ra, nghĩa là các dòng chứ giá trị ngoại lệ đã được loại bỏ.

Code[5]

```
sns.boxplot( y='city_development_index', data=df)
```

- In ra biểu đồ Boxplot sau khi xử lí ngoại lệ, các điểm ngoại lệ đã bị loại bỏ.

Đầu ra:



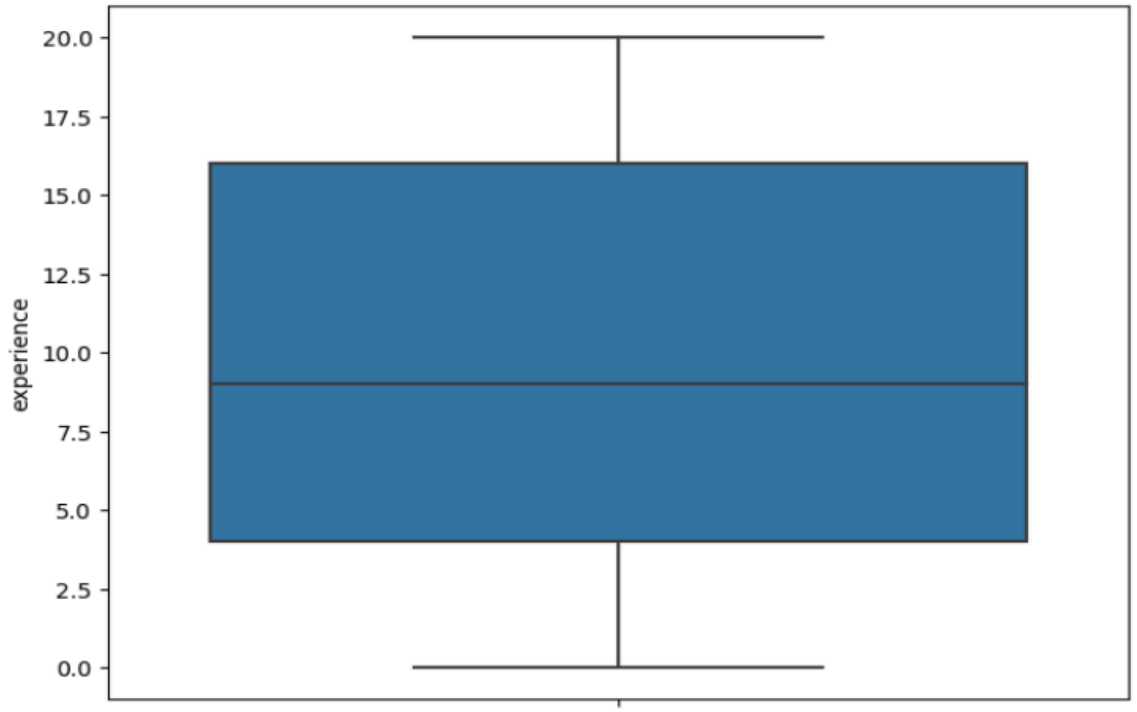
***) Thực hiện xử lí ngoại lên thuộc tính experience**

Code[6]

```
sns.boxplot( y='experience', data=df)
```

Thực hiện vẽ biểu đồ Boxplot cho thuộc tính experience để quan sát Min, Q1, Q2, Q3, Max và các điểm dữ liệu ngoại lệ

Đầu ra:



Quan sát biểu đồ không phát hiện dữ liệu ngoại lệ.

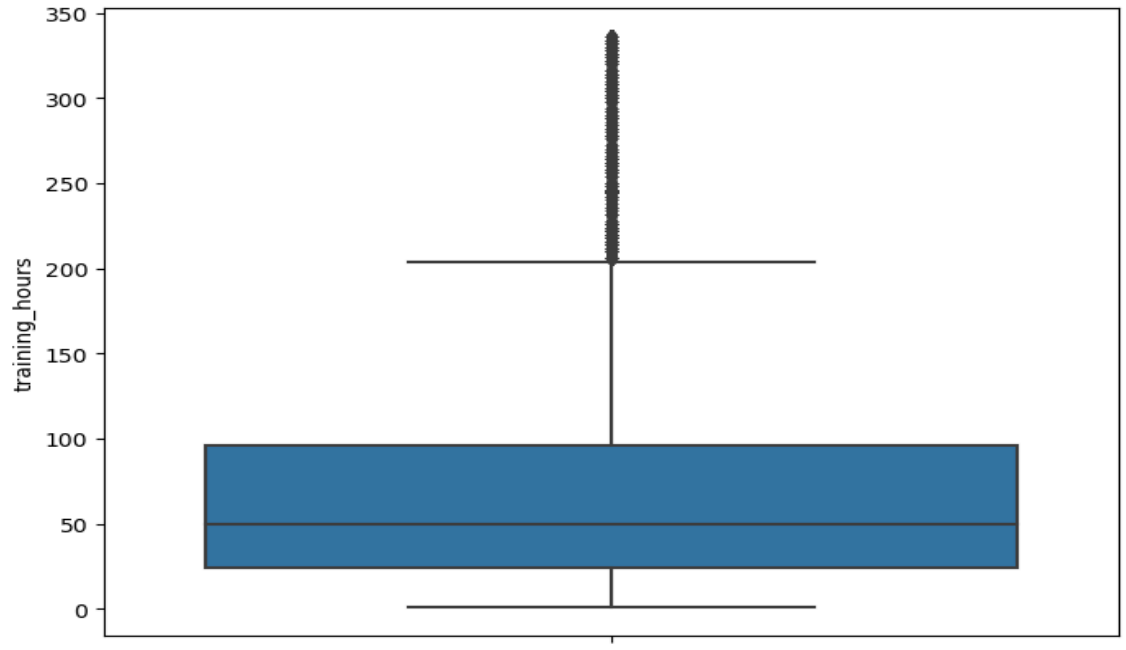
***) Thực hiện xử lí ngoại lên thuộc tính training_hours**

Code[7]

```
sns.boxplot( y='training_hours', data=df)
```

Thực hiện vẽ biểu đồ Boxplot cho thuộc tính training_hours để quan sát Min, Q1, Q2, Q3, Max và các điểm dữ liệu ngoại lệ.

Đầu ra:



Code[8]

```
IQR = df["training_hours"].quantile(0.75) - df["training_hours"].quantile(0.25)
# Giới hạn dưới được tính bằng cách lấy Q1 - 1.5*IQR
lower_hours_limit = df["training_hours"].quantile(0.25) - (IQR * 1.5)
# Giới hạn trên được tính bằng cách lấy Q3 + 1.5*IQR
upper_hours_limit = df["training_hours"].quantile(0.75) + (IQR * 1.5)

print(lower_hours_limit)
print(upper_hours_limit)
```

- IQR được tính bằng Q3(75%) - Q1(25%)
- Giới hạn dưới được tính bằng Q1 - 1,5*IQR
- Giới hạn trên được tính bằng Q3 + 1,5*IQR
- Thực hiện in giới hạn trên và giới hạn dưới

Code[9]

```
df["training_hours"] = np.where(df['training_hours'] > upper_hours_limit,
upper_hours_limit,
          np.where(df["training_hours"] < lower_hours_limit, lower_hours_limit,
df["training_hours"]))
```

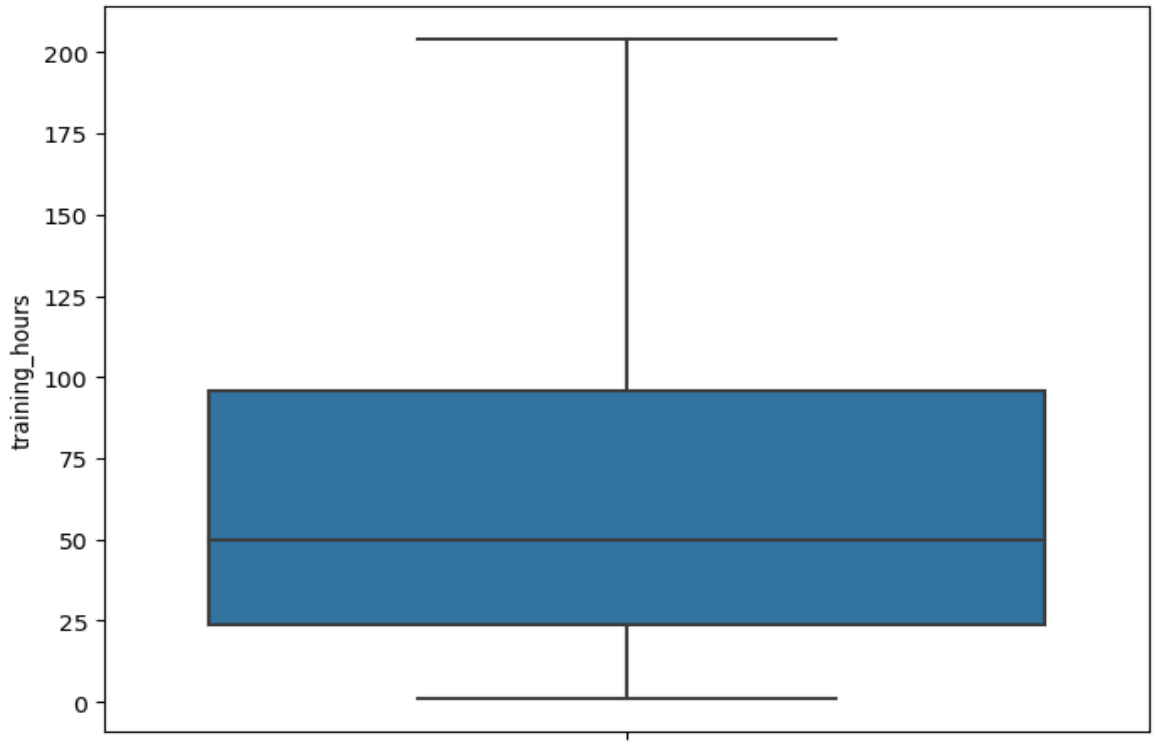
Thực hiện thay thế các giá trị ngoại lệ, những giá trị lớn hơn giới hạn trên gán bằng giá trị giới hạn trên, những điểm nhỏ hơn giới hạn dưới gán bằng giới hạn dưới, các điểm còn lại bằng chính nó

Code[10]

```
sns.boxplot( y='training_hours', data=df)
```

Biểu đồ Boxplot sau khi xử lí dữ liệu.

Đầu ra:



D. Chuẩn hóa dữ liệu

- Chuẩn hóa dữ liệu là quá trình biến đổi các giá trị trong tập dữ liệu thành một phạm vi chuẩn hoặc tiêu chuẩn nhất định.
- Chuẩn hóa là làm cho các đặc trưng trong tập dữ liệu có cùng phạm vi giá trị hoặc cùng trọng số, giúp cải thiện hiệu suất của các mô hình học máy.
- Chuẩn hóa dữ liệu là quá trình biến đổi dữ liệu ban đầu thành dạng có thể dễ dàng so sánh và xử lý mà không làm mất đi thông tin quan trọng.
- Một số kỹ thuật phổ biến để chuẩn hóa dữ liệu:
    - Min/Max Scaling
    - Mean Normalization
    - Maximum Absolute Scaling
    - Median and Quantile Scaling
    - Vector Unit Length Scaling

a, **Min/Max Scaling**

- **Khái niệm và định nghĩa:** Chuẩn hóa min-max chuyển đổi các giá trị dữ liệu vào một phạm vi cố định, thường là từ 0 đến 1 hoặc -1 đến 1.
- **Phương pháp:** Công thức chuẩn hóa min-max:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \times (\max_{\text{new}} - \min_{\text{new}}) + \min_{\text{new}}$$

Trong đó:

X là giá trị gốc của đặc trưng.

$X_{\min}$ là giá trị tối thiểu của đặc trưng trong tập dữ liệu.

$X_{\max}$ là giá trị tối đa của đặc trưng trong tập dữ liệu.

$\min_{\text{new}}$ là giá trị tối thiểu của phạm vi mong muốn (thường là 0).

$\max_{\text{new}}$ là giá trị tối đa của phạm vi mong muốn (thường là 1).

Ưu điểm:

- Dễ dàng thực hiện và hiểu.
- Bảo toàn phân phối của dữ liệu gốc.
- Đồng nhất hóa các đặc trưng và giảm thiểu ảnh hưởng của các giá trị ngoại lai.

Nhược điểm:

- Min-max scaling dễ bị ảnh hưởng bởi các giá trị ngoại lai.
- Không thích hợp cho các phân phối không đồng đều hoặc có nhiều giá trị ngoại lai.

Ví dụ:

Giả sử chúng ta có một tập dữ liệu về chiều cao và cân nặng của các người. Khi áp dụng min-max scaling, chúng ta có thể chia tỷ lệ cả hai đặc trưng sao cho chiều cao và cân nặng đều nằm trong phạm vi từ 0 đến 1, giúp cho việc so sánh giữa các đặc trưng trở nên dễ dàng hơn.

b, Mean Normalization

- **Mean normalization** là một kỹ thuật chuẩn hóa dữ liệu trong đó giá trị của mỗi điểm dữ liệu được điều chỉnh sao cho giá trị trung bình của các điểm dữ liệu là 0.
- Công thức để thực hiện mean normalization trên một đặc trưng X là:

$$\text{Xnorm} = \frac{X - mean(X)}{max(X) - min(X)}$$

Trong đó:

Xnorm là giá trị được chuẩn hóa của đặc trưng X.

$mean(X)$ là giá trị trung bình của đặc trưng X.

$max(X)$ và $min(X)$ là giá trị lớn nhất và nhỏ nhất của đặc trưng X tương ứng.

Ưu điểm:

- Đồng nhất hóa phạm vi: Mean normalization giúp đưa các đặc trưng về cùng một phạm vi giá trị, thường là trong khoảng [0, 1]. Điều này giúp cho các đặc trưng có cùng phạm vi, giúp mô hình dễ dàng học được các quan hệ giữa chúng một cách hiệu quả hơn.
- Giảm ảnh hưởng của giá trị ngoại lai: Mean normalization giúp giảm thiểu ảnh hưởng của các giá trị ngoại lai bằng cách đưa các giá trị gần về trung bình, làm cho chúng không còn gây ra ảnh hưởng lớn đến quá trình huấn luyện mô hình.
- Dễ hiểu và triển khai: Mean normalization dễ thực hiện và dễ hiểu. Không cần phải tính toán nhiều thông số như trong chuẩn hóa Z-score, và nó có thể được triển khai một cách nhanh chóng.

Nhược điểm:

- Không bảo toàn phân phối: Trong mean normalization, phân phối của dữ liệu không được bảo toàn như trong chuẩn hóa Z-score. Điều này có thể gây ra mất mát thông tin nếu quan tâm đến cấu trúc phân phối của dữ liệu.
- Khả năng biến đổi ảnh hưởng của giá trị ngoại lai: Trong một số trường hợp, các giá trị ngoại lai có thể làm biến đổi lớn đến phạm vi giá trị sau khi chuẩn hóa. Điều này có thể làm mất mát thông tin và làm giảm hiệu suất của mô hình.
- Không phù hợp cho các mô hình nhất quán tuyến tính: Các mô hình nhất quán tuyến tính như Linear Regression hoặc Logistic Regression có thể yêu cầu các đặc trưng có cùng độ lệch chuẩn. Trong trường hợp này, chuẩn hóa Z-score thường là lựa chọn phù hợp hơn.

c, Maximum Absolute Scaling

Khái niệm:

Maximum Absolute Scaling là một kỹ thuật tiền xử lý dữ liệu được sử dụng để chia tỷ lệ các giá trị của các đặc trưng dựa trên giá trị tuyệt đối lớn nhất của mỗi đặc trưng.

Phương pháp:

Công thức Maximum Absolute Scaling được áp dụng như sau:

$$X_{\text{scaled}} = \frac{X}{X_{\text{max\_abs}}}$$

Trong đó:

X là giá trị gốc của đặc trưng.

$X_{\text{max\_abs}}$ là giá trị tuyệt đối lớn nhất của đặc trưng trong tập dữ liệu.

Ưu điểm:

- Dễ dàng thực hiện.
- Bảo toàn phân phối của dữ liệu gốc.
- Không cần biết giá trị tối thiểu và tối đa của dữ liệu trước.

Nhược điểm:

- Các giá trị ngoại lai có thể ảnh hưởng đến kết quả chia tỷ lệ.
- Không thích hợp cho các phân phối có nhiều giá trị ngoại lai không phải là giá trị tuyệt đối lớn nhất.

Ví dụ:

Giả sử bạn có một tập dữ liệu về giá cổ phiếu, trong đó giá cổ phiếu có thể biến đổi từ -100 đến 100. Bằng cách sử dụng Maximum Absolute Scaling, mỗi giá trị trong tập dữ liệu sẽ được chia tỷ lệ dựa trên giá trị tuyệt đối lớn nhất là 100, như vậy giá trị lớn nhất sẽ trở thành 1 và giá trị nhỏ nhất sẽ trở thành -1.

d, Median and Quantile Scaling

Khái niệm:

Median and Quantile Scaling là một phương pháp trong tiền xử lý dữ liệu được sử dụng để chia tỷ lệ các giá trị của các đặc trưng trong tập dữ liệu dựa trên giá trị trung vị (median) hoặc phân vị (quantile) của chúng.

Phương pháp:

Cách thực hiện Median and Quantile Scaling thường bao gồm:

Tính toán giá trị trung vị hoặc các phân vị của các đặc trưng trong tập dữ liệu.

Chia tỷ lệ các giá trị của các đặc trưng dựa trên giá trị trung vị hoặc các phân vị đã tính toán.

Ưu điểm:

- Bảo toàn phân phối của dữ liệu gốc.
- Đồng nhất hóa các đặc trưng và giảm thiểu ảnh hưởng của các giá trị ngoại lai.
- Phù hợp với các mô hình yêu cầu dữ liệu có cùng phân phối.

Nhược điểm:

- Cần tính toán giá trị trung vị hoặc các phân vị của dữ liệu, đặc biệt là với các tập dữ liệu lớn.
- Không hiệu quả khi có nhiều giá trị ngoại lai.

Ví dụ:

Giả sử bạn có một tập dữ liệu gồm các đặc trưng biểu diễn điểm số của các học sinh trong một lớp học. Một trong những đặc điểm của dữ liệu là sự biến động lớn giữa các khoảng điểm. Bằng cách sử dụng Median and Quantile Scaling, bạn có thể chia tỷ lệ điểm số của các học sinh dựa trên phân vị của dữ liệu, giúp đồng nhất hóa và chuẩn hóa dữ liệu cho quá trình phân tích và huấn luyện mô hình.

e, Vector Unit Length Scaling

- Vector Unit Length Scaling là một kỹ thuật chuẩn hóa dữ liệu trong đó mỗi vectơ đặc trưng được chia cho độ dài của nó, biến đổi vectơ thành một vectơ có độ dài bằng 1. Kỹ thuật này cũng được gọi là chuẩn hóa L2 (L2 normalization) hoặc chuẩn hóa tổng độ dài.

Công thức để thực hiện chuẩn hóa L2 trên một vectơ X là:

$$\text{Xnorm} = \frac{X}{\|X\|2}$$

Trong đó,$\|X\|2$ là độ dài của vectơ X, được tính bằng căn bậc hai của tổng bình phương các phần tử trong vectơ.

Ưu điểm:

- Đồng nhất hóa độ dài của các vectơ
- Giảm ảnh hưởng của tỷ lệ giữa các đặc trưng.
- Giữ nguyên hướng của vectơ

Nhược điểm:

- Mất thông tin về độ lớn ban đầu của các vectơ
- Không giải quyết được vấn đề của giá trị ngoại lai
- Không phù hợp cho tất cả các loại dữ liệu

Ví dụ: Giả sử chúng ta có một vectơ đặc trưng X=[3,4], vectơ này có độ dài $\|X\|2 = \sqrt{3^2 + 4^2} = 5$. Để chuẩn hóa vectơ này, chúng ta chia mỗi phần tử của vectơ cho độ dài của nó:

$$\text{Xnorm} = \left[\frac{3}{5}, \frac{4}{5}\right]$$

Kết quả là một vectơ có độ dài bằng 1: $\text{Xnorm} = [0.6, 0.8]$

- **Trong dữ liệu trên chọn phương pháp:**
  Chuẩn hóa Min-Max

Bởi vì: Dễ dàng thực hiện và hiểu.
Bảo toàn phân phối của dữ liệu gốc.
Đồng nhất hóa các đặc trưng và giảm thiểu ảnh hưởng của các giá trị ngoại lai.

- **Dữ liệu đầu vào:**

	enrollee_id	city	city_development_index	relevent_experience	enrolled_university	education_level	experience	company_size	company_type	training_hours	target
0	8949	0.232734	0.920	1	0	2	20.0	8	6	36.0	1
1	29725	0.003540	0.776	0	0	2	15.0	2	2	47.0	0
2	11561	0.138247	0.624	0	2	2	5.0	8	6	83.0	0
4	666	0.006442	0.767	1	0	3	20.0	2	3	8.0	0
5	21651	0.002089	0.764	1	1	2	11.0	8	6	24.0	1
6	28806	0.045038	0.920	1	0	1	5.0	2	3	24.0	0
7	402	0.006790	0.762	1	0	2	13.0	0	2	18.0	1
8	27107	0.232734	0.920	1	0	2	7.0	2	2	46.0	1
9	699	0.232734	0.920	1	0	2	17.0	7	2	123.0	0
10	29452	0.138247	0.624	0	2	1	2.0	8	6	32.0	1

- **Dữ liệu đầu ra:**

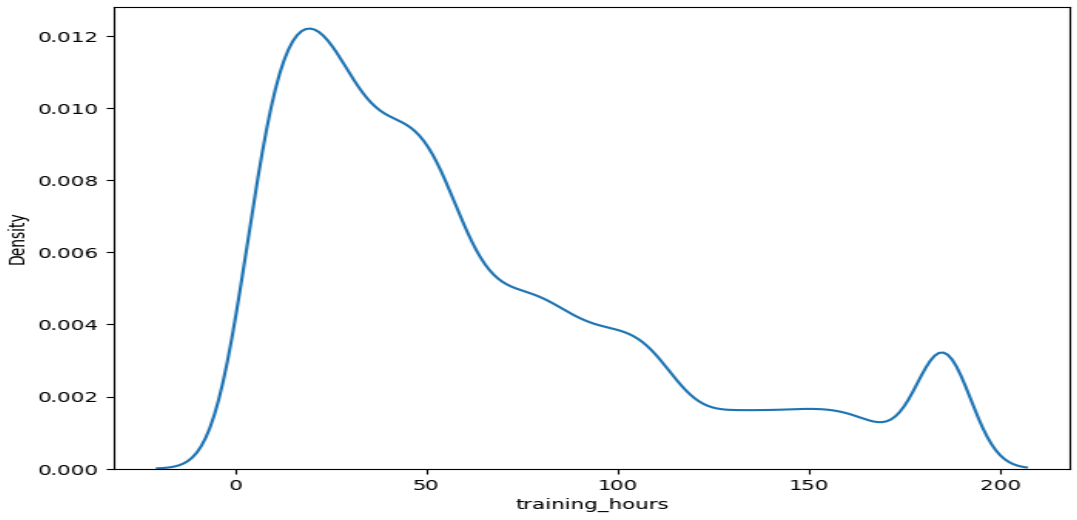
	enrollee_id	city	city_development_index	relevent_experience	enrolled_university	education_level	experience	company_size	company_type	training_hours	target
0	0.268073	1.000000	0.938298	1.0	0.0	0.50	0.202020	1.000	1.000000	0.189702	1.0
1	0.890500	0.014721	0.631915	0.0	0.0	0.50	0.151515	0.250	0.333333	0.249322	0.0
2	0.346326	0.593812	0.308511	0.0	1.0	0.50	0.050505	1.000	1.000000	0.444444	0.0
3	0.019923	0.027196	0.612766	1.0	0.0	0.75	0.202020	0.250	0.500000	0.037940	0.0
4	0.648611	0.008483	0.606383	1.0	0.5	0.50	0.111111	1.000	1.000000	0.124661	1.0
5	0.862968	0.193114	0.938298	1.0	0.0	0.25	0.050505	0.250	0.500000	0.124661	0.0
6	0.012014	0.028693	0.602128	1.0	0.0	0.50	0.131313	0.000	0.333333	0.092141	1.0
7	0.812067	1.000000	0.938298	1.0	0.0	0.50	0.070707	0.250	0.333333	0.243902	1.0
8	0.020911	1.000000	0.938298	1.0	0.0	0.50	0.171717	0.875	0.333333	0.661247	0.0
9	0.882321	0.593812	0.308511	0.0	1.0	0.25	0.020202	1.000	1.000000	0.168022	1.0

Code[0]

```
sns.kdeplot(df['training_hours'])
```

Vẽ minh họa biểu đồ phân phối Kde cho thuộc tính training_hours trước khi chuẩn hóa.

Đầu ra:



Code[1]

```
for columns in df.columns:
    # Thực hiện đi tính max và min của cột đó
    max = df[columns].max()
    min = df[columns].min()
    # Áp dụng công thức trên
    df[columns] = (df[columns] - min)/(max - min)
```

Sử dụng kĩ thuật chuẩn hóa Min/Max cho dữ liệu df về [0,1] theo công thức:

$$X_{new} = \frac{Xold - Min}{Max - Min}$$

Thực hiện duyệt từng thuộc tính dữ liệu df:

- Với mỗi thuộc tính tìm giá trị Min, Max.
- Sau đó thực hiện công thức chuẩn hóa và gán lại giá trị cho thuộc tính đấy.

Code[2]

```
df.head(10)
```

In ra 10 dòng dữ liệu đầu tiên sau khi chuẩn hóa, tất cả giá trị đã được đưa về khoảng giá trị [0,1].

Đầu ra:

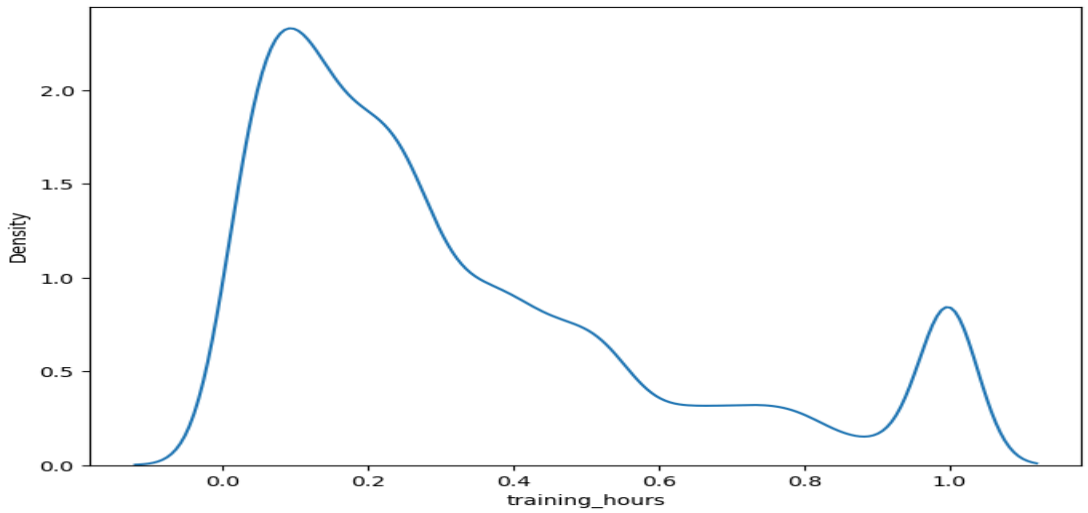
	enrollee_id	city	city_development_index	relevent_experience	enrolled_university	education_level	experience	company_size	company_type	training_hours	target
0	0.268073	1.000000	0.938298	1.0	0.000000	0.4	1.00	1.000	1.000000	0.172414	1.0
1	0.890500	0.015452	0.631915	0.0	0.000000	0.4	0.75	0.250	0.333333	0.226601	0.0
2	0.346326	0.619004	0.308511	0.0	0.666667	0.4	0.25	1.000	1.000000	0.403941	0.0
3	0.995836	0.012223	0.659574	0.0	1.000000	0.4	0.00	1.000	0.333333	0.251232	1.0
4	0.019923	0.029290	0.612766	1.0	0.000000	0.6	1.00	0.250	0.500000	0.034483	0.0
5	0.648611	0.005304	0.606383	1.0	0.333333	0.4	0.55	1.000	1.000000	0.113300	1.0
6	0.862968	0.194649	0.938298	1.0	0.000000	0.2	0.25	0.250	0.500000	0.113300	0.0
7	0.012014	0.029290	0.602128	1.0	0.000000	0.4	0.65	0.000	0.333333	0.083744	1.0
8	0.812067	1.000000	0.938298	1.0	0.000000	0.4	0.35	0.250	0.333333	0.221675	1.0
9	0.020911	1.000000	0.938298	1.0	0.000000	0.4	0.85	0.875	0.333333	0.600985	0.0

Code[3]

```
sns.kdeplot(df['training_hours'])
```

Vẽ minh họa biểu đồ Kde cho thuộc tính training_hours sau khi chuẩn hóa dữ liệu

Đầu ra:



Code[4]

```
df.shape
```

Kích thước dữ liệu df sau khi áp dụng các kĩ thuật tiền xử lí dữ liệu.
Đầu ra:

```
(19076, 22)
```

Code[5]

```
df.info()
```

Thông tin cơ bản của các thuộc tính trong dữ liệu sau khi áp dụng các kĩ thuật tiền xử lí dữ liệu.
Đầu ra:

```
<class 'pandas.core.frame.DataFrame'>
Index: 19076 entries, 0 to 19157
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19076 non-null  float64
 1   city                    19076 non-null  float64
 2   city_development_index  19076 non-null  float64
 3   relevent_experience     19076 non-null  float64
 4   enrolled_university     19076 non-null  float64
 5   education_level         19076 non-null  float64
 6   experience              19076 non-null  float64
 7   company_size            19076 non-null  float64
 8   company_type            19076 non-null  float64
 9   training_hours          19076 non-null  float64
 10  target                  19076 non-null  float64
 11  major_Arts              19076 non-null  float64
 12  major_Business Degree   19076 non-null  float64
 13  major_Humanities        19076 non-null  float64
 14  major_No Major          19076 non-null  float64
 15  major_Other             19076 non-null  float64
 16  major_STEM              19076 non-null  float64
 17  major_missing           19076 non-null  float64
 18  gender_Female           19076 non-null  float64
 19  gender_Male             19076 non-null  float64
 20  gender_Other            19076 non-null  float64
 21  gender_missing          19076 non-null  float64
dtypes: float64(22)
memory usage: 3.3 MB
```

E. ỨNG DỤNG MÔ HÌNH HỌC MÁY ĐỐI VỚI DỮ LIỆU

I, Giới thiệu về thuật toán phân lớp Support Vector Machine

a) Định nghĩa

- Bài toán tối ưu trong Support Vector Machine (SVM) là tìm đường phân chia sao cho margin là lớn nhất (Maximum Margin Classifier)
- Ý tưởng của SVM: Margin của một siêu phẳng được định nghĩa là khoảng cách từ các điểm gần nhất của lớp đó tới mặt phân chia. Margin của hai lớp phải bằng nhau và lớn nhất có thể
- Cần tìm một đường phân chia sao cho:
  - Khoảng cách từ điểm gần nhất của mỗi lớp tới đường phân chia là như nhau (margin, lề).
  - Margin này phải là cực đại

b) Đầu vào

Giả sử rằng các cặp dữ liệu trong tập huấn luyện là (x1, y1),(x2, y2), . . . ,(xN , yN) với vector xi ∈ R d thể hiện đầu vào của một điểm dữ liệu và yi là nhãn của điểm dữ liệu đó, d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi yi = 1 hoặc yi = −1 giống như trong PLA

c) Đầu ra

Mặt phẳng phân chia giữa hai classes là

$$\mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$$

d) Định nghĩa hàm mất mát

-Xét trong không gian hai chiều

Các điểm vuông xanh thuộc class 1

Các điểm tròn đỏ thuộc class -1

Mặt phẳng phân chia giữa hai classes là

wTx +b= $w1x1 + w2x2 + b=0$

Class 1 nằm về phía dương, class -1 nằm về phía âm của mặt phân chia

-Với cặp dữ liệu (xn ,yn) bất kỳ, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

-Vì *n* luôn cùng dấu với phía của *n* , nên *n* cùng dấu với *T n* , tử số luôn là 1 số không âm.

-Với mặt phân chia như trên, margin được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó:

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

-Bài toán tối ưu trong SVM chính là bài toán tìm và b sao cho margin này đạt giá trị lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}$$

-Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng ta có cách để đưa nó về bài toán đơn giản hơn

-Nhận xét quan trọng nhất là nếu ta thay vector hệ số bởi và bởi trong đó là một hằng số dương thì mặt phân chia không thay đổi, tức khoảng cách từ từng điểm đến mặt phân chia không đổi, tức margin không đổi

-Vậy bài toán tối ưu

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\}$$

-Có thể đưa về bài toán tối ưu có ràng buộc sau đây

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||_2}$$

$$\text{subject to:} \quad y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1, \forall n = 1, 2, \ldots, N$$

-Bằng phép lấy nghịch đảo, bài toán trên chuyển thành

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

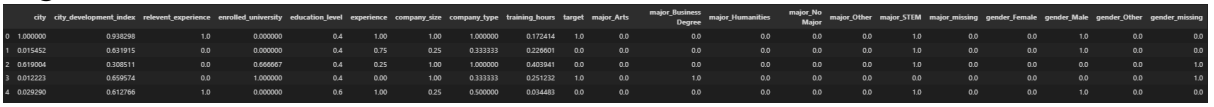
$$\text{thoả mãn: } 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N$$

e) Phương pháp

- Sau khi đã tìm được mặt phân cách wTx + b = 0
- Nhãn của bất kỳ một điểm được xác định bằng

class(x) = sgn(wTx + b)

- **Dùng thuật toán Super Vector Meachin đối với dữ liệu**
  - **Dữ liệu đầu vào** gồm 19076 bản ghi và 21 cột thuộc tính trong đó cột target là cột nhãn.



	city	city_development_index	relevent_experience	enrolled_university	education_level	experience	company_size	company_type	training_hours	target	major_Arts	major_Business Degree	major_Humanities	major_No Major	major_Other	major_STEM	major_missing	gender_Female	gender_Male	gender_Other	gender_missing
0	1.000000	0.938298	1.0	0.000000	0.4	1.00	1.00	1.000000	0.172414	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
1	0.015452	0.631915	0.0	0.000000	0.4	0.75	0.25	0.333333	0.229601	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
2	0.619004	0.308511	0.0	0.666667	0.4	0.25	1.00	1.000000	0.403941	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
3	0.012223	0.659574	0.0	1.000000	0.4	0.00	1.00	0.333333	0.251232	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.029290	0.612766	1.0	0.000000	0.8	1.00	0.25	0.500000	0.034483	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

  - **Dữ liệu đầu ra**: Giá trị dữ đoán với hàm kích hoạt là hàm dấu sgn(), đường tuyến tính phân chia 2 lớp dữ liệu.

Code[0]

```
from sklearn.svm import SVC
```

Đoạn mã này đang nhằm mục đích nhập lớp Support Vector Classification (SVC) từ thư viện scikit-learn

Code[1]

```
model = SVC(kernel='rbf', C=10)
model.fit(X_train.values, y_train.values)
y_pred = model.predict(X_test)
```

Tạo một mô hình Support Vector Classification (SVC) với hạt nhân là 'rbf' và tham số C là 11.

- kernel: Đây là loại hạt nhân được sử dụng trong thuật toán SVM. Trong trường hợp này, kernel='rbf' chỉ định hạt nhân Radial Basis Function (RBF). Hạt nhân RBF thường được sử dụng để chuyển đổi dữ liệu vào không gian chiều cao vô hạn để có thể phân chia các lớp không gian phức tạp.
- C: Tham số này là tham số điều chuẩn (regularization parameter) trong SVM. Nó quy định mức độ quan trọng của việc phân loại đúng các điểm dữ liệu huấn luyện so với việc tối thiểu hóa khoảng cách giữa các biên phân loại. Khi C lớn, mô hình SVM sẽ cố gắng tối đa hóa độ chính xác trên dữ liệu huấn luyện, có thể dẫn đến việc overfitting.

Huấn luyện mô hình trên dữ liệu huấn luyện

```
model.fit(X_train.values, y_train.values)
```

Dự đoán nhãn của dữ liệu kiểm tra

```
y_pred = model.predict(X_test)
```

```
# Kết quả chạy mô hình
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Ti le du doan dung theo SVM:")
print('accuracy_score:', accuracy_score(y_test,y_pred))
print('precision_score:', precision_score(y_test,y_pred))
print('recall_score:', recall_score(y_test,y_pred))
print('f1_score:', f1_score(y_test,y_pred))
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Dòng mã này import các hàm từ module sklearn.metrics để tính toán và đánh giá hiệu suất của mô hình phân loại gồm accuracy_score, precision_score, recall_score, f1_score

```
print("Ti le du doan dung theo SVM:")
print('accuracy_score:', accuracy_score(y_test,y_pred))
print('precision_score:', precision_score(y_test,y_pred))
print('recall_score:', recall_score(y_test,y_pred))
print('f1_score:', f1_score(y_test,y_pred))
```

- print("Ti le du doan dung theo SVM:"): Dòng này in ra một thông báo cho người dùng để nói rằng các chỉ số hiệu suất được tính dựa trên SVM.
- accuracy_score(y_test, y_pred): Hàm accuracy_score tính tỷ lệ dự đoán chính xác của mô hình bằng cách so sánh các nhãn dự đoán (y_pred) với nhãn thực tế (y_test). Nó là tỷ lệ giữa số lượng dự đoán đúng và tổng số dự đoán.
- precision_score(y_test, y_pred): Hàm precision_score tính tỷ lệ precision của mô hình bằng cách so sánh các nhãn dự đoán (y_pred) với nhãn thực tế (y_test). Precision đo lường khả năng của mô hình trong việc tránh việc gán nhãn sai cho các mẫu âm tính.
- recall_score(y_test, y_pred): Hàm recall_score tính tỷ lệ recall của mô hình bằng cách so sánh các nhãn dự đoán (y_pred) với nhãn thực tế (y_test). Recall đo lường khả năng của mô hình trong việc bắt được tất cả các mẫu dương tính.
- f1_score(y_test, y_pred): Hàm f1_score tính giá trị F1 của mô hình bằng cách sử dụng precision và recall. Giá trị F1 là trung bình điều hòa của precision và recall, cung cấp một phép đo tổng quát về hiệu suất của mô hình, đặc biệt hữu ích khi số lượng mẫu của các lớp không cân bằng.

```
# Biểu đồ thể hiện độ chính xác của mô hình qua các chỉ số dự đoán

import matplotlib.pyplot as plt
metrics_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
metrics_values = [accuracy_score(y_test, y_pred),
                  precision_score(y_test, y_pred),
                  recall_score(y_test, y_pred),
                  f1_score(y_test, y_pred)]

plt.bar(metrics_names, metrics_values, color=['blue', 'green', 'red', 'purple'])
plt.title('SVM Model Evaluation')
plt.xlabel('Metrics')
plt.ylabel('Values')
plt.show()
```

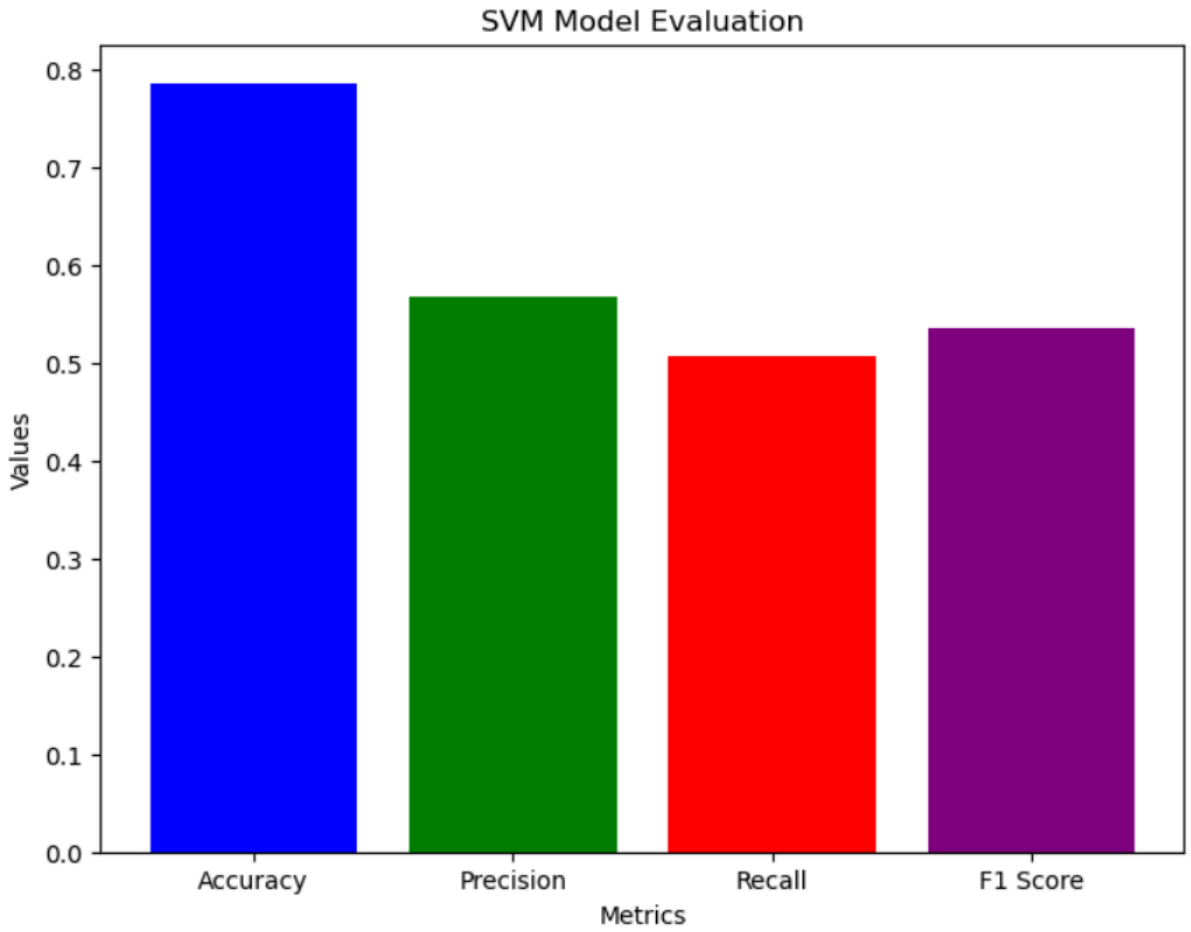
```
import matplotlib.pyplot as plt
```
import matplotlib.pyplot as plt: Dòng này import thư viện matplotlib, cụ thể là module pyplot, và đặt tên viết tắt là plt.

```
metrics_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
```
Dòng này tạo ra một list có tên là metrics_names, chứa các tên của các chỉ số đánh giá mà chúng ta muốn vẽ trên trục x của biểu đồ cột.

```
metrics_values = [accuracy_score(y_test, y_pred),
                  precision_score(y_test, y_pred),
                  recall_score(y_test, y_pred),
                  f1_score(y_test, y_pred)]
```
metrics_values = [accuracy_score(y_test, y_pred), precision_score(y_test, y_pred), recall_score(y_test, y_pred), f1_score(y_test, y_pred)]: Dòng này tạo ra một list có tên là metrics_values, chứa các giá trị của các chỉ số đánh giá tương ứng với các metrics_names. Các giá trị này được tính toán bằng cách sử dụng các hàm như accuracy_score, precision_score, recall_score, và f1_score. Các hàm này so sánh giá trị thực (y_test) với các dự đoán (y_pred) được thực hiện bởi mô hình.

```
plt.bar(metrics_names, metrics_values, color=['blue', 'green', 'red', 'purple'])
```
Dòng này tạo ra một biểu đồ cột bằng cách sử dụng hàm plt.bar(). Đối số đầu tiên metrics_names cung cấp các nhãn cho các cột trên trục x, đối số thứ hai metrics_values cung cấp chiều cao của các cột, và tham số color chỉ định màu của mỗi cột.

```
plt.title('SVM Model Evaluation')
plt.xlabel('Metrics')
plt.ylabel('Values')
plt.show()
```
- plt.title('SVM Model Evaluation'): Dòng này đặt tiêu đề của biểu đồ thành 'SVM Model Evaluation' bằng cách sử dụng plt.title().
- plt.xlabel('Metrics'): Dòng này đặt nhãn cho trục x của biểu đồ thành 'Metrics' bằng cách sử dụng plt.xlabel().
- plt.ylabel('Values'): Dòng này đặt nhãn cho trục y của biểu đồ thành 'Values' bằng cách sử dụng plt.ylabel().
- plt.show(): Dòng này hiển thị biểu đồ đã được tạo ra bằng các lệnh trước đó. Việc gọi plt.show() là quan trọng để thực sự vẽ biểu đồ lên màn hình.



KẾT LUẬN

Tổng kết:

- Áp dụng các kĩ thuật tiền xử lí dữ liệu để thực hiện:
    - + Xử lí dữ liệu thiếu
    - + Mã hóa dữ liệu
    - + Xử lí ngoại lệ
    - + Chuẩn hóa dữ liệu
- Áp dụng mô hình học máy SVM để dự đoán cho bài toán.

****

TÀI LIỆU THAM KHẢO

- Slide bài giảng môn Tiền xử lí dữ liệu của TS.Ta Quang Chieu
- Sách Machine Learning cơ bản của Vũ Hữu Tiệp
- Sách Data Preprocessing with Python for Absolute Beginners, nguồn: internet