

## Buổi 4. Tiền xử lý dữ liệu

### 1 Thông tin chung

#### Mục tiêu

- Giới thiệu các thao tác tiền xử lý dữ liệu như: làm sạch dữ liệu, xử lý các thuộc tính kiểu phân loại, thu giảm dữ liệu.

#### Kết quả đạt được

Sinh viên sau khi thực hành:

- Thành thạo các thao tác tiền xử lý dữ liệu.
- Cài đặt được các ví dụ thực hành.

**Thời gian thực hành:** 3 tiết

**Công cụ thực hành:** Google Colab, Anaconda

### 2 Nội dung thực hành

#### 2.1 Các thao tác tiền xử lý dữ liệu

Giai đoạn tiền xử lý dữ liệu thường sử dụng các kỹ thuật:

- Làm sạch dữ liệu
- Tích hợp dữ liệu
- Biến đổi dữ liệu
- Thu giảm dữ liệu

Ví dụ: Cho bảng dữ liệu gồm bốn thuộc tính như sau:

- Country
- Age
- Salary
- Purchased

```
[39]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data = pd.read_csv('data.csv')
data
```

```
[39]:
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Quan sát bảng dữ liệu, ta thấy cần xử lý một số vấn đề sau:

- Dữ liệu bị thiếu (NaN)
- Thuộc tính là thuộc kiểu phân loại
- Thu giảm dữ liệu

### 2.1.1 Xử lý dữ liệu bị thiếu

- Kiểm tra các thuộc tính bị thiếu dữ liệu.

```
[40]: data.isnull().sum()
```

```
[40]: Country      0
      Age          1
      Salary       1
      Purchased    0
      dtype: int64
```

*Nhận xét: Kết quả cho thấy hai thuộc tính Age và Salary bị thiếu dữ liệu.*

**Cách 1: Xóa các dòng bị thiếu dữ liệu.**

```
[41]: data_drop = data.copy()
      data_drop
```

```
[41]:
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[42]: data_drop.dropna(inplace = True)
      data_drop
```

```
[42]:   Country  Age  Salary Purchased
0  France  44.0  72000.0         No
1   Spain  27.0  48000.0         Yes
2  Germany 30.0  54000.0         No
3   Spain  38.0  61000.0         No
5  France  35.0  58000.0         Yes
7  France  48.0  79000.0         Yes
8  Germany 50.0  83000.0         No
9  France  37.0  67000.0         Yes
```

**Cách 2: Điền dữ liệu vào giá trị NaN dựa vào mean, median hay zero.**

```
[43]: age_mean = data['Age'].mean()
      print(age_mean)

      data['Age'].fillna(age_mean, inplace = True)
      data
```

```
38.77777777777778
```

```
[43]:   Country  Age  Salary Purchased
0  France  44.000000  72000.0         No
1   Spain  27.000000  48000.0         Yes
2  Germany 30.000000  54000.0         No
3   Spain  38.000000  61000.0         No
4  Germany 40.000000      NaN         Yes
5  France  35.000000  58000.0         Yes
6   Spain  38.777778  52000.0         No
7  France  48.000000  79000.0         Yes
8  Germany 50.000000  83000.0         No
9  France  37.000000  67000.0         Yes
```

```
[44]: salary_mean = data['Salary'].mean()
      print(salary_mean)

      data['Salary'].fillna(salary_mean, inplace = True)
      data
```

```
63777.77777777778
```

```
[44]:   Country  Age  Salary Purchased
0  France  44.000000  72000.000000         No
1   Spain  27.000000  48000.000000         Yes
2  Germany 30.000000  54000.000000         No
3   Spain  38.000000  61000.000000         No
```

4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

## 2.1.2 Xử lý đối với dữ liệu phân loại

Cách 1: Ánh xạ mỗi giá trị phân loại thành một giá trị số.

```
[45]: data['Country'].value_counts()
```

```
[45]: France      4
      Spain      3
      Germany    3
      Name: Country, dtype: int64
```

```
[46]: data_map = data.copy()
      data_map
```

```
[46]:
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

```
[47]: countries_mapping = {"France": 0, "Spain": 1, "Germany": 2}
      data_map['Country'] = data_map['Country'].map(countries_mapping)
      data_map
```

```
[47]:
```

	Country	Age	Salary	Purchased
0	0	44.000000	72000.000000	No
1	1	27.000000	48000.000000	Yes
2	2	30.000000	54000.000000	No
3	1	38.000000	61000.000000	No
4	2	40.000000	63777.777778	Yes
5	0	35.000000	58000.000000	Yes
6	1	38.777778	52000.000000	No
7	0	48.000000	79000.000000	Yes
8	2	50.000000	83000.000000	No

```
9          0  37.000000  67000.000000      Yes
```

### Cách 2: Sử dụng one-hot vector

```
[48]: from sklearn.preprocessing import OneHotEncoder

onehotencoder = OneHotEncoder()
data_country_encoded = onehotencoder.fit_transform(data[['Country']]).toarray()
data_country_encoded
```

```
[48]: array([[1., 0., 0.],
             [0., 0., 1.],
             [0., 1., 0.],
             [0., 0., 1.],
             [0., 1., 0.],
             [1., 0., 0.],
             [0., 0., 1.],
             [1., 0., 0.],
             [0., 1., 0.],
             [1., 0., 0.]])
```

```
[49]: country_names = data['Country'].unique()
country_names
```

```
[49]: array(['France', 'Spain', 'Germany'], dtype=object)
```

- Thêm vào các thuộc tính tương ứng với mỗi quốc gia riêng biệt.

```
[50]: for i in range(country_names.shape[0]):
        data['Country_' + country_names[i]] = data_country_encoded[:, i]

data
```

```
[50]:
```

	Country	Age	Salary	Purchased	Country_France	Country_Spain	\
0	France	44.000000	72000.000000	No	1.0	0.0	
1	Spain	27.000000	48000.000000	Yes	0.0	0.0	
2	Germany	30.000000	54000.000000	No	0.0	1.0	
3	Spain	38.000000	61000.000000	No	0.0	0.0	
4	Germany	40.000000	63777.777778	Yes	0.0	1.0	
5	France	35.000000	58000.000000	Yes	1.0	0.0	
6	Spain	38.777778	52000.000000	No	0.0	0.0	
7	France	48.000000	79000.000000	Yes	1.0	0.0	
8	Germany	50.000000	83000.000000	No	0.0	1.0	
9	France	37.000000	67000.000000	Yes	1.0	0.0	

  

	Country_Germany
0	0.0
1	1.0

```

2          0.0
3          1.0
4          0.0
5          0.0
6          1.0
7          0.0
8          0.0
9          0.0

```

- Xóa thuộc tính Country

```
[51]: data.drop('Country', axis = 1, inplace = True)
data
```

```
[51]:
```

	Age	Salary	Purchased	Country_France	Country_Spain	\
0	44.000000	72000.000000	No	1.0	0.0	
1	27.000000	48000.000000	Yes	0.0	0.0	
2	30.000000	54000.000000	No	0.0	1.0	
3	38.000000	61000.000000	No	0.0	0.0	
4	40.000000	63777.777778	Yes	0.0	1.0	
5	35.000000	58000.000000	Yes	1.0	0.0	
6	38.777778	52000.000000	No	0.0	0.0	
7	48.000000	79000.000000	Yes	1.0	0.0	
8	50.000000	83000.000000	No	0.0	1.0	
9	37.000000	67000.000000	Yes	1.0	0.0	

  

```

Country_Germany
0          0.0
1          1.0
2          0.0
3          1.0
4          0.0
5          0.0
6          1.0
7          0.0
8          0.0
9          0.0

```

- Thực hiện tương tự với thuộc tính Purchased

```
[52]: from sklearn.preprocessing import OrdinalEncoder

ordencoder = OrdinalEncoder()

data_labels = data.iloc[:, 3].values
data_purchased_encoded = ordencoder.fit_transform(data_labels.reshape(-1, 1))
data_purchased_encoded
```

```
[52]: array([[1.],
          [0.],
          [0.],
          [0.],
          [0.],
          [1.],
          [0.],
          [1.],
          [0.],
          [1.]])
```

```
[53]: data['Purchased'] = data_purchased_encoded
data
```

```
[53]:      Age      Salary  Purchased  Country_France  Country_Spain  \
0  44.000000  72000.000000         1.0             1.0             0.0
1  27.000000  48000.000000         0.0             0.0             0.0
2  30.000000  54000.000000         0.0             0.0             1.0
3  38.000000  61000.000000         0.0             0.0             0.0
4  40.000000  63777.777778         0.0             0.0             1.0
5  35.000000  58000.000000         1.0             1.0             0.0
6  38.777778  52000.000000         0.0             0.0             0.0
7  48.000000  79000.000000         1.0             1.0             0.0
8  50.000000  83000.000000         0.0             0.0             1.0
9  37.000000  67000.000000         1.0             1.0             0.0

      Country_Germany
0              0.0
1              1.0
2              0.0
3              1.0
4              0.0
5              0.0
6              1.0
7              0.0
8              0.0
9              0.0
```

### 2.1.3 Thu giảm dữ liệu

- Min-Max

```
[54]: data_num = data[['Age', 'Salary']]
data_num
```

```
[54]:      Age      Salary
0  44.000000  72000.000000
```

```
1 27.000000 48000.000000
2 30.000000 54000.000000
3 38.000000 61000.000000
4 40.000000 63777.777778
5 35.000000 58000.000000
6 38.777778 52000.000000
7 48.000000 79000.000000
8 50.000000 83000.000000
9 37.000000 67000.000000
```

```
[55]: from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler()
dataset_num_scaled = min_max_scaler.fit_transform(data_num)
dataset_num_scaled
```

```
[55]: array([[0.73913043, 0.68571429],
              [0.          , 0.          ],
              [0.13043478, 0.17142857],
              [0.47826087, 0.37142857],
              [0.56521739, 0.45079365],
              [0.34782609, 0.28571429],
              [0.51207729, 0.11428571],
              [0.91304348, 0.88571429],
              [1.          , 1.          ],
              [0.43478261, 0.54285714]])
```

```
[56]: from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()
dataset_num_scaled = std_scaler.fit_transform(data_num)
dataset_num_scaled
```

```
[56]: array([[ 7.58874362e-01,  7.49473254e-01],
              [-1.71150388e+00, -1.43817841e+00],
              [-1.27555478e+00, -8.91265492e-01],
              [-1.13023841e-01, -2.53200424e-01],
              [ 1.77608893e-01,  6.63219199e-16],
              [-5.48972942e-01, -5.26656882e-01],
              [ 0.00000000e+00, -1.07356980e+00],
              [ 1.34013983e+00,  1.38753832e+00],
              [ 1.63077256e+00,  1.75214693e+00],
              [-2.58340208e-01,  2.93712492e-01]])
```

```
[57]: data['Age'] = dataset_num_scaled[:, 0]
data['Salary'] = dataset_num_scaled[:, 1]
```



```
data
```

```
[57]:      Age      Salary  Purchased  Country_France  Country_Spain  \
0  0.758874  7.494733e-01        1.0            1.0          0.0
1 -1.711504 -1.438178e+00        0.0            0.0          0.0
2 -1.275555 -8.912655e-01        0.0            0.0          1.0
3 -0.113024 -2.532004e-01        0.0            0.0          0.0
4  0.177609  6.632192e-16        0.0            0.0          1.0
5 -0.548973 -5.266569e-01        1.0            1.0          0.0
6  0.000000 -1.073570e+00        0.0            0.0          0.0
7  1.340140  1.387538e+00        1.0            1.0          0.0
8  1.630773  1.752147e+00        0.0            0.0          1.0
9 -0.258340  2.937125e-01        1.0            1.0          0.0

      Country_Germany
0                0.0
1                1.0
2                0.0
3                1.0
4                0.0
5                0.0
6                1.0
7                0.0
8                0.0
9                0.0
```

### 3 Bài tập

Cho dữ liệu tiền thưởng của khách hàng dành cho nhân viên phục vụ tại địa chỉ:  
<https://www.kaggle.com/ranjeetjain3/seaborn-tips-dataset>

- Sinh viên hãy thực hiện các thao tác tiền xử lý dữ liệu.