baikiemtra1

June 27, 2024

Bài kiểm tra 1: Nhập môn dữ liệu lớn và học sâu - Họ và tên: Thái Thị Trà My - Mssv: 2174802010891

[368]:
```
import pandas as pd
```

1. Tạo một DataFrame từ dữ liệu đã cho

[369]:
```
# Tạo một DataFrame
data = {
    'Name': ['Alice', 'Bod', 'Charlie', 'David', 'Eva', 'Frank', 'Grace',␣
 ↪'Hannah', 'Ivan', 'Jack', 'Kelly', 'Liam', 'Mona', 'Nina','Oscar'],
    'Age': [25, 30, 35, 28, 22, 45, 34, 31, 27, 29, 33, 40, 26, 32, 36],
    'Salary': [50000, 60000, 70000, 55000, 52000, 80000, 72000, 68000, 61000,␣
 ↪59000, 63000, 77000, 53000, 66000, 75000]
}
```

[370]:
```
df= pd.DataFrame(data)
```

2. Hiển thị thông tin về DataFrame vừa tạo

[371]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Name    15 non-null     object
 1   Age     15 non-null     int64 
 2   Salary  15 non-null     int64 
dtypes: int64(2), object(1)
memory usage: 488.0+ bytes
```

[372]:
```
df
```

[372]:

	Name	Age	Salary
0	Alice	25	50000
1	Bod	30	60000
2	Charlie	35	70000

```
3     David   28   55000
4       Eva   22   52000
5     Frank   45   80000
6     Grace   34   72000
7    Hannah   31   68000
8      Ivan   27   61000
9      Jack   29   59000
10    Kelly   33   63000
11     Liam   40   77000
12     Mona   26   53000
13     Nina   32   66000
14    Oscar   36   75000
```

3. Lọc hàng trong DataFrame có 'Age' lớn hơn 28

[373]:
```
# Lọc các hàng mà Age > 28
filtered_df = df[df['Age'] > 28]
```

[374]:
```
filtered_df
```

[374]:
```
       Name  Age  Salary
1       Bod   30   60000
2   Charlie   35   70000
5     Frank   45   80000
6     Grace   34   72000
7    Hannah   31   68000
9      Jack   29   59000
10    Kelly   33   63000
11     Liam   40   77000
13     Nina   32   66000
14    Oscar   36   75000
```

4. Tính giá trị trung bình của cột 'Salary'

[375]:
```
TB_Salary = df['Salary'].mean()
```

[376]:
```
print("Giá trị trung bình của cột Salary: ")
print(TB_Salary)
```

```
Giá trị trung bình của cột Salary:
64066.666666666664
```

5. Nhóm dữ liệu theo cột 'Age' và tính tổng 'Salary' cho mỗi nhóm

[377]:
```
group_df = df.groupby('Age')['Salary'].sum().reset_index()
```

[378]:
```
group_df
```

[378]:
	Age	Salary
0	22	52000
1	25	50000
2	26	53000
3	27	61000
4	28	55000
5	29	59000
6	30	60000
7	31	68000
8	32	66000
9	33	63000
10	34	72000
11	35	70000
12	36	75000
13	40	77000
14	45	80000

6. Sắp xếp DataFrame theo cột 'Salary' giảm dần

[379]:
```
df.sort_values(by='Salary', ascending=False)
```

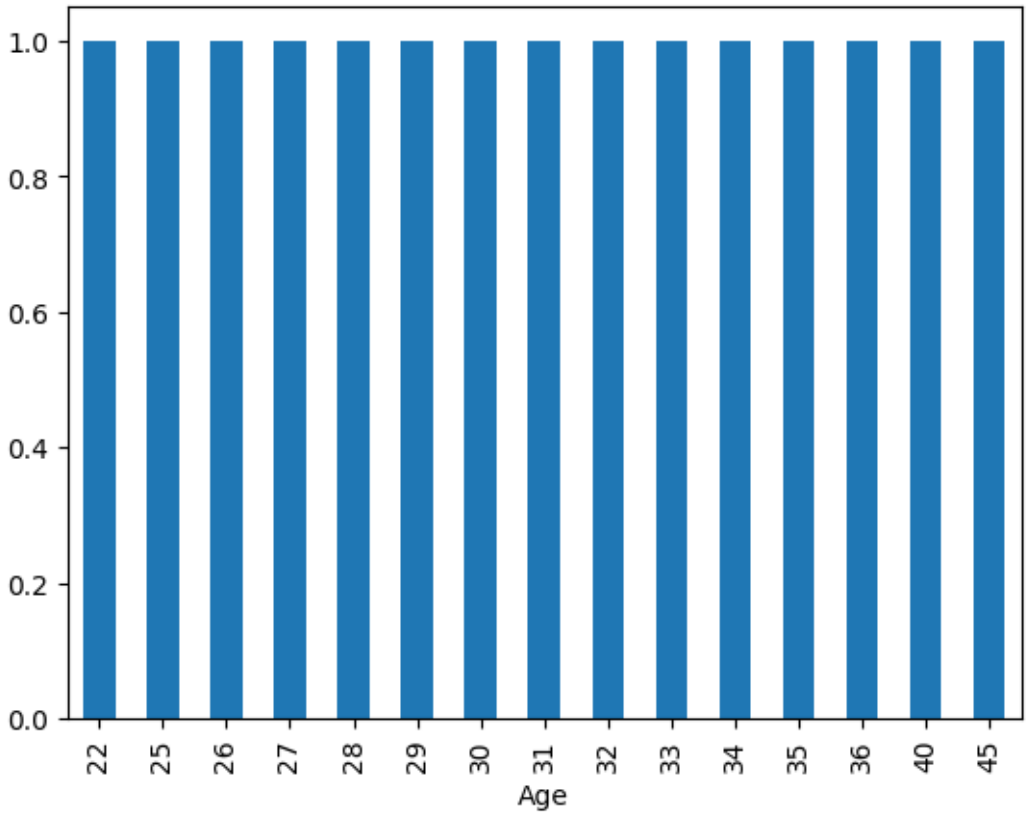
[379]:
	Name	Age	Salary
5	Frank	45	80000
11	Liam	40	77000
14	Oscar	36	75000
6	Grace	34	72000
2	Charlie	35	70000
7	Hannah	31	68000
13	Nina	32	66000
10	Kelly	33	63000
8	Ivan	27	61000
1	Bod	30	60000
9	Jack	29	59000
3	David	28	55000
12	Mona	26	53000
4	Eva	22	52000
0	Alice	25	50000

7. Vẽ biểu đồ cho cột 'Age'

[380]:
```
# Vẽ biểu đồ Histogram
c7 = df.groupby('Age')['Age'].agg('count')
```

[381]:
```
c7.plot.bar()
```

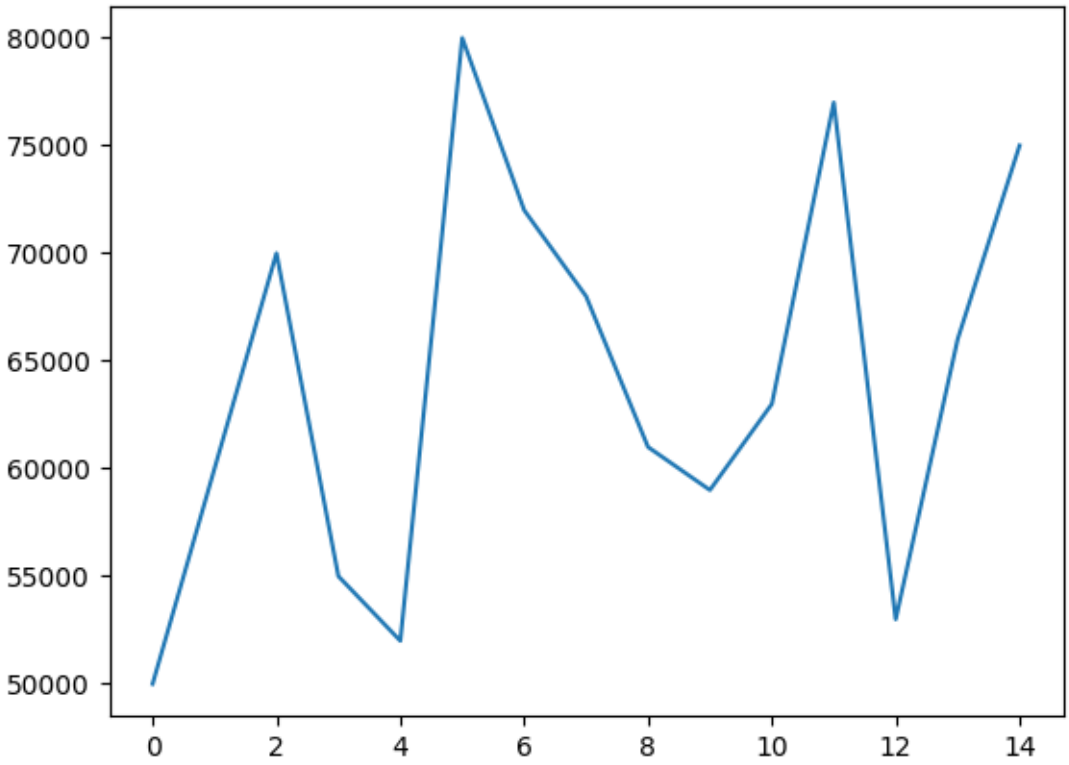
[381]:
```
<AxesSubplot:xlabel='Age'>
```



8. Vẽ biểu đồ đường cao cho cột 'Salary'

[382]:
```
df['Salary'].plot()
```

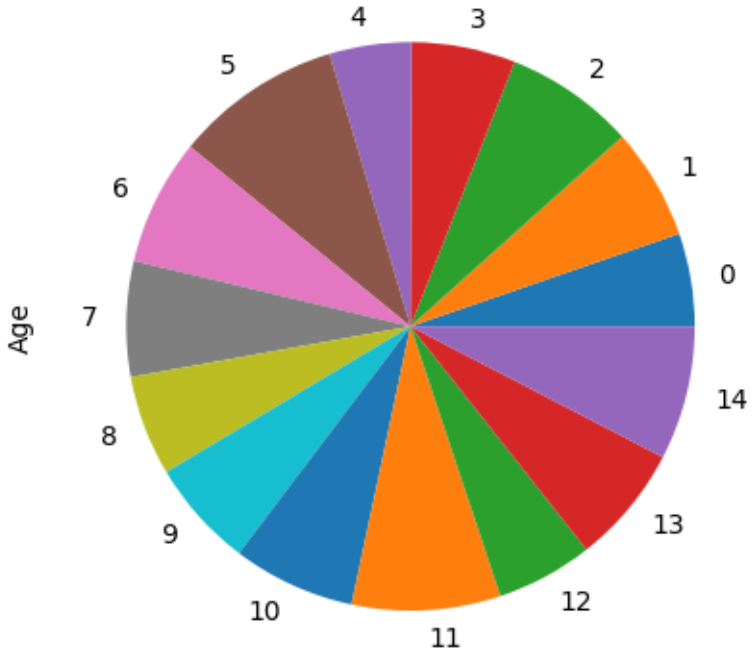
[382]:
```
<AxesSubplot:>
```



9. Vẽ biểu đồ tròn cho cột 'Age'

[383]:
```
df['Age'].plot.pie()
```

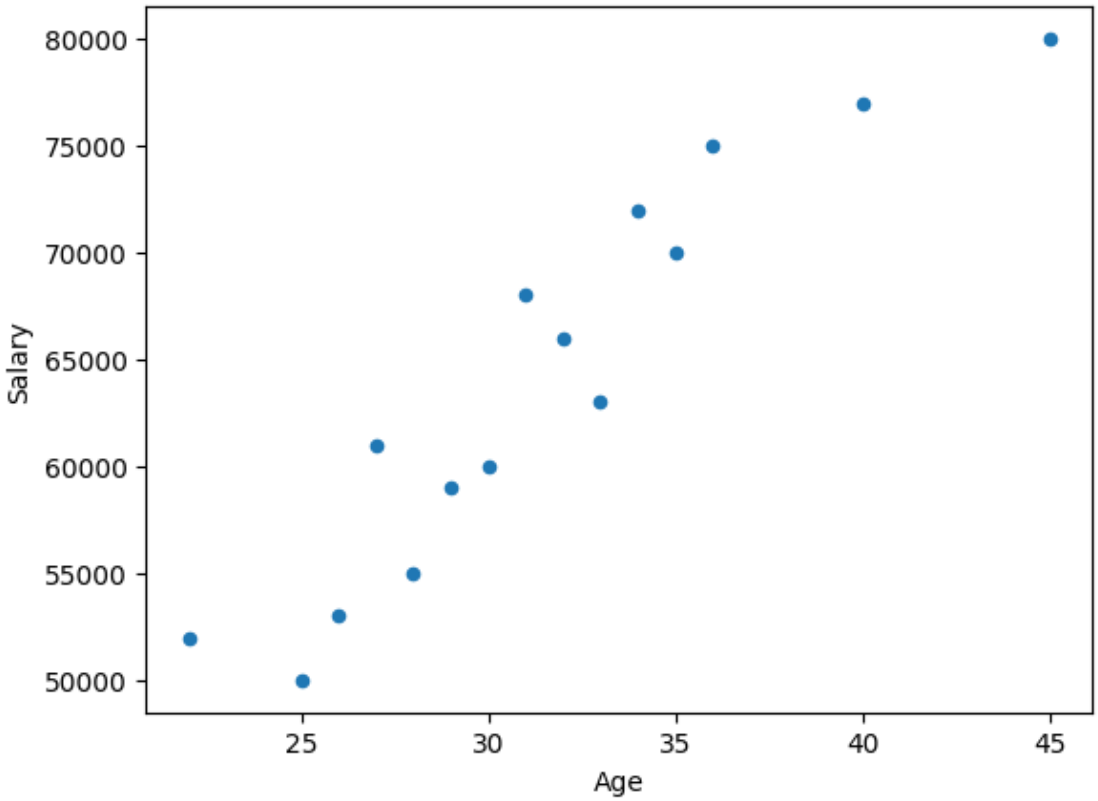
[383]:
```
<AxesSubplot:ylabel='Age'>
```



10. Vẽ biểu đồ phân tán cho ‘Age’ và ‘Salary’

[384]:
```
df.plot.scatter(x='Age', y='Salary')
```

[384]:
```
<AxesSubplot:xlabel='Age', ylabel='Salary'>
```



11. Kiểm tra xem có giá trị nào NaN trong DataFrame không

[385]:
```
df.isnull().sum()
```

[385]:
```
Name      0
Age       0
Salary    0
dtype: int64
```

12. Thay thế các giá trị của cột 'Age' lớn hơn 30 bằng giá trị trung bình của cột

[386]:
```
# Tính giá trị trung bình của cột 'Age'
TB_age = df['Age'].mean() #.round() nếu chỉ để kết quả số nguyên
```

[387]:
```
# Thay thế các giá trị của cột 'Age'
df.loc[df['Age'] > 30, 'Age'] = TB_age
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13968\217069998.py:2: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise an error in a
future version of pandas. Value '31.533333333333335' has dtype incompatible with
int64, please explicitly cast to a compatible dtype first.
  df.loc[df['Age'] > 30, 'Age'] = TB_age
```

[388]:
```
df
```

[388]:

	Name	Age	Salary
0	Alice	25.000000	50000
1	Bod	30.000000	60000
2	Charlie	31.533333	70000
3	David	28.000000	55000
4	Eva	22.000000	52000
5	Frank	31.533333	80000
6	Grace	31.533333	72000
7	Hannah	31.533333	68000
8	Ivan	27.000000	61000
9	Jack	29.000000	59000
10	Kelly	31.533333	63000
11	Liam	31.533333	77000
12	Mona	26.000000	53000
13	Nina	31.533333	66000
14	Oscar	31.533333	75000

13. Chuẩn hóa cột 'Age' về khoảng giá trị từ 0 đến 1

[389]:
```
# Chuẩn hoá cột 'Age' về khoảng giá trị từ 0 đến 1
    # Lấy giá trị ban đầu của Age trừ cho giá trị nhỏ nhất của cột Age
    # Chia cho hiệu của giá trị cột Age lớn nhất trừ giá trị cột Age nhỏ nhất
df['Normalized_age'] = (df['Age'] - df['Age'].min()) / (df['Age'].max() -␣
 ↪df['Age'].min())
```

[390]:
```
df
```

[390]:

	Name	Age	Salary	Normalized_age
0	Alice	25.000000	50000	0.314685
1	Bod	30.000000	60000	0.839161
2	Charlie	31.533333	70000	1.000000
3	David	28.000000	55000	0.629371
4	Eva	22.000000	52000	0.000000
5	Frank	31.533333	80000	1.000000
6	Grace	31.533333	72000	1.000000
7	Hannah	31.533333	68000	1.000000
8	Ivan	27.000000	61000	0.524476
9	Jack	29.000000	59000	0.734266
10	Kelly	31.533333	63000	1.000000
11	Liam	31.533333	77000	1.000000
12	Mona	26.000000	53000	0.419580
13	Nina	31.533333	66000	1.000000
14	Oscar	31.533333	75000	1.000000

14. Tạo một cột mới 'Age_group' phân loại tuổi thành 'Young, 'Middle-aged' và 'Old' dựa trên giá trị cột 'Age'

[391]:
```
# Định nghĩa hàm để phân loại tuổi thành các nhóm
def phanloai_age(age):
    if age <= 30:
        return 'Young'
    elif age <= 60 and age > 30:
        return 'Middle-aged'
    else:
        return 'Old'
```

[392]:
```
df['Age_group'] = df['Age'].apply(phanloai_age)
```

[393]:
```
df
```

[393]:

	Name	Age	Salary	Normalized_age	Age_group
0	Alice	25.000000	50000	0.314685	Young
1	Bod	30.000000	60000	0.839161	Young
2	Charlie	31.533333	70000	1.000000	Middle-aged
3	David	28.000000	55000	0.629371	Young
4	Eva	22.000000	52000	0.000000	Young
5	Frank	31.533333	80000	1.000000	Middle-aged
6	Grace	31.533333	72000	1.000000	Middle-aged
7	Hannah	31.533333	68000	1.000000	Middle-aged
8	Ivan	27.000000	61000	0.524476	Young
9	Jack	29.000000	59000	0.734266	Young
10	Kelly	31.533333	63000	1.000000	Middle-aged
11	Liam	31.533333	77000	1.000000	Middle-aged
12	Mona	26.000000	53000	0.419580	Young
13	Nina	31.533333	66000	1.000000	Middle-aged
14	Oscar	31.533333	75000	1.000000	Middle-aged

15. Tính toán tỷ lệ phần trăm thay đổi của cột 'Salary'

[394]:
```
# Tính tỷ lệ phần trăm thay đổi bằng phương thức pct_change()
df['Percentage_change'] = df['Salary'].pct_change() * 100
```

[395]:
```
df
```

[395]:

	Name	Age	Salary	Normalized_age	Age_group	Percentage_change
0	Alice	25.000000	50000	0.314685	Young	NaN
1	Bod	30.000000	60000	0.839161	Young	20.000000
2	Charlie	31.533333	70000	1.000000	Middle-aged	16.666667
3	David	28.000000	55000	0.629371	Young	-21.428571
4	Eva	22.000000	52000	0.000000	Young	-5.454545
5	Frank	31.533333	80000	1.000000	Middle-aged	53.846154
6	Grace	31.533333	72000	1.000000	Middle-aged	-10.000000
7	Hannah	31.533333	68000	1.000000	Middle-aged	-5.555556
8	Ivan	27.000000	61000	0.524476	Young	-10.294118

9	Jack	29.000000	59000	0.734266	Young	-3.278689
10	Kelly	31.533333	63000	1.000000	Middle-aged	6.779661
11	Liam	31.533333	77000	1.000000	Middle-aged	22.222222
12	Mona	26.000000	53000	0.419580	Young	-31.168831
13	Nina	31.533333	66000	1.000000	Middle-aged	24.528302
14	Oscar	31.533333	75000	1.000000	Middle-aged	13.636364

16. Tìm các giá trị trùng lặp trong DataFrame dựa trên cột 'Name' và loại bỏ các trùng lặp, giữ lại hàng đầu tiên

[396]:
```
# Tìm và loại bỏ các giá trị trùng lặp dựa trên cột 'Name', giữ lại hàng đầu␣
↪tiên
no_duplicates = df.drop_duplicates(subset=['Name'], keep='first')
```

[397]:
```
no_duplicates
```

[397]:

	Name	Age	Salary	Normalized_age	Age_group	Percentage_change
0	Alice	25.000000	50000	0.314685	Young	NaN
1	Bod	30.000000	60000	0.839161	Young	20.000000
2	Charlie	31.533333	70000	1.000000	Middle-aged	16.666667
3	David	28.000000	55000	0.629371	Young	-21.428571
4	Eva	22.000000	52000	0.000000	Young	-5.454545
5	Frank	31.533333	80000	1.000000	Middle-aged	53.846154
6	Grace	31.533333	72000	1.000000	Middle-aged	-10.000000
7	Hannah	31.533333	68000	1.000000	Middle-aged	-5.555556
8	Ivan	27.000000	61000	0.524476	Young	-10.294118
9	Jack	29.000000	59000	0.734266	Young	-3.278689
10	Kelly	31.533333	63000	1.000000	Middle-aged	6.779661
11	Liam	31.533333	77000	1.000000	Middle-aged	22.222222
12	Mona	26.000000	53000	0.419580	Young	-31.168831
13	Nina	31.533333	66000	1.000000	Middle-aged	24.528302
14	Oscar	31.533333	75000	1.000000	Middle-aged	13.636364