20250617_01

June 17, 2025

[1]:
```
# Getting some thoughts about merging.
import pandas as pd
```

[2]:
```
# First table
sales = pd.DataFrame({'store_id': [1, 2, 3],
                      'sales': [250, 400, 150]})
# Second table
locations = pd.DataFrame({'store_id': [1, 2, 4],
                          'city': ['Taipei', 'Kaohsiung', 'Tainan']})
```

[10]:
```
# Inner Join : keep only in both
inner = pd.merge(sales, locations, on = 'store_id', how = 'inner')
print(inner)
```

```
   store_id  sales       city
0         1    250     Taipei
1         2    400  Kaohsiung
```

[11]:
```
# Left Join : keep all from left (this case, sales)
left = pd.merge(sales, locations, on = 'store_id', how = 'left')
print(left)
```

```
   store_id  sales       city
0         1    250     Taipei
1         2    400  Kaohsiung
2         3    150        NaN
```

[12]:
```
# Right Join : keep all from right (this case, locations)
right = pd.merge(sales, locations, on = 'store_id', how = 'right')
print(right)
```

```
   store_id  sales       city
0         1  250.0     Taipei
1         2  400.0  Kaohsiung
2         4    NaN     Tainan
```

[13]:
```
# Outer Join : keep everything
outer = pd.merge(sales, locations, on = 'store_id', how = 'outer')
print(outer)
```

```
   store_id  sales       city
0         1  250.0     Taipei
1         2  400.0  Kaohsiung
2         3  150.0        NaN
3         4    NaN     Tainan
```

[14]:
```
# Dirty keys, dirty indeed.
sales_dirty = pd.DataFrame({'store_id': ['1 ', '2', '3'],
                            'sales': [250, 400, 150]})

locations_clean = pd.DataFrame({'store_id': ['1', '2', '4'],
                                'city': ['Taipei', 'Kaohsiung', 'Tainan']})

# Try merging
bad_merge = pd.merge(sales_dirty, locations_clean, on = 'store_id', how =␣
 ↪'left')
print(bad_merge)
```

```
  store_id  sales       city
0       1     250        NaN
1        2    400  Kaohsiung
2        3    150        NaN
```

[16]:
```
# Try fix this
# .str.strip() gets rid of all head and tail blank space
sales_dirty['store_id'] = sales_dirty['store_id'].str.strip()

# Merge again
good_merge = pd.merge(sales_dirty, locations_clean, on = 'store_id', how =␣
 ↪'left')
print(good_merge)
```

```
  store_id  sales       city
0        1    250     Taipei
1        2    400  Kaohsiung
2        3    150        NaN
```

[18]:
```
# Now try normalization and standardization
data = pd.DataFrame({'age': [18, 22, 25, 30, 35],
                     'income': [1000, 2500, 3000, 4000, 8000]})
```

[23]:
```
# Min-max normalization
normalized = (data - data.min()) / (data.max() - data.min())
print(normalized)
```

```
        age    income
0  0.000000  0.000000
1  0.235294  0.214286
2  0.411765  0.285714
```

```
3  0.705882  0.428571
4  1.000000  1.000000
```

[24]:
```
# Standardization
standardized = (data - data.mean()) / data.std()
print(standardized)
```

```
        age    income
0 -1.199251 -1.024168
1 -0.599625 -0.455186
2 -0.149906 -0.265525
3  0.599625  0.113796
4  1.349157  1.631083
```

[25]:
```
# Or we can do it in a fancier way
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Normalize
minmax = MinMaxScaler()
fancy_normalized = pd.DataFrame(minmax.fit_transform(data), columns = data.
 ↪columns)

# Standardize
standard = StandardScaler()
fancy_standardized = pd.DataFrame(standard.fit_transform(data), columns = data.
 ↪columns)
```

[26]:
```
fancy_normalized
```

[26]:
```
        age    income
0  0.000000  0.000000
1  0.235294  0.214286
2  0.411765  0.285714
3  0.705882  0.428571
4  1.000000  1.000000
```

[28]:
```
# This give different result since it use different std
fancy_standardized
```

[28]:
```
        age    income
0 -1.340803 -1.145055
1 -0.670402 -0.508913
2 -0.167600 -0.296866
3  0.670402  0.127228
4  1.508403  1.823606
```