Bài 2
Pandas cơ bản

Khóa học: Phân tích dữ liệu với Python

Nội dung

1. Giới thiệu về Pandas
2. Các kiểu dữ liệu cơ bản trong Pandas
3. Làm việc với DataFrame

1. Giới thiệu về Pandas

- Pandas là một thư viện mã nguồn mở trong Python được Wes McKinney phát triển vào năm 2008, cho phép phân tích và xử lý dữ liệu một cách linh hoạt, mạnh mẽ.
- Năm 2019 Pandas được đánh giá là một trong những công cụ khoa học dữ liệu phổ biến nhất trong phân tích và thu thập dữ liệu[1].

1. https://www.jetbrains.com/lp/pythondevelopers-survey-2019/

Các tính năng nổi bật của Pandas

- Cho phép làm việc với dữ liệu từ nhiều nguồn khác nhau như: csv, excel, json, html…
- Khả năng làm việc với các tập dữ liệu không đồng nhất với các kiểu dữ liệu đa dạng như kiểu chuỗi, kiểu số, kiểu thời gian…
- Khả năng xử lý dữ liệu khuyết thiếu, ngoại lai…
- Khả năng thêm bớt, lọc tách, phân tích, biến đổi, biểu diễn… dữ liệu một cách nhanh chóng, trực quan.
- Dễ dàng tích hợp với các thư viện khác của Python.
- …

Khai báo thư viện Pandas

Để làm việc với Pandas chúng ta sử dụng câu lệnh khai báo như sau:

```
import pandas as pd
```

- Thư viện pandas được viết bằng chữ in thường
- pd là tên đại diện dưới dạng thu gọn

2. Các kiểu dữ liệu cơ bản trong Pandas

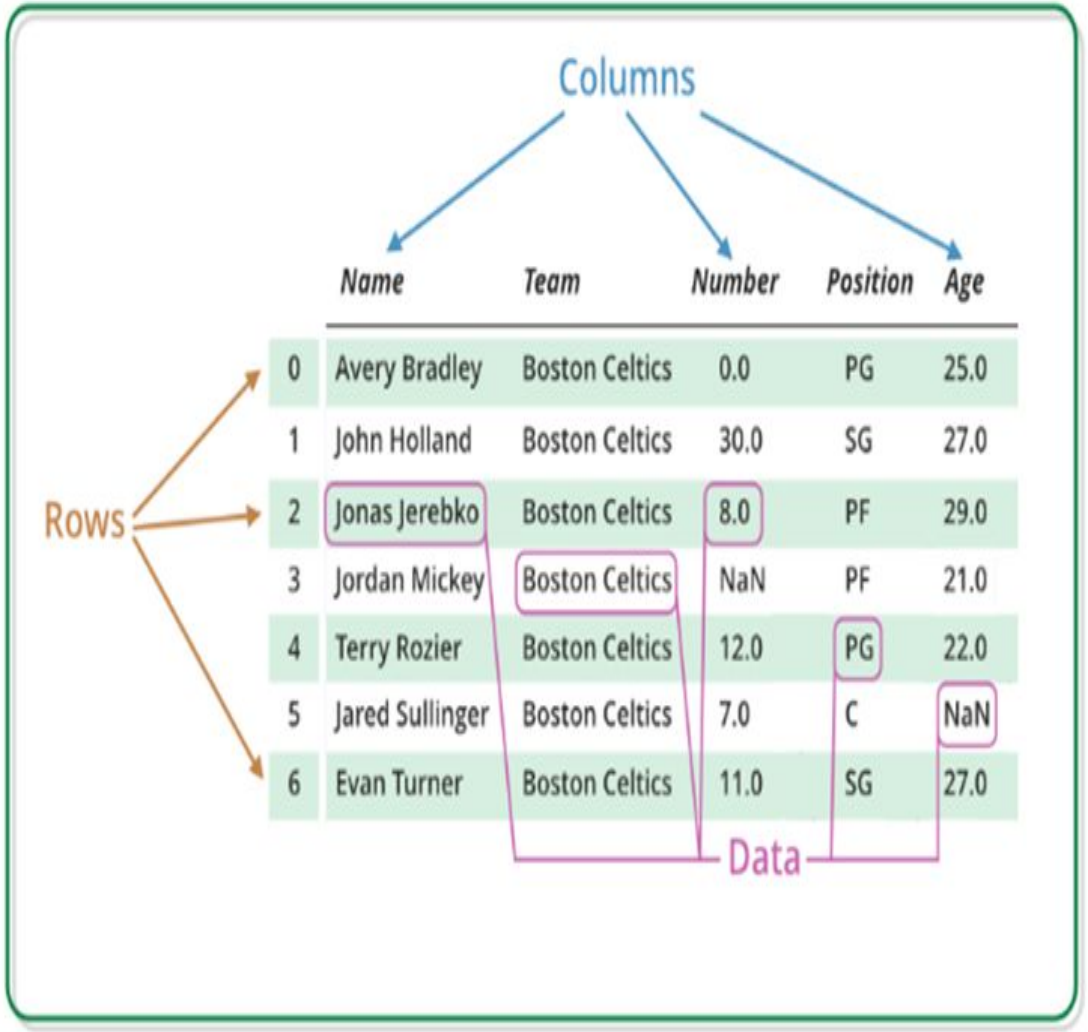
Pandas có hai kiểu dữ liệu cơ bản là Series và DataFrame, trong đó:

- DataFrame là kiểu dữ liệu hai chiều cho phép lưu trữ dữ liệu của một bảng tính.
- Series là kiểu dữ liệu một chiều đại diện cho các cột trong DataFrame.

Giới thiệu DataFrame

DataFrame là một cấu trúc dữ liệu hai chiều có dạng bảng bao gồm các dòng và các cột được gắn nhãn và lập chỉ mục.

- Các cột trong DataFrame mô tả các thuộc tính của dữ liệu có khả năng làm việc với các loại đối tượng khác nhau như kiểu chuỗi, kiểu số, kiểu ngày tháng...
- Các dòng mô tả các giá trị của dữ liệu tương ứng với các thuộc tính.



Tạo DataFrame rỗng

Tạo DataFram rỗng lưu vào biến df:

```
import pandas as pd

df = pd.DataFrame()
```

Tạo DataFrame từ List

- List là một danh sách biểu diễn một chuỗi các phần tử có kiểu dữ liệu đồng nhất hoặc khác nhau.
- Các phần tử được đặt trong cặp dấu ngoặc vuông [] và được ngăn cách nhau bởi dấu phẩy (,)
- Tạo DataFrame từ List:

Khai báo List:

```
lst = ['Kế toán','Kinh doanh','Tiếp thị']
```

Gán giá trị của List vào DataFrame

```
df = pd.DataFrame(lst)
```

Tạo DataFrame từ Tuple

- Tuple biểu diễn một chuỗi các phần tử cố định.
- Các phần tử được đặt trong cặp dấu ngoặc vuông () và được ngăn cách nhau bởi dấu phẩy (,)
- Tạo DataFrame từ Tuple:

Khai báo Tuple:

```
tuples = [('Kế toán', 2000),( 'Kinh doanh', 2100), ('Tiếp thị', 1900)]
```

Gán giá trị của Tuple vào DataFrame

```
df = pd.DataFrame(tuples, columns=['Nghề','Lương'])
```

Tạo DataFrame từ Dictionary

- Dictionary là tập hợp các cặp khóa giá trị theo dạng key:value, key và value được trình bày như một List có thể có kiểu dữ liệu bất kỳ.
- Dictionary được định nghĩa trong dấu ngoặc nhọn {}, giá trị của key tương ứng với các cột, giá trị của value tương ứng với các dòng.
- Tạo DataFrame từ Dictionary :

Khai báo Dictionary :

```
dic = {'Nghề':['Kế toán','Kinh doanh','Tiếp thị'],
       'Lương':[2000, 2100, 1900]}
```

Gán giá trị của Dictionary vào DataFrame

```
df = pd.DataFrame(dic)
```

Giới thiệu về Pandas Series

- Series là cấu trúc dữ liệu tuần tự một chiều có thể lưu trữ và xử lý các loại dữ liệu khác nhau như kiểu chuỗi, kiểu số, kiểu thời gian...
- Series là một trong các cột của DataFrame.
- Có thể tạo Series bằng cách sử dụng Dictionary, mảng NumPy và từ các giá trị vô hướng.

Tạo Series từ Dictionary

Khai báo Dictionary:

```
dict = {0 : 'Kế toán', 1 : 'Kinh doanh', 2 : 'Tiếp thị'}
```

Gán giá trị của Dictionary vào Series

```
seri = pd.Series(dict)
```

Tạo Series từ Numpy array

Khai báo thư viện

```
import pandas as pd
import numpy as np
```

Tạo Numpy array

```
arr = np.array([51,65,48,59,68])
```

Gán giá trị của array vào Series

```
seri = pd.Series(arr)
```

Tạo Series từ các giá trị vô hướng

Khai báo thư viện

```
import pandas as pd
import numpy as np
```

Tạo Series

```
seri = pd.Series(10, index=[0,1,2,3,4,5])
```

3. Làm việc với DataFrame

Trong phần này chúng ta tìm hiểu một số các hoạt động chính của DataFrame như sau:

- Tạo DataFrame từ các nguồn dữ liệu có sẵn (csv, xls, xlsx, json, fdf5, html...)
- Mô tả cấu trúc bộ dữ liệu DataFrame (số dòng, số cột, kiểu dữ liệu của các thuộc tính...)
- Chỉnh sửa cấu trúc bộ dữ liệu (đổi tên cột thuộc tính, thêm bớt số dòng số cột...)
- Truy cập, thay thế giá trị các ô dữ liệu...
- Xác định các giá trị thống kê mô tả của các biến định tính, định lượng
- Hợp nhất dữ liệu từ nhiều nguồn
- Tạo bảng tổng hợp pivot table...

Tạo DataFrame từ các nguồn dữ liệu có sẵn

Dữ liệu trong thực tế được thu thập từ rất nhiều nguồn với các định dạng khác nhau, pandas cung cấp các phương thức cho phép đọc dữ liệu từ các nguồn như csv, excel, json, hdf5, html... đưa vào DataFrame.

- Khai báo thư viện: import pandas as pd
- Đọc dữ liệu từ file csv: df=pd.read_csv('FoodPrice_in_Turkey.csv')
- Đọc dữ liệu từ file excel: df=pd.read_excel('house_price_đống đa.xlsx')
- Đọc dữ liệu từ file Json: df = pd.read_json('FoodPrice.json')
- Đọc dữ liệu từ file hdf5: df=pd.read_hdf('FoodPrice.h5', 'table')
- Đọc dữ liệu từ file html: df=pd.read_html('demo_FoodPrice.html')
- Đọc dữ liệu từ liên kết url: url = 'https://en.wikipedia.org/wiki/List_of_sovereign_states_and_dependent_territories_in_North_America'

```
df = pd.read_html(url)
```

Ghi dữ liệu từ DataFrame vào file khác nhau

Dữ liệu từ DataFrame df sau khi được phân tích xử lý có thể được lưu vào các file với các định dạng khác nhau.

- Đọc dữ liệu vào df: df=pd.read_csv('FoodPrice_in_Turkey.csv‘)
- Ghi dữ liệu từ DataFrame vào file csv: df.to_csv('demo_FoodPrice.csv')
- Ghi dữ liệu từ DataFrame vào file excel: df.to_excel('demo_FoodPrice.xlsx')
- Ghi dữ liệu từ DataFrame vào file Json dưới dạng cột:

  df.to_json('demo_FoodPrice.json',orient='columns')
- Ghi dữ liệu từ DataFrame vào file hdf5 dưới dạng bảng:

  df.to_hdf('demo_FoodPrice.h5', 'table')
- Ghi dữ liệu từ DataFrame vào file html: df.to_html('demo_FoodPrice.html')

Mô tả cấu trúc bộ dữ liệu DataFrame

Khi lần đầu tiên nhìn vào một tập dữ liệu, chúng ta muốn biết có bao nhiêu hàng, bao nhiêu cột, tên các cột và kiểu dữ liệu của chúng để hiểu hơn về bộ dữ liệu và chuẩn bị cho các bước phân tích xử lý tiếp theo.

- Xác định số dòng và số cột của df: df.shape
- Xác định số dòng: df.shape[0]
- Xác định số cột: df.shape[1]
- Xác định tên các cột thuộc tính: df.columns
- Xác định kiểu dữ liệu các cột thuộc tính: df.dtypes hoặc df.info()

Trong đó: object – kiểu chuỗi ký tự; int64 – kiểu số nguyên; float64 – kiểu số thực; datetime64 – kiểu ngày giờ.

Chỉnh sửa cấu trúc bộ dữ liệu

Chúng ta có thể đổi tên các cột thuộc tính và thêm bớt các cột, các dòng dữ liệu với các điều kiện khác nhau.

- Đổi tên cột thuộc tính ta sử dụng phương thức rename theo cú pháp:

  df.rename(columns={‘Tên cũ 1':‘Tên mới 1‘,...,‘Tên cũ n':'Tên mới n'},inplace=True)
- Thêm một cột mới với tất cả các bản ghi nhận giá trị rỗng NaN.

  df[‘Tên cột mới'] = 'NaN‘
- Thêm một cột mới với tất cả các bản ghi cùng nhận một giá trị cụ thể.

  Cách 1: df['Tên cột mới']= pd.Series(‘Giá trị cụ thể', index=df.index)

  Cách 2: df.insert(Vị trí muốn chèn,’Tên cột mới',pd.Series(‘Giá trị cụ thể', index=df.index))
- Thêm một dòng mới vào cuối DataFrame với các giá trị tương ứng với các cột được ngăn cách nhau bởi dấu phẩy.

  df = df.append({‘Cột1':‘Giá trị1',‘Cột2':‘Giá trị2‘,...,’Cộtn':‘Giá trịn'}, ignore_index = True)

Chỉnh sửa cấu trúc bộ dữ liệu (tiếp)

- Xóa một cột theo tên cột sử dụng hàm del:

  del df[‘Tên cột cần xóa’]

- Xóa một cột sử dụng phương thức pop(), kết quả trả về cột đã xóa:

  df.pop(‘Tên cột cần xóa‘)

- Xóa một cột sử dụng phương thức drop()

  df.drop('Tên cột cần xóa’, axis=1, inplace=True)

- Xóa nhiều cột sử dụng phương thức drop(), danh sách các cột được để trong một list:

  df.drop(['Tên cột cần xóa1’,..., 'Tên cột cần xóa n’], axis=1, inplace=True)

- Xóa một dòng có chỉ số cs sử dụng phương thức drop()

  df.drop(cs, axis=0, inplace=True)

- Xóa nhiều dòng sử dụng phương thức drop(), chỉ số các dòng được để trong một list:

  df.drop([cs1,...,csn], axis=0, inplace=True)

Truy cập dữ liệu

Trong Pandas, có 2 phương thức chính để truy cập dữ liệu:

- .iloc giúp truy cập hàng và cột thông qua các chỉ số của chúng. Tham số truyền vào là một số nguyên hoặc một tập các số nguyên. Cú pháp: .iloc[,].
- .loc giúp truy cập tới hàng và cột thông qua các chỉ số của hàng; tên hoặc biểu thức điều kiện của cột. Cú pháp: .loc[,]

Làm việc với .iloc

- Truy cập tới 1 dòng có chỉ số cs của df: df.iloc[cs]
- Truy cập tới nhiều dòng rời rạc: df.iloc[[cs1,...,csn]]
- Truy cập tới nhiều dòng liên tiếp nhau: df.iloc[cs_đầu : cs_cuối]
- Truy cập tới 1 cột có chỉ số cs: df.iloc[: , cs]
- Truy cập tới nhiều cột rời rạc: df.iloc[: , [cs1,...,csn]]
- Truy cập tới nhiều cột liên tiếp nhau: df.iloc[: , cs_đầu : cs_cuối]
- Truy cập tới các dòng, các cột rời rạc: df.iloc[[csd1,...,csdn] , [csc1,...,cscn]]
- Truy cập tới các dòng, các cột liên tiếp: df.iloc[csd_đầu : csd_cuối , csc_đầu : csc_cuối]
- Truy cập tới phần tử tại dòng csd cột csc: df.iloc[csd , csc]

Làm việc với .loc

- Truy cập tới 1 dòng có chỉ số cs của df: df.loc[cs]
- Truy cập tới nhiều dòng liên tiếp nhau: df.loc[cs_đầu : cs_cuối]
- Truy cập tới 1 cột có tên tc: df.loc[: , 'tc']
- Truy cập tới nhiều cột rời rạc có tên tc1,...,tcn df.loc[: , ['tc1',...,'tcn']]
- Truy cập tới phần tử tại dòng có chỉ số cs cột có tên tc: df.loc[cs , 'tc']
- Truy cập tới phần tử tại cột tên tc thỏa mãn điều kiện dk:

  df.loc[df.tc thỏa mãn điều kiện dk]

Thay thế giá trị các ô dữ liệu

- Thay các giá trị từ giá trị cũ gtc thành giá trị mới gtm trong toàn bộ tập dữ liệu:

  df.replace(gtc, gtm, inplace = True)

- Thay các giá trị từ giá trị cũ gtc thành giá trị mới gtm của cột có tên tc:

  df['tc'].replace(gtc, gtm, inplace = True)

Giá trị thống kê mô tả của biến định lượng

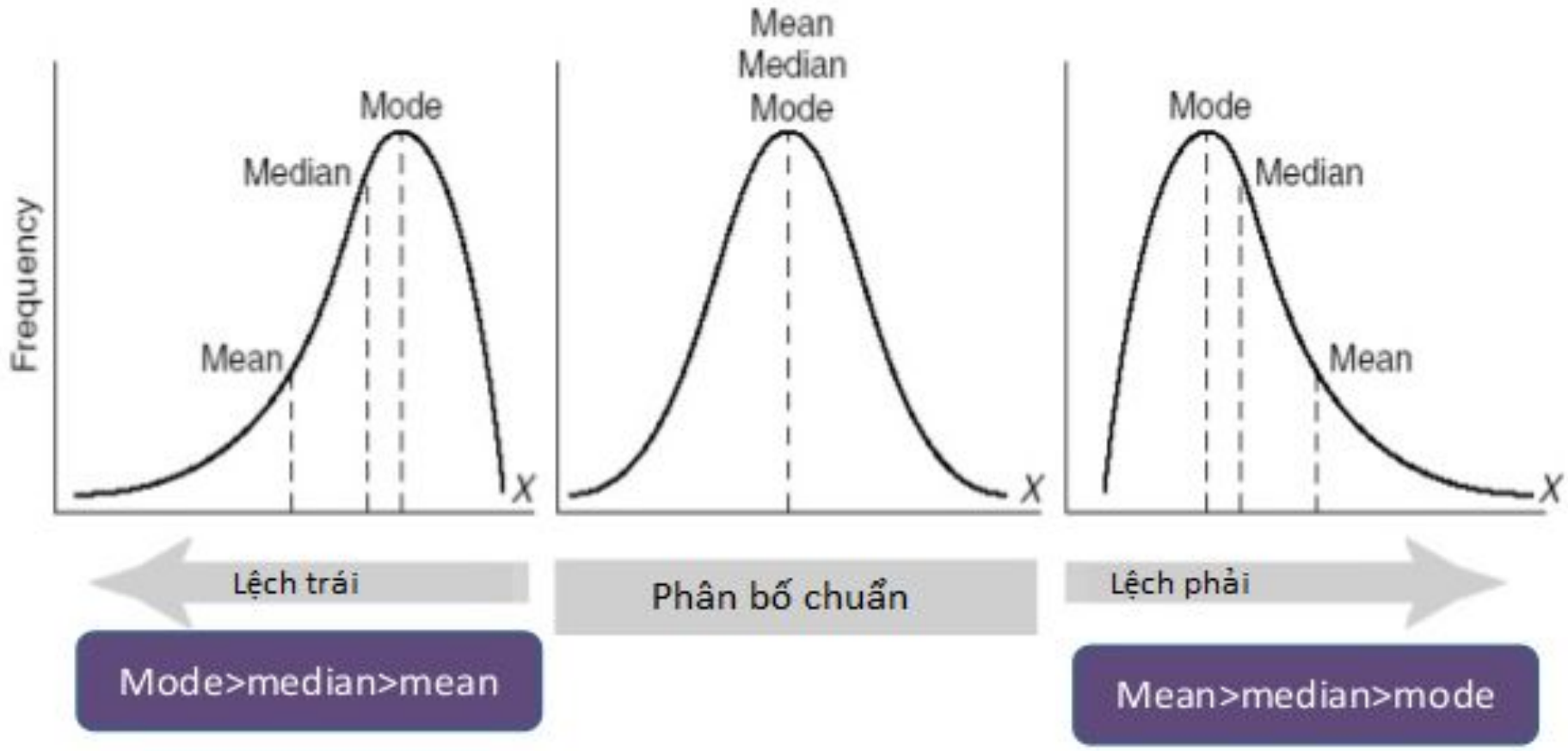
Thống kê là một phần rất quan trọng trong việc hiểu dữ liệu của bài toán từ đó đưa ra các nhận định và các kỹ thuật xử lý phù hợp và hiệu quả. Biến định lượng là các biến được biểu diễn bởi các con số.

- Đếm tần suất xuất hiện của biến định lượng df_dl: df_dl.count()
- Giá trị nhỏ nhất của biến định lượng df_dl: df_dl.min()
- Giá trị lớn nhất của biến định lượng df_dl: df_dl.max()
- Giá trị xuất hiện nhiều nhất của biến định lượng df_dl: df_dl.mode()
- Giá trị trung vị của biến định lượng df_dl: df_dl.median()
- Giá trị trung bình của biến định lượng df_dl: df_dl.mode()
- Tính giá trị thống kê của tất cả các biến định lượng trong DataFrame df:

  df.describe()

Giá trị thống kê mô tả của biến định lượng

Mối quan hệ giữa ba đại lượng mode, median, mean cho chúng ta biết hình dạng phân phối của tập dữ liệu.



Giá trị thống kê mô tả của biến định tính

Biến định tính mô tả các giá trị có thể quan sát được của một đại lượng nào đó không thể đo lường bằng các con số. Các giá trị thống kê mô tả của biến định tính df_dt gồm:

- Đếm tần suất xuất hiện của biến định tính df_dt: df_dt.count()
- Giá trị có độ dài nhỏ nhất của biến định tính df_dt: df_dt.min()
- Giá trị có độ dài lớn nhất của biến định tính df_dt: df_dt.max()
- Giá trị xuất hiện nhiều nhất của biến tính df_dt: df_dt.mode()

Hợp nhất dữ liệu từ nhiều nguồn

- Trong thực tế chúng ta cần phân tích bài toán mà các thông tin có thể được lưu trữ ở nhiều nguồn dữ liệu khác nhau. Pandas cung cấp các phương thức cho phép hợp nhất dữ liệu từ nhiều DataFrame khác nhau.
- Hợp nhất các cột có thể được thực hiện bằng phương thức concat, merge
- Hợp nhất các dòng có thể được thực hiện bằng phương thức concat, append
- Trong trường hợp số dòng, số cột của các DataFrame không bằng nhau, giá trị NaN sẽ được điền vào các ô thiếu dữ liệu.

Hợp nhất dữ liệu từ nhiều nguồn

Hợp nhất các cột:

- Hợp nhất 2 dataFrame có cùng chung một cột thuộc tính, DataFrame mới được tạo ra gồm các cột thuộc tính riêng và cột thuộc tính chung.

  pd.merge(dataFram1,dataFram2,on='Tên thuộc tính chung')
- Hợp nhất các dataFrame không cần cột thuộc tính chung, dataFrame mới được tạo ra bao gồm tất cả các cột thuộc tính của các dataFrame cần ghép.

  pd.concat([dataFram1,dataFram2,...], axis=1)

Hợp nhất các dòng:

- Hợp nhất các dataFrame lại với nhau, dataFrame mới được tạo ra bao gồm tất cả các dòng của các dataFrame cần ghép.

  pd.concat([dataFram1,dataFram2,...], axis=0)
- Hợp nhất 2 dataFrame, dataFram2 được chèn vào cuối dataFram1:

  dataFram1.append(dataFram2)

Bảng tổng hợp pivot table

Pivot table là một bảng tổng hợp dữ liệu, là công cụ hữu ích để tổng hợp, lọc và phân tích dữ liệu một cách dễ dàng, nhanh chóng.

- Cú pháp của pivot table gồm:

  dataFrame.pivot_table(values='a', index='b', columns='c', aggfunc='hàm')
- Kết quả tạo ra một bảng trong đó các giá trị của thuộc tính b nằm trên các hàng, các giá trị của thuộc tính c nằm trên các cột và các ô giao nhau là các giá trị của a được tính theo các hàm trong aggfunc (các ô nhận giá trị NaN nếu dữ liệu bị thiếu).
- Aggfunc có thể chứa các hàm: mean, sum, max, min

Tóm tắt

Qua bài học này, chúng ta đã nắm được các vấn đề cơ bản về vai trò của Pandas, cách khai báo thư viện Pandas và đối tượng quan trọng nhất của Pandas là DataFrame thông qua một số các chức năng chính của DataFrame như:

- cách tạo DataFrame;
- hiểu và chỉnh sửa được cấu trúc bộ dữ liệu trong DataFrame;
- Truy cập và chỉnh sửa các giá trị trong DataFrame;
- Hợp nhất dữ liệu từ nhiều DataFrame;
- Phân tích tổng hợp được các dữ liệu trong DataFrame...