Thực hành: Pandas (phần 1)

Khám phá dữ liệu

- *Pandas* là một package rất hiệu quả khi làm việc với dữ liệu dạng bảng. Nó cho phép chúng ta thực hiện các phép biến đổi và thống kê trên dữ liệu dạng bảng với tốc độ rất nhanh.

- *Data Frame*: Khi sử dụng thư viện pandas, một Data Frame là một cấu trúc dữ liệu dạng bảng hai chiều (hàng và cột), tương tự như bảng tính Excel hoặc bảng trong cơ sở dữ liệu. Nó được sử dụng để lưu trữ và thao tác dữ liệu có cấu trúc.

- *Series*: là mảng 1 chiều bao gồm một danh sách giá trị, và một mảng chứa index của các giá trị. Trong dữ liệu dạng bảng, mỗi Series được xem như là một cột của bảng đó.

1. Thư viện sử dụng

```
import pandas as pd
```

2. Đọc dữ liệu trong Pandas Python

(1). Đọc dữ liệu từ một le CSV

```
df = pd.read_csv(filepath, sep=',', names=NoDefault.no_default,
index_col=None, header=None)
```

Trong đó:

- `filepath` là đường dẫn đến file trong máy hoặc đường link URL
- `sep` dùng để nhận diện cách chia thành cột, nếu không truyền tham số này thì pandas tự hiểu là chia theo ','. Ngoài ra có thể chia theo các ký hiệu khác, ví dụ: `sep = "\t"`
- `names` là tên các cột của bảng. Nếu bảng đã có tên cột thì nên bỏ qua tham số này, pandas sẽ lấy dòng đầu tiên của file làm tên cột, nếu ko sử dụng tên cột thì để `header=None.`
- `index_col` dùng để chỉ định vị trí các cột dùng để làm index cho bảng. ví dụ: index_col="ID" (với ID là tên cột)

(2). Đọc dữ liệu từ một le TSV

```
df = pd.read_table(filename)
```

(3). Đọc dữ liệu từ một le Excel

```
df = pd.read_excel(filename)
```

hoặc:

```
x = pd.ExcelFile(filename)
```

Với *filename* là đường dẫn đến file

Để xem tên các sheets của *x* ta có thể dùng *x.sheet_names*. Sau đó để đọc từng sheet của x ta có thể dùng *.parse()*

```
df = x.parse(sheet_name)
```

Trong đó *sheet_name* là tên sheet cần đọc, các thông số còn lại tương tự như phần đọc file csv.

3. Xuất dữ liệu từ Pandas DataFrame

(1). Xuất dữ liệu từ DataFrame ra le CSV

```
df.to_csv(filename)
```

(2). Xuất dữ liệu từ DataFrame ra le Excel

```
df.to_excel(filename)
```

4. Các hàm và thuộc tính cơ bản trong pandas

- `df.head()`: Trả về 5 hàng đầu tiên của Data Frame (có thể thay đổi số lượng bằng cách truyền tham số, ví dụ: `df.head(10))`.

- `df.tail(n=8)`: Trả về 8 hàng đầu tiên của Data Frame

- `df.info()`: Cung cấp thông tin tổng quan về Data Frame, bao gồm số lượng hàng, cột, kiểu dữ liệu của từng cột và số giá trị không rỗng.

- `df.describe()`: Tạo thống kê mô tả (số lượng, trung bình, độ lệch chuẩn, giá trị min/max, phân vị) cho các cột số trong Data Frame.

- `df.shape`: Thuộc tính trả về một tuple biểu thị kích thước của Data Frame (số hàng, số cột). Ví dụ: (100, 5) nghĩa là 100 hàng và 5 cột.

- `df.size`: Cho biết số lượng phần tử của của Data Frame

- `df.columns`: Thuộc tính trả về danh sách tên các cột trong Data Frame.

- `df.dtypes`: Thuộc tính trả về kiểu dữ liệu của từng cột (ví dụ: int64, float64, object)

5. Truy cập dữ liệu/ lọc dữ liệu trong Data Frame

(1). Sử dụng [<tên cột>]

để lấy 1 Series của bảng, ví dụ:

```
df.gia_xe     # gia_xe là tên cột không chứa khoảng trắng
df['gia_xe']  # gia_xe là tên cột
df[['state', 'family_members']]  # chọn 1 bảng con có 2 cột ['state',
'family_members']
```

Lưu ý: `df[['state']]` sẽ trả về DataFrame trong khi `df['state']` trả về Series

(2). Sử dụng `.loc[]`

Truy cập dữ liệu bằng nhãn (label-based). Ví dụ:

```
df.loc[0, 'ten_cot']   #truy cập giá trị tại hàng 0, cột 'ten_cot '
df.loc['Pacific']
df.loc['Pacific', 'state']
df.loc['Pacific', 'individuals':'state_pop']
```

(3). Sử dụng `.iloc[]`

Truy cập dữ liệu bằng chỉ số index (integer-based). Ví dụ:

```
df. iloc [0, 1]   #truy cập giá trị tại hàng 0, cột 1
df.iloc[0]  # Khi truyền 1 giá trị nguyên, .iloc trả về giá trị của dòng
            # tại vị trí truyền vào với kiểu Series
df.iloc[:, 2]     # lấy dữ liệu theo cột có index cột là 2
```

Lưu ý: Sử dụng .columns.get_loc(<tên cột>) để lấy vị trí của cột

(4). Sử dụng .at[]

Truy cập một giá trị đơn bằng nhãn, nhanh hơn .loc[] cho truy cập đơn lẻ. Ví dụ:

```
df.at [0, ’ten_cot ’]
```

(5). Sử dụng .iat[]

Truy cập một giá trị đơn bằng chỉ số index, nhanh hơn .iloc[]. Ví dụ:

```
df. iat [0, 1]
```

(6). Lọc các hàng trong Data Frame thỏa mãn điều kiện

Ví dụ:

```
df[df[‘tuoi’] > 25] # lọc các hàng có giá trị cột ’tuoi ’ lớn hơn 25)
df.loc[df.ten_col1==15] #lọc ra các hàng có ten_col1==15
df.loc[(df.ten_col1==15)|(df.ten_col2== “h”)]   #lọc ra các hàng có
                                # ten_col1==15 hoặc ten_col2== “h”
df.loc[(df['individuals'] > 5000) & (df['individuals'] < 10000)]
```

6. Thêm, xóa, sửa cột trong Data Frame

- Thêm/sửa cột: Có thể tạo hoặc sửa cột mới bằng cách gán giá trị. Ví dụ:

```
# tạo cột mới bằng tổng của hai cột
df[‘cot_moi’] = df[‘cot1’] + df[‘cot2’]
```

- Xóa cột: Sử dụng df.drop().

```
df = df.drop(labels=None, axis=0) #xóa các dòng hoặc cột theo chỉ định
```

Trong đó:

- `labels:` Tên cột hoặc dòng cần xóa.
- `axis:` Mặc đinh giá trị 0 xóa theo dòng và 1 xóa theo cột.

```
# xóa cột ‘Humidity’
df = df.drop(columns= {'Humidity'})
```

hoặc

```
df = df.drop(‘Humidity’, axis=1)
```

Lưu ý: Tham số `inplace=True` có thể được sử dụng để sửa trực tiếp Data Frame mà không cần gán lại.

7. Một số thao tác khác trên Series/ DataFrame

(1). .value_counts()

Phương thức này trả số lần xuất hiện của các phần tử trong Series. Kết quả trả về mặc định sẽ sắp xếp theo số lần xuất hiện giảm dần và mặc định bỏ qua các giá trị null

(2). .unique() và .nunique()

Phương thức `.unique()` trả về các giá trị khác nhau của Series và .nunique() trả về số lượng các giá trị khác nhau của Series.

(3). .isnull()

(4). .notnull()

(5). .mean()

(6). .corr()

(7). .max()

(8). .min()

(9). .median()

(10). .std()

Bài tập

1. Tải các file dữ liệu

2. Đọc các file trên thành dataframe

3. Thực hiện các thao tác trên, quan sát và hiểu các lệnh đó.