Pandas 数据结构

学习目标

Learning Objectives

1. 掌握Series的常用属性及方法
2. 掌握DataFrame的常用属性及方法
3. 掌握更改Series和DataFrame的方法
4. 掌握如何导入导出数据

目录

Contents

1.1 创建Series

1. DataFrame和Series是Pandas最基本的两种数据结构
2. 在Pandas中，Series是一维容器，Series表示DataFrame的每一列

   可以把DataFrame看作由Series对象组成的字典，其中key是列名，值是Series

   Series和Python中的列表非常相似，但是它的每个元素的数据类型必须相同

1.1 创建Series

3. 创建 Series 的最简单方法是传入一个Python列表

```
import pandas as pd
s = pd.Series(['banana',42])
print(s)
```

输出结果

```
0    banana
1        42
dtype: object
```

1.1 创建Series

4. 创建Series时，可以通过index参数 来指定行索引

```
s = pd.Series(['Wes McKinney','Male'],index = ['Name','Gender'])
print(s)
```

输出结果

```
Name      Wes McKinney
Gender            Male
dtype: object
```

1.2 创建 DataFrame

1. 可以使用字典来创建DataFrame

```
name_list = pd.DataFrame(
    {'Name':['Tom','Bob'],
     'Occupation':['Teacher','IT Engineer'],
     'age':[28,36]})
print(name_list)
```

输出结果

```
Name   Occupation  age
0  Tom      Teacher   28
1   Bob  IT Engineer   36
```

1.2 创建 DataFrame

2. 创建DataFrame的时指定列的顺序和行索引

```
name_list = pd.DataFrame(
    data = {'Occupation':['Teacher','IT Engineer'],'Age':[28,36]},
    columns=['Age','Occupation'],index=['Tom','Bob'])
print(name_list)
```

输出结果

```
Age   Occupation
Tom   28        Teacher
Bob    36   IT Engineer
```

目录

Contents

2.1 Series常用属性

1. 加载CSV文件

```
data = pd.read_csv('data/nobel_prizes.csv',index_col='id')
data.head()
```

输出结果

id	year	category	overallMotivation	firstname	surname	motivation	share
941	2017	physics	NaN	Rainer	Weiss	"for decisive contributions to the LIGO detect...	2
942	2017	physics	NaN	Barry C.	Barish	"for decisive contributions to the LIGO detect...	4
943	2017	physics	NaN	Kip S.	Thorne	"for decisive contributions to the LIGO detect...	4

2.1 Series常用属性

1. 使用 DataFrame的loc 属性获取数据集里的一行，就会得到一个Series对象

```
first_row = data.loc[941]
type(first_row)
```

输出结果

```
pandas.core.series.Series
```

```
print(first_row)
```

输出结果

```
year                                                        2017
category                                                 physics
overallMotivation                                            NaN
firstname                                                 Rainer
surname                                                    Weiss
motivation          "for decisive contributions to the LIGO detect...
share                                                          2
Name: 941, dtype: object
```

2.1 Series常用属性

2. 可以通过 index 和 values属性获取行索引和值

```
first_row.index
```

输出结果

```
Index(['year', 'category', 'overallMotivation', 'firstname', 'surname',
  'motivation', 'share'],
 dtype='object')
```

```
print(first_row.values)
```

输出结果

```
[2017 'physics' nan 'Rainer' 'Weiss'
'"for decisive contributions to the LIGO detector and the observation of
gravitational waves"'
2]
```

2.1 Series常用属性

3. Series的keys方法，作用个index属性一样

```
data.keys()
```

输出结果

```
Index(['year', 'category', 'overallMotivation', 'firstname', 'surname',
  'motivation', 'share'],
 dtype='object')
```

2.1 Series常用属性

4. Series的一些属性

属性	说明
loc	使用索引值取子集
iloc	使用索引位置取子集
dtype或dtypes	Series内容的类型
T	Series的转置矩阵
shape	数据的维数
size	Series中元素的数量
values	Series的值

2.2 Series常用方法

1. 针对数值型的Series，可以进行常见计算

```
share = data.share  # 从DataFrame中 获取Share列（几人获奖）返回Series
share.mean()       #计算几人获奖的平均值
```

输出结果

```
1.982665222101842
```

```
share.max() # 计算最大值
```

输出结果

```
4
```

```
share.min() # 计算最小值
```

输出结果

```
1
```

```
share.std() # 计算标准差
```

输出结果

```
0.9324952202244597
```

2.2 Series常用方法

2. 通过value_counts()方法，可以返回不同值的条目数量

```
movie = pd.read_csv('data/movie.csv')     # 加载电影数据
director = movie['director_name']    # 从电影数据中获取导演名字 返回Series
actor_1_fb_likes = movie['actor_1_facebook_likes'] # 从电影数据中取出主演的facebook点赞数
director.head()  #查看导演Series数据
```

输出结果

```
0         James Cameron
1        Gore Verbinski
2            Sam Mendes
3     Christopher Nolan
4           Doug Walker
Name: director_name, dtype: object
```

2.2 Series常用方法

2. 通过value_counts()方法，可以返回不同值的条目数量

```
actor_1_fb_likes.head() #查看主演的facebook点赞数
```

输出结果

```
0     1000.0
1    40000.0
2    11000.0
3    27000.0
4      131.0
Name: actor_1_facebook_likes, dtype: float64
```

2.2 Series常用方法

2. 通过value_counts()方法，可以返回不同值的条目数量

```
director.value_counts()       # 统计不同导演指导的电影数量
```

输出结果

```
Steven Spielberg    26
Woody Allen         22
Clint Eastwood      20
Martin Scorsese     20
                    ..
Gavin Wiesen         1
Andrew Morahan       1
Luca Guadagnino      1
Richard Montoya      1
Name: director_name, Length: 2397, dtype: int64
```

2.2 Series常用方法

2. 通过value_counts()方法，可以返回不同值的条目数量

```
actor_1_fb_likes.value_counts()
```

输出结果

```
1000.0     436
11000.0    206
2000.0     189
3000.0     150
      ...
216.0        1
859.0        1
225.0        1
334.0        1
Name: actor_1_facebook_likes, Length: 877, dtype: int64
```

2.2 Series常用方法

3. 通过count()方法可以返回有多少非空值

```
director.count()
```

输出结果

```
4814
```

对比全部数据量

```
director.shape
```

输出结果

```
(4916,)
```

2.2 Series常用方法

4. 通过describe()方法打印描述信息

```
actor_1_fb_likes.describe()
```

输出结果

```
count      4909.000000
mean       6494.488491
std       15106.986884
min           0.000000
25%         607.000000
50%         982.000000
75%       11000.000000
max      640000.000000
Name: actor_1_facebook_likes, dtype: float64
```

2.2 Series常用方法

4. 通过describe()方法打印描述信息

```
director.describe()
```

输出结果

```
count                  4814
unique                 2397
top       Steven Spielberg
freq                     26
Name: director_name, dtype: object
```

2.2 Series常用方法

5. Series的一些方法

方法	说明
append	连接两个或多个Series
corr	计算与另一个Series的相关系数
cov	计算与另一个Series的协方差
describe	计算常见统计量
drop_duplicates	返回去重之后的Series
equals	判断两个Series是否相同
get_values	获取Series的值，作用与values属性相同
hist	绘制直方图
isin	Series中是否包含某些值
min	返回最小值

2.2 Series常用方法

5. Series的一些方法

方法	说明
max	返回最大值
mean	返回算术平均值
median	返回中位数
mode	返回众数
quantile	返回指定位置的分位数
replace	用指定值代替Series中的值
sample	返回Series的随机采样值
sort_values	对值进行排序
to_frame	把Series转换为DataFrame
unique	去重返回数组

2.3 Series的布尔索引

1. 从Series中获取满足某些条件的数据，可以使用布尔索引

```
scientists = pd.read_csv('data/scientists.csv')
print(scientists)
```

输出结果

	Name	Born	Died	Age	Occupation
0	Rosaline Franklin	1920-07-25	1958-04-16	37	Chemist
1	William Gosset	1876-06-13	1937-10-16	61	Statistician
2	Florence Nightingale	1820-05-12	1910-08-13	90	Nurse
3	Marie Curie	1867-11-07	1934-07-04	66	Chemist
4	Rachel Carson	1907-05-27	1964-04-14	56	Biologist
5	John Snow	1813-03-15	1858-06-16	45	Physician
6	Alan Turing	1912-06-23	1954-06-07	41	Computer Scientist
7	Johann Gauss	1777-04-30	1855-02-23	77	Mathematician

2.3 Series的布尔索引

2. 手动创建布尔值列表

```
#手动创建布尔值列表
bool_values = [False,True,True,False,False,False,False,False]
ages[bool_values]
```

输出结果

```
1    61
2    90
Name: Age, dtype: int64
```

2.3 Series的布尔索引

3. 筛选年龄大于平均年龄的科学家

```
ages = scientists['Age']
ages.mean()
```

输出结果

```
59.125
```

```
print(ages[ages>ages.mean()])
```

输出结果

```
1    61
2    90
3    66
7    77
Name: Age, dtype: int64
```

```
print(ages>ages.mean())
```

输出结果

```
0    False
1     True
2     True
3     True
4    False
5    False
6    False
7     True
Name: Age, dtype: bool
```

2.4 Series 的运算

1. Series和数值型变量计算时，变量会与Series中的每个元素逐一进行计算

```
ages+100
```

输出结果

```
0    137
1    161
2    190
3    166
4    156
5    145
6    141
7    177
Name: Age, dtype: int64
```

2.4 Series 的运算

1. Series和数值型变量计算时，变量会与Series中的每个元素逐一进行计算

```
ages*2
```

输出结果

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

2.4 Series 的运算

2. 两个Series之间计算，如果Series元素个数相同，则将两个Series对应元素进行计算

```
ages+ages
```

输出结果

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

2.4 Series 的运算

3. 元素个数不同的Series之间进行计算，会根据索引进行。

索引不同的元素最终计算的结果会填充成缺失值，用NaN表示

```
ages * pd.Series([1,100])
```

输出结果

```
0      37.0
1    6100.0
2       NaN
3       NaN
4       NaN
5       NaN
6       NaN
7       NaN
dtype: float64
```

2.4 Series 的运算

4. Series之间进行计算时，数据会尽可能依据索引标签进行相互计算

```
print(ages+rev_ages)
```

输出结果

```
0     74
1    122
2    180
3    132
4    112
5     90
6     82
7    154
Name: Age, dtype: int64
```

目录

Contents

3.1 DataFrame的常用属性和方法

1. DataFrame是Pandas中最常见的对象，

   Series数据结构的许多属性和方法在DataFrame中也一样适用

```
movie = pd.read_csv('data/movie.csv')
 # 打印行数和列数
movie.shape
```

输出结果

```
(4916, 28)
```

```
# 打印数据的个数
movie.size
```

输出结果

```
137648
```

```
# 该数据集的维度
movie.ndim
```

输出结果

```
2
```

3.1 DataFrame的常用属性和方法

```
# 该数据集的长度
len(movie)
```

输出结果

```
4916
```

```
# 各个列的值的个数
movie.count()
```

输出结果

```
color                      4897
director_name              4814
num_critic_for_reviews     4867
duration                   4901
                           ...
actor_2_facebook_likes     4903
imdb_score                 4916
aspect_ratio               4590
movie_facebook_likes       4916
Length: 28, dtype: int64
```

3.1 DataFrame的常用属性和方法

```
# 各列的最小值
movie.min()
```

输出结果

```
num_critic_for_reviews     1.00
duration                   7.00
director_facebook_likes    0.00
actor_3_facebook_likes     0.00
                           ...
actor_2_facebook_likes     0.00
imdb_score                 1.60
aspect_ratio               1.18
movie_facebook_likes       0.00
Length: 16, dtype: float64
```

3.1 DataFrame的常用属性和方法

```
movie.describe()
```

输出结果

	num_critic_for_reviews	duration	director_facebook_likes
count	4867.000000	4901.000000	4814.000000
mean	137.988905	107.090798	691.014541
std	120.239379	25.286015	2832.954125
min	1.000000	7.000000	0.000000
25%	49.000000	93.000000	7.000000
50%	108.000000	103.000000	48.000000
75%	191.000000	118.000000	189.750000
max	813.000000	511.000000	23000.000000

3.2 DataFrame的布尔索引

1. 同Series一样，DataFrame也可以使用布尔索引获取数据子集。

```
# 使用布尔索引获取部分数据行
movie[movie['duration']>movie['duration'].mean()]
```

输出结果

	color	director_name	num_critic_for_reviews	duration	director_facebook_l
0	Color	James Cameron	723.0	178.0	
1	Color	Gore Verbinski	302.0	169.0	5
2	Color	Sam Mendes	602.0	148.0	
3	Color	Christopher Nolan	813.0	164.0	220

3.2 DataFrame的布尔索引

2. 可以传入布尔值的列表，来获取部分数据，True所对应的数据会被保留

```
movie.head()[[True,True,False,True,False]]
```

输出结果

color	director_name	num_critic_for_reviews	duration	director_facebook_likes	a
0	Color	James Cameron	723.0	178.0	
1	Color	Gore Verbinski	302.0	169.0	
3	Color	Christopher Nolan	813.0	164.0	

3 rows × 28 columns

3.3 DataFrame的运算

1. 当DataFrame和数值进行运算时，DataFrame中的每一个元素会分别和数值进行运算

```
scientists*2
```

2. 两个DataFrame之间进行计算，会根据索引进行对应计算

```
scientists+scientists
```

3. 两个DataFrame会根据索引进行计算，索引不匹配的会返回NaN

```
first_half = scientists[:4]
scientists+first_half
```

目录

Contents

4.1 给行索引命名

1. 通过set_index()方法设置行索引名字

加载数据文件时，如果不指定行索引，Pandas会自动加上从0开始的索引

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

输出结果

	color	director_name	num_critic_for_reviews	duration	director_facebook_l
0	Color	James Cameron	723.0	178.0	
1	Color	Gore Verbinski	302.0	169.0	5
2	Color	Sam Mendes	602.0	148.0	
3	Color	Christopher Nolan	813.0	164.0	220

4.1 给行索引命名

1. 通过set_index()方法设置行索引名字

加载数据文件时，如果不指定行索引，Pandas会自动加上从0开始的索引

```
movie2 = movie.set_index('movie_title')
movie2
```

输出结果

movie_title	color	director_name	num_critic_for_reviews	duration	director_face
Avatar	Color	James Cameron	723.0	178.0	0.0
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0
Spectre	Color	Sam Mendes	602.0	148.0	0.0
The Dark Knight Rises	Color	Christopher Nolan	813.0	164.0	22000.0

4.1 给行索引命名

2. 加载数据的时候，可以通过通过index_col参数，指定使用某一列数据作为行索引

```
pd.read_csv('data/movie.csv', index_col='movie_title')
```

输出结果

movie_title	color	director_name	num_critic_for_reviews	duration	director_face
Avatar	Color	James Cameron	723.0	178.0	0.0
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0
Spectre	Color	Sam Mendes	602.0	148.0	0.0
The Dark Knight Rises	Color	Christopher Nolan	813.0	164.0	22000.0

4.1 给行索引命名

3. 通过reset_index()方法可以重置索引

```
movie2.reset_index()
```

输出结果

	movie_title	color	director_name	num_critic_for_reviews	duration	direct
0	Avatar	Color	James Cameron	723.0	178.0	
1	Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	
2	Spectre	Color	Sam Mendes	602.0	148.0	
3	The Dark Knight Rises	Color	Christopher Nolan	813.0	164.0	

4.2 DataFrame修改行名和列名

1. 通过rename()方法对原有的行索引名和列名进行修改

```
movie = pd.read_csv('data/movie.csv', index_col='movie_title')
movie.index[:5]
```

输出结果

```
Index(['Avatar', 'Pirates of the Caribbean: At World's End', 'Spectre',
  'The Dark Knight Rises', 'Star Wars: Episode VII - The Force Awakens'],
 dtype='object', name='movie_title')
```

```
movie.columns[:5]
```

输出结果

```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
  'director_facebook_likes'],
 dtype='object')
```

4.2 DataFrame修改行名和列名

1. 通过rename()方法对原有的行索引名和列名进行修改

```
idx_rename = {'Avatar':'Ratava', 'Spectre': 'Ertceps'}
col_rename = {'director_name':'Director Name', 'num_critic_for_reviews': 'Critical Reviews'}
movie.rename(index=idx_rename, columns=col_rename).head()
```

输出结果

movie_title	color	Director Name	Critical Reviews	duration	director_facebook_likes	acto
Ratava	Color	James Cameron	723.0	178.0	0.0	855.0
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0	1000.

4.2 DataFrame修改行名和列名

2. 将index 和 columns属性提取出来，修改之后，再赋值回去

输出结果

	color	Director Name	Critical Reviews	duration	director_facebook_likes	actor_
Ratava	Color	James Cameron	723.0	178.0	0.0	
Pirates of the Caribbean: At World's End	Color	Gore Verbinski	302.0	169.0	563.0	
Ertceps	Color	Sam Mendes	602.0	148.0	0.0	

movie.head()

4.3 添加、删除、插入列

1. 通过dataframe[列名]添加新列

```
movie = pd.read_csv('data/movie.csv')
movie['has_seen'] = 0
# 给新列赋值
movie['actor_director_facebook_likes'] = (movie['actor_1_facebook_likes'] +
                                          movie['actor_2_facebook_likes'] +
                                          movie['actor_3_facebook_likes'] +
                                          movie['director_facebook_likes'])
```

2. 调用drop方法删除列

```
movie = movie.drop('actor_director_facebook_likes', axis='columns')
```

4.3 添加、删除、插入列

3. 使用insert()方法插入列

loc 新插入的列在所有列中的位置（0,1,2,3...） column=列名 value=值

```
movie.insert(loc=0,column='profit',value=movie['gross'] - movie['budget'])
movie
```

输出结果

	profit	color	director_name	num_critic_for_reviews	duration	direc
0	523505847.0	Color	James Cameron	723.0	178.0	
1	9404152.0	Color	Gore Verbinski	302.0	169.0	
2	-44925825.0	Color	Sam Mendes	602.0	148.0	
3	198130642.0	Color	Christopher Nolan	813.0	164.0	

目录

Contents

5.1 pickle文件

1. pickle文件简介

如要保存的对象是计算的中间结果，或者保存的对象以后会在Python中复用，可把对象保存为.pickle文件

如果保存成pickle文件，只能在python中使用

文件的扩展名可以是.p，.pkl，.pickle

```
scientists = pd.read_csv('data/scientists.csv')
names = scientists['Name']
names.to_pickle('output/scientists_name.pickle')
scientists.to_pickle('output/scientists_df.pickle')
```

5.1 pickle文件

2. 读取pickle文件

可以使用pd.read_pickle函数读取.pickle文件中的数据

```
scientists_name = pd.read_pickle('output/scientists_name.pickle')
print(scientists_name)
```

输出结果

```
0       Rosaline Franklin
1          William Gosset
2    Florence Nightingale
3            Marie Curie
4          Rachel Carson
5              John Snow
6           Alan Turing
7          Johann Gauss
Name: Name, dtype: object
```

5.2 CSV文件

1. CSV文件介绍

CSV(逗号分隔值)是很灵活的一种数据存储格式

在CSV文件中，对于每一行，各列采用逗号分隔

除了逗号，还可以使用其他类型的分隔符，比如TSV文件，使用制表符作为分隔符

CSV是数据协作和共享的首选格式

```
names.to_csv('output/scientists_name.csv')
#设置分隔符为\t
scientists.to_csv('output/scientists_df.tsv',sep='\t')
```

不在csv文件中写行名

```
scientists.to_csv('output/scientists_df_noindex.csv',index=False)
```

5.3 Excel文件

1. 保存成Excel文件

Series这种数据结构不支持to_excel方法，想保存成Excel文件，需要把Series转换成DataFrame

```
names_df = names.to_frame()
import xlwt
names_df.to_excel('output/scientists_name_df.xls')
names_df.to_excel('output/scientists_name_df.xlsx')
```

把DataFrame保存为Excel格式

```
scientists.to_excel('output/scientists_df.xlsx',sheet_name='scientists',index=False)
```

5.3 Excel文件

2. 读取Excel文件

使用pd.read_excel() 读取Excel文件

```
pd.read_excel('output/scientists_df.xlsx')
```

#注意 pandas读写excel需要额外安装如下三个包

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple xlwt

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple openpyxl

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple xlrd

5.4 其它数据格式

1. feather文件

feather是一种文件格式，用于存储二进制对象

feather对象也可以加载到R语言中使用

feather格式的主要优点是在Python和R语言之间的读写速度要比CSV文件快

feather数据格式通常只用中间数据格式，用于Python和R之间传递数据

一般不用做保存最终数据

5.4 其它数据格式

2. 其他格式

导出方法	说明
to_clipboard	把数据保存到系统剪贴板，方便粘贴
to_dict	把数据转换成Python字典
to_hdf	把数据保存为HDF格式
to_html	把数据转换成HTML
to_json	把数据转换成JSON字符串
to_sql	把数据保存到SQL数据库

- 创建Series和DataFrame
  - pd.Series
  - pd.DataFrame
- Series常用操作
  - 常用属性
    - index
    - values
    - shape,size,dtype
  - 常用方法
    - max(),min(),std()
    - count()
    - describe()
  - 布尔索引
  - 运算
    - 与数值之间进行算数运算会对每一个元素进行计算
    - 两个Series之间进行计算会索引对齐
- DataFrame常用操作
  - 常用属性
  - 常用方法
  - 布尔索引
  - 运算
- DataFrame常用操作
  - 常用属性
  - 常用方法
  - 布尔索引
  - 运算
- 更改Series和DataFrame
  - 指定行索引名字
    - dataframe.set_index()
    - dataframe.reset_index()
  - 修改行/列名字
    - dataframe.rename(index=,columns =)
    - 获取行/列索引 转换成list之后，修改list再赋值回去
  - 添加、删除、插入列
    - 添加 dataframe['新列']
    - 删除 dataframe.drop
    - 插入列 dataframe.insert()
- 导入导出数据
  - pickle
  - csv
  - Excel
  - feather

传智教育旗下高端IT教育品牌