pandas库——矩阵运算

1.简介

Pandas是一个强大的分析结构化数据的工具集；它的使用基础是Numpy（提供高性能的矩阵运算）；用于数据挖掘和数据分析，同时也提供数据清洗功能。

主要提供Series和DataFrame这两种数据结构及其系列操作，

2.Series

它是一种类似于一维数组的对象，是由一组数据(各种NumPy数据类型)以及一组与之相关的数据标签(即索引)组成。仅由一组数据也可产生简单的Series对象。

2.1 创建Series

- 类似一维数组，是一种带有索引的序列。可以通过列表和字典创建，不用处理NaN(None)
- obj=pd.Series(obj,index=[])
- 获取索引：obj.index
- 获取数据：obj.values
- 预览数据：obj.head(n)
- 索引命名：obj.index.name=''
- 数据命名：obj.name=''

2.2 数据访问

- 判断索引是否存在：in
- 通过整型索引：iloc[int] 多个索引是用list
- 通过字符串索引：loc['']
- 以上两种方式也可以直接访问：[]

向量化操作：通过numpy的一些方法进行计算，速度会优于循环处理

3 DataFrame

DataFrame是Pandas中的一个表格型的数据结构，包含有一组有序的列，每列可以是不同的值类型(数值、字符串、布尔型等)，DataFrame即有行索引也有列索引，可以被看做是由Series组成的字典。

3.1 创建DataFrame

- 类似于多维数组/表格数据，**每列数据可以是不同的类型。**可以通过字典创建Series类型的列表创建，也可以读取数据文件
- 索引包括行索引（index）和列索引（label）
- pd.DataFrame(obj,index=[])
- 由Series列表创建

```
import pandas as pd
from faker import Faker

faker=Faker('zh-CN')
country1=pd.Series({'name':faker.name(),'job':faker.job(),'ID':faker.ssn()})
country2=pd.Series({'name':faker.name(),'job':faker.job(),'ID':faker.ssn()})
country3=pd.Series({'name':faker.name(),'job':faker.job(),'ID':faker.ssn()})
country4=pd.Series({'name':faker.name(),'job':faker.job(),'ID':faker.ssn()})
df=pd.DataFrame([country1,country2,country3,country4],index=
['c1','c2','c3','c4'])
```

	name	job	ID
c1	袁娟	西班牙语翻译	360502198012272491
c2	李刚	电脑维修	360981197911307232
c3	刘伟	营运主管	620702195704186659
c4	高辉	管道/暖通	330303195307282327

- 由文件读入
  - pd.**read_csv**('.csv/txt',index_col='',usecols=[],header=None,delimiter=',',names=[],dtype={'列名':类型,...},skiprows=n,nrows=n)
  - index_col给行索引加列名
  - usecols指定读取哪些列
  - 默认会将第一列作为索引名，若数据中第一列为数据，并非索引名时，通过header=None/0
  - delimiter分隔符,亦可使用sep
  - names指定列名，需要header=None，即不通过第一行读取
  - dtype指定每列类型
  - skiprows指定跳过行
  - nrows指定读取行数

```
import pandas as pd
iris=pd.read_csv('iris_data.csv',header=None,names=['Sepal
Length','Sepal Width','Petal Length','Petal Width','type'])
iris.head(10)
```

	Sepal Length	Sepal Width	Petal Length	Petal Width	type
0	5.5	2.5	4.0	1.3	Iris-versicolor

	Sepal Length	Sepal Width	Petal Length	Petal Width	type
1	6.3	2.5	5.0	1.9	Iris-virginica
2	5.5	2.6	4.4	1.2	Iris-versicolor
3	6.0	2.2	4.0	1.0	Iris-versicolor
4	6.0	2.2	5.0	1.5	Iris-virginica
5	6.3	2.8	5.1	1.5	Iris-virginica
6	5.5	2.4	3.8	1.1	Iris-versicolor
7	6.7	3.3	5.7	2.1	Iris-virginica
8	6.7	3.1	4.4	1.4	Iris-versicolor
9	5.6	3.0	4.5	1.5	Iris-versicolor

- series的to_frame(name=None)操作
- 读取数据库：pd.read_sql(sql_sentence,connection)

3.2 数据访问

- 直接使用[]访问时从行获取，使用loc时从列获取，使用iloc时则通过整型索引从列获取，均为Series对象
- 获取行索引：df.index
- 行或列索引
  - df[]列索引
  - df.loc[]行索引
  - df.iloc[]行索引
- 混合索引
  - 先行索引后列索引
  - 先列索引后行索引

```
#先列后行
df['name']['c2']
df['name'].loc['c2']
df['name'].iloc[1]

#先行后列
df.loc['c2']['name']
df.loc['c2'].loc['name']
df.iloc[1]['name']
```

3.3 数据操作

- 属性
  - shape/dtypes/ndim/index/columns/values
- 预览数据：df.head(n)

	0	1	2	3	4
0	5.5	2.5	4.0	1.3	Iris-versicolor
1	6.3	2.5	5.0	1.9	Iris-virginica
2	5.5	2.6	4.4	1.2	Iris-versicolor
3	6.0	2.2	4.0	1.0	Iris-versicolor
4	6.0	2.2	5.0	1.5	Iris-virginica
5	6.3	2.8	5.1	1.5	Iris-virginica
6	5.5	2.4	3.8	1.1	Iris-versicolor
7	6.7	3.3	5.7	2.1	Iris-virginica
8	6.7	3.1	4.4	1.4	Iris-versicolor
9	5.6	3.0	4.5	1.5	Iris-versicolor

- 数据信息：df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 5 columns):
5.5                99 non-null float64
2.5                99 non-null float64
4.0                99 non-null float64
1.3                99 non-null float64
Iris-versicolor    99 non-null object
dtypes: float64(4), object(1)
memory usage: 3.9+ KB
```

- 统计信息：df.describe()

	0	1	2	3

	0	1	2	3
count	100.000000	100.000000	100.000000	100.000000
mean	6.262000	2.872000	4.906000	1.676000
std	0.662834	0.332751	0.825578	0.424769
min	4.900000	2.000000	3.000000	1.000000
25%	5.800000	2.700000	4.375000	1.300000
50%	6.300000	2.900000	4.900000	1.600000
75%	6.700000	3.025000	5.525000	2.000000
max	7.900000	3.800000	6.900000	2.500000

- 插入列：df['列名']=[]

  若只给一个值时会广播操作，将该列的值全部赋值为该值

```
In [48]: df['age']=[23,25,56,42]
```

```
In [49]: df
```

Out[49]:

	name	job	ID	age
c1	袁娟	西班牙语翻译	360502198012272491	23
c2	李刚	电脑维修	360981197911307232	25
c3	刘伟	营运主管	620702195704186659	56
c4	高辉	管道/暖通	330303195307282327	42

- 获取某列的取值集合：set(list(df['列名']))
- 转置：df.T
- 排序
  - df.sort_values(by="col",ascending=True)
- 删除：df.drop([''],inplace=True,axis=0/1)
  - **inplace**为False时不修改原df，返回修改后的数据，若为True在原df上修改，返回None
  - **axis**指定删除方向，0为行索引（纵向计算），1为列索引（横向计算），默认为0
  - 这两个参数在很多情况都会用到
  - 也可以使用del关键词删除
- **注意**：从dataframe中取出数据进行操作后，会对原始数据产生影响；对获取的数据copy()操作之后，则不会对原数据产生修改。

4 Index

4.1 索引对象

- Series和DataFrame中的索引都是Index对象

- 不可变（immutable）,保证了数据的安全
- 常见Index
  - Index、Int64Index、MultiIndex、DatetimeIndex
- reset_index(list(""))重置索引,取其中的某些行
- 重命名列名：df.rename(columns={old:new,...},inplace=True)

```
iris.rename(columns={0:'Sepal Length',1:'Sepal Width',2:'Petal
Length',3:'Petal Width',4:'type'})
```

	Sepal Length	Sepal Width	Petal Length	Petal Width	type
0	5.5	2.5	4.0	1.3	Iris-versicolor
1	6.3	2.5	5.0	1.9	Iris-virginica
2	5.5	2.6	4.4	1.2	Iris-versicolor
3	6.0	2.2	4.0	1.0	Iris-versicolor

4.2 层级索引MultiIndex

- set_index([])设置层级索引
- 多级索引是读取数据方式：df.loc['第一级','第二级']
- swaplevel()交换索引层级
- sort_index(level=0)排序分层
- 返回索引唯一值：df.set_index("").index.unique()

4.3 Boolean Mask

- 获取某列指定取值的数据
  - df[df['列名']=='值'&...]

```
Iris_virginica=iris[iris['type']=='Iris-virginica']
```

5 数据清洗

5.1 缺失值处理

- 判断数据缺失：df.isnull()/notnull()
- 填充缺失数据
  - df.fillna(t.mean()/t.median()/0) 获取该列非nan列表并求统计值，注意只在该列上操作
  - df.ffill()由前一个值填充
  - df.bfill()由后一个值填充
- 删除缺失数据：df.dropna(axis=0,how="any",inplace=False)how的值为any时表示只要该行存在NaN即删除，all则全为NaN才删除

5.2 数据变形

- 重复数据
  - duplicated()判断是否重复
  - drop_duplicated([''],keep='last')去除重复数据
  - unique() 或者set() 均可以去重
- 表连接
  - df['列名'].map(dict)
  - dict为以df中该列的取值为键，map操作将dict中的值按键添加到df中
  - map中的dict可以通过lambda表达式指定

```
dict={'Iris-versicolor':'1','Iris-virginica':'2'}
iris['class']=iris[4].map(dict)
iris
```

	0	1	2	3	4	class
0	5.5	2.5	4.0	1.3	Iris-versicolor	1
1	6.3	2.5	5.0	1.9	Iris-virginica	2
2	5.5	2.6	4.4	1.2	Iris-versicolor	1
3	6.0	2.2	4.0	1.0	Iris-versicolor	1
4	6.0	2.2	5.0	1.5	Iris-virginica	2

- 替换值
  - df.replace([],[])
- 离散化与分箱
  - cats=pd.cut(data,bins,labels=[])
  - data为list数据，bins为list类型的分箱边界list类型,labels指定每个区间的标记
  - cats.categories返回边界索引
  - cats.codes分箱编码
  - pd.value_counts(cats)统计箱中元素个数
  - cats.get_values()获取数据的分箱值
- one-hot编码（哑变量）
  - pf.get_dummies(series/dataframe)

```
pd.get_dummies(iris[4])
```

	Iris-versicolor	Iris-virginica
0	1	0

	Iris-versicolor	Iris-virginica
1	0	1
2	1	0
3	1	0
4	0	1

- 向量化字符串操作
  - .str.contains('')
  - .str.split('',expand=True)
  - .str.cat(s1,sep='')
  - .str.endswith('')
  - .str.startswith('')
  - .str.findall('')
  - .str.isalnum()
  - .str.isalpha()
  - .str.isdecimal()
  - .str.isdigit()
  - .str.islower()
  - .str.isnumeric()
  - .str.isupper()
  - .str.join('')
- **分组（groupby）**
  - split->apply->combine
  - g=df.groupby('列名')返回可迭代对象，每次返回一个元组(group_name,group_data)

```
iris_rename=iris.rename(columns={0:'Sepal Length',1:'Sepal
Width',2:'Petal Length',3:'Petal Width',4:'type'})
groups=iris_rename.groupby('type')
groups['Sepal Length'].mean()
groups['Sepal Length'].max()
groups['Sepal Length'].min()
groups.size()
for g,f in groups:
    max=f['Sepal Length'].max()
    min=f['Sepal Length'].min()
    mean=f['Sepal Length'].mean()
    print("{} max:{},min:{},mean:{}".format(g,max,min,mean))
```

  - 也可以按指定的函数分组

```
iris_index=iris.set_index(0)
def get_rank_group(num):
    rank_group=''
    if num<5:
        rank_group=' <5'
    elif num<6:
        rank_group='5~6'
    elif num<7:
        rank_group='6~7'
    else:
```

```
        rank_group='>=7'
    return rank_group

groups=iris_index.groupby(get_rank_group)
groups
for g,f in groups:
    print('{}:{}'.format(g,len(f)))
```

```
 <5:2
5~6:31
6~7:54
>=7:13
```

- **聚合（aggregation）**
  - grouped.agg(fun)，数组产生标量的过程，mean/count/...
  - 常用于分组后的数据计算
  - 内置聚合函数：sum/mean/max/min/count/size/describe
  - 可以通过字典为每列指定不同的操作方法
  - 可自定义函数，传入agg方法
  - agg参数可以指定为list，按不同的操作分别计算
- **透视表**
  - df.pivot_table(values='',index='',columns='',aggfun=,margins=)
  - values透视表中的元素值
  - index透视表的行索引
  - columns透视表的列索引
  - aggfunc聚合函数，可以指定多个
  - margins表示是否对所有数据统计
- **合并merge**
  - 多个dataframe进行合并
  - pd.merge(df1,df2,how='inner/outer/left/right',on='列名',left_index=True,right_index=True,left_on='',right_on='',suffixes=('',))df1中按照列名合并df2(inner只返回df1和df2中该列交集的合并,outer返回该列并集的合并，left以df1为准，right以df2为准，没有的地方NaN填充)
  - how='inner/outer/left/right'
  - apply(func)