

pandas基础知识

一 : pandas介绍

二 : pandas核心数据结构及基础知识

 1. Series

 2. DataFrame

 代码练习

 3. 数据类型

 4. 基础属性

 代码练习

 5. 操作DataFrame数据

 1) 数据基本的查看方式

 2) 查看访问DataFrame数据—loc, iloc方法介绍

 代码练习 :

 3) 删除某列或某行数据

 4) 对数据进行排序

 代码练习

三 : pandas获取数据

 1. 文本文件读取

 2. movielens电影评分数据读取

 (1)电影详情表

 (2)评分表

 (3)用户表

 代码练习 movielens.ipynb

 3. 文本存储

 4. pandas描述性统计分析

 (1) 数值型特征的描述性统计

 (2) 类别型特征的描述性统计

 代码练习 : pandas_describe

四 : pandas统计分析

 1. 分组功能的实现

 2. 聚合函数agg

 代码练习 : groupby_agg.ipynb

五 : movielens电影评分数据分析

 代码练习 : movielens_analysis.ipynb

任务一 : 查看每一部电影的平均评分

 创建数据透视表

任务二 : 评分最高的十部电影

pandas基础知识

一 : pandas介绍

安装方法 : pip install pandas

1. 名称 : Python Data Analysis Library 或 pandas

2. 介绍 : 是基于NumPy 的一种工具 , 该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型 , 提供了高效地操作大型数据集所需的工具。

3. 定义1 : pandas提供了大量能使我们快速便捷地处理数据的函数和方法。

4. 定义2 : pandas是python里面分析结构化数据的工具集 , 基础是numpy , 图像库是matplotlib

二 : pandas核心数据结构及基础知识

1. Series

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（索引）组成，不要求数据类型是相同的。可以看作是一维数组

创建的方法统一为pd.Series(data,index) , index赋值必须是list类型

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

2. DataFrame

DataFrame是二维的关系表格型数据结构，含有一组有序的列，每列的数据类型是相同的，有这样一些参数：

- 1、data（方框内的数据）
- 2、index（行索引）
- 3、columns（列索引）
- 4、dtype（data的数据类型）

DataFrame统一的创建形式为：pd.DataFrame(data,columns,index,dtype)，比如如图所示

	a	b	c	d
2010-01-01	-0.758900	0.981442	0.372002	1.047799
2010-01-02	0.732313	-0.097231	-0.351529	-0.105151
2010-01-03	-0.946279	-1.498994	0.538853	1.992580
2010-01-04	1.366995	1.185058	-1.247609	1.932361
2010-01-05	-0.261246	0.522998	0.221616	0.477118
2010-01-06	-1.539353	0.393074	-1.228885	1.256865

代码练习

```

import pandas as pd
import numpy as np
# 1.Series创建
s1=pd.Series(data=[3,-5,'b',4],index=['a','b','c','d'])
print(s1)
# 2.创建一组DataFrame数据-date_range创建时间
date=pd.date_range('20100101',periods=6)
# print(date)
df=pd.DataFrame(np.random.randn(6,4),index=date,columns=list('abcd'))
df

```

3. 数据类型

pandas扩展了NumPy的类型系统，用dtypes属性来显示元素的数据类型，pandas主要有以下几种dtype：

- 字符串类型：object
- 整数类型：Int64 , Int32 , Int16, Int8
- 无符号整数：UInt64 , UInt32 , UInt16, UInt8
- 浮点数类型：float64 , float32
- 日期和时间类型：datetime64[ns]、 datetime64[ns, tz]、 timedelta[ns]
- 布尔类型：bool

4. 基础属性

函数	返回值
values	元素
index	索引
columns	列名
dtypes	类型
size	元素个数
ndim	维度数
shape	数据形状（行列数目）

代码练习

```

# 得到元素
df.values
# 得到列名称
df.columns
# 得到形状
df.shape

```

5. 操作DataFrame数据

1) 数据基本的查看方式

- 对列的访问：
- 对单列数据的访问：DataFrame的单列数据为一个Series。根据DataFrame的定义可以知晓 DataFrame是一个带有标签的二维数组，每个标签相当每一列的列名。df.a df['a']
- 对多列数据访问：访问DataFrame多列数据可以将多个列索引名称视为一个列表，df[['a','b']]
- 对某几行访问：
 - 如果只是需要访问DataFrame某几行数据的实现方式则采用数组的选取方式，使用“：“。
 - head和tail也可以得到多行数据，但是用这两种方法得到的数据都是从开始或者末尾获取的连续数据。默认参数为访问5行，只要在方法后方的“()”中填入访问行数即可实现目标行数的查看。

2) 查看访问DataFrame数据—loc, iloc方法介绍

1. loc方法是针对DataFrame索引名称的切片方法，如果传入的不是索引名称，那么切片操作将无法执行。利用loc方法，能够实现所有单层索引切片操作。loc方法使用方法如下。

DataFrame.loc[行索引名称或条件, 列索引名称]

2. iloc和loc区别是iloc接收的必须是行索引和列索引的位置。iloc方法的使用方法如下。

DataFrame.iloc[行索引位置, 列索引位置]

如图所示：得到红色框中的数据

	a	b	c	d
2010-01-01	0.537276	1.600419	1.037267	-0.416919
2010-01-02	1.466693	1.418624	-1.792067	1.595889
2010-01-03	0.694957	0.326015	0.463335	0.568509
2010-01-04	0.091378	0.223707	0.525408	-0.162677
2010-01-05	-0.685320	-1.928679	-0.541296	-0.401515
2010-01-06	0.428047	-1.245919	0.781141	0.469301

代码练习：

```
# 对a列数据查找
print(df.a)
print(df['a'])
# 对a,b两列数据
print(df[['a', 'b']])
# 打印前2行数据
print(df.head(2))
print(df[0:2])
# loc方法查找红色框中的数据
df2=df.loc['2010-01-01':'2010-01-04', ['a', 'b']]
df2
# iloc方法查找红色框中的数据
df3=df.iloc[:4,[0,1]]
print(df3)
# 按条件查询
```

```
df.loc[df.index<'20100104','a']
```

3) 删除某列或某行数据

删除某列或某行数据需要用到pandas提供的方法drop , drop方法的用法如下。

drop(labels, axis=0, level=None, inplace=False, errors='raise') , axis为0时表示删除行 , axis为1时表示删除列。

常用参数如下所示。

参数名称	说明
labels	接收string或array。代表删除的行或列的标签。无默认。
axis	接收0或1。代表操作的轴向。默认为0。
levels	接收int或者索引名。代表标签所在级别。默认为None。
inplace	接收boolean。代表操作是否对原数据生效。默认为False。

4) 对数据进行排序

- df.sort_index(axis=,ascending=) axis为0/1的参数 , 表示按行/按列排序 ;
- df.sort_values(by= , ascending=) by表示按哪一个columns参数排序。
- ascending为boolean参数 , False表示降序 , True表示升序。

代码练习

```
#删除a列
df1=df.drop('a',axis=1)
df1
#删除第一行
df2=df.drop(labels=date[0],axis=0)
df2
#排序
df_sort=df.sort_values(by='b',ascending=False)
df_sort
```

三 : pandas获取数据

我们平时接触最多的轻量级数据 , 一般是录入在Excel中 , 如果想要用Python处理这些数据 , 就需要将数据导出来 , 这一步是初始工作 , 但也是操作频繁必不可少的一步。如图所示

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

1. 文本文件读取

文本文件是一种由若干行字符构成的计算机文件，它是一种典型的顺序文件

1. 使用read_table来读取文本文件。

```
pandas.read_table(filepath_or_buffer, sep='\t', header='infer', names=None, index_col=None,
dtype=None, engine=None, nrows=None)
```

2. 使用read_csv函数来读取csv文件。

```
pandas.read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_col=None,
dtype=None, engine=None, nrows=None)
```

常见参数介绍如下所示：

参数名称	说明
filepath	接收string。代表文件路径。无默认。该字符串可以是一个URL。有效的URL方案包括http , ftp , s3和file
sep	接收string。代表分隔符。read_csv默认为”，“，read_table默认为制表符”[Tab]”。
header	接收int或sequence。表示是否将某行数据作为列名。默认为infer，表示自动识别。
names	接收array。表示列名。默认为None。
index_col	接收int、sequence或False。表示选择哪列作为行索引，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。
engine	接收c或者python。代表数据解析引擎。默认为c。
nrows	接收int。表示读取前n行。默认为None。

注意事项：

- read_table和read_csv函数中的sep参数是指定文本的分隔符的，如果分隔符指定错误，在读取数据的时候，每一行数据将连成一片。
- header参数是用来指定列名的，如果是None，我们需要添加一个默认的列名。采用names设置

- encoding代表文件的编码格式，常用的编码有utf-8、utf-16、gbk、gb2312、gb18030等。如果编码指定错误数据将无法读取，IPython解释器会报解析错误。

2. movielens电影评分数据读取

数据集来源：一组从20世纪90年末到21世纪初由MovieLens用户提供的电影评分数据。这些数据中包括电影评分、电影元数据（风格类型和年代）以及关于用户的人口统计学数据（年龄、邮编、性别和职业等）。MovieLens 1M数据集含有来自6000名用户对4000部电影的100万条评分数据。分为三个表：评分、用户信息和电影信息。

课堂目标：要求使用pandas合适的方法读取文件

User.dat中需要自定义列名为：['UserID','Gender','Age','Occupation','Zip-code']

movies.dat 中需要自定义列名为：['MovieID', 'Title', 'Genres']

ratings.dat 中需要自定义列名为：['UserID', 'MovieID', 'Rating', 'Timestamp']

(1)电影详情表

1::Toy Story (1995)::Animation Children's Comedy
2::Jumanji (1995)::Adventure Children's Fantasy
3::Grumpier Old Men (1995)::Comedy Romance
4::Waiting to Exhale (1995)::Comedy Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action Crime Thriller
7::Sabrina (1995)::Comedy Romance
8::Tom and Huck (1995)::Adventure Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action Adventure Thriller

(2)评分表

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
```

(3)用户表

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
```

代码练习 movielens.ipynb

```
import numpy as np
import pandas as pd
# 从用户表读取用户信息
users = pd.read_table('data/users.dat', header=None, names=['UserID', 'Gender', 'Age', 'Occupation', 'zip-code'], sep='::', engine='python')
# 打印列表长度，共有6040条记录
print(len(users))
# 查看前五条记录
users.head(5)# 同样方法，导入电影评分表
ratings = pd.read_table('data/ratings.dat', header=None, names=['UserID', 'MovieID', 'Rating', 'Timestamp'], sep='::', engine='python')
# 打印列表长度
```

```

print(len(ratings))
print(ratings.head(5))
# 同样方法，导入电影数据表
movies = pd.read_table('data/movies.dat', header=None, names=['MovieID',
'Title', 'Genres'], sep='::', engine='python')
print(len(movies))
print(movies.head(5))
# 导入完成之后，我们可以发现这三张表类似于数据库中的表
# 要进行数据分析，我们就要将多张表进行合并才有助于分析 先将users与ratings两张表合并再跟
movies合并
data = pd.merge(pd.merge(users, ratings), movies)
data.tail(5)
#进行存储
data.to_csv('data.csv', index=False)

```

3. 文本存储

文本文件的存储和读取类似，结构化数据可以通过pandas中的to_csv函数实现以csv文件格式存储文件。DataFrame.to_csv(path_or_buf=None, sep=',', na_rep='', columns=None, header=True, index=True, index_label=None, mode='w', encoding=None)

参数名称	说明	参数名称	说明
path_or_buf	接收string。代表文件路径。 无默认。	index	接收boolean，代表是否将行名（索引）写出。默认为True。
sep	接收string。代表分隔符。默认为","。	index_labels	接收sequence。表示索引名。默认为None。
na_rep	接收string。代表缺失值。默认为""。	mode	接收特定string。代表数据写入模式。默认为w。
columns	接收list。代表写出的列名。 默认为None。	encoding	接收特定string。代表存储文件的编码格式。默认为None。
header	接收boolean，代表是否将列名写出。默认为True。		

4. pandas描述性统计分析

(1) 数值型特征的描述性统计

数值型数据的描述性统计主要包括了计算数值型数据的完整情况、最小值、均值、中位数、最大值、四分位数、极差、标准差、方差、协方差和变异系数等。在NumPy库中一些常用的统计学函数如下表所示。

函数名称	说明	函数名称	说明
np.min	最小值	np.max	最大值
np.mean	均值	np.ptp	极差
np.median	中位数	np.std	标准差
np.var	方差	np.cov	协方差

pandas库基于NumPy，自然也可以用这些函数对数据框进行描述性统计

pandas还提供了一个方法叫作describe，能够一次性得出数据框所有数值型特征的非空值数目、均值、四分位数、标准差。

方法名称	说明	方法名称	说明
min	最小值	max	最大值
mean	均值	ptp	极差
median	中位数	std	标准差
var	方差	cov	协方差
sem	标准误差	mode	众数
quantile	四分位数	count	非空值数目
describe	描述统计		

(2) 类别型特征的描述性统计

- 描述类别型特征的分布状况，可以使用频数统计表。pandas库中实现频数统计的方法为value_counts。
- describe方法除了支持传统数值型以外，还能够支持对数据进行描述性统计，四个统计量分别为列非空元素的数目，类别的数目，数目最多的类别，数目最多类别的数目

代码练习：pandas_describe

```
#获取泰坦尼克号生存预测分析测试集数据
# 读取test.csv表格
test=pd.read_csv('data/test.csv')
print(test.head())
print(len(test))
# PassengerId => 乘客ID
# Pclass => 乘客等级(1/2/3等舱位)
# Name => 乘客姓名
# Sex => 性别
# Age => 年龄
# SibSp => 堂兄弟/妹个数
```

```

# Parch => 父母与小孩个数
# Ticket => 船票信息
# Fare => 票价
# Cabin => 客舱
# Embarked => 登船港口
# 数据描述性分析
# 数据完整性分析
test.info()
test.select_dtypes('int64').describe().T
# 数值型数据统计分析
test.describe()
test['Age'].mean()
test.Sex.describe()
test.select_dtypes('int64').describe().T
# 类别型数据统计分析
test.select_dtypes('object').describe().T
test.Sex.value_counts()

# 描述性数据分析练习
# 数值型
df=pd.DataFrame(np.arange(50).reshape(10,5),columns=list('abcde'))
print(df)
print(df.describe())
# 类别型
ss = pd.Series(['a', 'd', 'a', 'c', 'd', 'a'])
#value_counts 直接用来计算series里面相同数据出现的频率
print(ss.value_counts())
print(ss.describe())

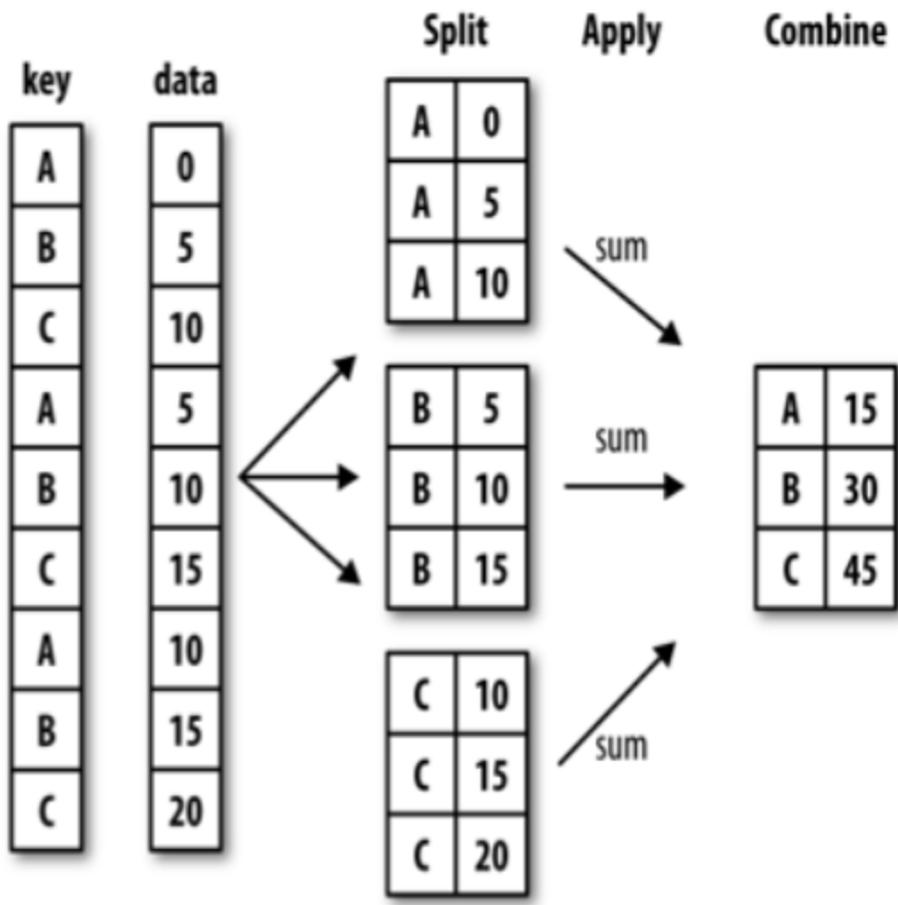
```

四：pandas统计分析

本阶段主要内容：

1. 使用分组聚合方式进行组内计算
2. 创建数据透视表
3. 应用案例：movielens电影评分数据分析

1. 分组功能的实现



使用Dataframe.groupby方法拆分数据：groupby方法的参数及其说明

该方法提供的是分组聚合步骤中的拆分功能，能根据索引或字段对数据进行分组。其常用参数与使用格式如下。

参数名称	说明
by	接收list, string, mapping或generator。用于确定进行分组的依据。无默认。
axis	接收int。表示操作的轴向，默认对行进行操作。默认为0。
level	接收int或者索引名。代表标签所在级别。默认为None。
as_index	接收boolean。表示聚合后的聚合标签是否以DataFrame索引形式输出。默认为True。
sort	接收boolean。表示是否对分组依据分组标签进行排序。默认为True。
group_keys	接收boolean。表示是否显示分组标签的名称。默认为True。

用groupby方法分组后的结果并不能直接查看，而是被存在内存中，输出的是内存地址。实际上分组后的数据对象GroupBy类似Series与DataFrame，是pandas提供的一种对象

GroupBox对象常用的描述性统计方法如下。

方法名称	说明	方法名称	说明
count	计算分组的数目，包括缺失值。	cumcount	对每个分组中组员的进行标记，0至n-1。
head	返回每组的前n个值。	size	返回每组的大小。
max	返回每组最大值。	min	返回每组最小值。
mean	返回每组的均值。	std	返回每组的标准差。
median	返回每组的中位数。	sum	返回每组的和。

2. 聚合函数agg

agg方法求统计量

1. 可以使用agg方法一次求出当前数据中所有菜品销量和售价的总和与均值，如
detail[['counts','amounts']].agg([np.sum,np.mean]))。
2. 对于某个字段希望只做求均值操作，而对另一个字段则希望只做求和操作，可以使用字典的方式，将两个字段名分别作为key，然后将NumPy库的求和与求均值的函数分别作为value，如
detail.agg({'counts':np.sum,'amounts':np.mean}))
3. 在某些时候还希望求出某个字段的多个统计量，某些字段则只需要求一个统计量，此时只需要将字典对应key的value变为列表，列表元素为多个目标的统计量即可，如
detail.agg({'counts':np.sum,'amounts':[np.mean,np.sum]}))
在正常使用过程中，agg函数和aggregate函数对DataFrame对象操作时功能几乎完全相同，因此只需要掌握其中一个函数即可。它们的参数说明如下表。

参数名称	说明
func	接收list、dict、function。表示应用于每行 / 每列的函数。无默认。
axis	接收0或1。代表操作的轴向。默认为0。

代码练习：groupby_agg.ipynb

```
#给定以下数据
df = pd.DataFrame({'A': [1, 1, 2, 2],
                   'B': [3, 4, 3, 4],
                   'C': np.random.randn(4)})

# print(df)
# 按照A列分组后聚合求最小值 df1接收
df1=df.groupby('A').agg('min')
print(df1)

# 按照A列分组和B列分组后聚合求最小值和最大值 df2接收
df2=df.groupby(['A','B']).agg(['min','max'])
print(df2)

# 按照A列分组后聚合后对B列求最小值和最大值,
# # 对C列求和, df3接收
df3=df.groupby('A').agg({'B':['min','max'],'C':'sum'})
print(df3)
```

五：movielens电影评分数据分析

代码练习：`movielens_analysis.ipynb`

任务一：查看每一部电影的平均评分

如图所示

```
Title
$1,000,000 Duck (1971)           3.027027
'Night Mother (1986)              3.371429
'Til There Was You (1997)         2.692308
'turbs, The (1989)                2.910891
...And Justice for All (1979)     3.713568
Name: Rating, dtype: float64
```

```
import numpy as np
import pandas as pd
data=pd.read_csv('data.csv')
data.head()
#对数据初步描述分析
data.describe()
data.info()
#查看每一部电影的平均评分
data_title=data.groupby('Title')['Rating'].mean()
data_title.head()
#查看每一部电影不同性别的平均评分
data_gender=data.groupby(['Title','Gender']).agg({'Rating':'mean'})
data_gender.head()
```

创建数据透视表

`pivot_table`函数常用参数及其说明, 利用`pivot_table`函数可以实现透视表，`pivot_table()`函数的常用参数及其使用格式如下。

```
pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean',
fill_value=None, margins=False, dropna=True, margins_name='All')
```

!

参数名称	说明
data	接收DataFrame。表示创建表的数据。无默认。
values	接收字符串。用于指定想要聚合的数据字段名，默认使用全部数据。默认为None。
index	接收string或list。表示行分组键。默认为None。
columns	接收string或list。表示列分组键。默认为None。
aggfunc	接收functions。表示聚合函数。默认为mean。
margins	接收boolearn。表示汇总（Total）功能的开关，设为True后结果集中会出现名为“ALL”的行和列。默认为True。
dropna	接收boolearn。表示是否删掉全为NaN的列。默认为False。

pivot_table函数主要的参数调节

- index表示透视表的行
- columns表示透视表的列
- values 表示聚合的数据
- aggfunc表示对分析对象进行的分析，一般默认为求平均值，可以指定
- margins表示添加每行每列求和的值，默认不添加。

代码练习

```
# 查看每一部电影不同性别的平均评分 data_gender接收
data_gender=data.pivot_table(index='Title',columns='Gender',values='Rating',aggfunc='mean')
data_gender.head()
#得到男女分歧最大的前五个电影
data_gender['diff']=np.fabs(data_gender.F-data_gender.M)
data_gender.head()
data_gender_sorted=data_gender.sort_values(by='diff',ascending=False)
data_gender_sorted.head()
```

任务二：评分最高的十部电影

```
#评分最高的十部电影
movies_stats=data.groupby('Title').agg({'Rating':[ 'size','mean']})
movies_stats.head()
# 被评论的次数>=100
atleast_100=movies_stats['Rating']['size']>=1000
movies_stats[atleast_100].sort_values(by=( 'Rating','mean'),ascending=False)[:10]
```