



第6章 数据存取

Jupyter金融应用从入门到实践

CSV是一种使用逗号作为分隔符的特殊的纯文本文件格式，全称为**Comma Separate Values**。文件以纯文本形式存储表格数据，可由任意数目的记录组成，记录间以某种换行符分隔。

1、使用Python读取
CSV文件的常用方式主要是通过Pandas的read_csv()。

read_csv()返回的数据格式是DataFrame。

[In]

```
1. import pandas as pd
2. path = "dataset/advertising.csv"
3. data = pd.read_csv(path)
4. data
```

[Out]

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2

2、用to_csv()，用于将结果存入csv中。

```
1. path = "dataset/new_advertising.csv"
2. data.to_csv(path, sep='\t', float_format='%.2f', header=0, index=0)
```

CSV参数配置

- path, 表示CSV文件所在的位置, 可以是本地的文件位置也可以是一个URL链接的位置。
- sep代表标识分隔符, 示例中的分隔符为英文逗号, 表示列与列之间用英文逗号分开, 该参数可以不填写, CSV的默认分隔符就是逗号。
- header表示文件列名的位置, 一般与names参数组合使用。header从0开始计数, header=0表示第一行是列名 (即表头)。
- names用于设置列名, 若希望使用自己设置的列名, 可以使用names定义, 这时返回的列名就会被自己定义的列名代替
- 要注意的是, 若不定义header只定义names, 那么header会默认为第一行就是数据并进行读取。如果CSV文件的第一行是表头, 那么names和header都无须指定。

[In]

```
1. import pandas as pd #导入 pandas 代码模块
2. path = "dataset/advertising.csv"
3. head = ["电视", "广播", "新闻", "销售"]
4. data = pd.read_csv(path, sep=',', header=0, names=head)
5. data
```

[Out]

	电视	广播	新闻	销售
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。**JSON**简洁和清晰的层次结构使它成为理想的数据格式。键值对这种类似字典的格式易于人们阅读和编写，同时也易于机器解析和生成。

1、JSON由花括号定义，括号里面的对象键值对由英文逗号分隔主

```
[In] 1. {
      2.     "TV": {
      3.         "0": 230.1,
      4.         "1": 44.5,
      5.         "2": 17.2
      6.     },
      7.     "Radio": {
      8.         "0": 37.8,
      9.         "1": 39.3,
     10.         "2": 45.9
     11.     },
     12.     "Newspaper": {
     13.         "0": 69.2,
     14.         "1": 45.1,
     15.         "2": 69.3
     16.     },
     17.     "Sales": {
     18.         "0": 22.1,
     19.         "1": 10.4,
     20.         "2": 12.0
     21.     }
     22. }
```

2、Pandas的read_json()函数读取JSON文

```
[In] 1. import os
      2. import pandas as pd
      3. path = "dataset/advertising.json"
      4. data = pd.read_json(path)
      5. data
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0

JSON参数配置

示例 1 : 当 JSON 格式是 {索引 -> [index], 列名 -> [columns], 数据 -> [values]} 这种类型

时, orient 参数应赋值为 "split"。这种 JSON 文件的键 (key) 的名字只能是 index、columns、

data 这 3 个, 多一个键 (key) 都不行, 少一个值 (value) 也不行。

[In]

```
1. json1 = '{
2.     "index": ["0", "1", "2"],
3.     "columns": ["TV", "Radio", "Newspaper", "Sales"],
4.     "data": [
5.         [230.1, 37.8, 69.2, 22.1],
6.         [44.5, 39.3, 45.1, 10.4],
7.         [17.2, 45.9, 69.3, 12.0]
8.     ]
9. }'
```

```
10. data = pd.read_json(json1, orient='split')
11. data
```

[Out]

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0

示例 2 : 当 JSON 文件格式是 [{列名 -> 值}, ..., {列名 -> 值}] 这种形式时, orient 参数

应赋值为 "records"。

[In]

```
1. json2 = '[{
2.     "TV": 230.1,
3.     "Radio": 37.8,
4.     "Newspaper": 69.2,
5.     "Sales": 22.1
6. }, {
7.     "TV": 44.5,
8.     "Radio": 39.3,
9.     "Newspaper": 45.1,
10.    "Sales": 10.4
11. }, {
12.    "TV": 17.2,
13.    "Radio": 45.9,
14.    "Newspaper": 69.3,
15.    "Sales": 12.0
16. }]'
```

```
17. ]'
```

```
18. data = pd.read_json(json2, orient='records')
19. data
```

[Out]

	Newspaper	Radio	Sales	TV
0	69.2	37.8	22.1	230.1
1	45.1	39.3	10.4	44.5
2	69.3	45.9	12.0	17.2

JSON参数配置

示例 3：当 JSON 文件格式是 {索引 -> {列名 -> 数值}}这种形式时（这种形式与上一种的唯一差别在于有自定义索引），orient 参数应赋值为"index"。

```
[In] 1. json3 = '{
2.     "0": {
3.         "TV": 230.1,
4.         "Radio": 37.8,
5.         "Newspaper": 69.2,
6.         "Sales": 22.1
7.     },
8.     "1": {
9.         "TV": 44.5,
10.        "Radio": 39.3,
11.        "Newspaper": 45.1,
12.        "Sales": 10.4
13.    },
14.    "2": {
15.        "TV": 17.2,
16.        "Radio": 45.9,
17.        "Newspaper": 69.3,
18.        "Sales": 12.0
19.    }
20. }'
21. data = pd.read_json(json3, orient='index')
22. data
```

示例 4：当 JSON 文件格式是 {列名-> {索引 -> 数值}}这种形式时（这种形式与上一种相似，只是做了索引和列名的位置对换），orient 参数应赋值为"columns"。

```
[In] 1. json4 = '{
2.     "TV": {
3.         "0": 230.1,
4.         "1": 44.5,
5.         "2": 17.2
6.     },
7.     "Radio": {
8.         "0": 37.8,
9.         "1": 39.3,
10.        "2": 45.9
11.    },
12.    "Newspaper": {
13.        "0": 69.2,
14.        "1": 45.1,
15.        "2": 69.3
16.    },
17.    "Sales": {
18.        "0": 22.1,
19.        "1": 10.4,
20.        "2": 12.0
21.    }
22. }'
23. data = pd.read_json(json4, orient='columns')
24. data
```

数据库能够存储大量数据，我们可以通过Pandas来直接读取数据库以实现高效、清晰的读写。Pandas使用sqlalchemy与数据库进行连接，支持Mysql、Oracle、SQLServer、SQLite等主流数据库。

```
[In] 1. # 导入模块
      2. import pandas as pd
      3. from sqlalchemy import create_engine
      4. # 初始化数据库连接，使用 pymysql 模块
      5. engine = create_engine('mysql+pymysql://usr:passwd@192.168.10.1:3306/jupyter')
      6. # 读取本地 CSV 文件
      7. df = pd.read_csv("dataset/advertising.csv", sep=',')
      8. # 将新建的 DataFrame 储存为 MySQL 中的数据表
      9. df.to_sql('advertising', engine, if_exists='replace', index= False)
     10. #读取结果
     11. pd.read_sql_table("advertising",engine).head()
```

[Out]

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

常用操作

示例 1：当使用 select 语句查询数据库时，会返回一个 DataFrame 类型的结果。

```
[In] 1. sql = ' select TV from advertising; '  
2. df = pd.read_sql_query(sql, engine)  
3. df.head()
```

示例 2：执行 insert 语句插入数据。

```
[In] 1. sql = ' insert into advertising (TV, Radio, Newspaper, Sales) values (1,1,1,1); '  
2. engine.execute(sql)
```

示例 3：执行 update 语句更新数据。

```
[In] 1. sql = 'update advertising set Sales=22.6 where TV=230.1'  
2. engine.execute(sql)
```

示例 4：执行 delete 语句删除数据。

```
[In] 1. sql = 'delete from advertising where TV=230.1'  
2. engine.execute(sql)
```


在读取各种文本文件时，除了使用上述的pandas内置方法外，还可以使用Python自带的open()函数。

以广告数据集中的 10 条数据新建一个 txt 文件用于读取。

```
[In] 1. f = open('dataset/new_advertising.txt','r',encoding='utf-8')
      2. f.read()

[Out] '230 38 69 22\n44 39 45 10\n17 46 69 12\n152 41 58 16\n181 11 58 18\n9 49 75 7\n58
      33 24 12\n120 20 12 13\n9 2 1 5\n200 3 21 16'
```

readline()一次读取一行，并定位到当前读取行的位置，一般配合 while 进行读取。

```
[In] 1. f = open('dataset/new_advertising.txt','r',encoding='utf-8')
      2. line = f.readline()
      3. while line:
      4.     print(line)
      5.     line =f.readline()
```

readlines()方法一次性读取全部行并返回一个列表，一般配合 for 循环进行读取。

```
[In] 1. f = open('dataset/new_advertising.txt','r',encoding='utf-8')
      2. for line in f.readlines():
      3.     line = line.strip()    #去除首尾空格，包括换行符
      4.     print(line)
```



谢谢！