(28条消息) java 类似dataframe_Pandas DataFrame 使用总结_简单的艾伦的博客-CSDN博客

5-7 minutes

Pandas 是一个非常好用的库，总结一下 Pandas DataFrame 常见用法，在使用的时候可以快速找到。

Pandas DataFrames 是具有带标签的行和列的二维数据结构，可以存储很多类型的数据。如果你熟悉 Excel 的话，可以将 Pandas DataFrames 看做类似于电子表格。

一、引入

```
import numpy as np

import pandas as pd
```

二、创建

```
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

c = ['a', 'b', 'c']

r = ['A', 'B', 'C']

df = pd.DataFrame(data=data, columns=c, index=r)
```

三、排序

按列、行名排序

```
# 行名排序 降序

df.sort_index(axis=0, ascending=False)

# 列名排序 降序

df.sort_index(axis=0, ascending=False)
```

按值排序

拿出来排序

```
df["a"].sort_values(ascending = False)
```

df 内排序

df.sort_values(['a', 'b', 'c'])

四、索引

位置索引

df.iloc[2] # 选择第二行所有数据, 是Series类型

df.iloc[[2]] # 选择第二行所有数据, 是DataFrame类型

df.iloc[:, 2] # 选择第二列所有数据, 是Series类型

df.iloc[:, [2]] # 选择第二列所有数据, 是DataFrame类型

df.iloc[:, 0:2] # 选择0到2列所有数据

df.iloc[[2,3], 0:2] # 选择2和3行, 0到2列所有数据

df.iat[1, 1] # 根据位置快速取出数据, 获取单个数据推荐这种方法

自定义索引

df.loc['top'] # 选择指定行数据, 是Series类型

df.loc[['top']] # 选择指定行数据, 是DataFrame类型

df.loc[:, 'xm'] # 选择指定列数据, 是Series类型(不推荐)

df.loc[:, ['xm']] # 选择指定列数据, 是DataFrame类型(不推荐)

df.loc[:, ['bj','xm']] # 选择多列数据(不推荐)

df.loc[:, 'bj':'xb'] # 选择多列之间所有数据, 列切片只能用这种方法

df.loc[['top','count'], 'bj':'xb'] # 选择指定行, 指定列数据

df.at['top', 'xm'] # 根据自定义索引快速取出数据, 获取单个数据推荐这种方法

布尔索引

选取所有出生日期大于等于1998年的数据, 这里是字符串比较

df[df['csrq']>='1998']

选取所有出生日期大于等于1997年小于1999年的数据

df[(df['csrq']>='1997')&(data['csrq']

```
# 选取所有出生日期大于等于1997年小于1999年的数据

df[df['csrq'].between('1997', '1999')]

# 选取所有出生日期大于等于1997年或者姓名为张三的数据

df[(df['csrq']>='1997')|(data['xm']=='张三')]

# 另一种选取方式(不推荐, 实测效率比上面低)

df[df.csrq>='1998']

# 选择字段值为指定内容的数据

df[df['xm'].isin(['张三','李四'])]
```

五、插入与删除

```
# 假设cj列本来不存在, 这样会在列尾添加新的一列cj, 值为s(Series对象), 原地

df['cj'] = s

# 在第1列位置插入一列dz(地址), 值为s, 原地

df.insert(0, 'dz', s)

# 在df中添加内容为df2(必须是DataFrame对象)的新列(添加列), 非原地

df.join(df2)

# 将df2中的行添加到df的尾部(添加行), 非原地

df.append(df2)

# 删除单列, 并返回删除的列, 原地

df.pop('xm')

# 删除指定行, 非原地

df.drop(1)

# 删除指定列, axis=1指第2维, axis默认0, 非原地

df.drop(['xm', 'xh'], axis=1)
```

六、DataFrame 重要方法与属性

```
'''重要属性'''
```

```
df.values # 查看所有元素的value

df.dtypes # 查看所有元素的类型

df.index # 查看所有行名

df.index = ['总数', '不同', '最多', '频率'] # 重命名行名

df.columns # 查看所有列名

df.columns = ['班级', '姓名', '性别', '出生日期'] # 重命名列名

df.T # 转置后的df, 非原地

'''查看数据'''

df.head(n) # 查看df前n条数据, 默认5条

df.tail(n) # 查看df后n条数据, 默认5条

df.shape() # 查看行数和列数

df.info() # 查看索引, 数据类型和内存信息

'''数据统计'''

df.describe() # 查看数据值列的汇总统计, 是DataFrame类型

df.count() # 返回每一列中的非空值的个数

df.sum() # 返回每一列的和, 无法计算返回空, 下同

df.sum(numeric_only=True) # numeric_only=True代表只计算数字型元素, 下同

df.max() # 返回每一列的最大值

df.min() # 返回每一列的最小值

df.argmax() # 返回最大值所在的自动索引位置

df.argmin() # 返回最小值所在的自动索引位置

df.idxmax() # 返回最大值所在的自定义索引位置

df.idxmin() # 返回最小值所在的自定义索引位置

df.mean() # 返回每一列的均值

df.median() # 返回每一列的中位数
```

df.var() # 返回每一列的方差

df.std() # 返回每一列的标准差

df.isnull() # 检查df中空值, NaN为True, 否则False, 返回一个布尔数组

df.notnull() # 检查df中空值, 非NaN为True, 否则False, 返回一个布尔数组

七、转换成 Numpy

df.values

np.array(df)

八、实战应用

获取交易对BTC/USDT最近10日的收盘标准差。

```
# 计算标准差

since_days = 10

test_symbol = 'BTC/USDT'

# 计算时间点

threeDayAgo = (datetime.datetime.now() - datetime.timedelta(days=since_days))

SinceTimeStamp = int(time.mktime(threeDayAgo.timetuple())) * 1000 # 转换为时间戳，*1000，转为毫秒时间戳13位

tickers_list = binance_exchange.fetch_ohlcv(test_symbol, timeframe='1d', since=SinceTimeStamp)

# print(len(tickers_list))

# print(tickers_list)

kline_data = pd.DataFrame(tickers_list)

kline_data.columns = ['Datetime', 'Open', 'High', 'Low', 'Close', 'Vol']

print(kline_data)

print("describe:\n", kline_data.describe())

std = kline_data['Close'].std()

print("标准差：", std)
```

打印:

Datetime Open High Low Close Vol

0 1590192000000 9170.00 9307.85 9070.00 9179.15 43526.296966

1 1590278400000 9179.01 9298.00 8700.00 8720.34 70379.866450

2 1590364800000 8718.14 8979.66 8642.72 8900.35 62833.910949

3 1590451200000 8900.35 9017.67 8700.00 8841.18 58299.770138

4 1590537600000 8841.00 9225.00 8811.73 9204.07 68910.355514

5 1590624000000 9204.07 9625.47 9110.00 9575.89 74110.787662

6 1590710400000 9575.87 9605.26 9330.00 9427.07 57374.362961

7 1590796800000 9426.60 9740.00 9331.23 9697.72 55665.272540

8 1590883200000 9697.72 9700.00 9381.41 9448.27 48333.786403

9 1590969600000 9448.27 9619.00 9421.67 9542.47 15797.593487

describe:

Datetime Open High Low Close \

count 1.000000e+01 10.000000 10.000000 10.000000 10.000000

mean 1.590581e+12 9216.103000 9411.791000 9049.876000 9253.651000

std 2.615890e+08 325.168891 282.355505 312.180668 339.899591

min 1.590192e+12 8718.140000 8979.660000 8642.720000 8720.340000

25% 1.590386e+12 8967.762500 9243.250000 8727.932500 8970.050000

50% 1.590581e+12 9191.540000 9456.555000 9090.000000 9315.570000

75% 1.590775e+12 9442.852500 9623.852500 9330.922500 9518.920000

max 1.590970e+12 9697.720000 9740.000000 9421.670000 9697.720000

Vol

count 10.000000

mean 55523.200307

std 16943.615232

min 15797.593487

25% 50166.657937

50% 57837.066549

75% 67391.244373

max 74110.787662

标准差： 339.8995912341039

相关文章：

Pandas DataFrame 总结

Python Pandas DataFrame 创建 (二十)

Python Pandas DataFrame 元素访问 (二十一)

关于找一找教程网

本站文章仅代表作者观点，不代表本站立场，所有文章非营利性免费分享。

本站提供了软件编程、网站开发技术、服务器运维、人工智能等等IT技术文章，希望广大程序员努力学习，让我们用科技改变世界。

[Pandas DataFrame 使用总结]http://www.zyiz.net/tech/detail-138134.html