Pandas和Numpy的使用

Numpy的使用

```
import numpy as np
import numpy.linalg as lg
#创建一维对象
a1 = np.array([1,2,3,4,5])

#创建二维对象
a2 = np.array([[1,2,3,4,5],[6,7,8,9,10]])

#获取矩阵行列数
print(a1.shape)#返回一个元组
print(a2.shape[0])#获取行数
print(a2.shape[1])#获取列数
print('----------------------')
#按行列截取矩阵
print(a2[0:1])#截取第一行
print(a2[1,2:5])#截取第二行，第3,4,5列，不包含头，包含尾
print(a2[1:2])#截取第二行
#或者可以是
print(a2[1,:])

print('-----------------------')

#按条件截取
b = a2[a2>6]#截取矩阵a2中大于6的元素，范围是一维数组
print(b)
print(a2>6)#返回的是Boolean值

a2[a2>6] = 0#将a2中大于6的元素变为0
print(a2)
print('------------------------')

#矩阵的合并
a3 = np.array([[1,2],[3,4]])
a4 = np.array([[5,6],[7,8]])
print(np.hstack([a3,a4]))#参数传入的时候以元组或列表的形式传入，hstack是横向合并
print(np.vstack([a3,a4]))#纵向合并
print(np.concatenate((a3,a4),axis=0))#纵向
print(np.concatenate((a3,a4),axis=1))#横向
print('---------------------------')

#通过函数创建矩阵
a = np.arange(10)#默认从0开始到10，不包括10，步长为1
print(a)
a1 = np.arange(5,10)#从5到10，不包括10，步长为1
print(a1)
```

```
a2 = np.arange(5,20,2)#从5到20，不包括20，步长为2
print(a2)
print('------------------------------')

#linspace用于创建指定数量等间隔的序列，实际生成一个等差序列
a = np.linspace(0,10,7)#首位是0，末位是10，含7个数的等差序列
print(a)
print('------------------------------')

#logspace用于生成等比数列
a = np.logspace(0,2,5)#首位是10的0次方，末位是10的2次方，含5个数
print(a)
print('------------------------------')

#ones创建全1矩阵，zeros创建全0矩阵，eye创建单位矩阵，empty创建空矩阵
a_ones = np.ones((3,3))
print(a_ones)
print('------------------------------')

a_zeros = np.zeros((3,3))
print(a_zeros)
print('------------------------------')

a_eye = np.eye(3)
print(a_eye)
print('------------------------------')

a_empty = np.empty((2,2))
print(a_empty)
print('------------------------------')

#fromstring可以将字符串转化为ndarray对象，需要将字符串数字化时这个方法比较有用，可以获得字符串的ASCII序
列
a = 'Aabcdef'
b = np.fromstring(a,dtype=np.int8)#一个字符是8位
print(b)
print('--------------------------')

#fromfunction可以根据矩阵的行号列号生成矩阵的元素
def fun(i,j):
    return i+j

a = np.fromfunction(fun,(5,6))#第一个参数为指定函数，第二个参数为列表list或元组tuple，说明矩阵的大小
print(a)
print('---------------------------')

#矩阵的运算
a = np.array([[1,2,3],[4,5,6]])
b = np.array([[6,5,4],[3,2,1]])
print(a + b)
print('------------------------')
print(a - b)
print('------------------------')
```

```
print(a * b)
print('------------------------')
print(a / b)
print('------------------------')
print(a % b)
print('------------------------')
print(a**2)
print('------------------------')

#常用矩阵函数
a = np.array([1,2,3])
b = np.sin(a)#对矩阵的每一个元素取正弦
print(b)
'''
np.cos(a)
np.tan(a)
np.arcsin(a)
np.arccos(a)
np.exp(a)
np.sqrt(a)
'''
#当矩阵中的元素不在定义域范围内，会产生RuntimeWarning，结果为nan
a = np.array([-2,0,1,2])
#print(np.arcsin(a))
print('-------------------------')

#矩阵乘法dot
a = np.array([[1,2,3],[4,5,6]])
b = np.array([[1,2],[3,4],[5,6]])
print(a.shape[1] == b.shape[0])#判断前列是否等于后行
print(a.dot(b))
print('-------------------')

#矩阵的转置transpose
a = np.array([[1,2,3],[4,5,6]])
print(a.transpose())

#另一种方法
print(a.T)
print('-----------------')

#矩阵的逆，求矩阵的逆需要先导入numpy.linalg，用linalg的inv函数来求逆。矩阵求逆的条件是矩阵的行数和列数相同。
a5 = np.array([[1,2],[3,4]])
print (lg.inv(a5))

a6 = np.eye(3)
print(lg.inv(a6))
print('----------------------')

#矩阵信息获取
a = np.array([[1,2,3],[4,5,6]])
print(a.max())#最大值
```

```
print(a.min())#最小值
print(a.max(axis=0))#每列的最大值
print(a.max(axis=1))#每行的最大值
print(a.argmax(axis=1))#获取最大值元素所在的位置
print('----------------------')


#平均值mean()
a = np.array([[1,2,3],[4,5,6]])
print(a.mean())
print(a.mean(axis=0))#每一列的平均值
print(a.mean(axis=1))#每一行的平均值

#方差var()
a = np.array([[1,2,3],[4,5,6]])
print(a.var())

#标准差std()
a = np.array([[1,2,3],[4,5,6]])
print(a.std())


#中值指的是将序列按大小顺序排列后，排在中间的那个值，如果有偶数个数，则是排在中间两个数的平均值。median()
a = np.array([[1,2,3],[4,5,6]])
print(np.median(a))
print(np.median(a,axis=1))#每一行的中值

#求和
a = np.array([[1,2,3],[4,5,6]])
print(a.sum())

#累积和,指的是该位置之前(包括该位置)所有元素的和。cumsum()
a = np.array([[1,2,3],[4,5,6]])
print(a.cumsum())
```

Pandas的使用

Pandas是基于Numpy的科学计算库。Pandas可以帮助创建一个非常牢固的用于数据挖掘与分析的基础。

```
import numpy as np
import pandas as pd

#生成数据表
#导入CSV或xlsx文件
df1 = pd.DataFrame(pd.read_csv('name.csv',header = 1))
#df1 = pd.DataFrame(pd.read_excel('name.xlsx'))
print(df1)

#用pandas创建数据表
df = pd.DataFrame({"id":[1001,1002,1003,1004,1005,1006],
```

```
 "date":pd.date_range('20130102', periods=6),
  "city":['Beijing ', 'SH', ' guangzhou ', 'Shenzhen', 'shanghai', 'BEIJING '],
 "age":[23,44,54,32,34,32],
 "category":['100-A','100-B','110-A','110-C','210-A','130-F'],
  "price":[1200,np.nan,2133,5433,np.nan,4432]},
  columns =['id','date','city','category','age','price'])
print(df)

#数据表信息查看
#维度查看
print(df.shape)
print('----------------------')

#数据表基本信息
print(df.info())
print('----------------------')

#每一列数据的格式
print(df.dtypes)
print('----------------------')

#某一列格式
print(df['date'].dtype)
print('----------------------')

#空值
print(df.isnull)
print('----------------------')

#查看某一列空值
print(df['price'].unique())
print('----------------------')

#查看数据表的值
print(df.values)
print('----------------------')

#查看列名称
print(df.columns)
print('----------------------')

#查看前5行数据，后5行数据
print(df.head())#默认前5行
print('----------------------')

print(df.tail())#默认后5行
print('----------------------')


#数据表清洗
#用数字0填充空值
df.fillna(value=0)
print(df.values)
```

```
print('----------------------')

#使用列price的均值对NA进行填充
df['price'].fillna(df['price'].mean())
print(df.values)
print('----------------------')

#清楚city字段的字符空格
df['city'] = df['city'].map(str.strip)

#大小写转换
df['city'] = df['city'].str.lower()
print(df.values)
print('----------------------')


#更改数据格式
df['age'].astype('float64')
print(df.dtypes)
print('----------------------')

#删除后出现的重复值
df['city'].drop_duplicates()
print(df.values)

#删除先出现的重复值
df['city'].drop_duplicates(keep='last')

#数据替换
df['city'].replace('sh','shanghai')
print(df.values)

#数据预处理
df1 = pd.DataFrame({'id':[1001,1002,1003,1004],
                    'gender':['male','female','male','female'],
                    'pay':['Y','N','Y','Y'],
                    'm-point':[10,20,12,30]})

#数据表合并
df_inner = pd.merge(df,df1,how='inner')#匹配合并，交集
df_left = pd.merge(df,df1,how='left')#左连接
df_outwe = pd.merge(df,df1,how='outer')#并集
print('----------------------')
print(df_inner)
print('----------------------')
print(df_outwe)
print('----------------------')
print(df_left)
print('----------------------')

#设置索引列
df_inner.set_index('id')
```

```
#按照特定列的值排序
df_inner.sort_values(by=['age'])
print(df_inner)

#按照索引列排序
df_inner.sort_index()

#如果price列的值>3000，group列显示high，否则显示low
df_inner['group'] = np.where(df_inner['price'] > 3000,'high','low')
print(df_inner)
print('-----------------------')


#对复合多个条件的数据进行分组标记
df_inner.loc[(df_inner['city'] == 'beijing') & (df_inner['price'] >= 1000),'sign'] = 1

print(df_inner)
print('-----------------------')

#对category字段的值依次进行分列，并创建数据表，索引值为df_inner的索引列，列名为category和size
print(pd.DataFrame((x.split('-') for x in df_inner['category']),index=df_inner.index,columns=
['category','size']))
print('-----------------------')

#完成分裂后的数据表和原df_inner数据表进行匹配
#df_inner = pd.merge(df_inner, np.split, right_index=True, left_index=True)

#数据提取
```