任务 2.1 数据的读取与显示

目录

2.1 数据的读取与显示

【任务描述】

Numpy的文件读取/写入主要有二进制的文件读/写和文件列表形式的数据读/写两种形式。Numpy提供了若干函数，可以将结果保存到二进制或文本文件中。

除此之外，Numpy还提供了许多从文件读取数据并将其转换为数组的方法。

【知识准备】

- Numpy对二进制的文件读取与写入分别采用了load和save方法，对文本文件采用loadtxt方法进行读取，采用savetxt方法对文本文件进行写入操作。学会读取/写入文件是利用Numpy进行数据处理的基础。除此之外，Numpy还提供了很多函数方法将数据转换成常用的数组格式。

【任务实施】

- （1）save函数以二进制的格式保存数据，load函数从二进制的文件中读取数据，save函数的语法格式如下：

```
Numpy.save(file, arr,
allow_pickle=True, fix_imports=True)
```

- 参数file接收str，表示要保存的文件的名称，需要指定文件保存的路径，如果未设置，那么将会保存到默认路径下面。
- 参数arr接收array_like，表示需要保存的数组。
- save函数就是将数组arr保存至名称为“file"的文件中，其文件的扩展名.npy是系统自动添加的。将二进制数据保存到文件中代码如下。

```
In[1]:import numpy as np  # 导入NumPy库
arr = np.arange(100).reshape(10, 10)  # 创建一个数组
np.save('D:\Anaconda\csv/save_arr', arr) # 保存数组
print('保存的数组为：\n', arr)
out[1]: 保存的数组为：
[[ 0  1  2  3  4  5  6  7  8  9]
[10 11 12 13 14 15 16 17 18 19]
[20 21 22 23 24 25 26 27 28 29]
[30 31 32 33 34 35 36 37 38 39]
[40 41 42 43 44 45 46 47 48 49]
[50 51 52 53 54 55 56 57 58 59]
[60 61 62 63 64 65 66 67 68 69]
[70 71 72 73 74 75 76 77 78 79]
[80 81 82 83 84 85 86 87 88 89]
[90 91 92 93 94 95 96 97 98 99]]
```

【任务实施】

（2）如果将多个数组保存到一个文件中，那么可以使用savez函数，其文件的扩展名为.npz，代码如下：

```
In[2]:arr1 = np.array([[1, 3, 5], [2, 4, 6]])
arr2 = np.arange(10, 20, 2)
np.savez('D:\Anaconda\csv/savez_arr', arr1, arr2)
print('保存的数组1为：', arr1)
print('保存的数组2为：', arr2)
out[2]: 保存的数组1为：  [[1 3 5] [2 4 6]]
保存的数组2为：  [10 12 14 16 18]
```

【任务实施】

（3）当需要读取二进制文件时，可以使用load函数，用文件名作为参数，存储时可以省略扩展名，但读取时不能省略扩展名，代码如下：

```
In[3]:# 读取含有单个数组的文件
loaded_data = np.load('D:\Anaconda\csv/save_arr.npy')
print('读取的数组为：\n', loaded_data)
out[3]:  读取的数组为：[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19][20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39][40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59][60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79][80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
In[4]:loaded_data1 = np.load('D:\Anaconda\csv/savez_arr.npz')
print('读取的数组1为：\n', loaded_data1['arr_0'])
print('读取的数组2为：', loaded_data1['arr_1'])
out[4]: 读取的数组1为：[[1 3 5][2 4 6]]
读取的数组2为：  [10 12 14 16 18]
```

【任务实施】

（4）在实际的数据分析任务中，更多地是使用文本格式的数据，如TXT或CSV格式，因此通常会使用savetxt函数、loadtxt函数和genfromtxt函数执行对文本格式数据的读取任务。

savetxt函数可将数组写到以某种分隔符隔开的文本文件中，其基本使用格式如下。

```
Numpy.savetxt(fname, X, fmt='%.18e', delimiter=' ', newline='\n', header='', footer='',
 comments='# ', encoding=None)
```

- 参数fname接收str，表示文件名。
- 参数X接收array_like，表示数组数据。
- 参数delimiter接收str，表示数据分隔符。

【任务实施】

loadtxt函数执行的是相反的操作，即将文件加载到一个二维数组中，其基本使用格式如下。

```
Numpy.loadtxt(fname, dtype=<class
 'float'>, comments='#', delimiter=None,
 converters=None, skiprows=0,
 usecols=None, unpack=False, ndmin=0,
 encoding='bytes', max_rows=None, *,
 like=None)
```

- loadtxt函数的常用参数主要有两个，分别是fname和delimiter。
- 参数fname接收str，表示需要读取的文件、文件名或生成器。
- 参数delimiter接收str，表示用于分隔数值的分隔符。

#对文件进行存储与读取，代码如下：

```
In[5]:arr = np.arange(10, 15, 0.5).reshape(5, -1)
print('创建的数组为: \n', arr)
# fmt='%d'表示保存为整数
np.savetxt('D:\Anaconda\csv/arr.txt', arr, fmt='%d',
 delimiter=',')
# 读入的时候也需要指定逗号分隔
loaded_data = np.loadtxt('D:\Anaconda\csv/arr.txt',
 delimiter=',')
print('读取的数组为: \n', loaded_data)
Out[5]: 创建的数组为:  [[10.  10.5] [11.
 11.5][12.  12.5]
[13.  13.5][14.  14.5]]
读取的数组为: [[10. 10.] [11. 11.] [12. 12.]
 [13. 13.][14. 14.]]
```

任务2.2 利用Numpy进行统计分析

目录

2.2 利用Numpy进行统计分析

【任务描述】

在Numpy中，数组计算十分简单和快速。通常比python数组计算快许多，特别是在进行数组统计和分析的时候。利用Numpy中常见的函数进行统计分析还可以对数据进行排序，去重等操作。

【知识准备】

在Numpy中，除了可以使用通用函数对数组进行比较、逻辑等运算之外，还可以使用统计函数对数组进行排序、去重、求最大和最小值以及求均值等统计分析。