Python数据分析

Python数据分析经常使用的库包括NumPy、pandas、matplotlib以及IPython等，在python中，字符就是unicode字符，字符串就是unicode字符数组，windows新建的txt默认的编码是txt，有的时候网页的信息需要写到txt中需要改成with open('D:/DataMining/machinelearninginaction/test.txt', 'w', encoding='utf-8') as file:

**Python之父Guido推荐的命名规范包括如下几点：**

1. 模块名和包名采用小写字母并且以下划线分隔单词的形式；
2. 类名采用以大写字母开头，并且以大写字母分隔单词的形式命名；
3. 全局或者类常量，全部使用大写字母，并且以下划线分隔单词；其余变量命名则是采用全部小写字母，并且以下划线分隔单词的形式命名。
4. 以上的内容如果是内部的，则使用下划线开头命名。

NumPy

• 快速高效的多维数组对象ndarray。

• 用于对数组执行元素级计算以及直接对数组执行数学运算的函数。

• 用于读写硬盘上基于数组的数据集的工具。

• 线性代数运算、傅里叶变换，以及随机数生成。

• 用于将C、C++、Fortran代码集成到Python的工具。

SciPy是一组专门解决科学计算中各种标准问题域的包的集合，主要包括下面这些包：

• scipy.integrate：数值积分例程和微分方程求解器。

• scipy.linalg：扩展了由numpy.linalg提供的线性代数例程和矩阵分解功能。

• scipy.optimize：函数优化器（最小化器）以及根查找算法。

• scipy.signal：信号处理工具。

• scipy.sparse：稀疏矩阵和稀疏线性系统求解器。

• scipy.special：SPECFUN（这是一个实现了许多常用数学函数（如伽玛函数）的

Fortran库）的包装器。

• scipy.stats：标准连续和离散概率分布（如密度函数、采样器、连续分布函数

等）、各种统计检验方法，以及更好的描述统计法。

• scipy.weave：利用内联C++代码加速数组计算的工具。

NumPy跟SciPy的有机结合完全可以替代MATLAB的计算功能（包括其插件工具箱）。

Pandas：pandas兼具NumPy高性能的数组计算功能以及电子表格和关系型数据库（如SQL）灵活的数据处理功能。它提供了复杂精细的索引功能，以便更为便捷地完成重塑、切片和切块、聚合以及选取数据子集等操作。

IPython：是一个 python 的交互式 shell，比默认的python shell 好用得多，支持变量自动补全，自动缩进，

Python社区已经广泛接受了一些常用模块的命名惯例：

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

也就是说，当你看到np.arange时，就应该想到它引用的是NumPy中的arange函数。这样做的原因是：在Python软件开发过程中，不建议直接引入类似NumPy这种大型库的全部内容（from numpy import \*）。

冻结二进制文件能够将程序的字节码、PVM（Python Virtual Machine）以及任何程序需要的Python支持文件捆绑在一起形成一个文件包，过程会有一些不同，但是实际的结果将会是一个单独的可执行的二进制程序，（例如window中的.exe文件），这个文件很容易地像客户分发。Windows中使用py2exe作为冻结二进制程序。

字符串中常使用的函数：

Find

Replace

Pdb命令行调试器

IPython

Cpaste复制代码

内置函数

|  |  |
| --- | --- |
| 函数名称 | 参数介绍 |
| read() | 一次读取整个文件，生成一个字符串类型 |
| readline() | 一次只读取一行，生成一个str类型，速度比readlines()慢很多，当没有足够的内存时可以分多次读取 |
| readlines() | 一次读取全部的文件，按行读取，读取文件放在一个list中 |
| Type() | 查看对象类型 |
| Str[0],list[0] | 注意list的切片和str切片的区别，前者是按照，进行切片，后者是按照字符串进行切片。 |
| 变量命名切勿和系统命名冲突，有一次将变量命名成了list，然后调用list()函数一直报错 |  |
| **newArray.dtype和type(newArray)的区别？** | 应该前者是array里面数据类型，后者为对象类型 |
| **切片器的好处在于无论是系统自带还是ndarray还是dataframe都可以使用切片器** |  |
| len()等python系统自带的方法使用时需要注意的是使用环境，如果对ndarray使用时，则表示ndarray的维度，ndarray中需要使用shape进行获取array的大小 |  |
| **编程的过程中需要非常注意的一点是字符串的’1’是不等于数字的1，需要进行类型转换才可以** |  |
| **注意append()和extend()用法的区别，extend()可以带上list,需要非常注意的一点为print(reducedVector.extend(vector[axis + 1:]))和reducedVector.extend(vector[axis + 1:])之后再print(reducedVector)的数值的区别，前者是对extend方法调用的返回结果，为None，后者为打印list extend之后的结果的** | 多个list append之后的效果为[['1', 'yes'], ['1', 'yes'], ['0', 'no'], ['0', 'no']]，而使用extend之后的效果为['1', 'yes', '1', 'yes', '0', 'no', '0', 'no'] |
| **Set()是python中集合的概念，也可以用来取list中的distinct数值** |  |
| **Zip** | for a,n in zip(name,age):可以将name和age拼接起来，使用起来还是比较方便的  name=('jack','beginman','sony','pcky')  age=(2001,2003,2005,2000)  zip(name,age)可以完成类似dict拼接的操作，按照key、value取出。 |
| **Item()** | Dict中的item方法用来将字典中的每对key和value组成一个元素，如果单独用for循环，只会取出key或者value |
| setdefault() | 对于dict需要熟悉setdefault()方法的使用，经常用来初始化dict，会检查dict对应的key是否赋值，如果没有赋值会将其赋值，比多一个if判断简洁很多(if 'summary' in key: summary = key.summary) |
| Re.sub() | 正则表达式中的sub可以现在比replace更复杂的替换。例如：replacedStr = re.sub("\d+", "222", inputStr)可以将input中的所有数字替换成222 |
| Python中的dict可以层层堆叠，即一层dict外面再加一层dict |  |
| Strip() | 该方法在python中使用频度也比较高 |
| 全局变量 | 使用全局变量时需要在方法外先声明全局变量，全局变量应该使用全部大写进行声明  在方法内调用全局变量时需要先使用global关键字声明全局变量，通常应该尽量避免使用全局变量 |
| 注意也可以通过列表生成式生成对象的列表生成式，这就是Python最灵活的地方 |  |
| 注意list的pop, del, remove的使用，调用方式分别为 list.pop(1), list.remove(1), del list[1] | Pop返回的是被删除的list对应的数值，del也是删除对应位置的数据，remove()括号中删除的数值为对应list中的数值，而不表示对应list中的位置,三者使用之间的区别一定要清楚，如果需要删除元素，务必先删除list后面的元素，再删除list前面的元素，因为删除后面的元素不会影响到前面元素的位置 |
| tmpFile = open ('D: /tmp.txt', 'a') | 其中a表示文件可以续写 |
| [(min([tmp[i] for tmp in dataTmp]), max([tmp[i] for tmp in dataTmp])) for i in range(len(dataTmp))] | 注意列表生成式可以嵌套使用，这里面的i相当于外层的for循环 |
| print([0.0]\*10) | 注意这种方法使用的灵活性 |
| [value for (key, value) in scores] | 可以取出list中类似dict的key、value对应的key或者value值，这种方法使用起来非常非常灵活，太赞了。 |
| [(0, 9)] \* len(people) \* 2 | List的复制 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Pandas

|  |  |  |
| --- | --- | --- |
| 函数名称 | 参数介绍 | 说明 |
| Series | Series是一种类似于一维数组的对象，它由一组数据（各种Numpy数据类型）以及一组与之相关的数据标签（即索引）组成，与Numpy数组相比，可以使用索引的方式选取series中的单个或者一组数据print(s[['a','e']]) |  |
| Series可以看做使用索引的字段，所以可以通过字段创建Series | Series本身以及索引都有一个name属性。  s.name  a.index.name |  |
| Dataframe | Dataframe是一个表格型数据结构 |  |
| pd.DataFrame(np.random.randn(5,5),columns = ['year','state','pop','debt','haha'],index = ['one','two','three','four','five']) | 使用columns和index分别创建行和列索引，使用pd1.year可以选择某一列的数据 |  |
| Frame.T | 对frame进行转置 |  |
| Frame.columns.name  Frame.index.name | 对行、列索引设置名称 |  |
| Frame.values | 显示frame的数值，不会显示行、列索引名称 |  |
| merge(left, right, how='inner', on=None, left\_on=None, right\_on=None,  left\_index=False, right\_index=False, sort=True,  suffixes=('\_x', '\_y'), copy=True) | Left：左表名  Right：右表名  How：关联条件，inner，left，right，outer  On：关联字段，如果关联字段为多个，可写成on=['key1', 'key2']  Left\_on:左表关联字段  Right\_on:右表关联字段  left\_index:  right\_index: |  |
| s = pd.Series([1,3,5,np.nan,6,8]) | 创建Series |  |
| s.values() |  |  |
| df.head(3) | 数据帧的前三行 |  |
| df.tail(3) | 数据帧的最后三行 |  |
| df.dtypes | 数据帧字段类型 |  |
| df.index | 数据帧索引字段，索引字段是不可以修改的，可以使用reindex()进行修改。 |  |
| Df.index.name  Df.columns.name | 修改行、列名称 |  |
| Df.drop() | 可以删除特特定的列 |  |
| df.columns | 数据帧的行值（即第一行非数据） |  |
| Df.index | 数据帧的列值（即第一列非数据），注意这里的数值和df.index.name 和df.columns.name之间的区别，后者为列名，不是列值 |  |
| Pd.to\_cvs(header, index\_col) | 其中header用来指定读入的第一行数据是否为行，header= None表示读入的第一行收据不为行，index\_col用来指定读入的数据第一列是否为列名称，index\_col = ‘Blog’Blog为第一列第一行的数据表示第一列数据为列名称 |  |
| df.values | 数据帧值 |  |
| Df.shape[0], df.shape[1] | 分表代表行数 |  |
| df.describe() | 数据帧描述 |  |
| df.T | 数据帧的转置 |  |
| Df.ix | 可以丢dataframe进行行、列标签的检索，参加P143 |  |
| df.sort\_index(axis=1, ascending=False) | 根据第一行进行排序 |  |
| f.sort\_values(by='B') | 根据B列进行排序 |  |
| df['A'] | 选择A列 |  |
| df[0:3] | 选择0-3行 |  |
| df['20130102':'20130104'] | 选择2-4行 |  |
| df.loc[dates[0]] | 选择第一行 |  |
| df.loc[:,['A','B']] | 选择A,B列 |  |
| df.loc['20130102':'20130104',['A','B']] | 选择区域 |  |
| df.iloc[3]  df.iloc[3:5,0:2]  df.iloc[[1,2,4],[0,2]]  df.iloc[1:3,:]  df.iloc[:,1:3]  df.iloc[1,1] | 根据位置去选取，这种方式用的还比较方便 |  |
| df1.dropna(how='any') | 丢掉数据帧中nan的数据 |  |
| df1.fillna(value=5) | 数据帧中nan中数据用5填充 |  |
| df.mean() | 数据帧按照列维度的均值 |  |
| df.mean(1) | 数据帧按照行维度的均值 |  |
| df.apply(**lambda** x: x.max() - x.min()) | 函数应用于数据帧 |  |
| df2 = pd.concat([df1.iloc[:,0:3],df1.iloc[:,0:3]]) | 数据帧的拼接，其中方法中有个参数axis，默认值为0表示按列进行拼接，如果设置成1表示按行拼接，同样在pandas中很多时候axis表是选择坐标轴 |  |
| df.append(s, ignore\_index=True) | 同样数据帧的append也可以完成数据拼接额操作 |  |
| df.groupby(['A','B']).sum() | 数据的group by操作 |  |
| print(pd.pivot\_table(df3, values='D', index=['A', 'B'], columns=['C'])) | 数据透视图 |  |
| df.to\_csv('foo.csv') | 导出csv |  |
| pd.read\_csv('foo.csv') | 导入csv |  |
| df.to\_excel('foo.xlsx', sheet\_name='Sheet1') | 导出excel |  |
| pd.read\_excel('foo.xlsx', 'Sheet1', index\_col=None, na\_values=['NA']) | 导入excel |  |
| unstack() | 用来处理层次化分组 |  |
| stack() | 用来处理层次化分组 |  |
|  | | |
| Df.apply() | 注意三种方法的区别 |  |
| Df.applymap() |  |
| Df.map() |  |
| Df.sort\_index() | 排序 |  |
| Df的索引值可以相同，这一点需要注意一下 | 可以使用obj.index.is\_unique()进行判断是否重复 |  |
| Axis参数可以对df进行或者列操作 | Axis=0表示对行进行操作，为1表示对列进行操作 |  |
| Count() describe() min(), max(), argmin(), argmax(),idxmin(), idxmax(), quantile(), sum(),mean(), median(), mad(), var(),std() | 分别表示数据的非NA数量，汇总信息，最小最大值，最小最大值的索引位置，最小最大值的索引值，样本分位点，值的总和，值的平均数，值的算术中位数，平均绝对离差，方差，标准差 |  |
| Dropna(),fillna(),isnull(),notnull()  Replace() | 删掉NAN，对NAN的数值进行填充 |  |
| 层次化索引，层次化索引在数据重塑和数据分组的操作中（比如透视表）中扮演着非常重要的角色，可以使用stack()和unstack()进行数据重排，可以使用swaplevel()对索引数值进行交换，并且使用sortlevel()对交换后的索引进行排序 |  |  |
| df.to\_csv('D:/2.csv',index = False, header = False) | 可以只控制不输出行和列索引值 |  |
| xls = pd.ExcelFile('D:/2.xlsx')  table = xls.parse('Sheet1')#读入excel内容 | 读取excel内容 |  |
| sql.read\_frame('select \* from ABSOLUTE ') |  |  |
| pd.merge(df1,df2,how='inner',left\_on='',right\_on='') |  |  |
| Pd.concat()  Np.where  Combine\_first() |  |  |
| Df.duplicated() | 数据是否重复 |  |
| df.drop\_duplicates() |  |  |
| 离散数据的分割问题 | P222 |  |
| Split() strip() |  |  |
| 常规字符串处理方法 | P228 |  |
|  |  |  |
|  |  |  |
|  |  |  |

Numpy

|  |  |  |
| --- | --- | --- |
| 函数名称 | 参数介绍 | 说明 |
| ndarray | n维数据对象，该对象是一个灵活的大数据容器，ndarray是一个通用的同构数据多维容器，**其中的所有元素必须是相同类型的**，每个数组都有一个shape(各维度数组的大小)和一个dtype(一个用户说明数组数据类型的对象)，生成ndarray最简单的办法就是使用array函数，它接受一切的序型的对象。 |  |
| Numpy中存在两种不同的数据类型，一种是array，一种是matrix，可以调用mat()函数将数组转化为矩阵 | | |
| 可以将python中的list转化成numpy中的array，使用array()方法即可转换，需要非常注意的一点是numpy中方法例如arange(),zeros()等生成的数据格式均是array()格式，而不是list格式，注意不同方式之间生成的数组，数据之间是否有空格 | | |
| .Id | 得到矩阵的逆 |  |
| np.arange(1,10,0.1) | 以0.1为步长进行增长，注意和python自带的range作一区分 |  |
| arr.shape  arr.dtype  arr.ndim | arr的几种基本属性 |  |
| np.arange | numpy 的内置range版本 |  |
|  |  |  |
| np.zeros((2,2)) | 生成全0数组,注意是两个括号，类似的方法还是ones(),eye(),empty() |  |
| ndarray | 同构数据多维容器，每一个数组都有一个shape(表示各个维度大小的元组)和一个dtype(一个用于说明数组类型的对象) |  |
| np.zeros() | 生成全零的数组 |  |
| np.ones() | 生成全一的数组 |  |
| arr2 = arr.astype('int64') | 转换数据类型 |  |
| reshape() | 改变矩阵的行列 |  |
| fancy indexing | 花式索引 |  |
| arr[[4,3,0,6]] | 可以选取数组中第4,3,0,6行数据,**即数组也具有切片性质，在使用切片的过程中需要非常注意的一点为numpy的数字只要对切片做任何的修改，即对原来的数据进行了修改，这和之前的python中变量的复制差异区别很大，主要原因为numpy用来处理大量的数据，大量的数据多个备份之间进行复制效率较低，如果要产生一个副本要使用copy()方法** |  |
| arr[[4,3,0,6],[ 4,3,0,6]] | 可以选取第4行、第4列，第3行第3列的数据 |  |
| np.random.randn(7,4) | 生成随机数 |  |
| arr.t | 数据的转置 |  |
| np.dot(arr.t,arr) | 矩阵的内积 |  |
| 更多具体的函数参加书中p110 |  |  |
| np.aqrt(arr) | 直接对数组进行开方操作 |  |
| np.maximum(x,y) | 对二元数组进行操作 |  |
| np.where(arr >0,2,-2) | 大于0则变成2，否则变成-2 |  |
| mean等聚合函数，既可以当做数组的实例方法调用，也可以当做顶级numpy函数使用 | arr.mean()等价于np.mean(arr) |  |
| arr1[arr1 < 0] = 0 | 将数组中小于0的数据都置为0 |  |
| arr.sort() | 数组排序 |  |
| np.unique() |  |  |
| x.dot(y) 相当于np.dot(x,y) | 矩阵的点乘 |  |
| Np.abs(frame) | 可以使用numpy的函数对df进行操作，也可以使用Frame.apply(f)对df进行操作 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Numpy数组的好处在于可以使用数组表达式代替循环，通常被称为是量化，一般来说，是量化数组运营要的纯Python方式快上一两个等级，甚至更多 |  |  |
|  |  |  |
|  |  |  |

Matplotlib

爬虫部分

Beautifusoup

|  |  |
| --- | --- |
| 常用方法 | 方法说明 |
| BeautifulSoup(c.read(), 'lxml') | 创建对象 |
| soup('a') | 对网页进行过滤 |
| link.attrs | 网页的属性，包括href等 |
|  |  |
|  |  |
|  |  |
|  |  |

数据可视化包：Bokeh, matplotlib, seaborn,ggplot

Blaze 是下一代的 NumPy

Python常用包

|  |  |
| --- | --- |
| 包名称 | 主要用途 |
| Scrapy | 网络爬虫 |
| Scikit-learn | 基于numpy和scipy的开源机器学习模块，包括分类、回归、聚类算法、主要算法有SVM、逻辑回归、朴素贝叶斯、Kmeans、DBSCAN等 |
| NLTK(Natural Language Toolkit) | 自然语言处理模块，包括字符传力和语言统计模型。 |
| Urllib urllib2 | 打开网页，下载文件，python3对urllib和urllib2进行了重构，拆分成了urllib.request, urllib.response, urllib.parse, urllib.error等几个子模块，这样的架构从逻辑和结构上说更加合理。在python3中需要使用import urllib.request |
| Requests | 提交各种类型的请求，支持重定向 |
| Beautifulsoup | 解析网页，对于语法不是很规范的web页面，beautiful soup具有很好的容错性，在python3中beautiful soup被缩写成了bs4 所以需要使用import bs4 |
| scikit-learn | 数据挖掘主要包 |
| from sqlite3 import dbapi2 as sqlit | Python中使用sqlite数据库 |
| import xml.dom.minidom | 可以用来解析XML |
|  |  |