Numpy的RandomState

rng = numpy.random.RandomState(23355)

arrayA = rng.uniform(0,1,(2,3))

该段代码的目的是产生一个2行3列的assarray，其中的每个元素都是[0,1]区间的均匀分布的随机数。这里可以看到有一个23355这个数字，其实，它是伪随机数产生器的种子对于某一个伪随机数发生器，只要该种子（seed）相同，产生的随机数序列就是相同的

Python中文网站http://python.usyiyi.cn/

ensemble复合，组合 regressor回归 estimator评价者，评估者

对于例如随机森林（RF），Bagging,boosting这些将弱分类器进行组合的集成学习算法使用时导入

from sklearn import ensemble

1.Numpy其shape的使用



e=[[ 1. 1.1]

[ 1. 1. ]

[ 0. 0. ]

[ 0. 0.1]]

则e.shape显示它的行列数，是4L,2L

e.shape[0]，表示输出行数4，e.shape[1]，表示输出列数2，

**2.[python中argsort()函数和sort、sorted函数以及lambda创建匿名函数的用法](http://www.cnblogs.com/yyxf1413/p/6253995.html)**

sort 是应用在 list 上的方法，sorted 可以对所有可迭代的对象进行排序操作。list的sort方法返回的是对已经存在的列表进行操作，而内建函数 sorted 方法返回的是一个新的 list，而不是在原来的基础上进行的操作。

Sorted的语法：

sorted(iterable[, cmp[, key[, reverse]]])

参数说明：

iterable -- 可迭代对象。

cmp -- 比较的函数，这个具有两个参数，参数的值都是从可迭代对象中取出，此函数必须遵守的规则为，大于则返回1，小于则返回-1，等于则返回0。

key -- 主要是用来进行比较的元素，只有一个参数，具体的函数的参数就是取自于可迭代对象中，指定可迭代对象中的一个元素来进行排序。

reverse -- 排序规则，reverse = True 降序 ， reverse = False 升序（默认）。

返回值：

返回重新排序的列表

案例：

students = [('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]

sorted(students, key=lambda s: s[2], reverse=True) # 按年龄降序排序

[('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]

注意：以上的lambda用于创建一个匿名函数，lambda只是一个表达式，它可以直接作为[python 列表](http://www.cnpythoner.com/post/50.html" \t "https://www.cnblogs.com/wanpython/archive/2010/11/01/_blank)或[python 字典](http://www.cnpythoner.com/post/62.html" \t "https://www.cnblogs.com/wanpython/archive/2010/11/01/_blank)的成员，此外像if或for或print这种语句就不能用于lambda中，lambda一般只用来定义简单的函数。

sorted(students, key=lambda s: s[2], reverse=True)意思是把students作为lambda匿名函数的输入，取列表中元组的第二项age作为key逆序进行排序

sorted对字典按值进行排序

import operator

d={**'a'**:1,**'b'**:3}  
dd=sorted(d.iteritems(),key=operator.itemgetter(1),reverse=True)

返回的dd是排序好的列表[('b', 3), ('a', 1)]

对于一个数值型列表，再加入一个与其对应的字符串型列表，一块组成迭代对象，按数值型列表进行排序，返回的是数值以及其最开始与其对应的字符串。

print(sorted(zip(map(**lambda** x: round(x, 4), rf.feature\_importances\_), names), reverse=**True**))

argsort()函数

import numpy as np

x=np.array([1,4,3,-1,6,9])

x.argsort()

我们发现argsort()函数是将x中的元素从小到大排列，提取其对应的index(索引)，然后输出到y。例如：x[3]=-1最小，所以y[0]=3,x[5]=9最大，所以y[5]=5

当num>=0时，np.argsort()[num]就可以理解为y[num];

当num<0时，np.argsort()[num]就是把数组y的元素反向输出，例如np.argsort()[-1]即输出x中最大值对应的index，np.argsort()[-2]即输出x中第二大值对应的index，依此类推。

该函数返回的是升序排序后的索引值，若想得到降序排序的索引值，再进一步用numpy的array的选择作用，常用切片进行反转，技巧如下：

movie\_ranks[np.argsort(movie\_ranks[:,0])[::-1]]

如上代码是先按movie\_ranks的第0列进行排序，[::-1]这样取切片得到的就是降序排序的索引值。

Python中\*\*两个[乘号](https://www.baidu.com/s?wd=%E4%B9%98%E5%8F%B7&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPjm3njDkPj6Yn1f1Pjc10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHRYnjc1PHTLnHcknHDdn1TYPs" \t "https://zhidao.baidu.com/question/_blank)就是[乘方](https://www.baidu.com/s?wd=%E4%B9%98%E6%96%B9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPjm3njDkPj6Yn1f1Pjc10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHRYnjc1PHTLnHcknHDdn1TYPs" \t "https://zhidao.baidu.com/question/_blank)，比如2\*\*4,结果就是2的4次方

2\*\*0.5相当于

3.numpy.tile()的用法

numpy.tile()是个什么函数呢，说白了，就是把数组沿各个方向复制

例子a = np.array([0, 1, 2]) **>>>**np.tile(a, 2) array([0, 1, 2, 0, 1, 2])

**>>>**np.tile(a, (2, 2)) array([[0, 1, 2, 0, 1, 2], [0, 1, 2, 0, 1, 2]])

**>>>**np.tile(a, (2, 1, 2)) array([[[0, 1, 2, 0, 1, 2]], [[0, 1, 2, 0, 1, 2]]])

**>>>**b =np.array([[1, 2], [3, 4]])np.tile(b, 2) array([[1, 2, 1, 2], [3, 4, 3, 4]])

**>>>**np.tile(b, (2, 1)) array([[1, 2],[3, 4], [1, 2], [3, 4]])

**>>>**c = np.array([1,2,3,4])

np.tile(c,(4,1))

array([[1, 2, 3, 4],

[1, 2, 3, 4],

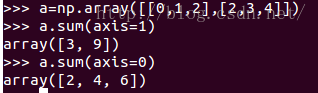
[1, 2, 3, 4],

[1, 2, 3, 4]])

1. numpy.sum()函数的使用

默认axis为None，表示将所有元素的值相加

对于一维数组而言，只有axis=0可以使用（没必要使用）axis=1表示按行相加 , axis=0表示按列相加

例子：

5.字典的get函数

dict\_data ={1:'one',2:'two',3:'three',4:'four'}

print(dict\_data.get(1))

print(dict\_data.get(3))

print(dict\_data.get(5))

print(dict\_data.get(5,'notfound'))#不存在时默认的值

输出：

one

three

None

not found

常用的是取字典的键对应的值，累计计数

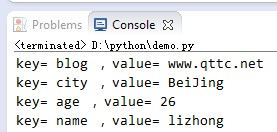
dict.get(key,0)+1 意思为取key对应的值，若不存在则初始为0再+1

6.python中字典的items()和iteritems()函数的用法

items函数，将一个字典以列表的形式返回，因为字典是无序的，所以返回的列表也是无序的，iteritems()返回一个迭代器

item()方法把字典中每对key和value组成一个元组，并把这些元组放在列表中返回。

person={'name':'lizhong','age':'26','city':'BeiJing','blog':'www.jb51.net'}  
for key,value in person.items():  
    print 'key=',key,'，value=',value



可见key接收了字典的key，value接收了字典的value值

person={'name':'lizhong','age':'26','city':'BeiJing','blog':'www.jb51.net'}  
for x in person.items():  
    print x



只有一个变量接收值的情况下，直接返回的是每一对key,value对应的元组

7.list的切片

A[0:2]取A列表第一行第二行所有列

A[1,:]取A列表第二行所有列

A[0:2,0:2]取A列表第一二行第一二列

L=range(10)  
**print** L[:-10:-1]意思是从后面开始输出，步长为1取到倒数第十个不包括倒数第十取倒数前9个。

输出：[9, 8, 7, 6, 5, 4, 3, 2, 1]

8.numpy的mat函数

把列表等转换成矩阵便于计算

9.矩阵中元素的排序sort

xMat=mat(xArr),xMat是numpy中矩阵。则xMat.sort(0)表示对矩阵中的每列进行升序排序，xMat.sort(1)表示对矩阵中的每行进行升序排序。

10.Matlab中eye()函数

Y = eye(n)：返回n\*n单位矩阵；

Y = eye(m,n)：返回m\*n单位矩阵；

Y = eye([m n])：返回m\*n单位矩阵；

Y = eye(size(A))：返回与A一样大小的单位矩阵；

11.[numpy中np.nonzero()函数](http://blog.csdn.net/u013698770/article/details/54632047)

参考网址：<http://blog.csdn.net/u013698770/article/details/54632047>

12.Python中的Map函数和astype函数进行数据类型转换

map函数的原型是map(function, iterable, …)，它的返回结果是一个列表。

参数function传的是一个函数名，可以是python内置的，也可以是自定义的。   
参数iterable传的是一个可以迭代的对象，例如列表，元组，字符串这样的。

这个函数的意思就是将function应用于iterable的每一个元素，结果以列表的形式返回。注意到没有，iterable后面还有省略号，意思就是可以传很多个iterable，如果有额外的iterable参数，并行的从这些参数中取元素，并调用function。如果一个iterable参数比另外的iterable参数要短，将以None扩展该参数元素

具体实例可参考：http://blog.csdn.net/csdn15698845876/article/details/73321593?locationNum=6&fps=1

**Numpy中的astype函数**

此函数用于数据类型的转化*(array(x[1:])).astype(float32)*

把x数组中的值都转换为float32类型

1. numpy中random的用法

具体参照：https://blog.csdn.net/kancy110/article/details/69665164

np.random.rand(3,2) 生成3行2列0-1之间的数

array([[ 0.14022471, 0.96360618], #random

[ 0.37601032, 0.25528411], #random

[ 0.49313049, 0.94909878]]) #random

**print** (np.random.rand(2,4))*#随机数组指定行和列注意随机出的数小于1***print** (np.random.rand()) *#随机生成一个0-1之间的浮点数*

**print** (np.random.rand(1)) *#随机生成一个只含一个（0-1）之间元素的数组***print** (np.random.rand(10)) *#生成10个0-1之间的一维数组*

* numpy.random.randn(d0, d1, ..., dn)：生成一个浮点数或N维浮点数组，取数范围：正态分布的随机样本数。

1. #numpy.random.randn(d0, d1, ..., dn)
2. **import** numpy as np
3. #无参
4. np.random.randn()#1.4872544578730051，不一定是[0,1)之间的随机数
5. #一个参数
6. np.random.randn(1)
7. np.random.randn(5)#生成形状为5的一维数组
8. #两个参数
9. np.random.randn(2,3)#生成2x3数组
10. #np.random.randn((2,3))#报错,参数必须是整数

* numpy.random.standard\_normal(size=None)：生产一个浮点数或N维浮点数组，取数范围：标准正态分布随机样本

1. **import** numpy as np
2. #numpy.random.standard\_normal(size=None)
3. #size为整数
4. np.random.standard\_normal(2)#array([-2.04606393, -1.05720303])
5. #size为整数序列
6. np.random.standard\_normal((2,3))
7. np.random.standard\_normal([2,3]).shape#(2, 3)

numpy.random.randint(low, high=None, size=None, dtype='l')：生成一个整数或N维整数数组，取数范围：若high不为None时，取[low,high)之间随机整数，否则取值[0,low)之间随机整数。

1. **import** numpy as np
2. #low=2
3. np.random.randint(2)#生成一个[0,2)之间随机整数
4. #low=2,size=5
5. np.random.randint(2,size=5)#array([0, 1, 1, 0, 1])
6. #low=2,high=2
7. #np.random.randint(2,2)#报错,high必须大于low
8. #low=2,high=6
9. np.random.randint(2,6)#生成一个[2,6)之间随机整数
10. #low=2,high=6,size=5
11. np.random.randint(2,6,size=5)#生成形状为5的一维整数数组
12. #size为整数元组
13. np.random.randint(2,size=(2,3))#生成一个2x3整数数组,取数范围：[0,2)随机整数
14. np.random.randint(2,6,(2,3))#生成一个2x3整数数组,取值范围：[2,6)随机整数
15. #dtype参数：只能是int类型
16. np.random.randint(2,dtype='int32')
17. np.random.randint(2,dtype=np.int32)
18. numpy.random.uniform(low,high,size)

功能：从一个均匀分布[low,high)中随机采样，注意定义域是左闭右开，即包含low，不包含high.  
参数介绍:    
    low: 采样下界，float类型，默认值为0；  
    high: 采样上界，float类型，默认值为1；  
    size: 输出样本数目，为int或元组(tuple)类型，例如，size=(m,n,k), 则输出m到n的k个样本，k缺省时输出1个值。  
返回值：ndarray类型，其形状和参数size中描述一致。

15.论numpy中matrix 和 array的区别

<http://blog.csdn.net/vincentlipan/article/details/20717163>

与许多矩阵语言不同，乘法运算符\*的运算在NumPy数组中是元素级别的。可以使用dot函数或方法执行矩阵乘积：

A = np.array( [[1,1],[0,1]] )

B = np.array( [[2,0],[3,4]] )

A\*B *# elementwise product*

**>>>**array([[2, 0],

[0, 4]])

1. dot(B)*# matrix product*

**>>>**array([[5, 4],

[3, 4]])

np.dot(A, B)*# another matrix product*

**>>>**array([[5, 4],

[3, 4]])

15.append（追加）和extend（扩展）的区别

<http://justjavac.iteye.com/blog/1827915>

Lists 的两个方法 extend 和 append 看起来类似，但实际上完全不同。extend 接受一个参数，这个参数总是一个 list，并且把这个 list 中的每个元素添加到原 list 中。

在这里list 中有 3 个元素 ('a'、'b' 和 'c')，并且使用另一个有 3 个元素 ('d'、'e' 和 'f') 的 list 扩展之，因此新的 list 中有 6 个元素。

另一方面，append 接受一个参数，这个参数可以是任何数据类型，并且简单地追加到 list 的尾部。在这里使用一个含有 3 个元素的 list 参数调用 append 方法。

原来包含 3 个元素的 list 现在包含 4 个元素。为什么是 4 个元素呢？因为刚刚追加的最后一个元素本身是个 list。List 可以包含任何类型的数据，也包括其他的 list。这或许是您所要的结果，或许不是。如果您的意图是 extend，请不要使用 append。

1. >>> li = ['a', 'b', 'c']
2. >>> li.extend(['d', 'e', 'f'])
3. >>> li
4. ['a', 'b', 'c', 'd', 'e', 'f']
5. >>> len(li)
6. 6
7. >>> li[-1]
8. 'f'
9. >>> li = ['a', 'b', 'c']
10. >>> li.append(['d', 'e', 'f'])
11. >>> li
12. ['a', 'b', 'c', ['d', 'e', 'f']]
13. >>> len(li)
14. 4
15. >>> li[-1]
16. ['d', 'e', 'f']

注意在对单元素进行操作时追加和扩展不同

A.extend([1])其中[]是必须的,因为参数必须是一个List

16.利用import matplotlib.pyplot as plt绘图

Plt.figure()#创建画布

x轴上下限设定xlim([a,b]);

y轴上下限设定ylim([a,b])

xticks和yticks: 为x,y轴的主刻度和次刻度设置颜色、大小、方向，以及标签大小

# [meshgrid函数的用法](http://www.cnblogs.com/haore147/p/3633086.html)

[A,B]=Meshgrid(a,b)  
生成size(b)Xsize(a)大小的矩阵A和B。它相当于a从一行重复增加到size(b)行，把b转置成一列再重复增加到size(a)列。

>> a=[1:2]

a =

     1     2

>> b=[3:5]

b =

     3     4     5

[A,B]=meshgrid(a,b)

A =

     1     2

     1     2

     1     2

B =

     3     3

     4     4

     5     5

[B,A]=meshgrid(b,a)

B =

     3     4     5

     3     4     5

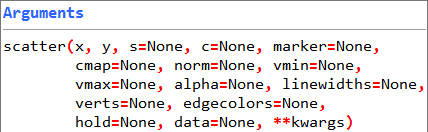
A =

     1     1     1

     2     2     2

plt.subplot(221)添加子画布，把画布划分为两行两列共四个模块，从左到右，从上到下分别是1234.

散点图绘制：



x,y是需要绘制散点的x,y坐标，s 点的大小（也就是面积）默认是20，c指点的颜色或颜色序列，默认为蓝色，其它的还可用。

c = 'r' (red); c = 'g' (green); c = 'k' (black) ; c = 'y'(yellow)



marker 形状，可选值，默认是圆。一般常用可选x（×），^（△），o（默认的圆），+（+），\*（\*）

**点型（标记marker）：**

像素： ','

圆形： 'o'

上三角： '^'

下三角： 'v'

左三角： '<'

右三角： '>'

方形： 's'

加号： '+'

叉形： 'x'

棱形： 'D'

细棱形： 'd'

三脚架朝下： '1'（像'丫'）

三脚架朝上： '2'

三脚架朝左： '3'

三脚架朝右： '4'

六角形： 'h'

旋转六角形： 'H'

五角形： 'p'

垂直线： '|'

水平线： '\_'

plt.scatter(x,y,c=**'r'**,marker=**'^'**,alpha=0.5,label=**'C1'**)

其中alpha=0.5点点的亮度，label，标签（注意要显示标签必须添加plt.legend(loc=**'lower right'**)这行代码，其中upper是上lower是下，left是左，right是右）

plt.title(**u'绘图'**)*#显示图表标题*plt.xlabel(**u'x轴'**)*#x轴名称*plt.ylabel(**u'y轴'**)*#y轴名称*plt.grid(True)*#显示网格线*

注意上面的汉字会显示乱码要在导入下面添加

plt.rcParams[**'font.sans-serif'**]=[**'SimHei'**] *#用来正常显示中文标签*plt.rcParams[**'axes.unicode\_minus'**]=False *#用来正常显示负号*

参考网址如下：

http://blog.csdn.net/luoganttcc/article/details/64440091

<http://blog.csdn.net/bbbeoy/article/details/71750027>

用plot()函数绘制线形图：

线型（linestyle 简写为 ls）：

实线： '-'

虚线： '--'

虚点线： '-.'

点线： ':'

点： '.'

pl.xlim(-1, 11) # 限定横轴的范围

pl.ylim(-1, 110) # 限定纵轴的范围

plt.plot(x, y, format\_string, \*\*kwargs): x为x轴数据，可为列表或数组；y同理；format\_string 为控制曲线的格式字符串， \*\*kwargs 第二组或更多的（x, y, format\_string）

format\_string: 由 颜色字符、风格字符和标记字符组成。

eg： plt.plot(a, a\*1.5,‘go-’,a, a\*2,‘\*’)第二条无曲线，只有点

pl.plot(x,y,'ob-',label=u'y=x^2曲线图',linewidth=2)

'ob-'中的o表示显示原点用圆显示,b表示线的颜色,-表示线条是实线

17.Python内置的range()和Numpy中的arange()函数的区别

Range(5)只有一个参数则默认从0开始步长为1输出0,1,2,3,4

Range(1,5)此时从1开始步长为1输出1,2,3,4

Range(1,5,2)此时从1开始步长为2输出1,3.注意写为range(5,2)并不会代表从0开始到5步长为2，这是不对的，此外对于步长也不可设置为小数

arange()函数具备以上特性但是其步长可以设置为小数。同时它返回的是一个ndarray()所以具备reshape()属性

当arange与浮点参数一起使用时，由于浮点数的精度是有限的，通常不可能预测获得的元素数量。出于这个原因，通常最好使用函数linspace，它接收我们想要的元素数量而不是步长作为参数：

from numpy import pi

np.linspace( 0, 2, 9 )# 9 numbers from 0 to 2

array([ 0.,0.25,0.5,0.75,1.,1.25,1.5,1.75,2.])

x = np.linspace( 0,2\*pi, 100 )# useful to evaluate function at lots of points

f = np.sin(x)

注意：range()是Python的内置函数调用应直接调用range()，

np.arange()才是numpy的内置函数



18.Anaconda安装包

Conda install 包的名字

19.字典的update函数

dict = {'Name': 'Zara', 'Age': 7}

dict2 = {'Sex': 'female' }

dict.update(dict2)

Value : {'Age': 7, 'Name': 'Zara', 'Sex': 'female'}

20.遍历多个序列

同时遍历两个或更多的序列，使用[zip()](http://python.usyiyi.cn/documents/python_278/library/functions.html" \l "zip" \o "zip)函数可以成对读取元素。

questions = ['name', 'quest', 'favorite color']

answers = ['lancelot', 'the holy grail', 'blue']

for q, a in zip(questions, answers):

print 'What is your {0}? It is {1}.'.format(q, a)

What is your name? It is lancelot.

What is your quest? It is the holy grail.

What is your favorite color? It is blue.

21.列表中放字典取值

List=[{'Age': 7, 'Name': 'Zara'},{'Sex': 'female'}]

List[0][‘Age’]=7

22.特征缩放

当以两个特征的和作为一个特征参与对结果类型进行预测时，其中一个特征的取值范围过大，导致其对结果的影响度完全其了主导作用而使得另一特征失去作用，（例如根据身高、体重之和判断一个人应该穿衣服的尺码，身高的取值范围相比体重的取值范围过大，而又想使得这两个特征对预测结果起同等重要的作用）这时应该用特征缩放。特征缩放的公式是(x-x(min))/(x(max)-x(min))根据这个公式所计算得到的值在0-1之间这既是优点又是缺点，优点是这会使得预估输出相对稳定，缺点是当特征中存在异常值时将会使得特征缩放处理变得非常麻烦。

案例：

*#encoding=utf-8***from** sklearn.preprocessing **import** MinMaxScaler  
**from** numpy **import** \*  
weights=array([[115.],[140.],[175.]]) *#之所以这样构造因为（）中的[]必须的，其余的每个值要代表该特征的一个样本数据因此要构造成3行1列*scale=MinMaxScaler()  
scaled\_weights=scale.fit\_transform(weights) *#注意调用此函数缩放的数组，其中的元素必须是小数***print** scaled\_weights

利用RBF核的SVM以及k-means算法会受特征缩放的影响，而决策树和逻辑回归不会

23.异常值处理

可能出现异常值的情况：

以下情况可能造成异常值的产生。1.感知器的错误。2.人为的输入错误。3.外部条件造成。4.遇到了极端情况的数据。前两个一般不用注意，后两个一些领域会注意。

异常值的处理方法：

1. 训练所有数据。2）找出训练之后，训练集中错误最多的点，去除这些点，一般占数据集的10%。3）再次训练新的数据。

24.reshape和resize的区别numpy中数组的重塑

array([[ 2., 8., 0., 6.],

[ 4., 5., 1., 1.],

[ 8., 9., 3., 6.]])

1. ravel() *# returns the array, flattened*

array([ 2., 8., 0., 6., 4., 5., 1., 1., 8., 9., 3., 6.])这是一个一维数组，无行和列的概念

注意它和a.reshape(1,-1)不一样,它返回的是二维数组有行和列的概念array([[ 2., 8., 0., 6., 4., 5., 1., 1., 8., 9., 3., 6.]])

a.resize((2,6))

array([[ 2., 8., 0., 6., 4., 5.],

[ 1., 1., 8., 9., 3., 6.]])

Reshape和resize的区别

Reshape有返回值，它不会直接对原来的数组进行修改，而是开辟一个新的空间，而resize无返回值，也就是它会直接对原数组进行修改。

25.关于读取文本的split()和strip()

Python split()通过指定分隔符对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串

str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。

str = "Line1-abcdef \nLine2-abc \nLine4-abcd";

print str.split( );

print str.split(' ', 1 );

以上实例输出结果如下：

['Line1-abcdef', 'Line2-abc', 'Line4-abcd']

['Line1-abcdef', '\nLine2-abc \nLine4-abcd']

一般分隔字符为字表符‘\t’或者是空格’ ’注意该函数能够识别出换行符，这两个分隔只是一行数据每个数据间的分隔

str = "Chris\_iven+Chris\_jack+Chris\_lusy"

print str.split("+")

print str.split("\_")

输出结果为：

['Chris\_iven', 'Chris\_jack', 'Chris\_lusy']

['Chris', 'iven+Chris', 'jack+Chris', 'lusy']

Python strip() 方法用于移除字符串头尾指定的字符（默认为空格以及换行符）

strip()方法语法：

str.strip([chars]);

参数

* chars -- 移除字符串头尾指定的字符。

返回值

返回移除字符串头尾指定的字符生成的新字符串。

实例

以下实例展示了strip()函数的使用方法：

str = "0000000this is string example....wow!!!0000000";

print str.strip( '0' );

以上实例输出结果如下：

this is string example....wow!!!

26.numpy.linalg模块

numpy.linalg模块包含线性代数的函数。使用这个模块，可以计算逆矩阵、求特征值、解线性方程组以及求解行列式等

# 使用inv函数计算逆矩阵  
inv = np.linalg.inv(A)

在计算逆矩阵之前要判断矩阵是否可逆，可以转换为求矩阵的行列式是否为0，不为0则可逆

# 使用det函数计算行列式  
print (np.linalg.det(F))

特征值和特征向量  
特征值（eigenvalue）即方程 Ax = ax 的根，是一个标量。其中，A 是一个二维矩阵，x 是一个一维向量。特征向量（eigenvector）是关于特征值的向量  
# numpy.linalg模块中，eigvals函数可以计算矩阵的特征值，而eig函数可以返回一个包含特征值和对应的特征向量的元组

# 调用eigvals函数求解特征值  
c0 = np.linalg.eigvals(C)  
print (c0)  
# [ 2. 1.]

# 使用eig函数求解特征值和特征向量

c1,c2 = np.linalg.eig(C)  
print (c1)  
# [ 2. 1.]   
print (c2)  
#[[ 0.89442719 0.70710678] #2.对应的特征向量  
# [ 0.4472136 0.70710678]]

4.奇异值分解  
# SVD（Singular Value Decomposition，奇异值分解）是一种因子分解运算，将一个矩阵分解为3个矩阵的乘积  
# numpy.linalg模块中的svd函数可以对矩阵进行奇异值分解。该函数返回3个矩阵——U、Sigma和V，其中U和V是正交矩阵，Sigma包含输入矩阵的奇异值。

import numpy as np

# 分解矩阵  
D = np.mat("4 11 14;8 7 -2")  
# 使用svd函数分解矩阵  
U,Sigma,V = np.linalg.svd(D,full\_matrices=False)

使用np.linalg.norm求向量的范数

顾名思义，linalg=linear+algebra，norm则表示范数，首先需要注意的是范数是对向量（或者矩阵）的度量，是一个****标量（scalar）****：

首先help(np.linalg.norm)查看其文档：

norm(x, ord=None, axis=None, keepdims=False)

这里我们只对常用设置进行说明，x表示要度量的向量，ord表示范数的种类，

| **参数** | **说明** | **计算方法** |
| --- | --- | --- |
| 默认 | 二范数：ℓ2 | x21+x22+…+x2n−−−−−−−−−−−−−−√ |
| ord=2 | 二范数：ℓ2 | 同上 |
| ord=1 | 一范数：ℓ1 | |x1|+|x2|+…+|xn| |
| ord=np.inf | 无穷范数：ℓ∞ | max(|xi|) |

numpy的logical\_and用法

Examples

np.logical\_and(**True**, **False**)

False

np.logical\_and([**True**, **False**], [**False**, **False**])

array([False, False], dtype=bool)

x = np.arange(5)

np.logical\_and(x>1, x<4)

array([False, False, True, True, False], dtype=bool)

27.异常处理和输出格式展示

**import** **math**

**while** True:

**try**:

text = raw\_input('> ')

**if** text[0] == 'q':

**break**

x = float(text)

y = math.log10(x)

**print** "log10({0}) = {1}".format(x, y)

**except** **ValueError**:

**print** "the value must be greater than 0"

一旦 try 块中的内容出现了异常，那么 try 块后面的内容会被忽略，****Python****会寻找 except 里面有没有对应的内容，如果找到，就执行对应的块，没有则抛出这个异常。

28.numpy的linspace()函数

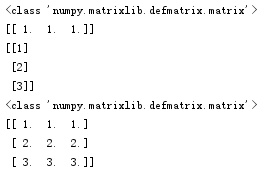
该函数用于构造等差数列np.linspace(0.5,9.5,10)从0.5到9.5构造含十个数的等差数列

# 29.numpy中的np.dot, np.multiply, \*

np.multiply对于矩阵而言是对应的元素相乘

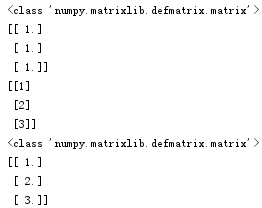
例1：

**from** numpy **import** \*  
a=mat(ones((1,3)))  
**print** type(a)  
**print** a  
b=[[1],[2],[3]]  
b=mat(b)  
**print** b  
**print** type(b)  
**print** multiply(a,b)



案例2.

修改上面的a=mat(ones((3,1)))

输出为：

np.multiply对于numpy的数组而言也是对应的元素相乘案例输出和上面相同，只不过类型是

因为np.multiply对于数组和矩阵而言都是对应的元素相乘，所以其中一个是矩阵一个是数组也是对应的元素相乘

\*对于数组而言也是对应的元素相乘，所得结果和用np.multiply操作数组一样

\*对于矩阵而言是运行矩阵相乘（注意一个乘数是矩阵一个是数组也是矩阵相乘）

dot()不管对于两个矩阵、两个数组、还是一个矩阵一个数组而言都是运行的矩阵相乘

30.enumerate()函数用法解析

enumerate()是python的内置函数

对于一个可迭代的（iterable）/可遍历的对象（如列表、字符串），enumerate将其组成一个索引序列，利用它可以同时获得索引和值

list1 = ["这", "是", "一个", "测试"]

for index, item in enumerate(list1):

print index, item

0 这

1 是

2 一个

3 测试

enumerate还可以接收第二个参数，用于指定索引起始值，

如：

list1 = ["这", "是", "一个", "测试"]

for index, item in enumerate(list1, 1):

print index, item

>>>1 这2 是3 一个4 测试

补充

如果要统计文件的行数，可以这样写：

count = len(open(filepath, 'r').readlines())

这种方法简单，但是可能比较慢，当文件比较大时甚至不能工作。

可以利用enumerate()：

count = 0

for index, line in enumerate(open(filepath,'r'))：

count += 1

31.文件读取

关于read()方法：

1、读取整个文件，将文件内容放到一个字符串变量中

2、如果文件大于可用内存，不可能使用这种处理

关于readline()方法：

1、readline()每次读取一行，比readlines()慢得多

2、readline()返回的是一个字符串对象，保存当前行的内容

关于readlines()方法：

1、一次性读取整个文件。

2、自动将文件内容分析成一个行的列表。

32.矩阵比较numpy数组筛选

两个列矩阵比较是对应的行进行比较，返回的是一个和两个矩阵行数相同的为True（两个列矩阵对应行上的元素相同则返回的矩阵此行为True）或False的列矩阵。注意全部为True或False的列矩阵可以进一步传到相同行的列矩阵中进行选择，例如True行处赋值给一个值。

33.Counter用法，根据字典的值的大小快速进行排序

1）d={**'a'**:1,**'b'**:3}  
dd=sorted(d.iteritems(),key=operator.itemgetter(1),reverse=True)

返回的dd是排序好的列表[('b', 3), ('a', 1)]

Print dd[0][0] 输出b

2）**from** collections **import** Counter

d={**'a'**:1,**'b'**:3}

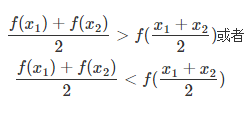
counts=Counter(d)  
**print** counts.most\_common(2) #2是值最大的前两个

输出的也是排序好的列表[('b', 3), ('a', 1)]

34.凸函数，梯度，梯度下降，拉格朗日

具体见博客：

有拉格朗日乘子法的地方，必然是一个组合优化问题。同时拉格朗日法是一定适合于凸问题的，不一定适合于其他问题。什么是凸问题？凸的就是开口朝一个方向（向上或向下）。更准确的数学关系就是：



如果满足第一个就是开口向上的凸，第二个是开口向下的凸

<http://blog.csdn.net/on2way/article/details/47729419>

<https://www.cnblogs.com/pinard/p/5970503.html>

L1,L2范数的介绍

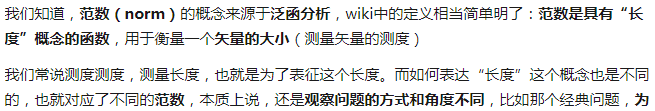
<http://blog.csdn.net/zouxy09/article/details/24971995>

几种常见范数的简单介绍

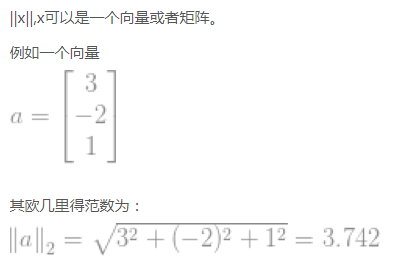
<http://blog.csdn.net/shijing_0214/article/details/51757564>

https://www.zhihu.com/question/20924039

范数的理解：



L0范数从严格意义来说不是一个范数，它表示的为向量中非0元素的个数。

L2范数：

即向量a的模的大小。上面的例子展示了怎样计算欧几里得范数，或者叫做L2范数。

最常用的是求其平方/n，称为均方误差

L2-norm(范数)也称为最小均方(least squares)，它是最小化目标值yi和估计值f(xi)平方和

L1范数：计算的是向量中所有元素的绝对值之和。

L1-norm(范数)，也叫作最小绝对偏差（leastabsolute deviations, LAD），最小绝对误差（least absolute errors，LAE）.它是最小化目标值yi和估计值f(xi)绝对差值的求和：

对于L1和L2范数而言：L2对异常点更加敏感。

L1范数：表示向量中每个元素绝对值的和

L2范数：即向量元素的平方和再开方，L2范数即欧氏距离

鲁棒性（Robustness）：最小绝对值偏差的方法应用领域很广，相比最小均方的方法，它的鲁棒性更好，LAD能对数据中的异常点有很好的抗干扰能力，异常点可以安全的和高效的忽略，这对研究帮助很大。如果异常值对研究很重要，最小均方误差则是更好的选择。

对于L2-norm，由于是均方误差，如果误差>1的话，那么平方后，相比L-norm而言，误差就会被放大很多。因此模型会对样例更敏感。如果样例是一个异常值，模型会调整最小化异常值的情况，以牺牲其它更一般样例为代价，因为相比单个异常样例，那些一般的样例会得到更小的损失误差。

以0范数和1范数作为正则项，可以求得稀疏解，以2范数作为正则项可以得到稠密解

**正则化解释**

应该把其称为规则化更好理解

首先需要明白为什么要正则化：**正则化是为了防止过拟合**

**规则化就是说给需要训练的误差函数加上一些规则（限制），让他们不要自我膨胀，从而防止过拟合**

**正则化的思想就是在损失函数中加入刻画模型复杂程度的指标。无论是哪一种正则化方式其基本的思想都是希望通过限制权重的大小，使得模型不能任意拟合训练数据中的随机噪声。**

**所谓的正则化就是在计算损失函数的过程中，在损失函数的后面再加一项，这样损失值所代表的输出与标准结果间的误差就会受到干扰，导致参数无法按照目标方向来调整，实现模型无法与样本完全拟合的结果，从而达到防止过拟合的效果。注意：添加的干扰项一定要具备如下两个特征，当欠拟合时，希望它对模型误差的影响越小越好，以便让模型快速拟合实际。当过拟合时，希望它对模型误差的影响越大越好，以便让模型不要产生过拟合的现象。**

**L1,L2正则化就是对损失函数加上例如L1,L2范数的规则。**

35.reshape()和astype()的用法

它俩基本都是针对numpy的array()类型使用的

若x是可重塑的对象（numpy的array类型），则x.reshape((-1,1))会把x重塑为1列以及自适应的行

通过reshape生成的新数组和原始数组公用一个内存，也就是说，假如更改一个数组的元素，另一个数组也将发生改变。

注意：Python的内置函数range()产生列表之后并没有reshape属性，Numpy的arange()产生的列表有reshape属性

36.对于方差和偏差的理解

参考：<https://www.zhihu.com/question/27068705>

为了取得好的泛化性能，则需使偏差较小，既能够充分拟合数据，并且使方差较小，即使得数据扰动产生的影响小

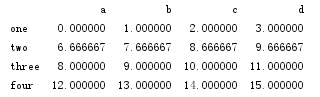
首先 Error = Bias + Variance  
Error反映的是整个模型的准确度，Bias反映的是模型在样本上的输出与真实值之间的误差，即模型本身的精准度，Variance反映的是模型每一次输出结果与模型输出期望之间的误差，即模型的稳定性。

37.Pandas中DataFrame的sum()函数

若一个DataFrame中的数据都是数字，用df表示此，则df.sum()返回一个含有列小计的Series,df.sum(axis=1)则会按行进行求和运算。

38.Pandas中填充DataFrame的nan值

df=DataFrame(np.arange(16).reshape((4,4)),columns=[**'a'**,**'b'**,**'c'**,**'d'**],index=[**'one'**,**'two'**,**'three'**,**'four'**])  
df.ix[1]=np.nan  
**print** df  
**print** df.fillna(df.mean())

填充后的输出为：

df.mean()是把nan值用每一列的均值填充，例如这列填充two行使用one,three,four行的值之和除以3.

例2:

train\_data['Embarked'] = train\_data['Embarked'].fillna('S')

意思是把Embarked这一列的所有缺失值填充为S

39.Pandas中对字符数据进行标记编码

sexEncoder=preprocessing.LabelEncoder()

train[**'Sex'**]=sexEncoder.fit\_transform(train[**'Sex'**])

注意：对于不同的字符列在对其进行标准化处理时要运用不同的preprocessing.LabelEncoder()

train\_data.loc[train\_data['Sex']=='male','Sex'] = 0 train\_data.loc[train\_data['Sex']=='female','Sex'] = 1

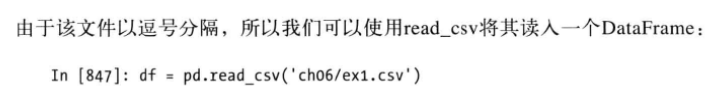
train\_data['Sex']=='male'是选出所有Sex列为male的行，选出这些行之后再把这些行的Sex列赋值为0

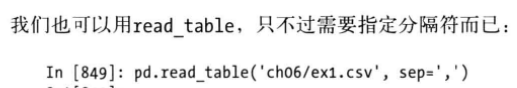
第二种方法：

target\_num\_map = {**'male'**:0, **'female'**:1}  
train\_data[**"Sex"**]=train\_data[**"Sex"**].apply(**lambda** x: target\_num\_map[x])

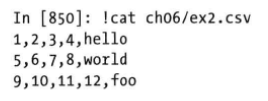
40.文件的读入

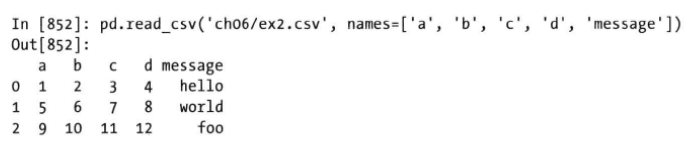
csv文件的读入





读入文件时为没有标题的文件指定列名





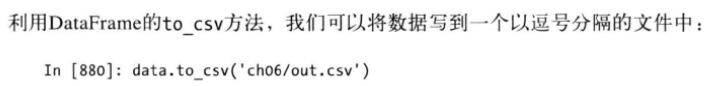
读取文件时跳过某些行

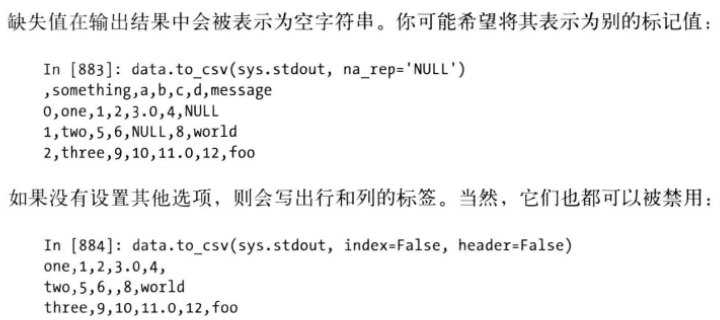




41.文件的输出

把DataFrame中的值输出到csv文件中





Index=False指定输出文件没有行索引，header=False指定输出文件没有列索引

42.Sigmoid()实现二分类说明及用法

注意利用Sigmoid()实现二分类并非是输出自动为0,1一样的输出需要自己进行转化。

例如：c=np.mat([0.4,0.6,0.1,0.8])

**def** sigmoid(inX):  
 pre=1.0/(1+np.exp(-inX))  
 pre=np.array(pre)  
 returnArr=np.zeros(np.shape(pre))  
 returnArr[:,np.nonzero(pre>0.5)[1]]=1  
 **return** returnArr

# 43.Pandas中loc iloc ix的用法以及区别

1. loc()按照行索引选取行

例如：

data = [[1,2,3],[4,5,6]]  
index = [**'e'**,**'f'**]  
columns=[**'a'**,**'b'**,**'c'**]  
df = pd.DataFrame(data=data, index=index, columns=columns)  
print(df.loc[**'e'**])

输出行索引为e的这一行的值

若为：

data = [[1,2,3],[4,5,6]]  
index = [1,2]  
columns=[**'a'**,**'b'**,**'c'**]  
df = pd.DataFrame(data=data, index=index, columns=columns)  
print(df.loc[1])

这里的行索引只能用1,2不能再用e,f

获取多行索引以上两种情况用如下方法：

df.loc[1:]

df.loc[‘e’:]

获取某些行某些列

df.loc[1,[**'b'**,**'a'**]]

1. iloc()通过行号以及列号的索引获取数据

例如：

data = [[1,2,3],[4,5,6]]  
index = [**'e'**,**'f'**]  
columns=[**'a'**,**'b'**,**'c'**]  
df = pd.DataFrame(data=data, index=index, columns=columns)

df.iloc[1]获取行号为1的数据  
print(df.iloc[1,[1]])获取行列号都为1的数据。

print(df.iloc[:,[1]])获取所有行列号为1的数据。

注意：这个函数只能通过行列号索引获取数据

1. ix()是以上两者的结合，获取行列时既可以通过行列的索引也可以通过行列的标签。

df.ix[:,df.ix[0]>2.5]

先选取第0行的值大于2.5的列，再选取所有行这些列的元素

44.pandas的drop()实现删除功能

例如：

data = [[1,2,3],[4,5,6]]  
index = [**'e'**,**'f'**]  
columns=[**'a'**,**'b'**,**'c'**]  
df = pd.DataFrame(data=data, index=index, columns=columns)

df=df.drop(**'e'**) 删除行索引为e的数据

df=df.drop([**'b'**,**'c'**],axis=1) 删除列索引为b,c的数据

注意：经过drop()处理后会返回一个新对象，若想替代原来的对象可以设置inplace=**True**

45.Pandas中基于DataFrame的apply()

Apply()即为在DataFrame的行、列以及整个DataFrame上应用某一函数。具体见http://blog.csdn.net/y12345678904/article/details/72757912

*# Define function to extract titles from passenger names*

def get\_title(name):

title\_search = re.search(' ([A-Za-z]+)\.', name)

*# If the title exists, extract and return it.*

if title\_search:

return title\_search.group(1)

return ""*# Create a new feature Title, containing the titles of passenger names*for dataset **in** full\_data:

dataset['Title'] = dataset['Name'].apply(get\_title)

例如对pandas的某一文本列进行处理

**def** word\_cut(df):  
 **return ' '**.join([con **for** con **in** jieba.cut(**''**.join(list(df[**'content'**]))) **if not** con **in** stopwords])  
df[**'content'**]=df.apply(word\_cut,axis=1)

给其传的是整个df，其主要目的是对df[**'content'**]中的内容进行分词处理，因为jieba.cut()接收的数据是字符串，所以要做相应的转换。

46.列表[]和元组()的区别与联系，以及字符串

他俩都是处理有序项目的数据结构，即为可以通过脚标取对应的值，同时他俩都可以存储不同的数据类型的值。它俩的区别在于，列表是可变的，即为可以通过a[1]=2这样进行改变，而元组是不可变的。因此元组相对而言更为安全。

字符串也是不可变的

47.列表切片

print(num2[1:5:2]) #取 索引为1 到 4的值，步长为2

# print(num2[::2]) #表示取所有的值，步长为2  
print(num2[::-1])#切片步长为负数，从后面往前面取值，相当于翻转了

48.Pandas读取数据之后如何根据标签列快速判断数据是否均衡？

print(train\_data[**'Survived'**].value\_counts())

输出该列每个值出现的次数。

# 49.numpy中array和asarray的区别

array和asarray都可以将结构数据转化为ndarray，但是主要区别就是当数据源是ndarray时，array仍然会copy出一个副本，占用新的内存，但asarray不会。

50.Python中zip()的用法

zip()函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。

如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 \* 号操作符，可以将元组解压为列表。

a = [1,2,3]

b = [4,5,6]

c = [4,5,6,7,8]

zipped = zip(a,b) # 打包为元组的列表

[(1, 4), (2, 5), (3, 6)]

zip(a,c) # 元素个数与最短的列表一致

[(1, 4), (2, 5), (3, 6)]

zip(\*zipped) # 与 zip 相反，可理解为解压，返回二维矩阵式

[(1, 2, 3), (4, 5, 6)]

51.Python中splitlines()的用法

Python splitlines() 按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。

str1 = 'ab c\n\nde fg\rkl\r\n'print str1.splitlines();

str2 = 'ab c\n\nde fg\rkl\r\n'print str2.splitlines(True)

以上实例输出结果如下：

['ab c', '', 'de fg', 'kl']

['ab c\n', '\n', 'de fg\r', 'kl\r\n']

52.列表向字符串的转换

list=['www', 'google', 'com']

str = " ".join(list)

print str

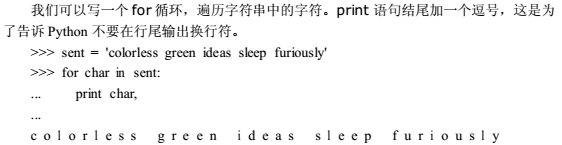
www google com

常用在sklearn的主题模型LDA中获得英文文章后要进行词型还原，会被转换为列表，随后又要进行词频统计，而其接收的参数各个文档组成一个list，而每个文章需要是一个str类型而不是List，因此需要进行转换。

53.字符串的isalpha()用法

判断字符串是否是由连续的字母组成不含空格以及非字母成分

54.print语句后加一个,号的意义



55.python的=(赋值操作)，[:]切片操作以及copy模块的copy()与deepcopy（）

例如a=[1,2,3]

b=a意思是相当于a指向了列表[1,2,3]这个地址，a存储的是指向列表的地址（指针）,b=a则是使得b也获得了列表[1,2,3]这个地址，b也获得了指向列表的地址（指针）。

a=[4,5,6]这时把列表[4,5,6]的地址赋给了a，a存储的不再是列表[1,2,3]的地址

也就是说=是把地址的引用赋值给了新变量，并不是再开辟新空间。

[:]切片后的结果开辟了新空间，不和原来指向一处

Copy.copy()是浅层赋值，只会为浅层次的开辟新空间，深层的则会指向原来的。

例如：bar=[1,2],foo=copy.copy(),使得bar[0]=3,则foo的值仍为[1,2]

若bar=[1,[2,3]],foo=copy.copy(),使得bar[1][0]=4,则foo的值也会变为[1,[4,3]]这就是所谓浅层复制，若使用copy.deepcopy()bar中的值无论怎么改变将不再会影响到foo的值

A=[‘a’]\*3 A=[‘a’,’a’,’a’]这样构造出的A中的三个值是同一对象，三个’a’是存储在内存中同一地址中的’a’，改变一个其余两个同时改变。

1. 假如dict为一个字典，则list(dict)返回一个包含字典中所有键的列表
2. .csv文件与.tsv文件的区别

.csv文件是以,作为分隔符的，而.tsv是以\t作为分隔符的

故读取.tsv文件为pd.read\_csv(**'labeledTrainData.tsv'**,sep=**'\t'**,header=0)

1. A.shape[-1]相当于切片的倒序，这里是输出最后一维，二维的则输出的是列数
2. Pandas处理时间序列数据的函数date\_range

****函数原型****

pandas.****date\_range****(start=None, end=None, periods=None, freq=’D’, tz=None, normalize=False, name=None, closed=None, \*\*kwargs)

****参数****

****start****：string或datetime-like，默认值是None，表示日期的起点。

****end****：string或datetime-like，默认值是None，表示日期的终点。

****periods****：integer或None，默认值是None，表示你要从这个函数产生多少个日期索引值；如果是None的话，那么start和end必须不能为None。

****freq****：string或DateOffset，默认值是’D’，表示以自然日为单位，这个参数用来指定计时单位，比如’5H’表示每隔5个小时计算一次。

****tz****：string或None，表示时区，例如：’Asia/Hong\_Kong’。

****normalize****：bool，默认值为False，如果为True的话，那么在产生时间索引值之前会先把start和end都转化为当日的午夜0点。

****name****：str，默认值为None，给返回的时间索引指定一个名字。

****closed****：string或者None，默认值为None，表示start和end这个区间端点是否包含在区间内，可以有三个值，’left’表示左闭右开区间，’right’表示左开右闭区间，None表示两边都是闭区间。

****返回值****

DatetimeIndex

例子：

dates = pd.date\_range(start\_date, end\_date, freq=**'M'**)  
*# Convert the data into time series data*data\_timeseries = pd.Series(data[:,column], index=dates)

一般都是用date\_range（）产生的时间序列数据作为索引

1. A=[1,2,3],则np.shape(A)输出为（3,）代表它是一维的，没有行列可言，3只是代表有三个元素。

B=np.zeros((3))返回的是一维的ndarray的数组，它的B.shape也为（3,）只是表示有三个元素,取值相当于一维数组的用脚标取值。

1. Numpy的diff()函数

numpy.diff(a, n=1,axis=-1)

沿着指定轴计算第N维的离散差值   
参数：   
a：输入矩阵   
n：可选，代表要执行几次差值   
axis：默认是最后一个

例如：

A = np.arange(2 , 14).reshape((3 , 4))

print('A:' , A)

# A: [[ 2 3 4 5]

[ 6 8 8 9]

[10 11 12 13]]

print(np.diff(A))

[[1 1 1]

[2 0 1]

[1 1 1]]

从输出结果可以看出，其实diff函数就是执行的是后一个元素减去前一个元素。

1. Numpy的矩阵合并

np.concatenate((a,b),axis=1)第一维精确到了一个个元素故为往一维数组中追加元素。表现为把两矩阵按列进行合并,

axis=0是按照第0维，第0维是一个个一维数组，故为在行方面追加

np.column\_stack()按列进行合并，参数接收一个元组

np.row\_stack()按行进行合并，参数接收一个元组

1. Collections中的defaultdict以及字典直接取键对应的值和get键对应值的区别

Defaultdict（）创建一个拥有默认值的字典，例如：

a=defaultdict(int) b=defaultdict(float) c=defaultdict(list)

d=defaultdict(dict)

a是创建默认取值为0的字典，b是创建默认取值为0.0的字典,c是创建默认取值为列表的字典。

注意上面说的个取值不是得到键的值

a[‘ee’]若字典a中不存在为ee的键返回的为0

a.get(‘ee’)若字典a中不存在为ee的键返回的为None,但可以为其加默认值操作为a.get(‘ee’,0)

1. pandas的groupby和agg以及reset\_index和sort\_values函数

例如：

**import** pandas **as** pd  
data=[  
 [1,2,3,4],[1,4,7,8],[2,8,9,4]  
]  
df=pd.DataFrame(data,columns=[**'a'**,**'b'**,**'c'**,**'d'**])  
df2=df[[**'a'**,**'b'**,**'c'**]]  
print(df2)  
print(df2.groupby(**'a'**)[**'b'**].agg(**'sum'**).reset\_index().sort\_values(**'a'**))

groupby是按照某些列进行分组把分组列相同的归为一组，注意它返回的不再是DataFrame类型的数据

agg函数是应用例如sum,mean,std这些聚合函数

reset\_index可以还原索引，不管原来的索引是什么，从新变为默认的整型索引，它一般和agg()连用

sort\_values（）按值进行排序，其中有个axis参数默认是0即为按列进行排序，若axis参数的值设为1则是按行进行排序。

上面的案例是按照a列进行分组取分组后的a,b列，对每个分组执行求和操作，然后重置索引，最后再按照a列的值进行排序操作

1. Pandas的head()和tail

head()默认是取前五行元素,head(n)取前n行

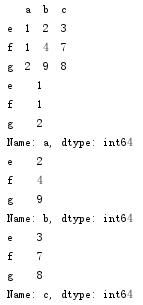
tail()默认是取后五行元素,tail(n)取后n行

1. Pandas的iterrows()

实现对DataFrame()进行行编列

print(df2)

**for** i,j **in** df2.iteritems():  
 print(j)



是Serise序列类型的

1. Pandas的DataFrame插入列

用的是insert()函数它有三个必须的参数，第一个loc指定插入列的列号，第二个column指定插入后列的名称，第三个value是插入的值

注意：df.insert(3,column=**'e'**,value=df2[**'d'**])

例如如上代码在df中插入df2的d列，若df2中d列的行索引在df中不存在则值为NAN只有插入的df2中的d列其对应的行索引在df中也有相同的行索引才有值

1. List的追加

list1=[**'a'**,**'b'**]  
list1+=[**'c'**]  
print(list1)

[**'a'**,**'b'**,**'c'**]直接把两个进行了合并

1. **[numpy的numpy.delete()/insert()/append()函数](https://blog.csdn.net/weixin_37895339/article/details/78442555)**
2. numpy.delete(arr,obj,axis=None)   
   arr:输入向量   
   obj:表明哪一个子向量应该被移除。可以为整数或一个int型的向量   
   axis:表明删除哪个轴的子向量，若默认，则返回一个被拉平的向量

a = np.array(np.arange(12).reshape(3,4))

a Out[301]: array([[ 0, 1, 2, 3], [ 4, 5, 6, 7], [ 8, 9, 10, 11]]) np.delete(a,1,0)

Out[302]: array([[ 0, 1, 2, 3], [ 8, 9, 10, 11]])

np.delete(a,1,1)

Out[303]: array([[ 0, 2, 3], [ 4, 6, 7], [ 8, 10, 11]])

np.delete(a,[0,1],1)

Out[304]: array([[ 2, 3], [ 6, 7], [10, 11]])

np.delete(a,np.s\_[::2],1)

Out[306]: array([[ 1, 3], [ 5, 7], [ 9, 11]])

1. numpy.insert(arr,obj,value,axis=None)   
   同理，value为插入的数值   
   arr:为目标向量   
   obj:为目标位置   
   value:为想要插入的数值   
   axis:为插入的维度

np.insert(a,1,[1,1,1,1],0)

Out[309]: array([[ 0, 1, 2, 3], [ 1, 1, 1, 1], [ 4, 5, 6, 7], [ 8, 9, 10, 11]])

1. numpu.append(arr,values,axis=None)   
   将values插入到目标arr的最后。   
   注意，这里values跟arr应该为相同维度的向量，例如，如下第一个为错误情况。

正确形式为

a Out[312]:

array([[ 0, 1, 2, 3], [ 4, 5, 6, 7], [ 8, 9, 10, 11]]) np.append(a,[[1,1,1,1]],axis=0) 注意这里一定要是[[1,1,1,1]]保持维度与a相同

Out[313]:

array([[ 0, 1, 2, 3], [ 4, 5, 6, 7], [ 8, 9, 10, 11], [ 1, 1, 1, 1]])

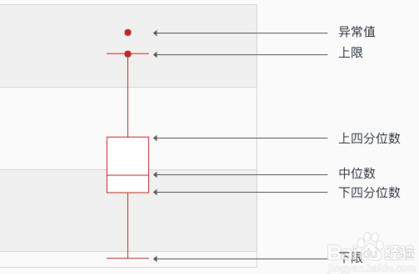
1. One\_hot独热编码转换为普通编码

Label=[np.argmax(row) for row in one\_hot\_data]

1. 箱型图分析

用seaborn绘制箱型图和直方图，具体参考：

<http://www.cnblogs.com/kylinlin/p/5236601.html>



想要学会箱形图的分析，一定要搞懂以下几个指标：

1、下四分位数Q1

（1）确定四分位数的位置。Qi所在位置=i（n+1）/4，其中i=1，2，3。n表示序列中包含的项数。

（2）根据位置，计算相应的四分位数。

例中：

Q1所在的位置=（14+1）/4=3.75，

Q1=0.25×第三项+0.75×第四项=0.25×17+0.75×19=18.5；

2、中位数（第二个四分位数）Q2

中位数，即一组数由小到大排列处于中间位置的数。若序列数为偶数个，该组的中位数为中间两个数的平均数。

例中：

Q2所在的位置=2（14+1）/4=7.5，

Q2=0.5×第七项+0.5×第八项=0.5×25+0.5×28=26.5

3、上四分位数Q3

计算方法同下四分位数。

例中：

Q3所在的位置=3（14+1）/4=11.25，

Q3=0.75×第十一项+0.25×第十二项=0.75×34+0.25×35=34.25。

4、上限

上限是非异常范围内的最大值。

首先要知道什么是四分位距如何计算的？

四分位距IQR=Q3-Q1，那么上限=Q3+1.5IQR

5、下限

下限是非异常范围内的最小值。

下限=Q1-1.5IQR

箱形图的价值

1.直观明了地识别数据批中的异常值

上文讲了很久的识别异常值，其实箱线图判断异常值的标准以四分位数和四分位距为基础，四分位数具有一定的耐抗性，多达25%的数据可以变得任意远而不会很大地扰动四分位数，所以异常值不会影响箱形图的数据形状，箱线图识别异常值的结果比较客观。由此可见，箱线图在识别异常值方面有一定的优越性。

2.利用箱线图判断数据批的偏态和尾重

对于标准正态分布的样本，只有极少值为异常值。异常值越多说明尾部越重，自由度越小（即自由变动的量的个数）；

而偏态表示偏离程度，异常值集中在较小值一侧，则分布呈左偏态；异常值集中在较大值一侧，则分布呈右偏态。

3.利用箱线图比较几批数据的形状

同一数轴上，几批数据的箱线图并行排列，几批数据的中位数、尾长、异常值、分布区间等形状信息便昭然若揭。如上图，可直观得看出第三季度各分公司的销售额大体都在下降。

1. corr()函数用于预测相关性系数

该函数是Pandas其DataFrame的内置函数，默认的用皮尔逊相关系数预测不同列的相关性，返回的也是一个DataFrame

1. 多个DataFrame用concat（）进行合并

df=pd.concat([df1,df2[[**'aa2'**,**'bb2'**]]],axis=1)

注意前面一定要用[]括起来，同时合并的两个DataFrame一定要行列数相同且每行的索引相同才有意义。

1. splitlines()

Python splitlines() 按照行('\r', '\r\n', \n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。

1. 字符串格式化输出format()用法

Python中格式化输出字符串使用format()函数, 字符串即类, 可以使用方法;

Python是完全面向对象的语言, 任何东西都是对象;

字符串的参数使用{NUM}进行表示, 0表示第一个参数, 1表示第二个参数, 以后顺次递加;

使用":", 指定代表元素需要的操作, 如":.3"小数点三位, ":8"占8个字符空间等;

数字(0, 1, ...)即代表format()里面的元素, 所以可以使用"."调用元素的方法;

#保留两位小数 print '{:.2f}'.format(321.33345)

#通过位置

print '{0},{1}'.format('chuhao',20)

print '{},{}'.format('chuhao',20)

print '{1},{0},{1}'.format('chuhao',20)

1. Pandas的read\_csv()常用的参数

Head=none就是没有列名，name=[‘a’,’b’]就是指定列名为a,b

1. 对于时间特征的处理

具体参考shijianxuliechuli中的time\_demo.py

1. 随机打乱训练集

np.random.seed(10)  
a=np.random.permutation(10)

为随机数的产生设置一个种子，这样每次产生的随机数就相同了，np.random.permutation(10)把0-9的整数进行随机的排序。