数据分析

什么是数据分析？

数据分析是指用适当的统计分析方法对收集来的大量数据进行分析，提取有用信息和形成结论而对数据加以详细研究和概括总结的过程。

数据分析经典案例

（一）啤酒与尿布

沃尔玛在对消费者购物行为分析时发现，男性顾客在购买婴儿尿片时，常常会顺便搭配几瓶啤酒来犒劳自己，于是尝试推出了将啤酒和尿布摆在一起的促销手段。没想到这个举措居然使尿布和啤酒的销量都大幅增加了。

（二）数据新闻让英国撤军

2010年10月23日《卫报》利用维基解密的数据做了一篇“数据新闻”。将伊拉克战争中所有的人员伤亡情况均标注于地图之上。地图上一个红点便代表一次死伤事件，鼠标点击红点后弹出的窗口则有详细的说明：伤亡人数、时间，造成伤亡的具体原因。密布的红点多达39万，显得格外触目惊心。一经刊出立即引起朝野震动，推动英国最终做出撤出驻伊拉克军队的决定。

（三）微软数据分析成功预测奥斯卡21项大奖

2013年，微软纽约研究院的经济学家大卫•罗斯柴尔德（David Rothschild）利用数据分析技术成功预测24个奥斯卡奖项中的19个，成为人们津津乐道的话题。后来，罗斯柴尔德再接再厉，成功预测第86届奥斯卡金像奖颁奖典礼24个奖项中的21个。

	ID	套餐金额	额外通话时长	额外流量	改变行为	服务合约	关联购买	集团用户	使用月数	流失用户
0	1	1	792.833333	-10.450067	0	0	0	0	25	0
1	2	1	121.666667	-21.141117	0	0	0	0	25	0
2	3	1	-30.000000	-25.655273	0	0	0	0	2	1
3	4	1	241.500000	-288.341254	0	1	0	1	25	0
4	5	1	1629.666667	-23.655505	0	0	0	1	25	0
5	6	1	182.000000	-115.863584	0	1	0	1	25	0
6	7	1	196.333333	221.294207	0	1	0	1	23	0
7	8	1	539.500000	81.160600	0	1	0	0	25	0
8	9	1	1037.166667	8.344393	0	1	0	1	25	0
9	10	2	289.000000	-131.784518	0	0	0	1	25	0

	index	title	community	years	housetype	square	floor	taxtype	totalPrice	unitPrice	followInfo
0	0	宝星华庭一层带花园，客厅挑高，通透四居室。房主自荐	宝星国际三期	底层(共22层)2010年建板塔结合	4室1厅	298.79平米	底层(共22层)2010年建板塔结合	距离15号线望京东站680米房本满五年	2598	86951	53人关注 / 共44次带看 / 一年前发布
1	1	三面采光全明南北朝向 正对小区绿地花园	顶秀青溪	中楼层(共11层)2008年建板塔结合	3室2厅	154.62平米	中楼层(共11层)2008年建板塔结合	距离5号线立水桥站1170米房本满两年随时看房	1000	64675	323人关注 / 共579次带看 / 一年前发布
2	2	沁园公寓 三居室 距离苏州街地铁站383米	沁园公寓	低楼层(共24层)1999年建塔楼	3室2厅	177.36平米	低楼层(共24层)1999年建塔楼	距离10号线苏州街站383米房本满五年	1200	67659	185人关注 / 共108次带看 / 一年前发布

	UserID	Gender	Age	Occupation	Zip-code	MovieID	Rating	Timestamp	Title	Genres
0	1	F	1	10	48067	1193	5	978300760	One Flew Over the Cuckoo's Nest (1975)	Drama
1	2	M	56	16	70072	1193	5	978298413	One Flew Over the Cuckoo's Nest (1975)	Drama
2	12	M	25	12	32793	1193	4	978220179	One Flew Over the Cuckoo's Nest (1975)	Drama
3	15	M	25	7	22903	1193	4	978199279	One Flew Over the Cuckoo's Nest (1975)	Drama
4	17	M	50	1	95350	1193	5	978158471	One Flew Over the Cuckoo's Nest (1975)	Drama
5	18	F	18	3	95825	1193	4	978156168	One Flew Over the Cuckoo's Nest (1975)	Drama
6	19	M	1	10	48073	1193	5	982730936	One Flew Over the Cuckoo's Nest (1975)	Drama
7	24	F	25	7	10023	1193	5	978136709	One Flew Over the Cuckoo's Nest (1975)	Drama
8	28	F	25	1	14607	1193	3	978125194	One Flew Over the Cuckoo's Nest (1975)	Drama
9	33	M	45	3	55421	1193	5	978557765	One Flew Over the Cuckoo's Nest (1975)	Drama

数据分析三驾马车

1. 统计学
2. 业务
3. 算法与编程

通过三种技能贯穿数据分析思想，培养自己的业务需求分析能力与编程能力，解决具体行业场景的数据分析问题。

课程设计

总结后端知识体系，了解数据分析、人工智能与后端的关系。

课程体系设计。学习的重点。学习方法。

徐铭 xuming@tedu.cn 15201603213 251041263

使用**python**做数据分析的常用库

1. numpy 基础数值算法
2. scipy 科学计算
3. matplotlib 数据可视化
4. pandas 序列高级函数

numpy

numpy概述

1. Numerical Python，数值的Python，补充了Python语言所欠缺的数值计算能力。
2. Numpy是其它数据分析及机器学习库的底层库。
3. Numpy完全标准C语言实现，运行效率充分优化。
4. Numpy开源免费。

numpy历史

1. 1995年，Numeric，Python语言数值计算扩充。
2. 2001年，Scipy->Numarray，多维数组运算。
3. 2005年，Numeric+Numarray->Numpy。
4. 2006年，Numpy脱离Scipy成为独立的项目。

numpy的核心：多维数组

1. 代码简洁：减少Python代码中的循环。
2. 底层实现：厚内核(C)+薄接口(Python)，保证性能。

numpy基础

ndarray数组

用np.ndarray类的对象表示n维数组

```
import numpy as np
ary = np.array([1, 2, 3, 4, 5, 6])
print(type(ary))
```

内存中的ndarray对象

元数据（metadata）

存储对目标数组的描述信息，如：ndim、dimensions、dtype、data等。

实际数据

完整的数组数据

将实际数据与元数据分开存放，一方面提高了内存空间的使用效率，另一方面减少对实际数据的访问频率，提高性能。

ndarray数组对象的特点

1. Numpy数组是同质数组，即所有元素的数据类型必须相同
2. Numpy数组的下标从0开始，最后一个元素的下标为数组长度减1

ndarray数组对象的创建

np.array(任何可被解释为Numpy数组的逻辑结构)

```
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6])
print(a)
```

np.arange(起始值(0),终止值,步长(1))

```
import numpy as np
a = np.arange(0, 5, 1)
print(a)
b = np.arange(0, 10, 2)
print(b)
```

np.zeros(数组元素个数, dtype='类型')

```
import numpy as np
a = np.zeros(10)
print(a)
```

np.ones(数组元素个数, dtype='类型')

```
import numpy as np
a = np.ones(10)
print(a)
```

ndarray对象属性的基本操作

数组的维度：np.ndarray.shape

```
import numpy as np
ary = np.array([1, 2, 3, 4, 5, 6])
print(type(ary), ary, ary.shape)
#二维数组
ary = np.array([
    [1,2,3,4],
    [5,6,7,8]
])
print(type(ary), ary, ary.shape)
```

元素的类型：np.ndarray.dtype

```
import numpy as np
ary = np.array([1, 2, 3, 4, 5, 6])
print(type(ary), ary, ary.dtype)
#转换ary元素的类型
b = ary.astype(float)
print(type(b), b, b.dtype)
#转换ary元素的类型
c = ary.astype(str)
print(type(c), c, c.dtype)
```

数组元素的个数：np.ndarray.size

```
import numpy as np
ary = np.array([
    [1,2,3,4],
    [5,6,7,8]
])
#观察维度，size，len的区别
print(ary.shape, ary.size, len(ary))
```

数组元素索引(下标)

数组对象[..., 页号, 行号, 列号]

下标从0开始，到数组len-1结束。

```
import numpy as np
a = np.array([[[1, 2],
               [3, 4]],
              [[5, 6],
               [7, 8]]])
print(a, a.shape)
print(a[0])
print(a[0][0])
print(a[0][0][0])
print(a[0, 0, 0])
for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        for k in range(a.shape[2]):
            print(a[i, j, k])
```

ndarray对象属性操作详解

Numpy的内部基本数据类型

类型名	类型表示符
布尔型	bool_
有符号整数型	int8(-128~127) / int16 / int32 / int64
无符号整数型	uint8(0~255) / uint16 / uint32 / uint64
浮点型	float16 / float32 / float64
复数型	complex64 / complex128
字串型	str_，每个字符用32位Unicode编码表示
日期类型	datetime64

自定义复合类型

```
# 自定义复合类型
import numpy as np

data=[
    ('zs', [90, 80, 85], 15),
    ('ls', [92, 81, 83], 16),
    ('ww', [95, 85, 95], 15)
]
#第一种设置dtype的方式
a = np.array(data, dtype='U3, 3int32, int32')
print(a)
print(a[0]['f0'], ":", a[1]['f1'])
print("====================================")
#第二种设置dtype的方式
b = np.array(data, dtype=[('name', 'str_', 2),
                          ('scores', 'int32', 3),
                          ('age', 'int32', 1)])
print(b[0]['name'], ":", b[0]['scores'])
```

```
print("====================================")

#第三种设置dtype的方式
c = np.array(data, dtype={'names': ['name', 'scores', 'ages'],
                    'formats': ['U3', '3int32', 'int32']})
print(c[0]['name'], ":", c[0]['scores'], ":", c.itemsize)
print("====================================")

#第四种设置dtype的方式
d = np.array(data, dtype={'name': ('U3', 0),
                    'scores': ('3int32', 16),
                    'age': ('int32', 28)})
print(d[0]['names'], d[0]['scores'], d.itemsize)

print("====================================")

#测试日期类型数组
f = np.array(['2011', '2012-01-01', '2013-01-01 01:01:01','2011-02-01'])
f = f.astype('M8[D]')
f = f.astype('i4')
print(f[3]-f[0])

f.astype('bool')
```

类型字符码

类型	字符码
np.bool_	?
np.int8/16/32/64	i1 / i2 / i4 / i8
np.uint8/16/32/64	u1 / u2 / u4 / u8
np.float/16/32/64	f2 / f4 / f8
np.complex64/128	c8 / c16
np.str_	U
np.datetime64	M8[Y] M8[M] M8[D] M8[h] M8[m] M8[s]

字节序前缀，用于多字节整数和字符串：
</>/[=]分别表示小端/大端/硬件字节序。

类型字符码格式

<字节序前缀><维度><类型><字节数或字符数>

3i4	释义
3i4	大端字节序，3个元素的一维数组，每个元素都是整型，每个整型元素占4个字节。
<(2,3)u8	小端字节序，6个元素2行3列的二维数组，每个元素都是无符号整型，每个无符号整型元素占8个字节。
U7	包含7个字符的Unicode字符串，每个字符占4个字节，采用默认字节序。

ndarray数组维度操作

视图变维（数据共享）： reshape() 与 ravel()

```
import numpy as np
a = np.arange(1, 9)
print(a)        # [1 2 3 4 5 6 7 8]
b = a.reshape(2, 4) #视图变维  : 变为2行4列的二维数组
print(b)
c = b.reshape(2, 2, 2) #视图变维     变为2页2行2列的三维数组
print(c)
d = c.ravel()   #视图变维    变为1维数组
print(d)
```

复制变维（数据独立）：flatten()

```
e = c.flatten()
print(e)
a += 10
print(a, e, sep='\n')
```

就地变维：直接改变原数组对象的维度，不返回新数组

```
a.shape = (2, 4)
print(a)
a.resize(2, 2, 2)
print(a)
```

ndarray数组索引操作

```
# 数组对象切片的参数设置与列表切面参数类似
#  步长+：默认切从首到尾
#  步长-：默认切从尾到首
数组对象[起始位置:终止位置:步长, ...]
# 默认位置步长：1
```

```
import numpy as np
a = np.arange(1, 10)
print(a)  # 1 2 3 4 5 6 7 8 9
print(a[:3])  # 1 2 3
print(a[3:6])   # 4 5 6
print(a[6:])  # 7 8 9
print(a[::-1])  # 9 8 7 6 5 4 3 2 1
print(a[:-4:-1])  # 9 8 7
print(a[-4:-7:-1])  # 6 5 4
print(a[-7::-1])  # 3 2 1
print(a[::])  # 1 2 3 4 5 6 7 8 9
print(a[:])  # 1 2 3 4 5 6 7 8 9
print(a[::3])  # 1 4 7
print(a[1::3])  # 2 5 8
print(a[2::3])  # 3 6 9
```

多维数组的切片操作

```
import numpy as np
a = np.arange(1, 28)
a.resize(3,3,3)
print(a)
#切出1页
print(a[1, :, :])
#切出所有页的1行
print(a[:, 1, :])
#切出0页的1行1列
print(a[0, :, 1])
```

ndarray数组的掩码操作

```
import numpy as np
a = np.arange(1, 10)
mask = [True, False,True, False,True, False,True, False,True]
print(a[mask])
```

多维数组的组合与拆分

垂直方向操作：

```
import numpy as np
a = np.arange(1, 7).reshape(2, 3)
b = np.arange(7, 13).reshape(2, 3)
# 垂直方向完成组合操作，生成新数组
c = np.vstack((a, b))
# 垂直方向完成拆分操作，生成两个数组
d, e = np.vsplit(c, 2)
```

水平方向操作：

```
import numpy as np
a = np.arange(1, 7).reshape(2, 3)
b = np.arange(7, 13).reshape(2, 3)
# 水平方向完成组合操作，生成新数组
c = np.hstack((a, b))
# 水平方向完成拆分操作，生成两个数组
d, e = np.hsplit(c, 2)
```

深度方向操作：（3维）

```
import numpy as np
a = np.arange(1, 7).reshape(2, 3)
b = np.arange(7, 13).reshape(2, 3)
# 深度方向（3维）完成组合操作，生成新数组
i = np.dstack((a, b))
# 深度方向（3维）完成拆分操作，生成两个数组
k, l = np.dsplit(i, 2)
```

长度不等的数组组合：

```
import numpy as np
a = np.array([1,2,3,4,5])
b = np.array([1,2,3,4])
# 填充b数组使其长度与a相同
b = np.pad(b, pad_width=(0, 1), mode='constant', constant_values=-1)
print(b)
# 垂直方向完成组合操作，生成新数组
c = np.vstack((a, b))
print(c)
```

多维数组组合与拆分的相关函数：

```
# 通过axis作为关键字参数指定组合的方向，取值如下：
# 若待组合的数组都是二维数组：
#   0：垂直方向组合
#   1：水平方向组合
# 若待组合的数组都是三维数组：
#   0：垂直方向组合
#   1：水平方向组合
#   2：深度方向组合
np.concatenate((a, b), axis=0)
# 通过给出的数组与要拆分的份数，按照某个方向进行拆分，axis的取值同上
np.split(c, 2, axis=0)
```

简单的一维数组组合方案

```
a = np.arange(1,9)      #[1, 2, 3, 4, 5, 6, 7, 8]
b = np.arange(9,17)     #[9,10,11,12,13,14,15,16]
#把两个数组摞在一起成两行
c = np.row_stack((a, b))
print(c)
#把两个数组组合在一起成两列
d = np.column_stack((a, b))
print(d)
```

ndarray类的其他属性

- shape - 维度
- dtype - 元素类型
- size - 元素数量
- ndim - 维数，len(shape)
- itemsize - 元素字节数
- nbytes - 总字节数 = size x itemsize
- real - 复数数组的实部数组
- imag - 复数数组的虚部数组
- T - 数组对象的转置视图
- flat - 扁平迭代器

```
import numpy as np
a = np.array([[1 + 1j, 2 + 4j, 3 + 7j],
              [4 + 2j, 5 + 5j, 6 + 8j],
              [7 + 3j, 8 + 6j, 9 + 9j]])
print(a.shape)
print(a.dtype)
print(a.ndim)
print(a.size)
print(a.itemsize)
```

```
print(a.nbytes)
print(a.real, a.imag, sep='\n')
print(a.T)
print([elem for elem in a.flat])
b = a.tolist()
print(b)
```