交通大数据

Python 高阶算法库介绍

- 郭延永
- guoyanyong@seu.edu.cn

Python高阶算法库

❑ 学习目标

- 了解Python模块安装与基本用法
- 认识Python标准库
- 学习Numpy基本用法
- 学习Pandas基本用法
- 了解Matplotlib、Scikit-Learn、TensorFlow

Python模块安装与基本用法

Python语言的最大优势之一：强大的模块功能

❑模块的使用方法

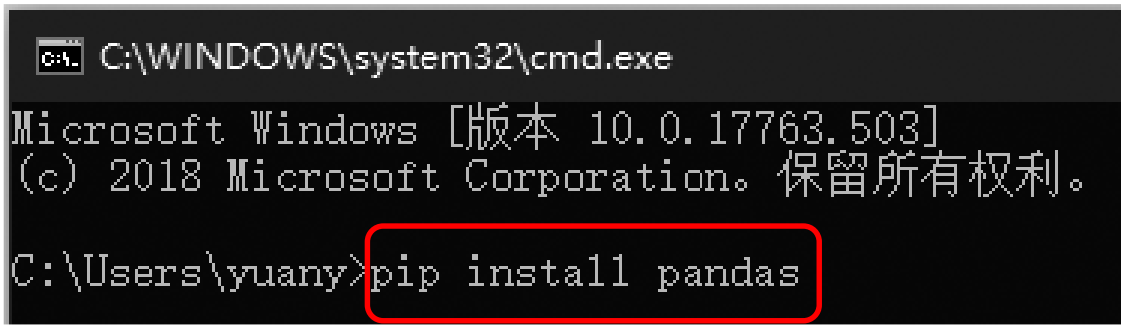
1.模块检索

PyPI（https://pypi.org/）是Python官方的模块发布网站，采用输入**关键词**搜索并获取**Python模块**

2.模块安装

（1）使用pip安装。

Windows系统下，在命令行中输入pip install 模块名即可，安装过程会自动进行，如图



Python模块安装与基本用法

Python语言的最大优势之一：强大的模块功能

❑模块的使用方法

2.模块安装

（1）使用pip安装。

检查Python及pip是否添加至环境变量

Anaconda环境下的Python路径：X:\Program Files\Anaconda3。

pip路径：X:\Program Files\Anaconda3\pkgs\pip-8.1.2-py35_0\Scripts

pip为Python3.5的版本

Python模块安装与基本用法

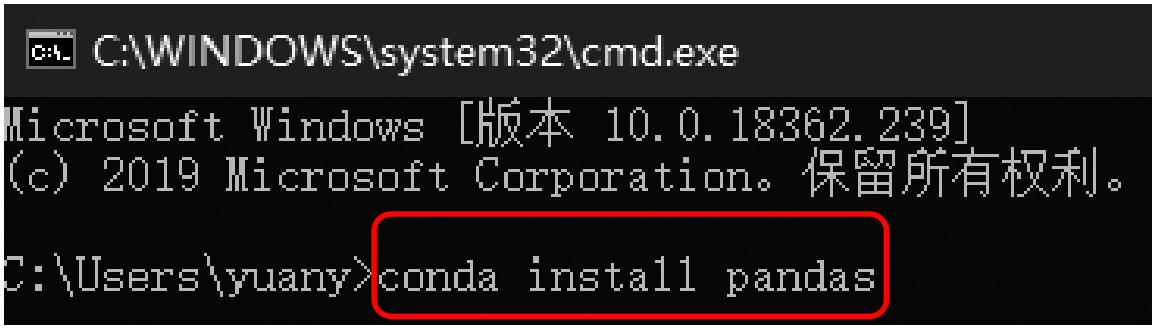
Python语言的最大优势之一：强大的模块功能

❑模块的使用方法

2.模块安装

（2）Anaconda环境下安装。

当安装了Anaconda环境时，在命令行中输入conda install 模块名即可，如图



Python模块安装与基本用法

Python语言的最大优势之一：强大的模块功能

❑模块的使用方法

2.模块安装

（3）手动安装whl文件。

某些特殊情况下，需要手动下载.whl文件进行安装。下载好.whl文件后，在存放文件的目录下，地址栏输入cmd，在弹出的命令行界面中输入“pip install”文件名即可手动安装。

Python模块安装与基本用法

Python语言的最大优势之一：强大的模块功能

❑模块的使用方法

3.模块使用

以导入pandas的文件读取函数为例，一种方法是导入pandas模块（设置别名为pd），使用别名pd，间接引用read_csv()函数读取文件名为′file.csv′的文件

```
1.  import pandas as pd
2.  pd.read_csv( 'file.csv' )
```

也可以用如下的方法导入read_csv()函数并读取文件。

```
1.  from pandas import read_csv
2.  read_csv( 'file.csv' )
```

其他模块使用方法大同小异，一般在官方文档中都能找到模块的详细调用方法

认识Python标准库

❑Python 标准库

1. math模块

（1）基本概念：math模块是Python中用于科学计算的内建模块，其中的函数都经过了特殊的算法优化，效率较高。

（2）常用函数

①常规数学运算函数

```
1.  import math          #调用
2.  math.ceil(x)         #向上取整
3.  math.floor(x)        #向下取整
4.  math.fabs(x)         #求绝对值
```

②幂函数与对数函数

```
1.  math.exp(x)          #返回e^x
2.  math.log(x, base)    #返回自然对数log(x),base为可选参数,若提供base,则返回log(x)/log(base)
3.  math.pow(x, y)       #返回x^y
4.  math.sqrt(x)         #返回x的平方根
```

认识Python标准库

❑Python 标准库

1. math模块

（1）基本概念：math模块是Python中用于科学计算的内建模块，其中的函数都经过了特殊的算法优化，效率较高。

（2）常用函数

①常规数学运算函数

②幂函数与对数函数

③三角函数

```
1.  math.sin(x)
2.  math.cos(x)
3.  math.tan(x)
```

④常用常数

```
1.  math.pi     #圆周率π
2.  math.e      #e
3.  math.inf    #无穷大
4.  math.nan    #空值
```

认识Python标准库

❑Python 标准库

2. time 模块

（1）基本概念：time模块是Python中用于处理时间的一个内建模块。

相关概念介绍：

时间戳：从1970年1月1日00:00:00开始按秒计算的偏移量。

元组（struct_time）：struct_time元组共有9个元素，如表。gmtime()，localtime()，strptime()这三个函数会返回struct_time元组。

属性	值
tm_year	年(例如2019)
tm_mon	月(1–12)
tm_mday	日(1–31)
tm_hour	时(0–23)
tm_min	分(0–59)
tm_sec	秒(0–6)
tm_wday	周几(0–6)0是周日
tm_yday	一年中第几天(1–366)
tm_isdst	是否夏令时(默认–1)

认识Python标准库

❑Python 标准库

2. time 模块

（2）常用函数：数据处理中经常遇到的情况是，将时间戳或字符串形式的时间转化为标准的时间格式，方便进行日期的运算。

```
1.  import time
2.  start = time.time()              #记录代码开始运行时间,time.time()用于获取当前时间戳
3.  a = 1566897866                   #类型1:时间戳数据,常见的时间输入格式
4.  c = time.localtime(a)            #将时间戳转换为时间格式
5.  print(c)
6.  #输出结果如下:
    time.struct_time(tm_year=2019, tm_mon=8, tm_mday=27, tm_hour=17, tm_min=24, tm_sec=26,
    tm_wday=1, tm_yday=239, tm_isdst=0)
7.  a = "2017-6-11 17:51:30"         #类型2:字符型时间数据
8.  c=time.strptime(a,"%Y-%m-%d %H:%M:%S")
9.  print(c)
10. #输出结果如下:
11. #time.struct_time(tm_year=2017, tm_mon=6, tm_mday=11, tm_hour=17, tm_min=51, tm_sec
    =30, tm_wday=6, tm_yday=162, tm_isdst=-1)
12. c.tm_year                        #年
13. c.tm_mon                         #月
14. c.tm_mday                        #日
15. c.tm_hour                        #小时
16. c.tm_min                         #分钟
17. c.tm_sec                         #秒
18. c.tm_wday                        #星期几
19. c.tm_yday                        #到当年1月1日的天数
20. end = time.time()                #运行结束时间
21. timeSpent = end - start          #计算运行时间
22. print("Time spent: {0} s".format(timeSpent))
```

认识Python标准库

❑Python 标准库

3. random 模块

（1）基本概念：random模块是Python内建的用于生成随机数的模块。

（2）常用函数

```
1.  import random
2.  random.random()                          #生成0-1之间均匀分布的随机浮点数
3.  random.normalvariate(0, 1)               #生成1个符合均值为0,方差为1正态分布的随机数
4.  [random.normalvariate(0, 1) for x in range(10)]    #生成长度为10的正态分布序列
5.  random.uniform(a, b)                     #生成[a, b]区间内的随机浮点数
6.  random.randint(a, b)                     #生成[a, b]区间内的随机整数
7.  random.choice(s)                         #从序列s中随机获取一个值
8.  random.shuffle(s)                        #将序列s中元素打乱
9.  random.sample(s, k)                      #从序列s中获取长度为k的片段
```

认识Python标准库

❑ Numpy 基本用法

numpy是一个用于科学计算的基础模块，它提供了高性能的**数组对象、矩阵对象**，以及众多的基础数学、线性代数、基本统计运算等领域所需的相关函数，是基于Python进行大数据处理中常用的第三方模块。

1. 安装 numpy

如果需要安装numpy，可以在cmd（Windows系统）或terminal（Linux系统）中输入**pip** install numpy进行安装。若已安装Anaconda环境，则无需再安装numpy。

2. ndarray 对象

ndarray是numpy中的一种最基本的**数组对象**，与原生的列表类型相比，ndarray的大小是固定的，同时其**内部数据类型需要一致**，开发者针对高级数学运算，对ndarray进行了优化，因此其在数学运算的效率、稳定性上均有较大的优势，基本操作如下所示。

认识Python标准库

❑ Numpy 基本用法

2. ndarray 对象

```
1.  import numpy as np
2.  a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])    #定义一个ndarray对象
3.  #[[1, 2, 3],
4.  #[4, 5, 6],
5.  #[7, 8, 9]]
6.  a.ndim                  #ndarray的维数(阶),输出为2
7.  a.shape                 #ndarray的形状,输出为(3, 3),对应(行数,列数)
8.  a.size                  #ndarray的元素个数,输出为9
```

认识Python标准库

❑ Numpy 基本用法

2. ndarray 对象

对ndarray进行复制有三种形式，分别为：引用，浅复制和深复制。

```
1.  b = a                    #引用
2.  print(id(a)== id(b))     #id()能够输出对象的内存地址,输出为True
3.  c = a.view()             #浅复制
4.  print(id(a)==id(c))      #输出为False
5.  d = a.copy()             #深复制
```

认识Python标准库

❑ Numpy 基本用法

3. 多维数组的下标存取

numpy中数组的下标存取方法如下，与内置的list对象的存取和切片方法类似：

```
1.  a[0, 1]                 #输出2  (行、列的索引都是由0开始)
2.  a[[0, 2], [1, 2]]       #输出为array([2, 9]),等价于np.array([a[0, 1], a[2, 2]])
3.  a[:2, 1:3]              #第0至1行,第1至2列array([[2, 3], [5, 6]])
4.  a[2, 1:]                #第2行,第1至最后1列
5.  a[:-1]                  #第0至倒数第2行
6.  a[1, :]                 #第1行,shape为(3,),一维,array([4, 5, 6])
7.  a[1:2, :]               #第1行,shape为(1,3), 二维,array([[4, 5, 6]])
8.  a[:,1]                  #第1列,shape为(3, ), 一维,array([2, 5, 8])
9.  a[:,1:2]                #第1列,shape为(3, 1), 二维,array([[2], [5], [8]])
```

```
Titanic="C:/Users/Lenovo/Desktop/titanic_train.csv"
 chad = pd.read_csv(Titanic, engine='python')
```

认识Python标准库

❑ Numpy 基本用法

3. 多维数组的下标存取

Numpy中还能直接用判断条件（布尔矩阵）取出符合某些条件的元素：

```
1.  a>2                    #输出为 array([[False, False, True],
2.                         #[True, True, True],
3.                         #[True, True, True]])
4.  a[a>2]                 #输出数组中大于2的值，为一个一维数组
5.  a[(a>2) & (a<6)]       #使用逻辑运算符连接多个条件，注意不能使用and,or之类的关键词
    进行连接，否则会报错
```

认识Python标准库

Numpy 基本用法

4. 数组运算

```
1.  #逐元素运算
2.  a + b
3.  a * b
4.  a/b
5.  a ** 2
6.  np.sin(a)
7.  #矩阵运算
8.  a.dot(b)                    #a与b矩阵相乘
9.  a = np.mat(a)               #转换为矩阵对象
10. a.I                         #逆矩阵
11. a.T                         #转置
12. a.trace()                   #迹
13. np.linalg.det(a)            #矩阵a的行列式
14. np.linalg.norm(a,ord=None)  #矩阵a的范数
15. np.linalg.eig(a)            #矩阵a的特征值和特征向量
16. np.linalg.cond(a,p=None)    #矩阵a的条件数
```

认识Python标准库

❑ Numpy 基本用法

5. 随机数生成

1. np.random.normal(0, 1, 100) #生成均值为0，方差为1（不是标准差），长度为100的正态分布样本
2. np.random.poisson(5, 100) #生成均值为5，长度为100的泊松分布样本
3. np.random.negative_binomial(1, 0.1, 100) #生成n=1, p=0.1，长度为100的负二项分布样本

6.知识拓展

numpy中还提供了许多方便实用的内置函数，以及一些高级编程机制，具体内容可参考官方文档进行了解并多加练习。

参见：Cox D R. The regression analysis of binary sequences［J］. Journal of the Royal Statistical Society: Series B （Methodological）, 1958, 20（2）: 215-232.

认识Python标准库

❑Pandas基本用法

Pandas是一个数据分析模块，它提供了快速、灵活并且直观的类似SQL中的表的数据结构，从而能够便捷高效地处理具有不同字段、不同来源的数据。

1. 安装Pandas

如果想安装Pandas，通过Win+R进入运行界面，输入cmd进入命令行窗口，输入pip install pandas。Anaconda 环境同样自带了Pandas，因此，如果已安装了Anaconda，则无需安装Pandas。

2. DataFrame 数据类型

Pandas的基本数据类型是DataFrame，它**类似于表的概念**，能够用标签对不同的列进行标识，从而能够更为直观便捷地对不同数据进行处理。

认识Python标准库

❑Pandas基本用法

2. DataFrame 数据类型

```
1.  import pandas as pd
2.  df = pd.DataFrame([[1, 2, 3], [4, 5, 6],
    [7, 8, 9]])
3.  df.columns = ['A', 'B', 'C'] #定义表头
4.  df['A']                      #取一列,
                                 输出为
5.  #0    1
6.  #1    4
7.  #2    7
8.  df[['A', 'B']]               #取多列,
                                 输出为
9.  #AB
10.  #0  1  2
11.  #1  4  5
12.  #2  7  8
13.   df[1:3]                   #取1到2行,输出为
14.   #A  B  C
15.   #1  4  5  6
16.   #2  7  8  9
17.   df.iloc[1:3, 2]           #多维索引,输出为
18.   #1    6
19.   #2    9
20.   #切片需要用.iloc
21.   df[df['A']>1]             #根据条件选择,输出为
22.   #A  B  C
23.   #1  4  5  6
24.   #2  7  8  9
```

认识Python标准库

❑Pandas基本用法

2. DataFrame 数据类型

每个DataFrame的一列是一个Series，可理解为每个DataFrame由多个Series组成。

```
1.  print(type(df.A))                          #输出<class 'pandas.core.series.Series'>
2.  tmp = pd.Series([1,2,3], name='tmp')       #创建一个名为tmp的Series
```

3.文件的读取与写入

```
1.  df = pd.read_csv('file.csv')                          #读取csv格式文件,自动识别第一行为表头
2.  df = pd.read_csv('file.csv', names=['A','B','C'])     #指定表头
3.  df = pd.read_csv('file.csv', index_col=0)             #指定索引列
4.  df.to_csv('newfile.csv')                              #在工作目录下新建csv文件
```

认识Python标准库

❑Pandas基本用法

4. 队列进行操作

（1）创建新列

在Pandas中，创建新列比较简单，例如，假设需要建立一个新的名为“new”的新列，值为1，按照如下代码进行操作：

```
1.  df['new'] = 1                                  #新建一列(添加为最后一列),赋值为1
2.  df['new1'] = range(len(df))                    #新建一列,赋值为由0开始的递增序列
3.  df['date'] = ['20190801', '20190702', '20190601']  #新建一列存储日期
```

（2）基于已有列进行计算

在实际的数据分析中，需要基于已有的字段进行处理，获得新的字段，对于简单的情形，可按以下方法进行：

```
1. df['new'] = df['A'] + 1                 #新建一列，赋值为列A加1
2. df['new'] = df['A'] + df['B']           #新建一列，赋值为列A与列B的和
3. #新建一列，赋值为列A与列B的商，转换为numpy数组进行计算，能提升性能
4. df['new'] = df['A'].values / df['B'].values
```

认识Python标准库

❑Pandas基本用法

4. 队列进行操作

在其他计算较为复杂的情况下，需要利用apply语句，传入自定义函数进行列的运算，通常来说，此种用法更为通用。

```
1.  #任务:新建一列,将date字段的月份取出,并转换为int类型,此处使用了匿名函数,
    也可直接传入已有函数的函数名
2.  df['new'] = df['date'].apply(lambda x: int(x[4:6]))
3.  #任务:新建一列,对两列进行apply操作,获得列A与列B的和
4.  df['new'] = df[['A', 'B']].apply(lambda x: x[0]+x[1], axis=1)
```

认识Python标准库

❑Pandas基本用法

5. 分组操作

分组操作能够将具有一个或多个相同字段的样本分为同一组，进而对这些组进行取平均值、最大值等操作，是一个常用的统计方法。

```
#划分
grouped = df.groupby(by=['A'])    #按列A进行分组
grouped['C']                      #取分组后的C列
df['C'].groupby(df['A'])          #等价于上面两行语句
```

在对数据进行分组后，若希望获得各组单独的数据，需要进行组间的迭代。

```
for name, group in grouped:
    print(name)
    print(group)
    grouped.get_group('xx')   #直接获取列A的值为 'xx' 的组,例如:grouped.get_group(1)
```

认识Python标准库

❑Pandas基本用法

5. 分组操作

分组的目的是获得各组的数据分布，或进行进一步的计算。

```
1.  grouped.sum()          #对各组进行单独求和
2.  grouped.size()         #获取各组的大小
3.  grouped.mean()         #获取各组的均值
4.  grouped.min()          #获取各组最小值
5.  grouped.max()          #获取各组最大值
6.  grouped.std()          #获取各组标准差
7.  grouped.var()          #获取各组方差
8.  #使用自定义函数
9.  grouped.aggregate(f)   #f为自定义函数,例如,np.mean,也可采用匿名函数的形式
```

认识Python标准库

❑Pandas基本用法

5. 分组操作

以下为一个小例子，统计不同车型的平均车速：

```
1.  df = pd.DataFrame({'VehicleType': ['Car', 'Truck', 'Car', 'Truck'], 'Speed': [67., 43., 72., 49.]})
2.  df.groupby(['VehicleType']).mean()
3.  #输出
4.  #VehicleType    Speed
5.  #Car        69.5
6.  #Truck       46.0
```

6. 数据拼接

在实际的数据处理过程中，往往存在相同来源数据的多个部分或不同来源的数据进行关联的需求，而Pandas也具有满足此类需求的功能。

Pandas中数据拼接的方法主要有三种，分别为append, concat以及merge方法。其中，append方法是简单的竖向拼接。

认识Python标准库

❑Pandas基本用法

6. 数据拼接

Pandas中数据拼接的方法主要有三种，分别为append, concat以及merge方法。其中，append方法是简单的竖向拼接。

```
1.  df = pd.DataFrame([[1, 2], [3, 4]], columns=list('AB'))
2.  df2 = pd.DataFrame([[5, 6], [7, 8]], columns=list('AB'))
3.  df.append(df2)
4.  #输出
5.  #    A  B
6.  #0   1  2
7.  #1   3  4
8.  #0   5  6
9.  #1   7  8
```

认识Python标准库

❑Pandas基本用法

6. 数据拼接

而concat方法的一大特点是，其能够同时拼接2个及2个以上的DataFrame，也能够自定义连接的方向以及索引匹配的方式。

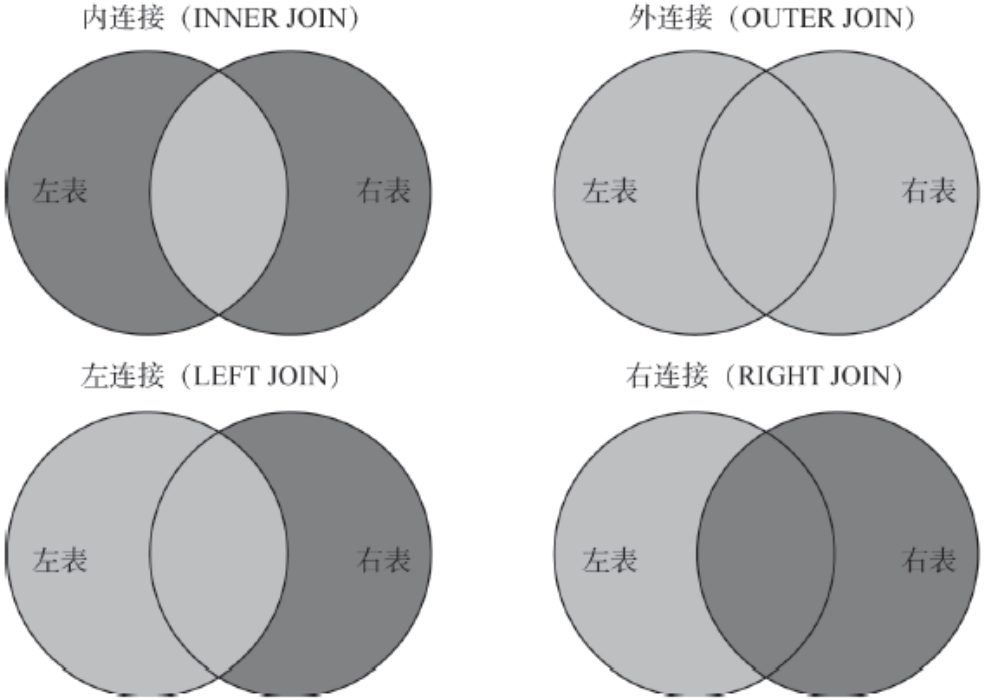
```
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number'])
df2 = pd.DataFrame([['c', 3], ['d', 4]], columns=['letter', 'number'])
pd.concat([df1, df2], axis=0)
#输出
#  letter  number
#0    a     1
#1    b     2
#0    c     3
#1    d     4
```

认识Python标准库

❑Pandas基本用法

6. 数据拼接

merge方法与SQL语言中表连接的方法类似，也最为灵活，在具体应用中可自定义连接的规则，图中列举了merge语句中能够实现的一些不同的连接方法。



认识Python标准库

❑Pandas基本用法

7.Pandas 中的统计特征函数

统计特征函数主要用于计算统计数据的均值、方差、标准差、分位数、相关系数和协方差等，这些统计特征反映出数据的总体分布特征。本小节所介绍的统计特征函数如表所示，它们主要作为Pandas的对象DataFrame或Series的统计方法使用。

函数名	函数功能
sum()	数据样本的总和(按列计算)
mean()	数据样本的算术平均数
median()	数据样本的算术中位数(50%分位数)
quantile()	样本的分位数(0到1)
min()、max()	数据样本的最大值和最小值
argmin()、agrmax()	最小值和最大值的索引位置(整数,Series类型适用)
var()	数据样本的方差
std()	数据样本的标准差
corr()	数据样本的Spearman(Pearson)相关系数矩阵
cov()	数据样本的协方差矩阵
diff()	数据样本的一阶差分(对时间序列很有用)
skew()	样本值的偏度(三阶阵)
kurt()	样本值的峰度(四阶阵)
describe()	样本的基本描述(基本统计量如均值、标准差等)

认识Python标准库

❑Pandas基本用法

7.Pandas 中的统计特征函数

上述统计特征函数的命令格式为D.function()，按列计算样本D的各类统计指标，样本D的数据类型可为DataFrame或者Series。这些统计函数也能够用在如2.8.4小节中的分组统计部分。某些统计函数可以添加函数选项，进行功能拓展，例如：

（1）corr()函数

```
D.corr(method= ' pearson ' )
```

样本D可为DataFrame类型，返回相关系数矩阵，method参数为计算方法，支持pearson（皮尔逊相关系数，默认选项）、kendall（肯德尔系数）、Spearman（斯皮尔曼系数）等；计算两个Series之间（或DataFrame的两列之间）的相关系数可采用S1.corr（S2, method=′ pearson′ ）的格式，其中S1、S2均为Series。

（2）describe()函数

```
D.describe()函数
```

括号里可以带一些参数，比如percentiles=[0.2, 0.4, 0.6, 0.8]就是指定只计算第20、40、60、80百分位数，而不是默认的第25、50、75百分位数。

认识Python标准库

❑高级统计特征函数

累积计算函数（**cum()**）

移动窗口函数（**rolling()**）

指数加权函数（**ewm()**）

累积计算函数（**cum()**）

函数名	函数功能
cumsum()	依次给出前1, 2,…,n个数的和
cumprod()	依次给出前1, 2,…,n个数的积
cummax()	依次给出前1, 2,…,n个数的最大值
cummin()	依次给出前1, 2,…,n个数的最小值

认识Python标准库

❑高级统计特征函数

移动窗口函数（rolling()）

函数名	函数功能
rolling(windowSize).count()	各窗口非NA观测值的数量，windowSize为移动窗口长度，下同
rolling(windowSize).sum()	移动窗口的和
rolling(windowSize).mean()	移动窗口的算术平均值
rolling(windowSize).median()	移动窗口的中位数
rolling(windowSize).var() rolling(windowSize).std()	移动窗口的方差和标准差。分母为n-1
rolling(windowSize).min() rolling(windowSize).max()	移动窗口的最小值和最大值
rolling(windowSize).corr()	移动窗口的相关系数
rolling(windowSize).cov()	移动窗口的协方差
rolling(windowSize).skew()	移动窗口的偏度（三阶矩）
rolling(windowSize).kurt	移动窗口的峰度（四阶矩）
ewm().mean	指数加权移动平均
ewm().var()、ewm().std()	指数加权移动方差和标准差
ewm().corr()、ewm().cov()	指数加权移动相关系数和协方差

认识Python标准库

❑高级统计特征函数

指数加权函数（ewm()）

函数名	函数功能
ewm().mean	指数加权移动平均
ewm().var()、ewm().std()	指数加权移动方差和标准差
ewm().corr()、ewm().cov()	指数加权移动相关系数和协方差

以上函数也是作用在DataFrame或Series对象上，其命令格式为D.function()。例如，D.rolling(windowSize).mean()语句的意思是对每windowSize 行计算一次均值，滚动计算；类似的，指数加权平均函数的用法为D.ewm().mean()。

认识Python标准库

❑ Matplotlib

1. Matplotlib 安装

方法一：按Win+R 进入运行界面，输入cmd 进入命令行窗口，输入pip install matplotlib 和pip installseaborn进行安装；

方法二：在Anaconda Navigator中搜索模块名，按提示进行安装。

2. Matplotlib 简介

Matplotlib是一种基本的绘图工具，它主要有以下几大类功能：

（1）基本绘图：Matplotlib自带了一些常用图形的绘制函数，例如散点图、条形图、直方图、堆积图、饼图、轮廓图等；

（2）精细化绘图：Matplotlib不仅能绘制基本的图表，还能基于各类参数对绘制图表的颜色、风格、坐标轴的范围和标签、图片的标注等进行深度定制和修改。

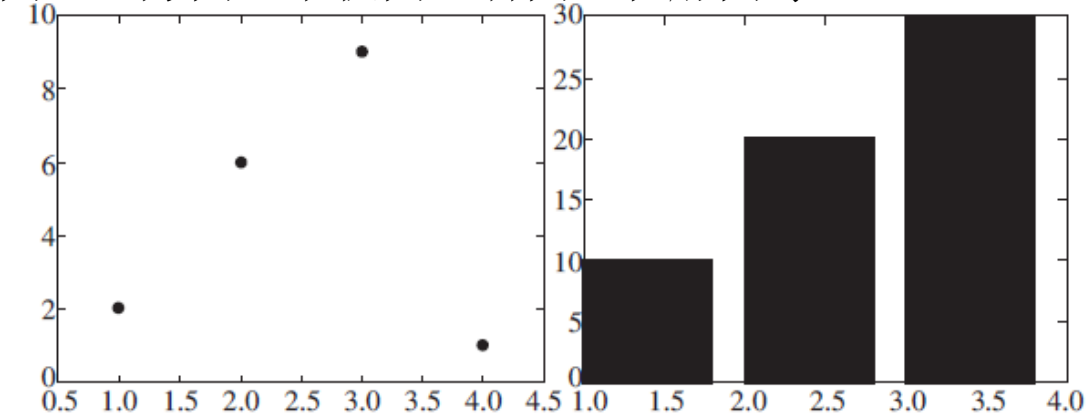
（3）拓展应用：Matplotlib还有其他高阶绘图功能，例如3D绘图、地理绘图等。

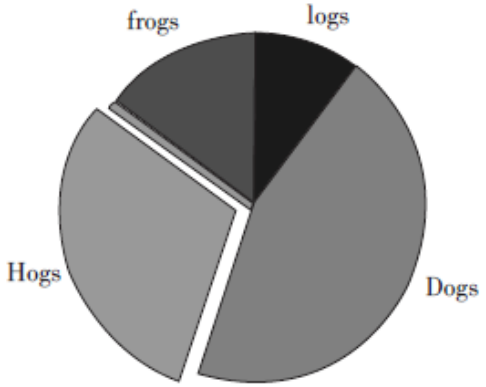
认识Python标准库

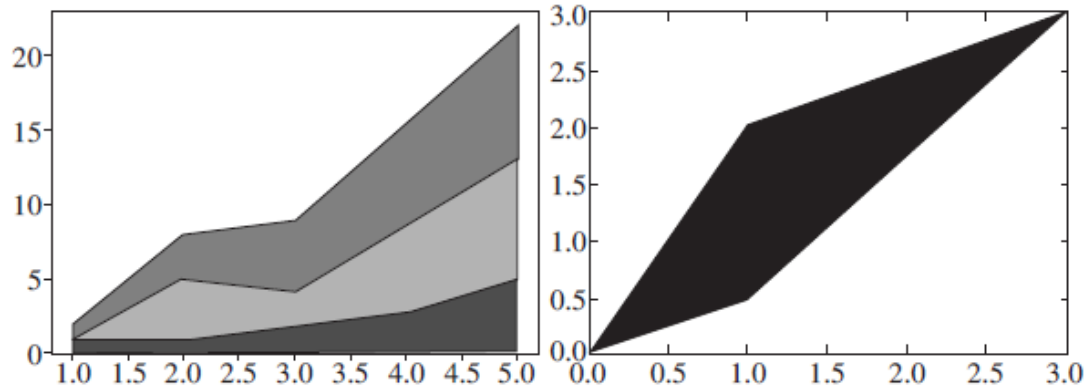
❑ Matplotlib

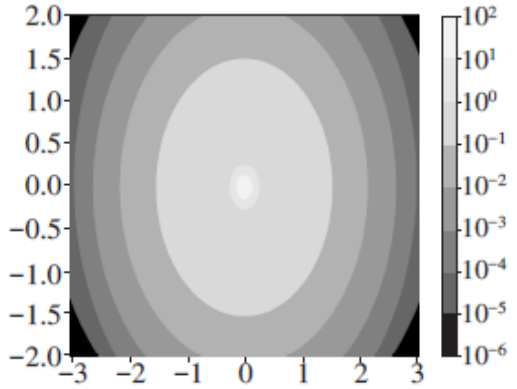
2. Matplotlib 简介

（1）基本绘图：Matplotlib自带了一些常用图形的绘制函数，例如散点图、条形图、直方图、堆积图、饼图、轮廓图等。







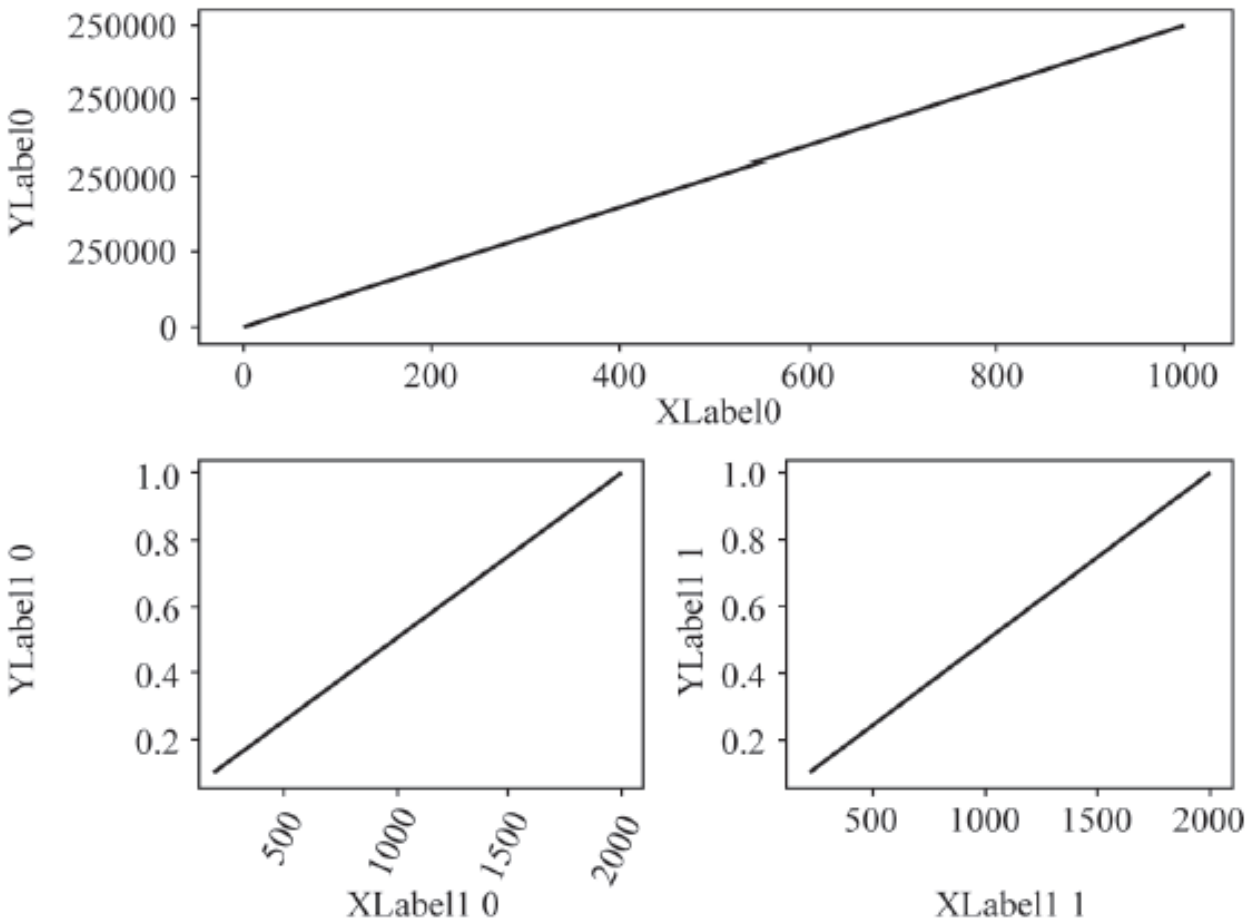


认识Python标准库

❑ Matplotlib

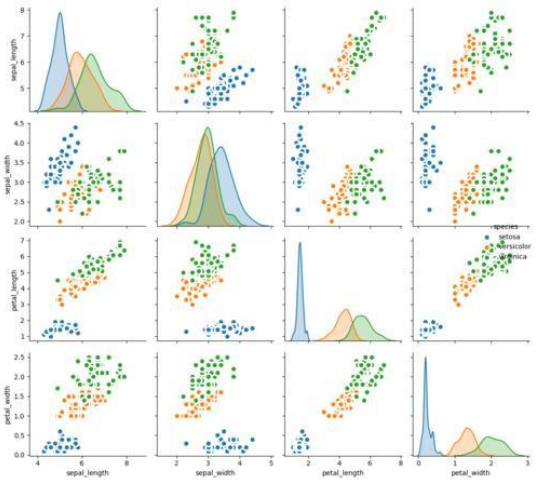
2. Matplotlib 简介

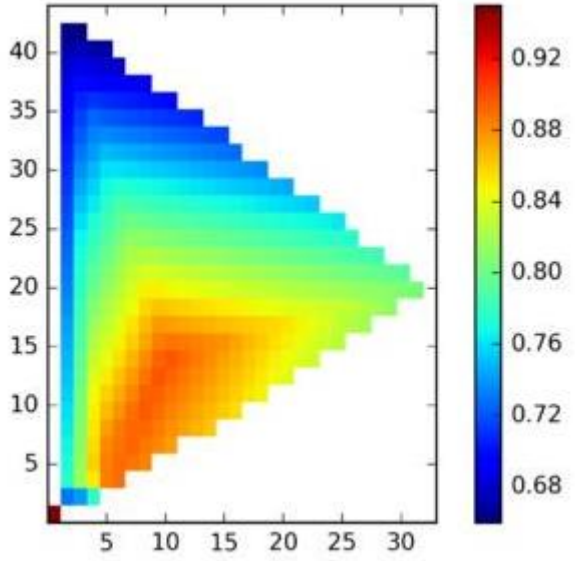
（2）精细化绘图：

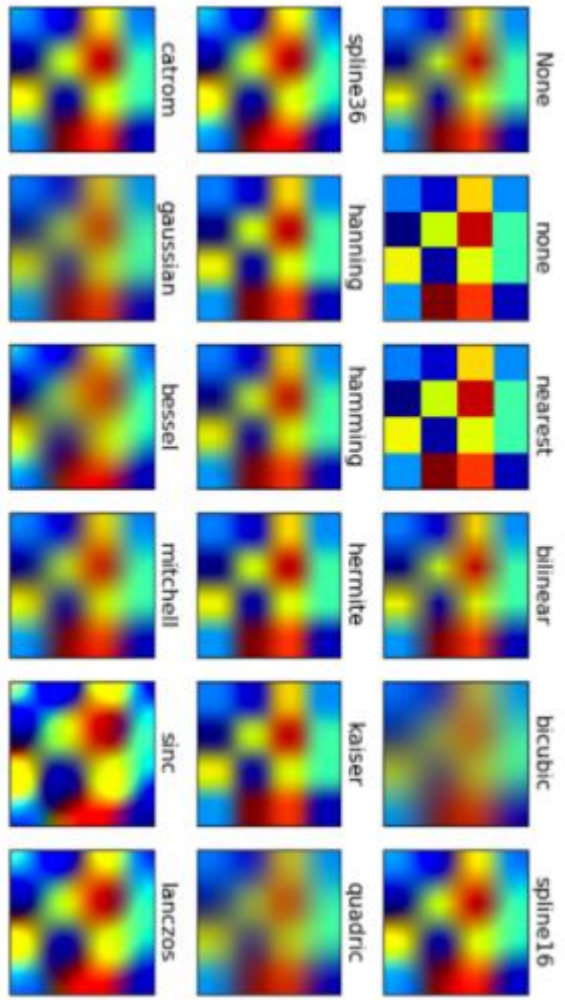


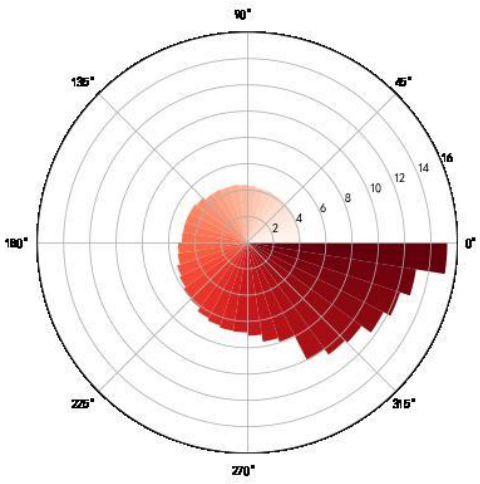
认识Python标准库

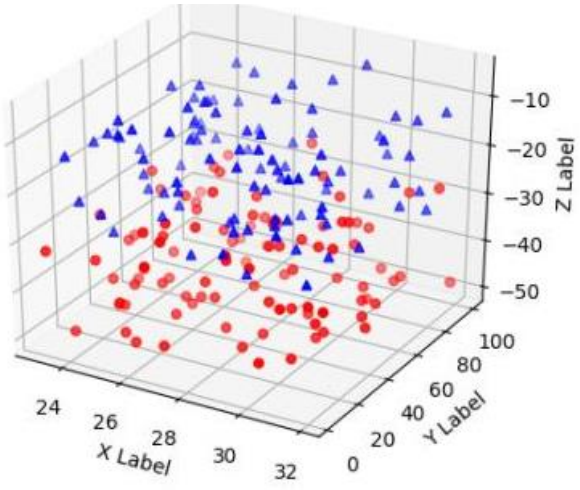
- Matplotlib









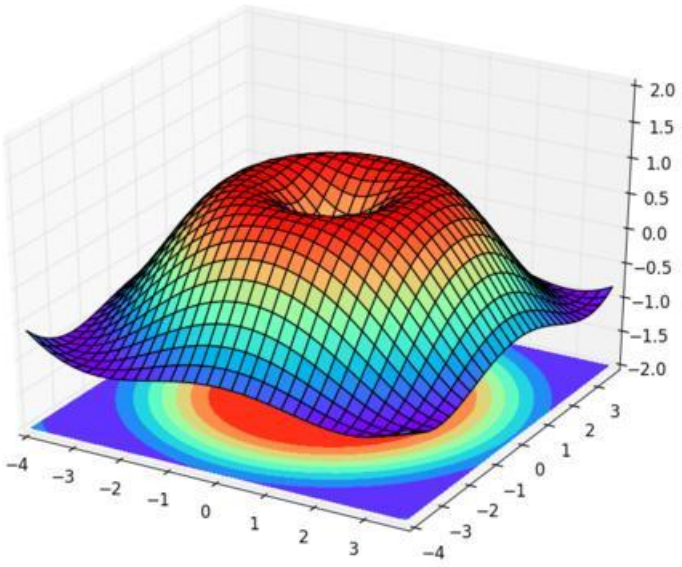


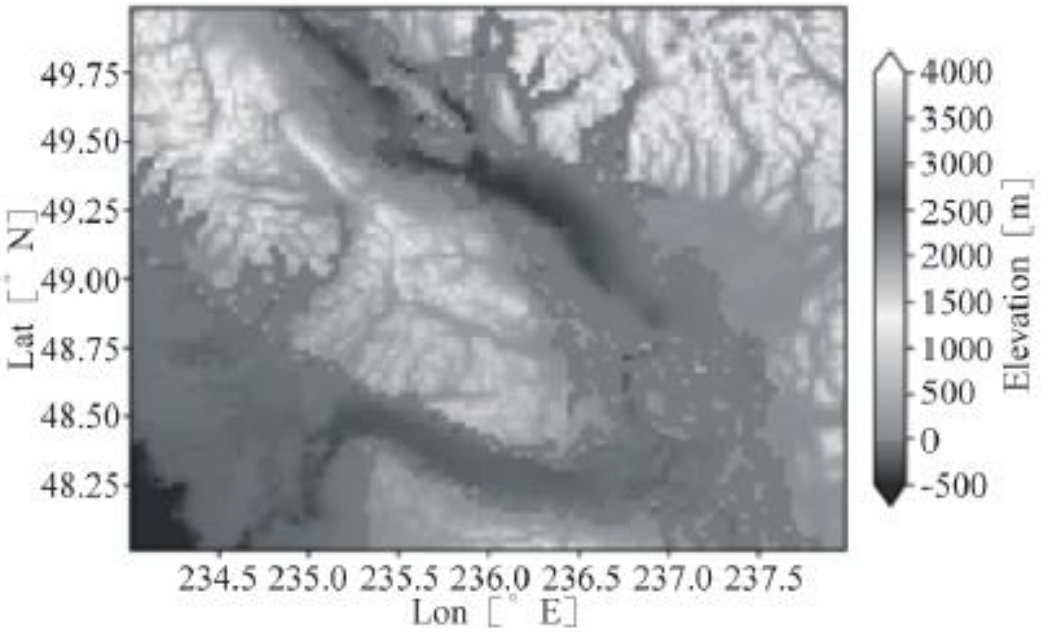
认识Python标准库

❑ Matplotlib

2. Matplotlib 简介

（3）拓展应用：Matplotlib还有其他高阶绘图功能，例如3D绘图、地理绘图等。





认识Python标准库

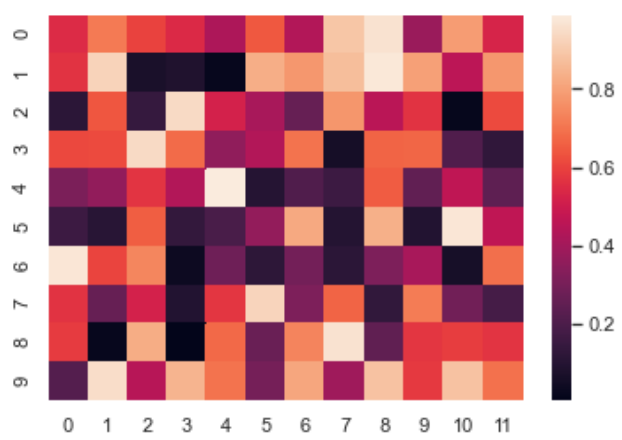
❑ Matplotlib

3. Seaborn 简介

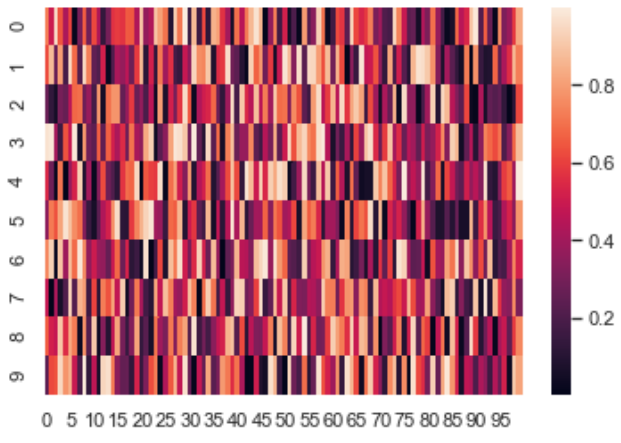
Seaborn是一个基于Matplotlib的高层级科学绘图模块。

Seaborn绘制热力图的代码及输出结果图：

```
1. Import  numpy as np
2. import seaborn as sns
3. np.random.seed(0)                          #将0设置为随机种子
4. sns.set()                                  #设置绘图风格
5. uniform_data = np.random.rand(10, 12)      #生成随机数据
6. ax = sns.heatmap(uniform_data)             #绘制热力图，输出结果如图所示
```



ttps:



s/index.html

认识Python标准库

❑ Scikit-learn

1. Scikit-learn 安装

方法一：按Win+R进入运行界面，输入cmd进入命令行窗口，输入pip install scikit-learn进行安装；

方法二：在Anaconda Navigator中搜索模块名，按提示进行安装。

2. Scikit-learn 简介

Scikit-learn（也称sklearn）是一个强大的机器学习工具模块，模块内封装了在构建机器学习模型时需要用到的相关功能，该模块的开发者对各类模型的调用方法进行了统一化处理，从而减少了使用者的学习成本，能够快捷方便地实现所需的功能。

认识Python标准库

❑ Scikit-learn

3. 数据预处理

原始数据往往充满了噪声和错误，需要经过清洗才能够作为模型的输入，否则将导致模型产出无意义的输出结果。Scikit-learn模块提供了一些接口函数，能够直接满足标准化、数据缩放、特征编码、缺失值填充、样本空间降维等数据预处理过程中常见的需求。

4. 常用模型

Scikit-learn中内置了许多常用的模型：

1）对于分类问题，涵盖了支持向量机、贝叶斯模型、决策树、集成学习等方法；

2）对于回归问题，最简单的线性模型、广义线性模型、支持向量机回归、决策树回归、岭回归等模型均包括在内；

3）对于聚类问题，模块实现了k-means、DBSCAN、层次聚类等多种常见算法，并在模型中提供了多样的聚类参数可供调节；

4）另外，模块也提供了模型参数标定、模型选择的方法。

认识Python标准库

❑ Scikit-learn

线性回归

```
from sklearn.linear_model import LinearRegression
# 定义线性回归模型
model = LinearRegression(fit_intercept=True, normalize=False,
    copy_X=True, n_jobs=1)
"""
参数
---
    fit_intercept：是否计算截距。False-模型没有截距
    normalize： 当fit_intercept设置为False时，该参数将被忽略。 如果为真，则回归前的回归系数x将通过减去平均值并除以l2-范数而归一化。
     n_jobs：指定线程数
"""
```

认识Python标准库

❑ Scikit-learn

逻辑回归LR

```
from sklearn.linear_model import LogisticRegression
# 定义逻辑回归模型
model = LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0,
    fit_intercept=True, intercept_scaling=1, class_weight=None,
    random_state=None, solver='liblinear', max_iter=100, multi_class='ovr',
    verbose=0, warm_start=False, n_jobs=1)

"""参数
---
    penalty: 使用指定正则化项（默认：l2）
    dual: n_samples > n_features取False（默认）
    C: 正则化强度的反，值越小正则化强度越大
    n_jobs: 指定线程数
    random_state: 随机数生成器
    fit_intercept: 是否需要常量
"""
```

认识Python标准库

❑ Scikit-learn

朴素贝叶斯算法

```
from sklearn import naive_bayes
model = naive_bayes.GaussianNB() # 高斯贝叶斯
model = naive_bayes.MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)
model = naive_bayes.BernoulliNB(alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None)
"""
文本分类问题常用MultinomialNB
参数
---
    alpha：平滑参数
    fit_prior：是否要学习类的先验概率；false-使用统一的先验概率
    class_prior：是否指定类的先验概率；若指定则不能根据参数调整
    binarize：二值化的阈值，若为None，则假设输入由二进制向量组成
"""
```

认识Python标准库

❑ Scikit-learn

决策树

```
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion='gini', max_depth=None,
    min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
    max_features=None, random_state=None, max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
     class_weight=None, presort=False)
"""参数
---
    criterion : 特征选择准则gini/entropy
    max_depth: 树的最大深度, None-尽量下分
    min_samples_split: 分裂内部节点, 所需要的最小样本树
    min_samples_leaf: 叶子节点所需要的最小样本数
    max_features: 寻找最优分割点时的最大特征数
    max_leaf_nodes: 优先增长到最大叶子节点数
    min_impurity_decrease: 如果这种分离导致杂质的减少大于或等于这个值, 则节点将被拆分。
"""
```

认识Python标准库

❑ Scikit-learn

支持向量机

```
from sklearn.svm import SVC
model = SVC(C=1.0, kernel='rbf', gamma='auto')
"""参数
---
    C: 误差项的惩罚参数C
    gamma: 核相关系数。浮点数, If gamma is 'auto' then 1/n_features will be used instead.
"""
```

认识Python标准库

❑ Scikit-learn

k近邻算法

```
from sklearn import neighbors
#定义kNN分类模型
model = neighbors.KNeighborsClassifier(n_neighbors=5, n_jobs=1) # 分类
model = neighbors.KNeighborsRegressor(n_neighbors=5, n_jobs=1) # 回归
"""参数
---
    n_neighbors:  使用邻居的数目
    n_jobs: 并行任务数
"""
```

认识Python标准库

❑ TensorFlow

1.TensorFlow 简介

TensorFlow是谷歌开发的一个开源**机器学习**框架，它提供了不同层级的模型接口，从而使开发者既能实现简洁快速的模型调用，也能借助底层接口实现高度定制化的机器学习模型构建。基于TensorFlow，能够实现**多层感知机、卷积神经网络、循环神经网络以及各类复杂的深度学习模型**。

TensorFlow能够在CPU和GPU上运行，其中**GPU**能够极大地提升神经网络的训练和推理速度，但对电脑的GPU具有一定的性能要求，在不符合性能要求的电脑上（如没有独立显卡的商务笔记本）无法使用。需要注意，若要在GPU上使用TensorFlow，除安装TensorFlow库外，还需要安装部分显卡驱动程序，较为复杂，有条件的读者可以自行查阅TensorFlow官方文档。以下先介绍TensorFlow的安装方法，然后对Google开发的基于网页的Colaboratory 和TensorFlow 的高阶API——Keras 进行介绍。其中，Colaboratory 集成了TensorFlow环境，无需配置即可使用TensorFlow，网站实现类似于JupyterNotebook的功能，十分方便，非常适合在初期学习阶段进行使用。

认识Python标准库

❑ TensorFlow

2. TensorFlow2 版本安装

正常情况下，在命令行中输入以下命令（命令行的打开方法见前文），即可安装TensorFlow2的最新稳定版本。

```
python -m pip install tensorflow
```

若提示类似“不是内部或外部命令”的错误信息，参考本章2.8.1小节。

安装完成后，在Python命令行中，输入以下代码进行验证，若能够正常输出，则代表安装成功：

```
1. import tensorflow as tf
2. hello = tf.constant('Hello, TensorFlow!')
3. print(hello)
```

认识Python标准库

❑ TensorFlow

3.Colaboratory 使用方法

（1）进入以下链接：https://colab.research.google.com/notebooks/welcome.ipynb

（2）新建notebook。在打开的网页中，点击“文件”→“新建Python3记事本”，登陆后即可开始使用。新建文件方法与编辑界面 如图所示。

认识Python标准库

❑ TensorFlow

4.TensorFlow 高阶API——Keras 简介

使用Tensorflow的底层API的一大优势是能够对模型结构进行深度的自定义和调整，但在交通等交叉学科中，使用经过高度封装的高阶API能够在保证模型效果的前提下，简便快捷地实现各种深度学习模型。**Keras是一个基于TensorFlow，对TensorFlow底层API进行深度封装的模块，**最初Keras是一个独立的模块，现已经整合进TensorFlow模块中。以下简要介绍Keras的基本逻辑和调用方法。

Keras中模型调用可概括为以下几个步骤：**模型定义、模型编译、模型训练、模型校验**。深度学习模型一般由前后“连接”的层组成，Keras也遵循这个逻辑，用户只需要定义一个模型，将所需要的层添加到模型中，并定义各层参数即可。模型定义完成之后，需要对模型进行编译，并定义使用的优化方法、损失函数以及评价指标。最后再进行模型的训练和校验。整个过程结构清晰，容易理解。

认识Python标准库

❑ TensorFlow

4.TensorFlow 高阶API-Keras 简介

```
1.  from tensorflow import keras
2.  vocab_size = 10000                                  #词汇表的大小,可理解为一个参数,
                                                          目前无需深究
3.  #步骤一:模型定义
4.  model = keras.Sequential()
5.  model.add(keras.layers.Embedding(vocab_size, 16))   #添加Embedding层
6.  model.add(keras.layers.GlobalAveragePooling1D())    #添加池化层
7.  model.add(keras.layers.Dense(16, activation=tf.nn.relu))   #添加全连接层
8.  model.add(keras.layers.Dense(1, activation=tf.nn.sigmoid))  #添加全连接层
9.  model.summary()                                     #输出模型结构
10. #步骤二:模型编译
11. model.compile(optimizer='adam',                     #使用adam方法进行参数优化
12.           loss='binary_crossentropy',               #损失函数选用二元交叉熵
13.           metrics=['acc'])                          #评价指标采用准确率
14. x_val = trainData[:10000]                           #训练集中前10000个样本作为验证集
15. partial_x_train = trainData[10000:]
16. y_val = trainLabels[:10000]
17. partial_y_train = trainLabels[10000:]
18. #步骤三:模型训练
19. history = model.fit(partial_x_train,                #输入自变量
20.                 partial_y_train,                    #输入因变量(标签)
21.                 epochs=40,                          #训练次数为40
22.                 batch_size=512,                     #分批训练,每批样本数为512
23.                 validation_data=(x_val, y_val),
24.                 verbose=1)                          #显示每次训练的进度
25. #步骤四:模型校验
26. results = model.evaluate(testData, testLabels)
27. print(results)
```

认识Python标准库

本章小结

本章介绍了一种常用的高级编程语言—Python。

主要包括以下几项内容：

1）对Python基础语法、特性以及环境配置进行了介绍；

2）介绍了Python中模块的安装和使用方法；

3）介绍了几个常用模块的功能和基本用法；

4）介绍了TensorFlow等拓展性模块。

在学习完本章后，应对Python的基本用法、应用范围及适用场景有初步的了解。

Python是一种高效的数据分析工具，它可以帮助我们对数据进行处理和分析，实现我们所构建的算法。

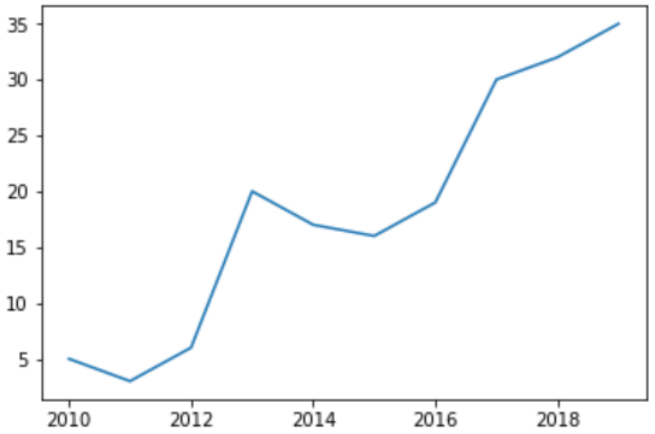
在掌握Python基本知识的基础上，基于Python进行交通大数据分析还需要更多理论和方法层面的支撑。后续的章节将进一步介绍相关内容。

认识Python标准库

❑ 练习

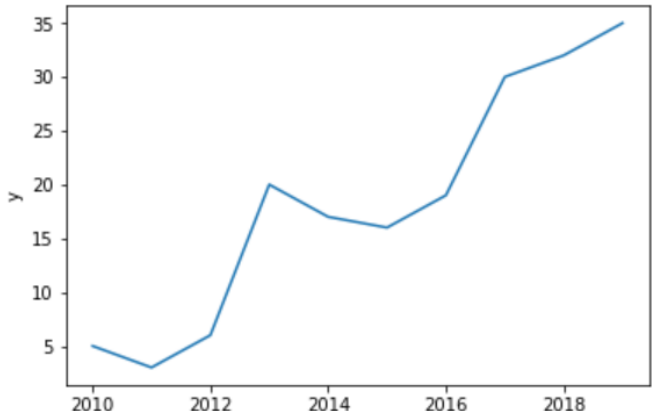
一、折线图

```
#import numpy as np
#import pandas as pd
#import matplotlib.pyplot as plt
# import seaborn as sns
```



Matplotlib

```
x = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019]
y = [5, 3, 6, 20, 17, 16, 19, 30, 32, 35]
plt.plot(x, y)
```



Seaborn

```
df = pd.DataFrame({'x': x, 'y': y})
sns.lineplot(x="x", y="y", data=df)
```

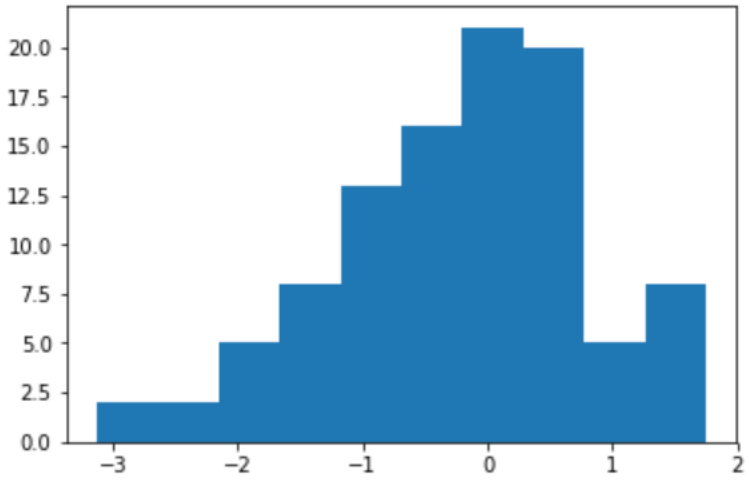
认识Python标准库

❑ 练习

二、直方图

a = np.random.randn(100)

s = pd.Series(a)



Matplotlib

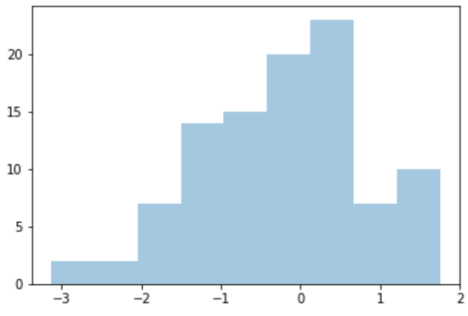
plt.hist(s)

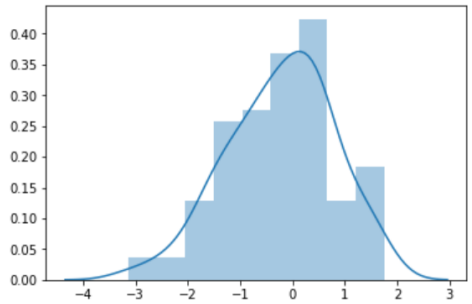
Seaborn

sns.distplot(s, kde=False)

plt.show()

sns.distplot(s, kde=True) plt.show()





认识Python标准库

❑ 练习

三、垂直条形图

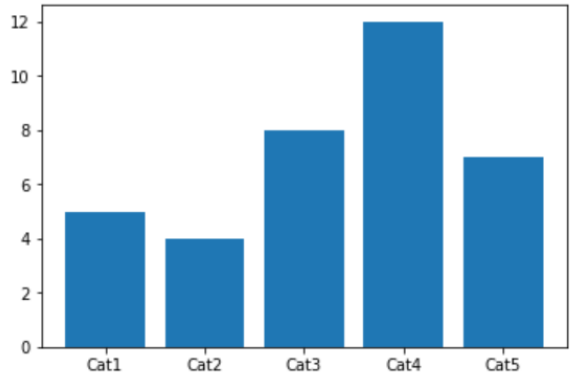
x = ['Cat1', 'Cat2', 'Cat3', 'Cat4', 'Cat5']

y = [5, 4, 8, 12, 7]

Matplotlib

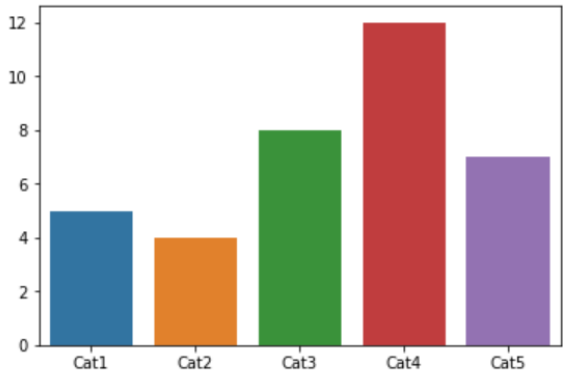
plt.bar(x, y)

plt.show()



Seaborn

sns.barplot(x,y)

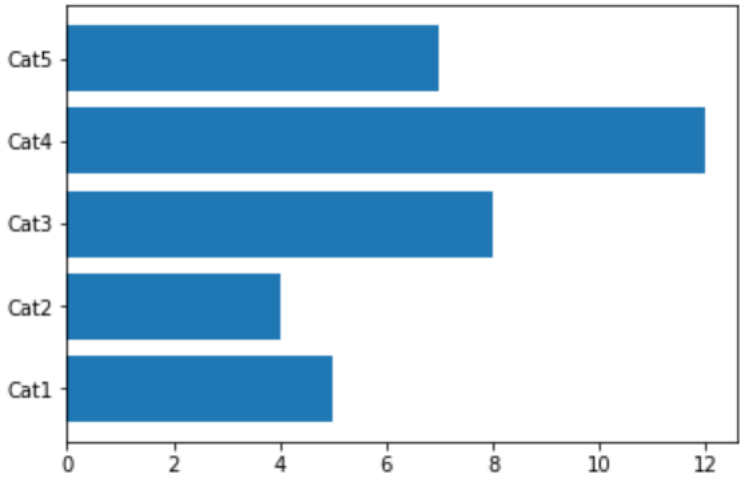


认识Python标准库

❑ 练习

四、水平条形图

```
x = ['Cat1', 'Cat2', 'Cat3', 'Cat4', 'Cat5']
y = [5, 4, 8, 12, 7]
plt.barh(x, y)
```



认识Python标准库

❑ 练习

五、饼图

```
nums = [25, 37, 33, 37, 6]
labels = ['High-school','Bachelor','Master','Ph.d', 'Others']
plt.pie(x = nums, labels=labels)
 plt.show()
```

认识Python标准库

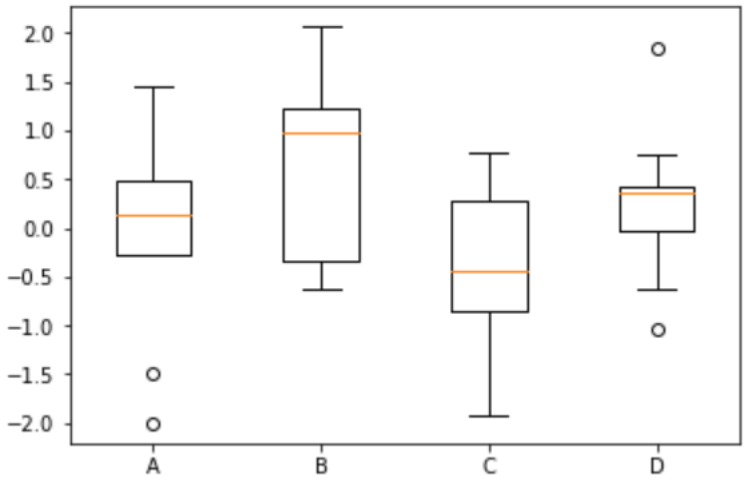
❑ 练习

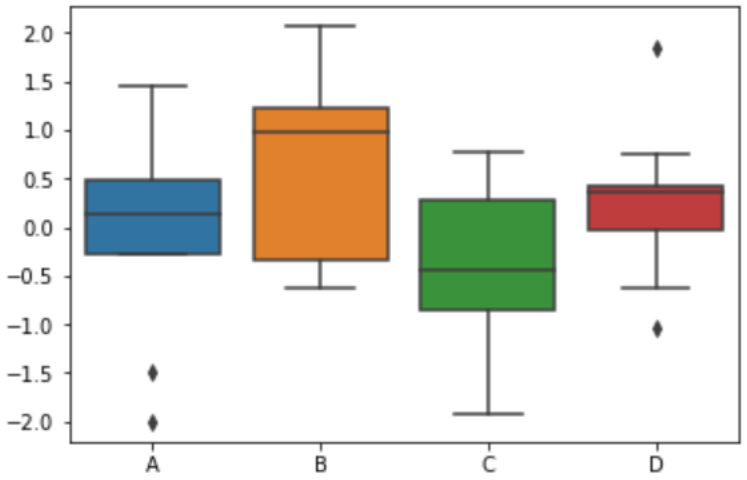
六、箱线图

```
data=np.random.normal(size=(10,4))
lables = ['A','B','C','D']

Matplotlib
plt.boxplot(data,labels=lables)
 plt.show()

Seaborn
df = pd.DataFrame(data, columns=lables)
 sns.boxplot(data=df)
 plt.show()
```



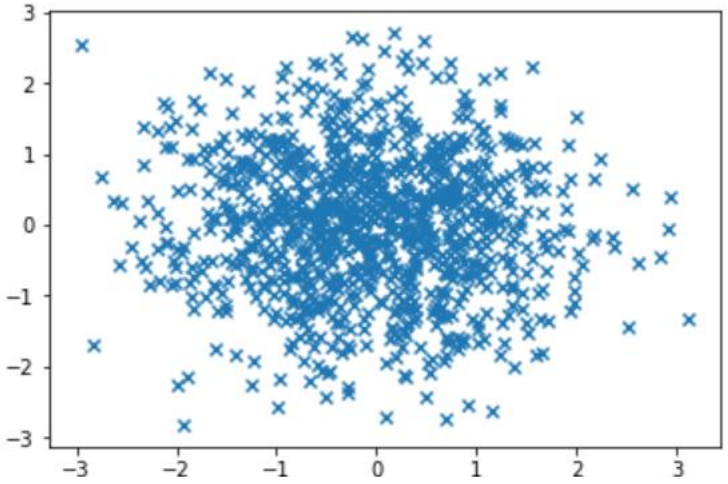


认识Python标准库

❑ 练习

七、散点图

```
N = 1000
x = np.random.randn(N)
y = np.random.randn(N)
```

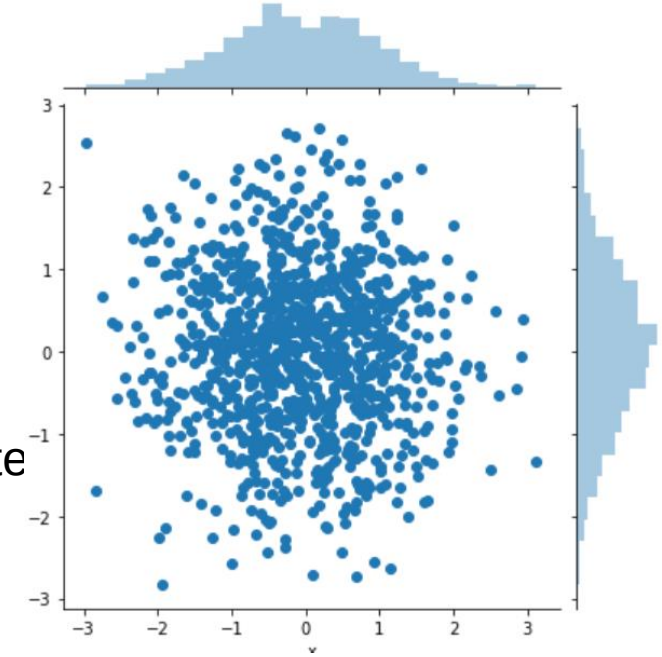


Matplotlib

```
plt.scatter(x, y,marker='x')
plt.show()
```

Seaborn

```
df = pd.DataFrame({'x': x, 'y': y})
 sns.jointplot(x="x", y="y", data=df, kind='scatte
 plt.show()
```



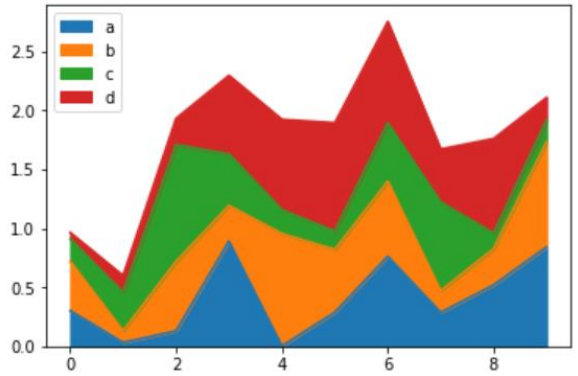
认识Python标准库

❑ 练习

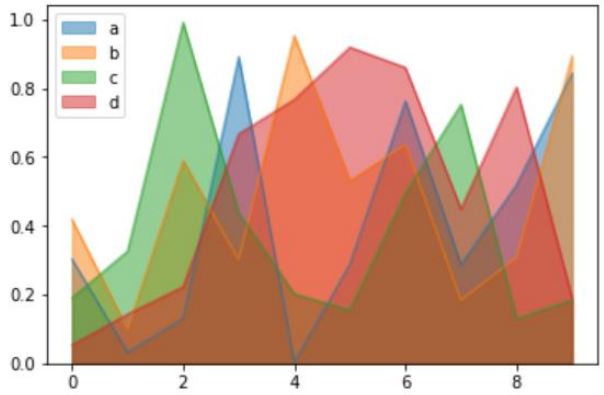
十一、面积图

```
df = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b','c', 'd'])
```

```
df.plot.area()
```




```
df.plot.area(stacked=False)
```



❑ 本章参考文献

［1］Mckinney W.利用Python进行数据分析［M］.北京：机械工业出版社，2018.

［2］UCImachine Learning Repository［EB/OL］［. 2020-06-10］.

［3］廖雪峰.Python教程［EB/OL］.［2020-06-10］.

［4］Python基础教程［EB/OL］.［2020-06-10］.

［5］The Python Standard Library［EB/OL］.［2020-06-10］.

［6］Numpy［EB/OL］.［2020-06-30］. https://www.numpy.org.

［7］Vutukuru A. Different Types of Joins in SQL Server［EB/OL］.［2019-01-08］.

［8］10Minutes to Pandas［EB/OL］.［2019-01-08］.

［9］Matplotlib Gallery［EB/OL］［. 2019-01-08］.

［10］Scikit-Learn User Guide［EB/OL］.［2019-01-08］.

［11］Text Classification with Preprocessed Text: Movie Reviews［EB/OL］. (2020-06-12)［2020-06-18］.