电子商务数据分析

第2章 数据采集与预处理

朱桂祥 (9120201070@nufe.edu.cn)
南京财经大学信息工程学院
江苏省电子商务重点实验室
电子商务信息处理国家级国际联合研究中心
电子商务交易技术国家地方联合工程实验室

https://github.com/zgx881205/Analysis-of-E-commerce-Data

南京财经大学 NANJING UNIVERSITY OF FINANCE & ECONOMICS

南京大学校训

自建校以来，南京大学就要求师生把“诚”作为立身立业、为人为学的根本，“诚朴雄伟，励学敦行”的校训就把“诚”放在了第一位。“诚”字也融入了南京大学百年发展历史，融入了培养学生、科学研究、服务社会、提振社会文化品质的所有细节和环节当中。“诚朴雄伟”：希望学子诚心向学，不计名利，承担复兴民族的责任；“励学敦行”：劝勉师生勤奋学习，在实践中展现自己的品格与抱负。南京大学校训凝聚着学校的核心价值观，她根植于中国优秀传统文化，延续着南大人百年来的悠长文脉。

http://tv.cctv.com/2014/10/24/VIDE1414112760898238.shtml

数据预处理

为什么要进行数据预处理

- **完整性（Completeness）**

数据的记录和信息是否完整，是否存在缺失的情况。数据的缺失主要有记录的缺失和记录中某个字段信息的缺失，两者都会造成统计结果的不准确，所以完整性是数据质量最基础的保障，而对完整性的评估相对比较容易。

- **一致性（Consistency）**

数据的记录是否符合规范，是否与前后及其他数据集合保持统一。数据的一致性主要包括数据记录的规范和数据逻辑的一致性。数据记录的规范主要是数据编码和格式的问题，比如网站的用户ID是15位的数字、商品ID是10位数字，IP地址一定是用“.”分隔的4个0-255的数字组成，及一些定义的数据约束，比如完整性的非空约束、唯一值约束等；数据逻辑性主要是指标统计和计算的一致性。

https://www.cda.cn/view/18013.html

数据预处理

- **准确性（Correctness）**

数据中记录的信息和数据是否准确，是否存在异常或者错误的信息。导致一致性问题的原因可能是数据记录的规则不一，但不一定存在错误；而准确性关注的是数据记录中存在的错误，比如字符型数据的乱码现象也应该归到准确性的考核范畴，另外就是异常的数值，异常大或者异常小的数值，不符合有效性要求的数值，如访问量Visits一定是整数、转化率一定是介于0到1的值等。

- **及时性（Timeliness）**

虽然说分析型数据的实时性要求并不是太高，但并不意味了就没有要求，分析师可以接受当天的数据要第二天才能查看，但如果数据要延时两三天才能出来，或者每周的数据分析报告要两周后才能出来，那么分析的结论可能已经失去时效性，分析师的工作只是徒劳；同时，某些实时分析和决策需要用到小时或者分钟级的数据，这些需求对数据的时效性要求极高。

https://www.cda.cn/view/18013.html

数据预处理

现实世界的数据：

- 不完整的
  缺少属性值或某些感兴趣的属性，或仅包含聚集数据。
- 含噪声的
  包含错误或存在偏离期望的离群值。
- 不一致的
  采用的编码或表示不同，如属性名称不同
- 冗余的
  如属性之间可以相互导出

数据错误的不可避免性：

- 数据输入和获得过程数据错误
- 数据集成所表现出来的错误
- 数据传输过程所引入的错误

数据预处理

数据预处理的形式:

- 数据清理

  补充缺失数据、平滑噪声数据、识别或删除离群点，解决不一致

- 数据集成

  集成多个数据库、数据立方或文件

- 数据变换

  规范化和聚集

- 数据归约

  简化数据、但产生同样或相似的结果

数据预处理

小结：

- 现实世界的数据一般是脏的、不完整的和不一致的。
- 数据预处理技术可以改进数据的质量，从而有助于提高其后的挖掘过程的精度和性能。
- 高质量的决策必然依赖于高质量的数据，因此数据预处理是知识发现过程的重要步骤。
- 检测异常数据、尽早地调整数据并归约待分析的数据，将在决策过程中得到高回报。

数据预处理

描述性数据汇总

获得数据的总体印象对于成功的数据预处理是至关重要的。

描述性数据汇总技术可以用来识别数据的典型性质，突显哪些数据值应当视为噪声或离群点。

- 动机：更好的理解数据。
- 主要内容：度量数据的中心趋势和离散程度、描述数据汇总的图形显示。
- 算数平均值：最常用
- 分布式度量:可以通过如下方法计算度量（即函数）：将数据集划分成较小的子集，计算每个子集的度量，然后合并计算结果，得到原（整个）数据集的度量值。sum()、count()、min()、max()
- 代数度量：可以通过应用一个代数函数于一个或多个分布度量计算的度。mean()、中列数
- 整体度量:必须对整个数据集计算的度量。例如，中位数、众数。

数据预处理

代数度量：

- 平均值 mean()：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- 加权平均：$\bar{x} = \frac{x_1 f_1 + x_2 f_2 + \cdots + x_n f_n}{n}$
- **截断均值**：去掉高、低极端值得到的均值。减小极端值对均值的影响。
- **中列数(midrange)**：是指样本中极大值与极小值的平均，(max+min)/2。

数据预处理

整体度量

- 中位数（median）：中位数是指将数据按大小顺序排列起来，形成一个数列，居于数列中间位置的那个数据：

$$M_e = \begin{cases} x_{\frac{n+1}{2}} & (n\text{为奇数}) \\ \dfrac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2} & (n\text{为偶数}) \end{cases}$$

设N个数值排序，若N为奇数，中位数是有序集的中间值；若N为偶数，中位数是中间两个值的平均值。

例如：1,3,5,7　　中位数4

　　1,3,5,6,7　中位数5

数据预处理

度量数据的离散程度:

- **极差:最大值与最小值之差**

- **四分位数:**

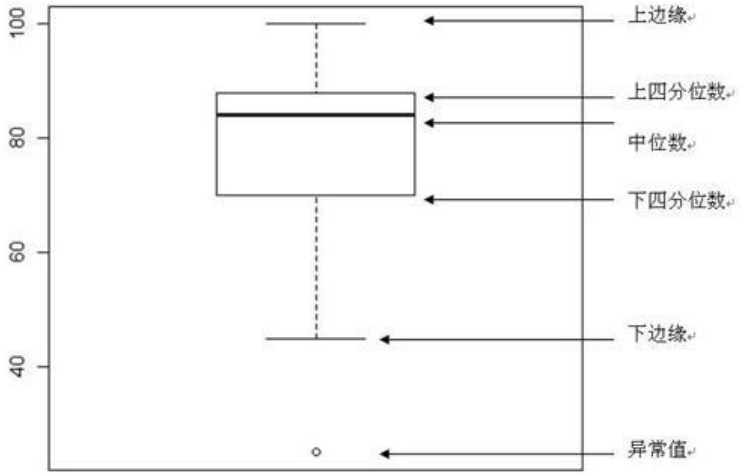
中位数是第50个百分位数，是第2个四分位数
第1个是第25个百分位数Q1

- **离群点outlier**

与数据的一般行为或模型不一致的数据对象

- **箱形图:**

箱形图主要用于反映原始数据分布的特征，还可以进行多组数据分布特征的比较。箱线图的绘制方法是：先找出一组数据的上边缘、下边缘、中位数和两个四分位数；然后，连接两个四分位数画出箱体；再将上边缘和下边缘与箱体相连接，中位数在箱体中间。



箱线图的功能

1.直观明了地识别数据集中的离群点
2.判断数据集的偏态和尾重
3.比较几批数据的形状

数据预处理

- 箱线图分析案例：

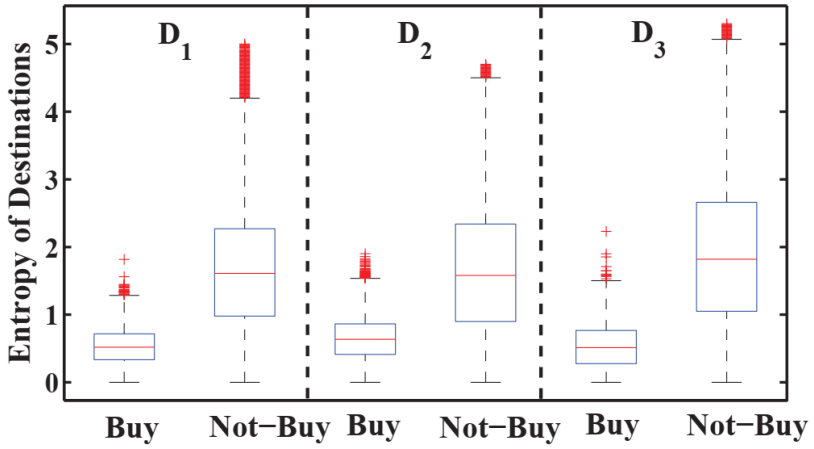


图 3.8 旅游区域分布的熵对购买意图的影响

我们调查顾客浏览的旅游包区域对在线购买决策的影响。箱线图比较了旅游区域分布的熵，此处较小的熵值代表用户聚焦于比较少的旅游区域。显而易见，在 3 个旅游数据集上，有购买行为的样本组中位数和变异系数都比较小。这意味着有很强购买意图的顾客更加喜欢浏览的旅游包以其感兴趣的地区为目的地。相比之下，广泛浏览各种旅游包的在线用户很难做出购买决定。

Guixiang Zhu, Zhiang Wu, Youquan Wang, Shanshan Cao, Jie Cao. Online purchase decisions for tourism e-commerce[J]. *Electronic Commerce Research and Applications*, 2019, 38: 100887.

数据预处理

数据规范化：

建模之前，都需要对数据进行标准化处理，以消除量纲的影响。如果对未标准化的数据直接进行建模，可能会导致模型对数值大的变量学习过多，而对数值小的变量训练不够充分，往往模型效果会不好。

■ 最小-最大规范化(Min-Max Normalization)：

也称为归一化，就是利用数据列中的最大值和最小值进行标准化处理，标准化后的数值处于[0,1]之间，但不改变原始数据的分布，计算方式为数据与该列的最小值作差，再除以极差。

$$y = (x - \min)/(\max - \min)$$

安装:
pip install sklearn
pip install numpy

问题：假设属性income的最小值和最大值分别是5000元和58000元。利用Min-Max规范化的方法将属性的值映射到0至1的范围内，那么属性income的16000元将被转化为多少？

数据预处理

数据规范化：

安装:
pip install sklearn
pip install numpy

- 最小-最大规范化(Min-Max Normalization) Python实现：

```
from sklearn import preprocessing
import numpy as np
x = np.array([[5000.],[58000.],[16000.]])
min_max_scaler = preprocessing.MinMaxScaler()
minmax_x = min_max_scaler.fit_transform(x)
print ('Income:',x)
print('After Min-Max Normalization:',minmax_x)
```

```
Income: [[ 5000.]
 [58000.]
 [16000.]]
After Min-Max Normalization: [[0.        ]
 [1.        ]
 [0.20754717]]
```

数据预处理

■ 零-均值规范化（Z-score Normalization）

也称为标准差标准化，经过处理的数据的均值为0，标准差为1:

$$x_i^* = \frac{x_i - \mu}{\sigma}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

$X=(x_1, x_2, \ldots, x_N)$,其中$x_i$为某个特征的原始值，$\mu$为该特征在所有样本中的平均值，$\sigma$为该特征在所有样本中的标准差，$x^*$为经过零-均值规范化处理后的特征值，符合标准正态分布，记为X ~N(0, 1) 。

若随机变量X服从一个数学期望为μ、方差为σ^2的正态分布，记为N(μ，σ^2)。其概率密度函数为正态分布的期望值μ决定了其位置，其标准差σ决定了分布的幅度。当μ = 0,σ = 1时的正态分布是标准正态分布。

数据预处理

数据规范化：

- **零-均值规范化 Python实现：**

安装:
pip install sklearn
pip install numpy

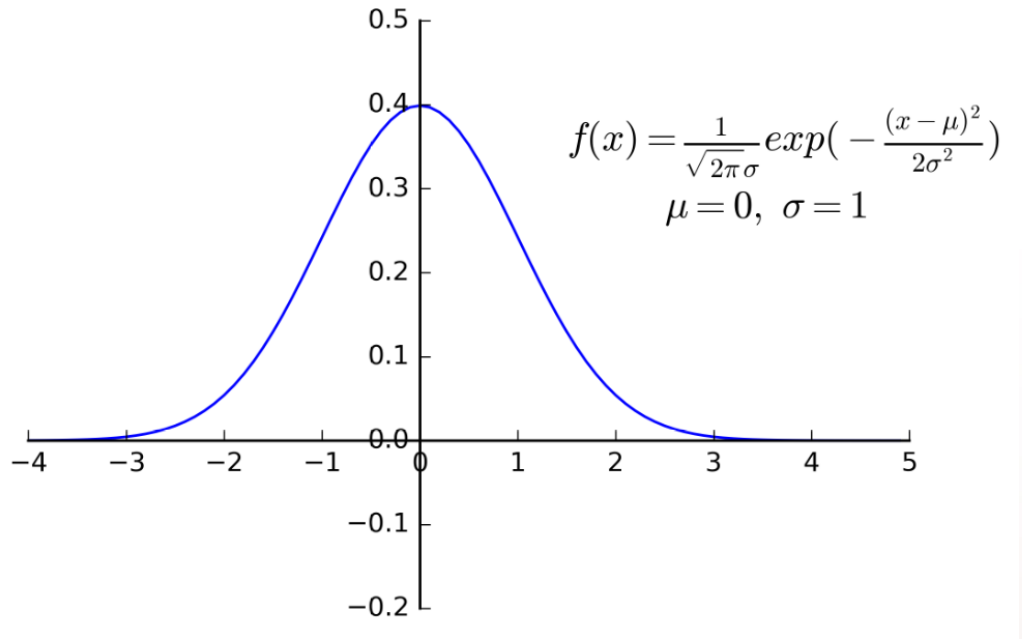
```
from sklearn import preprocessing
import numpy as np
X = np.array([[ 1., -1.,  2.], [ 2.,  0.,  0.],[ 0.,  1., -1.]])
X_scaled = preprocessing.scale(X)
print('X:',X)
print('After Z-score Normalization:',X_scaled)
```

```
X: [[ 1. -1.  2.]
 [ 2.  0.  0.]
 [ 0.  1. -1.]]
After Z-score Normalization: [[ 0.         -1.22474487  1.33630621]
 [ 1.22474487  0.         -0.26726124]
 [-1.22474487  1.22474487 -1.06904497]]
处理后数据的均值和方差: [0. 0. 0.] [1. 1. 1.]
```

数据预处理

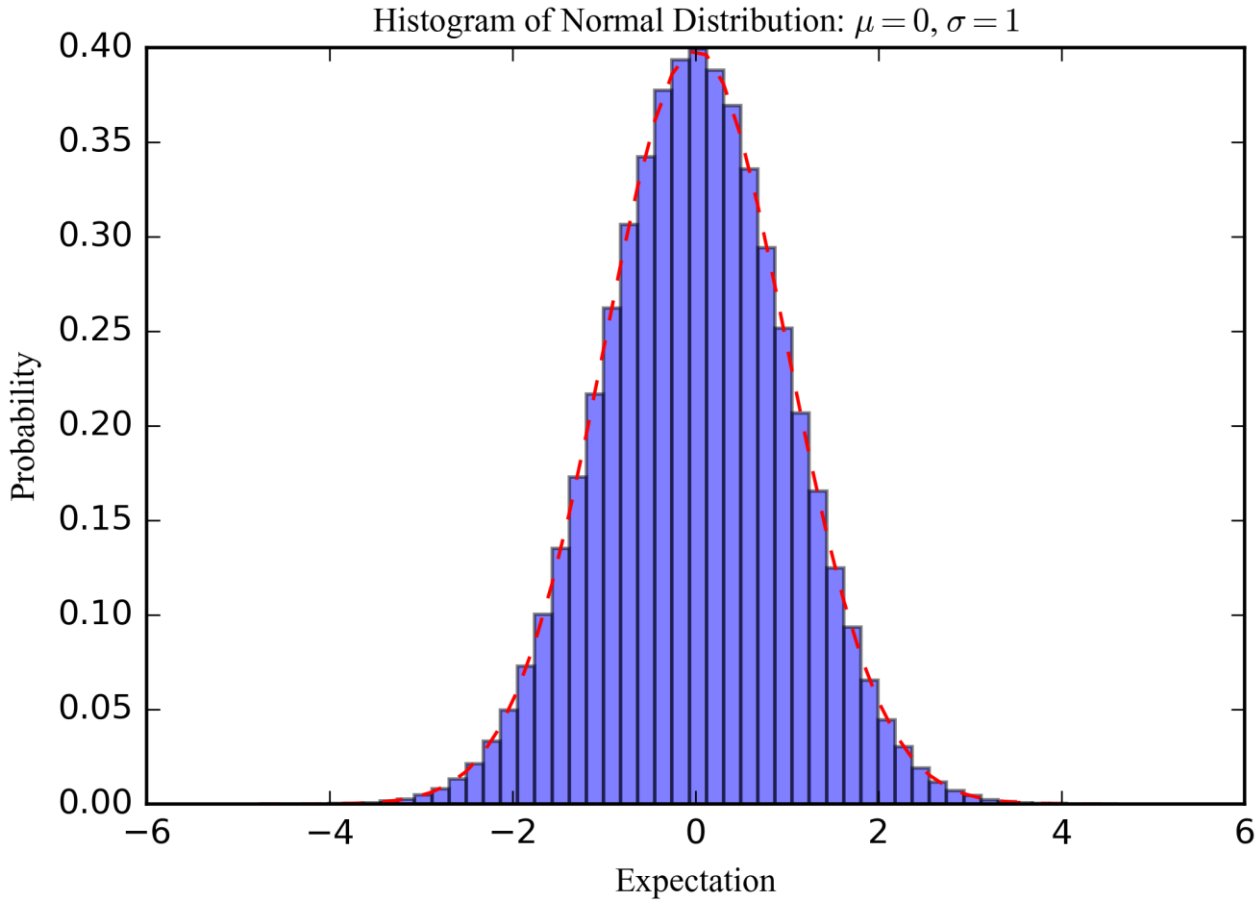
■ 零-均值规范化 Python实现：

```
import math
import pylab as pl
import numpy as np
def gd(x,m,s):
    left=1/(math.sqrt(2*math.pi)*s)
    right=math.exp(-math.pow(x-m,2)/(2*math.pow(s,2)))
    return left*right
def showfigure():
    x=np.arange(-4,5,0.1)
    y=[]
    for i in x:
        y.append(gd(i,0,1))
    pl.plot(x,y)
    pl.xlim(-4.0,5.0)
    pl.ylim(-0.2,0.5)
    ax = pl.gca()
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.spines['bottom'].set_position(('data',0))
    ax.yaxis.set_ticks_position('left')
    ax.spines['left'].set_position(('data',0))
    #add param
    label_f1 = "$\mu=0,\ \sigma=1$"
    pl.text(2.5,0.3,label_f1,fontsize=15,verticalalignment="top",
            horizontalalignment="left")
    label_f2 = r"$f(x)=\frac{1}{\sqrt{2\pi}\sigma}exp(-\frac{(x-\mu)^2}{2\sigma^2})$"
    pl.text(1.5,0.4,label_f2,fontsize=15,verticalalignment="top"
            ,horizontalalignment="left")
    plt.savefig("./Normal2.png", dpi=400)
    pl.show()
showfigure()
```



数据预处理

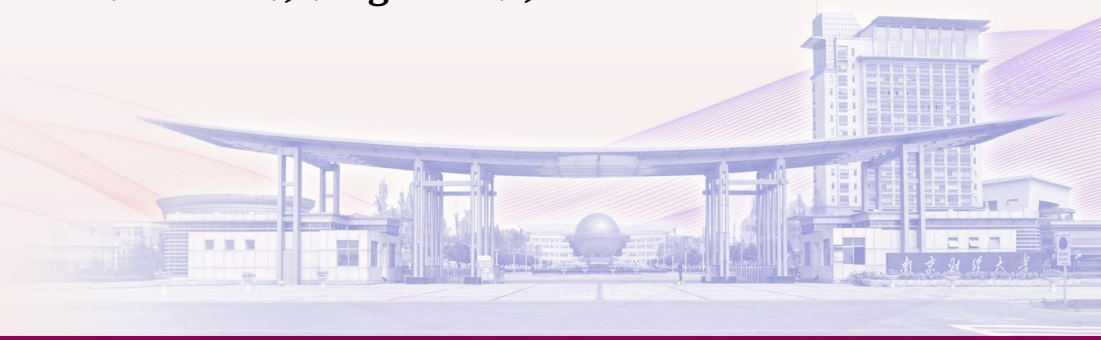
- 零-均值规范化 Python实现：



数据预处理

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

def demo():
    mu, sigma , num_bins = 0, 1, 50
    x = mu + sigma * np.random.randn(1000000)
    # 正态分布的数据
    n, bins, patches = plt.hist(x, num_bins, normed=True, facecolor = 'blue', alpha = 0.5)
    # 拟合曲线
    y = mlab.normpdf(bins, mu, sigma)
    plt.plot(bins, y, 'r--')
    plt.xlabel('Expectation', fontproperties='Times New Roman')
    plt.ylabel('Probability', fontproperties='Times New Roman')
    plt.title('histogram of normal distribution: $\mu = 0$, $\sigma=1$',
    fontproperties='Times New Roman')
    plt.subplots_adjust(left = 0.15)
    plt.show()
demo()
```



数据预处理

(1)数据清理

数据缺失问题:
(a)忽略元组或属性：当缺少类标号或元组有多个属性同时缺少值时可忽略元组；
(b)使用一个全局常量填充缺失值，如NULL;
(c)使用属性的中心度量（如均值或中位数）填充缺失值;
(d)使用最可能的值填充缺失值，可用逻辑回归、贝叶斯、决策树等模型来推理归纳确定。

异常值是数据集中偏离大部分数据的数据。从数值上表现为数据集中与平均值的偏差超过两倍标准差的数据。
异常值问题:
(a) 直接删除
(b) 按照缺失值处理

NULL (数据库)=None (python列表)=NaN (pandas)

数据预处理

数据预处理：
(2)数据集成：
把多个数据源的数据一起处理。

表 2-3. 原始数据

Id	MSZonging	LotArea	YearBuilt	MSSubClass
1	A	8450	1950	20
2	C	NaN	1947	20
3	RH	11250	1955	30
4	C	9550	1952	NaN

表 2-4. 原始数据

Id	RoofStyle	PoolArea	PoolQC	GarageArea
1	Flat	950	NaN	NaN
2	Flat	1000	Gd	NaN
3	Hip	1800	Ex	NaN
4	Shed	0	NaN	NaN

MSZonging：房子所在区域的类型，A 代表“农业”，C 代表“商业”，I 代表
“工业”，RH 代表“住宅密集区”，RL 代表“住宅稀疏区”。
LotArea：宅基地面积，单位为平房英尺。
YearBuilt：初始建造年份。
MSSubClass：住宅风格，“20”代表“一层，全新风格”，“30”代表“一层，
老式风格”，“40”表示“一层，有阁楼”，“50”表示“二层，新式风格”

RoofStyle：房顶风格，Flat 为平顶，Gable 为山墙型，Hip 为四坡屋顶，Shed 代表单坡屋顶。
PoolArea：游泳池面积，单位为平方英尺。
PoolQC：游泳池质量，Ex 表示“特别好”，Gd 表示“好”，TA 表示“普通”，
Fa 表示“可接受”，NA 表示“没有泳池”。
GarageArea：车库面积，单位平方英尺

数据预处理

(2)数据集成:把多个数据源的数据一起处理。

表 2-5. 预处理后的数据

Id	MSZonging	LotArea	YearBuilt	MSSubClass	RoofStyle	PoolArea	PoolQC
1	A	8450	1950	20	Flat	950	Gd
2	C	9750	1947	20	Flat	1000	Gd
3	RH	11250	1955	30	Hip	1800	Ex
4	C	9550	1952	20	Shed	0	NA

(1) LotArea,取了平均值。
(2) MSSubClas：按照多数原则
(3) PoolQC，按照PoolArea关联数据的相似性。

数据预处理

(3)数据变换

表 2-7　预处理后的数据

Id	LotArea	HouseAge	MSZonging _A	MSZonging _C	MSZonging _RH
1	0	68	1	0	0
2	0.464	71	0	1	0
3	1	63	0	0	1
4	0.393	66	0	1	0

LotArea= (Value-min)/(max-min）：
最小-最大规范化

对于“**MSZonging**”变量，假设其只有表格中的三种类型，即“**A**”，“**C**”和“**RH**”，则每一样本只会属于其中一种类型，当其属于某一类型时，该样本在该类型下标记为“**1**”，否则为“**0**”

对于字符串类型的变量，**A, C, RH**。

数据预处理

(4)数据归约：在保证数据完整性的基础上，降低数据规模。降低数据维度(Reduce Dimensionality)把3维的数据将为2维的数据，或者把2维的数据降低为1维的数据。
比如Principal Component Analysis (PCA),在机器学习领域用于降低维度的方法。其目的可以被简单
也

PCA降维

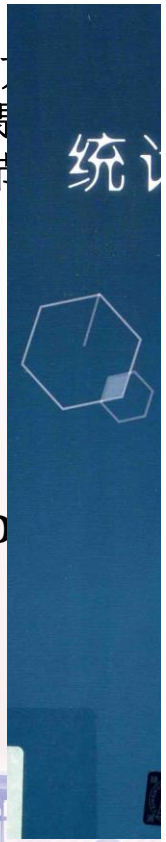
量x,通过一个
出所对应的原
特征脸（Eige

Turk M, Pentland A. Eigenfaces for recognition[J]. Journal of Cognitive Neuroscience, 1991, 3(1): 71-86.

数据预处理

(5)数据归约之降维技术：降维就是一种对高维度特征数据预处理方
保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提
际的生产和应用中，降维在一定的信息损失范围内，可以为我们节
成为应用非常广泛的数据预处理方法。

降维具有如下一些优点：
1）使得数据集更易使用。
2）降低算法的计算开销。
3）去除噪声。
4）使得结果容易理解。降维的算法有很多，比如奇异值分解(SVD
析(FA)、独立成分分析(ICA)。

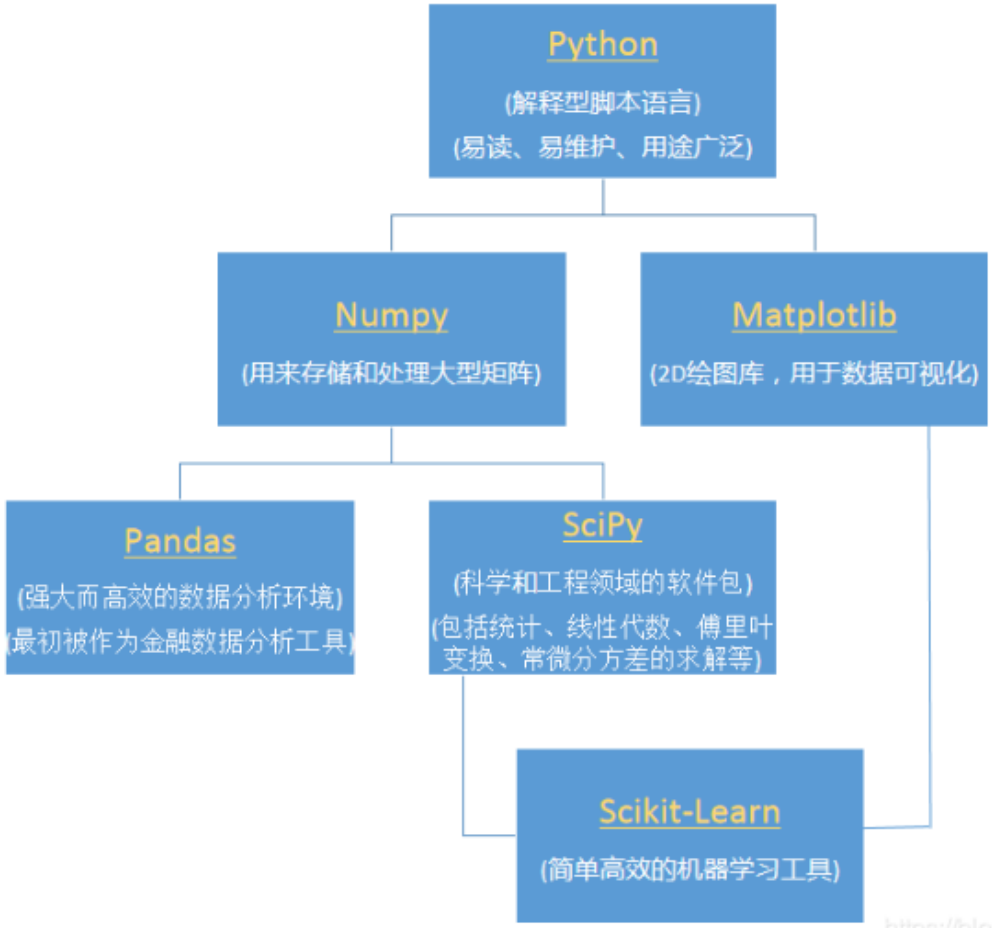


调用Python库:
from sklearn.decomposition import PCA

统计学习与方法（第二版），李航著.

数据预处理

■ **Python、Numpy、Pandas、SciPy、Scikit-Learn、Matplotlib**



安装:

pip install matplotlib
pip install numpy
pip install pandas
pip install scipy
pip install sklearn

附:

sklearn是scikit-learn的简称，是一个基于Python的第三方模块。sklearn库集成了一些常用的机器学习方法，在进行机器学习任务时，并不需要实现算法，只需要简单的调用sklearn库中提供的模块就能完成大多数的机器学习任务。sklearn库是在Numpy、Scipy和matplotlib的基础上开发而成的，因此在介绍sklearn的安装前，需要先安装这些依赖库。

数据预处理

■ **Python中标准的数据类型：**

1、字符串

```
str='this is string';
print str;
```

2、布尔类型

```
bool=False;
print bool;
bool=True;
print bool;
```

3、整数

```
int=20;
print int;
```

4、浮点数

```
float=2.3;
print float;
```

数据预处理

■ **Python中标准的数据类型：**

5、数字:包括整数、浮点数。

str(x) #将对象x转换为字符串

abs(x) 返回数字的绝对值，如abs(-10) 返回 10

exp(x) 返回e的x次幂(ex),如math.exp(1) 返回2.718281828459045 fabs(x) 返回数字的绝对值，如math.fabs(-10) 返回10.0

6、列表

list=['physics', 'chemistry', 1997, 2000];

nums=[1, 3, 5, 7, 8, 13, 20];

print "nums[0]:", nums[0]

"

#

p

"

d

p

"

```
list.append(obj) 在列表末尾添加新的对象
list.count(obj) 统计某个元素在列表中出现的次数
list.extend(seq) 在列表末尾一次性追加另一个序列中的多个值(用新列表扩展原来的列表)
list.index(obj) 从列表中找出某个值第一个匹配项的索引位置，索引从0开始
list.insert(index, obj) 将对象插入列表
list.pop(obj=list[-1]) 移除列表中的一个元素(默认最后一个元素)，并且返回该元素的值
list.remove(obj) 移除列表中某个值的第一个匹配项
list.reverse() 反向列表中元素，倒转
list.sort([func]) 对原列表进行排序
```

数据预处理

■ **Python中标准的数据类型：**

7、元组

Python的元组与列表类似，不同之处在于元组的元素不能修改；元组使用小括号()，列表使用方括号[]；元组创建很简单，只需要在括号中添加元素，并使用逗号(,)隔开即可，例如：

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
print "tup1[1:5]: ", tup1[1:3]
''' tup1[1:5]: ('chemistry', 1997) '''
```

8、字典

字典(dictionary)是除列表之外python中最灵活的内置数据结构类型。列表是有序的对象结合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。字典由键和对应的值组成。字典也被称作关联数组或哈希表。基本语法如下：

```
dict = {'name':'Zara','age': 7,'class':'First'};
print ( dict['name'])
" Zara"
```

数据预处理

■ **Python中标准的数据类型：**

9、日期

获取当前时间，例如：

import time, datetime;

```
import time, datetime;
today = datetime.date.today()
print (today)
```

```
2021-04-12
```

```
import time, datetime;
today_zero_time = datetime.datetime.strftime(today, '%Y-%m-%d %H:%M:%S')
print (today_zero_time)
```

```
2021-04-12 00:00:00
```

数据预处理

- **Python读取文件：**
  1.读取txt文

python常用的读取文件函数有三种read()、readline()、readlines()

✓ **read()　一次性读全部内容**
一次性读取文本中全部的内容，以字符串的形式返回结果

✓ **readline()　读取第一行内容**
只读取文本第一行的内容，以字符串的形式返回结果

✓ **readlines()　列表**
读取文本所有内容，并且以数列的格式返回结果，一般配合**for in**使用

数据预处理

■ Python读取文件：

```
with open("test.txt", "r") as f:  # 打开文件
    data = f.read()  # 读取文件
    print(data)
```

```
C:\Users\Administrator\PycharmProjects\untit
flora
zhang
我
你
他
```


readline()　读取第一行内容

只读取文本第一行的内容，以字符串的形式返回结果

```
with open("test.txt", "r") as f:
    data = f.readline()
    print(data)
```

```
C:\Users\Administrator\PycharmProjects'
flora
```

readlines()　列表

读取文本所有内容，并且以数列的格式返回结果，一般配合for in使用

```
with open("test.txt", "r") as f:
    data = f.readlines()
    print(data)
```

```
C:\Users\Administrator\PycharmProjects\untitled\venv\Scrip
['flora\n', 'zhang\n', '我\n', '你\n', '他']
```

readlines会读到换行符，可用如下方法去除：

```
with open("test.txt", "r") as f:
    for line in f.readlines():
        line = line.strip('\n')  #去掉列表中每一个元素的换行符
        print(line)
```

```
C:\Users\Administrator\PycharmProjects\untitled\venv\Scripts\py
flora
zhang
我
你
他
```

数据预处理

Python实践

Python读取文件：

2.读取csv

使用csv函数包，安装 pip install csv

使用方法：

```
import csv
def fileload(filename = ‘待读取.csv’):
    csvfile = open(filename, ‘r’,encoding = 'utf-8')
    data = csv.reader(csvfile)
    dataset = []
    for line in data:
        dataset.append(line)
    csvfile.close()
    return dataset
```

Python读取文件：

3.读取xlsx

使用xlrd函数包，安装： pip install xlrd

使用方法：

```
import xlrd
def fileload(filename = '待读取.xlsx'):
    dataset = []
    workbook = xlrd.open_workbook(filename)
    table = workbook.sheets()[0]
    for row in range(table.nrows):
        dataset.append(table.row_values(row))
    return dataset
```

数据预处理

Python实践

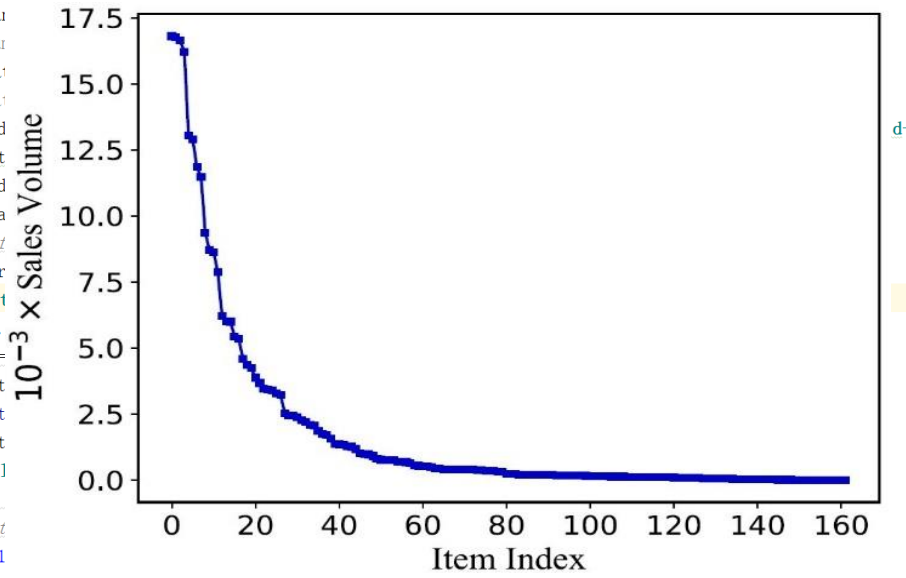
- **Python读取文件：4.pandas读取文件并计算绘图**

安装：

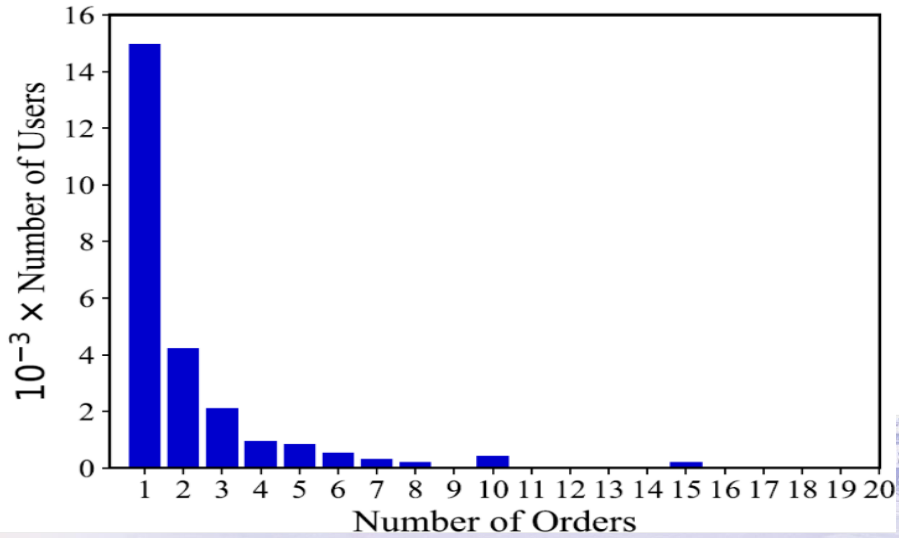
pip install numpy
pip install pandas
pip install matplotlib

A	B	C	D	E
	UserID	ItemID	Num	Price
0	0	0	1	368
1	1	0	12	368
2	1	1	12	368
3	1	2	12	398
4	2	0	12	368
5	2	1	12	368
6	2	2	24	398
7	3	0	12	368
8	3	1	12	368
9	3	2	12	398
10	4	0	12	368
11	4	1	12	368
12	4	2	12	398
13	5	0	12	368
14	5	1	12	368
15	5	2	12	398

```
import pa
import nu
import ma
import ma
datas = pd
ItemIDlist
Number = d
df = pd.Da
#print('dt
data=df.gr
print('dat
data=data.
ItemIndex=
SumNum=dat
Sales=list
Sales.sort
print('Sal
#for i in
    #print
Sales=[i/1
ax=plt.gc
#ax为两条
ax.spines
ax.spines
ax.spines
ax.spines
plt.plot(
plt.xlabe
plt.ylabe
#plt.xlab
#labels=[
plt.xtick
plt.ytick
plt.tight
plt.savef
```




```
d-itemid.xlsx')
```



数据预处理

■ 写入数据

1.写入txt

```
with open(“test.txt”,“w”) as f:
    f.write(“这是个测试！”) # 自带文件关闭功能
```

2.写入CSV

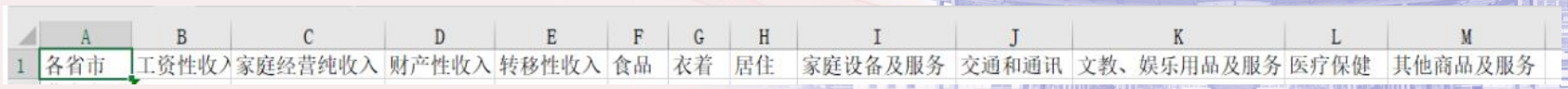
```
import csv
f = open('文件名.csv','w',encoding='utf-8')
csv_writer = csv.writer(f)
csv_writer.writerow(["姓名","年龄","性别"])
csv_writer.writerow(["l",'18','男'])
csv_writer.writerow(["c",'20','男'])
csv_writer.writerow(["w",'22','女'])
f.close()
```

数据预处理

■ 写入数据

3.写入xlsx

```
import openpyxl
workbook=openpyxl.Workbook() # 创建一个Workbook对象，相当于创建了一个Excel文件
worksheet = workbook.active #获取当前活跃的worksheet,默认就是第一个worksheet
worksheet.title="mysheet“
Project=['各省市', '工资性收入', '家庭经营纯收入', '财产性收入', '转移性收入', '食品', '衣着', '居住', '家庭设备及服务', '交通和通讯', '文教、娱乐用品及服务', '医疗保健', '其他商品及服务']
for i in range(len(Project)): #写入第一行数据，行号和列号都从1开始计数
    worksheet.cell(1, i+1,Project[i])
workbook.save(filename='DataSource\\myfile.xlsx')
```



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	各省市	工资性收入	家庭经营纯收入	财产性收入	转移性收入	食品	衣着	居住	家庭设备及服务	交通和通讯	文教、娱乐用品及服务	医疗保健	其他商品及服务

数据预处理

■ Python连接数据库

python连接微软的sql server数据库用的第三方模块叫做pymssql（document：http://www.pymssql.org/en/stable/index.html）。

```
import pymssql
conn = pymssql.connect(host='127.0.0.1',
        user='sa',
        password='123',
        database='SQLTest',
        charset='utf8')
#查看连接是否成功
cursor = conn.cursor()
sql = 'select * from student'
cursor.execute(sql)
#用一个rs变量获取数据
rs = cursor.fetchall()
print(rs)
```

打开IDLE，新建python程序:

运行结果:

```
PymssqlTest.py =======
[('0001', '张三    ', 18, '男', '文学院          '),
('0002', '李四    ', 19, '男', '理学院          '),
('0003', '王五    ', 20, '男', '文学院          '),
('0004', '赵六    ', 21, '女', '理学院          '),
('0005', '孙七    ', 18, '女', '文学院          '),
('0006', '周八    ', 19, '女', '理学院          '),
('0007', '吴九    ', 20, '男', '文学院          '),
('0008', '郑十    ', 21, '女', '理学院          ')]
>>>
```

数据预处理

Python连接数据库

在python中，在操作完“增删改”之后，还需要执行commit()才能真正提交代码执行，如果出意外的话就执行rollback()回滚到之前的状态，相当于之前的操作都白做了，这样也保护了数据库。

```
try:
  conn = pymssql.connect(host='127.0.0.1',
             user='sa',
             password='123',
             database='SQLTest',
             charset='utf8')
  cursor = conn.cursor()
  sql = 'insert into student values('0001', '张三', 18, '男', '文学院')'
  cursor.execute(sql)
  conn.commit()
except Exception as ex:
  conn.rollback()
  raise ex
finally:
  conn.close()
```

大家可以试一试将conn.commit()删去，然后看看数据库是否有变化。

数据预处理

■ Python中的列表（list）

列表是 Python 中最基本的数据结构。序列中的每个值都有对应的位置值，称之为索引，第一个索引是 0，第二个索引是 1，依此类推。

1. 列表的创建和访问

```
list = ['red', 'green', 'blue', 'yellow', 'white', 'black']
print( list[0] )
print( list[1] )
print( list[2] )
print( list[-1] )
print( list[-2] )
print( list[-3] )
print( list[0:3] )
```

```
red
green
blue
black
white
yellow
['red', 'green', 'blue']
```

数据预处理

■ Python中的列表（list）

2.删除列表元素

```
list = ['red', 'green', 'blue', 'yellow', 'white', 'black']
del list[0]
print(list)
```

```
['green', 'blue', 'yellow', 'white', 'black']
```

3.更新、添加列表元素

```
list = ['Google', 'Runoob', 'Amazon']
print ("第三个元素为 : ", list[2])
list[2] = 'Tik Tok'
print ("更新后的第三个元素为 : ", list[2])
list.append('Baidu')
print ("添加元素后的列表 : ", list)
```

```
第三个元素为 :  Amazon
更新后的第三个元素为 :  Tik Tok
添加元素后的列表 :  ['Google', 'Runoob', 'Tik Tok', 'Baidu']
```

数据预处理

- **Python中的列表（list）**

4.应用案例（点击流数据）
时序数据可以用list进行存取

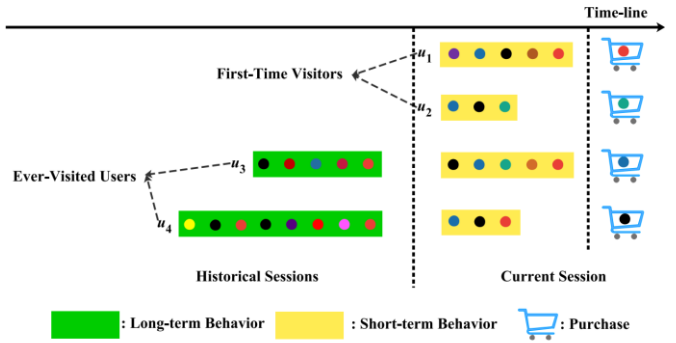


Fig. 2. Long-term and short-term behaviors with a cut time line.

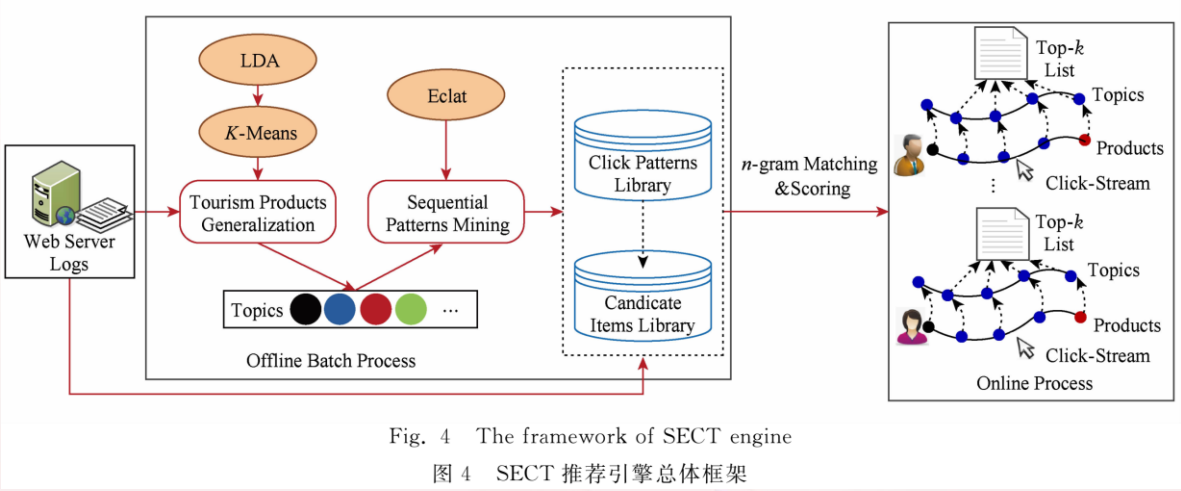


Fig. 4 The framework of SECT engine

图 4 SECT 推荐引擎总体框架

1. Guixiang Zhu, Youquan Wang, Jie Cao, Zhan Bu, Shuxin Yang，Weichao Liang, Jingting Liu. Neural attentive travel package recommendation via exploiting long-term and short-term behaviors[J]. ***Knowledge-Based Systems***, 2021，211：106511.

2. Guixiang Zhu, Jie Cao, Changsheng Li, Zhiang Wu. A recommendation engine for travel products based on topic sequential patterns[J]. ***Multimedia Tools and Applications*, 2017, 76(16): 17595-17612.**

Python中的中字典（dict）

字典(dictionary)是除列表之外python中最灵活的内置数据结构类型。列表是有序的对象结合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。字典由键和对应的值组成。字典也被称作关联数组或哈希表。

1.字典的格式

字典的每个键值对(key→value)都是用冒号：分割，每个键值对之间用逗号，分割，整个字典包括在花括号{}中，格式如下：

dict = {key1:value1, key2:value2}

2.字典的创建

In [3]:
```
dict = {
'name' : 'guodong',
'age' : '21',
'sex' : 'M',
'weight' : '140',
'height' : '180'}
print (dict)
```

```
{'weight': '140', 'age': '21', 'height': '180', 'name': 'guodong', 'sex': 'M'}
```

数据预处理

■ Python中的中字典（dict）

3.访问字段里的值

取值，根据索引(key)取值，取值时，索引用中括号括起来。

4.增加/删除

增加：索引使用中括号括起来；删除：索引使用小括号括起来。

```
print (dict['name'])
```

```
guodong
```

```
dict['id'] = '05'
print (dict)
```

```
{'weight': '140', 'height': '180', 'age': '21', 'id': '05', 'sex': 'M', 'name': 'guodong'}
```

```
dict.pop('id')
print (dict)
```

```
{'weight': '140', 'height': '180', 'age': '21', 'sex': 'M', 'name': 'guodong'}
```

数据预处理

■ Python中的中字典（dict）

5.统计键值对的数量

取值，根据索引(key)取值，取值时，索引用中括号括起来。

```
print(len(dict))
```

```
5
```

6.字典的遍历

```
for key,value in dict.items():
    print ("key:" + key,"value:" + value)
```

```
key:weight value:140
key:height value:180
key:age value:21
key:sex value:M
key:name value:guodong
```

数据预处理

■ Python中的中字典（dict）

6.字典使用案例

目的：绘制用户-购买不同产品数量的分布图

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
sheet = pd.read_excel('F:/Onedrive/userid-itemid.xlsx')
print (sheet)
Users= sheet['UserID'].values
Users=list(Users)
Distinct_Users = set(list(Users))#去重
dict={}
for user in Distinct_Users:
    dnum=Users.count(user)
    dict[user]=num
values = list(dict.values())  #{user: number of distinct products}
values.sort(reverse = True) #倒序
x=[i for i in range(0,len(values))] #x轴：user index
ax=plt.gca();#获得坐标轴的句柄
```

	Unnamed: 0	UserID	ItemID	Num	Price
0	0	0	0	1	368
1	1	1	0	12	368
2	2	1	1	12	368
3	3	1	2	12	398
4	4	2	0	12	368
...	...	...	...	...	...
43575	43575	31352	106	6	84
43576	43576	31353	106	6	84
43577	43577	31354	106	8	84
43578	43578	31355	106	8	84
43579	43579	31356	161	2	11

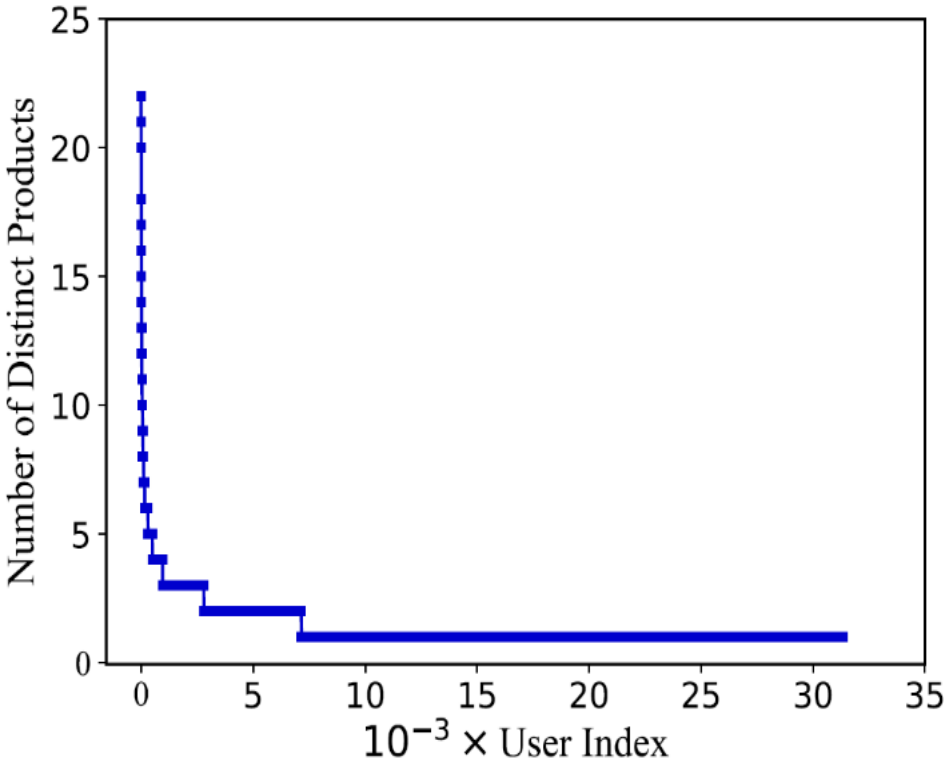
[43580 rows x 5 columns]

数据预处理

■ Python中的中字典（dict）

6.字典使用案例

```
ax.spines['top'].set_linev
ax.spines['bottom'].set_
ax.spines['left'].set_linev
ax.spines['right'].set_lin
plt.plot(x, values, 'medi
plt.ylabel("Number of I
plt.xlabel(r'$10^{-3}\tin
x=[0,5000,10000,15000,
label=[0,5,10,15,20,25,3
plt.xticks(x,label,fontpr
y=[0,5,10,15,20,25]
label=[0,5,10,15,20,25]
plt.yticks(y,label,fontpr
plt.tight_layout()
plt.savefig("./Figure1.ep
plt.show()
```




```
e'
n', fontsize=18)
 fontsize=18)
```

数据预处理

Python中的DataFrame

DataFrame是Pandas库中处理表的数据结构，可看作是python中的类似数据库的操作，是Python数据挖掘中最常用的工具。下面介绍DataFrame的一些常用方法。

1. DataFrame的创建

创建一个空的dataframe :

```
df=pd.DataFrame(columns={"a":"","b":"","c":""},index=[0])
```

Out:

```
   a   c   b
0 NaN NaN NaN
```

用list的数据创建dataframe：

```
a = [['2', '1.2', '4.2'], ['0', '10', '0.3'], ['1', '5', '0']]
df = pd.DataFrame(a, columns=['one', 'two', 'three'])
print df
```

Out:

```
 one  two three
0   2  1.2   4.2
1   0   10   0.3
2   1    5     0
```

数据预处理

■ Python中的DataFrame

用dict的数据创建DataFrame:

```
df=pd.DataFrame({'key':['a','b','c'],'data1':[1,2,3],'data2':[4,5,6]})
print(df)
```

```
   data1  data2 key
0      1      4   a
1      2      5   b
2      3      6   c
```

2. DataFrame遍历：

按行遍历iterrows():

```
for idx,item in df.iterrows():
    print(idx)
    print(item)
```

```
0
data1    1
data2    4
key      a
Name: 0, dtype: object
1
data1    2
data2    5
key      b
Name: 1, dtype: object
2
data1    3
data2    6
key      c
Name: 2, dtype: object
```

按列遍历iteritems():

```
for idx,item in df.iteritems():
    print(idx)
    print(item)
```

```
data1
0    1
1    2
2    3
Name: data1, dtype: int64
data2
0    4
1    5
2    6
Name: data2, dtype: int64
key
0    a
1    b
2    c
Name: key, dtype: object
```

数据预处理

■ Python中的DataFrame

3. 输出为excel或者csv格式、Numpy的矩阵格式、dict格式：

```
df.to_excel('path/filename.xls')
df.to_csv('path/filename.csv’ )
matrix = df.ax_matrix()
dict = df.to_dict(orient="dict")
```

```
dataframe:        ItemID  Num
0                      0    1
1                      0   12
2                      1   12
3                      2   12
4                      0   12
...                  ...  ...
43575                106    6
43576                106    6
43577                106    8
43578                106    8
43579                161    2

[43580 rows x 2 columns]
```

```
data            Num
ItemID
0             11849
1              6017
2             16207
3               558
4               203
...             ...
157            2271
158            6217
159            2373
160              29
161               2

[162 rows x 1 columns]
Sales: [16806, 16753, 16637, 16207, 13052, 12900, 11849, 11483, 9367, 8720, 8617, 7882, 6217,
```

mber})