



华中师范大学

# 自然语言处理

## 第二章 语料清洗

主讲教师：曾江峰

华中师范大学 信息管理学院

[jfzeng@ccnu.edu.cn](mailto:jfzeng@ccnu.edu.cn)

## 语料清洗

- ❖ 语料清洗是自然语言处理的第一步，对最终结果起到决定性的作用。
- ❖ 本章重点讲解语料的清洗策略，填充缺失值、消除异常值和平滑噪声数据等清洗方法。
- ❖ 介绍数据替换、数据映射、数据合并和数据补充等数据转换功能， `missingno` 库用于数据分析前的数据检查，查看数据集完整性。词云用于可视化地显示数据相关信息。



# 语料清洗

- 1 认识语料清洗
- 2 清洗策略
- 3 缺失值清洗
- 4 异常值清洗
- 5 重复值清洗
- 6 数据转换
- 7 Missingno 库
- 8 词云

# 1. 认识语料清洗

- 由于原始文本含有“脏”数据，无法直接用来训练模型，会严重干扰数据分析的结果，因此需要进行清洗，保留有用的数据。
- “脏”数据是指缺失值、异常值、离群点等数据。
- 语料清洗用于纠正语料数据中不一致的内容，通常会占据数据分析50%到80%的工作量。



# 1. 认识语料清洗

经过数据清洗的数据，应该满足以下特点。

(1) 所有值都是数字。

机器学习算法的所有数据都是数字，需要将非数字值（在类别或文本列中的内容）需要替换为数字标识符。

(2) 标识并清除无效值记录。

无效值无法反映实际问题。

(3) 识别并消除无关类别。

所有记录都需要使用一致类别。



# 1. 认识语料清洗

## 【例1】语料清洗

# 例1: 语料清洗

`str="???程序员是从事程序开发、维护的专业人员。%%# #aba一般将程序员分为程序设计人员和程序编码人员，<body>但两者的78界限并不非常清楚，软件从业人员分为初级程序员、高级程序员、系统分析员和项目经理四大类。"`

```
import re    # 引入正则表达式
pattern = re.compile(r'^\u4e00-\u9fa5')    # 过滤掉数字、符号、字符等特殊字符
chinese_txt = re.sub(pattern, '', str)
print(chinese_txt)
```

程序员是从事程序开发维护的专业人员一般将程序员分为程序设计人员和程序编码人员但两者的界限并不非常清楚软件从业人员分为初级程序员高级程序员系统分析员和项目经理四大类

# `str`文本中不仅包含中文字符，还包括数字、标签、英文字符、标点等非常规字符，  
# 这些都是无意义的信息，与文本内容所要表达的主题没有任何关联，通过正则表达式进行清洗



## 2. 清洗策略

### (1) 一致性检查

一致性检查是根据每个变量的合理取值范围和相互关系，检查数据是否合乎要求，发现超出正常范围的数据，例如，体重出现了负数、年龄超出正常值范围。SPSS、Excel 等软件能够自动识别超出范围的变量值。



## 2. 清洗策略

### (2) 格式内容检查

多源的数据往往在格式和内容上存在很多问题，例如，时间、日期、数值、全半角等显示格式不一致等。例如，性别字段，某来源为“男”和“女”，某来源为“0”和“1”，需要进行格式内容检查。





## 2. 清洗策略

### (3) 逻辑检查

通过逻辑推理发现不合理或者相互矛盾的问题数据。例如，“身份证号”和“年龄”两个字段，可以进行相互验证。



### 3. 缺失值清洗

缺失值是指记录的缺失和记录中某个字段信息的缺失，一般以空白、NaN 或其他占位符进行编码。缺失值处理一般采用如下方法：删除法和数据填充。

(1) 删除法：如果某个属性的缺失值过多，可以直接删除整个属性。

(2) 数据填充：对属性缺失的样本采用其他值，如前后值、中位数、均值进行替代。



### 3. 缺失值清洗

填充方法	方法描述
平均值	根据属性值类型，用该属性取值的平均值填充
中位数	根据属性值类型，用该属性取值的中位数填充
固定值	将缺失的属性值用一个常量替换
最近值	用最接近缺失值的属性值填充
最大、最小值	根据属性值类型，用最大、最小值进行填充

表3-1 常用填充方法



# 3. 缺失值清洗

## 3.1 Pandas处理

Pandas使用浮点值NaN表示缺失值，缺失值的处理函数有df.fillna()和df.dropna()两个函数。

函数名	Pandas缺失值处理函数
df.fillna(num)	用实数num填充缺失值
df.dropna()	删除DataFrame数据中的缺失数据

表3-2 pandas缺失值处理函数



# 3. 缺失值清洗

## 3.1 Pandas处理：df.fillna()用实数填充缺失值

DataFrame.fillna(value=None, method=None, axis=None, inplace=None, limit=None)

函数名	Pandas缺失值处理函数
value	用于填充缺失值的标量值或字典对象
method	插值方式
axis	数据删除维度：取值0为行；取值1为列
inplace	修改调用者对象而不产生副本
limit	可以连续填充的最大数量

表3-3 df.fillna()参数说明



## 3. 缺失值清洗

### 3.1 Pandas处理: df.fillna() 用实数填充缺失值

#### 【例2】 df.fillna() 举例

```
# 例2:df.fillna()用实值填充缺失值
from numpy import nan as NaN
import pandas as pd
df1= pd.DataFrame ([[1,2,3],[NaN,NaN,2],[NaN,NaN,NaN],[8,8,NaN]])
print ("df1:\n{}\n ".format(df1))
df2= df1.fillna(100)
print ("df2:\n{}\n ".format(df2))
```

df1:

	0	1	2
0	1.0	2.0	3.0
1	NaN	NaN	2.0
2	NaN	NaN	NaN
3	8.0	8.0	NaN

df2:

	0	1	2
0	1.0	2.0	3.0
1	100.0	100.0	2.0
2	100.0	100.0	100.0
3	8.0	8.0	100.0



# 3. 缺失值清洗

## 3.1 Pandas处理：df.dropna() 删除缺失数据所在的行或列

DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

函数名	Pandas缺失值处理函数
axis	数据删除维度：取值0为行；取值1为列
how	{'any','all'}, 默认为'any'，表示删除带有nan的行；'all'表示删除全为nan的行
thresh	int，表示保留至少int个非nan行
subset	部分标签中删除某些列
inplace	bool，是否修改源文件

表3-4 df.dropna()参数说明

[《自然语言处理入门》](#)



## 3. 缺失值清洗

### 3.1 Pandas处理: df.dropna() 用实数填充缺失值

#### 【例3】df.dropna() 举例

```
# 例3: df.dropna()删除DataFrame 缺失数据所在行或列
from numpy import nan as NaN
import pandas as pd
df1= pd.DataFrame ([[1,2,3],[NaN,NaN,2],[NaN,NaN,NaN],[8,8,NaN]])
print ("df1:\n{}\n ".format(df1))
df2= df1.dropna()
print ("df2:\n{}\n ".format(df2))
```

df1:

	0	1	2
0	1.0	2.0	3.0
1	NaN	NaN	2.0
2	NaN	NaN	NaN
3	8.0	8.0	NaN

df2:

	0	1	2
0	1.0	2.0	3.0





# 3. 缺失值清洗

## 3.2 Sklearn处理

Sklearn 中 `Imputer` 类 或 `SimpleImputer` 类用于处理缺失值。其中，`Imputer` 类在 `preprocessing` 模块中，而 `SimpleImputer` 类在 `sklearn.impute` 模块中。



# 3. 缺失值清洗

## 3.2 Sklearn处理

- Imputer 具体语法如下：

```
from sklearn.preprocessing import Imputer
```

```
imp = Imputer( missing_values = “NaN” , strategy = “mean” )
```

- SimpleImputer 具体语法如下：

```
from sklearn.impute import SimpleImputer
```

```
import numpy as np
```

```
imp = SimpleImputer( missing_values = np.nan , strategy = “mean” )
```

- 参数含义：missing\_values = np.nan：缺失值是 nan

strategy = “mean”：[用平均数、中位数等插值方法的数据](#)



# 3. 缺失值清洗

## 3.1 Sklearn处理

### 【例4】Sklearn中Imputer类或SimpleImputer类

```
# 例4: Imputer或SimpleImputer完成缺值填充
import pandas as pd
import numpy as np
# from sklearn.preprocessing import Imputer
from sklearn.impute import SimpleImputer
```

```
df = pd.DataFrame([['XXL',8,'black','class',22],
                  ['L',np.nan,'orange','class 1',17],
                  ['XL',10,'blue','class 2',19],
                  ['M',11,'green','class 3',np.nan],
                  ['M',7,'red','class 1',22]])
df.columns = ['size','price','color','class','boh']
print(df)
```

	size	price	color	class	boh
0	XXL	8.0	black	class	22.0
1	L	NaN	orange	class 1	17.0
2	XL	10.0	blue	class 2	19.0
3	M	11.0	green	class 3	NaN
4	M	7.0	red	class 1	22.0

```
# 1. 创建Imputer器
```

```
# imp = Imputer(missing-values = 'NaN',strategy = 'mean')
imp = SimpleImputer(missing_values = np.nan,strategy = 'mean')
```

```
# 2. 使用fit_transform()函数完成缺失值填充
```

```
df['price'] = imp.fit_transform(df[['price']])
```

```
print(df)
```

	size	price	color	class	boh
0	XXL	8.0	black	class	22.0
1	L	9.0	orange	class 1	17.0
2	XL	10.0	blue	class 2	19.0
3	M	11.0	green	class 3	NaN
4	M	7.0	red	class 1	22.0



## 4. 异常值清洗

“异常数据”又称为离群点，具有与其他数据的显著不同。

通常检测方法如下所示：

### (1) 基于邻近度的方法

通常可以在对象之间定义邻近性度量，异常对象是那些远离其他对象的对象。

### (2) 基于密度的方法

仅当一个点的局部密度显著低于它的大部分近邻时才将其分类为离群点。

### (3) 基于聚类的方法

聚类分析用于发现局部强相关的对象。

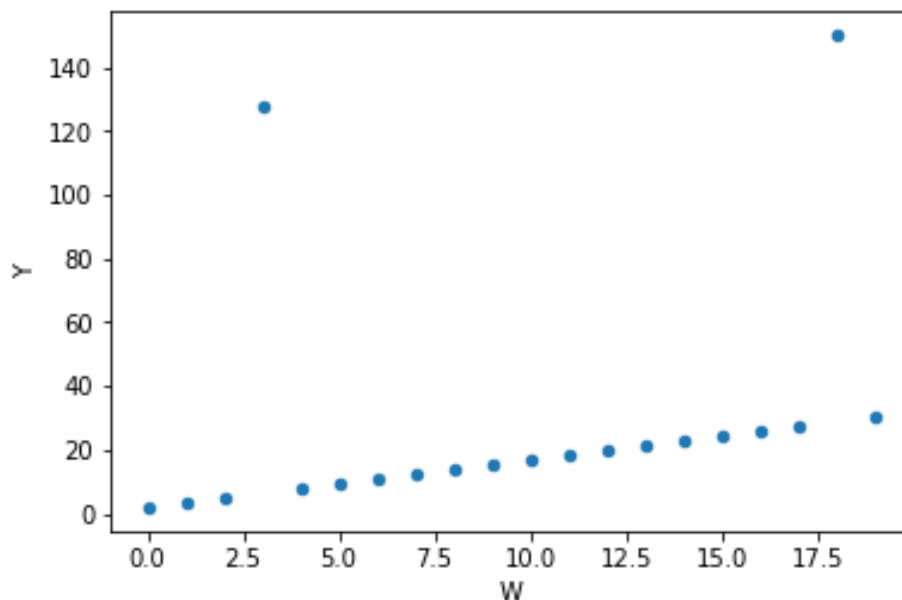
一般采用Z标准化得到的阈值作为判断标准，超过阈值则为异常。



# 4. 异常值清洗

## 4.1 散点图方法

散点图通过展示两组数据的位置关系，可以展示数据的分布和聚合情况，可以清晰直观地看出哪些值是离群点。



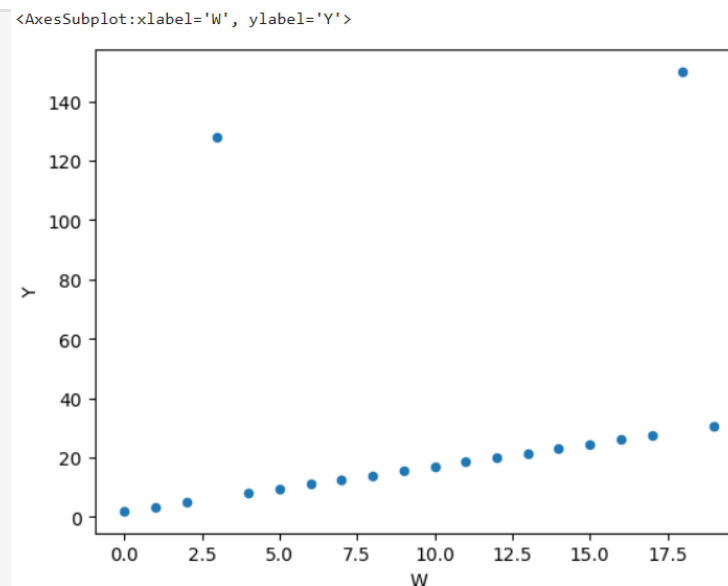
# 4. 异常值清洗

## 4.1 散点图方法

### 【例5】绘制散点图方法

```
# 例5: 绘制散点图
import numpy as np
import pandas as pd

wdf = pd.DataFrame(np.arange(20), columns = ['W'])
wdf['Y'] = wdf['W'] * 1.5 + 2
wdf.iloc[3,1] = 128
wdf.iloc[18,1] = 150
wdf.plot(kind = 'scatter', x = 'W', y = 'Y')
```



# 4. 异常值清洗

## 4.2 箱线图方法

箱线图又称箱形图或盒式图，不同于折线图、柱状图或饼图等传统图表只是数据大小、占比、趋势的呈现。箱线图包含统计学的均值、分位数、极值等统计量，用于分析不同类别数据平均水平差异，展示属性与中位数离散速度，并揭示数据间离散程度、异常值、分布差异等。箱线图是一种基于“五位数”摘要显示数据分布的标准化方法。

箱形图判断异常值的标准以四分位数和四分距离为基础，当数据在箱线图中超过上四分位1.5倍四分位距或下四分位1.5倍距离时，即小于 $Q_1 - 1.5IQR$ 或大于 $Q_3 + 1.5IQR$ 的值被认为是异常值。

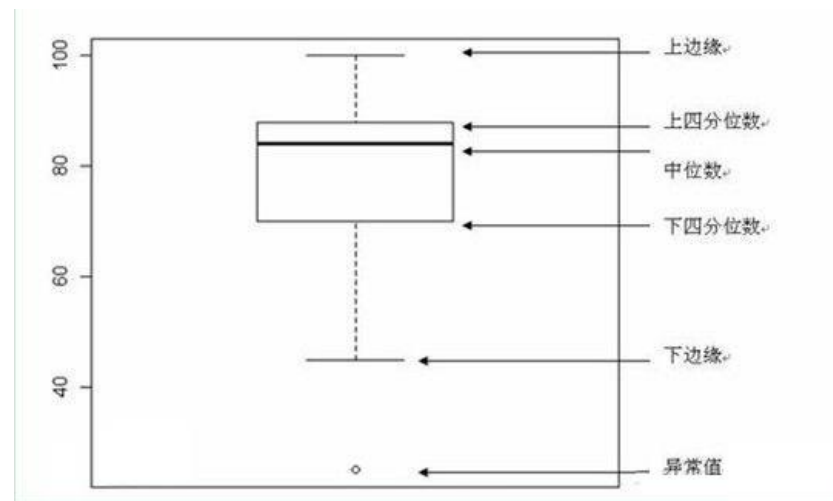


# 4. 异常值清洗

## 4.2 箱线图方法

“五位数”指箱型图的五个参数。

- (1) 下边缘 ( $Q_1$ ) 表示最小值。
- (2) 下四分位数 ( $Q_2$ ) 又称“第一四分位数”，由小到大排列后第25%的数字。
- (3) 中位数 ( $Q_3$ ) 又称“第二四分位数”，由小到大排列后第50%的数字。
- (4) 上四分位数 ( $Q_4$ ) 又称“第三四分位数”，由小到大排列后第75%的数字。
- (5) 上边缘 ( $Q_5$ ) 表示最大值。

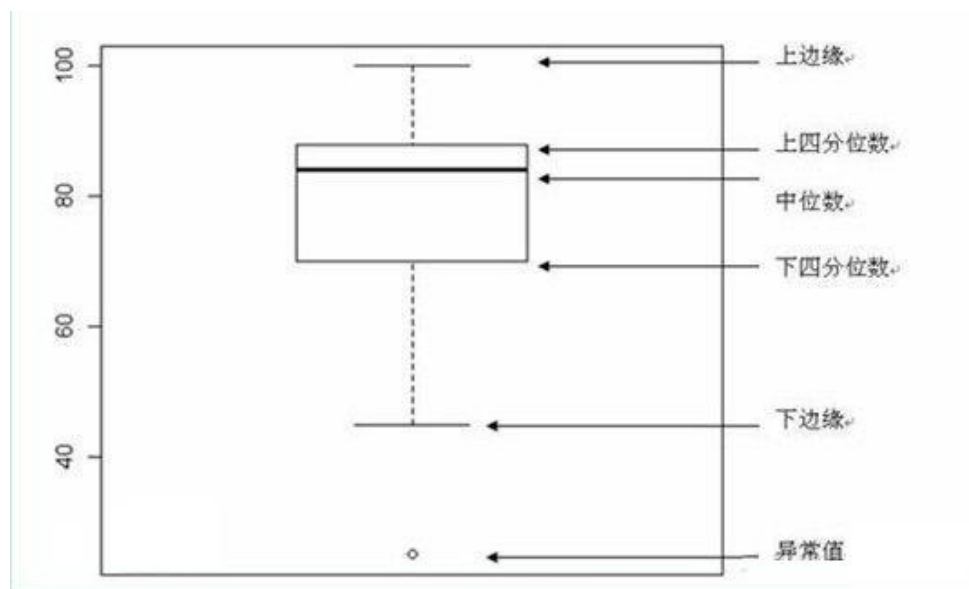




# 4. 异常值清洗

## 4.2 箱线图方法

箱形图判断异常值的标准以四分位数和四分距离为基础，当数据在箱线图中超过上四分位1.5倍四分位距或下四分位1.5倍距离时，即小于  $Q_1 - 1.5IQR$  或大于  $Q_3 + 1.5IQR$  的值被认为是异常值



# 4. 异常值清洗

## 4.2 箱线图方法

### 【例6】 绘制箱线图方法

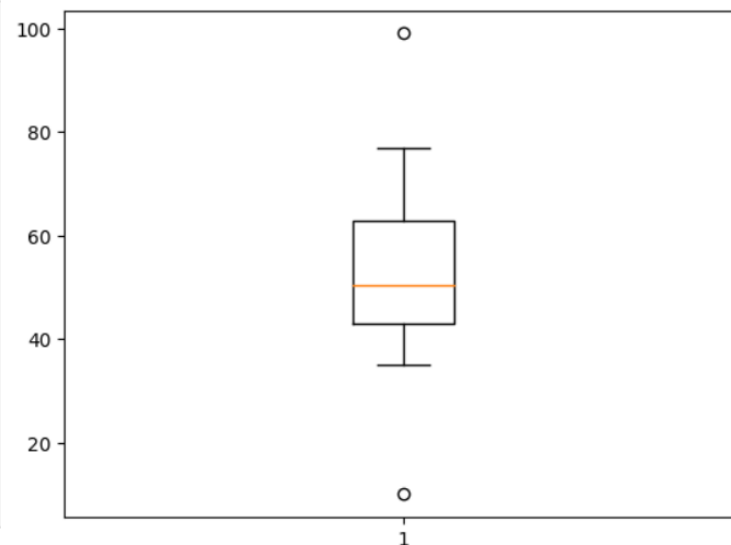
```
import numpy as np
import matplotlib.pyplot as plt

data = np.concatenate((np.random.randint(35,55,25),
                        np.random.randint(55,80,15)))

data[25] = 10
data[26] = 99
print(data)

plt.boxplot(data,showfliers = True)
plt.show()
```

```
[45 54 38 44 51 50 40 38 53 46 37 41 53 46 36 54 37 43 44 52 47 50 35 48
 43 10 99 76 65 73 77 68 76 70 62 61 65 68 59 57]
```



# 4. 异常值清洗

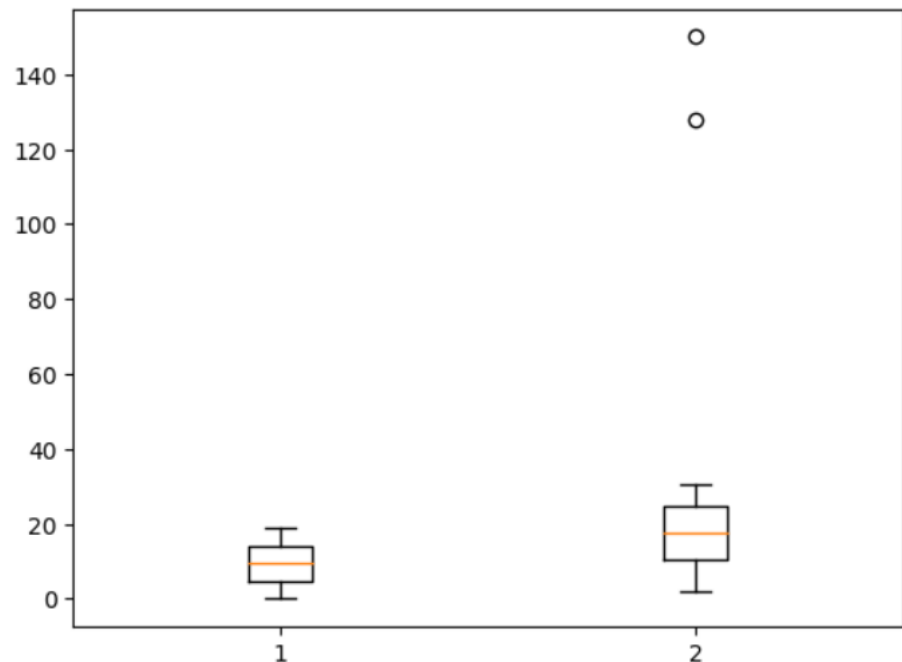
## 4.2 箱线图方法

### 【例6】绘制箱线图方法

```
# 例6: 绘制箱型图
import numpy as np
import pandas as pd

wdf = pd.DataFrame(np.arange(20), columns = ['W'])
wdf['Y'] = wdf['W'] * 1.5 + 2
wdf.iloc[3,1] = 128
wdf.iloc[18,1] = 150

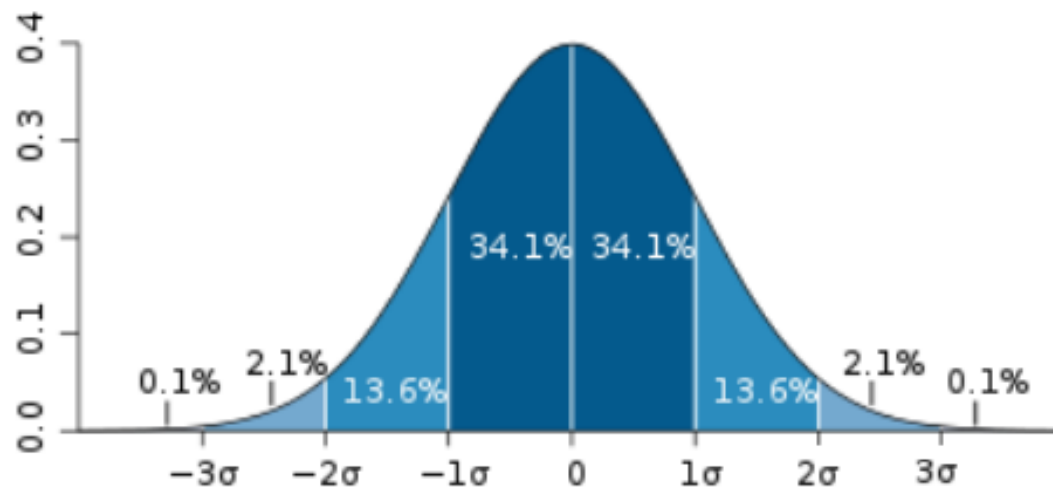
import matplotlib.pyplot as plt
plt.boxplot(wdf)
plt.show()
```



## 4. 异常值清洗

### 4.3 $3\sigma$ 法则

对于正态分布数据而言，数值分布在  $(\mu - \sigma, \mu + \sigma)$  中的概率为0.6827；在  $(\mu - 2\sigma, \mu + 2\sigma)$  中的概率为0.9545；在  $(\mu - 3\sigma, \mu + 3\sigma)$  中的概率为0.9973（ $\mu$ 为平均值， $\sigma$ 为标准差）。



# 4. 异常值清洗

## 4.3 $3\sigma$ 法则

计算步骤如下：

- (1) 首先判断数据大致上服从正态分布
- (2) 计算需要检验的数据列的平均值和标准差
- (3) 比较数据列的每个值与平均值的偏差是否超过3倍，如果超过3倍，则为异常值
- (4) 剔除异常值，得到规范的数据



# 4. 异常值清洗

## 4.3 $3\sigma$ 法则

### 【例7】 $3\sigma$ 法则

```
# 例7: 3sigma法则
import numpy as np
import pandas as pd
from scipy import stats

# 创建数据
data = [1222,87,77,92,68,75,77,80,78,123,3,23,32]
df = pd.DataFrame(data,columns = ['value'])

# 计算均值
u = df['value'].mean()
# 计算方差
std = df['value'].std()

print(stats.kstest(df, 'norm', (u, std)))
print('均值为: %.3f, 标准差为: %.3f'%(u, std))
print('-----')

# 识别异常值
error = df[np.abs(df['value'] - u) > 3 * std]

# 剔除异常值, 保留正常的数据
data_c = df[np.abs(df['value'] - u) <= 3 * std]
```

```
print('输出正常数据')
print(data_c)
print('输出异常数据')
print(error)
```

KstestResult(statistic=0.9995370512363594, pvalue=8.973421490365488e-44)

均值为: 156.692, 标准差为: 321.639

-----

输出正常数据

	value
1	87
2	77
3	92
4	68
5	75
6	77
7	80
8	78
9	123
10	3
11	23
12	32

输出异常数据

	value
0	1222

[《自然语言处理入门》](#)



## 4. 异常值清洗

异常值处理方法：

方法	方法描述
删除	直接删除含有异常值的记录
视为缺失值	利用缺失值处理方法
采用平均值修正	用前后两个观测值的平均值修正该异常值

表4-1 异常值处理方法



## 5. 重复值清洗

重复值会影响数据分析的准确性，消除重复值的基本思想是“排序和合并”，将数据进行排序后，比较邻近记录是否相似来检测记录是否重复。NumPy 和 Pandas 分别提供重复值的检测和清洗等。





# 5. 重复值清洗

## 5.1 NumPy处理

NumPy提供unique()函数返回数组元素的唯一值

### 【例8】使用NumPy的unique()函数去除重复值

```
# 例8: 使用Numpy的unique()函数去除重复值
import numpy as np
names = np.array(['红色', '蓝色', '红色', '白色', '白色', '红色', '绿色', '红色'])
print('原数组: ', names)
print('去重后的数组: ', np.unique(names))
```

原数组: ['红色' '蓝色' '红色' '白色' '白色' '红色' '绿色' '红色']

去重后的数组: ['白色' '红色' '绿色' '蓝色']



# 5. 重复值清洗

## 5.2 Pandas处理

在Pandas中，`uplicated()` 用于判断重复数据记录；  
`drop_duplicates()` 用于删除重复记录

`DataFrame(Series).drop_duplicates(self, subset=None, keep=' first' , inplace=False)`

参数	说明
subset	接收string或sequence，表示进行去重的列，默认全部列
keep	表示重复时保留第几个数据，first保留第一个，last保留最后一个
inplace	表示是否在原表上进行操作，默认为False

表5-1 drop\_duplicates()参数说明



# 5. 重复值清洗

## 5.2 Pandas处理

### 【例9】：df.duplicated() 举例

```
# 例9: df.duplicated()判断重复值并用df.drop_duplicates()删除
import pandas as pd

#生成异常数据
data1,data2,data3,data4 = ['a',3],['b',2],['a',3],['c',2]
df = pd.DataFrame([data1,data2,data3,data4],columns = ['col1','col2'])
print('数据为: \n',df)

# 判断重复数据记录
isDuplicated = df.duplicated()
print('重复值为: \n',isDuplicated)
print('删除数据记录中所有列值相同的记录\n',df.drop_duplicates())
print('删除数据记录中col1值相同的记录\n',df.drop_duplicates(['col1']))
print('删除数据记录中col2值相同的记录\n',df.drop_duplicates(['col2']))
print('删除数据记录中指定列 (col1/col2) 值相同的记录\n',df.drop_duplicates(['col2','col2']))
```

数据为:

	col1	col2
0	a	3
1	b	2
2	a	3
3	c	2

重复值为:

0	False
1	False
2	True
3	False

dtype: bool

删除数据记录中所有列值相同的记录

	col1	col2
0	a	3
1	b	2
3	c	2

删除数据记录中col1值相同的记录

	col1	col2
0	a	3
1	b	2
3	c	2

删除数据记录中col2值相同的记录

	col1	col2
0	a	3
1	b	2

删除数据记录中指定列 (col1/col2) 值相同的记录

	col1	col2
0	a	3
1	b	2



# 6. 数据转换

Pandas提供数据转换相关函数

函数名	说明
df.replace(a,b)	用b值替换a值
df['coll'].map()	对指定列进行函数转换，用于Series
pd.merge(df1,df2)	用于合并df1和df2，按照共有的列连接
df1.combine_first(df2)	用df2的数据补充df1的缺失值

表6-1 数据转换函数



# 6. 数据转换

## 6.1 数据值替换: `df.replace(a, b)` 用b值替换a值

### 【例10】`df.replace()` 举例

```
# 例10: df.replace(a,b)用b值替换a
import pandas as pd

# 创建数据集
df = pd.DataFrame(
    {'名称': ['产品1', '产品2', '产品3', '产品4', '产品5', '产品6', '产品7', '产品8'],
     '数量': ['A', '0.7', '0.8', '0.4', '0.7', 'B', '0.76', '0.28'],
     '金额': ['0', '0.48', '0.33', 'C', '0.74', '0', '0', '0.22'],
     '合计': ['D', '0.37', '0.28', 'E', '0.57', 'F', '0', '0.06']})

# 原DataFrame并没有改变, 改变的只是一个副本
print('df:\n{}\n'.format(df))
df1 = df.replace('A', 0.1)
print('df1:\n{}\n'.format(df1))

# 只需替换某个数据的部分内容
df2 = df['名称'].str.replace('产品', 'product')
print('df2:\n{}\n'.format(df2))

# 如果需要改变原数据, 需要添加常用参数inplace = True, 用于替换部分区域
df['合计'].replace({'D': 0.1111, 'F': 0.2222}, inplace = True)
print('df:\n{}\n'.format(df))
```

df:

	名称	数量	金额	合计
0	产品1	A	0	D
1	产品2	0.7	0.48	0.37
2	产品3	0.8	0.33	0.28
3	产品4	0.4	C	E
4	产品5	0.7	0.74	0.57
5	产品6	B	0	F
6	产品7	0.76	0	0
7	产品8	0.28	0.22	0.06

df2:

0	product1
1	product2
2	product3
3	product4
4	product5
5	product6
6	product7
7	product8

Name: 名称, dtype: object

df1:

	名称	数量	金额	合计
0	产品1	0.1	0	D
1	产品2	0.7	0.48	0.37
2	产品3	0.8	0.33	0.28
3	产品4	0.4	C	E
4	产品5	0.7	0.74	0.57
5	产品6	B	0	F
6	产品7	0.76	0	0
7	产品8	0.28	0.22	0.06

df:

	名称	数量	金额	合计
0	产品1	A	0	0.1111
1	产品2	0.7	0.48	0.37
2	产品3	0.8	0.33	0.28
3	产品4	0.4	C	E
4	产品5	0.7	0.74	0.57
5	产品6	B	0	0.2222
6	产品7	0.76	0	0
7	产品8	0.28	0.22	0.06



# 6. 数据转换

## 6.2 数据值映射: `df. [ 'coll' ].map()` 对特定列进行函数转换, 用于Series

### 【例11】 `df. [].map()` 举例

```
# 例11: df[].map()对指定列进行函数转换
import pandas as pd
import numpy as np
data = {'姓名':['周元哲','潘婧','詹涛','王颖','李震'], '性别':['1','0','0','0','1']}
df = pd.DataFrame(data)
df['成绩'] = [98,87,32,67,77]
print(df)

def grade(x):
    if x >= 90:
        return '优秀'
    elif x >= 80:
        return '良好'
    elif x >= 70:
        return '中等'
    elif x >= 60:
        return '及格'
    else:
        return '不及格'

df['等级'] = df['成绩'].map(grade)

print('\n',df)
```

	姓名	性别	成绩
0	周元哲	1	98
1	潘婧	0	87
2	詹涛	0	32
3	王颖	0	67
4	李震	1	77

	姓名	性别	成绩	等级
0	周元哲	1	98	优秀
1	潘婧	0	87	良好
2	詹涛	0	32	不及格
3	王颖	0	67	及格
4	李震	1	77	中等



# 6. 数据转换

## 6.3 数据值合并

`pd.merge(df1, df2)` 用于合并`df1`和`df2`，按照共有的列连接

`pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=True)`

参数	说明	参数	说明
left	拼接的左侧DataFrame对象	left_index	使用左侧DataFrame中的索引（行标签）作为连接键
right	拼接的右侧DataFrame对象	right_index	使用右侧DataFrame中的索引（行标签）作为连接键
on	要加入的列或索引级别名称	how	取值('left','right','outer','inner')，默认为'inner'。 inner表示取交集，outer表示取并集
left_on	左侧DataFrame中的列	sort	按字典顺序通过连接键对结果DataFrame进行排序
right_on	右侧DataFrame中的列		

表6-2 `pd.merge()`参数说明



# 6. 数据转换

## 6.3 数据值合并

### 【例12】pd.merge(df1, df2) 举例

```
# 例12: pd.merge(df1,df2)用于合并df1、df2，按照共有的列连接
import pandas as pd
left = pd.DataFrame({'key':['k0','k1','k2','k3'],'A':['A0','A1','A2','A3'],'B':['B0','B1','B2','B3']})
right = pd.DataFrame({'key':['k0','k1','k2','k3'],'C':['C0','C1','C2','C3'],'D':['D0','D1','D2','D3']})

# on参数传递的key作为连接键
result = pd.merge(left,right,on = 'key')

print('left:\n{}\n'.format(left))
print('left:\n{}\n'.format(right))
print('left:\n{}\n'.format(result))
```

	key	A	B
0	k0	A0	B0
1	k1	A1	B1
2	k2	A2	B2
3	k3	A3	B3

	key	C	D
0	k0	C0	D0
1	k1	C1	D1
2	k2	C2	D2
3	k3	C3	D3

	key	A	B	C	D
0	k0	A0	B0	C0	D0
1	k1	A1	B1	C1	D1
2	k2	A2	B2	C2	D2
3	k3	A3	B3	C3	D3





# 6. 数据转换

## 6.4 数据值补充: df1.combine\_first(df2)用df2的数据补充df1的缺失值

### 【例13】 df1.combine\_first(df2)举例

```
# 例13: df1.combine_first(df2)用df2的数据补充df1的缺失值
from numpy import nan as NaN
import numpy as np
import pandas as pd
a = pd.Series([np.nan,2.5,np.nan,3.5,4.5,np.nan],index = ['f','e','d','c','b','a'])
b = pd.Series([1,np.nan,3,4,5,np.nan],index = ['f','e','d','c','b','a'])
print(a)
print(b)
c = b.combine_first(a)
print(c)
```

f	NaN	f	1.0	f	1.0
e	2.5	e	NaN	e	2.5
d	NaN	d	3.0	d	3.0
c	3.5	c	4.0	c	4.0
b	4.5	b	5.0	b	5.0
a	NaN	a	NaN	a	NaN
dtype: float64		dtype: float64		dtype: float64	



## 7. Missingno 库

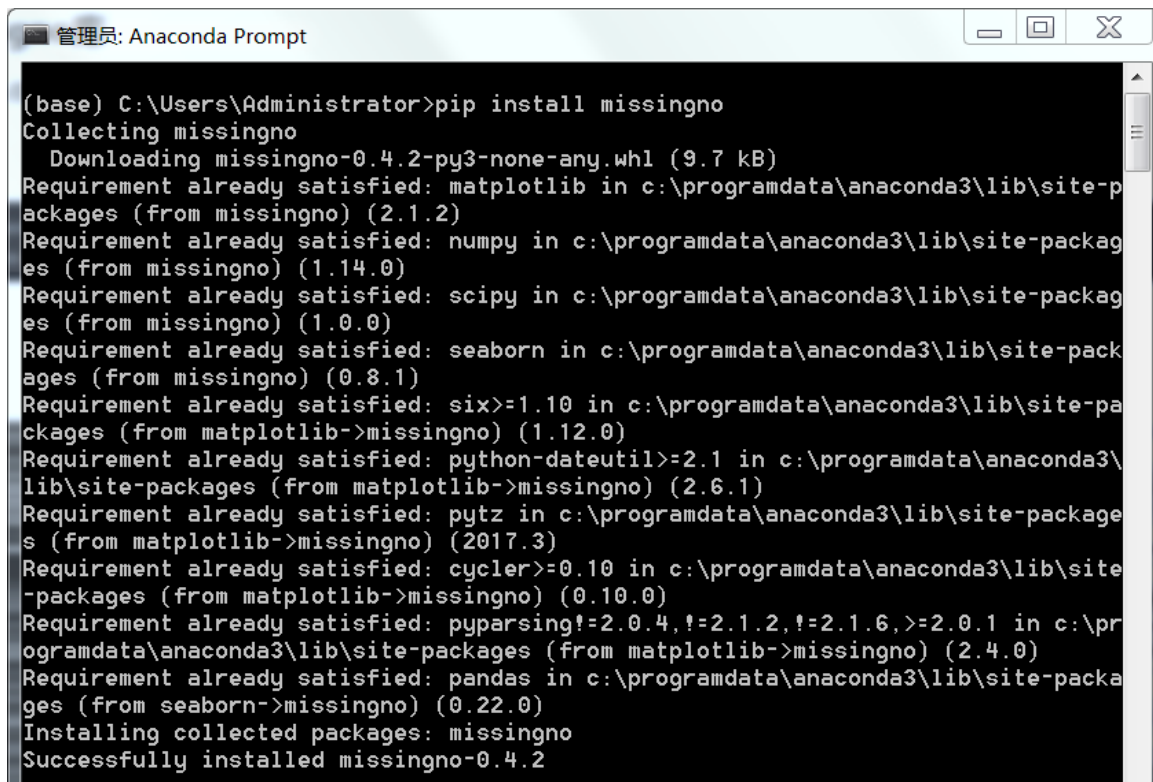
- Missingno 库用于缺失数据的可视化，便于快速直观地分析数据集的完整性

- 安装命令

`pip install missingno`

- 加载命令

`import missingno as msno`



```
(base) C:\Users\Administrator>pip install missingno
Collecting missingno
  Downloading missingno-0.4.2-py3-none-any.whl (9.7 kB)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from missingno) (2.1.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from missingno) (1.14.0)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from missingno) (1.0.0)
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-packages (from missingno) (0.8.1)
Requirement already satisfied: six>=1.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (1.12.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (2.6.1)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (2017.3)
Requirement already satisfied: cycloper>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (2.4.0)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (from seaborn->missingno) (0.22.0)
Installing collected packages: missingno
Successfully installed missingno-0.4.2
```

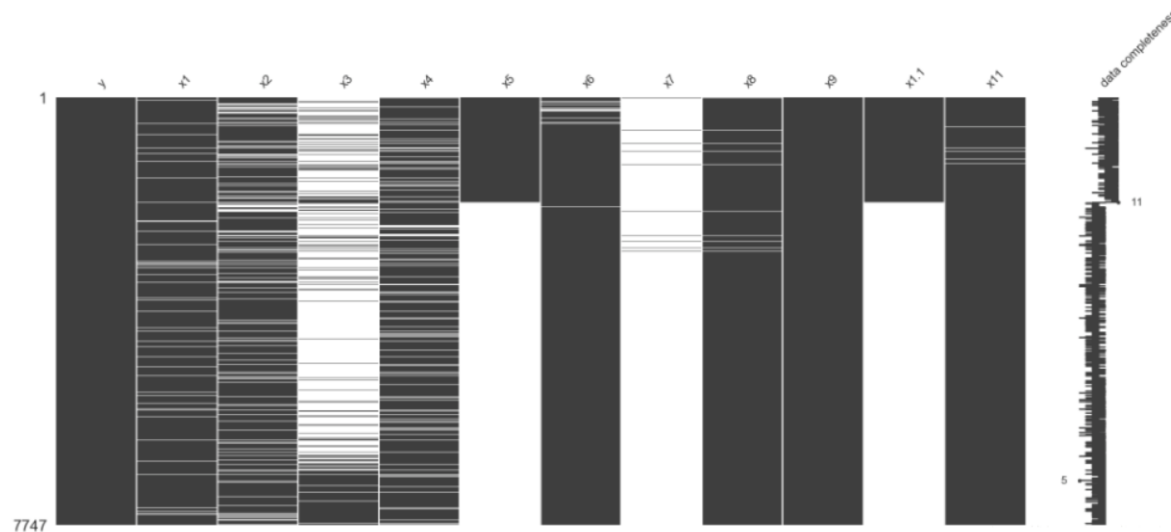


## 7. Missingno 库

功能1：无效矩阵的数据显示，快速直观地显示数据

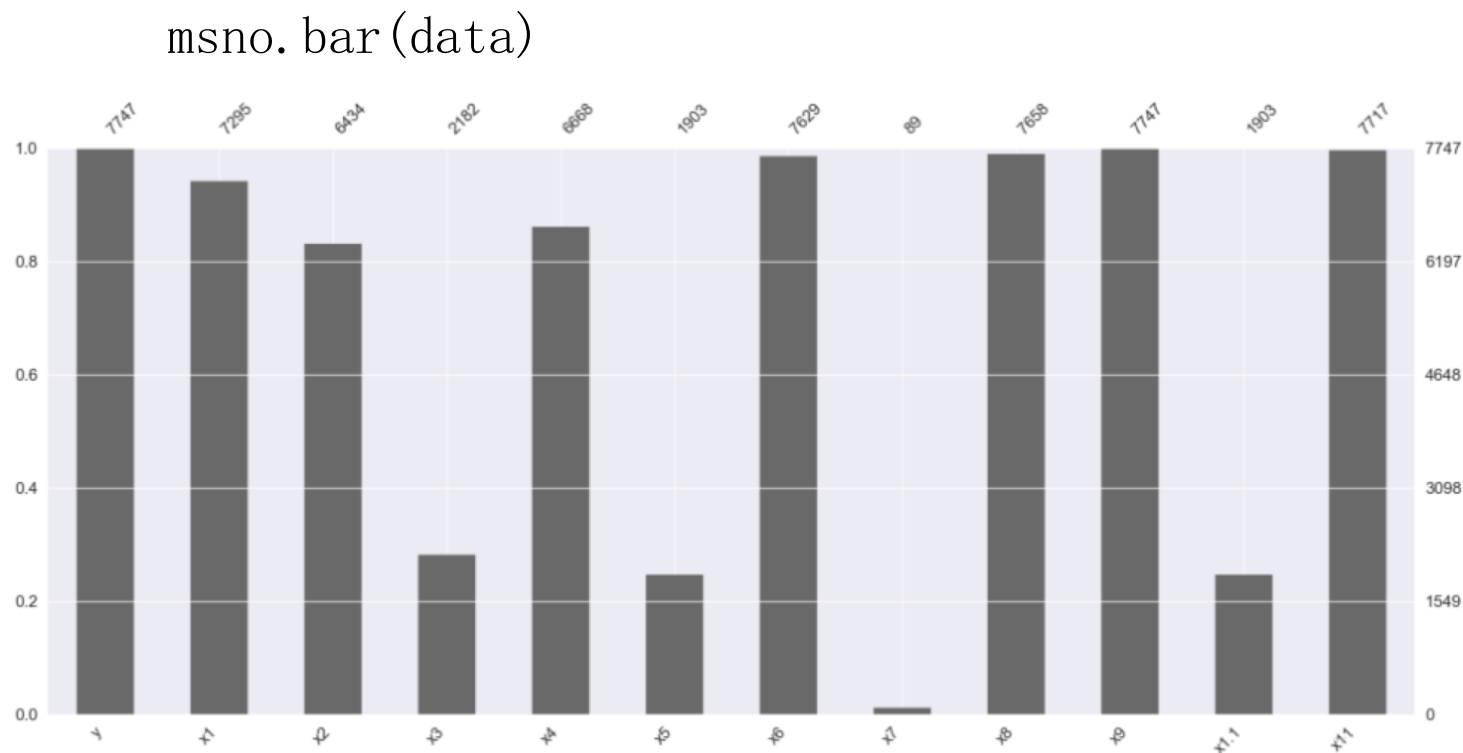
```
msno.matrix(data, labels=True)
```

无效矩阵的数据显示用于查看每个变量的缺失情况。下图中变量  $y$ ,  $X9$  数据完整，其他变量都有不同程度的缺失，尤其是  $X3$ ,  $X5$ ,  $X7$  等数据缺失严重。



## 7. Missingno 库

功能2: 列的无效可视化, 利用条形图可以直观地看出每个变量缺失的比例和数量情况

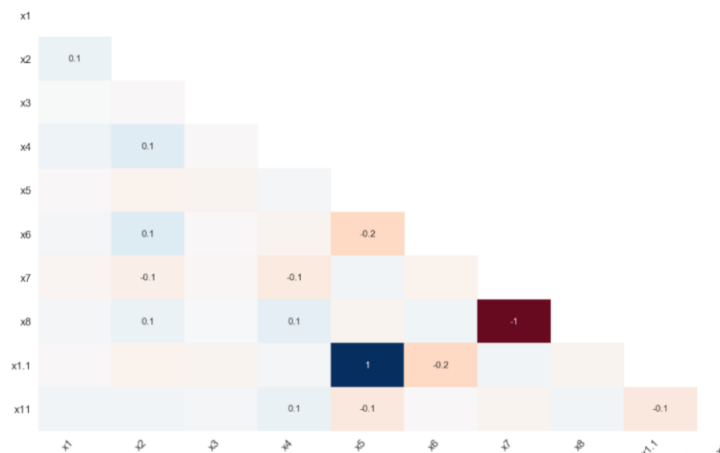


## 7. Missingno 库

功能3：热图相关性显示，用于说明变量间是否相互影响

```
msno.heatmap(data)
```

下图中X5与X1.1的缺失相关性为1，说明X5与X1.1正相关，即只要X5发生缺失，X1.1必然会缺失。X7和X8的相关性为一1，说明X7和X8负相关，X7缺失，X8不缺失；反之，X7不缺失，X8缺失。

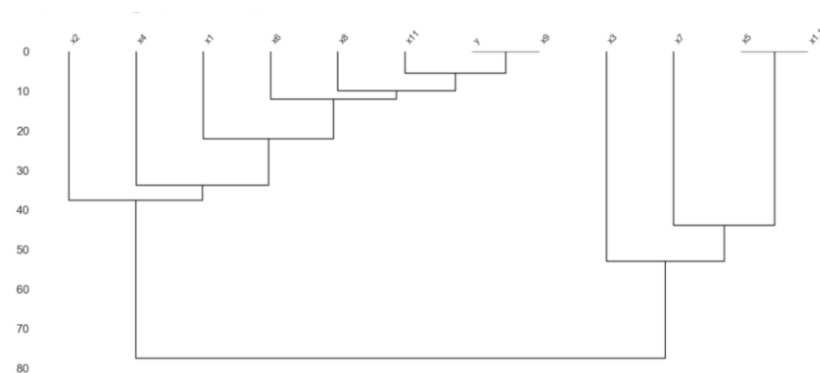


## 7. Missingo 库

功能4：树形图显示，使用层次聚类算法将无效性变量彼此相加

```
msno.dendrogram(data)
```

树形图中数据越完整，距离越接近零，越靠近y轴。数据分为左边数据和右边数据。其中，左边数据比较完整，Y和X9是完整数据，没有缺失值，距离为0；相对于其他变量，X11也比较完整，距离要比其他变量小，以此类推。右边数据的缺失值比较严重，由热图相关性得出X5和X1.1的相关性系数为1，距离为0，以此类推。



# 7. Missingno 库

## 【例14】Missingno例子

# 例14: Missingno 举例

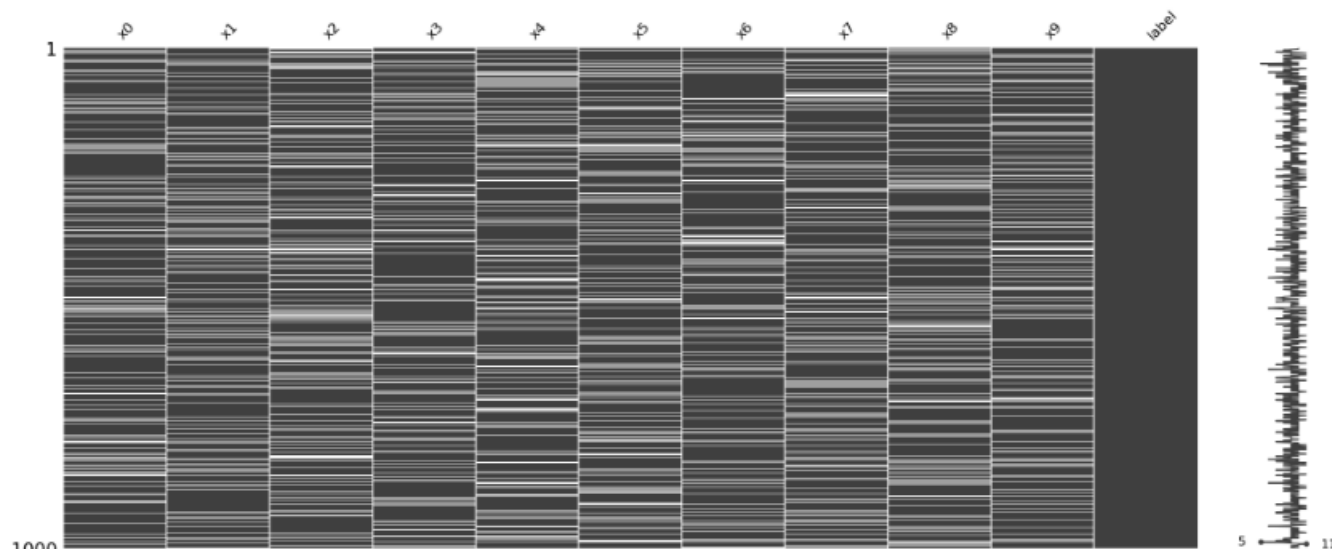
# Sklearn中make\_classification生成数据集, 增加随机Na值, 生成数据

```
import warnings
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
import missingno as msno
import matplotlib.pyplot as plt
from itertools import product
warnings.filterwarnings('ignore')
```

# 自定义数据集, 并随机产生2000个Na值, 分布在各个特征中

```
def getData():
    X1,y1 = make_classification(n_samples = 1000
                               ,n_features = 10
                               ,n_classes = 2
                               ,n_clusters_per_class = 1
                               ,random_state = 0
                               )
    for i,j in product(range(X1.shape[0]),range(X1.shape[1])):
        if np.random.random() >= 0.8:
            xloc = np.random.randint(0,10)
            X1[i,xloc] = np.nan
    return X1,y1
```

```
x,y = getData()
df = pd.DataFrame(x,columns = ['x%s'%str(i) for i in range(x.shape[1])])
df['label'] = y
msno.matrix(df)
plt.show()
```



## 8. 词云

word\_cloud是python的第三方库，称为词云，也叫做文字云，是对文本中出现频率较高的关键词给予视觉上的突出，形成“关键词云层”或“关键词渲染”，通过过滤掉大量的文本信息，直观和艺术的展示文本中词语的重要性，使得读者一眼便能领略文本的主旨。





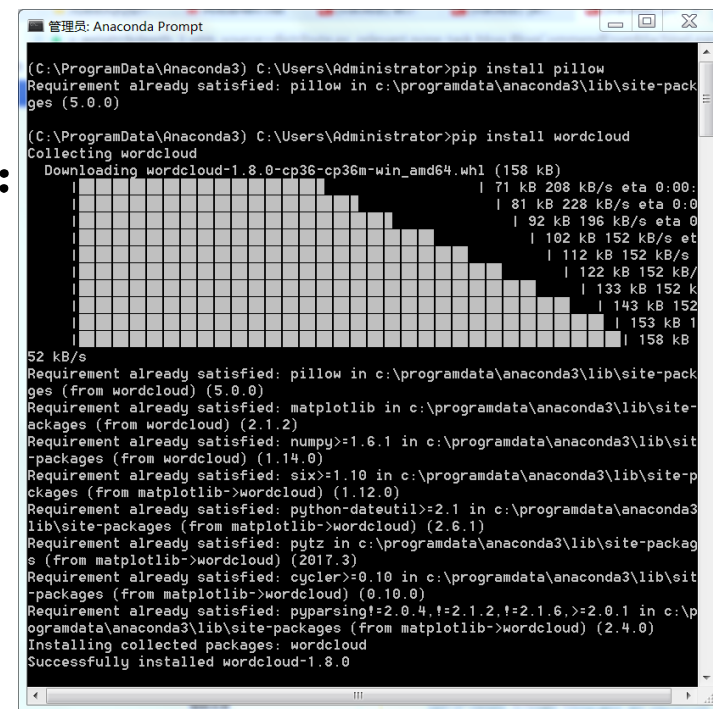
## 8. 词云

word\_cloud是python的第三方库，称为词云，也叫做文字云，是对文本中出现频率较高的关键词给予视觉上的突出，形成“关键词云层”或“关键词渲染”，通过过滤掉大量的文本信息，直观和艺术的展示文本中词语的重要性，使得读者一眼便能领略文本的主旨。

word\_cloud依赖于Numpy和Pillow，安装指令如下：

```
pip install pillow
```

```
pip install wordcloud
```



```
管理员: Anaconda Prompt
(C:\ProgramData\Anaconda3) C:\Users\Administrator>pip install pillow
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (5.0.0)

(C:\ProgramData\Anaconda3) C:\Users\Administrator>pip install wordcloud
Collecting wordcloud
  Downloading wordcloud-1.8.0-cp36-cp36m-win_amd64.whl (158 kB)
    | 71 kB 208 kB/s eta 0:00
    | 81 kB 228 kB/s eta 0:00
    | 92 kB 196 kB/s eta 0:00
    | 102 kB 152 kB/s eta 0:00
    | 112 kB 152 kB/s eta 0:00
    | 122 kB 152 kB/s eta 0:00
    | 133 kB 152 kB/s eta 0:00
    | 143 kB 152 kB/s eta 0:00
    | 153 kB 152 kB/s eta 0:00
    | 158 kB 152 kB/s eta 0:00
52 kB/s
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (5.0.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from wordcloud) (2.1.2)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.14.0)
Requirement already satisfied: six>=1.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.12.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.6.1)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2017.3)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.4.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.0
```



## 8. 词云

WordCloud 以词语为基本单位，根据文本中词语出现的频率等参数绘制词云的形状、尺寸和颜色等，生成词云的过程如下。

步骤1：使用 Pandas 读取数据并转换为列表。

步骤2：对列表数据使用分词工具 jieba 进行中文分词。

步骤3：使用 WordCloud 设置词云图片的属性、掩码和停止词等对象参数，加载词云文本，生成词云图像，保存输出词云文件。



# 8. 词云

## WordCloud常规方法

方法	方法描述
w=wordcloud.WordCloud(<参数>)	配置对象参数
w.generate(txt)	向WordCloud对象w中加载文本txt
w.to_file(filename)	将词云输出为图像文件，.png或.jpg

表8-1 WordCloud常规方法



# 8. 词云

## word\_cloud绘图参数

属性名	示例	说明
back_ground_color	back_ground_color='white'	指定背景色，可以使用16进制颜色
width	width=600	图像长度，默认400px
height	height=400	图像高度，默认200px
margin	margin=20	词与词之间的间距，默认2px
scale	scale=0.5	缩放比例，对图像整体进行缩放，默认为1
prefer_horizontal	prefer_horizontal=0.9	词在水平方向上出现的概率，默认为0.5
stopwords	stopwords=set('dog')	设置要过滤的词，以字符串或者集合作为接收参数，如不设置将使用默认的停用词库
relative_scaling	relative_scaling=1	词频与字体大小关联性，默认为5，值越小，变化越明显

表8-2 word\_cloud绘图



## 8. 词云

### 【例15】word\_cloud举例

```
# 例15: word_cloud举例
from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = 'cat fish cat cat cat cat cat cat dog dog cat dog'
wordcloud = WordCloud(
    font_path='msyh',    # 设置输出词云的字体
    max_font_size = 500,  # 设置字体的大小, 默认为200
    # background_color = 'white'
    width = 1000,height = 400,
    scale = 2,    # 设置图的词密度
    random_state = 50,    # random.Random用来生成随机颜色
).generate(text)    # generate()根据文本生成词云

# wordcloud.to_file('C:\\Users\\Desktop\\print.png')    # 输出词云文件到C:\\Users\\Desktop\\print.png
plt.imshow(wordcloud,interpolation = 'none')
plt.axis('off')    # 关闭x,y轴刻度
plt.savefig('C:\\Users\\Desktop\\print.png')
plt.show()
```



