数据清洗

1、缺失值处理
2、异常值处理
3、重复值处理

处理方式一般有2种：drop去掉、fill填充、interpolate插值拟合

In [1]:
```
import pandas as pd
```

In [2]:
```
data = {
    'name': ['Jack', 'Mary', 'Lily', 'Tom', 'Joe', None],
    'age': [18, None, 21, 25, 24, None],
    'score': ['A', 'B', 'A', None, None, None]
}

df = pd.DataFrame(data, columns = ['name', 'age', 'score'])
df
```

Out[2]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None
5	None	NaN	None

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
name     1
age      2
score    3
dtype: int64
```

1、dropna()

In [4]:
```
df.dropna()        # 删除有None的数据
```

Out[4]:

	name	age	score
0	Jack	18.0	A
2	Lily	21.0	A

In [5]:
```
df.dropna(how='all')     # 删除一组数据全部（all）是None的数据，how默认是
any
```

Out[5]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None

In [6]:
```
df
```

Out[6]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None
5	None	NaN	None

In [7]:
```
df.dropna(subset=['age'])     # 删除指定列数据为None的数据
```

Out[7]:

	name	age	score
0	Jack	18.0	A
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None

2、fillna(...)

In [8]:
```
df.fillna(0)
```

Out[8]:

	name	age	score
0	Jack	18.0	A
1	Mary	0.0	B
2	Lily	21.0	A
3	Tom	25.0	0
4	Joe	24.0	0
5	0	0.0	0

In [9]:
```
df.fillna({'age': 0, 'score': 'D'})
```

Out[9]:

	name	age	score
0	Jack	18.0	A
1	Mary	0.0	B
2	Lily	21.0	A
3	Tom	25.0	D
4	Joe	24.0	D
5	None	0.0	D

In [10]:
```
df.fillna(method='ffill')     # font fill     None的数据按照他的前一个数据填写
```

Out[10]:

	name	age	score
0	Jack	18.0	A
1	Mary	18.0	B
2	Lily	21.0	A
3	Tom	25.0	A
4	Joe	24.0	A
5	Joe	24.0	A

In [11]:
```
df.fillna(method='bfill')    # back fill      score列因为后面没有数据，所以没能填充
```

Out[11]:

	name	age	score
0	Jack	18.0	A
1	Mary	21.0	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None
5	None	NaN	None

In [12]:
```
df.fillna({'age': df.age.mean()})      # 用平均数填充
```

Out[12]:

	name	age	score
0	Jack	18.0	A
1	Mary	22.0	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None
5	None	22.0	None

In [13]:
```
df.fillna({'age': df.age.median()})      # 用中位数填充
```

Out[13]:

	name	age	score
0	Jack	18.0	A
1	Mary	22.5	B
2	Lily	21.0	A
3	Tom	25.0	None
4	Joe	24.0	None
5	None	22.5	None

In [14]:
```
df.fillna({'score': 'missing'})
```

Out[14]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	missing
5	None	NaN	missing

In [15]:
```
df.fillna({'score': 'missing'}, limit=1)
```

Out[15]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None

In [16]:
```
df.fillna({'score': 'missing'}, limit=1, inplace=True)   # 加上 inpl
ace=True 可以在运行后保存到原来的数据中
df
```

Out[16]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None

3、interpolate

In [17]:
```
df
```

Out[17]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None

In [18]:
```
df.interpolate()    # 默认线性插值
```

Out[18]:

	name	age	score
0	Jack	18.0	A
1	Mary	19.5	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	24.0	None

In [19]:
```
# df.interpolate?
```

插值方式： method : {'linear', 'time', 'index', 'values', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 'barycentric', 'krogh', 'polynomial', 'spline', 'piecewise_polynomial', 'from_derivatives', 'pchip', 'akima'}

In [20]:
```
df.interpolate(method='cubic')
```

Out[20]:

	name	age	score
0	Jack	18.0	A
1	Mary	17.0	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None

异常值处理

In [21]:
```
df.loc[6] = {'name': 'test', 'age': 999, 'score': 'C'}
```

In [22]:
```
df
```

Out[22]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None
6	test	999.0	C

In [23]:
```
q_upper = df['age'].quantile(0.75)
q_upper
```

Out[23]:
```
25.0
```

In [24]:
```
q_lower = df['age'].quantile(0.25)
q_lower
```

Out[24]:
```
21.0
```

In [25]:
```
val = q_upper - q_lower
val
```

Out[25]:
```
4.0
```

In [26]:
```
k = 1.5
```

In [27]:
```
df[df['age'] > q_upper + k * val]
```

Out[27]:

	name	age	score
6	test	999.0	C

In [28]:
```
df2 = df.drop(6)
```

In [29]:
```
df2
```
Out[29]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None

In [30]:
```
df[(df['age'] < q_upper + k * val) & (df['age'] > q_lower - k * val
)]
```
Out[30]:

	name	age	score
0	Jack	18.0	A
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None

重复值

drop_duplicates()

In [31]:
```
df
```
Out[31]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None
6	test	999.0	C

```
In [32]: df.loc[7] = {'name': 'test', 'age': 999, 'score': 'C'}

In [33]: df.loc[8] = {'name': 'test2', 'age': 999, 'score': 'C'}

In [34]: df
```

Out[34]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None
6	test	999.0	C
7	test	999.0	C
8	test2	999.0	C

```
In [35]: df.drop_duplicates()        # 一行数据与另一行数据完全相同才判定为重复
```

Out[35]:

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None
6	test	999.0	C
8	test2	999.0	C

```
In [36]: df.drop_duplicates(['age', 'score'])        # 只要age和score两列相同，就
         判定为重复值
```

```
Out[36]:
```

	name	age	score
0	Jack	18.0	A
1	Mary	NaN	B
2	Lily	21.0	A
3	Tom	25.0	missing
4	Joe	24.0	None
5	None	NaN	None
6	test	999.0	C