**轴向旋转**

**pivot()方法(行变列)**

DataFrame.pivot(index=None,columns=None,values=None)

index：表示新生成对象的行索引

columns：表示新生成对象的列索引

values：表示用来填充新生对象的数据

**melt()方法(列变行)**

DataFrame.**melt(id\_vars=None,value\_vars=None,var\_name=None,col\_level=None,ignore\_index=True)**

**id\_vars：表示无须被转换的列索引**

**value\_vars：表示要被转换的列索引**

**var\_name：表示自定义的列索引**

**col\_level：表示列索引的级别**

**ignore\_index：表示是否忽略索引**

**分组与聚合**

**分组操作**

df.groupby(by=None,axis=0,level=None,as\_index=True,sort=True,group\_keys=True)

by：表示分组的条件

axis：0或index表示按列；1或columns表示按行

level：**表示索引的级别**

as\_index：表示聚合后新数据的索引是否为分组标签的索引

sort：表示是否对分组索引进行排序

group\_keys：表示是显示分组标签的名称

**聚合操作**

**agg()**

df.agg({'a':'max','b':'sum'})#agg方法只需要传入{'列索引':'函数/方法名'}就可以聚合

**transform()**

transfrom(func,\*args,engine=None)

func：表示应用各分组的函数或方法

\*args：给func的位置参数

engine：cython表示用C语言来运行；numba表示用JIT来运行

这样可以保持与原数据相同的数据结构

**apply()**

df.apply(lambda x: sum(x.isnull())/len(x),axis= 0) #缺失比例

**哑变量处理**

pd.get\_dummies(data,prefix=None,prefix\_sep='\_',dummy\_na=False,columns=None,sparse=False,drop\_first=False)

data：待处理的数据

prefix：表示列索引名称的前缀

prefix\_sep：表示列索引名称的前缀的分隔符

dummy\_na：表示是否为Nan添加一列

columns：表示哑处理的列索引名称

sparse：表示哑处理的、是否稀疏

drop\_first：表示是否将K个分类级别的第一个级别删去

其实就是做假数据

**离散化处理**

pd.cut(x,bins,right=True,labels=None,retbins=False,precision=3,include\_lowest=False)

x：要处理的连续数据

bins：处理的依据(int的是面元的数目；其他则是划分的区间)

right：表示右端点是否为闭区间

labels：表示划分各区间的标签

retbins：表示是否返回区间的标签

precision：表示是区间标签的精度

include\_lowest：表示是否包含区间的左端点