**数据预处理**

<https://github.com/fuqiuai/sklearn-feature-engineering>

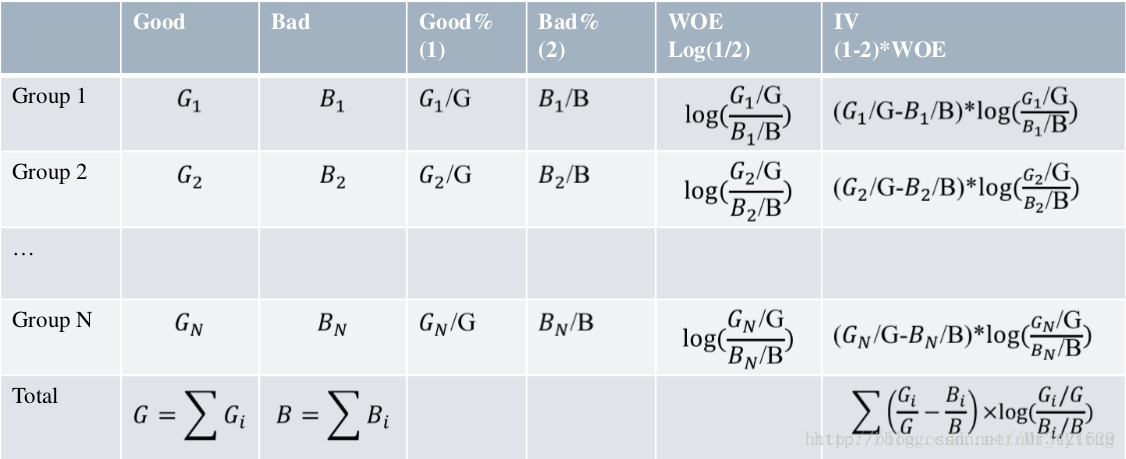
# 数据预处理之IV

## 图解IV

iv(信息量)值，变量的预测能力，只能处理二分类问题

|  |  |
| --- | --- |
| 区间范围 | 预测能力 |
| (-∞，0.02] | 无预测能力 |
| (0.02,0.1] | 弱预测能力 |
| (0.1,+∞] | 强预测能力，建议保留 |

自变量有group1、group2……groupN个值，因变量good指G个，bad值B个。



<https://blog.csdn.net/iModel/article/details/79420437> iv值理解

<https://blog.csdn.net/weixin_38940048/article/details/82316900> iv值计算代码

## python代码

# -\*- coding:utf-8 -\*-

from numpy import log

from pandas import DataFrame as df

import pandas as pd

def createDateset():

dataSet=[

[0, 1, 0],

[0, 0, 0],

[0, 1, 0],

[1, 0, 1],

[1, 0, 0],

[1, 1, 1],

[0, 1, 1],

[1, 1, 1],

[1, 0, 1],

[1, 0, 1]]

return dataSet

def calcWOE(dataset,col,targe):

subdata=df(dataset.groupby(col)[col].count())

suby=df(dataset.groupby(col)[targe].sum())

data=df(pd.merge(subdata,suby,how="left",left\_index=True,right\_index=True))

b\_total=data[targe].sum()

total=data[col].sum()

g\_total=total-b\_total

data["bad"]=data.apply(lambda x:round(x[targe]/b\_total,3),axis=1)

data["good"]=data.apply(lambda x:round((x[col]-x[targe])/g\_total,3),axis=1)

data.loc[data[‘good’]==0,’good’]=1

data["WOE"]=data.apply(lambda x:log(x.bad/x.good),axis=1)

return data.loc[:,["bad","good","WOE"]]

def calcIV(dataset):

dataset["IV"]=dataset.apply(lambda x:(x.bad-x.good)\*x.WOE,axis=1)

IV=sum(dataset["IV"])

return IV

if \_\_name\_\_ == '\_\_main\_\_':

data=createDateset()

data=df(data,columns=["x1","x2","y"])

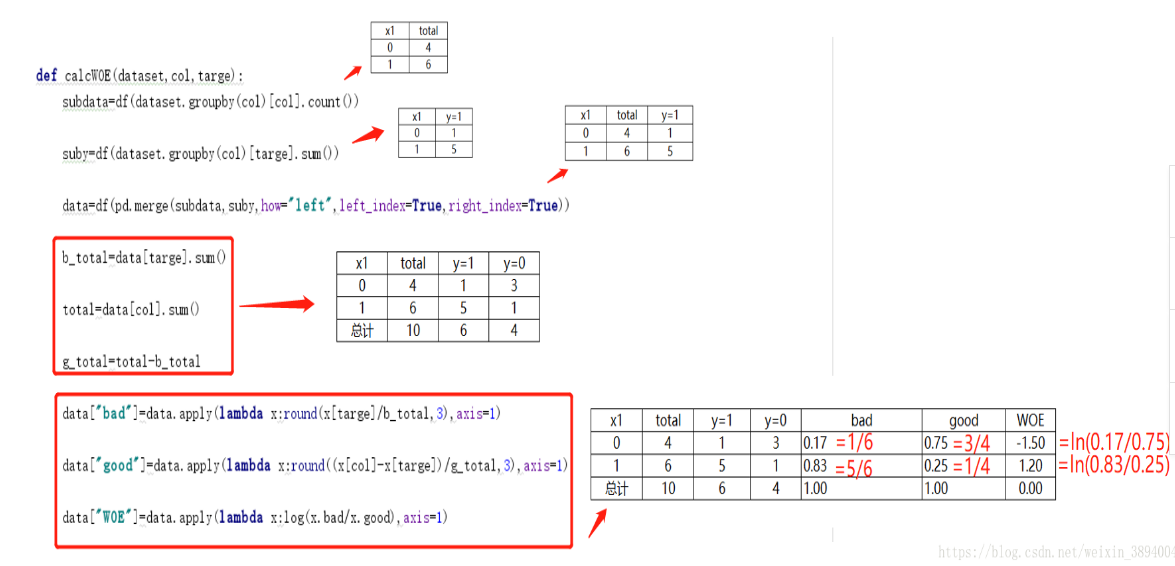
data\_WOE=calcWOE(data,"x1","y")

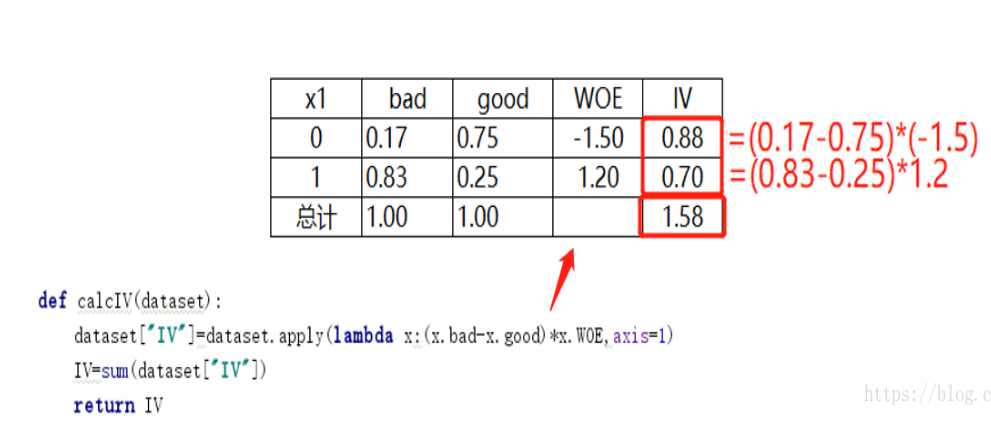
print(data\_WOE)

data\_IV=calcIV(data\_WOE)

print(data\_IV)

## python代码可视化





## 异常处理

使用IV其实有一个缺点，就是不能自动处理变量的分组中出现响应比例为0或100%的情况。那么，遇到响应比例为0或者100%的情况，我们应该怎么做呢？建议如下：

（1）如果可能，直接把这个分组做成一个规则，作为模型的前置条件或补充条件；

（2）重新对变量进行离散化或分组，使每个分组的响应比例都不为0且不为100%，尤其是当一个分组个体数很小时（比如小于100个），强烈建议这样做，因为本身把一个分组个体数弄得很小就不是太合理。

（3）如果上面两种方法都无法使用，建议人工把该分组的响应数和非响应的数量进行一定的调整。如果响应数原本为0，可以人工调整响应数为1，如果非响应数原本为0，可以人工调整非响应数为1.