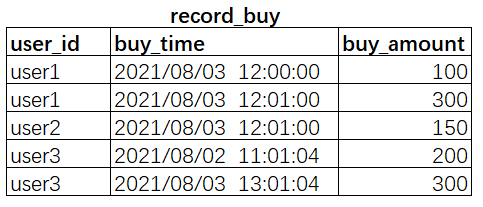
**一、SQL部分**

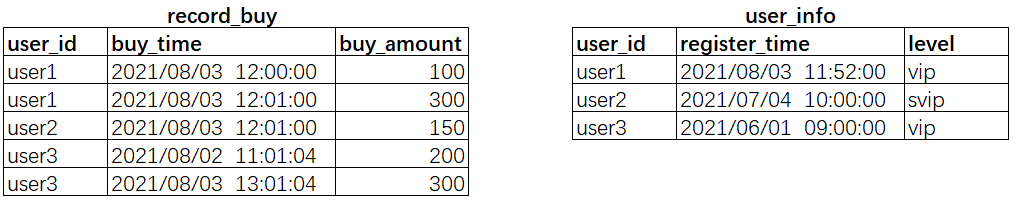
**1.**现有一张表record\_buy，包含客户的购买商品记录，一个客户可能有多条购买记录，如下所示。

**要求：**写出sql语句取出每个用户最近一笔的购买记录



**2.**现有一张用户信息表user\_info以及一张用户购买商品记录表record\_buy，一个客户可能有多条购买记录但只有一条用户信息，两表如下所示，

**要求：**写出sql语句，按注册月份统计vip客户数，按注册月份统计平均每次消费金额，按注册月份统计消费金额大于100的客户数



**二、PYTHON部分**

**3.** 现有一部分现金贷订单对应的贷后样本，该数据集为df(DataFrame格式)，数据维度为10000\*10，其中Y标签对应的变量名target，且好坏样本占比分别为3:1(1代表好样本，0代表坏样本)。其余9列为基础信息与多头等指标组成的变量，分别为[base1,base2,base3,base3,multi1,multi2,multi3,multi4,multi5,score1]。变量IV值的基本计算公式为IV=SUM[(x-y)\*log(x/y)]，其中x = bad\_bin/bad\_total,y = good\_bin/good\_total，bad\_bin为变量分箱后单个分箱内的坏样本数量，bad\_total为整体的坏样本数量；同理，good\_bin为变量分箱后单个分箱内的好样本数量，good\_total为整体的好样本数量。下面给出多变量等频分箱的计算代码，请结合该分箱代码与IV值计算公式，给出变量IV值计算的全流程代码(变量IV值的输出格式可以是json串，也可以是dataframe格式的数据)。

#等频分箱代码

**#需要注意的是，给出的等频分箱代码输出的结果是所有变量与其分箱节点所形成的json串，类似于**

**{var1:[float('-inf'),1,2,3,4,5,float('inf')],var2:[float('-inf'),4,5,6,7,8,float('inf')],var3:[float('-inf'),9,10,11,12,13,float('inf')]}；后续在进行IV值计算的过程中需要结合pd.cut()与groupby+apply函数计算所有变量在不同箱体下的好坏占比，结合IV值计算公式最终得到变量的IV值。各位同学也不要过于纠结数据本身和好坏占比等指标，这里只是举个例子而已，让数据更为形象化一些。变量IV值的计算在风控建模过程中相对基础也比较重要，这里主要是希望答题的同学能够在所写的代码中体现整个计算的逻辑与思路。**

import pandas as pd

import numpy as np

import toad

def QuantileMergeSplit(DF,collist,n\_bins\_,num\_nanvalues,nanvalue\_rate=0.05):

#DF：原始数据集，dataframe格式

#collist：需要计算分箱的字段，如[var1,var2,var3,var4]等

#n\_bins\_：箱体数量，等频分箱数量一般在10个左右

#num\_nanvalues：缺失值填充值，一般使用-9999979

#nanvalue\_rate：缺失值占比

#返回值(return)：输出所有变量分箱结果对应的json串，如

#{var1:[float('-inf'),1,2,3,4,5,float('inf')],var2:[float('-inf'),4,5,6,7,8,float('inf')],var3:[float('-inf'),9,10,11,12,13,float('inf')]}；

df = DF.copy()

c = toad.transform.Combiner()

#默认使用等频分箱操作进行数据切分

c.fit(df[collist],method='quantile',n\_bins=n\_bins\_,empty\_separate = False)

bins1 = c.export()

#将最终计算结果输出

for col in collist:

splits = bins1[col]

if (num\_nanvalues in df[col].values.tolist()) and (num\_nanvalues not in splits) and (df[(df[col]==num\_nanvalues)].shape[0]/df.shape[0]>=nanvalue\_rate):

splits.insert(0,num\_nanvalues)

if float('-inf') not in splits:

splits.insert(0,float('-inf'))

if float('inf') not in splits:

splits.append(float('inf'))

bins1[col] = splits

for x in bins1.keys():

bins1.update({x:[round(y,4) for y in bins1[x]]})

return bins1

**4.** 下面是一段计算手机设备信息的RFM时间切片衍生代码，请结合np.vectorize与gc.collect函数，对代码进行一定程度的优化(需要优化的地方已经做了相应的标注)

import pandas as pd

import numpy as np

import gc

import cate\_num\_feature,cal\_two\_timestamps

#注：cate\_num\_feature,cal\_two\_timestamps是两个写的外部包，不需要对这两个包及其子函数有太多的关注

times\_map = {1:'1d',3:'3d',7:'1w',14:'2w',21:'3w',30:'1m',60:'2m',90:'3m'}

def deviceinfofeature(DF,applytimecol,createdtimecol,catecollist,numcollist):

#DF：原始数据源

#applytimecol：申请时间对应的列名,str

#createdtime：入库时间对应的列名，str

#catecollist：离散型变量对应的变量名集合，list

#numcollist：连续型变量对应的变量名集合，list

global times\_map

#初始化全部为空值的返回值

timeprefixtotal = [x for x in times\_map.values()]

totalzerodict = cate\_num\_feature.zerodict(timeprefixtotal,catecollist,numcollist)

df = DF.copy()

#计算申请时间与入库时间之间的时间间隔，做了15分钟的延长；

#**需要优化的地方**

**df['created\_apply\_time\_internal'] =**

**df.apply(lambda x:cal\_two\_timestamps.CalTwotimeStampDiffDays(x[createdtimecol],x[applytimecol],15) if (x[createdtimecol]==x[createdtimecol] and x[applytimecol]==x[applytimecol]) else np.nan,axis=1)**

#缺失值替换

df[catecollist+numcollist] = df[catecollist+numcollist].replace(-1,np.nan)

df[catecollist+numcollist] = df[catecollist+numcollist].replace(-999979,np.nan)

#筛选申请时间前的所有入库数据

df1 = df[(df['created\_apply\_time\_internal']>=0)]

#如果入库时间之前，用户的设备相关历史信息完全没有，则返回初始化的结果，全部为空值

if df1.empty:

return totalzerodict

#否则对原始数据进行时间切片

else:

returndict = {}

for x in times\_map.keys():

timeprefix = times\_map[x]

#进行时间切片，同时需要对入库时间进行排序操作

df2 = df1[(df1['created\_apply\_time\_internal']<=x)]

df2.sort\_values(createdtimecol,inplace=True,ascending=True)

if df2.empty:

#如果时间切片的结果为空，这部分衍生变量全部返回空值

returndict.update(zerodict([timeprefix],catecollist,numcollist))

else:

#如果数值型变量非空，则进行字段衍生

if len(numcollist)>0:

returndict.update(cate\_num\_feature.gennumdict(df2,timeprefix,numcollist))

#如果字符型变量非空，则进行字段衍生

if len(catecollist)>0:

returndict.update(cate\_num\_feature.gencatedict(df2,timeprefix,catecollist))

#**需要优化的地方(for循环内存释放)**

#订单号与产品名字段更新

return returndict

**三、模型部分**

**5.**请根据以下数据计算KS

|  |  |  |
| --- | --- | --- |
| score | good | bad |
| 600-high | 400 | 4 |
| 500-600 | 600 | 20 |
| 400-500 | 600 | 50 |
| 300-400 | 200 | 30 |
| 200-300 | 300 | 50 |
| LOW-200 | 500 | 100 |
| 合计 | 2600 | 254 |

**6.**已有一份客户还款表现数据和对应的年龄数据，如果将客户年龄分为5段，然后根据客户还款表现将客户分为好客户与坏客户，已知每个年龄段内的好客户数量分别为、坏客户数量分别为(i从1到5)，请用公式表达各个年龄段的、以及年龄这个变量的IV

|  |  |  |
| --- | --- | --- |
| 年龄 | good | bad |
| 20-30 | G1 | B1 |
| 30-40 | G2 | B2 |
| 40-50 | G3 | B3 |
| 50-60 | G4 | B4 |
| 60-65 | G5 | B5 |