对所属不同时间的告警数据通过DBSCAN算法进行时间域的划分，经过本函数可以划分不同时间窗口的告警数据片段。

同一批告警集合，采用DataFrame格式，如下示意图，告警不同字段如下：‘告警标题(alarm\_title)’、‘设备名称(device\_name)’、‘发生时间(starttime)’和‘所属时间段(label)’

手机屏幕截图

描述已自动生成

def DBSCANApp(train):

features = train[['timestamp']].values

clustering = DBSCAN(eps=eps, min\_samples=ms).fit(features)#labal=226

labels = clustering.labels\_

num\_clusters = len(set(labels)) - (1 if -1 in labels else 0)

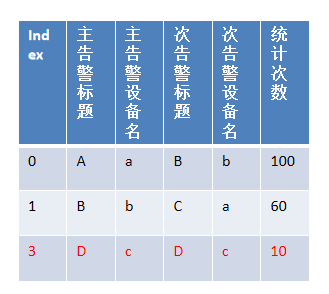
n\_noise = list(labels).count(-1)

train['label'] = labels

train = train[train['label'] != -1]

Return train

基于不同时间窗口的告警数据，采取eclat进行高效地挖掘频繁二元项，具体思路如下：

 ====> 

之后对根据主次告警对在整体样本中的统计次数进行概率统计分析，生成粗糙规则，字段为主告警标题(master\_title)、主告警设备(master\_device\_name)、主告警设备类型(master\_device\_type)、次告警标题(slave\_title)、次告警设备(slave\_device\_name)、次告警设备类型(slave\_device\_type)，结果如下：

：



Eclat算法具体代码如下：

def eclat(ec\_data):

patterns = {}

for item1 in SupL1:

for item2 in SupL1.keys():

freq = len(set(ec\_data[item1])&set(ec\_data[item2]))

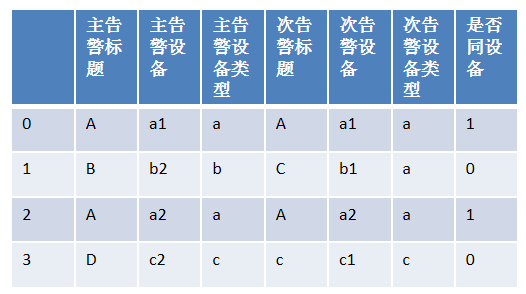
if freq>=support:

patterns.update({(item1,item2):freq})

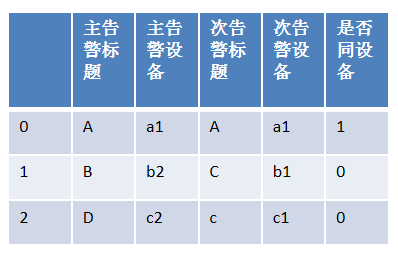
return patterns

合并不同地市生成的粗糙规则，对其进行规则泛化，具体思路如下：

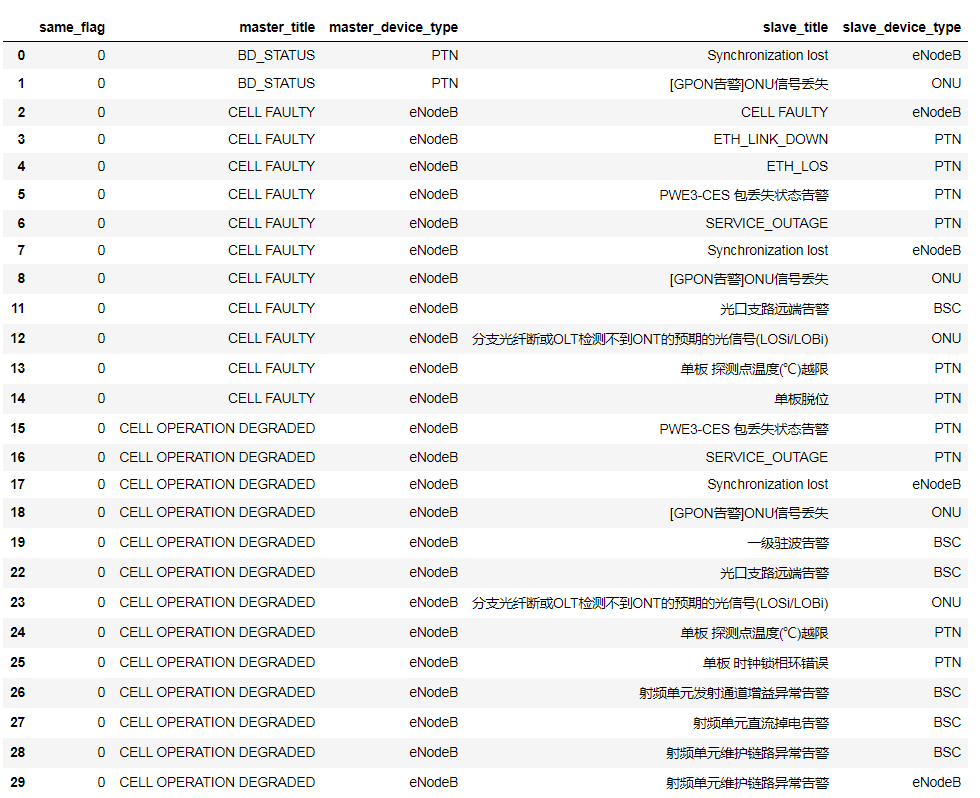
第一判断主次告警设备是否相同，打上标签。



对主告警标题、主告警设备类型、次告警标题、次告警设备类型、是否同设备进行合并，就进行规则泛化（groupby([...].sum())即可）。



字段为是否同设备(same\_flag)、主告警标题(master\_title)、主告警设备类型(master\_device\_type)、次告警标题(slave\_title)、次告警设备类型(slave\_device\_type)，结果如下：



def (rawrules):

aggfunc = {'support':'sum', 'master\_freq':'sum', 'slave\_freq':'sum','total\_itemsets':'sum'}

aggthis = rawrules.groupby(gpcols).agg(aggfunc)

aggthis['conf'] = aggthis['support'] / aggthis['master\_freq']

aggthis['lift'] = aggthis['support'] \* aggthis['total\_itemsets'] / ( aggthis['master\_freq'] \* aggthis['slave\_freq'])

#对于主次告警提升度大于0.7的规则进行保留

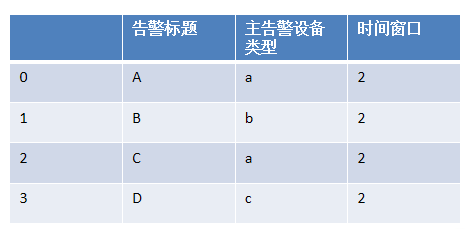
aggthis = aggthis[aggthis['lift']>0.7].copy()

aggthis.reset\_index(inplace=True)

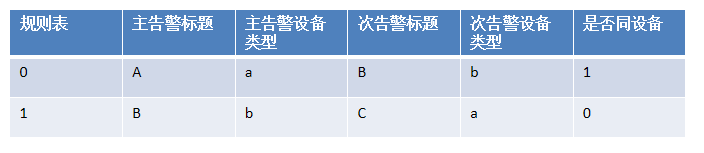
aggthis.to\_csv('MSNe\_Rules.csv', index=None, sep=',')

得到泛化后的规则之后，我们将规则用于实际告警数据的压缩，同样需要对实际告警数据进行聚类，过程同上；之后再应用泛化规则，具体思路如下：

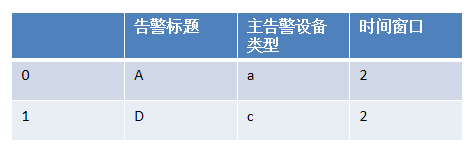
原始告警数据：



规则表



判断时间窗口内主次出现情况，若时间窗口内同事存在对应主次告警对，则删除次告警



具体代码如下：

def AlarmReduce():

HisRules = pd.read\_csv('/root/AI告警分析数据/batch/MSNe\_Rules.csv')

MasterId = set(HisRules['master\_title'].values)

SlaveId = set(HisRules['slave\_title'].values)

MasterSlaveAlarm=dict(HisRules.groupby('slave\_title').agg({'master\_title':'unique'}).reset\_index().values)

MasterSlaveAlarm = {key:set(values) for key, values in MasterSlaveAlarm.items()}

for idx, group in train.groupby('label'):

MasterTmp = set(group['alarm\_title'].values)&MasterId

SlaveTmp = set(group['alarm\_title'].values)&SlaveId

if MasterTmp and SlaveTmp:

for slave\_id in SlaveTmp:

if len(MasterTmp&MasterSlaveAlarm[slave\_id])>=1:

RemoveSlave.append(slave\_id)