数据探索之特征变量

在本部分主要对train、test中数据集进行探索，获取特征变量之间的相关关系。由于数据集结构相似，各个独立数据集之间没有相关关系，故以train/0.csv数据集为例进行探索。

整体情况

通过简略查看数据集和使用以下函数，可以看出数据集中不包含空值，除“time”外其余变量均为离散型，这也以为这所有数据均为有效数据，暂无需进行缺失值/异常值处理。

```
print(self.train.info())
print(self.train.describe(include="all"))
```

```
RangeIndex: 699 entries, 0 to 698
Data columns (total 5 columns):
Unnamed: 0      699 non-null int64
sysEname        699 non-null object
time            699 non-null object
triggername     699 non-null object
is_root         699 non-null int64
dtypes: int64(2), object(3)
memory usage: 27.4+ KB
```

特征拆分与编码

在该部分将对特征变量进行简单拆分/合并处理。由于“triggername”特征包含告警节点信息及告警信息，我们可以将其划分为“node_name”和“trigger”两列，便于进行数据探索。

```
    def add_feature(self):
        self.loaddata(0)
        name = []
        trigger = []
        for i in self.train["triggername"].tolist():
            name.append(i.split(" ", 1)[0])
            trigger.append(i.split(" ", 1)[1])
        self.train["node_name"] = name
        self.train["trigger"] = trigger
        self.train.drop(["triggername"], axis=1, inplace=True)
```

在此之后，观察数据集中其他特征，我们可以将“sysEname”与“node_name”精简化。考虑到系统名称和节点名称均为唯一标识，故可以删除这两列中不必要的文字说明，仅留下其编号作为标志。

查看数据集中的时间变量，可以发现每一个数据集代表一天中、一个小时内的告警信息，故可以进包留时间中的分、秒，便于表达。

最后，考虑对“trigger”变量进行特征编码，调用sklearn.preprocessing中的LabelEncoder库进行编码，具体实现如下：

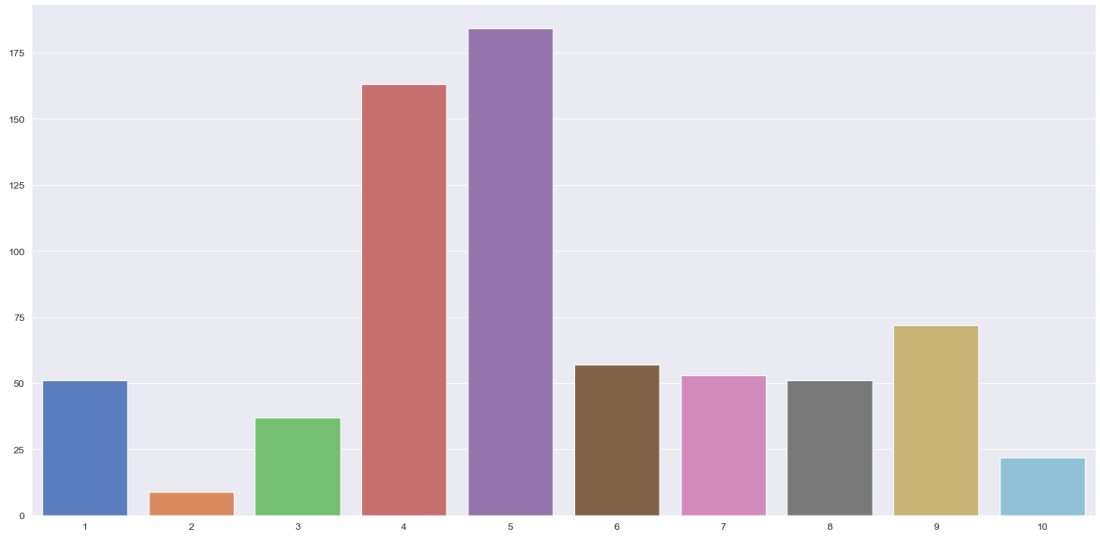
```
self.train["trigger"] = self.label.fit_transform(self.train["trigger"])
self.label_list = dict(zip(list(range(0, len(self.label.classes_))),
 self.label.classes_))

# self.label_list 存储编码与真实值的对应关系。
{0: '**** 请求延时大于5分钟', 1: '**application 停止运行', 2: 'FullGC平均耗时：
2118ms（大于阈值：1000ms）', 3: 'FullGC次数：32次（大于阈值：10次）', 4:
'HTTP:http://**********，慢响应（调用耗时超过1000ms）次数：309次（大于阈值：200次）',
5: 'JBOSS 8080端口的连接数大于400', 6: 'ping丢包率100%,服务器宕机', 7: 'ping丢包率当前
值为25%，持续2分钟平均大于20%', 8: 'sdd IO使用率持续30分钟大于90%', 9:
'url:http://node_22:80//访问失败', 10: 'url:http://node_73:80//访问失败', 11:
'url:http://node_85:80//访问失败', 12: '上CPU Steal Time持续5分钟超过10%', 13: '上
I/O等待负载大于50%', 14: '上I/O等待负载持续15分钟超过10%', 15: '分区剩余空间小于阀值10%',
16: '堆内存平均使用率：94.61%（大于阈值：90%）', 17: '日志中有No route to host信息',
18: '日志中有OutOfMemoryError信息', 19: '日志产生ERROR信息', 20: '硬盘Slot 00状态为
Failed', 21: '空闲CPU为1.52%,持续15分钟小于10%', 22: '空闲CPU为4.0%,持续15分钟小于
10%', 23: '空闲交换空间小于50%', 24: '端口6060通信异常', 25: '端口80通信异常', 26: '网
卡eth0 发送流量持续30分钟平均值大于700M', 27: '网卡eth0 接收流量持续10分钟平均值大于300M',
28: '网卡流量unknown'}
```

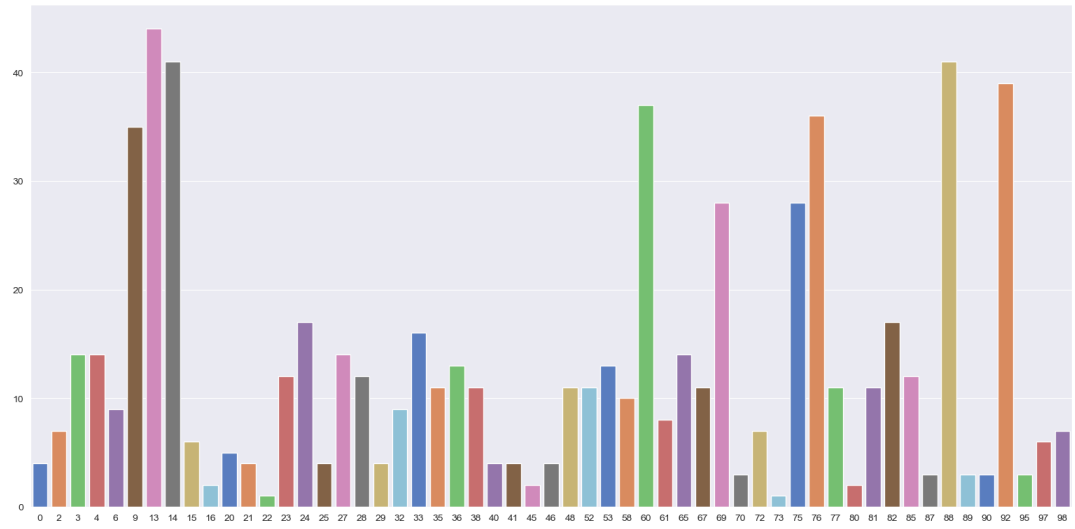
此时数据示例如下：

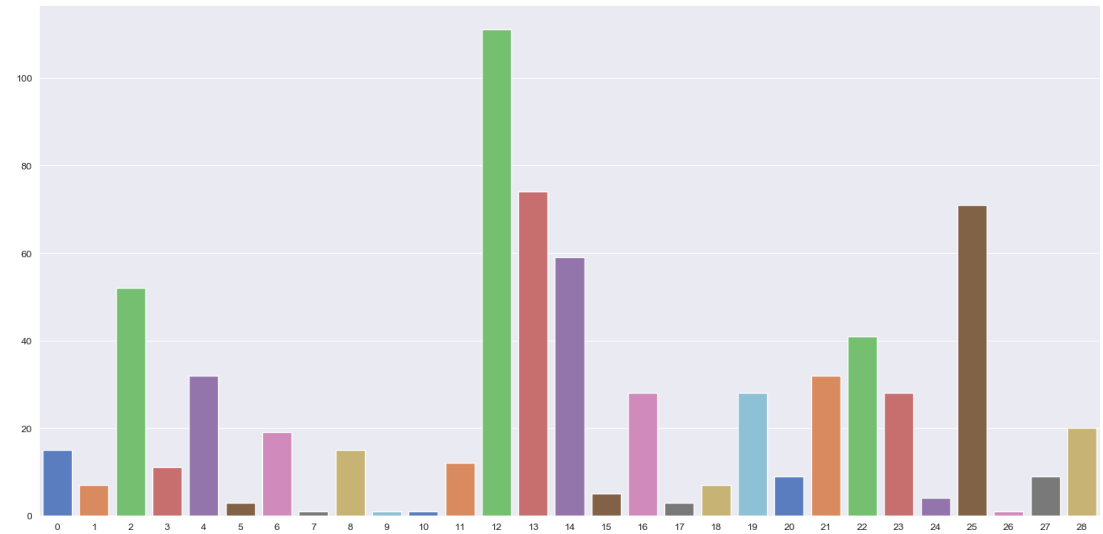
	Unnamed: 0	sysEname	time	is_root	node_name	trigger
0	0	8	14:02	0	87	4
1	1	9	14:03	0	92	12
2	2	9	14:04	0	75	23
3	3	5	14:05	1	60	25
4	4	7	14:05	0	53	4

单变量分析

由于数据集中特征较少，索引、时间列并无太多有效信息，故在这部分仅对“sysEname”、“node_name”、“trigger”进行分析，分别画出其柱形图。







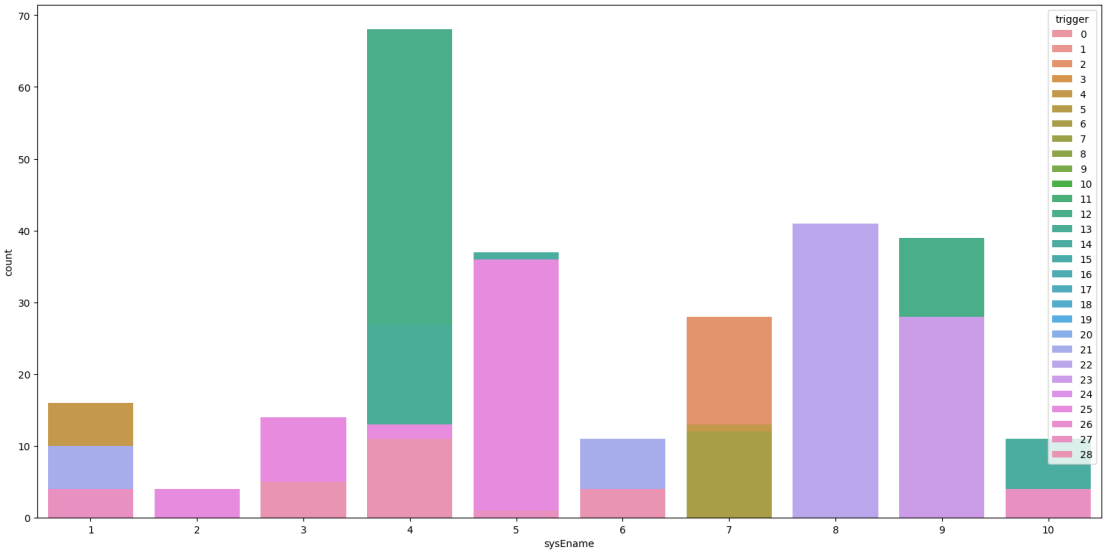
从上述图中可以看到，系统4和5、部分节点（9、13、14、88.....）的告警数据明显比其他系统多，而出现最多的告警信息为编号12。在此为多变量分析提供了思路：系统-节点、节点-告警信息、系统-告警信息等等。

多变量分析

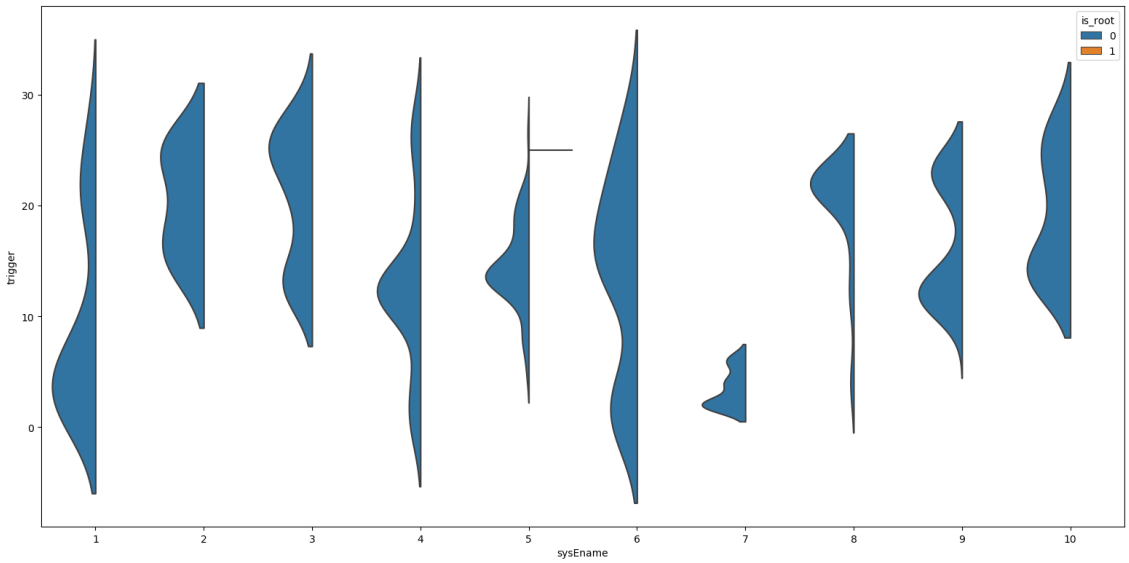
在单变量分析的基础上，我将对特征进行多变量分析。由于不同数据集中时间也是不相同的，所以本部分的探索主要集中于“sysEname”、“node_name”、“trigger”、“is_root”这四个特征。

- 系统-告警-根：

```
sns.countplot(x="sysEname", hue="trigger", data=self.train, dodge=False)
```

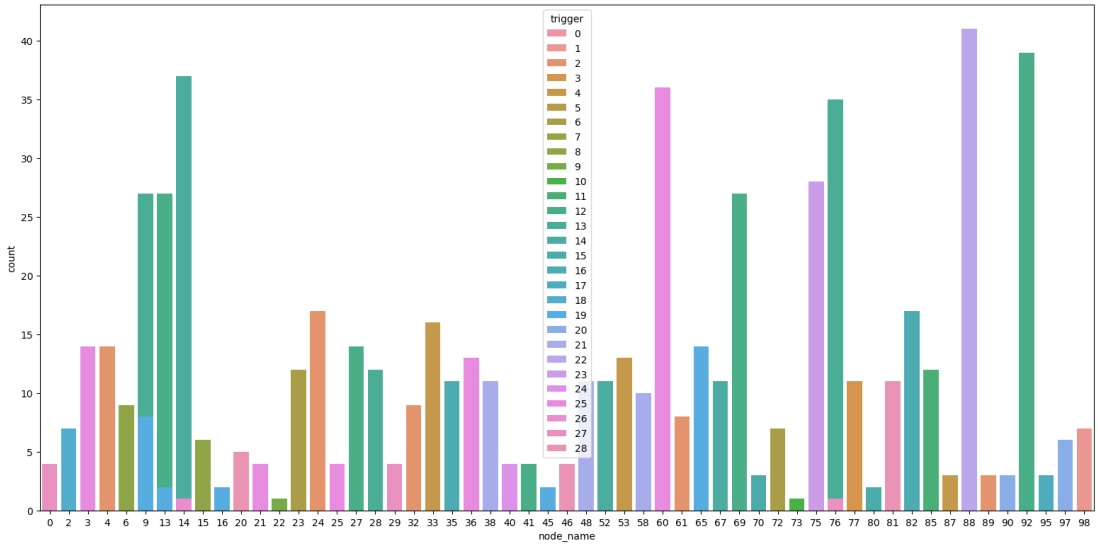



```
sns.violinplot(x="sysEname", y="trigger", data=self.train, hue="is_root",
split=True)
```

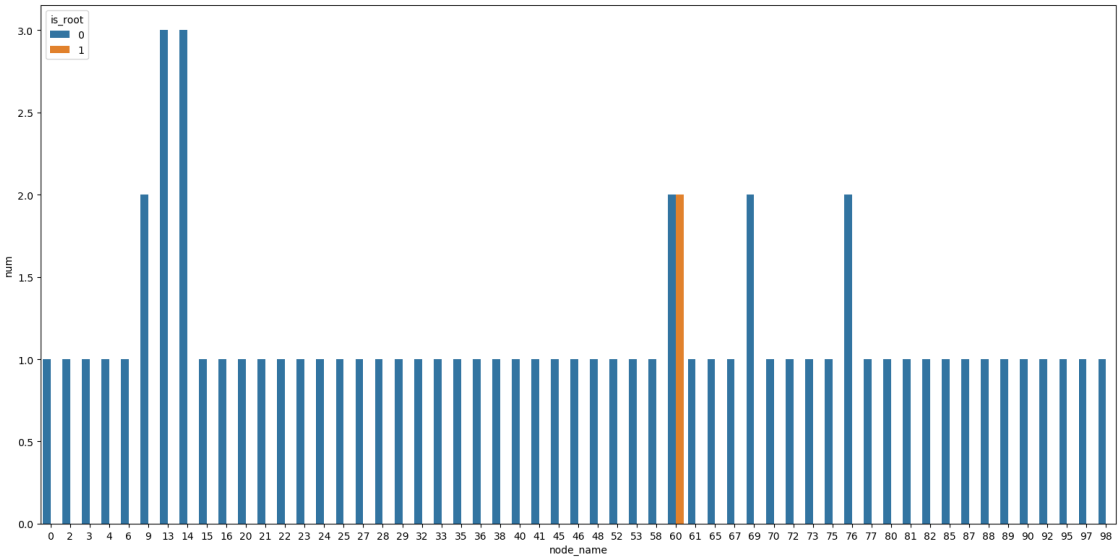


从上述图中可以看到，产生告警类型数量和根因所在的位置并无必然联系。

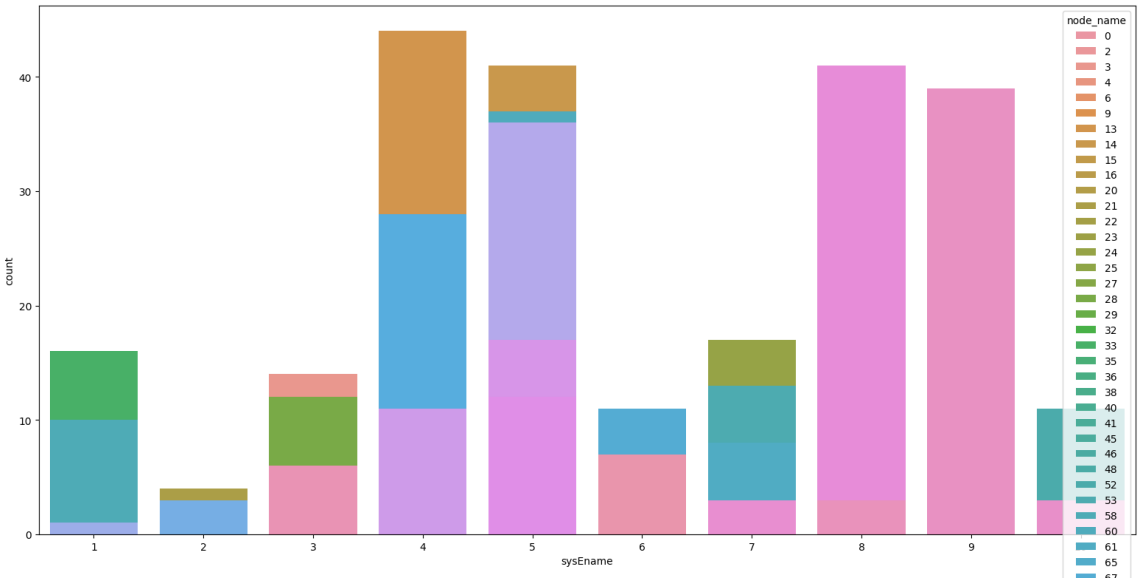
- 节点-告警-根：



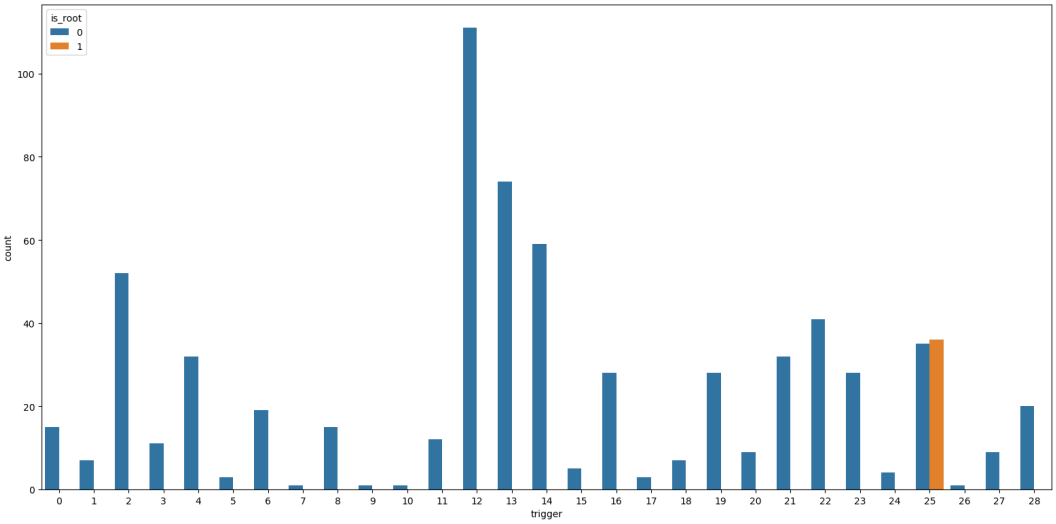
由图可见，一般一个节点仅会出现一种告警，而根因一般出现在有多种信息的节点之中。



- 系统-节点：



- 根-告警：



从中可以看到根因主要集中在一类告警信息中。

由于数据集的特征不适合做热力图，故数据探索-特征变量到此为止。

小结

从上述分析可知，根因告警时，其所在系统的节点受影响程度大，可导致其关联节点也可能产生告警。根因的造成可能来源于同类型告警，而此告警会引起根因出现不同类型告警。