# 代价敏感的即时缺陷预测：简单地无监督模型可能比有监督模型更好

这里只是做了十折交叉验证的实验。对于时间交叉验证以及跨项目验证没有实验，但是过程和算法都是一样的。

对于一个项目对应的数据集，先去掉不需要的两个属性transactionid，commitdate，然后调用util中的算法将数据集划分为10份，然后一份作为测试数据集test，其余的9份作为训练数据集train。把test和train传递给无监督和有监督模型进行处理。

## 1.无监督模型

### 1.1特征工程

1. 数据变换：[问题1]

对train和test中的lt、la、ld属性进行重新计算。

train["lt"]=train["lt"]\*train["nf"]

train["la"]=train["la"]\*train["lt"]

train["ld"]=train["ld"]\*train["lt"]

test["lt"]=test["lt"]\*test["nf"]

test["la"]=test["la"]\*test["lt"]

test["ld"]=test["ld"]\*test["lt"]

2. 构造新的特征：

构造的特征是为了计算模型的评价指标

（1）构造loc特征：利用util.py模块中的effortCal方法计算train和test数据的每一个样本（即每一行）的代码行数，记为train\_churn、test\_churn。这个属性最终是要用来衡量模型的性能的。

（2）构造bugdensity特征：利用上面计算出来的loc特征得到bugdensity特征，方法就是

train\_bugdensity=train["bug"]/(train\_churn+1)，

test\_bugdensity=test["bug"]/(test\_churn+1)，

这里加1是为了防止有的样本的代码行为0（我是这么理解的）。

（3）构造bug特征：就是简单的把train和test中的属性赋值给train\_bug，test\_bug。

（4）构造nuc特征：这里是利用train的npt和nf属性相乘得到train[“nuc”]，test的npt和nf属性相乘得到test[“nuc”]。

经过特征工程之后，train中的属性由15个变成了16个，新增了一个nuc属性，其中lt、la、ld、属性的数据都是经过数据变换的，其余属性的值没有发生变化。

test和train的操作是一样的。

由train和test各自产生了三个新变量train\_churn，train\_bugdensity和train\_bug，test\_churn，test\_bugdensity，test\_bug。这几个属性的产生全部都是为了计算模型的评价指标，以进行模型的比较进而评价模型的好坏，和建模及预测无关。

经过特征工程之后就可以用得到的train进行建模，然后用得到的模型对test进行预测，最后对test的预测结果计算Popt和ACC进行评价模型的性能。

### 1.2建模和预测：

train和test中的每一个属性取倒数就是一个算法，train和test中有16个属性，但是我们只要ns、nm、nf、entropy、lt、fix、ndev、pd、nuc、exp、rexp、sexp这几个属性。

### 1.3评价（计算Popt和ACC）：

用模型和预测中得到的y\_test和特征工程中的test\_churn，test\_bugdensity，test\_bug属性构造pd.DataFrame：

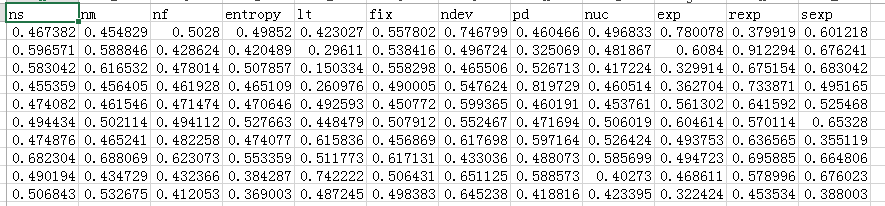
y\_test=pd.DataFrame({"per":y\_test,"num":test\_bug.values,"density":test\_bugdensity.values,"loc":test\_churn})，传到util.py模块中的Popt\_ACC函数中计算Popt和ACC。

Popt:

1. 对y\_test中的per进行逆序排序，得到新的y\_test，然后计算y\_test对应的面积，得到随机的面积mdl.
2. 对y\_test中的per进行逆序排序，density进行逆序排序, loc进行正序排序,得到新的y\_test，然后计算y\_test对应的面积，得到最好情况的面积opt.
3. 对y\_test中的per进行逆序排序，density进行正序排序, loc进行逆序排序,得到新的y\_test，然后计算y\_test对应的面积，得到最坏情况的面积wst.
4. 计算Popt=1-(opt-mdl)/(opt-wst)。

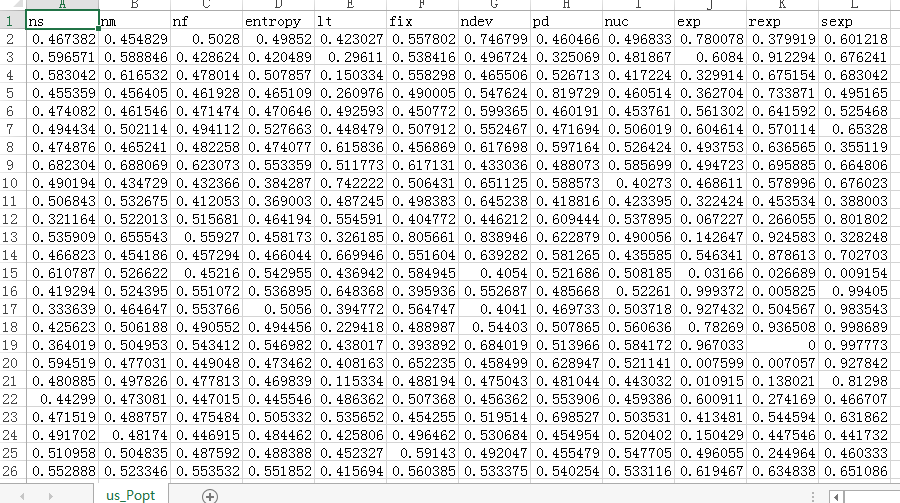
ACC:这里计算ACC主要采用的就是简单的划分法，不在这里进行叙述。

至此，得到了bugzilla项目的10次Popt和ACC的结果，保存到文件us\_bugzilla\_Popt.csv和us\_bugzilla\_ACC.csv中。



us\_bugzilla\_Popt.csv图

这里是对一个项目进行评估的方法，多个项目只需要一个for循环即可。对所有6个项目进行上述操作之后，得到的结果保存到文件us\_Popt.csv和us\_ ACC.csv中。因为每个项目都有10次结果，每个项目都采用了12种算法，所以us\_Popt.csv和us\_ ACC.csv中有6\*10行，12列。前10行是bugzilla项目中采用了12种算法得到的结果，其余的类推。



us\_Popt图

### 1.4问题：

1.为什么要改变着四个属性的值。

## 2.有监督模型

以bugzilla项目为例进行介绍。

### 2.1特征工程：

1.样本不均衡：

有个参数sample来决定是否进行采样处理来解决样本不均衡问题，这里的解决样本不均衡的方案是使缺陷变更和无缺陷变更的数量保持一致。util.py中的doSample方法。

2.构造新的特征：

构造的特征是为了计算模型的评价指标

（1）构造loc特征：利用util.py模块中的effortCal方法计算train和test数据的每一个样本（即每一行）的代码行数，记为train\_churn、test\_churn。这个属性最终是要用来衡量模型的性能的。

（2）构造bugdensity特征：利用上面计算出来的loc特征得到bugdensity特征，方法就是

train\_bugdensity=train["bug"]/(train\_churn+1)，

test\_bugdensity=test["bug"]/(test\_churn+1)，

这里加1是为了防止有的样本的代码行为0（我是这么理解的）。

（3）构造bug特征：就是简单的把train和test中的属性赋值给train\_bug，test\_bug。

3.数据变换：[问题2]

对数据进行log2变换，train和test中都有15个属性，在数据变换中去掉了5个属性 [“ns”,”nm”,”nf”,”fix”,“bug”]，只对剩下的10个属性进行数据变换。

4.相关性检查：

应用相关性检查，我们能看到rexp与exp的相关系数是0.928，nm与ns的相关系数是0.87，高度相关了，所以去掉高度相关的rexp和nm属性。

5.删除不需要的特征：

（1）如果一个属性的方差或者标准差为0，这也就意味着这个属性没有波动，或者说这个属性没有信息，这种属性是不需要的。

（2）删除la,ld两个属性。

经过特征工程之后，train中的属性由15个变成了11个['ns', 'nf', 'entropy', 'lt', 'fix', 'ndev', 'pd', 'npt', 'exp', 'sexp', 'bug']，去掉了nm，rexp（因为高度相关），la，ld（主动删除），其中["lt","exp" ,"sexp","ndev","pd","npt","entropy"]属性的数据都是经过log2变换的，ns、nf属性的值没有发生变化。

test中的属性还是15个。只不过la、ld、lt、exp、rexp、sexp、ndev、pd、npt、entropy属性中的值是经过log2变换的。

由train和test各自产生了三个新变量train\_churn，train\_bugdensity和train\_bug，test\_churn，test\_bugdensity，test\_bug。这几个属性的产生全部都是为了计算模型的评价指标，以进行模型的比较进而评价模型的好坏，和建模及预测无关。

经过特征工程之后就可以用得到的train进行建模，然后用得到的模型对test进行预测，最后对test的预测结果计算Popt和ACC进行评价模型的性能。

### 2.2建模和预测：[问题3]

这里用了sklearn中常用于二分类的几个经典模型：

nb=GaussianNB()

knn=KNeighborsClassifier()

dt=tree.DecisionTreeClassifier()

svc=SVC(kernel="rbf",probability=True)

rfc=RandomForestClassifier(n\_estimators=8)

gbtc=GradientBoostingClassifier()

adtc=AdaBoostClassifier()

但是没有用lr=LogisticRegression(penalty='l2')这个模型，主要的原因是用这个模型得到的预测结果y\_train和y\_test的值都是各不相同的，这就导致了在util.py模块中的computePopt函数中

mdl=computeArea(data\_mdl)，

opt=computeArea(data\_opt)，

wst=computeArea(data\_wst)

得到的mdl、opt、wst的结果完全相同，从而使得Popt=1-(opt-mdl)/(opt-wst)中的分母为0，导致错误。

为了防止别的几个二分类算法也或多或少的存在这些问题，我就把预测结果y\_train和y\_test的值全部保留5位小数。

### 2.3评价（计算Popt和ACC）：

用模型和预测中得到的y\_test和特征工程中的test\_churn，test\_bugdensity，test\_bug属性构造pd.DataFrame：

y\_test=pd.DataFrame({"per":y\_test,"num":test\_bug.values,"density":test\_bugdensity.values,"loc":test\_churn})，传到util.py模块中的Popt\_ACC函数中计算Popt和ACC。

Popt:

对y\_test中的per进行逆序排序，得到新的y\_test，然后计算y\_test对应的面积，得到随机的面积mdl.

对y\_test中的per进行逆序排序，density进行逆序排序, loc进行正序排序,得到新的y\_test，然后计算y\_test对应的面积，得到最好情况的面积opt.

对y\_test中的per进行逆序排序，density进行正序排序, loc进行逆序排序,得到新的y\_test，然后计算y\_test对应的面积，得到最坏情况的面积wst.

计算Popt=1-(opt-mdl)/(opt-wst)。

ACC:这里计算ACC主要采用的就是简单的划分法，不在这里进行叙述。

至此，得到了bugzilla项目的10次Popt和ACC的结果，保存到文件s\_bugzilla\_Popt.csv

和s\_bugzilla\_ACC.csv中。

这里是对一个项目进行评估的方法，多个项目只需要一个for循环即可。对所有6个项目进行上述操作之后，得到的结果保存到文件s\_Popt.csv和s\_ ACC.csv中。因为每个项目都有10次结果，每个项目都采用了7种算法，所以s\_Popt.csv和s\_ ACC.csv中有6\*10行，7列。前10行是bugzilla项目中采用了7种算法得到的结果，其余的类推。

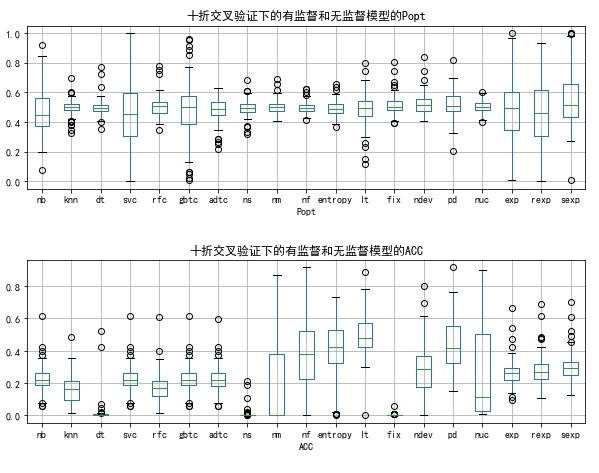
### 2.4问题：

2.为什么不进行标准化，而进行数据变换。去掉的5个属性中的fix和bug去掉是很好理解的，但是为什么“ns”,”nm”,”nf”,这三个变量也要去掉。

3.这几个模型的参数要怎么进行调整，这里所有模型全部采用默认的参数。

## 3.结果展示

从sqe/output/unsuprised/us\_Popt.csv和sqe/output/ unsuprised/us\_ACC.csv读取无监督模型得到的us\_Popt和us\_ACC，从sqe/output/suprised/s\_Popt.csv和sqe/output/unsuprised/s\_ACC.csv读取监督模型得到的s\_Popt和s\_ACC，然后在一幅图中画出它们的盒图。结果如图所示：



从图中可以看出在Popt上有监督和无监督模型是不行上下的，在ACC上，无监督模型比必有监督好，但是同时无监督模型的波动范围也比较大。