**PR预测阶段总结**

**写在前面：目前的训练方法为，以30天为一个round, gousios 与 ayonel 均如此，训练集数据集完全相同**

# gousios分析

## 属性

目前有11个属性

* num\_commits: **pr中的commit数**
* src\_churn： **pr中的代码更改行数**
* test\_churn: **pr中的测试代码行数**
* files\_changes **pr中的文件数**
* sloc **项目目前的总代码行数**
* team\_size **项目前3个月的活跃贡献者数**
* perc\_ext\_contribs **项目前3个月从pr中接受到的commit比例**
* commits\_files\_touched **项目前3个月的包含该pr中文件的commit数**
* test\_lines\_per\_kloc **代码测试覆盖比例**
* prev\_pullreqs **pr提交者历史的提交数**
* requester\_succ\_rate **pr提交者的提交通过率**

## 分析

经过我这几天的分析，我对pr预测这个题目的总结如下：

**这是一个不平衡分类问题。**

不平衡分类问题会造成一个问题，假设我有100个样本，其中95个都是正类，5个都是负类。那如果一个模型经过训练后，将这100个样本都预测成正类，那么我们最终得出的准确率为**95%**。但这个模型明显没有泛化能力，所以**对于不平衡数据集的分类问题，一般都不会单纯拿准确率来做度量指标。**

那我们现在来看gousios的方法，我将数据集与ayonel方法完全对齐后，目前的结果为：

|  |  |
| --- | --- |
| 项目 | 准确率 |
| Baystation12 | 0.86684048662 |
| bitcoin | 0.694662165672 |
| bootstrap | 0.754666607829 |
| brackets | 0.767460731561 |
| cakephp | 0.755835226082 |
| cdnjs | 0.748821021857 |
| cocos2d-x | 0.808409585286 |
| core | 0.808010417434 |
| dmd | 0.798890915036 |
| homebrew-cask | 0.770428148731 |
| ipython | 0.778380777884 |
| joomla-cms | 0.753149749887 |
| katello | 0.878233602066 |
| metasploit-framework | 0.788468339084 |
| openmicroscopy | 0.828993696165 |
| ppsspp | 0.856988822836 |
| puppet | 0.699894698614 |
| rails | 0.679733143005 |
| scala | 0.725426882713 |
| symfony | 0.696911156899 |
| sympy | 0.698748495339 |
| tgstation | 0.757299737221 |
| zendframework | 0.75663401033 |

目前所有项目的**平均准确率为0.768386**

这个结果看起来还行，但是存在**一个问题：**

**如果我使用一个我自己造的属性，该属性对所有pr的值都一样，比如设为0，之后用这个模型去进行分类，最后得出的准确率为0.752。**

可以看出，最终效果只提升了**1%**

这种做法其实跟将所有预测结果设为样本数较多的那一类 相同，我们来看一下数据，所有项目较多的那一类平均占比为**：0.746462**

那为什么不是0.752呢，因为我们的0.752训练的时候是30天为一个轮次，每个轮次的较多类样本比例取一个平均，所以与0.752有细微的区别。

结论是gousios的模型其实对于基准提升了（1%），这不是一个很好的模型。我又详细输出了gousios对每个pr的预测结果，并做了统计，我发现gousios的模型其实是把所有待分类pr都预测成了训练集中样本数较多的那一类（这种类我们称为正类），以下是详细统计：

|  |  |
| --- | --- |
| 项目 | 预测为正类的样本所占比例 |
| Baystation12 | 0.99941 |
| bitcoin | 0.99893 |
| bootstrap | 0.99917 |
| brackets | 0.99893 |
| cakephp | 0.99883 |
| cdnjs | 0.99891 |
| cocos2d-x | 0.99952 |
| core | 0.9996 |
| dmd | 0.99896 |
| homebrew-cask | 0.99978 |
| ipython | 0.99865 |
| joomla-cms | 0.99936 |
| katello | 0.99919 |
| metasploit-framework | 0.99912 |
| openmicroscopy | 0.99886 |
| ppsspp | 0.99912 |
| puppet | 0.99878 |
| rails | 0.9996 |
| scala | 0.99901 |
| symfony | 0.99947 |
| sympy | 0.99862 |
| tgstation | 0.99968 |
| zendframework | 0.99893 |

可以看到gousios的模型其实预测分类简单地分到了训练集的正类中，全是99+%。

所以可以看出，**gousios的方法并不是一个泛化能力强，优秀的模型，严格一点，这个模型完全不能用**

**（当然这与我们只截取了gousios的初始属性有关， gousios的初始属性其实并不优秀）**

# ayonel分析

先总结一句：目前优化方法还在于提出新的优秀的特征，我对之前的特征做了衰减之后，衰减之后的特征表现反而不好。

## 属性

（数值属性，按照重要性排序）

|  |  |  |
| --- | --- | --- |
| 属性名 | 解释 | 重要度 |
| last\_10\_pr\_rejected | 该pr之前的10个pr中拒绝的pr数 | 0.766663 |
| history\_pass\_pr\_num | 该pr提交者之前提交的通过pr数 | 0.766572 |
| last\_10\_pr\_merged | 该pr之前的10个pr中通过的pr数 | 0.764145 |
| commits | pr中的commits数 | 0.755164 |
| history\_commit\_num | 该pr提交者之前提交的pr数 | 0.752607 |
| null | 自造属性，值全为0，作为基准线 | 0.751923 |
| files\_changes | pr中文件数 | 0.750753 |
| commits\_files\_touched | 项目前3个月的包含该pr中文件的commit数 | 0.747506 |
| history\_commit\_passrate | pr提交者之前的pr通过率 | 0.743768 |
| team\_size | 项目前3个月的活跃贡献者数 | 0.743457 |
| history\_commit\_review\_time | 该提交者之前的平均审核通过时间（单位为小时） | 0.741614 |
| src\_deletion | 代码删除行数 | 0.740302 |
| text\_code\_proportion | pr文本中的代码占比 | 0.736792 |
| recent\_1\_month\_project\_pr\_num | 最近一个月项目的pr数 | 0.735597 |
| pr\_file\_rejected\_count | 历史中包含该pr的文件的pr拒绝数 | 0.735552 |
| src\_addition | 代码增加行数 | 0.733601 |
| pr\_file\_merged\_count | 历史中包含该pr的文件的pr通过数 | 0.732377 |
| src\_churn | 代码更改行数 | 0.729684 |
| pr\_file\_submitted\_count | 历史中包含该pr的文件的pr数 | 0.725718 |
| recent\_3\_month\_project\_pr\_num | 最近3个月项目的pr数 | 0.724423 |
| recent\_3\_month\_pr | 该提交者最近3个月的提交数 | 0.71954 |
| history\_pass\_pr\_num\_decay | 该提交者历史提交的通过pr数，衰减 | 0.717629 |
| pr\_file\_merged\_proportion | 历史中包含该pr的文件的pr通过占比 | 0.714855 |
| pr\_file\_rejected\_proportion | 历史中包含该pr的文件的pr拒绝占比 | 0.714338 |
| recent\_3\_month\_commit | 最近三个月项目接受的commits | 0.706288 |
| recent\_project\_passrate | 最近3个月项目的pr通过率 | 0.696543 |
| history\_pr\_num\_decay | 该提交者历史提交的pr数，衰减 | 0.688023 |
| pr\_file\_rejected\_count\_decay | 该提交者历史提交的拒绝pr数，衰减 | 0.687131 |
| perc\_ext\_contribs | 项目前3个月从pr中接受到的commit比例 | 0.681894 |
| pr\_file\_merged\_count\_decay | 历史中包含该pr的文件的pr通过数，衰减 | 0.681539 |
| body\_similarity\_merged | pr文本与前3个月通过pr的文本相似度 | 0.679058 |
| title\_similarity\_rejected | 标题与前3个月拒绝pr的文本相似度 | 0.676151 |
| pr\_file\_submitted\_count\_decay | 历史中包含该pr的文件的pr占比，衰减 | 0.674585 |
| recent\_3\_month\_project\_pr\_num\_decay | 最近3个月项目提交的pr数，衰减 | 0.674463 |
| title\_similarity\_merged | 标题与前3个月通过pr的文本相似度 | 0.67292 |
| last\_10\_pr\_merged\_decay | 该pr之前的10个pr中的通过数，衰减 | 0.672603 |
| recent\_1\_month\_project\_pr\_num\_decay | 最近1个月的项目接受pr数，衰减 | 0.67259 |
| body\_similarity\_rejected | body与前3个月拒绝pr的文本相似度 | 0.671277 |
| file\_similarity\_merged | 文件与前3个月通过pr的相似度 | 0.668442 |
| text\_similarity\_merged | 标题+body与前3个月通过pr的文本相似度 | 0.665557 |
| file\_similarity\_rejected | 文件与前3个月拒绝pr的相似度 | 0.664817 |
| text\_similarity\_rejected | 标题+body与前3个月拒绝pr的文本相似度 | 0.661867 |
| last\_10\_pr\_rejected\_decay | 该pr之前10个pr中的拒绝数，衰减 | 0.658729 |

以上是数值属性

|  |  |
| --- | --- |
| is\_reviewer\_commit | 是否是核心成员提交 |
| has\_test | 是否包含测试代码 |
| text\_forward\_link | 文本中是否包含外链 |
| last\_pr | 上一个pr是拒绝还是接受 |
| has\_body | 是否包含body |

以上是布尔属性。

还有一个类别属性：

|  |  |
| --- | --- |
| week | 表示提交时间是星期几 |

**总结：这些属性中，我上一次新提出来的属性表现很好，对属性衰减之后的属性表现不好。**

## 目前效果

目前采用前10名数值属性，3个bool属性以及一个分类属性，利用xgboost不调参进行分类后的结果大致在**80.1-80.3%**，以下为其中一个中间结果：

|  |  |
| --- | --- |
| 项目 | 准确率 |
| Baystation12 | 0.878589 |
| bitcoin | 0.728888 |
| bootstrap | 0.800349 |
| brackets | 0.813307 |
| cakephp | 0.761931 |
| cdnjs | 0.7911 |
| cocos2d-x | 0.830056 |
| core | 0.838653 |
| dmd | 0.862221 |
| homebrew-cask | 0.832795 |
| ipython | 0.789903 |
| joomla-cms | 0.795897 |
| katello | 0.895428 |
| metasploit-framework | 0.814255 |
| openmicroscopy | 0.848649 |
| ppsspp | 0.878734 |
| puppet | 0.74556 |
| rails | 0.731959 |
| scala | 0.757614 |
| symfony | 0.750787 |
| sympy | 0.736067 |
| tgstation | 0.779797 |
| zendframework | 0.775581 |
| 平均 | 0.801657 |

我们按照对gousios做的一样，统计一下预测为正类的样本所占比例：

|  |  |
| --- | --- |
| 项目 | 预测为正类的样本所占比例 |
| Baystation12 | 0.9751736704247808 |
| bitcoin | 0.8221797323135756 |
| bootstrap | 0.6479926728743703 |
| brackets | 0.9563758389261745 |
| cakephp | 0.776410998552822 |
| cdnjs | 0.7583378895571351 |
| cocos2d-x | 0.9231833910034603 |
| core | 0.9563698177353858 |
| dmd | 0.9809296781883194 |
| homebrew-cask | 0.6258503401360545 |
| ipython | 0.9166317553414327 |
| joomla-cms | 0.7178277697145622 |
| katello | 0.9903481558083419 |
| metasploit-framework | 0.866937531742001 |
| openmicroscopy | 0.9781733418097055 |
| ppsspp | 0.975383286547182 |
| puppet | 0.7759594778881745 |
| rails | 0.825244487638948 |
| scala | 0.9180391397016082 |
| symfony | 0.6592942345924453 |
| sympy | 0.8315103071098022 |
| tgstation | 0.9180924043309364 |
| zendframework | 0.5909172179510102 |

**平均比例为0.842920137**

**可以看出要比gousios的99+%要好很多，说明我的模型要比gousios具有更好的泛化能力以及解释能力，是更好的模型。**

# 评估指标

对于不平衡的数据集分类问题，通常都不会拿准确率AUC来作为评估指标，即使是gousios的原文中都采用了准确率AUC，精准率Presicion以及召回率recall。

不平衡分类问题也会采用这三种，以及F1

我也分别对gousios以及ayonel进行了以上precision，recall，f1指标的计算，结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 组织 | g\_precision | g\_recall | g\_f1 | a\_precisi | a\_recall | a\_f1 |
| Baystation12 | 0.894775 | 0.969677 | 0.930721 | 0.892957 | 0.992355 | 0.940036 |
| bitcoin | 0.742561 | 0.850972 | 0.793079 | 0.757082 | 0.904521 | 0.824261 |
| twbs | 0.806433 | 0.731335 | 0.76705 | 0.779962 | 0.881188 | 0.827491 |
| adobe | 0.82666 | 0.927792 | 0.874312 | 0.827571 | 0.982338 | 0.898337 |
| cakephp | 0.838232 | 0.914707 | 0.874801 | 0.860433 | 0.887622 | 0.873816 |
| cdnjs | 0.822555 | 0.884872 | 0.852576 | 0.865892 | 0.885615 | 0.875643 |
| cocos2d | 0.82057 | 0.930128 | 0.871921 | 0.828836 | 0.965551 | 0.891985 |
| owncloud | 0.862431 | 0.93191 | 0.895825 | 0.861548 | 0.979551 | 0.916767 |
| dlang | 0.883154 | 0.904181 | 0.893544 | 0.885143 | 0.986063 | 0.932882 |
| caskroom | 0.779568 | 0.845969 | 0.811412 | 0.848844 | 0.90082 | 0.87406 |
| ipython | 0.8496 | 0.93698 | 0.891153 | 0.858896 | 0.952861 | 0.903442 |
| joomla | 0.78043 | 0.772807 | 0.7766 | 0.764049 | 0.895097 | 0.824397 |
| Katello | 0.898251 | 0.96937 | 0.932456 | 0.897628 | 0.991524 | 0.942243 |
| rapid7 | 0.842158 | 0.904253 | 0.872101 | 0.855044 | 0.932892 | 0.892273 |
| openmicroscopy | 0.889845 | 0.97039 | 0.928374 | 0.887611 | 0.984834 | 0.933699 |
| hrydgard | 0.875204 | 0.94295 | 0.907815 | 0.868823 | 0.985423 | 0.923457 |
| puppetlabs | 0.743325 | 0.865789 | 0.799896 | 0.780605 | 0.86831 | 0.822125 |
| rails | 0.722318 | 0.79958 | 0.758988 | 0.733054 | 0.910012 | 0.812004 |
| scala | 0.771985 | 0.890093 | 0.826842 | 0.772573 | 0.940402 | 0.848266 |
| symfony | 0.695128 | 0.775803 | 0.733253 | 0.725634 | 0.859597 | 0.786955 |
| sympy | 0.760952 | 0.864748 | 0.809537 | 0.782851 | 0.888058 | 0.832142 |
| tgstation | 0.802204 | 0.917539 | 0.856004 | 0.80537 | 0.954711 | 0.873705 |
| zendframework | 0.744465 | 0.739011 | 0.741728 | 0.778349 | 0.785714 | 0.782014 |
|  | **0.810991** | **0.880037** | **0.843478** | **0.822555** | **0.926742** | **0.870957** |

可以看出精准率precision差距不大，根据precision定义，出现这种结果也正常，因为gousios的将所有样本都预测成了正类。而precision的定义是：

**(TP/TP+FP)**

**TP:实际为正类，预测也为正类**

**FP:本来为负类，预测为正类**

总结下就是分子为：实际是正类预测也为正类的样本数

分母为不管正负类，都预测成正类的样本数。

**recall以及f1值提升程度较大。**

对于不平衡数据集更常用的评估方式是ROC曲线，我们之后可以用来评估。

# 总结

**我觉得我们现在没有必要规避数据集不平衡的问题，目前gousios虽然在准确率上看起来很高，但是这个模型经不起考虑，是一个糟糕的模型，我们可以在论文中提到这一点，而我们的模型泛化能力确实要比它好。**

**我现在凭个人感觉，这个课题的上限最多提升1%，这还需要不停地去寻找新的属性，这次找的几个属性排名都很靠前。如果属性调不上去，看看能否在模型调参以及模型融合上优化一点，但我现在已经提了差不多快50个属性，确实感觉没多少可挖的了。**