黑白色的标志

中度可信度描述已自动生成

数据挖掘实验报告

|  |  |
| --- | --- |
| **实验日期：** | **2022.5.31** |
| **实验地点：** | **学武楼 A302** |
| **提交日期：** | **2022.6.26** |
| **学号 姓名：** | **22920192204246 刘赫昭**  **11920192203642 袁佳哲**  **34520192201612 谢健祥** |
| **专业年级：** | **软工2019级** |

# 一、人肝毒性数据集

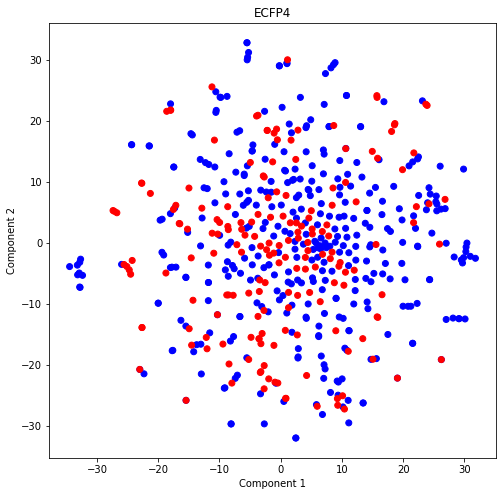
## 1、数据集：

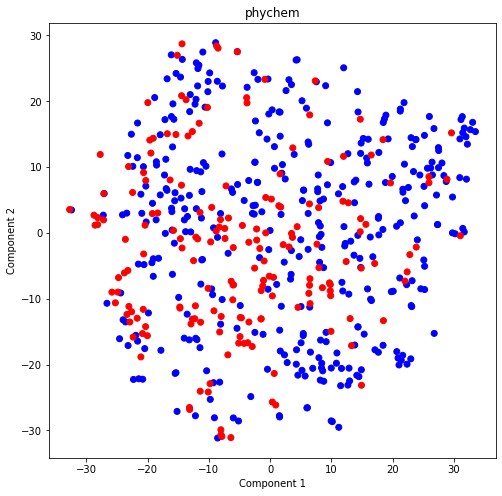
### （1）数据集总览

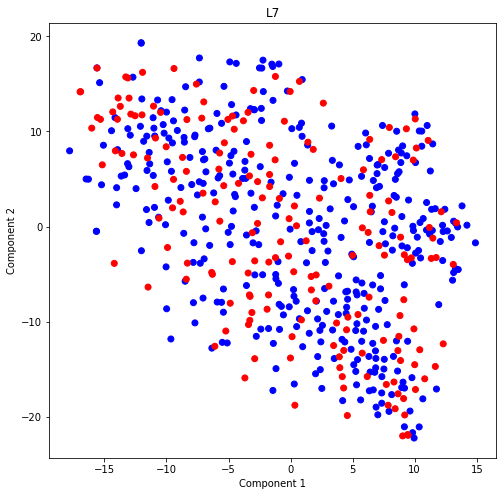
数据集总共分为三个版本，V1、V2、V3，其中V1的样本数是最多的，正样本1513个，负样本785个，但是属性只有ECFP4与phychem；V2版本的数据由ECFP4、phychem与L7 三类属性构成，样本数为554个；V3版本的数据由ECFP4、phychem与L5 三类属性构成，但是样本数只有252个，综上所述，V2数据集在样本数量与属性的完整性上都是较好的，所以我们决定在V2数据集上进行模型探索，再将得到的模型推广到其他两个数据集上。

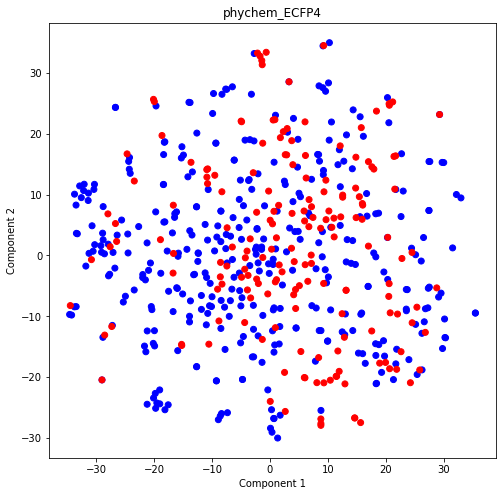
### （2）数据基本信息的查看：

首先将数据的各个属性使用TSNE算法降维成二维，并绘制出散点图：

****

****

****

****

从散点图的中可以看出，不同类型的属性中均无明显的聚簇，且两类数据夹杂在一起。使用sum函数求得每一个样本的ECFP4属性值的和，可以看出2000多维的属性中只有平均40个左右的1，其余全为0，ECFP4是十分稀疏的。

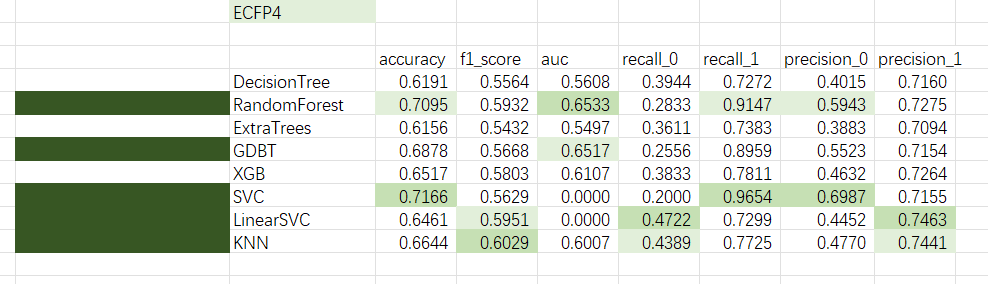
## 2、模型设计的基本思路：

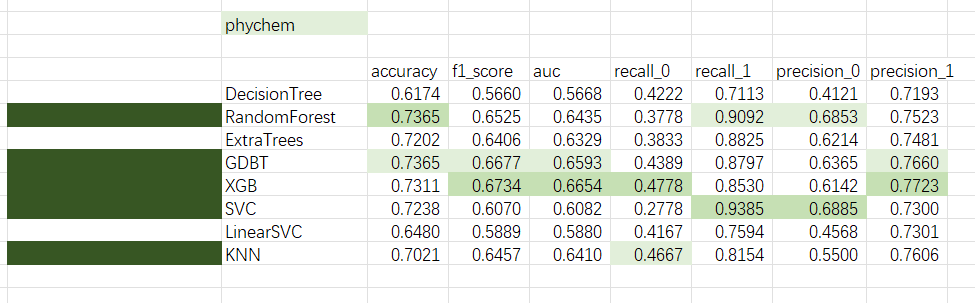
根据上述的数据基本信息，我们决定将不同的属性分开建模。对于每一个测试数据，先对其进行预分类，根据预分类的结果，再选择在此类上表现好的分类器对其进行分类。具体实现为尝试先对样本进行聚类，若判断为小类，则选择小类效果好的分类器对样本进行判断，否则使用大类的优势分类器进行判断。

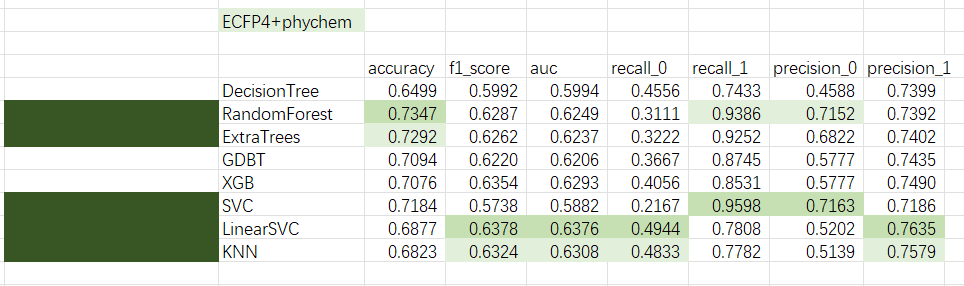
### （1）基分类器的选择：

首先将不同类型的属性直接输入各个算法，查看各种算法在不同属性上表现的基本情况，根据结果选出在某些指标突出的分类器作为集成学习的基分类器。

结果如下：



****

****

其中绿色底色的为突出表现的指标，左边绿色标记表示这类属性使用标记的分类器训练，作为集成的基分类器。每种属性选择了5种，总共有15个基分类器参与投票。

### （2）预分类模型的探索：

寻找最佳的聚类算法对样本进行预分类。目标选出表现最好的模型与数据的组合。

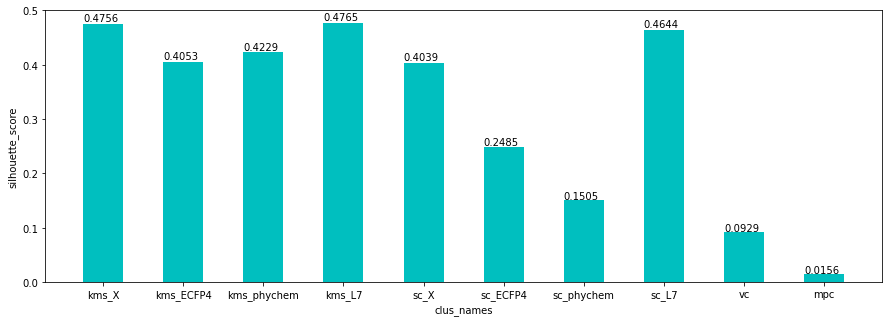
①首先考虑的是KMeans算法，根据以往的经验将数据预先降维2维之后再进行KMeans聚类分析。训练、测试完成之后使用silhouette\_score评价其聚类结果的好坏。

②根据上面绘制出的数据二维散点图以及sklearn文档的描述(Few clusters, even cluster size, non-flat geometry, transductive)，我们认为SpectralClustering能得到不错的结果，所以也将SpectralClustering的聚类结果进行了研究。同样使用silhouette\_score对其进行评价。

③此外，我们还尝试了在聚类时就使用投票算法，将不同数据建模的KMeans算法进行投票集成。使用ECFP4、phychem与L7训练三个基分类器，通过多数服从少数投票得到最终的结果。

④我们还尝试了基于距样本中心距离的自定义聚类函数，训练时计算出训练数据的几何中心。对于每一个测试数据，计算其与两个(共有两类标签)几何中心的距离，将它分为距离近的那一类。这样也不需要进行标签对齐。

将上述四种模型共10组成绩绘制成柱状图，结果如下：

****

综合上述尝试的结果，从柱状图中可以发现使用降维成2d的L7属性进行建模的KMeans算法能在聚类时得到最好的结果。所以在后续的模型中使用L7建模的KMeans算法进行预分类。

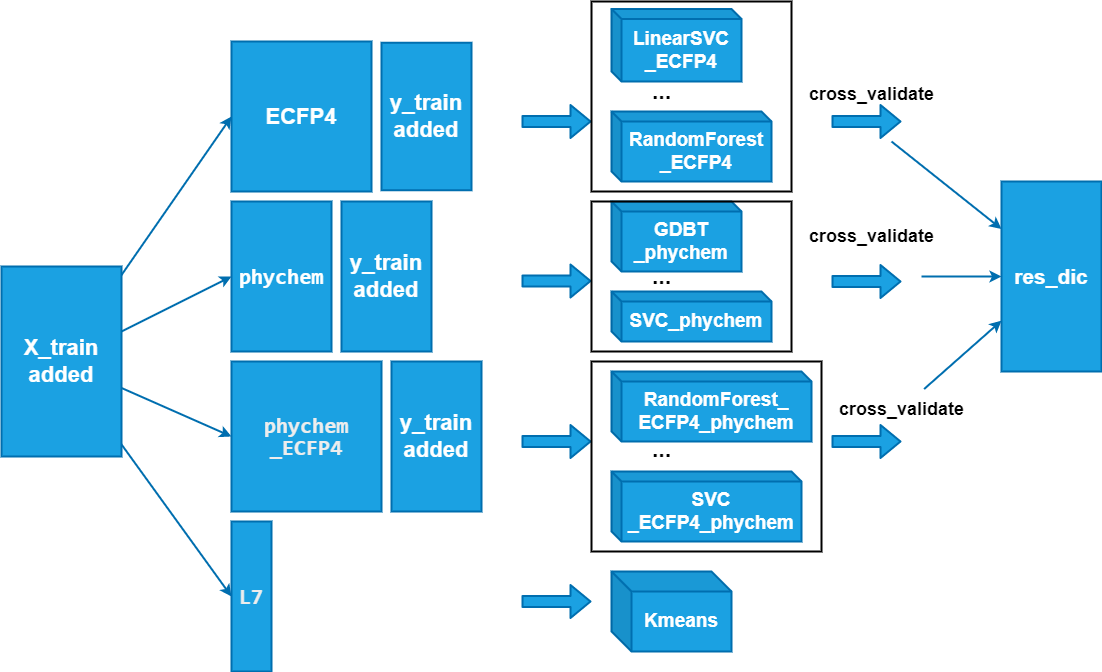
对于上述结果的原因，我们认为可能是：在上述的散点图中，只有L7属性的轮廓有条带状的倾向，其余属性的轮廓都大致呈圆形，条带状的数据分布是有利于聚类分析的。且各个属性的散点图轮廓均为凸图形，对于凸数据KMeans算法更能胜任。

## 3、模型的实现：

有了前面工作的基础，就可以开始实现模型了，我们先实现了一个简单的模型，之后再在它的基础上进行完善，大体过程为：预分类+投票-->预分类+投票+特征选择-->预分类+投票+特征选择+上采样-->预分类+投票+特征选择(自动选择最优阈值)+上采样(这个方案最终失败)

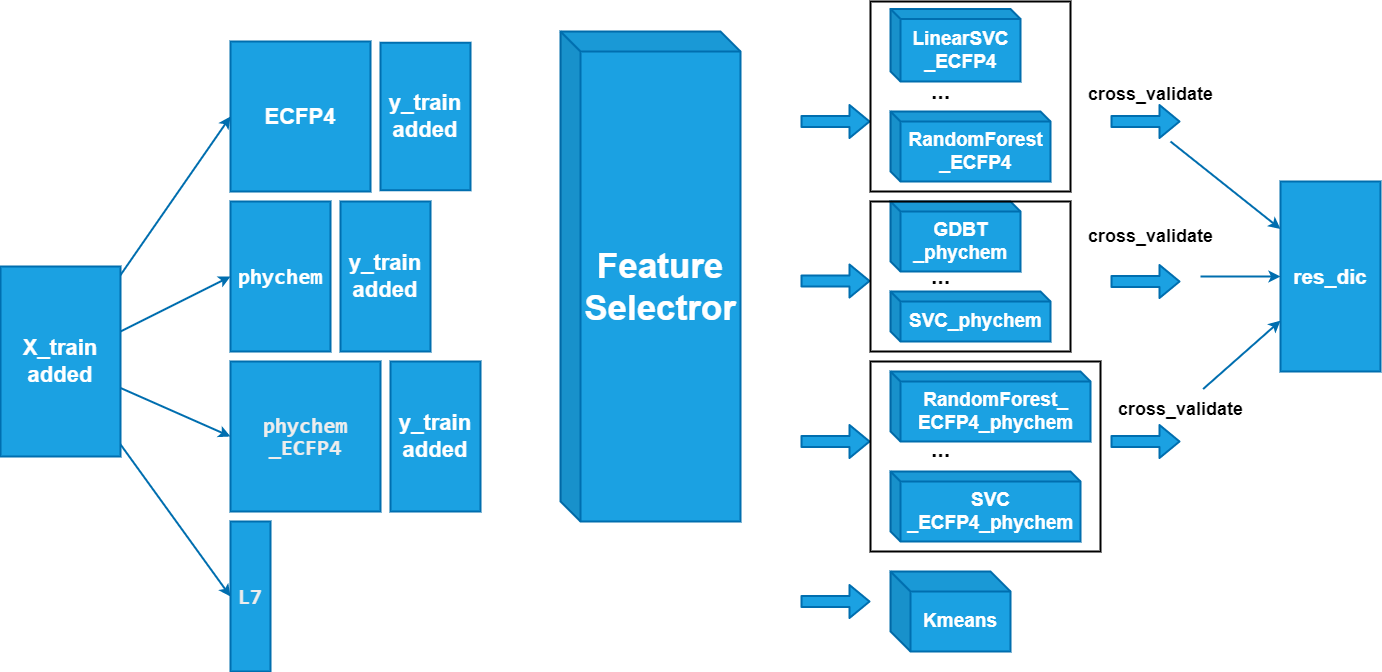
### （1）My\_vote\_clf

我们首先尝试了预分类+投票的模型，其训练流程如下：

****

### （2）My\_vote\_select\_clf

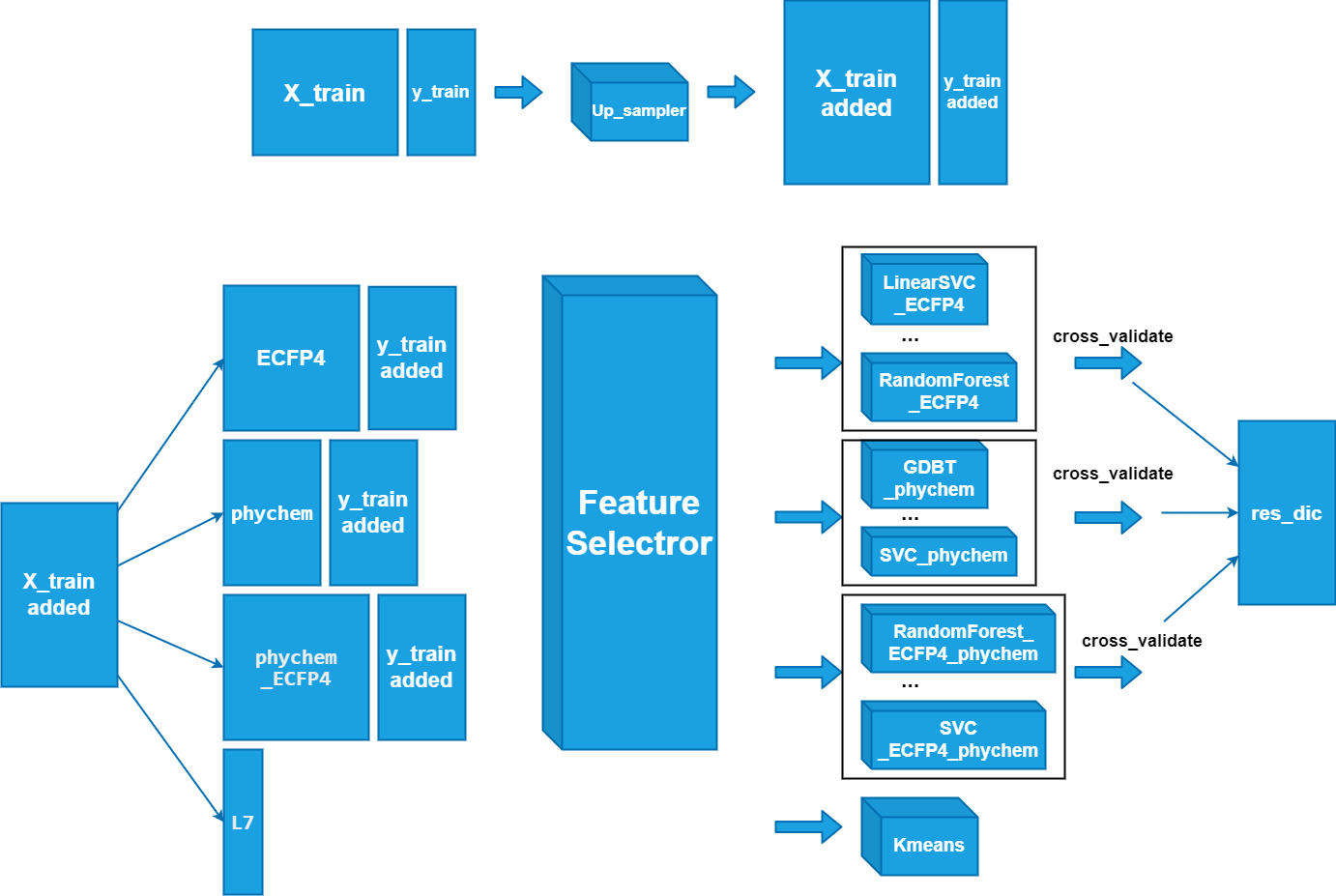
之后我们尝试引入特征选择。使用SelectFromModel算法进行自动特征选择，它通过分析数据在模型上的贡献度，将贡献度低于某个阈值(默认为所有属性贡献度的平均值)的属性剔除。其流程变为：

****

### （3）My\_vote\_select\_with\_up\_sample\_clf

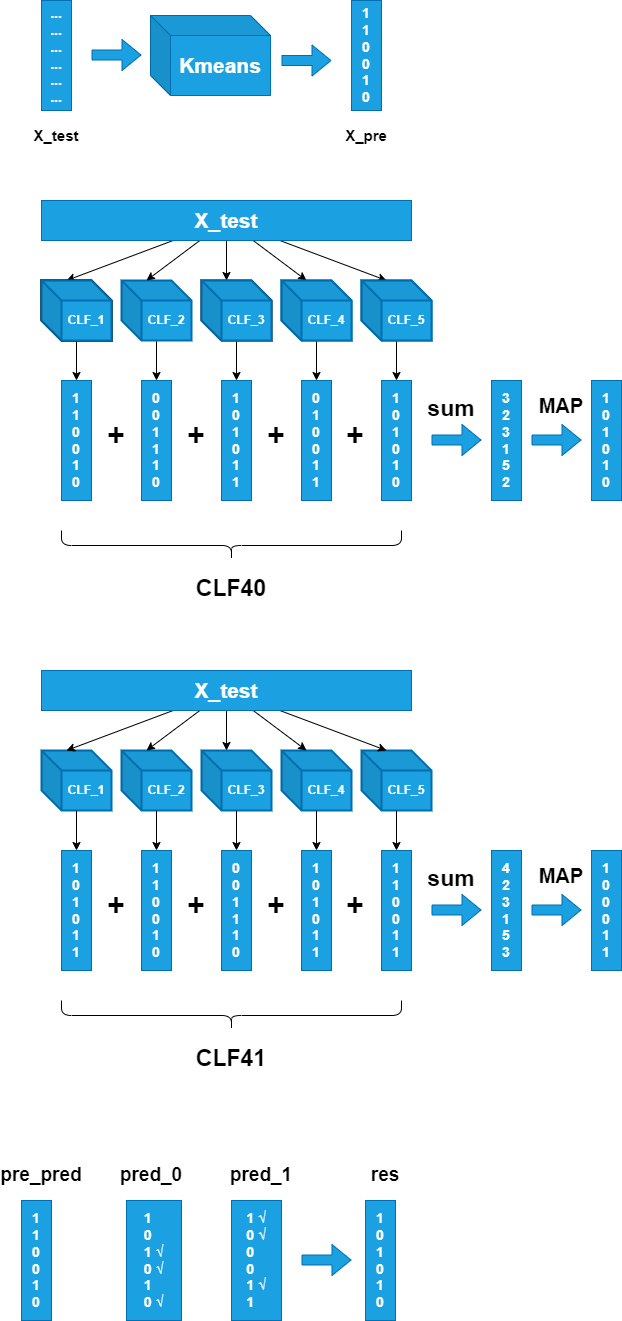
最后由于数据由些许不均衡，所以考虑引入上采样来提升结果。我们自己实现了一个简单的上采样器，他能通过复制的方式将训练集小类的数量补齐到与大类一致。加入上采样器之后我们得到了最终的模型：

**训练流程：**

****

对于一组训练数据，首先通过上采样器将小类的数量补齐至与大类一致，随后将补充后的训练数据根据属性类型划分为4部分，其中ECFP4、phychem、ECFP4+phychem使用SelectFromModel特征选择之后输入上面得到的表现好的各个基分类器。每个基分类器进行十折交叉验证，并将结果放入字典中，在预测时作为选择优势分类器的依据；剩余的L7属性用来训练KMeans模型，用于在预测时进行预分类。

**预测流程：**(各个模型的预测流程一致)

****

在预测时，首先通过KMeans模型得到预分类的结果，记为pre\_pred。之后从基分类器池中根据十折交叉验证的结果，选出recall\_0成绩top5的基分类器，他们在预测0类上有优势，通过投票将他们集成，得到一组结果，记为pred\_0；同样选出recall\_1成绩top5的对于1类预测有优势的一组基分类器，集成他们的结果得到pred\_1，最后根据pre\_pred选择最终的预测结果，如KMeans预测为0类，则选择pred\_0的对应结果为最终的结果，反之选择pred\_1的对应结果为最终的结果。

## 4、最佳参数的寻找：

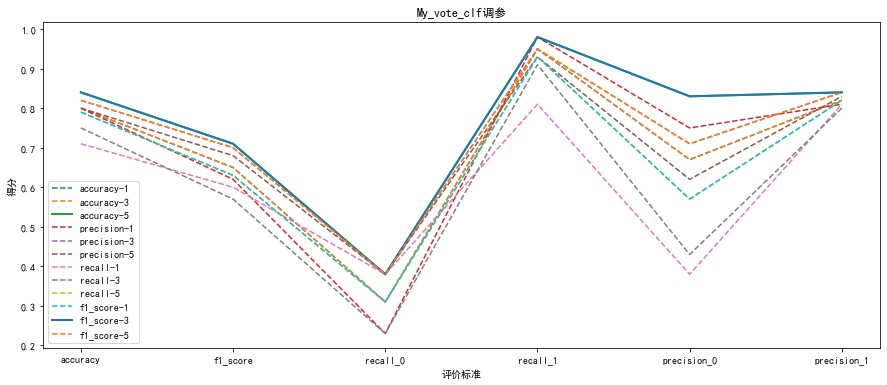
在最终的模型中，选择基分类器的排序依据和参与投票的基分类器个数被设为参数，可根据需求更改，如选出precision\_1成绩top3的基分类器等。SelectFromModel在初始化时可使用K值来调整阈值的范围，如SelectFromModel(estimator = RandomForest\_phychem,threshold=0.85\*mean）所以我们将它设为参数，并在之后进行调参。

利用类似于GridSearch的思路，对排序依据在['accuracy','precision','recall','f1\_score']中搜索；对参与投票的基分类器个数以步长2在(1,6)中进行搜索；对于特征选择的阈值系数K,我们按0.05的步长在（0.8,1.3）内进行搜索。

寻找最佳参数时，为了提高调参的效率，我们先使用原始数据，仅划划分为训练集与测试集进行一次测验，找出各个参数组合中相关指标(如recall\_0)较突出的进行十折交叉验证，得到最终的结果。

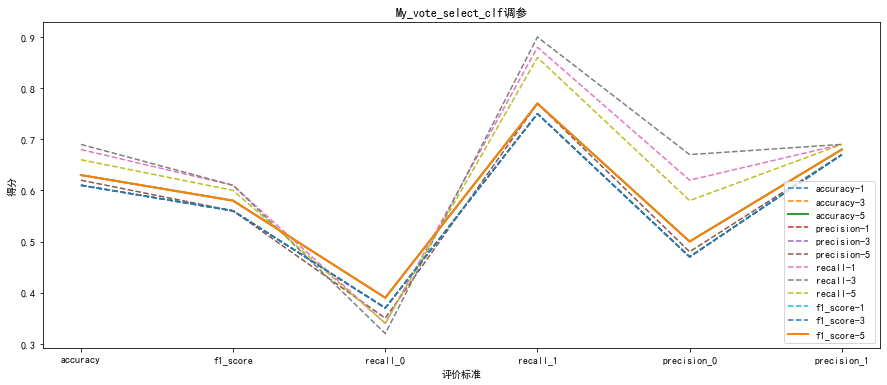
### （1）结果：

My\_vote\_clf：对于调参过程中可能的参数组合，绘制出折线图，其中每一条折现代表一组参数，实线为最佳的参数组合。

****

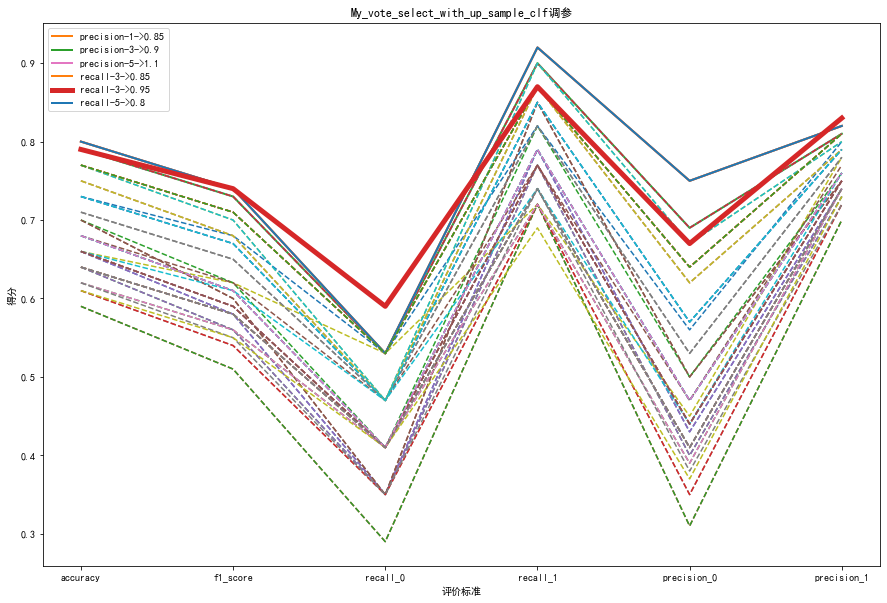
可以看出参数为score = 'accuracy'，topN = 5 以及score = 'f1\_score'，topN = 3单次验证的结果较为理想。

My\_vote\_select\_clf：

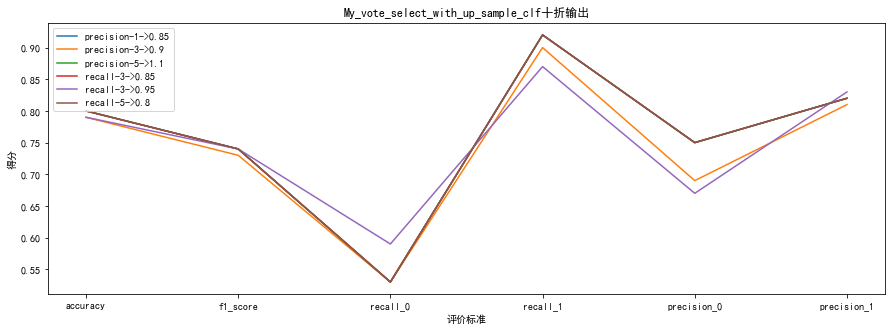
****

My\_vote\_select\_clf模型中两类的指标呈现此消彼长的趋势，结合实际情况我们选择recall\_0最高的score = 'accuracy'，topN = 5 以及score = 'f1\_score'，topN = 5为最佳的结果。

My\_vote\_select\_with\_up\_sample\_clf：

****

对于My\_vote\_select\_with\_up\_sample\_clf调参的输出较多，我们选择出了recall\_0指标较高的前几组参数，对他们都进行十折交叉验证，将输出结果绘制成折线图，结果如下。

****

参数为score = 'recall'，topN = 3，阈值系数为0.95为最佳的结果。

baseline定义：

V1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| feature\_  names | accuracy | f1\_score | auc | recall\_0 | recall\_1 | precision\_0 | precision\_1 |
| ECFP4+  phychem | 0.725858672 | 0.65784 | 0.649918 | 0.410191 | 0.889644 | 0.660318 | 0.744097 |

V2

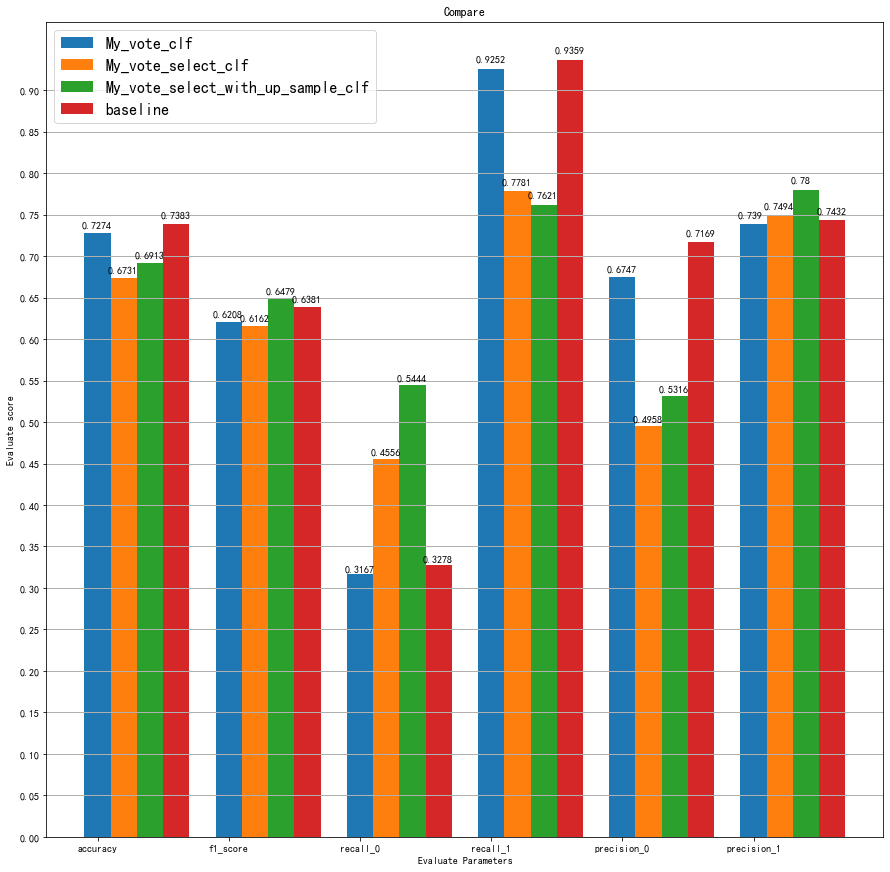
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| feature\_names | accuracy | f1\_score | auc | recall\_0 | recall\_1 | precision\_0 | precision\_1 |
| ECFP4+phychem | 0.738296478 | 0.638113285 | 0.631816817 | 0.327777778 | 0.935855856 | 0.716869748 | 0.743238771 |

V3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| feature\_names | accuracy | f1\_score | auc | recall\_0 | recall\_1 | precision\_0 | precision\_1 |
| ECFP4+phychem | 0.746118 | 0.532204 | 0.551366 | 0.135165 | 0.967568 | 0.686667 | 0.755595 |

### （2）分析：

各模型结果对比：

****

将各个模型各个指标数据绘制成柱状图，由于自定义的模型无法进行auc指标的计算，auc值全部为0，所以未将auc指标画在柱状图中。

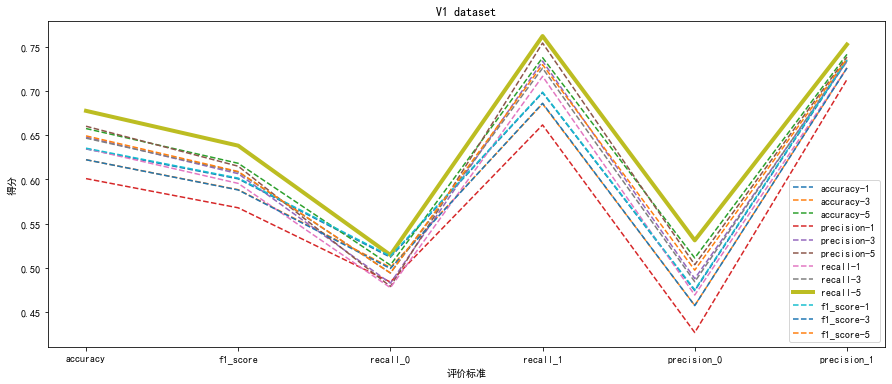
从中可以看出最基础的My\_vote\_clf模型(蓝色)各项指标基本与baseline(红色)齐平，有的甚至比baseline更低，我们猜测可能是由于未引入特征选择与上采样等特征处理的方法，My\_vote\_clf的训练数据与测试数据基本与baseline的训练、测试数据是一样的，而原始数据本身就是十分难以分类的，难分类的数据的抹平了算法之间的差异，使得基础模型的各项指标与baseline的差距不大，甚至更低。在引入了特征选择之后，My\_vote\_select\_clf模型的其他指标虽然相较于baseline有所下降，但是recall\_0提升了约14%，这符合我们的期望，模型对于小类的分类能力得到了提升。在加入了上采样机制之后，My\_vote\_select\_with\_up\_sample\_clf模型的各项指标比My\_vote\_select\_clf都有了些许提升，其中accuracy、precision\_0与baseline的差距减小，而f1\_score、precision\_1甚至超过了baseline，recall\_0的提升也达到了22%，其原因可能是上采样补齐了小类与大类的数量差距，使模型相较于未上采样是时加关注小类，故在recall\_0上能有继续的提升，而由于此任务为二分类，样本要么为0类，要么为1类，在0类分类准确性提升的同时，在1类上准确性也能得到提升，故f1\_score、precision\_1甚至能达到超过baseline的水平。

### （3）V1、V3数据集

V1版本数据集与V3版本数据集结果：

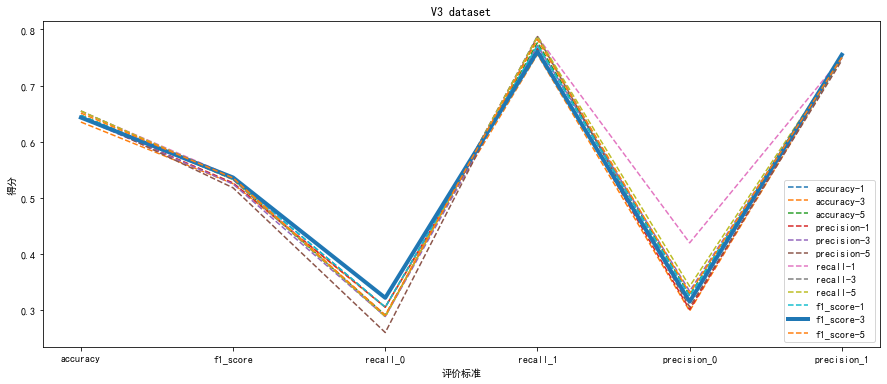
调参过程输出：

V1：由于V1数据集中没有L7属性，所以需要寻找新的属性集来训练KMeans模型由于预分类，通过比较各个属性集训练KMeans模型之后的silhouette\_score得分，选定phychem为新的用于预分类的属性集。V1使用的模型与调参方式与V2一致。

****

从折线图中可以看出参数为score = 'recall'，topN = 5为最佳的结果。

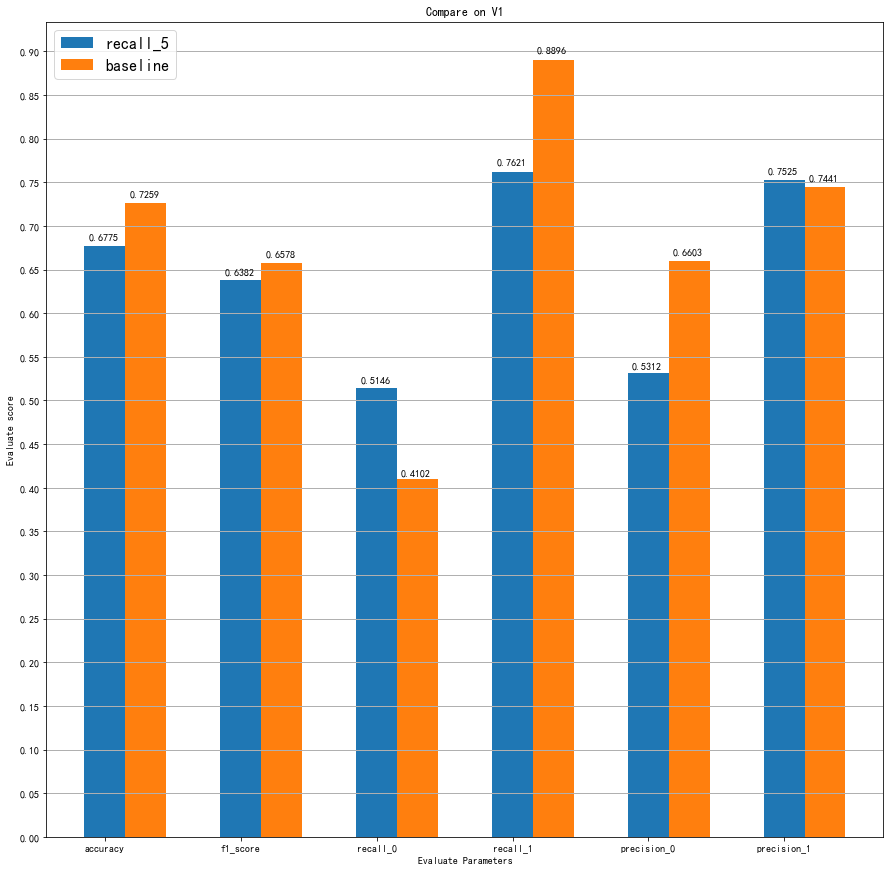
V3：V3使用的模型与调参方式与V2一致。

****

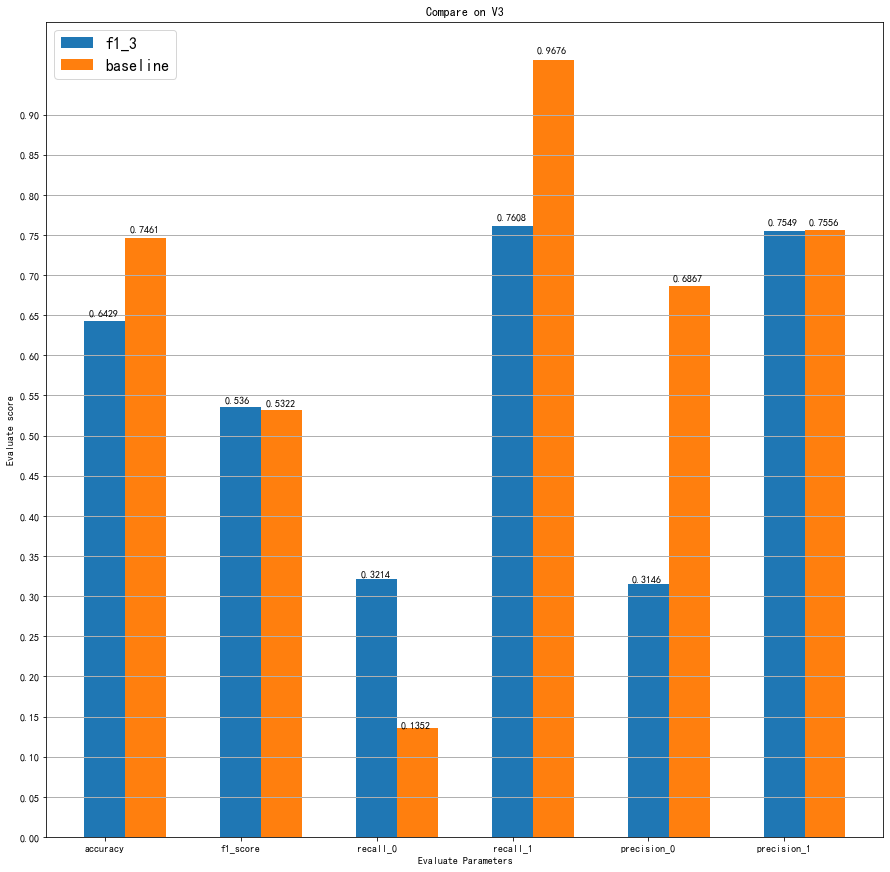
从折线图中可以看出参数为score = 'f1\_score'，topN = 3为最佳的结果。

与baseline的对比：

V1:

****

V1的各项指标中，recall\_0有较大的提升，约为10%，这是我们期望的，precision\_1也有1%的些许提升。其余指标均有不同程度的下降。这也印证了各个指标之间此消彼长的关系。

****

V3的情况与V1类似，除了recall\_0有19%左右的提升之外，其余指标均不同程度的下降。

## 5、总结

我们通过一步一步构建、完善模型的过程，最终得到了一个融合上采样+特征选择+预分类+投票的分类模型。我们主要对于三版本的数据集recall\_0指标进行提升，对于V1、V2、V3版本的数据分别达到了 10%、22%、19%的提升。Recall\_0召回率 = TP / (TP + FN)，表示的是样本中的0类有多少被预测正确了

对于人肝毒性数据集，越高的召回率说明更多的对人肝无毒的药物被正确分类，有一定的实际意义。

从此次实践中，我们更加意识到对特征进行处理的重要性，对特征的处理有时候比算法带来的提升更大。

不足与展望：

（1）由于模型最后的预测结果高度依赖于KMeans与分类的结果，所以预分类的效果直接影响了模型最终的预测效果，且预理论上最高的准确率即为模型的准确率，若预分类的准确率都高于模型的准确率，那之后的模型就没有意义了，有一定的局限性，且较低的预分类准确性可能引入新的误差。之后的工作可以集中在寻找准确率更高的预分类算法来逼近模型的极限，或者将预分类的结果作为参考而不是完全依赖于他的结果。

（2）特征选择算法我们只尝试了SelectFromModel一种，探索更好的特征选择方法可能会给模型带来新的提升。

（3）在探索特征选择的方法时，我们尝试使用LassoCV()自动选择出最适合当前分类器的特征选择阈值，但是由于自动选择出的阈值可能使训练数据中的所有属性都被剔除，原因未知，尝试解决之后无果，故放弃。之后可以深入探索其原因并尝试解决。

# 二、 药物联合作用数据集

## 1、综述

审查数据集后，我们总结其有如下特征：（1）特征非常多，共有4806个，可能会使用到特征抽取和降维等方法；（2）类分布不均，label\_0:label\_1:label\_2=29:2:1。我们用DecisionTree、RandomForest、ExtraTrees、GradientBoosting、XGB、SVC、LinearSVC、KNeighbor八种分类器对数据集进行处理。为了提高精确度，我们所有的模型训练和预测过程全部都是十折交叉验证。我们均是在经过十折验证划分后的训练集上进行处理，对测试集没有任何数据上的修改，训练集和测试集在理论上完全不会重合（除非原数据集上有重复数据），以此来规避过拟合现象产生。我们对数据集采用了离群点分析、重采样、特征抽取和二值化标签的方法进行处理。具体的实验流程和分析方法如下。

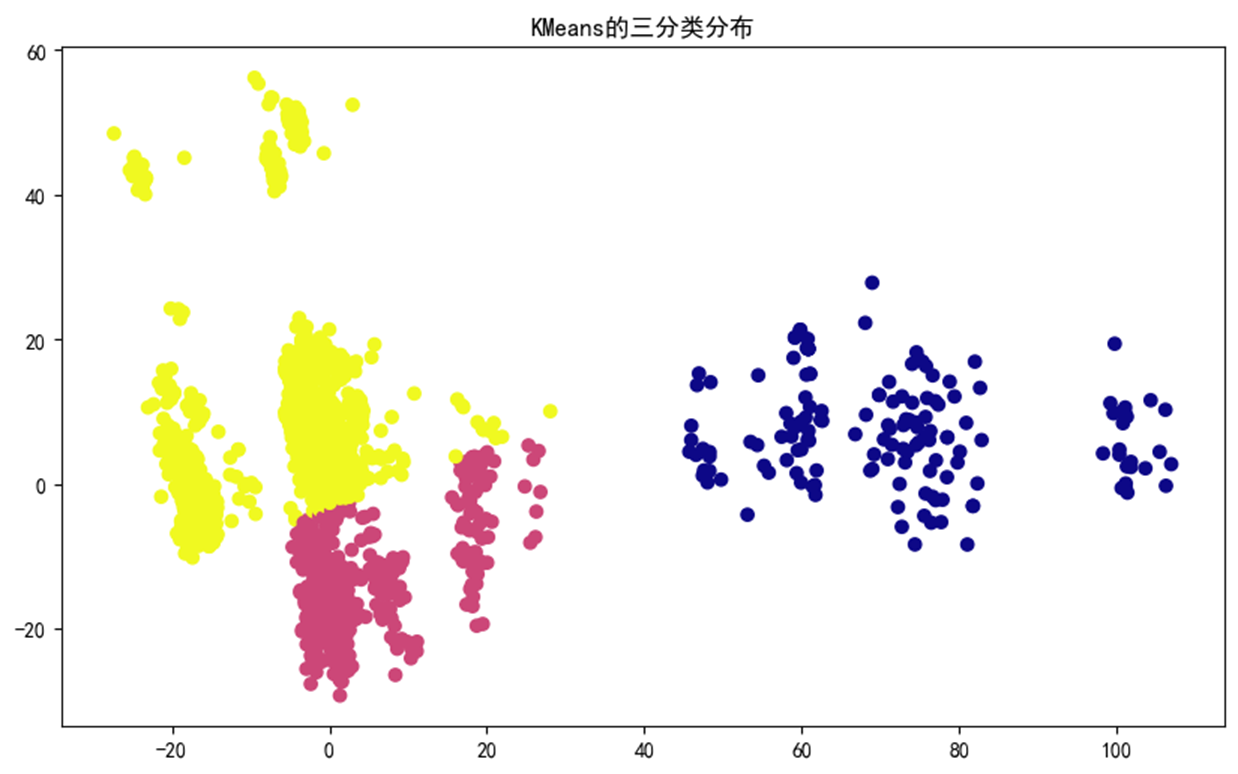
## 2、基准线定义

由于该数据集没有定义基线，我们对数据进行最基本的处理后直接进行多次十折交叉验证，取各分数的平均值作为我们这个数据集的基线。基线如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类器 | acc | f1 | auc | r0 | r1 | r2 | p0 | p1 | p2 |
| 决策树 | 0.629 | 0.349 | 0.569 | 0.660 | 0.325 | 0.256 | 0.914 | 0.123 | 0.008 |
| 随机森林 | 0.748 | 0.403 | 0.607 | 0.798 | 0.26 | 0.308 | 0.916 | 0.191 | 0.173 |
| 额外树 | 0.751 | 0.430 | 0.632 | 0.795 | 0.35 | 0.308 | 0.923 | 0.349 | 0.185 |
| GBDT | 0.762 | 0.414 | 0.613 | 0.814 | 0.26 | 0.288 | 0.917 | 0.234 | 0.146 |
| XGB | 0.836 | 0.442 | 0.621 | 0.901 | 0.145 | 0.309 | 0.915 | 0.192 | 0.192 |
| SVC | 0.910 | 0.366 | 0.629 | 0.999 | 0.095 | 0.02 | 0.913 | 0.15 | 0.04 |
| LinearSVC | 0.686 | 0.346 | 0.807 | 0.734 | 0.23 | 0.218 | 0.910 | 0.127 | 0.066 |
| KNN | 0.914 | 0.519 | 0.761 | 0.983 | 0.335 | 0.125 | 0.930 | 0.726 | 0.36 |

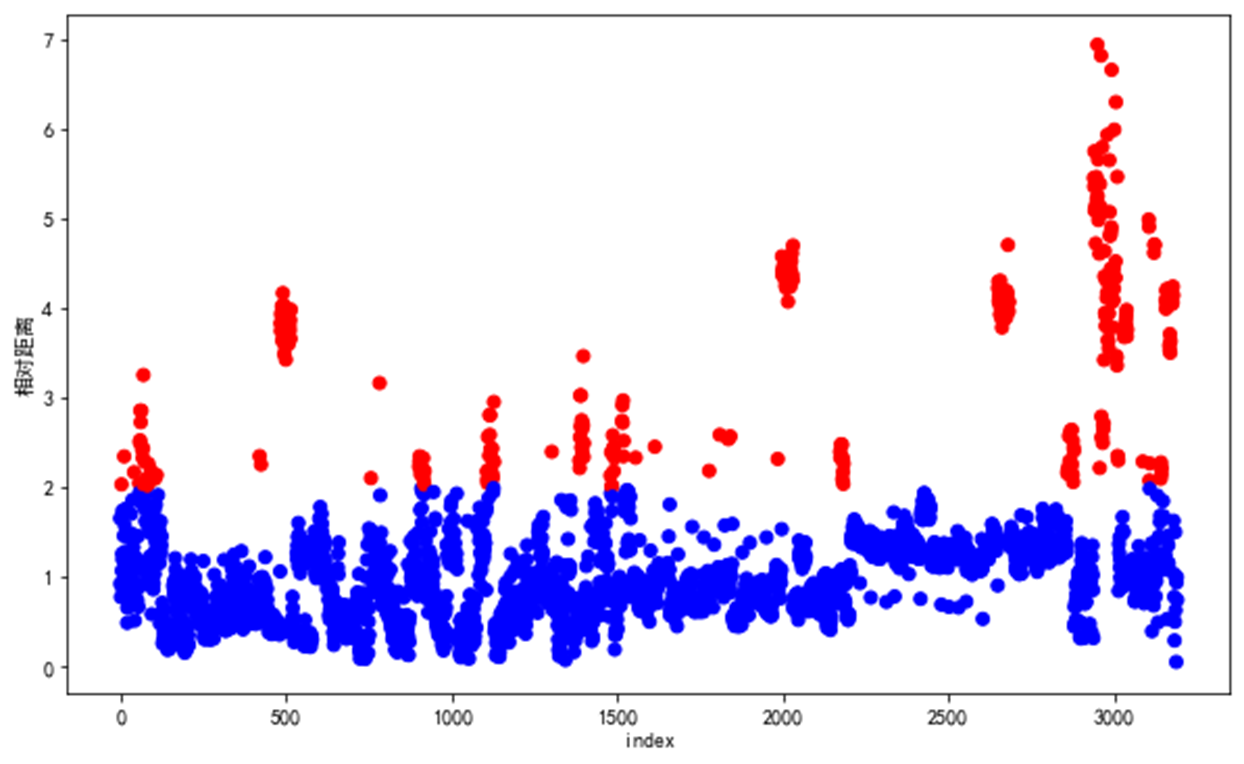
## 3、对数据进行离群点分离处理

因为原数据有将近5000个特征，所以我们先用PCA方法对数据进行降维处理，并用KMeans算法对降维后的数据进行聚类，以查看各类数据的分布情况，其结果如下：



从图中可见，各类的聚集效果较好，没有太过离群的点或群出现。但是我们对黄色点上方和蓝色点右方的两簇点是否离群是有争议的，所以我们为保险起见采用了离群点分离的方法对数据处理。

我们分别计算出每个点相对于三个聚簇中心点的距离，将距离大于2的点标记为离群点，结果如下：



找出离群点后，我们做如下处理：（1）将训练集中的离群点用上述方法提出；（2）用提出的训练集离群点单独训练一个模型，用来对测试集中的离群点进行标注；（3）用训练集中的剩余的非离群点训练另一个模型，用来对测试集中的非离群点进行标注；（4）将测试集用上述方法进行离群点分离操作，并将离群点用前一个模型训练，非离群点用后一个模型训练。整合评分后得到结果如下（其中基线为红色，auc暂时没有计算）：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类器 | acc | f1 | auc | r0 | r1 | r2 | p0 | p1 | p2 |
| 决策树 | 0.665  0.629 | 0.341  0.349 | 0  0.569 | 0.710  0.660 | 0.275  0.325 | 0.157  0.256 | 0.906  0.914 | 0.122  0.123 | 0.054  0.008 |
| 随机森林 | 0.787  0.748 | 0.339  0.403 | 0  0.607 | 0.861  0.798 | 0.09  0.26 | 0.089  0.308 | 0.903  0.916 | 0.055  0.191 | 0.089  0.173 |
| 额外树 | 0.764  0.751 | 0.360  0.430 | 0  0.632 | 0.830  0.795 | 0.17  0.35 | 0.097  0.308 | 0.907  0.923 | 0.264  0.349 | 0.084  0.185 |
| GBDT | 0.777  0.762 | 0.371  0.414 | 0  0.613 | 0.843  0.814 | 0.17  0.26 | 0.117  0.288 | 0.907  0.917 | 0.218  0.234 | 0.090  0.146 |
| XGB | 0.846  0.836 | 0.368  0.442 | 0  0.621 | 0.927  0.901 | 0.055  0.145 | 0.107  0.309 | 0.905  0.915 | 0.130  0.192 | 0.148  0.192 |
| SVC | 0.909  0.910 | 0.357  0.366 | 0  0.629 | 0.999  0.999 | 0.075  0.095 | 0.01  0.02 | 0.911  0.913 | 0.154  0.15 | 0.1  0.04 |
| LinearSVC | 0.779  0.686 | 0.335  0.346 | 0  0.807 | 0.846  0.734 | 0.2  0.23 | 0.038  0.218 | 0.906  0.910 | 0.075  0.127 | 0.041  0.066 |
| KNN | 0.900  0.914 | 0.437  0.519 | 0  0.761 | 0.972  0.983 | 0.245  0.335 | 0.048  0.125 | 0.922  0.930 | 0.417  0.726 | 0.187  0.36 |

与基线结果对比，发现效果并不是很出色。这印证了之前我们的想法：其离群效应不是很明显，而且离群点也没有过于明显的特征，所以得到的结果也不会较基线有太高的提升。

## 4、对数据进行重采样方法处理

针对数据集大类小类差距明显（0:1:2 = 29:2:1）的特征，我们考虑用重采样手段对数据进行预处理。使用dataframe自带的sample函数对三类数据进行重采样，将数据上下采样成0:1:2 = 1:1:1。并且为了尽量控制变量，我们采样后得到的数据总量近似不变。其结果如下：

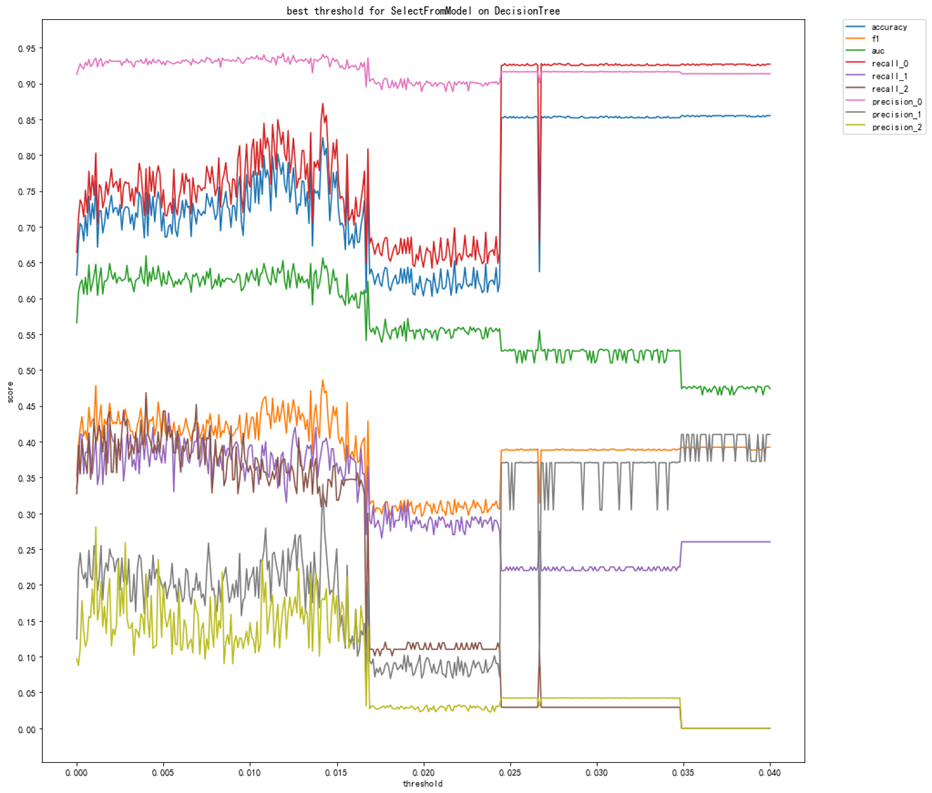
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类器 | acc | f1 | auc | r0 | r1 | r2 | p0 | p1 | p2 |
| 决策树 | 0.686  0.629 | 0.394  0.349 | 0.613  0.569 | 0.719  0.660 | 0.335  0.325 | 0.431  0.256 | 0.925  0.914 | 0.122  0.123 | 0.136  0.008 |
| 随机森林 | 0.817  0.748 | 0.444  0.403 | 0.653  0.607 | 0.874  0.798 | 0.255  0.26 | 0.308  0.308 | 0.921  0.916 | 0.219  0.191 | 0.224  0.173 |
| 额外树 | 0.841  0.751 | 0.478  0.430 | 0.667  0.632 | 0.897  0.795 | 0.295  0.35 | 0.337  0.308 | 0.928  0.923 | 0.243  0.349 | 0.291  0.185 |
| GBDT | 0.713  0.762 | 0.401  0.414 | 0.638  0.613 | 0.749  0.814 | 0.395  0.26 | 0.328  0.288 | 0.925  0.917 | 0.169  0.234 | 0.112  0.146 |
| XGB | 0.743  0.836 | 0.412  0.442 | 0.628  0.621 | 0.786  0.901 | 0.345  0.145 | 0.338  0.309 | 0.923  0.915 | 0.166  0.192 | 0.137  0.192 |
| SVC | 0.656  0.910 | 0.403  0.366 | 0.686  0.629 | 0.667  0.999 | 0.71  0.095 | 0.219  0.02 | 0.947  0.913 | 0.234  0.15 | 0.072  0.04 |
| LinearSVC | 0.575  0.686 | 0.329  0.346 | 0.723  0.807 | 0.592  0.734 | 0.43  0.23 | 0.376  0.218 | 0.917  0.910 | 0.123  0.127 | 0.070  0.066 |
| KNN | 0.653  0.914 | 0.424  0.519 | 0.739  0.761 | 0.657  0.983 | 0.665  0.335 | 0.521  0.125 | 0.961  0.930 | 0.209  0.726 | 0.112  0.36 |

通过与基线进行比较，我们发现两个小类1、2的recall值对各个模型都有或多或少的提升。因为重采样很好的平衡了样本，其小类recall分数的提升是可以预测的。但是重采样虽然效果很好，却不是很稳定，原因如下：（1）使用的重采样函数是dataframe自带的函数，其实现可能很粗糙；（2）我们将数据集中的类重采样成1:1:1，这个比值可能有待调整，比如2:1:1等；（3）如果原数据集中出现几个重复数据，那么重采样后可能会将重复数据放大，假如测试集中有相同的数据，那可能会出现很严重的过拟合现象。

## 5、对数据进行特征抽取处理

针对该数据集特征很多的特点，我们猜测有很多的属性其实对数据集分类无影响或影响很小；或有些特征和其他特征的相关性很高，只需要保留一个特征即可。所以我们使用SelectFromModel进行特征抽取，并寻找最佳参数threshold。

由于时间和资源问题，我们寻找最佳参数时无法将八个模型全部跑起来综合考虑，所以我们仅使用在决策树下的recall1评分来作为选取标准。具体流程如下：（1）不设定threshold值，查看默认threshold参数（0.00020）；（2）根据默认threshold参数值选取合适的参数查找范围和查找步长。设定threshold参数范围为0-0.08，步长为0.0001，分别针对决策树模型进行相应参数下的特征抽取；（3）对抽取特征值后的数据使用决策树进行训练并评分；（4）将各评分与对应的threshold值绘制为折线图，重点关注recall1的值，折线图如下所示；（5）取出一个recall1最高的点，将其对应的threshold值作为其他模型特征抽取的参数值，训练其他模型。得到结果如下：



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类器 | acc | f1 | auc | r0 | r1 | r2 | p0 | p1 | p2 |
| 决策树 | 0.725  0.629 | 0.439  0.349 | 0.646  0.569 | 0.756  0.660 | 0.425  0.325 | 0.428  0.256 | 0.934  0.914 | 0.241  0.123 | 0.146  0.008 |
| 随机森林 | 0.826  0.748 | 0.462  0.403 | 0.669  0.607 | 0.880  0.798 | 0.315  0.26 | 0.308  0.308 | 0.931  0.916 | 0.348  0.191 | 0.229  0.173 |
| 额外树 | 0.848  0.751 | 0.487  0.430 | 0.654  0.632 | 0.903  0.795 | 0.335  0.35 | 0.308  0.308 | 0.934  0.923 | 0.475  0.349 | 0.205  0.185 |
| GBDT | 0.813  0.762 | 0.449  0.414 | 0.669  0.613 | 0.868  0.814 | 0.305  0.26 | 0.277  0.288 | 0.926  0.917 | 0.298  0.234 | 0.183  0.146 |
| XGB | 0.850  0.836 | 0.462  0.442 | 0.677  0.621 | 0.913  0.901 | 0.22  0.145 | 0.309  0.309 | 0.923  0.915 | 0.303  0.192 | 0.295  0.192 |
| SVC | 0.911  0.910 | 0.367  0.366 | 0.622  0.629 | 1.000  0.999 | 0.095  0.095 | 0.02  0.02 | 0.913  0.913 | 0.164  0.15 | 0.04  0.04 |
| LinearSVC | 0.898  0.686 | 0.324  0.346 | 0.938  0.807 | 0.992  0.734 | 0.015  0.23 | 0  0.218 | 0.905  0.910 | 0.12  0.127 | 0  0.066 |
| KNN | 0.905  0.914 | 0.469  0.519 | 0.751  0.761 | 0.977  0.983 | 0.295  0.335 | 0.067  0.125 | 0.926  0.930 | 0.629  0.726 | 0.273  0.36 |

我们不难发现决策树的recall值有了明显提升，但是其他模型的recall值却提升不明显甚至下降。这是因为我们在threshold参数选取的时候只使用了决策树模型的recall值作为依据，而没有顾及其他模型。但是至少我们可以确定一点：特征抽取方法确实可以改善对各样本分类的效果，而且这种帮助是比重采样方法稳定很多的。只要再针对其他模型用同样的方法选取其特定的threshold参数，其他模型的recall1和recall2的值也会有稳定的提升。

## 6、对数据进行标签二值化处理

我们将训练集1、2类样本合为一类（假定为3类），与原0类样本进行建模，得到0\_3分类模型；然后使用训练集1、2类样本建模，得到1\_2分类模型。我们将测试集先用0\_3分类模型预测出0类（原0类）和3类（原1或2类），再将3类对应的测试样本再使用1\_2分类模型预测试1类和2类，把其与之前0类的预测结果结合评分。得到结果如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类器 | acc | f1 | auc | r0 | r1 | r2 | p0 | p1 | p2 |
| 决策树 | 0.626  0.629 | 0.352  0.349 | 0.0  0.569 | 0.654  0.660 | 0.345  0.325 | 0.374  0.256 | 0.917  0.914 | 0.107  0.123 | 0.094  0.008 |
| 随机森林 | 0.752  0.748 | 0.418  0.403 | 0.0  0.607 | 0.796  0.798 | 0.34  0.26 | 0.337  0.308 | 0.924  0.916 | 0.209  0.191 | 0.125  0.173 |
| 额外树 | 0.734  0.751 | 0.410  0.430 | 0.0  0.632 | 0.777  0.795 | 0.33  0.35 | 0.328  0.308 | 0.921  0.923 | 0.295  0.349 | 0.130  0.185 |
| GBDT | 0.746  0.762 | 0.374  0.414 | 0.0  0.613 | 0.801  0.814 | 0.235  0.26 | 0.219  0.288 | 0.914  0.917 | 0.147  0.234 | 0.090  0.146 |
| XGB | 0.793  0.836 | 0.404  0.442 | 0.0  0.621 | 0.854  0.901 | 0.16  0.145 | 0.327  0.309 | 0.913  0.915 | 0.151  0.192 | 0.180  0.192 |
| SVC | 0.910  0.910 | 0.366  0.366 | 0.0  0.629 | 0.999  0.999 | 0.095  0.095 | 0.02  0.02 | 0.913  0.913 | 0.15  0.15 | 0.04  0.04 |
| LinearSVC | 0.633  0.686 | 0.312  0.346 | 0.0  0.807 | 0.672  0.734 | 0.31  0.23 | 0.172  0.218 | 0.899  0.910 | 0.082  0.127 | 0.049  0.066 |
| KNN | 0.911  0.914 | 0.519  0.519 | 0.0  0.761 | 0.979  0.983 | 0.35  0.335 | 0.125  0.125 | 0.931  0.930 | 0.694  0.726 | 0.35  0.36 |

可以看出分类效果提升并不明显。我们猜想原因可能是大类和两个小类的样本数量差距太悬殊，就算是两个小类合为一类，其与大类样本数量比仍有3:29，和重采样下的1:1:1差距显著。

## 7、总结

我们使用了上述四种方法对数据进行预处理，其结果最好的是重采样方法，其稳定性最好的是特征抽取方法。从基线观察recall1和recall2的评分结果，我们发现没有哪种分类器对两个小类的分类有很好的效果，所以我们没有尝试使用投票器的方法来处理模型。

总的来说，此次实验可以说是开了一个好头，后续有很多可以实现和完善的工作，如：（1）将这些方法联合处理，得到更好的结果；（2）对其他模型进行特征抽取；（3）选择更好的重采样方法如bootstrap。为了实现以上工作，必须对每个模型的内部算法原理深入研究，从而知道它们分别适用于哪种数据类型，如何调整参数甚至修改内部实现等。