参考文献的分析

Raccuglia, P., Elbert, K., Adler, P. et al. Machine-learning-assisted materials discovery using failed experiments. Nature 533, 73–76 (2016). https://doi.org/10.1038/nature17439

作者从实验室笔记本中收集了水热合成反应的信息，并使用化学信息学软件添加了使用反应物名称计算得到的物理化学性质（比如有机反应物的配体分子量，氢键给体受体数，极性表面积，无机反应物的电离能，电子亲合能，电负性，硬度，原子半径，元素周期表中的位置)。数据的标签取值为1，2，3，4从小到大代表结晶结果从坏到好。以反应产生单晶或多晶为成功（3，4），反之为失败。

从这些数据中划分出测试集与验证集。以此使用weka软件训练机器学习模型来预测新反应的结果。虽然训练时的标签分为4类，但是评估模型时将分类结果化为成功与失败。使用不同模型得到的交叉验证结果(三折，反应物生成物取平均)如下，其中支持向量机（Pearson VII核）的表现最好，准确率达到了74%。

表1: 作者使用的各模型的性能

Technique (all from Weka library)	2-class accuracy (%)[a]
Decision tree (J48)	67.5
Random forest (size 100 and 1000)	69.8, 70.5
Logistic regression	69.2
k-Nearest neighbors (K = 1, 2, and 3)	69.1, 66.9, 68.4
SMO SVM	74.1

使用Weka库中的cfsSubseEval拥有的两种选择方法：最佳优先法和贪婪逐步法，按各特征对分类效果的影响进行选择。两种方法得到的最重要的6个特征不同，但是训练出来的模型交叉验证都得到了70%左右的准确率。由于使用选择后的特征导致支持向量机的性能下降，仍然使用全部数据训练支持向量机。

在商业化的有机化学品数据库中找到与训练集中使用的反应物的结构接近的物质，进行新的水热合成反应。用上述支持向量机预测新的反应，结果precision为89%，超过了人类直觉（precision为78%）。

由于支持向量机不具有可解释性，将其预测结果用决策树模型拟合，得到了反应成功需要的条件。

亮点在于利用了一直被忽视的失败数据，并且提出了根据机器学习模型预测新反应，并且将获得的反应数据训练模型的方法，并且不止步于机器学习，而是从模型中提取了化学直觉。但是使用的方法不够先进。引入的描述符本身就过于简单，只有反应物的一些简单的物理化学性质与反应条件，没有量子化学的信息比如电子密度，前线轨道能量。使用了最简单的支持向量机模型并且没有对参数比如软间隔大小，核函数的影响进行分析，也没有进行类别平衡。并且既然将SVM的预测结果用决策树拟合获得化学直觉，那为何不直接使用决策树模型学习?

问题:

测试集上89%的成功率（precision）究竟是如何计算的，我们用作者的模型预测计算的结果(见模型选择部分)只有81%。

数据集的划分可能存在问题，文献中的训练集所包含的负样本和测试集所包含的负样本具有非常不同的性质，以至于适应了训练集的模型对测试集的负样本描述或者很差（不平衡数据集）或者严重干扰了正样本的识别（平衡数据集）。

特征选择在不同的特征选择方法之间并不具有好的一致性，这一点让我们对于文中提出的“化学直觉”这件事情持保留态度。如果真的是足以让人能够总结和归纳规律从而进一步探索其机理和本质的“化学直觉”所对应的特征，应当和结果具有相当的、鲜明的相关性，为何挑选出来的特征甚至不能够训练出一棵差强人意的决策树？

进一步地，就从我们的化学直觉上讲，MOF的结构显然不是唯一的，而对于不同的MOF结构，却希望用一个普遍的、只依赖于数个十分简单的参量的模型就能够预测特定的反应物组合能否形成MOF，这相当于优化一个本身就有很多个稳定解的问题，单个决策树似乎并不很擅长这一点，而当前的数据集似乎还不能充分的反映这样多个稳定区域的特性（主要在于周边的不稳定区域行为），导致了神经网络不容易找出正确的规律。

如果作者先考虑对MOF根据结构特征以及一些底物组合特征进行聚类，再对各个类各自进行这样的处理，结果是否会更好，而分析结果得到的规律是否能够给出更加清晰的化学图像？

为何不直接用分类树拟合实验结果而是拟合SVM的结果？如果是为了解释SVM的决策面，可以生成更多随机数据，由支持向量机预测，再由决策树学习。

展望:

可以用委员会方法或者其他不是硬分类的方法给出成功的概率，并进行探索与利用的平衡。每次探索新的反应时综合考虑成功的概率以及对这一反应的了解程度，选择越策方差大或者成功概率大的反应，而不是像文献中的利用新反应物与已有反应物的相似程度来探索新的反应。此外，关于描述符的构建除了作者提供的简单的物化性质外，可以加入分子动力学模拟或量子化学方法得到的与微观结构相关的信息。

此外关于实验本身，可以用更精细的指标衡量结果的好坏，比如结晶的程度定量化为一个连续的数值。这样的话更适合用回归模型（而不是分类）来处理这一问题。

数据处理与分析

作者开放了使用的数据，并且已经划分成了训练集[train.csv][./data/train.csv]与测试集[test.csv][./data/test.csv]。每个数据有264维。有标签，值为1，2，3，4从低到高代表结晶情况从好到坏。

其中有32个feature取值为"Yes","No"，分别用数值1,0。其中一些feature中含有"?"表示参数不明，使用"?"为"Yes"的先验概率表示。

$$\frac{\text{这一feature中Yes的数量}}{\text{Yes的数量}+\text{No的数量}}$$

有6个feature取值为代表反应物名称的字符串，转换为独热编码。

其余feature取值为数值，不需要特殊处理。

由于数据大部分的feature都是直接由理论计算得到，不应当存在需要处理的异常值，因此不进行异常值的检测。事实上使用$z=\frac{z-\mu}{\sigma}$标准化后以$|z|<3$判定是否出现异常值，发现几乎所有特征都出现了异常值。数据本身不是正态分布，更接近于poisson分布，不应当因为偏离平均值过多而舍去。保存标准化的数据。

作者希望发现新的反应物，因此要求划分测试集时将同一反应物组合的反应全部放在测试集或者训练集。因此我们按照反应物组合将训练集数据分组。从中训练集划分出验证集时随机抽取反应物组合，再得到相应的数据。

结果保存在[Xx.npy][./processedData/Xx.npy]，[Y.npy][./processedData/Y.npy], [y.npy][./processedData/y.npy]

作者将标签为1，2的反应条件视作失败反应，标签为3，4视为成功反应。由此对训练集的标签比例作图。

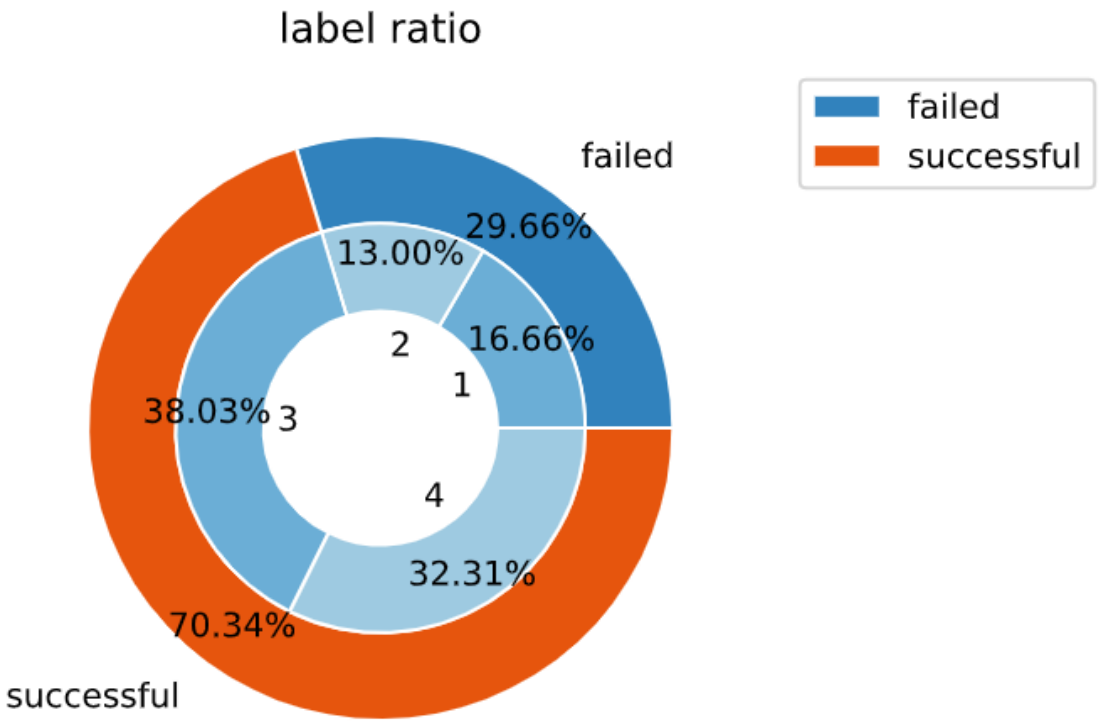

图1: 训练集中成功与失败数据的比例

可以看出，大部分反应成功，类别明显不均衡。

而测试集的数据更不均衡

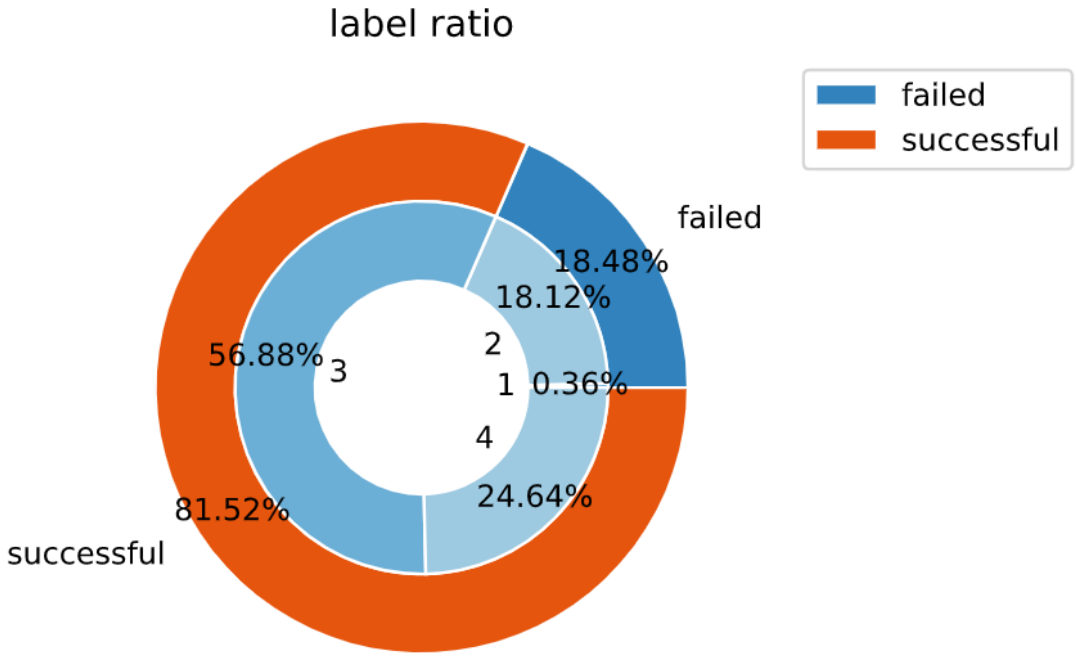

图2: 测试集中成功与失败数据的比例

对不同反应物类型的反应分组，得到每组中反应个数的分布如下。最大的一组由175个反应，绝大多数反应物组合的反应数目在20以下。

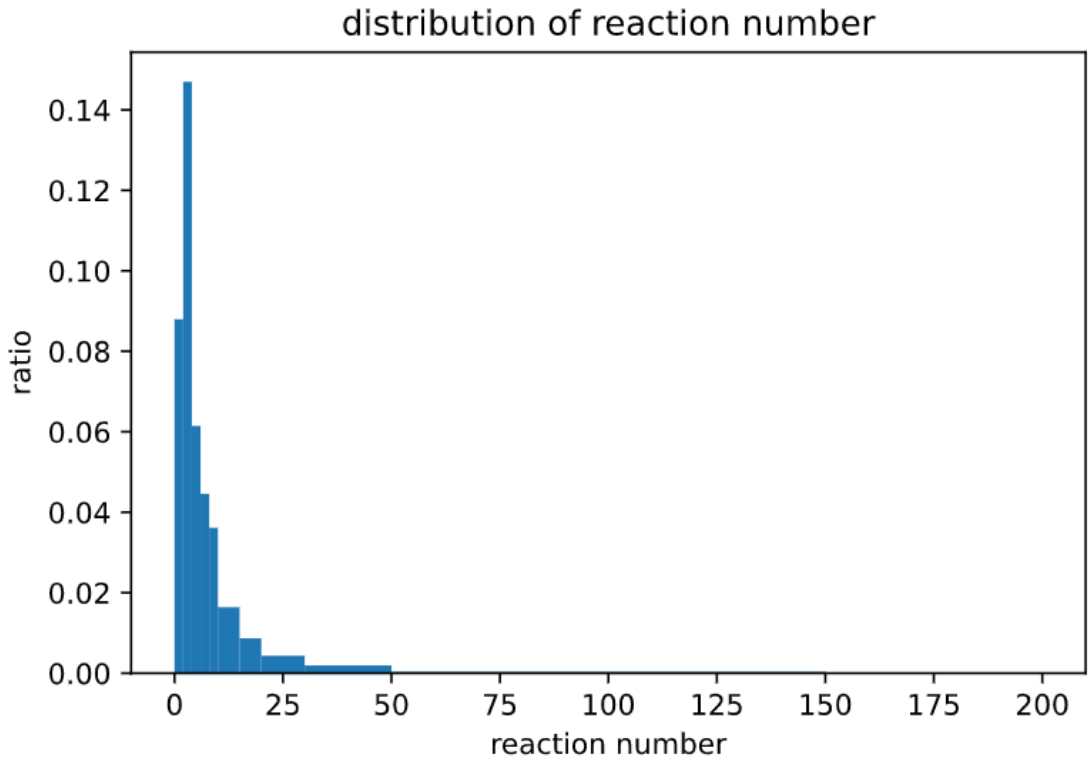

图3: 每组反应物反应个数的分布

为了考察反应物类型对反应结果的影响，对每个反应物组合，所有反应中成功反应的比例。发现反应物名称对反应结果有重大影响。但是作者的意图是发现新反应，因此将反应物名称从特征中剔除。

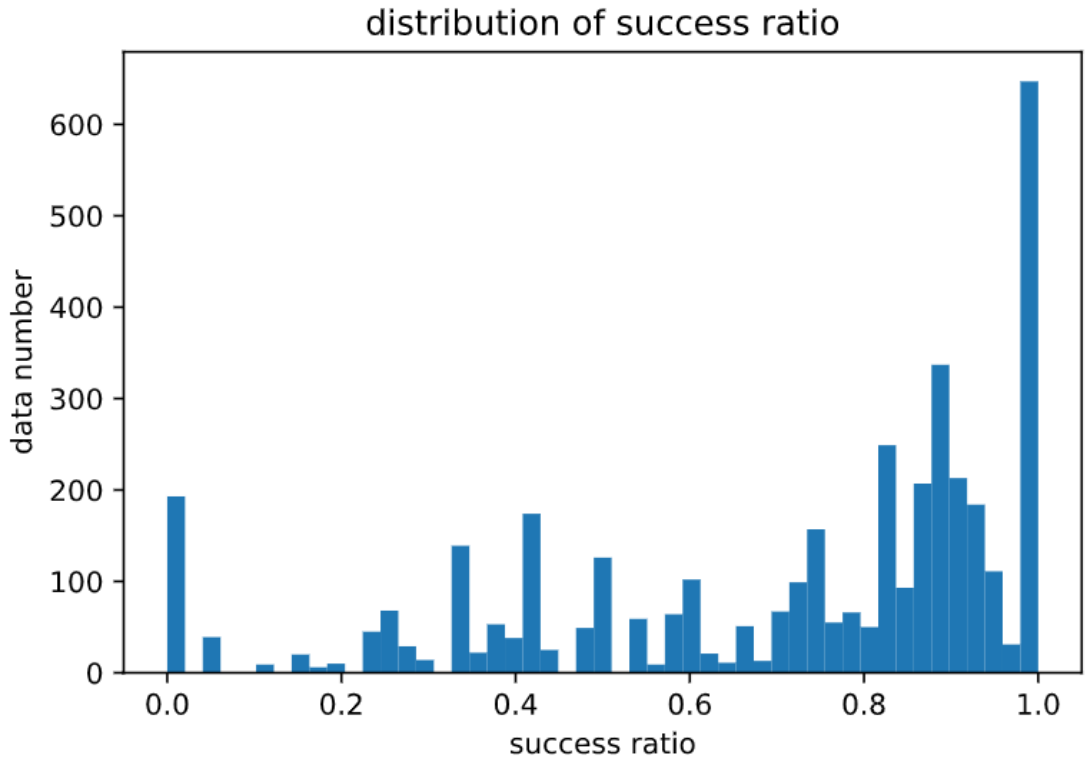

图4: 每组反应物反应成功比例的分布

模型选择

作者使用3折交叉验证进行超参数调节与模型评价，我们也同样使用。但是由于算力不足，我们没有像作者一样重复15次对结果取平均。

作者的衡量指标是accuracy，但我们认为这毫无意义，只要全部预测为1即可达到70%以上（即作者的模型的水平）。我们综合观察混淆矩阵，精度，召回率，准确率。由于实验代价比训练模型代价高，因此最看重精度。我们预测集上精度最高的模型达到了90%，而交叉验证的结果精度普遍超过了80%。就作者给出的训练集与测试集，我们做的更好。

作者使用了SVM，核函数为Pearson VII。我们使用sklearn中的SVC进行了复现。并对数据进行标准化

$$z = \frac{x - \mu}{\sigma}$$

类别不均衡问题可以直接使用SVC的class_weight="balanced"解决。

作者在测试集上的结果为

混淆矩阵

	prediction 1	**prediction 0**
true label 1	212	13
true label 0	48	3

accuracy=81%, precision=81%, recall=93%

使用作者使用的Pearson VII核函数，对不同的正则化强度（控制软间隔大小）进行实验，交叉验证的结果如下。

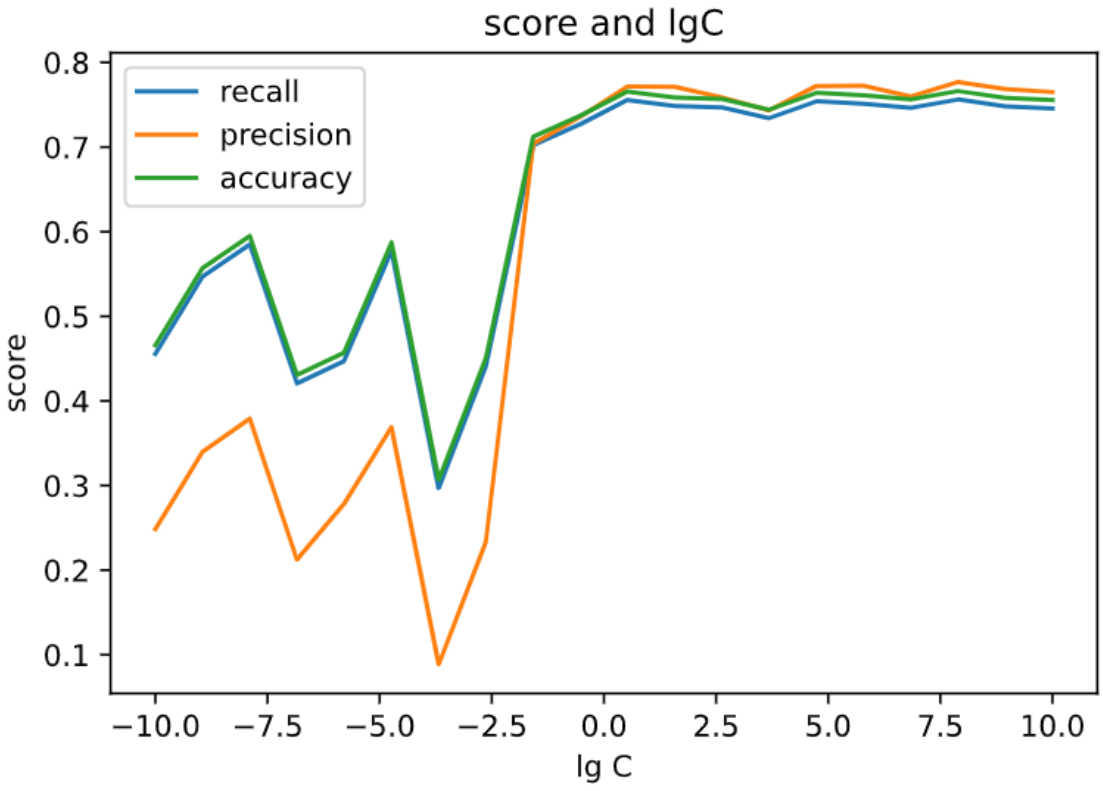

图5: 交叉验证性能与正则化强度的关系

设C=10，得到支持向量机模型应用于测试集上，发现效果极差，混淆矩阵为

	prediction 1	**prediction 0**
true label 1	225	0
true label 0	51	0

accuracy =81%, precision=81%, recall=100%

但是这个模型是全然无用的，无法预测0。

这一模型在训练集上给出结果的混淆矩阵为

	prediction 1	prediction 0
true label 1	2778	0
true label 0	21	1173

效果极佳，说明出现了过拟合现象。但是尝试增大软间隔大小（减小C）也不能在保持模型准确率合理的情况下，提高0的特异性。

这说明测试集与训练集的数据分布显著不同。

使用其他核函数：

rbf核，3折交叉验证结果如下，recall与另外两条曲线重合。

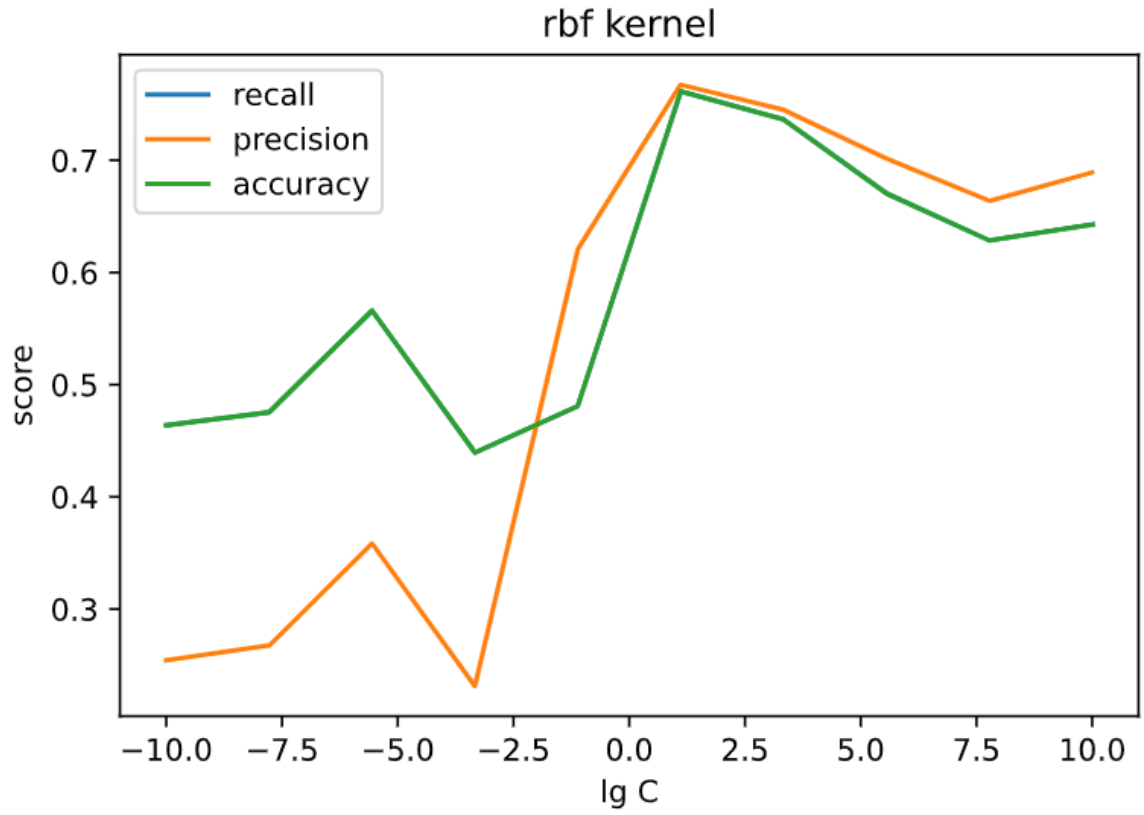

图6: 交叉验证性能与正则化强度的关系

使用C=100，得到的模型性能如下

	prediction 1	prediction 0
true label 1	116	109
true label 0	20	31

accuracy =53%, precision=85%, recall=51%

虽然这一模型的recall很低，但是precision非常高。

而使用sigmoid核，CV性能较差且不稳定。

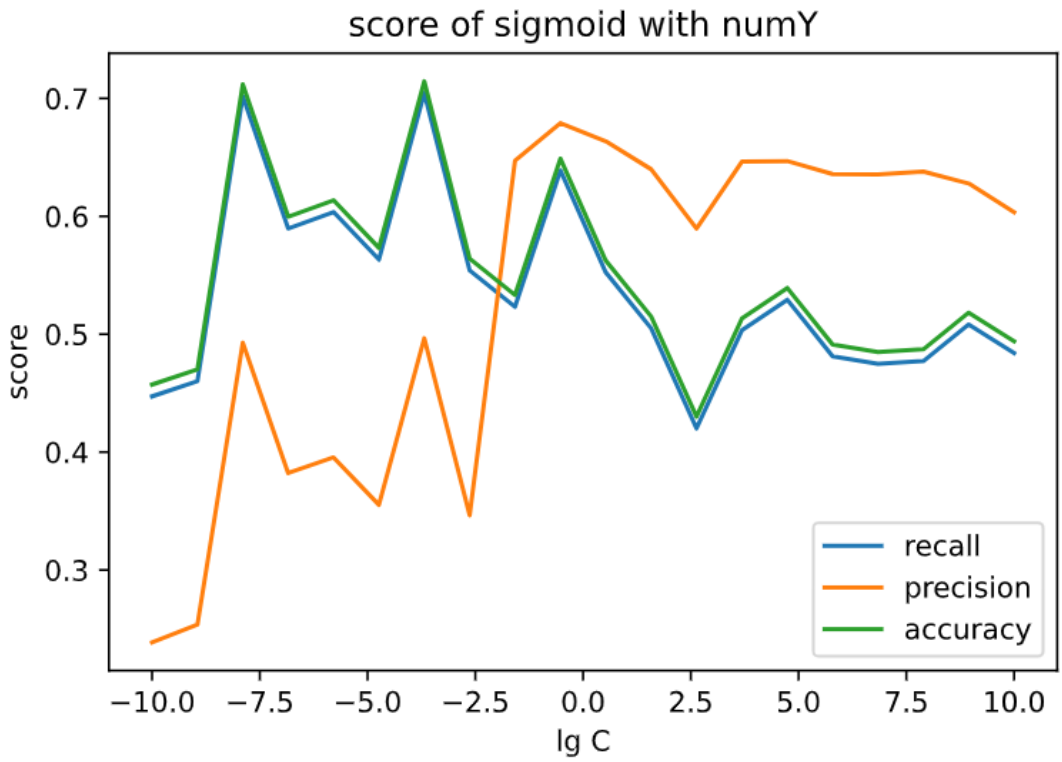

图7: 交叉验证性能与正则化强度的关系

尝试减小训练数据数目，防止过拟合，但是学习曲线如下图所示

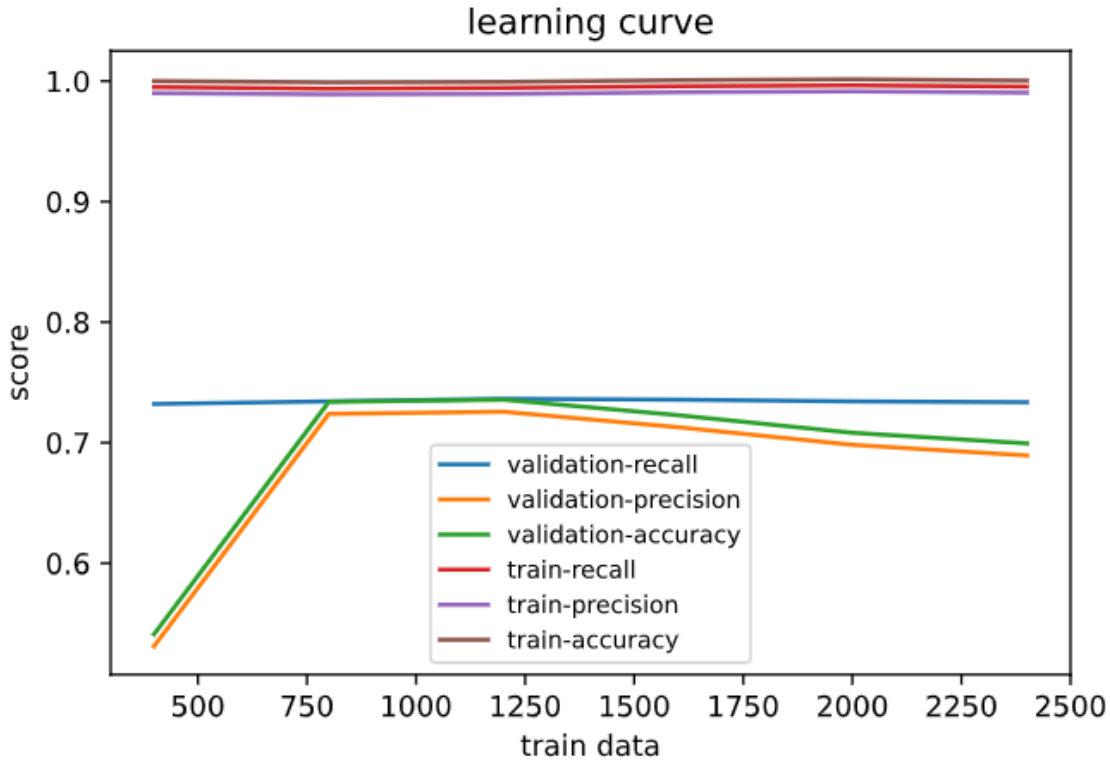

图8: 交叉验证性能与正则化强度的关系

发现学习的样本数目在1000左右时交叉验证性能最好，但是尝试发现这并没有给在测试集上的性能带来提升。

在测试过程中，我们发现了有趣的事实：直接使用successful/failed（boolY）进行训练，pearson VII核SVM得到的交叉验证结果比使用1，2，3，4(numY)的结果差。

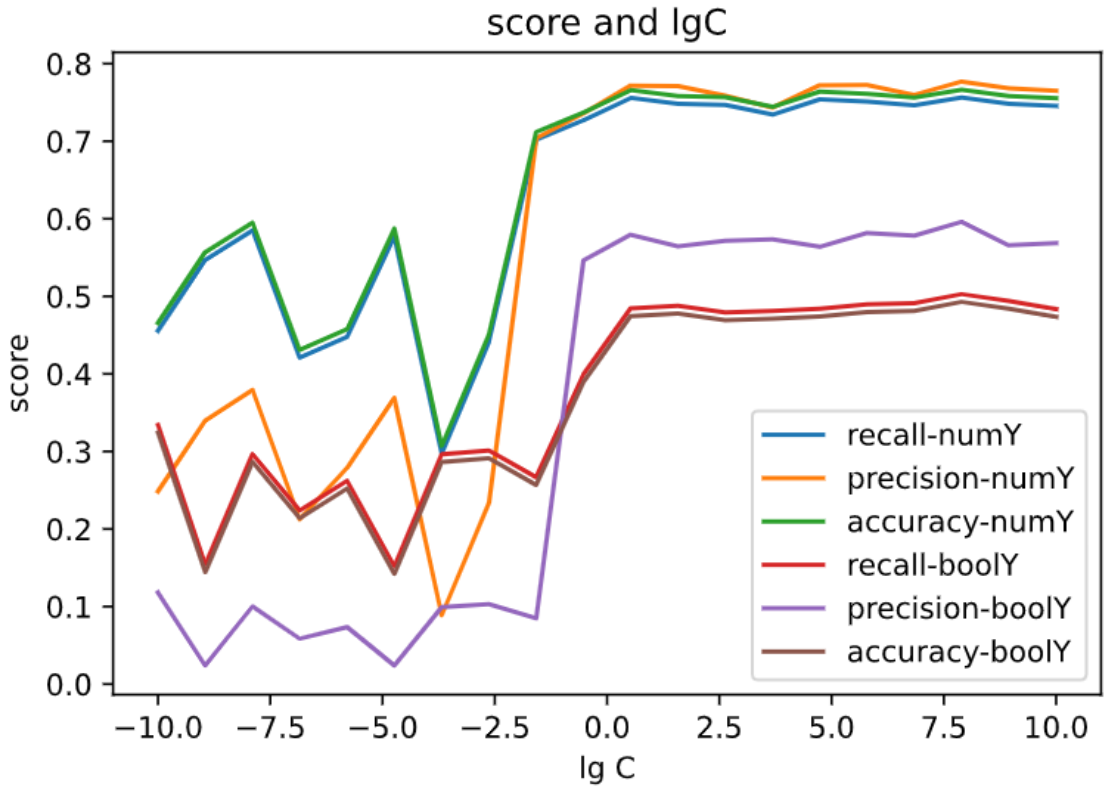

图9: 交叉验证性能与正则化强度的关系

但是更有趣的是在使用0，1作为标签后rbf核虽然交叉验证的precision下降了，但是在测试集上的效果更好了。

	prediction 1	**prediction 0**
true label 1	167	58
true label 0	16	35

accuracy =73%, precision=91%, recall=74%

而Pearson VII核性能没有提升。这一问题可能还是因为测试集的分布不同于训练集。

我们进一步想到，1，2，3，4这四个类别是有序的，而一般的多分类的one vs rest策略并不能利用这一有序性。因此我们改为直接拟合label，而不是分类。

我们采用线性模型：使用线性回归自带的normalize进行数据标准化，不考虑类别均衡问题。以2.5为阈值，如果预测值高于阈值则视为成功反应，反之视为失败反应。简单的线性拟合取得了良好的效果。在测试集上的结果如下

	prediction 1	**prediction 0**
true label 1	188	37
true label 0	44	7

accuracy =70%, precision=81%, recall=83%

考虑到有大量特征，因此进行了正则化，尝试了Lasso回归，Ridge回归，Logistic回归。三折交叉验证计算不同正则化强度下的得分，但是较简单的线性回归性能提升不大。

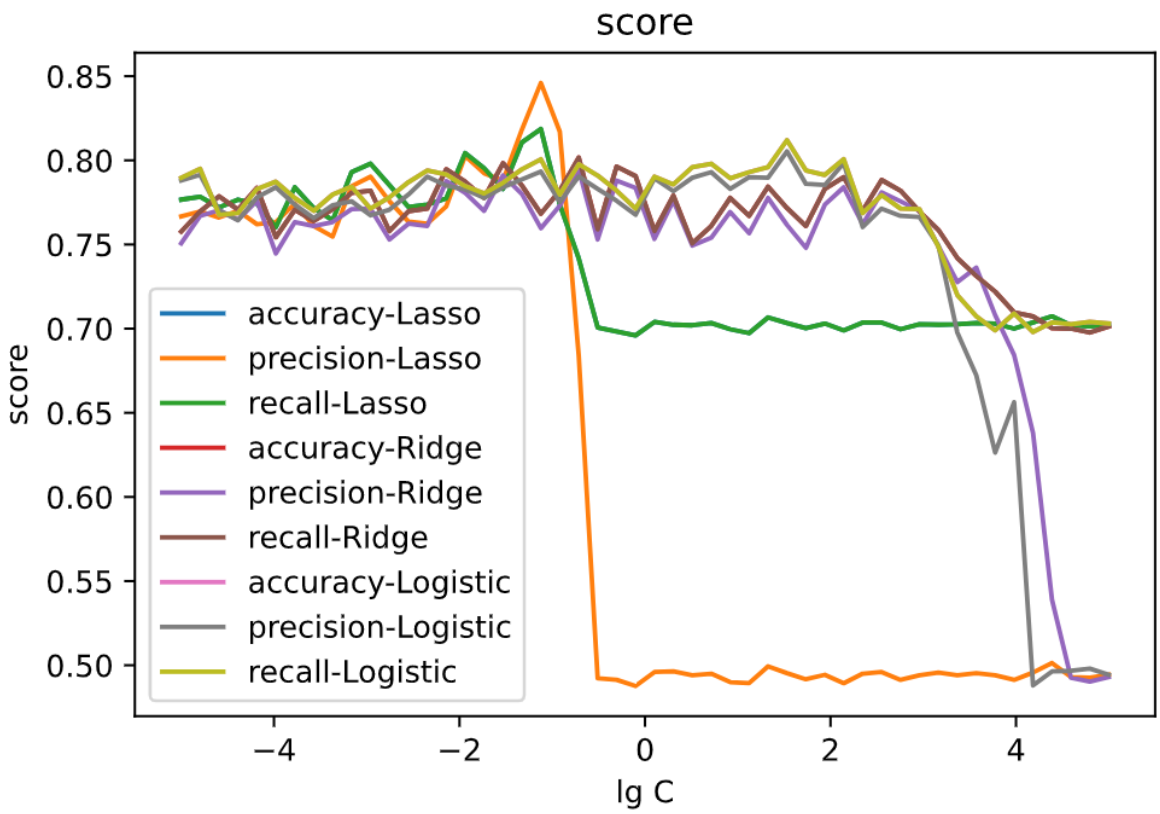

图10: 线性回归交叉验证性能与正则化强度的关系

尝试通过斜率的大小判定一个参数,但是观察到Lasso回归与岭回归在正则化强度很高时性能下降，并且出现了只能预测为成功的现象，说明将斜率小的特征忽略是不合理的。

使用性能最高C=0.1的Lasso回归对测试集进行预测。

	prediction 1	**prediction 0**
true label 1	196	29
true label 0	46	5

accuracy =72%, precision=81%, recall=87%

与简单的线性回归差别不大

尝试调整阈值，作出预测值分布的直方图，发现测试集的预测结果大多是分布在2.25-3.25之间

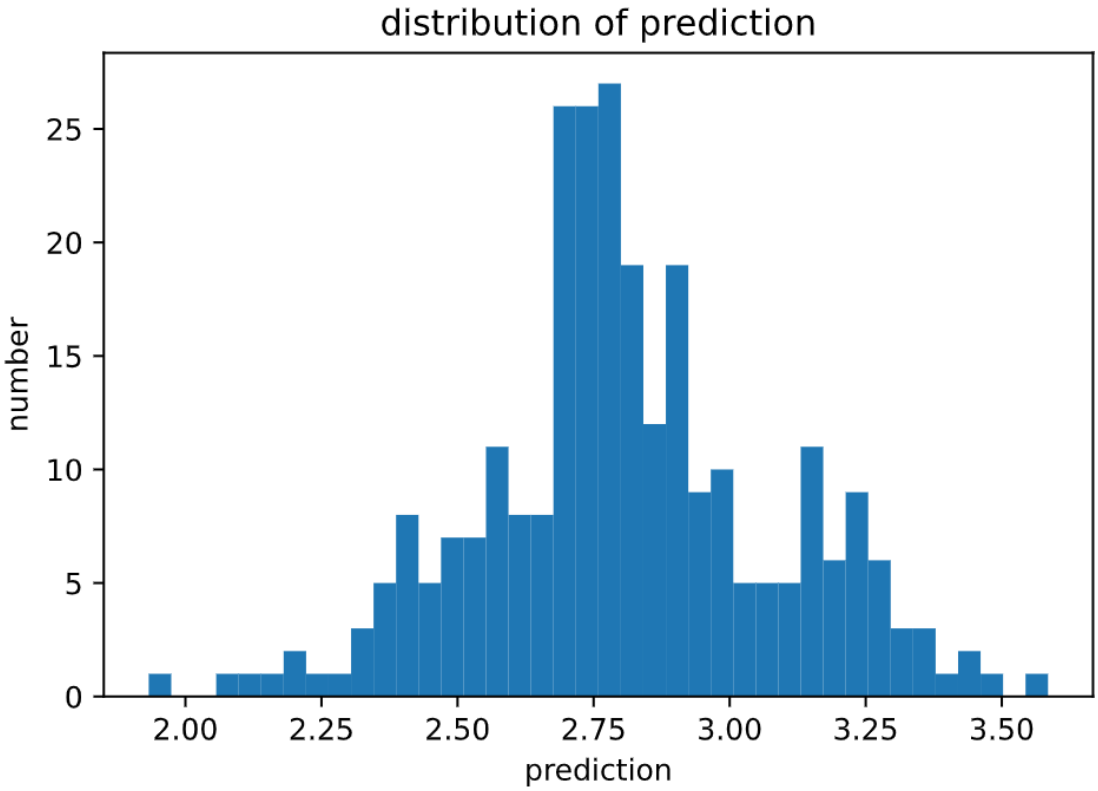

图11: 线性回归预测值分布的直方图

阈值可以在2.25到3.00之间调节，由于我们最在意的是precision，因此不做出ROC曲线而是作出precision随阈值变化的曲线。

出人意料的是阈值越大测试集上的precision越低，但是在验证集上没有这个问题。猜测还是测试集与训练集的数据显著不同。

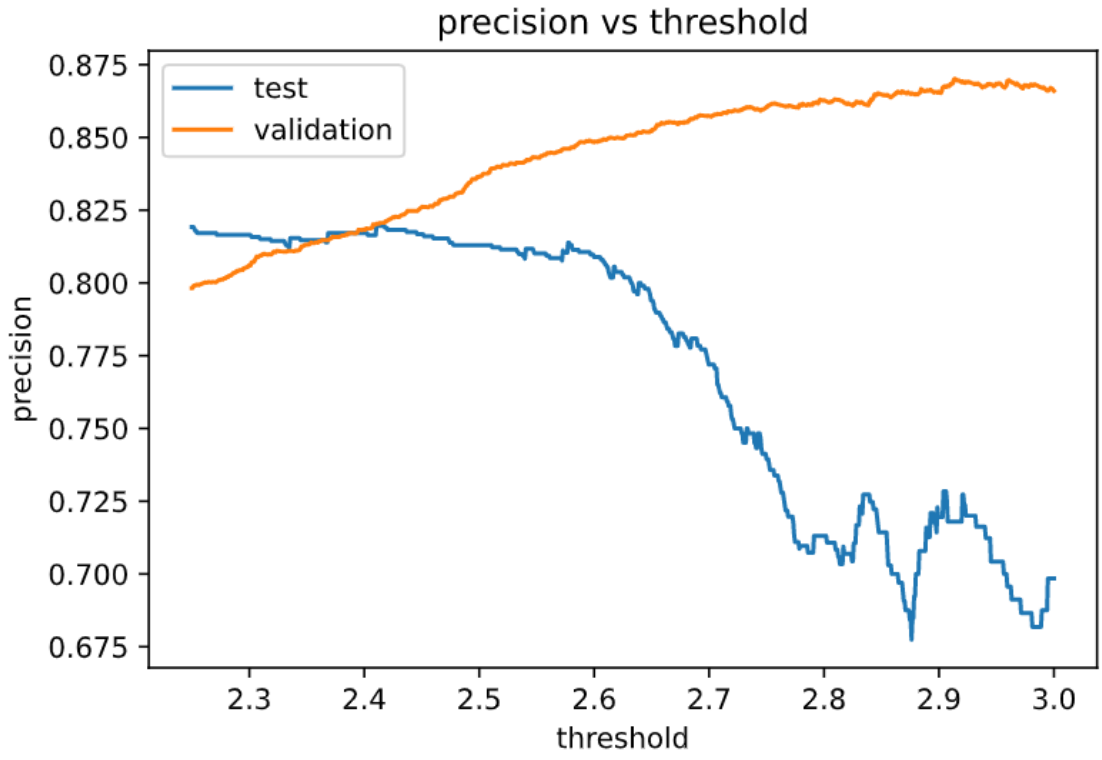

图12: 线性回归预测值分布的直方图

总结一下，各线性模型的交叉验证和在测试集上的表现如表2所示

表2: 各模型的表现

模型	CV-precision	CV-recall	test-precision	test-recall
作者(SVM(pearson VII))			89%(存疑)	
SVM(pearson VII)	76%	75%	81%	81%
SVM(rbf, numY)	75%	75%	85%	51%
SVM(rbf, boolY)			90%	73%
Lasso	84%	80%	81%	87%

文献中所用的特征选择方法sklearn没有实现，此处使用了sklearn中实现的另外两种特征选择方法：SelectKBest和RFE(recursive feature elimination)，希望对作者的结果得到某种交叉验证。

选择出来的特征和作者不尽相同，出现了文中没有提到的特征（比如'Na'/'K'/'XXXinorg2mass'），同时更多的反应条件相关的特征被选择（尤其是'pH'/'purity'在两种方法的结果中都出现了，这两个看上去倒还算是靠谱）。

SelectKBest并没有选择有机物的结构描述符，且用保留不同数目特征训练的决策树的预测结果precision变化趋势也不是单调的。用SelectKBest选择出来的结果训练的决策树表现并不尽人意。变化趋势以及决策树展示如下：

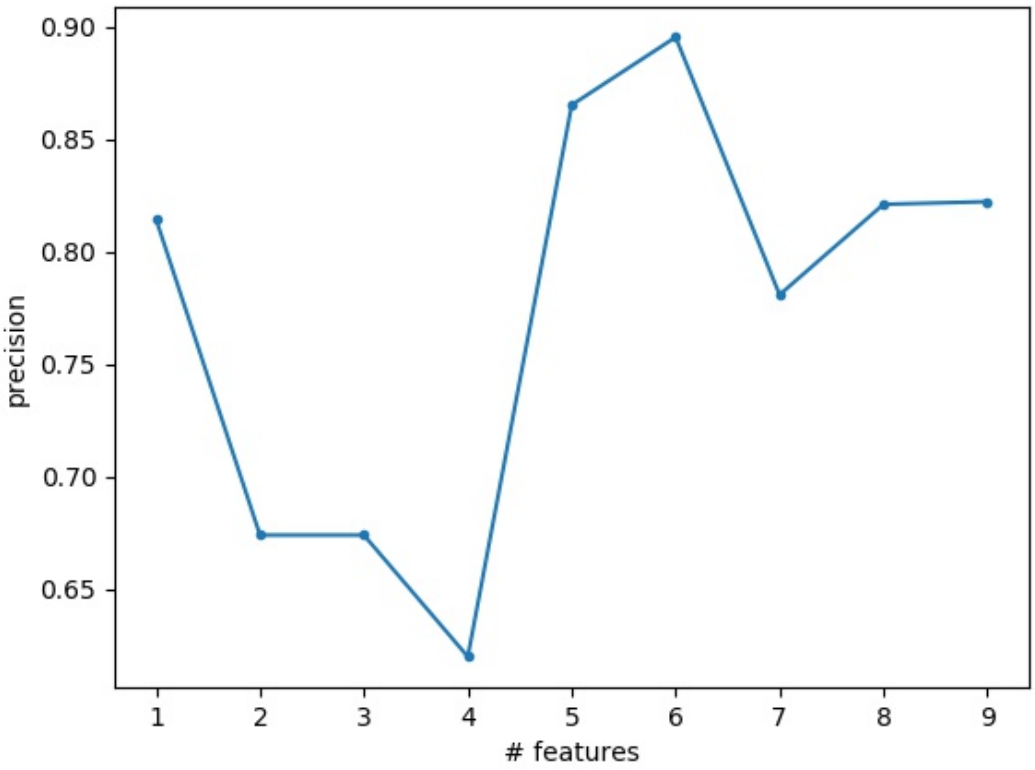

图13: 特征数目与决策树模型的表现

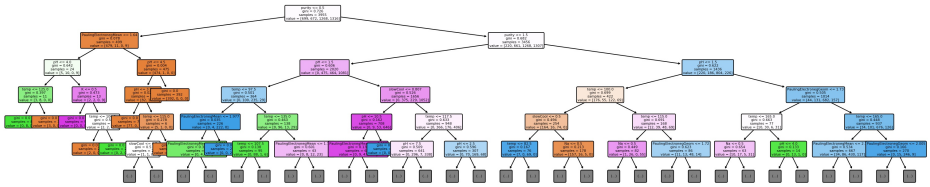

图14: 作出的决策树模型

而RFE给出了更加离谱的结果（只给出了两个负的结果，其中一个还是错的）。尽管如此，由于测试集与验证集的巨大不平衡，结果的precision、recall、accuracy看上去都非常好。作者所给出的“an accuracy of 70.7% (best-first) and 71.6% (greedy stepwise)”是否其实也是这样的结果?

我们试图训练一个小的神经网络（两层，每层8个神经元）作为分类器来从另一个角度重复上述工作。

可能是由于这个数据集的特征并不是十分明显，神经网络收敛很困难，且对于稍大的batch size（32）就基本不收敛。

在只使用作者给出的性质描述符的情况下，当不考虑训练集上的数据不平衡时（正样本数目大约是负样本数目的两倍），神经网络还算是能够比较好的识别出正样本（训练集上对正样本的recall达到0.945，测试集上为0.627），但很难识别出负样本（对负样本的recall在训练集上为0.303，测试集上0.098）。

很有意思的是，当使用调整抽样权重的方法平衡数据不平衡问题的时候，神经网络在测试集上达到对正负样本的recall分别为0.654/0.722（此时结果已经不再随着epoches增加），但在测试集上对正负样本的recall分别只有0.191和0.529. 负样本严重干扰了正样本的识别。

另一方面，考虑到文献中使用了决策树来评估特征选择的结果，我们可以进一步的尝试构建一个随机森林。使用了全数据集的结果在训练集和测试集上给出了0.834和0.841的accuracy。

随机森林给出的前九个特征重要度展示如下。使用这种方式选择出来的特征相对更加接近于作者的期望，尽管具有相同量级重要度的特征约有20多个。

表3：重要特征

rank	feature	importance
1	AtomicRadiusMinWeighted	0.0279
2	EAMinWeighted	0.0229
3	hardnessMinWeighted	0.0227
4	XXXinorg2mass	0.0224
5	PearsonElectronegMeanWeighted	0.0199
6	hardnessMaxWeighted	0.0198
7	org-water-moleratio	0.0196
8	EAMaxWeighted	0.0195
9	AtomicRadiusGeomWeighted	0.0193

这些特征衡量了原子半径，电子亲合能，硬度，电负性，有机物的亲水性。

代码结构

processData.py 数据处理的代码。这一部分的代码虽然很短，但为了处理源数据中的各种情况而没有包装，并且使用错误处理代替正常而复杂的条件判断。处理完后保存处理好的数据于./processedData。

utils.py为了简便地调用处理好的数据，以及一些通用的函数，将它们放在一个文件中。作者使用的策略是让模型产生将数据分为1，2，3，4四类，而评价模型时将1，2视作0，3，4视作1，计算precision。因此设计numout2boolout函数。

```
def numout2boolout(label):
    return label > 2.5
```

可以利用广播机制对向量进行处理，还可以转化线性回归模型的输出。

作者使用的训练集验证集划分要求同一反应组合的分在同一个集合中，因此我们不使用sklearn提供的CV函数，而是先对把反应物组合划分为测试集验证集再把组合中的数据放入测试集验证集中。

```
def CV_author(X, Y, n_splits, Model, params, scale=True, shuffle=True):
    if scale:
        X = StandardScaler().fit_transform(X)  # 标准化数据
    kf = KFold(n_splits=n_splits, shuffle=shuffle)  # 随机划分训练集与测试集中的反应组合

    for train_index_rc, test_index_rc in kf.split(reactantCombination):
        train_index = [
            i for rc in train_index_rc for i in reactantCombination[rc]]
        test_index = [
            i for rc in test_index_rc for i in reactantCombination[rc]]
        X_train, X_test = X[train_index], X[test_index]
        Y_train, Y_test = Y[train_index], Y[test_index]
        model = Model(**params)
        model.fit(X_train, Y_train)
        pred = model.predict(X_test)
        Y_test = numout2boolout(Y_test)
        pred = numout2boolout(pred)
        print("recall={:.3f}".format(
            recall_score(Y_test, pred, average='weighted')))
        print("precision={:.3f}".format(
            precision_score(Y_test, pred, average="weighted")))
        print("accuracy={:.3f}".format(accuracy_score(Y_test, pred)))
        print("confusion matrix is")
        print(confusion_matrix(Y_test, pred))
```

test函数求模型在预测集上的表现，类似CV_author。test与author函数实现了数据的标准化。

PUK_kernel是作者使用的SVM核函数，参考了以下项目:https://github.com/rlphilli/sklearn-PUK-kernel

learning_curve可以绘制学习曲线。

SVC.py 使用交叉验证方法测试不同核函数以及不同正则化强度的效果。

Linear_fit.py 通过交叉验证方法检验不同正则化强度的线性回归方法的效果将输出重定向到了./out/Ridge.out, ./out/Lasso.out, ./out/Logistic.out

feature_select.py 进行特征选择并利用选择后的特征训练决策树。

NN.py 搭建了一个两层的神经网络分类器并进行训练。

randomforest.py 训练随机森林，优化并给出特征重要度。

由于对得到的数据进行作图与分析的过程比较灵活，因此使用jupyter notebook。

dataAnal.ipynb 对得到的数据的分布进行分析。

outAnal.ipynb 对模型的输出进行分析。

每个人的工作

王希元：原始数据处理与分析(processData.py)，SVM(SVC.py)，线性回归方法(linear_fit.py), dataAnal.ipynb, outAnal.ipynb

尚游皓：神经网络(NN.py)，特征选择(feature_select.py)

连飞越：随机森林选择重要特征(randomforest.py)