Data Mining
Project Report

CLASSIFICATION AND CLUSTERING RESEARCH ON CODON USAGE

NING TANG 2020110533

December 28, 2023

School of Mathematics
Shanghai University of Finance and Economics
downing@stu.sufe.edu.cn

Contents

1 Abstract

这份报告将对数据集'Codon Usage' 中密码子使用频率数据分布进行聚类研究以及优化关于物种（'Kingdom'）分类任务的优化。在 Section2 Data Preprocess 中我们进行了数据预处理，去除了无效数据并对各类特征的统计分布进行描述与可视化。在 Section3-4 中，我们首先尝试通过 TSNE 方法对于数据进行降维，利用层次聚类方法进行标签找回，之后以物种特征为监督标签进行分类任务的研究，通过 SMOTE 重采样方法平衡数据集以及调整 GBDT 分类器，寻找到已知的最优分类器 (在测试集上 accuracy 与 balanced accuracy 分别达到 99.83% 与 78.48%)。最后我们在 Section5 中进行评估与不足分析。Code is available on https://github.com/downing777/DataMining23fall.git

2 Data Preprocess

2.1 Data Info

简要数据说明: 这个数据集是不同物种生物的 DNA 密码子的使用频率数据. 该数据集来自 CUTG 数据库, 并做了适当修改. 现该数据集内每条数据的 69 个属性如下:

1 - Kingdom: 物种, 是由 3 个字母构成, 包括 11 个物种;

2 - DNAtype: 基因组, 是 1 个整数标记, 包括 13 种构成;(Remark:type8 与 type10 并无数据样本, 故只有 11 类)

3 - SpeciesID: 物种 ID, 是 1 个整数标记, 原始 CUTG 数据库的物种标识;

4 - Ncodons: 密码子总数, 是 1 个整数;

5 - SpeciesName: 物种名称, 是 1 串字符串;

6-69 - 密码子使用频率 (frequency of usage), 是 1 个 5 位小数的浮点数, 核苷酸碱基包括'UUU', 'UUA', 'UUG', 'CUU', 等. 数据基本面板结构如下:

```
data = pd.read_csv('codon_usage.csv',low_memory=False)
data = data.set_index('SpeciesID')
```

SpeciesID	Kingdom	DNAtype	Ncodons	SpeciesName	UUU	UUC	UUA	UUG	CUU	CUC	...	CGG	AGA	AGG	GAU
100217	vrl	0	1995	Epizootic haematopoietic necrosis virus	0.01654	0.01203	0.00050	0.00351	0.01203	0.03208	...	0.00451	0.01303	0.03559	0.01003
100220	vrl	0	1474	Bohle iridovirus	0.02714	0.01357	0.00068	0.00678	0.00407	0.02849	...	0.00136	0.01696	0.03596	0.01221
100755	vrl	0	4862	Sweet potato leaf curl virus	0.01974	0.0218	0.01357	0.01543	0.00782	0.01111	...	0.00596	0.01974	0.02489	0.03126
100880	vrl	0	1915	Northern cereal mosaic virus	0.01775	0.02245	0.01619	0.00992	0.01567	0.01358	...	0.00366	0.01410	0.01671	0.03760
100887	vrl	0	22831	Soil-borne cereal mosaic virus	0.02816	0.01371	0.00767	0.03679	0.01380	0.00548	...	0.00604	0.01494	0.01734	0.04148

5 rows × 68 columns

Figure 1: Raw Dataframe

```
data.info( )
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13028 entries, 100217 to 9606
Data columns (total 68 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Kingdom      13028 non-null  object
 1   DNAtype      13028 non-null  int64
 2   Ncodons      13028 non-null  int64
 3   SpeciesName  13028 non-null  object
 4   UUU          13028 non-null  object
 5   UUC          13028 non-null  object
 6   UUA          13028 non-null  float64
 7   UUG          13028 non-null  float64
 8   CUU          13028 non-null  float64
 9   CUC          13028 non-null  float64
 10  CUA          13028 non-null  float64
 11  CUG          13028 non-null  float64
 12  AUU          13028 non-null  float64
 13  AUC          13028 non-null  float64
 14  AUA          13028 non-null  float64
 15  AUG          13028 non-null  float64
 16  GUU          13028 non-null  float64
 17  GUC          13028 non-null  float64
 18  GUA          13028 non-null  float64
 19  GUG          13028 non-null  float64
 20  GCU          13028 non-null  float64
```

Figure 2: Data type information

2.2 Preprocess and EDA

经初步探索，数据不存在重复，空值。注意到密码子'UUU' 与'UUC' 的数据类型为 object 而非 double, 可能存在异常数据类型，采用强制转化为数值类型的方式找出异常数据的索引，最终判断将其剔除 [fig:3]。同时，认为'Species Name' 是无效信息，将其剔除考虑范围。

```
data['UUU'] = pd.to_numeric(data['UUU'], errors='coerce')
data['UUC'] = pd.to_numeric(data['UUC'], errors='coerce')
data  = data.drop(['SpeciesName'], axis =1)
data.dropna(inplace=True)
```

为了进一步研究特征数据的分布与数值特征，随机挑选十二个密码子频率数据 (占比 20%) 并绘制其近似密度函数 (histogram) 与箱线图 (boxplot)[fig:4,fig:5]. 可以观察到，所随机选出的密码子频率特征的分布十分相似，接近低自由度下的卡方分布，样本点主要集中在均值附近，离群值 right skewed 且 long tail, 存在一些极端占比情况。例如有 9 个样本点'UUU' 密码子的占比达到 0.15 以上，其所属物种全部为'inv'. 密码子的极端占比可能对物种的判定有显著意

SpeciesID	Kingdom	DNAtype	Ncodons	SpeciesName	UUU	UUC	UUA	UUG
12440	vrl	0	1238	Non-A	non-B hepatitis virus	0.04362	0.0210	0.01292
353569	bct	0	1698	Salmonella enterica subsp. enterica serovar 4	12;I	-	0.0212	0.02356

2 rows × 68 columns

Figure 3: Abnormal Data

义。整体来看，如箱线图所示, 各特征的.25, .75 分位数与均值均集中在 0.02-0.04 之间，事实上对于 63 个特征全体并不存在显著异常的分布区间，在后续的工作中将尝试筛选重要特征与通过数据增强的方式增加数据的差异性，提升模型效果。

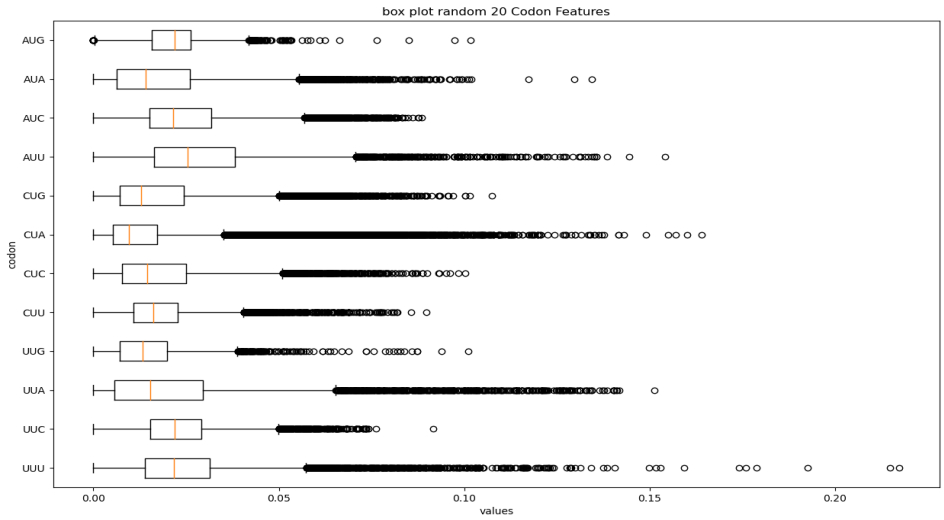

Figure 4: Random Feature Boxplot

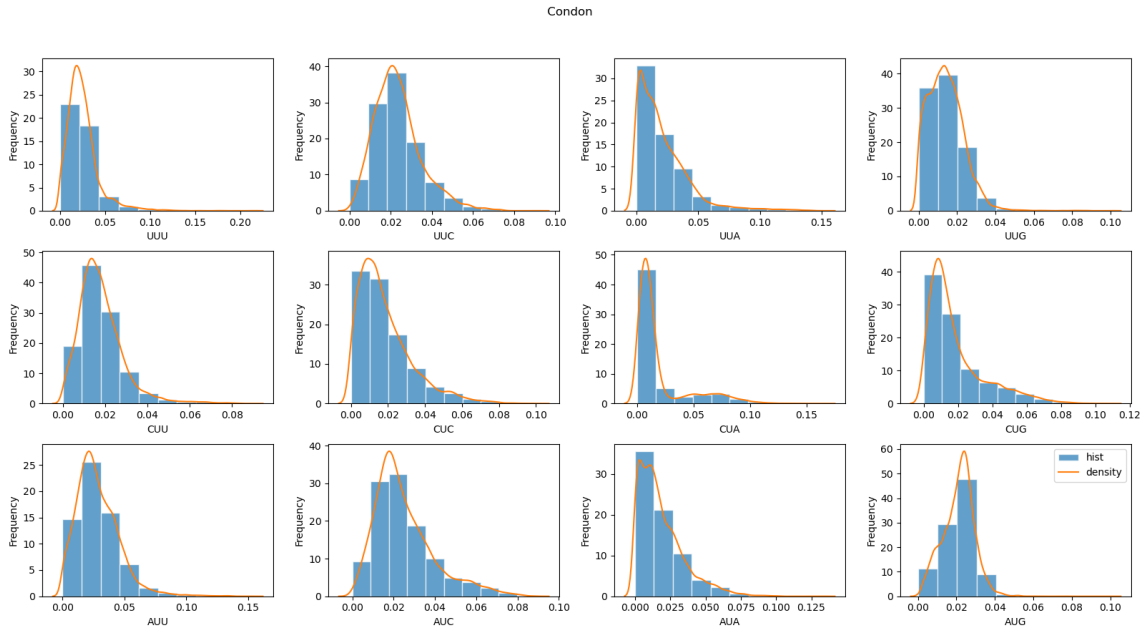

Figure 5: Random Feature Histogram

3 Clustering

在聚类分析中我们首先将所属的物种类别映射成数字类别。我们尝试通过聚类分析直接研究密码子频率数据的分布情况，这是一个极其高维的数据，首先尝试对其降维。

3.1 PCA Dimension Reduction

如 [fig:6] 所示是 PCA 降维的维数与方差解释占比的关系, 可以看到当维数较低时方差解释占比很低，有大量信息损失，当维数提升至 35 时解释占比才能达到 95%, 认为捕捉该数据特征不适合用 PCA 降维 (该数据大概率不满足线性可分的假定)。

3.2 TSNE Dimension Reduction and Clustering

我们尝试另外一种算法 TSNE[1]，这是一类非线性降维技术，降维后的数据对原始数据相似性非常好，并保持了数据点的局部结构, 被广泛用于可视化分析。降维结果如 [fig:7] 所示。

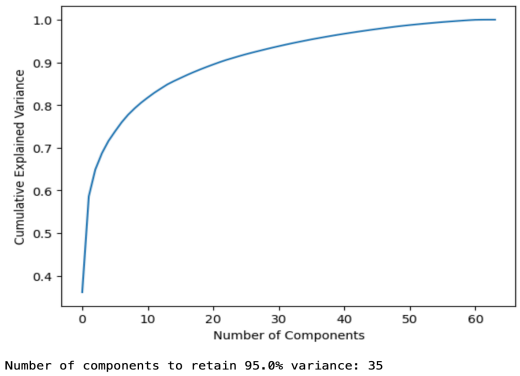

Figure 6: PCA:Dimension - CEV

```
X_std = preprocessing.scale(X)
tsne = TSNE(n_components=2)
X_transform = tsne.fit_transform(X_std)
```

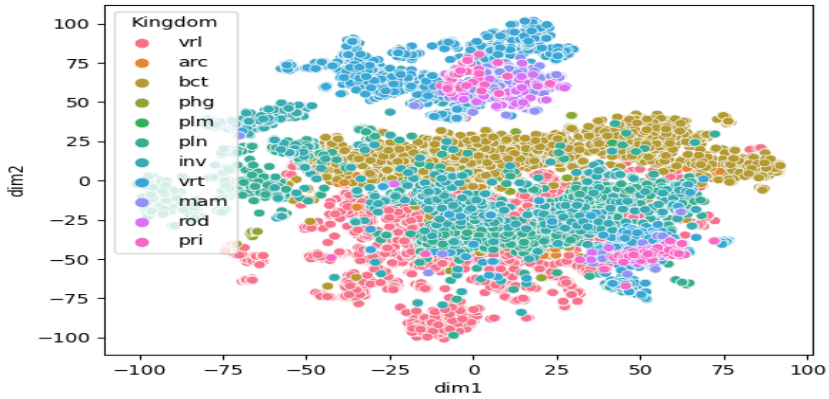

Figure 7: TSNE Reduction

观察可知降维后的数据分布得十分密集,且多样本类别如'bct', 'vrl' 呈现横向狭长的分布,Prototype-based 与 Density-based 的聚类方法可以预见的将会对这个数据集失效。考虑尝试层次聚类的方法。聚类结果如 [fig:8] 所示

当簇数 (n-clusters) 指定为 5，对于几个大类数据的找回效果 (recovery) 尚且可以，但与真实

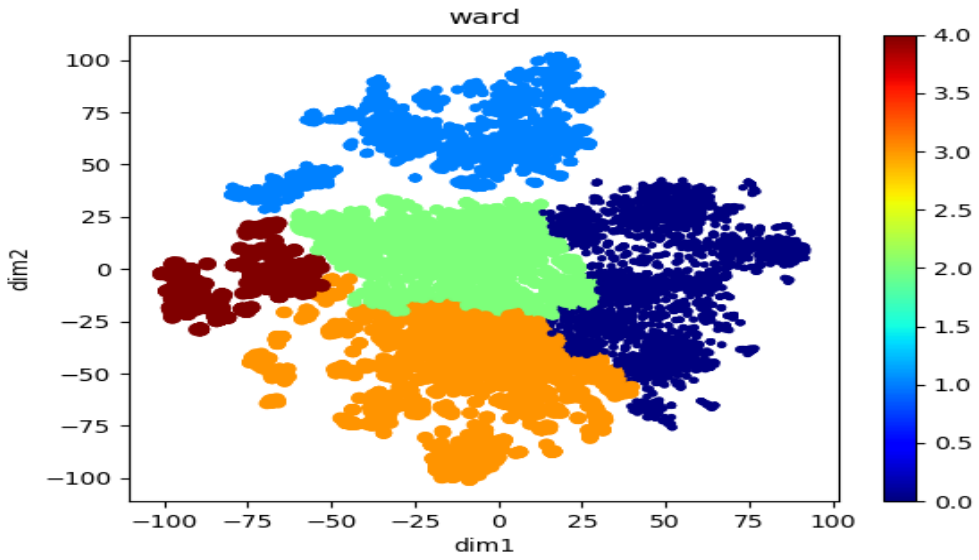

Figure 8: Hierarchy

的分布有不小的区别。当增大簇数，聚类效果将继续变差。事实上对于这样一种密集且线性不可分的数据仅通过聚类方法很难得到良好的区分效果，我们考虑有监督的分类任务。

4 Classification Task

4.1 Intuition and Related Work

数据属性'Kingdom' 与'DNAtype' 天然适合作为分类标签，在现实语境下都存在从密码子频率预测二者所属类别的研究价值，同时在该数据集中二者都存在 unbalanced labels 的问题，对于'DNAtype' 而言显得更为极端 [fig:9]。这里考虑研究通过密码子使用频率 ('Codons' frequency of usage') 作为类别预测物种 ('Kingdom') 的多分类任务。

事实上在开源社区关于该数据的'Kingdom Prediction' 分类任务有简易的测试报告，作者保留

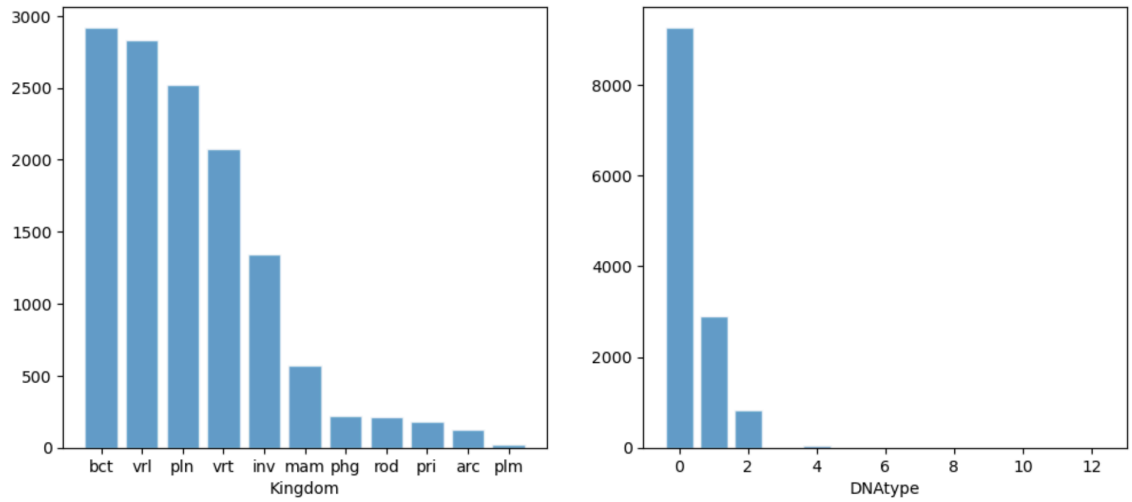

Figure 9: Labels' counts

了与标签正相关的特征并利用 package LazyClassifier 测试了 26 种分类器的分类效果 [fig:10]。其中表现最好的 XGBoost 分类器在训练集上的 Accuracy 与 Balanced Accuracy 分别达到了 0.93 与 0.78。关于这个测试，首先特征的选择笔者认为缺少统计意义上的支持，同时作为 inbalanced 分类问题，Balanced Accuracy 的结果表示分类器的分类效果并不佳而且测试集的预测评价并未给出，认为该分类结果较为粗略，可以以此为对比基础进行分类器的优化研究。

```
lazy=LazyClassifier()
models,predict=lazy.fit(x_train,x_test,y_train,y_test)
print(models)
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	\
XGBClassifier	0.93	0.78	None	0.93	
LabelSpreading	0.89	0.77	None	0.89	
LabelPropagation	0.89	0.77	None	0.89	
LGBMClassifier	0.92	0.75	None	0.92	
KNeighborsClassifier	0.89	0.74	None	0.89	
BaggingClassifier	0.87	0.69	None	0.86	
SVC	0.89	0.68	None	0.89	
RandomForestClassifier	0.89	0.66	None	0.88	
ExtraTreesClassifier	0.88	0.66	None	0.88	
DecisionTreeClassifier	0.80	0.64	None	0.80	
LinearDiscriminantAnalysis	0.77	0.59	None	0.77	
LogisticRegression	0.82	0.59	None	0.81	
ExtraTreeClassifier	0.71	0.57	None	0.71	
QuadraticDiscriminantAnalysis	0.56	0.54	None	0.55	
GaussianNB	0.55	0.53	None	0.56	
CalibratedClassifierCV	0.79	0.52	None	0.78	
LinearSVC	0.79	0.51	None	0.77	
SGDClassifier	0.78	0.51	None	0.77	
NearestCentroid	0.49	0.50	None	0.52	
Perceptron	0.72	0.50	None	0.72	
BernoulliNB	0.61	0.46	None	0.61	
PassiveAggressiveClassifier	0.67	0.45	None	0.68	
RidgeClassifier	0.72	0.35	None	0.67	
RidgeClassifierCV	0.72	0.35	None	0.67	
AdaBoostClassifier	0.35	0.21	None	0.30	
DummyClassifier	0.22	0.09	None	0.08	

Figure 10: Classifiers

4.2 Evaluation Metrics

模型的选择与优化与评价指标密不可分, 在任务中认为小样本类别和大样本类别同等重要 (在实际应用中对于小样本类别的准确预测可能至关重要, 因为这些类别可能具有特殊的生态或保护价值。)。对于一个 Inbalanced Multiclass Classification 问题中，常见的指标 accuracy 抑或是 F1-score 并不能够客观评价模型分类的效果，因此需要引进一些新的指标用于后续模型的对比与评价。

4.2.1 Recall

召回率, 指分类正确的正样本个数（TP）占真正的正样本个数（TP+FN）的比例:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

当模型倾向于预测多样本类别时，会造成精确率 (Precision) 偏高, 而 recall 值低。

4.2.2 Balanced Accuracy

对于多分类问题,Balanced Accuracy 定义为各类别 recall socres 的宏平均，即;

$$\text{Balanced Accuracy} = \frac{\sum_{i=1}^{n} \text{Recall}_i}{n}$$

这种等权平均的方式实际上增大了小样本类别的重要程度。

4.3 Ensemble Learning

从数据报告中不难发现集成学习算法所对应的分类器在该分类任务中明显强于单一分类器，例如 XGBClassifier、RandomForestClassifier，二者分别对应了集成学习中的 boosting 与 bagging 框架，其主要通过决策树生产基学习器，并综合所有基学习器的预测结果来改善单个基学习器的识别率和泛化性，在诸多实际分类任务中有着极为优异的表现效果，同时计算量小，可解释强，在深度学习模型具有极强分类能力的时代，仍保留着重要的应用与研究价值 (例如，残差神经网络 Res-Net 设计想法与 boosting 的架构有很大的相关性 [2])。由此，笔者决定基于这两类分类器，讨论集成学习的算法理论基础，并根据当前数据优化模型参数。

Bagging 架构中，每个基学习器对训练集 Bootstrap 采样得到的子集进行训练，最终投票综合得到预测类别，而 Boosting 架构则是有序训练基学习器，每一基学习器学习前一学习器的预测残差，最终加权得到预测分类。我们假设基模型的期望为 μ ，方差 σ^2 ，模型的权重为 r，基模型间的相关系数 ρ 相等。由于 Bagging 和 Boosting 的基模型都是线性组成的，那么有模型总体期望：

$$E[F] = E[\sum_{i=1}^{m} r_i f_i] = \sum_{i=1}^{m} r_i E[f_i]$$

总体方差：

$$\begin{aligned} Var[F] &= Var[\sum_{i=1}^{m} r_i f_i] \\ &= \sum_{i=1}^{r} Var[r_i f_i] + \sum_{i \neq j} Cov[r_i f_i, r_j f_j] \\ &= mr^2\sigma^2 + m(m-1)\rho r^2\sigma^2 \\ &= mr^2\sigma^2(1-\rho) + m^2 r^2 \sigma^2 \rho \end{aligned}$$

对于 Bagging 来说，每个基模型的权重等于 1/m 且期望近似相等，故我们可以得到：

$$E[F] = \mu \quad Var[F] = \frac{\sigma^2(1-\rho)}{m} + \sigma^2\rho$$

从上式可知，整体模型的期望接近基模型的期望，意味着整体偏差与基模型接近，Bagging 架构的基模型必须是强模型 (low bias, high variance), 同时增大基模型数量能够减少整体模型的方差，下限为 $\sigma^2\rho$, 一定程度上兼顾了准确度与过拟合的问题。

对 Boosting 架构而言，由于基模型共用同一套训练集，所以基模型间具有强相关性，故模型

间的相关系数近似等于 1,

$$E[F] = \sum_{i=1}^{m} r_i E[f_i] \quad Var[F] = \sigma^2$$

由上式知 Boosting 框架中的基模型必须为弱模型 (high bias, low variance)，此外 Boosting 框架中采用基于贪心策略的前向加法，整体模型的期望由基模型的期望累加而成，所以随着基模型数的增多，整体模型的期望值增加，整体模型的准确度提高。

4.3.1 Random Forest

Random Forest 是以 Classification Decision Tree 为基础的 Bagging 架构学习器，需要注意的是随机性不仅体现在了基学习器的样本的随机选择，同时对于每一 DT 的分枝过程中随机选择特征。这种方式以增大 bias 来减小方差，同时保证了随机性，无需限制特征选择与剪枝来抑制过拟合。

首先尝试无指定参数使用 RF 做分类预测, 模型结果如 [fig:11]：

```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
print("Accuracy on train data: {:.4f}".format(clf.score(X_train, y_train)))
print("Accuracy on test data: {:.4f}".format(clf.score(X_test, y_test)))
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

在不进行任何特征工程与参数调整的情况下，在训练集与测试集上的准确度分别达到 100% 与 89.3%, balanced accuracy 结果为 67%, 预测的准确度极高但可能存在过拟合的问题，对小样本类别例如'arc', 'phg', 'plm' 等的召回率 (Recall) 与 F1-score 低, 有一半以上该类别的样本点并未成功预测。

Hyperparameter Tuning

首先考虑超参数"n-estimator", 绘制其 accuracy 与 balanced accuracy 图像如下：结果显示单一调整基学习器的数量并不能够对默认 RF 有显著提升,accuracy 与 balanced accuracy 在测试集上的结果也仅达到 90% 与 67% 左右。考虑对决策树参数进行调整，限制树的最大深度 ('max depth') 以及分支样本数 ('min samples split') 避免模型在少数类别上过于拟合，使得决策树更加 robust。测试结果如下： 结果显示随着 max depth 的增加，测试集的 balanced accuracy 呈现先增后减的趋势，在 10 左右达到最大值，对于参数 Min Sample Split，其数值的增大会降低模型的精准度，但当其取值为 50 左右时, 测试集上的 balanced accuracy 提升。

```
Accuracy on train data: 1.0000
Accuracy on test data: 0.8932
              precision    recall  f1-score   support

         arc       0.82      0.35      0.49        26
         bct       0.91      0.96      0.93       753
         inv       0.93      0.67      0.78       342
         mam       0.88      0.81      0.84       137
         phg       0.91      0.54      0.67        56
         plm       0.00      0.00      0.00         6
         pln       0.88      0.93      0.91       629
         pri       0.90      0.56      0.69        50
         rod       0.85      0.60      0.71        48
         vrl       0.87      0.96      0.91       721
         vrt       0.91      0.97      0.94       489

    accuracy                           0.89      3257
   macro avg       0.81      0.67      0.72      3257
weighted avg       0.89      0.89      0.89      3257
```

Figure 11: Default Random Forest

认为可以调整这两个参数，分别取为 10，50，牺牲一部分的训练集上的准确度以提升小样本类别的预测精度，在经过 10-fold 交叉检验之后，最终测设报告如 [fig:15] 所示，训练集与测试集上的 accuracy 分别达到 95.3% 与 87.1%，测试集上的 balanced accuracy 达到 76.2%，较 default RF 提升了 14%:

```
clf = RandomForestClassifier(class_weight= 'balanced',max_depth = 10,...
    min_samples_split=50,random_state=42)
clf.fit(X_train, y_train)
print(classification_report(y_test, clf.predict(X_test)))
```

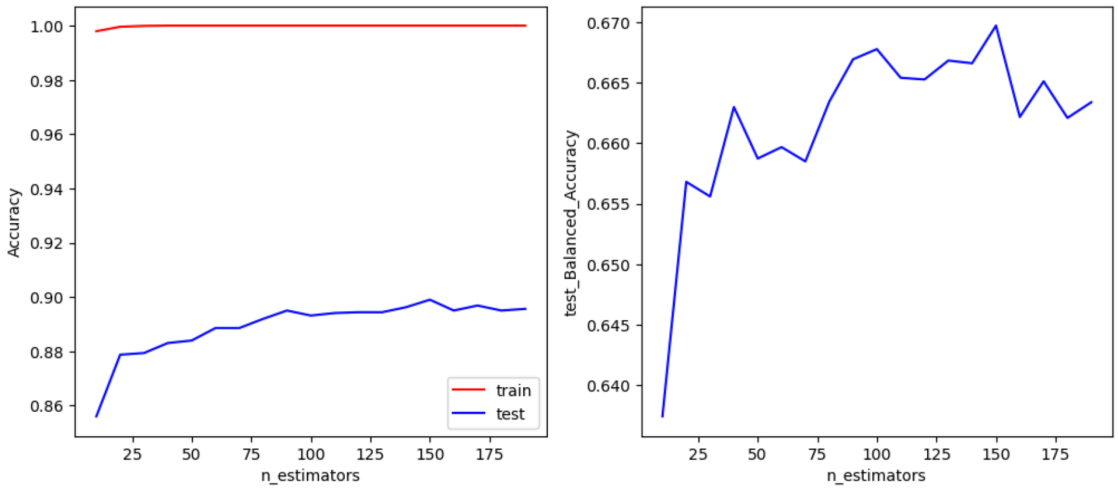

Figure 12: Estimators

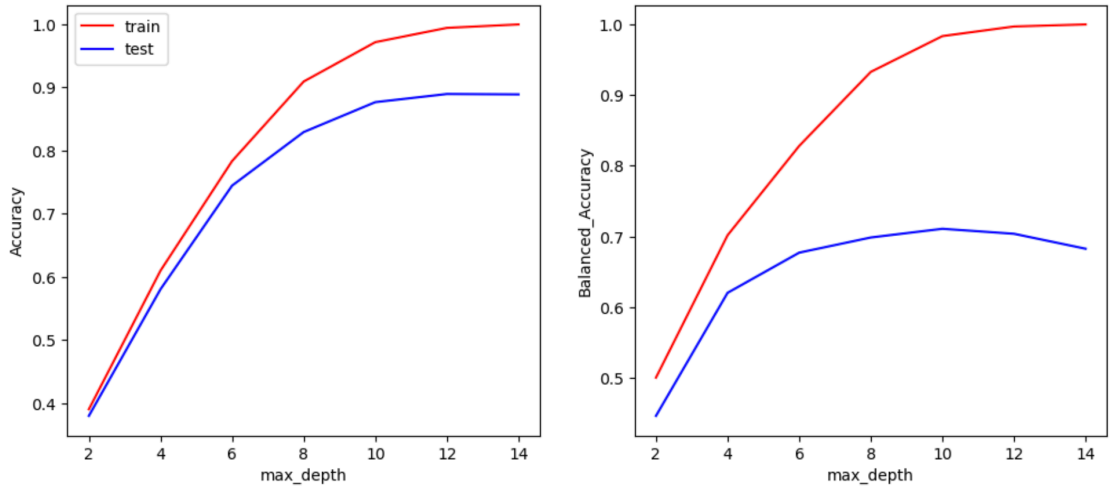

Figure 13: Max Depth

4.3.2 Gxradient Boosting Decesion Tree

与 RF 不同的是，GBDT 的基分类器是回归树 (Regression Decision Tree)[3], 回归树在分枝时会穷举每一个特征的每个阈值以可微损失函数 L 为标准找到最好的分割点, 模型的预测值为：

$$F_m(x) = \sum_{i=1}^{m} f_i(x)$$

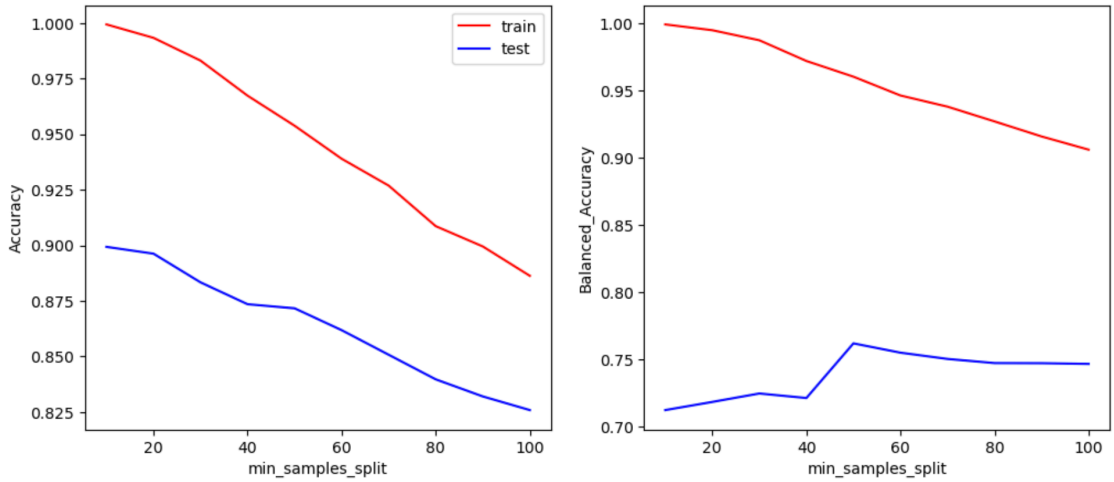

Figure 14: Min Sample Split

由于要同时考虑所有基模型，导致了整体模型的训练变成了一个非常复杂的问题。实际上模型的训练依赖于贪心的算法：

$$F_k(x) = F_{k-1}(x) + f_k(x)$$

此时，当我们训练好 k 棵树，根据样本特征，样本会落在每一个基学习器的某一叶子节点上并得到一个预测值，当前预测值即为所有预测值相加。对于分类问题，可以采取 Softmax 函数进行类别预测，L 为对数似然函数。Gradient 体现在于每一基学习器学习的残差可近似认为是损失函数关于当前预测值的负梯度 (在回归问题 MSE 损失函数下这是严格成立的)：

$$f_k(x) = y - F_k(x) \approx -\frac{\partial L(y, F_k(x))}{\partial F_k(x)}$$

在 package sklearn 中利用 HistGradientBoostingClassifier 实现了类似 XGBoost 的工程，在大样本量的数据集上 (n sample $> 10^4$) 高效计算 GBDT 算法。与 GBDT 略有不同的是，其显示的加入了参数的正则化项并在训练过程中利用了损失函数一阶与二阶导数信息，使模型自身具有良好的泛化性与稳健性。在 defalut 设置下，Balanced Accuracy 在训练集和测试集上分别达到了 1 与 72%，非常接近甚至优于优化过的 RF。

```
clf = HistGradientBoostingClassifier()
clf.fit(X_train, y_train)
train_pred = clf.predict(X_train)
test_pred = clf.predict(X_test)
```

	precision	recall	f1-score	support
arc	0.43	0.73	0.54	26
bct	0.90	0.92	0.91	753
inv	0.79	0.77	0.78	342
mam	0.88	0.78	0.83	137
phg	0.70	0.71	0.71	56
plm	1.00	0.17	0.29	6
pln	0.89	0.88	0.88	629
pri	0.60	0.80	0.68	50
rod	0.65	0.83	0.73	48
vrl	0.89	0.90	0.89	721
vrt	0.98	0.89	0.93	489
accuracy			0.87	3257
macro avg	0.79	0.76	0.74	3257
weighted avg	0.88	0.87	0.87	3257

Figure 15: Tuned Random Forest

```
print("Balanced Accuracy on train data: {:.4f}".format(balanced_accuracy_score(...
        y_train, train_pred)))
print("Balanced Accuracy on test data: {:.4f}".format(balanced_accuracy_score(...
        y_test, test_pred)))
>>>Balanced Accuracy on train data: 1.0000
>>>Balanced Accuracy on test data: 0.7230
```

类似于 RF 的方式调整 HGBT 并没有得到显著的优化，尝试通过重采样的方式处理数据集以近一步提升对小样本的分类能力。

4.3.3 SMOTE Resampling

SMOTE 是一类经典的处理不平衡分类数据的算法 [4], 其基本想法是在少数类别中随机取样 (n-sample = N), 之后对每一个样本寻找 KNN's(k default = 5), 最后利用 K 个近邻样本插值形成少数类别的合成样本。重采样之后训练集中的样本会取向均衡。

```
from imblearn.over_sampling import SMOTE
oversampler=SMOTE(random_state=42)
os_X_train,os_y_train=oversampler.fit_resample(X_train,y_train)
clf = HistGradientBoostingClassifier(random_state=42)
clf.fit(os_X_train, os_y_train)
>>>Accuracy on test data: 0.9984
>>>Accuracy on test data: 0.9260
>>>Balanced Accuracy on train data: 0.9983
>>>Balanced Accuracy on test data: 0.7848
```

	precision	recall	f1-score	support
arc	0.80	0.62	0.70	26
bct	0.95	0.95	0.95	753
inv	0.87	0.85	0.86	342
mam	0.90	0.89	0.89	137
phg	0.73	0.68	0.70	56
plm	1.00	0.33	0.50	6
pln	0.92	0.94	0.93	629
pri	0.84	0.64	0.73	50
rod	0.80	0.81	0.80	48
vrl	0.93	0.96	0.95	721
vrt	0.96	0.97	0.96	489
accuracy			0.92	3257
macro avg	0.88	0.78	0.82	3257
weighted avg	0.92	0.92	0.92	3257

Figure 16: SMOTE+HGBT

结合训练结果可知，在训练集与测试集上模型 accuracy 分别达到 99.84% 以及 99.26%，balanced accuracy 达到 99.83% 与 78.48%，不仅准确度极高，少数样本类别上的分类效果也得到提升，是目前找到的最优分类器。

5 Conclusion

通过 SMOTE 重采样方法平衡数据集以及调整 GBDT 分类器，得到已知的最优分类器，在测试集上 accuracy 与 balanced accuracy 分别达到 99.83% 与 78.48%。过程中可以认为我们几乎没有牺牲分类器准确率而使得分类器对于小样本类别的预测能力提升了 12%，是一个非常不错的结果。由于算力限制，继续对于分类器进行精细的大规模调参十分困难且耗时，可以预见仍有很大的优化空间，这里笔者不做继续研究。

References

[1] Wiki, "Tsne." [Online]. Available: https://en.wikipedia.org/wiki/T-distributed-stochastic-neighbor-embedding

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[3] Z. Zhihua, "Ensemble learning." [Online]. Available: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerEBR09.pdf

[4] S. SATPATHY, "Smote." [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/