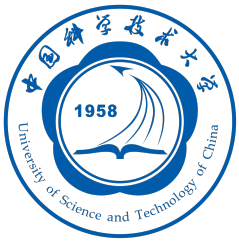


中国科学技术大学

University of Science and Technology of China

数据分析与实践 Lab5
机器学习分类模型训练与预测

姓名：邱文韬

学号：JL22110003

专业：计算机科学与技术

学院：计算机科学与技术学院

2023 年 5 月 23 日

摘要

本文基于 Pisa 2018 数据集，在数据分析与实践 Lab3：**数据分析与特征处理**的特征工程基础上，进一步进行特征处理，使特征能够适应模型需求，尝试训练决策树、随机森林、支持向量机以及 *KNN* 等分类模型，并根据模型的表现尝试不同的特征，并且发现特征的选择对模型性能的影响最大，对数据集采用交叉验证的方法，验证模型在不同数据集上的泛化能力，并不断调整模型参数，同样使用交叉验证的方法评估参数的好坏。最终得到的四个模型中，随机森林、支持向量机以及 *KNN* 的表现较好，*accuracy* 均在 0.83-0.85 范围，*f1-score* 在 0.80-0.84 范围，而决策树的表现相对较差，*accuracy* 为 0.78，*f1-score* 为 0.79。在实验过程中也遇到了一些问题，如数据集正负样本不平衡，导致模型预测 REPEAT=1 的表现相对较差，失败的过程记录详见第7节，以随机森林为例，其中尝试使用随机下采样以及 *SMOTE* 过采样平衡正负样本，随机下采样取得的 *accuracy* 为 0.73，*f1-score* 也为 0.73，*SMOTE* 过采样取得的 *accuracy* 为 0.83，*f1-score* 为 0.80。

关键字： 决策树，随机森林，支持向量机，*KNN*，交叉验证

目录

1 实验内容

1.1 实验介绍

实验五的任务基于实验三，是在实验三实现的数据分析基础上的拓展与延伸；

- 同样使用 PISA2018 数据集；
- 同学们须实现至少一种分类算法 (例如：决策树、KNN、朴素贝叶斯或者感知机、集成算法等)；
- 参考实验三中的特征工程，测试算法在 PISA2018 数据集上的预测性能，并撰写实验报告。
- 实验报告需要记录最终的方案流程，也鼓励大家记录每一次失败的尝试。

1.2 实验要求

- 代码实现可以使用现有的机器学习库，也可以自行编写实现算法
- 预测任务与实验三一致（实验三只是围绕预测目标进行数据分析和特征工程），即预测学生**是否会选择复读** (REPEAT)，并以**准确率 (*ACC*)** 作为评价指标 (也可以使用其它指标)。
- 请自行在 PISA2018 数据集上划分训练集和测试集 (4:1 比例、**交叉验证**)，汇报算法在测试集上的性能。
- 实验报告需包括实现算法的主要流程、关键技术以及算法的性能。

2 特征选取与特征处理

2.1 特征选取

首先，根据实验 3 的特征工程，初步选取以下特征，如表1，后续模型训练将选取他们的特征子集，以及尝试不同的特征组合等。

2.2 特征处理

参考实验 3 中的工作，对于连续型变量，考虑对缺失值使用均值填充，对异常值使用平均值替代或者删去，再进行归一化处理，与前述的工作类似，但对于离散变量，如 OCOD2，ESCS 等，考虑进行独热编码或者标签化，而 IC150Q06HA-IC152Q09HA 部分特征的缺失值如果直接删去会损失较多的数据，所以选择使用 *KNN* 进行缺失值填充，待预测特征存在样本不平衡的问题将在后文介绍以及其解决方法 [1]。

表 1: 特征选取

特征名称	特征含义
ISCEDL	ISCED level
HISEI	Index highest parental occupational status
BMMJ1	ISEI of mother
BFMJ2	ISEI of father
BSMJ	Students expected occupational status (SEI)
ESCS	Index of economic, social and cultural status
OCOD2	ISCO-08 Occupation code - Father
COBN_F	Country of Birth National Categories- Father
IC150Q06HA-IC152Q06HA	Time On Music
IC150Q08HA-IC152Q08HA	Time On PeformingArts
IC150Q09HA-IC152Q09HA	Time On VisualArts

3 分类算法

3.1 决策树

决策树 [2] 是一种基于树结构来进行决策的机器学习算法。它通过将数据集分成不同的子集，每个子集对应树上的一个节点，从而构建出一棵决策树。决策树的每个内部节点表示一个特征或属性，每个叶子节点表示一个类别。

使用决策树时，应注意特征选择应该考虑特征的熵、信息增益或基尼系数等指标，以选择最优的特征进行分裂。决策树容易过拟合，需要进行剪枝。决策树对于异常值和噪声敏感。

3.2 随机森林

随机森林 [3] 是一种基于决策树的集成学习算法。它通过使用多棵决策树来进行分类或回归，最终的结果是多棵决策树的预测结果的平均值或投票结果。

随机森林对于高维数据和大规模数据的处理能力较强。随机森林可以处理非线性关系的数据。随机森林的训练速度相对较慢，但预测速度较快。

3.3 支持向量机

支持向量机 [4] 是一种用于分类和回归的机器学习算法。它通过找到一个超平面，将样本分成不同的类别。超平面的选择是通过最大化间隔来实现的，即找到一个距离两个类别最近的样本的距离最大的超平面。支持向量机适用于二分类和多分类问题，但对于多分类问题

需要进行一些扩展。选择合适的核函数和超参数对支持向量机的性能影响很大。支持向量机对于数据量很大的情况下，训练时间会比较长。

3.4 *K* 近邻

K-近邻 [5] 算法是一种基于距离度量的机器学习算法。它通过找到和输入样本最近的 k 个邻居，并根据它们的类别来预测输入样本的类别。

K-近邻算法的建立过程包括选择距离度量、选择 k 值和分类。选择距离度量是指选择一种能够衡量样本之间距离的方法，如欧氏距离、曼哈顿距离等。选择 k 值是指选择一个合适的邻居个数，可以通过交叉验证等方法来确定。

K-近邻算法对于噪声和异常值敏感。选择合适的距离度量和 k 值对算法的性能影响很大。当特征维度很高时，*K*-近邻算法的性能可能会降低，需要进行降维或特征选择等操作来提高算法的效率。

4 模型构建

4.1 算法流程

使用 *sklearn* 中的 *DecisionTreeClassifier*、*RandomForestClassifier*、*SVC* 以及 *KNeighborsClassifier*，通过模型间的对比，最终确定随机森林的性能最佳，而对模型影响最大就是特征的选取以及处理，因此，实现算法时首先需要确定选取的特征是否正确，其次需要对特征进行一系列变换适应对应的模型，特征处理完毕后，对数据进行交叉验证，选取能够取得最佳得分的训练模型，然后再选取不同的特征组合或者子集，对比使用不同特征训练出的模型取得的效果，再对模型的参数进行调整，从而进一步提高算法的性能。

4.2 初步探索尝试

使用前述特征的子集：BFactor 、OCOD2、COBN_F 、ESCS 、HOMEPOS，其中 BFactor 为 BMMJ1+BFMJ2+HISEI+BSMJ 的组合特征，根据 *ROC* 曲线可以看出，这部分特征训练出的模型效果均不理想，选择准确率 (accuracy) 作为交叉验证的得分，结果如表2，进行 5 折交叉验证，得分差异不大，准确率最高为 0.8 左右，其中决策树的准确率最低。

其中，*SVM* 模型的训练时间过长，查询资料发现原因是，为了绘制 *SVM* 的 *ROC* 曲线，需要将 probability 设置为 True，而在 *Scikit-learn* 中，如果 *SVM* 模型的 probability 参数设置为 True，那么模型会使用 *Platt scaling* 或 *sigmoid calibration* 方法来估计样本属于正类的概率，这会增加模型的计算复杂度和运行时间，将其设置为 False 后，运行时间减少至 25s，相比较有大幅减少。

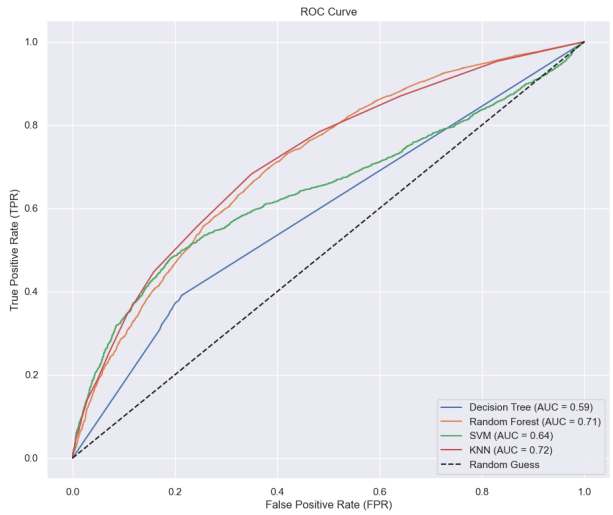


图 1: 模型 ROC 曲线

表 2: 不同模型的交叉验证得分和平均得分

模型	**交叉验证得分**	**平均得分**	**训练时间**	**accuracy**	**f1-score**
决策树	[0.7167, 0.7245, 0.7242, 0.7245, 0.7117]	0.720	0.187 秒	0.7181	0.721
随机森林	[0.8033, 0.7995, 0.8028, 0.7959, 0.8021]	0.801	5.014 秒	0.8012	0.765
支持向量机	[0.8011, 0.7969, 0.7986, 0.7953, 0.8024]	0.799	164.032 秒	0.8011	0.726
KNN	[0.8042,0.8032, 0.8019, 0.8018, 0.8032]	0.803	0.048 秒	0.8042	0.767

表 3: 不同模型的混淆矩阵和分类报告

模型	**混淆矩阵**	**分类报告**
Decision Tree	$\begin{bmatrix} 5460 & 1269 \\ 1109 & 598 \end{bmatrix}$	precision recall f1-score support 0 0.83 0.81 0.82 6729 1 0.32 0.35 0.33 1707
Random Forest	$\begin{bmatrix} 6378 & 351 \\ 1326 & 381 \end{bmatrix}$	precision recall f1-score support 0 0.83 0.95 0.88 6729 1 0.52 0.22 0.31 1707
SVM	$\begin{bmatrix} 6683 & 46 \\ 1632 & 75 \end{bmatrix}$	precision recall f1-score support 0 0.80 0.99 0.89 6729 1 0.62 0.04 0.08 1707
KNN	$\begin{bmatrix} 6469 & 260 \\ 1392 & 315 \end{bmatrix}$	precision recall f1-score support 0 0.82 0.96 0.89 6729 1 0.55 0.18 0.28 1707

5 参数调整

5.1 *KNN* 调参

使用网格化搜索 *KNN* 的最佳邻居个数，将其搜索过程可视化如图2，综合考虑计算时间以及准确率，选择邻居个数 17，后续更多的邻居个数对准确率的提升非常低，只能够提升 0.01-0.05 左右。加权方式选择 *distance*，即根据邻居之间的距离进行加权。较近的邻居会具有较高的权重，而较远的邻居会具有较低的权重。但调整参数后的 *KNN* 准确率提升并不大。

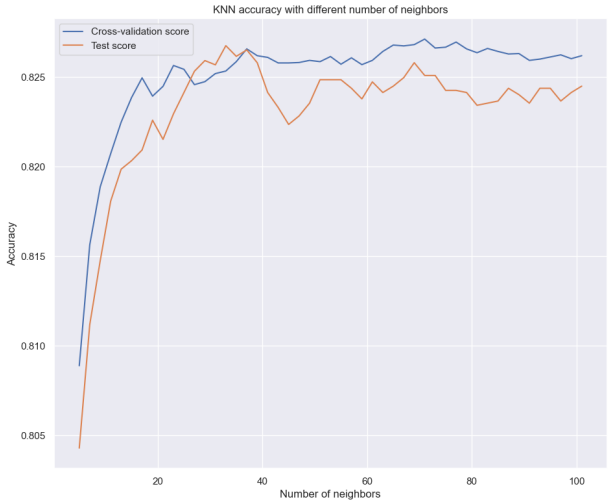


图 2: *KNN* 邻居个数与准确率变化

5.2 随机森林调参

通过网格搜索调整随机森林的参数。在调整过程中，我将参数范围设置为单一值，即最大深度为 10 和决策树数量为 100。经过交叉验证的评估，发现这组参数配置可以取得最佳的性能表现。有助于防止过度拟合，同时保持模型的预测能力。而决策树的数量被设置为 100。通过增加决策树的数量，可以提高模型的鲁棒性和泛化能力。

5.3 决策树调参

选择使用基尼指数 [6] 作为衡量决策树分裂质量的标准。基尼指数是一种衡量节点不纯度的指标。决策树的最大深度设置为 10，通过限制树的深度，防止过拟合，并促使模型更好地泛化。将叶节点所需的最小样本数设置为 2，意味着当节点上的样本数量小于 2 时，将

不再进行进一步的分裂，从而避免创建具有非常少样本的叶节点。设置分裂内部节点所需的最小样本数为 10。这样做可以确保每个内部节点至少有 10 个样本才会进行进一步的分裂。

5.4 *SVM* 调参

选择较小的 C 值，C 值代表错误分类样本的惩罚程度，这意味容忍错误分类的程度较高，而更强调模型的泛化能力。选择了阶数为 2 的多项式核函数，可以更好地拟合数据，同时避免过度复杂化模型。我选择了较大的核函数系数：1，模型在训练样本中更加敏感，有可能导致过拟合。但在调参过程中，发现较大的 gamma 值可以更好地捕捉特征之间的关系。使用径向基函数 (*RBF*) 作为 *SVM* 模型的核函数，因为我观察到数据集在原始特征空间中可能是非线性可分的，使用 *RBF* 核函数可以提高模型的灵活性和拟合能力。

6 最终方案与结果

6.1 方案

由于上述特征的表现不佳,选择使用前述的所有特征:IC150Q06HA-IC152Q09HA、OCOD2、COBN_F，BMMJ1、ISCEDL 等用于训练模型，根据 *ROC* 曲线, 如图3，可以看出随机森林算法的效果比较好，而交叉验证的结果显示，见表4，数据集的选择对模型性能影响不大，这几个模型对数据集具有较好的泛化能力。

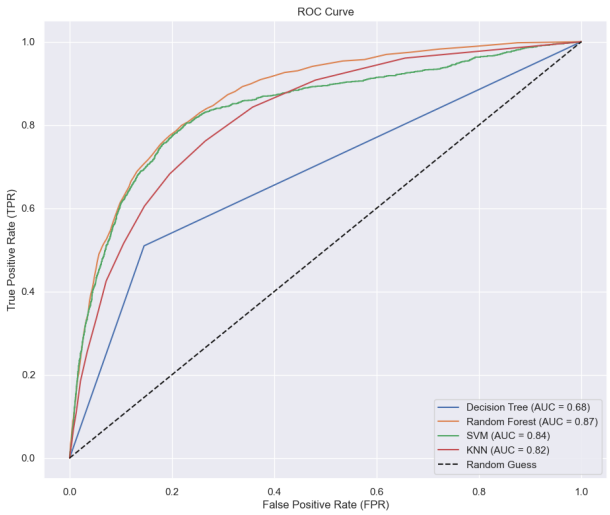


图 3: 模型 *ROC* 曲线

根据表5，可以看出数据正负样本不平衡的问题导致模型在预测 REPEAT = 1 的是表现相对更差，无论是精确率、召回率还是 F1 分数都要更低，所以可以考虑使用下采样或者

表 4: 不同模型的交叉验证得分和平均得分

模型	交叉验证得分	平均得分	训练时间	accuracy	f1-score
决策树	[0.783, 0.783, 0.790, 0.781, 0.781]	0.783	0.312 秒	0.786	0.785
随机森林	[0.850, 0.850, 0.853, 0.847, 0.842]	0.848	6.076 秒	0.849	0.841
支持向量机	[0.843, 0.842, 0.843, 0.840, 0.845]	0.842	22.186 秒	0.843	0.829
KNN	[0.824, 0.821, 0.823, 0.818, 0.830]	0.823	0.005 秒	0.824	0.807

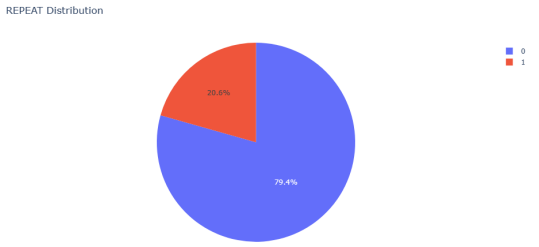
过采样让样本更平衡，但由于实验效果不佳，即虽然 REPEAT=1 的预测准确率提高了，但总体准确率降低了，所以最终没有选择该方案，具体实现过程在第7节过程记录中有介绍。

表 5: 不同模型的混淆矩阵和分类报告

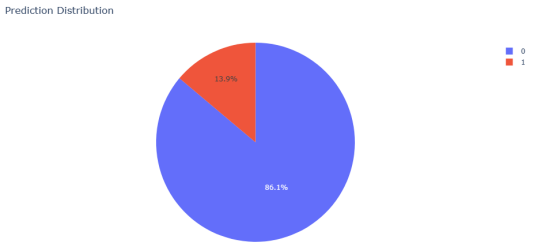
模型	混淆矩阵	分类报告
Decision Tree	$\begin{bmatrix} 5761 & 968 \\ 839 & 868 \end{bmatrix}$	precision recall f1-score support 0 0.87 0.86 0.86 6729 1 0.47 0.51 0.49 1707
Random Forest	$\begin{bmatrix} 6338 & 391 \\ 880 & 827 \end{bmatrix}$	precision recall f1-score support 0 0.88 0.94 0.91 6729 1 0.68 0.48 0.57 1707
SVM	$\begin{bmatrix} 6426 & 303 \\ 1023 & 684 \end{bmatrix}$	precision recall f1-score support 0 0.86 0.95 0.91 6729 1 0.69 0.40 0.51 1707
KNN	$\begin{bmatrix} 6377 & 352 \\ 1134 & 573 \end{bmatrix}$	precision recall f1-score support 0 0.85 0.95 0.90 6729 1 0.62 0.34 0.44 1707

6.2 结果可视化

将准确率最高的随机森林模型预测结果进行可视化展示。分别展示总体情况分布，如图4，交叉测试集预测情况分布，如图5，可以发现分布大体一致，但由于 REPEAT=1 的样本数较少，模型无法充分学习该类样本的特征，从而预测结果 REPEAT=1 的也会偏少。

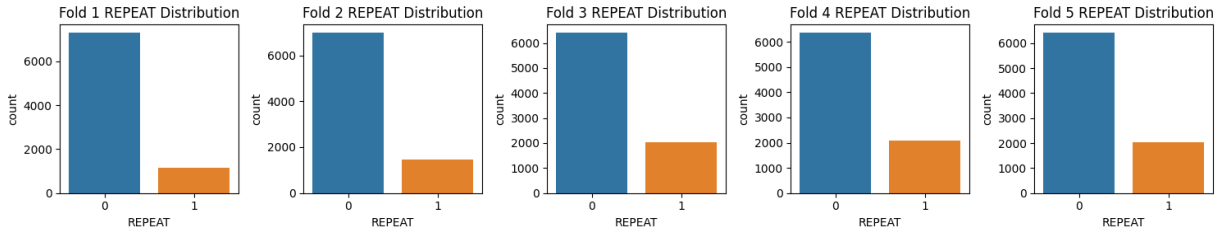


((a)) 真实结果分布

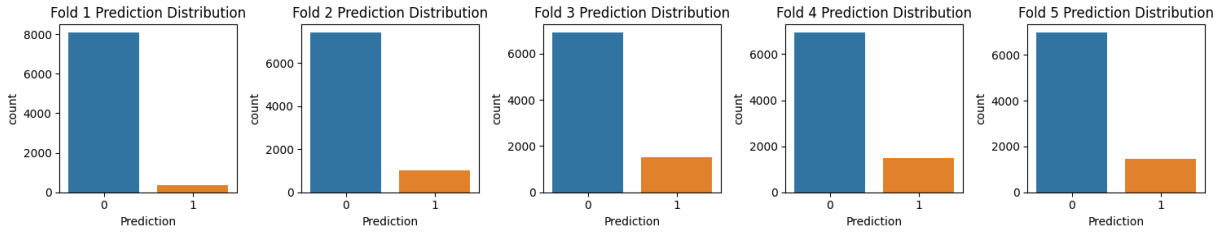


((b)) 预测结果分布

图 4: 随机森林真实数据与预测结果



((a)) 真实结果分布



((b)) 预测结果分布

图 5: 五折交叉预测结果分布

7 过程记录

7.1 离散变量取值范围过大影响分类效果

OCOD2 与 COBN_F 为离散变量，并且取值范围很大，这些特征对 *SVM* 模型在计算距离和间隔时产生较大的影响，导致 *SVM* 对这些特征更为敏感，因而预测效果不佳。

Decision Tree accuracy: 0.7560859755373471

Random Forest accuracy: 0.8274551715948225

SVM accuracy: 0.791117444484028

Decision Tree confusion matrix: $\begin{bmatrix} 5582 & 1080 \\ 974 & 785 \end{bmatrix}$

Random Forest confusion matrix: $\begin{bmatrix} 6288 & 374 \\ 1079 & 680 \end{bmatrix}$

SVM confusion matrix: $\begin{bmatrix} 6662 & 0 \\ 1759 & 0 \end{bmatrix}$

Decision Tree classification report:

precision	**recall**	**f1-score**	**support**
0.85	0.84	0.84	6662
0.42	0.45	0.43	1759
accuracy: 0.76, **macro avg:** 0.64, **weighted avg:** 0.76			

Random Forest classification report:

precision	**recall**	**f1-score**	**support**
0.85	0.94	0.90	6662
0.65	0.39	0.48	1759
accuracy: 0.83, **macro avg:** 0.75, **weighted avg:** 0.81			

SVM classification report:

precision	**recall**	**f1-score**	**support**
0.79	1.00	0.88	6662
0.00	0.00	0.00	1759
accuracy: 0.79, **macro avg:** 0.40, **weighted avg:** 0.63			

如果使用独热编码会导致特征由 12 列增加至 594 列，导致算法运行时间过长并且容易过拟合，所以使用 *LabelEncoder* 对离散特征进行标签编码，再使用 *StandardScaler* 对标签编码后的特征进行标准化。

7.2 正负样本不平衡

当正负样本不平衡时，模型在预测 REPEAT=1 的数据时，精确率相对较低。这是因为模型在训练过程中倾向于更多地学习和预测数量较多的类别，即 REPEAT=0。而对于 REPEAT=1，模型没有足够的样本和信息进行准确的学习，模型可能倾向于将更多的样本预测为数量较多的类别，从而导致相对较低的精确率。在这种情况下，模型可能会将一些实际属于 REPEAT=1 的样本错误地预测为 REPEAT=0，从而降低了预测的准确性。

Decision Tree accuracy: 0.7609547559672248

Random Forest accuracy: 0.8278114238213988

SVM accuracy: 0.820211376321102

Decision Tree confusion matrix: $\begin{bmatrix} 5615 & 1047 \\ 966 & 793 \end{bmatrix}$

Random Forest confusion matrix: $\begin{bmatrix} 6284 & 378 \\ 1072 & 687 \end{bmatrix}$

SVM confusion matrix: $\begin{bmatrix} 6478 & 184 \\ 1330 & 429 \end{bmatrix}$

Decision Tree classification report:

precision	recall	f1-score	support
0.85	0.84	0.85	6662
0.43	0.45	0.44	1759
accuracy: 0.76, macro avg: 0.64, weighted avg: 0.77			

Random Forest classification report:

precision	recall	f1-score	support
0.85	0.94	0.90	6662
0.65	0.39	0.49	1759
accuracy: 0.83, macro avg: 0.75, weighted avg: 0.81			

SVM classification report:

precision	recall	f1-score	support
0.83	0.97	0.90	6662
0.70	0.24	0.36	1759
accuracy: 0.82, macro avg: 0.76, weighted avg: 0.80			

7.3 使用随机下采样平衡正负样本

使用随机下采样 [7] 让 REPEAT=1 与 REPEAT=0 的样本数一致，训练集由原本的 33000 余条减少至 11000 余条，但训练出的模型性能不佳，虽然预测 REPEAT=1 的精确率有所提升，但是总体的准确率均降低了。

Decision Tree accuracy: 0.6484149855907781

Random Forest accuracy: 0.7273054755043228

SVM accuracy: 0.7280259365994236

Decision Tree confusion matrix: $\begin{bmatrix} 891 & 494 \\ 482 & 909 \end{bmatrix}$

Random Forest confusion matrix: $\begin{bmatrix} 1013 & 372 \\ 385 & 1006 \end{bmatrix}$

SVM confusion matrix: $\begin{bmatrix} 953 & 432 \\ 323 & 1068 \end{bmatrix}$

Decision Tree classification report:

precision	recall	f1-score	support
0.65	0.64	0.65	1385
0.65	0.65	0.65	1391
accuracy: 0.65, macro avg:0.65, weighted avg: 0.65			

Random Forest classification report:

precision	recall	f1-score	support
0.72	0.73	0.73	1385
0.73	0.72	0.73	1391
accuracy: 0.73, macro avg: 0.73, weighted avg: 0.73			

SVM classification report:

precision	recall	f1-score	support
0.75	0.69	0.72	1385
0.71	0.77	0.74	1391
accuracy: 0.73, macro avg: 0.73, weighted avg: 0.73			

7.4 使用 *SMOTE* 过采样方法平衡正负样本

SMOTE(Synthetic Minority Over-sampling Technique)[8]：*SMOTE* 是一种基于合成样本的过采样方法，它通过插值生成新的少数类样本。具体而言，*SMOTE* 选取一个少数类样本和其近邻样本，然后在这两个样本之间随机生成新的样本，从而扩大少数类样本的数量。

Decision Tree accuracy: 0.7609547559672248

Random Forest accuracy: 0.8278114238213988

SVM accuracy: 0.820211376321102

Decision Tree confusion matrix: $\begin{bmatrix} 5615 & 1047 \\ 966 & 793 \end{bmatrix}$

Random Forest confusion matrix: $\begin{bmatrix} 6284 & 378 \\ 1072 & 687 \end{bmatrix}$

SVM confusion matrix: $\begin{bmatrix} 6478 & 184 \\ 1330 & 429 \end{bmatrix}$

Decision Tree classification report:

precision	recall	f1-score	support
0.86	0.80	0.83	6662
0.39	0.49	0.44	1759
accuracy: 0.73, macro avg: 0.62, weighted avg: 0.76			

Random Forest classification report:

precision	recall	f1-score	support
0.88	0.87	0.87	6662
0.52	0.55	0.54	1759
accuracy: 0.80, macro avg: 0.70, weighted avg: 0.81			

SVM classification report:

precision	recall	f1-score	support
0.93	0.71	0.81	6662
0.42	0.78	0.55	1759
accuracy: 0.73, macro avg: 0.67, weighted avg: 0.82			

参考文献

[1] 安心小鱼, "数据分析-不平衡数据处理," Blog, CSDN, 1 2021, [Online]. Available: https://blog.csdn.net/weixin_36147808/article/details/112751796.

[2] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, no. 1, pp. 81–106, 1986.

[3] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[4] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[5] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees," Chapman & Hall, 1984.

[7] B. Shahbaba and R. M. Neal, "Nonlinear models using dirichlet process mixtures," Journal of Machine Learning Research, vol. 10, no. Aug, pp. 1829–1850, 2009.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," Journal of artificial intelligence research, vol. 16, pp. 321–357, 2002.