NIS3317 数据挖掘课程报告
Dry Beans 分类

郑宇森 **王鑫** **江彦泽**

{zys0794,wangxin.1,genius_j}@sjtu.edu.cn

1 引言

在本次研究中，我们选择了 Dry Bean Dataset[1]进行分类测试，并使用了 11 种不同的分类器模型，包括梯度提升分类器、XGB 分类器、基于树结构的分类器模型、集合元估计器、多层感知器模型、K 近邻分类器、支持向量机、决策树分类器、随机森林分类器、随机辐射下降分类器和朴素贝叶斯模型。我们对处理后的数据集进行了分类任务的训练和预测，并对分类的精度、召回率和 F1 score 进行了评估比较。**所有使用的分类器模型在评估中均获得了 90% 以上的 F1 score，这表明它们在该数据集上表现良好且可靠**。

2 数据分析

Dry Bean Dataset 数据集中包含 7 个类别，每个类别的数量如表 1所示。每个实例都包括 17 个特征（其中一个为类别特征），如表 2所示。

Table 1: Dry Bean Dataset 类别信息

DERMASON	SIRA	SEKER	HOROZ	CALI	BARBUNYA	BOMBAY
3546	2636	2027	1860	1630	1322	522

Table 2: Dry Bean Dataset 特征信息

特征	符号	定义
Area	A	bean 区域的面积及其边界内的像素数
Perimeter	P	bean 的周长，定义为其边界的长度
Major axis length	L	在 bean 上可以画出的最长的线的两端之间的距离
Minor axis length	l	在垂直于主轴的情况下，从豆子上可以画出的最长的线
Aspect ratio	K	Major axis length 和 Minor axis length 的关系
Eccentricity	Ec	偏心率
Convex area	C	能包含 bean 面积的最小凸形多边形中的像素数
Equivalent diameter	Ed	面积与 bean 面积相同的圆的直径
Extent	Ex	紧凑度，即界定框中的像素与 bean 面积的比率
Solidity	S	凸壳中的像素与 bean 中的像素的比率
Roundness	R	计算方式为 $4\pi A/P^2$
Compactness	CO	计算方式为 Ed/L
ShapeFactor1	$SF1$	形状因子
ShapeFactor2	$SF2$	形状因子
ShapeFactor3	$SF3$	形状因子
ShapeFactor4	$SF4$	形状因子
Class		bean 的类别

[1] https://archive.ics.uci.edu/ml/datasets/dry+bean+dataset

我们绘制了各个特征分布的直方图，如图 1所示，这能够帮助我们更加直观地理解特征分布的信息。我们还分析了不同类别下特征分布的情况，如图 2, 3所。我们发现 Area，Perimeter，Major axis length 等特征在不同类别的 bean 上的分布具有较大差异，而 Extent，Solidity，Roundness 等特征在不同类别的 bean 中的分布差异很小。这启发我们可以从中选择具有代表性和区分性的特征对 bean 进行分类。

3 数据预处理

3.1 数据清洗

数据清洗部分包括以下操作。

去除离群值. 首先对数据集去除空值和重复值。然后采用 z-score 方法（标准分数）将数据集中的特征数据标准化，z-score 表示一个数据点与平均值之差对应的标准差的数量。即：

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

我们认为 z-score 的绝对值大于某个值作为异常判定值，最终只保留正常数据。在具体的代码实现中，我们编写了 `removeOutliers_cat` 函数为分类中实际对原始数据进行异常值去除的函数。该函数按 drybeans 数据集中种子的品类为处理分界，在每一品类内单独处理异常值，将脱离各自品类平均值的标准差数量大于 4 的数据丢弃。在实验过程中，我们去除了 275 个异常值。

数据缩放. 同样使用公式 (1)对数据进行归一化和标准化。在程序中，我们定义了 `normalize_data` 函数对数据集进行数据缩放，在实现上选用的是 `StandardScaler` 类中的 `fit_transform` 方法。

缺失值处理. 我们使用了 `fillna` 函数对数据集中部分样本缺失的特征值进行了处理，方法是采用各自品类的该特征的平均值进行填充。

3.2 特征提取

特征分析. 为了选取具有高区分度的特征，我们对数据集中的 16 个特征进行了关联性分析。首先，我们绘制了 Area、Perimeter、AspectRatio、Eccentricity、Roundness 和 Compactness 这六个主要特征的散点图，如图 4所示。从分布散点图中可以清晰地看出，不同类别的 bean 样本会聚簇在不同的位置，表现出不同的特征分布。这也说明使用这些特征对 bean 的类别进行分类是可行的。我们还绘制了不同特征之间的相关性热图，如图 5所示，从中我们可以直观地了解到不同特征之间的相关性强弱，这有助于我们选取合适的特征进行分类。我们也计算了样本的各个特征与样本类别之间的互信息分数，如图 6所示。互信息度量了特征与类别之间相互依赖的程度，具体来说，互信息是 bean 的种类的不确定度由于已知 bean 的特定特征信息而减少的信息量。因此我们可以选取互信息高的特征进行分类。

去除冗余特征. 在图 3中可以明显看出，Area 与 ConvexArea，以及 Compactness 和 ShapeFactor3 两组特征具有较高的相似性。因此我们各删除了一个冗余特征，分别是 ConvexArea 和 Compactness，并得到的新的特征集。

构建衍生特征. 在表 2所列的特征之外，我们还计算了若干衍生特征。它们在不同的 bean 之间具有显著的差异性，可以提升分类精度。这些特征如表 3所示

Table 3: 衍生特征

特征	定义	说明
ShapeFactor5	L/P	长轴长度除以周长
ShapeFactor6	l/P	短轴长度除以周长
ShapeFactor7	$Ec \times A$	离心率乘以面积
ShapeFactor8	$Ec \times P$	离心率乘以周长
ShapeFactor9	$Ex \times A$	紧凑度乘以面积
ShapeFactor10	$Ex \times P$	紧凑度乘以周长

进行特征提取后，我们绘制了不同品类不同特征的分布散点图，如图 7所示。由图可见，处理后的数据集，在每种属性上各品类上可以明显的区分开，说明数据预处理后的数据集具备良好的分类特性，可以帮助分类模型更好地训练，从而达到较优的效果。

3.3 数据集构建

类别编码. 我们编写了 `encodingTarget` 函数和 `decodingTarget` 函数，使用预定义字典对 Class 特征进行值替换。替换规则为：按照表 1所列的类别顺序，分别将这些特征替换为 0–6 的数值表示。这样可以方便后续的分类训练过程。

数据集划分. 我们将数据集按照 0.72：0.13：0.15 的比例划分为训练集（train）、验证集（validation）、测试集（test）。具体操作是使用 `sklearn` 库中的 `train_test_split` 方法，先将原数据集划分为 0.85：0.15 的训练集（train）和测试集（test）。然后再使用 `train_test_split` 方法将此时的“训练集”进一步按照 0.85：0.15 的比例划分为训练集（train）和验证集（validation）。然后使用 `drop` 方法将标签从数据集中去除，将训练集、验证集、测试集分别切分为数据和标签两个部分。最终，我们训练集、验证集和测试集中的样本实例数分别为：9784, 1727, 2032。大量的样本有助于构建准确、鲁棒和泛化能力强的模型。

4 模型架构

我们在数据集上训练及验证了不同的分类模型，然后进行测试。

我们选取了梯度提升分类器（*GradientBoostingClassifier*）、*XGB* 分类器（*XGB Classifier*）、基于树结构的分类器模型（*LightGB Classifier*）、集合元估计器（*BaggingClassifier*）、多层感知器模型（*MultiLayerPerceptronClassifier*）、*K* 近邻分类器（*KNeighborsClassifier*）、支持向量机（*SupportVectorClassifier*）、决策树分类器（*DecisionTreeClassifier*）、随机森林分类器（*RandomForestClassifier*）、随机辐射下降分类器（*StochasticGradiantDescentClassifier*）、朴素贝叶斯模型（*Naive Bayes Model*）这十一种分类器模型对预处理后的数据集进行了分类任务的训练和预测，选取了精度、召回率、*F1_score* 等指标进行了评估。

Gradient Boosting Classifier. Gradient Boosting Classifier 是一种集成学习算法，它通过组合多个弱分类器（通常是决策树）来构建一个强分类器。它的算法原理如下：

1. 初始化模型：开始时，将强分类器初始化为一个简单的模型，通常选择一个常数作为初始预测值。然后，计算初始预测值与实际标签之间的残差（预测值减去实际值）。
2. 迭代训练：在每一轮迭代中，训练一个新的弱分类器来预测之前模型的残差。这个弱分类器被称为“基学习器”。
3. 残差拟合：使用基学习器对残差进行拟合。基学习器的任务是尽可能减小残差。可以使用各种机器学习算法作为基学习器，但通常使用决策树。
4. 更新模型：将新的基学习器添加到模型中，并更新模型的预测结果。新的预测结果是之前预测结果与新基学习器的预测结果的加权和。加权系数通常通过优化算法（如梯度下降）来确定。
5. 更新残差：计算更新后的预测结果与实际标签之间的残差。这些残差成为下一轮迭代的训练目标。
6. 迭代训练：重复步骤 3 到步骤 5，直到满足停止准则（如达到最大迭代次数或残差的变化不大）。
7. 得到最终模型：最终模型是所有基学习器的线性组合。它可以根据每个基学习器的权重对每个基学习器的预测结果进行加权平均。

Gradient Boosting Classifier 的关键思想是通过迭代训练一系列弱分类器来逐步改善模型的性能。每个新的弱分类器都会关注之前模型没有正确预测的样本，以此来减小残差。通过不断迭代，模型可以学习到更复杂的关系，从而提高整体的分类准确率。

XGBoost. XGBoost（eXtreme Gradient Boosting）是一种梯度提升树算法，特别用于解决回归和分类问题。XGBoost 的分类器算法原理如下：

1. 初始化模型：开始时，将强分类器初始化为一个常数值。对于分类问题，可以使用类别中样本的比例作为初始预测概率。
2. 迭代训练：在每一轮迭代中，训练一个新的弱分类器（决策树）来拟合之前模型的残差。

3. 残差拟合：使用基学习器（决策树）对残差进行拟合。基学习器的任务是尽可能减小残差。XGBoost 使用一种特殊的决策树模型，称为提升树（Boosting Tree），它是通过不断添加新的决策树来逐步改善模型的预测性能。
4. 更新模型：将新的基学习器添加到模型中，并更新模型的预测结果。新的预测结果是之前预测结果与新基学习器的预测结果的加权和。加权系数是通过优化算法（梯度下降）来确定的，它会最小化损失函数的值。
5. 正则化：为了防止过拟合，XGBoost 在每一轮迭代中引入了正则化项。正则化项包括树的复杂度和预测值的稀疏性，它们被添加到损失函数中，并在优化过程中进行惩罚。
6. 迭代训练：重复步骤 3 到步骤 5，直到满足停止准则（如达到最大迭代次数、残差的变化不大或达到早停策略定义的条件）。
7. 得到最终模型：最终模型是所有基学习器的线性组合。它可以根据每个基学习器的权重对每个基学习器的预测结果进行加权平均。

XGBoost 的关键思想是通过迭代训练一系列的提升树来逐步改善模型的性能。每个新的提升树都会关注之前模型没有正确预测的样本，以此来减小残差。通过不断迭代、加权组合和正则化，XGBoost 可以学习到更复杂的关系，从而提高整体的分类准确率。此外，XGBoost 还具有计算效率高、可扩展性强和对缺失值的处理能力等优点，使其成为广泛使用的机器学习模型。

LightGBM. LightGBM（Light Gradient Boosting Machine）是一种梯度提升树算法，它是一个基于决策树的集成学习框架。LightGBM 的分类器算法原理如下：

1. 数据准备：将训练数据划分为多个数据块（batches），每个数据块包含一部分样本。LightGBM 使用基于特征直方图的方法来高效地处理大规模数据。
2. 构建初始决策树：开始时，构建一个简单的决策树作为初始模型。可以将样本的初始预测值设置为常数或使用类别中样本的比例作为初始预测概率。
3. 迭代训练：在每一轮迭代中，LightGBM 通过增加新的叶子节点来扩展现有的决策树。每个叶子节点都代表一个子集的样本，通过优化算法来最小化损失函数。
4. 分裂节点选择：在每一轮迭代中，LightGBM 使用一种称为基于梯度的单边采样（Gradient-based One-Side Sampling，GOSS）的技术来选择需要分裂的节点。GOSS 通过保留梯度较大的样本，丢弃梯度较小的样本，以减少计算开销和过拟合风险。
5. 直方图优化：LightGBM 使用直方图算法来优化特征的离散化和存储。通过将特征值划分为不同的桶（bins），可以大幅减少计算开销。
6. 损失函数优化：LightGBM 使用梯度下降算法来最小化损失函数。在每个节点上，使用近似的方法计算梯度和 Hessian 矩阵，以加速训练过程。
7. 正则化：为了防止过拟合，LightGBM 引入了正则化项，包括叶子节点的复杂度和特征的稀疏性，它们被添加到损失函数中，并在优化过程中进行惩罚。
8. 提前停止策略：LightGBM 支持提前停止策略，可以根据验证集的表现来确定何时停止迭代，以防止过拟合。

通过以上步骤，LightGBM 迭代地构建多棵决策树，并将它们组合成最终的分类器。LightGBM 的算法原理主要集中在高效的数据处理、节点分裂选择、直方图优化和损失函数优化等方面，以实现快速训练和准确的分类效果。

Bagging Classifier. Bagging Classifier（自助聚合分类器）是一种集成学习方法，通过对训练数据进行有放回的自助采样，构建多个基分类器并进行投票或平均来进行预测。其算法原理如下：

1. 自助采样：从原始训练数据集中有放回地随机抽取样本，构建多个不同的训练子集。这意味着某些样本可能在一个子集中出现多次，而其他样本可能在某些子集中根本不出现。
2. 基分类器训练：对于每个训练子集，独立地训练一个基分类器。可以选择任何基分类器算法，如决策树、支持向量机、神经网络等。每个基分类器都根据对应的训练子集进行训练。
3. 集成预测：对于分类问题，通过投票的方式获得最终预测结果。每个基分类器对新样本进行预测，然后根据投票结果或平均值来确定最终预测类别。

4. 回归问题的处理：对于回归问题，通过对基分类器的预测结果进行平均来得到最终预测值。

BaggingClassifier 的关键思想是通过构建多个基分类器，利用每个基分类器对样本的不同采样和训练，来减小模型的方差和泛化误差，从而提高整体的预测性能。由于基分类器之间的独立性，BaggingClassifier 具有较好的抗噪声能力和泛化能力。

BaggingClassifier 适用于样本量较小、容易过拟合的问题，对于大规模数据集和高维特征空间，通常使用随机森林（Random Forest）算法，它是一种基于 Bagging 的决策树集成方法。

MultiLayer Perceptron Classifier. MultiLayer Perceptron Classifier（多层感知器分类器）是一种基于人工神经网络的分类算法，它由多个神经网络层组成。其算法原理如下：

1. 神经网络结构：多层感知器由输入层、若干个隐藏层和输出层组成。输入层接收输入特征，输出层给出最终的分类结果，隐藏层则用于处理输入特征的非线性映射。
2. 前向传播：通过前向传播，将输入特征从输入层传递到输出层。在每个神经元中，对输入进行加权和求和，然后应用激活函数来引入非线性。这样，通过多层的计算和激活函数的叠加，神经网络可以学习复杂的非线性关系。
3. 权重更新：神经网络通过反向传播算法来更新网络中的权重。反向传播算法基于损失函数计算梯度，并沿着网络的反方向传播梯度，以更新每个神经元的权重。常用的优化算法如梯度下降法（Gradient Descent）和随机梯度下降法（Stochastic Gradient Descent）等。
4. 激活函数：在神经网络中，激活函数引入非线性映射，增加网络的表达能力。常见的激活函数包括 Sigmoid 函数、ReLU 函数、tanh 函数等，它们使神经网络能够逼近任意复杂函数。
5. 训练过程：通过提供训练样本和对应的标签，神经网络通过迭代训练来调整权重，使得网络能够学习输入特征与标签之间的映射关系。训练过程中使用的损失函数通常是交叉熵（Cross-Entropy）或均方误差（Mean Squared Error）等。
6. 预测过程：经过训练后的多层感知器可以用于预测新样本的标签。将新样本的特征输入网络，通过前向传播得到输出结果，并将输出结果转化为预测的类别。

K-Nearest Neighbors. K 近邻分类器（K-Nearest Neighbors Classifier）是一种基于实例的分类算法，其算法原理如下：

1. 训练阶段：将训练数据集中的样本特征和对应的标签存储起来。
2. 预测阶段：对于一个新的测试样本，计算它与训练集中每个样本的距离（通常使用欧氏距离或其他距离度量方法）。根据距离，选择与测试样本最近的 K 个训练样本作为邻居。
3. 多数投票：根据 K 个最近邻样本的标签，进行多数投票。即，统计 K 个最近邻样本中属于各个类别的个数，选择票数最多的类别作为测试样本的预测标签。
4. 输出预测结果：将多数投票得到的类别作为测试样本的预测标签。

K 近邻分类器的关键思想是认为距离较近的样本在特征空间中具有相似的特征，因此具有相似的标签。通过选择最近的 K 个邻居来决定预测标签，K 的选择是一个重要参数，它会影响模型的复杂性和预测结果。较小的 K 值会使模型更加敏感，容易受到噪声的影响，而较大的 K 值会使模型更加平滑，可能忽略了数据的局部特征。

Support Vector Classifier. Support Vector Classifier（支持向量分类器）是一种基于支持向量机（Support Vector Machine，SVM）的分类算法。其算法原理如下：

1. 数据表示：支持向量分类器假设训练数据是由特征向量和对应的标签组成的。这些特征向量将数据映射到一个高维特征空间。
2. 寻找最优超平面：支持向量分类器的目标是找到一个最优超平面，用于将不同类别的样本分隔开。最优超平面被定义为能够使两个类别之间的间隔最大化的超平面。
3. 支持向量：支持向量是离最优超平面最近的训练样本点。它们对于定义最优超平面起到关键作用，因为它们决定了超平面的位置和方向。
4. 核函数：支持向量分类器使用核函数来将数据映射到高维特征空间。核函数可以有效地处理非线性分类问题，将数据从原始特征空间映射到一个更高维度的特征空间，使得在该特征空间中存在一个线性可分的超平面。

5. 分类预测：对于新的测试样本，支持向量分类器根据其在特征空间中的位置，判断其属于哪个类别。通过计算测试样本与最优超平面之间的距离或者应用决策函数，可以得出样本的分类结果。

支持向量分类器具有良好的泛化能力和对噪声的鲁棒性。它能够处理高维数据和非线性分类问题，并且不容易受到训练样本的影响。支持向量分类器可以通过调整超参数（如正则化参数和核函数选择）来适应不同的数据集和问题。

Decision Tree Classifier. Decision Tree Classifier（决策树分类器）是一种基于决策树的分类算法，其算法原理如下：

1. 树结构表示：决策树通过树结构来表示分类规则。树的每个节点代表一个特征，根节点表示最重要的特征，叶节点表示分类的结果（类别标签）。
2. 特征选择：在每个节点上，选择最佳的特征来进行划分。通常使用一些指标（如信息增益、基尼指数）来评估特征的重要性，选择能够最好地分离不同类别的特征。
3. 样本划分：根据选择的特征，将样本集划分为不同的子集，每个子集对应一个子节点。划分的目标是使得每个子节点中的样本尽可能属于同一类别。
4. 递归构建：对每个子节点，重复步骤 2 和步骤 3，直到满足终止条件。终止条件可以是达到预定的树深度、达到最小样本数、或者无法继续有效地划分样本。
5. 叶节点分类：当达到终止条件时，叶节点表示最终的分类结果。通常采用多数投票或者根据样本比例来确定叶节点的分类标签。
6. 预测分类：对于一个新的测试样本，从根节点开始根据特征值进行判断，沿着树的分支逐步向下，最终到达叶节点并获得分类结果。

决策树分类器具有直观、解释性强的特点。它可以处理离散特征和连续特征，适用于多分类和二分类问题。然而，决策树容易产生过拟合的问题，特别是在处理复杂的数据集时。为了减少过拟合，可以通过剪枝技术（如预剪枝或后剪枝）来降低决策树的复杂度。

Random Forest Classifier. Random Forest Classifier（随机森林分类器）是一种基于随机森林的集成学习算法，其算法原理如下：

1. 随机抽样：从训练集中随机有放回地抽取样本，构建多个不同的训练子集。这些子集被称为自助样本（bootstrap samples）。
2. 随机特征选择：对于每个子集，从原始特征中随机选择一部分特征用于构建决策树。这个过程可以减少特征间的相关性，增加模型的多样性。
3. 决策树构建：对于每个子集，使用上述随机选择的特征构建一个决策树模型。决策树的构建过程中通常采用信息增益、基尼指数等准则来选择最佳特征和划分点。
4. 集成预测：当需要进行预测时，每个决策树对输入样本进行预测，并根据投票或平均等方式得到最终的预测结果。分类问题采用投票策略，回归问题采用平均策略。

随机森林通过构建多个随机选择的决策树，并集成它们的预测结果，来提高模型的泛化能力和减少过拟合的风险。由于随机选择特征和样本的过程，随机森林能够有效地处理高维数据和大规模数据集，并具有较好的鲁棒性。此外，随机森林还能够评估特征的重要性，并用于特征选择和可视化分析。

Stochastic Gradient Descent Classifier. Stochastic Gradient Descent Classifier（随机梯度下降分类器）是一种基于随机梯度下降算法的分类器。它是一种线性分类器，适用于大规模数据集和高维特征空间。

算法原理如下：

1. 初始化权重：随机初始化分类器的权重向量。
2. 迭代更新权重：对于每个训练样本，计算其预测值，并根据预测结果和实际标签之间的差异来调整权重。这个过程使用随机梯度下降算法，即每次只使用一个样本来更新权重。
3. 学习率调整：随机梯度下降算法中，通常会设置一个学习率参数来控制每次权重更新的步长。学习率可以固定，也可以根据迭代次数逐渐减小。
4. 终止条件：重复迭代更新权重的过程，直到达到预定的迭代次数或达到收敛条件（如权重变化很小）。

5. 预测分类：使用更新后的权重进行分类预测，根据输入样本的特征和权重向量的线性组合来计算预测结果。

StochasticGradientDescentClassifier 基于随机梯度下降算法，通过迭代更新权重来优化分类器。它适用于处理大规模数据集和高维特征空间，并且可以在线更新模型，适用于增量学习。

但是 StochasticGradientDescentClassifier 是一种线性分类器，对于非线性问题可能表现不佳。在应用中，可以通过设置不同的损失函数和正则化项来改进模型性能，如使用不同的惩罚项、调整学习率等。

Naive Bayes Model. 朴素贝叶斯模型（Naive Bayes Model）是一种基于贝叶斯定理的概率分类模型。它基于特征之间的条件独立性假设（朴素性），通过计算给定特征条件下目标类别的后验概率来进行分类。

算法原理如下：

1. 准备数据集：收集带有特征和对应类别标签的训练数据。
2. 计算类别的先验概率：统计训练集中每个类别的样本数量，并计算每个类别的先验概率。
3. 计算特征的条件概率：对于每个特征，根据训练集中的样本计算在给定类别下该特征的条件概率。通常使用频率计数或概率密度函数进行估计。
4. 计算后验概率：对于给定的测试样本，通过将先验概率和条件概率相乘，计算出每个类别的后验概率。
5. 进行分类：选择具有最高后验概率的类别作为测试样本的预测类别。

朴素贝叶斯模型基于贝叶斯定理，通过计算后验概率来进行分类预测。它的核心思想是假设特征之间条件独立，这是朴素性的假设。尽管这个假设在现实中很少成立，但朴素贝叶斯模型仍然在许多实际问题中表现良好。

朴素贝叶斯模型具有以下优点：简单、高效、适用于高维数据、对于少量训练数据也能表现良好。但它也有一些限制，如对输入特征之间的依赖性敏感。

5 模型训练

在本节中，我们具体描述实验中对分类模型的代码实现。

Gradient Boosting Classifier. 创建 GradientBoostingClassifier 对象，定义参数如迭代次数（n_estimators）、学习率（learning_rate）、树的最大深度（max_depth）以及随机种子（random_state）。确定了参数网格（param_grid），执行 KFold 对象进行交叉验证，参数设定包括折数（n_splits）、随机种子（random_state）及数据打乱（shuffle）。使用 GridSearchCV 进行网格搜索后，利用 fit() 方法进行训练，并调用 predict() 方法进行预测。使用 f1_score 和 classification_report 函数进行模型性能评估。训练结果见表 4。

XGBoost. 创建 XGBClassifier 对象，设置参数如学习率（learning_rate）、随机种子（random_state）、目标函数（objective）、树的最大深度（max_depth）、正则化参数（reg_alpha）以及 gamma。利用 fit() 方法进行训练，并通过 KFold 进行交叉验证。利用 predict() 方法对验证集进行预测，再使用 f1_score 和 classification_report 进行性能评估。训练结果见表 5。

LightGBM. 创建 LGBMClassifier 对象，设置参数如目标函数（objective）、随机种子（random_state）、学习率（learning_rate）以及正则化参数（reg_alpha）。利用 fit() 方法进行训练，利用 predict() 对验证集进行预测，并采用 f1_score 和 classification_report 进行性能评估。训练结果见表 6。

Bagging Classifier. 创建 BaggingClassifier 对象，设置参数如随机种子（random_state）、基分类器的数量（n_estimators）以及是否使用袋外样本评估（oob_score）。利用 fit() 方法进行训练，利用 predict() 对验证集进行预测，并采用 f1_score 和 classification_report 进行性能评估。训练结果见表 7。

MultiLayer Perceptron Classifier. 创建 MLPClassifier 对象，参数设置如求解器（solver）、激活函数（activation）、正则化参数（alpha）、随机种子（random_state）、

最大迭代次数（max_iter）、早停策略（early_stopping）、验证集比例（validation_fraction）、权重重用（warm_start）、信息显示（verbose）、学习率调整策略（learning_rate）以及初始学习率（learning_rate_init）。利用 fit() 方法进行训练，利用 predict() 对验证集进行预测，并采用 f1_score 和 classification_report 进行性能评估。训练结果见表 8。

K-Nearest Neighbors. 创建 KNeighborsClassifier 对象,参数设置如算法选择(algorithm)、权重选择（weights）以及邻居数量（n_neighbors）。利用 fit() 方法进行训练，利用 predict() 对验证集进行预测，并采用 f1_score 和 classification_report 进行性能评估。训练结果见表 9。

Support Vector Classifier. 使用 sklearn 库中的 SVC 方法，我们实例化了一个支持向量分类器对象。核函数选择为径向基函数（RBF），正则化参数 C 设为 1.0，gamma 设为 0.20。训练集用于调用 fit() 方法训练模型，接着对验证集进行预测，最后生成了模型的分类报告及其 F1 分数。模型性能详见表 10。

Decision Tree Classifier. 我们创建了一个决策树分类器对象，设定其最大深度为 5，节点分裂所需的最小样本数为 16。通过训练集调用 fit() 方法训练模型，并对验证集进行预测。然后，生成模型的分类报告并计算 F1 分数。具体性能见表 11。

Random Forest Classifier. 创建了一个随机森林分类器对象，设定了 1000 棵决策树，最大深度为 7。该模型在训练集上进行训练，并在验证集上进行预测。生成的模型分类报告及其 F1 分数详见表 12。

Stochastic Gradient Descent Classifier. 我们创建了一个 SGDClassifier 对象，采用自适应学习率、初始学习率为 0.1，并设置了最大迭代次数为 1000。在训练集上调用 fit() 方法进行训练，之后在验证集上进行预测。模型性能见表 13。

Naive Bayes Model. 我们使用了 GaussianNB，对模型进行增量训练。然后在验证集上进行预测，生成模型的分类报告并计算 F1 分数。具体性能见表 14。

Voting Classifier. 我们还采用了 sklearn 库内置的 Voting Classifier 投票算法来集成以上 11 种分类模型, 使用投票的方式，根据各个分类器的预测结果进行投票决策，通过集成多个模型，可以充分利用每个模型的优点，抑制各自的缺点，提高模型的鲁棒性和泛化能力。集成模型和各个基分类器的 F1 分数如表 15所示。由结果可见，投票算法的 F1 score 为 0.93，有着很好的分类效果。

6 结果分析与讨论

我们的分类结果在附录 A中以表格的形式详细呈现。从结果中可以得到以下结论。

- 11 种分类器在测试集上的平均精度、召回率、F1 score 都超过了 90%，这说明这些分类器在测试集上表现非常优秀，能够高度准确地对数据进行分类。
- 对于不同种类的样本，分类的精度存在差异。例如，对于 DERMASON、SIRA、CALI、BARBUNYA 四种 bean 的类型，各个分类器的预测精度都在 95% 左右，而对于 HOROZ、BOMBAY 这两种 bean 类型，分类器的分类精度普遍较低，只有 90% 左右。我们认为这一现象是由于 Dry Beans 数据集中样本数量的不均衡造成的。从表 1中可见，DERMASON，SIRA 等类别的样本数量较多，而 BOMBAY 类别的样本数量较少，这会给分类器的训练和预测带来偏误。
- 对于 Dry Bean 的分类任务，多层感知机模型在测试集上的表现最优。它在各个类别上的分类精度、召回率、F1 Score 的均值接近 95%。这表明神经网络机器学习方法在分类预测等任务上具有较好的表现和应用前景。
- 投票算法能够有效集成所有基分类器的知识，根据不同分类器的表现得出更优的分类结果。

针对实验中出现的样本不均衡的问题，我们认为可以采用以下解决方法。

- 采集更多的数据：如果可能的话，我们可以采集更多的样本来增加数据集中样本数量较少的类别，这可以帮助分类器更好地学习数据集的特征，并提高分类的准确性。
- 过采样和欠采样：对于样本数量较少的类别，我们可以通过过采样或欠采样来增加或减少这些类别的样本数量。过采样方法包括复制样本、SMOTE 等，而欠采样方法包括删除样本、ClusterCentroids 等。

- 类别加权：通过调整分类器中不同类别的权重来解决样本不均衡的问题，例如，在分类器中引入类别权重参数，使得分类器更关注样本数量较少的类别。
- 结合不同分类器：结合多个分类器进行分类，使得每个分类器更专注于不同的类别，从而提高分类的准确性。
- 特征选择：选择更适合区分不同类别的特征，减少不必要的特征，可以有效提高分类器的准确性，同时降低样本不均衡的偏差。

综上，在本次作业中我们针对 Dry Bean 数据集进行了多个分类器的性能比较，并使用了许多数据预处理操作以减少模型在处理数据时的噪音和干扰，提高分类器的性能和准确性。结果中较高的分类精度证明了我们的实验操作和程序编写是正确有效的。我们的实验思路和方法也可以为实际应用中的分类问题提供参考和启示。

致谢

在本文的最后，我们要对回老师表达诚挚的感谢。回老师在数据挖掘课程中丰富详细的授课让我们收获颇丰。我们也要感谢助教老师在课程期间为我们解答问题和提供必要的支持，这对于我们学习这门课程有很大的帮助。最后，感谢您阅读到此，希望本文能对您有所启发。

A 分类结果

Table 4: Gradient Boosting Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.96	0.91	0.93	165	0.95	0.89	0.92	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.95	0.95	0.95	209	0.92	0.97	0.94	235
HOROZ	0.91	0.93	0.92	439	0.91	0.93	0.92	523
CALI	0.95	0.93	0.94	249	0.97	0.95	0.96	278
BARBUNYA	0.96	0.96	0.96	248	0.97	0.93	0.95	311
BOMBAY	0.86	0.88	0.87	345	0.86	0.88	0.87	398
accuracy			0.93	1727			0.93	2032
macro avg	0.94	0.94	0.94	1727	0.94	0.94	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

Table 5: XGBoost Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.96	0.92	0.94	165	0.94	0.89	0.91	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.94	0.95	0.95	209	0.93	0.96	0.95	235
HOROZ	0.91	0.94	0.93	439	0.92	0.94	0.93	523
CALI	0.96	0.92	0.94	249	0.97	0.96	0.96	278
BARBUNYA	0.95	0.96	0.96	248	0.96	0.94	0.95	311
BOMBAY	0.88	0.88	0.88	345	0.88	0.89	0.89	398
accuracy			0.93	1727			0.93	2032
macro avg	0.94	0.94	0.94	1727	0.94	0.94	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

B 数据可视化

Table 6: Light Gradient Boosting Machine Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.96	0.92	0.94	165	0.95	0.88	0.91	200
SIRA	1.00	1.00	1.00	72	0.99	1.00	0.99	87
SEKER	0.94	0.97	0.96	209	0.92	0.96	0.94	235
HOROZ	0.91	0.94	0.93	439	0.91	0.93	0.92	523
CALI	0.96	0.92	0.94	249	0.97	0.96	0.97	278
BARBUNYA	0.96	0.95	0.96	248	0.95	0.94	0.94	311
BOMBAY	0.89	0.89	0.89	345	0.88	0.87	0.88	398
accuracy			0.93	1727			0.93	2032
macro avg	0.95	0.94	0.94	1727	0.94	0.93	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

Table 7: Bagging Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.92	0.89	0.91	165	0.94	0.90	0.92	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.92	0.94	0.93	209	0.91	0.96	0.94	235
HOROZ	0.91	0.93	0.92	439	0.91	0.93	0.92	523
CALI	0.96	0.92	0.94	249	0.96	0.95	0.95	278
BARBUNYA	0.94	0.95	0.95	248	0.95	0.94	0.94	311
BOMBAY	0.88	0.87	0.88	345	0.87	0.87	0.87	398
accuracy			0.92	1727			0.92	2032
macro avg	0.93	0.93	0.93	1727	0.94	0.93	0.93	2032
weighted avg	0.92	0.92	0.92	1727	0.92	0.92	0.92	2032

Table 8: MultiLayer Perceptron Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.97	0.87	0.91	165	0.96	0.90	0.93	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.92	0.97	0.94	209	0.92	0.97	0.94	235
HOROZ	0.91	0.95	0.93	439	0.91	0.94	0.93	523
CALI	0.95	0.93	0.94	249	0.96	0.96	0.96	278
BARBUNYA	0.94	0.98	0.96	248	0.95	0.95	0.95	311
BOMBAY	0.91	0.85	0.88	345	0.90	0.86	0.88	398
accuracy			0.93	1727			0.93	2032
macro avg	0.94	0.94	0.94	1727	0.94	0.94	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

Table 9: K-Nearest Neighbors Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.94	0.90	0.92	165	0.95	0.88	0.91	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.93	0.96	0.94	209	0.90	0.97	0.93	235
HOROZ	0.92	0.92	0.92	439	0.93	0.92	0.93	523
CALI	0.97	0.93	0.95	249	0.97	0.95	0.96	278
BARBUNYA	0.95	0.94	0.95	248	0.95	0.95	0.95	311
BOMBAY	0.86	0.89	0.88	345	0.87	0.89	0.88	398
accuracy			0.93	1727			0.93	2032
macro avg	0.94	0.94	0.94	1727	0.94	0.94	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

Table 10: Support Vector Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.95	0.91	0.93	165	0.95	0.90	0.92	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.94	0.96	0.95	209	0.93	0.97	0.95	235
HOROZ	0.92	0.94	0.93	439	0.92	0.92	0.92	523
CALI	0.97	0.93	0.95	249	0.97	0.96	0.97	278
BARBUNYA	0.95	0.96	0.96	248	0.96	0.95	0.96	311
BOMBAY	0.88	0.88	0.88	345	0.87	0.89	0.88	398
accuracy			0.93	1727			0.93	2032
macro avg	0.94	0.94	0.94	1727	0.94	0.94	0.94	2032
weighted avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032

Table 11: Decision Tree Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.81	0.85	0.83	165	0.87	0.85	0.86	200
SIRA	1.00	1.00	1.00	72	0.99	1.00	0.99	87
SEKER	0.88	0.92	0.90	209	0.89	0.94	0.92	235
HOROZ	0.89	0.92	0.91	439	0.91	0.93	0.92	523
CALI	0.97	0.90	0.94	249	0.98	0.92	0.95	278
BARBUNYA	0.93	0.92	0.92	248	0.92	0.93	0.93	311
BOMBAY	0.86	0.83	0.84	345	0.88	0.85	0.87	398
accuracy			0.90	1727			0.91	2032
macro avg	0.91	0.91	0.91	1727	0.92	0.92	0.92	2032
weighted avg	0.90	0.90	0.90	1727	0.91	0.91	0.91	2032

Table 12: Random Forest Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.93	0.87	0.90	165	0.95	0.85	0.90	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.90	0.94	0.92	209	0.90	0.97	0.93	235
HOROZ	0.91	0.94	0.92	439	0.91	0.92	0.91	523
CALI	0.96	0.91	0.93	249	0.97	0.94	0.95	278
BARBUNYA	0.95	0.94	0.95	248	0.95	0.94	0.95	311
BOMBAY	0.87	0.88	0.87	345	0.85	0.88	0.86	398
accuracy			0.92	1727			0.92	2032
macro avg	0.93	0.92	0.93	1727	0.93	0.93	0.93	2032
weighted avg	0.92	0.92	0.92	1727	0.92	0.92	0.92	2032

Table 13: Stochastic Gradient Descent Classifier

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.93	0.90	0.91	165	0.94	0.90	0.92	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.93	0.94	0.93	209	0.93	0.95	0.94	235
HOROZ	0.92	0.91	0.91	439	0.93	0.87	0.90	523
CALI	0.96	0.92	0.94	249	0.96	0.94	0.95	278
BARBUNYA	0.95	0.95	0.95	248	0.96	0.93	0.95	311
BOMBAY	0.84	0.89	0.86	345	0.80	0.90	0.85	398
accuracy			0.92	1727			0.91	2032
macro avg	0.93	0.93	0.93	1727	0.93	0.93	0.93	2032
weighted avg	0.92	0.92	0.92	1727	0.92	0.91	0.91	2032

Table 14: Naive Bayes Model

Classes	Validation				Test			
	precision	recall	f1-score	support	precision	recall	f1-score	support
DERMASON	0.88	0.82	0.85	165	0.90	0.84	0.87	200
SIRA	1.00	1.00	1.00	72	1.00	1.00	1.00	87
SEKER	0.88	0.92	0.90	209	0.87	0.94	0.90	235
HOROZ	0.93	0.88	0.90	439	0.93	0.85	0.89	523
CALI	0.95	0.94	0.94	249	0.96	0.95	0.95	278
BARBUNYA	0.92	0.96	0.94	248	0.94	0.94	0.94	311
BOMBAY	0.83	0.88	0.85	345	0.80	0.89	0.85	398
accuracy			0.90	1727			0.90	2032
macro avg	0.91	0.91	0.91	1727	0.92	0.92	0.91	2032
weighted avg	0.90	0.90	0.90	1727	0.90	0.90	0.90	2032

Table 15: Voting Classifier F1 Score

Voting	SVC	MLP	LGB
0.9306102362204725	0.9320866141732284	0.9306102362204725	0.9261811023622047
GB	XGB	BAG	KN
0.9261811023622047	0.9315944881889764	0.922736220472441	0.9276574803149606
DT	RF	SGDC	NB
0.9104330708661418	0.9192913385826772	0.9138779527559056	0.9015748031496063

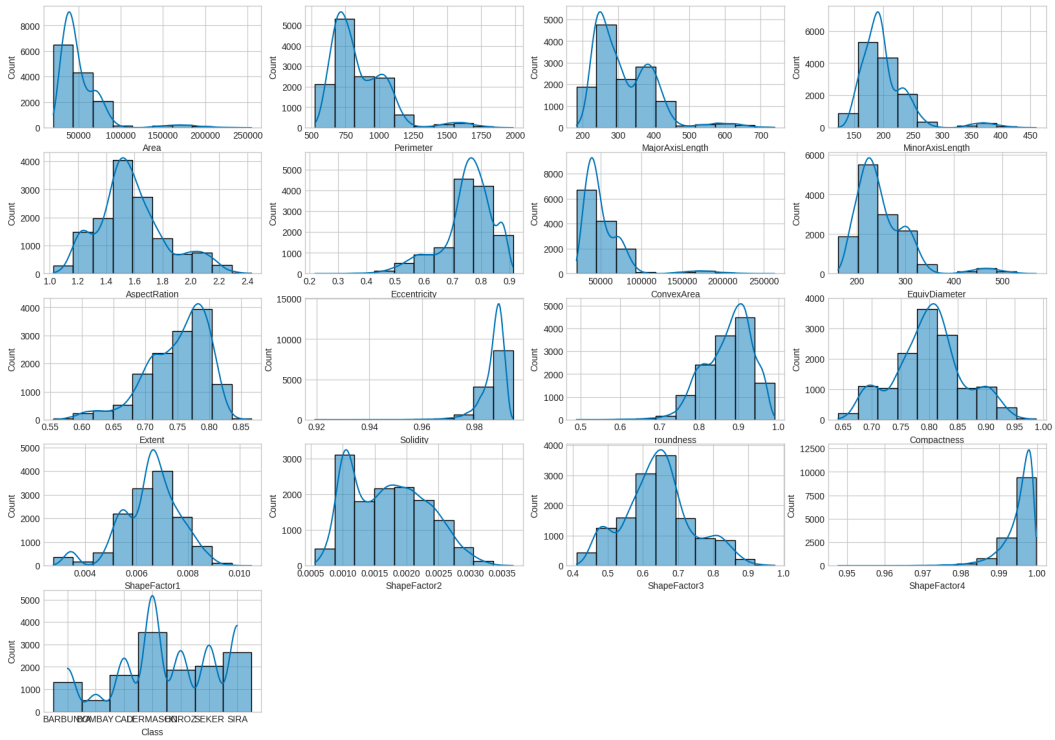

Figure 1: 各属性分布

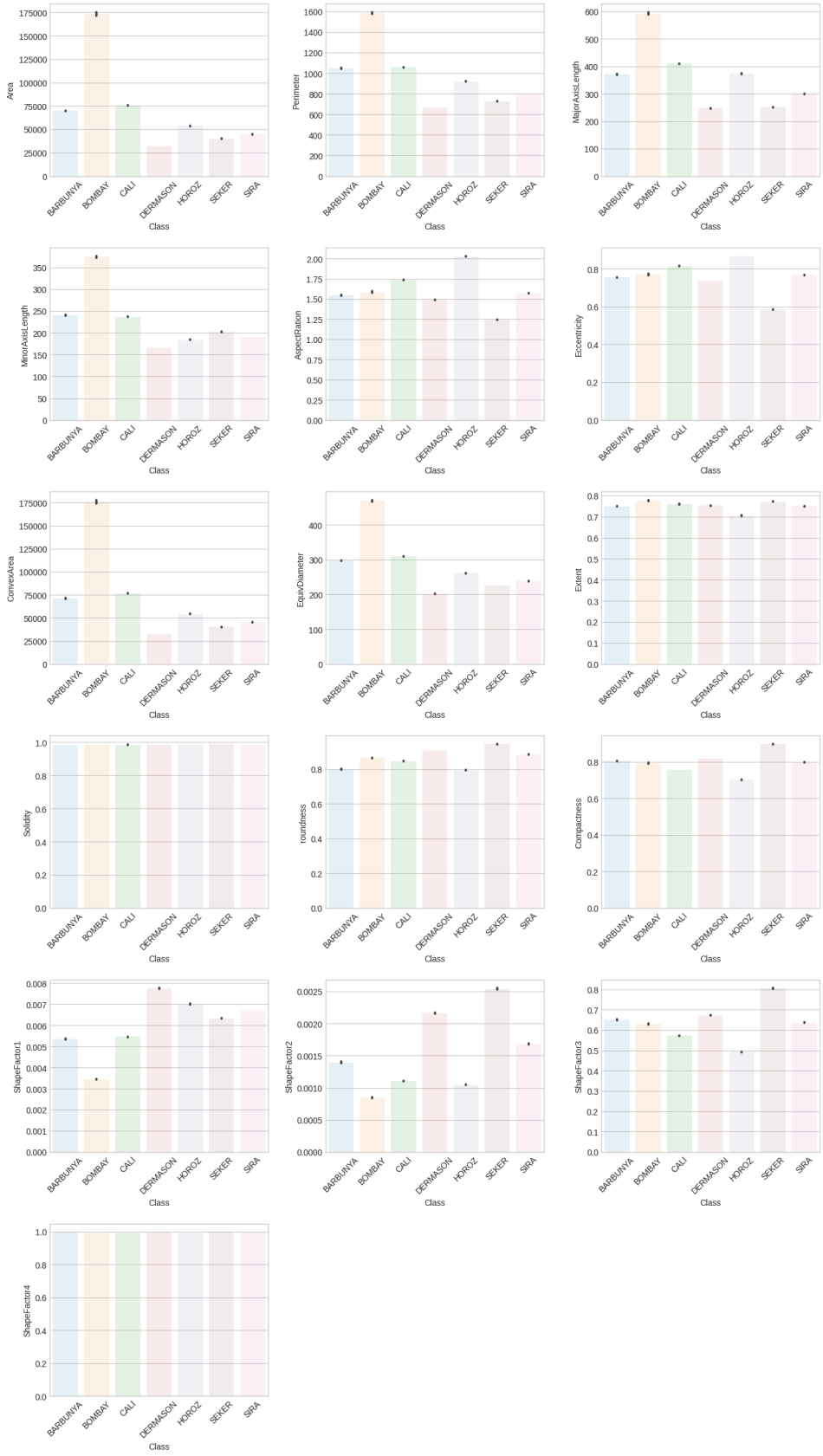

Figure 2: 不同类别下属性分布直方图

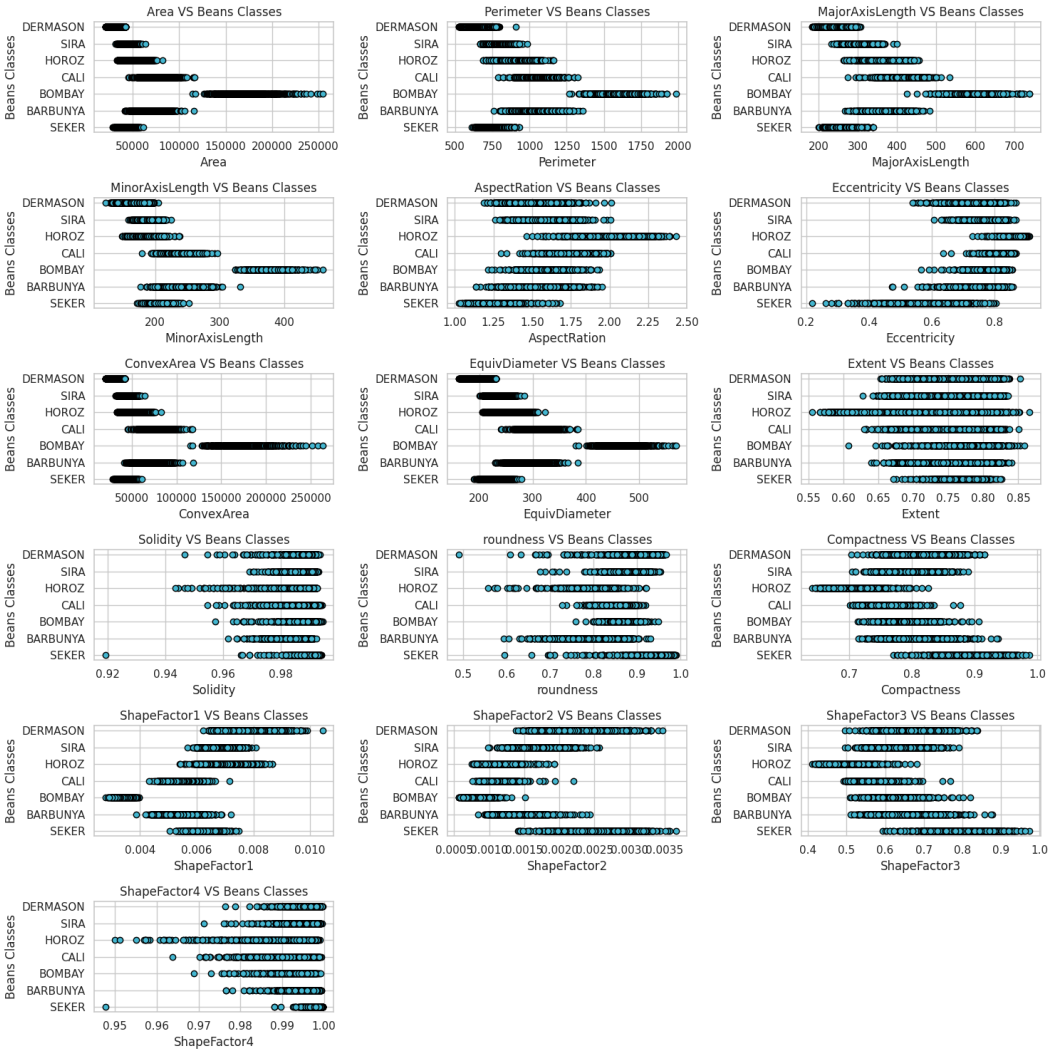

Figure 3: 不同类别下属性分布

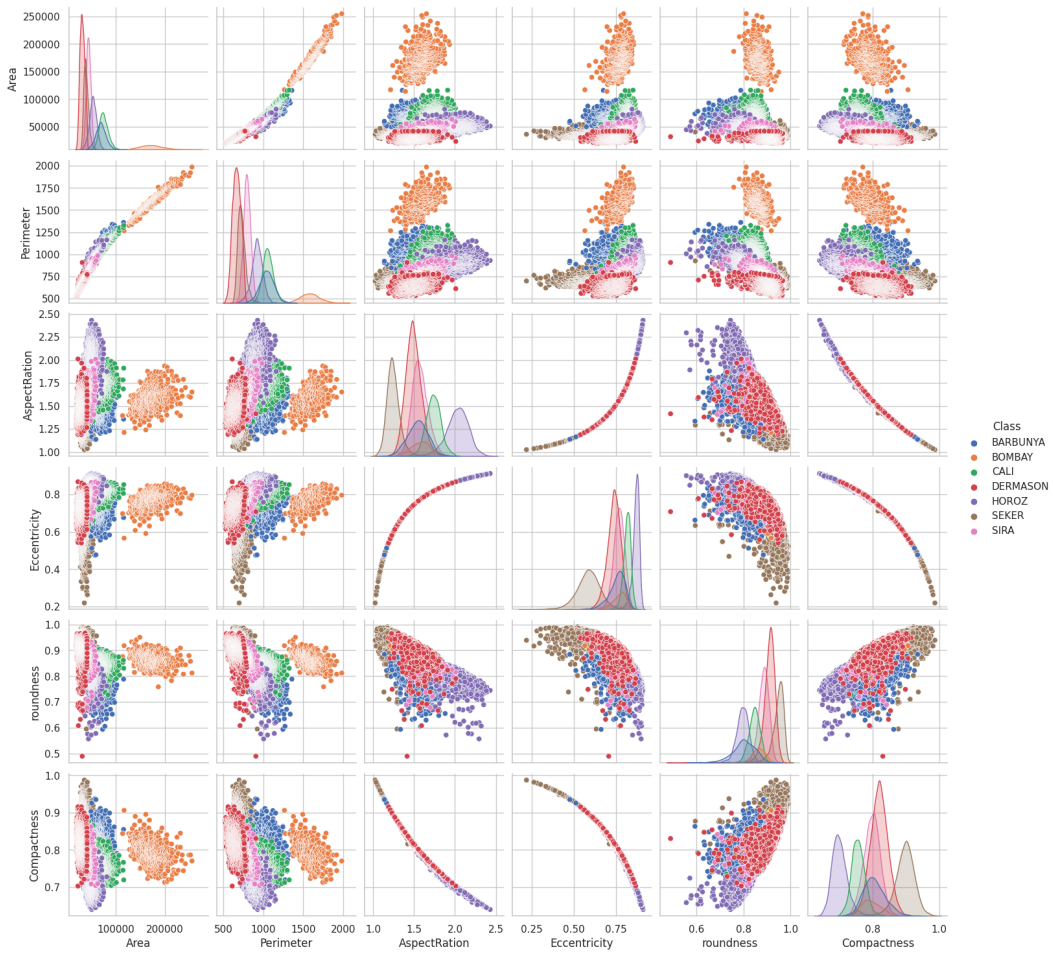

Figure 4: 强相关属性散点图

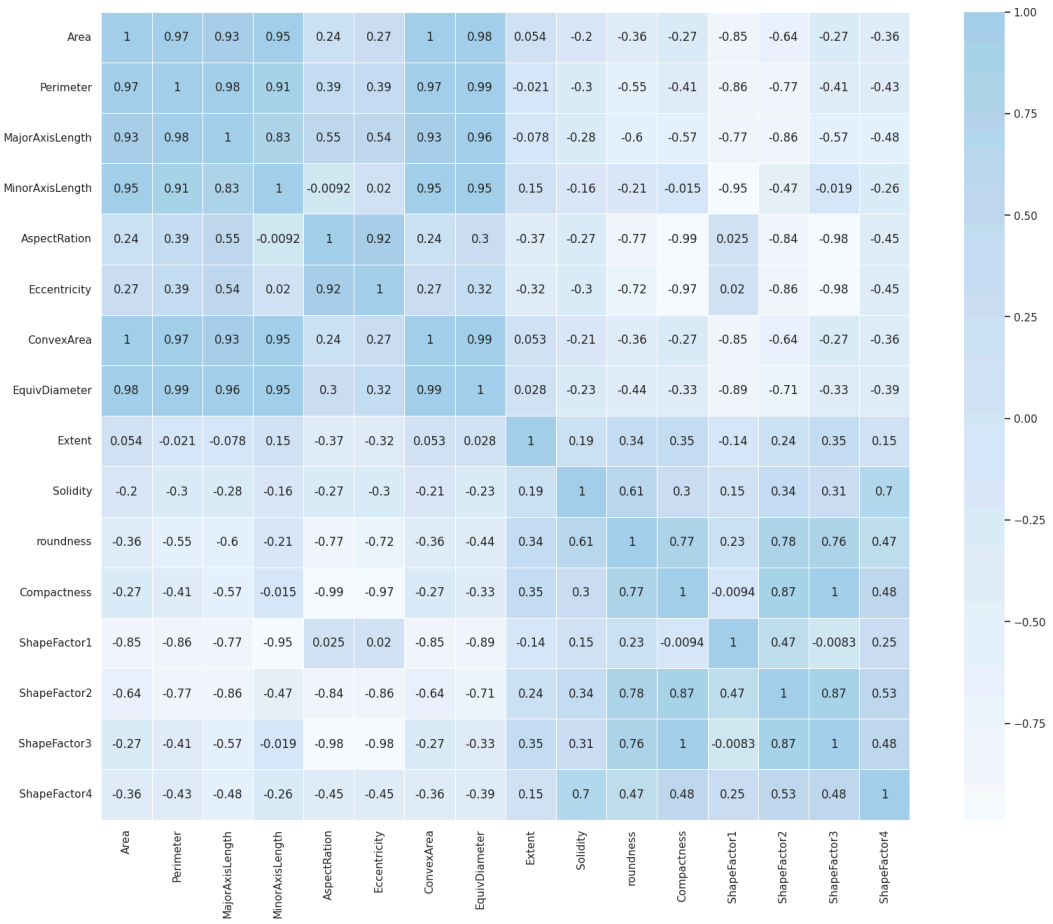

Figure 5: 各属性相关性热图

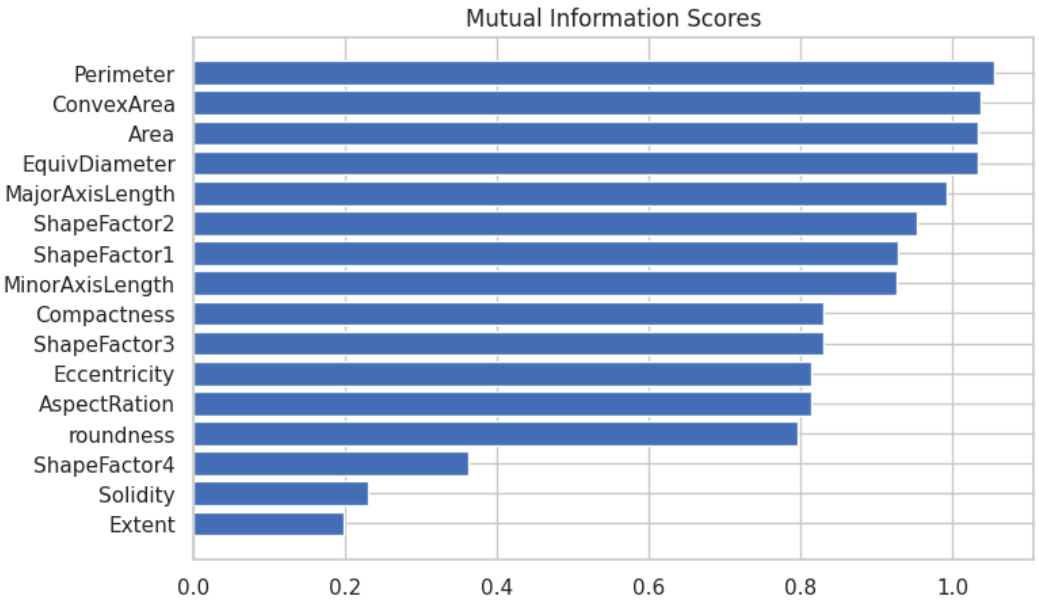

Figure 6: 特征与目标变量的互信息

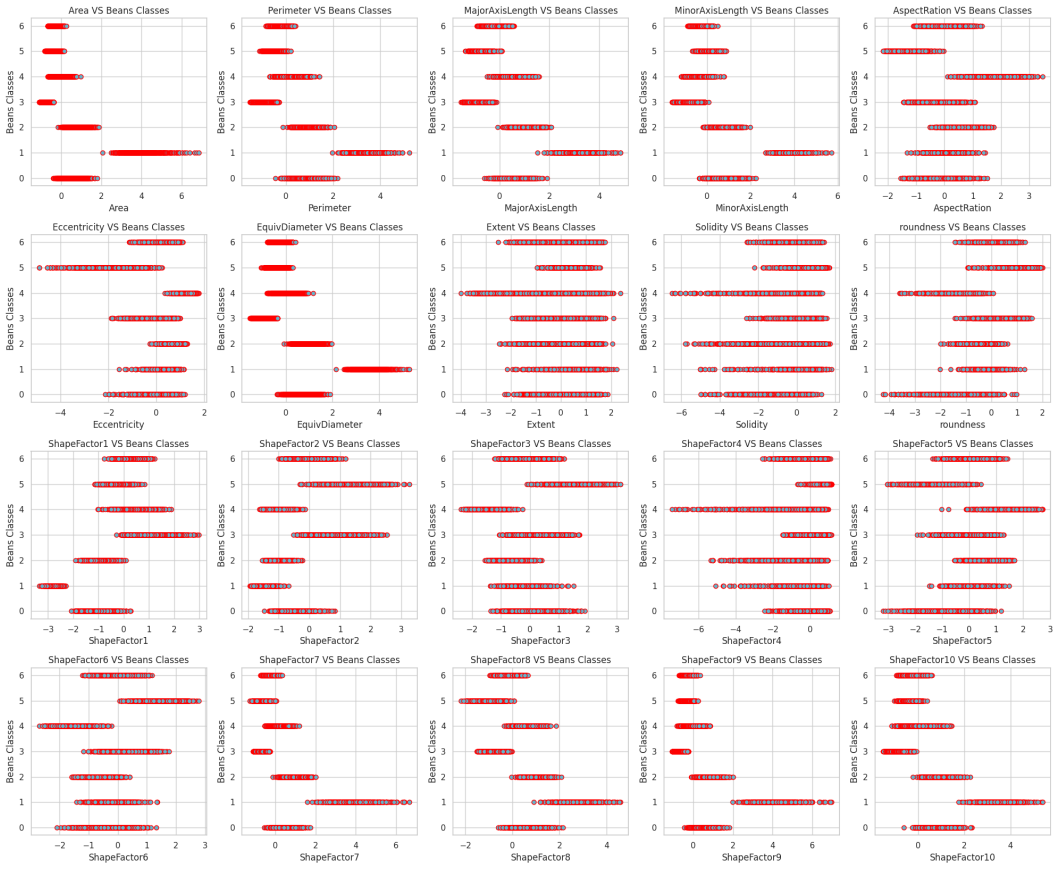

Figure 7: 预处理后不同类别下属性分布