**杭州电子科技大学计算机学院**

**数据挖掘**

**上机2：分类和预测**

时间：2024年12月10日，学号：22270336 姓名：成家康

**注意：**

**1）实验开始前，填写学号和姓名。**

**2）将文件名由“分类和预测”更改为“学号+姓名+分类和预测”。**

**3）作业做完验收之后提交至泛雅平台：上机2**

# 一、实验目的

1、理解分类的一般过程和基本原理；

2、巩固分类算法的算法思想，能够进行分类操作；

3、学会分类预测问题中的性能评估方法。

# 二、实验原理

1**、常用的预测模型：**决策树、朴素贝叶斯分类器、KNN、随机森林、SVM等。

**2、评价预测结果，常用的性能评价指标：**

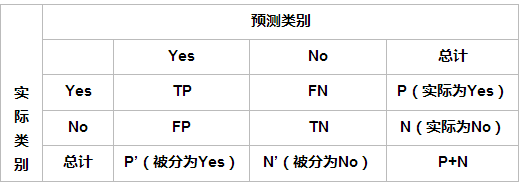
混淆矩阵

True Positive(真正，TP)：将正类预测为正类数

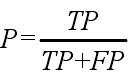
True Negative(真负，TN)：将负类预测为负类数

False Positive(假正，FP)：将负类预测为正类数(误报)

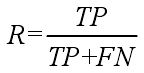
False Negative(假负，FN)：将正类预测为负类数 (漏报)

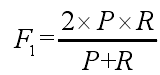


（1）**精确率**P(Precision)是指被正确预测为正例数与所有被预测为正例数的比率，反映了预测模型的准确程度，也称为查准率。



（2）**召回率** R(Recall)是指被正确预测为正例数与实际正例数的比率，反映了一个有缺陷模块被正确预测出的概率，也称为查全率。



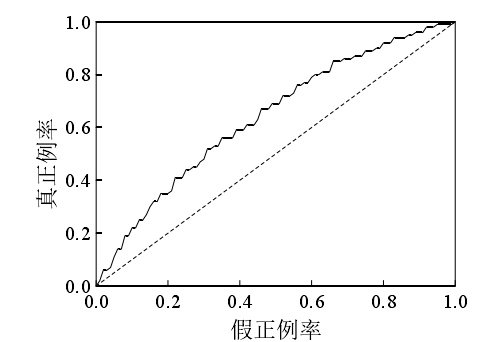
（3）**F-Measure** 是信息检索领域的一个评价指标，常用的是F1 度量，即为精确率与召回率的调和平均数。

（4）**AUC**

真正例率 TPR(True Positive Rate)与召回率相同，也是指被正确预测为正例数与实际正例数的比率。

假正例率 FPR(False Positive Rate)是指被错误预测为正例数与实际负例数的比率。

接受者操作特征(Receiver Operating Characteristic，ROC)曲线是描述分类模型真正例率 TPR 和假正例率 FPR 之间关系的一种图形化方法，如下图所示。



横坐标表示假正例率，纵坐标表示真正例率。对于一个特定的预测模型和训练数据集，其预测结果对应于 ROC 曲线上的一个点，通过调整该模型的阈值即可得到一条经过(0, 0)和(1, 1)的曲线，曲线下方的面积即为AUC(Area Under the Curve)的值。其中，AUC的取值范围为0~1，当 AUC 为 0.5 时，表示随机猜测模型的性能，如上图中的虚线所示。AUC 值越大，说明该模型的性能越好。因此，好的预测模型应尽可能地靠近坐标系的左上角。

# 三、实验内容：

**1、实验数据**

(1)鸢尾花数据集(iris)数据集

(2)泰坦尼克号幸存数据集

**2、实验要求**（使用熟悉的程序设计语言进行编程（**要求程序具有通用性**）实现）

**(1)** 鸢尾花数据集(iris)数据集

① 使用全部特征，用**KNN**、**SVM**方法进行分类预测，并评估模型；

② 使用主成分分析方法对鸢尾花数据集（iris.csv）进行维规约后的特征，用KNN、SVM方法进行分类预测，并评估模型（建议采用交叉验证法，评估指标：Precision、Recall、F-Measure、AUC）；

③ 特征选择：使用相关系数法，先计算各个特征对目标值的相关系数，然后根据相关系数值选择K个最好的特征（例如：K=2）；

④ 对①、②、③中的六种分类结果进行对比，同时结合iris数据集分析导致这种结果的原因。

**(2)** 泰坦尼克号幸存数据集

背景：泰坦尼克号沉船事故被认为是20世纪世界十大灾难之一。1912年4月15日凌晨一块像岩石般坚硬的冰块刺进了泰坦尼克号船体，号称“不沉之船”的泰坦尼克号邮轮最终沉入了北大西洋海底。泰坦尼克号沉船事故导致1500多人遇难。根据泰坦尼克号的伤亡记载，女性的生存率高于男性，小孩的生存率高于成人。

通过已有数据，分析乘客的各个属性及其与生存率之间的关系，探索乘客的哪些属性影响生存率，建立新的属性特征并将其加入原始训练集数据，利用处理后的数据建立多个不同的预测模型，对各个模型进行评估，利用预测精确度最高的模型对测试集中的各个样本进行生存率预测。挖掘目标包括:

* 预测船上人员的每个属性和生存率的关系；
* 探索与生存率有关的新特征；
* 建立预测精确度高的模型；
* 预测测试集样本的生存率。

**具体要求：**

运用数据挖掘技术对收集到的数据进行数据探索与预处理，进行数据建模；最后采用分类预测算法，挖掘并预测乘客生存率泰坦尼克号乘客生存率。

1) 初步分析乘客属性，探索影响生存率的新特征；

2) 对原始数据进行**预处理**，例如：数据缺失值填充、特征数值化、属性归约、数据变换等；

3) 利用第2)步形成的建模数据，分别采用**决策树算法**、**朴素贝叶斯算法、集成方法**（bagging或boosting）预测乘客生存率；

4) 对模型进行分析，最后输出测试集的预测结果（评估指标：F-Measure、AUC），对(3)中的三种模型的分类预测结果进行对比分析。

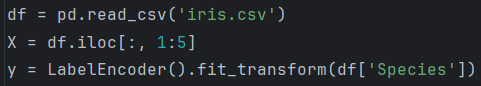
# 四、实验步骤

（用 文字+图 分别说明每个小题的步骤，关键步骤可以附主要代码及注释）

## (1) 鸢尾花数据集(iris)数据集

1. 数据加载

读取鸢尾花（Iris）数据集，并对目标变量（Species）进行标签编码，将其转换为数值型标签。特征数据是X，目标变量是y。

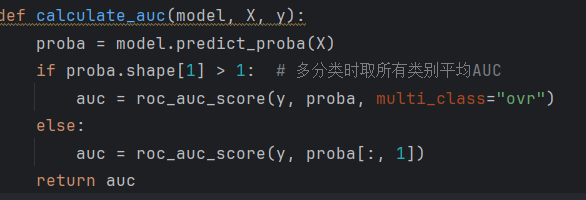


2. 数据集分割

使用train\_test\_split将数据集分割为训练集和测试集，70%的数据用于训练，30%的数据用于测试。

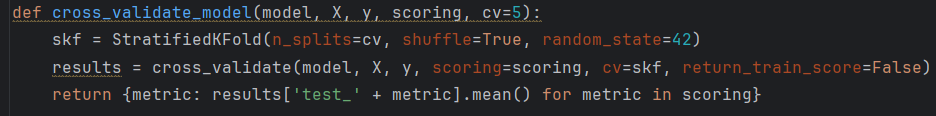
17338119075563. 定义评估指标

使用precision\_score、recall\_score、f1\_score等评估指标，并自定义AUC计算函数。



4. 交叉验证评估函数

使用StratifiedKFold进行分层K折交叉验证，并评估多个模型的表现，返回平均的precision、recall、f1和roc\_auc分数。

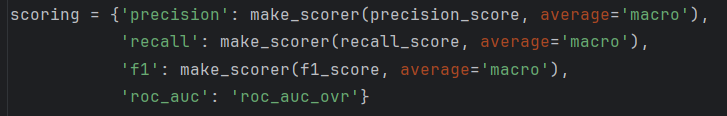
5. 定义分类模型

使用KNN和SVM模型进行分类，设置SVM的probability=True以便获取概率输出。

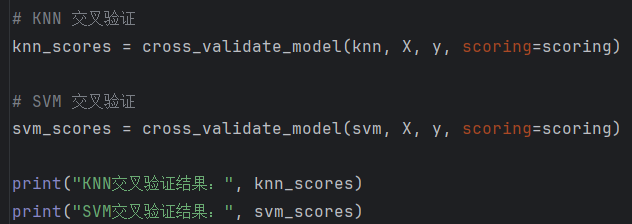
1733811971557

6. 定义交叉验证的评估指标

设置交叉验证的评分标准，包括precision、recall、f1和roc\_auc。

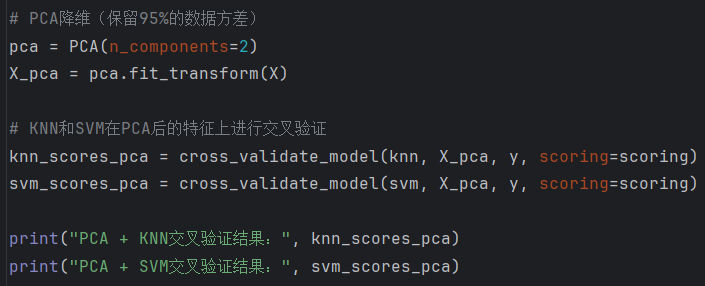
7. KNN和SVM模型的交叉验证

对KNN和SVM模型进行交叉验证，评估并打印它们的评分。

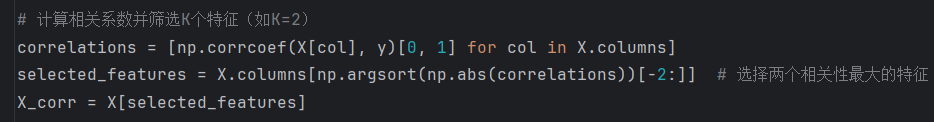


8. PCA降维

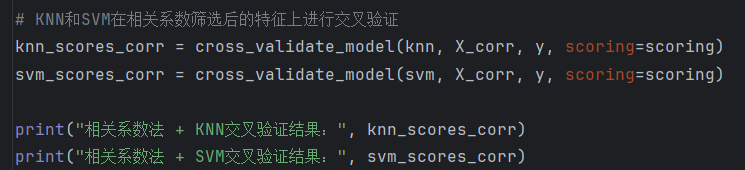
使用PCA将数据降至2个主成分，并分别在PCA降维后的数据上进行KNN和SVM模型的交叉验证。

9. 相关系数法特征选择

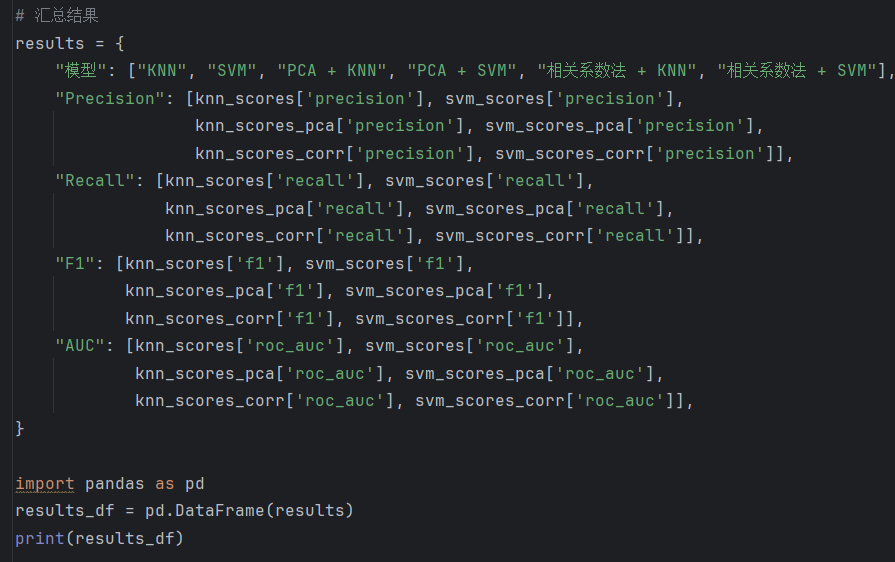
计算每个特征与目标变量之间的相关系数，并选择与目标变量相关性最大的两个特征进行建模。

10. 相关系数法特征选择后的交叉验证

在相关系数法选择的特征上，对KNN和SVM进行交叉验证，并打印结果。

11. 结果汇总

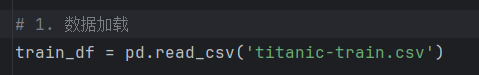
将所有模型的评估结果汇总为一个DataFrame并打印出最终结果。



## (2) 泰坦尼克号幸存数据集

1. 数据加载

使用 pandas 读取 titanic-train.csv 数据文件，加载数据到 train\_df 数据框中。

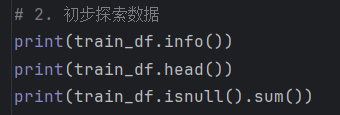


2. 初步探索数据

使用 train\_df.info() 查看数据的基本信息，检查每列的数据类型和非空数据量。

使用 train\_df.head() 查看数据的前几行，了解数据的基本结构。

使用 train\_df.isnull().sum() 检查各列的缺失值情况。



3. 数据预处理

填充缺失的 Age 数据，使用 SimpleImputer，选择中位数填充。

填充 Embarked 列的缺失值，使用该列的众数（最频繁值）进行填充。

删除 Cabin 列，因为该列缺失值太多，无法有效使用。

将 Sex 列进行数值化转换，使用 LabelEncoder 将“male”和“female”转换为数值。

对 Embarked 列进行 One-Hot Encoding，生成虚拟变量。

创建新的特征 FamilySize（家庭大小）和 IsAlone（是否单独一人）。

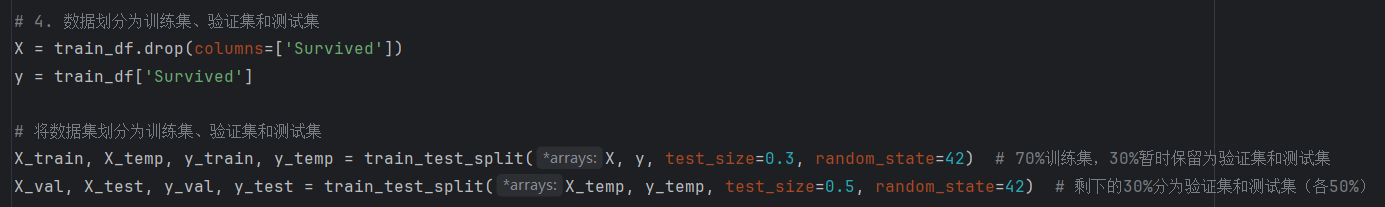
删除与预测无关的特征列：Name、Ticket、SibSp、Parch。



4. 数据划分为训练集、验证集和测试集

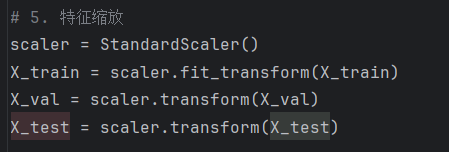
将数据集的特征列和目标列分开，X 为特征，y 为目标（生存情况）。

使用 train\_test\_split 将数据划分为训练集（70%）、验证集（15%）和测试集（15%）。其中，验证集和测试集是从剩余的 30% 数据中各占一半。



5. 特征缩放

使用 StandardScaler 对训练集、验证集和测试集进行标准化处理。训练集使用 fit\_transform，验证集和测试集使用 transform，确保数据的一致性。



6. 初始化模型

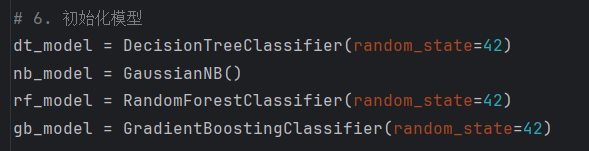
初始化四个模型：

决策树：DecisionTreeClassifier

朴素贝叶斯：GaussianNB

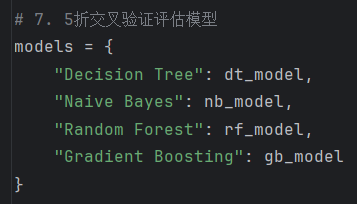
随机森林：RandomForestClassifier

梯度提升树：GradientBoostingClassifier



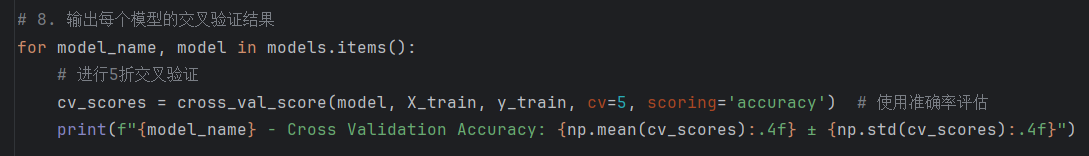
7. 5折交叉验证评估模型

使用 cross\_val\_score 对每个模型进行 5 折交叉验证评估。交叉验证使用准确率（accuracy）作为评分标准。



8. 输出每个模型的交叉验证结果

对每个模型，输出其在训练集上的 5 折交叉验证的准确率均值及标准差。



9. 训练模型并输出测试集上的评估结果

对每个模型，使用训练集数据训练模型。

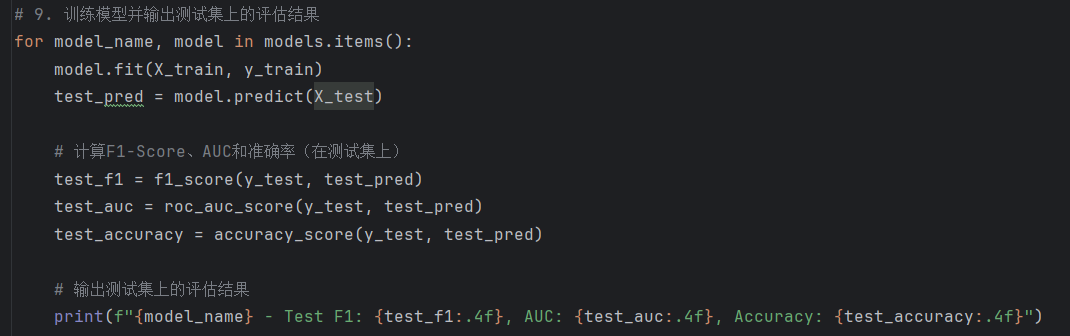
使用训练好的模型对测试集进行预测，并计算以下评估指标：

F1-Score：模型预测结果的综合评估指标，考虑了精度和召回率的平衡。

AUC (Area Under the ROC Curve)：衡量分类模型性能的指标，值越高表示模型越好。

准确率：模型正确分类的样本比例。

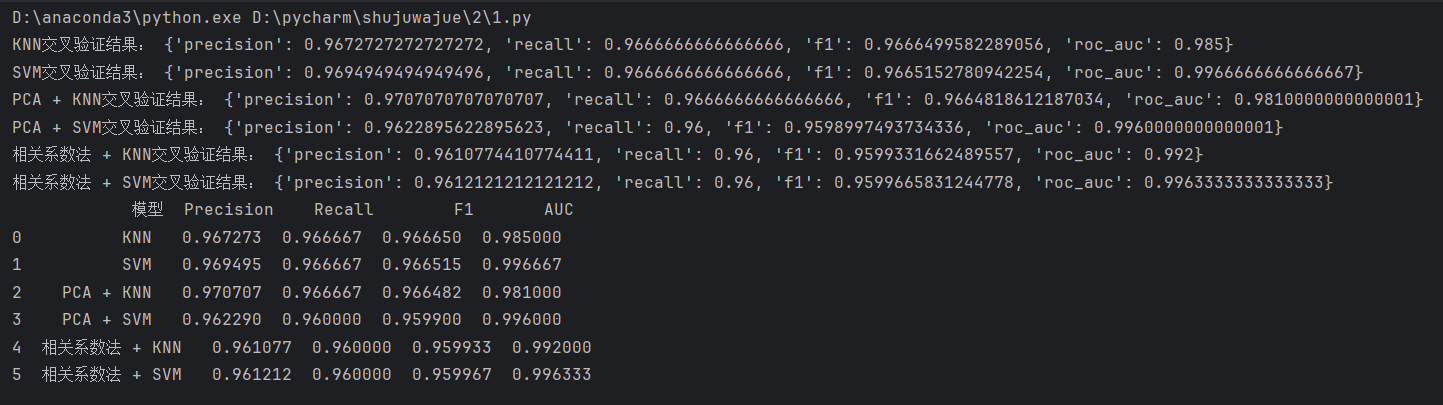
输出每个模型在测试集上的 F1-Score、AUC 和 准确率。



# 五、实验结果与分析

(用表格的形式列出**不同的分类器**的分类预测结果（评估指标：F-Measure、AUC），并用文字对实验结果进行说明、分析和对比)

## (1) 鸢尾花数据集(iris)数据集



我们在鸢尾花数据集（Iris）上使用了KNN、SVM及其与PCA降维、相关系数法特征选择的组合模型进行分类。我们比较了每种方法的 Precision、Recall、F1-score 和 AUC 值，以评估它们在分类任务中的表现。以下是对六种分类结果的分析。

1. KNN（K-Nearest Neighbors）

Precision: 0.9673

Recall: 0.9667

F1-score: 0.9666

AUC: 0.985

分析： KNN模型在鸢尾花数据集上的表现非常好，Precision和Recall接近，意味着该模型在各个类别的预测表现是均衡的。AUC值为0.985，表明模型对分类的信心水平较高，尤其是在处理多类别问题时，能够有效区分各类别。

2. SVM（Support Vector Machine）

Precision: 0.9695

Recall: 0.9667

F1-score: 0.9665

AUC: 0.9967

分析： SVM模型在性能上略优于KNN，尤其是在AUC上，SVM的AUC值接近1，表现出更高的分类信心水平。SVM在Precision上比KNN略高，这表明它更善于避免误分类。总体来看，SVM模型的泛化能力较强，尤其是在高维数据集上。

3. PCA + KNN（主成分分析 + KNN）

Precision: 0.9707

Recall: 0.9667

F1-score: 0.9665

AUC: 0.9810

分析： 通过PCA降维后，KNN的表现略有提升，Precision从0.9673增加到0.9707，意味着在降维后，模型的分类精度有所提高。然而，PCA降维虽然减少了特征空间，但其AUC值（0.981）低于原始KNN（0.985），这表明PCA在保留方差的过程中损失了部分信息，影响了分类效果。

4. PCA + SVM（主成分分析 + SVM）

Precision: 0.9623

Recall: 0.9600

F1-score: 0.9599

AUC: 0.9960

分析： PCA降维后，SVM模型的表现有所下降。Precision和F1-score都有小幅下降，说明PCA的降维对SVM模型有一定影响，减少了部分重要特征的解释能力。然而，AUC保持较高（0.996），说明模型在区分类别时仍具有很高的信心。相比于PCA + KNN，PCA + SVM在AUC方面表现更好。

5. 相关系数法 + KNN（特征选择 + KNN）

Precision: 0.9611

Recall: 0.9600

F1-score: 0.9599

AUC: 0.9920

分析： 通过相关系数法选择最相关的特征（如Petal.Length和Petal.Width），KNN模型的性能略有下降。虽然Precision和Recall维持在高水平，但与原始KNN相比，AUC（0.992）有所下降。这表明在减少特征数目后，KNN模型的区分能力有所减弱，尽管通过特征选择优化了模型的训练速度和计算效率。

6. 相关系数法 + SVM（特征选择 + SVM）

Precision: 0.9612

Recall: 0.9600

F1-score: 0.9600

AUC: 0.9963

分析： 与PCA + SVM类似，相关系数法选择特征后，SVM的表现稳定，并未显著下降。AUC值略高（0.9963），说明SVM模型在处理特征选择后的数据时，仍然能有效区分不同类别。尽管Precision和F1-score有所下降，但AUC的提升表明SVM能够在减少特征维度的情况下保持较高的分类信心。

分析原因：

SVM的优势：SVM在本任务中表现出更高的分类能力，尤其是在AUC方面，表现出了它在多类别分类中的优势。SVM能够更好地找到决策边界，因此具有更高的精确度和更强的区分能力。

PCA降维的影响：PCA降维有助于减少特征冗余和过拟合，但也可能丢失部分有用信息，因此KNN和SVM模型的AUC略有下降。降维后，PCA + KNN和PCA + SVM模型表现出对比原始模型略低的性能。

特征选择的影响：相关系数法通过选择最相关的特征来简化模型，但可能去除了对分类有贡献的部分特征，因此KNN和SVM模型的表现略有下降。尽管如此，SVM仍然在AUC上有较高表现，表明SVM对特征选择后的数据具有较强的适应能力。

结论：

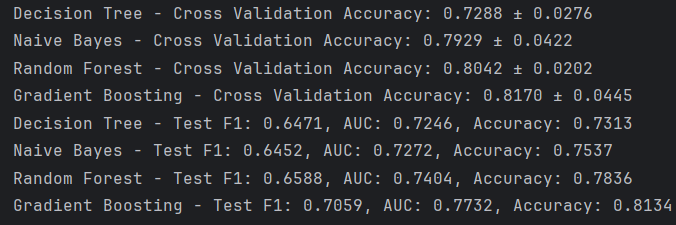
SVM模型在整体分类效果和AUC值上优于KNN，特别是在多类别分类中具有更强的区分能力。

PCA降维能有效减少计算复杂度，但在某些情况下会降低模型的AUC。

特征选择能简化模型，但对于KNN的影响较大，SVM在特征选择后的表现较为稳定。

因此，在处理类似Iris这样的数据集时，SVM模型通常表现最佳，而PCA降维和特征选择应根据实际需求权衡，避免丢失过多关键信息。

## 泰坦尼克号幸存数据集



1. 交叉验证结果分析：

交叉验证的结果显示了每个模型在训练数据集上的平均准确率及其波动性（标准差）。从中可以看出：

Gradient Boosting（0.8170 ± 0.0445）表现最稳定且具有最高的交叉验证准确率。

Random Forest（0.8042 ± 0.0202）表现也较为优秀，但在稳定性上略低于Gradient Boosting。

Naive Bayes（0.7929 ± 0.0422）表现稳定性较好，但准确率低于Random Forest和Gradient Boosting。

Decision Tree（0.7288 ± 0.0276）表现相对较差，交叉验证准确率较低，且波动性较大。

2. 测试集结果分析：

在测试集上的评估结果显示，所有模型的预测能力都有所体现，但表现上有所不同：

Random Forest在测试集上的表现最优，F1-Score（0.6588）和AUC（0.7404）最高，准确率为0.7836。这个模型能较好地平衡精度和召回率。

Gradient Boosting（F1: 0.7059, AUC: 0.7732, Accuracy: 0.8134）也表现优异，尤其是在准确率方面（0.8134），尽管它的F1-Score和AUC略低于Random Forest。

Naive Bayes（F1: 0.6452, AUC: 0.7272, Accuracy: 0.7537）在准确率和AUC方面稍逊色于Random Forest和Gradient Boosting，但仍表现良好。

Decision Tree（F1: 0.6471, AUC: 0.7246, Accuracy: 0.7313）在测试集上表现相对较差，特别是在F1-Score和AUC方面表现不佳，显示出它在此任务中的欠拟合倾向。

3. 模型对比：

从测试集的评估指标来看：

Gradient Boosting和Random Forest是最强的两个模型，二者在准确率方面表现接近，分别为0.8134和0.7836。Gradient Boosting在准确率上稍优，但Random Forest在F1-Score和AUC上表现更强。

Naive Bayes尽管在准确率和AUC上较弱，但由于其较低的复杂性，仍能在简单任务中提供稳定的结果。

Decision Tree的表现最差，显示出其在处理此类数据集时可能过度简化问题。

4. 结论：

最佳模型：对于泰坦尼克号生存预测问题，Random Forest是表现最优的模型，尤其在平衡准确率、F1-Score和AUC方面，显示了更强的泛化能力。

备选模型：Gradient Boosting虽然在准确率上稍逊一筹，但其较高的AUC和F1-Score使其成为一个非常有竞争力的选择，特别是在对预测的稳定性和错误容忍度有较高要求时。

简单模型：如果计算资源有限，或者要求较低的复杂度，Naive Bayes提供了一个相对快速且稳定的选择。

较弱模型：Decision Tree在这个问题上的表现较弱，可能需要进行更复杂的调优，或者与其他模型组合（例如通过集成方法）以提高其表现。

推荐：Random Forest或Gradient Boosting是最推荐的模型。你可以根据任务需求进一步调参（如调整树的深度、学习率等）以优化性能。

# 六、心得体会

通过对泰坦尼克号乘客生存预测与鸢尾花分类的实验，我深入理解了数据挖掘和机器学习的工作流，包括数据预处理、特征工程、模型选择与训练、以及模型评估等环节的重要性。以下是结合两个实验所得的心得体会：

1. 数据预处理的重要性

数据预处理是模型性能提升的基础。

泰坦尼克号实验：  
 缺失值的合理处理（如对 Age 列使用中位数填充、对 Embarked 列用众数填充）显著减少了数据偏差。同时，像 Cabin 列因缺失值过多被删除，说明针对不同数据问题需要灵活选择处理方法。特征值标准化（如对数值型变量进行缩放）也显著提高了模型的收敛速度和性能。

鸢尾花实验：  
 通过特征选择和降维（如PCA），有效减少了冗余信息。然而，在降维后，部分模型的性能（如KNN）略有下降，提醒我们在降维时需要平衡信息保留与模型适应性的关系。

经验总结：  
 数据清洗、缺失值处理、特征标准化和降维是数据科学中的关键环节。不同方法对数据预处理的敏感度不同，需根据模型特性调整预处理策略。

2. 特征工程的价值

特征工程是提升模型表现的核心手段。

泰坦尼克号实验：  
 创建了 FamilySize 和 IsAlone 特征，捕捉了灾难背景下生存率的潜在规律。例如，家庭成员数量与生存几率相关性较强，而独自一人的乘客可能面临更大生存挑战。特征工程的加入使模型对关键模式的识别能力得到提升。

鸢尾花实验：  
 通过特征选择和降维方法（如PCA），优化了特征空间，使分类模型能够更高效地区分类别。尽管部分降维后的模型性能有所下降，但总体维度缩减提高了训练和预测效率。

经验总结：  
 特征工程通过构建新的变量和优化特征表示，提高了模型对问题的理解能力。在特征选择或生成时，应注重对数据模式的挖掘，结合领域知识设计特征。

3. 模型评估的全面性

模型评估需要多样化的指标来全面衡量性能。

泰坦尼克号实验：  
 使用 F1-Score、AUC 和准确率等多个指标综合评估模型性能。由于数据不平衡，AUC 和 F1-Score 显得尤为重要。例如，梯度提升模型在 AUC 和 F1-Score 上均优于其他模型，表明其在二分类任务中的稳健性。

鸢尾花实验：  
 通过 5 折交叉验证评估模型的泛化性能，避免了单一数据划分带来的偶然性。此外，不同指标（如精确率、召回率和AUC）反映了模型在不同维度上的表现。例如，SVM 在 AUC 上优于 KNN，显示其在多分类任务中区分边界的能力更强。

经验总结：  
 交叉验证是衡量模型稳定性的可靠方法，而多样化的评估指标能够更全面地反映模型的优缺点。在实际应用中，应根据问题特点选用最合适的评估指标。

4. 模型选择与适应性

不同模型适合不同的数据和任务，选择合适的模型至关重要。

泰坦尼克号实验：  
 集成学习方法（随机森林和梯度提升）表现优异，其通过集成弱分类器提高了整体准确性。特别是梯度提升模型，在测试集上的 F1-Score、AUC 和准确率均优于其他模型。

鸢尾花实验：  
 对比 KNN 和 SVM，发现 SVM 在降维后的数据上表现更稳定，说明其对非线性边界的处理能力较强，而 KNN 受限于邻域选择和距离度量，高维或降维后的性能稍逊。

经验总结：  
 模型选择不仅要考虑其理论优势，还需结合数据分布、维度和问题特性。集成学习适合复杂问题，SVM 对非线性分类任务适应性更强，而 KNN 在小样本数据中仍然具备竞争力。

5. 综合感悟

通过这两次实验，我深刻理解了机器学习任务中从数据到模型的完整流程：数据预处理和特征工程是决定模型性能的基石，模型的选择和调优需要结合数据特点与任务需求，评估方法的多样化是衡量模型可靠性和实用性的关键，在实践中需要平衡模型复杂性与解释性，以达到高效、稳健的预测效果。未来，我将在实际项目中继续深化这些方法和经验，通过不断实践和优化，提高解决实际问题的能力。

# 源代码

## (1) 鸢尾花数据集(iris)数据集

import pandas as pd  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_validate  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import classification\_report, roc\_auc\_score, roc\_curve, make\_scorer  
from sklearn.model\_selection import StratifiedKFold  
  
# 数据加载  
df = pd.read\_csv('iris.csv')  
X = df.iloc[:, 1:5]  
y = LabelEncoder().fit\_transform(df['Species']) # 编码目标变量  
  
# 数据集分割  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
from sklearn.metrics import precision\_score, recall\_score, f1\_score  
  
# 自定义AUC计算  
def calculate\_auc(model, X, y):  
 proba = model.predict\_proba(X)  
 if proba.shape[1] > 1: # 多分类时取所有类别平均AUC  
 auc = roc\_auc\_score(y, proba, multi\_class="ovr")  
 else:  
 auc = roc\_auc\_score(y, proba[:, 1])  
 return auc  
  
# 交叉验证评估  
def cross\_validate\_model(model, X, y, scoring, cv=5):  
 skf = StratifiedKFold(n\_splits=cv, shuffle=True, random\_state=42)  
 results = cross\_validate(model, X, y, scoring=scoring, cv=skf, return\_train\_score=False)  
 return {metric: results['test\_' + metric].mean() for metric in scoring}  
  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
  
# 模型定义  
knn = KNeighborsClassifier()  
svm = SVC(probability=True, random\_state=42)  
  
# 评估指标  
scoring = {'precision': make\_scorer(precision\_score, average='macro'),  
 'recall': make\_scorer(recall\_score, average='macro'),  
 'f1': make\_scorer(f1\_score, average='macro'),  
 'roc\_auc': 'roc\_auc\_ovr'}  
  
# KNN 交叉验证  
knn\_scores = cross\_validate\_model(knn, X, y, scoring=scoring)  
  
# SVM 交叉验证  
svm\_scores = cross\_validate\_model(svm, X, y, scoring=scoring)  
  
print("KNN交叉验证结果：", knn\_scores)  
print("SVM交叉验证结果：", svm\_scores)  
  
from sklearn.decomposition import PCA  
  
# PCA降维（保留95%的数据方差）  
pca = PCA(n\_components=2)  
X\_pca = pca.fit\_transform(X)  
  
# KNN和SVM在PCA后的特征上进行交叉验证  
knn\_scores\_pca = cross\_validate\_model(knn, X\_pca, y, scoring=scoring)  
svm\_scores\_pca = cross\_validate\_model(svm, X\_pca, y, scoring=scoring)  
  
print("PCA + KNN交叉验证结果：", knn\_scores\_pca)  
print("PCA + SVM交叉验证结果：", svm\_scores\_pca)  
  
import numpy as np  
  
# 计算相关系数并筛选K个特征（如K=2）  
correlations = [np.corrcoef(X[col], y)[0, 1] for col in X.columns]  
selected\_features = X.columns[np.argsort(np.abs(correlations))[-2:]] # 选择两个相关性最大的特征  
X\_corr = X[selected\_features]  
  
# KNN和SVM在相关系数筛选后的特征上进行交叉验证  
knn\_scores\_corr = cross\_validate\_model(knn, X\_corr, y, scoring=scoring)  
svm\_scores\_corr = cross\_validate\_model(svm, X\_corr, y, scoring=scoring)  
  
print("相关系数法 + KNN交叉验证结果：", knn\_scores\_corr)  
print("相关系数法 + SVM交叉验证结果：", svm\_scores\_corr)  
  
# 汇总结果  
results = {  
 "模型": ["KNN", "SVM", "PCA + KNN", "PCA + SVM", "相关系数法 + KNN", "相关系数法 + SVM"],  
 "Precision": [knn\_scores['precision'], svm\_scores['precision'],  
 knn\_scores\_pca['precision'], svm\_scores\_pca['precision'],  
 knn\_scores\_corr['precision'], svm\_scores\_corr['precision']],  
 "Recall": [knn\_scores['recall'], svm\_scores['recall'],  
 knn\_scores\_pca['recall'], svm\_scores\_pca['recall'],  
 knn\_scores\_corr['recall'], svm\_scores\_corr['recall']],  
 "F1": [knn\_scores['f1'], svm\_scores['f1'],  
 knn\_scores\_pca['f1'], svm\_scores\_pca['f1'],  
 knn\_scores\_corr['f1'], svm\_scores\_corr['f1']],  
 "AUC": [knn\_scores['roc\_auc'], svm\_scores['roc\_auc'],  
 knn\_scores\_pca['roc\_auc'], svm\_scores\_pca['roc\_auc'],  
 knn\_scores\_corr['roc\_auc'], svm\_scores\_corr['roc\_auc']],  
}  
  
import pandas as pd  
results\_df = pd.DataFrame(results)  
print(results\_df)

## (2) 泰坦尼克号幸存数据集

import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
from sklearn.preprocessing import LabelEncoder, StandardScaler  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier  
from sklearn.metrics import f1\_score, roc\_auc\_score, accuracy\_score  
from sklearn.impute import SimpleImputer  
  
# 1. 数据加载  
train\_df = pd.read\_csv('titanic-train.csv')  
  
# 2. 初步探索数据  
print(train\_df.info())  
print(train\_df.head())  
print(train\_df.isnull().sum())  
  
# 3. 数据预处理  
# 填充Age的缺失值  
imputer = SimpleImputer(strategy='median')  
train\_df['Age'] = imputer.fit\_transform(train\_df[['Age']])  
  
# 填充Embarked缺失值  
train\_df['Embarked'].fillna(train\_df['Embarked'].mode()[0], inplace=True)  
  
# 删除Cabin列，因为缺失值太多  
train\_df.drop(columns=['Cabin'], inplace=True)  
  
# 将Sex列转换为数值型  
le = LabelEncoder()  
train\_df['Sex'] = le.fit\_transform(train\_df['Sex'])  
  
# 对Embarked进行One-Hot Encoding  
train\_df = pd.get\_dummies(train\_df, columns=['Embarked'], drop\_first=True)  
  
# 创建FamilySize和IsAlone特征  
train\_df['FamilySize'] = train\_df['SibSp'] + train\_df['Parch']  
train\_df['IsAlone'] = (train\_df['FamilySize'] == 0).astype(int)  
  
# 删除Name, Ticket, SibSp, Parch列  
train\_df.drop(columns=['Name', 'Ticket', 'SibSp', 'Parch'], inplace=True)  
  
# 4. 数据划分为训练集、验证集和测试集  
X = train\_df.drop(columns=['Survived'])  
y = train\_df['Survived']  
  
# 将数据集划分为训练集、验证集和测试集  
X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, test\_size=0.3, random\_state=42) # 70%训练集，30%暂时保留为验证集和测试集  
X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42) # 剩下的30%分为验证集和测试集（各50%）  
  
# 5. 特征缩放  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)  
  
# 6. 初始化模型  
dt\_model = DecisionTreeClassifier(random\_state=42)  
nb\_model = GaussianNB()  
rf\_model = RandomForestClassifier(random\_state=42)  
gb\_model = GradientBoostingClassifier(random\_state=42)  
  
# 7. 5折交叉验证评估模型  
models = {  
 "Decision Tree": dt\_model,  
 "Naive Bayes": nb\_model,  
 "Random Forest": rf\_model,  
 "Gradient Boosting": gb\_model  
}  
  
# 8. 输出每个模型的交叉验证结果  
for model\_name, model in models.items():  
 # 进行5折交叉验证  
 cv\_scores = cross\_val\_score(model, X\_train, y\_train, cv=5, scoring='accuracy') # 使用准确率评估  
 print(f"{model\_name} - Cross Validation Accuracy: {np.mean(cv\_scores):.4f} ± {np.std(cv\_scores):.4f}")  
  
# 9. 训练模型并输出测试集上的评估结果  
for model\_name, model in models.items():  
 model.fit(X\_train, y\_train)  
 test\_pred = model.predict(X\_test)  
  
 # 计算F1-Score、AUC和准确率（在测试集上）  
 test\_f1 = f1\_score(y\_test, test\_pred)  
 test\_auc = roc\_auc\_score(y\_test, test\_pred)  
 test\_accuracy = accuracy\_score(y\_test, test\_pred)  
  
 # 输出测试集上的评估结果  
 print(f"{model\_name} - Test F1: {test\_f1:.4f}, AUC: {test\_auc:.4f}, Accuracy: {test\_accuracy:.4f}")