**《机器学习基础》**

**Adult数据集**

**课程设计**

**引言**

在当今数字化蓬勃发展的时代，数据已然成为驱动各领域发展与决策的关键要素，而机器学习作为从海量数据中挖掘宝贵信息、洞察潜在规律的强大工具，正发挥着前所未有的重要作用。

本次课程设计聚焦于一个极具现实意义且内容丰富的数据集，其源自美国人口普查数据，涵盖了 48,842 个样本以及 15 个特征维度，犹如一幅描绘社会经济状况的精细画卷，囊括了年龄、工作类别、教育程度、资本收益等多个关键方面。这些特征不仅反映了个体在社会中的基本属性，更与收入水平和教育程度有着千丝万缕的内在联系。我们旨在运用机器学习这一强大技术，对这一数据集进行全面且深入的探索与分析，通过构建精准高效的预测模型，实现对收入水平（predclass）的准确分类预测以及对教育程度（education）的可靠回归预测。

在整个课程设计的过程中，全身心地投入到机器学习的各个环节，从繁杂的数据预处理工作开始，包括但不限于对缺失值的巧妙处理、对类别型数据的合理编码转换以及对数据的标准化操作，确保数据的质量和可用性。随后，在模型选择与调优的关键阶段，将深入研究和对比逻辑回归、支持向量机、随机森林等多种经典机器学习模型，依据数据集的独特特点和实际任务需求，精细调整模型的各项参数，运用交叉验证等科学方法确保模型具备出色的泛化能力，能够在未知数据上稳定且准确地发挥作用。最后，通过严谨的模型评估环节，运用准确率、混淆矩阵、分类报告、ROC 曲线和 AUC 等专业指标对模型性能进行全面且细致的评估，从而筛选出最优模型，并深入理解模型的优势与局限。

**目录**

[一、数据集介绍 1](#_Toc12659)

[二、数据分析 1](#_Toc7735)

[三、流程图 4](#_Toc24373)

[四、数据预处理 5](#_Toc9866)

[1. 数据清洗 5](#_Toc4361)

[2. 特征编码与标准化 6](#_Toc12743)

[五、模型选择与原理 6](#_Toc2509)

[（一）随机森林（Random Forest） 6](#_Toc549)

[（二）逻辑回归（Logistic Regression） 8](#_Toc19024)

[（三）支持向量机（Support Vector Machine） 9](#_Toc1769)

[（四）线性回归（Linear Regression） 10](#_Toc22468)

[（五）多项式回归（Polynomial Regression） 11](#_Toc16242)

[六、分类任务——预测preclass 12](#_Toc28444)

[（一）确定目标 12](#_Toc2992)

[（二）模型选择与调优 13](#_Toc21545)

[（三）模型评估 14](#_Toc32077)

[（四）特征重要性分析 16](#_Toc8868)

[（五）决策边界可视化 16](#_Toc5897)

[七、分类任务——使用“fnlwgt”作为权重预测preclass 17](#_Toc30861)

[（一）提取权重列 17](#_Toc5149)

[（二）模型选择与调优 18](#_Toc19590)

[（三）模型评估 18](#_Toc24488)

[（四）特征重要性分析 19](#_Toc6172)

[（五）决策边界可视化 20](#_Toc25701)

[八、回归任务 ——预测education 20](#_Toc32012)

[（一）确定特征与目标 20](#_Toc24386)

[（二）模型选择与调优 21](#_Toc1549)

[（三）模型评估 21](#_Toc29590)

[（四）相关性分析 21](#_Toc26286)

[（五）预测结果可视化 22](#_Toc30205)

[九、总结 23](#_Toc31549)

[十、参考文献 24](#_Toc19240)

[十一、感想 25](#_Toc4312)

[（一）遇到的困难 25](#_Toc25125)

[（二）收获 26](#_Toc32534)

[（三）感想 27](#_Toc19944)

[十二、附录 28](#_Toc10221)

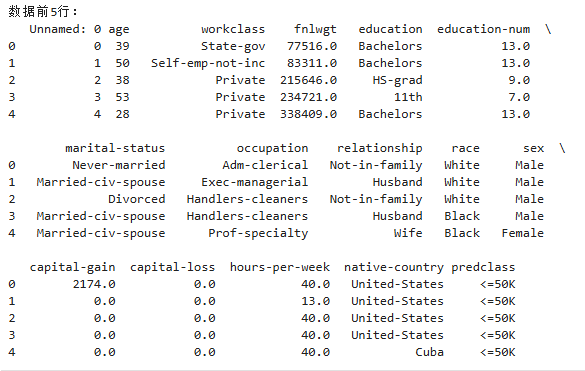
## 

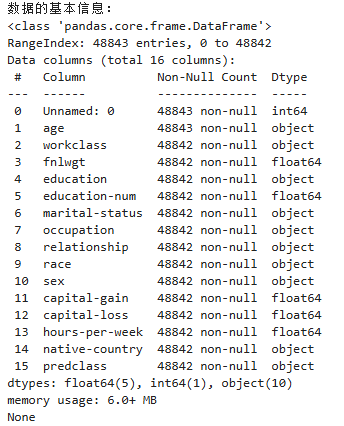
## **一、数据集介绍**

本实验所使用的数据集来源于美国人口普查数据，涵盖了 48,842 个样本，每个样本包含 15 个特征，具体特征信息如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性名** | **描述** | **数据类型** | **示例值** |
| Unnamed: 0 | 索引列 | int64 | 0, 1, 2, … |
| age | 年龄 | object | 39, 50, 38, … |
| workclass | 工作类别 | object | State-gov, Self-emp-not-inc, Private, … |
| fnlwgt | 权重 | float64 | 77516.0, 83311.0, 215646.0, … |
| education | 教育程度 | object | Bachelors, HS-grad, 11th, … |
| education-num | 受教育年份 | float64 | 13.0, 9.0, 7.0, … |
| marital-status | 婚姻状况 | object | Never-married, Married-civ-spouse, Divorced, … |
| occupation | 职业 | object | Adm-clerical, Exec-managerial, Handlers-cleaners, … |
| relationship | 家庭关系 | object | Not-in-family, Husband, Wife, … |
| race | 种族 | object | White, Black, … |
| sex | 性别 | object | Male, Female |
| capital-gain | 资本收益 | float64 | 40.0, 2174.0, 0.0, … |
| capital-loss | 资本损失 | float64 | 0.0, 0.0, 0.0, … |
| hours-per-week | 每周工作小时数 | float64 | 40.0, 40.0, 40.0, … |
| native-country | 原籍国家 | object | United-States, Cuba, … |
| predclass | 收入水平 | object | <=50K 、 > 50K、<=50K.、>50K. |

## **二、数据分析**

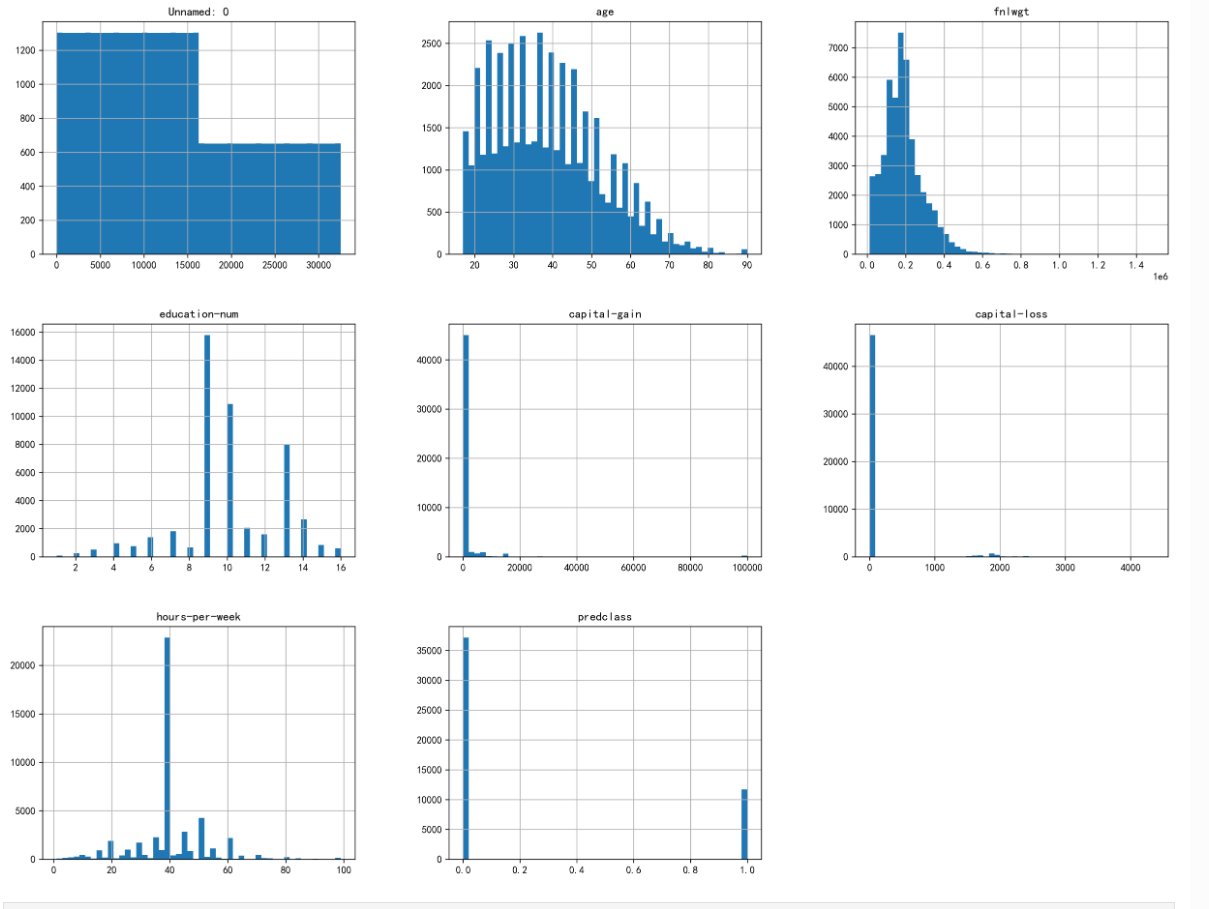




#### **（一）数据整体结构**

* **数据类型**：数据集以pandas.core.frame.DataFrame的形式呈现，这是 Python 中用于数据分析的常用数据结构，方便进行数据处理和分析操作。
* **数据规模**：数据集包含 48843 个条目，索引范围从 0 到 48842，表明数据量较大，能够为分析提供较为丰富的信息。
* **列信息**：
  + 共有 16 列，每列都有特定的含义和数据类型。其中，float64类型的列有 5 列，int64类型的列有 1 列，object类型的列有 10 列。这种数据类型的多样性反映了数据集所涵盖信息的丰富性和复杂性。

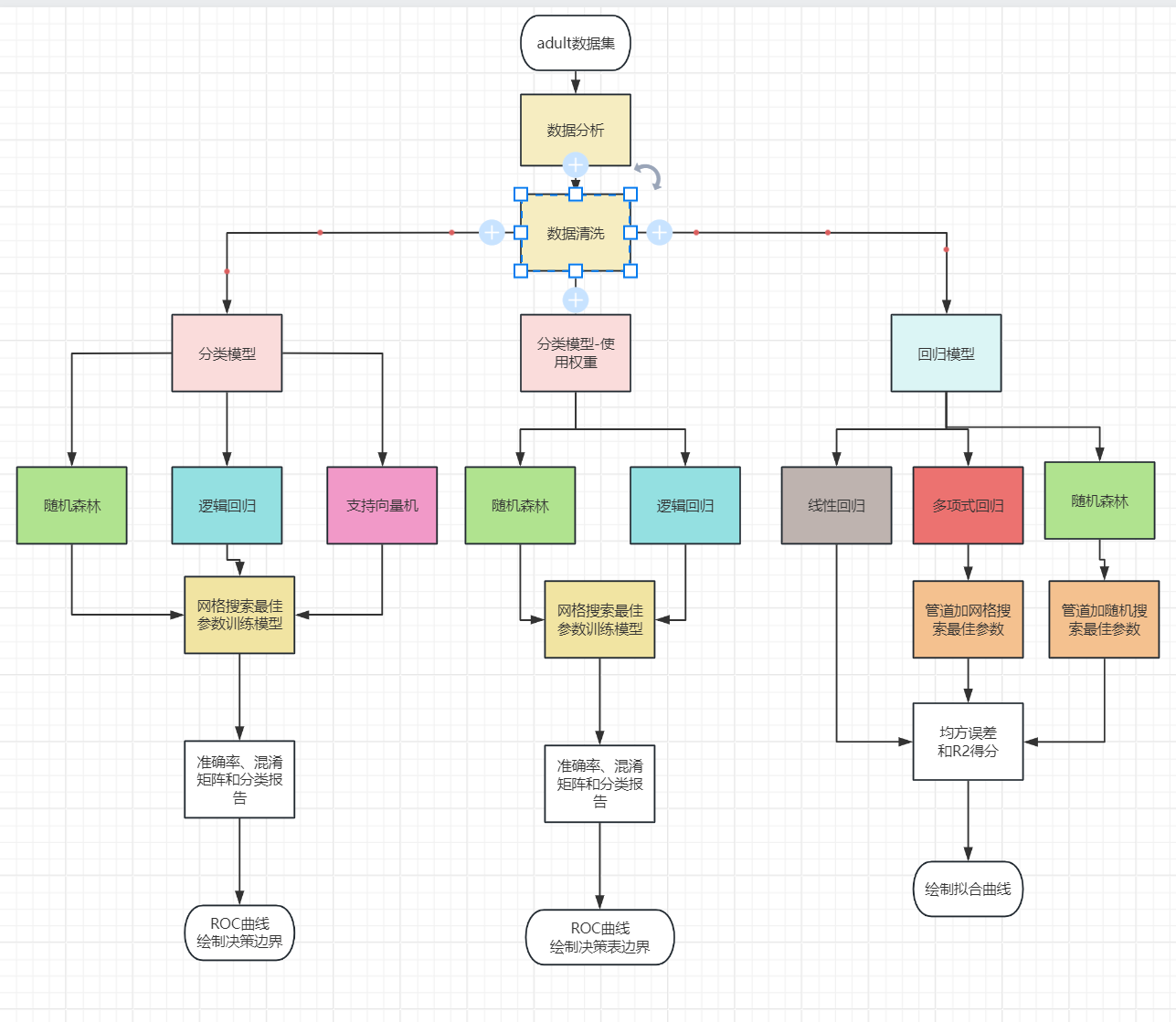
#### **（二）数据初步可视化分析**



通过绘制部分字段的直方图，对数据的分布情况进行了初步可视化，以便更直观地了解数据的特征：

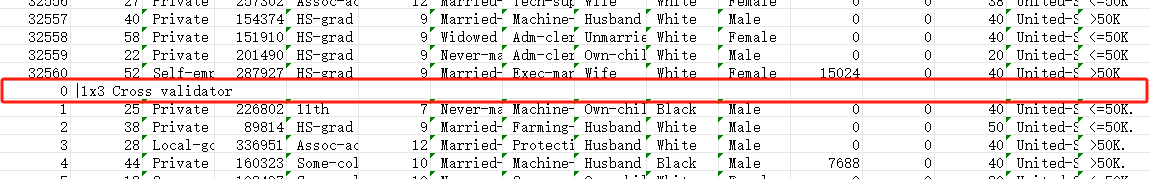
* **Unnamed: 0**：直方图显示该列数据在 0 到 15000 左右的区间内分布较为均匀，之后数量有所减少，但整体呈现出一定的规律性，可能与数据的采集或记录方式有关。
* **age**：该直方图呈现出一种近似正态分布但又不完全对称的形态。在 20 岁至 50 岁左右的区间内，数据分布较为密集，形成了多个较高的柱状条，其中 30 岁左右和 40 岁左右的区间柱状条高度相对较高，从 50 岁往后，数据量逐渐减少，柱状条的高度逐渐降低，到 80 岁及以上年龄段，数据量已经非常少，柱状条变得很矮，甚至在 90 岁左右几乎接近于 0。
* **fnlwgt**：该字段的直方图呈现出右偏态分布，大部分数据集中在较小的值域内，右侧有较长的尾巴，说明存在一些较大的权重值，但数量相对较少。这种分布可能反映了数据集中不同样本的权重差异。
* **education-num**：从直方图可以看出，教育程度数值主要集中在几个特定的值上，如 8、10、12、13 等，这些值可能对应着常见的教育阶段，如高中、大学等。分布的离散性表明教育程度在数据集中具有一定的分类特征。
* **capital-gain**：该字段的直方图显示大部分数据集中在 0 附近，只有少数数据具有较高的资本收益值，说明在数据集中，大多数人可能没有或只有较少的资本收益，而少数人获得了较高的收益。
* **capital-loss**：与资本收益类似，资本损失的直方图也显示大部分数据集中在 0 附近，只有极少数数据有较大的资本损失，表明资本损失在数据集中不是普遍现象。
* **hours-per-week**：直方图显示每周工作小时数主要集中在 40 小时左右，这是一个常见的全职工作时间，同时也有一些人工作时间较少或较多，分布呈现出一定的集中趋势和离散性。
* **preclass**:该直方图呈现出明显的不平衡性。其中，值为 0.0（代表收入类别为 “<=50K”）的柱状条高度非常高，大约有 35000 左右的数据量；而值为 1.0（代表收入类别为 “>50K”）的柱状条高度相对较低，仅有约 10000 左右的数据量。

## 三、流程图

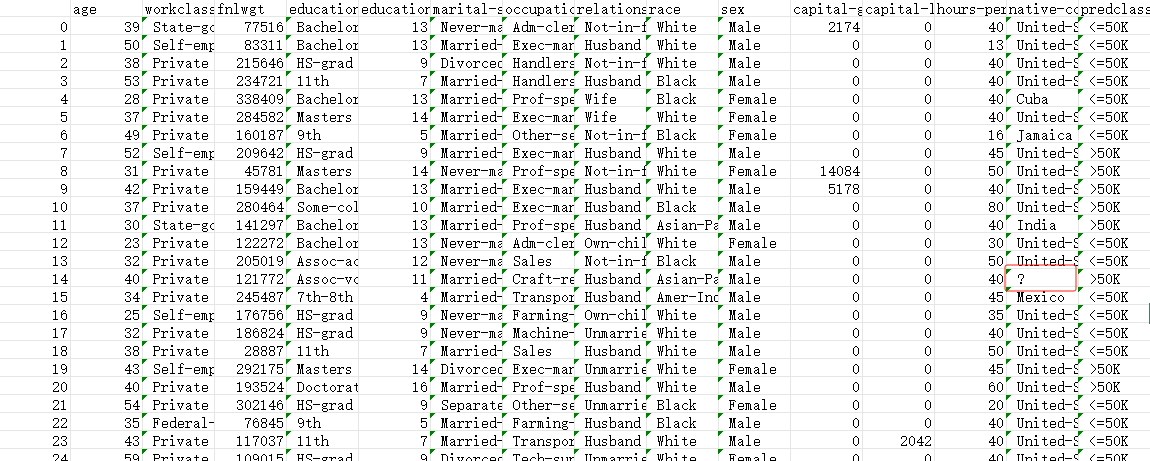


## **四、数据预处理**

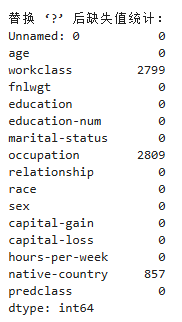
### **数据清洗**

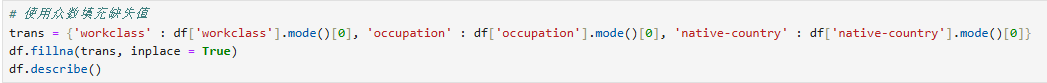


* + 原始数据集中age列数据类型为object，使用pd.to\_numeric函数将其转换为数值型，并将无法转换的值设为NaN，然后删除包含NaN的行。
  + 将predclass列中的类别标签<=50K和<50.映射为 0，>50K和>50.映射为 1，方便后续模型处理。



* + 发现数据集中存在?表示缺失值，使用np.nan替换?，并统计各列缺失值数量。对于缺失值较少的workclass（2799 个）、occupation（2809 个）和native-country（857 个）列，使用各自列的众数填充缺失值。





### **特征编码与标准化**

* + 对于数据集中的分类变量（object类型），使用OrdinalEncoder进行编码转换，将其转换为数值型特征。
  + 对除目标变量外的所有特征使用StandardScaler进行标准化处理，使数据特征具有零均值和单位方差，提高模型训练效果。

## **五、模型选择与原理**

### **（一）随机森林（Random Forest）**

#### **1. 模型选择依据**

* 当数据存在大量的噪声和异常值时，随机森林的集成特性使其能够有效地过滤掉这些干扰因素，因为多个决策树的综合结果可以降低单个异常值的影响。
* 在处理具有多种类型特征（如数值型、分类型、文本型等）的混合数据时表现良好，通过对不同类型特征进行适当的编码和处理后，可以将其整合到随机森林模型中进行分析。
* 适用于需要对特征重要性进行深入分析的场景，例如在市场营销中确定哪些因素（如产品特性、广告渠道、客户群体特征等）对销售业绩的影响最大。

#### **2. 原理**

* 随机森林由多个决策树组成，每个决策树是通过对原始训练数据进行有放回抽样（bootstrap sampling）得到一个新的训练子集来构建的。在每个节点上，从随机选择的一个特征子集（而不是所有特征）中选择最优的分裂特征和分裂点，以最小化节点内数据的不纯度（如基尼系数或信息熵）。在预测阶段，对于分类问题，通过对所有决策树的预测结果进行投票（多数投票）来确定最终的类别；对于回归问题，则对所有决策树的预测结果取平均值作为最终的预测值。

#### **3. 优点**

* 具有很高的预测准确性，尤其是在处理复杂的数据集和高维数据时，能够自动捕捉数据中的非线性关系和交互效应。
* 能够自动处理缺失值，在训练过程中，由于是通过有放回抽样构建决策树，缺失值可以在抽样过程中被合理处理，并且在预测时也可以通过一定的策略（如基于其他样本的预测结果）进行处理。
* 模型具有较好的稳定性和鲁棒性，对数据的微小变化不敏感，不同的训练数据集可能会产生相似的模型结果。

#### **4. 缺点**

* 模型解释性较差，由于是多个决策树的集成，难以直观地理解每个特征是如何影响预测结果的，虽然可以通过特征重要性指标（如特征在决策树中的分裂次数、信息增益等的平均值）来大致了解特征的重要性，但无法像线性模型那样给出明确的数学表达式。
* 训练和预测时间较长，尤其是当决策树数量较多或数据集规模较大时，需要大量的计算资源和时间。对于实时性要求较高的应用场景，可能不太适用。

### **（二）逻辑回归（Logistic Regression）**

#### **1. 模型选择依据**

* 在医疗诊断中，用于根据患者的症状、检查结果等多种因素预测疾病的发生概率，帮助医生进行早期诊断和决策。
* 当数据存在一定的线性趋势，但又不完全符合线性回归的要求（如目标变量是二元分类变量）时，逻辑回归是一种有效的替代方法。
* 在推荐系统中，可以用于预测用户对某个项目的喜欢或不喜欢（二分类），基于用户的历史行为、特征等信息进行建模。

#### **2. 原理**

* 逻辑回归的核心是逻辑函数（sigmoid 函数latexmath），其中latexmath
* 是特征的线性组合。通过最大似然估计法来估计模型的参数，目标是找到一组参数使得在给定训练数据的情况下，观察到这些数据的概率最大。在优化过程中，通常使用梯度下降法或其变种（如随机梯度下降、批量梯度下降等）来最小化损失函数（负对数似然函数）。

#### **3. 优点**

* 模型简单、易于理解和实现，训练速度相对较快，适合处理大规模数据集。
* 可以输出概率值，便于进行概率决策，例如可以根据设定的概率阈值（如 0.5）将样本分为不同的类别。
* 可以方便地进行正则化（如 L1 正则化和 L2 正则化），有助于防止过拟合，提高模型的泛化能力。

#### **4. 缺点**

* 对数据的线性假设较为严格，对于复杂的非线性关系，需要通过特征工程（如添加多项式特征、交互项等）来近似拟合，可能无法充分捕捉数据的非线性特征。
* 对于高度不平衡的数据集，可能会出现分类效果不佳的情况，需要采用一些特殊的处理方法（如过采样、欠采样、调整类别权重等）来提高模型在少数类样本上的性能。

### **（三）支持向量机（Support Vector Machine）**

#### **1. 模型选择依据**

* 在图像识别中，当特征提取已经较为成熟（如通过卷积神经网络提取到的图像特征向量），且需要对图像进行分类（如区分不同的物体类别）时，SVM 可以作为一种有效的分类器。
* 在文本分类任务中，对于经过词袋模型、TF - IDF 等特征提取方法处理后的文本数据，SVM 能够有效地挖掘文本特征之间的关系并进行分类。
* 当数据维度较高且样本数量相对较少时，SVM 的泛化能力通常优于一些其他模型，因为它通过核函数将数据映射到高维空间，从而更好地分离数据。

#### **2. 原理**

* 对于线性可分的数据，SVM 的目标是找到一个超平面latexmath，使得该超平面到两类最近的数据点（支持向量）的距离（间隔）最大。对于非线性可分的数据，通过引入核函数latexmath（如多项式核、高斯核等）将数据映射到高维特征空间latexmath，在高维空间中找到一个线性超平面latexmath。在训练过程中，通过求解一个带约束的二次规划问题来确定最优的超平面和支持向量。

#### **3. 优点**

* 具有较好的泛化能力，在高维空间中能够有效地分离数据，对过拟合有一定的抵抗能力。
* 对于小样本数据表现良好，通过核函数可以将低维空间中的非线性问题转化为高维空间中的线性问题进行处理。
* 一旦训练完成，预测速度较快，只需要计算测试样本与支持向量之间的核函数值。

#### **4. 缺点**

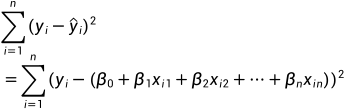
* 对核函数的选择和参数（如核函数的参数、惩罚参数）非常敏感，不同的参数组合可能会导致模型性能差异很大，需要通过大量的实验和交叉验证来确定合适的参数。
* 训练过程的计算复杂度较高，尤其是对于大规模数据集和高维数据，求解二次规划问题的时间和内存消耗较大。在实际应用中，可能需要采用一些优化策略（如简化核函数、使用近似算法等）来提高训练效率。

### **（四）线性回归（Linear Regression）**

#### **1. 模型选择依据**

* 在经济学中，用于分析经济变量之间的长期均衡关系（如消费函数、生产函数等），基于大量的历史数据进行建模。
* 在工程控制领域，用于建立控制系统的输入 - 输出关系模型，以实现对系统的精确控制和预测。
* 当需要对数据进行快速的初步分析和建模，并且假设数据存在线性关系时，线性回归是一个简单有效的选择。

#### **2. 原理**

* 线性回归模型假设目标变量与特征变量之间存在线性关系latexmath，其中是回归系数，是误差项（通常假设服从均值为 0、方差为latexmath的正态分布）。通过最小化残差平方和来估计回归系数。可以使用正规方程latexmath（当latexmath可逆时）或梯度下降等优化算法来求解最优的值。

#### **3. 优点**

* 模型简单直观，回归系数具有明确的经济意义或物理意义（如斜率表示变量之间的变化率），便于解释和理解。
* 计算效率高，尤其是使用正规方程求解时，对于小规模数据集可以快速得到最优解；即使使用梯度下降算法，在数据规模不是特别大的情况下也能较快收敛。
* 可以通过统计检验（如 t 检验、F 检验）对模型的显著性和回归系数的显著性进行评估，为模型的改进和验证提供依据。

#### **4. 缺点**

* 对数据的线性假设要求严格，如果数据存在明显的非线性关系，线性回归模型的拟合效果会很差，预测精度低。
* 对异常值敏感，一个异常值可能会对回归系数和预测结果产生较大的影响，导致模型的偏差较大。
* 只能处理数值型特征（虽然可以通过一些编码方法将分类特征转化为数值特征，但会损失一定的信息），对于复杂的非数值型数据（如文本、图像等）无法直接处理。

### **（五）多项式回归（Polynomial Regression）**

#### **1. 模型选择依据**

* 在物理学中，用于拟合一些物理实验数据，如物体的运动轨迹（在某些情况下可以用多项式来近似描述）、材料的应力 - 应变曲线等。
* 在气象学中，用于预测气温、降水量等随时间或其他因素的变化趋势，当变化趋势呈现出周期性的非线性特征时。
* 当数据的分布呈现出复杂的曲线形状，且无法通过简单的变换（如对数变换、指数变换等）使其线性化时，多项式回归可以作为一种尝试性的建模方法。

#### **2. 原理**

* 多项式回归是在普通线性回归的基础上，将原始特征进行多项式变换，例如对于一个特征x，可以构造新的特征latexmath（d为多项式的次数），然后将这些新特征与原特征一起作为回归模型的输入，即latexmath。通过最小化残差平方和的方法来估计多项式回归模型的系数，同样可以使用正规方程或梯度下降等方法进行求解。

#### **3. 优点**

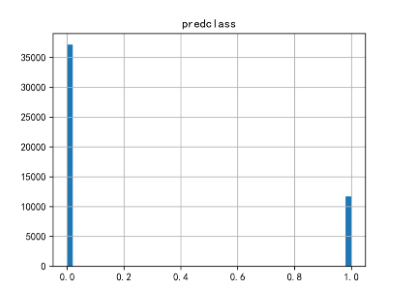
* 能够灵活地拟合各种非线性曲线，通过调整多项式的次数可以适应不同的数据特征和变化趋势，具有较强的灵活性。
* 可以通过对多项式系数的分析，了解数据的非线性变化模式和特征之间的高阶交互作用。

#### **4. 缺点**

* 容易出现过拟合问题，尤其是当多项式次数过高或数据量较少时，模型会过度拟合训练数据，导致在测试集上的性能下降。为了避免过拟合，需要合理选择多项式次数，并结合交叉验证等方法进行模型选择和优化。
* 随着多项式次数的增加，模型的复杂度迅速提高，计算量也会大幅增加，不仅训练时间变长，而且模型的可解释性变差，难以直观地理解高次多项式项与目标变量之间的关系。

## **六、分类任务——预测preclass**

### （一）确定目标



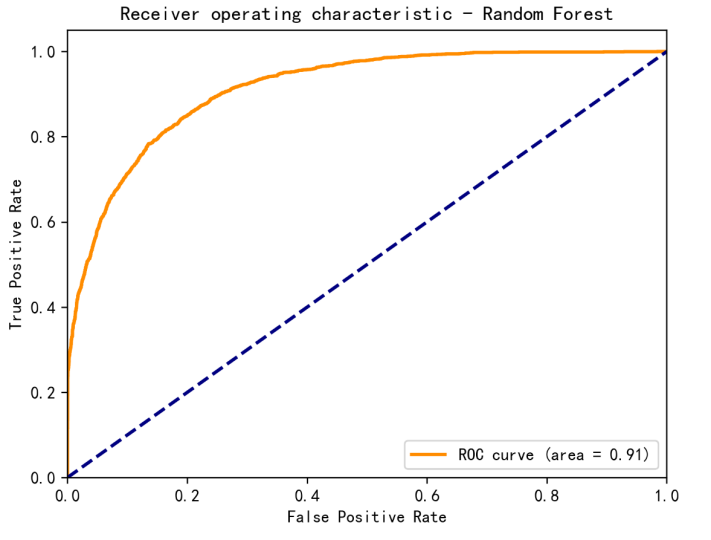
在数据分析中，经过对preclass的转换，观察到preclass只有两种可能，适合分类模型预测，所以选择**preclasss**作为目标列

### **（二）模型选择与调优**

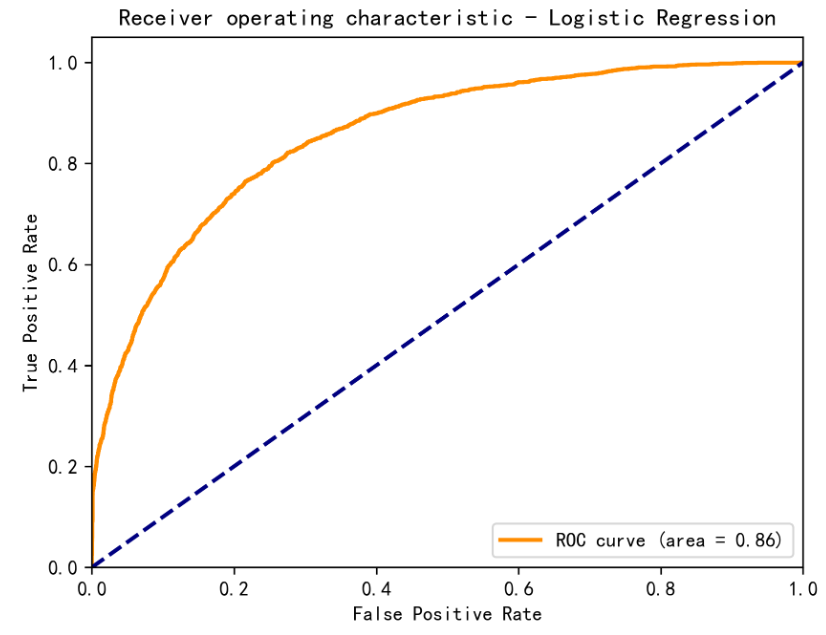
1. **随机森林分类器（RandomForestClassifier）**
   * 初始化随机森林分类器，设置随机种子为 42，最大深度为 10，最小样本分割数为 5，最大特征数为sqrt。
   * 定义参数网格param\_grid\_rf，对n\_estimators进行调优，取值为 [50, 100, 200]。
   * 使用GridSearchCV进行交叉验证和参数调优，选择最优模型。
2. **逻辑回归（LogisticRegression）**
   * 初始化逻辑回归模型，设置随机种子为 42。
   * 定义参数网格param\_grid\_lr，对正则化参数C进行调优，取值为 [0.1, 1, 10]。
   * 同样使用GridSearchCV进行交叉验证和参数调优。
3. **支持向量机（SVC）**
   * 初始化支持向量机模型，设置随机种子为 42，启用概率估计。
   * 定义参数网格param\_grid\_svm，对C和kernel进行调优，C取值为 [0.1, 1, 10]，kernel取值为 ['linear', 'rbf']。
   * 通过GridSearchCV寻找最优参数组合。

### **（三）模型评估**

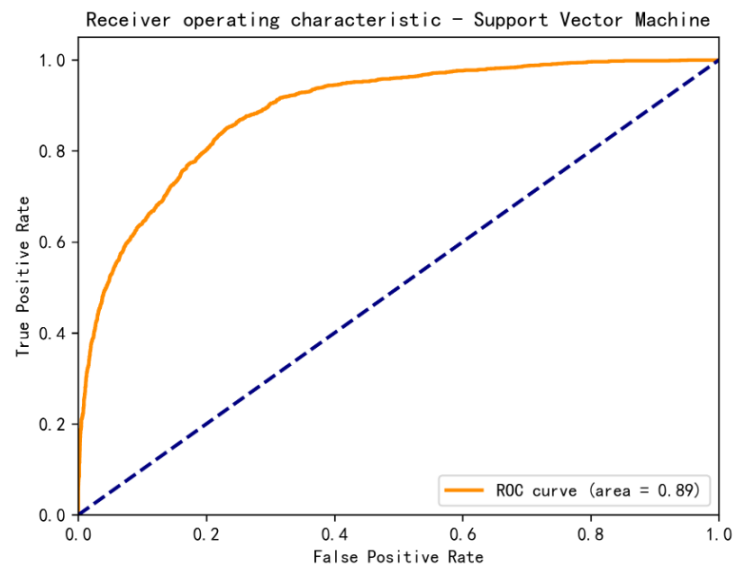
1. **准确率（Accuracy）**
   * 随机森林分类器在测试集上的准确率为 0.8567918927218753。
   * 逻辑回归的准确率为 0.8245470365441704。
   * 支持向量机的准确率为 0.8468625243115979。
2. **混淆矩阵（Confusion Matrix）**
   * 随机森林分类器的混淆矩阵为：latexmath，表示模型对不同类别样本的分类情况。
   * 逻辑回归的混淆矩阵为：latexmath。
   * 支持向量机的混淆矩阵为：latexmath。
3. **分类报告（Classification Report）**
   * 随机森林分类器的分类报告显示，
     + 类别 0 的精确率为 0.87，召回率为 0.96，F1 分数为 0.91；
     + 类别 1 的精确率为 0.81，召回率为 0.53，F1 分数为 0.64。
   * 逻辑回归的分类报告中，
     + 类别 0 的精确率为 0.84，召回率为 0.95，F1 分数为 0.89；
     + 类别 1 的精确率为 0.72，召回率为 0.44，F1 分数为 0.55。
   * 支持向量机的分类报告里，
     + 类别 0 的精确率为 0.86，召回率为 0.95，F1 分数为 0.90；
     + 类别 1 的精确率为 0.76，召回率为 0.54，F1 分数为 0.63。
4. **ROC 曲线与 AUC 值**
   * 绘制了各模型的 ROC 曲线，随机森林分类器的 AUC 值为 0.91，逻辑回归的 AUC 值为 0.86，支持向量机的 AUC 值为 0.89。AUC 值越接近 1，模型性能越好，说明随机森林分类器在区分正负样本方面表现较好。



随机森林——ROC曲线



逻辑回归——ROC曲线



支持向量机——ROC曲线

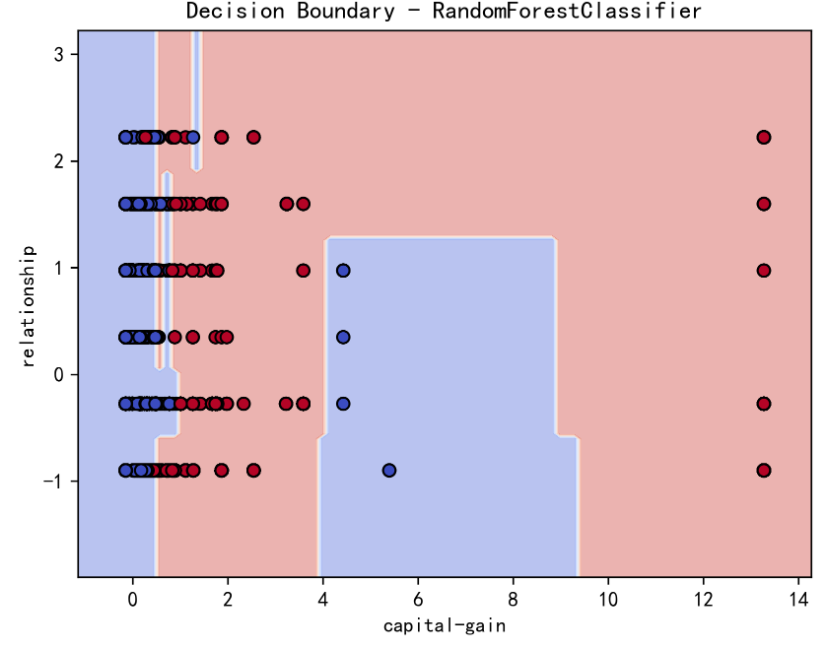
### **（四）特征重要性分析**



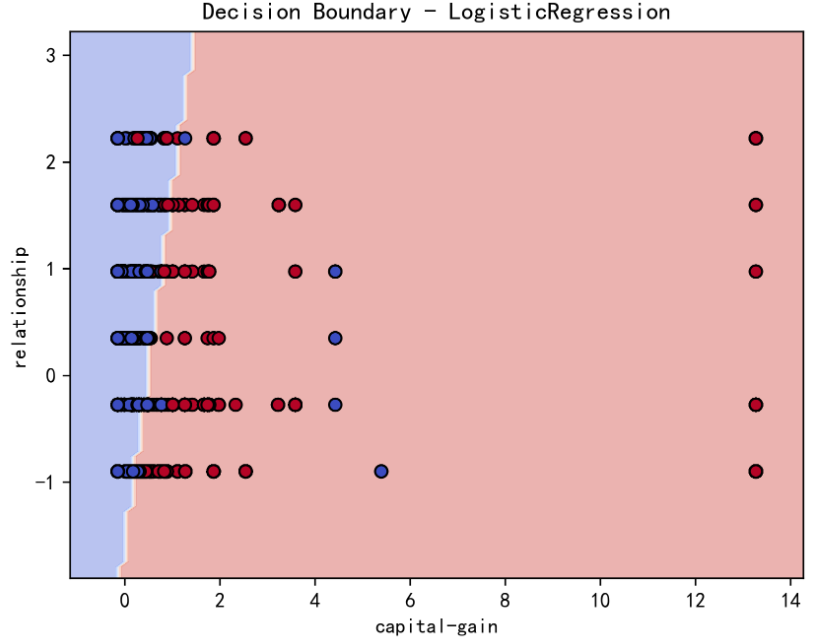
通过随机森林分类器计算特征重要性，发现relationship、capital-gain、marital-status等特征对模型预测较为重要。例如，relationship的特征重要性为 0.22113164161285986，capital-gain的特征重要性为 0.18926237604202620978。

### **（五）决策边界可视化**

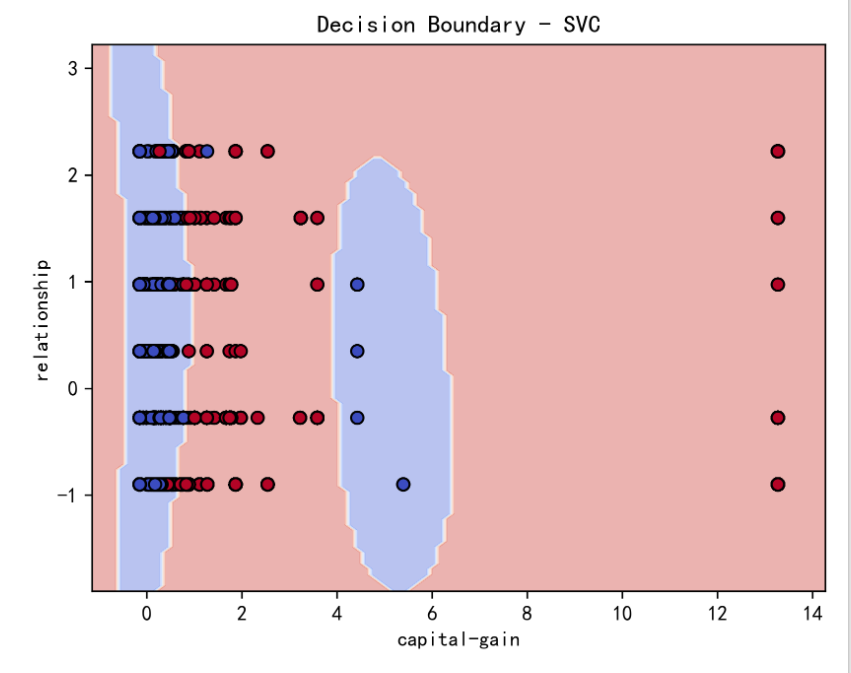
以relationship和capital-gain为特征，绘制了随机森林分类器、逻辑回归和支持向量机的决策边界，直观展示了模型对不同类别样本的划分情况。



随机森林-决策边界



逻辑回归——决策边界



支持向量机——决策边界

## **七、分类任务——使用“fnlwgt”作为权重预测preclass**

### **（一）提取权重列**



### **（二）模型选择与调优**

1. **随机森林分类器（RandomForestClassifier）**
   * 初始化随机森林分类器，设置随机种子为 42，最大深度为 10，最小样本分割数为 5，最大特征数为sqrt。
   * 定义参数网格param\_grid\_rf，对n\_estimators进行调优，取值为 [50, 100, 200]。
   * 使用GridSearchCV进行交叉验证和参数调优，选择最优模型。
2. **逻辑回归（LogisticRegression）**
   * 初始化逻辑回归模型，设置随机种子为 42。
   * 定义参数网格param\_grid\_lr，对正则化参数C进行调优，取值为 [0.1, 1, 10]。
   * 同样使用GridSearchCV进行交叉验证和参数调优。

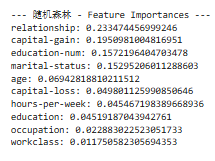
### **（三）模型评估**

1. **准确率（Accuracy）**
   * 随机森林分类器在测试集上的准确率为 0.8603746545193981。
   * 逻辑回归的准确率为 0.8248541304125294。
2. **混淆矩阵（Confusion Matrix）**
   * 随机森林分类器的混淆矩阵为：latexmath，表示模型对不同类别样本的分类情况。
   * 逻辑回归的混淆矩阵为：latexmath。
3. **分类报告（Classification Report）**
   * 随机森林分类器的分类报告显示，
     + 类别 0 的精确率为 0.87，召回率为 0.96，F1 分数为 0.91；
     + 类别 1 的精确率为 0.80，召回率为 0.56，F1 分数为 0.66。
   * 逻辑回归的分类报告中，
     + 类别 0 的精确率为 0.85，召回率为 0.94，F1 分数为 0.89；
     + 类别 1 的精确率为 0.71，召回率为 0.46，F1 分数为 0.56。
4. **ROC 曲线与 AUC 值**
   * 绘制了各模型的 ROC 曲线，随机森林分类器的 AUC 值为 0.91，逻辑回归的 AUC 值为 0.86AUC 值越接近 1，模型性能越好，说明随机森林分类器在区分正负样本方面表现较好。

|  |  |
| --- | --- |
| Y7RVS3Y5ABQAK | BIMFU3Y5ADQEQ |

随机森林——ROC曲线 逻辑回归——ROC曲线

### **（四）特征重要性分析**



通过随机森林分类器计算特征重要性，发现relationship、capital-gain、education-num等特征对模型预测较为重要。例如，relationship的特征重要性为 0.233474456999246，capital-gain的特征重要性为 0.1950981004816951。

### **（五）决策边界可视化**

以relationship和capital-gain为特征，绘制了随机森林分类器、逻辑回归和支持向量机的决策边界，直观展示了模型对不同类别样本的划分情况

|  |  |
| --- | --- |
| SFXVY3Y5ACQHA | F6JFY3Y5ADQBW |

随机森林——决策边界 逻辑回归——决策边界

## **八、回归任务 ——预测education**

### **（一）确定特征与目标**



通过特征编码，我发现教育程度和受教育年份有较大相关性，所以在后续的回归模型我选择**education列**作为目标

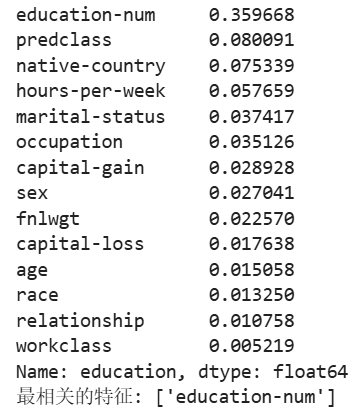
### **（二）模型选择与调优**

1. **线性回归（LinearRegression）**
   * 直接使用LinearRegression模型进行训练，未进行参数调优。
2. **多项式回归（PolynomialRegression）**
   * 先使用PolynomialFeatures对特征进行多项式转换，然后与LinearRegression结合构建多项式回归模型，尝试了不同的多项式次数。
   * 通过GridSearchCV对多项式次数进行调优，寻找最优的多项式回归模型。
3. **随机森林（RandomForestRegressor）**
   * 初始化随机森林回归器（RandomForestRegressor），设置随机种子为 42，通过RandomizedSearchCV对max\_depth、n\_estimators、min\_samples\_split和min\_samples\_leaf等参数进行随机搜索调优。

### **（三）模型评估**

1. **均方误差（MSE）**
   * 线性回归的均方误差为 12.905600663049183。
   * 多项式回归（最佳次数为 3）的均方误差为 0.36385920456646637。
   * 决策树回归（最优参数下）的均方误差为 0.0。
2. **R2 分数（R2 Score）**
   * 线性回归的 R2 分数为 0.12633837334608056。
   * 多项式回归（最佳次数为 20）的 R2 分数为 0.9753680721390434。
   * 随机森林回归（最优参数下）的 R2 分数为 1.0。R2 分数越接近 1，模型拟合效果越好，随机森林回归在该数据集上的拟合效果相对较好。

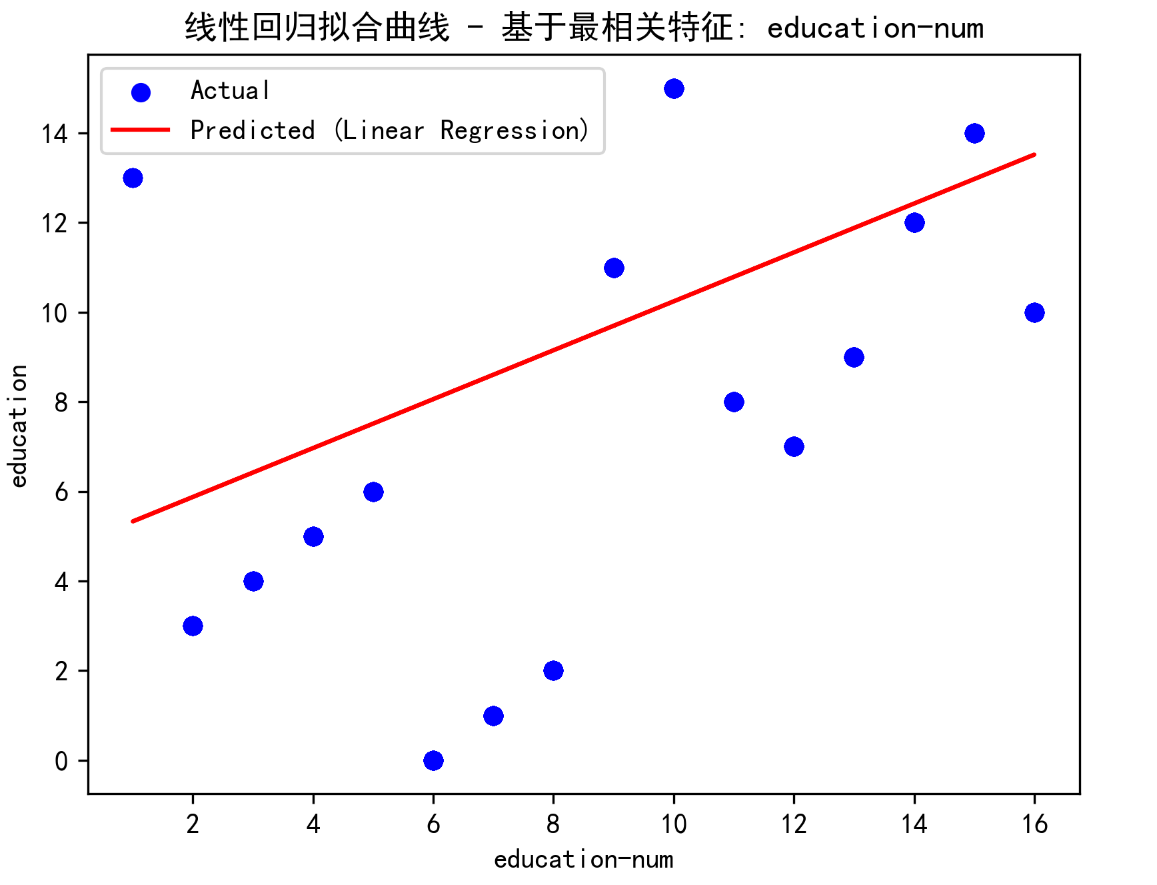
### **（四）相关性分析**

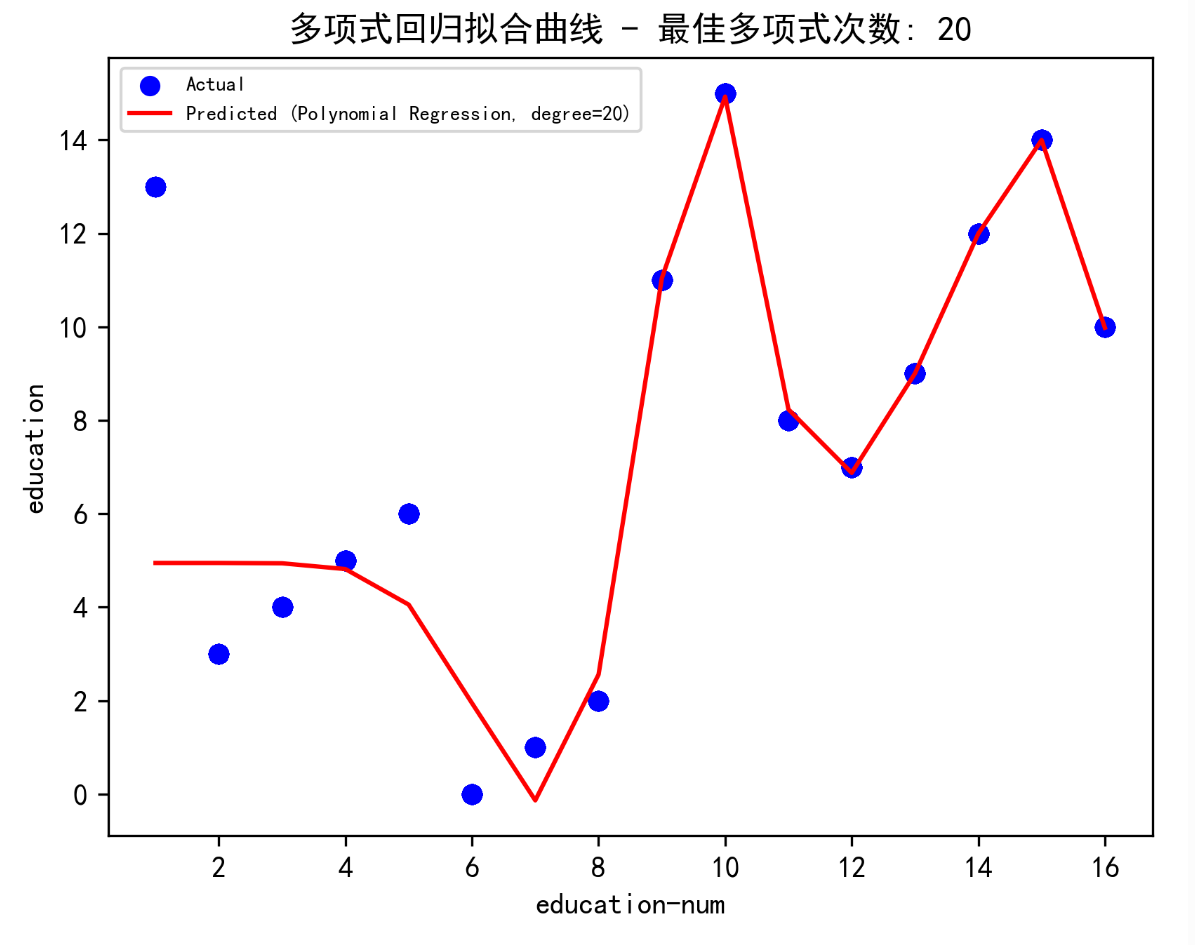


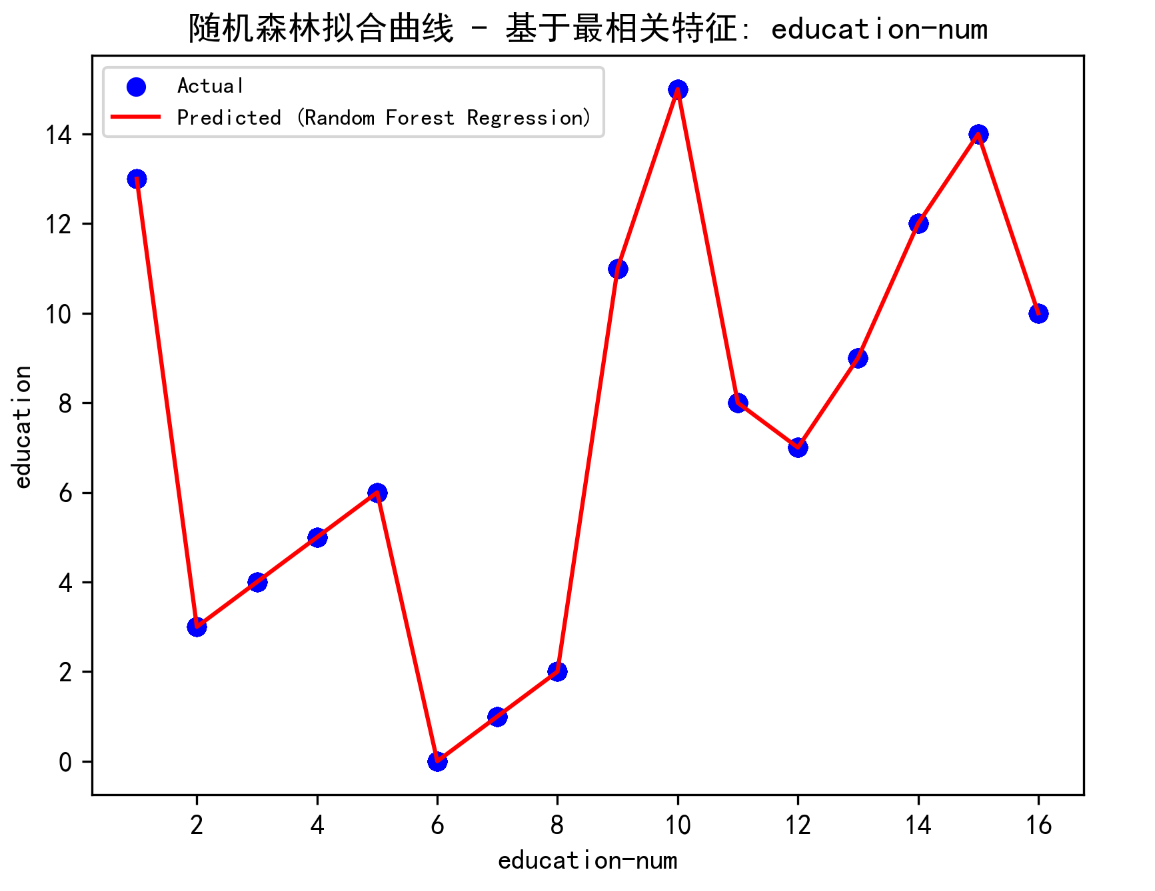
计算特征之间的相关性，发现education和education-num之间的相关性较高，相关系数为 0.359668。

### **（五）预测结果可视化**

绘制了线性回归、多项式回归和随机森林回归的预测结果与实际值的散点图，直观展示了模型的拟合效果。







## **九、**总结

1. **模型性能对比**
   * 在分类任务中，随机森林分类器表现最佳，具有较高的准确率、较好的召回率和 F1 分数，以及较高的 AUC 值，其决策边界能够较好地区分不同类别样本。逻辑回归和支持向量机也有一定的性能表现，但相对随机森林分类器稍逊一筹。
   * 在使用fnlwgt作为权重列后，随机森林和逻辑回归，训练结果，并没有明显变化，极小的提升准确率
   * 在回归任务中，随机森林回归在均方误差和 R2 分数方面表现较好，能够较好地拟合数据。多项式回归在合适的次数下也能取得较好的效果，而线性回归的拟合效果相对较弱。
2. **数据预处理的重要性**
   * 数据预处理步骤如缺失值处理、数据转换、特征编码和标准化等对模型性能有重要影响。合理的预处理可以提高模型的准确性和稳定性，使模型能够更好地学习数据中的模式。
3. **调优策略的影响**
   * 通过网格搜索（GridSearchCV）和随机搜索（RandomizedSearchCV）对模型参数进行调优，可以显著提高模型性能。不同模型对参数的敏感度不同，合理选择调优参数范围和方法能够找到最优的模型配置。
4. **特征工程的作用**
   * 特征重要性分析有助于理解数据中不同特征对模型预测的贡献程度，可指导特征选择和特征工程。在回归任务中，相关性分析可以发现特征之间的关系，避免多重共线性等问题，从而优化模型。
5. **可视化的帮助**
   * 可视化技术如 ROC 曲线、决策边界可视化、拟合曲线等有助于直观理解模型性能和数据特征之间的关系，能够辅助模型评估和调优过程，使结果更易于解释和分析。

## **十、参考文献**

[1] scikit-learn 官方文档. <https://scikit-learn.org/stable/documentation.html>

## 十一、感想

### **（一）困难**

1. **数据预处理的复杂性**
   * **缺失值处理的纠结**：在面对数据集时，发现多个特征存在缺失值，如工作类别、职业等。对于这些缺失值，起初尝试了多种填充方法，包括均值填充、众数填充等，但都存在一定问题。均值填充可能会使数据偏向平均值，影响数据的真实性；删除缺失值记录，会影响模型泛化能力。但缺失值数量少，所以最终选择众数填充。
2. **模型选择与调优的挑战**
   * **模型选择**：面对众多的机器学习模型，如逻辑回归、支持向量机、随机森林等，不知道该从何下手。每种模型都有其适用场景和优缺点，而数据集的特点又不是那么容易把握，导致在模型选择上犹豫不决，浪费了不少时间去尝试不同的模型，却难以快速找到最适合的那个。
   * **参数调优**：确定了模型后，参数调优又是一个巨大的挑战。例如，随机森林模型中的树的数量、最大深度、内部节点分裂所需最小样本数等参数，通过网格搜索等方法进行调优时，计算量非常大，也是换了好几列数据，而且经常陷入局部最优解，很难找到全局最优的参数组合，使得模型性能无法达到理想状态。
3. **结果评估与分析的困惑**
   * **评估指标的理解与应用**：课程设计中涉及到多种评估指标，如准确率、混淆矩阵、分类报告、ROC 曲线和 AUC 等。刚开始对这些指标的含义和计算方法理解不深，不知道如何根据这些指标来准确评估模型的性能。例如，在类别不平衡的情况下，准确率并不能很好地反映模型的实际效果，而混淆矩阵和分类报告中的精确率、召回率等指标又需要综合考虑，这让我在评估模型时感到很困惑。
   * **结果分析与改进的困难**：当模型评估结果不理想时，很难准确判断问题出在哪里。是数据预处理不够好，还是模型选择不合适，亦或是参数设置有问题？例如，模型出现过拟合或欠拟合现象时，不知道该从哪个方面入手去改进，是增加数据量、调整模型复杂度还是优化特征工程，需要不断地尝试和摸索，这个过程非常耗时且容易让人感到挫败。

### **（二）收获**

1. **知识与技能的提升**
   * **深入理解机器学习算法**：通过实际操作不同的机器学习模型，对它们的原理、工作机制以及适用场景有了更深刻的认识。不再只是停留在理论层面，而是能够根据数据的特点和实际问题，选择合适的模型并进行相应的调整。例如，明白了逻辑回归在处理线性可分问题上的优势，以及随机森林在处理复杂数据和高维数据时的强大能力，还有支持向量机在处理小样本和非线性数据时的独特之处。
   * **熟练掌握数据处理技巧**：学会了运用各种数据预处理技术，包括缺失值处理（如删除、填充等方法）、异常值处理（如基于统计方法的检测和处理）、特征编码（如独热编码）和数据标准化（如 标准化等）、流水线（pipline）。这些技能让我能够更好地准备数据，为模型训练提供高质量的输入，提高了模型的准确性和稳定性。
   * **编程能力的增强**：在课程设计中，大量使用 Python 及其相关的机器学习库（如 pandas、numpy、scikit-learn 等）进行代码编写和调试，编程能力得到了显著提升。能够更加熟练地运用这些工具进行数据读取、处理、模型构建、训练和评估等操作，并且学会了如何优化代码结构，提高代码的可读性和运行效率。
2. **问题解决能力的锻炼**
   * **培养了坚韧不拔的精神**：在遇到各种困难和挑战时，没有轻易放弃，而是通过查阅资料、请教老师和同学、不断尝试不同的方法等方式，逐步解决问题。这个过程锻炼了我的毅力和耐心，让我明白在面对困难时，只要坚持不懈，就一定能够找到解决办法。
   * **提高了独立思考和创新能力**：在解决问题的过程中，逐渐学会了独立思考，不再依赖现成的答案和方法。例如，在模型调优遇到瓶颈时，尝试结合多种技术和方法，如将特征工程与模型融合相结合，探索新的优化途径，培养了自己的创新思维和解决实际问题的能力。

### **（三）感想**

这次机器学习课程设计就像一场充满挑战的冒险之旅，虽然途中困难重重，但每一次克服困难都像是发现了宝藏，让我收获满满。

在面对数据预处理的复杂问题时，我曾感到无比的迷茫和无助，但正是这些困难促使我不断地学习和探索，让我逐渐掌握了数据处理的技巧和方法，也让我明白了数据质量对模型的重要性。就像盖房子需要坚实的地基一样，高质量的数据是构建优秀模型的基础。

模型选择和调优的过程让我深刻体会到了机器学习的魅力和复杂性。每一个模型都像是一个神秘的黑匣子，需要我们去不断地尝试和探索，才能找到打开它的正确方式。这个过程虽然艰辛，但当看到模型在不断调整后性能逐渐提升，那种成就感是无法言喻的。它让我明白，在机器学习的世界里，没有一蹴而就的成功，只有不断地尝试、调整和优化，才能找到最适合的解决方案。

结果评估与分析则让我学会了用更加客观和全面的眼光去看待问题。那些看似简单的评估指标背后，蕴含着丰富的信息，需要我们仔细地去解读和分析。通过对模型结果的评估，我不仅能够了解模型的优缺点，还能从中发现数据的潜在规律和问题，为进一步的改进提供方向。这让我意识到，在机器学习中，不仅仅是要构建出一个模型，更重要的是要能够理解和解释模型的结果，让模型真正为我们所用。

最可惜的可能是因为时间关系，没法进行其他集成学习方法，之后的更深入有机会可能还会再回来尝试优化。

通过这次课程设计，我对机器学习的兴趣更加浓厚了，也更加坚定了在这个领域继续深入学习和探索的决心。我相信，这次经历所积累的知识和经验，将成为我未来学习和工作的宝贵财富，激励我在机器学习的道路上不断前行，迎接更多的挑战和机遇。

这次课程设计是一次难忘的经历，它让我在知识、技能、思维等方面都得到了极大的锻炼和提升，我将永远铭记这段宝贵的经历，并将其转化为前进的动力。

**谢谢老师教导，下学期见!**

## **十二、附录**

**代码清单**

* + 以下是本课程设计中使用的 Python 代码：

1. import pandas as pd
2. import numpy as np
3. from sklearn.model\_selection import train\_test\_split
4. from sklearn.preprocessing import StandardScaler, OrdinalEncoder, PolynomialFeatures
5. from sklearn.ensemble import RandomForestClassifier
6. from sklearn.linear\_model import LogisticRegression
7. from sklearn.svm import SVC
8. from sklearn.model\_selection import GridSearchCV
9. from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc
10. import matplotlib.pyplot as plt
11. from sklearn.linear\_model import LinearRegression, Ridge, Lasso
12. from sklearn.metrics import mean\_squared\_error, r2\_score
13. from sklearn.ensemble import RandomForestRegressor
14. # 1. 读取数据
15. df = pd.read\_csv('adult.csv', header=0, low\_memory=False)
16. # 2. 查看数据基本信息
17. print("数据前5行：")
18. print(df.head())
19. print("\n数据的基本信息：")
20. print(df.info())
21. # 将age列转换为数字类型(存在)
22. df['age'] = pd.to\_numeric(df['age'], errors='coerce')
23. df.dropna(inplace = True)
24. # 对于predclass列，先进行映射转换（假设<=50K映射为0，>50K映射为1）
25. df['predclass'] = df['predclass'].map({' <=50K': 0, ' >50K': 1,' <=50K.': 0, ' >50K.': 1})
26. import numpy as np
27. # 删掉Cross validator行
28. df.dropna(inplace=True)
29. # 将 "?" 替换成空值 np.nan
30. df.replace(' ?', np.nan, inplace=True)
31. # 查看替换后缺失情况
32. print("\n替换 ‘?’ 后缺失值统计：")
33. print(df.isnull().sum())
34. # 使用众数填充缺失值
35. trans = {'workclass' : df['workclass'].mode()[0], 'occupation' : df['occupation'].mode()[0], 'native-country' : df['native-country'].mode()[0]}
36. df.fillna(trans, inplace = True)
37. df.describe()
38. df.isnull().sum()
39. #查看每个特征的数据分布情况
40. df.hist(bins=50, figsize=(20,15)) # 绘制数据集 df 中数值型特征的直方图
41. df.drop(['Unnamed: 0'], axis=1, inplace=True)
42. # 特征列与目标列分离
43. X = df.drop('predclass', axis=1)
44. y = df['predclass']
45. # 特征编码（OrdinalEncoder）
46. cat\_cols = X.select\_dtypes(include=['object']).columns
47. encoder = OrdinalEncoder()
48. X[cat\_cols] = encoder.fit\_transform(X[cat\_cols])
49. # 数据标准化
50. scaler = StandardScaler()
51. X\_scaled = scaler.fit\_transform(X)
52. # 划分训练集和测试集
53. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)
54. # 定义一个函数来绘制ROC曲线
55. def plot\_roc\_curve(fpr, tpr, roc\_auc, model\_name):
56. plt.figure()
57. plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc\_auc)
58. plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
59. plt.xlim([0.0, 1.0])
60. plt.ylim([0.0, 1.05])
61. plt.xlabel('False Positive Rate')
62. plt.ylabel('True Positive Rate')
63. plt.title(f'Receiver operating characteristic - {model\_name}')
64. plt.legend(loc="lower right")
65. plt.show()
66. # 定义一个函数来绘制决策边界
67. def plot\_decision\_boundary(model, X\_train\_2d, y\_train, featureA, featureB, x\_min, x\_max, y\_min, y\_max):
68. xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 100), np.linspace(y\_min, y\_max, 100))
69. Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])
70. Z = Z.reshape(xx.shape)
71. fig, ax = plt.subplots()
72. contour = ax.contourf(xx, yy, Z, alpha=0.4, cmap='coolwarm')
73. scatter = ax.scatter(X\_train\_2d[:, 0], X\_train\_2d[:, 1], c=y\_train, cmap='coolwarm', edgecolors='k')
74. ax.set\_xlabel(featureA)
75. ax.set\_ylabel(featureB)
76. ax.set\_title(f"Decision Boundary - {type(model).\_\_name\_\_}")
77. plt.show()
78. # 随机森林分类器
79. rf\_clf = RandomForestClassifier(random\_state=42, max\_depth=10, min\_samples\_split=5, max\_features='sqrt')
80. # 参数网格搜索
81. param\_grid\_rf = {
82. 'n\_estimators': [50, 100, 200]
83. }
84. grid\_search\_rf = GridSearchCV(rf\_clf, param\_grid\_rf, cv=3, scoring='accuracy', n\_jobs=-1)
85. grid\_search\_rf.fit(X\_train, y\_train)
86. best\_rf\_clf = grid\_search\_rf.best\_estimator\_
87. y\_pred\_rf = best\_rf\_clf.predict(X\_test)
88. # 随机森林模型评价
89. print("随机森林 - 测试集准确率:", accuracy\_score(y\_test, y\_pred\_rf))
90. print("随机森林 - 混淆矩阵:\n", confusion\_matrix(y\_test, y\_pred\_rf))
91. print("随机森林 - 分类报告:\n", classification\_report(y\_test, y\_pred\_rf))
92. # 随机森林 - 计算ROC曲线相关数据
93. y\_score\_rf = best\_rf\_clf.predict\_proba(X\_test)[:, 1]
94. fpr\_rf, tpr\_rf, thresholds\_rf = roc\_curve(y\_test, y\_score\_rf)
95. roc\_auc\_rf = auc(fpr\_rf, tpr\_rf)
96. plot\_roc\_curve(fpr\_rf, tpr\_rf, roc\_auc\_rf, 'Random Forest')
97. # 随机森林 - 获取特征重要性
98. feature\_importance\_rf = best\_rf\_clf.feature\_importances\_
99. features\_rf = X.columns
100. sorted\_idx\_rf = np.argsort(feature\_importance\_rf)[::-1]
101. print("\n--- 随机森林 - Feature Importances ---")
102. for idx in sorted\_idx\_rf[:10]:
103. print(f"{features\_rf[idx]}: {feature\_importance\_rf[idx]}")
104. # 随机森林 - 选择两个最重要特征绘制决策边界
105. featureA\_rf, featureB\_rf = features\_rf[sorted\_idx\_rf[0]], features\_rf[sorted\_idx\_rf[1]]
106. X\_train\_2d\_rf = X\_train[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
107. X\_test\_2d\_rf = X\_test[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
108. x\_min\_rf, x\_max\_rf = X\_train\_2d\_rf[:, 0].min() - 1, X\_train\_2d\_rf[:, 0].max() + 1
109. y\_min\_rf, y\_max\_rf = X\_train\_2d\_rf[:, 1].min() - 1, X\_train\_2d\_rf[:, 1].max() + 1
110. rf\_clf\_2d = RandomForestClassifier(n\_estimators=100, random\_state=42, max\_depth=10, min\_samples\_split=5)
111. rf\_clf\_2d.fit(X\_train\_2d\_rf, y\_train)
112. plot\_decision\_boundary(rf\_clf\_2d, X\_train\_2d\_rf, y\_train, featureA\_rf, featureB\_rf, x\_min\_rf, x\_max\_rf, y\_min\_rf, y\_max\_rf)
113. # 逻辑回归分类器
114. lr\_clf = LogisticRegression(random\_state=42)
115. # 参数网格搜索
116. param\_grid\_lr = {
117. 'C': [0.1, 1, 10]
118. }
119. grid\_search\_lr = GridSearchCV(lr\_clf, param\_grid\_lr, cv=3, scoring='accuracy', n\_jobs=-1)
120. grid\_search\_lr.fit(X\_train, y\_train)
121. best\_lr\_clf = grid\_search\_lr.best\_estimator\_
122. y\_pred\_lr = best\_lr\_clf.predict(X\_test)
123. # 逻辑回归模型评价
124. print("逻辑回归 - 测试集准确率:", accuracy\_score(y\_test, y\_pred\_lr))
125. print("逻辑回归 - 混淆矩阵:\n", confusion\_matrix(y\_test, y\_pred\_lr))
126. print("逻辑回归 - 分类报告:\n", classification\_report(y\_test, y\_pred\_lr))
127. # 逻辑回归 - 计算ROC曲线相关数据
128. y\_score\_lr = best\_lr\_clf.predict\_proba(X\_test)[:, 1]
129. fpr\_lr, tpr\_lr, thresholds\_lr = roc\_curve(y\_test, y\_score\_lr)
130. roc\_auc\_lr = auc(fpr\_lr, tpr\_lr)
131. plot\_roc\_curve(fpr\_lr, tpr\_lr, roc\_auc\_lr, 'Logistic Regression')
132. # 逻辑回归 - 选择两个最重要特征绘制决策边界
133. X\_train\_2d\_lr = X\_train[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
134. X\_test\_2d\_lr = X\_test[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
135. x\_min\_lr, x\_max\_lr = X\_train\_2d\_lr[:, 0].min() - 1, X\_train\_2d\_lr[:, 0].max() + 1
136. y\_min\_lr, y\_max\_lr = X\_train\_2d\_lr[:, 1].min() - 1, X\_train\_2d\_lr[:, 1].max() + 1
137. lr\_clf\_2d = LogisticRegression(random\_state=42)
138. lr\_clf\_2d.fit(X\_train\_2d\_lr, y\_train)
139. plot\_decision\_boundary(lr\_clf\_2d, X\_train\_2d\_lr, y\_train, featureA\_rf, featureB\_rf, x\_min\_lr, x\_max\_lr, y\_min\_lr, y\_max\_lr)
140. # 支持向量机分类器
141. svm\_clf = SVC(random\_state=42, probability=True)
142. # 参数网格搜索
143. param\_grid\_svm = {
144. 'C': [0.1, 1, 10],
145. 'kernel': ['linear','rbf']
146. }
147. grid\_search\_svm = GridSearchCV(svm\_clf, param\_grid\_svm, cv=3, scoring='accuracy', n\_jobs=-1)
148. grid\_search\_svm.fit(X\_train, y\_train)
149. best\_svm\_clf = grid\_search\_svm.best\_estimator\_
150. y\_pred\_svm = best\_svm\_clf.predict(X\_test)
151. # 支持向量机模型评价
152. print("支持向量机 - 测试集准确率:", accuracy\_score(y\_test, y\_pred\_svm))
153. print("支持向量机 - 混淆矩阵:\n", confusion\_matrix(y\_test, y\_pred\_svm))
154. print("支持向量机 - 分类报告:\n", classification\_report(y\_test, y\_pred\_svm))
155. # 支持向量机 - 计算ROC曲线相关数据
156. y\_score\_svm = best\_svm\_clf.predict\_proba(X\_test)[:, 1]
157. fpr\_svm, tpr\_svm, thresholds\_svm = roc\_curve(y\_test, y\_score\_svm)
158. roc\_auc\_svm = auc(fpr\_svm, tpr\_svm)
159. plot\_roc\_curve(fpr\_svm, tpr\_svm, roc\_auc\_svm, 'Support Vector Machine')
160. # 支持向量机 - 选择两个最重要特征绘制决策边界
161. X\_train\_2d\_svm = X\_train[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
162. X\_test\_2d\_svm = X\_test[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
163. x\_min\_svm, x\_max\_svm = X\_train\_2d\_svm[:, 0].min() - 1, X\_train\_2d\_svm[:, 0].max() + 1
164. y\_min\_svm, y\_max\_svm = X\_train\_2d\_svm[:, 1].min() - 1, X\_train\_2d\_svm[:, 1].max() + 1
165. svm\_clf\_2d = SVC(random\_state=42, probability=True)
166. svm\_clf\_2d.fit(X\_train\_2d\_svm, y\_train)
167. plot\_decision\_boundary(svm\_clf\_2d, X\_train\_2d\_svm, y\_train, featureA\_rf, featureB\_rf, x\_min\_svm, x\_max\_svm, y\_min\_svm, y\_max\_svm)
168. # 提取特征、目标变量和权重列
169. X = df.drop(['predclass', 'fnlwgt'], axis=1)
170. y = df['predclass']
171. weights = df['fnlwgt']
172. # 特征编码（OrdinalEncoder）
173. cat\_cols = X.select\_dtypes(include=['object']).columns
174. encoder = OrdinalEncoder()
175. X[cat\_cols] = encoder.fit\_transform(X[cat\_cols])
176. # 数据标准化
177. scaler = StandardScaler()
178. X\_scaled = scaler.fit\_transform(X)
179. # 划分训练集和测试集（包含权重）
180. X\_train, X\_test, y\_train, y\_test, weights\_train, weights\_test = train\_test\_split(X\_scaled, y, weights, test\_size=0.2, random\_state=42)
181. # 随机森林分类器
182. rf\_clf = RandomForestClassifier(random\_state=42, max\_depth=10, min\_samples\_split=5, max\_features='sqrt')
183. # 参数网格搜索
184. param\_grid\_rf = {'n\_estimators': [50, 100, 200]}
185. grid\_search\_rf = GridSearchCV(rf\_clf, param\_grid\_rf, cv=3, scoring='accuracy', n\_jobs=-1)
186. grid\_search\_rf.fit(X\_train, y\_train, sample\_weight=weights\_train)
187. best\_rf\_clf = grid\_search\_rf.best\_estimator\_
188. y\_pred\_rf = best\_rf\_clf.predict(X\_test)
189. # 随机森林模型评价
190. print("随机森林 - 测试集准确率:", accuracy\_score(y\_test, y\_pred\_rf))
191. print("随机森林 - 混淆矩阵:\n", confusion\_matrix(y\_test, y\_pred\_rf))
192. print("随机森林 - 分类报告:\n", classification\_report(y\_test, y\_pred\_rf))
193. # 随机森林 - 计算ROC曲线相关数据
194. y\_score\_rf = best\_rf\_clf.predict\_proba(X\_test)[:, 1]
195. fpr\_rf, tpr\_rf, thresholds\_rf = roc\_curve(y\_test, y\_score\_rf)
196. roc\_auc\_rf = auc(fpr\_rf, tpr\_rf)
197. plot\_roc\_curve(fpr\_rf, tpr\_rf, roc\_auc\_rf, 'Random Forest')
198. # 随机森林 - 获取特征重要性
199. feature\_importance\_rf = best\_rf\_clf.feature\_importances\_
200. features\_rf = X.columns
201. sorted\_idx\_rf = np.argsort(feature\_importance\_rf)[::-1]
202. print("\n--- 随机森林 - Feature Importances ---")
203. for idx in sorted\_idx\_rf[:10]:
204. print(f"{features\_rf[idx]}: {feature\_importance\_rf[idx]}")
205. # 随机森林 - 选择两个最重要特征绘制决策边界
206. featureA\_rf, featureB\_rf = features\_rf[sorted\_idx\_rf[0]], features\_rf[sorted\_idx\_rf[1]]
207. X\_train\_2d\_rf = X\_train[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
208. X\_test\_2d\_rf = X\_test[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
209. x\_min\_rf, x\_max\_rf = X\_train\_2d\_rf[:, 0].min() - 1, X\_train\_2d\_rf[:, 0].max() + 1
210. y\_min\_rf, y\_max\_rf = X\_train\_2d\_rf[:, 1].min() - 1, X\_train\_2d\_rf[:, 1].max() + 1
211. rf\_clf\_2d = RandomForestClassifier(n\_estimators=100, random\_state=42, max\_depth=10, min\_samples\_split=5)
212. # 传递原始的权重数组
213. rf\_clf\_2d.fit(X\_train\_2d\_rf, y\_train, sample\_weight=weights\_train)
214. plot\_decision\_boundary(rf\_clf\_2d, X\_train\_2d\_rf, y\_train, featureA\_rf, featureB\_rf, x\_min\_rf, x\_max\_rf, y\_min\_rf, y\_max\_rf)
215. # 逻辑回归分类器
216. lr\_clf = LogisticRegression(random\_state=42)
217. # 参数网格搜索
218. param\_grid\_lr = {'C': [0.1, 1, 10]}
219. grid\_search\_lr = GridSearchCV(lr\_clf, param\_grid\_lr, cv=3, scoring='accuracy', n\_jobs=-1)
220. grid\_search\_lr.fit(X\_train, y\_train, sample\_weight=weights\_train)
221. best\_lr\_clf = grid\_search\_lr.best\_estimator\_
222. y\_pred\_lr = best\_lr\_clf.predict(X\_test)
223. # 逻辑回归模型评价
224. print("逻辑回归 - 测试集准确率:", accuracy\_score(y\_test, y\_pred\_lr))
225. print("逻辑回归 - 混淆矩阵:\n", confusion\_matrix(y\_test, y\_pred\_lr))
226. print("逻辑回归 - 分类报告:\n", classification\_report(y\_test, y\_pred\_lr))
227. # 逻辑回归 - 计算ROC曲线相关数据
228. y\_score\_lr = best\_lr\_clf.predict\_proba(X\_test)[:, 1]
229. fpr\_lr, tpr\_lr, thresholds\_lr = roc\_curve(y\_test, y\_score\_lr)
230. roc\_auc\_lr = auc(fpr\_lr, tpr\_lr)
231. plot\_roc\_curve(fpr\_lr, tpr\_lr, roc\_auc\_lr, 'Logistic Regression')
232. # 逻辑回归 - 选择两个最重要特征绘制决策边界
233. X\_train\_2d\_lr = X\_train[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
234. X\_test\_2d\_lr = X\_test[:, [sorted\_idx\_rf[0], sorted\_idx\_rf[1]]]
235. x\_min\_lr, x\_max\_lr = X\_train\_2d\_lr[:, 0].min() - 1, X\_train\_2d\_lr[:, 0].max() + 1
236. y\_min\_lr, y\_max\_lr = X\_train\_2d\_lr[:, 1].min() - 1, X\_train\_2d\_lr[:, 1].max() + 1
237. lr\_clf\_2d = LogisticRegression(random\_state=42)
238. # 传递原始的权重数组
239. lr\_clf\_2d.fit(X\_train\_2d\_lr, y\_train, sample\_weight=weights\_train)
240. plot\_decision\_boundary(lr\_clf\_2d, X\_train\_2d\_lr, y\_train, featureA\_rf, featureB\_rf, x\_min\_lr, x\_max\_lr, y\_min\_lr, y\_max\_lr)
241. cat\_cols = df.select\_dtypes(include=['object']).columns
242. encoder = OrdinalEncoder()
243. df[cat\_cols] = encoder.fit\_transform(df[cat\_cols])
244. corr\_matrix = df.corr()
245. corr\_matrix.head()
246. # 将结果中与目标 "education" 相关的相关性值取出，并移除自身
247. target\_corr = corr\_matrix["education"].drop("education")
248. print(target\_corr.abs().sort\_values(ascending=False))
249. # 选出绝对值相关度最高的特征名
250. top\_feature = target\_corr.abs().sort\_values(ascending=False).head(1).index.tolist()
251. print("最相关的特征:", top\_feature)
252. # 构建新的 DataFrame，便于后续建模可视化
253. X = df[top\_feature] # 最相关特征
254. y = df["education"] # 分类目标
255. # 划分训练集和测试集
256. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
257. # 线性回归模型
258. lr = LinearRegression()
259. lr.fit(X\_train, y\_train)
260. y\_pred\_lr = lr.predict(X\_test)
261. print("\n--- Linear Regression ---")
262. print("均方误差:", mean\_squared\_error(y\_test, y\_pred\_lr))
263. print("R2得分:", r2\_score(y\_test, y\_pred\_lr))
264. # 绘制拟合曲线
265. plt.scatter(X\_test, y\_test, color='blue', label='Actual')
266. plt.plot(X\_test, y\_pred\_lr, color='red', label='Predicted (Linear Regression)')
267. plt.xlabel(top\_feature[0])
268. plt.xticks()
269. plt.ylabel("education")
270. plt.title(f"线性回归拟合曲线 - 基于最相关特征: {top\_feature[0]}")
271. plt.legend()
272. plt.show()
273. from sklearn.pipeline import Pipeline
274. # 定义要搜索的多项式次数范围，可根据实际情况调整范围和步长
275. param\_grid = {'polynomialfeatures\_\_degree': np.arange(1, 30)} # 注意这里参数名的写法，用"polynomialfeatures\_\_degree" 来指定是PolynomialFeatures步骤中的degree参数
276. # 创建管道，将多项式特征生成和线性回归组合在一起
277. pipe = Pipeline([
278. ('polynomialfeatures', PolynomialFeatures()),
279. ('linearregression', LinearRegression())
280. ])
281. # 创建GridSearchCV对象，用于搜索最佳参数组合
282. grid\_search = GridSearchCV(estimator=pipe, param\_grid=param\_grid, cv=3, # 5折交叉验证，可按需调整
283. scoring='neg\_mean\_squared\_error') # 以负均方误差作为评估指标，GridSearchCV会自动寻找使其最小化的值
284. # 进行网格搜索，拟合模型找到最佳参数
285. grid\_search.fit(X\_train, y\_train)
286. # 获取最佳多项式次数
287. best\_degree = grid\_search.best\_params\_['polynomialfeatures\_\_degree']
288. # 使用最佳的多项式次数重新构建多项式特征转换和拟合线性回归模型（这里通过管道的方式更方便了）
289. pipe\_best = Pipeline([
290. ('polynomialfeatures', PolynomialFeatures(degree=best\_degree)),
291. ('linearregression', LinearRegression())
292. ])
293. pipe\_best.fit(X\_train, y\_train)
294. y\_pred\_poly\_best = pipe\_best.predict(X\_test)
295. print("\n--- Polynomial Regression with Best Degree ---")
296. print("最佳多项式次数:", best\_degree)
297. print("均方误差:", mean\_squared\_error(y\_test, y\_pred\_poly\_best))
298. print("R2得分:", r2\_score(y\_test, y\_pred\_poly\_best)))
299. # 绘制拟合曲线
300. plt.scatter(X\_test.iloc[:, 0], y\_test, color='blue', label='Actual') # 使用 X\_test 的第一列作为 x 轴数据
301. # 对X\_test进行排序，以便绘制平滑的曲线
302. sorted\_idx = np.argsort(X\_test.iloc[:, 0]) # 使用 X\_test 的第一列进行排序
303. plt.plot(X\_test.iloc[sorted\_idx, 0], y\_pred\_poly\_best[sorted\_idx], color='red', label=f'Predicted (Polynomial Regression, degree={best\_degree})')
304. plt.xlabel(top\_feature[0])
305. plt.ylabel("education")
306. plt.title(f"多项式回归拟合曲线 - 最佳多项式次数: {best\_degree}")
307. plt.legend(loc='upper right', bbox\_to\_anchor=(0.52, 1.0), fontsize='x-small') # 缩小图例并调整位置
308. plt.show()
309. from sklearn.ensemble import RandomForestRegressor
310. from sklearn.model\_selection import RandomizedSearchCV
311. import numpy as np
312. # 定义要搜索的参数分布，这里以'max\_depth'（树的最大深度）、'n\_estimators'（决策树的数量）、'min\_samples\_split'（内部节点再划分所需最小样本数）和'min\_samples\_leaf'（叶子节点最少样本数）为例，你可以根据实际情况添加或调整其他参数
313. param\_distributions = {
314. 'max\_depth': np.arange(3, 11), # 树的最大深度的取值范围，可根据数据复杂度等调整
315. 'n\_estimators': np.arange(50, 201, 10), # 决策树的数量的取值范围，步长为10
316. 'min\_samples\_split': np.arange(2, 11), # 内部节点再划分所需最小样本数的取值范围
317. 'min\_samples\_leaf': np.arange(1, 6) # 叶子节点最少样本数的取值范围
318. }
319. # 创建随机森林回归模型对象（先不指定具体参数，后续由随机搜索确定合适值）
320. dt\_reg = RandomForestRegressor(random\_state=42)
321. # 创建RandomizedSearchCV对象，设置随机搜索的相关参数
322. random\_search = RandomizedSearchCV(estimator=dt\_reg,
323. param\_distributions=param\_distributions,
324. n\_iter=20, # 随机搜索尝试的参数组合数量，可根据计算资源和时间等调整
325. cv=5, # 5折交叉验证，可按需调整
326. scoring='neg\_mean\_squared\_error', # 以负均方误差作为评估指标，RandomizedSearchCV会自动寻找使其最小化的值
327. random\_state=42) # 保证每次运行随机搜索结果可复现
328. # 进行随机搜索，拟合模型找到合适参数
329. random\_search.fit(X\_train, y\_train)
330. # 获取最佳参数组合
331. best\_params = random\_search.best\_params\_
332. print("\n--- RandomForest Regression with Best Parameters ---")
333. print("最佳参数:", best\_params)
334. # 使用最佳参数重新构建决策树回归模型并拟合数据
335. best\_dt\_reg = RandomForestRegressor(\*\*best\_params, random\_state=42)
336. best\_dt\_reg.fit(X\_train, y\_train)
337. y\_pred\_dt\_best = best\_dt\_reg.predict(X\_test)
338. print("均方误差:", mean\_squared\_error(y\_test, y\_pred\_dt\_best))
339. print("R2得分:", r2\_score(y\_test, y\_pred\_dt\_best))
340. # 绘制拟合曲线
341. plt.scatter(X\_test.iloc[:, 0], y\_test, color='blue', label='Actual') # 使用 X\_test 的第一列作为 x 轴数据
342. # 对X\_test进行排序，以便绘制平滑的曲线
343. sorted\_idx = np.argsort(X\_test.iloc[:, 0]) # 使用 X\_test 的第一列进行排序
344. plt.plot(X\_test.iloc[sorted\_idx, 0], y\_pred\_dt\_best[sorted\_idx], color='red', label='Predicted (RandomForest Regression)')
345. plt.xlabel(top\_feature[0])
346. plt.ylabel("education")
347. plt.title(f"随机森林拟合曲线 - 基于最相关特征: {top\_feature[0]}")
348. plt.legend(loc='upper left', fontsize='small')
349. plt.show()