# 1. 导入必要的库

import pandas as pd # 用于数据操作和分析

import numpy as np # 用于数值计算

import matplotlib.pyplot as plt # 用于数据可视化

import seaborn as sns # 用于高级数据可视化

from sklearn.model\_selection import train\_test\_split, GridSearchCV # 用于数据集划分和参数调优

from sklearn.preprocessing import StandardScaler # 用于特征标准化

from sklearn.linear\_model import LinearRegression # 线性回归模型

from sklearn.tree import DecisionTreeRegressor # 决策树回归模型

from sklearn.svm import SVR # 支持向量机回归模型

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error # 用于评估模型性能

# 2. 加载数据集

data = pd.read\_csv("student-mat.csv") # 读取CSV文件

raws, cols = data.shape # 获取数据集的行数和列数

print(f"该数据集有 {raws} 个样本") # 打印样本数量

print(f"该数据集有 {cols-1} 个特征") # 打印特征数量（不包括目标变量）

# 3. 探索性数据分析（EDA）

print("数据集的前几行：")

print(data.head()) # 打印数据集的前几行

print("\n数据集的基本信息：")

print(data.info()) # 打印数据集的基本信息

print("\n数据集的描述统计信息：")

print(data.describe()) # 打印数据集的描述统计信息

# 4. 确定离散特征和数值特征

discrete\_features = data.select\_dtypes(include=['object']).columns # 选择离散特征

numerical\_features = data.select\_dtypes(include=['int64', 'float64']).columns # 选择数值特征

# 打印离散特征和数值特征

print("\n离散特征：", discrete\_features)

print("\n数值特征：", numerical\_features)

# 对离散特征进行独热编码

data = pd.get\_dummies(data, columns=discrete\_features) # 将离散特征转化为独热编码

# 将离散的数学成绩（G1, G2, G3）转化为有序类别

def convert\_to\_continuous(x):

x = int(x)

if x < 5:

return 1 # 差

elif 5 <= x < 10:

return 2 # 中

elif 10 <= x < 15:

return 3 # 好

else:

return 4 # 优

data['G1'] = data['G1'].apply(convert\_to\_continuous) # 转化 G1

data['G2'] = data['G2'].apply(convert\_to\_continuous) # 转化 G2

data['G3'] = data['G3'].apply(convert\_to\_continuous) # 转化 G3

# 确保数据集中只包含数值型特征

numerical\_features = data.select\_dtypes(include=['int64', 'float64']).columns

# 第一次做出箱型图发现有离群值，因此先替换离群值

def replace\_outliers(series):

Q1 = series.quantile(0.25)

Q3 = series.quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

return series.apply(lambda x: upper\_bound if x > upper\_bound else (lower\_bound if x < lower\_bound else x))

# 替换离群值并绘制箱型图

data\_no\_outliers = data[numerical\_features].apply(replace\_outliers) # 替换离群值

plt.figure(figsize=(10, 6))

sns.boxplot(data=data\_no\_outliers) # 绘制箱型图

plt.title("Boxplot of Student Performance Features without Outliers")

plt.xticks(rotation=45)

plt.show()

# 计算特征与目标变量（G3）之间的相关性

correlation = data.corr()['G3'].drop('G3')

# 排序特征重要性

sorted\_features = correlation.abs().sort\_values(ascending=False)

# 绘制特征重要性图

plt.figure(figsize=(12, 8))

sns.barplot(x=sorted\_features.values, y=sorted\_features.index, hue=sorted\_features.index, palette="viridis", legend=False)

plt.title("Feature Importance for G3")

plt.xlabel("Absolute Correlation")

plt.ylabel("Features")

plt.show()

# 相关性前四的特征

top\_features = sorted\_features.head(4).index

# 选取特征与目标变量（G3）之间的相关性

top\_correlation = data[top\_features].corr()

# 绘制热力图

plt.figure(figsize=(8, 6))

sns.heatmap(top\_correlation, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap of Top 4 Features with G3')

plt.show()

# 特征选择：选择相关性较高的特征，即前20个特征

top\_features = sorted\_features.head(20).index

# 选择特征和目标变量

X = data[top\_features] # 特征变量

y = data['G3'] # 目标变量

# 标准化选择特征可以确保模型在训练过程中更快地收敛，并且不会因为特征的尺度不同而对模型产生偏见

scaler = StandardScaler() # 创建一个标准化器对象

X\_scaled = scaler.fit\_transform(X) # 对特征进行标准化

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # 划分数据集

# 查看划分后的训练集和测试集大小

print("\n训练集大小:", X\_train.shape, y\_train.shape)

print("测试集大小:", X\_test.shape, y\_test.shape)

# 定义模型字典

models = {

"Linear Regression": LinearRegression(),

"Decision Tree": DecisionTreeRegressor(),

"Support Vector Machine": SVR()

}

pre\_optimization\_scores = {} # 存储调优前的性能指标

post\_optimization\_scores = {} # 存储调优后的性能指标

# 记录调优前的性能指标

plt.figure(figsize=(10, 6))

for name, model in models.items():

model.fit(X\_train, y\_train) # 训练模型

y\_pred\_pre = model.predict(X\_test) # 预测测试集

mse\_pre = mean\_squared\_error(y\_test, y\_pred\_pre) # 计算均方误差

r2\_pre = r2\_score(y\_test, y\_pred\_pre) # 计算R2评分

mae\_pre = mean\_absolute\_error(y\_test, y\_pred\_pre) # 计算平均绝对误差

pre\_optimization\_scores[name] = {"MSE": mse\_pre, "R2": r2\_pre, "MAE": mae\_pre}

# 可视化调优前性能

plt.bar(name, r2\_pre, color='red', alpha=0.5)

plt.title("Pre-Optimization Model Performance (R2 Score)")

plt.xlabel("Models")

plt.ylabel("R2 Score")

plt.ylim(0, 1)

plt.show()

# 输出每个模型的调优前性能指标

for name, scores in pre\_optimization\_scores.items():

print(f"Pre-Optimization Scores for {name}:")

print(f"MSE: {scores['MSE']}, R2: {scores['R2']}, MAE: {scores['MAE']}")

# 模型参数调优

best\_models = {}

for name, model in models.items():

if name == "Linear Regression":

best\_model = model # 线性回归模型不需要调优

elif name == "Decision Tree":

param\_grid = {'max\_depth': [None, 5, 10, 15]} # 决策树参数网格

grid\_search = GridSearchCV(model, param\_grid, cv=5, scoring='neg\_mean\_squared\_error') # 网格搜索

grid\_search.fit(X\_train, y\_train) # 训练模型

best\_model = grid\_search.best\_estimator\_ # 获取最佳模型

else: # SVM

param\_grid = {'C': [0.1, 1, 10, 100], 'epsilon': [0.01, 0.1, 1]} # SVM参数网格

grid\_search = GridSearchCV(model, param\_grid, cv=5, scoring='neg\_mean\_squared\_error') # 网格搜索

grid\_search.fit(X\_train, y\_train) # 训练模型

best\_model = grid\_search.best\_estimator\_ # 获取最佳模型

best\_models[name] = best\_model # 保存最佳模型

# 记录调优后的性能指标

y\_pred\_post = best\_model.predict(X\_test) # 预测测试集

mse\_post = mean\_squared\_error(y\_test, y\_pred\_post) # 计算均方误差

r2\_post = r2\_score(y\_test, y\_pred\_post) # 计算R2评分

mae\_post = mean\_absolute\_error(y\_test, y\_pred\_post) # 计算平均绝对误差

post\_optimization\_scores[name] = {"MSE": mse\_post, "R2": r2\_post, "MAE": mae\_post}

# 输出每个模型的调优后性能指标

for name, scores in post\_optimization\_scores.items():

print(f"\nPost-Optimization Scores for {name}:")

print(f"MSE: {scores['MSE']}, R2: {scores['R2']}, MAE: {scores['MAE']}")

# 可视化调优后的性能

plt.figure(figsize=(10, 6))

post\_optimization\_r2\_scores = [scores['R2'] for scores in post\_optimization\_scores.values()]

plt.bar(list(best\_models.keys()), post\_optimization\_r2\_scores, color='blue')

plt.title("Post-Optimization Model Performance (R2 Score)")

plt.xlabel("Models")

plt.ylabel("R2 Score")

plt.ylim(0, 1)

plt.show()

# 选出具有最高 R2 分数的模型

best\_model\_names = [name for name, scores in post\_optimization\_scores.items() if scores['R2'] == max(post\_optimization\_r2\_scores)]

print("Best Model(s):", best\_model\_names)

1.标准化通常只应用于特征（自变量），而不是目标变量（因变量）。这是因为目标变量通常是我们想要预测的变量，不应该改变它的分布或范围，而应该保持它原来的形式。目标变量的变化是由特征的变化引起的。

标准化特征可以确保模型在训练过程中更快地收敛，并且不会因为特征的尺度不同而对模型产生偏见。然而，对目标变量进行标准化可能会导致模型预测的解释变得困难，因为它们的实际含义会发生改变。因此，通常只对特征进行标准化，而保持目标变量不变。

2.将离散的数学成绩转化为有序的类别对于回归模型有几个好处：

减少数据的噪音影响： 原始的数学成绩可能受到许多因素的影响，比如个人努力程度、家庭背景等，导致数据中存在大量的噪音。将成绩转化为有序的类别可以降低这种噪音对模型的影响，使模型更专注于类别之间的差异。

提高模型的可解释性： 将数学成绩分为不同的类别可以更直观地解释模型的预测结果。例如，模型预测某个学生的成绩为“好”或“优”，比预测一个具体的数值更容易理解和解释。

适应业务场景需求： 在某些业务场景中，人们更关心学生的成绩等级而不是具体的数值。例如，在教育领域，学生的成绩通常被划分为差、中、好、优等级，而不是单纯地关注数值的大小。

降低过拟合的风险： 将数学成绩转化为有序的类别可以减少模型过拟合的风险。过拟合可能会导致模型在训练数据上表现良好，但在未知数据上表现不佳。通过将数据转化为有限的类别，可以减少模型学习过多的训练数据中的细节，从而降低过拟合的风险。

总的来说，将离散的数学成绩转化为有序的类别可以简化问题，提高模型的泛化能力，并使模型更符合实际应用场景的需求。

3.在回归问题中，通常使用的评估指标包括均方误差 (Mean Squared Error, MSE)、平均绝对误差 (Mean Absolute Error, MAE)、解释方差得分 (Explained Variance Score)、R2 分数等。这些指标用于衡量模型的预测值与实际值之间的差异程度。

均方误差 (MSE) 衡量预测值与实际值之间的平方误差的平均值，是最常用的回归模型评估指标之一。MSE 越小，表示模型的预测效果越好。

平均绝对误差 (MAE) 衡量预测值与实际值之间的绝对误差的平均值，也是常用的回归模型评估指标之一。

解释方差得分 衡量模型对数据方差的解释程度，取值范围在 0 到 1 之间，值越接近 1 表示模型对数据的解释能力越好。

R2 分数 (R-squared) 衡量模型对数据方差的解释程度，取值范围在负无穷到 1 之间，值越接近 1 表示模型对数据的解释能力越好。

4.将预测成绩问题转换为有序类别后，可以选择使用SVR（Support Vector Regression）或者使用分类器，如SVC（Support Vector Classifier）来解决问题。这取决于对于预测结果的理解和任务的目标。

SVR（Support Vector Regression）:

SVR是专门用于解决回归问题的支持向量机模型。

它在拟合过程中会考虑样本的有序性，尝试找到一个回归函数来最小化预测值与真实值之间的差距。

适用于连续型的预测结果，可以直接输出数值型的预测结果。

SVC（Support Vector Classifier）:

SVC是用于解决分类问题的支持向量机模型。

将预测成绩问题转换为有序类别，可以使用分类器来预测每个类别的概率或者标签。

适用于将预测结果转换为离散的类别，例如将成绩分为高、中、低三个等级。

因此，如果希望直接预测数值型的成绩结果，可以选择SVR。如果更关心成绩的等级分类，可以选择SVC。

5.对于决策树模型，通过定义了决策树的最大深度参数max\_depth的取值范围进行了调优。

对于支持向量机模型，通过定义了参数C和epsilon的取值范围进行了调优。

使用了网格搜索交叉验证（Grid Search Cross Validation），评分标准为负均方误差，通过在训练集上交叉验证找到最佳参数组合。