实验八：$scikit-learn$ 工具包

陈兴浩，王为，王亦辉

2024 年 11 月 19 日

1. 实验目标

利用 Python 工具包编写数据分析建模程序, 掌握常用工具包的使用方法及新工具包的自主学习方法, 能够结合具体问题选择合适的工工具包, 能够通过团队协作、合理分工高效解决问题。

2. 作业描述

根据第 9 章人工智能基础课件, 自学 scikit-learn 工具包; 并以给出的案例为基础, 尝试修改相关代码, 解决其他机器学习问题。例如, 利用前面给出的 UCI 机器学习存储库提供的公开数据集, 根据一个人的生活习惯预测其 BMI 指数。

3. 数据集简介

本数据集用于根据个体的饮食习惯和身体状况估算肥胖水平，包含来自墨西哥、秘鲁和哥伦比亚的样本数据。该数据集包含 2111 条记录和 17 个属性，标签变量为 `NObesity`（肥胖水平），可用于分类、回归和聚类任务。肥胖等级的标签包括：体重不足（Insufficient Weight）、正常体重（Normal Weight）、超重一级（Overweight Level I）、超重二级（Overweight Level II）、肥胖一级（Obesity Type I）、肥胖二级（Obesity Type II）以及肥胖三级（Obesity Type III）。

- **数据样本**：数据集包含了多个健康和生活方式的属性，例如性别（Gender）、年龄（Age）、身高（Height）、体重（Weight）、家族肥胖史（family_hist）、饮食频率（FAVC, FCVC）、每日用水量（CH2O）、运动频率（FAF）等。
- **生成方式**：数据的 77% 通过 Weka 工具和 SMOTE 过滤器合成生成，23% 数据通过网络平台直接从用户收集。
- **特征概览**：
  - **性别（Gender）**：男性或女性。
  - **年龄（Age）**：个体的年龄（整数）。
  - **身高（Height）和体重（Weight）**：身高（米）和体重（千克）。
  - **家族肥胖史（family_hist）**：是否有家族肥胖史。
  - **饮食习惯（FAVC, FCVC）**：食物摄入频率（如是否频繁进食高热量食物等）。
  - **其他生活习惯（CH2O，FAF，TUE，SMOKE，CALC，MTRANS）**：包括每日饮水量、运动频率、屏幕时间、吸烟习惯、饮酒频率以及主要交通方式。

4. 代码分析（以随机森林为例）

本节将对使用随机森林分类器的代码进行详细分析，以便理解数据处理、模型训练、预测以及评估步骤。

4.1. 数据加载与预处理

- **加载数据集**：使用 `pandas` 库加载数据集 `01.csv`。特征集（`X`）从数据集中剔除目标变量 `NObeyesdad`，而目标变量（`y`）则取 `NObeyesdad` 列。

- **类别编码**：由于特征中包含字符串类型的数据，使用 `pd.get_dummies()` 方法对所有字符串类型的列进行独热编码。此外，对目标变量 `y` 使用 `LabelEncoder` 将其转换为数值型编码，以便于模型的训练。
- **数据集划分**：使用 `train_test_split` 将数据划分为训练集和测试集，其中测试集占 20%。

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# 加载数据集
file_path = r".\01.csv"
df = pd.read_csv(file_path)

X = df.drop('NObeyesdad', axis=1)  # 特征
y = df['NObeyesdad']  # 目标变量

# 独热编码和标签编码
X = pd.get_dummies(X, columns=X.select_dtypes(
    include=['object']).columns)
le = LabelEncoder()
y = le.fit_transform(y)

# 数据集划分
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

4.2. 模型训练与预测

- **模型选择**：选择随机森林分类器（`RandomForestClassifier`）作为本例的机器学习模型。随机森林是一种基于决策树的集成模型，通过多个决策树的投票来减少单一模型的偏差和方差。
- **模型训练**：使用训练集 `X_train` 和 `y_train` 进行模型训练，设置 `n_estimators=100` 来构建 100 棵树，`random_state=42` 保证结果的可重复性。
- **模型预测**：训练完成后，使用测试集 `X_test` 进行预测，得到预测结果 `y_pred`。

```python
from sklearn.ensemble import RandomForestClassifier

# 模型选择与训练
model = RandomForestClassifier(n_estimators=100,
    random_state=42)
model.fit(X_train, y_train)

# 预测
y_pred = model.predict(X_test)
```

4.3. 模型评估

- **准确率**：使用 `accuracy_score` 计算模型在测试集上的分类准确率，并输出结果。
- **分类报告与混淆矩阵**：使用 `classification_report` 生成分类评估报告，报告包括精度（Precision）、召回率（Recall）、F1-score 等评估指标。同时，使用 `ConfusionMatrixDisplay.from_predictions` 绘制混淆矩阵，以便分析分类器的性能。
- **均方误差 (MSE)**：通过 `mean_squared_error` 计算模型在测试集上的均方误差（MSE），用于衡量模型预测值与实际值之间的平均平方差。

```python
from sklearn import metrics
from sklearn.metrics import accuracy_score,
    mean_squared_error

# 模型评估
accuracy = accuracy_score(y_test, y_pred)
print("分类准确率:", accuracy)

# 转换回原始标签
y_test_original = le.inverse_transform(y_test)
y_pred_original = le.inverse_transform(y_pred)

# 分类报告
print(metrics.classification_report(y_test_original,
     y_pred_original, digits=4))

# 混淆矩阵
metrics.ConfusionMatrixDisplay.from_predictions(
    y_test, y_pred)
```

```
17
18  # 计算并输出MSE
19  mse = mean_squared_error(y_test, y_pred)
20  print("测试集上的MSE指标:%.4f" % mse)
```

此代码通过对数据进行预处理、模型训练、预测和评估，完整展示了随机森林分类器的应用流程。

4.4. 代码总结

本代码展示了从数据加载、预处理、模型训练到评估的完整机器学习流程，使用随机森林分类器对肥胖水平进行预测。该方法在分类准确率上表现良好，且通过混淆矩阵与分类报告可以深入了解模型的分类效果。均方误差（MSE）也为进一步评估模型性能提供了参考。

5. 实验结果

通过模型模拟与最终测试集的误差计算，我们依次得到了如下结果，图片展现的是预测结果和真实结果的热力图，右对角线上所表示的数值体现了模型预测的准确性，其余格子上的数值是预测失败的值。我们同时给出准确率和 MSE。

5.1. 随机森林

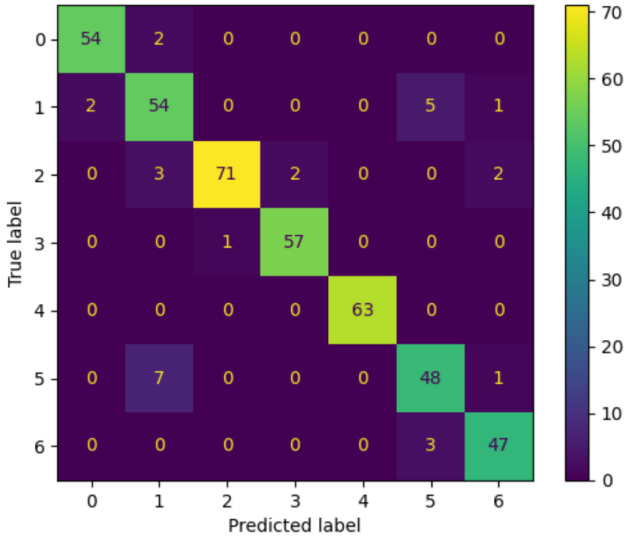

Figure 1: 随机森林

- mse：0.6217
- 准确率：93.14%

5.2. k-近邻

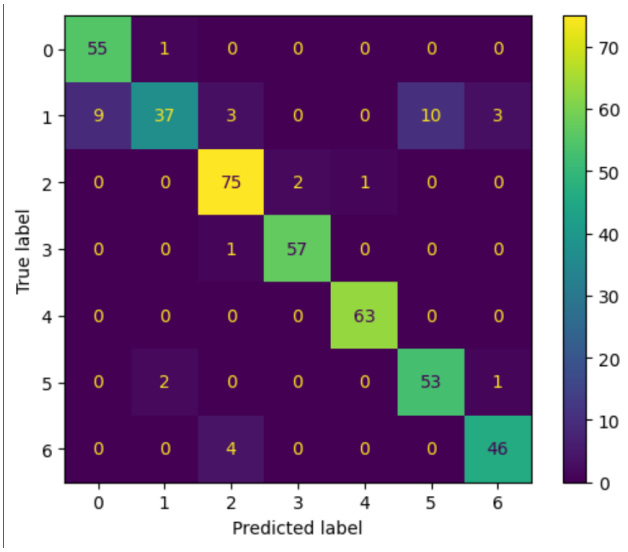

Figure 2: k-近邻

- mse：0.8322
- 准确率：91.25%

5.3. 朴素贝叶斯

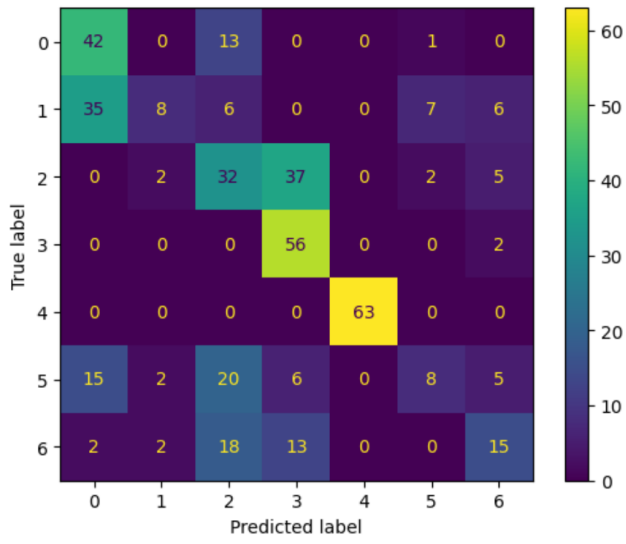

Figure 3: 朴素贝叶斯

- mse：3.9669
- 准确率：52.96%

5.4. 梯度提升树

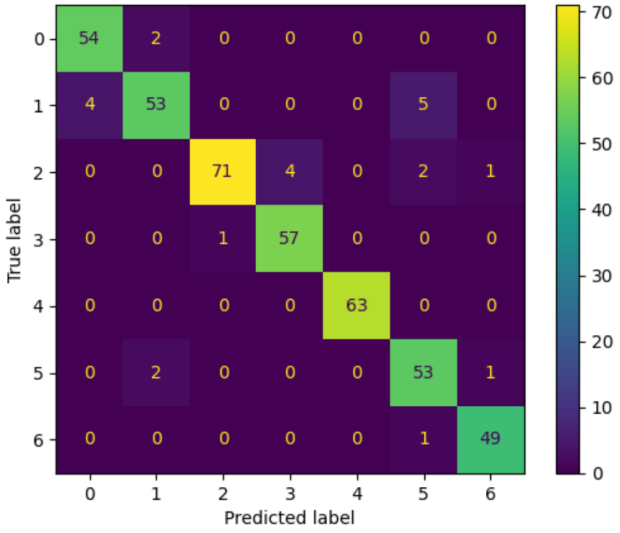

Figure 4: 梯度提升树

- mse：0.3759
- 准确率：94.56%

5.5. 结果分析

- 随机森林模型在此任务中表现出色，以较低的均方误差（MSE）和较高的准确率表现出其在处理多维分类问题上的稳健性。随机森林通过多个决策树的集成，有效降低了过拟合的风险，并在特征选择中具有较好的解释性。该模型的表现仅次于梯度提升树，在多类别分类任务中展示了优异的泛化能力。
- K-近邻模型也表现良好，准确率为 91.25%，但均方误差略高于随机森林。这种模型在特征空间上直接基于数据点的距离做出预测，因此对于边界清晰、分布均匀的数据有较好的效果。然而，对于维度较高或类别界限不清晰的数据，KNN 可能会表现出一定的误差。在本实验中，KNN 的准确率虽高，但其对数据分布的敏感性较强，性能稍逊于随机森林。
- 朴素贝叶斯模型的表现明显不如其他模型，准确率仅为 52.96%，均方误差较高（3.9669）。这可能与朴素贝叶斯模型假设特征之间相互独立有关。对于特征间有较强相关性或复杂依赖关系的数据集，朴素贝叶斯难以捕捉这种依赖结构，导致预测效果较差。因此，该模型不适合此类复杂特征的多类别分类任务，尤其在该实验数据集的上下文中。
- 梯度提升树模型在此实验中表现最佳，具有最低的 MSE 和最高的准确率（94.56%）。梯度提升树通过迭代地建立模型并逐步优化误差，能够很好地拟合复杂数据，同时抑制过拟合。其出色的表现说明该模型在处理多维数据并捕捉非线性关系方面具备明显优势，尤其适用于复杂任务中的类别边界清晰的情况。

5.6. 性能对比

模型表现对比：梯度提升树和随机森林模型表现优异，均显著高于 K-近邻和朴素贝叶斯。这表明集成模型在多维复杂数据集上具有显著的优势，尤其是在需要捕捉特征间复杂关系的任务中。下图展示了不同模型的性能对比情况。

数据适配性：该实验数据集的特征间可能存在相关性和非线性关系，因此，线性假设较强的朴素贝叶斯表现欠佳，而集成模型（如梯度提升树和随机森林）更适合此数据。

模型选择建议：对于类似肥胖预测的任务，可以优先考虑梯度提升树或随机森林。K-近邻模型在特征空间密集、分布均匀的情况下也具备一定的优势，但在维度高且数据复杂的情境下，其表现可能受限。

在本小节中，我们对比了四个分类模型（梯度提升树、朴素贝叶斯、K 近邻和随机森林）的性能表现。比较指标包括：

- **分类准确率（Accuracy）**：衡量模型在测试集上的整体预测准确程度。
- **精确率（Precision）**：表示预测为某一类的样本中，真正属于该类的比例。

- **召回率（Recall）**：表示属于某一类的样本中被正确预测的比例。
- **F1 分数（F1-score）**：精确率和召回率的调和平均数，衡量分类模型的综合性能。
- **均方误差（MSE）**：表示模型预测值与实际值之间的平均平方误差。

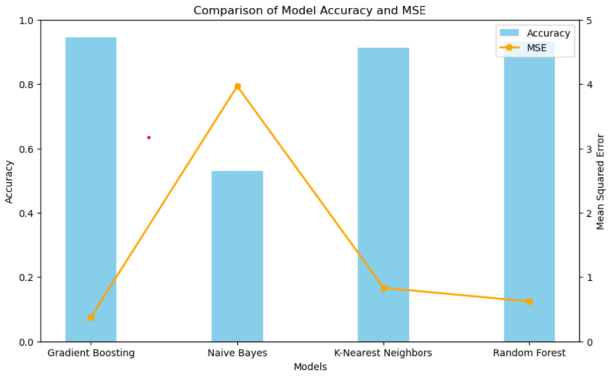

Figure 5: 性能对比

图6展示了各模型在分类准确率和均方误差方面的性能对比。通过图表可以观察到，梯度提升树和随机森林的表现相对较好，朴素贝叶斯的表现相对较差。这一比较结果为选择最适合的分类模型提供了依据。

5.7. 鲁棒性分析

在本小节中，我们对随机森林、K 近邻、朴素贝叶斯以及梯度提升树四种模型的鲁棒性进行了分析，重点考察了模型在不同测试集划分上的分类准确率表现。图 6 展示了不同模型在多次实验中的分类准确率波动情况。以下是对各模型的详细分析：

- **随机森林**：随机森林模型的分类准确率在不同的测试集划分中表现较为稳定，大多数实验的准确率维持在 0.93 至 0.95 之间，最高达到 0.9503。这样的表现表明，随机森林能够较好地适应数据集中的噪声和随机性，具有较高的鲁棒性。

  mathematica 复制代码

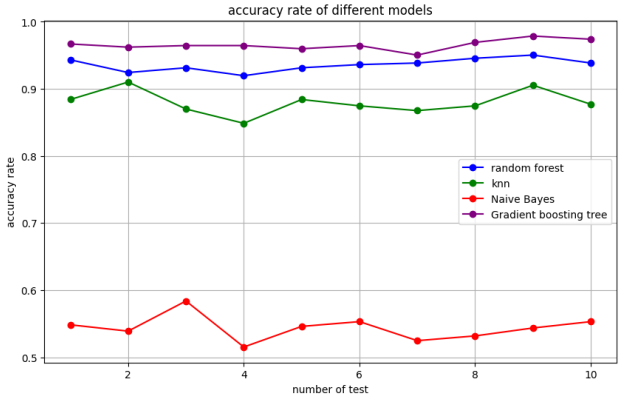

Figure 6: 鲁棒性

- **K 近邻（KNN）**：K 近邻模型的准确率波动较大，在不同测试集划分中准确率介于 0.84 到 0.91 之间，显示出对数据集变化的敏感性。K 近邻对特征空间密集和分布均匀的数据集具有一定的适应性，但当数据维度较高时，K 近邻模型的鲁棒性有所下降。
- **朴素贝叶斯**：朴素贝叶斯模型的分类准确率较低，通常在 0.51 至 0.58 之间，且波动较大。由于朴素贝叶斯假设特征独立性，而本数据集的特征间存在较强的相关性，因此导致其表现欠佳。该模型的鲁棒性较弱，难以应对复杂的数据关系。
- **梯度提升树（GBDT)**：梯度提升树的分类准确率表现优异且较为稳定，在多次实验中的准确率大多超过 0.96，最高达到 0.9787。梯度提升树模型能够有效地捕捉特征间的复杂非线性关系，具有很强的鲁棒性，适合用于复杂数据的分类任务。

综上所述，梯度提升树和随机森林模型在不同测试集划分中表现出较高的鲁棒性，准确率波动较小且整体表现优异。而 K 近邻和朴素贝叶斯在面对数据集变化时表现出较大的波动，尤其是朴素贝叶斯在处理复杂数据时的鲁棒性较弱。因此，在实际应用中，对于类似的复杂数据集，建议优先选择梯度提升树或随机森林模型。

模型	分类准确率	宏平均精确率	宏平均召回率	宏平均 F1 分数	测试集上的 MSE
梯度提升树	0.9456	0.9444	0.9484	0.9456	0.3759
朴素贝叶斯	0.5296	0.5395	0.5282	0.4843	3.9669
K 近邻	0.9125	0.9142	0.9128	0.9073	0.8322
随机森林	0.9314	0.9305	0.9322	0.9310	0.6217

Table 1: 不同模型的性能对比