08022311陈鲲龙第五六七次作业

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

train\_data = pd.read\_csv('dna.scale.tr', header=None, sep=' ', names=['class'] + [f'feature\_{i}' for i in range(1, 181)])

val\_data = pd.read\_csv('dna.scale.val', header=None, sep=' ', names=['class'] + [f'feature\_{i}' for i in range(1, 181)])

test\_data = pd.read\_csv('dna.scale.t', header=None, sep=' ', names=['class'] + [f'feature\_{i}' for i in range(1, 181)])

def preprocess(data):

    X = []

    y = []

    for index, row in data.iterrows():

        y.append(int(row['class']))

        features = row[1:].values

        feature\_dict = {}

        for feature in features:

            if pd.notna(feature):

                try:

                    idx, val = str(feature).split(':')

                    feature\_dict[int(idx)] = int(val)

                except ValueError:

                    print(f"Skipping invalid feature format: {feature}")

        X.append([feature\_dict.get(i, 0) for i in range(180)])

    return np.array(X), np.array(y)

X\_train, y\_train = preprocess(train\_data)

X\_val, y\_val = preprocess(val\_data)

X\_test, y\_test = preprocess(test\_data)

le = LabelEncoder()

y\_train = le.fit\_transform(y\_train)

y\_val = le.transform(y\_val)

y\_test = le.transform(y\_test)

dt\_model = DecisionTreeClassifier(random\_state=42)

rf\_model = RandomForestClassifier(random\_state=42)

nb\_model = GaussianNB()

svm\_model = SVC(random\_state=42)

dt\_model.fit(X\_train, y\_train)

y\_train\_pred\_dt = dt\_model.predict(X\_train)

y\_test\_pred\_dt = dt\_model.predict(X\_test)

rf\_model.fit(X\_train, y\_train)

y\_train\_pred\_rf = rf\_model.predict(X\_train)

y\_test\_pred\_rf = rf\_model.predict(X\_test)

nb\_model.fit(X\_train, y\_train)

y\_train\_pred\_nb = nb\_model.predict(X\_train)

y\_test\_pred\_nb = nb\_model.predict(X\_test)

svm\_model.fit(X\_train, y\_train)

y\_train\_pred\_svm = svm\_model.predict(X\_train)

y\_test\_pred\_svm = svm\_model.predict(X\_test)

train\_errors = {

    'Decision Tree': 1 - accuracy\_score(y\_train, y\_train\_pred\_dt),

    'Random Forest': 1 - accuracy\_score(y\_train, y\_train\_pred\_rf),

    'Naive Bayes': 1 - accuracy\_score(y\_train, y\_train\_pred\_nb),

    'SVM': 1 - accuracy\_score(y\_train, y\_train\_pred\_svm)

}

test\_errors = {

    'Decision Tree': 1 - accuracy\_score(y\_test, y\_test\_pred\_dt),

    'Random Forest': 1 - accuracy\_score(y\_test, y\_test\_pred\_rf),

    'Naive Bayes': 1 - accuracy\_score(y\_test, y\_test\_pred\_nb),

    'SVM': 1 - accuracy\_score(y\_test, y\_test\_pred\_svm)

}

print("训练错误率：", train\_errors)

print("测试错误率：", test\_errors)

models = ['Decision Tree', 'Random Forest', 'Naive Bayes', 'SVM']

train\_errors\_vals = list(train\_errors.values())

test\_errors\_vals = list(test\_errors.values())

x = np.arange(len(models))

width = 0.35

fig, ax = plt.subplots()

rects1 = ax.bar(x - width/2, train\_errors\_vals, width, label='训练错误率')

rects2 = ax.bar(x + width/2, test\_errors\_vals, width, label='测试错误率')

ax.set\_xlabel('模型')

ax.set\_ylabel('错误率')

ax.set\_title('不同模型的训练与测试错误率对比')

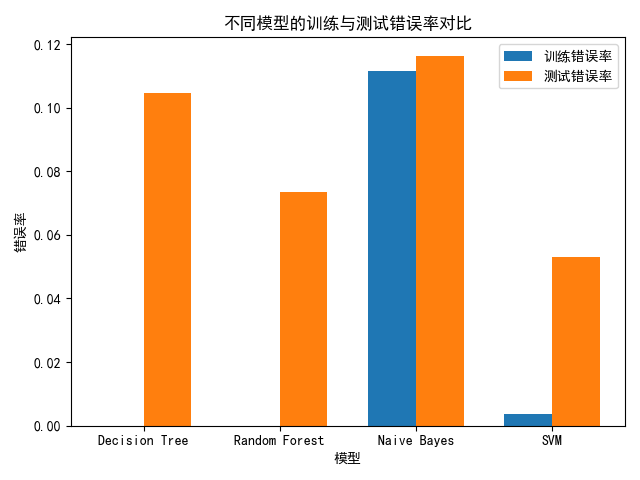
ax.set\_xticks(x)

ax.set\_xticklabels(models)

ax.legend()

fig.tight\_layout()

plt.show()



根据实验结果，以下是不同方法在训练集和测试集上的表现总结：

1.决策树 (Decision Tree)

训练集错误率为 0.0，说明模型完全拟合了训练数据。但在测试集上的错误率为 10.46%，表现出一定程度的过拟合现象。尽管如此，其测试集错误率在几种方法中表现较为良好。

2.随机森林 (Random Forest)

训练集错误率为 0.0，说明该模型对训练数据的拟合程度很高。在测试集上的错误率为 7.34%，优于决策树，表现出随机森林通过集成多个弱分类器有效缓解了过拟合问题。

3.朴素贝叶斯 (Naive Bayes)

训练集错误率为 11.14%，较高的错误率反映了朴素贝叶斯假设的简化性（属性之间的条件独立性假设）。在测试集上的错误率为 11.64%，略高于其他方法，显示该模型对数据复杂关系的捕捉能力有限。

4.支持向量机 (SVM)

训练集错误率为 0.36%，测试集错误率为 5.31%。SVM 在训练集和测试集上的表现都较好，说明其在高维空间中的分类能力较强，并且通过合适的超参数选择很好地平衡了模型的泛化能力。

总结：

随机森林和支持向量机在测试集上的错误率最低，显示了较强的泛化能力，其中 SVM 的测试集错误率最低（5.31%），是性能最优的模型。而决策树虽然在训练集上表现完美，但在测试集上存在过拟合风险。朴素贝叶斯的性能相对较差，可能与其简单的假设不适合该数据集有关。