集成学习

实验内容：动手实现分类决策数、基于决策树和集成学习实现随机森林；随机森林的sklearn调用与探究。

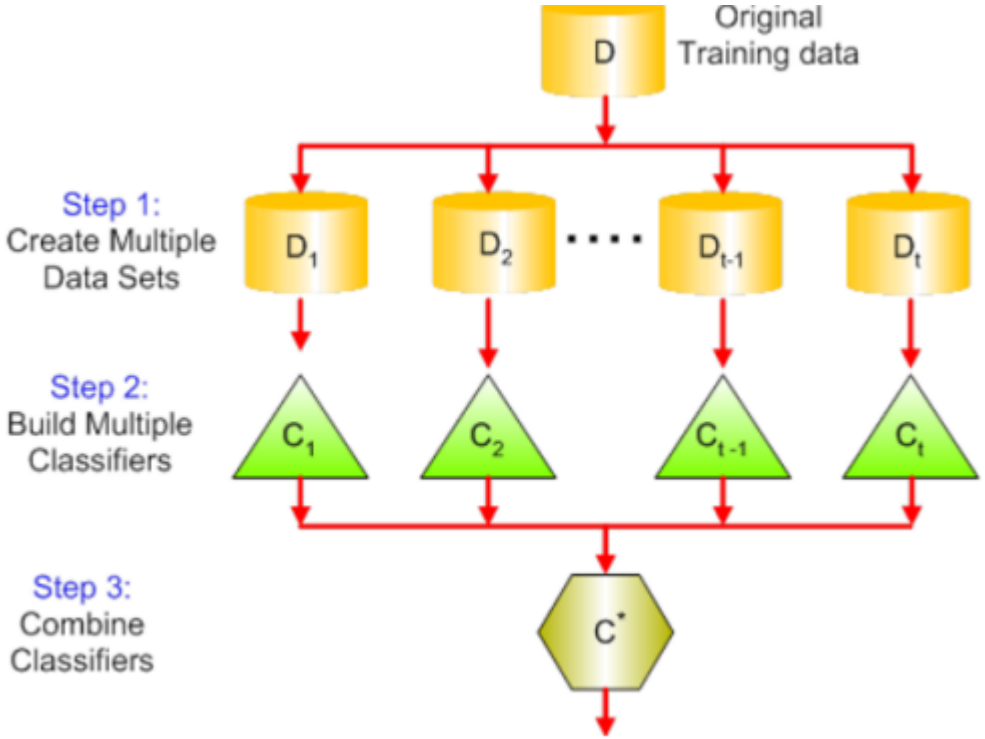
1. 集成学习

集成学习是通过构建并结合多个学习器来完成学习任务的方法。其核心思想是将多个弱学习器（weak learners）组合成一个强学习器（strong learner），从而提升整体模型的泛化能力和预测准确率

集成学习主要包括以下几种类型：

- Bagging：通过对数据集进行有放回的随机采样，生成多个子数据集，训练多个基学习器，并对结果进行平均或投票
- Boosting：通过顺序训练多个基学习器，每个基学习器关注被前一个学习器错误分类的样本，最终将多个基学习器的结果进行加权组合
- Stacking：通过训练多个基学习器，并使用一个元学习器来组合这些基学习器的预测结果

2.Bagging与随机森林



构建完全不同的数据子集是Bagging算法的主要思想。通过**自助采样**对训练集有放回采样产生若干不同但有交集的子集，然后基于每个子集训练基学习器。

假设数据集 D 有 m 个样本，从中**有放回采样** m 次到数据集 D_i 中，这种采样方法有可能导致某个样本会被选择好多次，也有可能某个样本一次也不会被选择。

对于一直不会被选择的样本，每次采样时不被选择的概率为 $1-\frac{1}{m}$，则 m 次采样不被选择的概率为 $(1-\frac{1}{m})^m$

$$\lim_{m\to\infty}(1-\frac{1}{m})^m=\frac{1}{e}\approx 0.368$$

理论上 36.8% 的样本不会被用于训练。通过自助采样得到若干自助采样集，利用这些子集训练出各自的基学习器，最终结果由基学习器各自的结果结合给出。

常用的聚合策略包括：

- 简单平均
- 加权平均
- 绝对多数投票：若某个标签计票过半，则预测为该标记，否则拒绝预测
- 相对多数投票：预测为计票最多的结果，若同时有多个标签计票最高，则随机选取
- 加权投票
- 学习法（Stacking策略）

随机森林算法

- 使用Bagging方法形成每颗**决策树**的训练集（从数据集角度增加随机性）
- 在构建每棵树时，假设当前节点共有 d 个可用属性，指定一个属性数 k ，从 d 个属性中随机抽取 k 个属性作为决策树分裂属性集，并从分裂属性集中选择最优属性用于数据划分（从特征角度增加随机性）
- 每颗树任其生长，不进行剪枝
- 对每个决策树的预测结果进行聚合

3. 随机森林的sklearn调用

In [1]:
```
# 导入必要的库
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.datasets import fetch_openml
import matplotlib.pyplot as plt
```

In [2]:
```
# 加载数据集
data = fetch_openml(name='adult', version=2, as_frame=True)
```

```
X = data.data
y = data.target
```

In [ ]:
```
# 数据预处理：对分类变量进行编码
label_encoders = {}
for column in X.select_dtypes(include=['category']).columns:
    label_encoders[column] = LabelEncoder()
    X[column] = label_encoders[column].fit_transform(X[column])

# 数据预处理：对标签进行编码
label_encoders_y = LabelEncoder()
y = label_encoders_y.fit_transform(y)
```

In [4]:
```
# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# 创建随机森林模型
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [6]:
```
# 训练模型
rf.fit(X_train, y_train)
```

Out[6]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [7]:
```
# 预测测试集
y_pred = rf.predict(X_test)
```

In [8]:
```
# 评估模型
# sklearn提供了许多常用模型评估函数，不需要自己重新造轮子
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}\n")
print(report)
```

```
Accuracy: 0.8625243115979118

              precision    recall  f1-score   support

           0       0.89      0.93      0.91      7479
           1       0.74      0.64      0.68      2290

    accuracy                           0.86      9769
   macro avg       0.82      0.78      0.80      9769
weighted avg       0.86      0.86      0.86      9769
```

4.探究基学习器数量和随机属性选取数量对随机森林的影响

- 基学习器数量（T）：表示采样多少棵决策树进行学习。

在sklearn的RandomForestClassifier中由**n_estimators**设定

- 随机属性选取数量（k）：在寻找最佳分裂时要考虑的特征数量。若属性数为 d，通常选择 $\sqrt{d}$ 或 $log_2 d$。

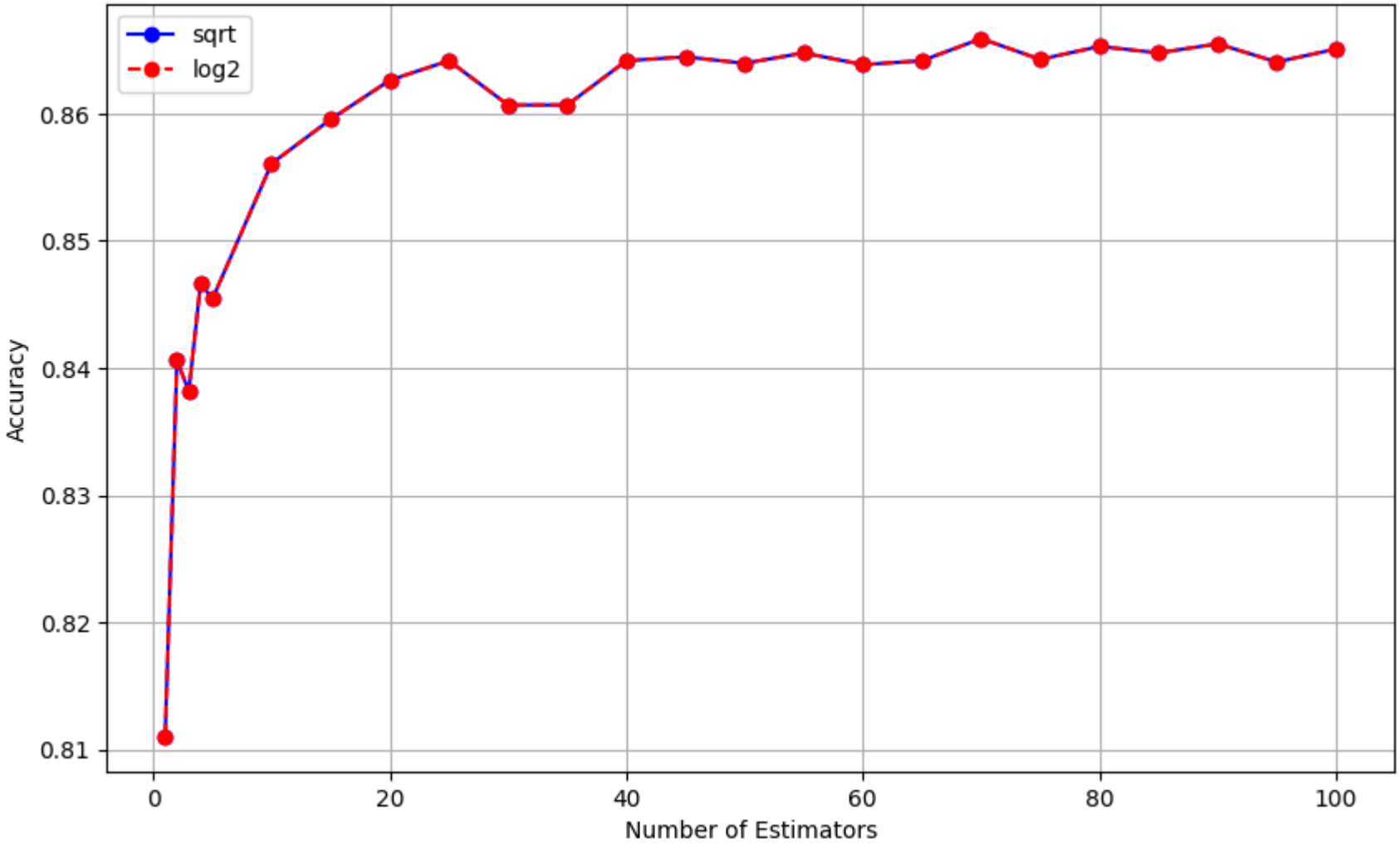
在sklearn的RandomForestClassifier中由**max_features**设定，可设置的值分别为"sqrt"和"log2"

In [9]:
```python
n_estimators_list = [1,2,3,4] + [5*(i+1) for i in range(20)] # 定义基学习器的数量范围

accuracies1 = []

accuracies2 = []
```

In [10]:
```python
# 探究max_features = "sqrt"的情况下，基学习器数量的影响
# 训练不同数量基学习器的随机森林模型，并记录准确率
for n_estimators in n_estimators_list:
    rf = RandomForestClassifier(n_estimators=n_estimators, random_state=1, max_features='sqrt')  #控制随机属性选取数量
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies1.append(accuracy)
```

In [11]:
```
# 探究max_features = "log2"的情况下，基学习器数量的影响
for n_estimators in n_estimators_list:
    rf = RandomForestClassifier(n_estimators=n_estimators, random_state=1, max_features='log2')  #控制随机属性选取数量
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies2.append(accuracy)
```

In [12]:
```
# 可视化结果
plt.figure(figsize=(10, 6))
plt.plot(n_estimators_list, accuracies1, marker='o', linestyle='-', color='b')
plt.plot(n_estimators_list, accuracies2, marker='o', linestyle='--', color='r')
plt.xlabel('Number of Estimators')
plt.ylabel('Accuracy')
plt.legend(['sqrt','log2'])
plt.grid(True)
plt.show()
```



结果分析

- 随机属性数量选取策略对本数据以及sklearn实现的“优质代码”没有影响
- 基学习器数量的增加，有效提升了预测准确率，随着数量的增加最终可能趋于稳定范围
- 基学习器数量的增加，学习时间增加，所以基学习器数量恰当即可，并非越多越好

5. 动手实践

- 请继续完成实验七，自己实现一个决策树分类器，并对adult数据集进行分类，检验决策树的有效性。
- **基于自己实现的决策树**，进一步实现随机森林，集成策略可以是Bagging，也可以采用其他集成策略。并对adult数据集进行分类，与单决策树做对比。