西北大学

智能信息系统综合实践
实验报告

题　　目：　集成学习

年　　级：　2020

专　　业：　软件工程

姓　　名：　何铖俊

一、题目（原题目）

- AdaBoost算法

1. 采用AdaBoost算法对一个自选数据集进行分类实验
2. 测试不同分类器和集成策略对性能的影响
3. 讨论和图示基分类器的偏差和方差

二、解题步骤（思路+代码）

本次实验将使用 Breast Cancer Wisconsin（诊断）数据集：这个数据集包含了肿瘤样本的一些测量数据，目的是将这些肿瘤样本分类为良性或恶性。
Breast Cancer 数据集是一个用于分类任务的标准数据集，用于对良性和恶性乳腺肿瘤进行分类。该数据集由威斯康星大学（University of Wisconsin）医学院玻璃学实验室收集，并由 UCI 机器学习库提供。
该数据集包含了 569 个样本，每个样本包含 30 个数值特征。这些特征基于数字化的乳腺组织图像计算得到，包括肿瘤的半径、质地、周长、面积、光滑度、紧密度等。
具体特征列表如下：
半径 (mean of distances from center to points on the perimeter)
纹理 (standard deviation of gray-scale values)
周长
面积
光滑度 (local variation in radius lengths)
COMPACTNESS（密度，计算方式：perimeter² / area - 1.0）
CONCAVITY（凹度，计算方式：severity of concave portions of the contour）
CONCAVE POINTS（凹点，计算方式：number of concave portions of the contour）
SYMMETRY（对称性）
FRACTAL DIMENSION（分形维度，计算方式："coastline approximation" - 1）
这些特征的值范围不同，有些特征被标准化处理过。每个样本的特征值均为实数。样本被标记为“良性”或“恶性”，其中 357 个样本标记为良性（B），212 个标记为恶性（M）。
但由于 breast_cancer 数据集特征数太多了，因此需要对其进行可视化分析并进行降维

对 Breast Cancer Wisconsin 数据集进行可视化分析

步骤：
1、导入必要的包和函数，包括 pandas、numpy、matplotlib.pyplot、seaborn 和 load_breast_cancer()函数。

2、使用 load_breast_cancer()函数导入数据集，并将数据转化为 pandas DataFrame 的形式。其中，X 为 30 个特征的值，y 为样本的类别标签。

3、统计样本的类别分布，使用 sns.countplot()函数绘制条形图。

4、查看各个特征之间的相关性，使用 sns.heatmap()函数绘制热力图，并添加相关系数的注释。

5、绘制各个特征的分布直方图，使用 plt.subplots()函数创建子图，并使用 for 循环遍历每一个特征，分别绘制其分布直方图。最后使用 plt.tight_layout()函数调整子图的布局。

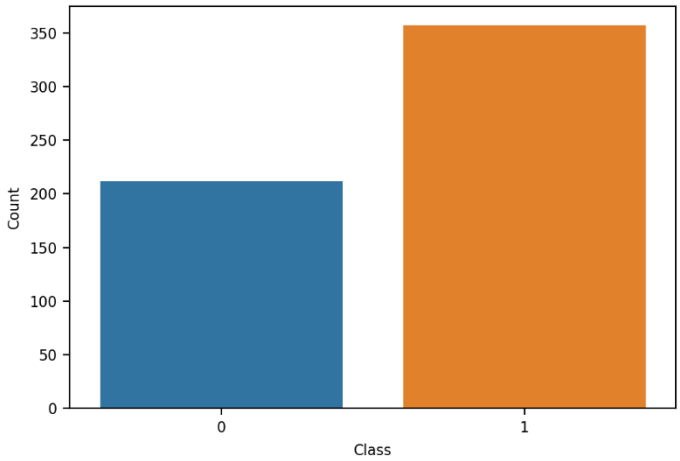
可视化代码如下

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
from sklearn.datasets import load_breast_cancer

# 导入数据集
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
# 设置字体大小
mpl.rcParams['font.size'] =8

# 统计样本的类别分布
plt.figure(figsize=(6,4))
sns.countplot(x=y)
plt.xlabel("Class")
plt.ylabel("Count")
plt.show()

# 查看特征之间的相关性
plt.figure(figsize=(30,40))
sns.heatmap(X.corr(), annot=True, cmap="coolwarm")
plt.show()

# 绘制各个特征的分布直方图
fig, axs = plt.subplots(nrows=10, ncols=3, figsize=(5,10))
for i in range(30):
    row, col = i//3, i%3
    axs[row, col].hist(X.iloc[:,i], bins=30, color="lightblue",
```

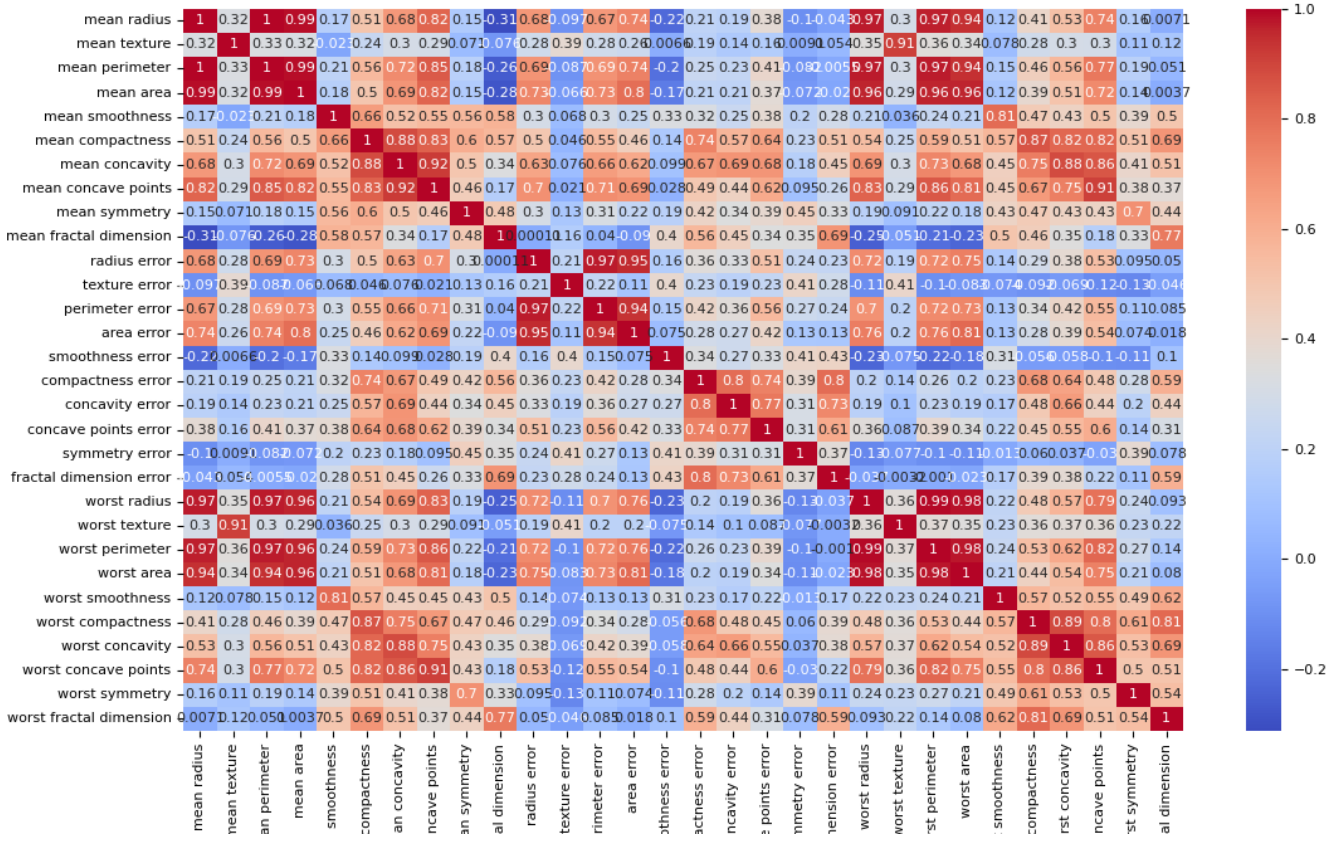
```
edgecolor="white")
    axs[row, col].set_title(X.columns[i])
plt.tight_layout()
plt.show()
```

最终结果

类别分布



特征之间的相关性

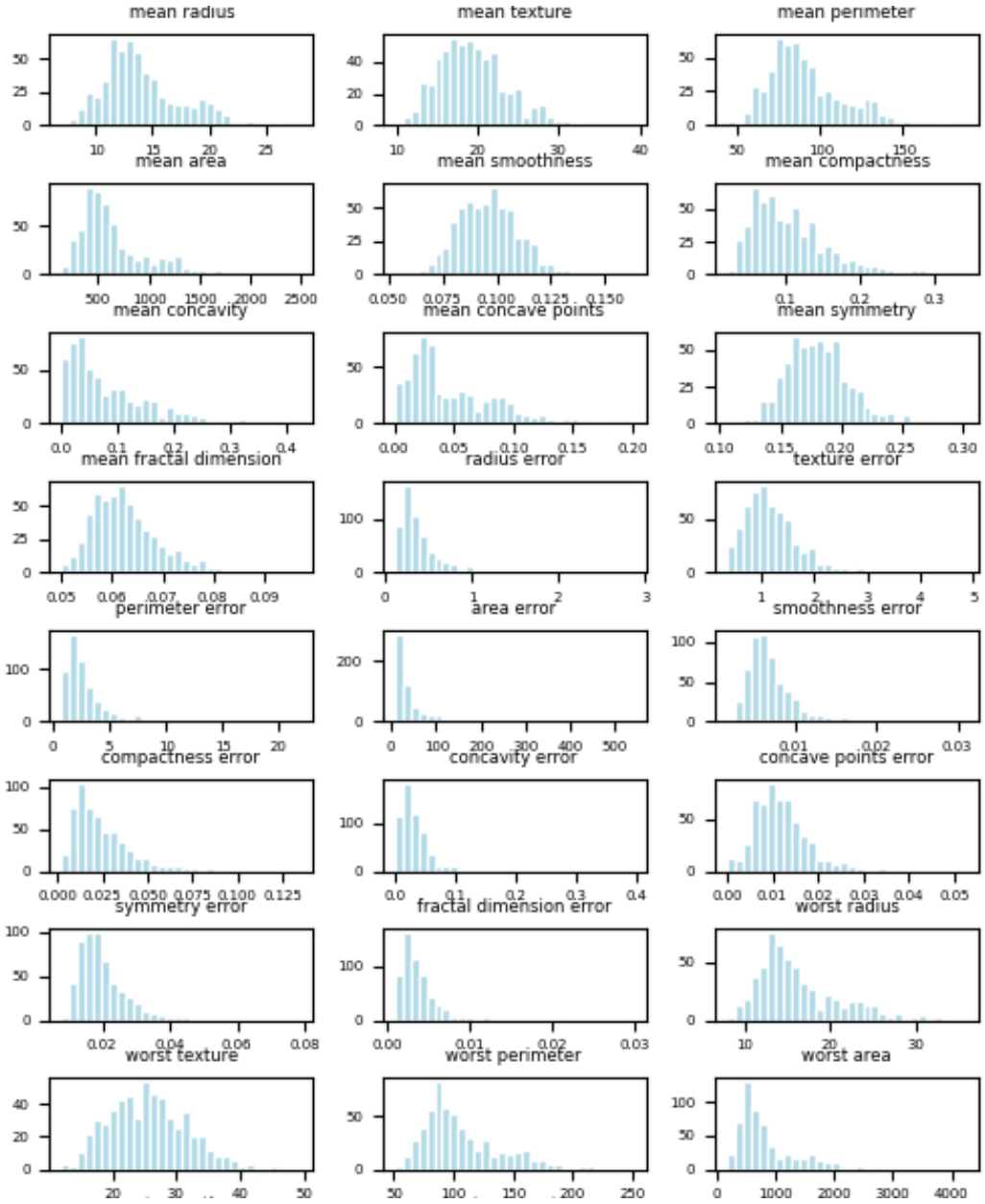


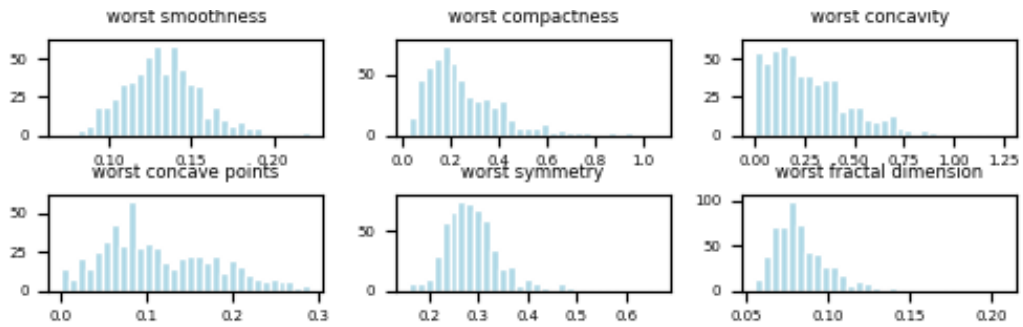
特征之间的相关性数值反映了它们之间的线性关系强度和方向。在机器学习领域中，对特征之间的相关性进行分析可以帮助我们进行特征选择、数据预处理、模型调优等任务。

具体来说，当特征之间的相关性较强时，我们可以考虑进行特征选择，保留其中一个特征，以减少冗余信息，提高模型训练效率和泛化性能。另外，当特征之间的相关性较弱时，我们可以考虑进行特征组合，将它们组合成新的特征，从而提高模型的预测能力。

此外，在特征选择和数据预处理中，我们还可以使用相关性矩阵进行特征筛选，选择与目标变量相关性较高的特征进行模型训练，提高模型的预测精度。

各个特征的分布直方图





分布直方图的主要意义在于：

1、展示数据的整体分布特征：通过绘制分布直方图，我们可以看到数据的整体分布特征，如数据是否对称、是否存在多个峰值、是否存在异常值等。

2、描述数据的中心趋势和离散程度：分布直方图可以帮助我们直观地了解数据的中心趋势和离散程度，如均值、中位数、众数、标准差等。

3、检查数据的偏态和峰度：分布直方图还可以帮助我们检查数据是否呈现偏态（如正偏态和负偏态）和峰度（如高峰态和平峰态），这对于后续的建模分析非常重要。

综合上述分析
应该舍弃掉'mean symmetry', 'smoothness error'特征

```
# 转换为 pandas DataFrame 对象
df = pd.DataFrame(X, columns=data.feature_names)

# 删除特定的特征
df.drop(['mean symmetry', 'smoothness error'], axis=1, inplace=True)

# 再次转换为 numpy.ndarray 对象
X = df.to_numpy()
```

将 pandas DataFrame 对象转换为 numpy.ndarray 类型的对象，而 numpy.ndarray 对象没有 drop() 方法。解决方法是在转换数据类型之前先对数据进行处理，或者使用 pandas 对象进行数据处理。

对数据进行处理后我们进行后续的分类

题目一

- AdaBoost算法

1. 采用AdaBoost算法对一个自选数据集进行分类实验

先写出一个 adaboost 调用模块，用默认参数对 breast cancer 数据集进行分类实验

在本次实验直接调用 sklearn.ensemble 库中的 AdaBoostClassifier

参数如下

base_estimator：弱学习器的基础模型，默认为决策树模型，也可以选择其他模型。

选择不同的基模型会对算法的性能产生影响，因此需要根据具体的应用场景和数据特点选择合适的基模型。同时，基模型的复杂度也会影响算法的性能，过于简单的模型会导致欠拟合，过于复杂的模型会导致过拟合。因此，在使用不同的基模型时，需要进行模型选择和调优，以获得更好的分类性能。

n_estimators：弱学习器的最大迭代次数，默认为 50。

learning_rate：每个弱学习器的权重缩减系数，默认为 1.0。该参数取值较小会使得算法收敛得更慢，但泛化能力会更好，反之则会收敛得更快，但泛化能力可能会降低。

algorithm：AdaBoost 算法的实现方式，默认为'SAMME.R'，即 RealBoost 算法。也可以选择'SAMME'算法。

在上述代码中，使用了 AdaBoostClassifier 类来实现 AdaBoost 算法，并通过 algorithm 参数指定了基分类器的类型。algorithm 参数的可选值包括：

SAMME.R（加权平均）：这是 AdaBoost 算法的默认算法，使用基分类器的预测概率来计算加权错误率，并根据加权错误率调整样本权重。此外，SAMME.R 还对基分类器的权重进行了修正，以避免过拟合。

SAMME：这是 AdaBoost 算法的原始算法，使用基分类器的预测结果来计算加权错误率，并根据加权错误率调整样本权重。但是，SAMME 不对基分类器的权重进行修正，容易导致过拟合。

random_state：随机数生成器的种子，默认为 None，即使用默认的随机数生成器。

（先采用采用决策树作为基分类器，弱学习器的最大迭代次设为 10，其余

参数都为默认值）

```
# 将数据集分为训练集和测试集

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# 创建 AdaBoost 分类器，以决策树作为基分类器
n_estimators = 10
base_estimator = DecisionTreeClassifier(max_depth=1)
ada_clf = AdaBoostClassifier(base_estimator=base_estimator,
                             n_estimators=n_estimators,
                             algorithm='SAMME',
                             learning_rate=1.0)

# 训练 AdaBoost 分类器
ada_clf.fit(X_train, y_train)
```

计算精确度

```
# 在测试集上进行预测
y_pred = ada_clf.predict(X_test)


# 计算 accuracy
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
```

```
Accuracy: 0.9766081871345029
```

题目二

2. 测试不同分类器和集成策略对性能的影响

具体来说，AdaBoost 算法使用的基函数需要满足两个条件：
基函数的输出结果必须是一个二元分类结果（正例或反例）。
基函数必须比随机猜测的分类器稍微好一些，即它的错误率不能超过 50%。
基于这些条件，AdaBoost 算法可以采用各种不同的基函数
在题目一种我们使用的是决策树作为基分类器，下面分别其他基本分类器（都采用默认参数）

决策树

```
clf1= DecisionTreeClassifier(max_depth=1)
```

```
Accuracy: 0.9766081871345029
```

逻辑回归

```
clf2 = LogisticRegression()
```

```
Accuracy: 0.9707602339181286
```

SVM

```
clf3 = SVC()
```

```
Accuracy: 0.631578947368421
```

随机森林

```
clf4 = RandomForestClassifier()
```

```
Accuracy: 0.9766081871345029
```

梯度提升树

```
clf5 = GradientBoostingClassifier()
```

```
Accuracy: 0.9590643274853801
```

朴素贝叶斯

```
clf6 = GaussianNB()
```

```
Accuracy: 0.9649122807017544
```

从上述几种可以粗略得到结论是：除了 svm 以外其他分类器作为基分类器在 breast cancer 数据集进行 adaboost 二分类表现效果优异，但 svm 默认参数效果不是很好

为什么 svm 默认参数效果不是很好？

```
clf3 = SVC(kernel='linear')
```

```
Accuracy: 0.9707602339181286
```

```
clf3 = SVC(kernel='poly')
```

```
Accuracy: 0.8128654970760234
```

```
clf3 = SVC(kernel='sigmoid')
```

```
Accuracy: 0.631578947368421
```

采用 svm 不同的 kernel 参数，可以得出结论，breast cancer 集更适用 svm 使用于线性核函数
可能原因：样本量足够大：当样本量足够大时，SVM 的线性核函数通常具有较好的泛化性能。但是当样本量较小时，线性核函数可能会过拟合。

采用不同的集合策略

Stacking

```
import pandas as pd
from sklearn.datasets import fetch_openml, load_breast_cancer
import numpy as np
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold, cross_val_predict,
train_test_split
from sklearn.linear_model import LogisticRegression

data = load_breast_cancer()
```

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

# 转换为 pandas DataFrame 对象
df = pd.DataFrame(X, columns=data.feature_names)

# 删除特定的特征
df.drop(['mean symmetry', 'smoothness error'], axis=1, inplace=True)

# 再次转换为 numpy.ndarray 对象
X = df.to_numpy()


# 将数据集分为训练集和测试集

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# 定义基本分类器
base_classifiers =
[AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
n_estimators=2, algorithm='SAMME'),

AdaBoostClassifier(DecisionTreeClassifier(max_depth=2),
n_estimators=5, algorithm='SAMME'),

AdaBoostClassifier(DecisionTreeClassifier(max_depth=3),
n_estimators=10, algorithm='SAMME')]

# 训练和评估基本 AdaBoost 分类器
for clf in base_classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print("基本 AdaBoost 分类器的精确度: ", accuracy)

# 创建 stacking 模型
stacking_model = LogisticRegression(random_state=0)

# 定义 KFold 交叉验证的折数
n_folds = 5

# 创建数组以存储第二层训练数据和测试数据
```

```
second_layer_train_data = np.zeros((X_train.shape[0],
len(base_classifiers)))
second_layer_test_data = np.zeros((X_test.shape[0],
len(base_classifiers)))

# 循环遍历基本分类器
for i, clf in enumerate(base_classifiers):
    # 创建一个 KFold 交叉验证对象
    kf = KFold(n_splits=n_folds, shuffle=True, random_state=0)

    # 使用 cross_val_predict 获取第二层训练数据
    second_layer_train_data_i = cross_val_predict(clf, X_train,
y_train, cv=kf, method='predict_proba')

    # 存储第二层训练数据
    second_layer_train_data[:, i] = second_layer_train_data_i[:, 1]

    # 在完整的训练数据上训练基本分类器
    clf.fit(X_train, y_train)

    # 使用训练好的基本分类器获取第二层测试数据
    second_layer_test_data_i = clf.predict_proba(X_test)

    # 存储第二层测试数据
    second_layer_test_data[:, i] = second_layer_test_data_i[:, 1]

stacking_model.fit(second_layer_train_data, y_train)
y_pred = stacking_model.predict(second_layer_test_data)
accuracy = accuracy_score(y_test, y_pred)
print("stacking 后 Accuracy:", accuracy)
```

以上代码通过 sklearn 库实现了 stacking 的过程，该过程使用了 AdaBoost 作为基本分类器，并使用逻辑回归作为第二层分类器。首先，使用 fetch_openml 方法加载 MNIST 数据集，并将其划分为训练集和测试集。然后，定义三个基本分类器，并创建一个逻辑回归模型作为第二层分类器。

接下来，使用 KFold 和 cross_val_predict 方法生成第二层训练数据。对于每个基本分类器，我们使用 KFold 方法将训练数据集划分为 K 个折，然后使用 cross_val_predict 方法对每个折进行预测

最终结果为：

```
基本 AdaBoost 分类器的精确度： 0.8947368421052632
基本 AdaBoost 分类器的精确度： 0.9473684210526315
基本 AdaBoost 分类器的精确度： 0.9590643274853801
stacking后Accuracy: 0.9649122807017544
```

从结果上可以得出 stacking 后的分类效果更好

Voting

VotingClassifier 类可以接受以下参数：

estimators：一个列表，包含要结合的分类器，每个分类器是一个元组，元组包含两个值，一个是分类器的名称，另一个是分类器本身。例如：estimators=[('clf1', clf1), ('clf2', clf2)]。
voting：结合分类器的方式。可以是 hard（默认值）或 soft。hard 表示采用少数服从多数的方式，即取预测结果中出现次数最多的类别作为最终结果；soft 表示对所有分类器的预测结果进行平均，然后选择得分最高的类别作为最终结果。
weights：一个列表，包含每个分类器的权重。如果未指定，则所有分类器的权重相等。
n_jobs：并行运行的作业数。默认值为 None，表示不并行计算。如果指定为-1，则使用所有可用的 CPU。

```
# 导入必要的库
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import AdaBoostClassifier, VotingClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
import numpy as np
import matplotlib.pyplot as plt

# 加载 breast_cancer 数据集
data = load_breast_cancer()

X = data.data
y =data.target

# 将数据集分为训练集和测试集

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

```
# 以决策树为基估计器创建 AdaBoost 分类器

n_estimators = 10
base_estimator = DecisionTreeClassifier(max_depth=1)

# 用 SAMME 算法训练 AdaBoost 分类器
ada_clf_samme = AdaBoostClassifier(base_estimator=base_estimator,
                                   n_estimators=n_estimators,
                                   algorithm='SAMME',
                                   learning_rate=1.0)
ada_clf_samme.fit(X_train, y_train)

# 用 SAMME.R 算法训练 AdaBoost 分类器
ada_clf_samme_r = AdaBoostClassifier(base_estimator=base_estimator,
                                     n_estimators=n_estimators,
                                     algorithm='SAMME.R',
                                     learning_rate=1.0)
ada_clf_samme_r.fit(X_train, y_train)


# 将 AdaBoost 分类器与投票相结合
voting_clf = VotingClassifier([('samme', ada_clf_samme), ('samme_r',
ada_clf_samme_r)], voting='hard')

voting_clf.fit(X_train, y_train)

# 评估 AdaBoost 分类器和投票分类器
y_pred_samme = ada_clf_samme.predict(X_test)
accuracy_samme = accuracy_score(y_test, y_pred_samme)
print("Accuracy with SAMME algorithm:", accuracy_samme)

y_pred_samme_r = ada_clf_samme_r.predict(X_test)
accuracy_samme_r = accuracy_score(y_test, y_pred_samme_r)
print("Accuracy with SAMME.R algorithm:", accuracy_samme_r)

y_pred_voting = voting_clf.predict(X_test)
accuracy_voting = accuracy_score(y_test, y_pred_voting)
print("Accuracy with voting:", accuracy_voting)
```

最终结果：

```
Accuracy with SAMME algorithm: 0.9766081871345029
Accuracy with SAMME.R algorithm: 0.9707602339181286
Accuracy with voting: 0.9766081871345029
```

从结果可以得出 voting 对分类性能是有提升的

题目三

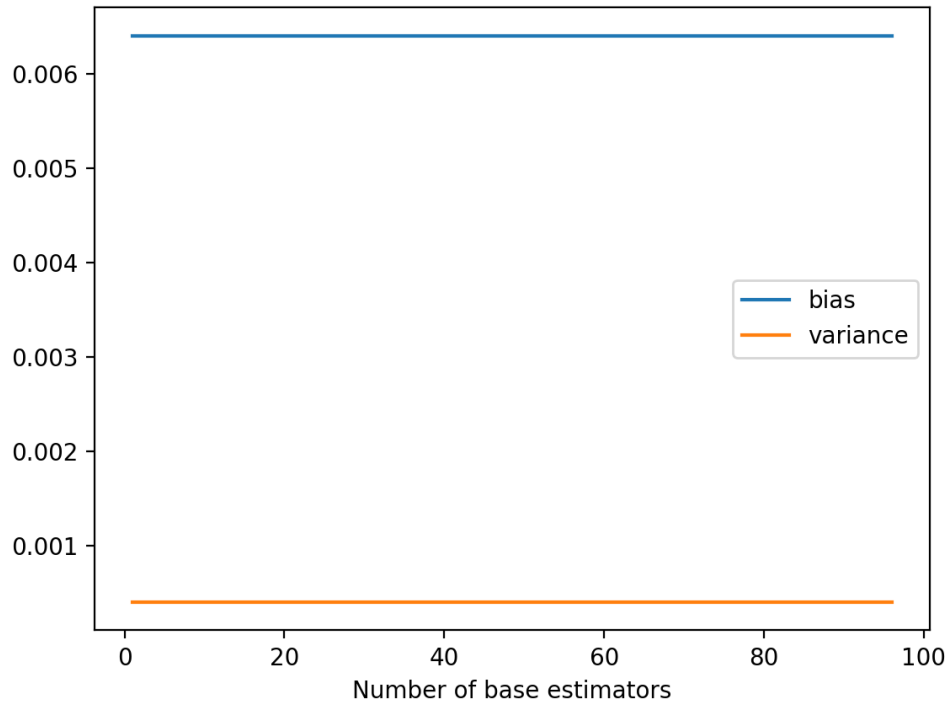
3. 讨论和图示基分类器的偏差和方差

1、从 base estimators 数量上进行分析

```python
from sklearn.datasets import load_breast_cancer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt

data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
# 定义函数计算误差
def calc_error(model, X, y):
    y_pred = model.predict(X)
    return np.mean(y_pred != y)

# 定义函数计算偏差和方差
def calc_bias_variance(model, X_train, y_train, X_test, y_test):
    train_errors = []
    test_errors = []
    num_estimators = range(1, 101, 5)
    for n in num_estimators:
        model.n_estimators = n
        model.fit(X_train, y_train)
        train_errors.append(calc_error(model, X_train, y_train))
        test_errors.append(calc_error(model, X_test, y_test))
    bias = np.mean(train_errors)
    variance = np.var(test_errors, ddof=1)
    return bias, variance, num_estimators
```

```
# 初始化 AdaBoostClassifier 模型
base_model = DecisionTreeClassifier(max_depth=1)
model = AdaBoostClassifier(base_estimator=base_model)

# 计算偏差和方差
bias, variance, num_estimators = calc_bias_variance(model, X_train,
y_train, X_test, y_test)

# 绘制偏差和方差图
plt.plot(num_estimators, np.ones_like(num_estimators) * bias,
label='bias')
#plt.plot(num_estimators, variance, label='variance')
plt.xlabel('Number of base estimators')
plt.legend()
plt.show()
```

最终结果

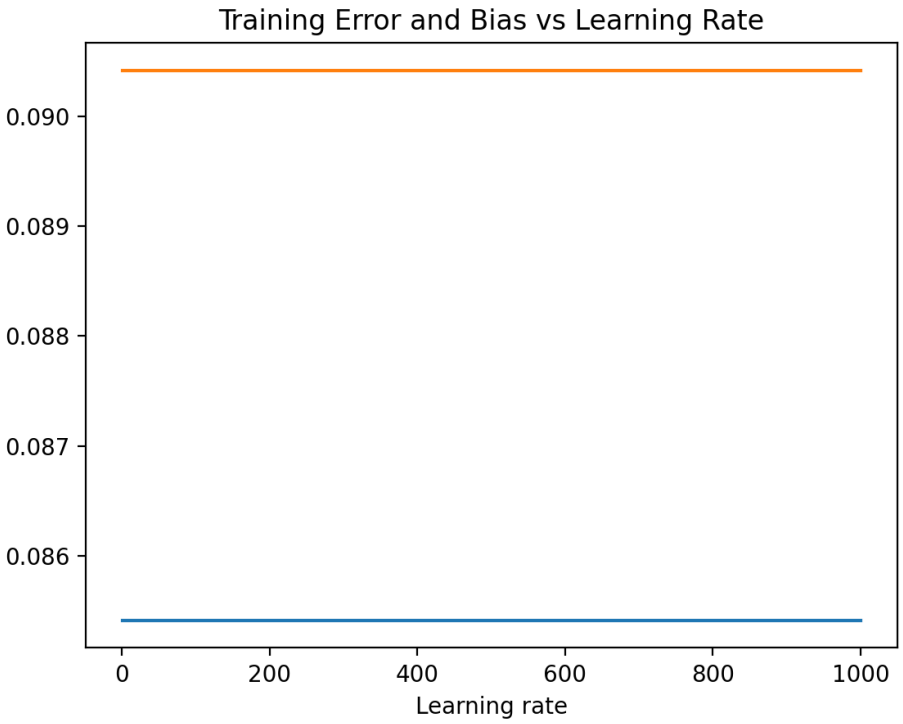


2、从 learning_rate 角度进行分析

```
# 定义存储误差和偏差的列表
dt_errors = []
dt_biases = []
for i in range(1, 1001):
    ada_clf = AdaBoostClassifier(base_estimator=clf1,
```

```
                                n_estimators=n_estimators,
                                algorithm='SAMME',
                                learning_rate=i)
    # 训练 AdaBoost 分类器
    ada_clf.fit(X_train, y_train)
    dt_scores = cross_val_score(clf1, X_train, y_train, cv=5)
    dt_error = 1 - dt_scores.mean()
    dt_bias = dt_error - (1 - 0.5 * np.log(len(X_train))) /
len(X_train)
    # 将误差和偏差存入列表
    dt_errors.append(dt_error)
    dt_biases.append(dt_bias)
# 定义学习率范围
learning_rates = np.linspace(1, 1000, num=1000)
# 绘制训练误差和偏差变化曲线
plt.plot(learning_rates, dt_errors, label='Training error')
plt.plot(learning_rates, dt_biases, label='Training bias')
plt.title('Training Error and Bias vs Learning Rate')
plt.xlabel('Learning rate')
plt.show()
```

最终结果



从上述两个角度的图像可知，经过数据处理的 breast cancer 数据集在 learning rate 和基分类器数量的变化下，偏差和误差并没有

什么大的变化

三、总结（心得体会）

1、应当在实验过程中保持一个刨根问底的求学态度，才可以在学习过程中将收获最大化。

2、在进行训练之前，应该先对数据集的总体分布、特征有个大概的印象和了解，方便后续使用恰当的方法进行分类模型和预测。

3、当一种方法行不通时，应当敢于尝试其他的方法，而不是死脑经盯着一处。

4、实验过程中不能心急，急躁的情绪只会让结果越来越不符合自己的心理预期，导致恶性循环。