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：　张琬琪

一、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

【数据集选取说明】：

总共选取了两类数据集，分别为：**Mnist 手写字数据集（题目 1 使用）与鸢尾花数据集（题目 2 与题目 3 使用）**，之前几次实验中已经做过数据分析，这里不再赘述。

题目 1：

①【解题思路】：

1）读取 mnist 数据集，并进行格式整理，将数据和标签分别提取存入；随机划分数据集，使用训练集训练模型，测试集测试模型效果，代码见图 1：

```
raw_data = pd.read_csv('mnist_data_70000.csv', header=0)
data = raw_data.values
imgs = data[0::, 1::]
labels = data[::, 0]
x_train, x_test, y_train, y_test = train_test_split(imgs, labels,
                                                    test_size=0.3,random_state=42)
```

图 1

2）调取 sklearn 中 AdaBoost 分类模型。

关于 AdaBoost 的简单说明介绍：

AdaBoost 是一种有效而实用的 Boosting 算法，它以一种高度自适应的方式按顺序训练弱学习器。针对分类问题，AdaBoost 算法根据前一次的分类效果调整数据的权重，在上一个弱学习器中分类错误的样本的权重会在下一个弱学习器中增加，分类正确的样本的权重则相应减少，并且在每一轮迭代时会向模型加入一个新的弱学习器。**不断重复调整权重和训练弱学习器，直到误分类数低于预设值或迭代次数达到指定最大值，最终得到一个强学习器。**简单来说，AdaBoost **算法的核心思想就是调整错误样本的权重，进而迭代升级。**

AdaBoost **模型参数说明：**

class sklearn.ensemble.AdaBoostClassifier （base_estimator = None，n_estimators = 50，learning_rate = 1.0，algorithm = ‘SAMME.R’，random_state = None ）

默认的基分类器为弱分类器：DecisionTreeClassifier

在实验中，只对参数 random_state 输入了整数，设置了随机数。

3）利用训练数据训练模型，并使用测试集进行模型测试。

代码见图 2：

```
model = AdaBoostClassifier(random_state=1)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
predicted_probas = model.predict_proba(x_test)
```

图 2

4）以列表的形式输出实际值和预测值，并输出查看分类相关的具体信息以及混淆矩阵的输出。

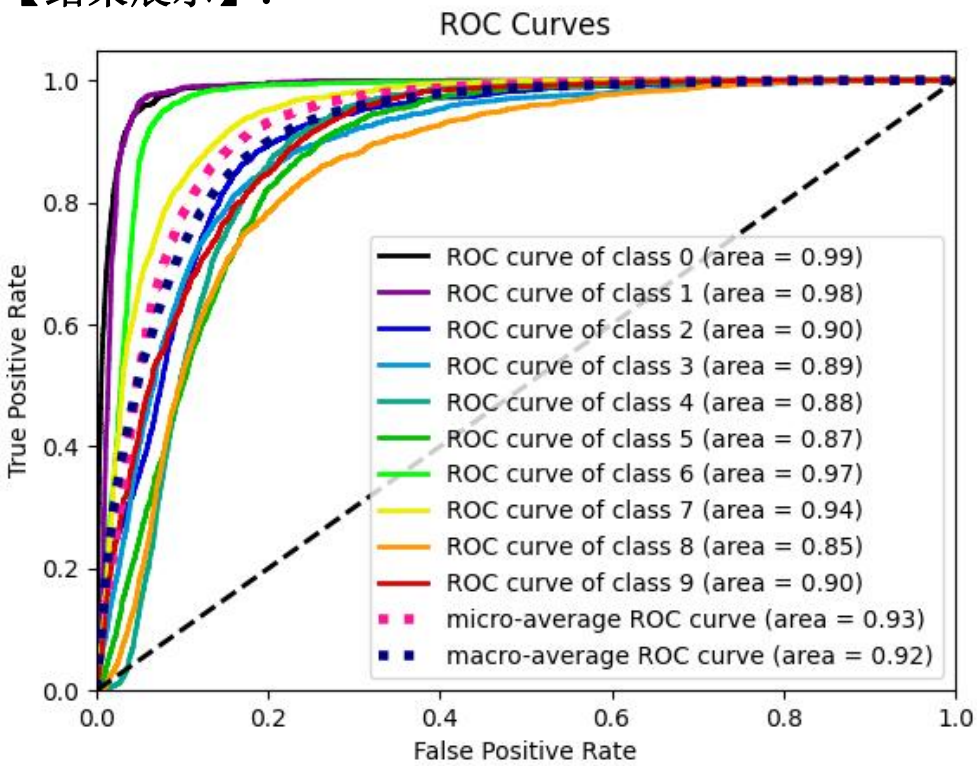
5）绘制 ROC 曲线，进一步观察和评估该数据集上的分类模型。

代码见图 3：

```
result=pd.DataFrame()
result['预测值']=list(y_pred)
result['实际值']=list(y_test)
result.head()
print(result)
print('\t')
#查看所有测试集数据的预测准确度
print('分类完毕后的基本信息:\t')
print(classification_report(y_test, y_pred))
print('混淆矩阵:\t')
print(metrics.confusion_matrix(y_test, y_pred))#混淆矩阵
print('\t')
print('mnist数据集预测准确度：{:.2f}%'.format(model.score(x_test, y_test) * 100))

skplt.metrics.plot_roc(y_test, predicted_probas)#绘制ROC曲线
plt.show()
```

图 3

②【结果展示】：



01_AdaBoost

```
D:\developlAPP\python3.10.4\python.exe D:\科目学习类\pythonhomew
       预测值  实际值
0        4    6
1        1    1
2        5    8
3        4    4
4        7    3
...    ...  ...
20995    8    7
20996    6    6
20997    5    5
20998    4    4
20999    1    1

[21000 rows x 2 columns]

分类完毕后的基本信息:
              precision    recall  f1-score   support

           0       0.87      0.89      0.88      2064
           1       0.85      0.94      0.90      2422
           2       0.72      0.61      0.66      2078
           3       0.69      0.67      0.68      2163
           4       0.67      0.62      0.64      2001
           5       0.69      0.64      0.67      1879
           6       0.77      0.82      0.80      2092
           7       0.65      0.68      0.66      2221
           8       0.69      0.72      0.70      2013
           9       0.55      0.57      0.56      2067

    accuracy                           0.72     21000
   macro avg       0.72      0.72      0.71     21000
weighted avg       0.72      0.72      0.72     21000

混淆矩阵:
[[1828    0   66    7    4   83   48    2   15   11]
 [   0 2279   14   18    3    8   13   64   21    2]
 [  56   66 1259   78   60   26  301   54  166   12]
 [  54   68   41 1451   10  241   70   57  103   68]
 [  11    8   52   34 1239   24   15  313   71  234]
 [  48   49   35  222   60 1210   50   21  112   72]
 [  37   22  153    4   82   44 1719    2   28    1]
 [  16   31   33   28   32   19    0 1505   59  498]
 [  29  112   34  211   23   47   14   27 1446   70]
 [  11   35   50   52  330   48    0  282   86 1173]]

mnist数据集预测准确度：71.95%

Process finished with exit code 0
```

题目 2：

在 sklearn 中调用集成学习的使用方法，从基模型的角度上可以简单地将方法分为两类，以下小结了对应的方法和模型类型：

方法名称	基模型
Adaboost	单个模型
Bagging	单个模型
Gradient boost	决策树
Voting	多个模型
Staking	多个模型

①【解题思路】：

数据处理同题目 1，但这里的实验数据选用了鸢尾花数据集，直接调取，数据集典型且能使实验完成效率更高。代码见图 4：

```
iris = load_iris()
X = iris.data[:, :2]  # 获取花卉两列数据集
y = iris.target  # 获取label
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.2,
                                                    random_state=42)
```

图 4

1-不同分类器对性能的影响：

1）根据前文叙述，并结合题意，本次实验首先在 Adaboost 基础上选取了不同的单个基分类器（单个模型表示每次只能在该集成学习模型中输入一个基分类器模型，而不能一次输入多个不同的基分类器模型）。实验中共选取了四个基分类器，见图 5：

```
model_dict = {
    'Logistic Regression':
        (
            LogisticRegression()
        ),
    'RandomForestClassifier':
        (
            RandomForestClassifier()
        ),
    'GaussianNB':
        (
            GaussianNB()
        ),
    'DecisionTreeClassifier':
        (
            DecisionTreeClassifier()
        )
}
```

图 5

2）利用 for 循环，完成不同类型的基分类器模型训练与预测后，并输出相关分类报告信息。代码见图 6：

```
for model_name, (model) in model_dict.items():
    clf = AdaBoostClassifier(model, n_estimators=50,random_state=1)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test) # 进行预测
    print('Afaboost_iris数据集_{}模型的分类报告信息数据：'.format(model_name))
    print(classification_report(y_test, y_pred))
```

图 6

3）拓展实验，为了进一步在宏观上，横向上对采用单个基分类器模型的集成学习进行对比，便又选取调用了 Bagging 模型

关于 Bagging 的简单说明介绍：

Bagging 的思路是训练 k 个独立的基学习器，对于每个基学习器的结果进行结合来获得一个强学习器。

Bagging 要求基分类器对样本分布敏感，常用的基分类器为决策树、神经网络。KNN、线性分类器由于过于“稳定”不适合作为基

分类器。因此在这里实验中的基模型，只选取了 RandomForestClassifier，DecisionTreeClassifier。

完成 Bagging 不同类型的基分类器模型训练与预测后，同样输出相关分类报告信息。

代码见图 7：

```
model_dict = {

    'RandomForestClassifier':
        (
            RandomForestClassifier(n_estimators=50, random_state=1)
        ),
    'DecisionTreeClassifier':
        (
            DecisionTreeClassifier()
        )
}

for model_name, (model) in model_dict.items():
    clf = BaggingClassifier(model, n_estimators=50,random_state=1)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test) # 进行预测
    acc = clf.score(x_test, y_test)
    f1 = f1_score(y_test, y_pred, average='micro')
    print('Bagging_iris数据集_{}模型的分类报告信息数据：'.format(model_name))
    print(classification_report(y_test, y_pred))
    print('\t')
```

图 7

2-不同集成策略对性能的影响：

在不同分类器的解决方案中，虽然选取了不同的集成学习模型，但是二者皆为只能进行单个学习模型的输入，集成策略本质上相同。

因此在实验中，选择了 sklearn 中的 Voting 模型以进行集成策略上的对比实验。

关于 **Voting:**

Voting 是一种多模型集成的常用方法，它先将多个模型进行投票，再按照投票结果对模型进行组合。投票法又可以被划分为硬投票与软投票，实验中使用硬投票（预测结果是所有投票结果最多出现的类。）

本次实验中将 Logistic Regression， 高斯朴素贝叶斯，随机森林与决策树四个基分类器进行集成。

最后输出相关分类报告信息，**在这里前几问的基础上，又加入了交叉验证得分的输出，（交叉验证是一种非常常用的对于模型泛化能力进行评估方法，既可以解决数据集的数据量不够大问题，也可以解决参数调优的问题。）多角度探寻和发现，组合分类器的效果。**

代码见图 8：

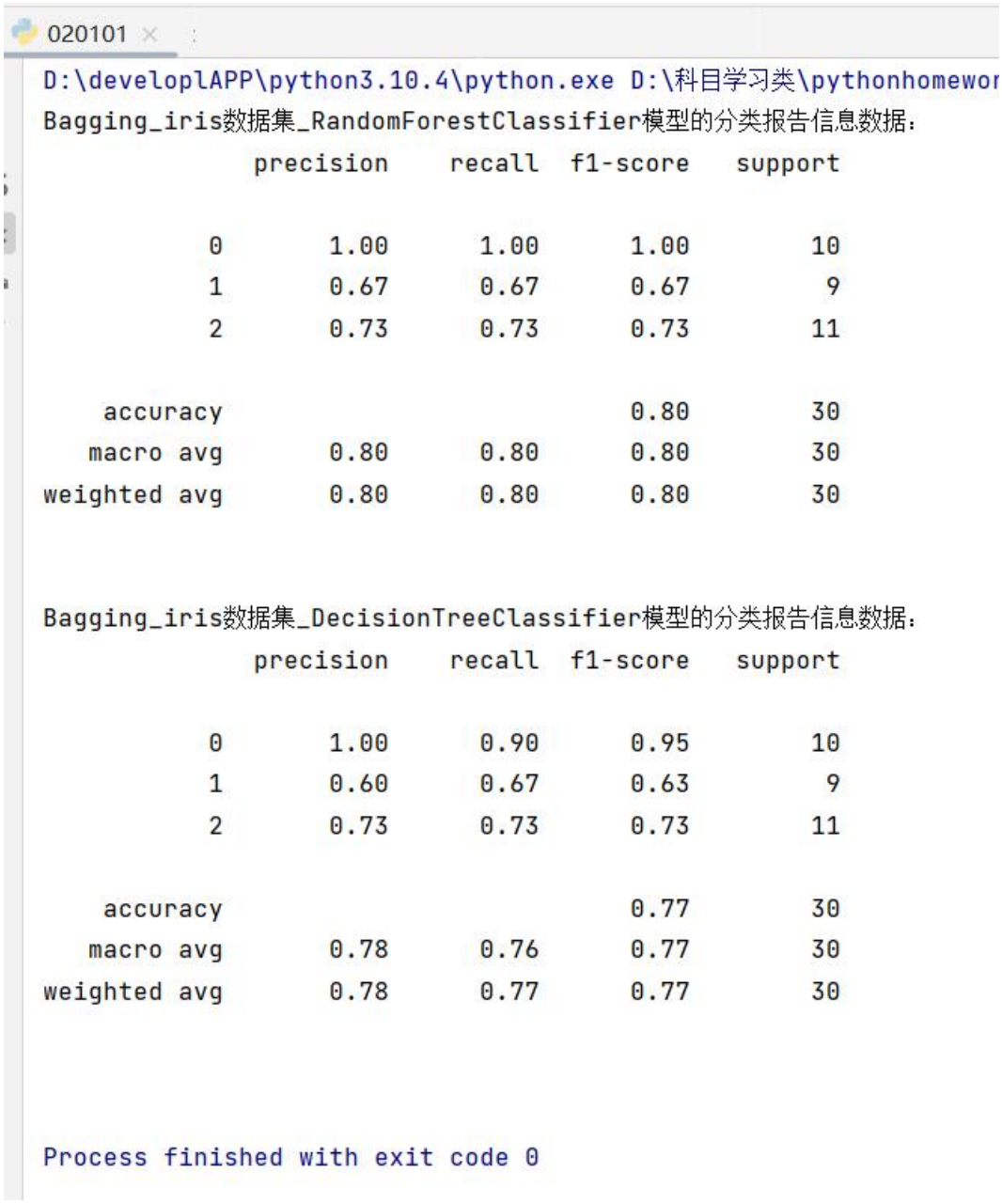
```
clf1 = LogisticRegression(random_state=1)
clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
clf3 = GaussianNB()
clf4 = DecisionTreeClassifier()

eclf = VotingClassifier(
    estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3),('dt',clf4)],
    voting='hard')

eclf.fit(x_train,y_train)
y_pred = eclf.predict(x_test) # 进行预测
print('iris数据集Voting模型的分类报告信息数据：')
print(classification_report(y_test, y_pred))
print('\t')

for clf, label in zip([clf1, clf2, clf3, eclf],
     ['Logistic Regression', 'Random Forest', 'naive Bayes', 'DecisionTree','Ensemble']):
    scores = cross_val_score(clf, X, y, scoring='accuracy', cv=5)
    print("交叉验证平均得分：%0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

图 8

②【结果展示】：

```
D:\developlAPP\python3.10.4\python.exe D:\科目学习类\pythonhc
Afaboost_iris数据集_Logistic Regression模型的分类报告信息数据：
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.53      0.89      0.67         9
           2       0.80      0.36      0.50        11

    accuracy                           0.73        30
   macro avg       0.78      0.75      0.72        30
weighted avg       0.79      0.73      0.72        30

Afaboost_iris数据集_GaussianNB模型的分类报告信息数据：
              precision    recall  f1-score   support

           0       1.00      0.50      0.67        10
           1       0.36      1.00      0.53         9
           2       0.00      0.00      0.00        11

    accuracy                           0.47        30
   macro avg       0.45      0.50      0.40        30
weighted avg       0.44      0.47      0.38        30

Afaboost_iris数据集_DecisionTreeClassifier模型的分类报告信息数据：
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        10
           1       0.55      0.67      0.60         9
           2       0.70      0.64      0.67        11

    accuracy                           0.73        30
   macro avg       0.75      0.73      0.74        30
weighted avg       0.75      0.73      0.74        30

Afaboost_iris数据集_RandomForestClassifier模型的分类报告信息数据：
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.67      0.67      0.67         9
           2       0.73      0.73      0.73        11

    accuracy                           0.80        30
   macro avg       0.80      0.80      0.80        30
weighted avg       0.80      0.80      0.80        30
```

通过实验结果来看，不同基分类器对于集成效果确实会造成不同程度的影响，而在本次实验中随机森林模型表现较好。

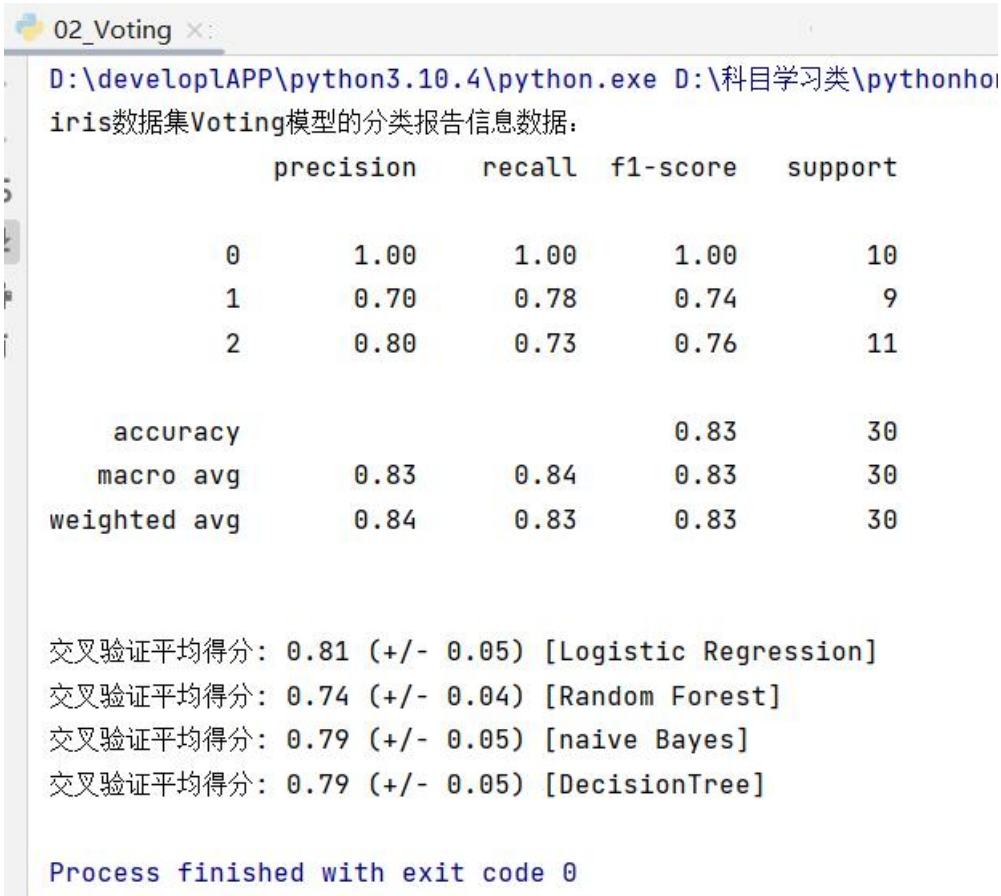


结合数据来看，在 **Bagging** 中，同样也是随机森林模型效果会更好。

横向与 **Adaboost** 进行对比，在本实验数据集上 **Bagging** 整体效果表现更佳。

补充：

随机森林是将多个决策树的预测组合成一个模型。逻辑就是一个由许多平庸的模型组成的模型，仍然优于一个单一的好模型。观察随机森林的主流表现，也确实如此。且随机森林不易产生过拟合。



结合实验数据来看，结合了多种类基分类器的投票方法整体效果都要比 Adaboost 与 Bagging 更佳。

题目 3：

①【解题思路】：

在题目二—Adaboost 基础上，进行基分类器的方差和偏差分析。

简要说明：

模型方差---模型的方差是模型在拟合不同的训练数据时性能的变化大小。它反映特定数据对模型的影响。

模型偏差---偏差用于衡量一个模型拟合的输入和输出之间关系与真实情况的近似程度。获得模型的刚度：模型对于输入和输出之间的函数的假设强度。

1）计算方差和偏差。

Sebastian Raschka 建立的 mlxtend 库提供了 bias_variance_decomp()函数，可以对一个模型采用多重自采样（multiple bootstrap samples）的方式来评估偏差和方差。

2）输出偏差和方差结果。注：这里增加了一项：模型的评估误差-即：偏差加上方差。

3）通过绘制柱状图，将不同的分类器的偏差和方差图示化。

计算代码见图 9：

```
mse, bias, var = bias_variance_decomp(model, x_train, y_train, x_test, y_test,
                                      loss='mse', num_rounds=500, random_seed=1)
num_list.append(bias)
num_list2.append(var)

# summarize results
print('Afaboost_iris数据集_{}模型的误差相关信息数据：'.format(model_name))
print('MSE:%.3f' % mse)
print('Bias:%.3f' % bias)
print('Variance:%.3f' % var)
print('\t')
```

图 9

绘图代码见图 10：

```
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False

name_list = ['Logistic Regression', 'Random Forest', 'naive Bayes', 'DecisionTree']
x = list(range(len(num_list)))
width = 0.3  # 柱子的宽度
index = np.arange(len(name_list))
plt.bar(index, num_list, width, color='blue', tick_label=name_list, label='偏差')
plt.bar(index + width, num_list2, width, color='yellow', hatch='\\', label='方差')
plt.legend(labelspacing=1)

for a, b in zip(index, num_list):  # 柱子上的数字显示
    plt.text(a, b, '%.3f'%b ,ha='center', va='bottom', fontsize=10)
for a, b in zip(index + width, num_list2):
    plt.text(a, b, '%.3f'%b,ha='center', va='bottom', fontsize=10)

plt.title('Adaboost_不同基分类器的偏差和方差展示')
plt.ylabel('数值大小')
plt.legend(loc='best')
plt.show()
```

图 10

通过课件学习与相关资料学习，了解了集成学习与偏差和方差的关系。

我们所需要讨论分析的基分类器（弱分类器）的错误是偏差和方差两种错误之和。

偏差错误主要是由于分类器的表达能力有限导致的系统性错误，表现在训练误差不收敛。

方差错误主要是由于分类器对于样本分布过于敏感，导致在训练样本数较少时，产生过拟合。

②【结果展示】：

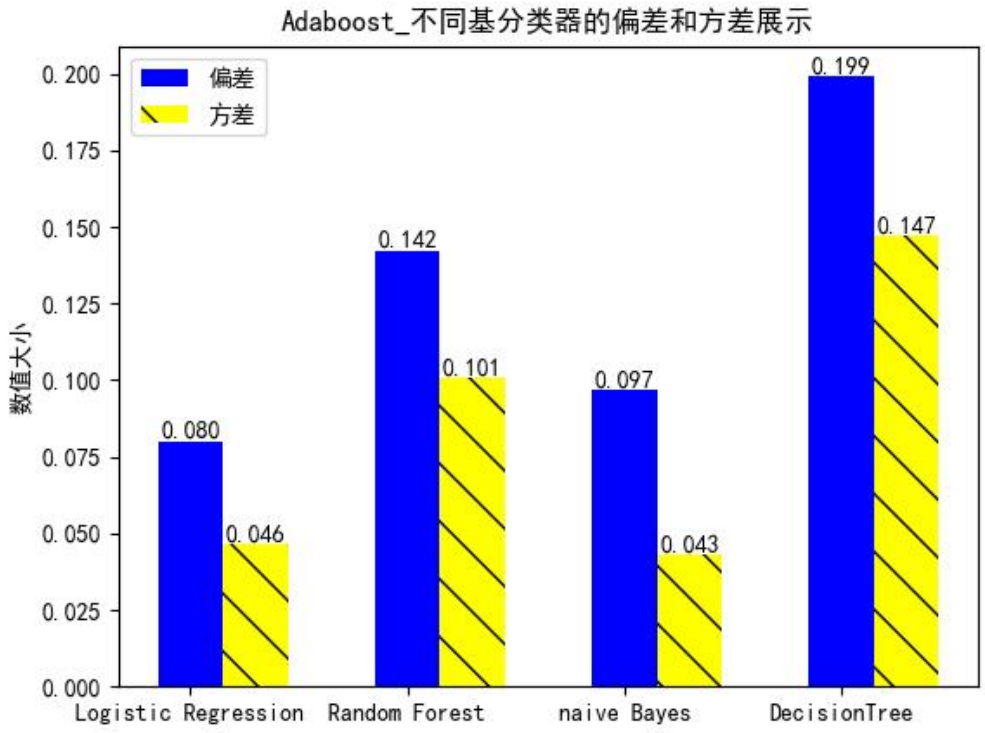
```
Afaboost_iris数据集_Logistic Regression模型的误差相关信息数据：
MSE:0.126
Bias:0.080
Variance:0.046
```

```
Afaboost_iris数据集_RandomForestClassifier模型的误差相关信息数据：
MSE:0.243
Bias:0.142
Variance:0.101
```

```
Afaboost_iris数据集_GaussianNB模型的误差相关信息数据：
MSE:0.140
Bias:0.097
Variance:0.043
```

```
Afaboost_iris数据集_DecisionTreeClassifier模型的误差相关信息数据：
MSE:0.346
Bias:0.199
Variance:0.147
```

注：**MSE-评估误差　Bias-偏差　Variance-方差**



三、总结

①集成学习是将多个模型放在一起，通过某种方式将其组合起来，从而得到一个统一的预测模型。而在集成之前就要有基本的模型用于集成。

目前所用的集成学习模式主要分为两种：

采用同一基模型，在样本的不同子集上进行训练，从而得到对应的多个不同的具体模型。这类学习方法的代表就是**boosting, bagging**。

采用多种基模型，在同一样本上进行训练，将多个模型组合起来得到具体模型。**voting** 和 **stacking** 都是采用这类方式。

②在多次实验中都是用到 **sklearn** 库方法，而在 **sklearn** 中实现这样的集成学习同样是非常简单高效，只需要一句代码即可。训练和预测和直接使用机器学习模型是完全一样的语法。它尽最大可能减小了程序员去记忆工具模式的工作，从而可以将更多精力集中在问题本身。

③通过本次实验学习到：衡量一个机器学习模型的性能，可以用偏差和方差作为依据：

一个高偏差的模型，总是会对数据分布做出强假设，比如线性回归。而一个高方差的模型，总是会过度依赖于它的

训练集，例如未修剪的决策树。我们会希望一个模型的偏差和方差都很低，但更多情况下我们需要在二者之间做出权衡。

④在实验过程中出现了一个警告（如下所示），通过查找资料发现这类警告--根本原因还是在于模型结构问题，即代码问题，或者数据集没有进行随机打乱等，但在实验中具体情况各有差异。

```
D:\developlAPP\python3.10.4\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in label
  _warn_prf(average, modifier, msg_start, len(result))
```