数据挖掘与大数据分析

技术报告

姓名：宋书波

2020数据科学与大数据技术

摘要：本次作业要求从*UCI*数据集库中下载两个数据集（*IRIS*和乳腺癌威斯康星州数据集）。第一项任务是使用这些数据集进行分类和回归任务，并使用随机森林和*AdaBoost*算法，比较同一数据集上不同模型的效果，得到不同大小数据集对回归器性能的影响。第二项任务是使用乳腺癌数据集训练一个*Stacking*方法模型，并与其他模型进行比较。

1、引言

本次作业旨在掌握机器学习中的常见算法——随机森林、AdaBoost和Stacking方法，并对两个数据集进行分类和回归任务。通过完成作业，我们可以：

1. 熟悉UCI数据集仓库，了解IRIS和Breast Cancer Wisconsin (Diagnostic) Data Set两个数据集；
2. 掌握Scikit-learn包中实现随机森林和AdaBoost分类器以及回归器的方法，并能够进行分类和回归任务；
3. 了解Stacking方法的基本思想，利用Scikit-learn包中实现该方法，并将其应用于Breast Cancer数据集中，训练一个Stacking模型；
4. 通过对数据集的分析、模型训练和测试等步骤，提高对机器学习算法的理解和应用能力，加深对机器学习的认识。

以上内容对于我们将来进一步学习和应用机器学习算法具有重要的指导意义，也可以提升机器学习算法实践能力和解决实际问题的能力。

2、算法

随机森林

随机森林算法(Random Forest)是集成学习(Ensemble Learning)中的一种分类与回归方法。它是通过建立多个决策树来完成任务，每个决策树的训练数据子集和特征子集都是随机选择的，这样可以减少过拟合的风险。

随机森林算法的详细流程

1. 从原始数据集中有放回地抽取n个样本作为一个训练集，不包含其余样本的袋装数据集称为“袋装样本(bagging sample)”；
2. 随机选择k个特征，其中k远小于总特征数，在这k个特征中分别选出最优特征，将之作为节点分裂的依据；
3. 每个决策树通过递归二分的方式生成子节点，直至无法继续划分或达到设置的停止条件；
4. 重复步骤1-3，建立m棵决策树；
5. 对新数据进行预测时，将所有决策树的预测结果进行投票或平均，得到最终的预测结果。

随机森林实现算法

分类公式

假设训练集$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 x_i 是输入特征向量，y_i 是对应的类别标签。我们假设共有 K 个类别。

1. 对于每棵树 T，重复以下步骤：
   - 从训练集 D 中有放回地进行自助采样（bootstrap sampling），得到一个新的训练集 D_T。
   - 用 D_T 训练一棵决策树 T。
2. 对于每个样本 x，进行如下操作：
   - 对于每棵树 T，使用 T 进行预测，得到一个类别标签预测 y_T。
   - 统计所有树的预测结果，选择预测结果最多的类别作为最终的预测类别。

回归公式

假设训练集 $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 x_i 是输入特征向量，y_i 是对应的目标值。

1. 对于每棵树 T，重复以下步骤：
   - 从训练集 D 中有放回地进行自助采样（bootstrap sampling），得到一个新的训练集 D_T。
   - 用 D_T 训练一棵决策树 T。
2. 对于每个样本 x，进行如下操作：
   - 对于每棵树 T，使用 T 进行预测，得到一个目标值预测 y_T。
   - 统计所有树的预测结果，计算平均值作为最终的预测目标值。

Adaboost

Adaboost（自适应增强）是一种集成学习方法，用于解决分类和回归问题。它通过训练一系列弱分类器（或回归器），每个弱分类器都在前一个分类器的误分类样本上进行调整，最终将它们组

合成一个强分类器（或回归器）。

Adaboost算法的基本步骤

1. 初始化训练集的权重：将每个样本的权重初始化为相等值，通常为 $1/n$，其中 n 是训练集中样本的数量。
2. 对于每个弱分类器 $t = 1, 2, \ldots, T$，重复以下步骤：
   - 使用当前样本权重训练一个弱分类器 h_t。弱分类器可以是任何具有较高错误率的简单分类器，例如决策树桩（仅考虑单个特征的简单决策树）。
   - 计算弱分类器 h_t 的分类误差率 ϵ_t，即在加权训练集上被错误分类的样本的权重之和。
   - 计算弱分类器 h_t 的权重系数 $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$。较低错误率的弱分类器将获得较高的权重系数，表示其对最终分类器的贡献较大。
   - 更新样本权重：将正确分类的样本的权重降低，并将错误分类的样本的权重增加。这样，下一个弱分类器将更关注前一个分类器错误分类的样本。
3. 对于每个样本，进行如下操作：
   - 对于每个弱分类器 t，使用弱分类器 h_t 进行预测，得到一个类别标签预测 y_t。
   - 统计所有弱分类器的预测结果，通过加权投票或加权求和的方式得到最终的预测类别。

最终，Adaboost将多个弱分类器的预测结果进行加权组合，生成一个强分类器的预测结果。它的关键思想是通过逐步调整样本权重来关注于被错误分类的样本，从而提高整体分类性能。Adaboost在实际应用中表现出色，并且在许多机器学习任务中都取得了很好的效果。

Adaboost分类公式

假设训练集 $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 x_i 是输入特征向量，y_i 是对应的类别标签。我们假设共有 K 个类别。

1. 初始化训练集的权重 $w_i = \frac{1}{n}$，其中 n 是训练集中样本的数量。
2. 对于每个弱分类器 $t = 1, 2, \ldots, T$，重复以下步骤：
   - 使用当前权重 w_i 训练一个弱分类器 h_t。
   - 计算弱分类器 h_t 的分类误差率 $\epsilon_t = \sum_{i=1}^{n} w_i \cdot 1(h_t(x_i) \neq y_i)$，其中 $1(\cdot)$ 是指示函数。
   - 计算弱分类器 h_t 的权重系数 $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$。
   - 更新样本权重 $w_i = \frac{w_i \cdot \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z}$，其中 Z 是规范化因子，使得权重之和为1。
3. 对于每个样本 x，进行如下操作：
   - 对于每个弱分类器 t，使用弱分类器 h_t 进行预测，得到一个类别标签预测 y_t。

- 统计所有弱分类器的预测结果，通过加权投票或加权求和的方式得到最终的预测类别。

Adaboost回归公式

假设训练集 $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 x_i 是输入特征向量，y_i 是对应的目标值。

1. 初始化训练集的权重 $w_i = \frac{1}{n}$，其中 n 是训练集中样本的数量。
2. 对于每个弱回归器 $t = 1, 2, \ldots, T$，重复以下步骤：
   - 使用当前权重 w_i 训练一个弱回归器 h_t。
   - 计算弱回归器 h_t 的加权平均绝对误差（WAPE） $\epsilon_t = \frac{\sum_{i=1}^{n} w_i \cdot |h_t(x_i) - y_i|}{\sum_{i=1}^{n} w_i}$。
   - 计算弱回归器 h_t 的权重系数 $\alpha_t = \frac{1}{2} \ln\left(\frac{1}{\epsilon_t (1 - \epsilon_t)}\right)$。
   - 更新样本权重 $w_i = \frac{w_i \cdot \exp(-\alpha_t \cdot |h_t(x_i) - y_i|)}{Z}$，其中 Z 是规范化因子，使得权重之和为1。
3. 对于每个样本 x，进行如下操作：
   - 对于每个弱回归器 t，使用弱回归器 h_t 进行预测，得到一个目标值预测 y_t。
   - 统计所有弱回归器的预测结果，通过加权求和的方式得到最终的预测目标值。

以上是Adaboost回归的详细公式，其中权重的更新和弱回归器的权重系数是Adaboost算法的关键步骤。Adaboost回归通过逐步调整样本权重和弱回归器的权重来逼近目标值，从而提高整体回归性能。

Stacking

Stacking（堆叠）是一种集成学习方法，通过组合多个基学习器的预测结果来生成最终的预测结果。它的核心思想是将多个基学习器的输出作为新的特征，然后使用一个元学习器来整合这些特征，产生最终的预测结果。

Stacking方法的基本步骤

1. 第一阶段（训练基学习器）：
   - 将训练集分成多个折（例如K折交叉验证）。
   - 对于每个折 k，使用除了第 k 折以外的所有折的数据作为训练集。
   - 使用不同的基学习器（例如决策树、支持向量机、随机森林等）训练模型，并对第 k 折的数据进行预测。得到基学习器在第 k 折数据上的预测结果。
2. 第二阶段（训练元学习器）：
   - 将第一阶段得到的基学习器的预测结果作为新的训练集，每个样本对应于基学习器的预测结果和真实的目标值。

- 使用一个元学习器（例如逻辑回归、线性回归、神经网络等）对新的训练集进行训练，目标是学习如何将基学习器的预测结果组合起来得到最终的预测结果。

3. 预测阶段：

- 对于新的样本，先使用每个基学习器进行预测，得到基学习器的预测结果。
- 将基学习器的预测结果作为新的特征输入给训练好的元学习器，并使用元学习器生成最终的预测结果。

Stacking的优势在于能够利用多个不同的基学习器的优点，通过组合它们的预测结果来提高整体的预测性能。它能够适应不同类型的数据和模型，具有较好的灵活性和泛化能力。然而，Stacking也可能面临一些挑战，如模型的选择、训练数据的划分等，需要谨慎处理。

Stacking方法公式

假设我们有一个训练集 $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 x_i 是输入特征向量，y_i 是对应的目标值。

1. 第一阶段（训练基学习器）：

- 将训练集 D 分成 K 折，每折为 D_k。
- 对于每个折 k，重复以下步骤：
  - 使用除了第 k 折以外的所有折的数据作为训练集。
  - 使用基学习器 $M_1, M_2, \ldots, M_m$ 训练模型，得到基学习器在第 k 折数据上的预测结果 $P_k^1, P_k^2, \ldots, P_k^m$，其中 P_k^i 是模型 M_i 在第 k 折数据上的预测结果。

2. 第二阶段（训练元学习器）：

- 将第一阶段得到的基学习器的预测结果作为新的训练集 $\tilde{D} = \{(P_1^1, P_1^2, \ldots, P_1^m, y_1), (P_2^1, P_2^2, \ldots, P_2^m, y_2), \ldots, (P_n^1, P_n^2, \ldots, P_n^m, y_n)\}$。
- 使用元学习器 M_{meta}（如逻辑回归、线性回归等）训练元模型，得到最终的元学习器模型。

3. 对于新的样本 x，进行如下操作：

- 对于每个基学习器 M_i，使用 M_i 进行预测，得到基学习器的预测结果 $P^{(i)}$。
- 使用元学习器 M_{meta} 对基学习器的预测结果进行组合，得到最终的集成学习模型的预测结果。

以上是Stacking方法的详细公式，其中包括两个阶段：第一阶段用于训练基学习器，第二阶段用于训练元学习器。Stacking通过将基学习器的预测结果作为输入，训练一个元学习器来组合基学习器的输出，从而提高整体的预测性能。

3、实验及结果分析

3.1 数据集

从 UCI dataset repository 中下载以下数据集[1]

- IRIS
- Breast Cancer Wisconsin (Diagnostic) Data Set **IRIS** 数据集包含三个品种的鸢尾花（setosa，versicolor和virginica）的测量值每个样本包含四个特征：萼片长度，萼片宽度，花瓣长度和花瓣宽度（均以厘米为单位）。数据集包含150个样本，每个品种50个样本。 **D4: Breast Cancer Wisconsin (Diagnostic) Data Set** 数据集包含569个乳腺肿瘤细胞样本，其中212个是恶性的，357个是良性的.每个样本包含30个特征，包括肿瘤细胞的半径，纹理，平滑度，紧密度，分界点等等数据集中所有特征的值都已经归一化到0到1之间。

3.2 文件详情

```
IRIS_分类.py: 其中包含随机森林、Adaboost方法在IRIS数据集上的使用。
IRIS_回归.py: 其中包含随机森林、Adaboost方法在IRIS数据集上的使用。
WDBC_分类.py: 其中包含随机森林、Adaboost方法以及Stacking方法在WDBC数据集上的使用。
WDBC_回归.py: 其中包含随机森林、Adaboost方法在WDBC数据集上的使用。
其中，Stacking方法使用的基分类器为：KNN、决策树以及SVM，元学习器采取神经网络。
```

3.3 实验结果分析

数据集处理

IRIS数据集中D_{train}:D_{test}=75%:25% WDBC数据集中D_{train}:D_{test}=80%:20% WDBC数据集中D_{train}:D_{test}=90%:10% 回归任务中数据处理：

```
IRIS数据集：去除最后一列，第一列作为target,其余列作为data.
WDBC数据集：去除id, diagnosis列,fractal_dimension_worst列作为target,其余列作为data.
```

IRIS数据集分类

采取随机森林以及Adaboost方法 实验结果如下图所示：

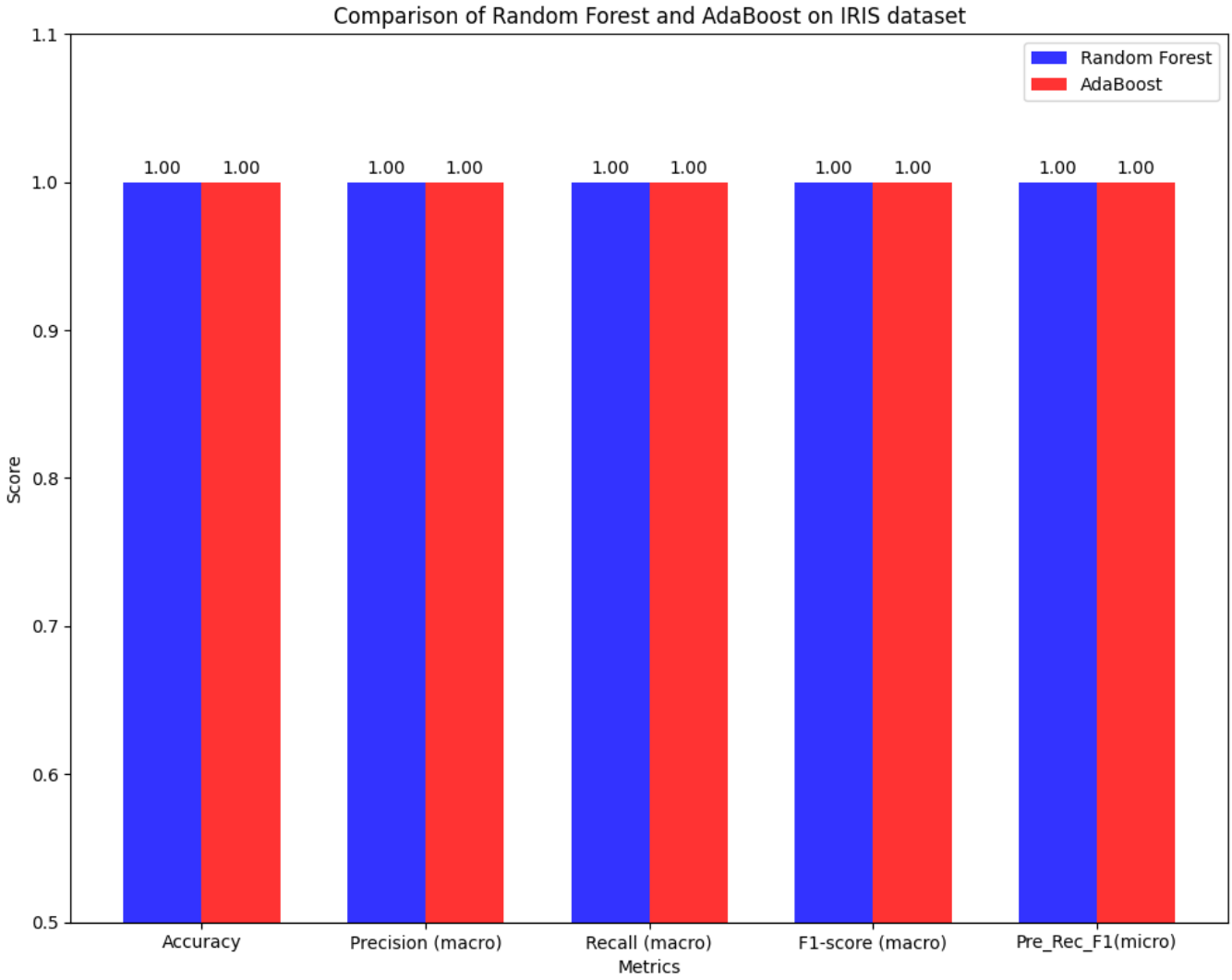

图1 IRIS分类（随机森林、Adaboost）

最终实验得到的结果是随机森林方法以及Adaboost方法在IRIS数据集上进行分类任务效果都较好，评价指标非常高，这可能与数据集的规模有关。

WDBC数据集分类

采取随机森林、Adaboost以及Stacking方法 实验结果如下图所示：

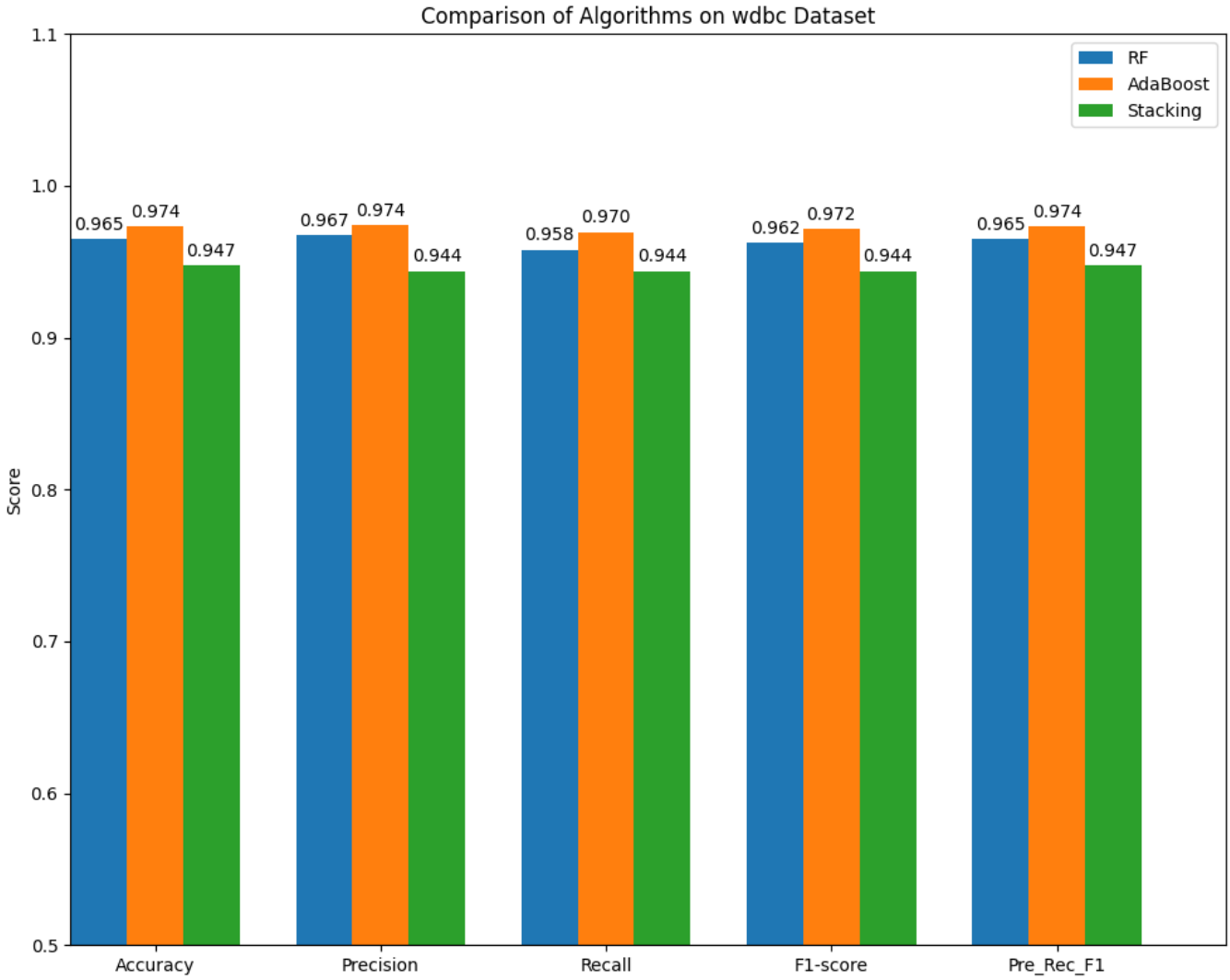

图2 WDBC分类（随机森林、Adaboost、Stacking）

实验结果得到的基本结论是在WDBC数据集上Adaboost性能>随机森林>Stacking,Adaboost表现最好，Stacking相对最差，但是总体来看三者的性能都较好。

IRIS数据集回归

采取随机森林以及Adaboost方法 实验结果如下图所示：

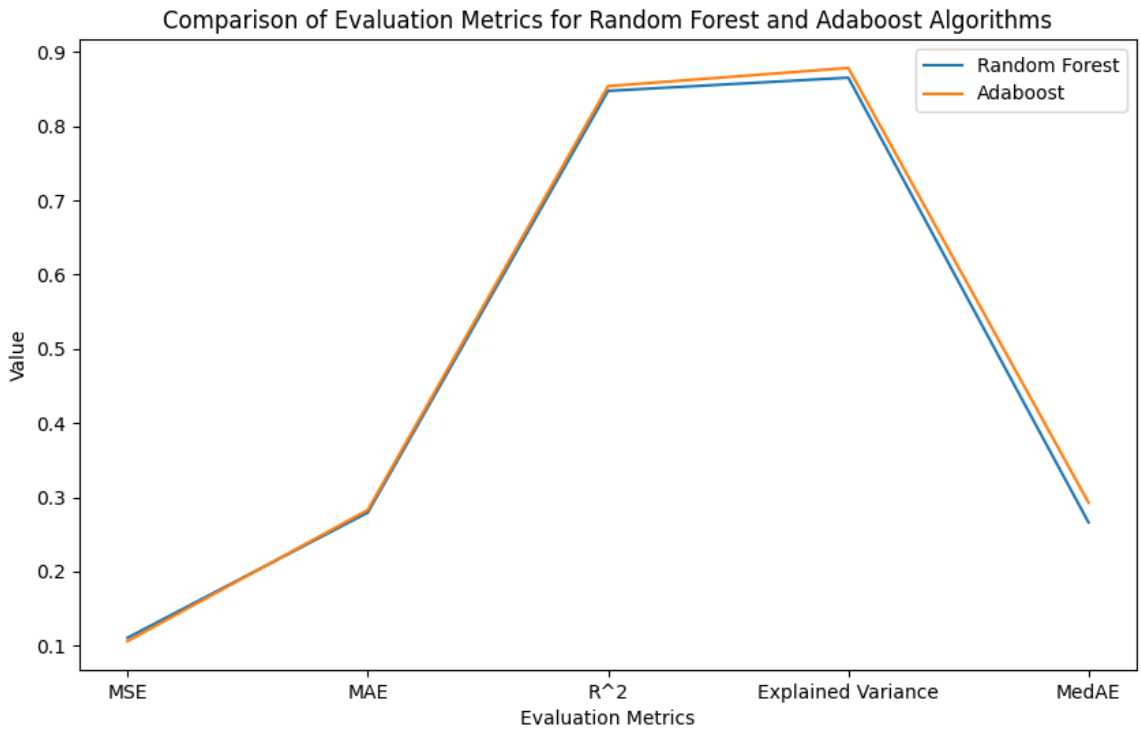

图3 IRIS回归（随机森林、Adaboost)

实验结果得到的结论是随机森林回归器与Adaboost回归器性能相差不大，随机森林的性能要稍好于Adaboost,得到的解释方差和中位绝对误差均小于Adaboost.随机森林相较于Adaboost回归器来说，在规模较小的数据集上可能效果要稍好于Adaboost.

WDBC数据集回归

采取随机森林以及Adaboost方法

WDBC数据集中D_{train}:D_{test}**=80%:20%** 评价指标如下:

```
随机森林算法
均方误差（MSE）: 6.648829813517526e-05
平均绝对误差（MAE）: 0.004625507894736838
决定系数（R²）: 0.8135500105425925
解释方差分数: 0.813600107584113
中位绝对误差（MedAE）: 0.003579249999999999

Adaboost算法
均方误差（MSE）: 7.859403212123333e-05
平均绝对误差（MAE）: 0.005608262005441805
决定系数（R²）: 0.7796024733461095
解释方差分数: 0.7844200491965807
中位绝对误差（MedAE）: 0.004253399481068676
```

实验结果如下图所示：

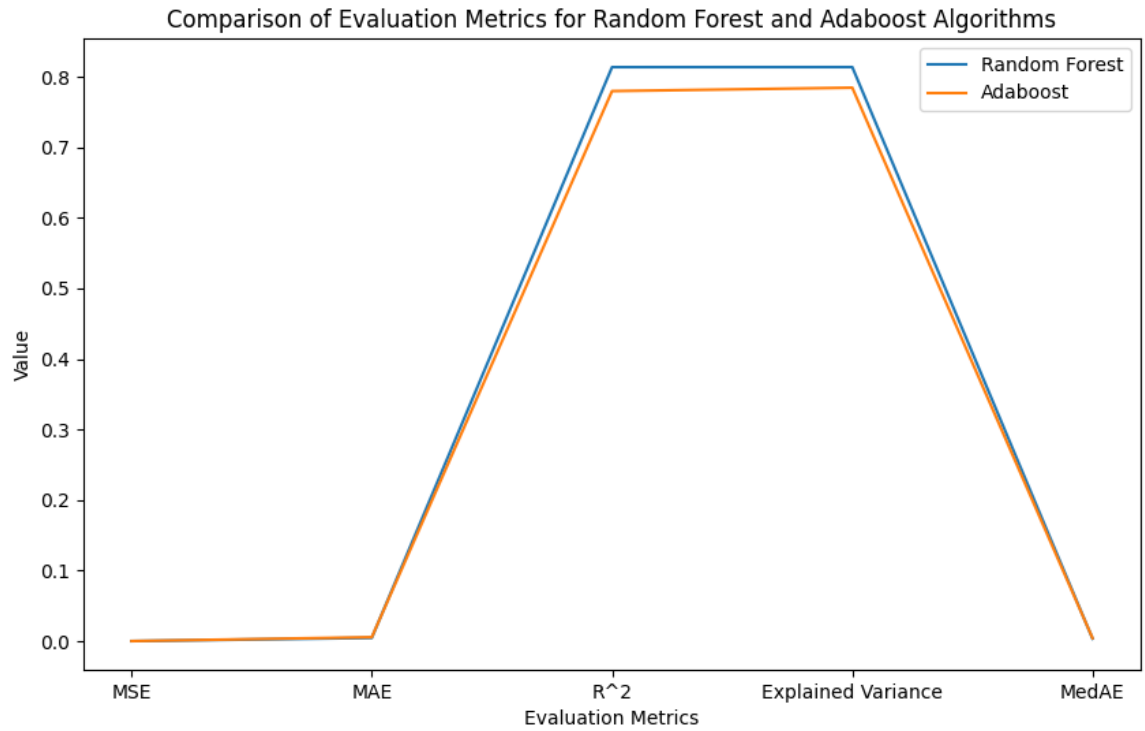

图4 WDBC回归（随机森林、Adaboost)

WDBC数据集中D_{train}:D_{test}=90%:10% 评价指标如下:

```
随机森林算法
均方误差（MSE）: 2.933823755578941e-05
平均绝对误差（MAE）: 0.004187694736842104
决定系数（R²）: 0.8605772376456842
解释方差分数: 0.8654848476116914
中位绝对误差（MedAE）: 0.0032643000000000533

Adaboost算法
均方误差（MSE）: 4.7291880492087496e-05
平均绝对误差（MAE）: 0.005376938772303012
决定系数（R²）: 0.7752569627743066
解释方差分数: 0.8028345038592419
中位绝对误差（MedAE）: 0.004890691823899437
```

实验结果如下图所示：

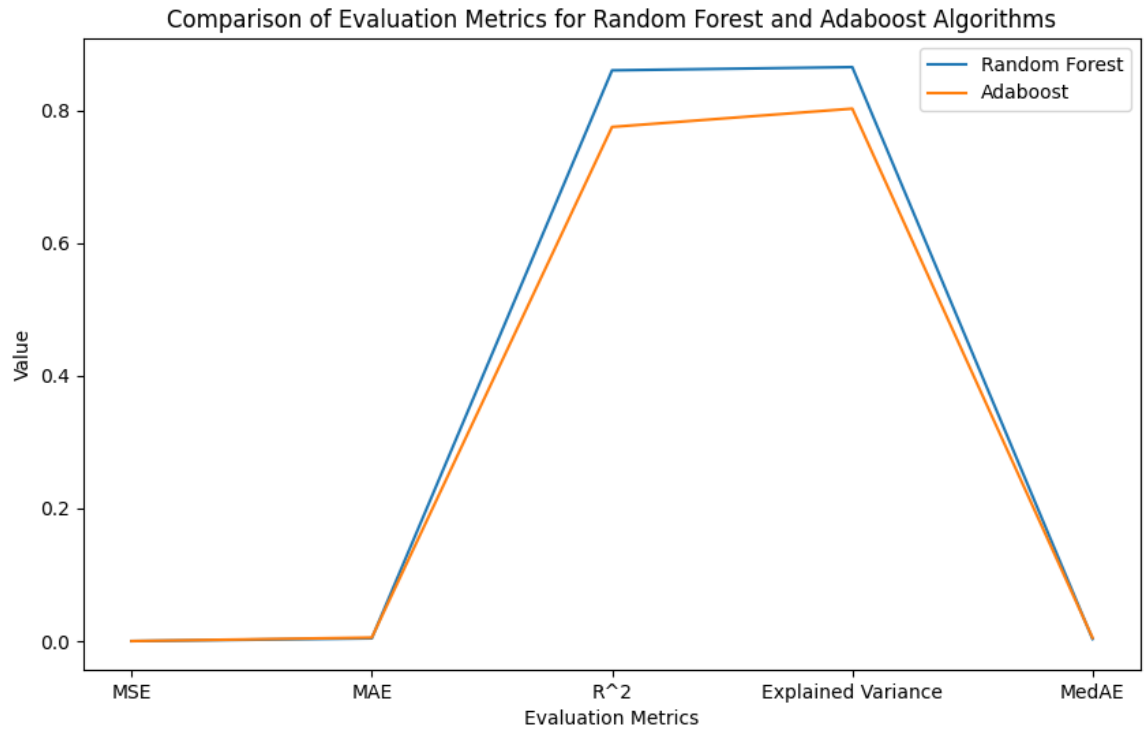

图5 WDBC回归（随机森林、Adaboost）

以下是将不同数据集划分比例下，随机森林和Adaboost算法的性能指标进行对比的表格。数据集划分比例分别为0.9：0.1和0.8：0.2。

算法	划分比例	均方误差（MSE）	平均绝对误差（MAE）	决定系数（R^2）	解释方差分数	中位绝对误差（MedAE）
随机森林	0.9：0.1	2.9338	0.0041876	0.86057	0.86548	0.0032643
Adaboost	0.9：0.1	4.7291	0.0053769	0.77525	0.80283	0.0048906
随机森林	0.8：0.2	6.6488	0.0046255	0.81355	0.81360	0.0035792
Adaboost	0.8：0.2	7.8594	0.0056082	0.77960	0.78442	0.0042533

表1 WDBC回归数据集划分影响

从表格中可以观察到，随机森林和Adaboost算法在不同数据集划分比例下的性能指标有所差异。

- 对于随机森林算法，随着训练集比例的降低（从0.9：0.1到0.8：0.2），MSE和MAE值略有增加，R^2和解释方差分数略有下降，而中位绝对误差（MedAE）的变化不明显。
- 对于Adaboost算法，随着训练集比例的降低（从0.9：0.1到0.8：0.2），MSE和MAE值略有增加，R^2和解释方差分数略有提高，中位绝对误差（MedAE）的变化不明显。

这些差异可能是由于不同的数据集划分比例导致训练集和测试集之间的样本分布不同所致。通常情况下，更大的训练集比例可能有助于模型更好地捕捉数据的特征，提高模型的拟合能力和泛化能力。

在给定的情况下，随机森林和Adaboost算法在两种不同的数据集划分比例下表现出略微不同的性能。随机森林在较大的训练集比例（0.9：0.1）下表现更好，具有更低的MSE和MAE，更高的R^2和解释方差分数，以及较小的中位绝对误差（MedAE）。原因可能是因为随机森林在更多的训练样本上有更好的机会学习数据的特征，从而提高了性能。

Adaboost算法在较小的训练集比例（0.8：0.2）下略微优于较大的训练集比例。它表现出稍微更高的R^2和解释方差分数，但MSE和MAE值略高。原因可能是因为Adaboost算法在较小的训练集上更容易进行加权调整，以减小错误样本的影响，并提高整体性能。

总体而言，适当的数据集划分比例对于算法的性能具有一定的影响，但最终结果还取决于数据集的特征和算法的选择。在实践中，可以通过交叉验证等技术来评估不同的划分比例，并选择表现最佳的比例来训练和评估模型。

4、结论

本次作业中，主要是关于随机森林、Adaboost以及Stacking等模型。其中Adaboost采取决策树作为基分类器，Stacking采取的基学习器为KNN、SVM以及决策树，元学习器为神经网络。最终得到结论如下：相同数据集的分类情况下，数据规模较大，Adaboost性能>随机森林>Stacking,Adaboost表现最好，Stacking相对最差，但是总体来看三者的性能都较好；数据规模较小的情况则是随机森林以及Adaboost性能相近。相同数据集的回归情况下，随机森林的效果要好于Adaboost回归器。同一数据集的不同划分比例的情况下,数据集划分比例越大，模型的拟合能力以及泛化性能就越好。

参考文献

[1].Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.