Introduction to Machine Learning
Ensemble Learning Report

姓名: 任一

学号:2018011423

ry18@mails.tsinghua.edu.cn

2020 年 5 月 18 日

实验环境	
操作系统:	Windows10 家庭版 18362.72
Python 版本:	Python 3.7.3

1 实验概述

在本次作业中，我自己实现了 Bagging 和 Adaboost 这 2 种集成学习算法，并尝试了 DTree, SVM, MLP 这 3 种基学习器。此外由于 Bagging 算法易于并行，除了串行的 Bagging 外，我还实现了并行化的 Bagging 算法，提升了训练速度。

本实验的运行方法是，在代码所在文件夹下，运行 python3 ensemble.py, 根据命令行的提示输入是否在本地测试、需要运行的集成学习方法、基学习器类型、基学习器数量，然后即可得到相应的运行结果。详尽的命令行提示有利于第一次的使用者方便地运行。

```
(base) D:\Tsinghua\2020Spring\MachineLearning\HW\HW_EnsembleLearning>python ensemble.py
Please input exec mode, input 0 for local train and test, 1 for output predicted file: 0
Please choose base estimator, input 0 for Decision Tree, 1 for SVM, 2 for MLP: 0
Please choose ensemble method, input 0 for Serial Bagging, 1 for Adaboost, 2 for Parallel Bagging: 2
Please input ensemble number: 5
Vectorizing text with BOW model...
train_matrix size:[176000, 10000]
test_matrix size:[44000, 10000]
============================== Ensemble Info ==============================
Ensemble Method: Parallel Bagging
Mode: Local Test Mode
Ensemble Number: 5
Base Estimator Info: DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=11,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort='deprecated',
                     random_state=None, splitter='best')
===========================================================================
Start ensemble training...
Training finished, time cost = 48.97s
RMSE: 1.0281225307993482
```

图 1: 详尽的命令行提示与训练信息

2 实验思路

2.1 文本特征选取

在本实验中，我使用了 Bag Of Words 模型，对 Amazon Reviews 中的 summary 和 reviewText 进行了向量化。具体来说是调用了 sklearn 的 CountVectorizer 模块，选取文本中出现**频数最高**的 10000 个词，并去除停用词。这样的处理可以有效限制训练中涉及的词汇数量，提高训练速度，同时在一定程度上提高训练准确率。

为了测试选词数量对训练效果的影响，我使用最大深度为 11 的 DTree 基学习器，选取不同数量的特征词，得到的实验结果如下：

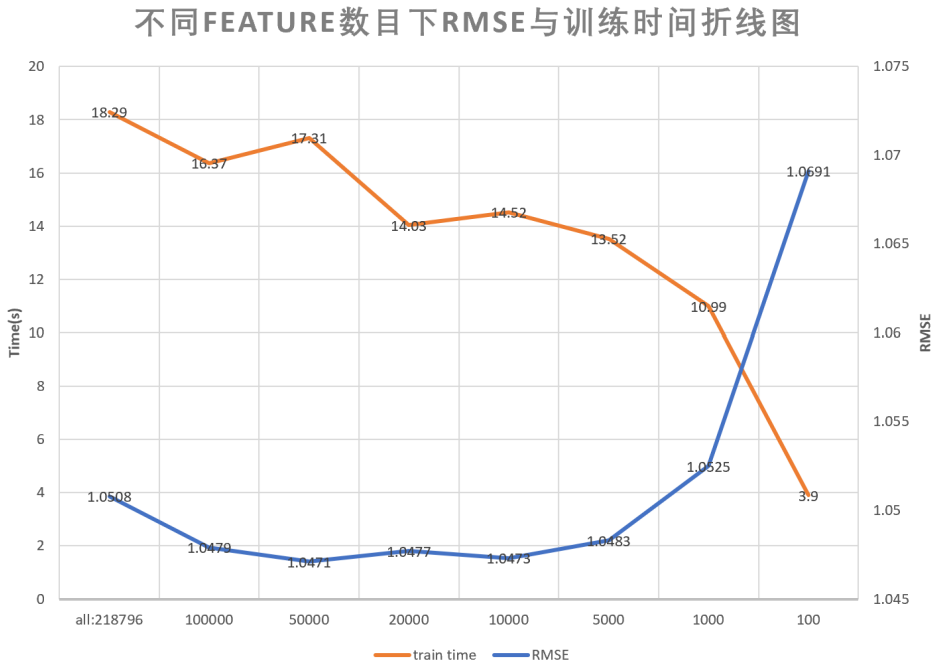

图 2: 不同数量特征下，RMSE 与训练时间变化折线图

从图中可以看出，选择词频最高的 10000-50000 个特征词，可以取得较小的 RMSE，特征词选取过多或过少都会影响效果。这可以理解为，选取特征词太多易引入噪声，选取特征词太少难以全面把握文本的特征。此外训练时间也基本随着特征词汇数量下降而下降。综合 RMSE 和训练时间，我选取了 10000 作为特征词的数量。

2.2 基学习器选取

本次实验中，我调用了 sklearn 中的 SVM 和 DTree，同时也实现了 MLP 这三种基学习器。

对于 DTree 来说，其优势在于训练速度快、可解释性强、实现简单易于理解等。但是 DTree 很容易陷入过拟合。为了解决这个问题，我的解决方案是**限制 DTree 的最大深度**，经过一定尝试，我发现在最大深度为 11 时，本地测试的 RMSE 是最小的。此外训练时间也随着 DTree 最大深度增加而增加。综合考虑 RMSE 与训练时间，我设置 DTree 的最大深度为 11. 测试结果如下：

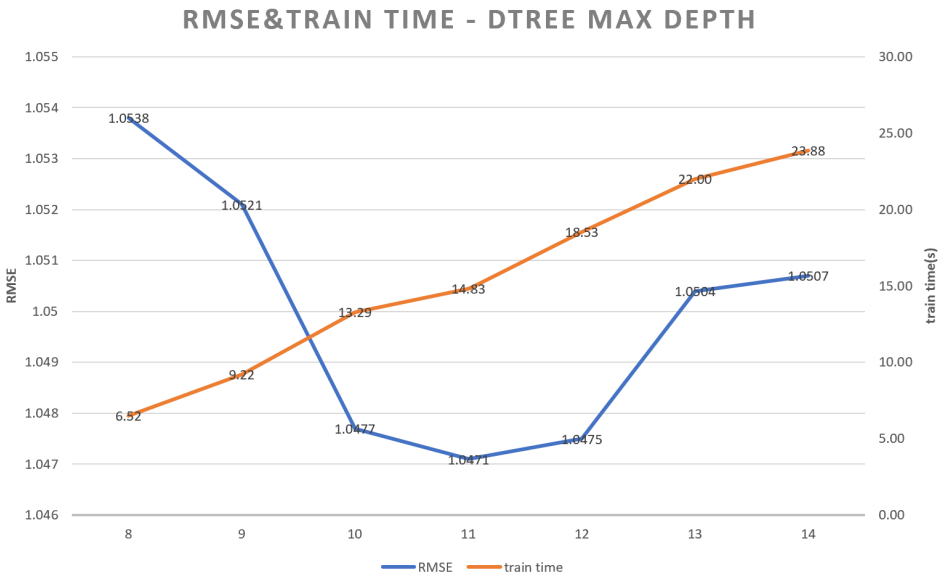

图 3: 最大深度不同时，DTree 的 RMSE 和训练时间

对 SVM 来说，其优势在于理论基础扎实、泛化能力强、善于处理高维数据、内存占用小等等。但是其缺点也很明显，例如训练速度慢、对噪声敏感。为了提升 SVM 的训练效率，我使用的了 sklearn 的 LinearSVR, 该模块使用了线性核函数，底层实现相对普通的 SVR 算法更为高效。此外，为了提升 SVM 对噪声的容错性，我通过引入**松弛变量**，给 SVM 带来了**" 软间隔"**，即允许 SVM 在一些样本上出错。经过一定的参数调试，我认为当松弛变量 $\epsilon = 0.30$ 时，SVM 在本地测试中 RMSE 最小。

2.3 集成学习算法实现

在本实验中，我实现了 Bagging 算法的串行和并行版本，以及 Adaboost 算法。

在集成时中，我分别尝试了集成分类任务和回归任务。经过尝试我发现，回归任务得到的准确率会比分类任务稍高一些。例如下图中，我分别尝试了集成不同数量的做分类和回归的 DTree(最大深度为 11), 得到的 RMSE 如图所示，可以看出，集成回归任务的效果会更好一些，因此我最终采用了集成回归任务的基分类器。

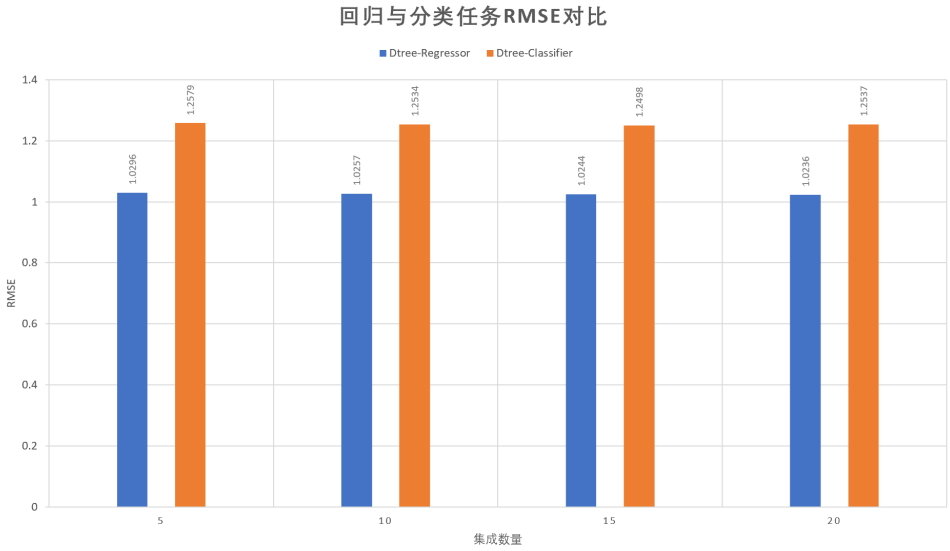

图 4: 分类任务与回归任务在不同集成数量下的 RMSE 对比

对于 Bagging 算法而言，其关键思想在于，通过 Bootstrapping 抽样，从初始数据集中每次可以产生不同的数据集进行训练，从而增大了**数据样本的扰动**。从偏差-方差分解的角度看，Bagging 算法通过这样的数据扰动，主要是能够降低学习器预测的**方差**，理论上，这种数据扰动对决策树、神经网络等对训练样本变化较为敏感的基学习器效果较为显著，而对 SVM, KNN, Naive Bayes 这样对训练样本变化不敏感的学习器效果可能不甚明显。[1]

此外，由于 Bagging 算法便于并行，我利用 concurrent.futures 模块，实现了并行化的 Bagging 算法，提升效率。经过测试，在服务器上若要使用串行 Bagging 算法集成 1000 个 SVM，需要 40h 的训练时间，若使用 80 线程的并行 Bagging 算法，训练时间仅需 1h，速度提升较为明显。

对于 Adaboost.M1 算法而言，其关键思想在于通过调整训练数据的权重，可以更好地专注于训练出错的样本。从偏差-方差分解的角度来看，Adaboost 算法通过这样的权重调整，主要能够降低学习器预测的**偏差**，因此能够从一个弱学习器训练出一个效果较好的集成学习器，例如 Adaboost 集成决策树桩即可得到很好的效果. [2] 但是 Adaboost 由于在训练过程中更加专注训练集中预测出错的样本，我认为 Adaboost 存在一定过拟合的风险，尤其是在集成较强的学习器的时候，例如 DTree, SVM, MLP 等等，这一点在后续对实验数据的分析中也会体现。

Adaboost 算法中需要对样本赋予权值，并且在训练过程中调整权值。在此处我没有使用带权的样本进行训练，即 re-weighting 的做法，而是使用了 re-sampling 的做法，即每次训练时，以样本的权值为概率，抽取训练的样本。这样的实现方法与 re-weighting 没有本质区别，且实现简单易行。

[1]此处分析参考周志华《机器学习》8.3 节 Bagging 与随机森林的内容。

[2]此处分析参考周志华《机器学习》8.2 节 Boosting 的内容。

不过 Adaboost.M1 更适合集成做分类任务的基学习器，对于回归任务的基学习器，训练过程中的加权错误率 ϵ_i 难以界定。因此通过资料的查找，我扩展了 ϵ_i 的定义，使之可以进行回归任务。[3]

3 实验结果与分析

3.1 不同集成方法下，学习器效果分析

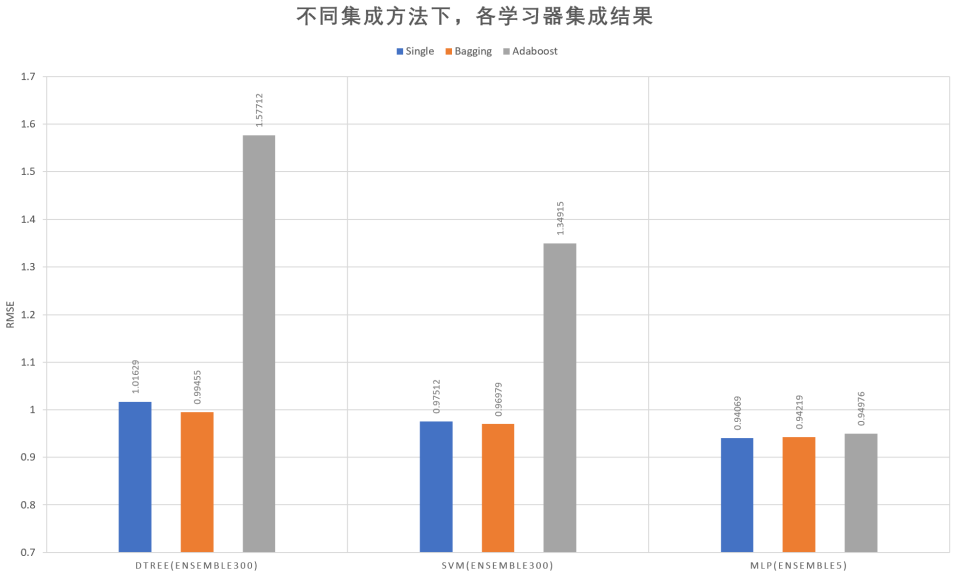

图 5: 不同集成方法和不同基学习器下的 RMSE 值对比

从集成学习的方法来看，Bagging 集成 300 个基学习器时，能够在 DTree 和 SVM 上取得优于个体学习器的结果，而 Adaboost 集成 300 个基学习器的结果相较个体学习器却明显变差。

为了排除是我的 Adaboost 算法实现问题，我套用 sklearn 的 Adaboost 算法并将集成个数同样设置为 300，得到的结果与我的 Adaboost 结果接近，说明这不是我的算法的问题。通过进一步思考，我认为这是在上述 Adaboost 测试中，我集成了 300 个学习器，这个集成数量过多。由于 Adaboost 算法会对训练集的样本权值进行不断调整，更加关注于训练中出错的样本，这可能导致集成过多时，Adaboost 算法出现严重的过拟合现象。关于 Adaboost 算法效果与集成个体学习器数量的关系，我将在 3.2 的分析中展开。

从各学习器之间的对比来看，MLP 的效果最好，SVM 次之，DTree 效果位居最后。我认为这与模型的表达能力有关。MLP 表达能力较强，同时由于我只使用了单层 MLP，每个特征词的数量作为每个神经元的输入，而每个神经元都会有一个权值。权值绝对值较大的词对输出结果影响较大。举例来说，我们可以认为'excellent' 这个词的权值为正并且绝对值很大，'awful' 这个词的权值为负并且绝对值很大，而'eat' 这个词的权值的绝对值就可能很小，对预测结果影响不大。由于 MLP 自身表达能力很强，并且该场景也很适合 MLP，所以 MLP 的效果明显优于其他两个学习器。同时 SVM 的对高维空间的数据表达能力和拟合能力也较强，因此效果居第 2 位也是合理的。

[3]此处参考了https://www.cs.utexas.edu/~dpardoe/papers/ICML10.pdf

3.2 集成学习器数量对 **Adaboost** 算法影响

下图为在 Adaboost 集成结果与集成学习器的数量关系的折线图。

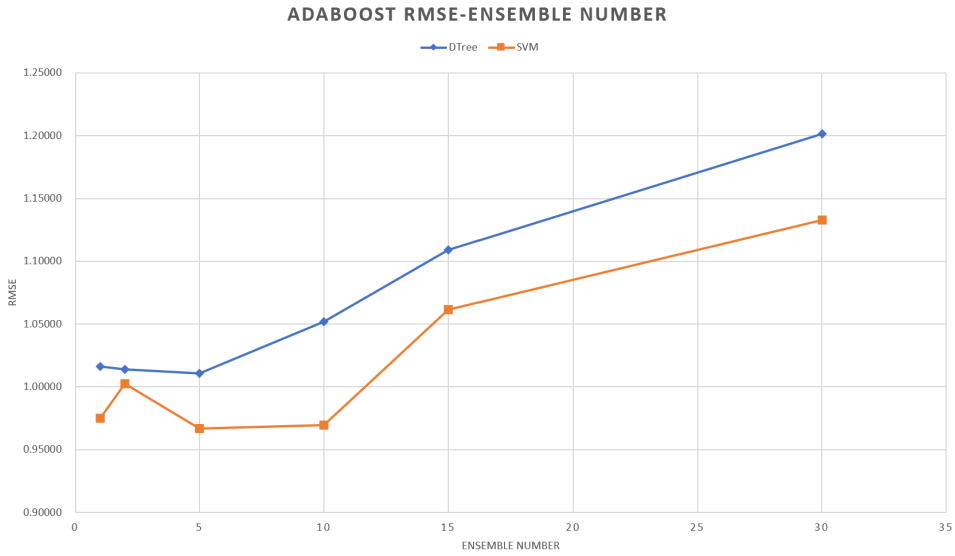

图 6: Adaboost 算法中 RMSE 与集成学习器数量

从总体来说，Adaboost 能够在基学习器数量较少时，能够有一定集成的效果，但是当学习器数量过多时，RMSE 会明显上升，可以理解为是陷入了过拟合。从方差-偏差角度来分析，Adaboost 算法能够减小针对训练集的预测偏差，但也有一定的过拟合风险。

从不同的基学习器角度来看，使用 DTree 做基学习器的 Adaboost 算法很容易陷入过拟合。当基学习器数量为 5 时 RMSE 最低，此后再增加基学习器数量，RMSE 反而会上升。我认为这在一定程度上也反映了 DTree 自身就容易陷入过拟合的特点。

而对于使用 SVM 做基学习器的 Adaboost 算法，其过拟合现象相较 DTree 就有一定缓解，在基学习器数量在 5-10 时，集成效果最好，继续增加学习器数量 RMSE 也同样会上升。我认为这与 SVM 本身相较 DTree 有较好的稳定性和泛化能力有关。

3.3 集成学习器数量对 **Bagging** 算法影响

下图为在 Bagging 集成结果与集成学习器的数量关系的折线图。

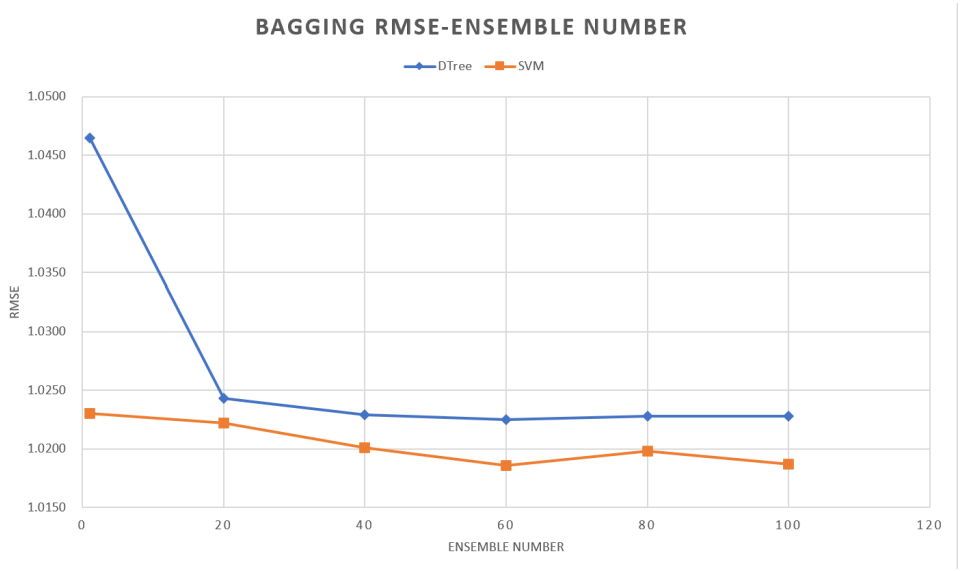

图 7: Bagging 算法中 RMSE 与集成学习器数量

从总体来看，Bagging 算法的效果，随着学习器数量增加没有特别明显的变化，RMSE 有小幅下降，但并没有像 Adaboost 一样陷入过拟合。

我认为这和 Bagging 算法本身的特性有关，Bagging 算法主要靠 Bootstrapping 方法，每次给个体学习器提供独立的不同训练集，可以认为每个个体学习器是独立的，没有相互的直接联系和干扰。而 Adaboost 算法会随着训练的进行，更加关注训练出错的样本，这在一定程度上就可能造成过拟合。

从方差-偏差的角度来看，Bagging 算法通过数据的扰动，可以降低预测的方差，使集成学习器效果随着集成数量增加收敛于一个较为稳定的值，这与折线图中的趋势也是较为吻合的。

从个体学习器的特征来看，可以明显看出，DTree 在 Bagging 的集成下，当个体学习器数量增加时，RMSE 下降明显，尤其是从 1-5 这一段来看。而 SVM 相对来说在 Bagging 集成后效果虽有小幅上升，但不甚明显。这样的现象也充分反映了 DTree 易受样本扰动的不稳定特性和 SVM 不易受到样本扰动的稳定性。

4 Kaggle 平台评测结果

截至 5 月 17 日 23:45，我在 Kaggle 平台上排名为第 23，RMSE 为 0.94069，使用的方法是 Bagging 集成的 MLP. 这充分展示出了 MLP 本身强大的表达能力，以及 Bagging 算法利用 MLP 对数据样本扰动较为敏感的特性，能够达到较好的集成结果。

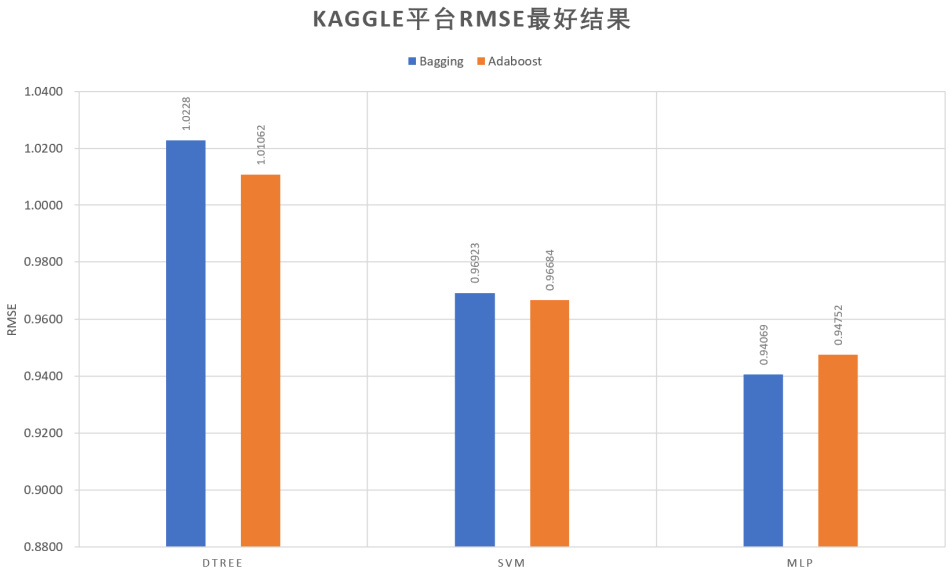

图 8: Kaggle 平台各学习算法最好效果

此外，关于课程要求实现的 DTree 和 SVM 算法，它们使用 Adaboost 集成的效果会略优于 Bagging，这可以理解为 Adaboost 可以在一定程度上降低预测的偏差，因此可以达到较好的效果。此外 SVM 效果会优于 DTree，我认为这与 SVM 自身表达能力较强有关。

5 总结

在本次实验中，我学习了手动实现 Bagging 和 Adaboost 算法，并尝试集成了不同的基学习器 (DTree, SVM, MLP). 在这个过程中，我感受到了集成学习的确能在一定程度上提升学习器的预测准确率。此外，我也结合所学到的理论知识，尝试对实验中出现的现象进行猜想和解释，并通过进一步的实验证明我的猜想。在这个过程中，我加深了对理论知识的理解，也增强了实践的能力。感谢老师和助教给予的悉心指教！