实验三 集成学习

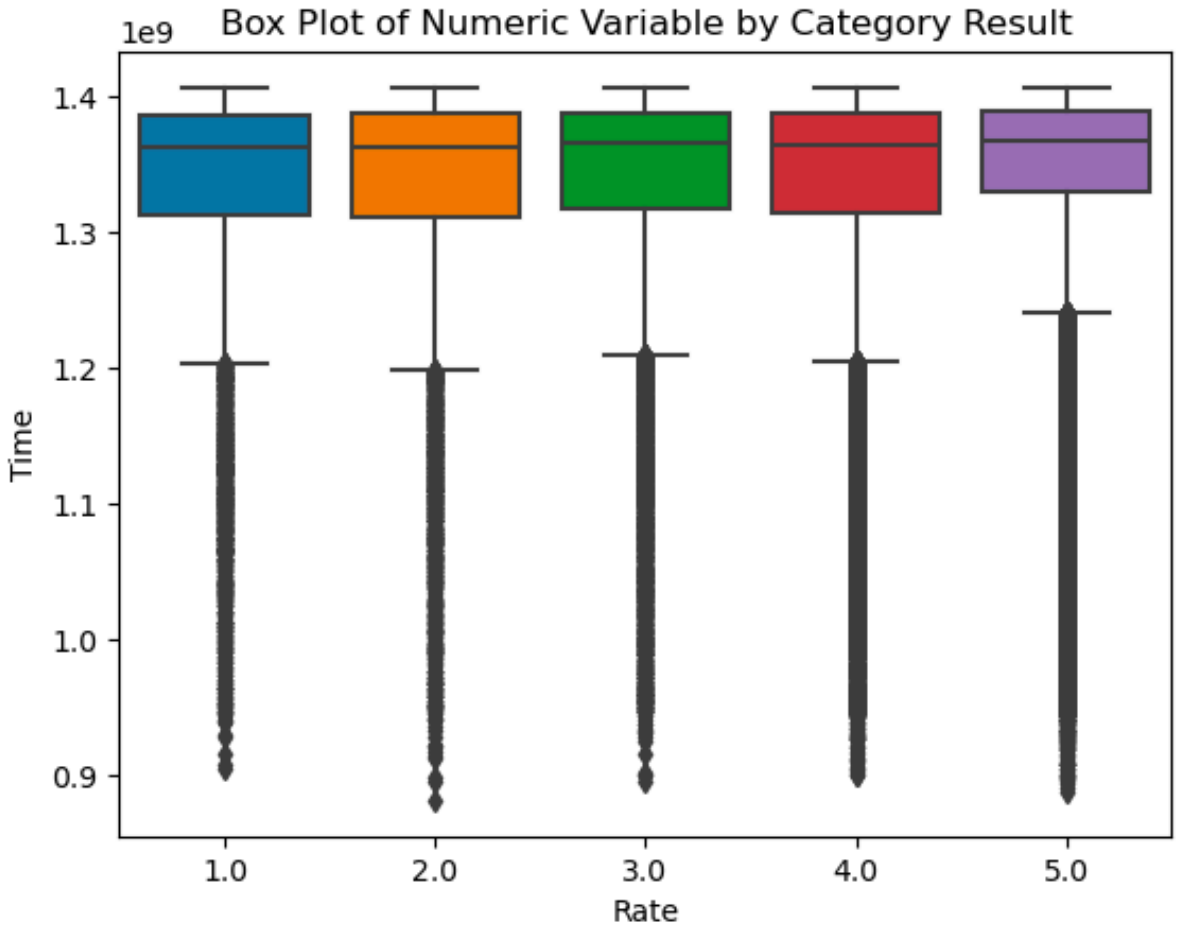
田田 经12计18 2021011048

一、数据预处理分析

在这一部分，我们对提供的数据进行预处理，将数据从原始的格式转化成能够让模型处理的数值类型。我们需要对 reviewerID，asin，unixReviewTime，summary 和 reviewText 进行处理。

通过简单统计，总共有219997条不含缺失值的评论数据，其中总共有125391个不同的用户名 ID，111697个不同的物品 ID，和5499个不同的评论时间戳，三个维度的数据都有比较多的重复。并且这三个维度本质上都是类别型变量，我们只需要两者是否相同，不存在大小关系或者是顺序关系（时间戳或许存在顺序关系），因此可以直接整数编码编号处理。为了避免编号导致的顺序关系（可能导致在 SVM 中出现问题），我们可以把编号转化成二进制对应的高维向量。

对于这三个变量的直观解释，我们认为同一个人的打分 / 同一个物品的得分应该有规律性，因此 reviewID 和 asin 可以用来判断是否是同一个人 / 同一件物品，而评论的时间戳 unixReviewTime 感觉起来应该没什么关系。但对时间戳和评论分数做相关系数计算和卡方检验之后发现，两者的相关系数是0.0377，卡方检验的 p 值是 2.31*10^-41，并不足以认为两个分布完全独立。对于时间戳和打分绘制箱线图也发现，不同打分对应的时间戳分布有着可以观察出来的差别，因此也将时间戳作为特征向量的一部分，使用 minmax 归一化处理。



对于 summary 和 reviewText，在了解了常用的处理方式，并在网络上查询了建议之后，我们决定首先对于两者进行常规的去除标点、分词、去除停用词、提取词干和词性还原操作；然后采用逆文档频率 TF-IDF 的方式整体处理数据，综合考虑每个词出现的词频和普遍性，来使得在一类文档里面经常出现但在其他文档中不怎么出现的词能够有更高的权重，从而取得更好的分类效果；最后使用 LDA 的方式，给每个 summary 和 reviewText 分配一个对应的主题，来帮助对于评分的预测。

在具体操作中，结合之前的经验（选取过多的特征词并不利于模型的表现），我们给 summary 选取了100个特征词和20个主题，给 reviewText 选取了300个特征词和20个主题。和之前的三个变量提取出来的特征向量合并到一起之后，最终输入的特征向量是475维的（summary 和 reviewText 对应的440维的特征词和主题，asin 和 reviewerID 对应的两个17维的二进制向量和 unixReviewTime 对应的一维的向量）。

然后，我们把数据按照9:1的方式随机分成训练集和测试集，准备开始训练。在这一部分观察发现，overall的评分分布并不均衡，评分整体偏重于5分这一侧，需要在设计模型的时候考虑。

二、集成学习算法实现

在这一部分，我解释我对需要实现的两种集成学习的算法的理解，并具体解释我实现的代码逻辑。

1、Bagging

Bagging 是一种并行式的集成学习方法，它通过从原始数据集中进行有放回抽样（bootstrap采样）来创建多个子集。每个子集都是通过随机抽样从原始数据中得到的，这样就使得每个子集都有一些差异。然后，针对每个子集，使用相同的学习算法进行训练。最后，将所有学习器的预测结果进行平均或者投票来得到最终的预测。

在我实现的 bagging 算法中，我预留了整体学习器数量 n_estimators ，采用的学习器类型 base_classifier 和每次抽样比例 sample_size 三个超参数。每次我们抽取一部分的样本（实际操作中是0.8），用这部分样本训练一个基分类器，然后使用多数票投票的方式决定最终的预测结果。

2、AdaBoost

AdaBoost 是一种串行式的集成学习方法，它通过迭代训练一系列弱学习器来构建一个强学习器。在每一轮迭代中，它根据前一轮的结果调整样本的权重，将更多的关注放在前一轮分类错误的样本上。每个弱学习器都会根据调整后的样本权重进行训练，然后根据其分类准确度给出一个权重，这样误分类率低的学习器会获得更大的权重。最终，将所有弱学习器的结果进行加权组合，得到最终的预测结果。

在这一部分，我们一开始忽视了课件中提及的 Adaboost 分类器和 python 库中实现的分类器都是二分类的，而我们需要处理的问题是五分类的，导致一开始预测的结果总是全都是0。

在最终的实现中，我们首先实现了二分类的 SimpleAdaBoost 分类器，然后将五分类的问题拆分成对于所有两两组合（总共10组）的分类，使用10个 SimpleAdaBoost 分类器组合来处理这个五分类的问题。模型有整体分类轮数 n_estimators 和 采用的基学习器类型 base_estimators 两个超参数。

三、分析组合表现

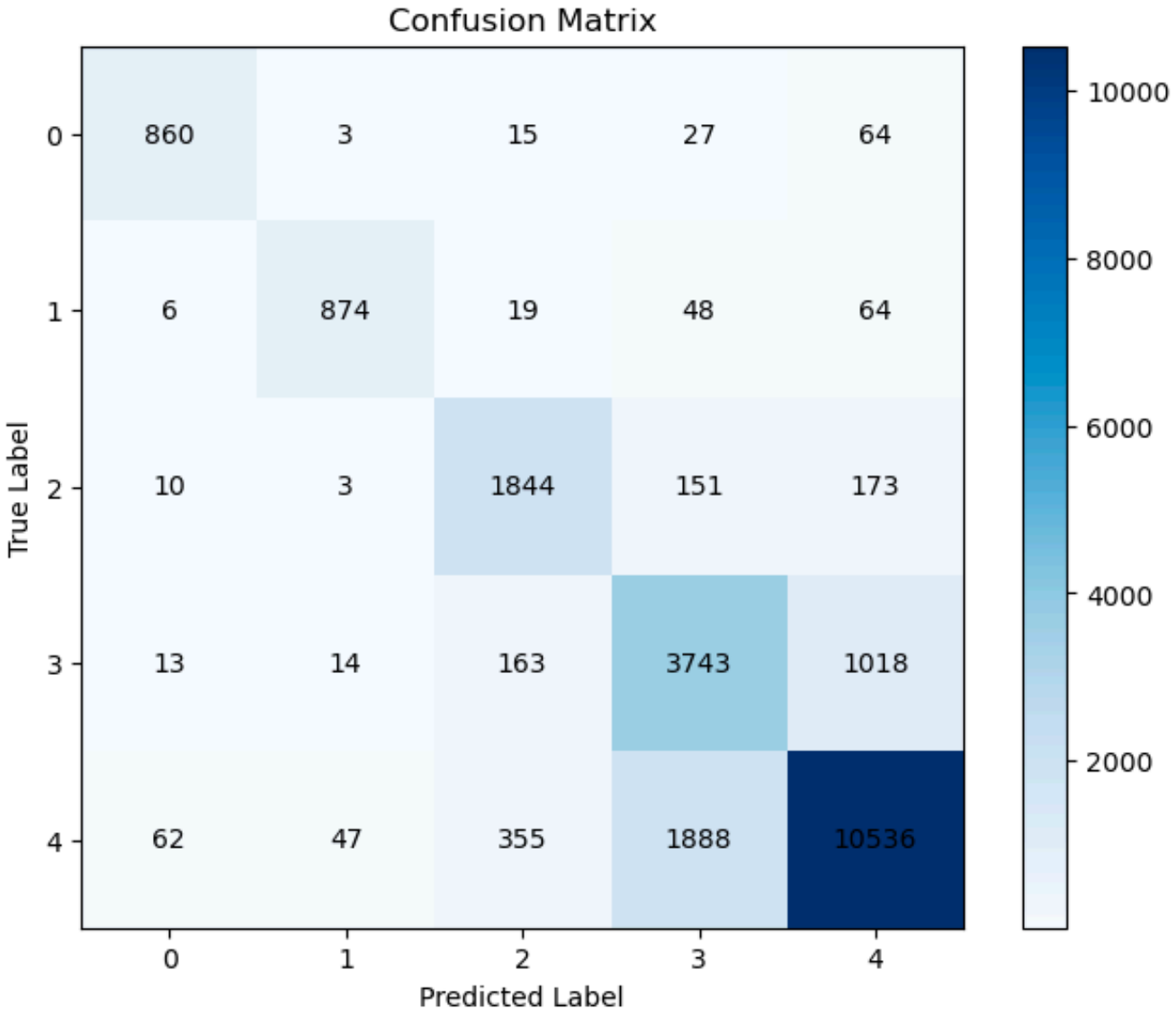
在这一部分，我们报告对于每个组合具体使用的超参数和观察，每个组合的具体性能和调试中的思考。

在模型性能和表现的衡量上，我们使用 accuracy，precision 和 recall 三个参数定量衡量模型的分类能力，使用混淆矩阵直观判断分类的情况和问题所在，使用 MAE 和 RMSE 定量衡量最终的整体误差情况。

1、Bagging和SVM组合

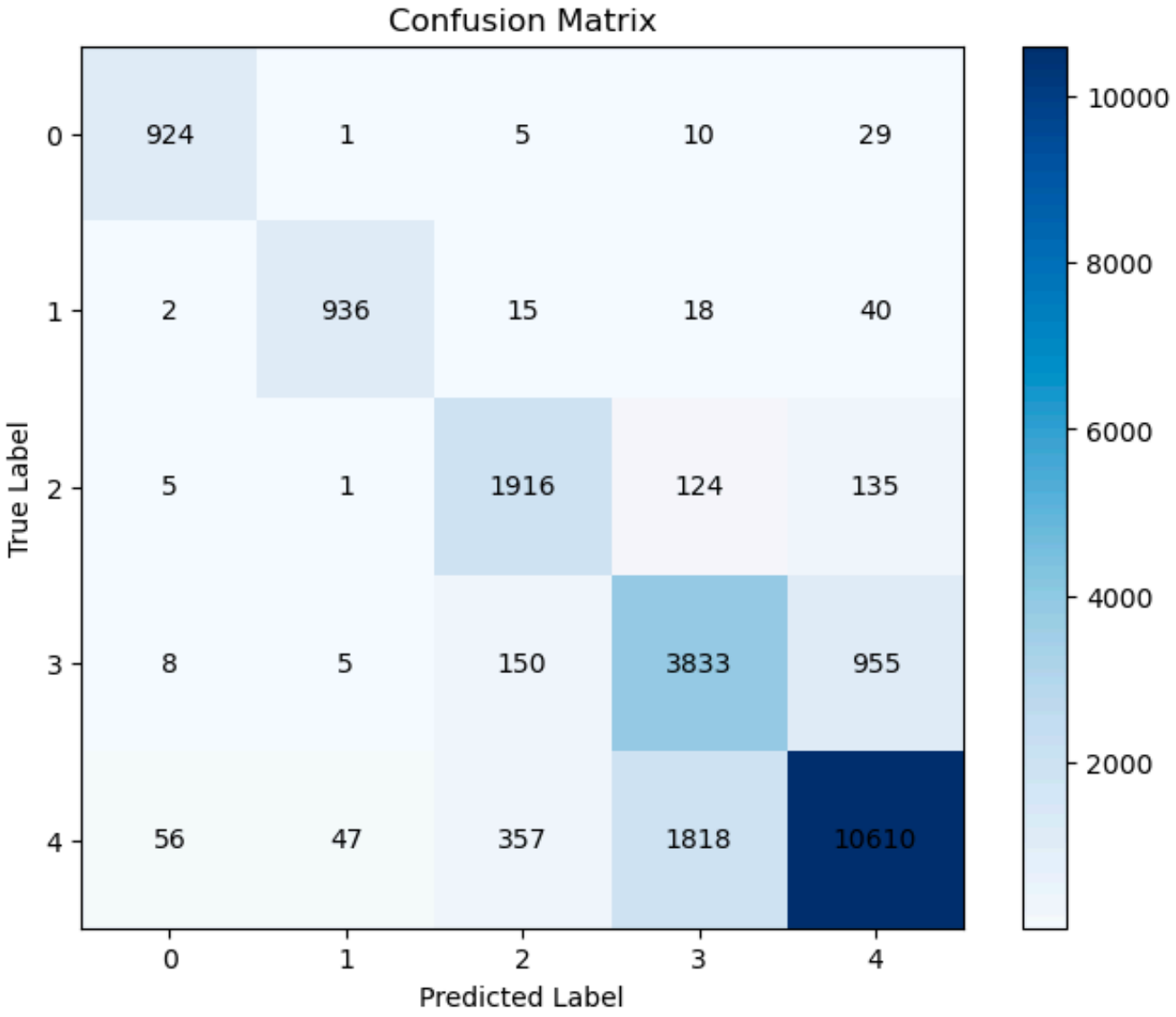
一开始，我们没有设置基分类器的参数，设定 bagging 算法总共训练15个基分类器，最终的结果是 accuracy 为64.46%，precision 为58.20%，对于SVM采用的就是默认的初始参数。

而后注意到， overall 评分更多集中在高分段，低分的比较少，因此我们需要针对不均衡的数据情况进行修正，所以调整SVM的参数，使用高斯核函数来保证比较好的分类能力，将 gamma 参数shelli设定为0.1，使用 balanced 作为处理类别分布的手段，来平衡原本不均衡的分类结果，将正则化参数设定为1来避免过拟合。同样训练15个基分类器，最终 accuracy 为81.17%，precision 为82.61%，recall 为83.42%，混淆矩阵如下：



可以看出，模型有着比较好的分类性能，也没有明显严重的误分类，问题主要发生在对于4分和5分的区分上。但我们认为想要提升模型在这一部分的表现，可能主要需要在数据预处理上下更多的功夫，调整特征词的数量和主题的数量，增加模型的结构可能影响不大。

为了验证我们的设想，我们保持参数不变，将整体的基分类器数量增加到了30，结果发现的确可以提升性能，但是整体的提升并不是非常明显（因为计算的时间增加了三倍，但所有指标的增加幅度都在5%以内）。accuracy 为82.81%，precision 是84.41%，recall 是87.11%，混淆矩阵的结果如下：

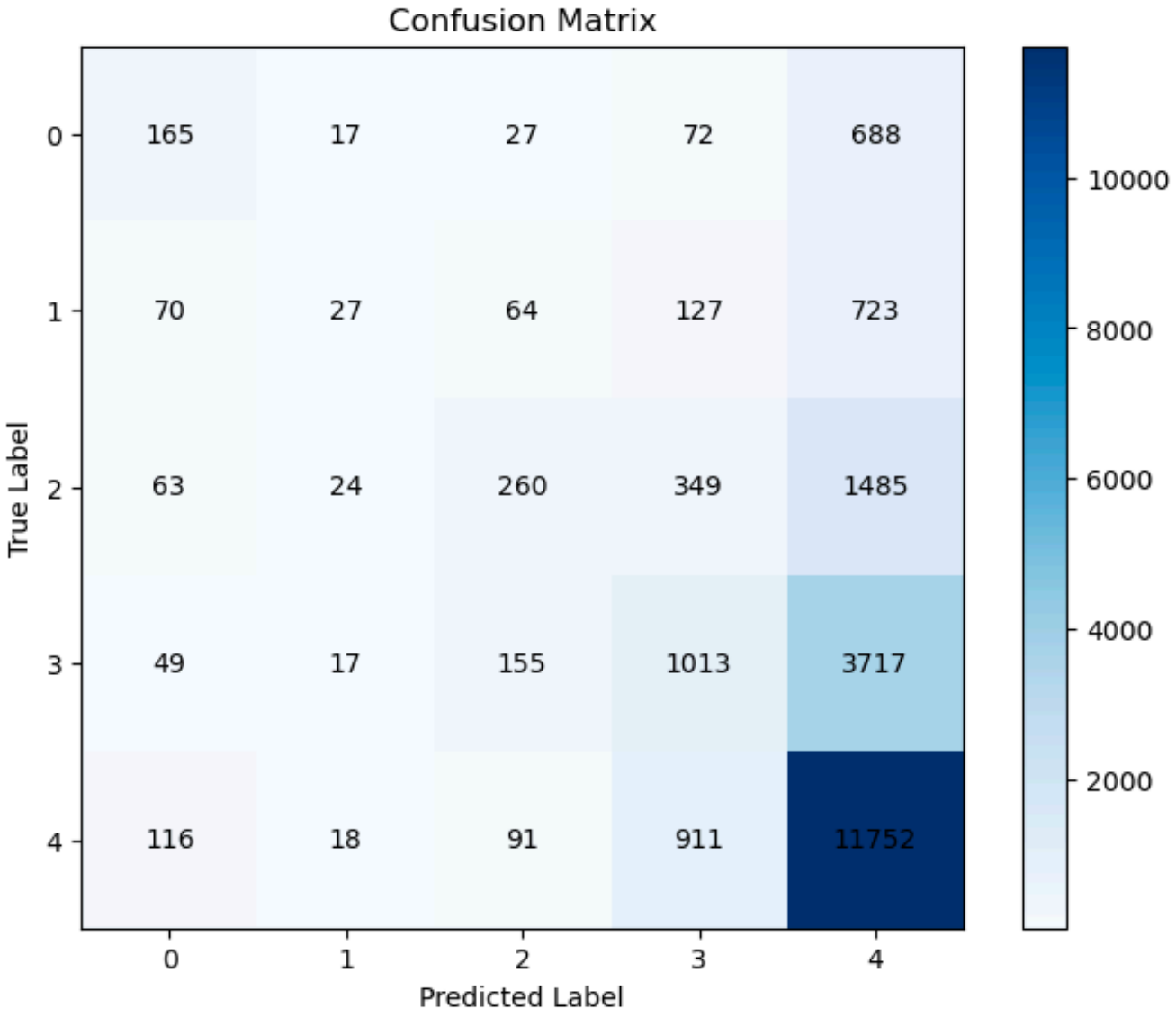


可以看出，在各个层面相较采用15个基分类器的组合都有进步。由于比较长的计算时间，无法再去具体尝试修改基分类器的数量和 SVM 基分类器的具体参数，来找到最合适的组合。

2、Bagging和Decision Tree组合

在这一部分，我们训练了30个基分类器，将 DecisionTree 的样本权重设定为 balanced 来平衡不同评分占比的失衡，并采用信息熵作为节点分类的衡量标准，同样每次抽取80%的样本数据用来训练基分类器。

最终的 accuracy 为60.08%，precision 为42.07%，recall 是28.65%，混淆矩阵的结果如下：



从中可以发现，模型的 accuracy 还算比较不错，但是 precision 和 recall 都比较低，结合混淆矩阵可以发现，模型的预测结果中五分的比例过高，将太多的1分到4分的结果预测成为了5分。虽然由于整体结果中五分的占比很高，模型取得了还可以的正确率，但是实际的分类能力并不理想。

3、AdaBoost和SVM组合

这一部分我们采用10个基分类器，采用 linear 内核的 SVM 分类器，对于权重设置采用 balanced 算法。由于这部分中 SVM 的运算速度非常慢，为了降低运算的负荷，我们使用原本就是随机抽样的训练集的前20000个样本用于训练，采用测试集的前2000个样本用来测试。

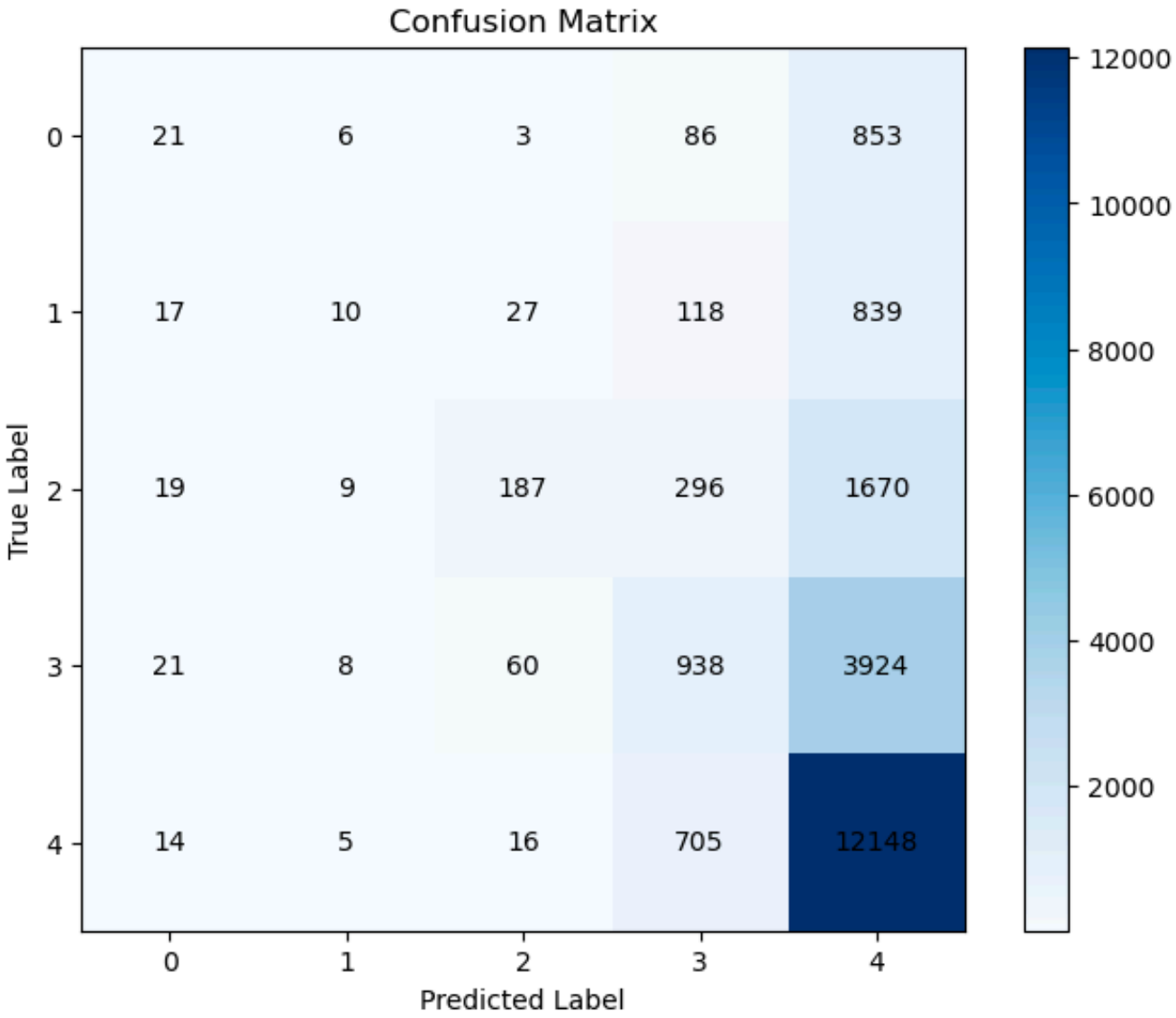
不知道为什么，无论怎么训练或者采用什么样的内核，只要训练次数比较高，最终的结果都是将所有数值预测为5分，修改参数得到的其他预测结果还不如直接预测所有值为5分的。

推测可能是数据中评价为5分的样本太多，导致了五分类的情况并不均衡，从而导致比较敏感的 AdaBoost 和 SVM 组合出现了问题。或者是数据预处理的时候，我提取了维度太高的特征向量，从而导致了这个组合的偏离。

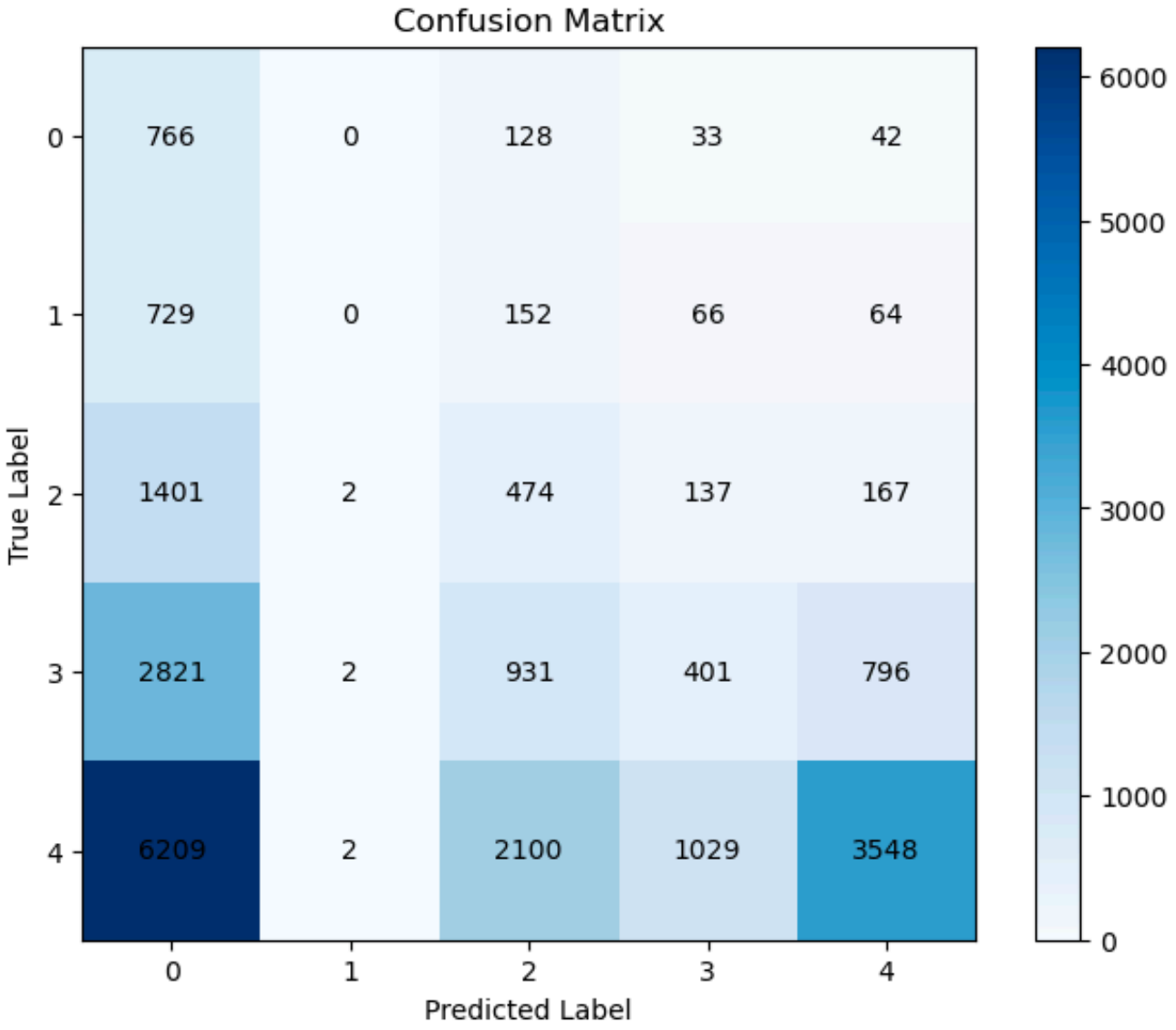
4、AdaBoost和Decision Tree组合

在这一部分，我们针对不同的决策树最大深度和整体基分类器数量尝试了两个不同的组合。

一开始，我们规定决策树的最大深度是3，总共采用10个基分类器，最终预测的结果为：accuracy 是60.47%，precision 是43.85%，recall 是24.98%，和之前分析的结果类似，基本都是预测结果为五分，所以accuracy 看起来还可以。混淆矩阵的结果也证实了这一点：



为了提升模型的性能，我们尝试增加决策树的深度并增加基分类器的数量，采用最深为5层的决策树作为基分类器，并规定训练30个基分类器。但是最终的结果非常糟糕，所有指标都接近20%，并且整体预测的结果基本都是0（即认为全都是1分），混淆矩阵如下图所示。这也是我实现的 AdaBoost 分类器遇到的一个比较典型的问题：随着基分类器越来越复杂，训练的总数越来越多，最终的预测结果就越来越接近0。我认为这应该是和模型训练前初始值实现或者每次训练后权重修正有关，但是无法解决这个问题。



5、对比分析

在这一部分，我们将整体比较四个模型的最好的表现。为了有一个比较的基准，我们引入第五个模型，这个模型无脑将所有的结果都预测为五分。五个模型预测结果的各项指标如下：

	accuracy	precision	recall	F1	MAE	RMSE
BaggingSVM	82.81%	84.41%	87.11%	85.57%	0.2168	0.5827
BaggingDecisionTree	60.07%	42.07%	28.65%	29.82%	0.6695	1.228
AdaBoostSVM	58.1%	11.62%	20%	14.70%	0.7645	1.352
AdaBoostDecisionTree	60.47%	43.85%	24.99%	24.52%	0.6833	1.258
全部预测为五分	58.58%	11.71%	20%	14.78%	0.7374	1.319

整体对比观察可以发现，Bagging 集成学习模型和 SVM 基学习器的组合有着最好的效果，达到了最高的 accuracy，precision，recall，f1 和最低的 MAE 与 RMSE；而另外两个模型的表现比较接近。除去 AdaboostSVM 之外的三个模型的结果也都超越了【全部预测为五分】这一模型，说明训练后的，除了 AdaBoost 和 SVM 组合之外的三个模型都有了最基本的分类能力。

从中我们同样可以发现，accuracy 等衡量判断能力的指标和 MAE 等衡量拟合程度的指标基本满足负相关，这也符合我们的基本认知，增加了模型结果的可信度。

从训练过程中可以发现，Adaboost 分类器对于初始化的结果非常敏感，很容易偏离到最终预测的结果全是0（1分）的情况，这也符合我们课上学习到的 Adaboost 对于噪声和异常值敏感的已知。

当我们固定集成学习算法不变，比较两种基分类器的性能的时候，可以发现在使用 Bagging 集成学习算法的时候， SVM 分类器的性能比较明显的优于 DecisionTree 的性能，推测是因为我们生成的特征向量维度比较高，更加适合 SVM 处理。而 DecisionTree 经常是在训练集上表现良好，但是在测试集中表现不佳，Adaboost 集成的 DecisionTree 经常出现过拟合使得在训练集上预测正确率100%，但是在测试集中低于【全部预测为五分】模型的情况。因此之后如果使用 DecisionTree，要注意控制模型的规模和输入数据的维度，以防止过拟合。

四、总结和反思

整体而言，我认为这次大作业中我在数据预处理的部分做的比较好，预先定性定量分析了各个变量的影响，没有出现之前到模型训练的时候才发现数据预处理有问题的情况。

但是，没有对训练的计算量有着准确的估计，经常苦等四五个小时发现还是无法训练成功只能重启。如果之后再做类似的任务的话，应该先使用规模较小的数据完整过一次流程，估计一下使用全部数据需要的运算时间，如果时间太长的话就适当减少数据量（因为增加数据量所带来的额外收益有着严重的边际效益递减，不如减少数据量从而缩短训练时间，换来更多的调试和尝试不同参数组合的机会）。