

数 据 挖 掘 实 验 报 告

实验题目 基于THUCnews的短文本分类

专业班级 计算机科学与技术 16-1 班

学生姓名 曹辰鹏

学生学号 *2019218117*

指导教师 张玉红

实验地点 第五机房

2018-2019 学年第二学期

目录

[实验1： 分类任务 2](#_Toc107077243)

[1.1 实验目的 2](#_Toc107077244)

[1.2 数据集 2](#_Toc107077245)

[1.3 实验环境 2](#_Toc107077246)

[1.4 实验内容 2](#_Toc107077247)

[1.4.1预处理 2](#_Toc107077248)

[1.4.2搭建模型训练 5](#_Toc107077249)

[1.4.3使用其他模型进行训练 5](#_Toc107077250)

[1.4.4 实验分析 9](#_Toc107077251)

[1.5 实验总结 17](#_Toc107077252)

# 实验1： 分类任务

## 实验目的

1）理解分类任务；

2）考察学生对数据预处理步骤的理解，强化预处理的重要性；

3）基模型可以调用已有的包，训练学生熟悉数据挖掘的基本框架；

4）学会多维度的对模型进行评估以及模型中参数的讨论。

## 数据集

1）新闻文本分类为中文数据集，需要进行一定的预处理，包括分词， 去停用词等。本文使用的是广泛的新闻文本分类THUCnews

2）数据中的其他问题可自行酌情处理； 数据说明：自行划分 train 和 test，一般按 7：3 划分。

## 实验环境

开发平台：Windows10 操作系统

开发环境：VScode、Python3.6.5、pandas、sklearn、matplotlib ，tensorflow等等

## 实验内容

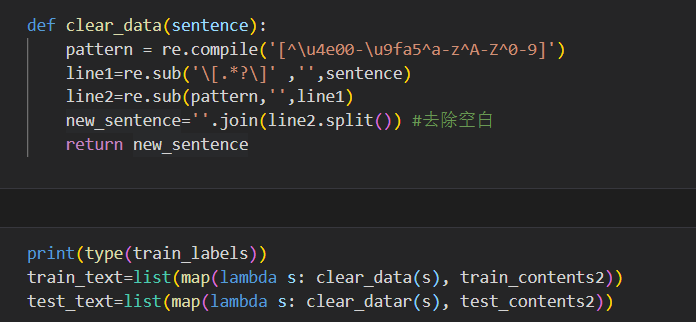
### 1.4.1预处理

#### 1.4.1.1清除特殊字符

在中文新闻文本中，文本里可能会含有一些特殊字符，比如说\*。这些字符对于数据最后的分类结果没有什么意义，甚至可能会影响最终的分类结果。所以，我们首先要进行数据的清洗。将数据转化为完全由中英问和日常标点符号的文本作为输入数据。我们直接写一个函数，用来将字符串进行匹配，及那个不匹配的内容从字符串设为空白。可以直接使用re包进行处理。最后将得到的字符串去掉空白，就可以得到我们想要的标准的数据。

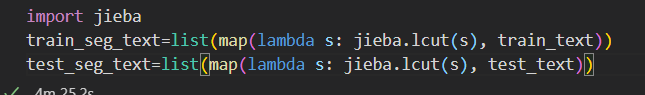
之后，我们将所有的训练数据和测试数据（二者都是列表结构，元素为文本字符串），遍历每一个文本元素，将原来的数据进行清洗。

具体代码如下：



#### 1.4.1.2中文分词

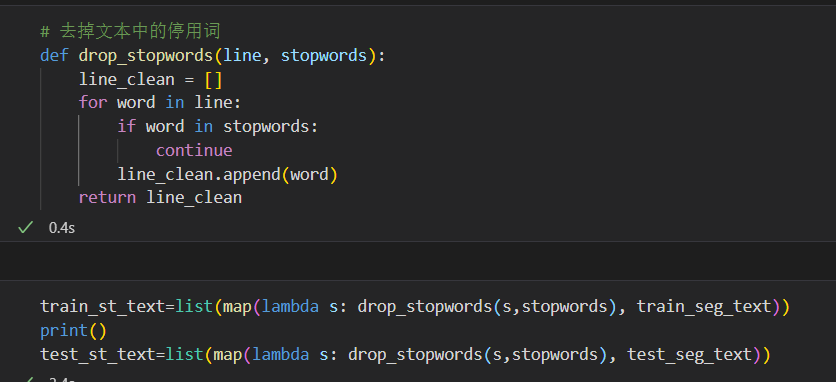
中文分词就是将一个个单独的汉字切分成一个一个单独的词，是将连续的字按照一定的规范重新排列组合成词序列的过程。中文的字每一个是由很多意思的，结成词语之后才能较为精确地表达含义。我们借用第三方库实现文本的精确模式分词。保证最后的分词结果没有冗余。



#### 1.4.1.3去除停用词

当我们使用jieba分词之后，只要是句子中出现的词语都会被划分，但是有些词语是没有实际意义的。这对于提取文本的信息并没有帮助。而且，去掉一定数量的停用词之后，还能进一步减少计算量。去除停用词还能过滤掉一些敏感信息的关键词。我们在本次实验中使用的是应用较为广泛的百度停用词表。先读取停用词表里的词语信息，之后，对每一个分词之后的列表元素（也是列表，是某一文本分词之后的结果）进行去停用词。之后得到结果。

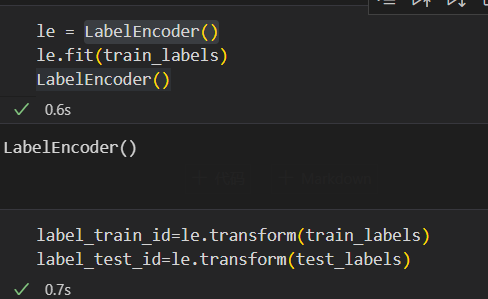
代码如下：



#### 1.4.1.4 映射标签

之前的数据都是文本和标签单独放置的，但是是按照顺序进行排列的。在之后的标签和数据对应很方便。但是标签是字符串，我们想将标签参与学习就要将标签转换成编码。同时将原来的训练数据的标签和测试数据的标签都转换为对应的编码。使用sklearn中的LabelEncoder可以很简单就做到。

代码如下：

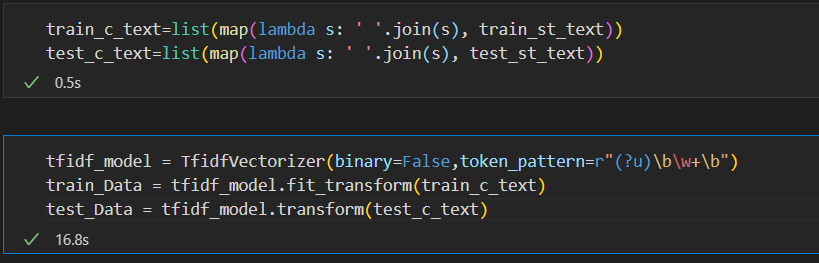


#### 1.4.1.5 转换文本表示方式

文本由多种表示方式，比如说BOW，主题模型以及词向量等。我们首先考虑的是BOW表示方式，统计每个词出现的次数。但是我们的数据超过了上万条，可以尝试考虑其他的文本信息，而不仅仅是关注本文档的信息，所以我们可以将统计结果转化为TF-IDF值的形式存在。处理好的数据相当于是进行了一次特征提取。我们准备使用sklearn中的TfidfVectorizer

实现。其中的fit\_transform()函数会先得到输入数据的词表，之后得到每个词的个数，得到TF-IDF值，转化为矩阵输出出来。

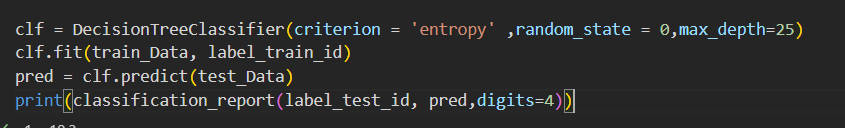
代码如下：



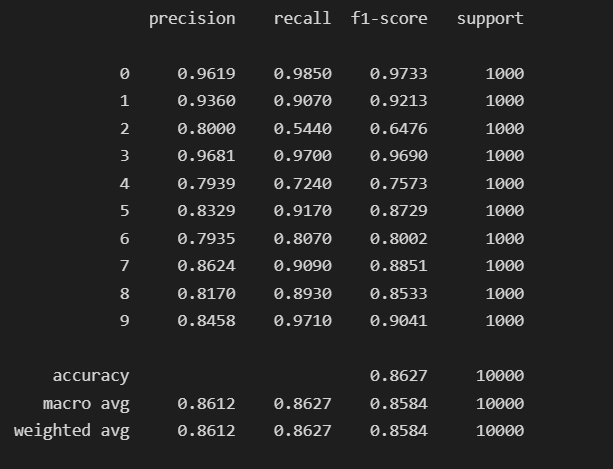
### 1.4.2搭建模型训练

我们先尝试使用比较简单的决策树模型对文本进行分类。决策树划分的标准我们使用信息熵，通过限制决策树的最大深度和最大的叶子节点数来观察实验现象。考虑到数据量较大，决策树的深度可能会比较深，我们在后续进行改变层数的过程中，以10为初始最大层数开始增长。

代码如下：



实验中，我们预先没有设置层数，层数可以任意增长。此时的结果如下：

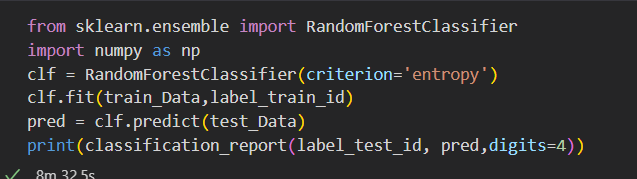


### 1.4.3使用其他模型进行训练

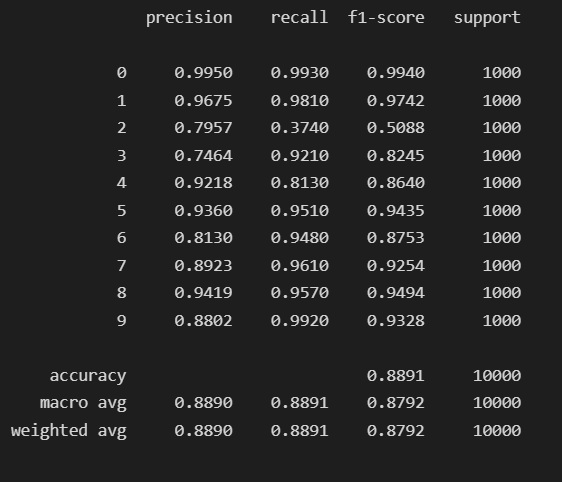
我们除了使用决策树进行训练，还加入了随机森林，贝叶斯分类等模型。

#### 1.4.3.1随机森林分类

随机森林是一个包含多个决策树的分类器，输出结果是由所有基于训练集训练得到的树分类器模型投票决定的，属于最简单的Bagging思想应用。集成学习是通过建立几个模型组合来解决单一预测问题。其工作原理为生成多个分类器，各自学习和做出预测。这谢谢预测最后会综合形成一个但预测。理论上来讲，随机森林产生的结果会随着试验次数的累加而不同，但最终的结果应该会优于决策树。本小节使用随机森林对文本进行分类，使用的分裂条件依旧是信息熵，不限制最大深度。使用sklearn库中ensemble包中随机森林模型进行试验。代码如下：



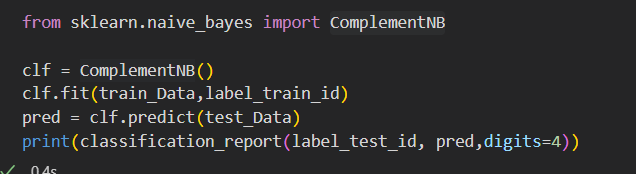
分类结果如下：

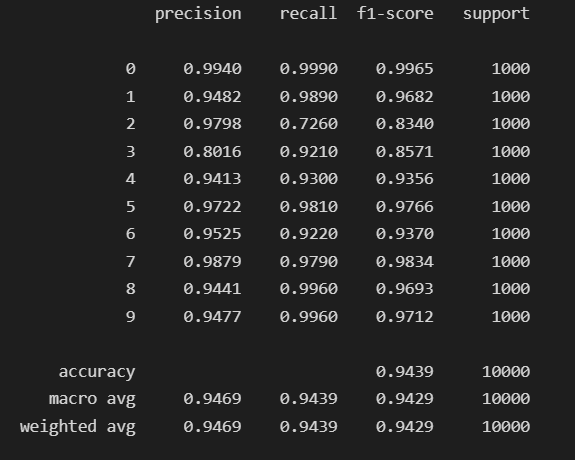


可以看到：使用这个模型的训练效果是比决策树分类模型的结果更好的。最后的f1-score比原来的增加0.02，模型的效果好上很多。但是我们的2、6类的分类效果没有得到显著的提升。

#### 1.4.3.2贝叶斯分类

本文使用的是最基础的朴素贝叶斯分类。朴素贝叶斯模型依托于贝叶斯公式，对于给出的待分类项，求解出出现的条件下各个类别出现的概率，最大的那个即为所属的类别。代码如下：



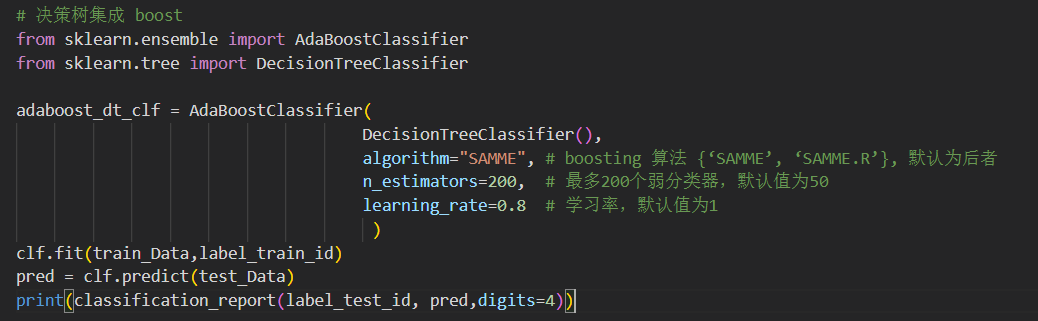


可以看到，简单的贝叶斯网络得到了0.9429的得分。比我们使用的决策树和随机森林的效果要好得多。

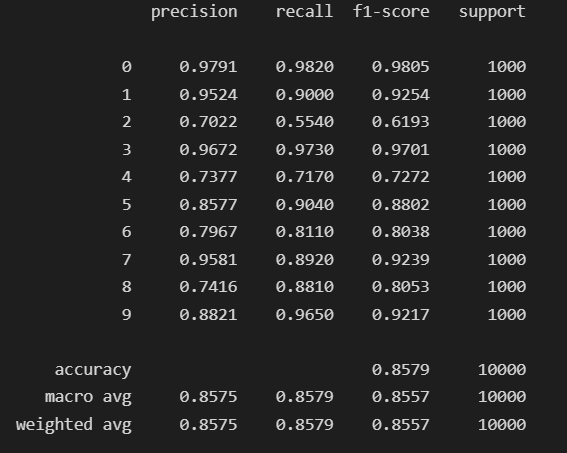
综合分析原因，一是，我们使用的是bow的表示方法，对于贝叶斯模型来说非常友好。二是，当前数据量是非常少的，对于使用贝叶斯模型也是非常适合。

#### 1.4.3.3决策树集成boost

我们在上课的时候讲过，决策树的boost集成。所以可以使用boost进行集成决策树来得到分类器。代码如下：



结果如下：

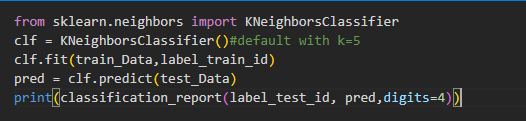


从结果可以看到，实际上的boost结果和决策树相差不大。甚至效果更差。

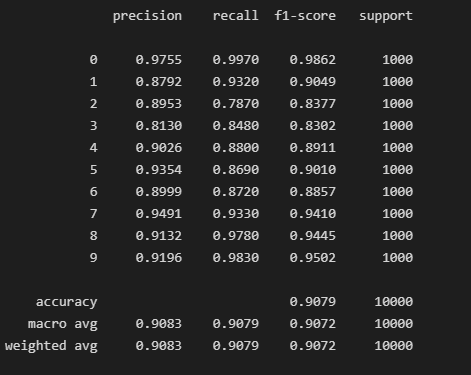
#### 1.4.3.4 KNN分类

KNN算法是一个比较简单的算法，其核心思想是一个样本在特征空间中的K个最临近的样本中最多数的那个类别就是该样本的类别，并且具有此类别上的特性。此方法通过最邻近的一个或几个样本来确定待分类样本的类别。可以使用sklearn库中的函数直接实现分类任务。

代码如下：



最后训练得到的结果如下：

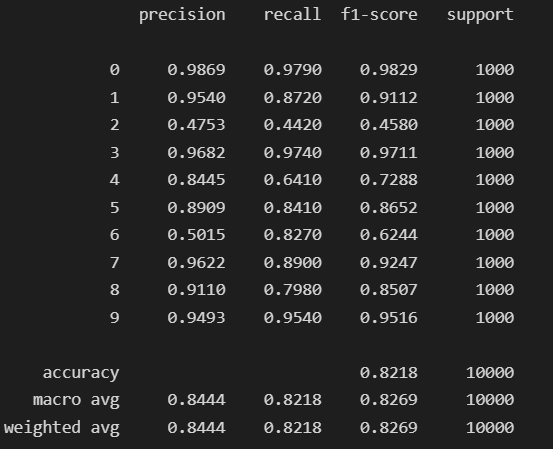
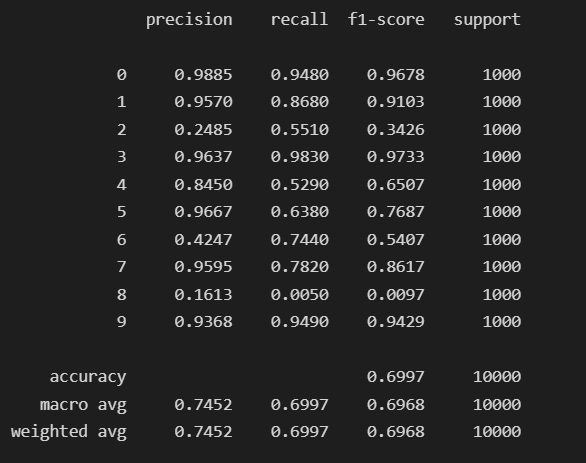


### 1.4.4 实验分析

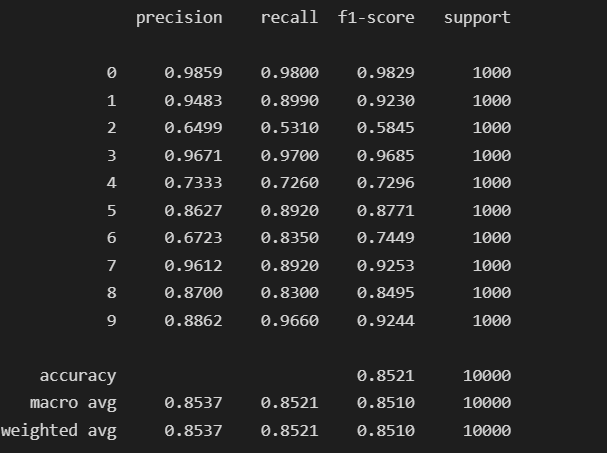
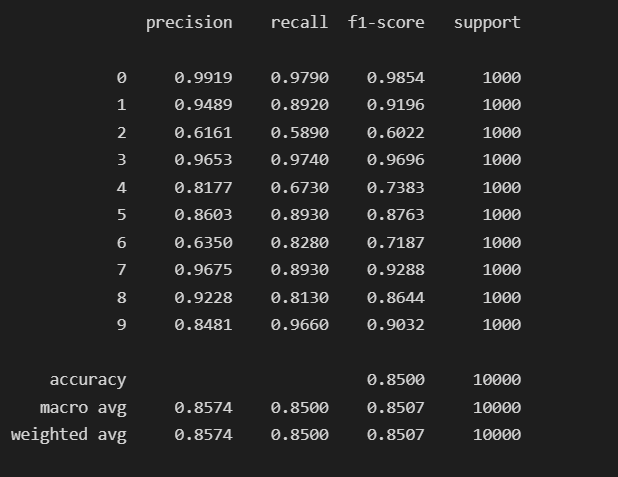
#### 1.4.4.1 决策树最大深度最佳取值分析

对于分类的评估，我们使用准确率和召回率的调和平均数，也就是F1-score来判断。因为这个评判标准综合和准确率和召回率，考虑相对来说更加全面合理。经过我们之前预处理步骤之后的数据，分别令决策树的最大深度以10开始，以5为步长取值，一直到F1-score稳定为止。

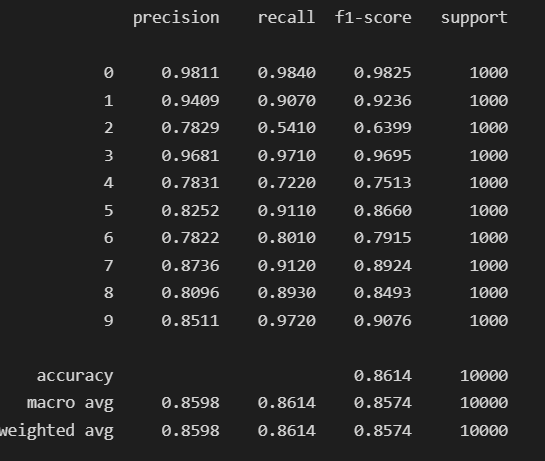
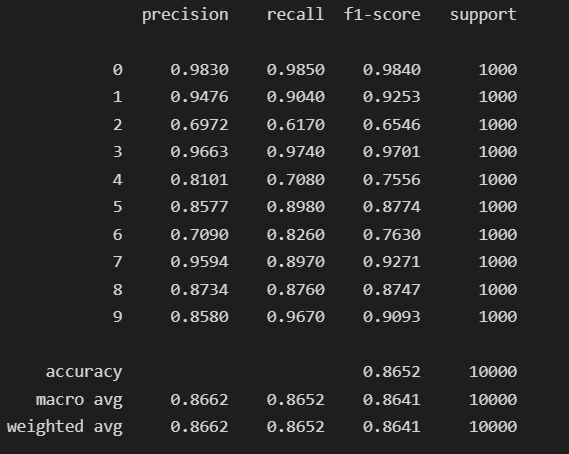
可以得到如下结果：



深度为 10 深度为15



深度为20 深度为25



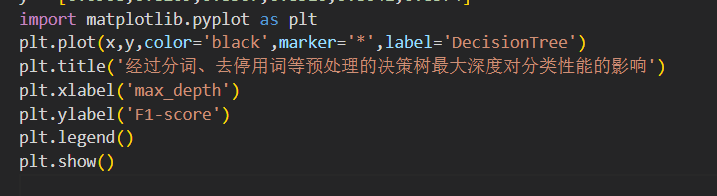
深度为 30 最大深度为50

可以看出来，随着深度的增加，模型的各个评分都在变好，但是到达20层之后的最大深度，增长的速度就减慢，而且和不限层数时的结果相差不大。

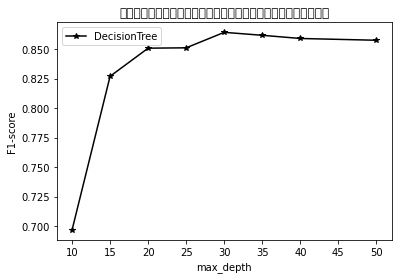
整理结果可得到如下结果：



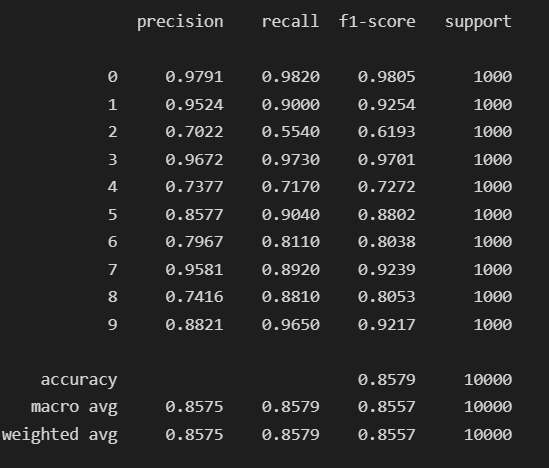
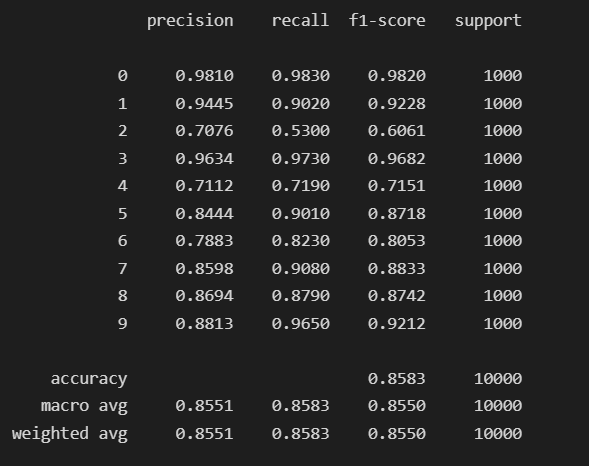
其中x代表横坐标最大深度，纵坐标代表记录的F1-score。作图代码如图所示：



最终的结果如下：

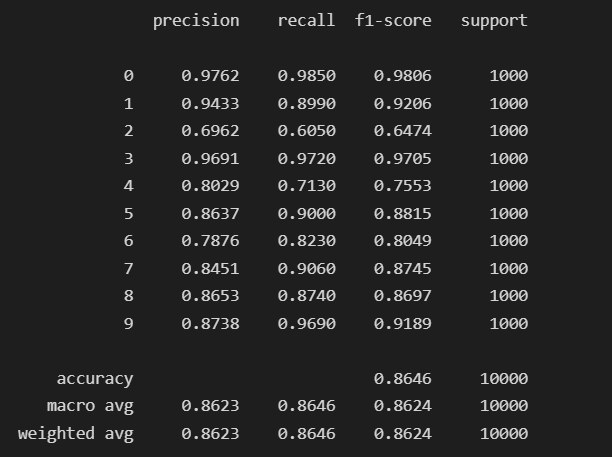


可以看到当决策树的最大深度为30时，已经有了最佳性能。从图中可以看到随着深度增加，F1的值也在逐渐增大，在最大深度为30时达到顶峰。从最大F1值之后逐渐增大最大深度这个约束，决策树的F1指标始终小于0.8641这个值 。所以最佳取值应该是。决策树的停止分裂条件也就是在30附近。我们再做一次实验，将最大深度分别设为28和32，可以得到如下结果。

深度为28 深度为32

可以看出25，32作为最大层数时，效果和20层时相差不大。再以31为最大层数训练。最后得到结果如下：



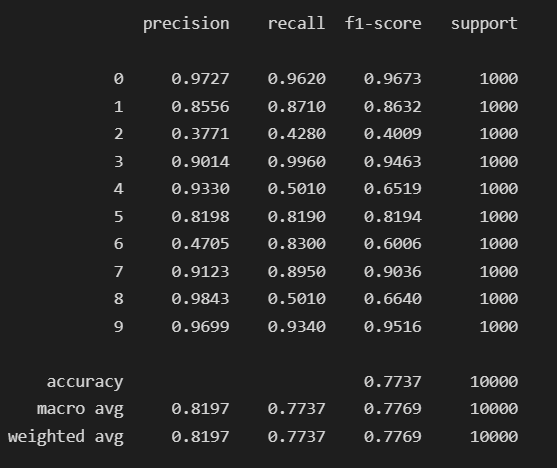
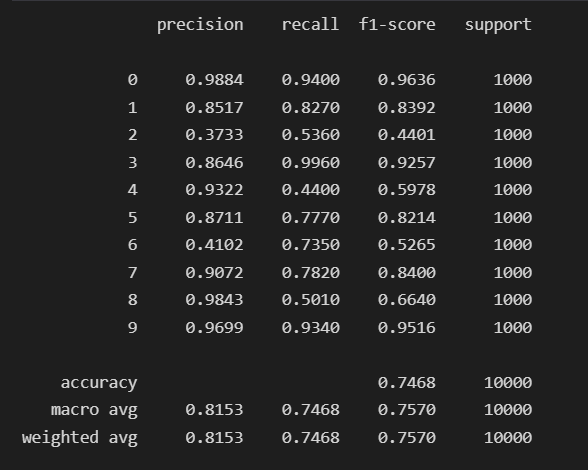
此结果也没有比最大层数为30时的结果更好。

综上所述，目前来说最大深度为30的效果最好。

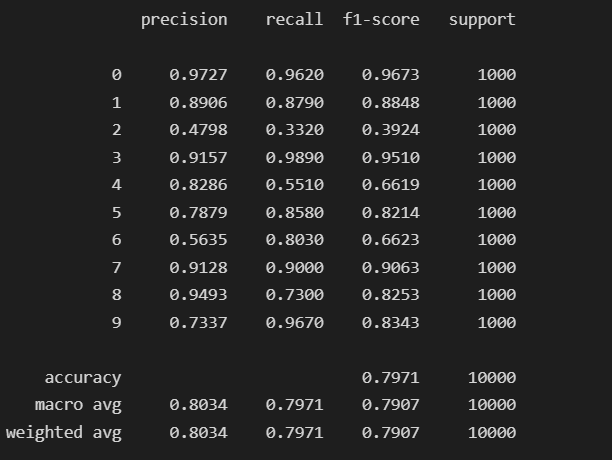
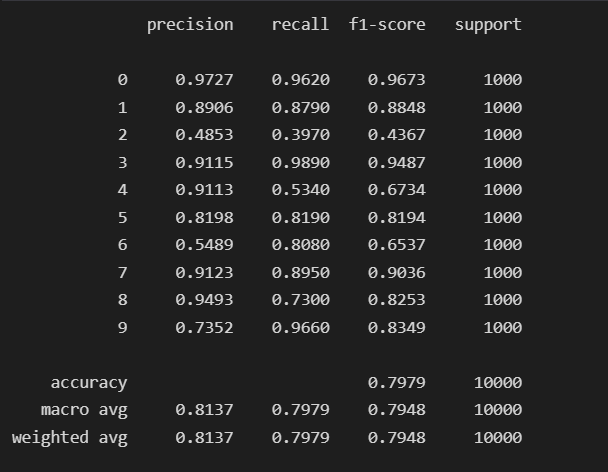
#### 1.4.4.2 决策树最大叶子节点数最佳取值分析

当限制决策树的最大叶子节点时，我们可以防止过拟合。当我们平时对模型进行训练的时候，我们可以将此变量进行限制，看能否得到一个较为更好的效果。

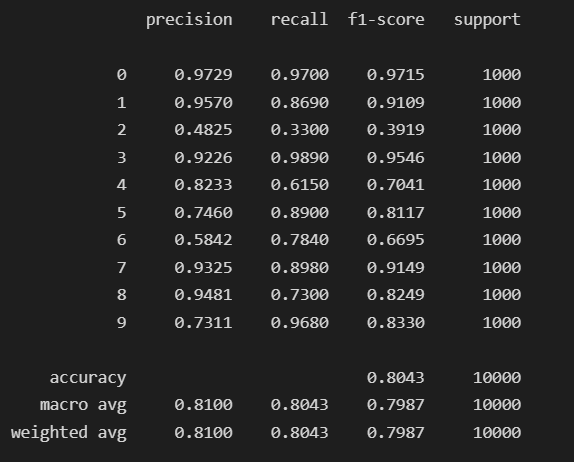
当我们改变决策树最大的叶子节点数时。以20开始10为步长进行训练



叶子节点数最大为20 叶子节点数最大为30

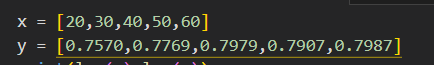


最大叶子节点数为40 最大叶子节点数为50

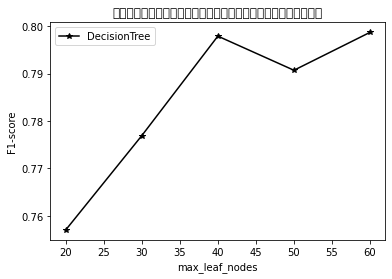


最大叶子节点数为60

最终得到的结果如下：



按照上述的作图代码，可得如下图像：



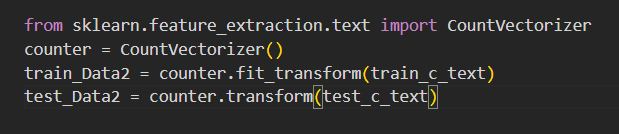
可以看到，当我们进行限制最大叶子节点数的时候，模型的效果跌落很多。我们这个的模型原本的训练不存在过拟合现象，当我们的将限制过拟合的手段使用出来时，会导致原本的模型效果更差。而且在限制最大节点数为40之后，其实模型的效果是差不多的。我们根本无法找到一个合适的值作为最大叶子节点数。但是我们可以让模型自己寻找合适的最大叶子节点数

而且从图中可以很明显看到2、6号的分类效果很差。2、6号数据在此模型中无法达到很好的训练效果。

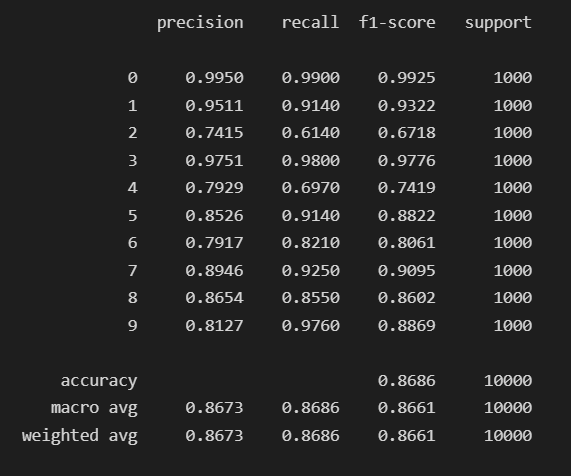
最开始不限制的决策树模型对于一些除了2、6号以外的类部分训练的效果也不是很好，但是我们通过限制最大深度，最大叶子节点数，可以让这些类的分类效果显著提升。例如0号类，原本的精确度是0.9619，但是在限制最大深度之后，每次训练的精确度都在0.9791以上。说明训练数据中的一部分适合较为较为浅的决策树模型，而还有一部分并不适合，需要更精确的分类标准。需要加深决策树的深度和增大决策树的叶子节点数。所以我们后续直接改变最大深度为50，得到的结果总体效果并不好，但是2、6类的分类效果却显著提升。所以，我们可以得到结论，2、6类需要的更精确的分类标准才能很好地分类，需要很深的深度保证他们的训练效果。但是，其他类并不需要很深，反而由于特别深的深度会导致原有的效果下降，这很可能是因为原有的较浅深度有效减缓了过拟合，所以在测试集上的效果很好。

#### 1.4.4.3 预处理性能分析

我之前预处理的时候，是将文本转化成TF-IDF矩阵进行运算。我们也可以将TF-IDF矩阵换成词频矩阵，每个矩阵的元素的行对应一个文档，每一个列对应一个词语，元素对应的是一个文档某个词与出现的次数。我们将这个作为文本的特征提取得到输入数据。代码如下：



依旧使用决策树进行训练，得到结果如下：

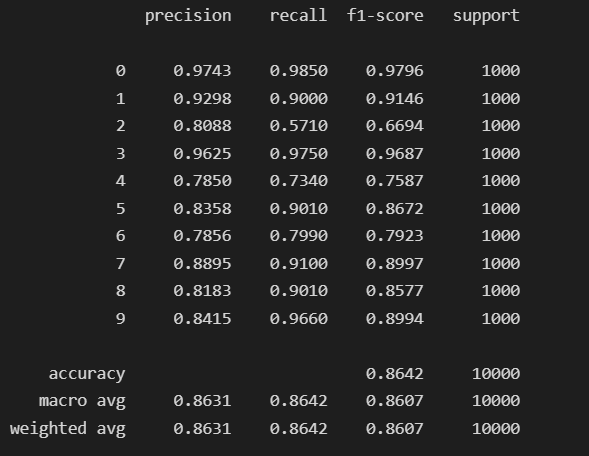


可以看到，最终的结果比我们一开始使用TF-IDF值的效果提高了0.11。所以我们实际应该使用BOW来提取文本特征。

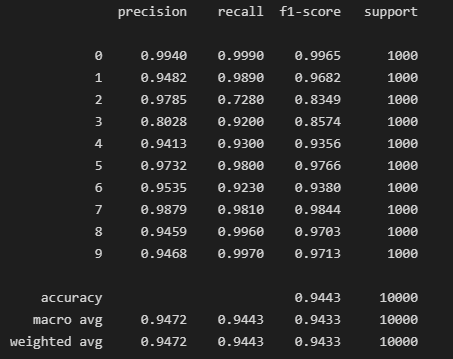
#### 1.4.4.4 噪声分析

原来的数据中，文本和标签是一一对应的，而且标签能准确地描述文本的内容。但是，如果，我们产生一些标签和内容不对应的数据加入到数据集中。这些数据就会扰动原来的高质量的数据。理论上会造成一定程度的分类效果下降。我实验的时候一共加入了十分之一的噪声数据。对决策树进行训练。

结果如下：

****

我们可以发现结果并没有出现很大的变化。可以证明决策树的抗噪性能还是比较好的。同时我们又使用其他模型进行了训练。比如说是boost集成的决策树分类。结果如下：



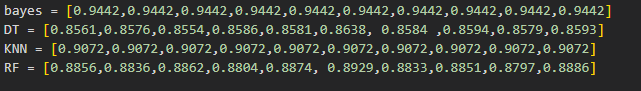
我们可以明显发现，boost集成的结果比原来没有加入噪声的时候效果更好。我认为是因为boost在集成的时候，会增大错误分类数据的权重，减少正确分类数据的权重，在迭代学习的过程中不断更新权重，很可能在不断的迭代学习过程中，这些错误数据一直被误分类，从而更加重视这些数据，学习到了这些错误数据的特征，相当于是学习的过程中“过拟合”了这些数据。Boosting对注意力都集中在比较难分和错分的数据上，主要任务集中在噪声数据上，这些因素导致了最后的结果。从实验结果中也可以体现这一点。

有一些噪声会对原有的分类缠上正向作用的。噪声能够更好地让模型知道哪些是错误的，进而产生更好的效果。相对来说，一些噪声，例如说是和原有数据很相近的噪声数据，就会让模型更好地关注到他们，形成一个对照。知道什么是错误分类，什么是正确的分类。

#### 1.4.4.5 众多模型分类评估

针对进行数据预处理之后的数据，分别采用决策树分类，随机森林分类，朴素贝叶斯分类，以及KNN进行训练分析。秉持着控制变量的思想，其中和决策树相关的模型都使用信息熵进行分类，所有模型都基于同一个数据集进行分类。因为每一次决策树的分裂都是不同的，我们可以试着实验多次，得到各模型在一定范围内的结果，进而对比各模型之间的效果。将每一种方法都继续10次实验，使用F1值评估模型的性能指标。对每一次的F1值都进行统计。由于每次

可得到如下结果：

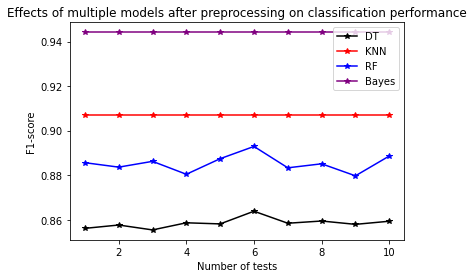


我们可以通过绘图更加直观地感受到他们的差别。

绘图代码如下：



最终，可以得到如下折线图：



可以看到，针对THUC数据集，朴素贝叶斯的分类效果最好，KNN次之。

随机森林和决策树的效果相差不多。但是总体来看，随机森林的分类效果要好于决策树的。随机森林的基本单元就是决策树，本质上随机森林属于集成学习，其每一个决策树打分决策一般都会好过一个决策树的分类效果，这是我们学习的最简单的bagging思想。这和之前介绍集成学习的工作原理相一致，实验结果也验证了这一点。

另外我们可以发现贝叶斯和KNN的计算结果的单一确定，没有任何变化。但是决策树和随机森林的分类结果却是在某一区间上下震荡。随机森林可能是因为每次训练出来的多个分类器都有差别，然后最终的投票选择也会产生一定的区别。而决策树则可能是因为由于考虑信息熵计算的先后顺序。

针对THUCnews数据集，使用词频、BOW进行特征提取之后选择模型最好使用朴素贝叶斯。贝叶斯分类不仅运算比较快，而且结果也非常好。决策树模型和随机森林模型是效果最差的分类器。决策树是不适合做文本分类的。

## 实验总结

通过这次实验，我熟悉了THUCnews数据集。此数据集是应用比较广泛的中文分类数据集。没有错误很严重的数据，而且数据量也非常多。此次实验是选中一部分数据集，抽取数据得到的。针对这次的实验，我的工作主要有一下几点：

1. 对原有的数据进行了预处理。由于使用的是中文的数据分类，所以需要进行分词。英文自带空格，所以不需要进行分词处理。而且，中文分词之后，还需要对分词后的数据进行处理，去掉停用词。结果需要将词语按照中间有空格的形式结合成一整个句子。
2. 对每一个文档经过预处理之后的数据，实现数据的向量化表示。文本的向量化表示有很多种形式，我们选择最简单的BOW和TF-IDF形式，二者同时使用，得到的结果使用同一模型进行训练。虽然最后的结果相差不多，但是本质上已经有所不同。
3. 对模型的一些超参数进行选择。由于我们一开始是使用决策树和随机森林进行训练的，后续才考虑了其他模型。所以实验对象一开始就是决策树。我们对决策树的超参数进行了选择。不同条件下的最大深度对模型的分类效果有很深的影响。同时我们也可以发现，我们限制最大叶子节点的时候，对于模型来说已经造成了精度下降的状况。实际上，限制最大叶子节点数是为了限制模型的过拟合现象。但是我们的模型一开始并没有出现过拟合现象，所以当我们限制最大叶子节点数的时候，模型的精度不可避免地下降了很多。
4. 对原有的数据集加入了噪声数据。噪声在文本中指的是不能正确描述当前场景的数据。我又找了一个数据集，随机对文本打上标签，很容易理解，这部分数据有很多是错误的。在图像中表是的就是离群点。理论上来讲，加入噪声是会对原来的模型产生一定影响的。实际情况证明也是如此。但是，精度实际并没有下降很多。所以决策树还是有一定的抗干扰能力的。但是，对于一些比较敏感的模型来说，噪声是非常影响模型效果的。
5. 分别使用了多个模型，进行比较。我对不同的模型使用相同的数据集，进行了10次独立重复实验。并且以F1值作为评价标准绘制了折现图，对多个模型的实际情况做了对比。并对不同模型在该数据集、此处理方式上的使用性进行了分析。

通过这从实验，我学会了使用sklearn库方便地对数据进行预处理任务，还能够使用库中提供的函数对处理完的数据实现分类任务。我基本实现了所有建议的任务，还选做了噪声对任务处理的影响这一个实验。对整个的分类任务有了很多的了解，自己对于大数据挖掘方面的知识有了长足的进步。