**---（3）朴素贝叶斯算法实现**

学生姓名： 学 号： 专业班级：

实验类型：□ 验证 ■ 综合 □ 设计 □ 创新 实验日期： 11.20 实验成绩：

**一、实验目的**

学会利用朴素贝叶斯算法对数据集进行分类，掌握朴素贝叶斯算法的思想。

**二、实验内容**

根据朴素贝叶斯算法原理实现对数据集的分类，编译、运行程序并观察程序的输出，并分析实验结果，写出实验报告。

**三、实验要求**

1. 分析实验代码；

2. 分析运行结果；

3. 画出程序流程图。

**四、实验步骤**

1. 编写实验代码；

2. 编译、运行程序；

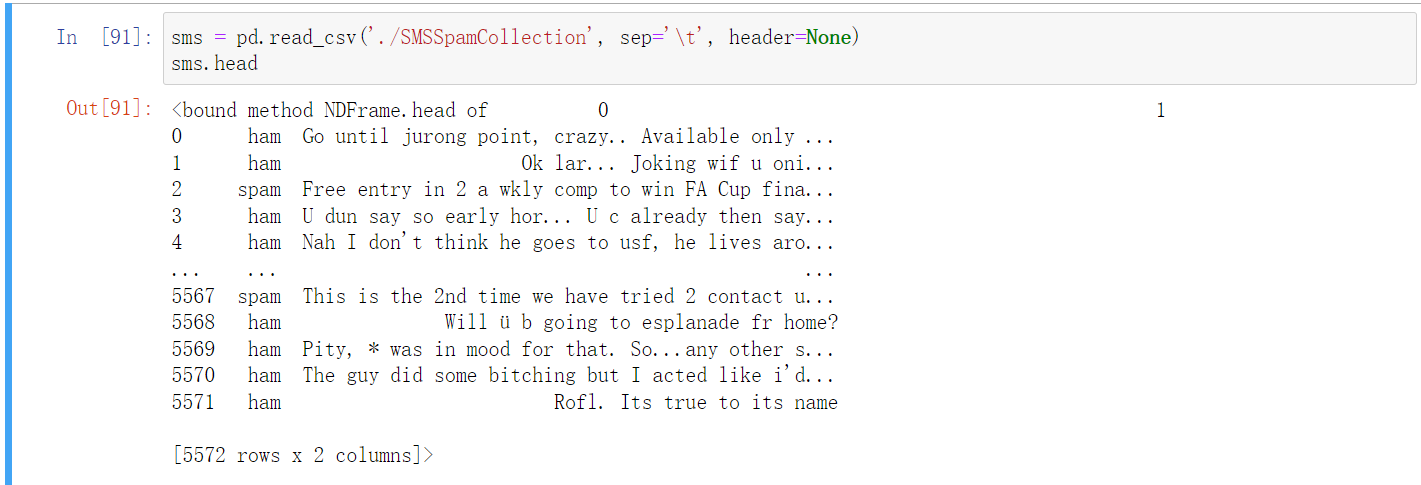
3. 观察程序的输出并分析实验结果。

**五、实验数据及处理结果**

**1. 查看数据集**

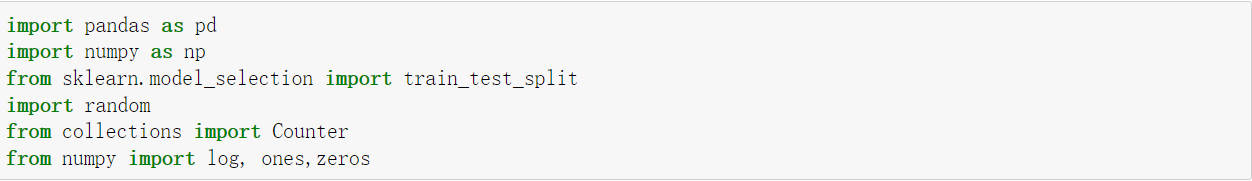
SMSSpamCollection是为手机垃圾邮件研究收集的一组带有SMS标签的公共数据。它包含5,572条英文短信，根据是合法或垃圾邮件进行标记。

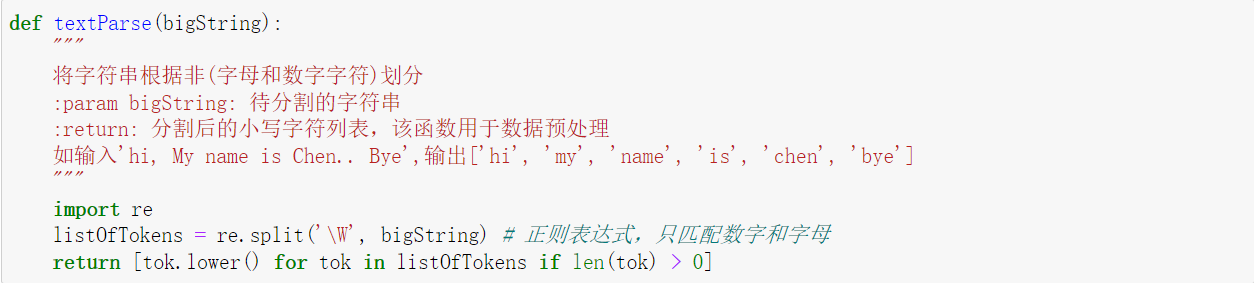
使用Jupyter Notebook查看数据集。代码如下如下：

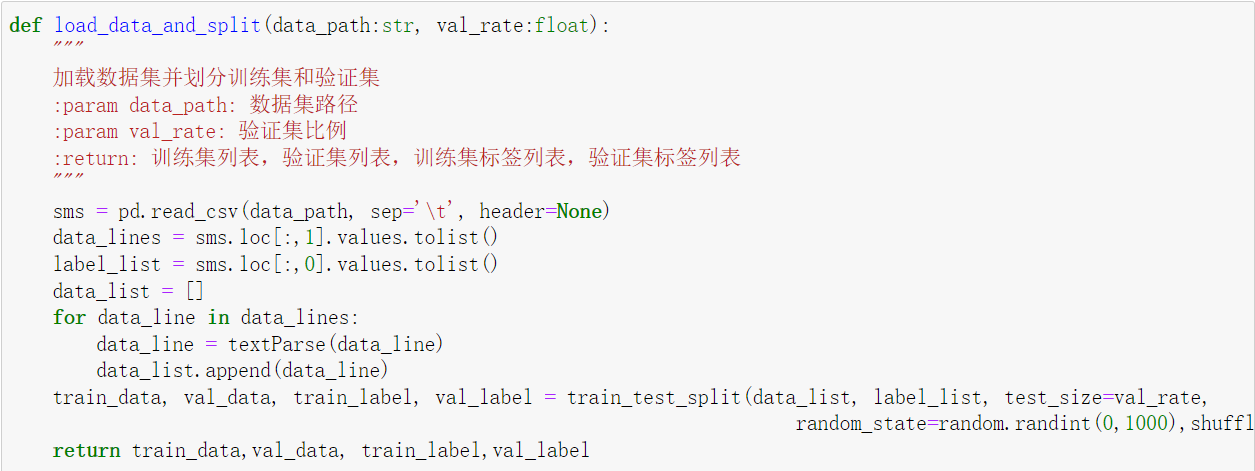
 由上图可知数据集共包含5572行数据。每行分为两部分，第一部分为邮件标签，分为ham(合法)以及spam(垃圾)，第二部分为邮件内容。实验正是根据邮件内容预测某条邮件的类别。本数据集涉及的问题为二分类问题。

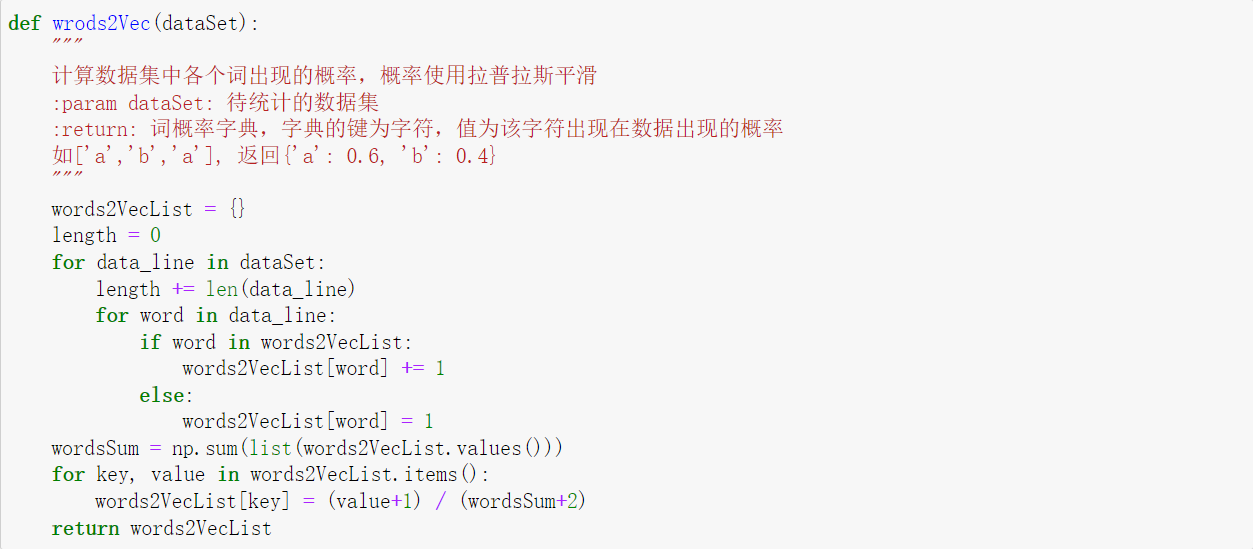
**2. 朴素贝叶斯算法实现**

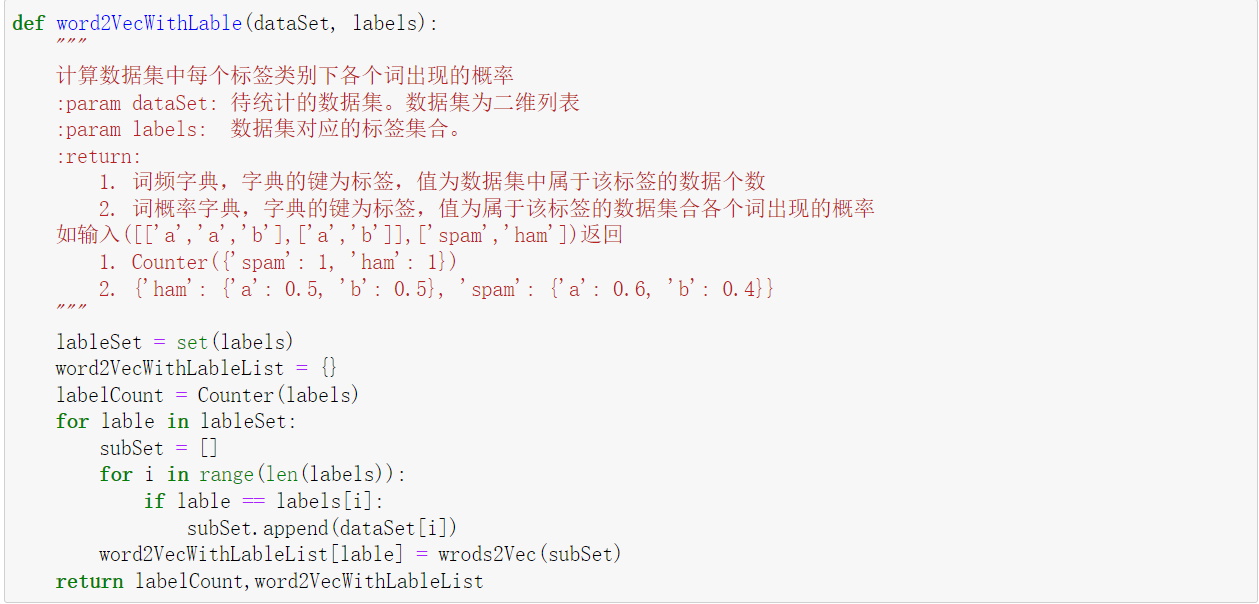
代码实现如下：



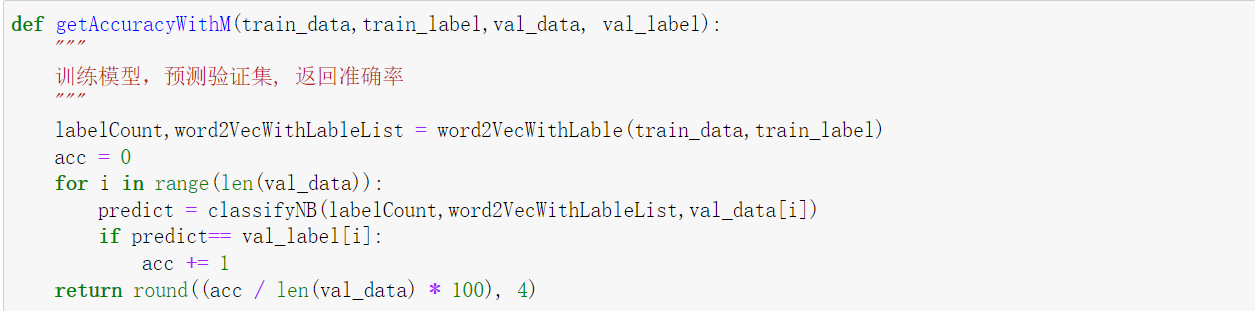










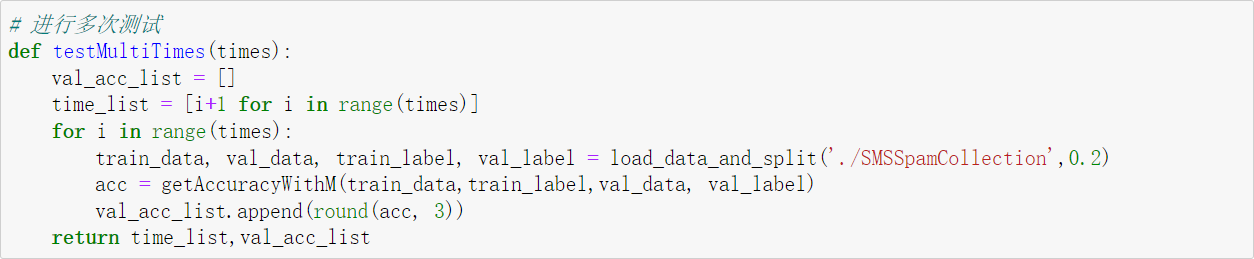


算法的流程图如下：

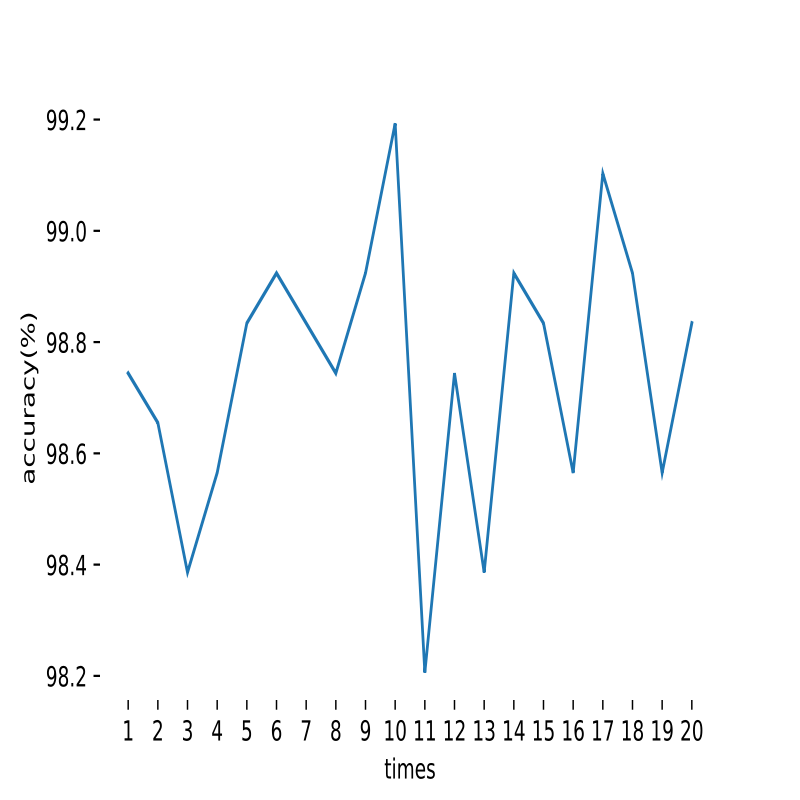


1. **多次运行**

注意到划分训练集和验证集时，程序会将数据集打乱并进行随机采样，所以每次运行结果不尽相同。为了观察模型的稳定性，下面进行多次测试并进行结果可视化。代码及运行结果如下：







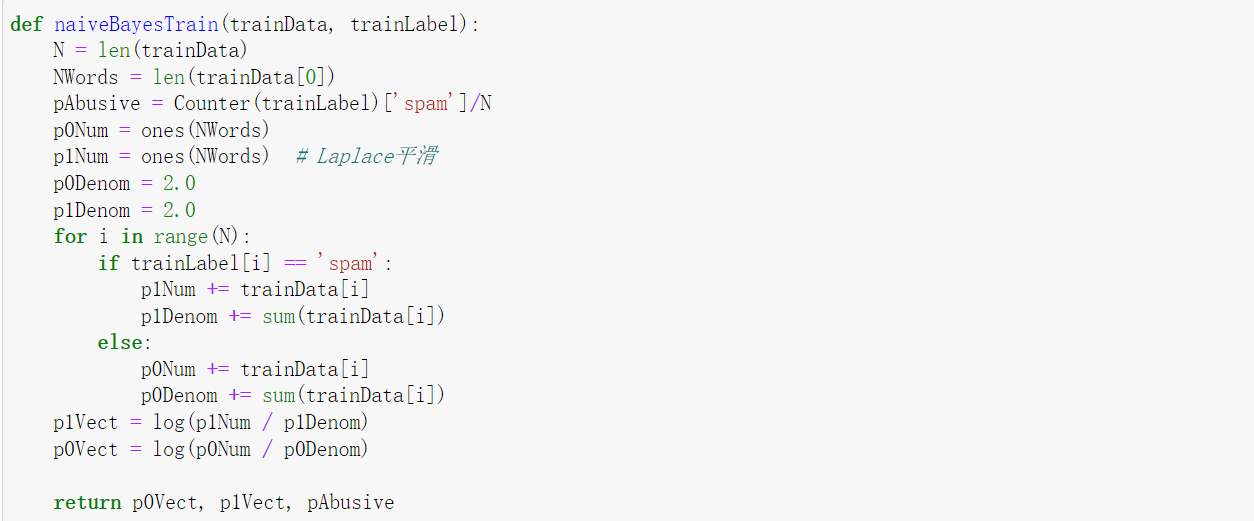
可以发现尽管每次运行测试集和验证集均不相同，但是模型在验证集上获得的准确率均保持在98%以上，且曲线稳定，波动不大。说明模型能够达到基本要求。

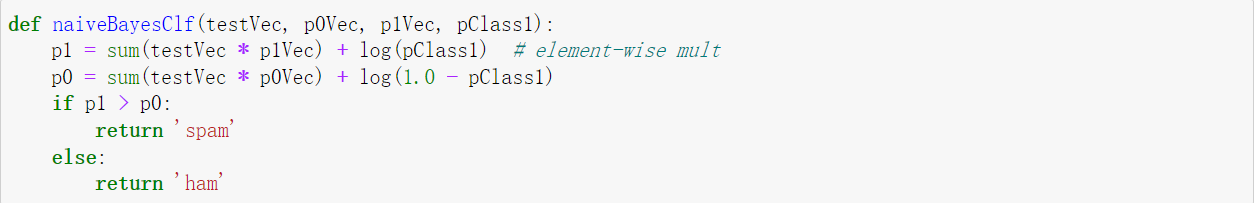
1. **模型对比**

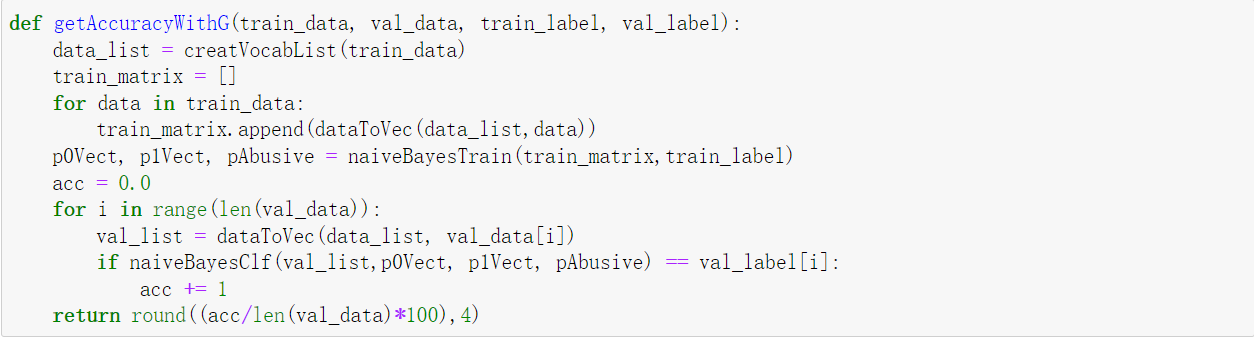
本次实验代码核心思想参考学长的实验报告中的代码，如下：











上述代码使用矩阵来存储各个词在每个类别下出现的概率。其程序流程如下：

1. 程序首先将训练集中出现的词合并成一个集合，每个词仅在集合中出现一次，称为词集合。如对应训练集[[‘a’,‘e’，‘b’]，[‘c’,‘a’,‘a’]，[‘d’,‘c’]]，其生成的词集合为[‘a’,‘b’,‘c’,‘d’,‘e’]。
2. 训练集映射成矩阵：将训练集中的每个样本映射成一个同词集合长度相同的列表，称为词位置列表。列表元素仅为0和1。0代表集合中该位置对应的词在样本中没有出现，1则表示出现。如对于词集合[‘a’,‘b’,‘c’,‘d’,‘e’]。样本[‘d’,‘c’]按照映射规则生成的词位置列表为[0,0,1,1,0]。所有列表组合成二维矩阵，称训练集矩阵。
3. 分类：依据标签向量，将训练集矩阵划分为两类(一类是ham,一类为spam)。每类构成一个矩阵，称为类别矩阵。
4. 计算类别矩阵中各个词出现的概率(先验概率)：对类别矩阵按行求和，得到该类别下每个词出现的次数，称为词频数列表。同2中的例子，如果某个类别矩阵求和后结果为[4，5，0，0，1]，则表示在该类别下，‘a’出现4次，‘b’出现一次，‘e’出现1次。将词频数列表的每个元素除以列表元素之和即得到该类别下各个词出现的概率，称为频率列表。如[4, 5, 0, 0, 1]对应的频率列表为[0.4, 0.5, 0, 0, 0.1]。表示在该类别下，词‘a’出现的概率为0.4，词‘b’出现的概率为0.5...然后对频率列表的每个元素取log，得到log列表。
5. 预测：对于输入的一个封邮件，首先将其映射成2中提及的词位置列表。然后将该词位置列表与各个类别在4中生成的log列表按照对应位置元素相乘再求和，得到类别和。最大类别和对应的类别即为该邮件的预测类别。

为了方便举例，在4中计算频率列表时没有使用Laplace平滑。而在程序中进行了严格的Laplace平滑，请注意区分。

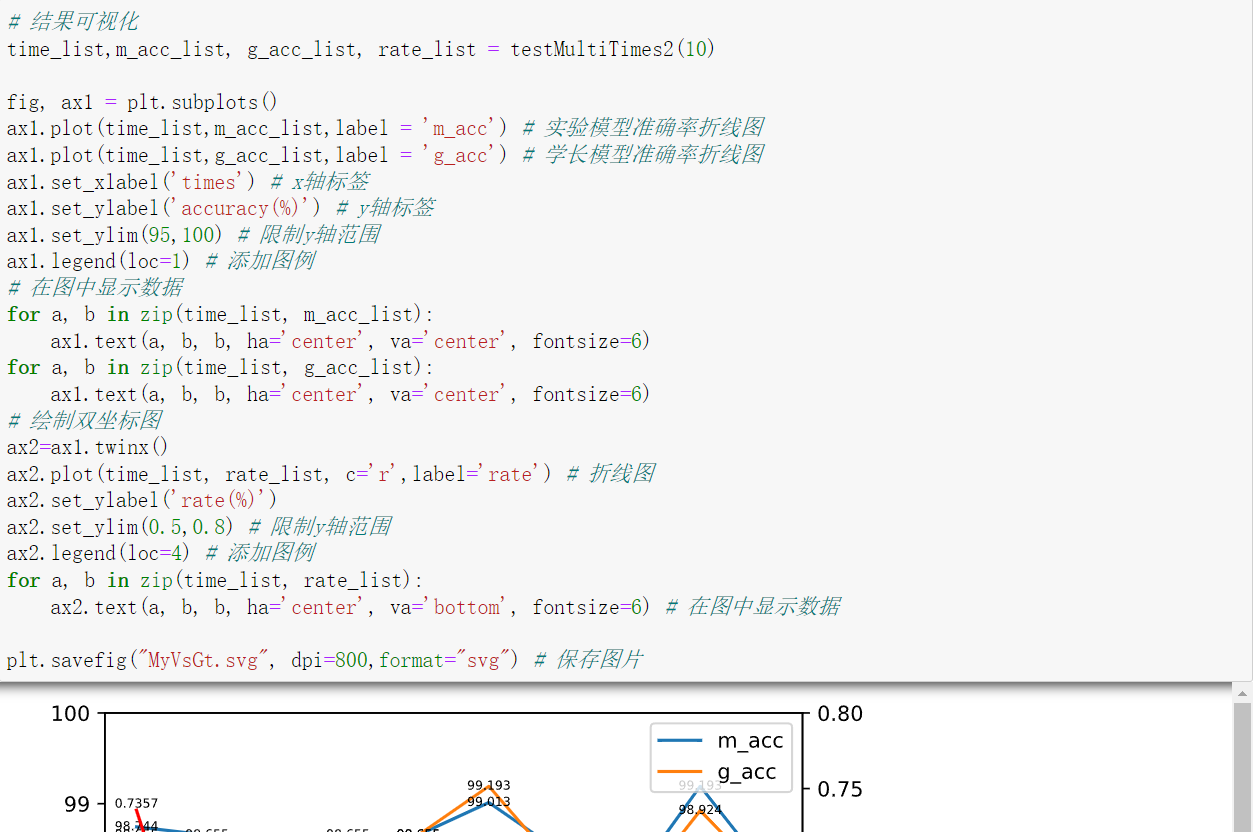
仔细分析上述代码不难发现，其存在如下两个问题：

1. 只能执行二分类问题。对于多分类问题无能为力。
2. 训练集映射成的训练矩阵为稀疏矩阵。生成该矩阵的过程耗费大量时间,

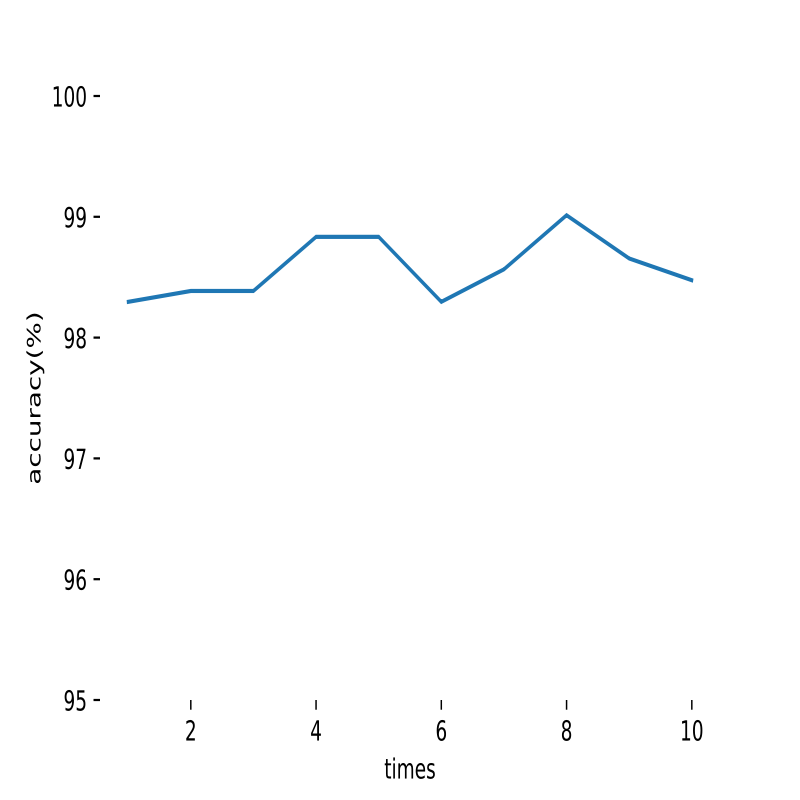
同时浪费了大量的空间储存无意义的0。

上述缺点极大地限制了该程序在其他数据集中的应用，尤其是在规模较大的数据集上。这也是我选择用字典代替矩阵存储数据的主要原因。为了直观感受两个模型之间的差距，下面从运行时间以及准确率两个角度进行对比。代码如下：





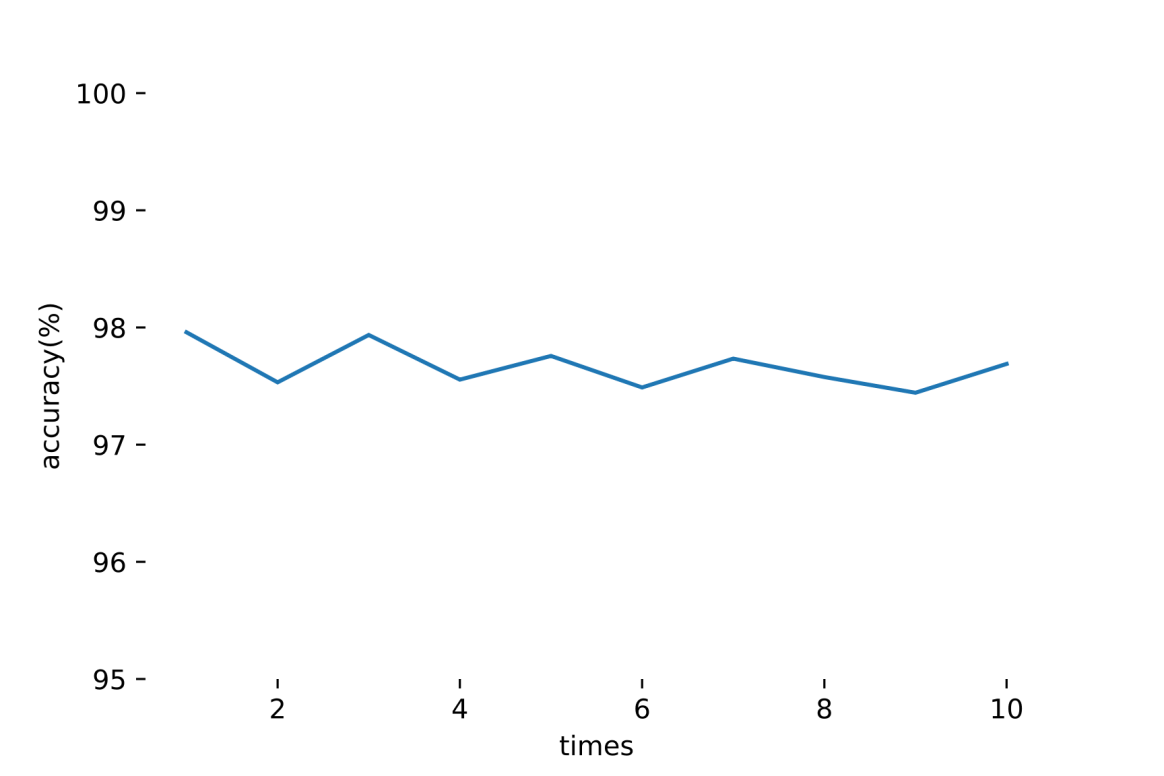
运行结果如下：



其中，m\_acc指实验模型的准确率，g\_acc指学长模型的准确率，rate指实验模型的运行时间与学长模型的运行时间的比值(单位 %)。

由上图不难发现，实验模型的准确率曲线大部分点在学长模型的准确率曲线上方，说明实验模型有着更好的预测能力。同时，比值曲线的值均不超过0.8%,说明实验模型的运行时间仅占学长模型运行时间的0.8%以下，运行速度的提升是巨大的。而且这个比值将会随着数据集的增大而减小，也即学长的模型的时间复杂度不是随数据规模线性增长的。

上述的实验验证了本实验模型的优势。为了更好地评估实验模型的性能，下面将实验模型、学长的模型以及python标准库sklearn中的MultinomialNB(多项式朴素贝叶斯)模型进行对比。代码大同小异，此处便直接展示结果。

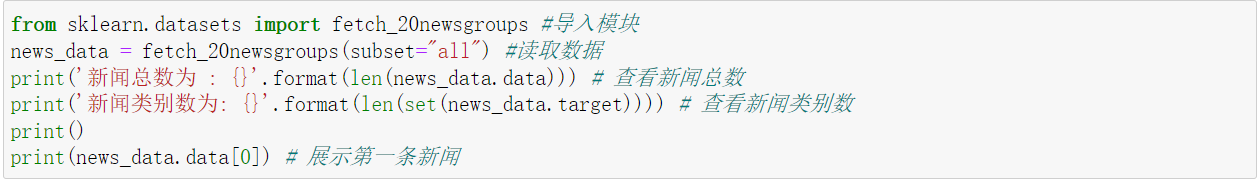


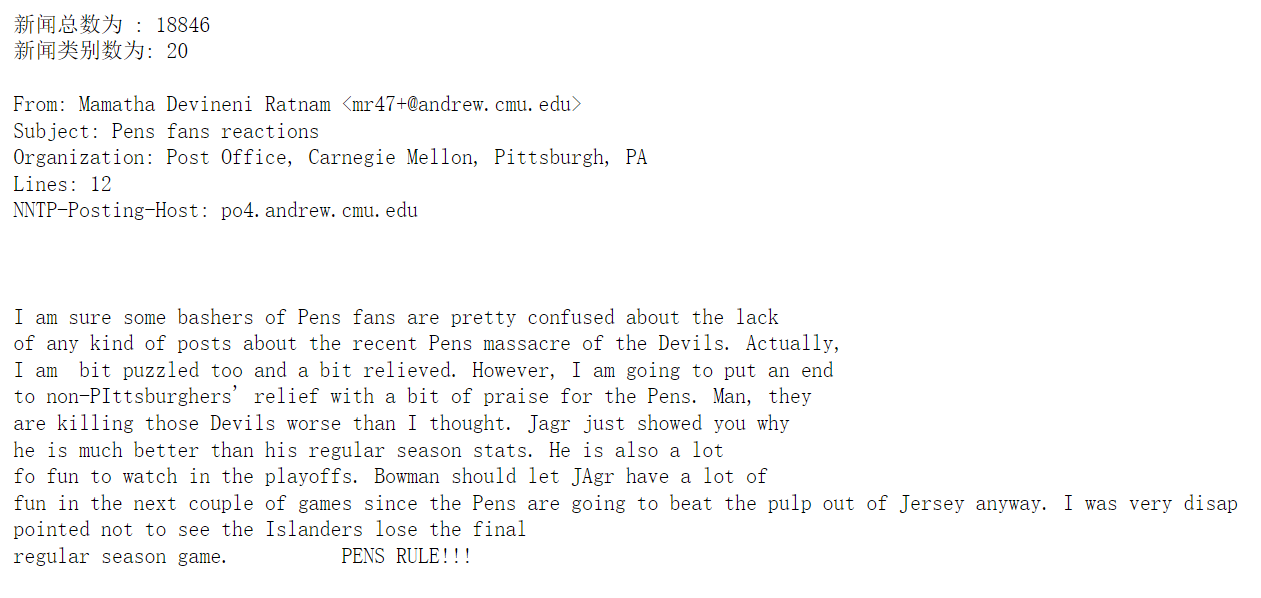
其中，m\_acc指实验模型的准确率，g\_acc指学长的模型的准确率，b\_cc指多项式朴素贝叶斯模型的准确率。不难发现，实验模型的准确率与多项式朴素贝叶斯模型准确率大致相同，均高于学长的模型的准确率。由此不难说明，实验模型具有较的分类能力以及较好的性能。

1. **20newsgroups**

上述实验验证了在中小型数据集上，实验模型拥有不错的分类预测能力。但是无法直接肯定该模型在大型词分类数据集上拥有同样的效果。下面设计实验进行验证。

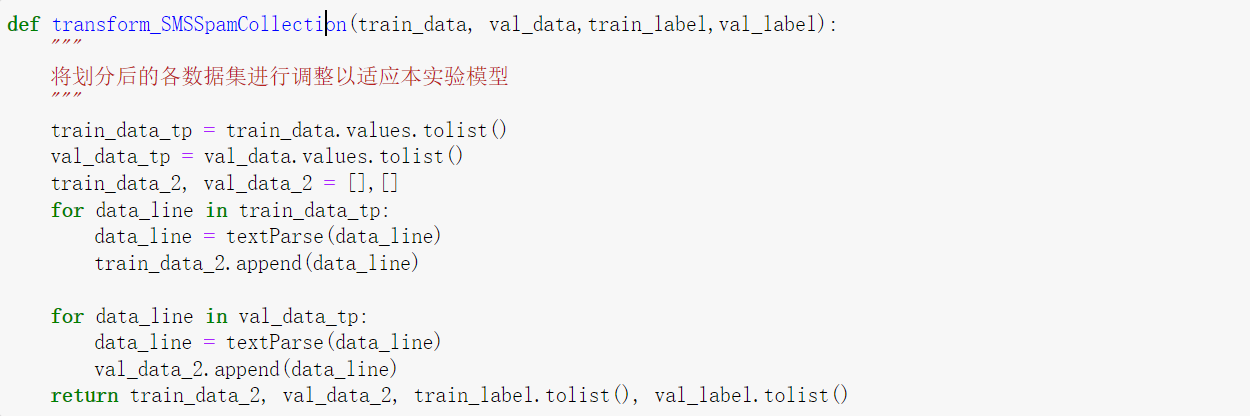
20newsgroups[数据集](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%9B%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/imstudying/article/details/_blank)是用于文本分类、文本挖据和信息检索研究的国际标准数据集之一。数据集收集了大约20,000左右的新闻组文档，均匀分为20个不同主题的新闻组集合。一些新闻组的主题特别相似(e.g. comp.sys.ibm.pc.hardware/ comp.sys.mac.hardware)，还有一些却完全不相关 (e.g misc.forsale /soc.religion.christian)。使用 JupyterNotebook查看该数据集如下：



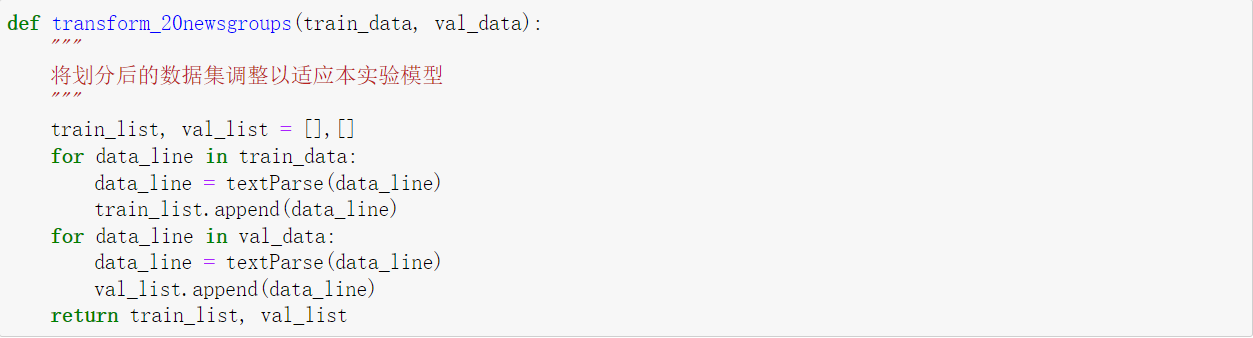


可以看到，20newsgroups数据集共收集18846条新闻，新闻类别数为20。同时每个样本的词数量较SMSSpamCollection数据集中的样本的词数量大大增加。如此庞大的数据量对模型的性能提出了更高的要求。如果使用学长的模型进行训练，则程序将会提示内存不足。因为学长的模型在实现过程中使用了大量的稀疏矩阵，这无疑会大幅增大程序的内存开销，同时也需要耗费大量时间。于是，本次验证仅将实验模型与sklearn库中的多项式朴素贝叶斯(MultinomialNB)模型(简称标准库模型)进行对比。由于划分训练集和测试集时为随机采样，所以本次验证将重复进行50次预测，取50次预测结果的准确率的平均值作为模型的最终准确率。代码实现如下：









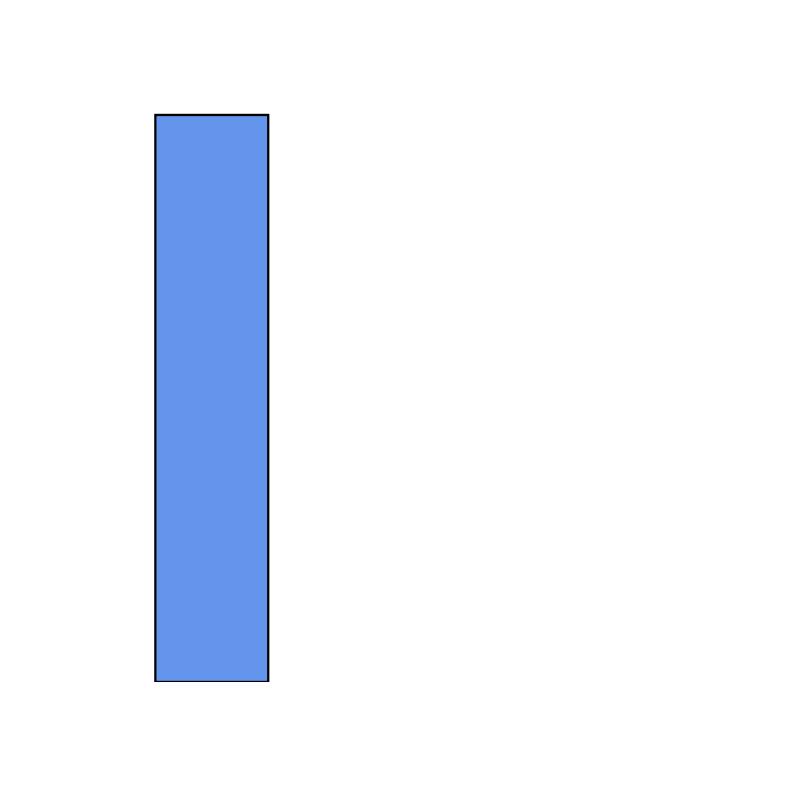




结果可视化：



实验结果：



其中，m\_acc表示本实验模型的平均准确率，b\_acc为标准库模型的平均准确率。

由图不难发现，在SMSSpamCollection数据集上，实验模型与标准库模型均获得了98%以上的超高准确率，且两者的准确率无明显差距，这与前文的结论保持一致，此处不再重复赘述。而在大数据集20newsgroups上，两模型的准确率均有所降低。这不仅仅与20newsgroups庞大的数据量有关，更与其涉及20个新闻类别且存在若干个相近的类别的数据集特点有关。分类类别越多，样本之间越相似，模型犯错的几率也越大。但是，实验模型的平均准确率却明显高于标准库模型，甚至超过2%之多。若以准确率为评判标准，则实验模型的分类能力超过标准库模型。但是从执行时间的角度来看，在两者准确率相差不大的情况下，实验模型所消耗的时间是标准库模型的9倍以上，很显然是后者更加优秀。总之，两模型各有优劣，需要根据数据集的具体情况以及结合需求侧重灵活选择模型。

综上，在中小数据集上，实验模型有着与标准库的一致预测能力。在大型数据集上，其分类能力同样在可接受的范围。至此实验完成。

**六、实验总结**

在开始实验前，我先阅读了19级学长的实验报告中的代码，发现其参考《机器学习实战》(Peter Harrington著)第4章 基于概率论的分类方法：朴素贝叶斯。该章节所展示的代码使用矩阵作为主要数据结构表示各个词是否出现以及出现的概率。由于一封邮件出现的词的数量仅占所有邮件出现的词的总数的10%以下，所以该矩阵为稀疏矩阵，这也意味着矩阵中存储着大量无用的信息。而生成这些无用的信息代价无疑是巨大的。程序并为对此进行优化，而是将其当作普通矩阵处理，这是其最大的缺点。在仔细分析了该程序后，我决定摒弃矩阵，选择字典来存储每个词出现的概率，这样便能最大化地保留有用信息丢弃无用信息，从而减少时间和空间的浪费。

实验结果证明我的分析是正确的---本次实验首先对学长的代码进行改进，得到了本次实验的模型。该模型进行多次预测均获得98%以上的准确率。为了更好地说明改进后的实验模型有着更高的运行效率，实验又对比了两个模型的运行时间以及准确率。结果发现本实验的模型多次运行的平均准确率高于原模型的平均准确率，同时运行时间仅占原模型的0.8%以下，运行速度得到极大地提升。为了更加客观地评价模型，实验又将实验模型与sklearn库中的多项式朴素贝叶斯模型进行对比，结果显示两者的准确率相差无几。

最后，为了测试实验模型在大型数据集上的分类能力，又将实验模型在20newsgroups数据集上进行训练并预测。发现其预测能力甚至超过标准库中的多项式朴素贝叶斯模型，但同时其时间开销也是后者的9倍之多。两者均有优劣。

综上验证可知，实验模型有着较为出众的分类能力以及较高的运行效率。由于实验代码较长，实验报告仅展示核心代码。完整代码已放至

[ChenWeilinx/machine\_learning: 机器学习与数据挖掘实验 (github.com)](https://github.com/ChenWeilinx/machine_learning)

**七、参考资料**

《数据挖掘原理与实践 蒋盛益，李霞，郑琪编著》

《机器学习实战 Peter Harrington著》