

电子科技大学信息与软件工程学院

# 实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 机器学习技术与应用

理论教师 黄 俊

实验教师 杨 珊

# 电子科技大学

## 实验报告

学生姓名：袁昊男      学号：2018091618008      指导教师：杨珊

实验地点：在线实验      实验时间：2020.04.29

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：垃圾邮件分类的贝叶斯分类模型

三、实验学时：4 学时

四、实验原理：

在分类（Classification）问题中，常常需要把一个事物分到某个类别。一个事物具有很多属性，把它的众多属性看做一个向量，即  $x = (x_1, x_2, x_3, \dots, x_n)$ ，用  $x$  这个向量来代表这个事物。类别也是有很多种，用集合  $Y = \{y_1, y_2, \dots, y_m\}$  表示。如果  $x$  属于  $y_1$  类别，就可以给  $x$  打上  $y_1$  标签，意思是说  $x$  属于  $y_1$  类别。这就是所谓的分类（Classification）。 $x$  的集合记为  $X$ ，称为属性集。一般  $X$  和  $Y$  的关系是不确定的，你只能在某种程度上说  $x$  有多大可能性属于类  $y_1$ ，比如说  $x$  有 80% 的可能性属于类  $y_1$ ，这时可以把  $X$  和  $Y$  看做是随机变量， $P(Y|X)$  称为  $Y$  的后验概率（posterior probability），与之相对的， $P(Y)$  称为  $Y$  的先验概率（prior probability）。在训练阶段，我们要根据从训练数据中收集的信息，对  $X$  和  $Y$  的每一种组合学习后验概率  $P(Y|X)$ 。分类时，来了一个实例  $x$ ，在刚才训练得到的一堆后验概率中找出所有的  $P(Y|X)$ ，其中最大的那个  $y$ ，即为  $x$  所属分类。根据

贝叶斯公式，后验概率为 
$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}。$$

在比较不同  $Y$  值的后验概率时，分母  $P(X)$  总是常数，因此可以忽略。先验概率  $P(Y)$  可以通过计算训练集中属于每一个类的训练样本所占的比例容易地估计。

在文本分类中，假设我们有一个文档  $d \in X$ ， $X$  是文档向量空间（document space），和一个固定的类集合  $C = \{c_1, c_2, \dots, c_j\}$ ，类别又称为标签。显然，文档向量空间是一个高维度空间。我们把一堆打了标签的文档集合  $\langle d, c \rangle$  作为训练样

本,  $\langle d, c \rangle \in X \times C$ 。例如:  $\langle d, c \rangle = \{\text{Beijing joins the World Trade Organization, China}\}$  对于这个只有一句话的文档, 我们把它归类到 China, 即打上 china 标签。

我们期望用某种训练算法, 训练出一个函数  $\gamma$ , 能够将文档映射到某一个类别:  $\gamma: X \rightarrow C$  这种类型的学习方法叫做有监督学习, 有一个监督者监督着整个学习过程。朴素贝叶斯分类器是一种有监督学习。

## 五、实验目的:

- 1、掌握贝叶斯分类器基本原理。
- 2、掌握条件概率的计算方法。
- 3、掌握朴素贝叶斯交叉验证方法。
- 4、掌握 Python 数据分析常用库 pandas 及其函数式编程。
- 5、了解拉普拉斯平滑。

## 六、实验内容:

- 1、收集数据, 收集垃圾邮件文本数据以及停用词。
- 2、加载数据, 使用 pandas 加载数据, 并使用函数式编程 (lambda) 来对数据进行预处理。
- 3、训练算法, 分析并掌握 train() 函数的训练算法。
- 4、测试算法, 观察错误率, 确保分类器可用。
- 5、使用算法, 构建完整的程序, 封装上述的算法。

## 七、实验器材 (设备、元器件):

- 1、PC 电脑。
- 2、Python。

## 八、实验步骤:

### 1、数据处理

```
1. # file.py
2.
3. import shutil, os
4.
5. f = open("../demo/index", 'r', encoding='gbk')
6. dic = []
7. for line in f.readlines():
8.     line = line.strip('\n') # 去掉换行符\n
9.     b = line.split(' ') # 将每一行以空格为分隔符转换成列表
10.    dic.append(b)
11.
12. x = dic[0][0]
13. # dic = dict(dic)
```

```

14. # print(dic)
15.
16. spam_num = 000
17. norm_num = 000
18. spam_count = 0
19. norm_count = 0
20.
21. for i in range(len(dic)):
22.     if dic[i][0]=='spam':
23.         shutil.copy(dic[i][1], "../demo/spam/" + str(spam_num))
24.         print("spam: " + str(spam_num))
25.         spam_num = spam_num + 1
26.         spam_count = spam_count + 1
27.     else:
28.         shutil.copy(dic[i][1], "../demo/norm/" + str(norm_num))
29.         print("norm: " + str(norm_num))
30.         norm_num = norm_num + 1
31.         norm_count = norm_count + 1
32.
33. print("spam_count: " + str(spam_count))
34. print("norm_count: " + str(norm_count))

```

**说明：**根据 index.txt 文件中的 spam 与 ham 将原始数据归档。标签为 spam 的为垃圾邮件，标签为 ham 的为正常邮件。共得到 42854 封垃圾邮件、21766 封正常邮件。并从中随机选择 2000 封作为测试集样本，剩余邮件作为训练集样本。

## 2、jieba 分词

```

1. # spamProcess.py / class spamEmailBayes
2.
3. # 导入停用词表
4. def getStopWords(self):
5.     stopList=[]
6.     for line in open("../data/stop.txt", 'r', encoding='gbk', errors='ignore'):
7.         stopList.append(line[:len(line)-1])
8.     return stopList;
9.
10. # jieba 分词，保存字典
11. def get_word_list(self,content,wordsList,stopList):
12.     #分词结果放入 res_list
13.     res_list = list(jieba.cut(content))
14.     for i in res_list:
15.         if i not in stopList and i.strip()!='' and i!=None:
16.             if i not in wordsList:
17.                 wordsList.append(i)
18.
19. # 若 jieba 处理后的词已在字典中，则频度加 1，否则添加到字典
20. def addToDict(self,wordsList,wordsDict):
21.     for item in wordsList:
22.         if item in wordsDict.keys():
23.             wordsDict[item]+=1
24.         else:
25.             wordsDict.setdefault(item,1)

```

```

26.
27. # 获得文件路径
28. def get_File_List(self,filePath):
29.     filenames=os.listdir(filePath)
30.     return filenames

```

**说明:**首先使用 open 函数打开(以 gbk 编码方式)中文停用词表 stop.txt, 字符串处理后保存至 stopList 列表; get\_word\_list()方法使用 jieba 库分词函数处理邮件内容, 并将分词结果保存至 wordsList 列表中; 若 jieba 库处理后的得到的词汇已在 wordsDict 字典中, 则该词词频加 1, 否则直接添加到字典中。

### 3、计算条件概率

```

1. # spamProcess.py / class spamEmailBayes
2.
3. # 计算每个邮件中条件概率  $p(s|w)$  得出对分类影响最大的 15 个词
4. def getTest-
    Words(self,testDict,spamDict,normDict,normFilelen,spamFilelen):
5.     wordProbList={}
6.     for word,num in testDict.items():
7.         if word in spamDict.keys() and word in normDict.keys():
8.             #该文件中包含词个数
9.             pw_s=spamDict[word]/spamFilelen
10.            pw_n=normDict[word]/normFilelen
11.            ps_w=pw_s/(pw_s+pw_n)
12.            wordProbList.setdefault(word,ps_w)
13.        if word in spamDict.keys() and word not in normDict.keys(
14.            ):
15.            pw_s=spamDict[word]/spamFilelen
16.            pw_n=0.01
17.            ps_w=pw_s/(pw_s+pw_n)
18.            wordProbList.setdefault(word,ps_w)
19.        if word not in spamDict.keys() and word in normDict.keys(
20.            ):
21.            pw_s=0.01
22.            pw_n=normDict[word]/normFilelen
23.            ps_w=pw_s/(pw_s+pw_n)
24.            wordProbList.setdefault(word,ps_w)
25.        if word not in spamDict.keys() and word not in normDict.k
26.            eys():
27.            #若该词不在 spam 字典中, 概率设为 0.4
28.            wordProbList.setdefault(word,0.4)
29.
30.        after = sorted(wordProbList.items(),key=lambda d:d[1],re-
31.            verse=True)[0:15]
32.        file = open("./wordprobList.txt", 'a')
33.        for i in range(len(after)):
34.            s = str(after[i]).replace('[', '').replace(']', '') # 去
35.            除[],这两行按数据不同, 可以选择
36.            s = s.replace('"', '').replace(',', ' ') + '\n' # 去除单引
37.            号, 逗号, 每行末尾追加换行符
38.            file.write(s)
39.        file.close()

```

```

34.     # print("保存文件成功")
35.
36.     # print(after)
37.     return wordProbList

```

**说明：**计算每个邮件中分词的条件概率，得出对分类影响最大的 15 个词。在计算过程中，若该词只出现在垃圾邮件的词典中，则令其概率为 0.01；若都未出现，则令其概率为 0.4。这里取的 0.01 与 0.4 做的假设是根据前人做的一些研究工作得出的。将分词结果与条件概率计算结果保存至 wordprobList.txt。

#### 4、贝叶斯条件概率与测试集正确率计算

```

1. # spamProcess.py / class spamEmailBayes
2.
3. # 计算贝叶斯概率
4. def calBayes(self,wordList,spamdict,normdict):
5.     ps_w=1
6.     ps_n=1
7.
8.     for word,prob in wordList.items() :
9.         # print(word+"/"+str(prob))
10.        ps_w*=(prob)
11.        ps_n*=(1-prob)
12.    p=ps_w/(ps_w+ps_n)
13.
14.    return p
15.
16. # 计算预测结果正确率
17. def calAccuracy(self,testResult):
18.     rightCount=0
19.     errorCount=0
20.     for name ,catagory in testResult.items():
21.         if (int(name)<1000 and cata-
22.            gory==0) or(int(name)>1000 and catagory==1):
23.             rightCount+=1
24.         else:
25.             errorCount+=1
26.     return rightCount/(rightCount+errorCount)

```

**说明：**根据贝叶斯公式计算后验概率。已知词向量  $w = (w_1, w_2, w_3, \dots, w_n)$  的条件下求包含该词向量邮件是否为垃圾邮件的概率。即求：  $P(s|w)$ ，其中  $w = (w_1, w_2, w_3, \dots, w_n)$ ， $s$  表示分类为垃圾邮件。根据贝叶斯定理和全概率公式：

$$\begin{aligned}
 & P(s|w_1, w_2, w_3, \dots, w_n) \\
 &= \frac{P(s, w_1, w_2, w_3, \dots, w_n)}{P(w_1, w_2, w_3, \dots, w_n)} = \frac{P(w_1, w_2, w_3, \dots, w_n | s) P(s)}{P(w_1, w_2, w_3, \dots, w_n)} \\
 &= \frac{P(w_1, w_2, w_3, \dots, w_n | s) P(s)}{P(w_1, w_2, w_3, \dots, w_n | s) P(s) + P(w_1, w_2, w_3, \dots, w_n | s') P(s')} \quad (1)
 \end{aligned}$$

根据朴素贝叶斯的条件独立假设，并设先验概率  $P(s) = P(s') = 0.5$ ，(1) 式可化为：

$$= \frac{\prod_{j=1}^n P(w_j | s)}{\prod_{j=1}^n P(w_j | s) + \prod_{j=1}^n P(w_j | s')}$$

再利用贝叶斯定理  $P(w_j | s) = \frac{P(s | w_j) P(w_j)}{P(s)}$ ，上式可化为：

$$= \frac{\prod_{j=1}^n P(s | w_j)}{\prod_{j=1}^n P(s | w_j) + \prod_{j=1}^n P(s' | w_j)} = \frac{\prod_{j=1}^n P(s | w_j)}{\prod_{j=1}^n P(s | w_j) + \prod_{j=1}^n (1 - P(s | w_j))} \quad (2)$$

因此可以用(2)式来计算概率。

在 2000 个测试样本中，文件名小于 1000 的是正常邮件，大于 1000 的是垃圾邮件；对应正常邮件标签为 0，垃圾邮件标签为 1。因此如果在 testResult 字典中邮件文件名小于 1000 且标签为 0，或邮件文件名大于 1000 且标签为 1，则说明判断正确；反之判断错误。

## 5、数据结构初始化

```
1. # main.py
2.
3. from src.spam.spamEmail import spamEmailBayes
4. import re
5. # spam 类对象
6. spam=spamEmailBayes()
7.
8. # 保存词频的字典
9. spamDict={}
10. normDict={}
11. testDict={}
12.
13. # 保存每封邮件中出现的词
14. wordsList={}
15. wordsDict={}
16.
17. # 保存预测结果
18. testResult={}
19.
20. # 分别获得正常邮件、垃圾邮件及测试文件名称列表
21. normFileList=spam.get_File_List(r"../data/normal")
22. spamFileList=spam.get_File_List(r"../data/spam")
23. testFileList=spam.get_File_List(r"../data/test")
24.
25. # 获取训练集中正常邮件与垃圾邮件的数量
```

```

26. normFilelen=len(normFileList)
27. spamFilelen=len(spamFileList)
28.
29. # 获得停用词表，用于对停用词过滤
30. stopList=spam.getStopWords()

```

**说明：**为接下来的计算过程准备各种数据结构，如：保存每封邮件的词频字典、保存每封邮件中出现词的列表、保存预测结果的列表 `testResult`；并将垃圾邮件训练集、正常邮件训练集、测试邮件集读入，计算训练集与测试集大小；读入停用词表。

## 6、计算词频

```

1. # main.py
2.
3. # 获得正常邮件中的词频
4. for fileName in normFileList:
5.     wordsList.clear()
6.     for line in open("../data/normal/"+fileName, 'r', encoding='gbk', errors='ignore'):
7.         # 过滤掉非中文字符
8.         rule=re.compile(r"^\u4e00-\u9fa5")
9.         line=rule.sub("",line)
10.        # 将每封邮件出现的词保存在 wordsList 中
11.        spam.get_word_list(line,wordsList,stopList)
12.        # 统计每个词在所有邮件中出现的次数
13.        spam.addToDict(wordsList, wordsDict)
14.    normDict=wordsDict.copy()
15.
16. # 获得垃圾邮件中的词频
17. wordsDict.clear()
18. for fileName in spamFileList:
19.     wordsList.clear()
20.     for line in open("../data/spam/"+fileName, 'r', encoding='gbk', errors='ignore'):
21.         rule=re.compile(r"^\u4e00-\u9fa5")
22.         line=rule.sub("",line)
23.         spam.get_word_list(line,wordsList,stopList)
24.         spam.addToDict(wordsList, wordsDict)
25. spamDict=wordsDict.copy()

```

**说明：**计算正常邮件与垃圾邮件中的词频。步骤为：依次读入邮件、使用正则表达式过滤非中文字符、调用 `get_word_list()` 方法分词，将每封邮件中出现的词保存在列表中、统计每个词的词频。

## 7、测试

```

1. # main.py
2.
3. # 测试邮件
4. for fileName in testFileList:
5.     testDict.clear()
6.     wordsDict.clear()

```



```

7.     wordsList.clear()
8.     for line in open("../data/test/"+fileName, 'r', encoding='gbk', errors='ignore'):
9.         rule=re.compile(r"[\u4e00-\u9fa5]")
10.        line=rule.sub("",line)
11.        spam.get_word_list(line,wordsList,stopList)
12.        spam.addToDict(wordsList, wordsDict)
13.        testDict=wordsDict.copy()
14.        #通过计算每个文件中 p(s|w)来得到对分类影响最大的 15 个词
15.        wordProbList = spam.getTest-
Words(testDict,spamDict,normDict,normFilelen,spamFilelen)
16.
17.        #对每封邮件得到的 15 个词计算贝叶斯概率
18.        p=spam.calBayes(wordProbList, spamDict, normDict)
19.        if(p>0.9):
20.            testResult.setdefault(fileName,1)
21.        else:
22.            testResult.setdefault(fileName,0)

```

**说明：**正常邮件与垃圾邮件词频统计完成后进行测试邮件集的贝叶斯概率计算。首先读入测试邮件集、使用正则表达式过滤非中文字符、根据停用词表过滤掉停用词、计算每个文件中的条件概率来得到对分类影响最大的 15 个词、对这 15 个词进行贝叶斯概率计算。若概率大于 0.9（阈值），则判为垃圾邮件（标签为 1），否则判为正常邮件（标签为 0）。

## 8、计算测试集准确率

```

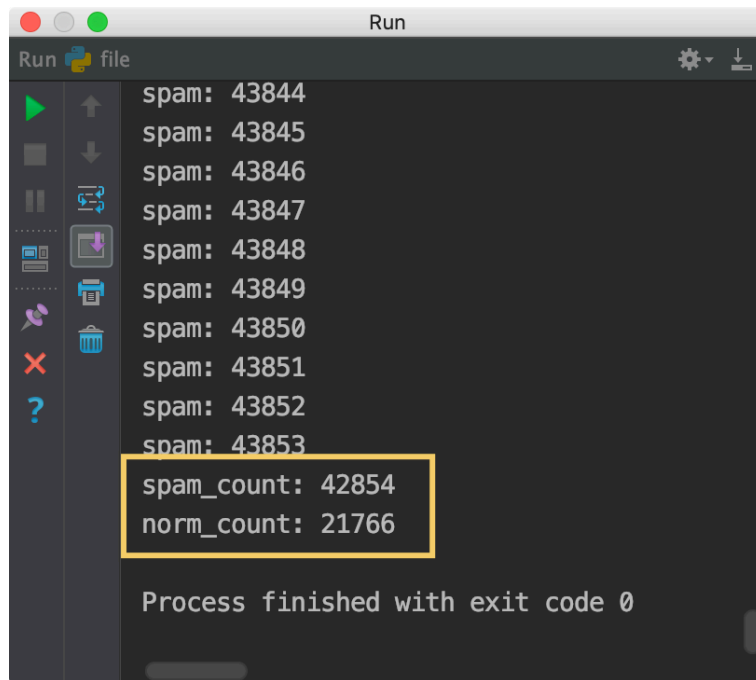
1. main.py
2.
3. #计算分类准确率（测试集中文件名低于 1000 的为正常邮件）
4. testAccuracy=spam.calAccuracy(testResult)
5. testOutput = open("./testOutput.txt", 'w', encoding="utf-8")
6.
7. for i,ic in testResult.items():
8.     if ((int(i) <= 1000) and (int(ic) == 0)) or ((int(i) > 1000)
and (int(ic) == 1)):
9.         print(i + "/" + str(ic) + " correct", file=testOutput)
10.    else:
11.        print(i + "/" + str(ic) + " wrong", file=testOutput)
12.
13. print("test accuracy: " + str(testAccuracy))

```

**说明：**调用 calAccuracy()方法计算测试集准确率，并将测试集判断结果输出到 testOutput.txt 文件保存。

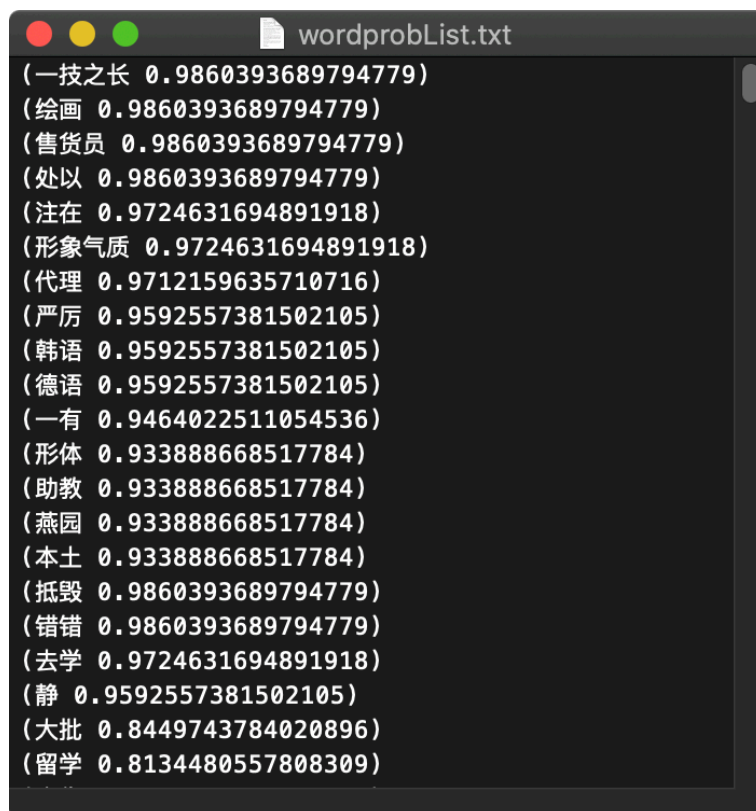
## 九、实验数据及结果分析

### 1、原始数据归档



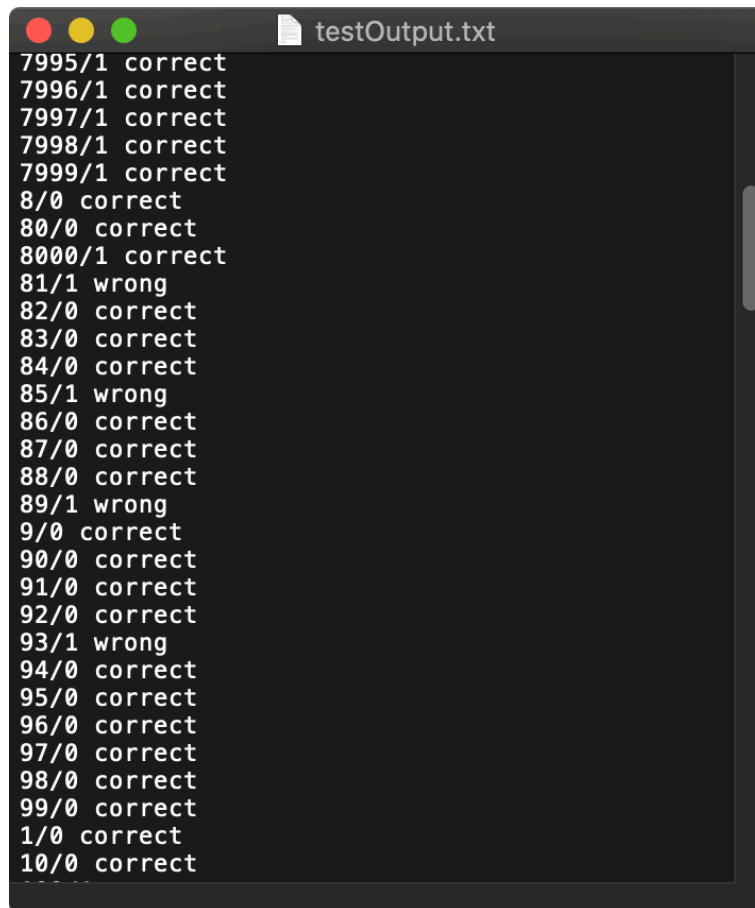
```
Run file
spam: 43844
spam: 43845
spam: 43846
spam: 43847
spam: 43848
spam: 43849
spam: 43850
spam: 43851
spam: 43852
spam: 43853
spam_count: 42854
norm_count: 21766
Process finished with exit code 0
```

## 2、分词概率计算结果




```
wordprobList.txt
(一技之长 0.9860393689794779)
(绘画 0.9860393689794779)
(售货员 0.9860393689794779)
(处以 0.9860393689794779)
(注在 0.9724631694891918)
(形象气质 0.9724631694891918)
(代理 0.9712159635710716)
(严厉 0.9592557381502105)
(韩语 0.9592557381502105)
(德语 0.9592557381502105)
(一有 0.9464022511054536)
(形体 0.933888668517784)
(助教 0.933888668517784)
(燕园 0.933888668517784)
(本土 0.933888668517784)
(抵毁 0.9860393689794779)
(错错 0.9860393689794779)
(去学 0.9724631694891918)
(静 0.9592557381502105)
(大批 0.8449743784020896)
(留学 0.8134480557808309)
```

## 3、测试集判断结果



```
7995/1 correct
7996/1 correct
7997/1 correct
7998/1 correct
7999/1 correct
8/0 correct
80/0 correct
8000/1 correct
81/1 wrong
82/0 correct
83/0 correct
84/0 correct
85/1 wrong
86/0 correct
87/0 correct
88/0 correct
89/1 wrong
9/0 correct
90/0 correct
91/0 correct
92/0 correct
93/1 wrong
94/0 correct
95/0 correct
96/0 correct
97/0 correct
98/0 correct
99/0 correct
1/0 correct
10/0 correct
```

#### 4、测试集准确性结果



```
Run ttss
/Volumes/B00TCAMP/袁昊男/学习/大二下/机器学习技术与应用/实验/实验1数据集和实验指导书/BayesSpam
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/pk/zl3ywwg96jg35h4br2_69qmc0000gn/T/jieba.cache
Loading model cost 1.233 seconds.
Prefix dict has been built successfully.
test accuracy: 0.951530612244898
Process finished with exit code 0
```

## 十、实验结论

本实验通过应用朴素贝叶斯分类模型，结合 jieba 库完成了对垃圾邮件的分类。得到了垃圾邮件中特定词汇出现的概率，测试集对训练好模型的识别准确率达到了 95.15%，可以通过调整阈值  $\alpha$  或增加训练样本来达到更高的准确率。

## 十一、总结及心得体会

思考题解答：本实验在估计概率时使用的是最大后验概率。

### **极大似然估计（MLE，频率学派）：**

- 定义：根据已知样本，希望通过调整模型参数来使得模型能够最大化样本情况出现的概率。
- 优点：比其他估计方法简单，大部分传统的机器学习算法都采用了该方法进行参数估计、收敛性，只要训练样本集够大，理论上可以接近无偏估计。
- 缺点：必须要假设分布模型，选择假设模型时必须非常慎重，若偏差太大会导致估计结果非常差。

### **最大后验概率（MAP，贝叶斯学派）：**

- 定义：最大化在给定数据样本的情况下模型参数的后验概率，是贝叶斯派模型参数估计的常用方法。
- 优点：考虑了参数本身的分布，也就是先验分布，融入了要估计量的先验分布在其中，故最大后验估计可以看做规则化的最大似然估计。
- 缺点：比极大似然估计方法更复杂。

## **十二、对本实验过程及方法、手段的改进建议**

实验部分课堂讲授部分较简单，学生需要花费更多的课下时间来学习相关的算法知识并实现。给出的邮件数据集数量太大，课程实验没有必要使用如此庞大的数据集，运行起来很费计算资源和时间。

**报告评分：**

**指导教师签字：**