基于先验为多项式分布的朴素贝叶斯的垃圾邮件分类

主讲人：王利猛 王瑾


目录
Content
01 需求分析
02 系统设计
03 系统实现
04 总结分析


1
需求分析
Demand analysis

需求分析

先验为多项式分布的朴素贝叶斯(MultinomialNB):

多项式朴素贝叶斯是先验为多项式分布的朴素贝叶斯。他的假设特征是由一个简单多项式分布生成的。多项式分布可以描述各种类型样本出现次数的概率，因此多项式朴素贝叶斯非常适合用于描述出现次数的特征。该模型常用于文本分类。


2
系统设计
System design.

总体设计方案

数据集预处理

- 加载数据
- 所有单词化为小写
- 删除停用词
- 词形还原

训练

- 按7:3进行数据集切分
- 特征提取形成词库
- 标准化

模型评估

- 生成ROC曲线


3
系统实现
System implementation

数据集预处理

```
[103] 1 #加载数据  0  ham正常邮件  1spam垃圾邮件
      2 data  =  pd.read_csv('/content/NB_mail/spam_ham_dataset.csv')
      3 data  =  data.iloc[:,2:]
      4 data.head()
```

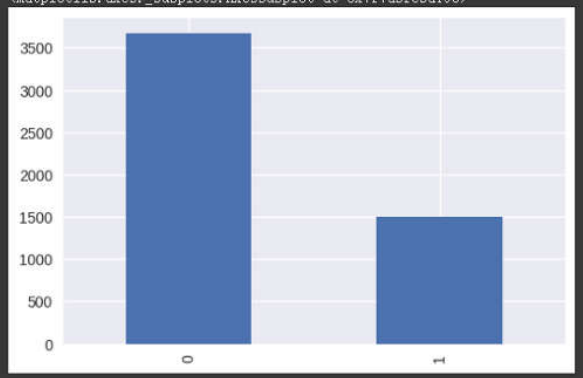
	text	label_num
0	Subject: enron methanol ; meter # : 988291\r\n...	0
1	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	Subject: photoshop , windows , office . cheap ...	1
4	Subject: re : indian springs\r\nthis deal is t...	0

```
1 print('这份数据包含{}条邮件'.format(data.shape[0]))
2 print('正常邮件一共有{}条'.format(data['label_num'].value_counts()[0]))
3 print('垃圾邮件一共有{}条'.format(data['label_num'].value_counts()[1]))
```

```
这份数据包含5171条邮件
正常邮件一共有3672条
垃圾邮件一共有1499条
```

```
[105] 1 plt.style.use('seaborn')
      2 plt.figure(figsize=(6,4),  dpi=100)
      3 data['label_num'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7abfcba190>
```



获得数据集

- 数据来源：

  https://github.com/Youngphone/spam_email

- 数据集大小：

  一个5.24MB的CSV文件；共5171条邮件，包括3672条正常邮件，1499条垃圾邮件

加载数据

- 训练标签：0代表正常邮件（ham）

  1代表垃圾邮件（spam）

数据集预处理

```
[106]  1 #邮件中含有大小写，先将单词替换为小写
       2 data['text'] = data['text'].str.lower()
       3 data.head()
```

◆将邮件中大写的单词替换成小写

```
[107]  1 #删除邮件中英语停用词
       2 #https://zhuanlan.zhihu.com/p/34671514
       3 import nltk
       4 nltk.download('stopwords')
       5 stop_words=set(stopwords.words('english'))
       6 stop_words.add('subject')
```

◆删除邮件中英语停用词

停用词：英语中比如a，the，he等；每个页面几乎都包含了这些词汇，如果搜索引擎它们当关键字进行索引，那么所有的网站都会被索引，而且没有区分度，所以一般把这些词直接去掉，不可当做关键词。

```
[108]  1 #词形还原函数
       2 #https://www.cnblogs.com/jclian91/p/9898511.html
       3 def text_process(text):
       4         tokenizer = RegexpTokenizer('[a-z]+')
       5         token = tokenizer.tokenize(text)
       6         lemmatizer = WordNetLemmatizer()
       7         token = [lemmatizer.lemmatize(w) for w in token if lemmatizer.lemmatize(w) not in stop_words]
       8         return token
```

```
[109]  1 nltk.download('wordnet')
       2 #用wordnet实现词形还原
       3 #https://zhuanlan.zhihu.com/p/26527203
       4 data['text'] = data['text'].apply(text_process)
       5 data.head()
```

◆词形还原

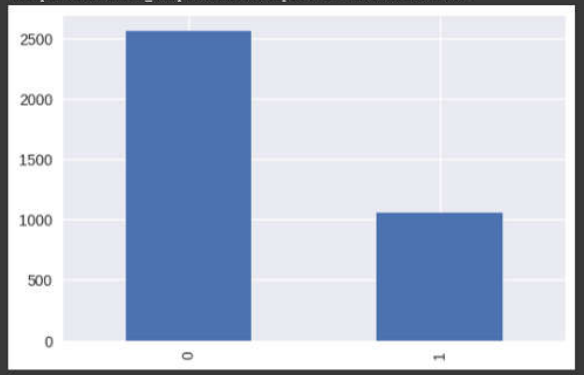
去掉单词的词缀，提取单词的主干部分，通常提取后的单词会是字典中的单词，比如，单词“cars”词形还原后的单词为“car”，单词“ate”词形还原后的单词为“eat”。

训 练

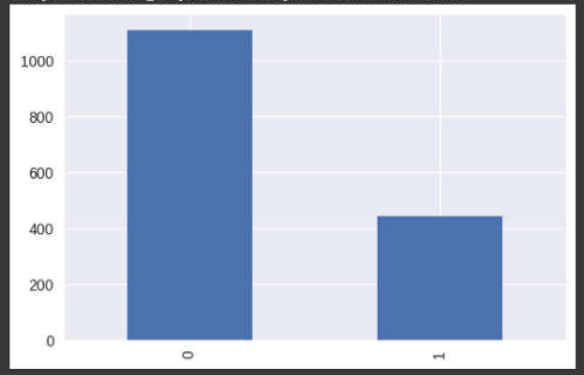
```
[110] 1 X = data['text']
      2 y = data['label_num']
      3 X_train, X_test, y_train,y_test = train_test_split(X,y,train_size = 0.7)#训测比7:3

[111] 1 train = pd.concat([X_train,y_train],axis=1)
      2 test = pd.concat([X_test,y_test],axis = 1)

[112] 1 ham_train = train[train['label_num'] == 0] # 正常邮件
      2 spam_train = train[train['label_num'] == 1] # 垃圾邮件

[113] 1 print('训练集含有{}封邮件，测试集含有{}封邮件'.format(train.shape[0], test.shape[0]))
      训练集含有3619封邮件，测试集含有1552封邮件
```

将数据集以7:3的比例划分成训练集和测试集，其中训练集含有3619封邮件，测试集含有1552封邮件。

```
1 #训练集中垃圾邮件与正常邮件数量
2 print(train['label_num'].value_counts())
3 plt.figure(figsize=(6, 4), dpi=100)
4 train['label_num'].value_counts().plot(kind='bar')

0    2562
1    1057
Name: label_num, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f7abdc5dc90>
```




```
1 #测试集中垃圾邮件与正常邮件数量
2 print(test['label_num'].value_counts())
3 plt.figure(figsize=(6, 4), dpi=100)
4 test['label_num'].value_counts().plot(kind='bar')

0    1110
1     442
Name: label_num, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f7abdb9aed0>
```



训练集中：

(0)正常邮件：2562

(1)垃圾邮件：1057

测试集中：

(0)正常邮件：1110

(1)垃圾邮件：442

训 练

```
[116] 1 #  各取30封组成词库
      2 ham_train_part  =  ham_train['text'].sample(30,  random_state=42)
      3 spam_train_part  =  spam_train['text'].sample(30,  random_state=42)
      4 part_words  =  []
      5 for  text  in  pd.concat([ham_train_part,spam_train_part]):
      6         part_words  +=  text
```

```
[117] 1 part_words_set  =  set(part_words)#去除重复单词
      2 print('单词表一共有{}个单词'.format(len(part_words_set)))

单词表一共有2440个单词
```

```
      1 #  将正常邮件与垃圾邮件的单词都整理为句子，单词间以空格相隔，CountVectorizer()的句子里，单词是以空格分隔的
      2 train_part_texts  =  [' '.join(text)  for  text  in  np.concatenate((spam_train_part.values,  ham_train_part.values))]
      3 #  训练集所有的单词整理成句子列表
      4 train_all_texts  =  [' '.join(text)  for  text  in  train['text']]
      5 #  测试集所有的单词整理成句子列表
      6 test_all_texts  =  [' '.join(text)  for  text  in  test['text']]
```

```
[119] 1 cv  =  CountVectorizer()
      2 #https://blog.csdn.net/weixin_38278334/article/details/82320307
      3 part_fit  =  cv.fit(train_part_texts)  #  以部分句子为参考
      4 train_all_count  =  cv.transform(train_all_texts)  #  对训练集所有邮件统计单词出现次数
      5 test_all_count  =  cv.transform(test_all_texts)  #  对测试集所有邮件统计单词出现次数
      6 tfidf  =  TfidfTransformer()
      7 train_tfidf_matrix  =  tfidf.fit_transform(train_all_count)#标准化
      8 test_tfidf_matrix  =  tfidf.fit_transform(test_all_count)
      9 #print(train_all_count)
     10 #print(train_tfidf_matrix)
```

1. 从训练集和测试集中各取30个“text”组成词库
2. 将训练集和测试集中的邮件整理成句子列表
3. 分别统计训练集和测试集中对应词库里的句子里单词出现的次数
4. 对结果进行标准化

训练

```
  (0, 1993)      0.19460262697058953
  (0, 1683)      0.2683531639348823
  (0, 1549)      0.232334571256264
  (0, 1544)      0.3292801481356707
  (0, 1460)      0.564485237005128
  (0, 863)       0.22872989770422503
  (0, 835)       0.38374374611833206
  (0, 443)       0.25846264631371035
  (0, 311)       0.2986226703818847
  (0, 54)        0.23132872202616178
  (1, 2249)      0.162032615891801
  (1, 2202)      0.6928634875675248
  (1, 1521)      0.36581578860265457
  (1, 1433)      0.20245954775609362
  (1, 1418)      0.3216092244551748
  (1, 1403)      0.21353126050694476
  (1, 882)       0.18123601861703367
  (1, 521)       0.17844535175453674
  (1, 24)        0.3242795535323569
  (2, 2448)      0.06888272513494491
  (2, 2427)      0.053723435193662517
  (2, 2383)      0.3333133772512531
  (2, 2379)      0.10370095302650319
  (2, 2267)      0.06787210134672222
  (2, 2202)      0.27409352332891673
  :      :
```

最终形成

训练集的X为3619*2440矩阵，3619指有3619个训练样本，2440指该样本（即经过预处理后的text）的每个单词对应词库中单词出现的次数。Y为3619*1向量，对应该样本的有监督学习标签。同理，测试集的X为1552*2440矩阵，Y为1552*1向量。

将X矩阵中的元素（出现次数）进行标准化

标准化结果的稀疏矩阵如图所示。

评 估

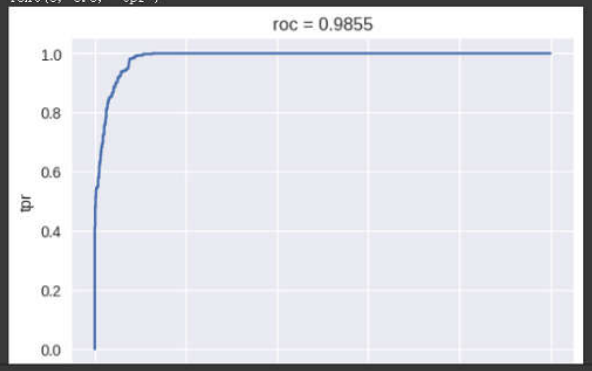
```
[120] 1 #建立模型
      2 #https://blog.csdn.net/TeFuirnever/article/details/100125386
      3 model = MultinomialNB()
      4 model.fit(train_tfidf_matrix, y_train)
      5 model.score(test_tfidf_matrix,y_test)
```

```
0.9368556701030928
```

```
[121] 1 y_pred = model.predict_proba(test_tfidf_matrix)
      2 fpr, tpr, thresholds = roc_curve(y_test, y_pred[:, 1])
      3 auc = auc(fpr, tpr)
```

```
1 # roc 曲线
2 plt.figure(figsize=(6, 4), dpi=100)
3 plt.plot(fpr, tpr)
4 plt.title('roc = {:.4f}'.format(auc))
5 plt.xlabel('fpr')
6 plt.ylabel('tpr')
```

```
Text(0, 0.5, 'tpr')
```



模型评估：

ROC曲线图是反映敏感性与特异性之间关系的曲线。横坐标X轴为 1 – 特异性，也称为假阳性率（误报率），X轴越接近零准确率越高；纵坐标Y轴称为敏感度，也称为真阳性率（敏感度），Y轴越大代表准确率越好。

曲线下方部分的面积被称为AUC（Area Under Curve），用来表示预测准确性，AUC值越高，也就是曲线下方面积越大，说明预测准确率越高。曲线越接近左上角（X越小，Y越大），预测准确率越高。

利用MultinomialNB()函数建立模型，训练后可知左侧输出的分数接近0.94，即正确率在94%左右。AUC接近1.


4
总结分析

总结分析

随着互联网和多媒体技术的进一步发展，邮件分类技术将与图像识别、语音识别融合，比如像邮件的分类、语音邮件的分类,多媒体数据库索引等。这就进一步要求邮件分类技术在文本的处理方法、.克服噪音干扰、分类精度方面有进一步的提高。总之,邮件分类技术也会是将来研究的热点问题,它的发展同时会推动数据库技术,网络技术的共同进步。


欢 迎 指 正