**垃圾短信过滤案例**

# 1简介

随着全球手机使用量的增长，一种创造垃圾短信的新途径已经为声名狼藉的营销市场开放了。这些广告商利用短信服务（SMS）文本信息，以潜在消费者为目标，给他们发送不需要的广告，即垃圾短信。这种类型的垃圾短信十分麻烦，它与垃圾邮件不同，因为很多手机用户需要为收到的每一条短信缴费。

研究一种可以过滤垃圾短信的分类算法，将会为移动电话供应商提供一种很好用的工具。由于朴素贝叶斯算法在垃圾邮件的过滤上，已经获得了很大成果，我们自然会考虑将此算法推广到垃圾短信过滤上。

然而，相对于垃圾邮件来说，垃圾短信的过滤有额外的挑战。由于短信通常限制为160个字符，可以用来确定一条短信是否为垃圾短信的文本量减少了，另外很多人喜欢采用短信术语简写形式，这进一步模糊了合法信息和垃圾信息的界限。

本实验使用从网站上收集的垃圾短信改编数据。

# 2环境及数据准备

## 2.1安装依赖

本案例使用了第三方提供的可视化工具wordcloud，请大家完成此工具包的安装，可以使用pip实现。

import pandas as pd  
import string  
import matplotlib.pyplot as plt  
from sklearn.feature\_extraction.text import CountVectorizer  
from wordcloud import WordCloud  
from sklearn import metrics  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.model\_selection import train\_test\_split

## 2.2数据说明

该数据集包含短信的文本信息，而且带有表明该信息是否为垃圾短信的标签。垃圾短信标记为spam，而非垃圾短信标记为ham。

垃圾短信和非垃圾短信在使用的词语方面有较大的差异，我们的朴素贝叶斯分类器将利用词频差异来确定短信消息更像是垃圾短信还是非垃圾短信，根据短信中所有单词提供的证据计算垃圾短信和非垃圾短信的概率。

## 2.3数据读入

我们使用pandas提供的read\_csv方法导入数据，保存为一个对象，并输出前五行，验证导入成功。

sms\_raw = pd.read\_csv(**"D:/input/sms\_spam.csv"**)  
sms\_raw.head()

使用dataframe中的describe方法，可以看到数据文件的特征。

sms\_raw.describe()

可以看到，数据文件包含了5567条短信，每条短信有两个特征变量：type和text。其中type的值为ham或spam，而text存储整个短信文本。

查看两个特征的数据类型。

sms\_raw.dtypes

# 3数据预处理

## 3.1特征编码

标签特征type是一个字符串特征。由于它是一个分类变量，将其转换为一个数值型变量将有利于后续的数据处理。其中1表示为垃圾短信，0表示为非垃圾短信。

sms\_raw[**'type'**] = pd.Series(sms\_raw[**'type'**].factorize()).iloc[0]

查看编码后的特征分布。

sms\_raw.groupby(**'type'**).count()

## 3.2无效数据清洗

短信就是由词、空格、数字和标点符号组成的文本字符串。处理这种类型的复杂数据需要大量的思考和工作，一方面需要考虑如何去除数字和标点符号，如何处理没有意义的单词如and、but、or等，以及如何将句子分解为单个的单词。

幸运的是，scikit-learn向我们提供了这些方法。

我们的数据集现在包含5568条短信的原始文本内容。在将文本内容分解为单词之前，我们需要进行一些清理步骤去除标点符号和可能会影响结果的其他字符。

首先我们利用map方法去除文本内容中的标点符号和数字，并将文本中所有字符串都转换为小写形式。

*# 将所有字母转换为小写*def tolower(text):  
 return text.lower()  
*# 去除标点符号和数字*def removePunctuation(text):  
 return text.translate({None: string.punctuation+(**'1234567890'**)})  
*# 进行变换*sms\_raw[**'text'**] = sms\_raw[**'text'**].map(removePunctuation).map(tolower)

在分析文本数据时，一个常见的做法就是去除填充词，比如and、to、but等，这些词称为停用词（stop word）。我们使用sklearn包中的方法去除停用词并计算文本的词频分布。

count\_vect = CountVectorizer(stop\_words=**"english"**,decode\_error=**'ignore'**)  
sms\_counts = count\_vect.fit\_transform(sms\_raw[**'text'**])

查看变换后的数据集大小。

sms\_counts.shape

由于原始词频矩阵维度过高，我们考虑使用TF-IDF进行特征提取，设定最小词频数为5.

tfidf\_vect = TfidfVectorizer(stop\_words=**"english"**,decode\_error=**'ignore'**,min\_df=5)  
sms\_tfidf = tfidf\_vect.fit\_transform(sms\_raw[**'text'**])

查看现在数据集的大小

sms\_tfidf.shape

可以看到，数据维度显著降低。

## 3.3数据集划分

此前的实验中，我们已经多次进行过训练集和测试集的划分，本案例同样需要进行这一步。

train\_x, test\_x, train\_y, test\_y = train\_test\_split(sms\_tfidf, sms\_raw[**'type'**], test\_size=0.3)

查看训练集和测试集垃圾短信和非垃圾短信的占比是否相同。

train\_y.value\_counts()/len(train\_y)  
test\_y.value\_counts()/len(test\_y)

可以看到，训练集和测试集两种样本的分布比例基本相同，数据集划分合理。

## 3.4可视化

词云是一种可视化地描绘单词出现在文本数据中频率的方式。词云是由随机分布在词云图中的单词构成的，经常出现在文本中的单词会以较大的字体呈现，而不太常见的单词将会以较小的单词呈现。

最近，这种图愈发流行了，因为它提供了一种观察社交媒体网站上热门话题的方式。

Wordcloud包提供了一个简单的函数来创建这种类型的图形，我们将应用这个函数使短信中的单词类型可视化。比较垃圾短信和非垃圾短信的词云有助于我们预测朴素贝叶斯分类器是否可能成功。

查看全部信息的词云：

wc = WordCloud()

wc.generate(**''**.join(sms\_raw[**'text'**]))  
plt.imshow(wc)

plt.show() //查看垃圾短信的词云：

wc.generate(**''**.join(sms\_raw[**'text'**][sms\_raw[**'type'**]==1]))  
plt.imshow(wc)

plt.show() //查看非垃圾短信的词云

wc.generate(**''**.join(sms\_raw[**'text'**][sms\_raw[**'type'**]==0]))  
plt.imshow(wc)

plt.show() //可以看出，垃圾短信和非垃圾短信的词云有很大的不同。

# 4构建和训练模型

此前已经将原始短信转换为一个标准数据表，此时便使用sklearn包中的朴素贝叶斯分类器快速构建和训练模型。

sms\_classifier = MultinomialNB().fit(train\_x,train\_y)

# 5模型性能评估

为了评估短信分类器，我们用测试集来测试模型性能。

首先获取预测值：

*# 对测试集的样本进行预测*test\_pred = sms\_classifier.predict(test\_x)

与真实值比较，查看模型对正样本和负样本的预测情况。

metrics.confusion\_matrix(test\_y, test\_pred)

可以看到，有9条非垃圾短信被错误地归类为垃圾短信，比例为0.62%，而有64条垃圾短信被错误地划分为非垃圾短信，比例为29%。考虑到我们在这个案例中几乎没有对数据进行其他专业的处理，这个水平是可以接受的。

同时，我们也可以查看其他性能参数：

print(metrics.classification\_report(test\_y, test\_pred))