**程序报告**

学号：2013750 姓名：管昀玫

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

====================================================================

垃圾短信 (Spam Messages，SM) 是指未经过用户同意向用户发送不愿接收的商业广告或者不符合法律规范的短信。本实验要求识别出垃圾短信息。

1) 任务提供包括数据读取、基础模型、模型训练等基本代码  
2) 参赛选手需完成核心模型构建代码，并尽可能将模型调到最佳状态  
3) 模型单次推理时间不超过 10 秒

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

====================================================================

**1.停用词**

停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为 Stop Words（停用词）。  
这些停用词都是人工输入、非自动化生成的，生成后的停用词会形成一个停用词库。

使用停用词可以降低无关词语的干扰。

**2.文本向量化方法**

目前拥有的数据是长度不统一的文本数据，而绝大多数机器学习算法需要的输入是向量，因此文本类型的数据需要经过处理得到向量。  
我们可以借助 sklearn 中 **CountVectorizer** 来实现文本的向量化，CountVectorizer 实际上是在统计**每个词出现的次数**，这样的模型也叫做**词袋模型**。

与 CountVectorizer 类似的还有 TfidfVectorizer 。  
TF-IDF 算法是创建在这样一个假设之上的：  
对区别文档最有意义的词语应该是那些在文档中出现频率高的词语，因此选择特征空间坐标系取 TF 词频作为测度，就可以体现同类文本的特点。  
另外考虑到单词区别不同类别的能力，TF-IDF 法认为一个单词出现的文本频数越小，它区别不同类别文本的能力就越大。  
因此引入了逆文本频度 IDF 的概念，以 TF 和 IDF 的乘积作为特征空间坐标系的取值测度，并用它完成对权值 TF 的调整，调整权值的目的在于突出重要单词，抑制次要单词。  
在本质上 IDF 是一种试图抑制噪声的加权，并且单纯地认为文本频率小的单词就越重要，文本频率大的单词就越无用。

**3.朴素贝叶斯算法**

朴素贝叶斯实现分类的原理是基于贝叶斯公式，给定一个样本，计算该样本条件下每个类别的条件概率。具体公式此处不再赘述。

**4.构建Pipeline**

构建 PipleLine 可以将数据处理和数据分类结合在一起，这样输入原始的数据就可以得到分类的结果，方便直接对原始数据进行预测。

5.数据归一化

使用StandardScaler 或者 MaxAbsScaler，对数据进行归一化，可以使梯度下降收敛更快。

1. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

====================================================================

# 导入相关的包

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.externals import joblib

from sklearn.preprocessing import MaxAbsScaler

from sklearn.preprocessing import StandardScaler

# 数据集的路径

data\_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms\_pub.csv"

# 读取数据

sms = pd.read\_csv(data\_path, encoding='utf-8')

sms\_pos = sms[(sms['label'] == 1)]

sms\_neg = sms[(sms['label'] == 0)].sample(frac=1.0)[: len(sms\_pos)]

sms = pd.concat([sms\_pos, sms\_neg], axis=0).sample(frac=1.0)

def read\_stopwords(stopwords\_path):

"""

读取停用词库

:param stopwords\_path: 停用词库的路径

:return: 停用词列表

"""

with open(stopwords\_path, 'r', encoding='utf-8') as f:

stopwords = f.read()

stopwords = stopwords.splitlines()

return stopwords

# 停用词库路径

stopwords\_path = r'scu\_stopwords.txt'

# 读取停用词

stopwords = read\_stopwords(stopwords\_path)

# 构建训练集和测试集

X = np.array(sms.msg\_new)

y = np.array(sms.label)

pipeline = Pipeline([

('tfidf', TfidfVectorizer(token\_pattern=r"(?u)\b\w+\b", stop\_words=stopwords)),

('MaxAbsScaler', MaxAbsScaler()),

('clf', MultinomialNB())

])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42, test\_size=0.6)

pipeline.fit(X\_train, y\_train)

joblib.dump(pipeline, 'results/pipeline.model')

y\_pred = pipeline.predict(X\_test)

print("在测试集上的 f1-score ：")

print(metrics.f1\_score(y\_test, y\_pred))

print('在测试集上的准确率：')

print(metrics.accuracy\_score(y\_test, y\_pred))

# 在所有的样本上训练一次，充分利用已有的数据，提高模型的泛化能力

pipeline.fit(X, y)

# 保存训练的模型，请将模型保存在 results 目录下

from sklearn.externals import joblib

pipeline\_path = 'results/pipeline.model'

joblib.dump(pipeline, pipeline\_path)

# 加载训练好的模型

from sklearn.externals import joblib

# ------- pipeline 保存的路径，若有变化请修改 --------

pipeline\_path = 'results/pipeline.model'

# --------------------------------------------------

pipeline = joblib.load(pipeline\_path)

def predict(message):

"""

预测短信短信的类别和每个类别的概率

param: message: 经过jieba分词的短信，如"医生 拿 着 我 的 报告单 说 ： 幸亏 你 来 的 早 啊"

return: label: 整数类型，短信的类别，0 代表正常，1 代表恶意

proba: 列表类型，短信属于每个类别的概率，如[0.3, 0.7]，认为短信属于 0 的概率为 0.3，属于 1 的概率为 0.7

"""

label = pipeline.predict([message])[0]

proba = list(pipeline.predict\_proba([message])[0])

return label, proba

1. **实验结果**

（实验结果，必填）

====================================================================



1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

====================================================================

CountVectorizer的参数需要自己调。以下列举几个有用的参数：

*ngram\_range：tuple (min\_n, max\_n)*

要提取的不同n-gram的n值范围的下边界和上边界。 将使用n的所有值，使得min\_n <= n <= max\_n。

max\_df： float in range [0.0, 1.0] or int, default=1.0

在构建词汇表时，忽略文档频率严格高于给定阈值的术语（语料库特定的停用词）。 如果是float，则参数表示文档的比例，整数绝对计数。 如果词汇表不是None，则忽略此参数。

min\_df：float in range [0.0, 1.0] or int, default=1

构建词汇表时，请忽略文档频率严格低于给定阈值的术语。 该值在文献中也称为截止值。 如果是float，则参数表示文档的比例，整数绝对计数。 如果词汇表不是None，则忽略此参数。