程序报告

学号：2213041　　姓名：李雅帆

一、问题重述

==========

1.实验背景

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。随着手机的普及，垃圾短信在日常生活日益泛滥，已经严重的影响到了人们的正常生活娱乐，乃至社会的稳定。在面对量级如此巨大的短信数据时，为了保证更良好的用户体验，如何从数据中挖掘出更多有意义的信息为人们免受垃圾短信骚扰成为当前亟待解决的问题。

==========

2.实验要求

我们的实验要通过在所给数据集上进行机器学习模型训练，实现对垃圾短信的分类识别功能,根据一段中文文本（ 200 个中文字符以内），预测这段文本是否为垃圾短信。要求如下：

（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

二、设计思想

==========

1.设计思想与流程

（1）导入必要的库和设置环境变量：

代码开始时导入必要的库，并设置环境变量。

（2）读取停用词库：

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）。使用 read_stopwords() 函数从 scu_stopwords.txt 读取停用词库，该函数返回一个停用词列表。

（3）构建文本处理 Pipeline：

使用 Pipeline 类来构建文本处理流程，包括文本向量化和分类器模型的组合。目前使用了词袋模型（CountVectorizer）和朴素贝叶斯分类器（MultinomialNB）。

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；朴素贝叶斯是一种基于贝叶斯公式的监督学习算法，并假设每个特征是独立的，该方法在“垃圾

邮件分类”、“恶意邮件检测”等领域有着广泛应用。

（4）模型训练与评估：

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。使用 pipeline.fit() 函数在训练集上训练模型，并使用测试集对模型进行评估，输出混淆矩阵、分类报告以及 F1 分数。

（5）模型保存与加载：

使用 joblib.dump() 函数将训练好的模型保存到指定路径；使用 joblib.load() 函数加载保存的模型。

（6）短信分类预测：

定义了 predict() 函数，接收一个短信文本，使用训练好的模型进行分类预测，并返回分类标签和每个类别的概率。

==========

2.优化方向

（1）停用词处理改进：

停用词库中的读取部分已经完成，但在实际使用中，可以考虑使用更全面、精准的停用词库，或者动态地根据数据集来选择停用词。另外，可以尝试去除数字、标点符号等非自然语言信息。

（2）特征提取方法改进：

当前代码使用的是简单的词袋模型（CountVectorizer）进行特征提取，可以尝试其他方法如 TF-IDF（TfidfVectorizer）等，这些方法可能会更好地捕捉关键词的重要性。

（3）模型选择和调参：

目前使用了朴素贝叶斯分类器（MultinomialNB），但没有进行参数调优。可以使用交叉验证等方法来选择最佳的参数，比如平滑参数 alpha 等。另外，除了朴素贝叶斯，还可以尝试其他算法如支持向量机（SVM）、随机森林等，以提高分类性能。

（4）模型评估改进：

代码中使用了混淆矩阵和分类报告来评估模型性能，可以考虑使用交叉验证等更加严谨的评估方法，确保评估结果的可靠性。

（5）异常处理和数据验证：

没有对输入数据进行异常处理或者验证，可以添加一些代码来确保输入数据的格式和内容正确。

三、代码内容

==========

1.读取停用词。

```
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
# ----------  停用词库路径，若有变化请修改  ----------
stopwords_path = r'scu_stopwords.txt'
# ------------------------------------------------
```

```
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，如 ['嘿', '很', '乎', '会', '或']
    """
    stopwords = []
    # -----------  请完成读取停用词的代码  ------------
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()
    #----------------------------------------------


    return stopwords
#  读取停用词
stopwords = read_stopwords(stopwords_path)
```

==========

2．构建一个机器学习流水线（Pipeline），用于文本分类任务。

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import ComplementNB
import pandas as pd
import numpy as np

from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import ComplementNB

# pipline_list 用于传给 Pipline 作为参数
pipeline_list = [

    ('cv', CountVectorizer(token_pattern=r"(?u)\b\w+\b", stop_words=stopwords)),
    ('classifier', MultinomialNB())
]
```

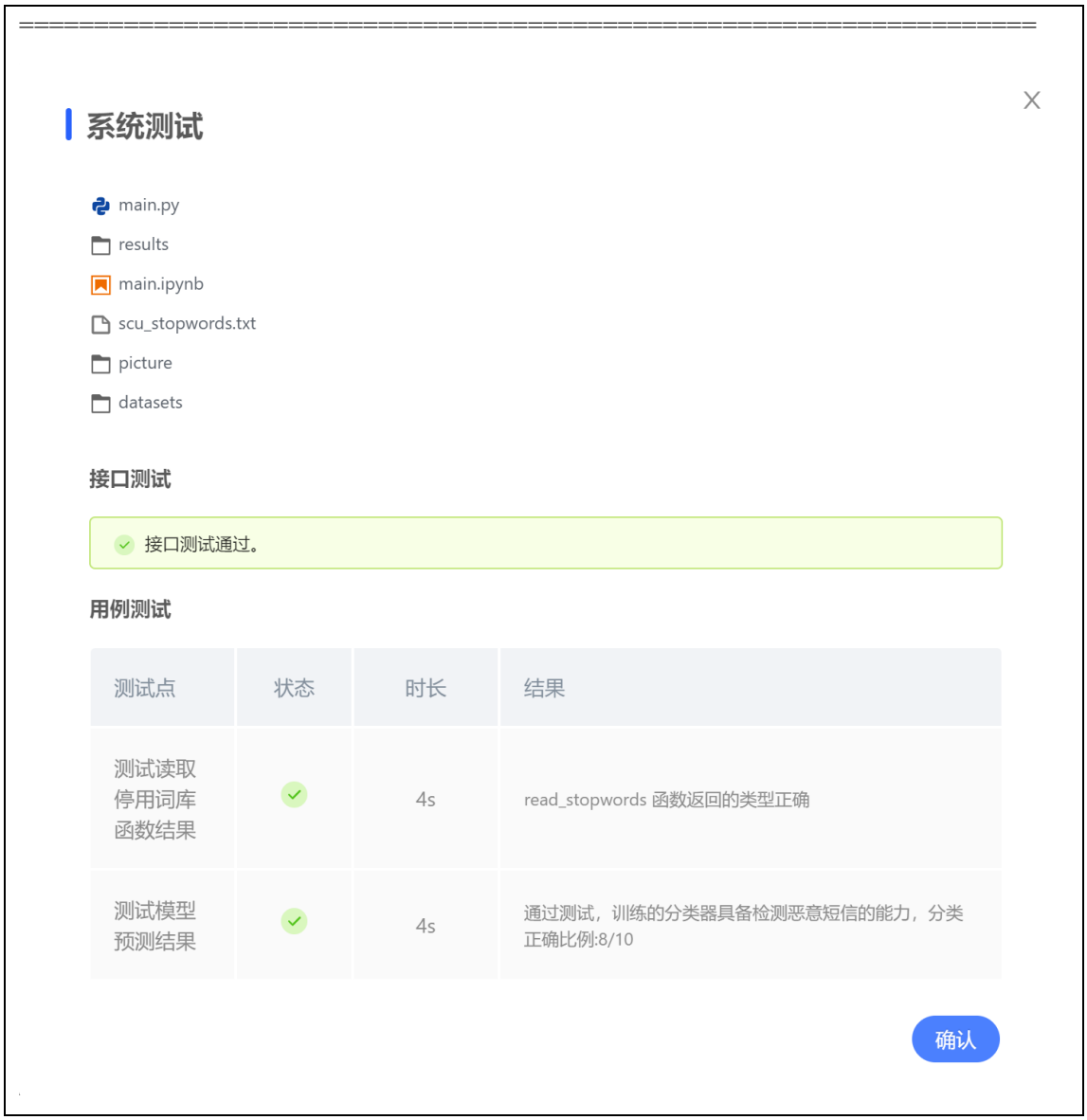
==========

3. 针对构建的机器学习流水线（Pipeline）进行训练和评估。

```
#  搭建  pipeline
pipeline = Pipeline(pipeline_list)
```

```
# 训练 pipeline
pipeline.fit(X_train, y_train)

# 对测试集的数据集进行预测
y_pred = pipeline.predict(X_test)
# 在测试集上进行评估
from sklearn import metrics
print("在测试集上的混淆矩阵：")
print(metrics.confusion_matrix(y_test, y_pred))
print("在测试集上的分类结果报告：")
print(metrics.classification_report(y_test, y_pred))
print("在测试集上的 f1-score ：")
print(metrics.f1_score(y_test, y_pred))

==========================================================================
4．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
# 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
import joblib
pipeline_path = 'results/pipeline.model'
joblib.dump(pipeline, pipeline_path)

==========================================================================
5. 加载之前保存的训练好的模型，并定义 predict()函数用于对新的短信文本进行分类预测。
# 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):
    label = pipeline.predict([message])[0]
    proba = list(pipeline.predict_proba([message])[0])
    return label, proba

==========================================================================
6.测试用例
label, proba = predict('医生 拿 着 我 的 报告单 说 ： 幸亏 你 来 的 早 啊')
print(label, proba)
```

四、实验结果



五、总结

1.改用 TF-IDF + 标准化 StandardScaler 进行数据处理，并将原来的 MultinomialNB 基于多项式的朴素贝叶斯换成 ComplementNB 后通过测试发现：原平台给出的方法识别恶意短信的能力更强，分类的正确比例更高。

2.了解了文本分类的基本概念和流程，包括数据准备、特征提取、模型选择、训练和评估等步骤。

3.了解了常见的模型评估指标，包括准确率、召回率、F1 分数等，以及如何使用交叉验证和网格搜索等技术来调优模型参数。