程序报告

学号：2213045　　姓名：胡进喆

一、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

可以使用基于 Python 的 Pandas、Numpy、Sklearn 等库进行相关特征处理，使用 Sklearn 框架训练分类器，也可编写深度学习模型，使用过程中请注意 Python 包（库）的版本。

二、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。

在此次实验里，采用的仍是四川大学实验室停用词，并未做更改。

2.文本向量化方法

向量化文本数据是一种常见的预处理步骤，用于将文本转换为数值型向量，以便机器学习法能够处理。CountVectorizer 和 TfidfVectorizer 是常用的工具之一。

CountVectorizer 将文本转换为词频矩阵，其中每个单词的出现次数被计算并编码为向量的一个元素。这种方法简单有效，但不考虑单词在语料库中的重要性。

相比之下，TfidfVectorizer 使用了 TF-IDF 算法，这个算法考虑了词语在文档中的频率以及在整个语料库中的稀有程度。这样的处理方式可以突出重要单词并抑制常见但无意义的单词，使得向量化结果更能反映文本的特点。

总的来说，CountVectorizer 适用于简单的文本处理任务，而 TfidfVectorizer 更适用于需要考虑单词重要性的场景，如文本分类和信息检索。

3.朴素贝叶斯算法

朴素贝叶斯算法基于贝叶斯公式，通过计算后验概率来进行分类。其核心假设是所有特征之间相互独立。这意味着给定类别的情况下，每个特征对于分类的影响是相互独立的，这种假设简化了模型的计算。

在垃圾邮件分类或恶意邮件检测等任务中，朴素贝叶斯算法可以用于判断一封邮件是否属于某个类别，如垃圾邮件或正常邮件。它的工作原理是基于先验概率和条件概率：

先验概率：指在没有任何其他信息的情况下，某个事件发生的概率。在垃圾邮件分类中，先验概率就是某封邮件是垃圾邮件或正常邮件的概率。

条件概率：指在已知其他相关信息的情况下，某个事件发生的概率。在朴素贝叶斯算法中，条件概率指的是给定类别的情况下，每个特征出现的概率。

朴素贝叶斯算法根据贝叶斯公式计算后验概率，并选择具有最高后验概率的类别作为预测结果。在实际应用中，需要通过训练数据集来估计先验概率和条件概率，然后用这些概率来进行分类。

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

数据归一化是一种常见的数据预处理技术，它通过将数据按照一定的规则进行缩放，使得数据分布在一个特定的范围内，有助于提高模型的训练效率和性能。其中，StandardScaler 和 MaxAbsScaler 是两种常用的归一化方法。

StandardScaler：.使用 StandardScaler 时，将数据进行均值归一化，使得数据的均值为 0，标准差为 1。具体操作是，对于每个特征，将其减去该特征的均值，然后除以该特征的标准差。这种归一化方式适用于特征的分布近似为高斯分布的情况，同时不受异常值的影响。

MaxAbsScaler：使用 MaxAbsScaler 时，将数据按最大绝对值进行缩放，使得数据的最大绝对值为 1。具体操作是，对于每个特征，将其除以该特征的最大绝对值。这种归一化方式适用于特征的分布不受限制的情况，可以保留数据的原始分布。

对于梯度下降等迭代优化算法，数据归一化可以使得各个特征的权重更新更加平稳，避免某些特征对模型训练过程的影响过大，从而加快模型收敛速度。在实际应用中，根据数据的分布情况和模型的要求，选择合适的归一化方法进行数据预处理，有助于提高模型的性能和稳定性。

三、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）
==========

```
    # 忽略警告信息
import warnings
warnings.filterwarnings('ignore')


# 设置环境变量，禁用HDF5 文件锁定
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"


# 导入 pandas 库，用于数据处理
import pandas as pd


# 导入 numpy 库，用于数值计算
import numpy as np


# ----------------- 导入相关的库 -----------------
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import ComplementNB
from sklearn.preprocessing import StandardScaler


# 构建训练集和测试集
from sklearn.model_selection import train_test_split


# 在测试集上进行评估
from sklearn import metrics


# ---------- 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']
        """
        # ----------- 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


    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


    # 读取停用词
    stopwords = read_stopwords(stopwords_path)


    # pipline_list 用于传给 Pipline 作为参数
    pipeline_list = [
        # --------------------------- 需要完成的代码 ------------------------------


        # ========================= 以下代码仅供参考 ============================
```

```
        ('cv', CountVectorizer(token_pattern=r"(?u)\b\w+\b", stop_words=stopwords,ngram_range=(1,3))),
        ('classifier', MultinomialNB())
        # ========================================================================


        # ------------------------------------------------------------------------
    ]


    # 搭建 pipeline
    pipeline = Pipeline(pipeline_list)


    # 数据集的路径
    data_path = "./sms_pub.csv"
    # 读取数据
    sms = pd.read_csv(data_path, encoding='utf-8')
    # 显示前 5 条数据
    sms.head()
    # 显示数据集的一些信息
    sms.groupby('label').describe()


    X = np.array(sms.msg_new)
    Y = np.array(sms.label)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=11, test_size=0.1)
    print("总共的数据大小", X.shape)
    print("训练集数据大小", X_train.shape)
    print("测试集数据大小", X_test.shape)


    # 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)


```

```
105.
106. print("在测试集上的混淆矩阵：")
107. print(metrics.confusion_matrix(Y_test, Y_pred))
108. print("在测试集上的分类结果报告：")
109. print(metrics.classification_report(Y_test, Y_pred))
110. print("在测试集上的 f1-score ： ")
111. print(metrics.f1_score(Y_test, Y_pred))
112.
113. # 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
114. pipeline.fit(X, Y)
115. # 保存训练的模型，请将模型保存在 results 目录下
116. import joblib
117.
118. pipeline_path = 'results/pipeline.model'
119. joblib.dump(pipeline, pipeline_path)
120.
121. # 加载训练好的模型
122. import joblib
123.
124. # ------- pipeline 保存的路径，若有变化请修改 --------
125. pipeline_path = 'results/pipeline.model'
126. # ----------------------------------------------------
127. pipeline = joblib.load(pipeline_path)
128.
129.
130. # 测试用例
131.
132.
133. result_list = ['2015 年 招标 师 考试 辅导 招生 方',
134.                '南京 游泳 培训 泳动 奇迹 游泳 培训 一对二 教学 成人 初学者 包教包会',
135.                '宠物 乘坐 飞机 需要 提前 预定 有 氧舱',
136.                'SDOUG 目前 所有 报名 通道 全部 截止',
137.                '本 公司 新到 各种规格 辐射 松 ， 澳松 ， 花旗 松 ， 无节材 ， 价格 优惠 ！ 欢迎 各位 新 老
    客户 来 人 来电 选购 … … 上海 傲寒 国际贸易 有限公司 业务经理 ： 陈强',
138.                '是否 考 金融 之类 研究生 值得 思考',
```

```
            '亚马逊 在线 零售商 公布 第二季度 业绩 实现 了 盈利',
            '本 宝宝 不 还是 进去 了 hhhhh',
            '今年 有 17 名 台湾 大学生 走进 嘉兴',
            '然而 自己 并 不是 从事 医生 这个 职业 只是 恰好 有 一身 行头 而已 看 完医龙 后 对 这
    份 职业 真的 是 充满 了 敬意 …',]

for i in result_list:
    label, proba = predict(i)
    print(label, proba)
```

四、实验结果

（实验结果，必填）

==========

五、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**）

==========

1. 是否达到目标预期？

达到，经过 pycharm 计算，在测试集上的 f1-score 可以达到较高的 94%

2. 改进的方向

（1）调节 CountVectorizer 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。

（2）更换更好的停用词库，选择南开大学实验室的停用词库

（3）采用更好的数据归一化方法，此次实验采用的是：
*('scaler', StandardScaler(with_mean=False)),*可以尝试采用 MaxAbsScaler 的归一化方法

（4）适当调节分类器的参数，提高模型的表现，例如：

```
# 定义要搜索的参数网格
param_grid = {
    'classifier__alpha': [0.1, 0.5, 1.0]  # 在这里添加你希望搜索的参数值
}
```

3. 实现过程中遇到的困难

（1）主要是对参数的不断调试，找到最佳的模型参数消耗了大量的时间：将 **ngram_range** 的范围改为（1,3）时，在 MO 上的运行超出了预期的运行时间，导致最后并没有得到良好的实验结果

（2）选择合适的优化方法，很多方法对于模型性能的优化都没有得到多少的提升，在选取方法方面消耗了大量时间