程序报告

学号：2211290　　姓名：姚知言

一、问题重述

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，构建垃圾短信识别模型。通过补充读取停用词的代码以及 pipeline_list 的实现，调节文本向量化参数，对数据进行归一化，调节分类器参数，更换停用词库等方式，提高垃圾短信识别模型正确率。

二、设计思想

首先确定数据集的划分，约定为 random_state=55，测试集占比 test_size=0.2，进行以下实验。
对停用词读取模块进行编写，完成对停用词文档的读取以及按行分割成为列表的操作。然后考虑更换停用词库，起初尝试将百度停用词库、中文停用词库、哈工大停用词库与四川大学停用词库合并起来测试，但结果并不理想。对这些词库进行分别测试后，发现百度停用词库与四川大学停用词库表现较好，最终提取百度停用词库中的中文部分与四川大学停用词库合并作为实验词库，不更改其他参数的情况下 f1-score=0.9213（参考四川大学停用词库得分 f1-score=0.9200）。
尝试调整文本向量化的参数：ngram_range=(1,2)的时候 f1-score=0.9330，ngram_range=(1,3)的时候 f1-score 接近 0.94，但运行时间较长，或出现时间不足的风险。增加了 max_df=0.6，对 f1-score 几乎没有影响，这类词汇可能以及被停用词去除，增加 min_df=0.01 使得 f1-score 快速下降，遂不保留。
尝试使用 tfidf 向量化方法，在相同参数下 f1-score=0.8920。
使用 MaxAbsScaler 对其进行归一化，tfidf 测试组 f1-score 提高到 0.9387，countvector 测试组提高到 0.9508。
尝试使用 StandardScaler 进行归一化，两组结果 f1-score 均不足 0.9。
以下对所选方法：CountVectorizer，MaxAbsScaler 进行贝叶斯分类器的选择。
BernoulliNB 组测试 f1-score 不足 0.9，ComplementNB 组 f1-score=0.9004。
在进行其他测试后，最终确定模型：CountVectorizer，MaxAbsScaler，MultinomialNB。相关参数见代码内容。

三、代码内容

```
#读取停用词代码
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
stopwords_path = r'results/baidu_scu_stopwords.txt'
def read_stopwords(stopwords_path):
    stopwords = []
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()
    return stopwords
```

```
stopwords = read_stopwords(stopwords_path)
==========================================================================
#构建 pipeline 代码
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
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import StandardScaler
#最终选择 pipeline
pipeline_list =[
    ('cv',CountVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
    ('scaler', MaxAbsScaler()),
    ('classifier', MultinomialNB())
]
#以下为备选方案
pipeline_list01=[
    ('cv', CountVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
    ('scaler', MaxAbsScaler()),
    ('classifier', BernoulliNB())
]
pipeline_list02=[
    ('cv', CountVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
    ('scaler', MaxAbsScaler()),
    ('classifier', ComplementNB())
]
pipeline_list03=[
    ('cv', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
    ('scaler', MaxAbsScaler()),
    ('classifier', BernoulliNB())
]
pipeline_list1 = [
    ('cv', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
    ('scaler', MaxAbsScaler()),
    ('classifier', MultinomialNB())
```

```
]
pipeline_list2 = [
        ('cv', CountVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
        ('scaler', StandardScaler(with_mean=False)),
        ('classifier', MultinomialNB())
]
pipeline_list3 = [
        ('cv', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords,ngram_range=(1,2))),
        ('scaler', StandardScaler(with_mean=False)),
        ('classifier', MultinomialNB())
]
```

四、实验结果

五、总结

在本次实验中，通过对 Pandas、Numpy、Sklearn 等库的使用，进一步了解了机器学习的实现方式。通过对参数的逐步调整与优化，我意识到对于不同要求的问题和不同类型的数据，对于模型选择适合的方法和参数也是非常重要的，对最终结果影响很大。在本次实验中，我没有重新分配随机数种子和测试集分割比例进行验证，这可能成为后续的优化方向。然而，总体上我已经对不同方法和参数进行了充分选择，也得到了还算不错的结果。