程序报告

学号：2313721　　姓名：许洋

一、问题重述

1.实验背景

垃圾短信 (SpamMessages，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。

据 360 公司 2020 年第一季度有关手机安全的报告提到，360 手机卫士在第一季度共拦截各类垃圾短信约 34.4 亿条，平均每日拦截垃圾短信约 3784.7 万条。

大数据时代的到来使得大量个人信息数据得以沉淀和积累，但是庞大的数据量缺乏有效的整理规范；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

任务提供包括数据读取、基础模型、模型训练等基本代码

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

模型单次推理时间不超过 10 秒

3.实验环境

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

(1)模型构建与训练

数据预处理

首先，我们从sms_pub.csv文件中读取短信数据，并对正负样本进行平衡处理：

正样本（恶意短信）保持不变。

负样本（正常短信）随机抽样，使其数量与正样本一致。

```
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
stopwords_path = r'scu_stopwords.txt'
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)
```

停用词处理

停用词是指对分类无意义的高频词汇（如“我”、“你”）。我们通过读取停用词文件scu_stopwords.txt，生成停用词列表：

```
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
```

```
stopwords = read_stopwords(stopwords_path)
```

特征提取

我们使用TfidfVectorizer进行文本特征提取，具体参数如下：

stop_words=stopwords : 使用自定义的停用词列表。

ngram_range=(1, 2) : 提取单字和双字组合的特征。

token_pattern=r"(?u)\b\w+\b" : 包括单字符在内的更宽容分词规则。

max_df=0.95 : 去除文档中出现频率超过 95% 的高频词汇。

min_df=2 : 去除仅出现一次的稀有词汇，避免噪声。

模型选择

我们选择了互补朴素贝叶斯（Complement Naive Bayes, CNB）作为分类器，原因如下：

CNB 是朴素贝叶斯的一种改进版本，特别适用于不平衡数据集。

在多分类问题中表现良好，且计算效率高。

Pipeline构建

为了简化流程，我们将特征提取、归一化和分类器集成到一个 Pipeline 中：

```
# pipline_list用于传给Pipline作为参数
pipeline_list = [
    ('tfidf', TfidfVectorizer(
        stop_words=stopwords,
        ngram_range=(1, 2),
        token_pattern=r"(?u)\b\w+\b",  # 加入更宽容的分词规则（包括单字符）
        max_df=0.95,                   # 去除文档中95%以上出现的词（如“我”“你”）
        min_df=2                       # 去除只出现一次的稀有词，避免噪声
    )),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', ComplementNB(alpha=1.0))
]
```

数据划分与训练评估

将数据划分为训练集和测试集（8:2），并使用训练集拟合模型，

```
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=42, test_size=0.2)
# 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))
```

模型保存与加载

训练好的模型被保存为pipeline.model文件，便于后续加载和使用：

```
# 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)
```

(2)利用训练模型进行预测

首先我们读取停用词，然后导入我们测试出最好的模型

```
stopwords_path=r'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)
# ------- pipeline 保存的路径，若有变化请修改 --------
pipeline_path = 'results/pipeline.model'
# -----------------------------------------------
pipeline = joblib.load(pipeline_path)
```

负责加载训练好的模型并提供预测接口。

```
def predict(message):
    label = pipeline.predict([message])[0]
    proba = list(pipeline.predict_proba([message])[0])
    return label, proba
```

三、代码内容

(1)train.py

```
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import ComplementNB
from sklearn.preprocessing import MaxAbsScaler
data_path = "./datasets/5f9ae242cae5285cd734b91e-
momodel/sms_pub.csv"
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)
stopwords_path = r'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)
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
pipeline_list = [
    ('tfidf', TfidfVectorizer(
        stop_words=stopwords,
        ngram_range=(1, 2),
        token_pattern=r"(?u)\b\w+\b",  # 加入更宽容的分词规则（包括单字
符）
        max_df=0.95,                   # 去除文档中95%以上出现的词
（如“我”“你”）
        min_df=2                       # 去除只出现一次的稀有词，避免噪声
```

```
    )),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', ComplementNB(alpha=1.0))  # 显式声明默认值，更可调
]
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=42, test_size=0.2)
pipeline = Pipeline(pipeline_list)
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
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))
pipeline.fit(X, y)
from sklearn.externals import joblib
pipeline_path = 'results/pipeline.model'
joblib.dump(pipeline, pipeline_path)
```

(2)main.py

```
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import ComplementNB
from sklearn.preprocessing import MaxAbsScaler
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.externals import joblib
stopwords_path=r'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)
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
```

四、实验结果

- 输出结果如下：

```
在测试集上的混淆矩阵:
[[15275   518]
 [  221 15645]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98     15793
           1       0.97      0.99      0.98     15866

    accuracy                           0.98     31659
   macro avg       0.98      0.98      0.98     31659
weighted avg       0.98      0.98      0.98     31659

在测试集上的 f1-score :
0.9769271597614662
```

```
0 [0.9999943901180939, 5.609881911730399e-06]
```

- 平台检测结果：

测试点	状态	时长	结果
测试模型预测结果	✓	8s	通过测试，训练的分类器具备检测恶意短信的能力，分类正确比例:10/10
测试读取停用词库函数结果	✓	10s	read_stopwords 函数返回的类型正确

五、总结

通过机器学习技术构建了一个垃圾短信分类系统，能够高效区分正常短信和恶意短信。在数据预处理阶段，通过对正负样本的平衡处理和停用词过滤，有效减少了噪声对模型的影响。特征提取采用 TfidfVectorizer，结合单字和双字组合（n-gram），增强了模型对上下文信息的捕捉能力。模型选择方面，互补朴素贝叶斯（CNB）表现优异，尤其适用于不平衡数据集。最终模型在测试集上的准确率和 F1 分数均达到 98%，分类结果均衡且误判率低。

通过本次实验，我深刻体会到数据质量和特征工程对模型性能的重要性，同时也认识到不同算法在特定场景中的适用性。未来可以尝试引入深度学习方法（如 BERT）以进一步提升分类效果，并开发实时预测接口以满足实际应用需求。感谢指导老师和开源社区的支持，使我能够顺利完成实验并从中获益良多。