程序报告

学号：2213924　　姓名：申宗尚

一、问题重述

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。

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。

实验要求：

1. 使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

首先，我们进行数据分析。

在提供的数据集/datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv 中，提供了如下的数据格式以供训练：

	label	message	msg_new
1	0	和垄断地位的前提条件之一	垄断 地位 的 前提条件 之一
2	1	、冰丝女王长半裙、口 皇	丝 女王 长半裙 、口 皇
3	0	大常州一场壮观的视觉盛宴	州 一场 壮观 的 视觉 盛宴
4	0	原因不明的泌尿系统结石等	不明 的 泌尿系统 结石 等
5	0	认盐城拉回来的麻麻的嫁妆	城 拉回来 的 麻麻 的 嫁妆
6	0	到自减肥、跳减肥健美操、	减肥 、 跳 减肥 健美操 、
7	1	惠顾【全金釜韩国烧烤店】	【 全金 釜 韩国 烧烤店 】
8	0	扒器人是扫地机的最佳伴侣	人 是 扫地机 的 最佳 伴侣
9	1	！！ 预约电话：xxxxxxxxxx	预约 电话 ： xxxxxxxxxx
10	0	特别容易招惹粉刺、黑头等	容易 招惹 粉刺 、 黑头 等
11	0	滇市法院成立爱心救助基金	法院 成立 爱心 救助 基金
12	1	xxxxxxxxxxx 李伟%	x x x x x x 李伟 %
13	1	详情进店咨询。美丽热线x	进店 咨询 。 美丽 热线 x
14	0	品牌墙/文化墙设计参考	品牌 墙 / 文化 墙 设计 参考

共有 786610 行数据，其中第一列为 label 值，0 表示正常短信，1 表示垃圾短信，第二列为短信内容，第三列 msg_new 表示分词过后的短信内容，由于分类模型通过单词/短语出现的频率进行训练，后续的特征提取、向量化和分类操作前，均需要对文本进行分词化，因此在实验中，我们使用 label 和 msg_new 进行模型训练。

分词使用的库是中文分词库 jieba https://github.com/fxsjy/jieba

同时，我们导入 panda 库，使用 panda.read_csv 方法进行数据的读取。

```
import pandas as pd
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
sms = pd.read_csv(data_path, encoding='utf-8')
```

同时，我们为了数据学习的高效性和正确性，定义一些停用词集合。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）。

在本次实验中，直接使用提供的四川大学机器智能实验室停用词库。

接着，我们进行文本向量化。这次实验中主要尝试了两种方法：CountVectorizer 和 TfidfVectorizer，由于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。

其中，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。而 TfidfVectorizer 则是基于 TF-IDF 算法的，在词出现次数的基础上增加了“文本频率小的单词更重要”的思想，复杂性更高。

在本次实验中，CountVectorizer 在后续的实验效果中表现并不出色，稳定在 8/10 的正确率左右，因此，后面改用了 TfidfVectorizer 进行文本向量化

随后，我们划分训练集和测试集，一般的数据集会划分为两个部分：

训练数据：用于训练，构建模型

测试数据：在模型检验时使用，用于评估模型是否有效

他们的比例可以调整，会影响最终模型训练的效果，是一个优化模型的方式。

在实验中，我们采用的分类器进行了多次尝试，首先，我们尝试了朴素贝叶斯方法，其实现原理基于贝叶斯公式，给定一个样本，计算该样本下每个类别的条件概率：

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

并假设每个特征独立，因此该公式化为：

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

由于分母确定，结果只与分子有关：

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

求出最大的条件概率，其对应的类别就是该样本所属的类别。

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

朴素贝叶斯有两种实现方法，分别为 BernoulliNB 和 ComplementNB，我们分别尝试，评价如下：

from sklearn.naive_bayes import BernoulliNB

from sklearn.naive_bayes import ComplementNB

BernoulliNB:

```
总共的数据大小 (786610,)
训练集数据大小 (707949,)
测试集数据大小 (78661,)
在测试集上的混淆矩阵:
[[69600  1178]
 [  672  7211]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     70778
           1       0.86      0.91      0.89      7883

    accuracy                           0.98     78661
   macro avg       0.93      0.95      0.94     78661
weighted avg       0.98      0.98      0.98     78661

在测试集上的 f1-score:
0.886307767944936
```

ComplementNB:

```
[ 5260  2631]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     70770
           1       1.00      0.33      0.50      7891

    accuracy                           0.93     78661
   macro avg       0.96      0.67      0.73     78661
weighted avg       0.94      0.93      0.92     78661

在测试集上的 f1-score:
0.4999524940617577
```

最后分类器的效果仍不是特别理想，可以看到，ComplementNB 有过学习的嫌疑，BernoulliNB 性能较为让人满意，但由于仍想要找到更好的分类器算法，而且实验只是二分类，实验中尝试了 LogisticRegression 分类器：

逻辑回归(Logistic Regression)是一种常用的统计学习方法，用于解决二分类问题。尽管名字中带有"回归"，但实际上逻辑回归是一种分类算法，用于预测目标变量的可能取值为两个类别中的一个。逻辑回归的核心思想是将线性回归模型的输出通过一个逻辑函数（也称为 sigmoid 函数）映射到[0, 1]区间，以表示概率。

其学习效果如下：

```
在测试集上的混淆矩阵:
[[70150   466]
 [ 1261  6784]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     70616
           1       0.94      0.84      0.89      8045

    accuracy                           0.98     78661
   macro avg       0.96      0.92      0.94     78661
weighted avg       0.98      0.98      0.98     78661

在测试集上的 f1-score:
0.8870872834259561
预测的类别: 0 概率: [0.9999991839317574, 8.160682425986268e-07]
```

其中，召回率(Recall)：指在所有实际为正例的样本中，模型成功预测为正例的样本数量占比。换句话说，召回率衡量了模型对于正例样本的识别能力，即模型有多少能力找出所有的正例。计算公式：Recall = (TP) / (TP + FN)

F1 值(f1-score)：是综合考虑了模型的精确率 (Precision) 和 召回率 (Recall) 的指标。F1 值是精确率和召回率的调和平均数，它将两者都考虑在内，因此可以更全面地评估模型地性能。F1 值地计算公式为：F1 = 2 x (Precision x Recall) / (Precision + Recall)

可以发现，虽然 Logistic Regression 算法实现简单，但其特别适用于二分类问题，学习效果明显优于两种贝叶斯。

随后，我们通过构建 Pipeline 的方式，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，从而方便直接对原始数据进行预测。

三、代码介绍

在代码中，我们首先导入需要的头文件和库名：

```python
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MaxAbsScaler
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, f1_score
#from sklearn.externals
import joblib
```

随后，我们加载数据，并通过 pandas 的 read_csv 进行数据的读取：

```
# 设置环境变量
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"

# 加载数据
data_path = 'datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv'
data = pd.read_csv(data_path,encoding='utf-8')
data = data.sample(frac=1).reset_index(drop=True)
x = data['message']
y = data['label']



X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42,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)
```

停用词的读取：

```
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 = [line.strip() for line in f]
    # ------------------------------------------

    return stopwords
# 读取停用词
stopwords_path = './scu_stopwords.txt'
stopwords = read_stopwords(stopwords_path)
```

定义、配置 Pipeline，设置文本向量化工具、scalar 和分类器（参数"(?u)\b\w+\b"是为了防止单文本个例被删除，影响训练结果），同时进行模型的训练和测试：

```
# 定义和配置pipeline
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b", stop_words=stopwords)),
    ('scaler', MaxAbsScaler()),
    ('classifier',LogisticRegression(C=1000, penalty='l2'))
    #BernoulliNB(class_prior=[class_weights[0], class_weights[1]],alpha=1.0)
    ])
# 训练模型
pipeline.fit(X_train, y_train)
# 测试模型
y_pred = pipeline.predict(X_test)
```

进行模型的评估：

```
# 评估模型
print("在测试集上的混淆矩阵：")
print(confusion_matrix(y_test, y_pred))
print("在测试集上的分类结果报告：")
print(classification_report(y_test, y_pred))
print("在测试集上的 f1-score：")
print(f1_score(y_test, y_pred, average='binary'))
```

保存模型：

```
# 保存模型，确保目标目录存在
pipeline_path = 'results/pipeline.model'
os.makedirs(os.path.dirname(pipeline_path), exist_ok=True)
joblib.dump(pipeline, pipeline_path)
```

最终，预测函数：

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

四、代码内容

import os

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import MaxAbsScaler
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, f1_score
#from sklearn.externals
import joblib

# 设置环境变量
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"

# 加载数据
data_path = 'datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv'
data = pd.read_csv(data_path,encoding='utf-8')
data = data.sample(frac=1).reset_index(drop=True)
x = data['message']
y = data['label']



X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=42,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)


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 = [line.strip() for line in f]
    # -----------------------------------------------

    return stopwords
# 读取停用词
```

```
stopwords_path = './scu_stopwords.txt'
stopwords = read_stopwords(stopwords_path)
#class_weights = {0: 0.1, 1: 1200}
# 定义和配置 pipeline
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b", stop_words=stopwords)),
    ('scaler', MaxAbsScaler()),
    ('classifier',LogisticRegression(C=1000, penalty='l2'))
    #BernoulliNB(class_prior=[class_weights[0], class_weights[1]],alpha=1.0)
    ])
# 训练模型
pipeline.fit(X_train, y_train)
# 测试模型
y_pred = pipeline.predict(X_test)

# 评估模型
print("在测试集上的混淆矩阵：")
print(confusion_matrix(y_test, y_pred))
print("在测试集上的分类结果报告：")
print(classification_report(y_test, y_pred))
print("在测试集上的 f1-score：")
print(f1_score(y_test, y_pred, average='binary'))

# 保存模型，确保目标目录存在
pipeline_path = 'results/pipeline.model'
os.makedirs(os.path.dirname(pipeline_path), exist_ok=True)
joblib.dump(pipeline, pipeline_path)

# 预测函数
def predict(message):
    label = pipeline.predict([message])[0]
    proba = list(pipeline.predict_proba([message])[0])
    return label, proba

# 测试预测
label, proba = predict('医生 拿 着 我 的 报告单 说 ： 幸亏 你 来 的 早 啊')
print("预测的类别：", label, "概率：", proba)
```

五、实验结果

正确率 10/10，平均测试时间 1.2s，完成任务要求

六、总结

本次解答基本达到了目标预期，通过使用 Pipeline 的构建，使用 Logistic Regression 分类器，进行数据集、测试集的划分、使用停用词、文本向量化等工具，实现了一个中文文本二分类的分类器，达到了垃圾短信检测的目标。本次实验的收获如下：

1、对于模型的选择：分类器有多种算法，贝叶斯就有 3 种(BernoulliNB, ComplementNB, MultinomiaNB)，而最终选择了二分类最有效的 Logistic Regression 分类器，期间经历了漫长的学习、比对过程，增长了自己对于 sklearn 库结构的理解

2、参数的选择： 参数（测试集、训练集的比例，停用词…）的存在会使模型的最终训练效果不同，实验中为了达到最好的测试效果，对于 Recall 值、f1-score 等评判标准进行了系统的学习，同时进行了多次调整，多次尝试，最终平稳到能够令人满意的结果。

3、文本向量化的学习： 在先前的学习中，只接触过词袋模型，TfidfVectorizer 的学习，使我掌握了 TF-IDF 特征提取方法，将文本数据转换为数值特征向量。

4、Pipeline 的运用：本次实验我理解了机器学习中 Pipeline 的概念，以及如何将多个处理步骤组合成一个整体

总的来说，这次实验提高了我对于**机器学习、分类器、sklearn 库、python 程序语言编写**的熟练度。