垃圾短信分类问题实验报告

学号：2312900　姓名：禹相祐　专业：计算机科学与技术

一. 问题重述

1.1 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。

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。

1.2 实验要求

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

1.3 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. 设置并导入停用词 & 处理数据

```
# 数据和停用词文件路径
data_path="./datasets/5f9ae242cae5285cd734b91emomodel/sms_pub.csv"
stopwords_path = r'scu_stopwords.txt'

# 读取数据
sms = pd.read_csv(data_path, encoding='utf-8')
# 筛选正样本
sms_pos = sms[(sms['label'] == 1)]
# 筛选负样本并采样使其数量与正样本相同
sms_neg = sms[(sms['label'] == 0)].sample(frac=1.0)[:len(sms_pos)]
```

```
# 合并正样本和负样本并打乱顺序
sms = pd.concat([sms_pos, sms_neg], axis=0).sample(frac=1.0)

# 读取停用词函数
def read_stopwords(stopwords_path):
    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. 构建模型并利用训练集训练模型

此处采用 tfidf + MaxAbsScaler + LogisticRegression:

我们采用 tfidf 对文本信息转换成数值向量（处理文本），用 MaxAbsScaler 对向量进行缩放最后采用 LogisticRegression 对向量进行分类。

```
pipeline = Pipeline([
    ('tfidf',TfidfVectorizer(stop_words=stopwords,ngram_range=
(1, 2))),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', LogisticRegression(max_iter=1000))
])
```

3. 利用模型预测测试集并打印结果

```
# 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))
```

三． 代码内容

最后此次实验的完整代码：

```
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.linear_model import LogisticRegression
from sklearn.preprocessing import MaxAbsScaler

# 数据和停用词文件路径
data_path="./datasets/5f9ae242cae5285cd734b91emomodel/sms_pub.
csv"
stopwords_path = r'scu_stopwords.txt'

# 读取数据
sms = pd.read_csv(data_path, encoding='utf-8')
# 筛选正样本
sms_pos = sms[(sms['label'] == 1)]
# 筛选负样本并采样使其数量与正样本相同
sms_neg=sms[(sms['label']== 0)].sample(frac=1.0)[: len(sms_pos)]
# 合并正样本和负样本并打乱顺序
sms = pd.concat([sms_pos, sms_neg], axis=0).sample(frac=1.0)
```

```
# 读取停用词函数
def read_stopwords(stopwords_path):
    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)

# 提取特征和标签
X = np.array(sms.msg_new)
y = np.array(sms.label)

# 划分训练集和测试集
X_train, X_test, y_train, y_test=train_test_split(
X,y, random_state=22, test_size=0.2)

# 输出数据集数量信息
print("the number of all the datas", X.shape)
print("the number of the train datas", X_train.shape)
print("the number of the test datas", X_test.shape)

pipeline = Pipeline([
    ('tfidf',TfidfVectorizer(stop_words=stopwords,ngram_range=
(1, 2))),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', LogisticRegression(max_iter=1000))
])

# 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))
# 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)
# 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):
    """
    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
# 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

打印 accuracy & f1-score 等属性，我们可以看出 tfidf + MaxAbsScaler + LogisticRegression 的组合效果十分不错：

```
在测试集上的混淆矩阵:
[[15568   337]
 [  261 15493]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     15905
           1       0.98      0.98      0.98     15754

    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 :
0.981066362715299
```

最后在 mo 上测试的结果如图：只实现了 7/10，必须得承认效果没有在测试集上表现得那么好，可能模型有一点过学习了。

五. 总结

这次的实验相比于上次的黑白棋难度还是低不少的,而且自己能发挥的地方也特别的多。无论是像 tfidf 对于文本的处理方式，还是 Logistic Regression 这样分类器的选择都是多种多样的，加在一起组合成模型就更加多样了。可能因为我上学期选修过陈晨老师的 Python 课，期末大作业刚好也是新闻文本的分类，与此次实验强相关，所以对我来说不会特别的难。但能自己亲自搭建一个模型也十分有成就感。