程序报告

学号：2212046　　姓名：王昱

一、问题重述

- 垃圾短信检测问题：

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) 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

==========

- 对问题的理解：

本次实验要求我们应用机器学习相关内容，使用 Python 的 Pandas、Numpy、Sklearn 等库进行相关特征处理之后，使用 Sklearn 框架训练分类器，通过调参和对模型的选择与优化，得出一个最优的模型，可以正确识别垃圾短信。

二、设计思想

2.1 设计步骤

2.1.1 数据读取：使用 python 的 pandas 包中的 read_csv 进行数据的读取。

2.1.2 数据集分析：为了确定使用的模型以及如何对使用的模型进行调参，所以要进行对数据集的分析，包括正负样本的数量，有无空值等信息，这些都会影响到模型的性能。

2.1.3 数据处理：

- 清洗数据，去除无关信息和特殊字符。
- 读取停用词库，对文本内容进行分词操作，将停用词表读取到列表中进行保存。
- 特征提取：将文本数据转换为可以用于机器学习模型的数值特征，常用方法包括词袋模型和 TF-IDF：

(1) CountVectorizer：统计每个词出现的次数，将一句话中的词频整合作为向量，用于后面的训练。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。

(2) TF-IDF:F-IDF 是一种统计方法，用以评估一个词语对于一个文档集或一个语料库中的其中一份文档的重要程度。它是通过将词频（TF）与逆文档频率（IDF）相乘得到的值。公式：TF-IDF(t,d)=TF(t,d)×IDF(t)。目的在于突出重要单词，抑制次要单词。在本质上 IDF 是一种试图抑制噪声的加权，并且单纯地认为文本频率小的单词就越重要，文本

频率大的单词就越无用.

- 数据标准化：在数据处理中，标准化是一个非常重要的步骤，尤其是在机器学习和数据挖掘领域。标准化的目的是将数据调整到统一的尺度上，这样可以避免因为特征的量纲不同而导致的模型偏差。

MaxAbsScaler() 标准化：它来自于 Python 的 scikit-learn 库。这个方法的主要目的是将每个特征的数据按其最大绝对值进行缩放，使得每个特征的最大绝对值在训练集中为 1.0。MaxAbsScaler 会对每个特征独立地进行缩放，而不会移动/中心化数据，因此不会破坏数据的稀疏性。这个方法特别适合处理稀疏数据，例如文本数据的词频矩阵。这种标准化适用于稀疏矩阵，故本次实验使用该方法进行数据的标准化。

2.1.4 模型选择：

- svm/神经网络：svm 是天生的结构风险最小化分类，相对很稳定并且过拟合风险较小（别给太膨胀的核函数参数就行）。在样本数量少的时候，很多模型（尤其是深度神经网络）容易过拟合，svm 在这方面就挺不错。但当样本数量达到一定量级之后，svm 的计算复杂度会显著上升，神经网络的泛化能力也得以提升。加之神经网络结构设计灵活，这时 svm 就不再是最优的选择。
- 决策树系列：基于信息论的分类器，十分适合离散特征的处理。对于离散特征，其它很多模型都需要对其进行编码，得到很稀疏的编码向量再进行模型拟合，因较大的过拟合风险对样本量有较高要求。决策树则统计离散特征后直接计算信息增益等，再加上有各类集成树模型的存在（比如快速稳定可并行的 xgb）可以进一步提升性能，所以在很多离散特征满天飞的算法比赛里树模型独占鳌头。
- 贝叶斯分类器：

1. GaussianNB 先验为高斯分布的朴素贝叶斯，适合于特征分布大部分是连续值的样本。

2. MultinomialNB 先验为多项式分布的朴素贝叶斯，适合于特征的分大部分是多元离散值的样本。

3. BernoulliNB 先验为伯努利分布的朴素贝叶斯，适合于特征是二元离散值或稀疏的多元离散值的样本。

本次实验选择的分类器：朴素贝叶斯是一种基于贝叶斯公式的监督学习算法，并假设每个特征是独立的，该方法在“垃圾邮件分类”、“恶意邮件检测”等领域有着广泛应用。因此在本次垃圾短信分类实验采用朴素贝叶斯。

分析本实验所用的样本集，是向量化之后的文本特征值，属于多元离散的样本分布，MultinomialNB 的性能通常会高于 BernoulliNB，特别是在包含很多非零特性的数据集（即大型文档）上。因此判断 MultinomialNB 会更适合，经实验验证，MultinomialNB 分类效果确实优于其他两种。

2.1.5 模型搭建与训练：

- 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。本实验最终选择使用：TfidfVectorizer + 停用词表+MaxAbsScaler()，以及 MultinomialNB()分类器 搭建模型。因此在 pipline 里传入这三个参数。
- 对训练集和测试集按照一定比例进行划分，并且将训练好的模型进行保存。

2.1.6 模型效果评估：

- 二类分类问题常用的评价指标是准确率(accuracy)精准度（precision）、召回率（recall）、F1-Score，AUC。
- 在测试集上分析其混淆矩阵及评测指标：

Accuracy： (TP + TN) / (TP + FP + FN + TN) 表示模型预测正确的比例

Precision：TP / (TP + FP) 表示模型正确预测正类的频率

Recall： TP / (TP + FN) 表示在所有实际正类的标签中，模型正确识别了几个

F1 = 2 / (1 / Precision + 1 / Recall) 精确率与召回率的调和平均值

通过这些值判断模型训练的结果较好，为了防止过拟合，再在所有的数据训练一次，这样可以充分利用已有的数据，提高模型的泛化能力。

==========

2.2 实现流程

2.2.1 导入相关的库

- 首先导入实验要用到的库，为后面应用做准备

```
import os
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.externals import joblib
from sklearn.preprocessing import MaxAbsScaler
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
```

2.2.2 数据读取

- 利用 pandas 库读取指定路径的数据

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

- 导入停用词库并读取

```
stopwords_path = r'scu_stopwords.txt'
def read_stopwords(stopwords_path):
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()
    return stopwords
stopwords = read_stopwords(stopwords_path)
```

2.2.3 数据分析

- 为了文本的数据更加可视化，我使用了 matplotlib 等工具进行了数据集的分析。

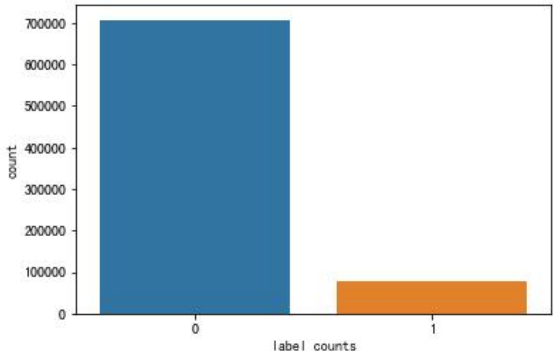
```
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
sms = pd.read_csv(data_path, encoding='utf-8')
#显示数据集信息
sms.info()
# 显示前 5 条数据
sms.head()
#显示正负样本的比例
sms['label'].value_counts()
```

sns.countplot(sms.label)

plt.xlabel('label counts')

运行结果：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786610 entries, 0 to 786609
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype 
---  ------   --------------   ----- 
 0   label    786610 non-null  int64 
 1   message  786610 non-null  object
 2   msg_new  786610 non-null  object
dtypes: int64(1), object(2)
memory usage: 18.0+ MB
Text(0.5, 0, 'label counts')
```



如图所示，依次是文本的信息以及正负样本的比例，可以看到，正负样本的比例很不均衡由于训练数据中正负样本不均衡（正负样本比例约为 1:10），将会导致拟合效果较差，因此读入后在负样本中随机取出一定数量作为实验用样本，使正负样本数量相同，这样可以使正负样例的数目相同，这样的话模型训练结果会好，代码如下：

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)

2.2.4 模型参数选择

- MultinomialNB()的参数：

① alpha：浮点型可选参数，默认为 1.0，如果这个参数设置为 0，就是不添加平滑；

② fit_prior：布尔型可选参数，默认为 True。布尔参数 fit_prior 表示是否要考虑先验概率，如果是 false，则所有的样本类别输出都有相同的类别先验概率。

③ class_prior：可选参数，默认为 None。

④ 经过实验验证得，调整参数之后，发现有平滑的比没有平滑的效果更好，且 alpha=1 时，效果最好。

⑤ 本实验没有统一的类别先验概率，所以 fit_prior、class_prior 使用默认参数。

- TfidfVectorizer(min_df=3,ngram_range(1,2))

① ngram_range 是词组切分的长度范围，经过验证这个效果最好。

② min_df 也是经过多次测试得到的，这里取 3。

2.2.5 模型搭建与训练集测试及划分

from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()

X = np.array(sms.msg_new)

y = np.array(sms.label)

pipeline= Pipeline([

```
    ('tfidf',TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",stop_words=stopwords)),
      ('MaxAbsScaler', MaxAbsScaler()),
      ('clf', MultinomialNB())
      ])
      X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)
      pipeline.fit(X_train, y_train)
```

2.2.6 模型性能评估

通过上文提到的几个指标对模型训练的结果进行评估。

```
      joblib.dump(pipeline, 'results/pipeline.model')
      y_pred = pipeline.predict(X_test)
      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))
      print('在测试集上的准确率：')
      print(metrics.accuracy_score(y_test, y_pred))
```

训练结果如下：

```
在测试集上的混淆矩阵：
[[7462  426]
 [  73 7869]]
在测试集上的分类结果报告：
              precision    recall  f1-score   support

           0       0.99      0.95      0.97      7888
           1       0.95      0.99      0.97      7942

    accuracy                           0.97     15830
   macro avg       0.97      0.97      0.97     15830
weighted avg       0.97      0.97      0.97     15830

在测试集上的 f1-score ：
0.9692677218698035
在测试集上的准确率：
0.9684775742261529
```

可以看到，模型的训练效果很好，达到了很不错的效果。

2.2.7 模型预测

加载训练好的模型

```
      from sklearn.externals import joblib
      pipeline_path = 'results/pipeline.model'
      pipeline = joblib.load(pipeline_path)
```

预测

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

三、代码内容

完整的代码内容如下：

main:

```
# 导入相关的包
```

```
import os
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.externals import joblib
from sklearn.preprocessing import MaxAbsScaler
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"

# ----------  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
    """
    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)

# 数据集的路径
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
# 读取数据
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)

# 构建训练集和测试集
X = np.array(sms.msg_new)
y = np.array(sms.label)
```

```
pipeline= Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 2),token_pattern=r"(?u)\b\w+\b",
stop_words=stopwords)),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('clf', MultinomialNB())
])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)
pipeline.fit(X_train, y_train)

# 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
```

模型训练：

```
import os
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.externals import joblib
from sklearn.preprocessing import MaxAbsScaler
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
```

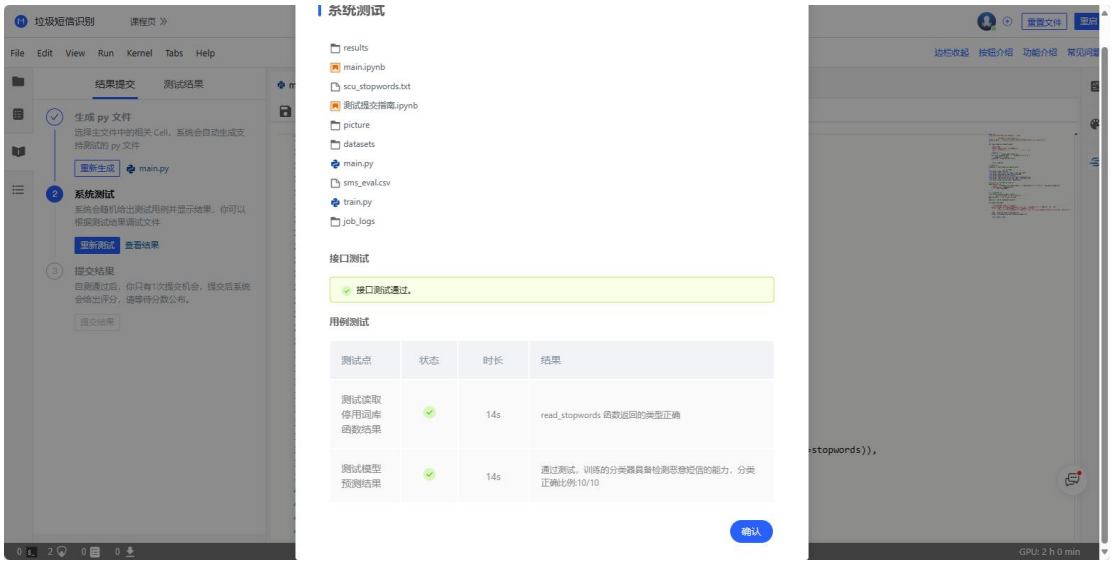
```
# ----------  停用词库路径，若有变化请修改  ----------
stopwords_path = r'scu_stopwords.txt'
# -----------------------------------------------------

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
    """
    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)# 数据集的路径
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
# 读取数据
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)

# 构建训练集和测试集
X = np.array(sms.msg_new)
y = np.array(sms.label)

pipeline= Pipeline([
    ('tfidf', TfidfVectorizer(token_pattern=r"(?u)\b\w+\b", stop_words=stopwords)),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('clf', MultinomialNB())
])

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)
pipeline.fit(X_train, y_train)joblib.dump(pipeline, 'results/pipeline.model')

y_pred = pipeline.predict(X_test)
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))
print('在测试集上的准确率：')
print(metrics.accuracy_score(y_test, y_pred))
```

四、实验结果

平台检测结果：

测试两次，一次结果为 10/10，另一次结果为 9/10，说明模型分类垃圾短信的能力较好。



五、总结

1.模型训练初期，效果并不是很好，只有 8/10 的正确率，后面通过对数据进行分析，发现正负样本比例十分不均衡，因此我使用了一种方法来平衡正负样本的个数：

```
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)
```

随机取出了和正样例相等个数的负样例，从而平衡了样本数据的类别。随后把样本数据随机打乱来作为训练数据。同时，我还尝试了其他方法，比如对正负样例加权，进行过采样和欠采样等等，综合下来采用了模型效果最好的一种方法。

==========

2.模型参数的选择：

使用 K 折交叉验证评估模型性能

在 k 折交叉验证中，我们不重复地随机将训练数据集划分为 k 个，其中 k-1 个用于模型的训练，剩余的 1 个用于测试。重复此过程 k 次，我们就得到了 k 个模型及对模型性能的评价。我们可以用 AUC 为指标，这样能很好的反应模型的效果，取出 AUC 结果最好的一组参数。

==========

3.实验过程中遇到的困难主要是特征提取方法的选择，TF-IDF 方法虽然简单，但可能无法充分表达短信中的语义信息。为了提升性能，可以从以下方面进行优化：

1、采用更复杂的特征提取方法，如 word2vec 等。

2、尝试不同的分类算法，如支持向量机、决策树等。

3、调整模型的超参数，如朴素贝叶斯分类器的平滑参数，TF-IDF 的 ngram 参数等。