# 程序报告

学号：2310422　　姓名：谢小珂

## 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

**本次实验围绕垃圾短信识别任务展开，核心目标是通过机器学习方法构建一个高效、准确的分类模型，能够自动区分正常短信（label=0）和恶意/垃圾短信（label=1）。以下是实验的设计思想及关键步骤：**

**（1）数据预处理**

○数据加载：

•使用Pandas读取CSV格式数据集，字段包括原始文本（message）、分词后文本（msg_new）和标签（label）。

○停用词过滤：

•加载四川大学停用词库（scu_stopwords.txt），去除无意义的常见词（如“的”“了”），降低特征维度

• 通过 splitlines()将停用词文本转换为列表，供后续向量化工具调用。

○数据集划分：

• 按 9:1 比例切分训练集和测试集，确保模型评估的可靠性。

**（2）文本向量化**

○TF-IDF 向量化（TfidfVectorizer）：

• 优势：结合词频（TF）和逆文档频率（IDF），突出重要词汇，抑制常见词。
• 参数调优：max_df=0.9（忽略高频词）、min_df=10（忽略低频词），平衡特征空间大小。

○备选方案：CountVectorizer 词袋模型，通过 ngram_range 可捕捉词组特征。

**（3）模型选择与训练**

○算法选择：朴素贝叶斯分类器（ComplementNB）：

• 适用性：适合高维稀疏文本数据，计算效率高。
• 参数设置：alpha=0.1（平滑参数防止零概率问题）、norm=True（归一化提升数值稳定性）。

○Pipeline 构建：

• 将 TfidfVectorizer 和 ComplementNB 封装为流水线，简化训练与预测流程，避免数据泄露。
• 优势：一键完成从原始文本到分类结果的端到端处理。

**（4）模型评估与优化**

○评估指标：

• 混淆矩阵：直观展示分类结果（TP、TN、FP、FN）。
• 分类报告：输出精确率（Precision）、召回率（Recall）、F1-score（平衡指标）。

○优化策略：

• 特征层面：尝试不同的停用词库或扩充领域相关停用词。
• 模型层面：调节 TfidfVectorizer 的 ngram_range 或切换为 CountVectorizer。
• 超参数调优：如调整朴素贝叶斯的 alpha 值或尝试其他分类器（如 MultinomialNB）。

**（5）模型部署与应用**

○全数据训练：

• 最终使用全部数据（训练集+测试集）重新训练模型，提升泛化能力。

○模型保存：

• 通过 joblib 将流水线（pipeline）保存为 pipeline.model，便于后续加载预测。

○预测接口：

• 输入：分词后的短信文本（如“医生 拿 着 报告单”）。
• 输出：预测标签（0/1）及类别概率（如[0.2, 0.8]）。

## 三、代码内容

本次实验采用双脚本模块化设计，通过 train.py 和 main.py 分离模型训练与预测流程，确保代码可维护性和复用性。

1.train.py

该脚本用于训练垃圾短信分类模型，主要功能包括：加载数据集（含分词后的短信文本和标签）、过滤停用词、使用 TF-IDF 将文本向量化、训练朴素贝叶斯分类器，并评估模型性能（输出混淆矩阵和 F1 分数），最终将训练好的模型保存为 pipeline.model 文件供后续预测使用。

```python
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"

# ---------- 停用词库路径配置 -------------
stopwords_path = r'scu_stopwords.txt'  # 停用词库文件路径
# ----------------------------------------

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 = []
    # 读取停用词文件内容并按行分割成列表
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()  # 将文本按行分割成列表
    return stopwords

# 读取停用词
stopwords = read_stopwords(stopwords_path)

# ----------------- 导入所需的库 -----------------
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, ComplementNB
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pandas as pd
import numpy as np
import joblib

# ----------------- 数据加载与预处理 -----------------
# 读取短信数据集
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
sms = pd.read_csv(data_path, encoding='utf-8')

# 将数据转换为 numpy 数组格式
```

```
X = np.array(sms.msg_new)  # 短信文本特征
y = np.array(sms.label)    # 短信标签（0=正常，1=垃圾）

# 划分训练集和测试集（测试集占 10%）
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42,
test_size=0.1)

# 打印数据集大小信息
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)

# ----------------- 模型构建 -----------------
# 定义 Pipeline 流程：TF-IDF 向量化 + ComplementNB 分类器
pipeline_list = [
    # TF-IDF 文本向量化
    ('tf', TfidfVectorizer(
        stop_words=stopwords,  # 使用停用词表
        max_df=0.9,           # 忽略在 90%以上文档中出现的词
        min_df=10             # 忽略在少于 10 个文档中出现的词
    )),
    # ComplementNB 分类器（适用于不平衡数据）
    ('classifier', ComplementNB(
        alpha=0.1,  # 平滑参数
        norm=True   # 启用归一化
    ))
]

# 构建 Pipeline
pipeline = Pipeline(pipeline_list)

# ----------------- 模型训练与评估 -----------------
# 在训练集上训练模型
pipeline.fit(X_train, y_train)

# 在测试集上进行预测
y_pred = pipeline.predict(X_test)

# 评估模型性能
print("在测试集上的混淆矩阵：")
print(metrics.confusion_matrix(y_test, y_pred))  # 输出混淆矩阵

print("在测试集上的分类结果报告：")
print(metrics.classification_report(y_test, y_pred))  # 输出分类报告
```

```python

print("在测试集上的 f1-score : ")
print(metrics.f1_score(y_test, y_pred))  # 输出 F1 分数

# ----------------- 最终模型训练与保存 -----------------
# 使用全部数据重新训练模型，提高泛化能力
pipeline.fit(X, y)

# 保存训练好的模型到 results 目录
pipeline_path = 'results/pipeline.model'
joblib.dump(pipeline, pipeline_path)  # 使用 joblib 保存模型
print("模型保存成功")
```

2. main. py

该脚本用于实时预测短信类别，通过加载训练好的模型文件，输入分词后的短信文本即可返回分类结果（0/1）及概率分布。

```python
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"

# ---------- 停用词库路径配置 -------------
# 注意：如需更改停用词库路径，请修改此处变量
stopwords_path = r'scu_stopwords.txt'  # 默认使用当前目录下的四川大学停用词库
# ------------------------------------------------

def read_stopwords(stopwords_path):
    """
    读取停用词库文件并转换为列表
    :param stopwords_path: 停用词库文件路径
    :return: 包含所有停用词的列表，例如：['的', '了', '在', '是', '我']
    """
    stopwords = []
    # 使用 utf-8 编码读取停用词文件
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()  # 读取整个文件内容
    stopwords = stopwords.splitlines()  # 按行分割为列表
    return stopwords

# 加载停用词库
stopwords = read_stopwords(stopwords_path)

# 加载训练好的模型
import joblib
# 模型路径配置（如需更改模型位置请修改此处）
pipeline_path = 'results/pipeline.model'  # 默认从 results 目录加载模型
# ------------------------------------------------
```

```python
# 加载预训练的分类 pipeline
pipeline = joblib.load(pipeline_path)

def predict(message):
    """
    对输入的短信文本进行分类预测
    :param message: 经过 jieba 分词的短信文本字符串，词语间用空格分隔
                    示例："医生 拿 着 报告单 说 幸亏 来 的 早"
    :return:
        label: 整型预测结果，0 表示正常短信，1 表示垃圾短信
        proba: 概率列表，格式为[正常短信概率，垃圾短信概率]
               示例：[0.85, 0.15]表示 85%概率是正常短信
    """
    # 使用 pipeline 进行预测（注意输入需要是列表形式）
    label = pipeline.predict([message])[0]  # 获取预测类别
    proba = list(pipeline.predict_proba([message])[0])  # 获取概率分布并转换为列表

    return label, proba
```

## 四、实验结果

测试结果如下图所示：

## 五、总结

1.通过这次垃圾短信分类实验，我收获了很多课堂上没有的实战经验，这次真正自己动手实现了从数据清洗、特征提取到模型训练的全过程。特别是处理中文文本时，学会了用 jieba 分词和停用词过滤这些实用技巧。

2.在调参时才发现，原来 max_df、min_df 这些参数对结果影响这么大。通过反复尝试，终于明白了 TF-IDF 中"忽略高频词和低频词"的实际作用，这种理解比单纯看公式直观多了。

3.课本上讲的朴素贝叶斯算法看起来很简单，但实际应用到短信分类时，要考虑分词、停用词、特征选择这么多细节，让我对机器学习有了更立体的认识。

4.优化方向

特征工程：

- 尝试 ngram_range=(1,2)捕捉词组特征（如“免费领取”）。
- 引入自定义停用词（如特定广告词汇）。

模型升级：

- 测试深度学习模型（如 TextCNN、BERT）以提升长文本分类效果。
- 集成多个分类器（如随机森林+SVM）进行投票决策。

部署优化：

- 将模型封装为 API 服务，支持批量预测。
- 添加日志监控，跟踪模型在线表现。