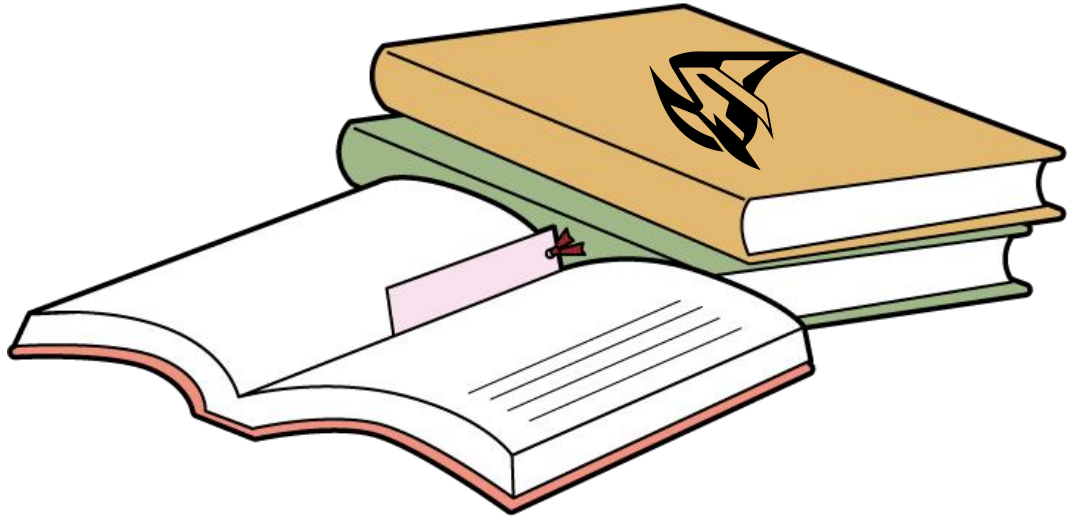
垃圾短信识别
--基于朴素贝叶斯分类



汇报人：叶飘飘

目录

CONTENTS

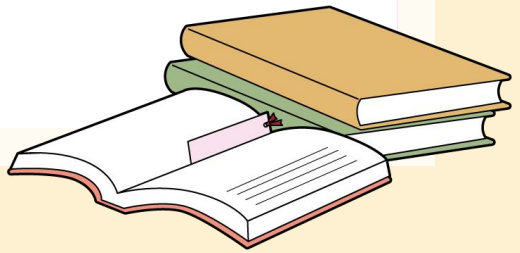
研究背景

第一部分

背景

日常生活中，人们深受垃圾短信之扰。垃圾短信的识别，成为各移动运营商不得不面对和考虑的问题。垃圾短信影响了用户的生活，损害了用户的利益，部分违法垃圾短信还影响了社会的稳定和国家的安全。因此对垃圾短信进行识别和过滤成为目前各运营商面临的重要任务之一。

垃圾短信识别是一个典型的二元分类问题。对任何一条短信，可以根据它的文本内容将其分类为垃圾短信和非垃圾短信。本案例主要介绍针对一个文本数据集，基于朴素贝叶斯分类法，如何进行垃圾短信的识别。



贝叶斯分类法介绍

第二部分

引言

贝叶斯分类：是一种基于贝叶斯定理的概率分类方法，用来预测一个样本属于某个特定类的概率，从而进行分类。

主要有朴素贝叶斯分类法和贝叶斯信念网络法两种。

朴素贝叶斯分类法在分类的计算过程中做了假设，即假定属性之间是相互独立的。做此假设的目的是为了简化计算。

但实际上，在某些情况下属性间是不独立的，这时就要采用贝叶斯信念网络分类法，该方法采用图形模型来表示属性之间的依赖关系。

在这里主要对朴素贝叶斯分类法进行介绍。

1.垃圾邮件过滤

- 应用场景：电子邮件系统的垃圾邮件过滤。
- 工作原理：利用朴素贝叶斯分类器，依据邮件中“免费”“优惠”等关键词，计算邮件为垃圾或非垃圾的概率。

2.文本分类

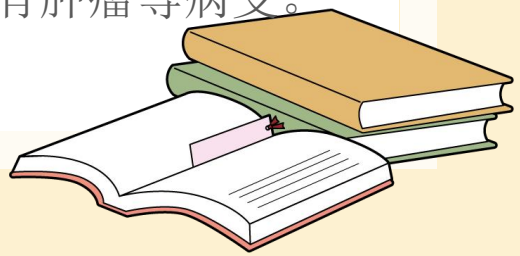
- 应用场景：新闻分类、情感分析、文档分类等。
- 工作原理：新闻分类时，贝叶斯分类器根据“体育”“科技”等关键词自动分类新闻；情感分析通过分析情感词汇判断文本情感倾向。

3.推荐系统

- 应用场景：电商、音乐平台的个性化推荐。
- 工作原理：根据用户购买记录、评分等历史行为，贝叶斯分类器预测用户对商品的兴趣度，推荐商品或音乐。

4.图像分类

- 应用场景：图像识别
- 工作原理：贝叶斯分类器依据图像颜色、纹理、形状等特征分类图像。如医学图像分析中，根据X光片或MRI图像特征判断是否有肿瘤等病变。



贝叶斯定理

- **贝叶斯定理公式**:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}$$

  - $P(C_k|X)$: 后验概率，表示在给定特征 X 的情况下，样本属于类别 C_k 的概率。
  - $P(X|C_k)$: 似然概率，表示在类别 C_k 下，特征 X 出现的概率。
  - $P(C_k)$: 先验概率，表示类别 C_k 在训练集中的出现概率。
  - $P(X)$: 证据因子，表示特征 X 在所有类别中的出现概率。

朴素贝叶斯分类法

- **朴素贝叶斯的假设**: 特征之间相互独立，即:

$$P(X|C_k) = P(x_1|C_k) \times P(x_2|C_k) \times \cdots \times P(x_n|C_k)$$

- **分类过程**:

1. 计算每个类别的先验概率 $P(C_k)$。
2. 计算每个特征在给定类别下的条件概率 $P(x_i|C_k)$。
3. 根据贝叶斯定理计算后验概率 $P(C_k|X)$。
4. 选择后验概率最大的类别作为分类结果。

连续型特征变量

对于连续型特征，通常假设其服从高斯分布（正态分布），即：

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{C_k}^2}} e^{-\frac{(x_i-\mu_{C_k})^2}{2\sigma_{C_k}^2}}$$

计算步骤：

1. 计算每个类别 C_k 下特征 x_i 的均值 μ_{C_k} 和标准差 σ_{C_k}。
2. 使用高斯概率密度函数计算 $P(x_i|C_k)$。

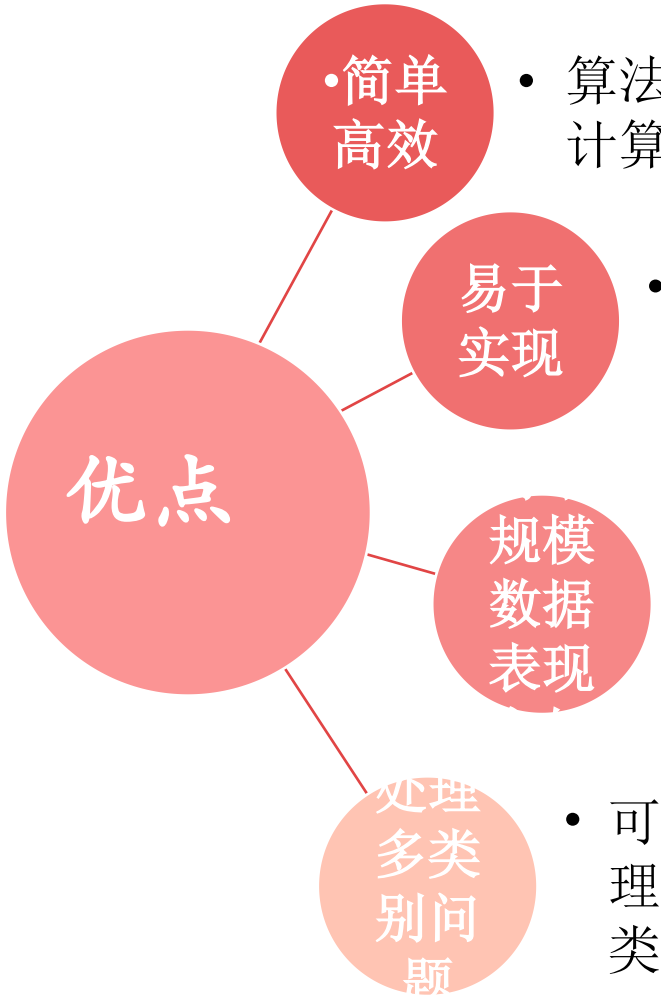
如果特征 x_i 的取值为 a，则：

$$P(x_i = a|C_k) = \frac{1}{\sqrt{2\pi\sigma_{C_k}^2}} e^{-\frac{(a-\mu_{C_k})^2}{2\sigma_{C_k}^2}}$$

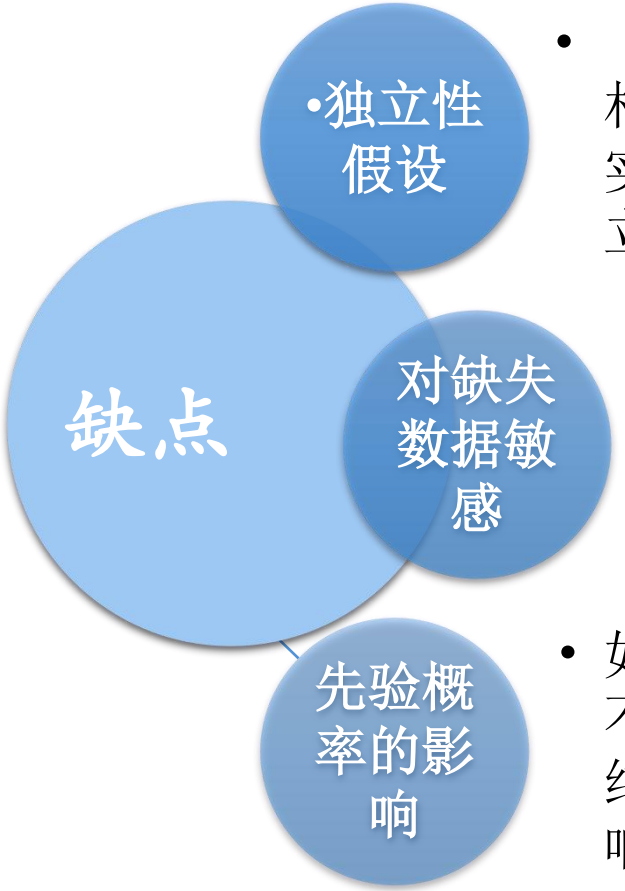
如果特征向量 $X = \{x_1, x_2, \ldots, x_n\}$ 包含多个连续型特征，假设特征之间相互独立（朴素贝叶斯假设），则联合条件概率为：

$$P(X|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

优缺点



- 算法简单，计算速度快。
- 适用于大规模数据集。
- 即使在数据量较少的情况下也能表现良好。
- 可以轻松处理多类别分类问题。



- 假设特征之间相互独立，现实中往往不成立。
- 如果某个特征在训练集中没有出现，会导致概率为0，影响分类结果。
- 如果先验概率不准确，分类结果会受到影响。

数据来源与准备

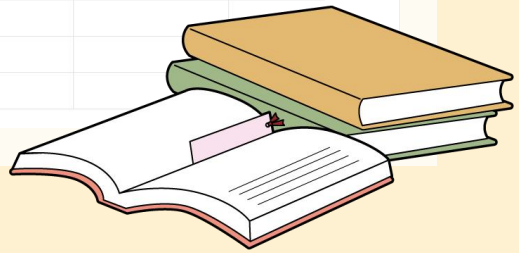
第三部分

数据来源与说明

数据集包含80万条短信信息。原始数据的格式如下：第一列是每一条短信的编号，第二列是短信类别标签（0代表该条短信不是垃圾短信，1代表该条短信是垃圾短信），最后一列是短信内容。

部分短信类别和内容数据如下所示。原始数据集与代码来源：自然语言处理小案例：基于文本内容的垃圾短信分类_自然语言处理文本分类80w垃圾短信csv下载-CSDN博客

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	0	商业秘密的秘密性那是维系其商业价值和垄断地位的前提条件之一												
2	1	南口阿玛施新春第一批限量春装到店啦					春暖花开淑女裙、冰蓝色公主衫			气质粉小西装、冰丝女王长半裙、			皇	
3	0	带给我们大常州一场壮观的视觉盛宴												
4	0	有原因不明的泌尿系统结石等												
5	0	23年从盐城拉回来的麻麻的嫁妆												
6	0	感到自减肥、跳减肥健美操、												
7	1	感谢致电杭州萧山全金釜韩国烧烤店，本店位于金城路xxx号。韩式烧烤等，价格实惠、欢迎惠顾【全金釜韩国烧烤店】												
8	0	这款UVe智能杀菌机器人是扫地机的最佳伴侣												
9	1	一次价值xxx元王牌项目；可充值xxx元店内项目卡一张；可以参与V动好生活百分百抽奖机会一次！预约电话：xxxxxxxxxx												
10	0	此类皮肤特别容易招惹粉刺、黑头等												
11	0	乌兰察布丰镇市法院成立爱心救助基金												
12	1	(长期诚信在本市作各类资格职称（以及印 /章、牌、 ……等。祥：x x x x x x x x x x x 李伟%												
13	1	《依林美容》三．八．女人节倾情大放送活动开始啦！！！！超值套餐等你拿，活动时间x月x日一x月xx日，										详情进店咨询。美丽热线x		
14	0	品牌墙/文化墙设计参考												
15	0	苏州和无锡两地警方成功破获了一起劫持女车主的案件												
16	0	自然之友苏州小组今日下午按原计划举办小组读书活动暨“我为城市量体温”启动仪式												
17	0	共查扣违法三轮车304辆、残疾证23本												
18	0	扬州公司工作人员正在扬州瓜洲镇境内												
19	0	而我不愿意将就~今天调研团的小伙伴们来到了丁莲芳生产基地												
20	0	任何职业的entrylevel都是最简单枯燥的重复												



数据来源与说明

本案例中数据预处理的总体思路是通过数据清理、集成、变换等方法将原始数据中不完整数据、不一致数据、重复数据等去除。

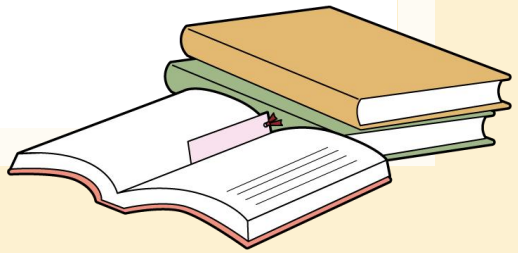
主要有两个步骤：

第一步

对短信文本进行中文分词，去除其中的无用词，例如特殊符号、数字、X等后，把返回的结果存入文本文件中；

第二步

将每一条完成分词的短信变成相对应的特征向量。



前期准备

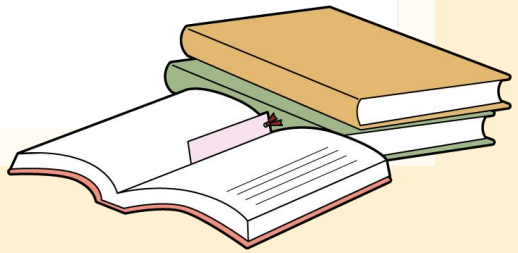
- 安装jieba包。

Jieba是用Python语言写成的一个工业界的分词开源库，用于分词，代码清晰，扩展性好，对已收录词和未收录词都有相应的处理算法。Windows操作系统下，在cmd命令行中输入pip install jieba命令，然后在Python程序中import导入jieba包即可。

- 使用re包是为了支持正则表达式，在分词过程中能够快速地去除不必要的垃圾词。使用时只需在Python程序中直接import导入re包即可。

- 安装pandas包。

pandas包是一个含有更高级的数据结构和工具的数据分析包，它的核心数据结构是一位序列和二维表，可以很好地处理二维结构的csv数据文件。



python代码详解

第四部分

代码详解（data_process）

```
import pandas as pd
import re
import jieba

def data_process(file='message80W1.csv', random_state=42):
    # 读取CSV文件，设置第一列为索引
    data = pd.read_csv(file, header=None, index_col=0)
    data.columns = ['label', 'message']
    n = 5000

    # 随机抽取正常和垃圾短信各n条，（可选）设置随机数种子
    a = data[data['label'] == 0].sample(n, random_state=random_state)
    b = data[data['label'] == 1].sample(n, random_state=random_state)
```

代码详解（data_process）

```
# 合并数据并去重
data_new = pd.concat([a, b], axis=0)
data_dup = data_new['message'].drop_duplicates()

# 数据清洗：删除特定字符x
data_qumin = data_dup.apply(lambda x: re.sub('x', '', x))
# 或者可以用：data_qumin = data_dup.str.replace('x', '')

# 加载自定义词典并分词
jieba.load_userdict('newdic1.txt')
data_cut = data_qumin.apply(jieba.lcut)
```

代码详解（data_process）

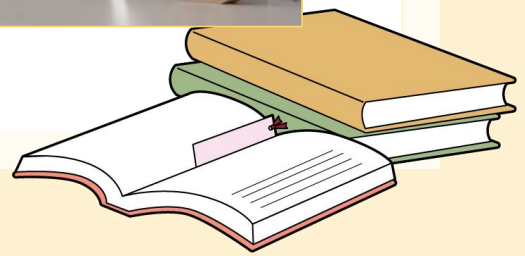
```
    # 加载停用词表并去除停用词
    stopWords = pd.read_csv('stopword.txt',
                            encoding='utf-8', sep='/n', header=None,
                            engine='python')
    stopWords = ['≮', '≯', '≠', ' ', '会', '月', '日', '-']
                + list(stopWords.iloc[:, 0])
    data_after_stop = data_cut.apply(lambda x: [word for word in x if word
                                     not in stopWords])
    # 获取标签并拼接文本
    labels = data_new.loc[data_after_stop.index, 'label']
    adata = data_after_stop.apply(lambda x: ' '.join(x))
    return adata, data_after_stop, labels

# 调用数据处理函数
adata, data_after_stop, labels = data_process(random_state=42)
```

分词结果展示

message
['株洲市', '芦淞区', '嘉盛雅苑', '栋', '楼顶', '一间', '木屋']
['PrimeDay', '促销', '活动', '凌晨']
['发送', '歌曲', 'demo', '一步步', '走', '录制']
['清', '融资', '分享']
['南京', '襟', '防营', '陆军', '警察队', '统辖', '图案']
['名为', 'TheSkysphere']
['搞了半天', '关机', '百度', '关掉']
['145', '水', '宝宝', '防晒', '乳液']
['Amazon', '2200', '多用户', '评论']
['玩具', '机器人', '公司', 'WobbleWorks', '旗下', '称为', '全球', '首款', '3D', '打印', '笔', '3Doodler', '成功', '3D', '技术', '带来', '便利', '发挥', '极致']
['发起', '投票', '爱女心切', '理解']
['妈妈', '玩儿', '电脑', '收拾', '整理', '家']
['QQ', '好友', '发', '验证码']
['现已', '环', '卫星', '直径', '30', '厘米', '冰块']
['今晚', '幾個', '同學', '聊聊', '聊原', '來', '有人', '覺得', 'cool', '高', '竇', '好多', '男朋友', '好多', '追', '係', '們', '猜', '對', '老實', '真的', '要扮', 'cool', '扮高', '竇才', '保護', '麻', '至於', '喜歡', '還有', '什麼', '選擇']
['微软', 'Windows', '操作系统', '工程院', '工程师', '23', '清华', '同学', '分享', '实战', '中', '例子']
['判决', '被告', '李文', '赔偿', '原告', '李明', '1798']
['办', '信用卡', '升', '一级', '代理']
['推荐', '这家', '微店', '有缘', '商铺']
['眼睛', '真相', '瞬间', '输', '一败涂地']
['曾任', '潜艇', '部队', '司令', '海军', '总司令', '第三', '帝国', '国家元首', '武装部队', '统帅']
['爸妈', '说', '手机', '相机', '美颜', '大脸', '拍成', '瓜子脸']
['老婆', '爱', '家里', '来场', '坦克', '大战']
['临时', '南京大屠杀', '纪念馆', '云锦', '博物馆']
['汽车', '改装成', '4G', '多功能', '通讯', '指挥车']
['我今', '一把', '花千骨', '剧情', '公布']

至此，文本分词结束，并在此过程中去除了一些无用的词，但是这些数据仍无法直接用分类方法进行分类，因为大部分的分类方法期望的输入是固定长度的特征向量而不是不同长度的文本文件，所以要进行第二步 —— 特征向量的提取。中文词转特征向量的方法有多种，这里使用TF-IDF 法。

它是一种用于信息检索和文本挖掘的常用加权技术，用于评估一个词语在文档中的重要程度。它结合了词频（TF）和逆文档频率（IDF）两个指标，能够有效地反映词语在文档中的重要性。



代码详解（model）

```
from data_process import data_process
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix
import jieba

# 调用数据处理函数
adata, data_after_stop, labels = data_process()

# 分割训练样本和测试样本，训练样本占80%，测试样本占20%
data_tr, data_te, labels_tr, labels_te = train_test_split(adata,
labels, test_size=0.2, random_state=26)
```

代码详解（model）

```
# 创建CountVectorizer对象
countVectorizer = CountVectorizer()

# 对训练集的文本数据进行拟合和转换，生成词频矩阵
data_tr = countVectorizer.fit_transform(data_tr)

# 使用已有的词汇表对测试集进行转换
data_te = countVectorizer.transform(data_te)

# 计算训练集、测试集的TF-IDF特征向量
tfidf_transformer = TfidfTransformer()
X_tr = tfidf_transformer.fit_transform(data_tr).toarray()
X_te = tfidf_transformer.transform(data_te).toarray()
```

代码详解（model）

```
# 创建高斯朴素贝叶斯分类器
model = GaussianNB()

# 使用训练集数据进行模型训练
model.fit(X_tr, labels_tr)

# 使用训练好的模型对测试集进行预测
predictions = model.predict(X_te)

# 计算预测准确率
accuracy = accuracy_score(labels_te, predictions)
print(f"预测准确率: {accuracy * 100:.2f}%")

# 计算混淆矩阵
cm = confusion_matrix(labels_te, predictions)
print("混淆矩阵:")
print(cm)
```

代码详解（model）

```
tn, fp, fn, tp = cm.ravel()
print(f"真阴性 (TN): {tn}")
print(f"假阳性 (FP): {fp}")
print(f"假阴性 (FN): {fn}")
print(f"真阳性 (TP): {tp}")

# 计算精确率、召回率和F1分数
precision = tp / (tp + fp) if (tp + fp) > 0 else 0
recall = tp / (tp + fn) if (tp + fn) > 0 else 0
f1 = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0

print(f"精确率 (Precision): {precision:.2f}")
print(f"召回率 (Recall): {recall:.2f}")
print(f"F1 分数: {f1:.2f}")
```

代码详解（model）

```
# 用户输入短信的分类
def classify_message(message):
    # 分词
    words = " ".join(jieba.lcut(message))
    # 转换为特征向量
    message_vec = countVectorizer.transform([words])
    message_tfidf = tfidf_transformer.transform(message_vec).toarray()
    # 预测
    prediction = model.predict(message_tfidf)
    return "垃圾短信" if prediction[0] == 1 else "正常短信"
```

代码详解（model）

```
# 支持用户多次输入
while True:
    user_message = input("请输入一条短信（输入 'q' 退出）：")
    if user_message.lower() == 'q':
        print("程序已退出。")
        break
    result = classify_message(user_message)
    print(f"分类结果: {result}")
```

代码详解（model）

输出结果

```
预测准确率: 89.14%
混淆矩阵:
[[820 154]
 [ 63 961]]
真阴性 (TN): 820
假阳性 (FP): 154
假阴性 (FN): 63
真阳性 (TP): 961
精确率 (Precision): 0.86
召回率 (Recall): 0.94
F1 分数: 0.90
```

89.59%的准确率表明模型在大多数情况下能够正确预测。并且对于用户输入的短信，识别准确率也较高。

总体来看，这个模型的性能表现较好，具有较高的准确率、精确率和召回率，F1分数也较高，说明模型在分类任务中表现优异。

```
请输入一条短信（输入 'q' 退出）：【高德打车】您有3张打车券即将过期，最高减12元，快来使用 https://surl.amap.com/E5B9ZKw1YH 拒收请回复R
分类结果: 垃圾短信
请输入一条短信（输入 'q' 退出）：【财商学习】尊敬的9478用户，您账号的理财课程已生效，请尽快激活 b.u5x.cn/0ANC1OT 拒收请回复R
分类结果: 垃圾短信
请输入一条短信（输入 'q' 退出）：【阿里拍卖】“阿里司法拍卖瑞士滚...”的保证金已释放至您的银行卡，详见https://c.tb.cn/n5.00LyZS
分类结果: 垃圾短信
请输入一条短信（输入 'q' 退出）：今年3月12日，是中国第47个植树节。履行植树义务、共建美丽中国。黑龙江省人民政府绿化委员会办公室、黑龙江省林业和草原局宣。
分类结果: 正常短信
请输入一条短信（输入 'q' 退出）：q
程序已退出。
```

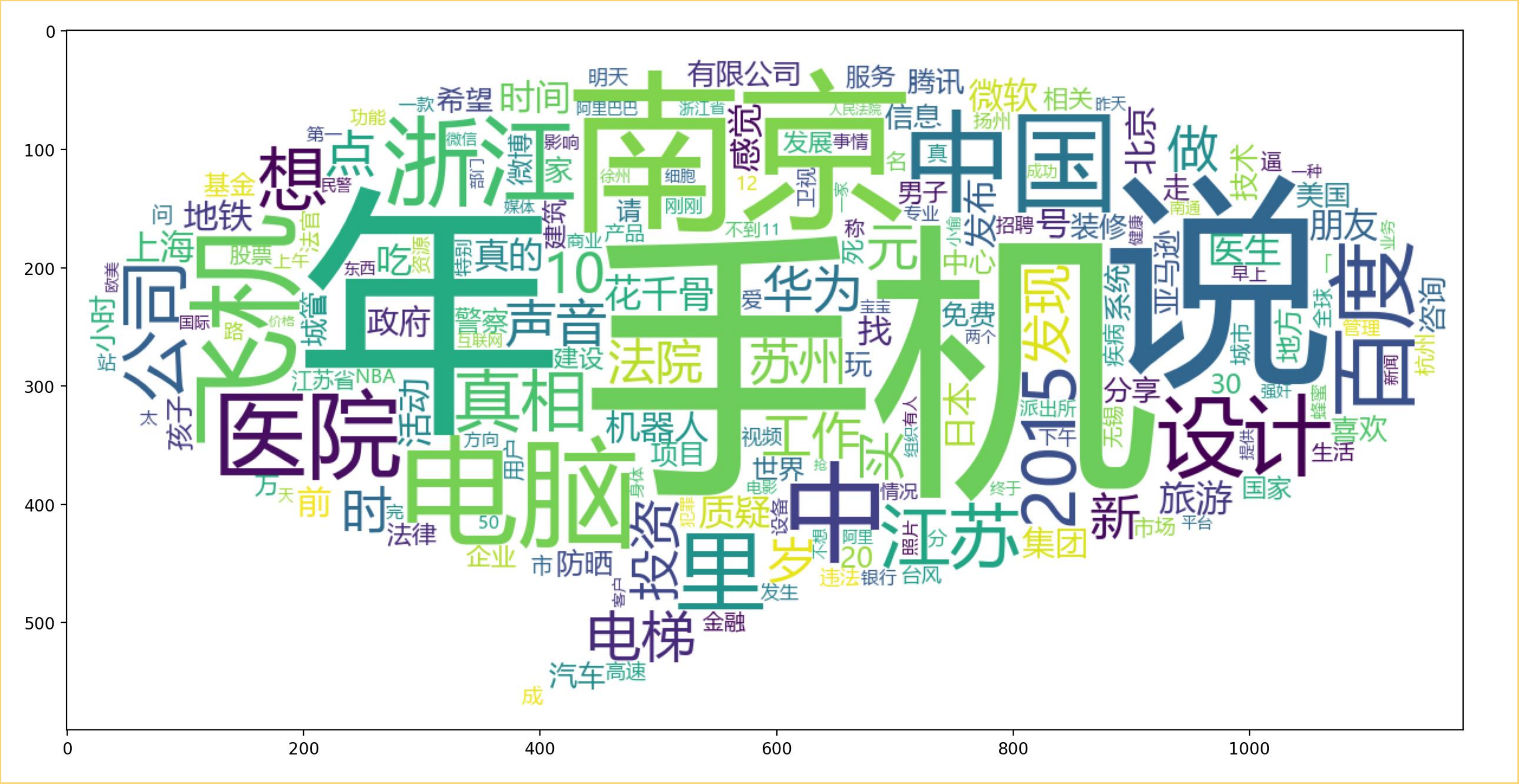
代码详解（word_cloud）

```
from data_process import data_process
from wordcloud import WordCloud
import matplotlib.pyplot as plt

word_fre = {}          # 创建一个空字典,用于存储每个词语的频率
adata, data_after_stop, labels = data_process()

for i in data_after_stop[labels == 0]:    # 遍历每条正常短信的分词结果
    for j in i:
        if j not in word_fre.keys():
            word_fre[j] = 1
        else:
            word_fre[j] += 1
```

代码详解（word_cloud）

```
mask = plt.imread('duihuakuan.jpg')     # 读取图片文件,作为词云的背景形状
wc = WordCloud(mask=mask, background_color='black',
font_path=r'C:\Windows\Fonts\msyh.ttc')

# 根据word_fre字典中的词频数据生成词云
wc.fit_words(word_fre)

# 使用 plt.imshow 显示生成的词云图像
plt.imshow(wc)

plt.show()
```

词云图


手机
年
说
南京
中国
浙江
飞机
医院
电脑
中
里
百度
设计
真相
公司
江苏
2015
发现
投资
电梯
想
做
声音
法院
苏州
华为
工作
花千骨
10
发布
北京
朋友
医生
旅游
新
机器人
买
元
时
真的
时间
希望
有限公司
服务
腾讯
微软
相关
信息
感觉
发展
事情
地铁
上海
基金
点
政府
警察
建设
活动
孩子
小时
前
法律
集团
日本
免费
亚马逊
疾病
系统
咨询
杭州
装修
技术
美国
美国
喜欢
生活
国家
市场
质疑
企业
防晒
汽车
高速
金融
发生
银行
台风
违法
项目
世界
视频
玩
找
男子
分享
30
20
50
江苏省NBA
0
100
200
300
400
500
600
800
1000

汇报完毕
谢谢观看