你能将商品名称自动化分类吗？

姓名：于沁涵 学号：1910227

摘要

商品分类是商品流通的基础步骤之一，探究商品自动化分类问题具有较高的价值。然而商品分类还存在一些困难，目前普遍存在三大困难有：数据大、信息少、人工繁。

本文针对上述商品分类面对的困难，提出了一套基于商品名称的商品自动化分类体系，建立朴素贝叶斯模型进行分类预测。研究表明：以生鲜类为例的朴素贝叶斯模型，最终得到测试集上的预测准确率为 93.9%，其混淆矩阵二级品类分类情况良好，因此经过上述讨论，我们可以得出此商品自动化分类器表现效果良好，可用于文本分类。

一、背景介绍

商品分类，顾名思义就是根据商品的性质、特点将其划分到合适的类别中。在现代商业社会，商品分类是商品流通的基础步骤之一，它是一项可以为顾客、品牌商、零售商三方都带来收益的商业流程活动。

商品分类有很多的好处，首先对于顾客而言，商品分类可以使其提高搜索效率、降低时间成本、愉悦购物体验；对于品牌商而言，商品分类可以使其优化运营管理、精确商品定位、提高整体利润；对零售商而言，商品分类可以使其优化分类体系、加强商品管理、构筑零售品牌。除此之外，商品分类还为各类商业运营活动带来了丰富的想象空间，如可以利用商品分类数据进行基于商品类别的产品推荐，从购买商品属性上挖掘用户特征、门店特征，进行购物篮分析等等，从而创造潜在价值。

然而商品分类还存在一些困难，目前普遍存在三大困难：数据大、信息少、人工繁。由于商品种类五花八门，商品分类问题往往涉及庞大的数据量。与此同时，新产品层出不穷，商品分类需要紧跟商品更新迭代的速度。采用传统的人工标注的方式进行商品分类和审核已经无法满足大数据时代下商品分类的需求，这种方式不仅工作量大，费时费力，而且判断标准较为主观，误判率较高。因此，对商品进行自动化分类已成为当前的主要发展趋势。但在商品分类的实际应用场景中，自动化分类可借助的信息非常少，尤其是对于规模较小的超市便利店来说，往往只能拿到商品名称的信息，因此有用信息缺乏是商品自动化分类面临的主要挑战。

本文针对上述商品分类面对的困难，提出了一套基于商品名称的商品自动化分类体系，它可以完全利用商品名称信息，以自动化的分类技术大批量高效率

的对海量商品进行分类处理，从而延伸商品价值。

二、数据来源和说明

本文数据借鉴某知名电商的商品分类体系，以食品饮料与保健食品（下文简称食品饮料类）、生鲜两个大类为例进行研究，数据共有 69494 个。其中食品饮料类包括 6 个二级品类、42 个三级品类，共 48970 个数据，生鲜类共有 7 个二级品类、40 个三级品类，共有 20524 个数据。其中，商品名称和品类的变量类型均为文本，具体变量说明如表 1:

表 1 数据变量说明

一级品类	二级品类	三级品类
生鲜类	猪羊羊肉	猪肉、羊肉、牛肉、内脏类
	饮品甜品	其他、冷藏果蔬汁、冰激凌
	水果	柚子、香蕉、苹果、牛油果、柠檬、芒果、梨、蓝莓、火龙果、更多水果、橙子、草莓
	蔬菜	叶菜类、鲜菌菇、茄果瓜类、根茎类、葱姜蒜椒 半加工蔬菜
	禽肉蛋品	鸭肉、其他禽类、鸡肉、蛋类
	冷冻食品	速冻半成品、面点、火锅丸串
	海鲜水产	鱼类、蟹类、虾类、其他水产、海产礼盒、海产干货、海参、贝类
食品饮料类	地方特产	云南、新疆、四川、山西、其他特产、内蒙古、湖南、福建、东北、北京
	粮油调味	有机食品、调味品、食用油、南北干货、米面杂粮、方便食品
	茗茶	养生茶、乌龙茶、铁观音、其它茶、普洱、绿茶 龙井、花果茶、花草茶、红茶、黑茶、白茶
	食品礼券	粽子、月饼、卡券、大闸蟹
	休闲食品	休闲零食、肉干肉脯、蜜饯果干、坚果炒货、饼干蛋糕
	饮料冲调	饮用水、饮料、牛奶乳品、冲饮谷物、成人奶粉

三、描述性分析

（一）品类分布

在选定的商品分类体系中，我们将食品饮料和生鲜类定义为两个一级品类；每个一级品类又进一步分为多个二级品类和三级品类。因此，为了更宏观的展示品类的分布情况，我们绘制了饼图观察一级品类下二级品类的分布情况，如图 1 和图 2：

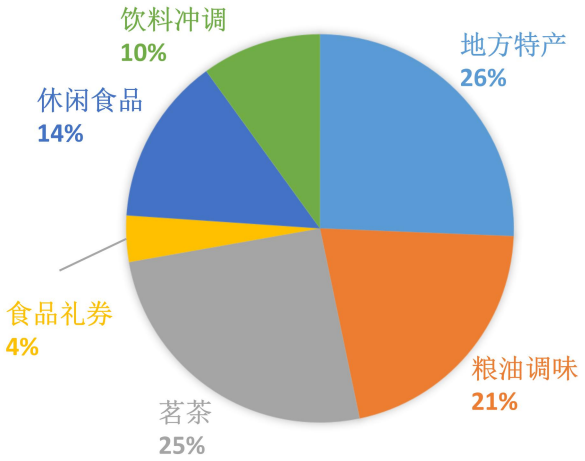

图 1 食品饮料类的二级品类分布图

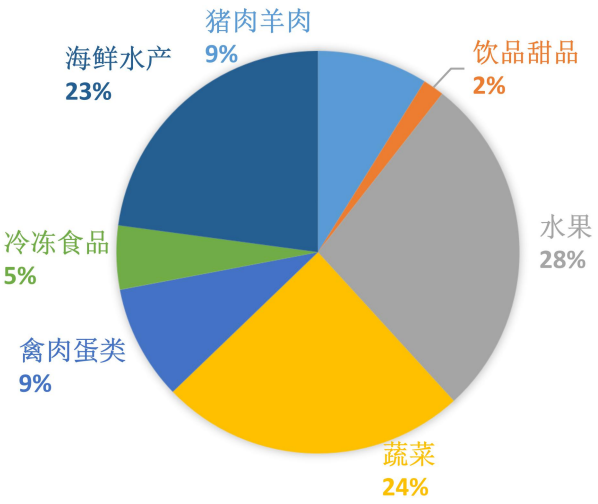

图 2 生鲜类的二级品类分布图

我们发现食品饮料类下共有 6 个二级品类，其中地方特产、茗茶、粮油调味总占比达 70%左右，是食品品类下占比较高的二级品类；生鲜品类下共有 7 个二级品类，其中水果、蔬菜、海鲜水产的总占比达 75%左右，是该品类下占比较高的二级品类。

（二）商品名称的文本分析

由于商品名称是文本数据，我们首先对其进行文本分词等预处理操作。图 3 展示了商品名称中总词数的直方图。可以看出，食品饮料类和生鲜类的产品名称描述词基本都分布在 0～5 个词左右。同时，我们又分别绘制了生鲜类和食品饮料类的分词次数直方图，如图 4、5。我们发现食品饮料类的描述词略微多于生鲜类。

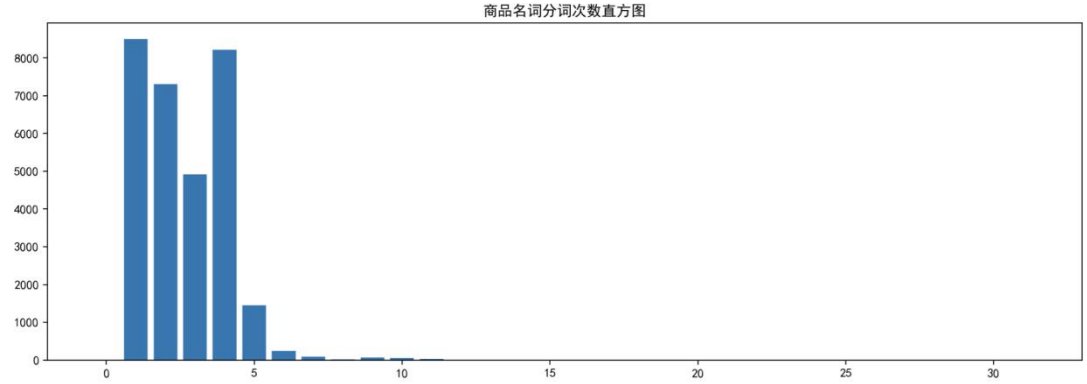

图 3 商品名称中总词数的直方图

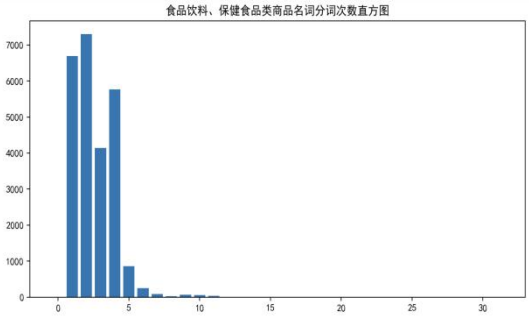

图 4 食品饮料类的分词次数直方图

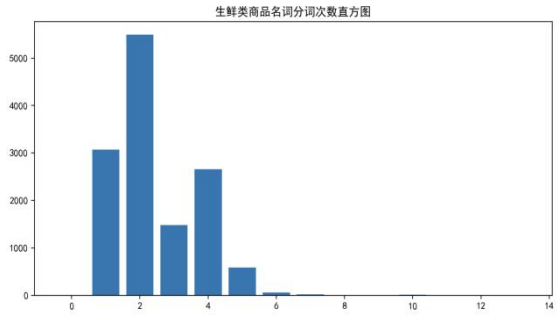

图 5 生鲜类的分词次数直方图

（三）词云图

为了研究不同品类下商品名称的文本特点，我们以一级品类、二级品类的词云图观察每个类别的语言特点。首先，我们绘制一级品类的词云图，如图 3。

图 3 中左图为生鲜类词云图，右图为食品饮料类词云图，可以看到生鲜类中新鲜、海鲜、水果蔬菜、进口等词出现次数最多，这说明人们在生鲜类商品中，商品最多以新鲜来标称，这是因为新鲜、进口等词可以反映商品的质量，这也是生鲜类产品的一个特点；而对于食品饮料中，茶叶、特产、礼盒袋、零食等词出现次数较多，这说明人们在购买食品饮料类商品时，商品多以礼盒的形式呈现，而茶叶和特产则在食品饮料类颇受欢迎，这两者也多以礼盒的形式进行出售，因此在词云图中这几个词出现频率最多。

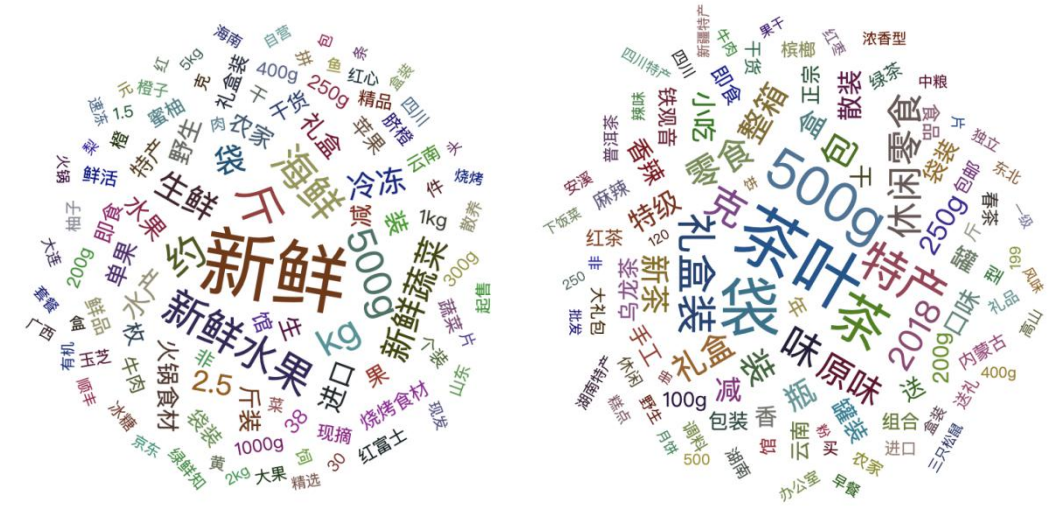

图 6 一级品类商品名称 top100 高频词的词云图

接着，我们绘制一级品类下二级品类的词云图。如图 7～10。可以看出不同级别品类下关键词的种类与词频各不相同，但具有共同趋势；商品的分级品类越细，其呈现的词云结果越能直接反映出该品类产品的特征。

食品饮料类—地方特产　　食品饮料类—茗茶　　食品饮料类—粮油调味

食品饮料类—食品礼券　　食品饮料类—休闲食品　　食品饮料类—饮料冲调

图 7 二级品类食品饮料类词云图

生鲜类—海鲜水产　　生鲜类—冷冻食品　　生鲜类—蔬菜

生鲜类—水果　　生鲜类—饮品甜品　　生鲜类—猪牛羊肉

生鲜类—禽肉蛋品

图 8 二级品类生鲜类词云图

四、商品的自动化分类模型的建立

从上述品类的词云图输出及结论可知，不同品类的商品具有不同的特征词及特征词比例。因此可以认为，根据商品名称的文本特征对商品进行分类，是具有合理性和可行性的。接下来，我们将以商品名称的分词结果为自变量，以商品对应品类为因变量，借助常见的机器学习算法，建立商品的自动化分类模型。

在分类器的选择中，很多分类器都可以选择，但一般来说，朴素贝叶斯分类器对文本数据的分类效果较好，因此我们以朴素贝叶斯分类器来进行展示。朴素贝叶斯分类器是一个很简单但是稳健的分类器，其思想基础是对于任意商品名称，求解在此商品名称出现的条件下，各个类别出现的概率。概率最大的类别就认为该商品名称属于该类别。

朴素贝叶斯建立在两个强假设下：

- 所有属性同等重要
- 样本数据的所有属性之间相互独立

在本文中，文本之间相互独立，符合建模条件，因此我们可以对文本数据建立朴素贝叶斯模型，观察商品分类情况。在此，因为数据的庞大，考虑到代码运行情况，我们以生鲜类为例，对训练集建立朴素贝叶斯分类器。最后，使用建立的分类器对测试集中生鲜大类下的商品进行预测，并计算测试集上的预测准确率。

（一）数据预处理

首先，我们进行数据预处理，从全部分词结果中挑选出用于分类的特征，具体步骤如下：

（1）对所有商品名称进行分词处理，并去掉数字字母组合。

（2）去掉低频词（包含该词的商品数量<10）、高频词（在多于 75%的商品名称中出现）。

（3）将分词后的数据集按照 7:3 比例随机拆分为训练集和测试集。

（4）保留高频词中的 top1000 的词语，作为分类器的特征。

利用 Python 软件，我们得到 top1000 高频词的前 20 名特征，得到直方图如下：

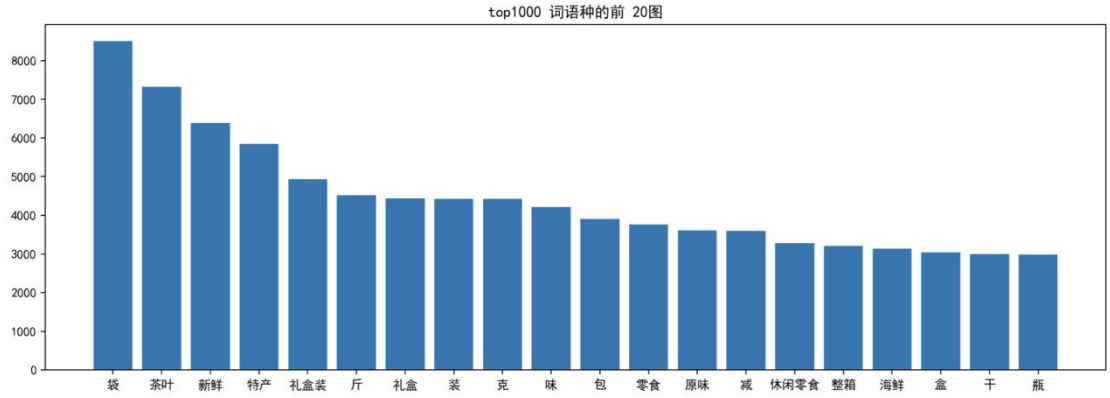

图 11 一级品类商品名称 top100 高频词的前 20 个特征直方图

可以看出袋、茶叶、新鲜、特产这些高频词在词云图中也是占比比较大的，其中食品饮料类略高于生鲜类。

（二）文档-词频矩阵

基于文本预处理的结果，我们进一步构造文档-词频矩阵。考虑到此时数据集中包括的特征词仍然非常庞大，得到文档词频矩阵后，我们进一步简化特征，将其转换成 0-1 矩阵，即包含该高频词取值为 1，不包含该高频词取值为 0。

最后，以训练集为例，得到前 6 篇文档在前 5 个特征词上的取值情况如表 2。从表 2 看出新鲜、特产、礼盒袋比较具有意义的分词也在高频特征词内，这与词云图也相符。

表 2 前 6 篇文档在前 5 个特征词的文档-词频矩阵

文档\特征词	1	2	3	4	5
袋	0	0	0	0	0
茶叶	1	0	0	0	0
新鲜	0	0	1	1	0
特产	0	1	0	0	0
礼盒装	1	0	1	0	0
斤	0	0	0	0	0

（三）朴素贝叶斯模型的建立

以文档-词频矩阵为自变量，每个商品名称对应的二级分类为因变量，以生鲜类为例，对训练集建立朴素贝叶斯分类器。最后，使用建立的分类器对测试集中生鲜大类下的商品进行预测，并得到测试集上的预测准确率为 93.9%，可以初步判断分类效果良好。

观察上述自动化分类器在“生鲜”各个二级品类上的预测准确率，可以得到混淆矩阵如图 12。从混淆矩阵可以看出分类器预测的效果还不错，错判最多的是禽肉蛋品和冷冻食品，这可能是由于两者包含的商品含有一定的交叉性。但是

总体上看，模型的预测能力还是较强的。

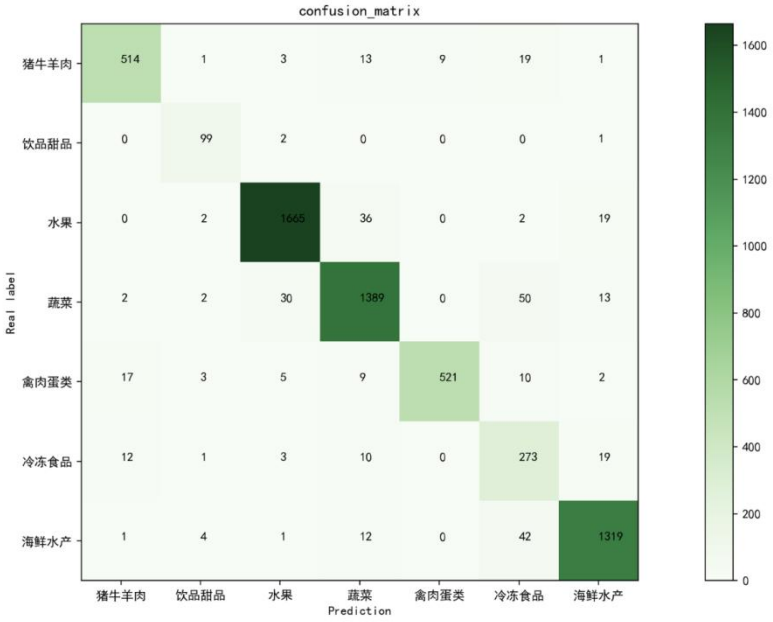

图 12 生鲜类朴素贝叶斯分类器混淆矩阵

五、结论与展望

本文针对上述商品分类面对的困难，提出了一套基于商品名称的商品自动化分类体系，建立朴素贝叶斯模型进行分类预测。我们以生鲜类为例，最终得到测试集上的预测准确率为 93.9%，并通过混淆矩阵观察二级品类分类情况也良好，因此经过上述讨论，我们可以得出此商品自动化分类器表现效果良好，可用于文本分类。

附录 1　Python 语言程序代码

```
# import related packages
import jieba
import os
import numpy as np
import pandas as pd

# read data and cache in words
os.chdir(r'/Users/yuqinhan1229/Desktop/')
txt = open("userdict.txt", encoding="utf-8").read()
words = txt.split('\n')
data = pd.read_csv("catalogs.csv")

# determine whether words is repetitive
if len(words) == len(set(words)):
    print("words are not repetitive")
    max_len = 4

# create a word dictionary
dt = {}
for word in words:
    dt[word] = len(word)

# 删除相同的类别
distri =  data.drop_duplicates(subset = ['first','second','third'])
distri.to_csv('data.csv')
# 查看数据量
data.iloc[:,1].value_counts()

# 任务三
## 加载包
import jieba
from jieba.analyse import extract_tags
from sklearn.feature_extraction.text import TfidfVectorizer
import re
import jieba.analyse
```

```
import codecs
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['font.sans-serif']=['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False # 用来正常显示负号

##定义停词函数
def stopwordlist():
    stopwords = [line.strip() for line in open('stopwords.txt',
encoding='UTF-8').readlines()]
    # ---停用词补充,视具体情况而定---
    i = 0
    for i in range(19):
        stopwords.append(str(10 + i))
    # ----------------------

    return stopwords

##定义分词及统计词频函数
def seg_word(line):
    # seg=jieba.cut_for_search(line.strip())
    seg = jieba.cut(line.strip())
    temp = ""
    counts = {}
    wordstop = stopwordlist()
    for word in seg:
        if word not in wordstop:
            if word != ' ':
                temp += word
                temp += '\n'
                counts[word] = counts.get(word, 0) + 1#统计每个词出现的次数
    return    temp #显示分词结果


##输出分词并去停用词的有用的词到 txt
def output(inputfilename, outputfilename):
```

```
    inputfile = open(inputfilename, encoding='UTF-8', mode='r')
    outputfile = open(outputfilename, encoding='UTF-8', mode='w')
    for line in inputfile.readlines():
        line_seg = seg_word(line)
        outputfile.write(line_seg)
    inputfile.close()
    outputfile.close()
    return outputfile


##生成输入文件
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
f = open('name.txt','w')
for line in name:
    f.write(line+'\n')
f.close()

inputfilename = 'name.txt'
outputfilename = 'a1.txt'
output(inputfilename, outputfilename) ##输出全部商品分词文件
f = open("a1.txt", encoding="utf-8").read()
wordsall = f.split('\n')

##对分词进行计数统计做成词典
dict = {}
for key in wordsall:
    dict[key] = dict.get(key, 0) + 1
wordsall = sorted(dict.items(),key = lambda x:x[1],reverse= True) ##对频率进行排序

##分词长度和频率分别放入列表中，生成数组
keys = []
values = []
for pair in wordsall:
```

```
        keys.append(len(pair[0]))
        values.append(pair[1])

x = np.array(keys)
y = np.array(values)
##画图：商品名词分词次数直方图
plt.figure(figsize=(15, 5), dpi=200)
plt.bar(x, y)
plt.title('商品名词分词次数直方图')

data1 = data[data.iloc[:,1]=="生鲜"]
data1 = data1.dropna()
##生成生鲜类输入文件
txt1 = data1.iloc[:,0]
name = []
for i in range(len(txt1)):
        name.append(str(txt1.iloc[i]))
f = open('name.txt','w')
for line in name:
        f.write(line+'\n')
f.close()
inputfilename = 'name.txt'
outputfilename = '生鲜.txt'
output(inputfilename, outputfilename) ##输出生鲜类商品分词文件
f = open("生鲜.txt", encoding="utf-8").read()
words_s = f.split('\n')
##对分词进行计数统计做成词典
dict = {}
for key in words_s:
        dict[key] = dict.get(key, 0) + 1
words_s = sorted(dict.items(),key = lambda x:x[1],reverse= True) ##对频率进行排序
##分词长度和频率分别放入列表中，生成数组
keys = []
values = []
for pair in words_s:
        keys.append(len(pair[0]))
        values.append(pair[1])
```

```
x = np.array(keys)
y = np.array(values)
##画图：商品名词分词次数直方图
plt.figure(figsize=(10, 5), dpi=200)
plt.bar(x, y)
plt.title('生鲜类商品名词分词次数直方图')

data1 = data[data.iloc[:,1]=="食品饮料、保健食品"]
data1 = data1.dropna()
##生成食品类输入文件
txt1 = data1.iloc[:,0]
name = []
for i in range(len(txt1)):
    name.append(str(txt1.iloc[i]))
f = open('name.txt','w')
for line in name:
    f.write(line+'\n')
f.close()
inputfilename = 'name.txt'
outputfilename = '食品饮料、保健食品.txt'
output(inputfilename, outputfilename) ##输出食品饮料、保健食品类商品分词文件
f = open("食品饮料、保健食品.txt", encoding="utf-8").read()
words_p = f.split('\n')
##对分词进行计数统计做成词典
dict = {}
for key in words_p:
    dict[key] = dict.get(key, 0) + 1
words_p = sorted(dict.items(),key = lambda x:x[1],reverse= True) ##对频率进行排序
##分词长度和频率分别放入列表中，生成数组
keys = []
values = []
for pair in words_p:
    keys.append(len(pair[0]))
    values.append(pair[1])
```

```
x = np.array(keys)
y = np.array(values)
##画图：商品名词分词次数直方图
plt.figure(figsize=(10, 5), dpi=200)
plt.bar(x, y)
plt.title('食品饮料、保健食品类商品名词分词次数直方图')


# 任务四
import pandas as pd
import jieba
import pyecharts.options as opts
from pyecharts.charts import WordCloud

##画生鲜类前 100 个词云图
c = (WordCloud().add("", words_s[0:100], word_gap = 0, word_size_range=[10, 50],
rotate_step = 20, width = 2000, height = 2000, is_draw_out_of_bound = False)
        .set_global_opts(title_opts=opts.TitleOpts(title="")), ))
c.render_notebook()

##画食品饮料、保健食品类词云图
c = (WordCloud().add("", words_p[0:100], word_gap = 0, word_size_range=[10, 50],
rotate_step = 30, width = 2000, height = 2000, is_draw_out_of_bound = False)
        .set_global_opts(title_opts=opts.TitleOpts(title="")), ))
c.render_notebook()

import matplotlib.pyplot as plt
import jieba
from wordcloud import WordCloud
##画二级分类词云图
##猪肉羊肉
data2 = data[data.iloc[:,2]== '猪肉羊肉' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
        name.append(str(txt.iloc[i]))
```

```
name = ''.join(name)
words = jieba.lcut(name)         #精确分词
newtxt = ''.join(words)       #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('猪肉羊肉.jpg')
##饮品甜品
data2 = data[data.iloc[:,2]== '饮品甜品' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
        name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)         #精确分词
newtxt = ''.join(words)       #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('饮品甜品.jpg')
##水果
data2 = data[data.iloc[:,2]== '水果' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
        name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)         #精确分词
newtxt = ''.join(words)       #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('水果.jpg')
##蔬菜
data2 = data[data.iloc[:,2]== '蔬菜' ]
```

```
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('蔬菜.jpg')
##禽肉蛋类
data2 = data[data.iloc[:,2]== '禽肉蛋类' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('禽肉蛋类.jpg')
##冷冻食品
data2 = data[data.iloc[:,2]== '冷冻食品' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
```

```
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('冷冻食品.jpg')
##海鲜水产
data2 = data[data.iloc[:,2]== '海鲜水产' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('海鲜水产.jpg')


##地方特产
data2 = data[data.iloc[:,2]== '地方特产' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('地方特产.jpg')
##粮油调味
data2 = data[data.iloc[:,2]== '粮油调味' ]
data2.dropna()
txt = data.iloc[:,0]
```

```
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('粮油调味.jpg')
##茗茶
data2 = data[data.iloc[:,2]== '茗茶' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('茗茶.jpg')
##食品礼券
data2 = data[data.iloc[:,2]== '食品礼券' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
    name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)        #精确分词
newtxt = ''.join(words)      #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
```

```
wordcloud.to_file('食品礼券.jpg')
##休闲食品
data2 = data[data.iloc[:,2]== '休闲食品' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
        name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)          #精确分词
newtxt = ''.join(words)       #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('休闲食品.jpg')
##饮料冲调
data2 = data[data.iloc[:,2]== '饮料冲调' ]
data2.dropna()
txt = data.iloc[:,0]
name = []
for i in range(len(txt)):
        name.append(str(txt.iloc[i]))
name = ''.join(name)
words = jieba.lcut(name)          #精确分词
newtxt = ''.join(words)       #空格拼接
font=r'/Users/yuqinhan1229/Library/Fonts/SimHei.ttf'#引号里面写字体的路径
wordcloud = WordCloud(font_path =
font,background_color='White').generate(newtxt)
wordcloud.to_file('饮料冲调.jpg')

##任务五
import re
import os
from sklearn import model_selection
#对分词后的数据集进行拆分
data1 = data[data.iloc[:,1]=="生鲜"]
txt = data.iloc[:,0]
```

```
name = []
jieba.load_userdict(r"userdict.txt")
for i in range(len(txt)):
    x = seg_word(str(txt[i]))
    x = x.split('\n')
    name.append(x)

#分词结果需要进行预处理，去掉数字字母组合。
for j in range(len(name)):
    for i in range(len(name[j])):
            name[j][i] = re.sub('[\d]','',name[j][i]) # [0-9]
            name[j][i] = re.sub('[a-zA-Z]','',name[j][i])


#去掉低频词（包含该词的商品数量<10）、高频词（在多于 75%的商品名称中出现）。
#数据集上划分训练集和测试集 7：3
sample_train, sample_test = model_selection.train_test_split(data, test_size=0.3)
data['name'] = name
##对分词进行计数统计做成词典
dict = {}
for x in name:
    for key in x:
        dict[key] = dict.get(key, 0) + 1
wordsall = sorted(dict.items(),key = lambda x:x[1],reverse= True) ##对频率进行排序
wordsall.pop(5) #删除符号
#用列表生成删除的词
words = []
for i in range(len(wordsall)):
    word = wordsall[i][1]
    if word<10 or word>len(wordsall)*3/4:
        words.append(wordsall[i])
#删除指定词
for x in words:
    wordsall.remove(x)
#保留高频词中的 top1000 的词语，作为分类器的特征。
```

```
wordsall = wordsall[0:1000]
#查看最终保留的 top1000 词语种的前 20，并输出结果
keys = []
values = []
for pair in wordsall[0:20]:
    keys.append(pair[0])
    values.append(pair[1])

x = np.array(keys)
y = np.array(values)
##画图：商品名词分词次数直方图
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(15, 5), dpi=200)
plt.bar(x, y)
plt.title('top1000 词语种的前 20 图')
#任务六
#根据上个任务选择的 top1000 词，构建文档-词频矩阵.包含该高频词取值为 1，不包含该高频词取值为 0。
#训练集
w = np.zeros((1000,len(sample_train)))
for i in range(1000):
    for j in range(len(sample_train)):
        if str(wordsall[i][0]) in str(sample_train.iloc[j,0]):
            w[i,j]=w[i,j]+1
#测试集
w_test = np.zeros((1000,len(sample_test)))
for i in range(1000):
    for j in range(len(sample_test)):
        if str(wordsall[i][0]) in str(sample_test.iloc[j,0]):
            w_test[i,j]=w_test[i,j]+1
#最后，以训练集为例，输出其前 6 篇文档在前 5 个特征词上的取值情况
exp = w[0:6,0:5]
dt = pd.DataFrame(exp)
dt['分词'] =
[wordsall[0][0],wordsall[1][0],wordsall[2][0],wordsall[3][0],wordsall[4][0],wordsall[
5][0]]
```

```
dt

#任务七
#以文档词频矩阵为自变量，以每个商品名称对应的二级分类为因变量，建立自动化分类模型。
#很多分类器都可以选择，但一般来说，朴素贝叶斯分类器对文本数据的分类效果较好，因此我们以朴素贝叶斯分类器来进行展示。
##建立模型
from sklearn.naive_bayes import MultinomialNB
X_train = w.transpose()
y_train = sample_train.iloc[:,2]
X_test = w_test.transpose()
y_test = sample_test.iloc[:,2]
clf = MultinomialNB()
clf = clf.fit(X_train, y_train)
y_pred=clf.predict(iris.data)
print("多项分布朴素贝叶斯，样本总数： %d 错误样本数 : %d" %
(iris.data.shape[0],(iris.target != y_pred).sum()))
model.fit(X_train, y_train)

##测试集进行预测验证
y_pred = model.predict(X_test)
print("model accuracy:", metrics.accuracy_score(y_test, y_pred))

##任务八
###混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred, labels=['猪肉羊肉','饮品甜品','水果','蔬菜','禽肉蛋类','冷冻食品','海鲜水产']) #绘制混淆矩阵
classes = ['猪肉羊肉','饮品甜品','水果','蔬菜','禽肉蛋类','冷冻食品','海鲜水产']
confusion_matrix = np.array([(0, 0, 0, 0,0,0,0), (0,99,2,0,0,0,1), (0,2,1665,36,0,2,19),
(0,2,30,1389,0,50,13),(0,0,0,0,0,0,0),(0,1,3,10,0,273,19),(0,4,1,12,0,42,1319)],
dtype=np.int64)
plt.figure(figsize=(15,7),dpi=200)
plt.imshow(confusion_matrix, interpolation="nearest", cmap=plt.cm.Greens)
plt.title("confusion_matrix")
plt.colorbar()
```

```
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes) #设置 X 轴刻度间隔
plt.yticks(tick_marks, classes)
iters = np.reshape([[[i,j] for j in range(7)] for i in range(7)],(confusion_matrix.size,2))
for i, j in iters:
    plt.text(j, i, format(confusion_matrix[i, j])) #显示数字
plt.ylabel('Real label')
plt.xlabel('Prediction')
plt.tight_layout()
plt.show()
```