

BEIJINGINSTITUTEOFPETROCHEMICALTECHNOLOGY

文本分析与大数据可视化

小组报告

|  |  |
| --- | --- |
| 课程名称 | 《文本分析与大数据可视化》 |
| 标 题 | 网易云音乐《成都》评论的文本聚类与分类 |
| 专 业 | 大数据管理与应用 |
| 班 级 | 数管201 |
| 团队组号 | 2 |
| 指导老师 | 宋长宏 |
| 提交时间 | 2023/5/24 |

目录

[一、 作业简介 1](#_Toc136211662)

[1. 作业名称 1](#_Toc136211663)

[2. 数据来源 1](#_Toc136211664)

[二、 小组成员及分工 1](#_Toc136211665)

[三、 文本聚类 1](#_Toc136211666)

[1. 语料加载 1](#_Toc136211667)

[1.1引入所需要的Python 依赖库 1](#_Toc136211668)

[1.2加载停用词字典 2](#_Toc136211669)

[1.3加载语料 2](#_Toc136211670)

[2. 分词、去停用词、特征向量提取 2](#_Toc136211671)

[2.1定义分词、去停用词的函数 2](#_Toc136211672)

[2.2调用函数、生成训练数据 3](#_Toc136211673)

[2.3抽取数据查看结果 3](#_Toc136211674)

[2.4抽取特征 3](#_Toc136211675)

[3. 选择算法、聚类、评测 4](#_Toc136211676)

[3.1TF-IDF的中文文本 K-means 聚类 4](#_Toc136211677)

[3.2定义聚类结果 4](#_Toc136211678)

[3.3可视化 5](#_Toc136211679)

[3.4算法评测 6](#_Toc136211680)

[四、 文本分类 8](#_Toc136211681)

[1. 语料加载 8](#_Toc136211682)

[1.1引入库 8](#_Toc136211683)

[1.2加载停用词字典 9](#_Toc136211684)

[1.3加载语料 9](#_Toc136211685)

[2. 分词、去停用词、特征向量提取 9](#_Toc136211686)

[2.1定义分词、去停用词和批量打标签的函数 9](#_Toc136211687)

[2.2调用函数、生成训练数据 10](#_Toc136211688)

[2.3抽取数据查看结果 10](#_Toc136211689)

[2.4抽取词向量特征 10](#_Toc136211690)

[3. 选择算法、训练 11](#_Toc136211691)

[4. 评测、可视化 11](#_Toc136211692)

[4.1评估、计算 AUC 值 11](#_Toc136211693)

[4.2进行测试集的预测 11](#_Toc136211694)

[4.3可视化 12](#_Toc136211695)

[五、 总结及感想 13](#_Toc136211696)

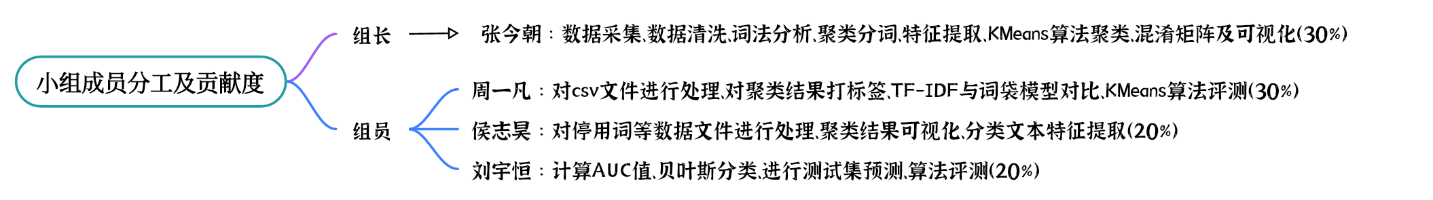
1. 作业简介
2. 作业名称

本小组的题目为“网易云音乐之《成都》的评论文本聚类与文本分类”。我们选择这个标题的名称的原因是小组成员都非常喜欢这首歌曲，并且习惯使用网易云音乐APP，加之小组成员经常阅读网易云音乐歌曲的评论，尤其是热门歌曲。于是便以此首歌曲作为典型案例来开展项目。

1. 数据来源

该首歌曲的链接为[https://music.163.com/#/song?id=436514312](https://music.163.com/" \l "/song?id=436514312)，此链接页面底部即为评论区，评论区首页为35条（15条精彩评论+20条按时间排序评论），其他页为20条按时间排序评论。采集过程初期，拟使用requests和BeautifulSoup模块并调用网易云音乐的API接口进行爬取，但由于API接口的调用方式为POST，并且使用AES加密，采集难度偏大，且本项目的重点内容不在于采集数据部分。因此最后调整思路，使用selenium和edge浏览器进行采集。

1. 小组成员及分工



1. 文本聚类
2. 语料加载

1.1引入所需要的Python 依赖库

import jieba

import pandas as pd

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import TfidfVectorizer

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans

from sklearn import metrics

import warnings

warnings.filterwarnings("ignore")

1.2加载停用词字典

加载stopwords.txt文件。

#加载停用词

stopwords=pd.read\_csv('data\\stopwords.txt',index\_col=False,quoting=3,sep="\t",names=['stopword'], encoding='utf-8')

stopwords=stopwords['stopword'].values

1.3加载语料

加载csv文件，删除nan行，并提取要分词的content列转换为list列表。

#加载语料

music\_df = pd.read\_csv('data\\ChengDu（Ultra）.csv',encoding='utf-8', header=None, names=['content'])

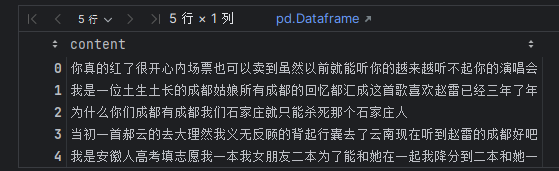
#删除语料的nan行

music\_df.dropna(inplace=True)

#转换

music = music\_df['content'].tolist()

music\_df.head()



1. 分词、去停用词、特征向量提取

2.1定义分词、去停用词的函数

content\_lines为语料列表；sentences 为预先定义的 list，用来存储分词后的结果。

#定义分词和打标签函数preprocess\_text

#参数content\_lines即为上面转换的list

#参数sentences是定义的空list，用来储存打标签之后的数据

def preprocess\_text(content\_lines, sentences):

for line in content\_lines:

segs = jieba.lcut(line)

segs = [v for v in segs if not str(v).isdigit()] #去数字

segs = list(filter(lambda x: x.strip(), segs)) #去左右空格

segs = list(filter(lambda x: len(x) > 1, segs)) #长度为1的字符

segs = list(filter(lambda x: x not in stopwords, segs)) #去掉停用词

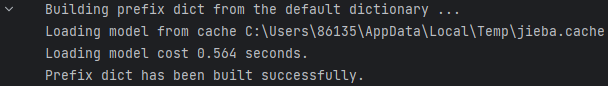
sentences.append((" ".join(segs))) # 打标签

2.2调用函数、生成训练数据

#调用函数，生成训练集

sentences = []

preprocess\_text(music,sentences)

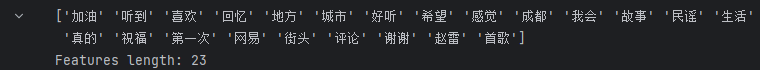


2.3抽取数据查看结果

生成训练集同时观察一下前十条数据。

for sentence in sentences[:10]:

print(sentence)



2.4抽取特征

将文本中的词语转换为词频矩阵，统计每个词语的tf-idf权值，获得词在对应文本中的tf-idf权重。

#将文本中的词语转换为词频矩阵 矩阵元素a[i][j] 表示j词在i类文本下的词频

vectorizer = TfidfVectorizer(sublinear\_tf=True, max\_features=23)

# 统计每个词语的tf-idf权值

transformer = TfidfTransformer()

# 计算tf-idf

tfidf = vectorizer.fit\_transform(sentences)

# 获取词袋模型中的所有词语

word = vectorizer.get\_feature\_names()

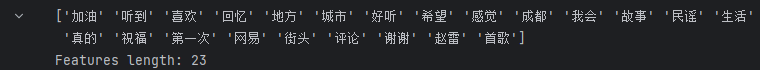
print(word)

# 将tf-idf矩阵抽取出来，元素w[i][j]表示j词在i类文本中的tf-idf权重

weight = tfidf.toarray()

#查看特征大小

print ('Features length: ' + str(len(word)))



1. 选择算法、聚类、评测

3.1TF-IDF的中文文本 K-means 聚类

numClass = 11 # 聚类分几簇

clf = KMeans(n\_clusters=numClass,max\_iter=10000,init="k-means++",tol=1e-6)

pca = PCA(n\_components=2) # 输出两维

TnewData = pca.fit\_transform(weight)

s = clf.fit(TnewData)

print("s:",s)



3.2定义聚类结果

利用可视化函数plot\_cluster，该函数包含3个参数，其中result表示聚类拟合的结果集；newData表示权重weight降维的结果，这里需要降维到2维，即平面可视化；numClass表示聚类分为几簇，绘制代码第一部分绘制结果newData，第二部分绘制聚类的中心点。

# 绘制聚类中心

def plot\_cluster(result, newData, numClass):

plt.figure(figsize=(6, 4))

Lab = [[] for i in range(numClass)]

index = 0

for labi in result:

Lab[labi].append(index)

index += 1

color = ['oy', 'ob', 'og', 'cs', 'ms', 'bs', 'ks', 'ys', 'yv', 'mv', 'bv', 'kv', 'gv', 'y', 'm', 'b', 'k', 'g'] \* 3

for i in range(numClass):

x1 = []

y1 = []

for ind1 in newData[Lab[i]]:

try:

y1.append(ind1[1])

x1.append(ind1[0])

except:

pass

plt.plot(x1, y1, color[i])

#绘制初始中心点

x1 = []

y1 = []

for ind1 in clf.cluster\_centers\_:

try:

y1.append(ind1[1])

x1.append(ind1[0])

except:

pass

plt.plot(x1, y1, "rv") # 绘制中心

plt.show()

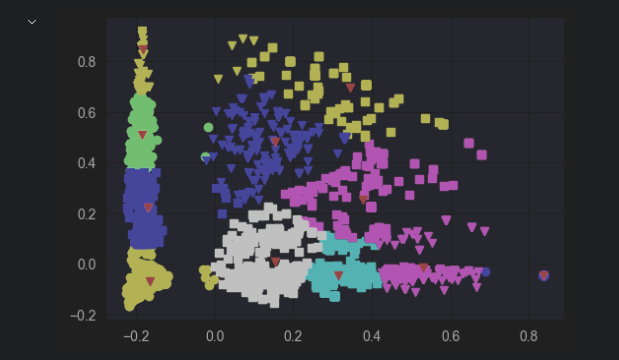
3.3可视化

对数据降维到2维，最后绘制聚类结果。

pca = PCA(n\_components=2)#输出二维

newData = pca.fit\_transform(weight)

plot\_cluster(result,newData,numClass)



聚类结果中能看到是分了11个中心点和11个簇，右边的几簇聚类效果较差。

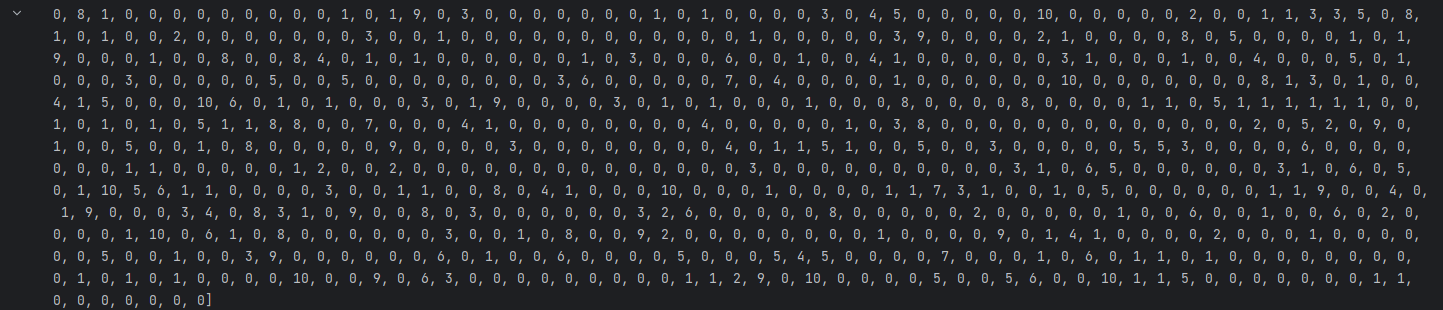
其中前4类极其接近，整体聚类效果在可接受范围内。

3.4算法评测

3.4.1查看聚类结果、贴标签

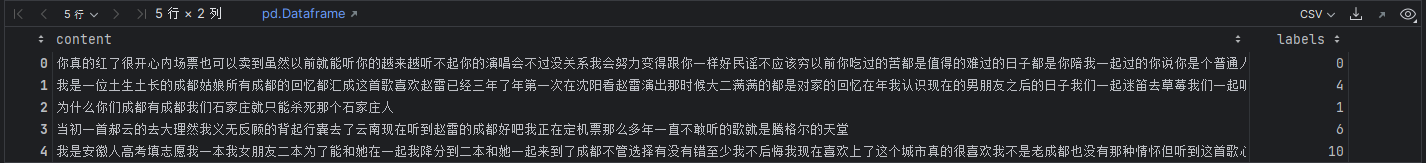
result = list(clf.predict(TnewData))#查看聚类结果

print(result)



music\_df['labels'] = result # 贴标签

music\_df.head()



通过显示更多结果，我们可以分析到，聚类结果比较符合我们的认知。

3.4.2存储到本地

#存储

music\_df.to\_csv('data\\labels.csv')

3.4.3KMeans算法效果评价

def km\_sse\_cs():

"""

KMeans算法效果评价

1、簇内误方差(SSE, sum of the squared errors)，手肘法，肘部法，其大小表明函数拟合的好坏。

使用图形工具肘部法，根据簇的数量来可视化簇内误方差。下降率突然变缓时即认为是最佳的k值（拐点）。

当KMeans算法训练完成后，可以通过使用内置inertia属性来获取簇内的误方差。

2、轮廓系数法（Silhouette Coefficient）结合了聚类的凝聚度（Cohesion）和分离度（Separation）

平均轮廓系数的取值范围为[-1,1]，系数越大，聚类效果越好。当值为负时，暗含该点可能被误分了。

:return:

"""

# 存放设置不同簇数时的SSE值

sse\_list = []

# 轮廓系数

silhouettes = []

data=TnewData

# 循环设置不同的聚类簇数

for i in range(2, 20):

model = KMeans(n\_clusters=i)

model.fit(data)

# kmeans算法inertia属性获取簇内的SSE

sse\_list.append(model.inertia\_)

# 轮廓系数

silhouette = metrics.silhouette\_score(data, model.labels\_, metric='euclidean')

silhouettes.append(silhouette)

# 绘制簇内误方差曲线

plt.subplot(211)

plt.title('KMeans 簇内误方差')

plt.plot(range(2, 20), sse\_list, marker='\*')

plt.xlabel('簇数量')

plt.ylabel('簇内误方差(SSE)')

# 绘制轮廓系数曲线

plt.subplot(212)

plt.title('KMeans 轮廓系数')

plt.plot(range(2, 20), silhouettes, marker='o')

plt.xlabel('簇数量')

plt.ylabel('轮廓系数')

plt.tight\_layout()

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

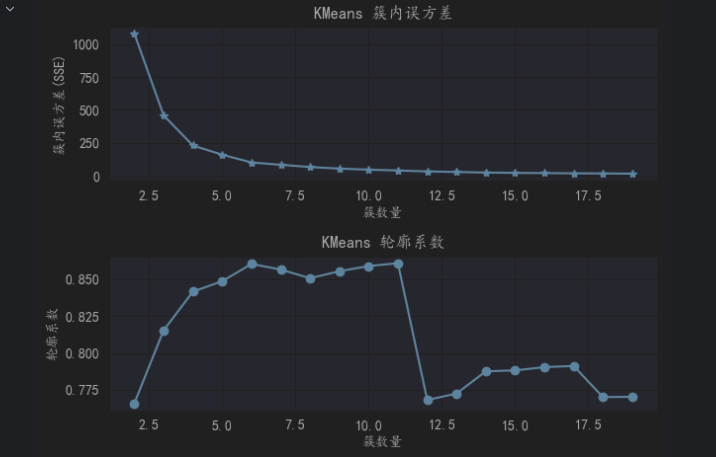
import matplotlib as mpl

# 汉字字体,优先使用楷体，如果找不到楷体，则使用黑体

mpl.rcParams['font.sans-serif'] = ['KaiTi', 'SimHei', 'FangSong']

# KMeans算法

km\_sse\_cs()



通过簇内误方差我们得出最佳K值为2左右，但由于聚类数据量达20000.通过轮廓系数我们可以知道在簇数量为11左右的时候聚类效果最好。但簇不利于我们进行尝试，所以我们选择了次选项，把后续的内容定为分成11簇。

1. 文本分类
2. 语料加载

1.1引入库

import jieba # 分词包

import matplotlib.pyplot as plt # 数据可视化

import pandas as pd # 数据分析

import seaborn as sns # 数据可视化

from sklearn.feature\_extraction.text import CountVectorizer # 特征提取

from sklearn.model\_selection import train\_test\_split # 数据集划分

from sklearn.metrics import confusion\_matrix # 混淆矩阵

# 定义朴素贝叶斯模型，然后对训练集进行模型训练，直接使用 sklearn 中的 MultinomialNB

from sklearn.naive\_bayes import MultinomialNB

import warnings # 忽略警告

import random # 随机数

plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签

plt.rcParams['axes.unicode\_minus'] = False # 解决保存图像是负号'-'显示为方块的问题

warnings.filterwarnings("ignore") # 忽略警告

1.2加载停用词字典

# 加载停用词

stopwords = pd.read\_csv('data\\stopwords.txt', index\_col=False, quoting=3, sep="\t", names=['stopword'], encoding='utf-8')

stopwords = stopwords['stopword'].values

停用词字典集我们也利用了去除特殊词汇的字典，因为收集到的评论有小部分是中性无关词语。

1.3加载语料

*# 加载语料*

*music\_df = pd.read\_csv('data\\labels.csv', encoding='utf-8')*

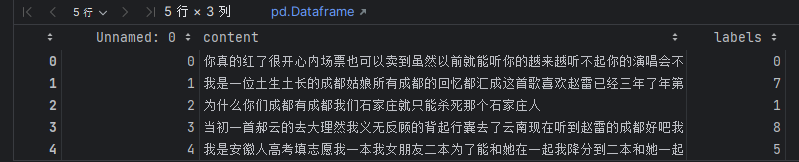
*music\_df.head()*

# 删除语料的nan行

music\_df.dropna(inplace=True)

content = music\_df['content'].tolist()

labels = music\_df['labels'].tolist()



1. 分词、去停用词、特征向量提取

2.1定义分词、去停用词和批量打标签的函数

函数包含3个参数：content\_lines参数为语料列表；sentences参数为预先定义的list，用来存储分词并打标签后的结果。

#定义分词和打标签函数preprocess\_text

#参数content\_lines即为上面转换的list

#参数sentences是定义的空list，用来储存打标签之后的数据

def preprocess\_text(content\_lines, sentences):

for line in content\_lines:

segs = jieba.lcut(line)

segs = [v for v in segs if not str(v).isdigit()] #去数字

segs = list(filter(lambda x: x.strip(), segs)) #去左右空格

segs = list(filter(lambda x: len(x) > 1, segs)) #长度为1的字符

segs = list(filter(lambda x: x not in stopwords, segs)) #去掉停用词

sentences.append((" ".join(segs))) # 打标签

2.2调用函数、生成训练数据

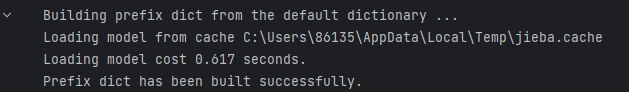
# 转换

music = music\_df['content'].tolist()

#调用函数，生成训练集

sentences = []

preprocess\_text(music, sentences)



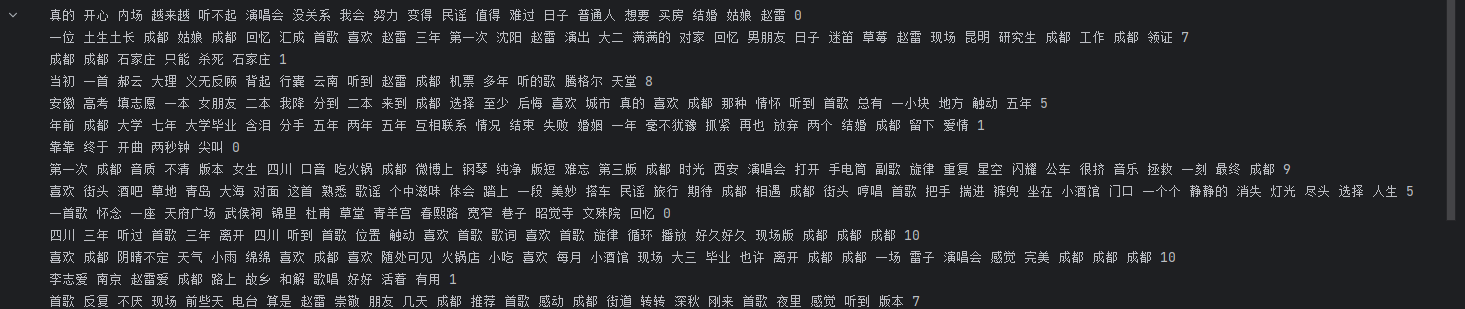
2.3抽取数据查看结果

将得到的数据集打散，生成更可靠的训练集同时观察一下前20条数据。

random.shuffle(sentences)

for sentence in sentences[:20]:

print(sentence[0], sentence[1]) #下标0是词列表，1是标签



2.4抽取词向量特征

2.4.1抽取词袋模型特征

vec = CountVectorizer(

analyzer='word', # tokenise by character ngrams

max\_features=23, # keep the most common 1000 ngrams

)

2.4.2sk-learn 对数据切分，分成训练集和测试集

x, y = zip(\*sentences)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=42)

len(x\_train)



2.4.3把训练数据转换为词袋模型

#把训练数据转换为词袋模型：

vec.fit(x\_train)

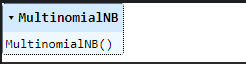


1. 选择算法、训练

定义朴素贝叶斯模型，然后对训练集进行模型训练，直接使用 sklearn 中的 MultinomialNB

classifier = MultinomialNB()

classifier.fit(vec.transform(x\_train), y\_train)



1. 评测、可视化

4.1评估、计算 AUC 值

#评估、计算 AUC 值

print(classifier.score(vec.transform(x\_test), y\_test))



得到的评分结果为：0.9434113177364527

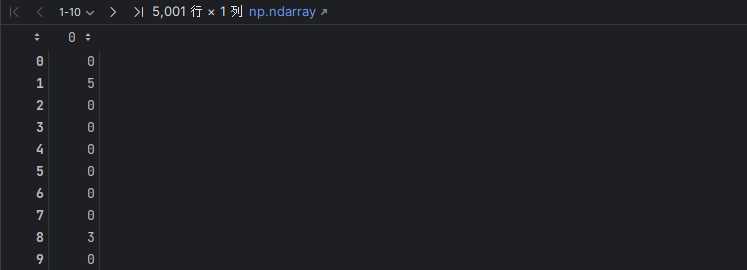
调整训练集测试集比例后评分在0.93到0.98之间，没有太大差别，虽然能正常得出结果但也可以看出我们的语料库存在问题。不能很好的进行分析。后续的预测错误率应该会比较大。综合来看主要错误率会体现在后四类中。

4.2进行测试集的预测

# 进行测试集的预测：

pre = classifier.predict(vec.transform(x\_test))

pre



4.3可视化

4.3.1混淆矩阵

# 混淆矩阵

# 真实标签

y\_true = y\_test

# 预测标签

y\_pred = classifier.predict(vec.transform(x\_test)).tolist()

c = confusion\_matrix(y\_true, y\_pred)

c



4.3.2热力图

# 绘制热力图

# fmt参数用于指定数值的格式

# cmap参数用于指定颜色映射

# annot参数用于指定是否在热力图中显示数值

# 绘制热力图

plt.figure(figsize=(8, 6))

sns.heatmap(c, annot=True, cmap='summer', fmt='d', vmax=300)

plt.xlabel('预测标签')

plt.ylabel('真实标签')

plt.show()



混淆矩阵热力图主要是看对角线上的数值，在8：2比例的测试集训练集设置下，正确预测1、2、4、7、8类的准确率很高，当然我们也可以预见其他类被错误预测的结果，导致9、10类的预测结果非常差。调整比例后也是如此，正确预测的数量主要还是第一类。抛开算法的因素，虽然结果不太准确，但总体的过程和结果比较符合我们的预期。

1. 总结及感想

**张今朝**：本次的《文本分析与大数据可视化》上机课已经圆满结束，通过这几周的实践与学习，让我们对于文本分析与大数据可视化的整个一个最基本的过程有了更深的理解与体会。

1. 聚类和分类过程：

在文本分析中，聚类和分类是常用的技术手段。聚类是将相似的文本样本归为一类，而分类则是根据预定义的类别标签将文本样本进行分类。聚类过程可以通过聚类算法（如K-means）实现，而分类过程则可以使用机器学习算法（如贝叶斯分类）来实现。

1. 特征提取：为了进行文本分析，我们需要将文本数据转化为机器可以理解和处理的形式。特征提取是这个过程中的关键一步，它可以将文本数据转化为数值型的特征向量。常用的特征提取方法包括词袋模型，将文本转化为词频向量表示。
2. K-means算法：是一种常用的聚类算法，它通过迭代的方式将样本划分为K个簇。算法的核心思想是通过最小化样本与各个簇中心的距离来不断更新簇的中心位置，直到达到收敛条件。K-means算法可以帮助我们发现数据中的相似模式和结构。
3. 贝叶斯分类：是一种基于概率统计的分类方法。它利用贝叶斯定理计算文本属于每个类别的概率，并选择概率最大的类别作为分类结果。
4. 混淆矩阵：是一种用于评估分类模型性能的矩阵表示。它展示了模型预测结果与真实标签之间的对应关系。通过混淆矩阵，我们可以计算出准确率、召回率、F1值等指标，评估模型的分类效果。
5. 热力图：是一种常用的大数据可视化技术，用于可视化数据的分布和关联程度。在文本分析中，可以使用热力图来展示词语的共现关系、相关性等。热力图通过颜色的深浅来表示不同区域的数值大小，帮助我们直观地理解数据。其中matplotlib绘制heatmap，方法比较繁琐，要调用很多辅助函数才能实现效果更好的热图。Seaborn是基于matplotlib的Python可视化库。它使得作图更加容易，不需要经过大量的调整就能使你的图达到预定期望。

总之，在小组成员的互帮互助情况下，我们顺利完成了我们的任务并且收获颇丰，同时也认识到了自身的不足，还是有很多东西是需要自己去主动学习的。

**周一凡**：在本课程中，我们深入学习了文本分析和大数据可视化的核心概念和技术。通过理论知识的学习和实践项目的实施，我们获得了丰富的经验和技能，可以有效地处理和分析大规模的文本数据，并将数据转化为可视化形式，以便更好地理解和传达数据的信息。以下是本课程的关键内容和总结。

文本分析的分类过程：文本分类是将文本按照其主题或类别进行分类的过程。我们学习了几个关键步骤：

1. 数据预处理：对文本数据进行清洗、标记化和去除停用词等处理，以准备数据进行后续分析。
2. 特征提取：利用词袋模型和TF-IDF等方法，将文本转化为向量表示，以捕捉文本的重要特征。
3. 训练和评估分类模型：使用机器学习算法如朴素贝叶斯、支持向量机等，通过已标记的训练数据对分类模型进行训练，并使用评估指标如准确率、召回率等来评估分类模型的性能。
4. 文本分析的聚类过程：文本聚类是将文本按照相似性进行分组的过程。
5. 数据预处理：同样需要对文本数据进行清洗和标记化等处理，确保数据质量。
6. 特征提取：与文本分类类似，将文本转化为特征向量表示。
7. 聚类算法应用：我们重点学习了K-means算法，它能够根据文本的相似性将其聚类为预先指定的类别。K-means算法通过迭代计算文本与聚类中心的距离，将文本分配给最近的聚类中心。
8. K-means算法： K-means算法是一种常用的聚类算法，它通过迭代计算文本与聚类中心的距离，将文本分配给距离最近的聚类中心。该算法需要预先指定聚类的数量K，并通过不断迭代优化聚类结果。K-means算法是一种高效的聚类方法，适用于大规模文本数据的处理。
9. 大数据可视化是将大规模的数据以图形方式呈现，以帮助人们理解数据的模式和趋势。我们学习了matplotlib等工具，提供了丰富的功能和库，使我们能够创建出高质量的大数据可视化。

通过本课程的学习，我们掌握了文本分析和大数据可视化的核心技术和方法。我们学会了如何进行文本分类和聚类，利用词袋分析和K-means算法等进行文本特征提取和聚类分析。此外，我们也学会了如何使用图表、图形和交互式可视化等技术，将大数据转化为可视化形式，以便更好地理解和传达数据的信息。

**侯志昊**：在这门课程中，我探索了文本分析与大数据可视化的精彩世界，通过深入学习相关概念、方法和技术，获得了宝贵的知识与实践经验。以下是我对这门课程的总结和反思。

为了进行文本分析，特征提取是至关重要的一步。我学会了使用词袋模型，将文本转化为词频向量的表示形式。词袋模型的优势在于简单高效，适用于许多文本分析任务。通过特征提取，我能够将文本数据转化为机器可以理解和处理的形式，为后续的分析奠定基础。

在课程中，我深入学习了K-means算法，这是一种常用的聚类算法。K-means算法通过迭代的方式将样本划分为K个簇，通过最小化样本与簇中心之间的距离来不断更新簇的中心位置，直到达到收敛条件。通过掌握K-means算法，我能够发现数据中的聚类模式，为数据的进一步分析提供了指导。

此外，贝叶斯分类也是课程中的重要内容。贝叶斯分类是一种基于概率统计的分类方法，通过计算文本属于每个类别的概率，并选择概率最大的类别作为分类结果。我了解了贝叶斯分类器的原理和应用。

在评估文本分类模型的性能方面，我学习了混淆矩阵的使用。混淆矩阵提供了一种直观的方式来评估模型的分类效果，展示了模型预测结果与真实标签之间的对应关系。通过混淆矩阵，我们可以计算准确率、召回率、F1值等指标，深入了解模型的分类表现。

大数据可视化是将庞大的数据转化为直观可视形式的技术，能够帮助我们更好地理解和传达数据的信息。在课程中，我学会了使用热力图进行大数据可视化。热力图通过颜色的深浅来展示数据的分布和关联程度。

**刘宇恒：**本次的文本分析与大数据可视化课程中让我学到了很多有用的东西，不仅拓展了我对这个专业的专业知识，并且对我以后做毕业设计也提供了思路和帮助。本次的小组实验我们小组选取的题目是网易云音乐《成都》评论的文本聚类与分类，在完成题目的同时我学到了以下知识：

1. 语料库的加载：通过引入第三方模块的方法来加载我们爬取到的语料库以及我们自己创建的停用词字典。
2. 分词、去停用词及特征向量的提取：将语料库中的文本进行分词并去除停用词，读取得到的结果文件并返回一个二维列表，为向量化做准备。
3. KMeans算法聚类：使用sklearn库的TfidfVectorizer算法进行KMeans算法聚类，传入一个分词的二维列表和一个聚类方法名，返回的是一个向量化后的二维矩阵，创建一个对象，对象用于将文档向量化，将列表转换为一个矩阵。该矩阵的每一行都代表一个文档的向量化表示，最后进行聚类，将聚类结果写入文档。
4. 对聚类结果打标签：使用KMeans算法的predict函数输出标签结果，再用pandas模块为csv文件贴上标签。
5. 评测：通过簇内误方差的手肘法，肘部法，其大小表明函数拟合的好坏。通过轮廓系数法，该方法结合了聚类的凝聚度和分离度，系数越大，聚类效果越好。
6. 可视化：通过绘图工具画出混淆矩阵的热力图，以便清晰的展示出结果。