合肥工業大學

# 数据挖掘实验报告

实验名称 20news-bydate数据集的分类

姓名 付炎平

班级 物联网工程19-2班

学号 2019217819

2022 年 6 月 24 日

目录

# 实验： 基于 20news-bydate 数据集的分类任务

## 一 实验目的

1）理解分类任务

2）考察学生对数据预处理步骤的理解，强化预处理的重要性

3）基模型可以调用已有的包，训练学生熟悉数据挖掘的基本框架

4）学会多维度的对模型进行评估以及模型中参数的讨论

## 二 实验任务

基于 20news-bydate数据集（包含训练集和测试集 ）完成分类任务，要满足以下要求：

1）要有针对数据特点的预处理步骤

2）原则上不限制模型，决策树，NB，NN，SVM，random forest 均可，且不限于上述方法

3）文本可采用 BOW，主题模型以及词向量等多种表示方式，图像数据集可采用 LBP，HOG，SURF 等特征表示方式。

4）实现一个或多个基本分类模型，并计算其评估指标如准确率，召回率等

5）对模型中的关键参数，（如决策树中停止分裂条件，NN 中层数等参数）进行不同范围的取值，讨论参数的最佳取值范围。

6）对比分析不同的特征表示方法对结果的影响。

7）若对同一数据采用两种或多种模型进行了分类，对多种模型结果进行对比，以评估模型对该数据集上分类任务的适用性。

## 三 实验环境

开发平台：Windows10 操作系统

开发环境：VSCode、Jupyter Notebook、Anaconda、pandas、sklearn、matplotlib 等等

## 四 实验内容

### 4.1 数据集加载

在这里，我使用sklearn.datasets里封装的fetch_20newsgroups对数据集进行加载,具体代码如下所示：

```
categories = ['alt.atheism', 'soc.religion.christian','comp.graphics', 'sci.med']
twenty_train = fetch_20newsgroups(data_home='./dataset',subset='train',
                                  categories=categories,shuffle=True,random_state=42)

twenty_test = fetch_20newsgroups(data_home='./dataset',subset='test',
                                 categories=categories, shuffle=True, random_state=42)
```

加载完的数据集内容如图所示：

```
['From: sd345@city.ac.uk (Michael Collier)\nSubject: Converting images to HP LaserJ
et III?\nNntp-Posting-Host: hampton\nOrganization: The City University\nLines: 14\n
\nDoes anyone know of a good way (standard PC application/PD utility) to\nconvert t
if/img/tga files into LaserJet III format.  We would also like to\ndo the same, con
verting to HPGL (HP plotter) files.\n\nPlease email any response.\n\nIs this the co
rrect group?\n\nThanks in advance.  Michael.\n-- \nMichael Collier (Programmer)
The Computer Unit,\nEmail: M.P.Collier@uk.ac.city                The City Universit
y,\nTel: 071 477-8000 x3769                      London,\nFax: 071 477-8565
EC1V 0HB.\n',
 "From: ani@ms.uky.edu (Aniruddha B. Deglurkar)\nSubject: help: Splitting a trimmin
g region along a mesh \nOrganization: University Of Kentucky, Dept. of Math Science
s\nLines: 28\n\n\n\n\tHi,\n\n\tI have a problem, I hope some of the 'gurus' can hel
p me solve.\n\n\tBackground of the problem:\n\tI have a rectangular mesh in the uv
domain, i.e  the mesh is a \n\tmapping of a 3d Bezier patch into 2d. The area in th
is domain\n\twhich is inside a trimming loop had to be rendered. The trimming\n\tlo
op is a set of 2d Bezier curve segments.\n\tFor the sake of notation: the mesh is m
ade up of cells.\n\n\tMy problem is this :\n\tThe trimming area has to be split up
into individual smaller\n\tcells bounded by the trimming curve segments. If a cell
\n\tis wholly inside the area...then it is output as a whole ,\n\telse it is trivia
```

### 4.2 去停用词和去符号-数据预处理1

从加载完数据集的结果可以看出，加载的数据中有很多没有意义的标点和字符，在这里我们需要进行去停用词和去掉没有意义的字符。

#### 4.2.1 加载停用词库

加载停用词库，在这里我使用Pandas库选择加载百度的停用词库，百度的停用词库如图所示：


加载停用词库的代码如图所示：

```
import pandas as pd
stopwords=pd.read_table('stopwords/baidu_stopwords.txt',
                        names=['words'],encoding='utf-8')
```

### 4.2.2 去停用词和去无意义的标点和字符

在这里我使用re模块来匹配去掉无意义符号，并分别在训练集和测试集下进行，代码如下所示：

```
import re
train_lst=[]
test_lst=[]
for sentence in twenty_train.data:
    out = re.sub(r'[^\w\s]','',sentence)#去掉符号
    words = [word for word in word_tokenize(out)
             if word not in list(stopwords['words'])]#去停用词
    lst=' '.join(words)
    train_lst.append(lst)

for sentence in twenty_test.data:
    out = re.sub(r'[^\w\s]','',sentence)#去掉符号
    words = [word for word in word_tokenize(out)
             if word not in list(stopwords['words'])]#去停用词
    lst=' '.join(words)
    test_lst.append(lst)
```

进行去停用词和去符号后，数据如下所示，可以看出，处理后比处理前的数据更加整齐。

```
['From sd345cityacuk Michael Collier Subject Converting images HP LaserJet III Nntp
PostingHost hampton Organization The City University Lines 14 Does a good standard
PC applicationPD utility convert tifimgtga files LaserJet III format We converting
HPGL HP plotter files Please email response Is correct group Thanks advance Michael
Michael Collier Programmer The Computer Unit Email MPCollierukaccity The City Unive
rsity Tel 071 4778000 x3769 London Fax 071 4778565 EC1V 0HB',
 'From animsukyedu Aniruddha B Deglurkar Subject Splitting a trimming region a mesh
Organization University Of Kentucky Dept Math Sciences Lines 28 Hi I a problem I ho
pe gurus solve Background problem I a rectangular mesh uv domain mesh a mapping a 3
d Bezier patch 2d The area domain inside a trimming loop rendered The trimming loop
a set 2d Bezier curve segments For sake notation mesh made cells My problem The tri
mming area split individual smaller cells bounded trimming curve segments If a cell
wholly inside areathen output a trivially rejected Does body thiss algo Any appreci
ated Thanks Ani To irritated human stay cool divine',
 'From djohnsoncsucsdedu Darin Johnson Subject Re harrassed work prayers Organizati
on CSE Dept UC San Diego Lines 63 Well Ill email apply people Ill post Ive working
company years engineering jobs Im female Yesterday I counted realized occasions Ive
sexually harrassed company I dreaded coming back work today What boss kind question
Your boss person bring problems If heshe action higher higher Sexual harrassment to
```

最后，我们将处理后的数据保存到本地，以便于后面进行加载，在这里我使用numpy模块中的savetxt函数，把数据以txt格式保存到本地，代码如下所示：

```
np.savetxt('words_news_train.txt',train_lst,fmt = '%s',encoding='utf-8',delimiter="",")
np.savetxt('words_news_test.txt',test_lst,fmt = '%s',encoding='utf-8',delimiter="",")
```

## 4.3 数据特征表示

在本次实验中，使用了两种方法进行数据特征表示，分别为：Word2Vec、BOW，Word2Vec考虑了上下文的语义，而BOW仅考虑词出现的权重，即词频。

### 4.3.1 使用Word2Vec进行特征表示

（1）加载初步处理后的数据集

分别读取训练数据和测试数据，读取数据代码如下：

```
with open('words_news_train.txt','r',encoding='utf-8') as f:
    data=f.readlines()
train_lst=[x.split() for x in train_lst]
with open('words_news_test.txt','r',encoding='utf-8') as f:
    test_lst=f.readlines()
test_lst=[x.split() for x in test_lst]
```

对训练数据和测试数据进行拼接，便于将Word转换成Vector

```
data=np.hstack((train_lst,test_lst))
```

（2）将Word转换成Vector

在这里，我使用gensim包中的Word2Vec函数将每个单词转换成向量，我设置vector_size=100，表示转换后的向量维度是100，并进行训练Word2Vec模型，同时把他保存到本地。代码如下所示：

```
import gensim
model=gensim.models.Word2Vec(data,sg=1,vector_size=100,
                             window=5,min_count=2,negative=3,
                              sample=0.001,hs=1,workers=4)

model.wv.save_word2vec_format("word2vec_model_news")
```

转换后，每个单词都表示成一个100维的向量，由于训练Word2Vec模型需要的时间较长，我把他保存到本地，后面直接加载模型，不用再训练，以节约时间，保存到本地后的模型如下所示，可以看出，每一个Word都表示成立一个(-1,+1)的向量：

```
a 0.053413805 -0.07320565 0.17762284 0.16722503 -0.01104569 -0.05786019 0.106959246 0.31263828 -0.058994092 -0.06564393 -(
I -0.017672839 -0.084464915 0.30494398 0.08918359 -0.00025737705 -0.31384277 0.1705954 0.33411524 -0.093506195 -0.2648057
The 0.1865195 -0.09260624 0.117180385 0.30306837 0.061658934 -0.029619088 0.36443076 0.2304577 -0.043698374 0.013103894 -(
From -0.5450499 -0.13613887 -0.72320664 0.5063916 -0.74545777 -0.23662983 -0.557002 0.19827703 -0.37181303 0.0072915847 0
Subject -0.48097134 -0.4227019 -0.64812076 0.74147475 -0.69876164 -0.25024426 -0.6391545 0.047732443 -0.73313516 0.249614
Lines -0.53465825 0.18698147 0.038461108 0.074372694 -0.5046065 -0.17019005 0.28133544 0.18954219 -0.33974308 -0.00947173
Organization -0.23819402 -0.20017722 -0.2835975 0.6489391 -0.6492073 0.025489887 0.05925367 0.13210474 -0.52376425 0.6087
In -0.0066147577 -0.19009665 -0.2199209 -0.026968269 -0.50561595 0.103127256 -0.22525983 0.3830554 0.014738854 -0.3000892
Re -0.6020807 -0.22174743 -0.62623817 0.92016506 -0.785342 -0.24012585 -0.7675532 0.05393868 -0.65915173 0.3743142 0.3123
writes -0.395611 -0.085336044 -0.68670326 -0.1512404 -0.5233444 -0.29212427 -0.7932336 0.3820883 -0.41130883 -0.48490793
God -0.13626978 0.214973 0.22909239 -0.0195536 0.19283077 -0.08580701 -0.18533531 0.34732607 -0.40335304 -0.16120219 -0.1
people -0.12865213 -0.14317343 0.29849264 -0.09995341 0.20860015 -0.17841092 0.050671317 0.35940087 -0.02380531 -0.011433
If 0.19352463 -0.24311805 0.2413259 0.2056831 0.14745161 -0.013347772 0.47220424 0.5834012 -0.06771184 -0.123552024 -0.17
article -0.42582673 -0.022791762 -0.41848627 -0.25393444 -0.8087407 -0.065421574 -0.4285398 0.26916215 -0.17139095 -0.506
dont 0.09770854 -0.052024495 0.29331374 -0.012445173 0.23385277 -0.21311283 0.12462201 0.34143552 -0.10561041 -0.25624922
It 0.14546582 -0.07220724 0.2058764 0.30767435 0.06773731 -0.11569689 0.20950642 0.37955174 -0.16257294 0.12173384 -0.114
This 0.1631253 -0.111145854 0.0438012 0.3808959 -0.07463843 -0.13644987 0.1814125 0.16613333 -0.11362711 0.059163727 -0.3
University 0.014703431 -0.051155373 0.19748107 0.30362234 -0.09444537 0.21045296 0.3920514 -0.14523235 -0.44187218 0.6697
A 0.17203537 -0.41161746 -0.10925877 0.4952868 0.32610443 -0.32070473 0.32510495 0.50112146 0.2464129 -0.13840158 -0.0884
time 0.2495605 -0.1687899 0.46597868 -0.021434814 -0.057957962 -0.33537716 -0.18973209 0.50491714 -0.0887099 0.034437448
Im -0.060240913 -0.1883546 0.07698275 0.32285628 -0.29445392 -0.5826545 -0.09919851 0.2991459 0.04113226 -0.23343243 -0.1
But 0.02354515 -0.10635812 0.34355566 0.22622588 0.042136848 -0.096647054 0.07296093 0.45769045 0.0058053294 -0.069188125
good 0.11140191 -0.09896823 0.034812156 0.16179852 0.06842892 -0.24039286 -0.14167047 0.23892646 -0.21893121 -0.26609135
```

（3）加载Word2Vec模型

在这里使用gensim.models中的KeyedVectors来加载训练好的Word2Vec模型，代码如下所示：

```
import numpy as np
import gensim
from gensim.models import KeyedVectors,word2vec,Word2Vec
model = KeyedVectors.load_word2vec_format('word2vec_model_news')#加载模型
```

（3）将一段文本转换成向量

将词向量转换成文本向量，由于每个文本的单词个数并不相同，所以我将每段文本中所有单词计算平均，并把它作为每段文本的特征表示，在这里，我分别将训练数据和测试数据进行文本特征表示，代码如下所示：

```
train_text_vec = np.zeros((len(train_lst), 100))
s=np.zeros(100)
for i,sentence in enumerate(train_lst):
    lenth=len(sentence)
    vec=0
    if lenth==0:
        train_text_vec[i]=0
        continue
    for word in sentence:
        try:
            vec=vec+model[word]
        except:
            pass
    train_text_vec[i]=vec/lenth
test_text_vec = np.zeros((len(test_lst), 100))
s=np.zeros(100)
for i,sentence in enumerate(test_lst):
    lenth=len(sentence)
    vec=0
    if lenth==0:
        test_text_vec[i]=0
        continue
    for word in sentence:
        try:
            vec=vec+model[word]
        except:
            pass
    test_text_vec[i]=vec/lenth
```

（4）数据标准化

要想调用 sklearn 库的模型，并进行训练，需要将数据的取值范围转换到0到1之间，由于将本文转换后的特征范围在-1到1之间，需要进行数据的标准化，在这里我使用最大最小值标准化，并使用sklearn.preprocessing中的MinMaxScaler函数实现，代码如下所示：

```
from sklearn import preprocessing

def min_max_normalization(np_array):
    min_max_scaler = preprocessing.MinMaxScaler()
    ret = min_max_scaler.fit_transform(np_array)
    return ret
x_train=min_max_normalization(x_train)
x_test=min_max_normalization(x_test)
```

标准化后的数据如下所示，可以看出，标准化后数据在0到1之间。

```
array([[0.60817738, 0.39235591, 0.55268781, ..., 0.48573955, 0.39232166,
        0.61539456],
       [0.62434657, 0.42826337, 0.58633492, ..., 0.51474397, 0.38371116,
        0.74043573],
       [0.66013533, 0.34253185, 0.77099887, ..., 0.6019036 , 0.52952493,
        0.62986023],
       ...,
       [0.56073889, 0.38629818, 0.62247827, ..., 0.59525892, 0.48934857,
        0.73732418],
       [0.62843586, 0.40097772, 0.60259401, ..., 0.64697877, 0.44545292,
        0.7610825 ],
       [0.15000476, 0.18923838, 0.49824716, ..., 0.82904627, 0.63884693,
        0.69283631]])
```

### 4.3.2 使用BOW词袋模型进行向量化

在把词转换为向量时，通过统计每个词在文本中出现的次数就可以得到该文本基于词的特征，再将各个文本的词和对应的词频放在一起。

（1）统计词频

在词袋模型的统计词频这一步，我们会得到该文本所有词的词频，有了词频，我们就能用词向量来表示该文本了。在这里我使用sklearn中的CountVectorizer来统计词频，代码如下所示：

```
from sklearn.feature_extraction.text import CountVectorizer
# 在测试集上的性能评估
import numpy as np
Counter = CountVectorizer(stop_words='english')
Counter.fit(twenty_train.data)
Counter.fit(twenty_test.data)
x_train_count=Counter.transform(twenty_train.data)
x_test_count=Counter.transform(twenty_test.data)
```

统计结果如下所示，可以看出每段文本中所有词的词频。

```
(0, 119)      2
(0, 423)      1
(0, 1767)     2
(0, 2352)     1
(0, 3121)     2
(0, 3417)     1
(0, 4224)     1
(0, 7156)     4
(0, 7457)     3
(0, 7718)     1
(0, 8124)     1
(0, 8128)     2
(0, 8241)     1
(0, 10092)    1
(0, 10791)    2
(0, 11873)    1
(0, 12038)    2
(0, 12369)    1
(0, 13170)    1
(0, 13417)    1
(0, 13651)    1
(0, 14342)    1
(0, 14376)    2
(0, 14383)    1
(0, 14709)    2
```

（2）TF-IDF特征表示

将词频转换成特征，在这里，使用sklearn中的TfidfTransformer()来将词频转换成特征，转换代码如下所示：

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidfer = TfidfTransformer()
tfidfer.fit(x_train_count)
tfidfer.fit(x_test_count)
x_train=tfidfer.transform(x_train_count)
x_test=tfidfer.transform(x_test_count)
```

转换结果如下所示，可以看出，文本中所有词都转换成了0到1之间的浮点型的值，每段文本都转换成了向量。

```
  (0, 29646)    0.04728728123850258
  (0, 28947)    0.10320987558561678
  (0, 28696)    0.07111316200246669
  (0, 28673)    0.09534201036277591
  (0, 28409)    0.12499073487561276
  (0, 27641)    0.1200758316716855
  (0, 27426)    0.05386091919421854
  (0, 27406)    0.09985392396080776
  (0, 27244)    0.09885872085777618
  (0, 26527)    0.01840677363566864
  (0, 26160)    0.07383239074753992
  (0, 23790)    0.07593831618324601
  (0, 22290)    0.09791457679895682
  (0, 21854)    0.03797896306193983
  (0, 21607)    0.14029773938256326
  (0, 20983)    0.11261252720957661
  (0, 20965)    0.07769266908807906
  (0, 20342)    0.01997833681952289
  (0, 19659)    0.03964839413846657
  (0, 18423)    0.21582857691298465
  (0, 17358)    0.10731722842291665
  (0, 17196)    0.018431283324557245
  (0, 17149)    0.03833708279461463
  (0, 16814)    0.2805954787651265
  (0, 16549)    0.03797896306193983
```

4.4 模型构建和分类

使用skleran中的分类器来训练数据并对测试数据进行分类，并评估分类结果。

4.4.1使用朴素贝叶斯分类器

使用sklearn中的MultinomialNB分类器进行分类，并评估分类结果，代码如下所示：

```
# 朴素贝叶斯分类
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
text_clf =MultinomialNB()
text_clf.fit(x_train, twenty_train.target)
predicted = text_clf.predict(x_test)
print(classification_report(twenty_test.target, predicted, target_names=categories))
acc_score1=accuracy_score(predicted,twenty_test.target)
recall_score1=recall_score(predicted,twenty_test.target,average='macro')
f1_score1=f1_score(predicted,twenty_test.target,average='macro')
pre_score1=precision_score(predicted,twenty_test.target,average='macro')
```

分类结果如下：

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.96 | 0.72 | 0.82 | 319 |
| soc.religion.christian | 0.94 | 0.96 | 0.95 | 389 |
| comp.graphics | 0.95 | 0.88 | 0.91 | 396 |
| sci.med | 0.78 | 0.97 | 0.87 | 398 |
| accuracy | | | 0.89 | 1502 |
| macro avg | 0.91 | 0.88 | 0.89 | 1502 |
| weighted avg | 0.90 | 0.89 | 0.89 | 1502 |

### 4.4.2 使用随机梯度下降分类器

使用sklearn中的SGDClassifier分类器进行分类，并评估分类结果，代码如下所示：

```
# 随机梯度下降分类器
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import classification_report
text_clf =SGDClassifier(loss='hinge',penalty='l2',
                        alpha=1e-3, random_state=42,
                         max_iter=5, tol=None)
text_clf.fit(x_train, twenty_train.target)
predicted = text_clf.predict(x_test)
print(classification_report(twenty_test.target, predicted, target_names=categories))
acc_score2=accuracy_score(predicted,twenty_test.target)
recall_score2=recall_score(predicted,twenty_test.target,average='macro')
f1_score2=f1_score(predicted,twenty_test.target,average='macro')
pre_score2=precision_score(predicted,twenty_test.target,average='macro')
```

分类结果如下：

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.96 | 0.82 | 0.88 | 319 |
| soc.religion.christian | 0.91 | 0.97 | 0.94 | 389 |
| comp.graphics | 0.95 | 0.92 | 0.93 | 396 |
| sci.med | 0.89 | 0.96 | 0.92 | 398 |
| accuracy | | | 0.92 | 1502 |
| macro avg | 0.93 | 0.92 | 0.92 | 1502 |
| weighted avg | 0.92 | 0.92 | 0.92 | 1502 |

### 4.4.3 使用KNN分类器

使用sklearn中的KNeighborsClassifier分类器进行分类，并评估分类结果，代码如下所示：

```
# KNN分类器
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
text_clf =KNeighborsClassifier()
text_clf.fit(x_train, twenty_train.target)
predicted = text_clf.predict(x_test)
print(classification_report(twenty_test.target, predicted, target_names=categories))
acc_score3=accuracy_score(predicted,twenty_test.target)
recall_score3=recall_score(predicted,twenty_test.target,average='macro')
f1_score3=f1_score(predicted,twenty_test.target,average='macro')
pre_score3=precision_score(predicted,twenty_test.target,average='macro')
```

分类结果如下：

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.79 | 0.86 | 0.83 | 319 |
| soc.religion.christian | 0.85 | 0.88 | 0.86 | 389 |
| comp.graphics | 0.84 | 0.75 | 0.80 | 396 |
| sci.med | 0.84 | 0.85 | 0.85 | 398 |
| accuracy | | | 0.83 | 1502 |
| macro avg | 0.83 | 0.84 | 0.83 | 1502 |
| weighted avg | 0.84 | 0.83 | 0.83 | 1502 |

### 4.4.4 使用SVM分类器

使用sklearn中的SVC分类器进行分类，并评估分类结果，代码如下所示：

```
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.svm import SVC
text_clf =SVC(decision_function_shape = 'ovo')
text_clf.fit(x_train, twenty_train.target)
predicted = text_clf.predict(x_test)
print(classification_report(twenty_test.target, predicted, target_names=categories))
acc_score4=accuracy_score(predicted,twenty_test.target)
recall_score4=recall_score(predicted,twenty_test.target,average='macro')
f1_score4=f1_score(predicted,twenty_test.target,average='macro')
pre_score4=precision_score(predicted,twenty_test.target,average='macro')
```

分类结果如下：

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.97 | 0.74 | 0.84 | 319 |
| soc.religion.christian | 0.86 | 0.98 | 0.92 | 389 |
| comp.graphics | 0.93 | 0.90 | 0.92 | 396 |
| sci.med | 0.88 | 0.96 | 0.92 | 398 |
| accuracy | | | 0.90 | 1502 |
| macro avg | 0.91 | 0.89 | 0.90 | 1502 |
| weighted avg | 0.91 | 0.90 | 0.90 | 1502 |

## 五 实验分析

### 5.1 各个模型的分类结果对比

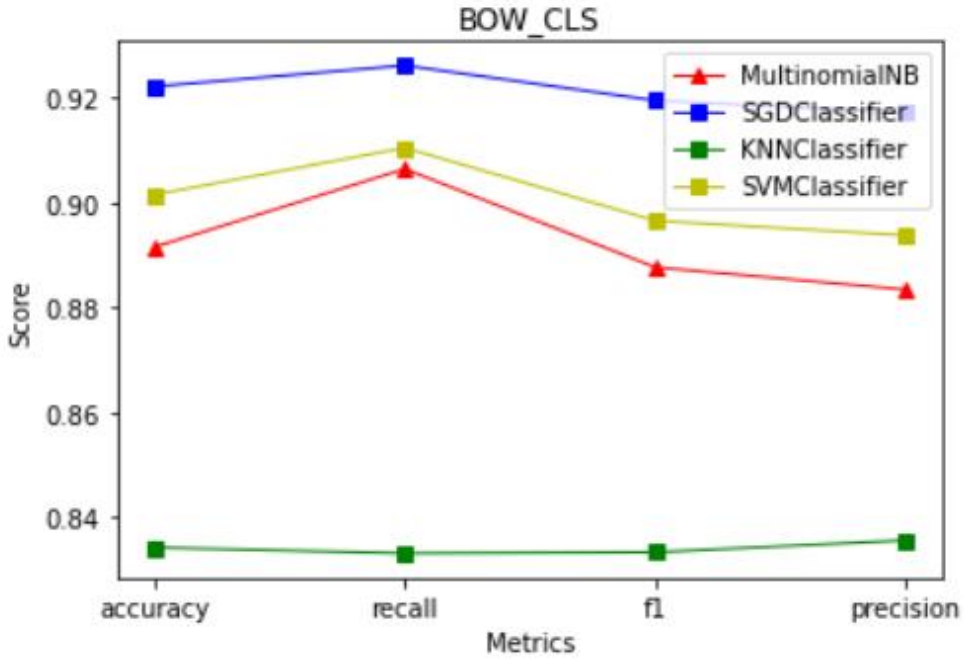
在本次实验中，使用了两种方法进行文本特征表示，分别为BOW、Word2Vec这两种方式，对应这两种文本特征表示方式对应的各个模型的分类结果也不相同，于是，我分别将这两种文本特征表示方式的各个模型的分类结果评估进行对比。

#### 5.5.1 基于BOW文本表示方式的分类结果

我使用了matplotlib.pyplot中的plt模块来对分类结果进行可视化，代码如下所示：

```
import matplotlib.pyplot as plt
import numpy as np

Mul_score=[acc_score1,recall_score1,f1_score1,pre_score1]
SGD_score=[acc_score2,recall_score2,f1_score2,pre_score2]
KNN_score=[acc_score3,recall_score3,f1_score3,pre_score3]
SVM_score=[acc_score4,recall_score4,f1_score4,pre_score4]

score=['accuracy','recall','f1','precision']
plt.plot(score, Mul_score, color="r", linestyle="-",
         marker="^", linewidth=1, label="MultinomialNB")
plt.plot(score, SGD_score, color="b", linestyle="-",
         marker="s", linewidth=1, label="SGDClassifier")
plt.plot(score, KNN_score, color="g", linestyle="-",
         marker="s", linewidth=1, label="KNNClassifier")
plt.plot(score, SVM_score, color="y", linestyle="-",
         marker="s", linewidth=1, label="SVMClassifier")
plt.legend()
plt.ylabel("Score", loc='center')
plt.xlabel("Metrics", loc='center')
plt.title("基于BOW文本表示方式的分类结果")

plt.show()
```

可视化结果如图所示：



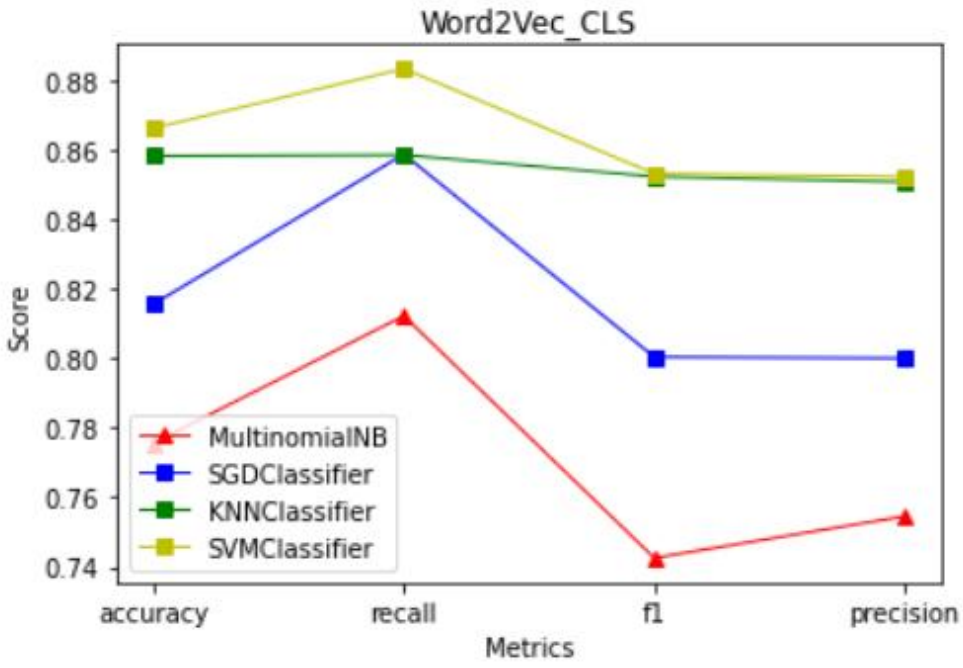

可以看出在BOW下,随机梯度下降分类器SGDClassifier的分类效果最好。

### 5.5.2 基于Word2Vec文本表示方式的分类结果

我使用了matplotlib.pyplot中的plt模块来对分类结果进行可视化，代码如下所示：

```
import matplotlib.pyplot as plt
import numpy as np

Mul_score=[acc_score1,recall_score1,f1_score1,pre_score1]
SGD_score=[acc_score2,recall_score2,f1_score2,pre_score2]
KNN_score=[acc_score3,recall_score3,f1_score3,pre_score3]
SVM_score=[acc_score4,recall_score4,f1_score4,pre_score4]
score=['accuracy','recall','f1','precision']
plt.plot(score, Mul_score, color="r", linestyle="-",
         marker="^", linewidth=1, label="MultinomialNB")
plt.plot(score, SGD_score, color="b", linestyle="-",
         marker="s", linewidth=1, label="SGDClassifier")
plt.plot(score, KNN_score, color="g", linestyle="-",
         marker="s", linewidth=1, label="KNNClassifier")
plt.plot(score, SVM_score, color="y", linestyle="-",
         marker="s", linewidth=1, label="SVMClassifier")
plt.legend()
plt.ylabel("Score", loc='center')
plt.xlabel("Metrics", loc='center')
plt.title("World2Vec_CLS")

plt.show()
```

可视化结果如图所示：




可以看出，在Word2Vec文本表示方式下，SVMClassifier的分类效果最好。

总体来看，基于BOW文本特征表示方式的分类效果比基于Word2Vec文本特征表示方式的分类效果好。

综上所述，使用BOW文本特征表示方式的SGDClassifier分类器分类效果最好。

## 5.6 探讨关键模型性能

在上述所有模型中，SGDClassifier模型的表现最好，下面我将探讨SGDClassifier中的参数Alpha对分类结果的影响,参数Alpha是与正则项相乘的常数，值越高，正则化越强，我将设置Alpha不同的取值进行实验，实验20次，实验代码如下：

```
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
from sklearn.linear_model import SGDClassifier
acc_score=[]
re_score=[]
f_score=[]
pre_score=[]
Alpha=[]
for i in range(20):
    alpha=0.000001*(i+1)*2
    Alpha.append(alpha)
    text_clf =SGDClassifier(loss='hinge',penalty='l2',
                            alpha=alpha, random_state=42,
                             max_iter=1000, tol=None)
    text_clf.fit(x_train, twenty_train.target)
    predicted = text_clf.predict(x_test)
    acc_score.append(accuracy_score(predicted,twenty_test.target))
    re_score.append(recall_score(predicted,twenty_test.target,average='macro'))
    f_score.append(f1_score(predicted,twenty_test.target,average='macro'))
    pre_score.append(precision_score(predicted,twenty_test.target,average='macro'))
```

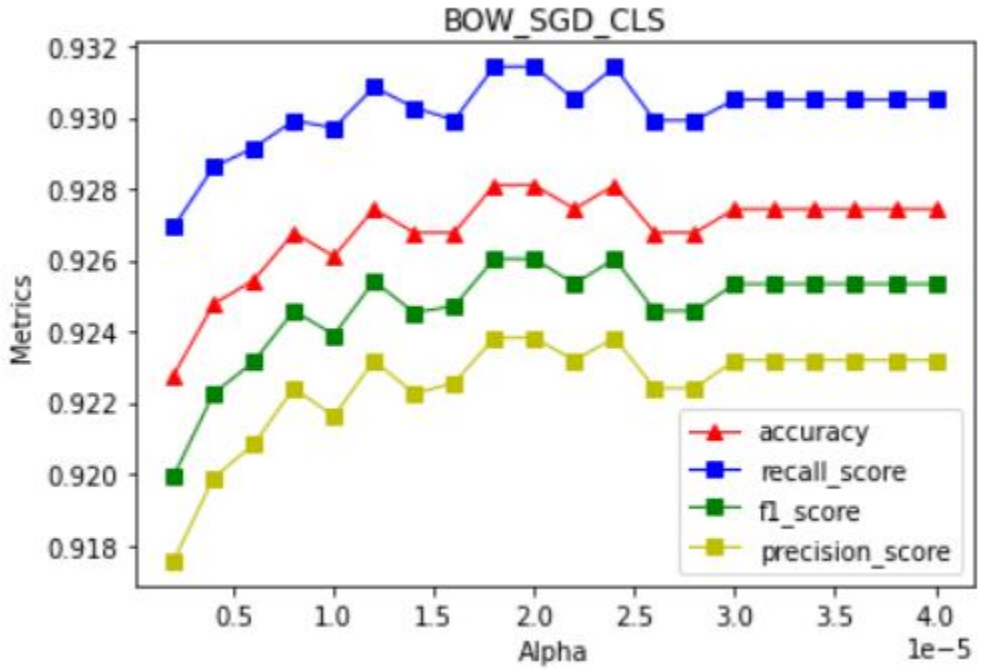
Alpha取值如下，可以看出Alpha取了20组不同的值。

```
[2e-06,
 4e-06,
 6e-06,
 8e-06,
 9.999999999999999e-06,
 1.2e-05,
 1.4e-05,
 1.6e-05,
 1.8e-05,
 1.9999999999999998e-05,
 2.2e-05,
 2.4e-05,
 2.6e-05,
 2.8e-05,
 2.9999999999999997e-05,
 3.2e-05,
 3.4e-05,
 3.6e-05,
 3.7999999999999995e-05,
 3.9999999999999996e-05]
```

我将Alpha不同的取值对应的模型分类结果通过可视化表示出来，可视化代码如下所示：

```
import matplotlib.pyplot as plt
import numpy as np


plt.plot(Alpha, acc_score, color="r", linestyle="-",
         marker="^", linewidth=1, label="accuracy")
plt.plot(Alpha, re_score, color="b", linestyle="-",
         marker="s", linewidth=1, label="recall_score")
plt.plot(Alpha, f_score, color="g", linestyle="-",
         marker="s", linewidth=1, label="f1_score")
plt.plot(Alpha, pre_score, color="y", linestyle="-",
         marker="s", linewidth=1, label="precision_score")
plt.legend()
plt.ylabel("Metrics", loc='center')
plt.xlabel("Alpha", loc='center')
plt.title("BOW_SGD_CLS")

plt.show()
```

可视化结果如下：




从上图可以看出，随着Alpha值的增大，分类效果先上升，然后再下降，当Alpha取值为2.4e-05时，分类效果最好。

5.7 数据预处理性能分析

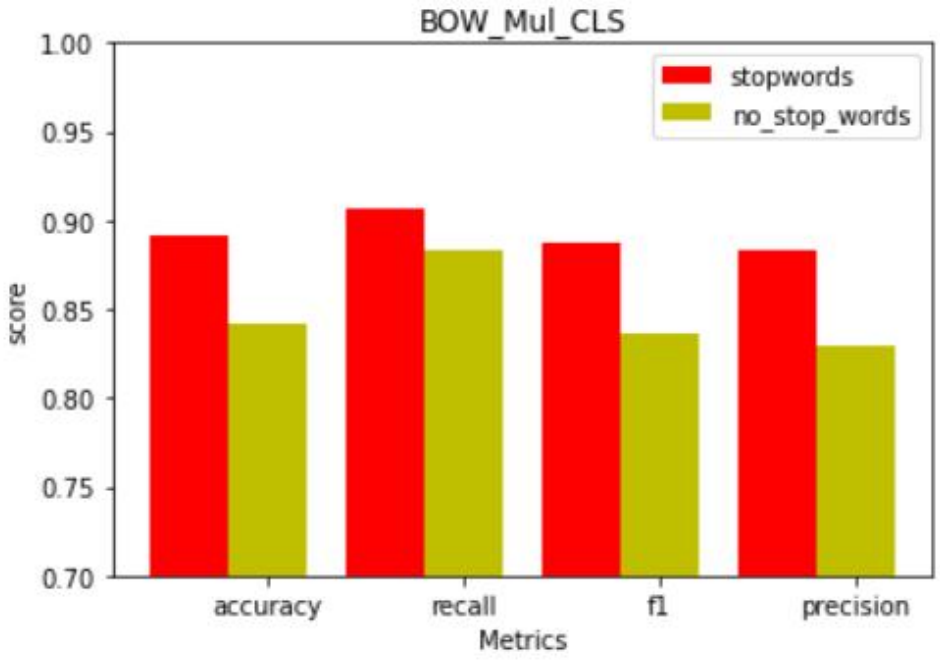
在本次实验中，在数据预处理的部分采用了去停用词的数据预处理的方法，为了验证去停用词对模型分类结果的影响，在这里，我将进行实验，分别进行去停用词和不去停用词来使用模型进行分类，并且使用基于BOW的文本表示方法来验证。

通过改变sklearn中CountVectorizer中的参数stop_words=None或者stop_words='english'来设置停用词，并将结果可视化出来，可视化代码如下所示：

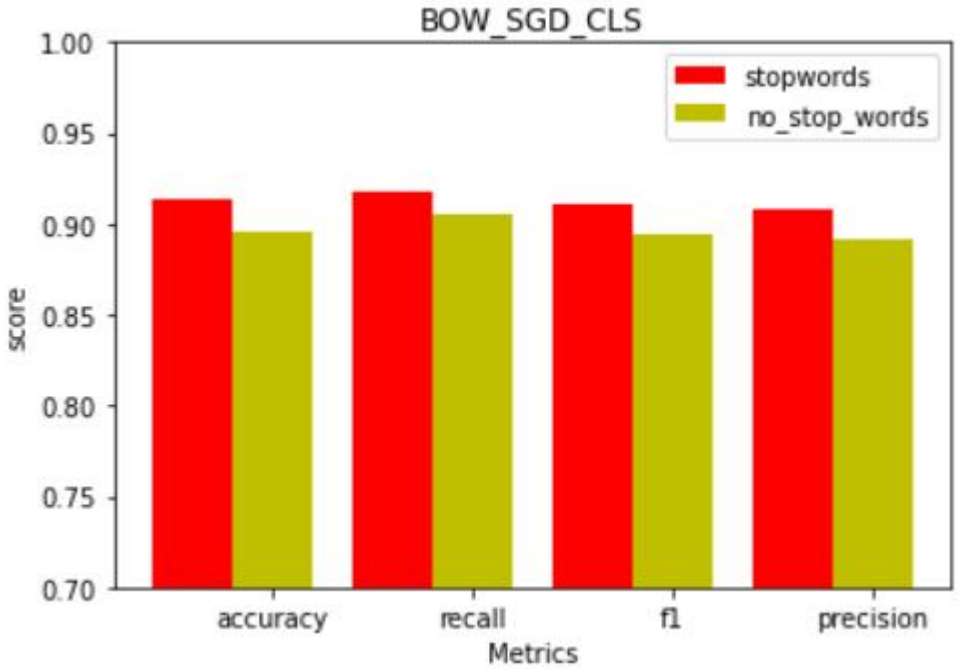
```
Mul_score_1=[acc_score1_1,recall_score1_1,f1_score1_1,pre_score1_1]
SGD_score_1=[acc_score2_1,recall_score2_1,f1_score2_1,pre_score2_1]
KNN_score_1=[acc_score3_1,recall_score3_1,f1_score3_1,pre_score3_1]
SVM_score_1=[acc_score4_1,recall_score4_1,f1_score4_1,pre_score4_1]

import matplotlib.pyplot as plt

name_list = ['accuracy','recall','f1','precision']
x =list(range(len(num_list)))
total_width, n = 0.8, 2
width = total_width / n
plt.title("BOW_Mul_CLS")
plt.bar(x, Mul_score, width=width, label='stopwords',fc = 'r')
for i in range(len(x)):
    x[i] = x[i] + width
plt.bar(x, Mul_score_1, width=width, label='no_stop_words',
        tick_label = name_list,fc = 'y')

plt.ylim(0.7,1)
plt.xlabel("Metrics", loc='center')
plt.ylabel("score", loc='center')
plt.legend()
plt.show()
```

可视化结果：

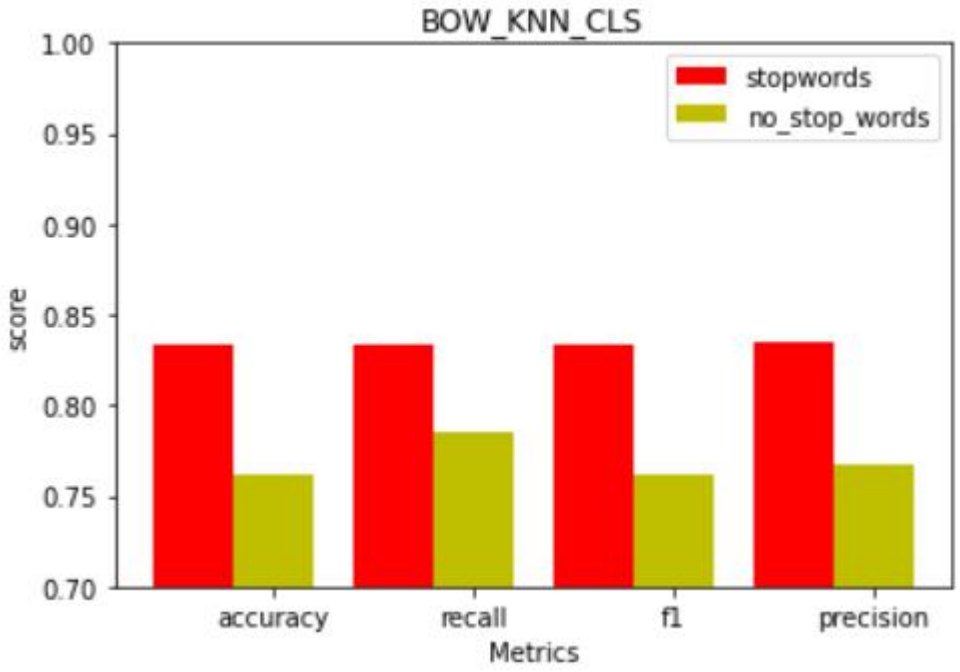
(1) MultinomialNB



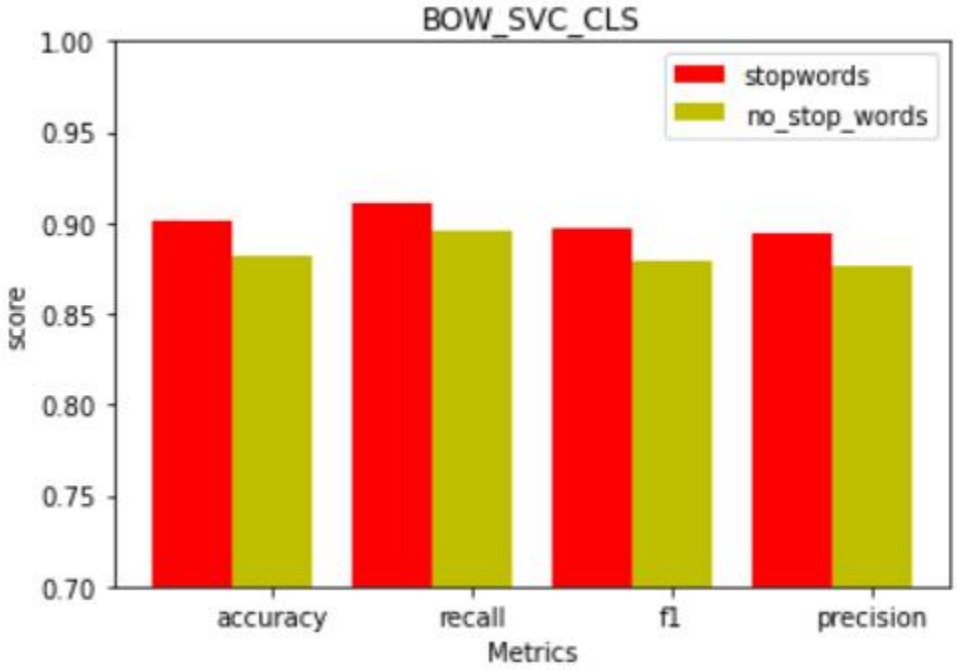
(2) SGDClassifier




(3) KNeighborsClassifier




(4) SVC




从上述四个模型的结果可以看出，去停用词比不去停用词的模型分类结果好。

六 实验总结

通过这次实验，我熟悉了 20news-bydate这个数据集，并且学会了如何进行数据预处理、文本向量化、模型构建、模型评估、模型分析。

在数据预处理中，我使用了去停用词和去无意义的符号两种方法。在文本向量化中，我也使用了两种方法，分别是BOW、Word2Vec，这两种方法都能将文本转换成特征向量，但是转换的原理不一样，最后对于模型的分类的影响也不一样。

在模型构建中，我使用了四种分类器，分别为：朴素贝叶斯分类器、随机梯度下降分类器、KNN分类器、SVM分类器，这四种分类器在不同文本表示方法中表现的性能也不一样。在模型构建完之后，需要对模型进行评估，在本实验中我使用了四种评估指标，分别为:accuracy、recall_score、f1_score、precision;用这四种评估指标来评估模型表现的好坏，同时也验证了不同模型在BOW和Word2Vec这两种表示方法的性能也不一致。最后，我对实验的结果进行了分析，并对表现最好的模型的参数进行了分析，通过多次实验，找到了时模型分类效果最好的该模型参数的值，并且，我对数据预处理也进行了分析，对是否使用停用

词对模型分类结果的影响进行的分析，结果表明去停用词对模型性能的增加有一定的帮助。

通过本次实验，我学会了如何使用sklearn，并且使用sklearn中的一些模块来帮助我更好的完成实验，是我对Python更加的熟悉，并且对文本分类的步骤有了更好的理解，同时对数据挖掘课程的内容有了一个很好的巩固，我在本次实验中学到了很多知识。

七 验收过程老师提出的问题与解答

（1）Word2Vec和TFIDF有有什么区别？

答：Word2Vec是基于上下文语义的，而TFIDF是基于词频的，Word2Vec把上下文的语义作为特征把词转换成向量，而TFIDF是把词在文中出现的次数作为词的特征转换成向量。

（2）在本次实验中，Word2Vec与TFIDF相比哪个更好？

答：在本次实验中TFIDF在各个模型上的表现总体上比Word2Vec要好。

（3）在本次实验中使用了哪几种模型，表现的性能如何？

答：在本次实验中，我使用了朴素贝叶斯、随机梯度下降、KNN、SVM这四种模型，在Word2Vec下，SVM表现比其他模型好，在BOW下，随机梯度下降模型表现比其他模型好，两种文本特征提取中，表现最好的是BOW文本特征提取下的随机梯度下降模型。