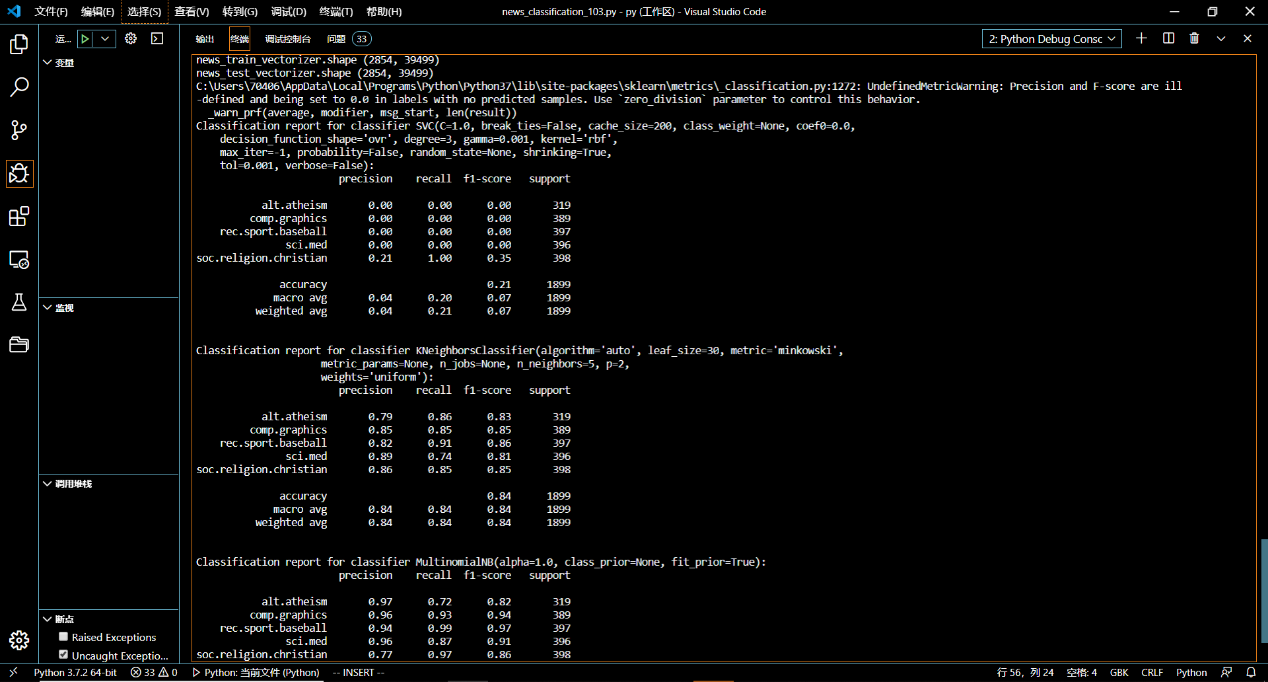
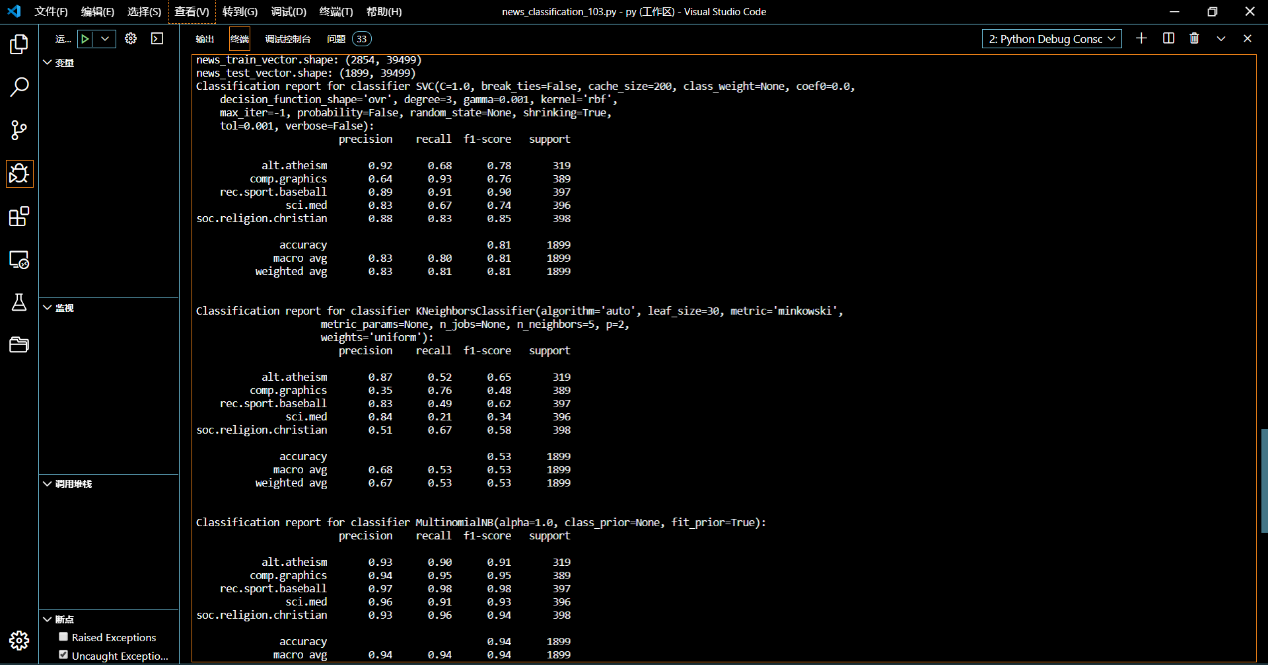
**REPORT**

**031804103 陈翰泽**

**相关代码与文件已上传至GitHub：**<https://github.com/Holmze/SklearnClassification>

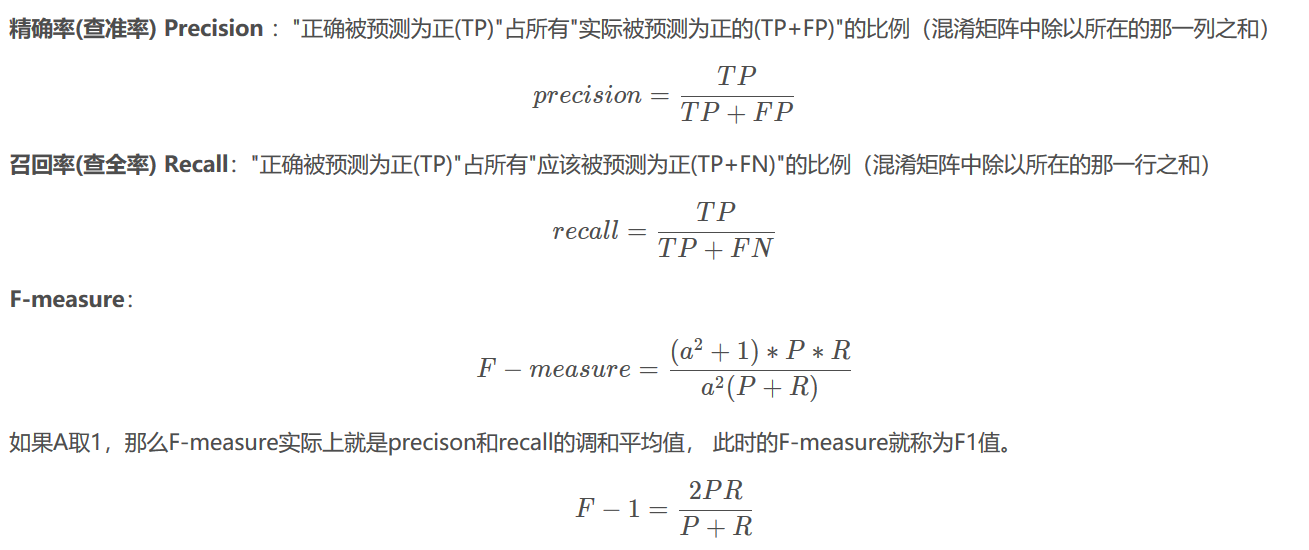
实验结果：



代码

1. **from** sklearn.datasets **import** fetch\_20newsgroups
2. # sample\_cate 指定需要下载哪几个主题类别的新闻数据
3. sample\_cate = ['alt.atheism', 'soc.religion.christian', 'comp.graphics', 'sci.med', 'rec.sport.baseball']
4. # 需要从网络上下载，受连接外网速度限制, 可能要耐心等待几分钟时间
5. newsgroups\_train = fetch\_20newsgroups(subset='train', categories=sample\_cate, shuffle=True)
6. # 以上得到训练集，以下代码得到测试集
7. newsgroups\_test = fetch\_20newsgroups(subset='test', categories=sample\_cate, shuffle=True)
9. **print**(len(newsgroups\_train.data))
10. **print**(len(newsgroups\_test.data))
12. **import** matplotlib.pyplot as plt
13. **from** sklearn **import** datasets, svm, metrics
14. **from** sklearn.neighbors **import** KNeighborsClassifier
15. **from** sklearn.model\_selection **import** train\_test\_split
16. **from** sklearn.naive\_bayes **import** MultinomialNB
17. **from** sklearn.feature\_extraction.text **import** CountVectorizer
18. **from** sklearn.feature\_extraction.text **import** TfidfVectorizer
20. count\_vectorizer = CountVectorizer(stop\_words='english')
21. cv\_news\_train\_vector = count\_vectorizer.fit\_transform(newsgroups\_train.data)
22. **print**("news\_train\_vector.shape:",cv\_news\_train\_vector.shape)
23. cv\_news\_test\_vector = count\_vectorizer.transform(newsgroups\_test.data)
24. **print**("news\_test\_vector.shape:",cv\_news\_test\_vector.shape)
26. svc\_clf = svm.SVC(gamma=0.001)
27. svc\_clf.fit(cv\_news\_train\_vector,newsgroups\_train.target)
28. cv\_svc\_predict = svc\_clf.predict(cv\_news\_test\_vector)
29. **print**("Classification report for classifier %s:\n%s\n" % (svc\_clf,metrics.classification\_report(newsgroups\_test.target,cv\_svc\_predict,target\_names=newsgroups\_test.target\_names)))
31. knn\_clf = KNeighborsClassifier()
32. knn\_clf.fit(cv\_news\_train\_vector,newsgroups\_train.target)
33. cv\_knn\_predict = knn\_clf.predict(cv\_news\_test\_vector)
34. **print**("Classification report for classifier %s:\n%s\n" % (knn\_clf,metrics.classification\_report(newsgroups\_test.target,cv\_knn\_predict,target\_names=newsgroups\_test.target\_names)))
36. nb\_clf = MultinomialNB()
37. nb\_clf.fit(cv\_news\_train\_vector,newsgroups\_train.target)
38. cv\_nb\_predict = nb\_clf.predict(cv\_news\_test\_vector)
39. **print**("Classification report for classifier %s:\n%s\n" % (nb\_clf,metrics.classification\_report(newsgroups\_test.target,cv\_nb\_predict,target\_names=newsgroups\_test.target\_names)))
41. ###################################################################
43. tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')
44. tfidf\_news\_train\_vector = tfidf\_vectorizer.fit\_transform(newsgroups\_train.data)
45. **print**("news\_train\_vectorizer.shape",tfidf\_news\_train\_vector.shape)
46. tfidf\_news\_test\_vector = tfidf\_vectorizer.transform(newsgroups\_test.data)
47. **print**("news\_test\_vectorizer.shape",tfidf\_news\_train\_vector.shape)
49. svc\_clf2 = svm.SVC(gamma=0.0001)
50. svc\_clf2.fit(tfidf\_news\_train\_vector,newsgroups\_train.target)
51. tfidf\_svc\_predict2 = svc\_clf2.predict(tfidf\_news\_test\_vector)
52. **print**("Classification report for classifier %s:\n%s\n" % (svc\_clf2,metrics.classification\_report(newsgroups\_test.target,tfidf\_svc\_predict2,target\_names=newsgroups\_test.target\_names)))
54. knn\_clf = KNeighborsClassifier()
55. knn\_clf.fit(tfidf\_news\_train\_vector,newsgroups\_train.target)
56. tfidf\_knn\_predict = knn\_clf.predict(tfidf\_news\_test\_vector)
57. **print**("Classification report for classifier %s:\n%s\n" % (knn\_clf,metrics.classification\_report(newsgroups\_test.target,tfidf\_knn\_predict,target\_names=newsgroups\_test.target\_names)))
59. nb\_clf = MultinomialNB()
60. nb\_clf.fit(tfidf\_news\_train\_vector,newsgroups\_train.target)
61. tfidf\_nb\_predict = nb\_clf.predict(tfidf\_news\_test\_vector)
62. **print**("Classification report for classifier %s:\n%s\n" % (nb\_clf,metrics.classification\_report(newsgroups\_test.target,tfidf\_nb\_predict,target\_names=newsgroups\_test.target\_names)))

3.1



TN：算法预测为负例（N），实际上也是负例（N）的个数，即算法预测对了（True）；  
FP：算法预测为正例（P），实际上是负例（N）的个数，即算法预测错了（False）；  
FN：算法预测为负例（N），实际上是正例（P）的个数，即算法预测错了（False）；  
TP：算法预测为正例（P），实际上也是正例（P）的个数，即算法预测对了（True）。

3.2

宏平均 Macro-average

Macro F1：将n分类的评价拆成n个二分类的评价，计算每个二分类的F1 score，n个F1 score的平均值即为Macro F1。

微平均Micro-average

Micro F1：将n分类的评价拆成n个二分类的评价，将n个二分类评价的TP、FP、TN、FN对应相加，计算评价准确率和召回率，由这2个准确率和召回率计算的F1 score即为Micro F1。

(TP + FP) / (TP + TN + FP + FN)，实际上就是accuracy，分母就是输入分类器的预测样本个数，分子就是预测正确的样本个数（无论类别）。

一般来讲，Macro F1、Micro F1高的分类效果好。Macro F1受样本数量少的类别影响大。

宏平均比微平均更合理，但也不是说微平均一无是处，具体使用哪种评测机制，还是要取决于数据集中样本分布。

2.1

通常，在训练有监督的机器学习模型的时候，会将数据划分为训练集、验证集和测试集，**划分比例一般为0.6 : 0.2 : 0.2。**对原始数据进行三个集合的划分，是为了能够选出效果（可以理解为准确率）最好的、泛化能力最佳的模型。

* **训练集（Training set）**

作用是用来拟合模型，通过设置分类器的参数，训练分类模型。后续结合验证集作用时，会选出同一参数的不同取值，拟合出多个分类器。

* **验证集(Cross Validation set)**

作用是当通过训练集训练出多个模型后，为了能找出效果最佳的模型，使用各个模型对验证集数据进行预测，并记录模型准确率。选出效果最佳的模型所对应的参数，即用来调整模型参数。如svm中的参数c和核函数等。

* **测试集(Test set)**

通过训练集和验证集得出最优模型后，使用测试集进行模型预测。用来衡量该最优模型的性能和分类能力。即可以把测试集当做从来不存在的数据集，当已经确定模型参数后，使用测试集进行模型性能评价。

对原始数据进行三个数据集的划分，也是为了防止模型过拟合。当使用了所有的原始数据去训练模型，得到的结果很可能是该模型最大程度地拟合了原始数据，亦即该模型是为了拟合所有原始数据而存在。当新的样本出现，再使用该模型进行预测，效果可能还不如只使用一部分数据训练的模型.

2.2

- One-Hot:是一种将非数值型的特征值(或称为属性)转换为数值型的数据的编码方法。一般是将类别数据编码成为对应的数值数据以供后续的算法使用。使用哑编码保证了两两类别(假设类别间相互独立)间的空间距离是相等的，这样避免了人为引入额外的类别差异性，进而有利于后续(比如loss函数)的计算。

- 词袋法(Bag of word，简称BOW)：该模型忽略了文本的语法和语序，用一组无序的单词(words)来表示一段文字或一个文档(我们也可以认为一段文字也是一个文档，下文均用文档来表示)，词袋法的核心是使用单词在文档中出现的次数(频数)来表示文档。

- 词集法(Set of word，简称SOW)：是词袋法的的一个变种，和词袋法原理一样，也是以文档中的单词来表示文档的一种模型，区别在于：词袋法使用的是单词的频数，而在词集法中使用的是单词是否出现，出现赋值为1，否则赋值为0

- 词频–逆文档频率(TF-IDF):在词袋法或词集法中，使用的是单词的词频或者是否存在来进行文档的表征，但是不同的单词在不同的文档中出现的次数不同，而且有些单词仅仅在某一些文档中出现(比如专业名词等)，也就是说不同单词对于文本而言具有不同的重要性，那么，如何评估一个单词对于一个文本的重要性呢？在讨论之前，我们有如下的假设：

1. 单词的重要性随着它在文本中出现的次数成正比，也就是单词出现的次数越多，该单词对于文档越重要。
2. 2，与此同时，单词的重要性会随着在语料库中出现的频率成反比下降，也就是单词在语料库中出现的频率越高，表示该单词越常见，也就是该单词对于特定文本的重要性越低。

* TF的意思是词频(Item Frequency)
* IDF的意思是逆向文件频率(Inverse Document Frequency)。

2.3

**高斯模型：用来处理连续型特征变量的，当使用此模型时，我们会假定特征属于高斯分布，然后基于训练样本计算特征均值和标准差，这样就可以得到此特征下每一个属性值的先验概率。**

多项式模型：高斯分布相反，多项式模型主要适用于离散特征的先验概率计算，即**贝叶斯定理求条件概率 + 拉普拉斯平滑**

**伯努利模型：**伯努利模型对应的是伯努利分布，是一种只有“是"或"否"两种结果的随机变量分布，比如抛硬币的正或反就是非常典型的伯努利分布。**在伯努利模型中，所有特征的取值都会变成0或1，如果特征本身并不是0或1，则会自动设定某个阈值，将低于阈值地设为0，高于阈值地设为1，从而将特征0-1化。然后在所有训练样本中计算0-1的概率，作为特征的条件概率**

2.6

利用机器学习算法进行回归、分类或者聚类时，评价指标，即检验机器学习模型效果的定量指标，metrics函数的作用就是这个

classification\_report函数用于显示主要分类指标的文本报告．在报告中显示每个类的精确度，召回率，F1值等信息。