* [任务描述](https://www.educoder.net/tasks/8mks3ofebnzw#%E4%BB%BB%E5%8A%A1%E6%8F%8F%E8%BF%B0)
* [相关知识](https://www.educoder.net/tasks/8mks3ofebnzw#%E7%9B%B8%E5%85%B3%E7%9F%A5%E8%AF%86)
  + [数据简介](https://www.educoder.net/tasks/8mks3ofebnzw#%E6%95%B0%E6%8D%AE%E7%AE%80%E4%BB%8B)
  + [文本向量化](https://www.educoder.net/tasks/8mks3ofebnzw#%E6%96%87%E6%9C%AC%E5%90%91%E9%87%8F%E5%8C%96)
  + [MultinomialNB](https://www.educoder.net/tasks/8mks3ofebnzw#MultinomialNB)
* [编程要求](https://www.educoder.net/tasks/8mks3ofebnzw#%E7%BC%96%E7%A8%8B%E8%A6%81%E6%B1%82)
* [测试说明](https://www.educoder.net/tasks/8mks3ofebnzw#%E6%B5%8B%E8%AF%95%E8%AF%B4%E6%98%8E)

**任务描述**

本关任务：使用sklearn完成新闻文本主题分类任务。

**相关知识**

为了完成本关任务，你需要掌握如何使用sklearn提供的MultinomialNB类与文本向量化。

**数据简介**

本关使用的是20newsgroups数据集，20newsgroups数据集是用于文本分类、文本挖据和信息检索研究的国际标准数据集之一。数据集收集了18846篇新闻组文档，均匀分为20个不同主题（比如电脑硬件、中东等主题）的新闻组集合。

部分数据如下：

1. From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>
2. Subject: Pens fans reactions
3. Organization: Post Office, Carnegie Mellon, Pittsburgh, PA
4. Lines: 12
5. NNTP-Posting-Host: po4.andrew.cmu.edu
6. I am sure some bashers of Pens fans are pretty confused about the lack
7. of any kind of posts about the recent Pens massacre of the Devils. Actually,
8. I am bit puzzled too and a bit relieved. However, I am going to put an end
9. to non-PIttsburghers relief with a bit of praise for the Pens. Man, they
10. are killing those Devils worse than I thought. Jagr just showed you why
11. he is much better than his regular season stats. He is also a lot
12. fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
13. fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to see the Islanders lose the final
14. regular season game. PENS RULE!!!

其中新闻文本对应的主题标签，已经用0-19这20个数字表示。

**文本向量化**

由于数据集中每一条数据都是很长的一个字符串，所以我们需要对数据进行向量化的处理。例如，I have a apple！\n I have a pen!可能需要将该字符串转换成向量如[10, 7, 0, 1, 2, 6, 22, 100, 8, 0, 1, 0]。

sklearn提供了实现词频向量化功能的CountVectorizer类。想要对数据进行向量化，代码如下：

1. from sklearn.feature\_extraction.text import CountVectorizer
2. #实例化向量化对象
3. vec = CountVectorizer()
4. #将训练集中的新闻向量化
5. X\_train = vec.fit\_transform(X\_train)
6. #将测试集中的新闻向量化
7. X\_test = vec.transform(X\_test)

但是仅仅通过统计词频的方式来将文本转换成向量会出现一个问题：长的文章词语出现的次数会比短的文章要多，而实际上两篇文章可能谈论的都是同一个主题。

为了解决这个问题，我们可以使用tf-idf来构建文本向量，sklearn中已经提供了tf-idf的接口，示例代码如下：

1. from sklearn.feature\_extraction.text import TfidfTransformer
2. #实例化tf-idf对象
3. tfidf = TfidfTransformer()
4. #将训练集中的词频向量用tf-idf进行转换
5. X\_train = vec.fit\_transform(X\_train\_count\_vectorizer)
6. #将测试集中的词频向量用tf-idf进行转换
7. X\_test = vec.transform(X\_test\_count\_vectorizer)

**MultinomialNB**

MultinomialNB是sklearn中多项分布数据的朴素贝叶斯算法的实现，并且是用于文本分类的经典朴素贝叶斯算法。在本关中建议使用MultinomialNB来实现文本分类功能。

在MultinomialNB实例化时alpha是一个常用的参数。

* alpha: 平滑因子。当等于1时，做的是拉普拉斯平滑；当小于1时做的是Lidstone平滑；当等于0时，不做任何平滑处理。

MultinomialNB类中的fit函数实现了朴素贝叶斯分类算法训练模型的功能，predict函数实现了法模型预测的功能。

其中fit函数的参数如下：

* X：大小为[样本数量,特征数量]的ndarry，存放训练样本
* Y：值为整型，大小为[样本数量]的ndarray，存放训练样本的分类标签

而predict函数有一个向量输入：

* X：大小为[样本数量,特征数量]的ndarry，存放预测样本

MultinomialNB的使用代码如下：

1. clf = tree.MultinomialNB()
2. clf.fit(X\_train, Y\_train)
3. result = clf.predict(X\_test)

**编程要求**

填写news\_predict(train\_sample, train\_label, test\_sample)函数完成新闻文本主题分类任务，其中：

* train\_sample：原始训练样本，类型为ndarray
* train\_label：训练标签，类型为ndarray
* test\_sample：原始测试样本，类型为ndarray

**测试说明**

只需返回预测结果即可，程序内部会检测您的代码，预测正确率高于0.8视为过关。

开始你的任务吧，祝你成功！

参考答案

1. from sklearn.feature\_extraction.text import CountVectorizer # 从sklearn.feature\_extraction.text里导入文本特征向量化模块
2. from sklearn.naive\_bayes import MultinomialNB
3. from sklearn.feature\_extraction.text import TfidfTransformer
4. def news\_predict(train\_sample, train\_label, test\_sample):
5. '''
6. 训练模型并进行预测，返回预测结果
7. :param train\_sample:原始训练集中的新闻文本，类型为ndarray
8. :param train\_label:训练集中新闻文本对应的主题标签，类型为ndarray
9. :test\_sample:原始测试集中的新闻文本，类型为ndarray
10. '''
11. # \*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*#
12. vec = CountVectorizer()
13. train\_sample = vec.fit\_transform(train\_sample)
14. test\_sample = vec.transform(test\_sample)
15. tfidf = TfidfTransformer()
16. train\_sample = tfidf.fit\_transform(train\_sample)
17. test\_sample = tfidf.transform(test\_sample)
18. mnb = MultinomialNB(alpha=0.01) # 使用默认配置初始化朴素贝叶斯
19. mnb.fit(train\_sample, train\_label) # 利用训练数据对模型参数进行估计
20. predict = mnb.predict(test\_sample) # 对参数进行预测
21. return predict
22. # \*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*#