* [任务描述](https://www.educoder.net/tasks/juo6w39seifn" \l "任务描述)
* [相关知识](https://www.educoder.net/tasks/juo6w39seifn#相关知识)
  + [准确度的缺陷](https://www.educoder.net/tasks/juo6w39seifn#准确度的缺陷)
  + [混淆矩阵](https://www.educoder.net/tasks/juo6w39seifn#混淆矩阵)
* [编程要求](https://www.educoder.net/tasks/juo6w39seifn#编程要求)
* [测试说明](https://www.educoder.net/tasks/juo6w39seifn#测试说明)

任务描述

本关任务:填写python代码，完成confusion\_matrix函数实现二分类混淆矩阵的构建。

相关知识

为了完成本关任务，你需要掌握：

* 准确度的缺陷
* 混淆矩阵

准确度的缺陷

准确度这个概念相信对于大家来说肯定并不陌生，就是正确率。例如模型的预测结果与数据真实结果如下表所示：

| **编号** | **预测结果** | **真实结果** |
| --- | --- | --- |
| 1 | 1 | 2 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |
| 4 | 1 | 1 |
| 5 | 2 | 3 |

很明显，连小朋友都能算出来该模型的准确度为3/5。

那么准确对越高就能说明模型的分类性能越好吗？非也！举个例子，现在我开发了一套癌症检测系统，只要输入你的一些基本健康信息，就能预测出你现在是否患有癌症，并且分类的准确度为0.999。您认为这样的系统的预测性能好不好呢？

您可能会觉得，哇，这么高的准确度！这个系统肯定很牛逼！但是我们知道，一般年轻人患癌症的概率非常低，假设患癌症的概率为0.001，那么其实我这个癌症检测系统只要一直输出您没有患癌症，准确度也可能能够达到0.999。

假如现在有一个人本身已经患有癌症，但是他自己不知道自己患有癌症。这个时候用我的癌症检测系统检测发现他没有得癌症，那很显然我这个系统已经把他给坑了（耽误了治疗）。

看到这里您应该已经体会到了，一个分类模型如果光看准确度是不够的，尤其是对这种样本**极度不平衡**的情况（10000条健康信息数据中，只有1条的类别是患有癌症，其他的类别都是健康）。

混淆矩阵

想进一步的考量分类模型的性能如何，可以使用其他的一些性能指标，例如精准率和召回率。但这些指标计算的基础是**混淆矩阵**。

继续以癌症检测系统为例，癌症检测系统的输出不是有癌症就是健康，这里为了方便，就用1表示患有癌症，0表示健康。假设现在拿10000条数据来进行测试，其中有9978条数据的真实类别是0，系统预测的类别也是0，有2条数据的真实类别是1却预测成了0，有12条数据的真实类别是0但预测成了1，有8条数据的真实类别是1，预测结果也是1。

如果我们把这些结果组成如下矩阵，则该矩阵就成为**混淆矩阵**。

| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | 9978 | 12 |
| 1 | 2 | 8 |

混淆矩阵中每个格子所代表的的意义也很明显，意义如下：

| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | 预测0正确的数量 | 预测1错误的数量 |
| 1 | 预测0错误的数量 | 预测1正确的数量 |

如果将正确看成是True，错误看成是False，0看成是Negtive，1看成是Positive。然后将上表中的文字替换掉，混淆矩阵如下：

| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | TN | FP |
| 1 | FN | TP |

因此TN表示真实类别是Negtive，预测结果也是Negtive的数量；FP表示真实类别是Negtive，预测结果是Positive的数量；FN表示真实类别是Positive，预测结果是Negtive的数量；TP表示真实类别是Positive，预测结果也是Positive的数量。

很明显，当FN和FP都等于0时，模型的性能应该是最好的，因为模型并没有在预测的时候犯错误。即如下混淆矩阵：

| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | 9978 | 0 |
| 1 | 0 | 22 |

**所以模型分类性能越好，混淆矩阵中非对角线上的数值越小。**

编程要求

根据提示，填写python代码，完成confusion\_matrix函数实现二分类混淆矩阵的构建。

confusion\_matrix函数中的参数：

* y\_true：数据的真实类别，类型为ndarray
* y\_predict：模型预测的类别，类型为ndarray

测试说明

平台会对你编写的代码进行测试，期望您的代码根据输入来输出正确的混淆矩阵，以下为其中一个测试用例（y\_true表示真实类别，y\_predict表示预测类别）：

测试输入：  
{'y\_true':[1, 0, 0, 1, 0, 1, 0], 'y\_predict':[0, 1, 0, 1, 0, 1, 0]}

预期输出：

1. [[3 1]
2. [1 2]]

开始你的任务吧，祝你成功！

import numpy as np

def confusion\_matrix(y\_true, y\_predict):

'''

构建二分类的混淆矩阵，并将其返回

:param y\_true: 真实类别，类型为ndarray

:param y\_predict: 预测类别，类型为ndarray

:return: shape为(2, 2)的ndarray

'''

#\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*#

def TN(y\_true, y\_predict):

return np.sum((y\_true == 0) & (y\_predict == 0))

def FP(y\_true, y\_predict):

return np.sum((y\_true == 0) & (y\_predict == 1))

def FN(y\_true, y\_predict):

return np.sum((y\_true == 1) & (y\_predict == 0))

def TP(y\_true, y\_predict):

return np.sum((y\_true == 1) & (y\_predict == 1))

return np.array([

[TN(y\_true, y\_predict), FP(y\_true, y\_predict)],

[FN(y\_true, y\_predict), TP(y\_true, y\_predict)]

])

#\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*#

* [任务描述](https://www.educoder.net/tasks/urwq9bk84vxp#%E4%BB%BB%E5%8A%A1%E6%8F%8F%E8%BF%B0)
* [相关知识](https://www.educoder.net/tasks/urwq9bk84vxp#%E7%9B%B8%E5%85%B3%E7%9F%A5%E8%AF%86)
  + [精准率](https://www.educoder.net/tasks/urwq9bk84vxp#%E7%B2%BE%E5%87%86%E7%8E%87)
  + [召回率](https://www.educoder.net/tasks/urwq9bk84vxp#%E5%8F%AC%E5%9B%9E%E7%8E%87)
  + [精准率与召回率之间的关系](https://www.educoder.net/tasks/urwq9bk84vxp#%E7%B2%BE%E5%87%86%E7%8E%87%E4%B8%8E%E5%8F%AC%E5%9B%9E%E7%8E%87%E4%B9%8B%E9%97%B4%E7%9A%84%E5%85%B3%E7%B3%BB)
  + [应该选精准率还是召回率作为性能指标？](https://www.educoder.net/tasks/urwq9bk84vxp#%E5%BA%94%E8%AF%A5%E9%80%89%E7%B2%BE%E5%87%86%E7%8E%87%E8%BF%98%E6%98%AF%E5%8F%AC%E5%9B%9E%E7%8E%87%E4%BD%9C%E4%B8%BA%E6%80%A7%E8%83%BD%E6%8C%87%E6%A0%87%EF%BC%9F)
* [编程要求](https://www.educoder.net/tasks/urwq9bk84vxp#%E7%BC%96%E7%A8%8B%E8%A6%81%E6%B1%82)
* [测试说明](https://www.educoder.net/tasks/urwq9bk84vxp#%E6%B5%8B%E8%AF%95%E8%AF%B4%E6%98%8E)

#### 任务描述

本关任务:填写python代码，完成precision\_score函数和recall\_score函数分别实现计算精准率和召回率。

#### 相关知识

为了完成本关任务，你需要掌握：

* 精准率
* 召回率

##### 精准率

**精准率(Precision)**指的是模型预测为Positive时的预测准确度，其计算公式如下：

Precisioin=\frac{TP}{TP+FP}*Precisioin*=​*TP*+*FP*​​*TP*​​

假如癌症检测系统的混淆矩阵如下：

| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | 9978 | 12 |
| 1 | 2 | 8 |

则该系统的精准率=8/(8+12)=0.4。

0.4这个值表示癌症检测系统的预测结果中如果有100个人被预测成患有癌症，那么其中有40人是真的患有癌症。**也就是说，精准率越高，那么癌症检测系统预测某人患有癌症的可信度就越高。**

##### 召回率

**召回率(Recall)**指的是我们关注的事件发生了，并且模型预测正确了的比值，其计算公式如下：

Recall=\frac{TP}{FN+TP}*Recall*=​*FN*+*TP*​​*TP*​​

假如癌症检测系统的混淆矩阵如下：

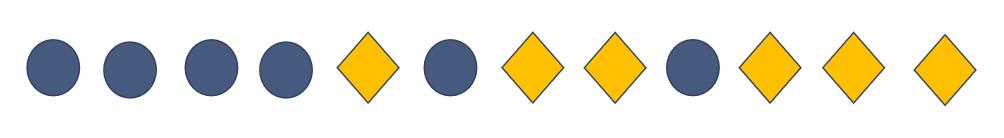
| **真实\预测** | **0** | **1** |
| --- | --- | --- |
| 0 | 9978 | 12 |
| 1 | 2 | 8 |

则该系统的召回率=8/(8+2)=0.8。

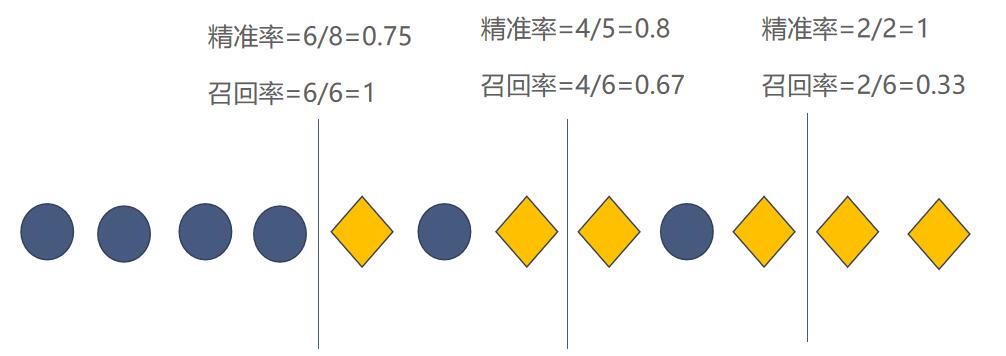
从计算出的召回率可以看出，假设有100个患有癌症的病人使用这个系统进行癌症检测，系统能够检测出80人是患有癌症的。**也就是说，召回率越高，那么我们感兴趣的对象成为漏网之鱼的可能性越低。**

##### 精准率与召回率之间的关系

假设有这么一组数据，菱形代表Positive，圆形代表Negtive。



现在需要训练一个模型对数据进行分类，假如该模型非常简单，就是在数据上画一条线作为分类边界。模型认为边界的左边是Negtive，右边是Positive。如果该模型的分类边界向左或者向右移动的话，模型所对应的精准率和召回率如下图所示：



从上图可知，**模型的精准率变高，召回率会变低，精准率变低，召回率会变高。**

##### 应该选精准率还是召回率作为性能指标？

到底应该使用精准率还是召回率作为性能指标，其实是**根据具体业务来决定的。**

比如我现在想要训练一个模型来预测我关心的股票是涨(Positive)还是跌(Negtive)，**那么我们应该主要使用精准率作为性能指标**。因为精准率高的话，则模型预测该股票要涨的可信度就高（很有可能赚钱！）。

比如现在需要训练一个模型来预测人是(Positive)否(Negtive)患有艾滋病，**那么我们应该主要使用召回率作为性能指标**。因为召回率太低的话，很有可能存在漏网之鱼（可能一个人本身患有艾滋病，但预测成了健康），这样就很可能导致病人错过了最佳的治疗时间，这是非常致命的。

#### 编程要求

根据提示，填写python代码，完成precision\_score函数和recall\_score函数分别实现计算精准率和召回率。

precision\_score函数中的参数:

* y\_true：数据的真实类别，类型为ndarray
* y\_predict：模型预测的类别，类型为ndarray

recall\_score函数中的参数:

* y\_true：数据的真实类别，类型为ndarray
* y\_predict：模型预测的类别，类型为ndarray

#### 测试说明

平台会对你编写的代码进行测试，期望您的代码根据输入来输出正确的精准率和召回率，以下为其中一个测试用例(y\_true表示真实类别，y\_predict表示预测类别)：

测试输入：  
{'y\_true':[1, 0, 0, 1, 0, 1, 0], 'y\_predict':[0, 1, 0, 1, 0, 1, 0]}

预期输出：  
0.666667, 0.666667

开始你的任务吧，祝你成功！