Hans Journal of Data Mining 数据挖掘, 2022, 12(4), 351-367
Published Online October 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.124033

Hans汉斯

基于混淆矩阵的机器学习分类评价指标研究及Python实践

姚世祎，杨盛腾，李裕梅

北京工商大学数学与统计学院，北京

收稿日期：2022年9月7日；录用日期：2022年10月7日；发布日期：2022年10月17日

摘　要

基于混淆矩阵的机器学习分类指标体系是在衡量各个分类器的分类效果中最常用的，本文对这些指标的计算原理进行了全面的整理，其中，对二分类的G-mean值和Matthews相关系数做了三分类及更多分类上的推广定义，并利用UCI的机器学习数据的Vehicle Silhouettes (汽车轮廓)数据集进行了基于Sklearn的相应python实验，给出了相应python代码以及运行结果等。特别地，对于推广到多分类的两个指标定义了相应的python函数，并进行了相应实验和验证。本文为最常用的机器学习分类评价指标的选取提供理论和python实践及相应代码的参考，为广大学者选用指标提供了依据。

关键词

机器学习分类，评价指标，G-Mean值多分类定义，Matthews相关系数多分类定义，Vehicle Silhouettes (汽车轮廓)数据集，Python实践

Research on Performance Measure Indicators of Machine Learning Classification Based on Confusion Matrix and Corresponding Python Practice

Shiyi Yao, Shengteng Yang, Yumei Li

School of Mathematics and Statistics, Beijing Technology and Business University, Beijing

Received: Sep. 7th, 2022; accepted: Oct. 7th, 2022; published: Oct. 17th, 2022

Abstract

The machine learning classification performance measure system based on confusion matrix is

文章引用: 姚世祎, 杨盛腾, 李裕梅. 基于混淆矩阵的机器学习分类评价指标研究及 Python 实践[J]. 数据挖掘, 2022, 12(4): 351-367. DOI: 10.12677/hjdm.2022.124033

the most commonly used in measuring the classification effect of each classifier, and the calculation principle of these performance measures are comprehensively listed in this paper. Among them, the G-mean value and Matthews correlation coefficient of two classifications are generalized and defined in three or more classification problems. Moreover, using the Vehicle Silhouettes dataset in UCI, the corresponding python experiments are implemented in the bases of Sklearn, and corresponding python codes and running results are given. In particular, the corresponding python functions are defined for the two generalized measures, and corresponding experiments and validations are also implemented. This paper provides a theoretical basis and python practice and corresponding codes for the selection of the most commonly used machine learning classification performance measures for the majority of scholars to select appropriate performance measures.

Keywords

Machine Learning Classification, Performance Measures, Definition of G-Mean in Multi-Classification, Definition of Matthews Correlation Coefficient in Multi-Classification, Vehicle Silhouettes Dataset, Python Practice

1. 引言

机器学习分类实验的效果，主要根据验证集或测试集的分类预测情况用指标进行衡量，而各种分类指标从不同的方面对预测结果进行了计算和度量。各种研究论文中，最常用的是基于混淆矩阵的分类评价指标体系[1]-[7]，其中，最常用的计算指标之一是分类准确率(Accuracy) [2]-[7]，用来衡量整体效果，是总整体上看有多少样本被分类正确；其次是分类准确率(Precision) [1] [3] [4]，用于衡量正确预测的正类样本数占所有被预测为正类的样本数的比例，这个指标只关注正类样本的情况；经常和 Accuracy 或 Precision 一起用，还有召回率(Recall) [1] [2] [3] [4]，它关注分类正确的正类样本个数占所有正类个数的比例；再者是混淆矩阵(Confusion Matrix) [4] [7] [8]，它给出每类样本被分到各个类别的数目，可以从它计算出很多其它的分类指标，比如 Accuracy、Precision、Recall 等，其中 Precision 和 Recall 都只关注正类样本的分类情况，这可能是在研究某些问题的时候，确实只关注正类样本就够了。在这些可计算的分类性能度量的指标基础上，还有一些别的指标，有的是致力于改进前面一些指标单方面的不足，综合和均衡前面多个指标，比如 F1-score [1] [3] [7]，目的在于均衡 Precision 和 Recall，构造一个综合的分类衡量指标；G-mean 值[9]，是对正类和负类的 Recall 进行综合表达；和 Matthews 相关系数[10]，是根据混淆矩阵构造的一个综合指标；Kappa 系数也是根据混淆矩阵中的值进行综合的一个衡量样本实际类别与模型预测类别一致程度的指标[7]。关于分类性能衡量的可视化手段，有 P_R 曲线[7]和 ROC 曲线[11] [12]，其中，P-R 曲线，是根据分类器对于 Precision 还是 Recall 的重视程度而取不同的阈值得到；ROC 曲线，是利用正类中被预测为正类的比例，以及负类中被预测成负类的比例，分别做成横纵坐标，形成的曲线，该曲线下的面积称为 AUC 值，这个值越大，分类效果越好。

对于这些指标，多数论文是选择某些指标来衡量其分类效果，很少有论文综合研究这些指标，其中，文献[13]对分类指标做了综述，从基于错误率的、基于混淆矩阵的和基于统计显著性检验的三大类性能度量方面进行了指标综述，列出了每一个指标的就算原理，并针对一个取值范围在[0, 1]的数据集，采用最近邻(KNN)和最优子集回归两种方法进行回归和分类实验，将计算得到的各个指标值列在一起进行了对

比和相关性分析，得到的结论是：在进行算法性能度量时，应根据实际的任务需求来选择适用的性能度量指标，且应综合分析多种性能度量指标的结果，给出一个相对准确的结论。这说明各个指标具有自己的特点，并不能相互代替，研究者们可以根据不同的需求选择使用不同的指标。但这个论文并没有给出实践代码以供参考，并且对于大家最常使用的基于混淆矩阵的指标体系总结的不够全面。我们希望把最常用的基于混淆矩阵的指标进行更全面的总结，并且给出在一份实际的机器学习数据集上针对 BP 网络进行的 python 实验及指标计算过程和相关代码等，为读着提供更有效的参考。

另外，这些衡量指标，大部分指标既可以针对二分类情况，又可以针对多分类情况，但有很少几个指标目前为止都只针对二分类情况，比计算指标中的 G-mean 值和 Matthews 相关系数，又比如可视化手段的 P_R 曲线和 ROC 曲线。本文针对 G-mean 值和 Matthews 相关系数，做了三分类以上的推广，定义出相应的 python 函数，并进行了相应的验证实验。

2. 实验数据集及混淆矩阵

2.1. 实验数据集

为验证机器学习各种分类评价指标，本文使用 UCI 数据库中 Vehicle Silhouettes (汽车轮廓)数据集中前三个类别的数据和 sklearn 中的 MLPClassifier 分类器，进行 BP 神经网络分类实验，并将各种分类评价指标进行总结、分析。

原始数据利用从汽车轮廓中提取的一组特征，将给定的样本分类为四种类型的车辆之一。其中共前三个类别包括 647 条样本，每条样本有 18 个指标，倒数第二列为样本的真实类别，包括 opel、saab、bus、van 四种类型，最后一列为类别标号。下面表 1 展示了该数据集的部分数据。

Table 1. Some data of Vihicle Silhouettes dataset
表 1. Vehicle Silhouettes 数据集部分数据

	X1	X2	X3	X4	X5	X6	X7	X8	...	X18	class
1	85	44	70	205	103	52	149	45	...	183	bus
2	107	57	106	172	50	6	255	26	...	183	bus
3	97	43	73	173	65	6	153	42	...	204	bus
4	88	46	74	171	68	6	152	43	...	195	bus
...	...	...	...	...	...	...	...	...	...	...	...
647	86	36	78	146	58	7	135	50	...	195	saab

导入所需的程序包，代码如下：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
```

先对各指标数据分别进行归一化，再取出归一化后的数据集中前两类的数据，共 430 条样本，其中70%作为训练集，30%作为测试集。将训练集代入二分类 BP 网络中进行训练，并得到测试集的预测类别结果，代码如下：

```
#生成二分类训练集和测试集
X_binary = vehicle.iloc[:430, :18]
X_binary = (X_binary-np.min(X_binary))/(np.max(X_binary)-np.min(X_binary))
Y_binary = vehicle.iloc[:430, 19]
X_binary_train, X_binary_test, Y_binary_train, Y_binary_test = train_test_split(
    X_binary, Y_binary, test_size = 0.3, random_state = 8)
#训练二分类 BP 神经网络
clf_binary = MLPClassifier(solver = 'lbfgs', hidden_layer_sizes = (5, 5),
    activation = 'logistic', max_iter = 30, random_state = 0)#共两个隐藏层，每层 5 个节点，激活函数为 logistic 函数，最大训练步数 30 步。
clf_binary.fit(X_binary_train, Y_binary_train)
Y_binary_pred = clf_binary.predict(X_binary_test)
```

先对各指标数据分别进行归一化，再取出归一化后的数据集中前三类的数据，共 647 条样本，其中70%作为训练集，30%作为测试集。将训练集代入多分类 BP 网络中进行训练，并得到测试集的预测类别结果，代码如下：

```
#生成多分类训练集和测试集
X_multi = vehicle.iloc[:, :18]
X_multi = (X_multi-np.min(X_multi))/(np.max(X_multi)-np.min(X_multi))
Y_multi = vehicle.iloc[:, 19]
X_multi_train, X_multi_test, Y_multi_train, Y_multi_test = train_test_split(
    X_multi, Y_multi, test_size = 0.3, random_state = 8)
#训练多分类器 BP 神经网络
clf_multi = MLPClassifier(solver = 'lbfgs', hidden_layer_sizes = (8, 6),
activation = 'logistic', max_iter = 200, random_state = 0)#共两个隐藏层，第一层 8 个节点，第二层 6 个节点，激活函数为 logistic 函数，最大训练步数 200 步。
clf_multi.fit(X_multi_train, Y_multi_train)
Y_multi_pred = clf_multi.predict(X_multi_test)
```

2.2. 混淆矩阵及相应 Python 实现

混淆矩阵(Confusion Matrix)，是评价机器学习分类精度的一种常见指标，其每一行代表样本的真实类别，每一列代表样本的预测类别。

1) 二分类混淆矩阵

在二分类情况下，混淆矩阵如图 1 所示，由 TP，FN，FP，TN 组成。

其中 TP 为实际是正类，预测为正类的样本数；FN 为实际是正类，预测为负类的样本数；FP 为实际是负类，预测为正类的样本数；TN 为实际是负类，预测为负类的样本数。相应 Python 实验代码如下：

```
confusion_binary = confusion_matrix(Y_binary_test, Y_binary_pred)
```

得到二分类混淆矩阵为：

$$\begin{pmatrix} 61 & 2 \\ 8 & 58 \end{pmatrix}$$

从混淆矩阵中可以看出，第 0 类只有 2 个样本被错误预测为第 1 类；第 1 类有 8 个样本被错误预测为第 0 类，说明整体预测效果较好。

2) 多分类混淆矩阵

在多分类情况下，以三分类为例，把其中一类当正类，另外两类当负类，分别针对三类中每一类进行计算，混淆矩阵如图 2 所示，由 T00，F01，F02，F10，T11，F12，F20，F21 和 T22 九个元素组成。

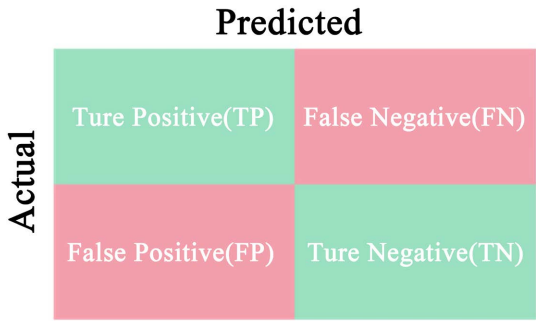

Figure 1. Confusion matrix for binary classification
图 1. 二分类混淆矩阵

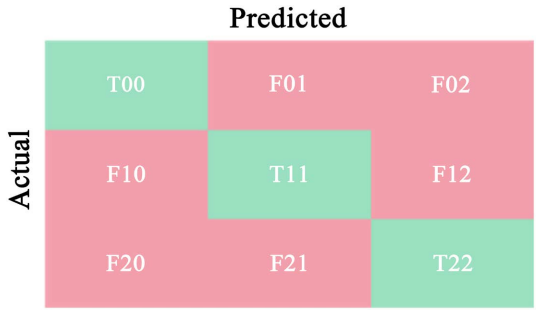

Figure 2. Confusion matrix for three classification
图 2. 三分类混淆矩阵

图中第一行代表第 0 类的情况，T00 表示本身是第 0 类，又被模型预测到第 0 类的样本数；F01 表示本身是第 0 类，但是被模型预测到第 1 类的样本数；F02 表示本身是第 0 类，但是被模型预测到第 2 类

的样本数。

图中第二行代表第 1 类的情况，F10 表示本身是第 1 类，但是被模型预测到第 0 类的样本数；T11 表示本身是第 1 类，又被模型预测到第 1 类的样本数；F12 表示本身是第 1 类，但是被模型预测到第 2 类的样本数。

图中第三行代表第 2 类的情况，F20 表示本身是第 2 类，但是被模型预测到第 0 类的样本数；F21 表示本身是第 2 类，但是被模型预测到第 1 类的样本数；T22 表示本身是第 2 类，又被模型预测到第 2 类的样本数。

相应 Python 实验代码如下：

```
confusion_multi = confusion_matrix(Y_multi_test, Y_multi_pred)
```

得到的三分类混淆矩阵为：

$$\begin{pmatrix} 64 & 0 & 0 \\ 3 & 42 & 17 \\ 5 & 17 & 47 \end{pmatrix}$$

从混淆矩阵中可以看出，第 0 类样本全部预测正确；第 1 类有 3 个样本被错误预测为第 0 类，有 17 个样本被错误预测为第 2 类；第 2 类有 5 个样本被错误预测为第 0 类，有 17 个样本被错误预测为第 1 类，说明整体预测效果较好。

三类以上情况与以上类似。

3. 计算指标

3.1. 单个计算指标

1) 准确率

准确率(Accuracy)，是指正确分类样本数占样本总量的比例，可用来评价模型整体预测效果。

在二分类情况下，如下面的公式：

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

公式中 TP (True Positive)代表本身是正类，又被模型预测到正类的样本数；TN (True Negative)代表本身是负类，又被模型预测到负类的样本数；FP (False Positive)代表本身是负类，但是被模型预测到正类的样本数；FN (False Negative)代表本身是正类，但被模型预测到负类的样本数。相应 Python 实验代码如下：

```
accuracy_binary = accuracy_score(Y_binary_test, Y_binary_pred)
```

得到测试样本的准确率为 0.9225，说明模型整体预测效果很好。

在多分类情况下，准确率就等于被模型正确预测的样本数总数占总样本数的比例，如下面的公式：

$$\text{Accuracy} = \frac{\sum_{i=0}^{n-1} T_{ii}}{\sum_{i=0}^{n-1} T_{ii} + \sum_{i=0}^{n-1} \sum_{j=0, j\neq i}^{n-1} F_{ij}}$$

公式中 T_{ii} 表示本身是第 i 类，又被模型预测到第 i 类的样本数；F_{ij} 表示本身是第 i 类，但是被模型预测到第 j 类的样本数。相应 Python 实验代码如下：

```
accuracy_multi = accuracy_score(Y_multi_test, Y_multi_pred)
```

得到测试样本的准确率为 0.7846，说明模型整体预测效果较好。

2) 召回率

召回率(Recall)，是指分类正确的正类样本个数占所有正类个数的比例，可用来评价正类样本的预测效果。由于在实际分类工作中，正类样本数往往是少数，被错分的代价也更高，所以模型拥有较高的正类召回率是很重要的。

在二分类情况下，如下面的公式：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

公式中 TP (True Positive)代表本身是正类，又被模型预测到正类的样本数；FN (False Negative)代表本身是正类，但被模型预测到负类的样本数。相应 Python 实验代码如下：

Recall_binary = recall_score(Y_binary_test, Y_binary_pred, average = None)

若设第 0 类为正类，得到的召回率为 0.9683；若设第 1 类为正类，得到的召回率为 0.8788。

在多分类情况下，可以计算每一类的召回率，某类的召回率就是该类正确预测的样本数占该类总样本数的比例，如下面的公式：

$$\text{Recall}_i = \frac{T_{ii}}{T_{ii} + \sum_{j=0, j\neq i}^{n-1} F_{ij}}$$

公式中 T_{ii} 表示本身是第 i 类，又被模型预测到第 i 类的样本数；F_{ij} 表示本身是第 i 类，但是被模型预测到第 j 类的样本数。相应 Python 实验代码如下：

Recall_multi = recall_score(Y_multi_test, Y_multi_pred, average = None)#average 参数设为 None 表示分别计算每一类的召回率

若设第 0 类为正类，得到的召回率为 1，因为第 0 类的全部样本都被正确预测为第 0 类；若设第 1 类为正类，得到的召回率为 0.6774；若设第 2 类为正类，得到的召回率为 0.6812，第 1 类和第 2 类的召回率较低。

3) 精确率

精确率(Precision)，是正确预测的正类样本数占所有被预测为正类的样本数的比例。二分的类情况如下：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

公式中 TP (True Positive)代表本身是正类，又被模型预测到正类的样本数；FP (False Positive)代表本身是负类，但是被模型预测到正类的样本数。相应 Python 实验代码如下：

Precision_binary = precision_score(Y_binary_test, Y_binary_pred, average = None)

若设第 0 类为正类，得到的精确率为 0.8841；若设第 1 类为正类，得到的精确率为 0.9667。

在多分类情况下，可以计算每一类的精确率，某类的精确率就是该类正确预测的样本数占所有被预测为该类的样本数的比例，如下面的公式：

$$\text{Precision}_j = \frac{T_{jj}}{T_{jj} + \sum_{i=1, i\neq j}^{n-1} F_{ij}}$$

公式中 T_{jj} 表示本身是第 j 类，又被模型预测到第 j 类的样本数；F_{ij} 表示本身是第 i 类，但是被模型预测到第 j 类的样本数。相应 Python 实验代码如下：

Precision_multi = precision_score(Y_multi_test, Y_multi_pred, average = None)#average 参数设为 None 表示分别计算每一类的精确率

若设第 0 类为正类，得到的精确率为 0.8889；若设第 1 类为正类，得到的精确率为 0.7119；若设第 2 类为正类，得到的精确率为 0.7344。

3.2. 综合计算指标

1) G-mean 值

了解 G-mean 值之前，首先应明确灵敏度(Sensitive)和特异度(Specificity)。灵敏度(Sensitive)与召回率(Recall)相同，是指正确预测的正类样本数占所有正类样本数的比例。特异度(Specificity)是指正确预测的负类样本数占所有负类样本数的比例。实际上，Sensitive 表示正类的召回率，Specificity 表示负类的召回率，如下面的公式：

$$\text{Sensitive} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}$$

在二分类情况下，G-mean 值等于灵敏度和特异度的几何平均数。G-mean 值以相同的权重综合了正类和负类的召回率，所以对数据样本在各类的分布不敏感，即使数据很不平衡，G-mean 值也能综合考虑到各类召回率，很好地评价一个分类器的性能。G-mean 值越高，说明模型整体预测效果越好，如下面的公式：

$$\text{G-mean} = \sqrt{\text{Sensitive} \times \text{Specificity}}$$

公式中 Sensitive 表示正类的召回率，Specificity 表示负类的召回率，相应 Python 实验代码如下：

Gmean_binary = np.sqrt(Recall_binary[0]*Recall_binary[1])

得到的 G-mean 值为 0.9224。

在多分类情况下，我们认为也可以将各个类别的召回率求几何平均数得到多分类的 G-mean 值，其值越高，说明模型整体预测效果越好，如下面的公式：

$$\text{G-mean} = \sqrt[n]{\prod_{i=0}^{n-1} \text{Recall}_i}$$

公式中 Recall_i 表示第 i 类的召回率，将 n 个类别的召回率相乘再开 n 次方，得到其几何平均值，即多分类 G-mean 值。相应 Python 实验代码如下：

Gmean_multi = math.pow(Recall_multi[0]*Recall_multi[1]*Recall_multi[2], 1/3)

得到的 G-mean 值为 0.7727。

对于三分类以上的情况，也可以将装有各类召回率的列表代入如下函数求 G-mean 值，代码如下：

```
import math
def Gmean(Recall_list):
    G = 1; n=len(Recall_list)
    for Recall in Recall_list:
        G = G*Recall
    return math.pow(G, 1/n)
```

2) Matthews 相关系数

Matthews 相关系数(Matthews correlation coefficient, MCC)，用一个值综合混淆矩阵，度量真实值与预测值之间的相关性。

在二分类情况下，公式中分母含有四项，分子含有两项，如下面的公式：

$$\mathrm{MCC}=\frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

分母中第一项(TP + FP)表示所有被预测为正类的样本数，第四项(TN + FN)表示所有被预测为负类的样本数，第二项(TP + FN)表示测试集中实际为正类的样本数，第三项(TN + FN)表示测试集中实际为负类的样本数。如果模型将所有测试样本全部预测为正类(分母第四项为 0)或负类(分母第一项为 0)，整个公式分母也为 0，应认为这是一个随机分类器，Matthews 相关系数应指定为 0。如果测试集中本没有正类样本(分母第二项为 0)，或负类样本(分母第三项为 0)，则 Matthews 相关系数没有意义。

分子中第一项(TP × TN)表示所有被正确预测的样本数的乘积，第二项(FP × FN)表示所有被错误预测的样本数的乘积。

Matthews 相关系数值介于[−1, 1]之间：

MCC = 1 时，FP 和 FN 均为 0，则分子第二项为 0，分母为 $\sqrt{\mathrm{TP}^2\mathrm{TN}^2}$，这表示每一类测试样本全部都正确预测到该类，分类器是完美的，分类结果全部正确。

MCC = 0 时，一种情况是由于模型将所有测试样本全部预测为正类或负类，此时分类器是随机分类器；另一种情况是公式的分子两项之差恰好为 0，此时 MCC 值也恰好为 0。

MCC = −1 时，TP 和 TN 均为 0，则分子第一项为 0，分母为 $\sqrt{\mathrm{FP}^2\mathrm{FN}^2}$，这表示每一类测试样本全部都被错误预测，分类器是最差的，分类结果全部错误。

相应 Python 实验代码如下：

```
MCC_binary = matthews_corrcoef(Y_binary_test, Y_binary_pred)
```

得到的 Matthews 相关系数值为：0.8489。

在多分类情况下，我们认为也可以将 Matthews 相关系数推广到多分类问题上，如下面的公式：

$$\mathrm{MCC}=\frac{\left(\prod_{i=0}^{n-1}T_{ii}\right)^{n-1}-\prod_{i=0}^{n-1}\prod_{j=0}^{n-1}F_{ij}}{\sqrt{\prod_{i=0}^{n-1}\prod_{j=0,j\neq i}^{n-1}\left(T_{ii}+F_{ij}\right)\times\prod_{j=0}^{n-1}\prod_{i=0,i\neq j}^{n-1}\left(T_{jj}+F_{ij}\right)}}$$

以三分类情况为例，如下面的公式：

$$\mathrm{MCC}=\frac{\left(T_{00}T_{11}T_{22}\right)^2-F_{01}F_{02}F_{10}F_{12}F_{20}F_{21}}{\sqrt{\begin{array}{l}(T_{00}+F_{01})(T_{00}+F_{02})(T_{11}+F_{10})(T_{11}+F_{12})(T_{22}+F_{20})(T_{22}+F_{21})\\(T_{00}+F_{10})(T_{00}+F_{20})(T_{11}+F_{01})(T_{11}+F_{21})(T_{22}+F_{02})(T_{22}+F_{12})\end{array}}}$$

与二分类情况类似，公式中分子表示所有被正确预测的样本数的乘积减去所有被错误预测的样本数的乘积；分母中第一行共六项，每类两项，共三类，表示实际属于各类的测试样本被正确预测的样本数(T_{ii})分别与错误预测至其他各类的样本数之和；分母第二行共六项，每类两项，共三类，表示实际属于各类的测试样本被正确预测的样本数(T_{ii})分别与其他各类错误预测至该类的样本数之和。

可以将混淆矩阵代入如下函数求得多分类的 Matthews 相关系数，代码如下：

```
import math
def MCC(confusion_matrix):
    n = confusion_matrix.shape [1]
    T = 1; F = 1
```

```
    for i in range(n):
        T = T*confusion_matrix[i,i]
    for i in range(n):
        for j in range(n):
            if j! = i:
                F = F*confusion_matrix[i,j]
    Actual = 1; Predict = 1
    for i in range(n):
        for j in range(n):
            if j! = i:
                Actual = Actual*math.sqrt(float(
                    confusion_matrix[i, i] + confusion_matrix[i, j]))
    for j in range(n):
        for i in range(n):
            if i! = j:
                Predict = Predict*math.sqrt(float(
                    confusion_matrix[j, j] + confusion_matrix[i, j]))
    T_pow = math.pow(T, n-1)
    AP = Actual*Predict
    return (T_pow-F)/AP
#将多分类混淆矩阵代入函数中
MCC_multi = MCC(confusion_multi)
```

得到的多分类 Matthews 相关系数为：0.4520。

将二分类混淆矩阵代入函数中验证，得到的 Matthews 相关系数结果与 sklearn 中 matthews_corrcoef 函数得到的结果同为 0.8489。

3) F1-Score

F-Score 也称 F-Measure 是一种用来综合衡量模型召回率和精确率的指标。召回率(Recall)和精确率(Precision)是一对矛盾的度量，一般情况下，Recall 值高时 Precision 值往往偏低，Precision 值偏高时 Recall 值往往偏低。当分类置信度高时，准确率 Precision 偏高，召回率 Recall 值偏低；而分类置信度低时，准确率 Precision 偏低，召回率 Recall 值就偏高。

$$\text{F-Score} = \frac{\left(a^2+1\right)\times P\times R}{a^2\times P + R}$$

其中 a 表示权重因子，P (Precision)表示精确率，R (Recall)表示召回率，将上式变形后可得 $\frac{a^2+1}{\text{F-Score}} = \frac{a^2}{P} + \frac{1}{R}$。用类别数 2 分别除以上式两边，并将系数整理至右边可得

$$\text{F-Score} = \frac{2}{\frac{2a^2}{a^2+1}\times\frac{1}{P} + \frac{2}{a^2+1}\times\frac{1}{R}}$$

可以看出，F-Score 是召回率和精确率的加权调和平均值。且对于参数 a，当 $a > 1$ 时表示精确率比召回率更重要，当 $a < 1$ 时表示召回率比精确率更重要。$a = 1$ 时二者同样重要，由此可以引出 F1-Score。

F1-Score 是 F-Score 在 $a = 1$ 时的特例，是最常用的综合衡量模型召回率和精确率的指标，是召回率和精确率的调和平均值。

在二分类情况下，如下面的公式：

$$\text{F1-Score} = \frac{2PR}{P+R}, \quad \frac{2}{\text{F1-Score}} = \frac{1}{P} \times \frac{1}{R}$$

公式中 P 表示精确率，R 表示召回率，相应 Python 实验代码如下：

F1_binary = f1_score(Y_binary_test, Y_binary_pred, average = None)

若设第 0 类为正类，得到的 F1-Score 值为 0.9242；若设第 1 类为正类，得到的 F1-Score 值为 0.9206。

在多分类情况下，需要为 f1_score 函数设置合适的 average 参数(如表 2 所示)，不能使用默认的‘binary’，可以根据需求计算几种不同的 F1-Score 值。

Table 2. Some parameters of average
表 2. Average 部分参数

参数	含义
binary	返回指定类的 F1-Score (返回默认正类的值)，只适用于二分类情况
None	返回每一类的 F1-Score
micro	分别将各类指定为正类，其他类均为负类，得到各类的混淆矩阵，将各类混淆矩阵相加，得到多分类混淆矩阵，计算全局 F1-Score
macro	返回各类 F1-Score 的简单平均值
weighted	返回各类 F1-Score 的加权平均值，以各类真实样本数占比为权重

相应 Python 实验代码如下：

F1_multi = f1_score(Y_multi_test, Y_multi_pred, average = None)

若设第 0 类为正类，得到的 F1-Score 值为 0.9412；若设第 1 类为正类，得到的 F1-Score 值为 0.6942；若设第 2 类为正类，得到的 F1-Score 值为 0.7068。

F1_multi_micro = f1_score(Y_multi_test, Y_multi_pred, average = 'micro')

得到的全局 F1-Score 值为 0.7846。

F1_multi_macro = f1_score(Y_multi_test, Y_multi_pred, average = 'macro')

得到的各类简单平均 F1-Score 值为 0.7807。

F1_multi_weighted = f1_score(Y_multi_test, Y_multi_pred, average = 'weighted')

得到的各类加权平均 F1-Score 值为 0.7797。

4) Kappa 系数

Kappa 系数是一种基于混淆矩阵计算的，用于衡量样本实际类别与模型预测类别一致程度的综合指标。如下面的公式：

$$\text{Kappa} = \frac{P_o - P_e}{1 - P_e}$$

其中，P_o 值是指正确分类的样本数除以总样本数，也就等于模型的分类准确率 Accuracy；P_e 值是指各类实际样本数与被模型预测到该类的样本数的乘积之和，除以混淆矩阵中各项之和的平方。

在二分类情况下，如下面的公式：

$$P_o = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}},\quad P_e = \frac{(\mathrm{TP}+\mathrm{FN})\times(\mathrm{TP}+\mathrm{FP})+(\mathrm{TN}+\mathrm{FP})\times(\mathrm{TN}+\mathrm{FN})}{(\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN})^2}$$

P_e 公式中分子含有四项，其中第一项(TP + FN)表示正类的实际样本数；第二项(TP + FP)表示被模型预测到正类的样本数；第三项(TN + FP)表示负类的实际样本数；第四项(TN + FN)表示被模型预测到负类的样本数。分母表示混淆矩阵中各项之和的平方。相应的 python 代码如下：

Kappa_binary = cohen_kappa_score(Y_binary_test, Y_binary_pred)

得到的 Kappa 值为：0.8452。

在多分类情况下，如下面的公式：

$$P_o = \frac{\sum_{i=0}^{n-1} T_{ii}}{\sum_{i=0}^{n-1} T_{ii} + \sum_{i=0}^{n-1}\sum_{j=0,j\neq i}^{n-1} F_{ij}},\quad P_e = \frac{\sum_{i=0}^{n-1}\left(\left(T_{ii}+\sum_{j=0,j\neq i}^{n-1} F_{ij}\right)\times\left(T_{ii}+\sum_{j=0,j\neq i}^{n-1} F_{ji}\right)\right)}{\left(\sum_{i=0}^{n-1} T_{ii} + \sum_{i=0}^{n-1}\sum_{j=0,j\neq i}^{n-1} F_{ij}\right)^2}$$

P_e 公式中分子的 n 项中，每一项又包含两项的乘积，其中第一项表示第 i 类的实际样本数；第二项表示被模型预测到第 i 类的样本数。分母表示混淆矩阵中各项之和的平方。相应的 Python 代码如下：

Kappa_multi = cohen_kappa_score(Y_multi_test, Y_multi_pred)

得到的 Kappa 值为：0.6768。

Kappa 系数取值介于[−1, 1]之间，Kappa 系数值越大说明模型分类效果越好，“样本实际类别”与“模型预测类别”这两者的一致程度越高，具体地说，就是：Kappa 值在 0.21~0.40 内可以认为两者具有“可接受”的一致性；Kappa 值在 0.41~0.60 内可以认为两者具有“中等”的一致性；Kappa 值在 0.61~0.80 内可以认为两者具有“较高”的一致性；Kappa 值 > 0.81 可以认为两者几乎完全一致。根据 Kappa 系数公式，当 $P_o = 1$ 且 $P_e \neq 1$ 时，即所有样本都被正确预测且不全属于同一类别时，Kappa 值为 1，说明样本实际类别与模型预测类别两者完全一致。

另外，Kappa 系数倾向于在不平衡数据集上得到更低的值。根据 P_e 值的公式，对于样本数在各类别中分布越不平衡的混淆矩阵，P_e 值会越高，相应地，在模型分类准确率 Accuracy (或者 P_o)相同时，Kappa 系数值会更低。

4. 基于混淆矩阵的可视化指标

1) P-R 曲线

P-R 曲线是由准确率(Precision)和召回率(Recall)共同构成的曲线，其中横轴为召回率，纵轴为准确率。很多二分类器输出的是一个概率值(因为输出层的激活函数一般用了 sigmoid 或者 softmax 函数，它们的输出值都在 0 到 1 之间)，将这个概率值与一个指定的阈值进行比较，若概率大于阈值，则分为正类，否则分为负类。实际上，可以根据概率值由大到小将测试样本排序，最可能是正例的样本排在最前面，最不可能是正例的样本排在最后面。分类的过程就相当于在这个排序中选择一个介于[0, 1]之间的阈值，阈值越接近 1，分类器越重视准确率，阈值越接近 0，分类器越重视召回率。选取的阈值大小不同，分类结果不同。通过改变阈值的大小，分别计算 P 值和 R 值，就得到了 P-R 曲线。

在二分类情况下，相应 Python 代码如下：

Y_prob0 = clf_binary.predict_proba(X_binary_test)[:, 0]

Y_prob1 = clf_binary.predict_proba(X_binary_test)[:, 1]

precision0，recall0，thresholds0 = precision_recall_curve(

```
    Y_binary_test, Y_prob0, pos_label = 0)
precision1，recall1，thresholds1 = precision_recall_curve(
    Y_binary_test, Y_prob1, pos_label = 1)
plt.figure()
plt.title('Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.plot(recall0, precision0, label = 'Class0')
plt.plot(recall1, precision1, label = 'Class1')
plt.legend(loc = 3)
plt.show()
```

根据代码得到的两分类 P-R 曲线如图 3 所示。

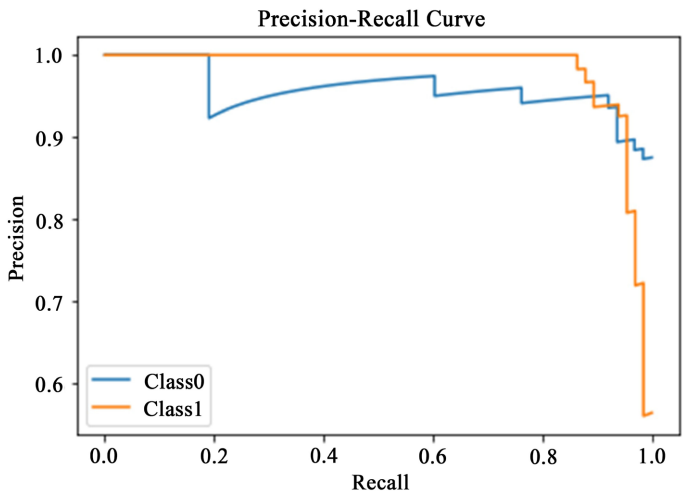

Figure 3. P-R curve
图 3. P-R 曲线

图 3 中蓝色为第 0 类为正类时的 P-R 曲线，橙色为第 1 类为正类时的 P-R 曲线。现实中的 P-R 曲线可能是非单调、不平滑的，在很多局部有上下波动。

2) ROC 曲线

ROC 曲线，又称受试者工作特征曲线 (Receiver operating characteristic curve)，其纵轴为真正例率(TPR)，即正类中被预测为正类的比例；横轴为假正例率(FPR)，即负类中被预测为负类的比例。如下面的公式：

$$\mathrm{TPR}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\quad \mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$$

利用有限个样本绘制 ROC 曲线图时，无法得到平滑的曲线，只能得到近似 ROC 曲线。对指定正类，设真正例个数为 n_p，假正例个数为 n_F，将所有被预测为正类的样本按分类器输出的正类概率从大到小排序。先把分类阈值设置为 1，即将所有样本均预测为负类，此时真正例率和假正例率均为 0，从坐标原点(0, 0)处开始绘制 ROC 曲线。再对排序好的测试样本，从第一位开始，每次都将分类阈值减小至该样本

的输出概率，即依次将每个样本划分为正类。设前一个标记点为(x, y)，若该样本为真正例，则标记$\left(x, y+\frac{1}{n_P}\right)$为当前点；若该样本为假正例，则标记$\left(x+\frac{1}{n_F}, y\right)$为当前点，用线段依次连接各标记点可得该正类的近似 ROC 曲线。

如果多数的正类概率大的测试样本确实属于正类，即被分类器正确预测为正类，则 ROC 曲线会呈现先向上再向右增长的趋势，这样的 ROC 曲线会靠近图左上角，ROC 曲线越靠近左上角说明分类器对该类的预测效果越好；如果 ROC 曲线与第一象限角平分线重合，说明分类器是随机分类器；如果 ROC 曲线靠近右下角，说明预测效果比随机分类器还差，分类器没有预测价值。

在二分类情况下，可以通过 sklearn 中自带的 plot_roc_curve 函数通过一行代码直接画出各类的 ROC 曲线图(如图 4 和图 5 所示)，相应 Python 代码：

```
plot_roc_curve(clf_binary, X_binary_test, Y_binary_test, pos_label = 0)
plot_roc_curve(clf_binary, X_binary_test, Y_binary_test, pos_label = 1)
```

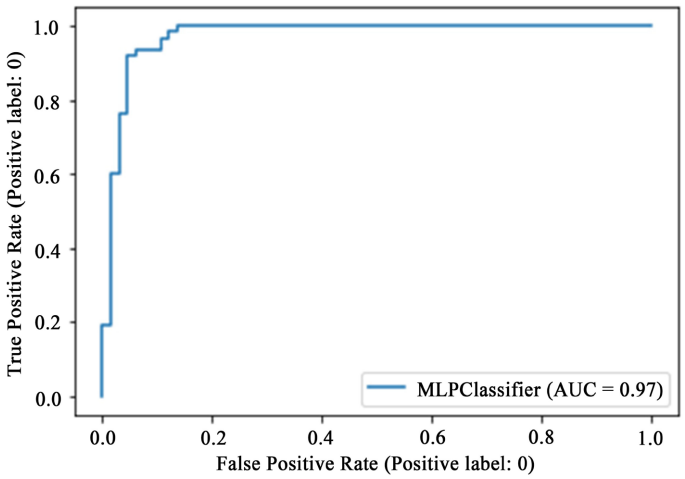

Figure 4. ROC curve of Class 0 for the binary classification
图 4. 二分类第 0 类 ROC 曲线

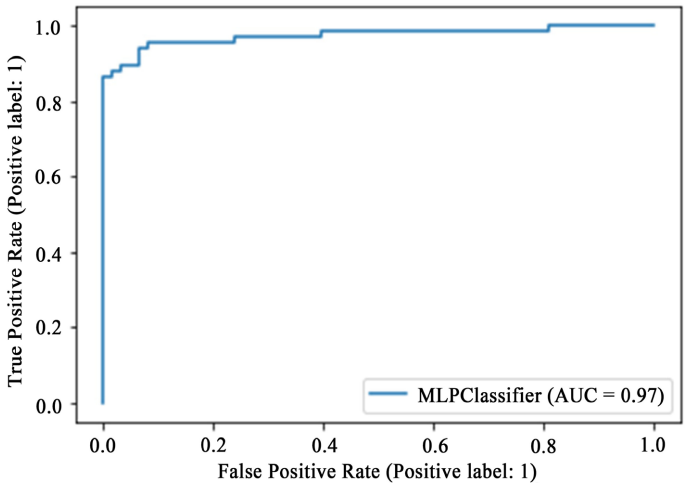

Figure 5. ROC curve of Class 1 for the binary classification
图 5. 二分类第 1 类 ROC 曲线图

从第 0 类和第 1 类的 ROC 曲线图中可以明显看出，第 0 类 ROC 曲线的线下面积达到 0.97，第 1 类 ROC 曲线的线下面积也达到 0.97，且两条 ROC 曲线都靠近左上角，说明分类器预测效果非常好。

也可以通过 sklearn 中自带的 roc_curve 函数，将两个类别的 ROC 曲线画在一张图上(如图 6 所示)。相应 Python 代码：

```
Y_binary_prob0 = clf_binary.predict_proba(X_binary_test)[:, 0]
Y_binary_prob1 = clf_binary.predict_proba(X_binary_test)[:, 1]
FP0_binary，TP0_binary，thresholds0_binary = roc_curve(
    Y_binary_test, Y_binary_prob0, pos_label = 0)
FP1_binary，TP1_binary，thresholds1_binary = roc_curve(
    Y_binary_test, Y_binary_prob1, pos_label = 1)
plt.title('ROC')
plt.plot(FP0_binary, TP0_binary, label = 'Class0')
plt.plot(FP1_binary, TP1_binary, label = 'Class1')
plt.xlim([−0.1, 1.1])
plt.ylim([−0.1, 1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc = 4)
plt.show()
```

在多分类情况下，将多个类别的 ROC 曲线画在一张图上(如图 7 所示)。相应 Python 代码：

```
Y_multi_prob0 = clf_multi.predict_proba(X_multi_test)[:, 0]
Y_multi_prob1 = clf_multi.predict_proba(X_multi_test)[:, 1]
Y_multi_prob2 = clf_multi.predict_proba(X_multi_test)[:, 2]
FP0_multi，TP0_multi，thresholds0_multi = metrics.roc_curve(
    Y_multi_test, Y_multi_prob0, pos_label = 0)
FP1_multi，TP1_multi，thresholds1_multi = metrics.roc_curve(
    Y_multi_test, Y_multi_prob1, pos_label = 1)
FP2_multi，TP2_multi，thresholds2_multi = metrics.roc_curve(
    Y_multi_test, Y_multi_prob2, pos_label = 2)
plt.title('ROC')
plt.plot(FP0_multi, TP0_multi, label = 'Class0')
plt.plot(FP1_multi, TP1_multi, label = 'Class1')
plt.plot(FP2_multi, TP2_multi, label = 'Class2')
plt.xlim([−0.1, 1.1])
plt.ylim([−0.1, 1.1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc = 4)
plt.show()
```

从图中可以看出，第 0 类的 ROC 曲线最靠近左上角，说明分类器对第 0 类的预测精度最高。第 1 类

和第 2 类的 ROC 曲线有交叉，无法直接判断哪一条的线下面积更大，这就要借助 AUC 值来进行判断。

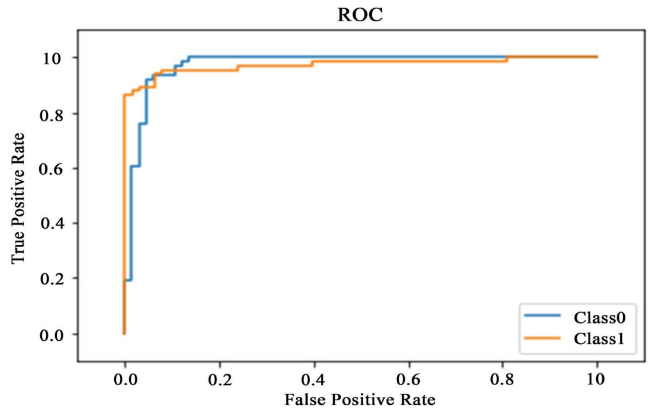

Figure 6. ROC curve of binary classification
图 6. 二分类 ROC 曲线

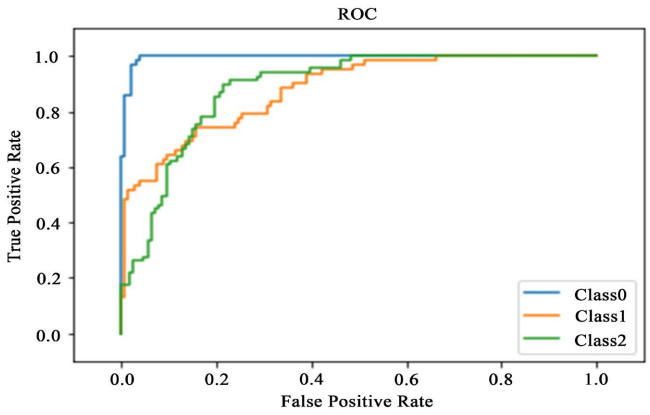

Figure 7. ROC curve for multi-category classification
图 7. 多分类 ROC 曲线

3) AUC 值

AUC 值(Area under curve)是指 ROC 曲线的右侧线下面积，可以直观衡量 ROC 曲线所表示的分类器预测效果的好坏。如果一类 ROC 曲线将另一类 ROC 曲线完全包住，说明分类器对前一类的预测效果好于后一类；如果两类 ROC 曲线有交叉，则要通过 AUC 值的大小来判断哪一类的预测效果好。

在二分类情况下，相应 Python 实验代码如下：

AUC0_binary = auc(FP0_binary, TP0_binary) 得到第 0 类的 AUC 值为 0.9733。	AUC1_binary = auc(FP1_binary, TP1_binary) 得到第 1 类的 AUC 值为 0.9733。

在多分类情况下，相应 Python 实验代码如下：

AUC0_multi = auc(FP0_multi, TP0_multi) 得到第 0 类的 AUC 值为 0.9948。	AUC1_multi = auc(FP1_multi, TP1_multi) 得到第 1 类的 AUC 值为 0.8789。 AUC2_multi = auc(FP2_multi, TP2_multi) 得到第 2 类的 AUC 值为 0.8862。

5. 结论

本文将机器学习的分类指标的原理进行了全面的整理和分析，针对 G-mean 值和 Matthews 相关系数做了三分类及更多分类上的推广定义，并用 UCI 数据集中的 Vehicle Silhouettes 数据在 BP 网络上进行训练和测试，然后根据测试情况计算各个分类指标，并给出相应的 python 代码，以及推广到三分类以上的 G-mean 值和 Matthews 相关系数的 python 函数定义。从理论和 python 实践方面都给读着提供参考。

基金项目

北京工商大学教育教学改革研究项目(jg215203)。

参考文献

[1] Yang, M., Yuan, Y. and Liu, G. (2022) SDUNet: Road Extraction via Spatial Enhanced and Densely Connected UNet. *Pattern Recognition*, **126**, Article ID: 108549. https://doi.org/10.1016/j.patcog.2022.108549

[2] Wang, X., Yu, X., Guo, L., Liu, F. and Xu, L. (2020) Student Performance Prediction with Short-Term Sequential Campus Behaviors. *Information*, **11**, 201. https://doi.org/10.3390/info11040201

[3] Wisanwanichthan, T. and Thammawichai, M. (2021) A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM. *IEEE Access*, **9**, 138432-138450. https://doi.org/10.1109/ACCESS.2021.3118573

[4] Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd Edition, O'Reilly Media, Incorporation, Sebastopol. https://doi.org/10.1016/j.landusepol.2022.106282

[5] Janusa, J. and Ertunç, E. (2022) Towards a Full Automation of Land Consolidation Projects: Fast land Partitioning Algorithm Using the Land Value Map. *Land Use Policy*, **120**, Article ID: 106282.

[6] Beeche, C., Singh, J.P, Leader, J.K., *et al.* (2022) Super U-Net: A Modularized Generalizable Architecture. *Pattern Recognition*, **128**, Article ID: 108669. https://doi.org/10.1016/j.patcog.2022.108669

[7] 张开放, 苏华友, 窦勇. 一种基于混淆矩阵的多分类任务准确率评估新方法[J]. 计算机工程与科学, 2021, 43(11): 1910-1919.

[8] 于营, 杨婷婷, 杨博雄. 混淆矩阵分类性能评价及 Python 实现[J]. 现代计算机, 2021(20): 70-73+79.

[9] 郭华平, 董亚东, 邬长安, 等. 面向类不平衡的逻辑回归方法[J]. 模式识别与人工智能, 2015, 28(8): 686-693.

[10] 刘洋. 基于 filter-wrapper mRMR 改进的 K 阶依赖贝叶斯网络分类模型[D]: [硕士学位论文]. 长春: 吉林大学, 2018.

[11] Alizadeh, S.H., Hediehloo, A. and Harzevili, N.S. (2021) Multi Independent Latent Component Extension of Naive Bayes Classifier. *Knowledge-Based Systems*, **213**, Article ID: 106646. https://doi.org/10.1016/j.knosys.2020.106646

[12] 王彦光, 朱鸿斌, 徐维超. ROC 曲线及其分析方法综述[J]. 广东工业大学学报, 2021, 38(1): 46-53.

[13] 杨杏丽. 分类学习算法的性能度量指标综述[J]. 计算机科学, 2021, 48(8): 209-219.