华中師範大學

自然语 言处 理
第十三章 评 价指标

主讲教师：曾江峰

华中师范大学 信息管理学院

jfzeng@ccnu.edu.cn

评价指标

- 信息分类的评价指标一般有以下几个：混淆矩阵、准确率、精准率、召回率、F1 Score 值、ROC 曲线、 AUC 面积和分类评估报告。
- 本章重点介绍了 Sklearn 的分类评价指标的函数和方法。讲解了中文分词的指标以及未登录词和登录词召回率。

评价指标

1.Sklearn中的评价指标

2.混淆矩阵

3.准确率

4.精确率

5.召回率

6.F1 Score

7.综合实例

8.ROC曲线

9.AUC面积

10.分类评估报告

11.NLP评价指标

1.Sclearn中的评价指标

- 评价模型的合理性、有效性，不同的机器学习任务有不同的评价指标，同一任务有时也会因为侧重点的不同具有不同的评价指标。
- sklearn.metrics模型评价指标有混淆矩阵、准确率、召回率、F1 Score、ROC曲线和AUC面积等。

1. Sklearn中的评价指标

术 语	Sclearn函数	术 语	Sclearn函数
混淆矩阵	confusion_matrix	ROC曲线	roc_curve
准确率	accuracy_score	AUC面积	roc_auc_score
召回率	recall_score	分类 评 估报告	classification_report
F1 Score	f1_score		

表1-1 分类问题的评价指标

2. 混淆矩阵

2.1 认识混淆矩阵

- 混淆矩阵又称为可能性表格、错误矩阵或分类矩阵，用于评估模型的预测精度，检查模型是否在预测时出现明显的错误。
- 混淆矩阵由n行n列组成，列代表预测值，行代表真实值。每列总数表示预测为该类别的数据数目，每行总数表示该类别数据的真实数目。
- 混淆矩阵的所有正确的预测结果都在对角线上，对角线之外的数据是预测错误结果。

2. 混淆矩阵

2.1 认识混淆矩阵

混淆矩阵		混淆矩阵	
		正例（Positive）	反例（Negative）
真实值	正例（True）	真阳性（TP）	真阴性（TN）
	反例（False）	假阳性（FP）	假阴性（FN）

表2-1 混淆矩阵

(1)样本全集 $= TP \cup FP \cup FN \cup TN$

(2)一个样本属于且只属于$N \times N$集合中的一个

2. 混淆矩阵

2.1 认识混淆矩阵

【例1】混淆矩阵

现有27只动物，其中8只猫、6条狗和13只兔子。将8只猫的其中3只预测成了狗;6条狗的其中有 1条预测成兔子，还有2条预测成猫。13只兔子的其中有2条被预测成了狗。

混淆矩阵		混淆矩阵		
		猫	狗	兔子
真实值	猫	5	3	0
	狗	2	3	1
	兔子	0	2	11

表2-2 混淆矩阵举例

2. 混淆矩阵

2.2 Pandas计算混淆矩阵

混淆矩阵本质上就是列联表，Pandas提供crosstab()函数求得列联表

pd.crosstab(index,columns,values=None)

参数	说明
index	指定了要分组的列，最终作为行
columns	指定了要分组的列，最终作为列
values	指定了要聚合的值（由行列共同影响）

表2-3 confusion_matrix()参数说明

2. 混淆矩阵

2.2 Pandas计算混淆矩阵

【例2】Pandas的crosstab()函数

```
# 例2: Pandas的crosstab()函数
import pandas as pd
results = pd.DataFrame()
results['True'] = [1,2,2,2,2]
results['Pred'] = [2,2,1,2,1]
print(pd.crosstab(results['True'],results['Pred']))   # pd.crosstab得到混淆矩阵
```

```
Pred  1  2
True
1     0  1
2     2  2
```


2. 混淆矩阵

2.3 Sklearn计算混淆矩阵

sklearn.metrics模块提供confusion_matrix()函数用于混淆矩阵

sklearn.metrics.confusion_matrix(y_true, y_pred, labels)

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值
labels	指定类别对应的数字

表2-4 confusion_matrix()参数说明

2. 混淆矩阵

2.3 Sklearn计算混淆矩阵

【例3】Sklearn的confusion_matrix()函数

```
# 例3: Sklearn中的confusion_matrix()函数
from sklearn.metrics import confusion_matrix
y_true = [2,0,2,2,0,1]
y_pred = [0,0,2,2,0,2]
print('confusion_matrix:\n',confusion_matrix(y_true,y_pred))

y_true = ['cat','ant','cat','cat','ant','bird']
y_pred = ['ant','ant','cat','cat','ant','cat']
print('confusion_matrix:\n',confusion_matrix(y_true,y_pred,labels = ['ant','bird','cat']))
```

```
confusion_matrix:
 [[2 0 0]
 [0 0 1]
 [1 0 2]]
confusion_matrix:
 [[2 0 0]
 [0 0 1]
 [1 0 2]]
```


3. 准确率

混淆矩阵		混淆矩阵	
		正例（Positive）	反例（Negative）
真实值	正例（True）	真阳性（TP）	真阴性（TN）
	反例（False）	假阳性（FP）	假阴性（FN）

3.1 认识准确率

准确率（Accuracy，缩写ACC）是最常用的分类性能指标。

准确率(Accuracy)=预测正确样本数/总样本数。公式如下所示：

$$ACC = \frac{TP + TN}{P + N}$$

3. 准确率

3.2 Sklearn计算准确率

sklearn.metrics模块提供accuracy_score()函数

sklearn.metrics.accuracy_score(y_true, y_pred, normalize)

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值
normalize	默认值为True，返回正确分类的比例；False返回正确分类的样本数

表3-1 accuracy_score()参数说明

3. 准确率

3.2 Sklearn计算准确率

【例4】accuracy_sore()举例

```
# 例4: accuracy_score()举例
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = [0,2,1,3]
y_true = [0,1,2,3]
print(accuracy_score(y_true,y_pred))
print(accuracy_score(y_true,y_pred,normalize = False))
```

```
0.5
2
```


4. 精确率

4.1 认识精确率

混淆矩阵		混淆矩阵	
		正例（Positive）	反例（Negative）
真实值	正例（True）	真阳性（TP）	真阴性（TN）
	反例（False）	假阳性（FP）	假阴性（FN）

精确率（Precision）又称为查准率，容易和准确率混淆。精确率只是针对预测正确的正样本而不是所有预测正确的样本，精准率是正确预测的正例数/预测正例总数。公式如下所示：

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

预测结果为正例当中确实为正例的概率

4. 精确率

4.1 认识精确率

sklearn.metrics模块提供precision_score()函数

sklearn.metrics.precision_score(y_true, y_pred)

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值

表4-1 precision_score()参数说明

4. 精确率

4.1 认识精确率

【例5】Sklearn计算精确率

```
# 例5: Sklearn计算精确率
from sklearn.metrics import precision_score
import numpy as np
y_true = np.array([1,0,1,1])
y_pred = np.array([0,1,1,0])    # 预测正例是1
p = precision_score(y_true,y_pred)
print(p)
```

```
0.5
```


5. 召回率

5.1 认识召回率

召回率（Recall）就是所有准确的条目有多少被检索出来，召回率可理解为查全率。查准率和查全率是一对矛盾的度量。一般来说，查准率高时，查全率往往偏低；而查全率高时，查准率往往偏低。召回率是正确预测的正例数/实际正例总数，计算公式如下所示：

$$Recall = \frac{TP}{TP + TN}$$

5. 召回率

5.2 Sklearn计算召回率

sklearn.metrics模块提供recall_score()函数

recall_score(y_true, y_pred,average)

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值
average	可取值有micro、macro、weighted

表5-1 recall_score()参数说明

5. 召回率

5.2 Sklearn计算召回率

【例6】Sklearn计算召回率

```
# 例6: Sklearn计算召回率
from sklearn.metrics import recall_score
y_true = [0,1,2,0,1,2]
y_pred = [0,2,1,0,0,1]
print(recall_score(y_true,y_pred,average = 'macro'))
print(recall_score(y_true,y_pred,average = 'micro'))
print(recall_score(y_true,y_pred,average = 'weighted'))
print(recall_score(y_true,y_pred,average = None))
```

```
0.3333333333333333
0.3333333333333333
0.3333333333333333
[1. 0. 0.]
```


6. F1 Score

6.1 认识F1 Score

F1分数（F1 Score）用于衡量二分类模型精确度，是精确率和召回率的调和值，变化范围在0-1。F1 Score计算公式如下所示：

$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

6. F1 Score

6.1 认识F1 Score

sklearn.metrics模块提供f1_score()函数

```
f1_score(y_test, predictions, average="micro")
```

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值

表6-1 f1_score()参数说明

6. F1 Score

6.2 认识F1 Score

【例7】Sklearn计算F1 Score

```
# 例7：Sklearn计算F1值
from sklearn import metrics
y_true = [0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2]
y_pred = [1,1,0,2,2,0,1,1,1,1,2,1,1,2,2,1,2,2,2,2,2,2,1,1]
F1 = metrics.f1_score(y_true,y_pred,average = 'micro')
print('F1:',F1)
```

```
F1: 0.625
```


7. 综合实例

准确率、精确率、召回率、F1 Score汇总如下

	公式	意义	
准确率（ACC）	准确率（ACC）：$ACC = \frac{TP + TN}{P + N}$；精确率（P值）：$Precision = \frac{TP}{TP + FP}$；召回率（R值）：$Recall = \frac{TP}{TP + FN}$；F1 Score（F1-Score）：$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	预测正确的结果占总样本的百分比；所有被预测为正的样本中实际为正的样本的概率；实际为正的样本中被预测为正样本的概率；F1 Score指标综合了精确率与召回率的产出结果	预测正确的结果占总样本的百分比	样本不均衡，准确率就会失效。
精确率（P值）	准确率（ACC）：$ACC = \frac{TP + TN}{P + N}$；精确率（P值）：$Precision = \frac{TP}{TP + FP}$；召回率（R值）：$Recall = \frac{TP}{TP + FN}$；F1 Score（F1-Score）：$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	预测正确的结果占总样本的百分比；所有被预测为正的样本中实际为正的样本的概率；实际为正的样本中被预测为正样本的概率；F1 Score指标综合了精确率与召回率的产出结果	所有被预测为正的样本中实际为正的样本的概率	只针对正样本
召回率（R值）	准确率（ACC）：$ACC = \frac{TP + TN}{P + N}$；精确率（P值）：$Precision = \frac{TP}{TP + FP}$；召回率（R值）：$Recall = \frac{TP}{TP + FN}$；F1 Score（F1-Score）：$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	预测正确的结果占总样本的百分比；所有被预测为正的样本中实际为正的样本的概率；实际为正的样本中被预测为正样本的概率；F1 Score指标综合了精确率与召回率的产出结果	实际为正的样本中被预测为正样本的概率	只针对正样本
F1 Score（F1-Score）	准确率（ACC）：$ACC = \frac{TP + TN}{P + N}$；精确率（P值）：$Precision = \frac{TP}{TP + FP}$；召回率（R值）：$Recall = \frac{TP}{TP + FN}$；F1 Score（F1-Score）：$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	预测正确的结果占总样本的百分比；所有被预测为正的样本中实际为正的样本的概率；实际为正的样本中被预测为正样本的概率；F1 Score指标综合了精确率与召回率的产出结果	F1 Score指标综合了精确率与召回率的产出结果	

表7-1 StandardScaler()参数说明

7. 综合实例

二分类可以指定正例和负例，那么在多分类任务中，Precision、Recall、F1这三个指标又将如何计算呢？

首先，将n分类任务转换成n个二分类任务；
然后，分别计算n个分类任务中的Precision、Recall、F1；
最后，加权求和得到n分类任务的Precision、Recall、F1。

7. 综合实例

7.1 数学计算评价指标

【例8】已知猫、猪、狗的混淆矩阵如表7-2所示。现计算准确率、精确率、召回率、F1 Score

		预测值		
		猫	狗	猪
真实值	猫	10	3	5
	狗	1	15	6
	猪	2	4	20

表7-2 混淆矩阵举例

7. 综合实例

7.1 数学计算评价指标

【例8】计算准确率、精确率、召回率、F1 Score

1. 准确率

在总共讨论的66个动物中，预测正确的有

$$10 + 15 + 20 = 45$$

个样本，因此准确率为

$$45/66 = 68.2\%$$

7. 综合实例

7.1 数学计算评价指标

【例8】计算准确率、精确率、召回率、F1 Score

现只讨论猫，将表7-2合并为二分问题，转变为如表7-3所示

		预测值	
		猫	不是猫
真实值	猫	10	8
	不是猫	3	45

表7-3 猫的混淆矩阵

7. 综合实例

7.1 数学计算评价指标

【例8】计算准确率、精确率、召回率、F1 Score

2. 精确率

预测13只猫中，只有10只预测正确，3只不是猫

$$Precision(\text{猫}) = 10/13 = 76.9\%$$

3. 召回率

18只猫中，只有10只是猫，8只不是猫

$$Recall(\text{猫}) = 10/18 = 55.6\%$$

7. 综合实例

7.1 数学计算评价指标

【例8】计算准确率、精确率、召回率、F1 Score

4. F1 Score

$$F1-core=\frac{2\times 0.769\times 0.556}{0.769+0.556}=64.54\%$$

7. 综合实例

7.2 Python计算评价指标

load_digits数据集是sklearn.datasets中内置的手写数字图片数据集，本例计算load_digits数据集的分类的评价指标

【例9】计算评价指标

```
# 例9：计算评价指标
import numpy as np
import pandas as pd
from sklearn import datasets
```

```
d = datasets.load_digits()
x = d.data
y = d.target.copy()    # 防止原来数据改变
print(len(y))
```

```
1797
```

```
y[d.target == 9] = 1
y[d.target != 9] = 0
print(y)
```

```
[0 0 0 ... 0 1 0]
```

```
# 统计各个数据出现的个数
print(pd.value_counts(y))
```

```
0    1617
1     180
dtype: int64
```

```
# 划分数据集为训练数据和测试数据
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=666)
```

```
# 使用计算学习算法—逻辑回归算法进行数据分类
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver = "newton-cg")
log_reg.fit(x_train,y_train)
print(log_reg.score(x_test,y_test))
```

```
0.9844444444444445
```


7. 综合实例

7.2 Python计算评价指标

【例9】计算评价指标

```
y_pred = log_reg.predict(x_test)
```

```
# 计算TN、FP、FN和TP
def TN(y_true,y_pred):
    return np.sum((y_true == 0) & (y_pred == 0))
def FP(y_true,y_pred):
    return np.sum((y_true == 0) & (y_pred == 1))
def FN(y_true,y_pred):
    return np.sum((y_true == 1) & (y_pred == 0))
def TP(y_true,y_pred):
    return np.sum((y_true == 1) & (y_pred == 1))
print(TN(y_test,y_pred))
print(FP(y_test,y_pred))
print(FN(y_test,y_pred))
print(TP(y_test,y_pred))
```

```
404
1
6
39
```

```
# 混淆矩阵的定义
def confusion_matrix(y_true,y_pred):
    return np.array([
        [TN(y_true,y_pred),FP(y_true,y_pred)],
        [FN(y_true,y_pred),TP(y_true,y_pred)]
    ])
print(confusion_matrix(y_test,y_pred))
```

```
[[404   1]
 [  6  39]]
```

```
# 精准率
def precision(y_true,y_pred):
    try:
        return TP(y_true,y_pred) / (FP(y_true,y_pred) + TP(y_true,y_pred))
    except:
        return 0.0
print(precision(y_test,y_pred))
```

```
0.975
```


7. 综合实例

7.2 Python计算评价指标

【例9】计算评价指标

```
# 召回率
def recall(y_true,y_pred):
    try:
        return TP(y_true,y_pred) / (FN(y_true,y_pred) + TP(y_true,y_pred))
    except:
        return 0.0
print(recall(y_test,y_pred))
```

```
0.8666666666666667
```


8. ROC曲线

8.1 认识ROC曲线

ROC全称是“受试者工作特征”（Receiver Operating Characteristic）曲线，用于描述混淆矩阵中FPR-TPR两个量之间的相对变化情况。ROC曲线的横轴是FPR（False Positive Rate，伪阳率），纵轴是TPR（True Positive Rate，真阳率）。ROC曲线用于描述样本的真实类别和预测概率，计算公式如下所示：

$$FPR = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{TP + FN}$$

8. ROC曲线

8.1 认识ROC曲线

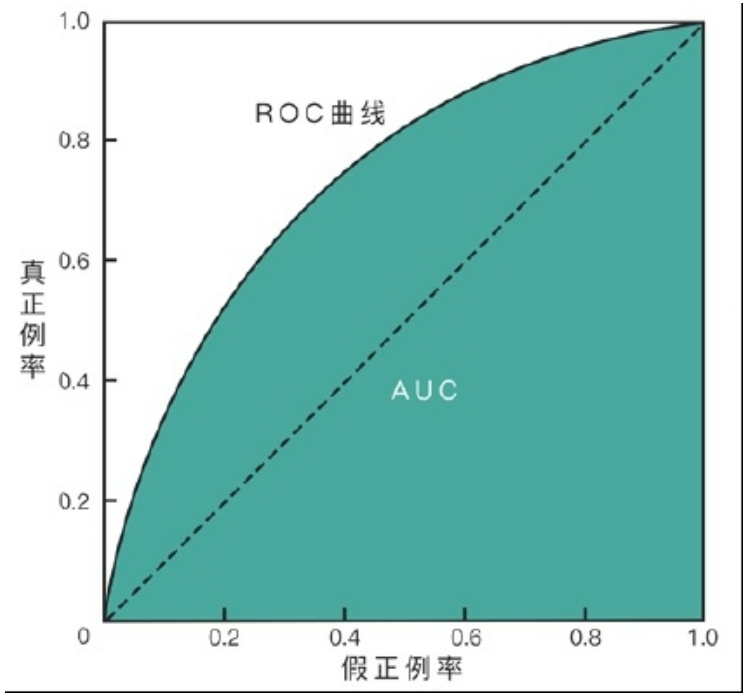
ROC曲线中的四个点和一条线如下：

点(0,1)：即 FP=0， TPR=1，意味着FN=0且 FP=0，将所有样本都正确分类。

点(1,0)：即FPR=1， TPR=0，最差分类器，避开了所有正确答案。

点(0,0)：即FPR=TPR=0，意味着FP=TP=0,将所有样本都预测为负类。

点(1,1)：即 FPR=TPR=1，分类器把所有样本都预测为正类。



8. ROC曲线

8.2 Sklearn计算ROC曲线

sklearn.metrics模块提供roc_curve()函数

sklearn.metrics.roc_curve(y_true, y_score)

参数	说明
y_true	每个样本的真实类别，0为（反例），1为（正例）
y_score	预测得分，可以是正类的估计概率

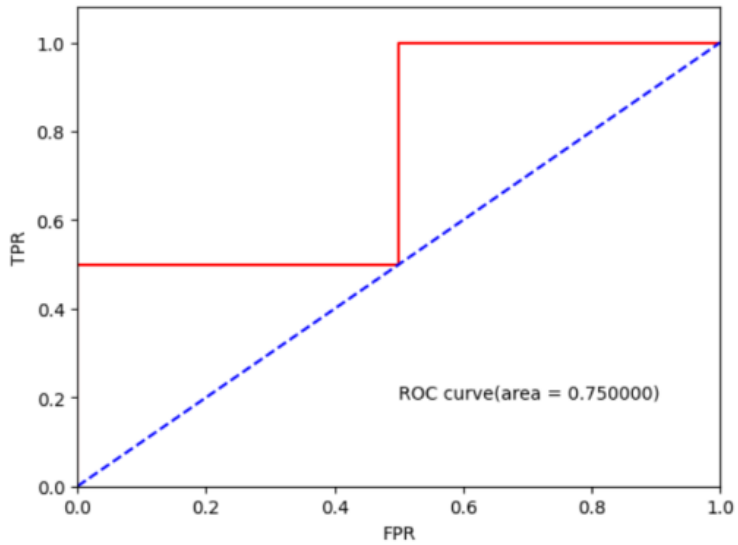
表8-1 roc_curve()参数说明

8. ROC曲线

8.2 Sklearn计算ROC曲线

【例10】roc_curve()举例

```
fpr: [0.  0.  0.5 0.5 1. ]
tpr: [0.  0.5 0.5 1.  1. ]
[1.8  0.8  0.4  0.35 0.1 ]
auc 0.75
```



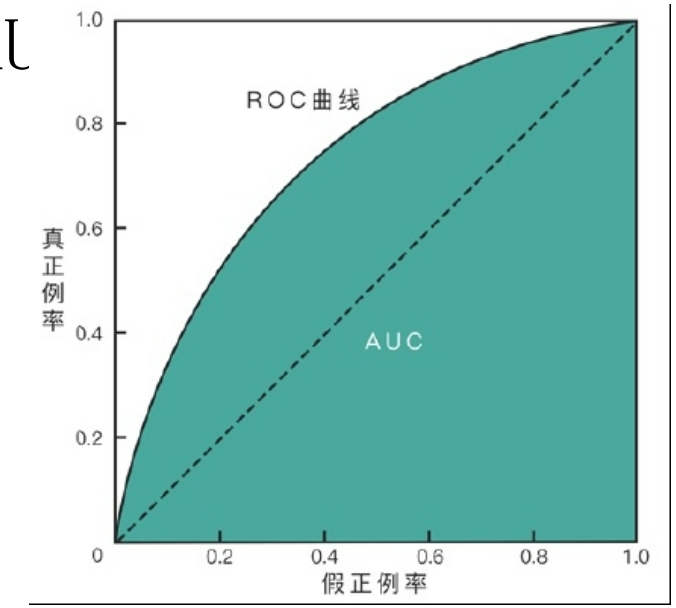

```python
# 例10:roc_curve()举例
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.metrics import roc_auc_score

# y_true = np.array([0,0,1,1])
# y_scores = np.array([0.1,0.4,0.35,0.8])

y_true = np.array([1,1,2,2])
y_scores = np.array([0.1,0.4,0.35,0.8])

#计算AUC
auc_test = roc_auc_score(y_true,y_scores)
```

```python
# 计算ROC
fpr,tpr,thresholds = metrics.roc_curve(y_true,y_scores,pos_label = 2)
print('fpr:',fpr)
print('tpr:',tpr)
print(thresholds)

plt.plot(fpr,tpr,color = 'red')
plt.plot([0,1],[0,1],color = 'blue',linestyle = '--')
plt.xlim(0.0,1.0)
plt.ylim(0.0,1.08)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.annotate(xy = (.4,.2),xytext = (.5,.2),text = 'ROC curve(area =% 02f)' %auc_test)

from sklearn.metrics import auc
print('auc',metrics.auc(fpr,tpr))
```


9. AUC面积

9.1 认识AUC面积

AUC(Area Under Curve)是指ROC曲线下的面积，由于ROC曲线一般都处于y=r直线上方，所以AUC的取值为0.5-1。AUC越接近1，检测方法真实性越高，当AU
实性最低，无应用价值。



9. AUC面积

9.2 Sklearn计算AUC面积

sklearn.metrics模块提供roc_auc_score()函数

sklearn.metrics.roc_auc_score(y_true,y_score)

参数	说明
y_true	每个样本的真实类别，0为（反例），1为（正例）
y_score	预测得分，可以是正类的估计概率

表9-1 roc_auc_score()参数说明

9. AUC面积

9.2 Sklearn计算AUC面积

【例11】roc_auc_score()举例

```
# 例11: roc_auc_score()举例
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0,0,1,1])
y_scores = np.array([0.1,0.4,0.35,0.8])
print(roc_auc_score(y_true,y_scores))
```

```
0.75
```


10. 分类评估报告

分类评估报告显示每个类的精确度、召回率、F1 Score等信息。

sklearn.metrics模块提供classification_report()函数

sklearn.metrics.classification_report(y_true,y_pred,labels,target_names)

参数	说明
y_true	真实目标值
y_pred	估计器预测目标值
labels	指定类别对应的数字
target_names	目标类别名称

表10-1 classification_report()参数说明

10. 分类评估报告

【例12】classification_report()举例

```
# 例12: classification_report()举例
from sklearn.metrics import classification_report
y_true = [0,1,2,2,2]
y_pred = [0,0,2,2,1]
target_names = ['class_0','class_1','class_2']
print(classification_report(y_true,y_pred,target_names = target_names))
```

```
              precision    recall  f1-score   support

     class_0       0.50      1.00      0.67         1
     class_1       0.00      0.00      0.00         1
     class_2       1.00      0.67      0.80         3

    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```


11. NLP评价指标

11.1 中文分词精确率和召回率

【例13】文本“武汉市长江大桥”进行中文分词的标准答案[‘武汉市’，‘长江大桥’]，现有两种分词结果，如表11-1所示

	分词结果	分词区间	
标准答案	[‘武汉市’,‘长江大桥’]	[1,2,3],[4,5,6,7]	A
分词结果1	[‘武汉’,‘市长’,‘江大桥’]	[1,2],[3,4],[5,6,7]	B’
重合部分	无	0	
分词结果2	[‘武汉’,‘市长’,‘江大桥’]	[1,2],[3,4],[5,6,7]	B’
重合部分	[‘武汉’,‘市长’,‘江大桥’]	[1,2],[3,4],[5,6,7]	

表11-1 两种分词结果

11. NLP评价指标

11.1 中文分词精确率和召回率

【例13】文本“武汉市长江大桥”进行中文分词的标准答案[‘武汉市’，‘长江大桥’]，现有两种分词结果，如表11-1所示

分析：分词结果与标准答案的交集为重合部分。对于分词结果1，由于没有重合部分，根据计算公式得到精确率和召回率均为0，分词结果1错误；对于分词结果2，重合部分为标准答案，精确率和召回率均为1，分词结果2正确。

11. NLP评价指标

11.1 中文分词精确率和召回率

【例14】中文文本“结婚的和尚未结婚的”进行中文分词的标准答案为［‘结婚’，‘的’，‘和’，‘尚未’，‘结婚’，‘的’］，分词结果如表11-2所示

	分词结果	分词区间	
标准答案	[‘结婚’,‘的’,‘和’,‘尚未’,‘结婚',‘的’]	[1,2],[3,3],[4,4],[5,6],[7,8],[9,9]	A
分词结果	[‘结婚’,‘的’,‘和尚’,‘未结婚',‘的’]	[1,2],[3,3],[4,5],[6,7,8],[9,9]	B
重合部分	[‘结婚’,‘的’,‘的’]	[1,2],[3,3],[9,9]	

表11-1 两种分词结果

11. NLP评价指标

11.1 中文分词精确率和召回率

【例14】中文文本“结婚的和尚未结婚的”进行中文分词的标准答案为[‘结婚’，‘的’，‘和’，‘尚未’，‘结婚’，‘的’]，分词结果如表11-2所示

根据精确率和召回率计算公式如下

精确率为： $3/5 = 0.6$

召回率为： $3/6 = 0.5$

11. NLP评价指标

11.2 未登录词和登录词召回率

IV是“登录词”(In Vocabulary)，也就是已经存在于字典中的词。IV Recall是IV的召回率，计算公式如下：

$$IV\ Recall = \frac{\text{重复词区间在词典中出现的词}}{\text{标准分词中在词典中出现的词}}$$

11. NLP评价指标

11.2 未登录词和登录词召回率

OOV是“未登录词”(Out Off Vocabulary)，也就是新词，是在已知词典中不存在的词。OOV Recall是OOV的召回率，计算公式如下：

$$OOV\ Recall = \frac{\text{重复词区间未在词典中出现的词}}{\text{标准分词中未在词典中出现的词}}$$

11. NLP评价指标

11.2 未登录词和登录词召回率

【例15】已知字符串为“结婚的和尚未结婚的都应该好好考虑一下人生大事”，词典为['结婚','尚未','的','和','青年','都','应该','好好考虑','自己','人生','大事']

（1）标准答案A

分词结果：['结婚',' 的',' 和',' 尚未',' 结婚 ','的',' 都',' 应该',' 好好',' 考虑',' 一下',' 人生','大事']

分词区间：[1,2],[3,3],[4,4],[5,6],[7,8],[9,9],[10,10],[11,12],[13,14],[15,16],[17,18],[19,20],[21,22]

（2）分词结果B

分词结果：['结婚',' 的','和尚','未结婚 ','的 ','都',' 应该',' 好好考虑',' 一下',' 人生大事']

分词区间：[1,2],[3,3],[4,5],[6,7,8],[9,9],[10,10],[11,12],[13,14,15,16],[17,18],[19,20,21,22]

11. NLP评价指标

11.2 未登录词和登录词召回率

【例15】已知字符串为“结婚的和尚未结婚的都应该好好考虑一下人生大事”，词典为['结婚','尚未','的','和','青年','都','应该','好好考虑','自己','人生','大事']

（3）重复词语 A∩B

分词结果：['结婚',' 的',' 的',' 都',' 应该',' 一下']

分词区间：[1,2], [3,3], [9,9],[10,10],[11,12],[17,18]

代入求得：$Recall = 6/10 = 0.6$　　$Precision = 6/13 = 0.4615$

$$F1 - core = \frac{2 \times 0.6 \times 0.4615}{0.6 + 0.4615} = 0.5217$$

11. NLP评价指标

11.2 未登录词和登录词召回率

【例15】已知字符串为“结婚的和尚未结婚的都应该好好考虑一下人生大事”，词典为['结婚','尚未','的','和','青年','都','应该','好好考虑','自己','人生','大事']

重复词区间在词典中出现的词=['结婚',' 的',' 的',' 都',' 应该']，个数为5；标准分词在词典中出现的词=['结婚',' 的',' 和',' 尚未',' 结婚 ','的',' 都',' 应该',' 人生','大事']，个数为10。因此，IV的召回率计算如下所示：

$$IV\ Recall = \frac{\text{重复词区间在词典中出现的词}}{\text{标准分词中在词典中出现的词}} = \frac{5}{10} = 0.5$$

重复词区间未在词典中出现的词=[' 一下']，个数为1；标准分词未在词典中出现的词=[' 好好',' 考虑',' 一下']，个数为3。因此，OOV的召回率计算如下所示：

$$\text{重复词区间未在词典中出现的词} \quad 1$$

11. NLP评价指标

11.2 未登录词和登录词召回率

【例15】代码

```python
# 例15：未登录词和登录词召回率
import re
def to_region(segmentation: str) -> list:
    #将分词结果转换为区间
    region = []
    start = 0
    for word in re.compile("\\s+").split(segmentation.strip()):
        end = start + len(word)
        region.append((start, end))
        start = end
    return region
```

```python
def prf(gold: str, pred: str, dic) -> tuple:
#计算OOV_R, IV_R
    A_size, B_size, A_cap_B_size, OOV, IV, OOV_R, IV_R = 0, 0, 0, 0, 0, 0, 0
    A, B = set(to_region(gold)), set(to_region(pred))
    A_size += len(A)
    B_size += len(B)
    A_cap_B_size += len(A & B)
    text = re.sub("\\s+", "", gold)

    for (start, end) in A:
        word = text[start: end]
        if word in dic:
            IV += 1
        else:
            OOV += 1
    for (start, end) in A & B:
        word = text[start: end]
        if word in dic:
            IV_R += 1
        else:
            OOV_R += 1
    p, r = A_cap_B_size / B_size * 100, A_cap_B_size / A_size * 100
    return p, r, 2 * p * r / (p + r), OOV_R / OOV * 100, IV_R / IV * 100
```


11. NLP评价指标

11.2 未登录词和登录词召回率

【例15】代码

```
if __name__ == '__main__':
# dic为词典
    dic = ['结婚', '尚未', '的', '和', '青年', '都', '应该', '好好考虑', '自己',  '人生', '大事']
    # gold为标准答案
    gold = '结婚 的 和 尚未 结婚 的 都 应该 好好 考虑 一下 人生 大事'
    # pred为分词结果
    pred = '结婚 的 和尚 未结婚 的 都 应该 好好考虑 一下 人生大事'
print("Precision:%.2f\n Recall:%.2f\n F1:%.2f\n OOV-R:%.2f\n IV-R:%.2f\n" % prf(gold, pred, dic))
```

```
Precision:60.00
 Recall:46.15
 F1:52.17
 OOV-R:33.33
 IV-R:50.00
```


Thank you!