Метрики качества

Задача классификации

Выборка: $X^l = \{x_1, \dots, x_l\}$

Ответы: $y_i \in \{0,1\}, i = 1, \dots, l$

Алгоритм: $a: X \to Y$

Задача: измерить качество алгоритма a на выборке X^l.

Accuracy

Доля правильных ответов на выборке:

$$\frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i]$$

Пример

- 950 объектов класса 0;
- 50 объектов класса 1;
- $a(x) = 0$ для всех x.

Доля правильных ответов $a(x)$: 95%

Базовая доля правильных ответов:

$$BaseRate = \arg\max_{y_0 \in \{0,1\}} \frac{1}{l} \sum_{i=1}^{l} [a(x_i) = y_i]$$

Матрица ошибок

	$y = 1$	$y = 0$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = 0$	False Negative (FN)	True Negative (TN)

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Пример

Задача медицинской диагностики ($y = 1$ – больные, $y = 0$ – здоровые)

	$y = 1$	$y = 0$
$a(x) = 1$	20	50
$a(x) = 0$	5	1000

Доля правильных ответов: 94.9%

	$y = 1$	$y = 0$
$a(x) = 1$	0	0
$a(x) = 0$	25	1050

Доля правильных ответов: 97.6%

Точность и полнота

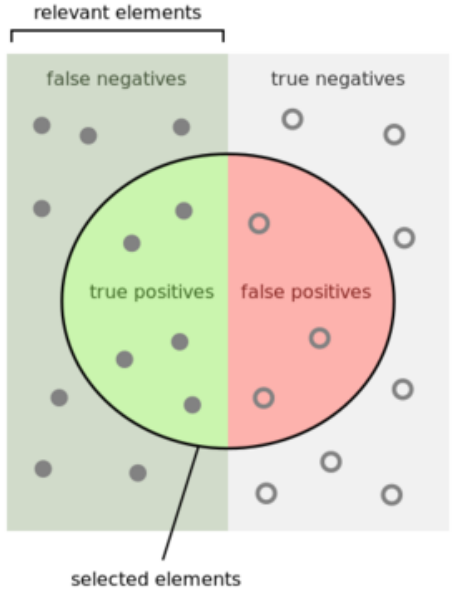
Точность (precision) — насколько можно доверять классификатору

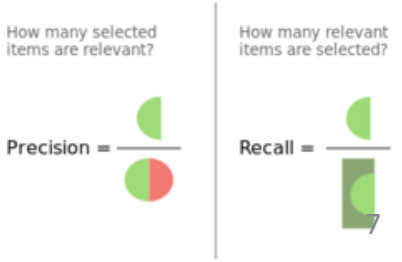
$$precision = \frac{TP}{TP + FP}$$

	$y = 1$	$y = 0$
$a(x) = 1$	20	50
$a(x) = 0$	5	1000

Точность классификатора: 28.6%

Точность константного классификатора: 0%



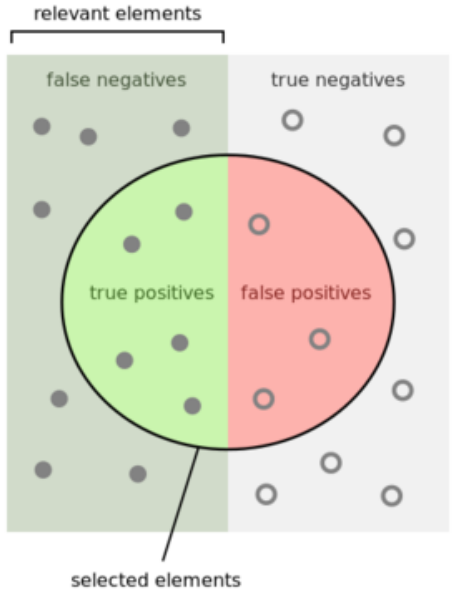


Точность и полнота

Полнота (recall) — как много объектов класса 1 находит классификатор

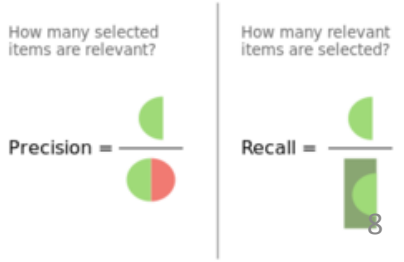
$$recall = \frac{TP}{TP + FN}$$

	$y = 1$	$y = 0$
$a(x) = 1$	20	50
$a(x) = 0$	5	1000



Точность классификатора: 80%

Точность константного классификатора: 0%



Пример

Пример 1: определение мошеннических действий на банковских счетах.

Важнее **полнота**: лучше проверить лишний раз, чем пропустить вредоносные действия

Пример 2: поиск вражеских самолетов для автоматического уничтожения ракетой

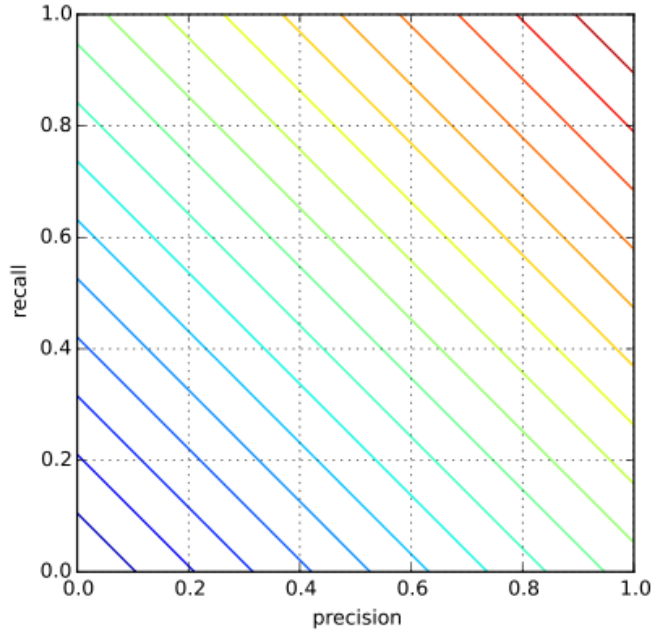
Важнее **точность**: нельзя допустить стрельбы по своему самолету.

Усреднение точности и полноты

$$A = \frac{1}{2}(precision + recall)$$

Если *precision* = 0.05, *recall* = 1, то A = 0.525.

Если *precision* = 0.525, *recall* = 0.525, то A = 0.525



Усреднение точности и полноты

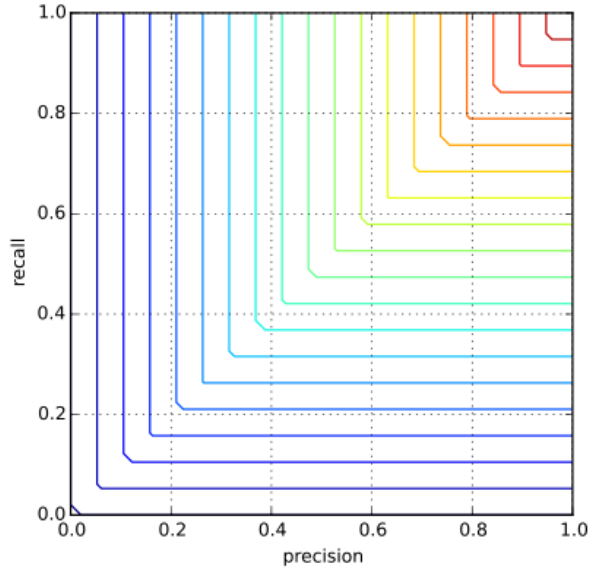
$$M = \min(precision, recall)$$

Если precision = 0.05, recall = 1, то M = 0.05.

Если precision = 0.525, recall = 0.525, то M = 0.525.

Если precision = 0.2, recall = 1, то M = 0.2.

Если precision = 0.2, recall = 0.3, то M = 0.2.



Усреднение точности и полноты

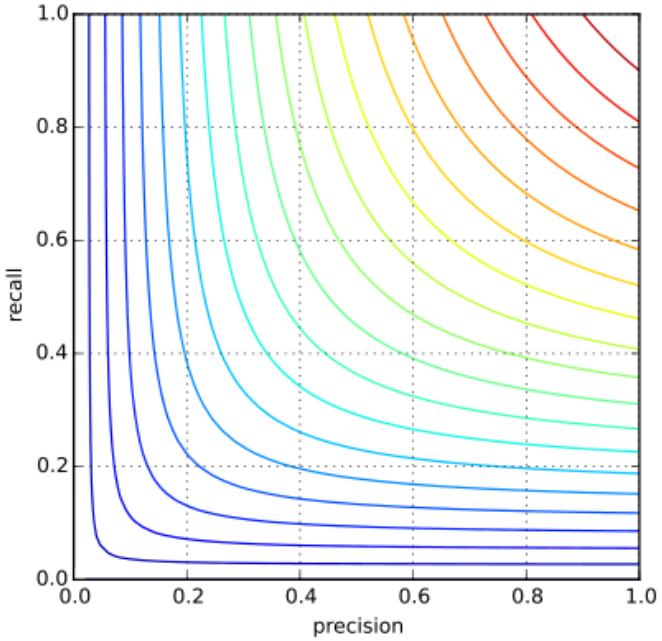
$$F = \frac{2 * percision * recall}{percision + recall}$$

Если precision = 0.05, recall = 1, то F = 0.1.

Если precision = 0.525, recall = 0.525, то F = 0.525.

Если precision = 0.2, recall = 1, то F = 0.33.

Если precision = 0.2, recall = 0.3, то F = 0.24.



Правило классификации

$$a(x) = [b(x) > t]$$

$b(x)$ - оценка принадлежности классу 1

t – порог классификации

Линейный классификатор

$$a(x) = [\langle w, x \rangle > 0]$$

Метод k ближайших соседей

$$a(x) = \left[\sum_{i=1}^{k} \left[y^{(i)} = 1\right] > k/2\right]$$

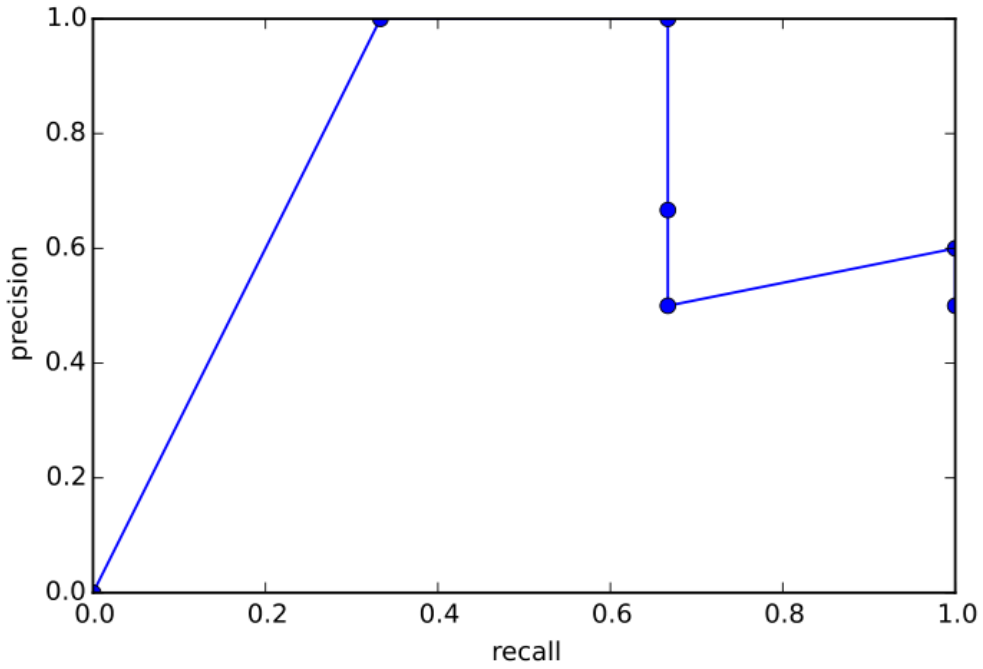
PR-кривая

1. Сортировка объектов по возрастанию оценки $b(x)$
2. Перебор порогов классификации, начиная с максимального:

$$t_l = b\left(x_{(l)}\right), \ldots t_1 = b\left(x_{(1)}\right)$$

3. Подсчёт для каждого порога точности и полноты
4. Отображение соответствующей точки в осях «полнота-точность»

Пример



$b(x)$	0.14	0.23	0.39	0.52	0.73	0.90
y	0	1	0	0	1	1

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

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

	$y = 1$	$y = 0$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = 0$	False Negative (FN)	True Negative (TN)

PR-кривая

1) Если выборка идеально разделима, то кривая пройдет через точку (1,1);

2) Чем больше площадь под кривой, тем лучше.

AUC-PRC (Area Under Precision-Recall curve) — мера качества для $b(x)$.

ROC-кривая

ROC — «reciever operating characteristic»

1. По оси *X*: False Positive Rate, доля ошибочных положительных классификаций:

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

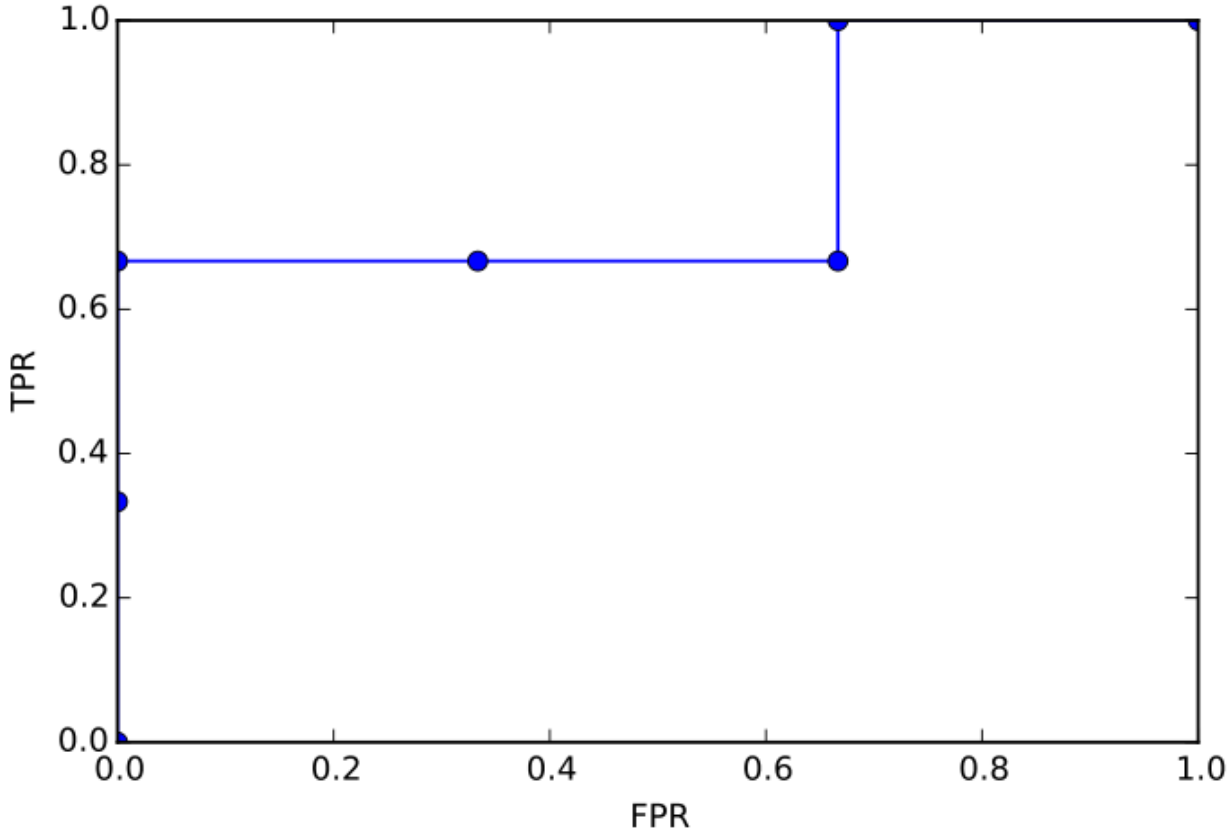
FPR называется специфичностью алгоритма.

2. По оси *Y*: True Positive Rate, доля правильных положительных классификаций:

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

TPR называется чувствительностью алгоритма.

Пример



$b(x)$	0.14	0.23	0.39	0.52	0.73	0.90
y	0	1	0	0	1	1

Свойства

1. Левая точка: всегда (0,0) (все объекты относим к классу 0);
2. Правая точка: всегда (1,1) (все объекты относим к классу 1);
3. Если выборка идеально разделима, то кривая пройдет через точку (1,0);
4. Площадь меняется от 1/2 до 1;
5. Чем больше площадь под кривой, тем лучше.

AUC-ROC (Area Under ROC-curve) — мера качества для $b(x)$.

ROC-кривая

$$TPR = \frac{TP}{TP + FN} \; FPR = \frac{FP}{FP + TN}$$

1. Метрики качества нормируются на размеры классов, ROC-кривая не изменится при перемене соотношения классов.

2. Имеет проблемы при сильном дисбалансе классов.

PR-кривая

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

1. Точность нормируется на число положительных прогнозов, изменится при перемене соотношения классов.

2. Максимально возможная площадь под PR-кривой зависит от соотношения классов.

3. Хорошо подходит для измерения качества при сильном дисбалансе классов

Пример

Алгоритм 1:

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

$$FPR = \frac{FP}{FP + TN} = \frac{10}{10 + 999890} = 0.00001$$

Алгоритм 2:

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

$$FPR = \frac{FP}{FP + TN} = \frac{1910}{1910 + 997990} = 0.00191$$

Алгоритм 1:

$$precision = \frac{TP}{TP + FP} = 90/(90 + 10) = 0.9$$

$$recall = \frac{TP}{TP + FN} = 90/(90 + 10) = 0.9$$

Алгоритм 2:

$$precision = \frac{TP}{TP + FP} = \frac{90}{90 + 1910} = 0.045$$

$$recall = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

Взвешенная точность (weighted accuracy)

We will first consider evaluation measures for classification in the general case of two or more classes (i.e., $M \geq 2$) [1]. The most common measure is the probability that an instance of the test set is classified correctly. This is usually referred to as (weighted) accuracy *WA*, or weighted average recall or recognition rate.

$$\begin{aligned} \text{WA} &= \frac{\#\,\text{correctly classified test instances}}{\#\,\text{test instances}} \\ &= \frac{\sum_{i=1}^{M} \left|\left\{\underline{x} \in \mathcal{T}_i \mid \hat{y} = i\right\}\right|}{|\mathcal{T}|}. \end{aligned} \tag{7.86}$$

If this rate is given per class i, one speaks of the class-specific recall RE_i:

$$\text{RE}_i = \frac{\left|\left\{\underline{x} \in \mathcal{T}_i \mid \hat{y} = i\right\}\right|}{T_i}. \tag{7.87}$$

With $p_i = T_i/|\mathcal{T}|$ as the prior probability of class i in the test set further holds:

$$\text{WA} = \sum_{i=1}^{M} p_i \,\text{RE}_i. \tag{7.88}$$

Невзвешенная точность (unweighted accuracy)

If balance of instances among classes is (highly) unbalanced, one can prefer to exchange the priors p_i for all classes by the constant weight $\frac{1}{M}$. This is known as unweighted accuracy UA or unweighted average recall:

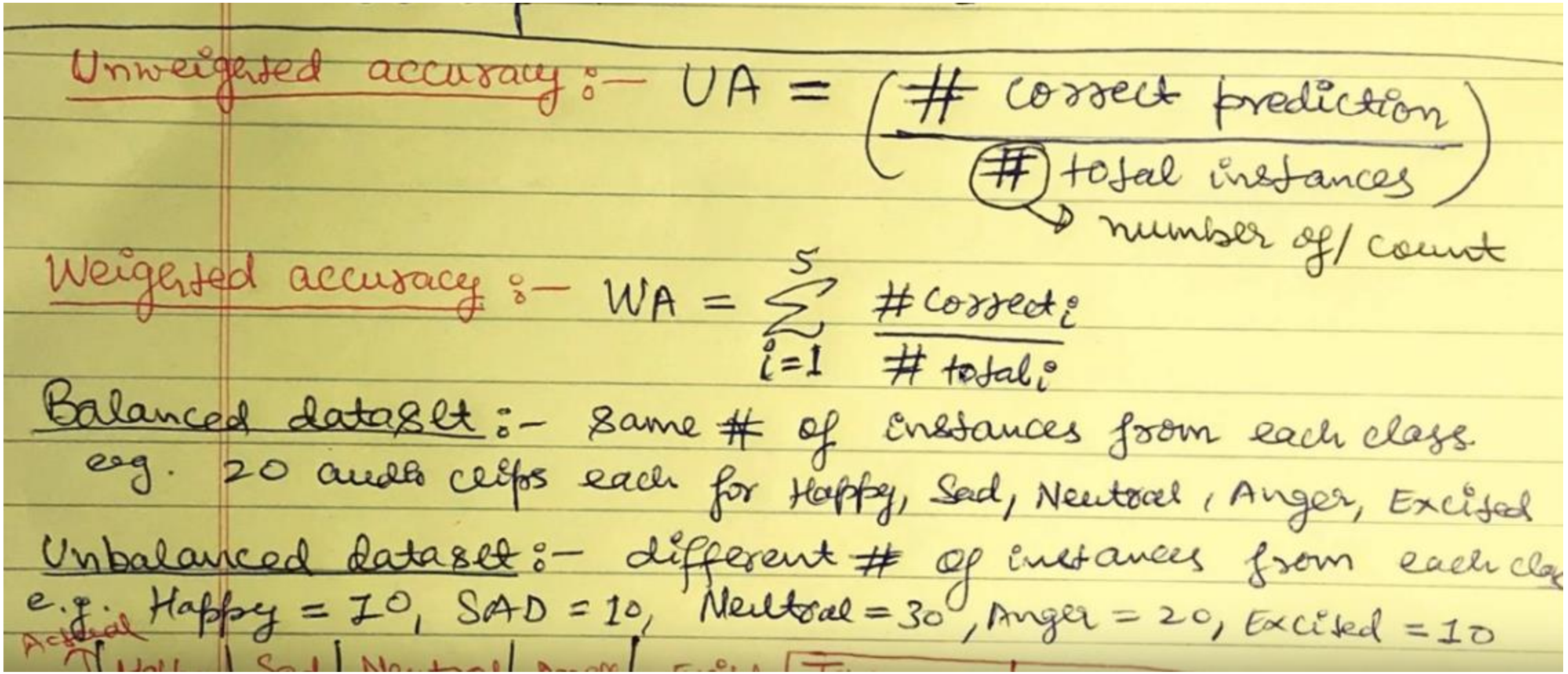
$$\mathrm{UA} = \frac{\sum_{i=1}^{M} \mathrm{RE}_i}{M}. \tag{7.90}$$

Weighted vs Unweighted accuracy

Weighted accuracy is computed by taking the average, over all the classes, of the fraction of correct predictions in this class (i.e. the number of correctly predicted instances in that class, divided by the total number of instances in that class). Unweighted accuracy is the fraction of instances predicted correctly (i.e. total correct predictions, divided by total instances).

The distinction between these two measures is useful especially if there exist classes that are under-represented by the samples. Unweighted accuracy gives the same weight to each class, regardless of how many samples of that class the dataset contains. Weighted accuracy weighs each class according to the number of samples that belong to that class in the dataset.

Weighted vs Unweighted accuracy



```python
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
import numpy as np

iris = datasets.load_iris()
X = iris.data
y = iris.target

# Add noisy features
random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

# Limit to the two first classes, and split into training and test
X_train, X_test, y_train, y_test = train_test_split(X[y < 2], y[y < 2],
                                                    test_size=.5,
                                                    random_state=random_state)

# Create a simple classifier
classifier = svm.LinearSVC(random_state=random_state)
classifier.fit(X_train, y_train)
y_score = classifier.decision_function(X_test)
```

PR Python

2 класса

```python
from sklearn.metrics import average_precision_score
average_precision = average_precision_score(y_test, y_score)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))
```

Compute average precision (AP) from prediction scores

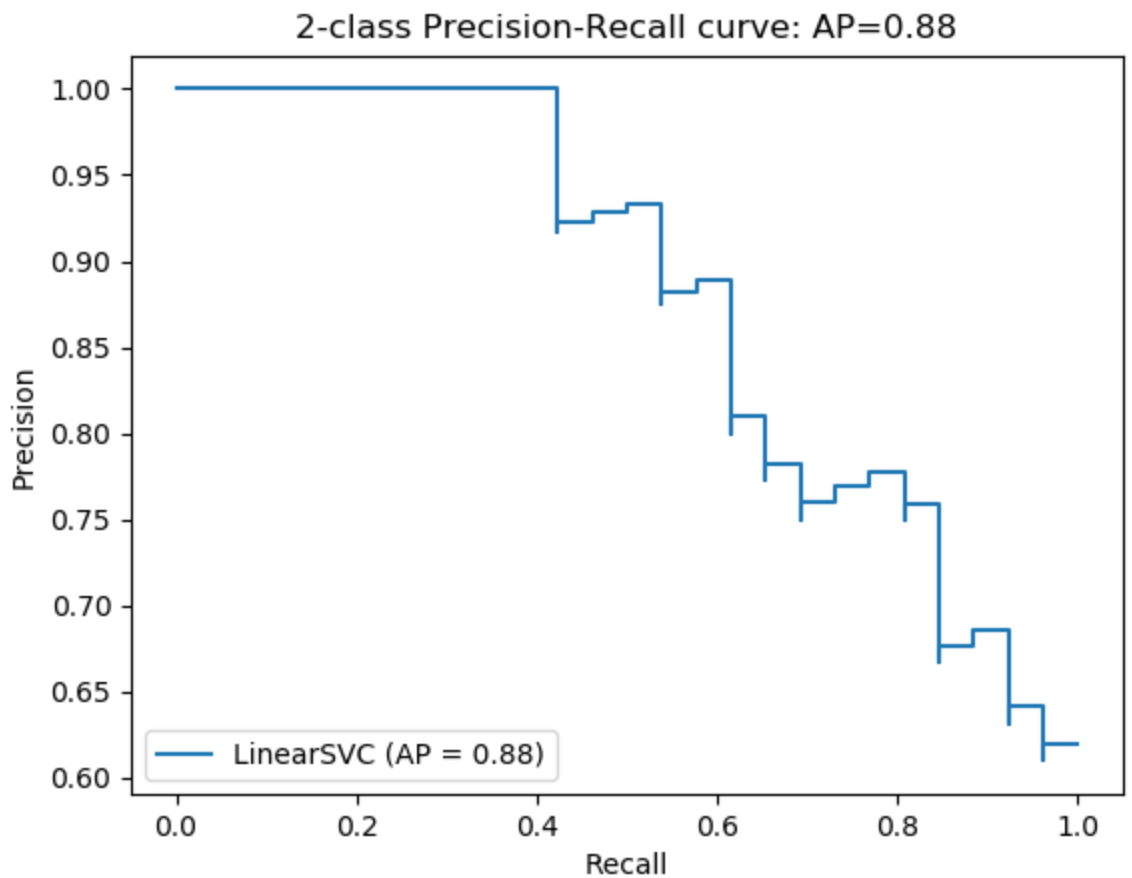
AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are the precision and recall at the nth threshold [1]. This implementation is not interpolated and is different from computing the area under the precision-recall curve with the trapezoidal rule, which uses linear interpolation and can be too optimistic.

Note: this implementation is restricted to the binary classification task or multilabel classification task.

```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
import matplotlib.pyplot as plt

disp = plot_precision_recall_curve(classifier, X_test, y_test)
disp.ax_.set_title('2-class Precision-Recall curve: '
                   'AP={0:0.2f}'.format(average_precision))
```



```python
from sklearn.preprocessing import label_binarize

# Use label_binarize to be multi-label like settings
Y = label_binarize(y, classes=[0, 1, 2])
n_classes = Y.shape[1]

# Split into training and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.5,
                                                    random_state=random_state)

# We use OneVsRestClassifier for multi-label prediction
from sklearn.multiclass import OneVsRestClassifier

# Run classifier
classifier = OneVsRestClassifier(svm.LinearSVC(random_state=random_state))
classifier.fit(X_train, Y_train)
y_score = classifier.decision_function(X_test)
```

```python
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score

# For each class
precision = dict()
recall = dict()
average_precision = dict()
for i in range(n_classes):
    precision[i], recall[i], _ = precision_recall_curve(Y_test[:, i],
                                                        y_score[:, i])
    average_precision[i] = average_precision_score(Y_test[:, i], y_score[:, i])

# A "micro-average": quantifying score on all classes jointly
precision["micro"], recall["micro"], _ = precision_recall_curve(Y_test.ravel(),
    y_score.ravel())
average_precision["micro"] = average_precision_score(Y_test, y_score,
                                                     average="micro")
print('Average precision score, micro-averaged over all classes: {0:0.2f}'
      .format(average_precision["micro"]))
```

PR Python 3 класса

```
In [6]: Y

Out[6]: array([[1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
               [1, 0, 0],
```

numpy.ravel

numpy.**ravel**(*a, order='C'*)

Return a contiguous flattened array.

It is equivalent to `reshape(-1, order=order)`.

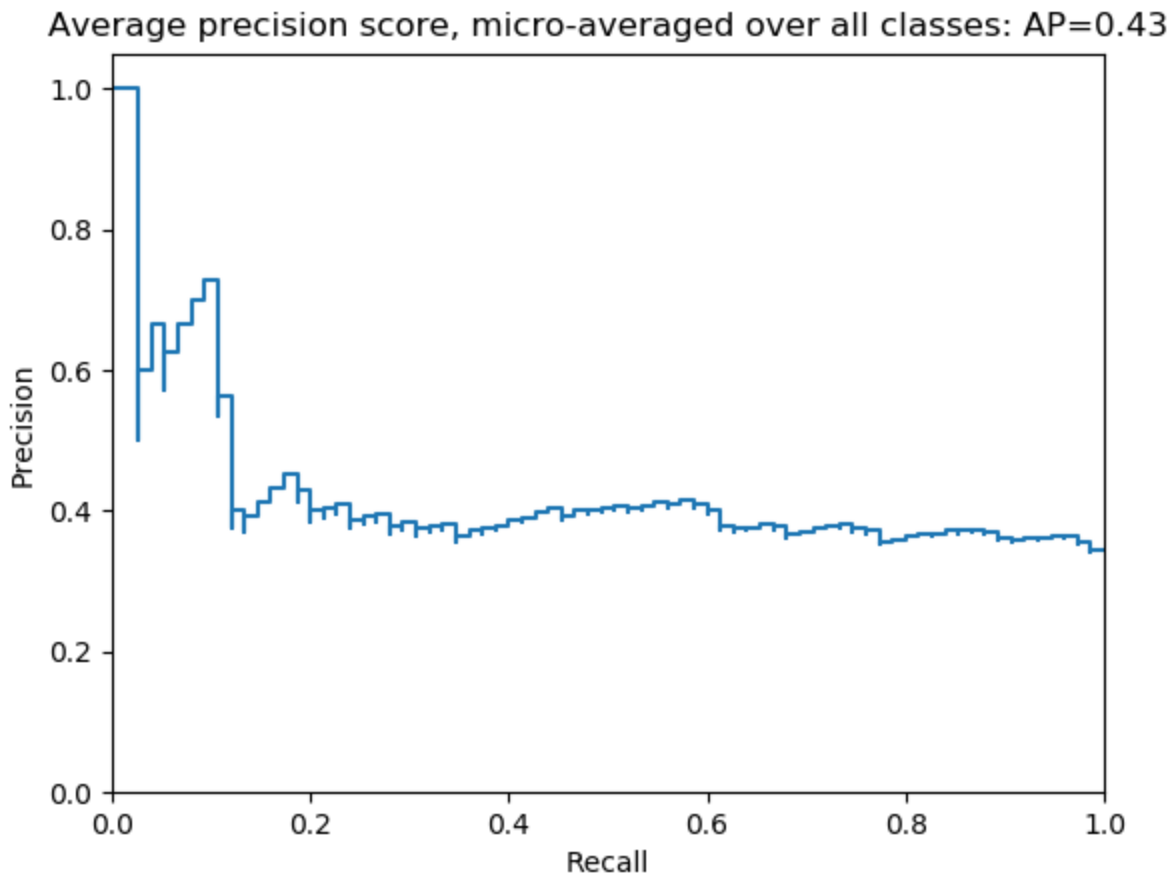
```
>>> x = np.array([[1, 2, 3], [4, 5, 6]])
>>> np.ravel(x)
array([1, 2, 3, 4, 5, 6])
```

```
>>> x.reshape(-1)
array([1, 2, 3, 4, 5, 6])
```

```
>>> np.ravel(x, order='F')
array([1, 4, 2, 5, 3, 6])
```

После бинаризации получаем только два класса: 0 и 1

```
plt.figure()
plt.step(recall['micro'], precision['micro'], where='post')

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title(
    'Average precision score, micro-averaged over all classes: AP={0:0.2f}'
    .format(average_precision["micro"]))
```



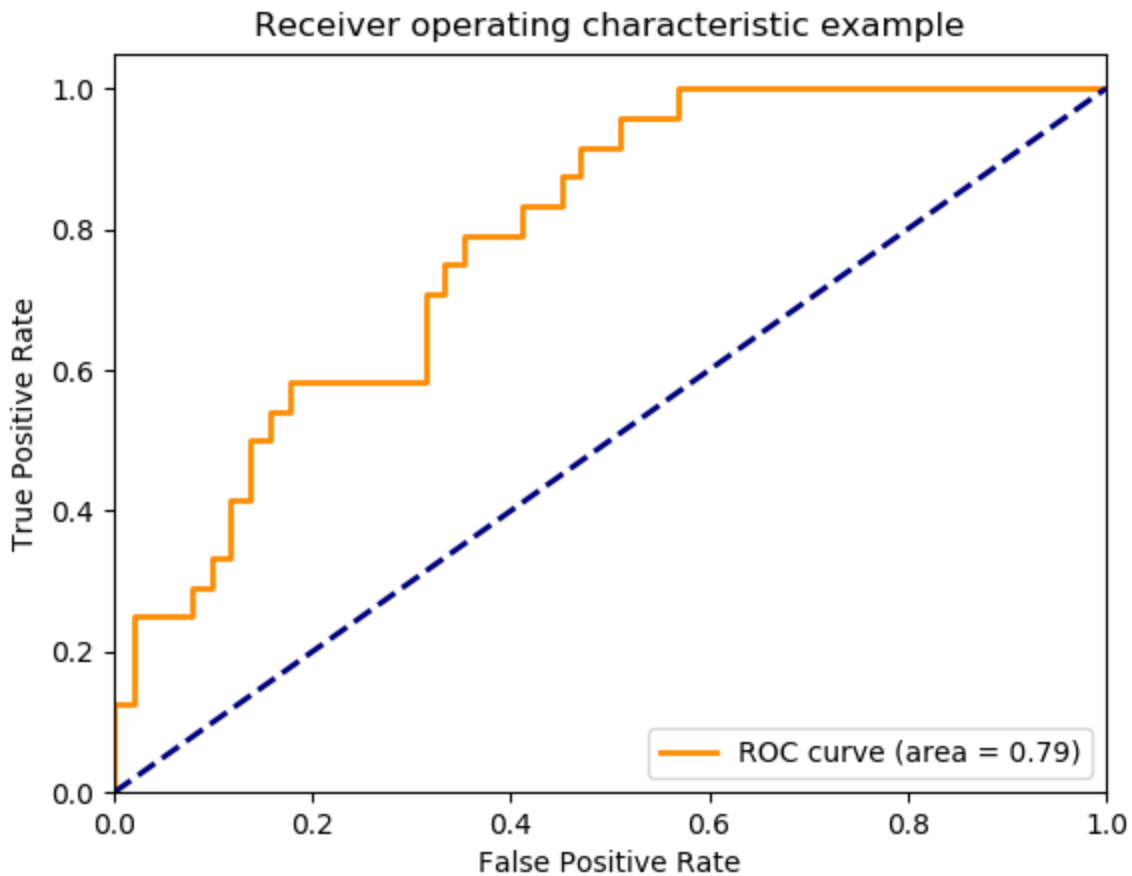
ROC Python

2 класса

```
# Import some data to play with
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Binarize the output
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]

# Add noisy features to make the problem harder
random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

# shuffle and split training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5,
                                                    random_state=0)

# Learn to predict each class against the other
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True,
                                 random_state=random_state))
y_score = classifier.fit(X_train, y_train).decision_function(X_test)

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

Plot of a ROC curve for a specific class

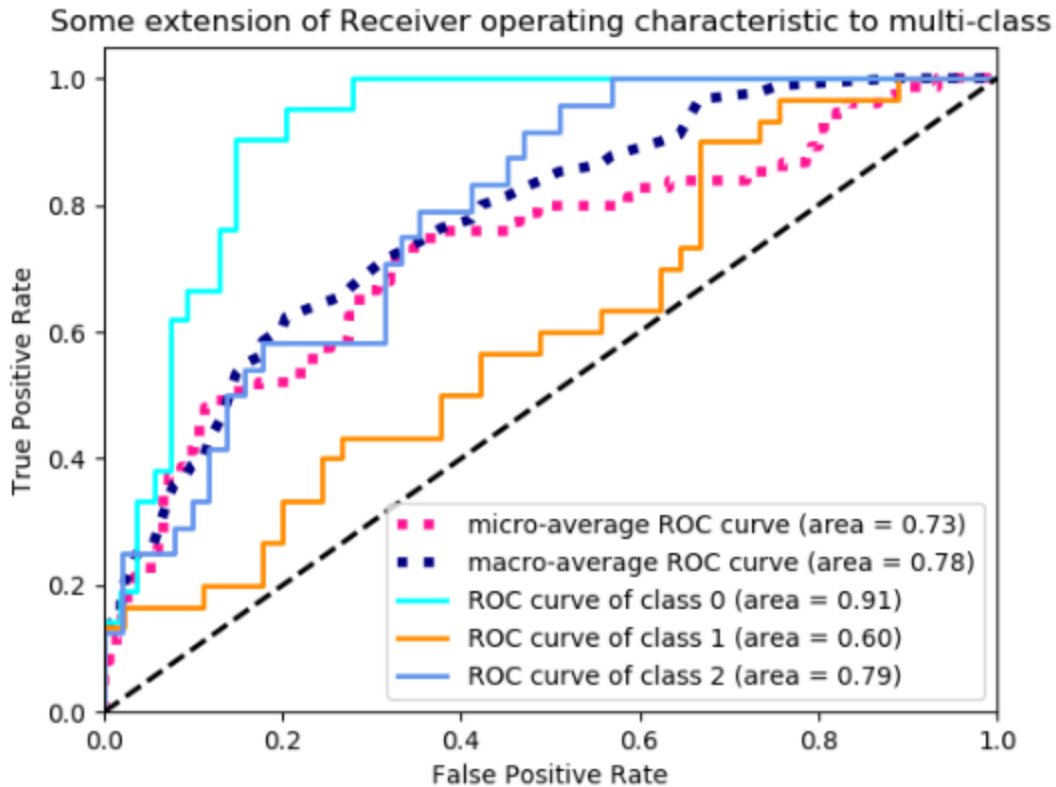
```
plt.figure()
lw = 2
plt.plot(fpr[2], tpr[2], color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.79)

```
# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```

ROC Python

3 класса


Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.73)
macro-average ROC curve (area = 0.78)
ROC curve of class 0 (area = 0.91)
ROC curve of class 1 (area = 0.60)
ROC curve of class 2 (area = 0.79)

Ссылки

- https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
- https://stackoverflow.com/questions/56090541/how-to-plot-precision-and-recall-of-multiclass-classifier
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html#sklearn.metrics.average_precision_score
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.label_binarize.html#sklearn.preprocessing.label_binarize
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#area-under-roc-for-the-multiclass-problem

Sklearn examples

PR: http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

ROC: http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

Fscore: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

Ссылки

Метрики оценки качества для различных задач:

http://scikit-learn.org/stable/modules/model_evaluation.html

Метрики оценки качества Kaggle:

https://www.kaggle.com/wiki/Metrics