precision_score

sklearn.metrics.precision_score(*y_true*, *y_pred*, ***, *labels=None*, *pos_label=1*, *average='binary'*, *sample_weight=None*, *zero_division='warn'*) [source]

Compute the precision.

The precision is the ratio `tp / (tp + fp)` where `tp` is the number of true positives and `fp` the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The best value is 1 and the worst value is 0.

Support beyond term:`binary` targets is achieved by treating multiclass and multilabel data as a collection of binary problems, one for each label. For the binary case, setting `average='binary'` will return precision for `pos_label`. If `average` is not `'binary'`, `pos_label` is ignored and precision for both classes are computed, then averaged or both returned (when `average=None`). Similarly, for multiclass and multilabel targets, precision for all `labels` are either returned or averaged depending on the `average` parameter. Use `labels` specify the set of labels to calculate precision for.

Read more in the User Guide.

Parameters:

y_true : ***1d array-like, or label indicator array / sparse matrix***

Ground truth (correct) target values.

y_pred : ***1d array-like, or label indicator array / sparse matrix***

Estimated targets as returned by a classifier.

labels : ***array-like, default=None***

The set of labels to include when `average != 'binary'`, and their order if `average is None`. Labels present in the data can be excluded, for example in multiclass classification to exclude a “negative class”. Labels not present in the data can be included and will be “assigned” 0 samples. For multilabel targets, labels are column indices. By default, all labels in `y_true` and `y_pred` are used in sorted order.

> ! *Changed in version 0.17:* Parameter `labels` improved for multiclass problem.

pos_label : ***int, float, bool or str, default=1***

The class to report if `average='binary'` and the data is binary, otherwise this parameter is ignored. For multiclass or multilabel targets, set `labels=[pos_label]` and `average != 'binary'` to report metrics for one label only.

average : ***{'micro', 'macro', 'samples', 'weighted', 'binary'} or None, default='binary'***

This parameter is required for multiclass/multilabel targets. If `None`, the metrics for each class are returned. Otherwise, this determines the type of averaging performed on the data:

`'binary'`:

Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

`'micro'`:

Calculate metrics globally by counting the total true positives, false negatives and false positives.

`'macro'`:

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

`'weighted'`:

Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

`'samples'`:

Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from `accuracy_score`).

sample_weight : ***array-like of shape (n_samples,), default=None***

Sample weights.

zero_division : ***{"warn", 0.0, 1.0, np.nan}, default="warn"***

Sets the value to return when there is a zero division.

Notes:

- If set to "warn", this acts like 0, but a warning is also raised.
- If set to `np.nan`, such values will be excluded from the average.

Added in version 1.3: `np.nan` option was added.

Returns:

precision : *float (if average is not None) or array of float of shape (n_unique_labels,)*

Precision of the positive class in binary classification or weighted average of the precision of each class for the multiclass task.

See also

`precision_recall_fscore_support`

Compute precision, recall, F-measure and support for each class.

`recall_score`

Compute the ratio `tp / (tp + fn)` where `tp` is the number of true positives and `fn` the number of false negatives.

`PrecisionRecallDisplay.from_estimator`

Plot precision-recall curve given an estimator and some data.

`PrecisionRecallDisplay.from_predictions`

Plot precision-recall curve given binary class predictions.

`multilabel_confusion_matrix`

Compute a confusion matrix for each class or sample.

Notes

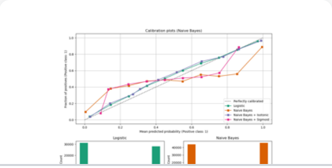
When `true positive + false positive == 0`, precision returns 0 and raises `UndefinedMetricWarning`. This behavior can be modified with `zero_division`.

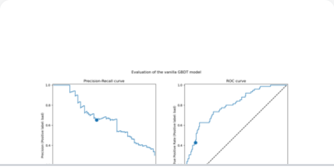
Examples

```
>>> import numpy as np
>>> from sklearn.metrics import precision_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> precision_score(y_true, y_pred, average='macro')
0.22...
>>> precision_score(y_true, y_pred, average='micro')
0.33...
>>> precision_score(y_true, y_pred, average='weighted')
0.22...
>>> precision_score(y_true, y_pred, average=None)
array([0.66..., 0.        , 0.        ])
>>> y_pred = [0, 0, 0, 0, 0, 0]
>>> precision_score(y_true, y_pred, average=None)
array([0.33..., 0.        , 0.        ])
>>> precision_score(y_true, y_pred, average=None, zero_division=1)
array([0.33..., 1.        , 1.        ])
>>> precision_score(y_true, y_pred, average=None, zero_division=np.nan)
array([0.33...,        nan,        nan])
```

```
>>> # multilabel classification
>>> y_true = [[0, 0, 0], [1, 1, 1], [0, 1, 1]]
>>> y_pred = [[0, 0, 0], [1, 1, 1], [1, 1, 0]]
>>> precision_score(y_true, y_pred, average=None)
array([0.5, 1. , 1. ])
```

Gallery examples


Calibration plots (Naive Bayes)
Fraction of positives (Positive class: 1)
Mean predicted probability (Positive class: 1)
Perfectly calibrated
Logistic
Naive Bayes
Naive Bayes + Isotonic
Naive Bayes + Sigmoid
Logistic
Naive Bayes
Count


Evaluation of the vanilla GBDT model
Precision-Recall curve
ROC curve
Precision (Positive label: bad)
True Positive Rate (Positive label: bad)