1. Accuracy

It gives the ratio to the number of items that have been predicted/classified correctly to the number of times the items have been predicted/classified in total. It basically tells how many times the algorithm predicted the output correctly. In mathematics, we use the following equation to calculate it:

**Accuracy = (items predicted/classified correctly / total number of items predicted)**

**= (true positives + true negatives) / (true positives + true negatives + false positives + false negatives)**

The definition for true positive, true negative, false positive and false negative has been provided in the section below.

**sklearn** package has a function named accuracy\_score which can be used to calculate the accuracy:

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_test, y\_pred))

2. Confusion Matrix

It helps measure how well a classifier performed when tested on real data. The name basically gives a notion of how many times the classifier got confused before arriving at various different solutions amongst which some were correct and some were incorrect. The confusion matrix is usually a **2 x 2 matrix**. The entries of the confusion matrix depict the number of times each class of the dataset occured in the question.

To construct a sample confusion matrix, let us consider the following example:

Let us consider a classifier that predicts whether India will win a certain cricket match or not. The following would be the confusion matrix for the same:

|  |  |  |
| --- | --- | --- |
|  | **Predicted: Won** | **Predicted: Lost** |
| **Actual: Won** | True Positives | False Positives |
| **Actual: Lost** | False Negatives | True Negatives |

In the above matrix, the **columns** are those which have been predicted by the classifier. On the other hand, the **rows** are the actual classes of the dataset.

Before you start getting confused about the entries in the confusion matrix, read ahead:

**True Positives (TP):** When the classifier **correctly** predicted that India would win and India did win.

**True Negatives (TN):** When the classifier **correctly** predicted that India would not win and India didn't win.

**False Positives (FP):** When the classifier **incorrectly** predicted that India would not win but India ended up winning the match.

**False Negatives (FN):** When the classifier **incorrectly** predicted that India would win but India ended up losing the match.

Instead of memorizing the above said terms, understand them in a simple way:

The **true** in the "true positive" and "true negative" basically tells whether the classifier predicted the output correctly or not. The **negatives** and **positives** in the above expressions tells whether the classifier predicted the positive outcome or the negative outcome.

**sklearn** has a confusion\_matrix function, which can be used as follows:

from sklearn.metrics import confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

Here y\_test are the test values and y\_pred are the values that have been predicted by the classifier.

## 3. Precision

Precision tells the number of the predicted positive values which are actually correct. This metric is used when the objective is to **reduce the number of false positives** in the confusion matrix.

**precision = (true positives) / (true positives + false positives)**

## 4. Recall

Recall is a metric that tells the frequency of the correct predictions that are positive values. This metric is used when the objective is to **reduce the number of false negatives** in the confusion matrix. It is also known as "**sensitivity**" or the "**true positive rate**" (TPR).

**recall = (true positives) / (true positives + false negatives)**

5. F1-score

The F1 score is the **harmonic mean of recall and precision**. This metric is used when precision and recall are both used as metrics in analysing a model's performance. There should be a careful balance between precision and recall.

When we try to optimize recall, then the algorithm ends up predicting outputs which belong to positive class but also predicts too many false positives, consequently leading to low precision.

On the other hand, if we try to optimize precision, then the algorithm ends up predicting very few positive results (those that have the highest probability of being positive) and the recall would be a very low value.

**F1 score = 2 \* (( precision \* recall ) / ( precision + recall ))**

The function, classification\_report, which can be found in the **sklearn.metrics** package, gives the precision, recall and f1-score

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

The output would be:

precision recall f1-score support

0 0.81 0.88 0.84 241

1 0.78 0.68 0.73 152

accuracy 0.80 393

precision recall f1-score support

macro avg 0.80 0.78 0.79 393

weighted avg 0.80 0.80 0.80 393

6. ROC curve (Receiver Operating Characteristic Curve)

This is a visual way of measuring the performance of a binary classifier. It is the ratio of true positive rate (recall or TPR) and false positive rate (FPR).

False positive rate is the metric which tells how often it predicts the negative result incorrectly. It can be expressed as following:

**FPR = ((False positives) / (false positive + true negative))**

ROC curve shows how the **recall versus precision relationship** changes when the threshold value is varied in the classifier. Threshold here refers to the data points which are above a certain limit, and are considered as positive. TPR is plotted on the y-axis and FPR is plotted on the x-axis.

In sklearn, ROC curve can be expressed as follows:

from sklearn.metrics import roc\_curve

import matplotlib.pyplot as plt

%matplotlib inline

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_prob)

7. AUC (Area under curve)

This metric is used to find the area under the ROC curve. This value is usually between 0 and 1, wherein a value closer to 1 or 1 itself means that the model provides a very good classification performance.

It can be implemented in sklearn.metric in the following way:

from sklearn.metrics import roc\_auc\_score

roc\_auc\_score(y\_test, y\_pred\_prob)

A larger area under the curve indicates that the algorithm gives high recall and precision values. This area is also known as average precision and can be visualized using the following code:

from sklearn.metrics import average\_precision\_score

average\_precision\_score(y\_test, y\_pred\_prob)

8. Precision-recall curve

Balancing the precision recall value can be a tricky task. This trade-off can be represented using the precision-recall curve.

In sklearn.metrics, it can be represented as follows:

from sklearn.metrics import precision\_recall\_curve

precision, recall, thresholds = precision\_recall\_curve(y\_test, y\_pred\_prob)

Link - https://www.studytonight.com/post/how-good-is-my-machine-learning-model-how-do-i-improve-its-performance