Download from finelybook www.finelybook.com If you are confused about the confusion matrix, Figure 3-2 may help.

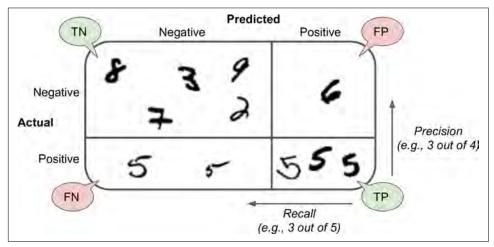


Figure 3-2. An illustrated confusion matrix

Precision and Recall

Scikit-Learn provides several functions to compute classifier metrics, including precision and recall:

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision score(y train 5, y pred) # == 4344 / (4344 + 1307)
0.76871350203503808
>>> recall score(y train 5, y train pred) # == 4344 / (4344 + 1077)
0.79136690647482011
```

Now your 5-detector does not look as shiny as it did when you looked at its accuracy. When it claims an image represents a 5, it is correct only 77% of the time. Moreover, it only detects 79% of the 5s.

It is often convenient to combine precision and recall into a single metric called the F_1 *score*, in particular if you need a simple way to compare two classifiers. The F₁ score is the harmonic mean of precision and recall (Equation 3-3). Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values. As a result, the classifier will only get a high F₁ score if both recall and precision are high.

Equation 3-3. F_1 score

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

Download from finelybook www.finelybook.com To compute the F₁ score, simply call the f1_score() function:

```
>>> from sklearn.metrics import f1 score
>>> f1_score(y_train_5, y_pred)
0.78468208092485547
```

The F_1 score favors classifiers that have similar precision and recall. This is not always what you want: in some contexts you mostly care about precision, and in other contexts you really care about recall. For example, if you trained a classifier to detect videos that are safe for kids, you would probably prefer a classifier that rejects many good videos (low recall) but keeps only safe ones (high precision), rather than a classifier that has a much higher recall but lets a few really bad videos show up in your product (in such cases, you may even want to add a human pipeline to check the classifier's video selection). On the other hand, suppose you train a classifier to detect shoplifters on surveillance images: it is probably fine if your classifier has only 30% precision as long as it has 99% recall (sure, the security guards will get a few false alerts, but almost all shoplifters will get caught).

Unfortunately, you can't have it both ways: increasing precision reduces recall, and vice versa. This is called the precision/recall tradeoff.

Precision/Recall Tradeoff

To understand this tradeoff, let's look at how the SGDClassifier makes its classification decisions. For each instance, it computes a score based on a decision function, and if that score is greater than a threshold, it assigns the instance to the positive class, or else it assigns it to the negative class. Figure 3-3 shows a few digits positioned from the lowest score on the left to the highest score on the right. Suppose the decision threshold is positioned at the central arrow (between the two 5s): you will find 4 true positives (actual 5s) on the right of that threshold, and one false positive (actually a 6). Therefore, with that threshold, the precision is 80% (4 out of 5). But out of 6 actual 5s, the classifier only detects 4, so the recall is 67% (4 out of 6). Now if you raise the threshold (move it to the arrow on the right), the false positive (the 6) becomes a true negative, thereby increasing precision (up to 100% in this case), but one true positive becomes a false negative, decreasing recall down to 50%. Conversely, lowering the threshold increases recall and reduces precision.