Classifier performance metrics

Fraida Fund

Suppose we have a series of data points $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$ and there is some (unknown) relationship between $\mathbf{x_i}$ and y_i. Furthermore, the target variable y is constrained to be either a 0 or 1: a 1 label is considered a *positive* label, and a 0 label is considered a *negative* label.

We also have a black box *model* that, given some input $\mathbf{x_i}$, will produce as its output an estimate of y_i, denoted $\hat{y}_i$. This model is called a *binary classifier*.

The question we will consider in these notes - without knowing any details of the classifier model - is how can we evaluate the performance of the classifier?

Possible outcomes

Consider a classifier model that is trained to identify cat photographs. Its output is $\hat{y} = 1$ if it thinks the photograph is of a cat, and $\hat{y} = 0$ otherwise.

For each prediction the classifier makes, there are four possible outcomes:

- **True positive**: $y = 1, \hat{y} = 1$. This is a *correct* prediction.
- **False positive**: $y = 0, \hat{y} = 1$. This is called *Type 1* error. (Also known as a *false alarm*.)
- **False negative**: $y = 1, \hat{y} = 0$. This is called *Type 2* error. (Also known as a *missed detection*.)
- **True negative**: $y = 0, \hat{y} = 0$. This is a *correct* prediction.

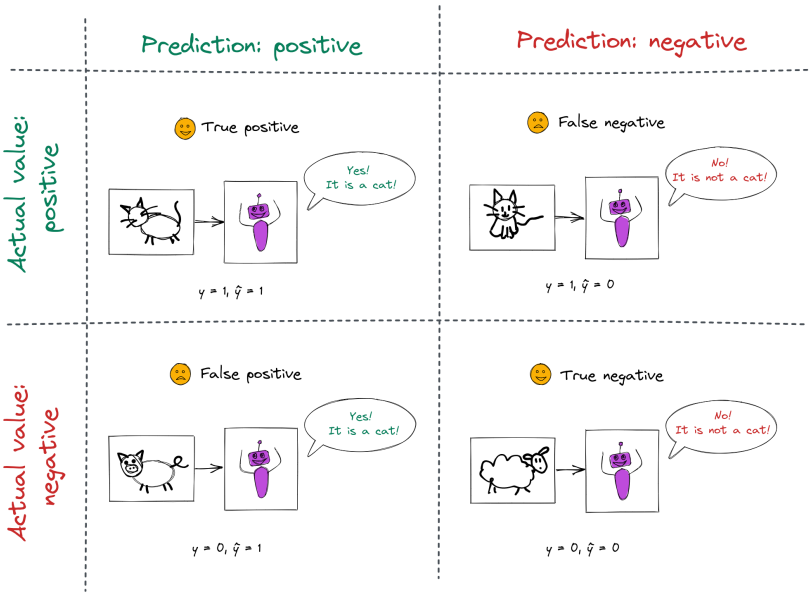

Figure 1: Four outcomes for a cat photograph classifier.

The number of *true positive* (TP) outcomes, *true negative* (TN) outcomes, false positive (FP) outcomes, and false negative (FN) outcomes, are often presented together in a **confusion matrix**:

Actual $\downarrow$ Pred. $\rightarrow$	1	0
1	TP	FN
0	FP	TN

We may also define two more quantities:

- The number of actual positive values (when $y = 1$) **P** = TP+FN, is the sum of the "actual positive" cells.
- The number of actual negative values (when $y = 0$) **N** = FP+TN is the sum of the "actual negative" cells.

The **total population**, P + N, is the total number of samples.

Metrics related to error

The most basic classifier performance metric is **accuracy**, defined as

$$\frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{P + N}$$

i.e., the portion of samples classified correctly.

However, accuracy is not always a useful metric. For example, imagine you are training a model to classify credit card transactions as fraudulent (1) or not fraudulent (0), but only 1% of transactions are fraudulent. A very basic classifier that *always* outputs 0 will have 99% accuracy! It is clear that accuracy is not a very useful metric here.

For a data set with highly imbalanced classes ($P >> N$ or $P << N$), **balanced accuracy** is often a more appropriate metric:

$$\frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$$

Balanced accuracy gives the proportion of correct predictions in each class, averaged across classes.

In addition to the overall accuracy, a number of other metrics are used in various contexts. These are defined in terms of the four basic numbers described above: TP, FN, FP, TN.

- **True Positive Rate** (TPR) also called *recall* or *sensitivity*:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = P(\hat{y} = 1 | y = 1)$$

- **True Negative Rate** (TNR) also called *specificity*:

$$TNR = \frac{TN}{N} = \frac{TN}{FP + TN} = P(\hat{y} = 0 | y = 0)$$

- **Positive Predictive Value** (PPV) also called *precision*:

$$PPV = \frac{TP}{TP + FP} = P(y = 1|\hat{y} = 1)$$

- **Negative Predictive Value** (NPV):

$$NPV = \frac{TN}{TN + FN} = P(y = 0|\hat{y} = 0)$$

- **False Positive Rate** (FPR):

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TNR = P(\hat{y} = 1|y = 0)$$

- **False Discovery Rate** (FDR):

$$FDR = \frac{FP}{FP + TP} = 1 - PPV = P(y = 0|\hat{y} = 1)$$

- **False Negative Rate** (FNR):

$$FNR = \frac{FN}{FN + TP} = 1 - TPR = P(\hat{y} = 0|y = 1)$$

- **False Omission Rate** (FOR):

$$FOR = \frac{FN}{FN + TN} = 1 - NPV = P(y = 1|\hat{y} = 0)$$

These metrics are illustrated in the following table:

	Actual – Positive	Actual – Negative
Predicted – Positive	**True Positive (TP)** PPV = $\frac{TP}{TP+FP}$ TPR = $\frac{TP}{TP+FN}$	**False Positive (FP)** FDR = $\frac{FP}{TP+FP}$ FPR = $\frac{FP}{FP+TN}$
Predicted – Negative	**False Negative (FN)** FOR = $\frac{FN}{TN+FN}$ FNR = $\frac{FN}{TP+FN}$	**True Negative (TN)** NPV = $\frac{TN}{TN+FN}$ TNR = $\frac{TN}{TN+FP}$

Figure 2: Selected classifier metrics.

Another metric, known as F1 score, combines precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) in one metric:

$$F_1 = 2\left(\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right)$$

The most appropriate choice of metric for evaluating a classifier depends on the context - for example, whether there is class imbalance, and what the relative cost of each type of error is.

Tradeoff between FPR and TPR using thresholds

It is trivial to build a classifier with no Type 1 error (no false positives) - if the classifier predicts a negative value for all samples, it will not produce any false positives. However, it also won't produce any true positives! (Similarly, it is trivial to build a classifier with no Type 2 error, by predicting a positive value for all samples. This model will have no false negatives, but also no true negatives.)

We can often adjust the tradeoff between the FPR and TPR, depending on the *cost* of each type of error. Many classifiers are actually **soft decision** classifiers, which means that their output is a probability, $P(y = 1|\mathbf{x})$.

(This is in contrast to **hard decision** classifiers, whose output is a label, e.g. 0 or 1.)

We get a "hard" label from a "soft" classifier by setting a threshold t, so that:

$$\hat{y} = \begin{cases} 1, & P(y = 1|\mathbf{x}) \geq t \\ 0, & P(y = 1|\mathbf{x}) < t \end{cases}$$

By tuning this **threshold** we can adjust the tradeoff between FPR and TPR.

For example, consider our cat photo classifier from earlier, but suppose it is a soft decision classifier:

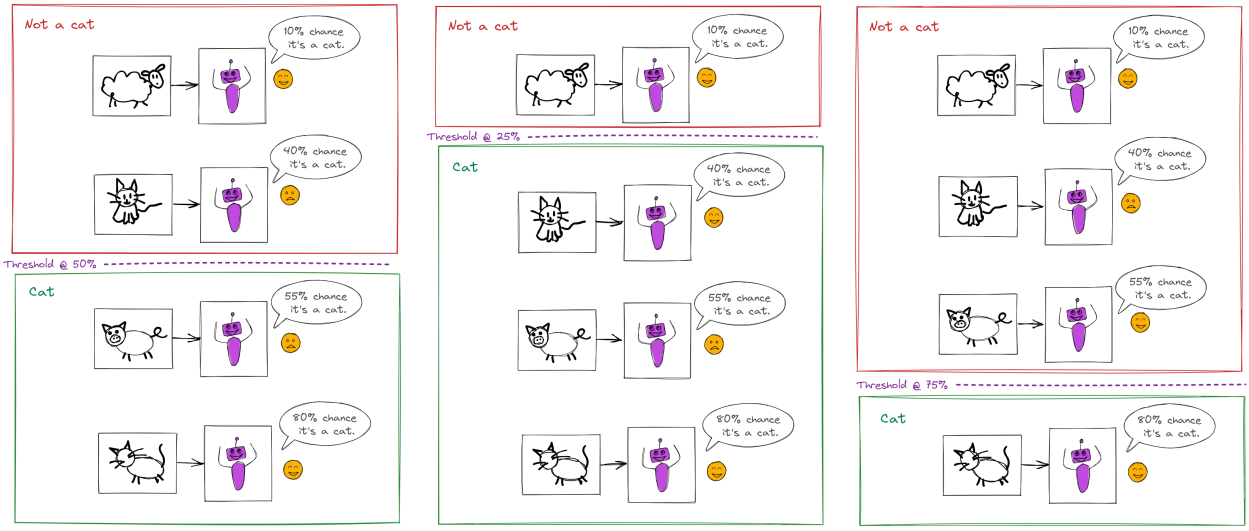

Figure 3: Soft decision classifier for cat photos.

The performance of this classifier depends on where we set the threshold t for a positive prediction:

- If we set the threshold at 50%, this classifier has one TP, one TN, one FP, and one FN on the data shown. ($TPR = 0.5, FPR = 0.5$.)
- What if the cost of missing a "true" cat is high, but the cost of accidentally classifying a non-cat as a cat is low? Then we might set the threshold at 25%. The classifier then has two TPs, one TN, and one FP. ($TPR = 1, FPR = 0.5$.)
- What if the cost of missing a "true" cat is low, but the cost of accidentally classifying a non-cat as a cat is high? Then we might set the threshold at 75%. The classifier then has one TP, two TNs, and one FN. ($TPR = 0.5, FPR = 0$.)

The label applied by the classifier depends on where we set the threshold, the error metrics above also depend on where we set the threshold. But, it's useful to be able to evaluate the classifier performance in general, instead of for a specific threshold. We do this by plotting the TPR vs. FPR for every possible threshold, like in this plot:

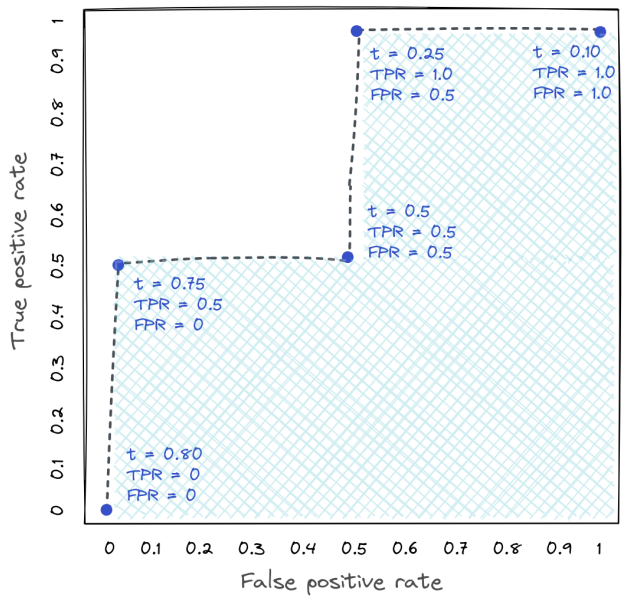

Figure 4: Plot of TPR vs. FPR for the cat photo classifier.

This plot is know as the **ROC curve** (receiver operating characteristic). The shaded area underneath the ROC curve is known as the **AUC** (area under the curve), and it is a classification-threshold-invariant way of evaluating the classifier performance.

A random classifier that doesn't use any information about the problem will have an AUC of 0.5 (if both classes are equally prevalent in the data). A perfect classifier will have an AUC of 1. A typical machine learning model will have an AUC somewhere between the two, with a number closer to 1 being a better score.

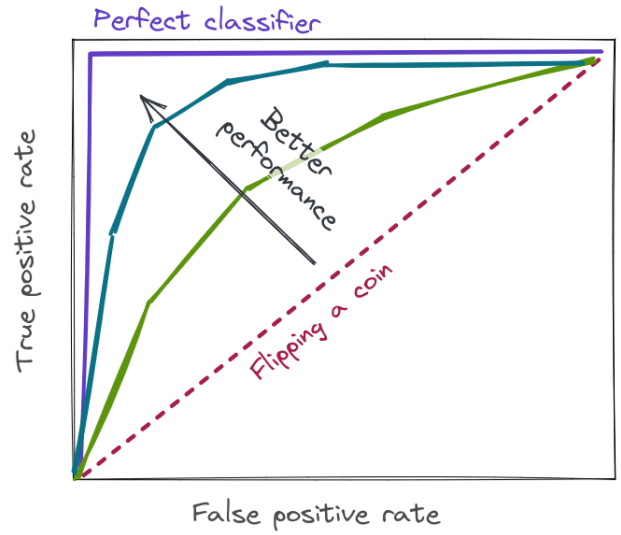

Figure 5: Plot of TPR vs. FPR for the cat photo classifier.

Multi-class classifier performance metrics

So far, we have only discussed a binary classifier. For a multi-class classifier, the output variable is no longer restricted to 0 or 1; instead, we have $y \in 1, 2, \cdots, K$ where K is the number of classes.

The same performance metrics apply to a multi-class classifier, with some minor modifications:

- The **accuracy** is the number of correct labels, divided by number of samples
- The **balanced accuracy** is a direct extension of two-class version: compute the per-class accuracy, and average across classes.
- For other metrics, we can use pairwise comparisons between one class and all others, to compute a per-class version of the metric.
- A **soft-decision classifier** will produce a vector of probabilities, one for each class.

The error of a multi-class classifier can also be visualized using a confusion matrix, for example:

Target	Selected									Acc
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
1	**137**	13	3	0	0	1	1	0	0	0.89
2	1	**55**	1	0	0	0	0	6	1	0.86
3	2	4	**84**	0	0	0	1	1	2	0.89
4	3	0	1	**153**	5	2	1	1	1	0.92
5	0	0	3	0	**44**	2	2	1	2	0.82
6	0	0	2	1	4	**35**	0	0	1	0.81
7	0	0	0	0	0	0	**61**	2	2	0.94
8	0	0	0	1	0	0	0	**69**	3	0.95
9	0	0	0	0	0	0	0	2	**26**	0.93
										0.89

The diagonal elements (correct decisions) are marked in bold. Column "Acc" provides the specific accuracy for each key.

Figure 6: Example of a multi-class confusion matrix, via Cross Validated.

Evaluating a classifier - some considerations

There is no universal rule for what makes a "good" classifier. It's a common misconception that a "good" classifier should achieve some high accuracy e.g. 95%, 99%, etc. Yet, we have seen that even a very bad classifier will have high accuracy sometimes (if there is class imbalance). Meanwhile, for some very difficult problems, even a classifier with much lower accuracy may be useful (if it still has higher accuracy than any alternative solution). Finally, not all types of errors are equally "bad" - we may prefer a classifer that makes more errors overall but fewer "bad" errors, over one that has fewer overall errors but more of the "bad" type.

To decide whether a machine learning classifier is doing a "good job", here are some helpful questions to ask yourself:

- Does the model have better performance than a "simple" model that *always* predicts the more common class (i.e. "prediction by mode")?
- Does the model have better performance than an alternative solution (e.g. a rule-based implementation), if one is available?
- Are all types of error equally "expensive" in context, or are some types (e.g. false positive, false negative) more costly? Is the rate of the "expensive" error small? (Also note that different "stakeholders" may care more about some types of errors than other types.)

Questions

(You can check your answers to the first four questions here.)

1. In which of the following scenarios does the accuracy value suggest that the ML model is doing a good job?

- In the game of roulette, a ball is dropped on a spinning wheel and eventually lands in one of 38 slots. Using visual features (the spin of the ball, the position of the wheel when the ball was dropped, the height of the ball over the wheel), an ML model can predict the slot that the ball will land in with an accuracy of 4%.
- A deadly, but curable, medical condition afflicts .01% of the population. An ML model uses symptoms as features and predicts this affliction with an accuracy of 99.99%.
- An expensive robotic chicken crosses a very busy road a thousand times per day. An ML model evaluates traffic patterns and predicts when this chicken can safely cross the street with an accuracy of 99.99%.

2. Consider a classification model that separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to precision?

- Probably increase
- Probably decrease

3. Consider a classification model that separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to recall?

- Always stay constant
- Either decrease, or stay the same
- Always increase

4. Consider two models—A and B—that each evaluate the same dataset. Which one of the following statements is true?

- If model A has better recall than model B, then model A is better.
- If Model A has better precision than model B, then model A is better.
- If model A has better precision and better recall than model B, then model A is probably better.