Error Metrics for Skewed Analysis

In the context of evaluation and of error metrics, there is one important case, where it's particularly tricky to come up with an appropriate error metric, or evaluation metric, for our learning algorithm, that case is the case of what's called **skewed classes**.

Consider the problem of cancer classification, where we have features of medical patients and we want to decide whether or not they have cancer (this is like the malignant versus benign tumor classification example that we had earlier).

Cancer classification example

- Train logistic regression model $h_\theta(x)$, ($y = 1$ if cancer, $y = 0$ otherwise).
- We have trained the regression classifier model and let's say we test our classifier on a test set and we find that we got 1% error on test set (99% correct diagnoses).
- But only 0.50% of patients have cancer (in this case, the 1% error no longer looks so impressive).

So this setting of when the ratio of positive to negative examples is very close to one of two extremes, where in this case, the number of positive examples is much smaller than the number of negative examples, this is what we call the case of **skewed classes**.

- LOTS more of one class than another
- So standard error metrics aren't so good

So the problem with using classification error or classification accuracy as our evaluation metric is the following:

- We have one learning algorithm that's getting 99.2% accuracy (0.8% error).
- We make a change to our algorithm and we now are getting 99.5% accuracy (0.5% error).
- So, is this an improvement to the algorithm or not? One of the nice things about having a single real number evaluation metric is this helps us to quickly decide if we just need a good change to the algorithm or not. By going from 99.2% accuracy to 99.5% accuracy.

If we have very skewed classes it becomes much harder to use just classification accuracy, because we can get very high classification accuracies or very low errors, and it's not always clear if doing so is really improving the quality of our classifier because predicting $y = 0$ all the time doesn't seem like a particularly good classifier. When we're faced with such a skewed classes therefore we would want to come up with a different error metric or a different evaluation metric.

Precision/Recall

- Two new metrics - **precision** and **recall**
- $y = 1$ in presence of rare class that we want to detect.

Let's say we are evaluating a classifier on the test set, for the examples in the test set the actual class of that example in the test set is going to be either 1 or 0 (there is a binary classification problem).

And what our learning algorithm will do is it will predict some value for the class and our learning algorithm will predict the value for each example in our test set and the predicted value will also be either 1 or 0. Considering this, classification can be:

- True positive (we guessed 1, it was 1)
- False positive (we guessed 1, it was 0)
- True negative (we guessed 0, it was 0)
- False negative (we guessed 0, it was 1)

Precision: How often does our algorithm cause a false alarm?

Of all patients where we predicted $y = 1$, what fraction actually has cancer?

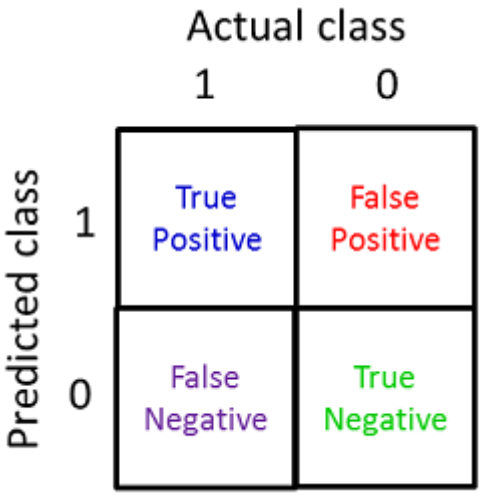
- $\text{Presicion} = \text{True positives} / \#\text{ predicted positives}$
- $\text{Precision} = \text{True positives} / \#\text{ (true positives} + \text{false positives)}$
- High precision is good (i.e. the proportion closer to 1).
- We want a big number, because we want false positive to be as close to 0 as possible.

Recall: How sensitive is our algorithm?

Of all patients that actually have cancer what fraction did we correctly detect as having cancer?

- $\text{Presicion} = \text{True positives} / \#\text{ actual positives}$
- $\text{Precision} = \text{True positives} / \#\text{ (true positives} + \text{false negatives)}$
- High recall is good (i.e. the proportion closer to 1).
- We want a big number, because you want false negative to be as close to 0 as possible.

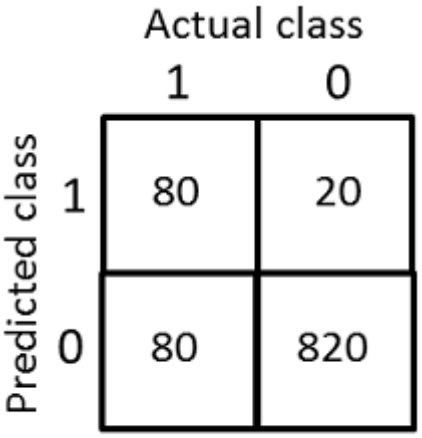
Video Question: Precision and recall are defined according to:



$$\text{Precision} = \frac{\text{True positives}}{\#\text{ predicted as positive}} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\#\text{ actual positives}} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

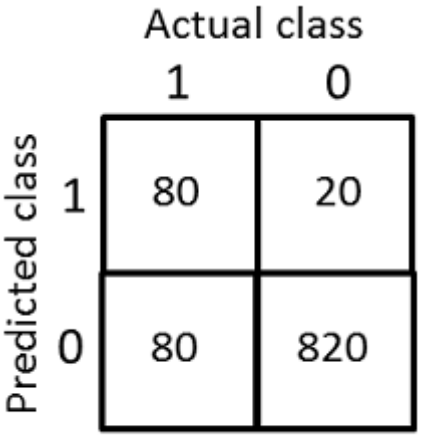
Your algorithm's performance on the test set is given to the right. What is the algorithm's precision?



0.8

- 0.5
- 0.08
- 0.1

Your algorithm’s performance on the test set is given to the right. What is the algorithm’s recall?



0.5

- 0.8
- 0.08
- 0.1

Generally, even for settings where we have very skewed classes, it's not possible for an algorithm to sort of "cheat" and somehow get a very high precision and a very high recall by doing some simple thing like predicting y equals 0 all the time or predicting y equals 1 all the time, we're much more sure that a classifier of a high precision or high recall actually is a good classifier, and this gives us a more useful evaluation metric that is a more direct way to actually understand whether, our algorithm may be doing well.

Note: Typically we say the presence of a rare class is what we're trying to determine (e.g. positive (1) is the existence of the rare thing).