**Classification vs. Regression**

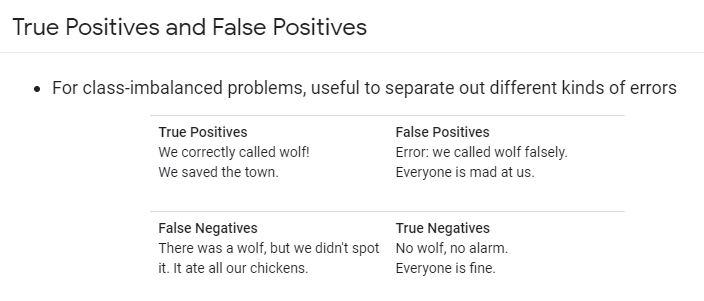
* Sometimes, we use logistic regression for the probability outputs -- this is a regression in (0, 1)
* Other times, we'll threshold the value for a discrete binary classification
* Choice of threshold is an important choice, and can be tuned

**Evaluation Metrics: Accuracy**

* How do we evaluate classification models?
* One possible measure: Accuracy
  + the fraction of predictions we got right

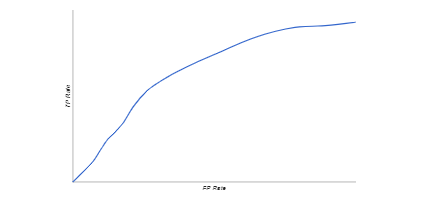
**Accuracy Can Be Misleading**

* In many cases, accuracy is a poor or misleading metric
  + Most often when different kinds of mistakes have different costs
  + Typical case includes *class imbalance*, when positives or negatives are extremely rare



**A ROC Curve**

Each point is the TP and FP rate at one decision threshold.



ROC = Receiver Operator Characteristic Curve

TP/FP = True Positive/ False Positive

## Prediction Bias

* Logistic Regression predictions should be unbiased.
  + average of predictions == average of observations
* Bias is a canary.
  + Zero bias alone does not mean everything in your system is perfect.
  + But it's a great sanity check.
* If you have bias, you have a problem.
  + Incomplete feature set?
  + Buggy pipeline?
  + Biased training sample?
* Don't fix bias with a calibration layer, fix it in the model.
* Look for bias in slices of data -- this can guide improvements.

# Classification: Thresholding



**Estimated Time:** 2 minutes

Logistic regression returns a probability. You can use the returned probability "as is" (for example, the probability that the user will click on this ad is 0.00023) or convert the returned probability to a binary value (for example, this email is spam).

A logistic regression model that returns 0.9995 for a particular email message is predicting that it is very likely to be spam. Conversely, another email message with a prediction score of 0.0003 on that same logistic regression model is very likely not spam. However, what about an email message with a prediction score of 0.6? In order to map a logistic regression value to a binary category, you must define a **classification threshold** (also called the **decision threshold**). A value above that threshold indicates "spam"; a value below indicates "not spam." It is tempting to assume that the classification threshold should always be 0.5, but thresholds are problem-dependent, and are therefore values that you must tune.

# Classification: True vs. False and Positive vs. Negative



**Estimated Time:** 5 minutes

In this section, we'll define the primary building blocks of the metrics we'll use to evaluate classification models. But first, a fable:

**An Aesop's Fable: The Boy Who Cried Wolf (compressed)**

A shepherd boy gets bored tending the town's flock. To have some fun, he cries out, "Wolf!" even though no wolf is in sight. The villagers run to protect the flock, but then get really mad when they realize the boy was playing a joke on them.

[Iterate previous paragraph N times.]

One night, the shepherd boy sees a real wolf approaching the flock and calls out, "Wolf!" The villagers refuse to be fooled again and stay in their houses. The hungry wolf turns the flock into lamb chops. The town goes hungry. Panic ensues.

Let's make the following definitions:

* "Wolf" is a **positive class**.
* "No wolf" is a **negative class**.

We can summarize our "wolf-prediction" model using a 2x2 [confusion matrix](https://developers.google.com/machine-learning/glossary#confusion_matrix) that depicts all four possible outcomes:

|  |  |
| --- | --- |
| True Positive (TP):   * Reality: A wolf threatened. * Shepherd said: "Wolf." * Outcome: Shepherd is a hero. | False Positive (FP):   * Reality: No wolf threatened. * Shepherd said: "Wolf." * Outcome: Villagers are angry at shepherd for waking them up. |
| False Negative (FN):   * Reality: A wolf threatened. * Shepherd said: "No wolf." * Outcome: The wolf ate all the sheep. | True Negative (TN):   * Reality: No wolf threatened. * Shepherd said: "No wolf." * Outcome: Everyone is fine. |

A **true positive** is an outcome where the model correctly predicts the positive class. Similarly, a **true negative** is an outcome where the model correctly predicts the negative class.

A **false positive** is an outcome where the model incorrectly predicts the positive class. And a **false negative** is an outcome where the model incorrectly predicts the negative class.

# Classification: Accuracy



**Estimated Time:** 6 minutes

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictionsTotal number of predictions

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

Accuracy=TP+TNTP+TN+FP+FN

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Let's try calculating accuracy for the following model that classified 100 tumors as [malignant](https://wikipedia.org/wiki/Malignancy) (the positive class) or [benign](https://wikipedia.org/wiki/Benign_tumor) (the negative class):

|  |  |
| --- | --- |
| True Positive (TP):   * Reality: Malignant * ML model predicted: Malignant * **Number of TP results: 1** | False Positive (FP):   * Reality: Benign * ML model predicted: Malignant * **Number of FP results: 1** |
| False Negative (FN):   * Reality: Malignant * ML model predicted: Benign * **Number of FN results: 8** | True Negative (TN):   * Reality: Benign * ML model predicted: Benign * **Number of TN results: 90** |

Accuracy=TP+TNTP+TN+FP+FN=1+901+90+1+8=0.91

Accuracy comes out to 0.91, or 91% (91 correct predictions out of 100 total examples). That means our tumor classifier is doing a great job of identifying malignancies, right?

Actually, let's do a closer analysis of positives and negatives to gain more insight into our model's performance.

Of the 100 tumor examples, 91 are benign (90 TNs and 1 FP) and 9 are malignant (1 TP and 8 FNs).

Of the 91 benign tumors, the model correctly identifies 90 as benign. That's good. However, of the 9 malignant tumors, the model only correctly identifies 1 as malignant—a terrible outcome, as 8 out of 9 malignancies go undiagnosed!

While 91% accuracy may seem good at first glance, another tumor-classifier model that always predicts benign would achieve the exact same accuracy (91/100 correct predictions) on our examples. In other words, our model is no better than one that has zero predictive ability to distinguish malignant tumors from benign tumors.

Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set**, like this one, where there is a significant disparity between the number of positive and negative labels.

In the next section, we'll look at two better metrics for evaluating class-imbalanced problems: precision and recall.