Stanford MEDICINE

Stanford | Center for Health Education

FUNDAMENTALS OF MACHINE LEARNING FOR HEALTHCARE

MODULE 4 - EVALUATION AND METRICS FOR MACHINE LEARNING IN HEALTHCARE

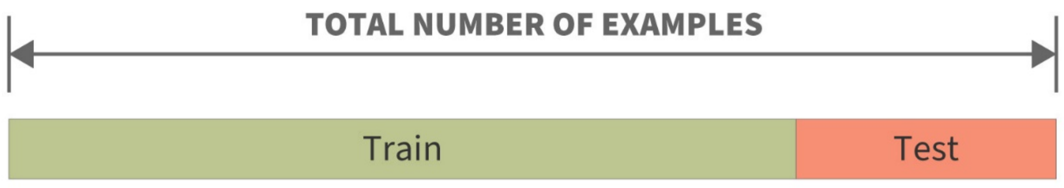
LEARNING OBJECTIVES

- Learn important approaches for leveraging data to train, validate, and test machine learning models
- Assess model training behavior and understand important concepts like underfitting and overfitting in healthcare settings.
- Begin developing an intuition regarding hyperparameters and the downstream effects of hyperparameter tuning.
- Determine the correct set of metrics for model evaluation and understand the most common metrics used in machine learning for clinical research especially the receiver operating curve and precision recall curve

CRITICAL EVALUATION OF MODELS AND STRATEGIES FOR HEALTHCARE APPLICATIONS

It is very important to consider how we evaluate the performance of our models.

Recall that we typically split our data into 3 sets - **train, validation**, and **test**.



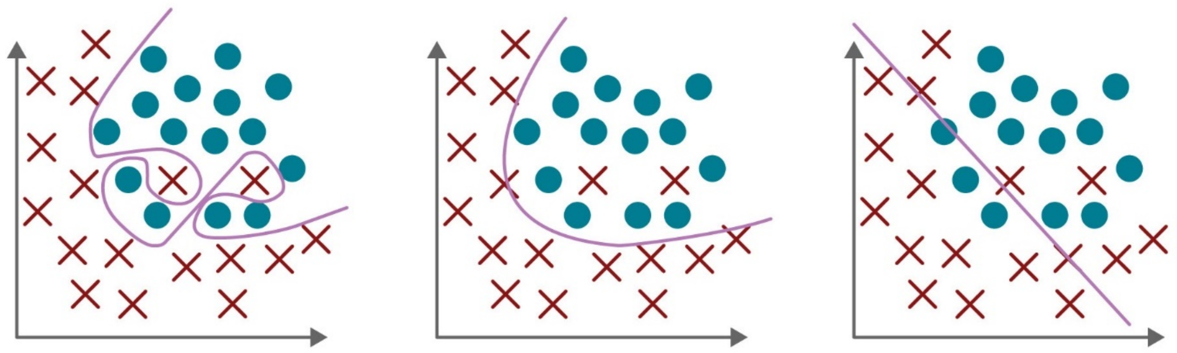
- Usually 70-80% of your data is used for training and the rest for testing the model performance. (i.e we use a 80%–20% or 70%–30% **train-test split**)

- ○ In the special case of time series data, most people hold aside all data from the most recent time point and train the model on the data before that (i.e. training and validation data from 2012-2015 and test 2016-2017)

- You may also hear “train-validation-test” split
    - Recall that the validation set is used to choose hyperparameters. It is different from the test set, which contains data you never see while tuning your model
    - Note that sometimes you may hear the validations set referred to as the dev or development set. Some research scientists reserve the term validation set to refer to the test set as the validation set

Another method for shuffling the training and testing data is something called **cross-validation,** or **“k-fold cross validation”**

- Similar to a train/test split, but instead of creating one split, multiple splits are applied to more subsets multiple times
- In a typical 80-20 train-test split, you simply assign 80% of the data at random to the train set, the rest goes to test. In k-fold cross validation, we repeat this process k times, so that a different, random, 80% of the data ends up in train each time. Each split creates a “fold.” You then train on k-1 of these folds, holding out the last one to use as the test set
- Doing this over and over allows us to get many different estimations model parameters
- This method can be particularly advantageous when there is not much data to begin with



Over-fitting: fitting too strongly to the training data

Appropriate-fitting

Under-fitting: unable to capture underlying trend of data

Overfitting occurs when the model begins to memorize the random fluctuations, anomalies, and noise in the training dataset. At this point, the model will have a high training accuracy despite the fact that it will no longer be extracting relevant generalizable signal from the data, meaning the test accuracy will suffer.

- Extended **training time** can lead to overfitting
  - If trained for too long on the same data, a model will start to overfit, harming the performance of the system on new data
  - To try and guarantee that our model is extracting only generalizable signal from the train set, we want to stop just before the error on the validation set starts increasing
- Excessive **model complexity** can lead to overfitting
  - A very complex model (i.e. a very deep neural network) that can accommodate a high number of feature weights, will frequently experience overfitting - especially on small datasets
    - For instance, if your feature space was the same size as your data set, the model could directly memorize your data by associating each data point with a unique feature

Underfitting occurs when a statistical model is unable to obtain a good fit of the trends in the data, leading to poor performance on the training and testing data.

- This occurs when a model is too simple to fit more complex trends in the data

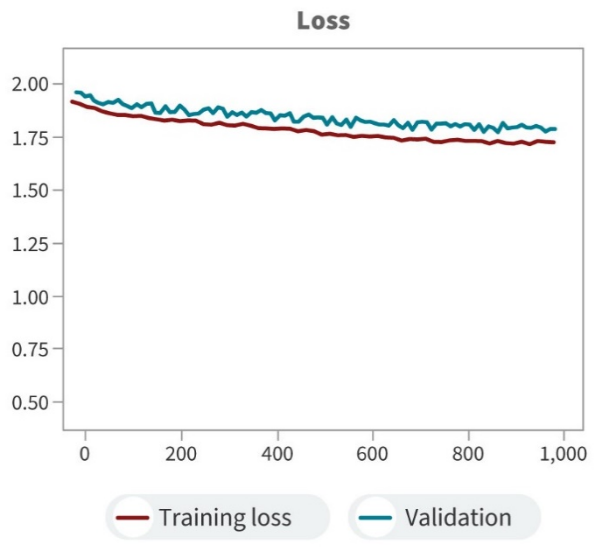
Appropriate fitting is the goal, and one of the major challenges facing machine learning practitioners.

- In order to hit the sweet spot of appropriate fitting, we can tweak hyperparameters and modify algorithmic design choices
  - The gains achieved in this way will grow smaller over time as the model gets closer and closer to the optimal weights

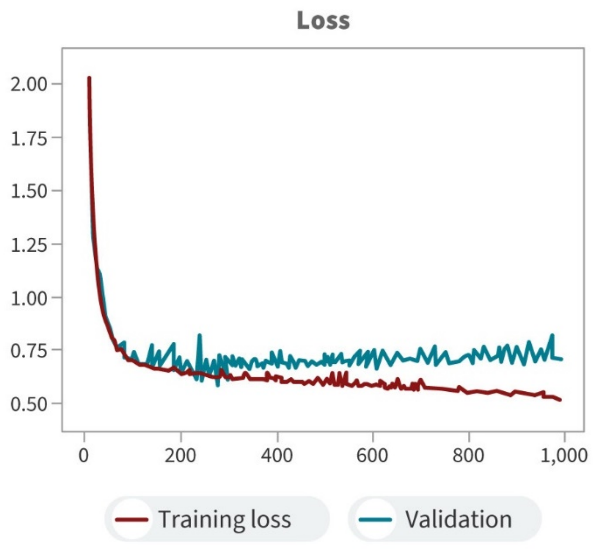
We use **learning curves** - plots of model performance over time - to monitor the learning process in algorithms that learn incrementally from training data

- They are used to diagnose problems such as model underfitting or overfitting, to sanity-check and to debug code and implementations
- Often learning curves plots **model loss** over time, or as the model is exposed to more and more training data

Plotting the model's learning curve for both the training and validation sets side by side can be very **useful for debugging**.

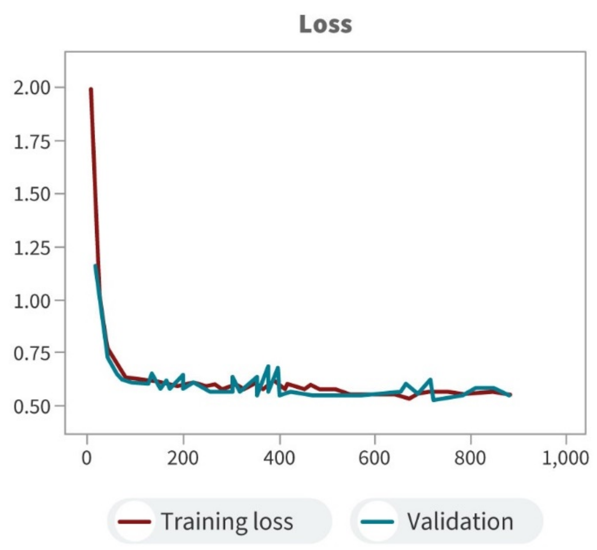


- The loss curve for an **underfit** model usually looks something like this, where both the training and validation loss curves don't decrease much and stay relatively flat lines at high loss, indicating that the model is unable to learn from the training dataset to reduce the loss. Indeed, even seeing just a flat training loss curve, without plotting a validation loss curve, is a good reason to suspect underfitting. Sometimes the curves may also show oscillating noisy values but these will still be centered around high loss values without significant trends of decrease.



- The loss curve for an **overfit** model usually looks something like this where the training loss continues to decrease over time, but the validation loss decreases to a point and then begins to increase again.

A good fit is our goal when training machine learning models, and occurs at the sweet spot where the model is neither underfitting nor overfitting.



- When we have a good fit, the training and validation losses will both decrease, at a large rate of decrease initially and then smaller over time, until they reach a point of stability (in other words they converge). People also refer to this as **reaching a plateau**
- Ideally, the training and loss curves will plateau to similar values, but you will often see a small gap between the two where the training curve converges to a lower loss value than the validation loss. This gap between the training loss and validation loss is referred to as the **"generalization gap"**, and intuitively we can expect this to happen because the model is directly optimizing to perform well on the training set

A second type of learning curve, the plot of the final performance metric (e.g accuracy) to **visualize model progress during training**

- This is useful to get a sense of actual model performance, since lower loss usually corresponds to better model performance but it doesn't tell us whether the accuracy is at a level we are happy with or not

STRATEGIES TO ADDRESS UNDERFITTING AND OVERFITTING

A model **underfits** when it is unable to learn features from the training set, which means that it performs poorly on both the training set and the validation set.

Reasons why a model underfits:

- The model is not expressive enough for the data that you have.
- The samples in the training set do not have the information required to make the right decisions.

To fix underfitting:

- Train your model for more time
- Increase the capacity of your model

Reasons why the model **overfits**:

- It is learning features that are specific to the training set that cannot be found in the validation set

To fix overfitting, broadly speaking, consider **regularization**– which means to force the model to learn and retain general insights that allow it to extrapolate what it has learned to unseen data.

- **Weight decay** or **L1/L2 regularization** means penalizing the model for using too many of its parameters.
    - Using weight decay means adding a value in the loss that represents the magnitude of the parameter values in the model. If the model has many non-zero or large weights, then the magnitude will be large
    - If the model has few non-zero or large weights, the magnitude will be small. Because the model seeks to minimize the loss, this constraint forces the model to have small-valued weights
    - The strength of the effects of weight decay is a very common hyperparameter
- **Dropout** means randomly setting parameter values to zero in the model during training.
    - Picking a layer of the model that randomly "drops out" output values and sets them to zero
    - Intuitively, this makes that layer more unreliable, and the model thus has to build in redundancy into its subsequent layers
    - The need for redundancy means that the model cannot be as complex, hence dropout has the effect of regularization
    - The probability of dropping out a given neuron is a very common hyperparameter.
- **Data augmentation** means randomly warping / transforming the samples in the training set in order to prevent the model from learning any features that are too specific.
    - Common augmentations include: Rotation, random crops, resizing, color and brightness, flipping them horizontally / vertically

The model overfits the training set because it gets too familiar with the samples. Data augmentation constantly changes what the samples look like, so you slow down / prevent the model from memorizing the training set.

Note: Be careful with the transformations. The samples must still be representative of the label they are affiliated with.

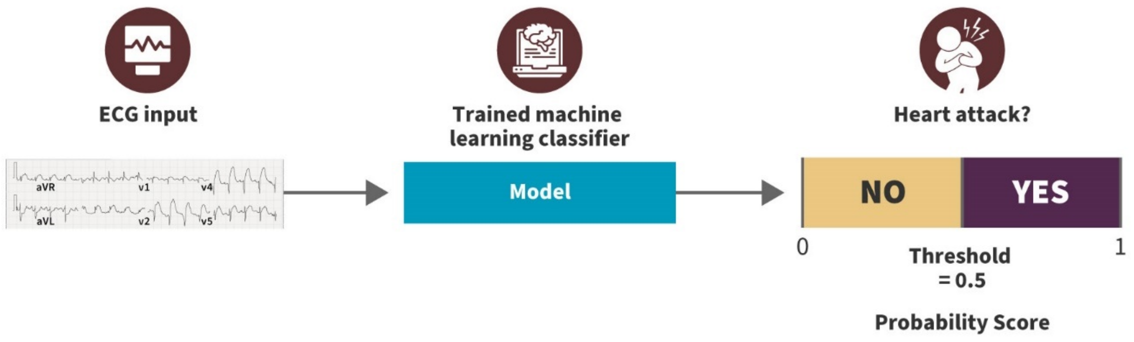
STATISTICAL APPROACHES TO MODEL EVALUATION

Picking the right metric is critical for evaluating machine learning models.

Accuracy is not always the best metric. If 90% of samples in the test set are benign, then a model that predicts that everything is benign would achieve a 90% accuracy.

Choosing a threshold is not always straightforward.

- The output of a trained machine learning classifier for categorical labeling tasks will typically be a probability score (usually between 0 and 1) for the desired output label
- If we are trying to understand the performance of our classifier in more concrete terms, in particular typical ways in which we might use it in the real world, then we have to choose a "threshold" that will binarize the predicted label into a specific category prediction, i.e. convert that probability to either 0 or 1

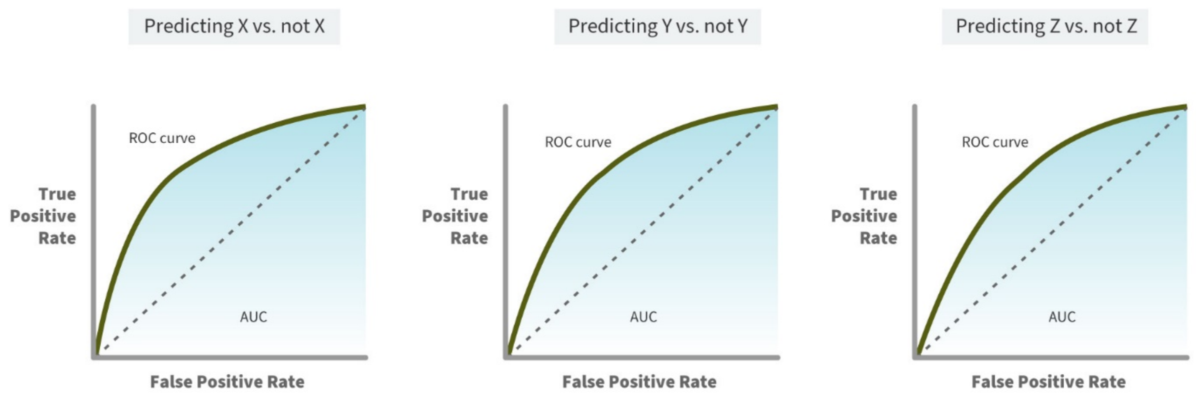


1. The most common approach here is to choose a threshold of 0.5 as the middle ground so that anything greater is a "positive" decision for the label and anything less is a "negative" decision for the label
2. With that threshold the common metrics used in medical testing can then be calculated

- However, 0.5 seems relatively arbitrary given the model's ability to produce more nuanced values

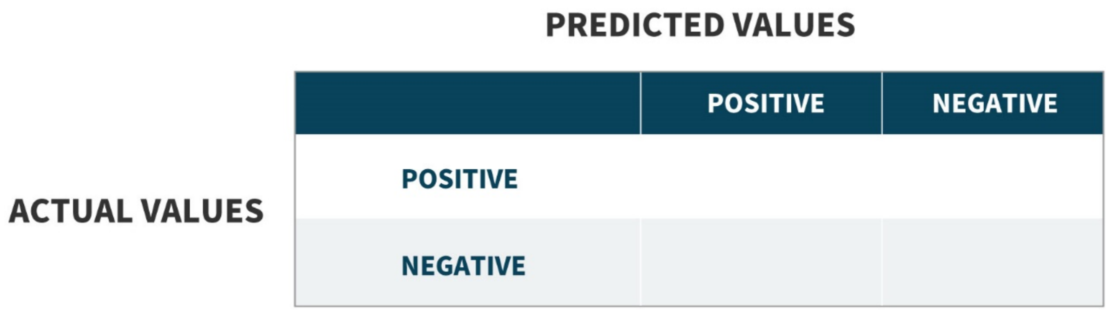
The **receiver operating characteristic (ROC)** curve is a metric for evaluating the model performance that considers all thresholds simultaneously.

1. Algorithms that were trained using discrete labels (such as disease / no disease) are most suited to this approach



2. If our model can detect multiple classes we would plot an AUC ROC curve for each one - so for example, If you have three classes named X, Y and Z, you will have one ROC for X classified against Y and Z, another ROC for Y classified against X and Z, and a third one of Z classified against Y and X

In order to understand the implications for our medical task in an ROC analysis, the knowledge on basics of statistical testing is needed. The fundamental analysis of performance for machine learning classification problem is a table that contains different combinations of predicted and actual values, which is known as the **confusion matrix**.



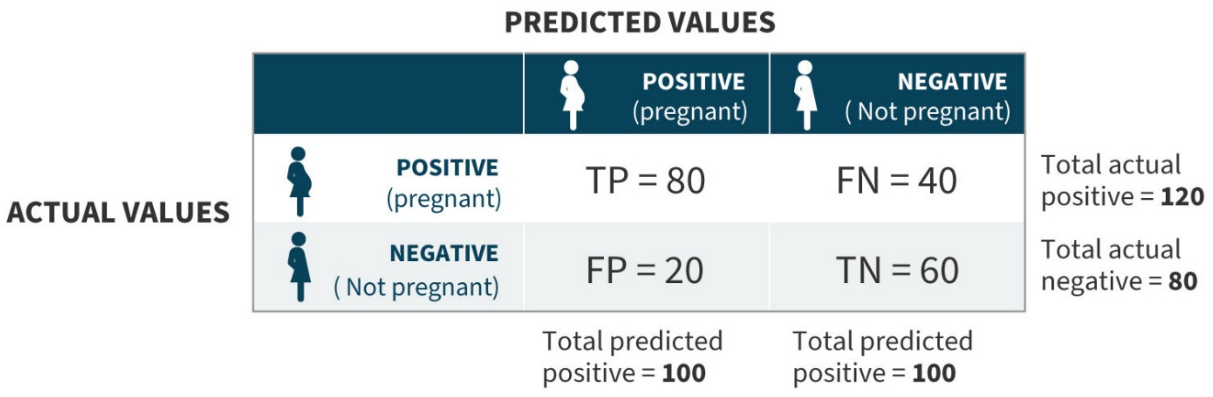
		PREDICTED VALUES	
		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE		
	NEGATIVE		

The table allows us to derive metrics such as: Precision, Recall / Sensitivity, Specificity, Accuracy.

Example: We have a smartphone app that can predict pregnancy using the heart rate function on a wearable device.

- We have a trained our machine learning model

- We want to see how it performs on the hold- out test set of 200 cases with 120 positives (user was pregnant) and 80 negatives (user was not pregnant)
- When we ran our model, it predicted 100 negatives and 100 positives



PREDICTED VALUES

ACTUAL VALUES		POSITIVE (pregnant)	NEGATIVE (Not pregnant)	
	POSITIVE (pregnant)	TP = 80	FN = 40	Total actual positive = **120**
	NEGATIVE (Not pregnant)	FP = 20	TN = 60	Total actual negative = **80**
		Total predicted positive = **100**	Total predicted positive = **100**	

- The four boxes of the confusion matrix:
    - True positive (TP): Cases that were positive (pregnant) and our model predicted positive (pregnant)
    - True negative (TN): Cases that were negative (not pregnant) and our model predicted negative (not pregnant)
    - False positive (FP): Cases that were negative (not pregnant) but we predicted positive (pregnant)
    - False negative (FN): Cases that were positive (pregnant) but we predicted negative (not pregnant)

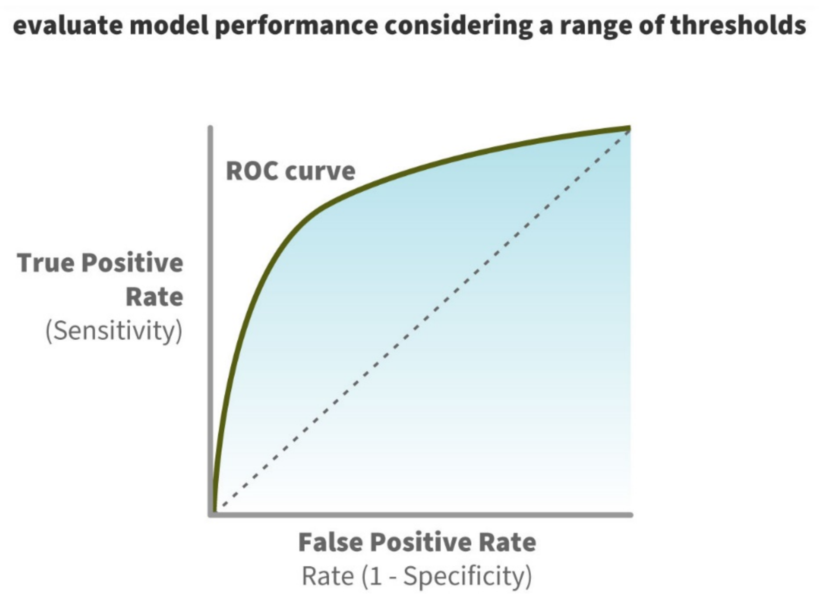
Metric Definitions

- **Accuracy**: Number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0.
- **Sensitivity** or **recall**: When the patient is pregnant how often does the test predict pregnant? In other words, out of all positive datapoints, how many did the model predict as positive?
- **Specificity**: When the patient is not pregnant how often does the test predict not pregnant?
- **Precision** (**positive predictive value**): How often is the model correct when predicting positive?
- **Negative predictive value**: How often is the model correct when predicting negative? Note that both positive and negative predictive values are influenced by the prevalence of conditions in the test set and this can be misleading

IMPORTANT METRICS FOR CLINICAL MACHINE LEARNING

The receiver-operating characteristic curve, or **ROC curve** is a plot where the sensitivity of the model is shown on the y-axis and the false positive ratio is shown on the x-axis.

ROC curves enable us to assess the performance of the model over its entire operating range. In other words, we can see what happens to our model's performance with thresholds from 0.0 to 1.0.

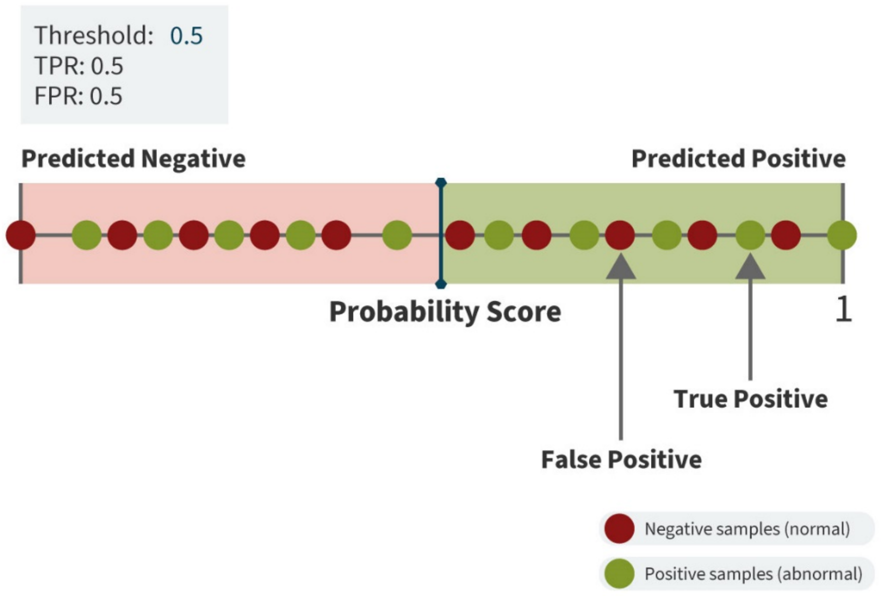


The area under the ROC curve (ROC-AUC or AUROC) gives us a single number that summarizes the efficacy of our model as measured by the ROC curve.

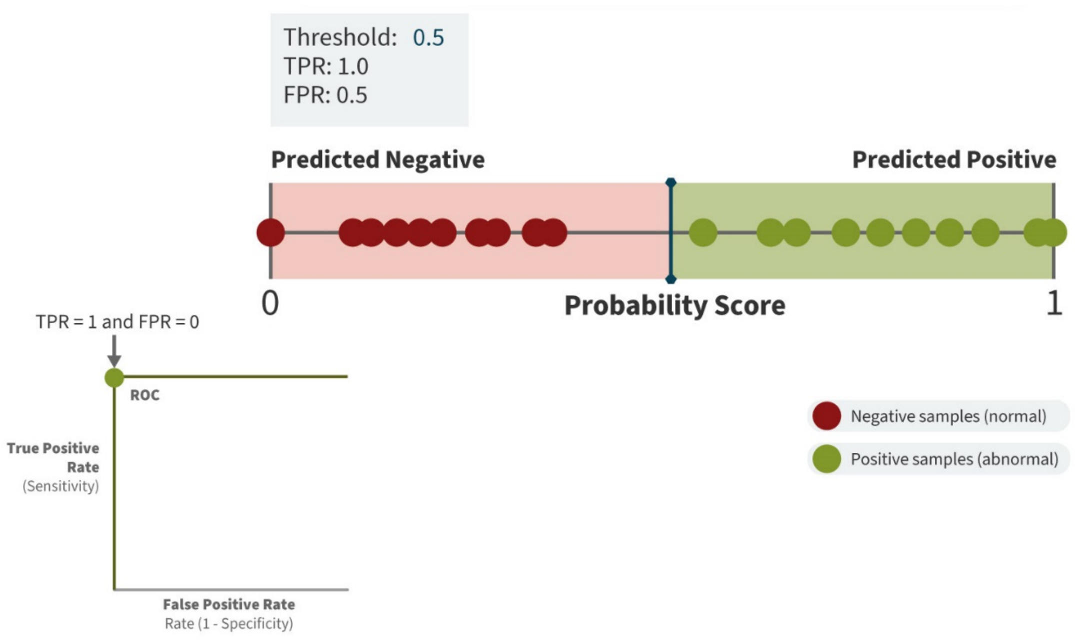
- The maximum AUROC achievable is 1.0– a perfect classifier
- A random AUROC is 0.5– a completely random classifier

How ROC curves work:

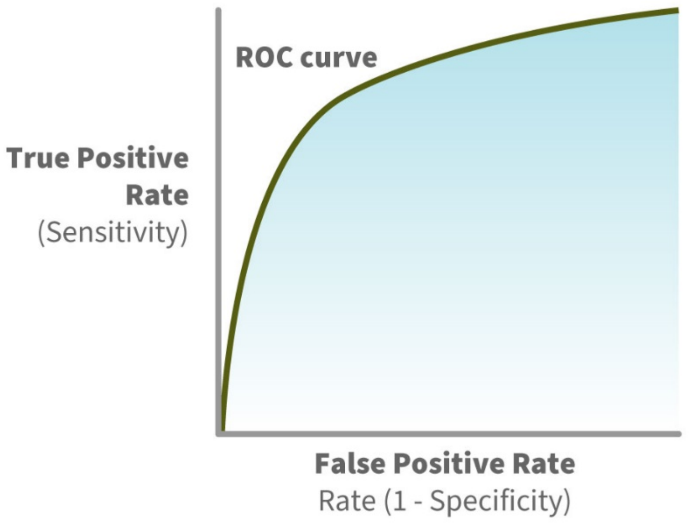
- Our random classifier evaluated on this dataset, will output a random probability score of each example being a positive that's between 0 and 1. We can visualize where each example falls on the probability spectrum like this



- If we set our threshold to be 0.5 such that all examples with scores above 0.5 are predicted to be positives, in this case you can see that the true positive rate (in other words the number of true positives, which are the red dots in the predicted positive region, divided by the number of actual positives, which are the red dots everywhere) will be about 0.5. Similarly, looking at the red dots, the false positive rate (the number of false positives divided by the number of actual negatives) will be about 0.5
- If we increase our threshold, our true positive rate increases to 0.75, but the FPR also increases to 0.75. In other words, decreasing the threshold increases true positives, but many more false positives are predicted as well. This leads to another point on the ROC curve where TPR and FPR are now both 0.75. In general, every adjustment to the threshold to increase TPR equivalently increases FPR, which creates the diagonal line ROC curve that we showed earlier, with Area under the Curve of 0.5
- A perfect classifier, one that predicts a probability score between 0.5 and 1 for every actual positive and a probability between 0 and 0.5 for every actual negative, would have a very different probability spectrum
- For all thresholds of interest, we have a TPR of 1 and a FPR of 0. Thus, the AUROC of this classifier looks like the following:



- You are unlikely to have either a random classifier or a perfect one. The ROC for a "good" classifier would be something like this, with AUC of 0.9



- The probability scores output by the model largely separate the examples, but the model cannot perfectly discriminate between the two classes. So at a threshold of 0.5, you'll have a TPR of 0.8 and a FPR of 0.2, for example, and at a threshold of 0.4, you'll get a higher TPR of 0.9, but also a less desirable FPR of 0.4

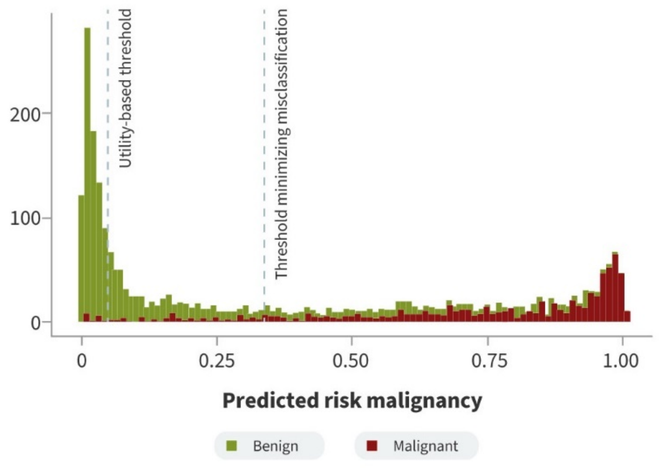
There is a fundamental tradeoff between TPR and FPR: as you increase sensitivity, or TPR, you typically decrease specificity, or 1-FPR. The threshold you choose for the classifier implementation of predicting positive vs. negative is also referred to as the **operating point**.

Comparing the performance between two or more classifiers:

- Method 1: Performance characteristics at particular operating points can be compared
- Method 2: Overall performance can be compared using the AUC.
- **Caution:** many published reports compare AUCs in absolute terms: "Classifier 1 has an AUC of 0.9, and classifier 2 has an AUC of 0.8, so classifier 1 is clearly better". But this does not necessarily hold. Statistical analyses are necessary to verify that these claims are significant

Choosing an operating point:

- Example 1: If we are choosing an operating point for a classifier to screen for cancer, for example, we'd probably rather put up with a higher false positive rate in order to make sure we catch as many true positives as possible as well, since it's most important to identify potential cancer sufferers.
- Example 2: If our classifier will be used to make a decision about whether to pursue a high-risk treatment or not, we probably want a different threshold of sensitivity / specificity trade-off, since we don't want to subject a patient to unnecessary risks unless we are very certain that they need it.



1. By **Minimizing Misclassification**: choose an operating point that minimizes the sum of false positives and false negatives.
2. By **Optimizing Utility:** assign a different utility value for each type of outcome and optimize total utility. NOTE: the utility values are usually subjective!

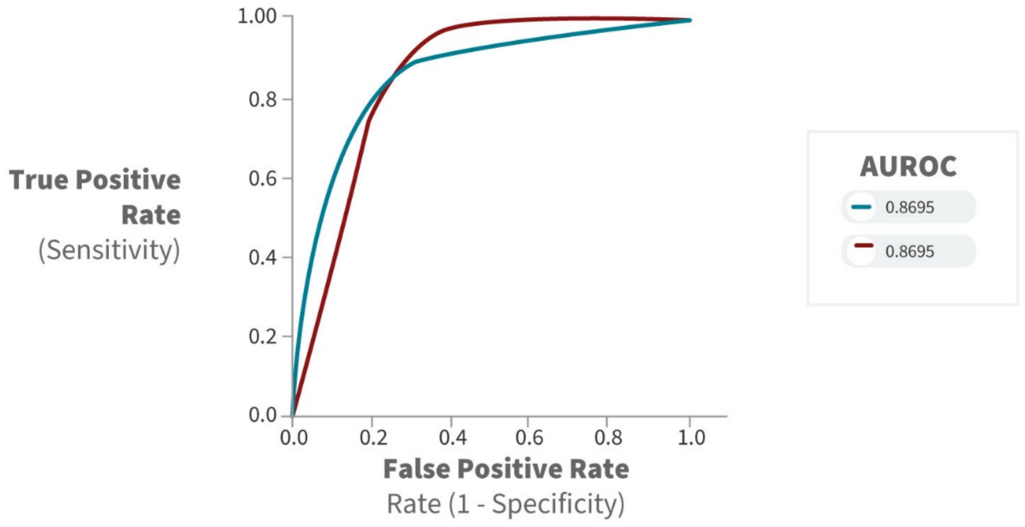
Frequencies of predicted risk of malignancy and possible risk thresholds

- Choosing an operating point can be based on maximizing a utility measurement. We can subjectively measure utility by manually assigning a value to true positives, false positives, true negatives, and false negatives

- The utility or cost of different possible outcomes can be expressed in any framework that makes sense for the clinical problem at hand

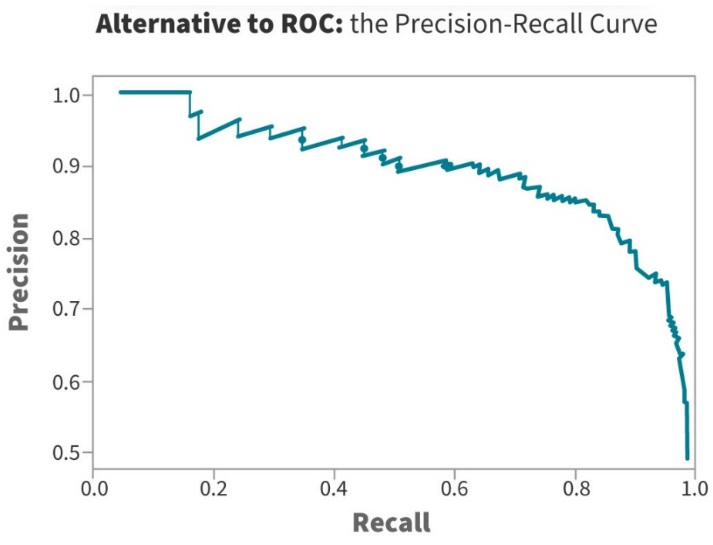
Other important considerations when using ROC curves:

- The shape of ROC curves can matter in evaluation, for example, a classifier can have a lower AUROC than another classifier yet a higher utility because of the shape of its ROC curve



  - At the far left of the plot, the model doesn't predict any positives and so both the TPR and the FPR are zero
  - At the far right of the plot, where the model classifies everything as positive, we have a TPR of 1, but also a FPR of 1 since all true negatives are predicted as positive in this case
  - The shape of the curve as we progress from left to right tells us about the tradeoff between TPR and FPR. If a model's ROC curve arcs towards the top-left more than another model's curve, that tells us that the first model achieves a higher TPR than the second at some fixed FPR
- ROC curves can also be misleading with imbalanced datasets, where the number of true positive labels is very different from the number of true negative labels

Precision recall (PR) curves are more robust to imbalance data.



- The y-axis describes precision and the x-axis describes recall at various thresholds
- PR curves cross each other more frequently and it can be more difficult to interpret and compare. However, overall, a curve that appears above another curve in a PR plot generally corresponds to better performance.
- The key difference: the number of true-negative results is not used for making a PRC
- And similar to ROC curves, the AUC (the area under the precision-recall curve) score can be used as a single performance measure for precision-recall curves. As the name indicates, it is an area under the curve calculated in the precision-recall space

The best practice is to evaluate and report both the AUROC curve and area under the PR curve, along with statistical error bars around the average classifier performance so that the most complete information on the performance can be evaluated in general.

CITATIONS AND ADDITIONAL READINGS

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: data mining, inference, and prediction.* New York: Springer.

Wang, F., Kaushal, R., & Khullar, D. (2020). Should health care demand interpretable artificial intelligence or accept "black box" medicine?. https://www.acpjournals.org/doi/10.7326/M19-2548