performance Metrics

Outline

1. Metrics

2. Precision and recall

3. Receiver Operating Characteristic (ROC) curves

Outline

1. **Metrics**

2. Precision and recall

3. Receiver Operating Characteristic (ROC) curves

Metrics

It is extremely important to use **quantitative metrics** for evaluating a machine learning model

- Until now, we relied on the **cost function value** for regression and classification
- Other metrics can be used to **better evaluate** and understand the model
- **For classification**
  - ✔ Accuracy/Precision/Recall/F1-score, ROC curves,…
- **For regression**
  - ✔ Normalized RMSE, Normalized Mean Absolute Error (NMAE),…

Classification case: metrics for skewed classes

Disease dichotomic classification example

Train logistic regression model $h(\boldsymbol{x})$, with $y = 1$ if disease, $y = 0$ otherwise.

Find that you got 1% error on test set (99% correct diagnoses)

Only 0.50% of patients **actually have** disease

The $y = 1$ class has very few examples with respect to the $y = 0$ class

If I use a predictor that **predicts always the $\mathbf{0}$ class,** I get 99.5% of accuracy!!

For **skewed classes,** the accuracy metric can be deceptive

Outline

1. Metrics

2. **Precision and recall**

3. Receiver Operating Characteristic (ROC) curves

Precision and recall

Suppose that $y = 1$ in presence of a **rare class** that we want to detect

Precision *(How much we are precise in the detection)*

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

$$\frac{\text{True Positive}}{\text{\# Predicted Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall *(How much we are good at detecting)*

Of all patients that actually have the disease, what fraction did we correctly detect as having the disease?

$$\frac{\text{True Positive}}{\text{\# Actual Positive}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Confusion matrix

Actual class

Predicted class

	1 (p)	0 (n)
1 (Y)	True positive (TP)	False positive (FP)
0 (N)	False negative (FN)	True negative (TN)

Trading off precision and recall

Logistic regression: $0 \le h(\boldsymbol{x}) \le 1$

- Predict 1 if $h(\boldsymbol{x}) \ge 0.5$
- Predict 0 if $h(\boldsymbol{x}) < 0.5$

These thresholds can be different from 0.5!

$\rightarrow$ *At different thresholds, correspond different confusion matrices!*

Suppose we want to predict $y = 1$ (disease) only if very confident

- Increase threshold $\rightarrow$ Higher precision, lower recall

Suppose we want to avoid missing too many cases of disease (avoid false negatives).

- Decrease threshold $\rightarrow$ Higher recall, lower precision

F1-score

It is usually better to compare models by means of one number only. The $F1 - \text{score}$ can be used to combine precision and recall

	Precision(P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	$\boxed{0.444}$
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

The best is Algorithm 1

Algorithm 3 predict always 1

Average says not correctly that Algorithm 3 is the best

$$\text{Average} = \frac{P+R}{2}$$

$$F_1\text{score} = 2\,\frac{PR}{P+R}$$

- $P = 0 \text{ or } R = 0 \Rightarrow F_1\text{score} = 0$
- $P = 1 \text{ and } R = 1 \Rightarrow F_1\text{score} = 1$

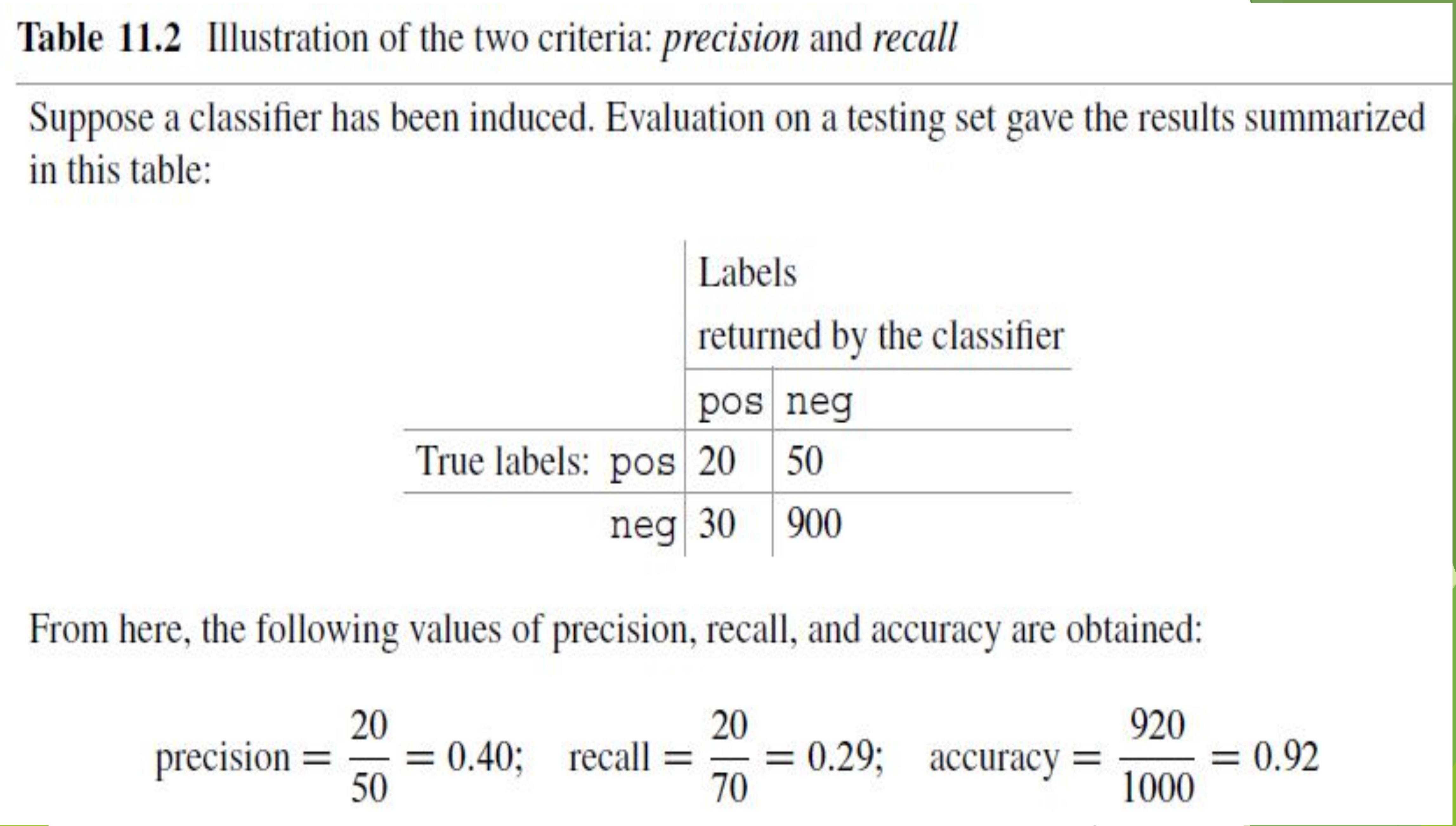


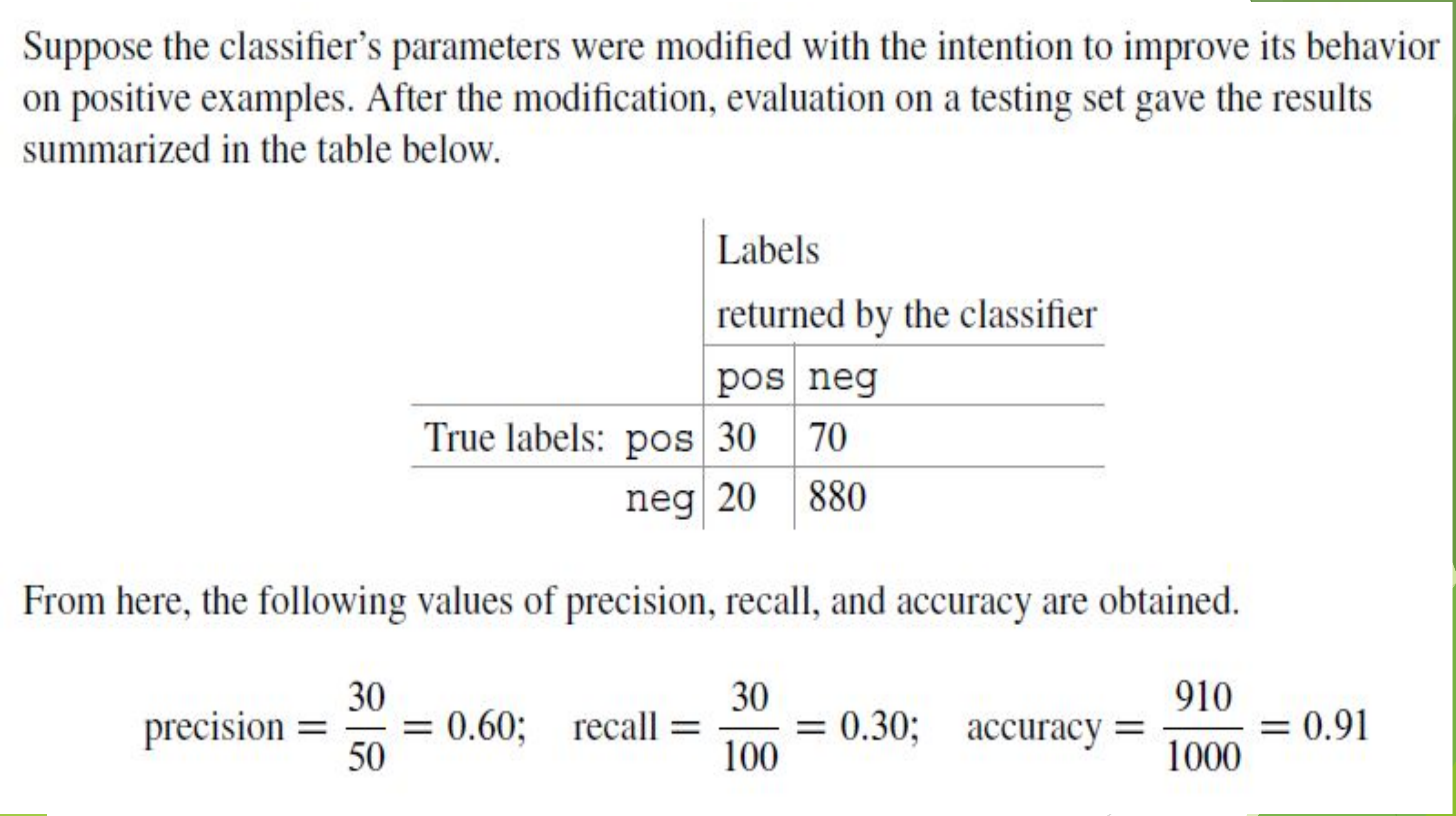
Table 11.2 Illustration of the two criteria: *precision* and *recall*

Suppose a classifier has been induced. Evaluation on a testing set gave the results summarized in this table:

		Labels returned by the classifier	
		pos	neg
True labels:	pos	20	50
	neg	30	900

From here, the following values of precision, recall, and accuracy are obtained:

$$\text{precision} = \frac{20}{50} = 0.40; \quad \text{recall} = \frac{20}{70} = 0.29; \quad \text{accuracy} = \frac{920}{1000} = 0.92$$



Suppose the classifier's parameters were modified with the intention to improve its behavior on positive examples. After the modification, evaluation on a testing set gave the results summarized in the table below.

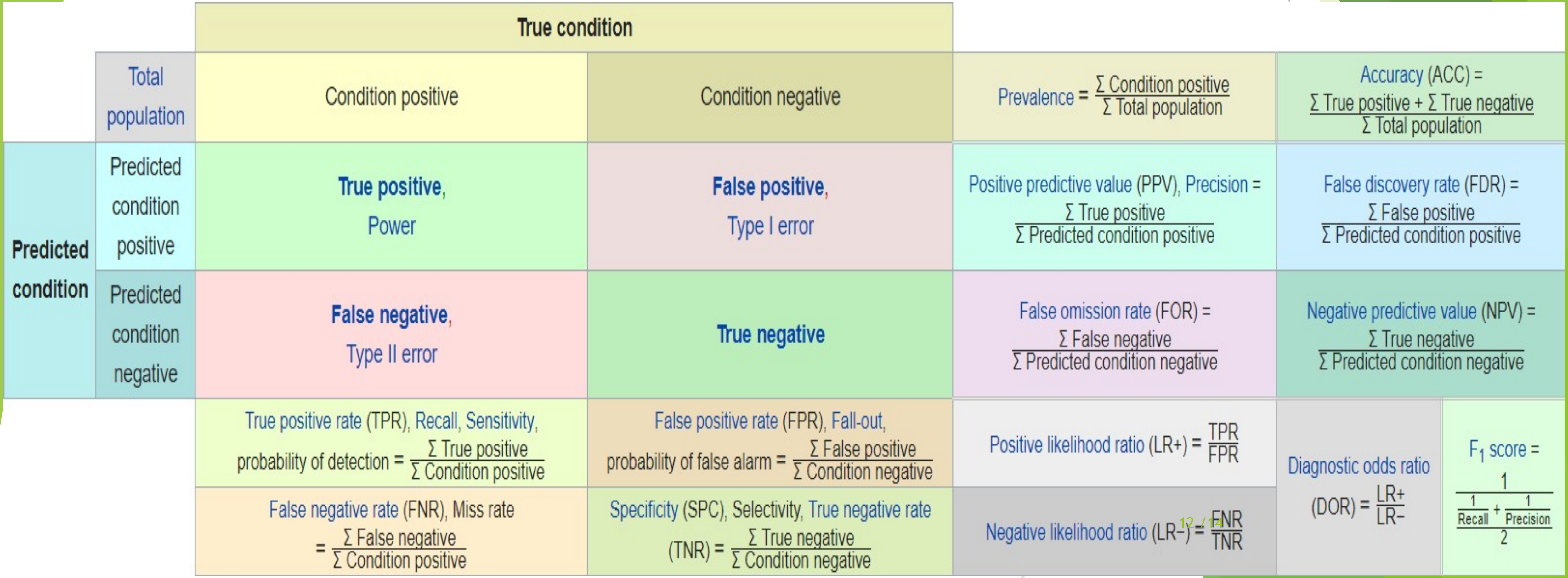
	Labels returned by the classifier	
	pos	neg
True labels: pos	30	70
neg	20	880

From here, the following values of precision, recall, and accuracy are obtained.

$$\text{precision} = \frac{30}{50} = 0.60; \quad \text{recall} = \frac{30}{100} = 0.30; \quad \text{accuracy} = \frac{910}{1000} = 0.91$$

Summaries of the confusion matrix

Different metrics can be computed from the confusion matrix, depending on the class of interest *(https://en.wikipedia.org/wiki/Precision_and_recall)*



		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Outline

1. Metrics

2. Precision and recall

3. **Receiver Operating Characteristic (ROC) curves**

Ranking instead of classifying

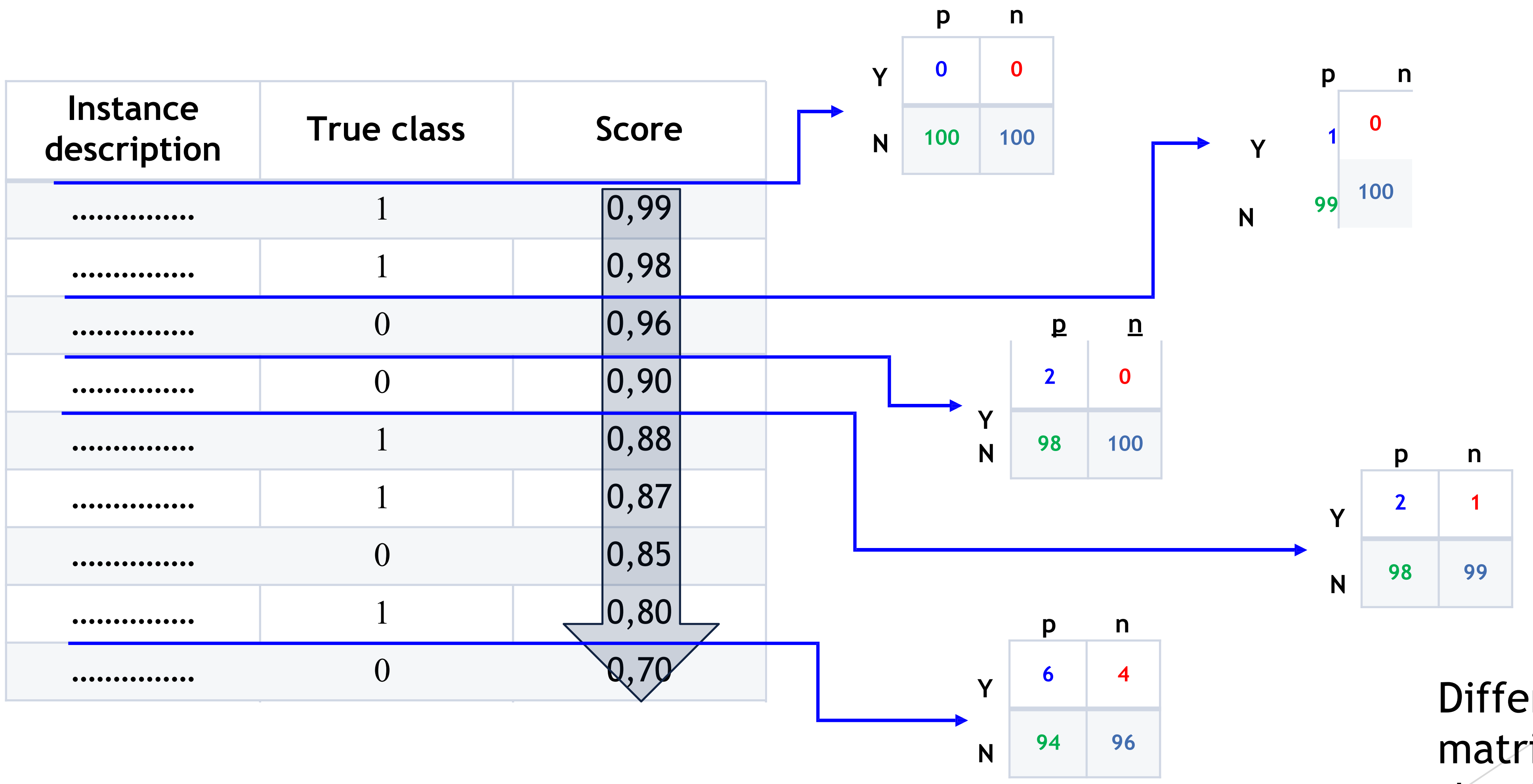
Classifiers such as logistic regression can output a **probability** of belonging to a class (or something similar).

- We can use this to **rank** the different istances and take actions on the cases at top of the list

- We may have a **budget**, so we have to target most promising individuals

- Ranking enables to use different techniques for **visualizing** model performance

Ranking instead of classifying



Instance description	True class	Score
....	1	0,99
....	1	0,98
....	0	0,96
....	0	0,90
....	1	0,88
....	1	0,87
....	0	0,85
....	1	0,80
....	0	0,70

$$\begin{array}{c|cc} & p & n \\ \hline Y & 0 & 0 \\ N & 100 & 100 \end{array}$$

$$\begin{array}{c|cc} & p & n \\ \hline Y & 1 & 0 \\ N & 99 & 100 \end{array}$$

$$\begin{array}{c|cc} & p & n \\ \hline Y & 2 & 0 \\ N & 98 & 100 \end{array}$$

$$\begin{array}{c|cc} & p & n \\ \hline Y & 2 & 1 \\ N & 98 & 99 \end{array}$$

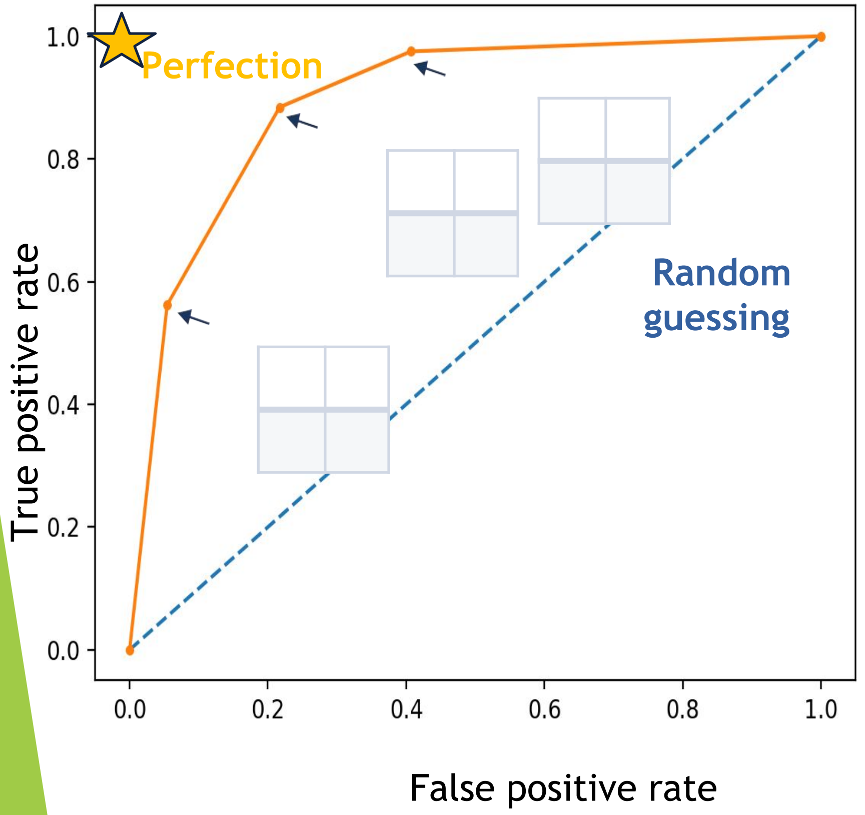
$$\begin{array}{c|cc} & p & n \\ \hline Y & 6 & 4 \\ N & 94 & 96 \end{array}$$

Adapated from [1]

Different confusion matrices by changing the **threshold**

ROC curves

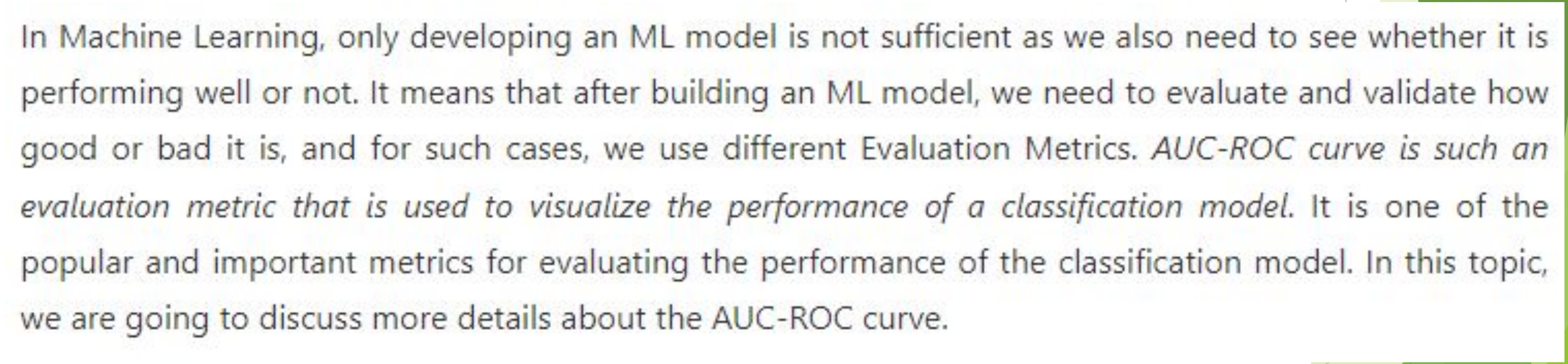
ROC curves are a very general way to **represent and compare** the performance of different models (on a binary classification task)



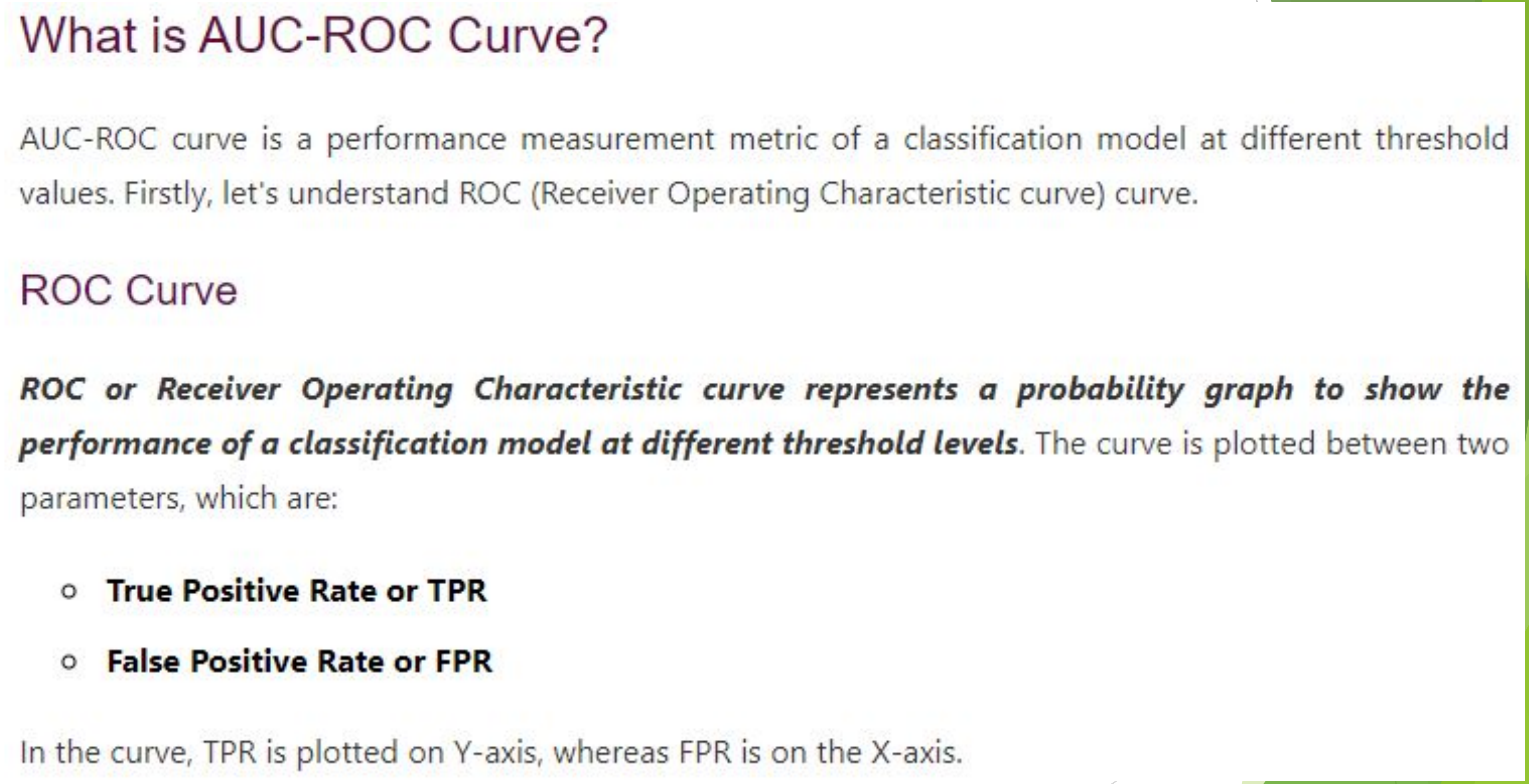
Observations

- 0,0 : (predict always negative)
- 1,1 (: predict always positive)
- Diagonal line: random classifier
- Below diagonal line: worse than random classifier
- Different classifiers can be compared
- Area Under the Curve (AUC): probability that a randomly chosen positive instance will be ranked ahead of randomly chosen negative instance

AUC-ROC Curve in Machine Learning



In Machine Learning, only developing an ML model is not sufficient as we also need to see whether it is performing well or not. It means that after building an ML model, we need to evaluate and validate how good or bad it is, and for such cases, we use different Evaluation Metrics. *AUC-ROC curve is such an evaluation metric that is used to visualize the performance of a classification model.* It is one of the popular and important metrics for evaluating the performance of the classification model. In this topic, we are going to discuss more details about the AUC-ROC curve.



What is AUC-ROC Curve?

AUC-ROC curve is a performance measurement metric of a classification model at different threshold values. Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve.

ROC Curve

ROC or Receiver Operating Characteristic curve represents a probability graph to show the performance of a classification model at different threshold levels. The curve is plotted between two parameters, which are:

- **True Positive Rate or TPR**
- **False Positive Rate or FPR**

In the curve, TPR is plotted on Y-axis, whereas FPR is on the X-axis.

TPR:

TPR or True Positive rate is a synonym for Recall, which can be calculated as:

$$TPR = \frac{TP}{TP + FN}$$

FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

Here, **TP**: True Positive

FP: False Positive

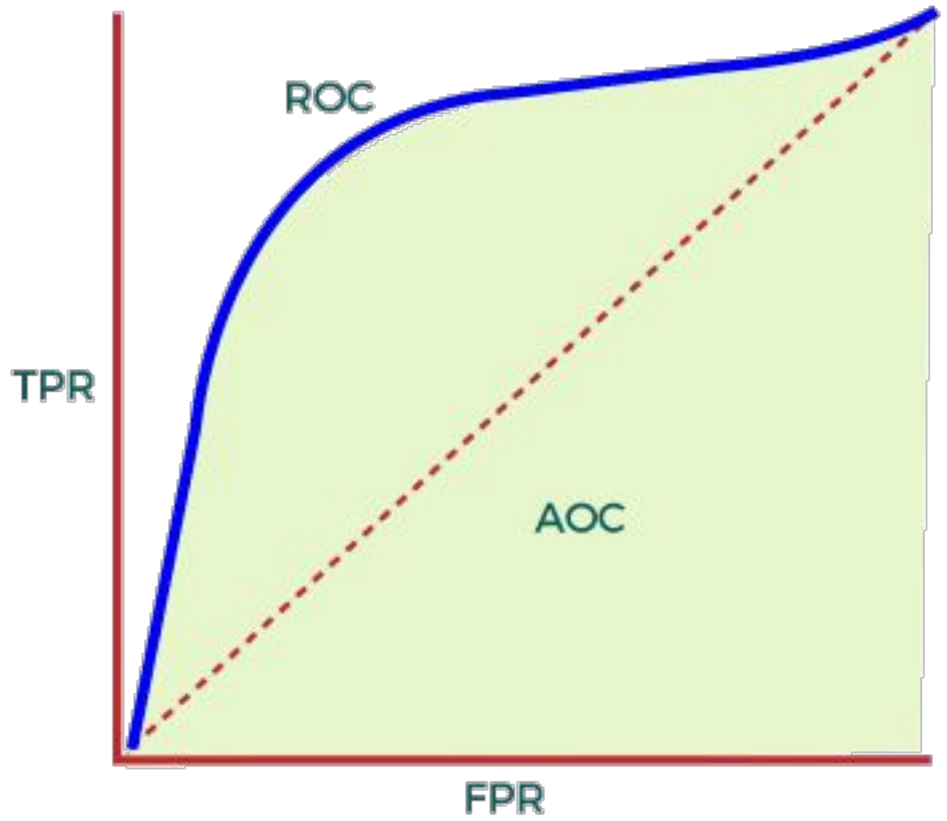
TN: True Negative

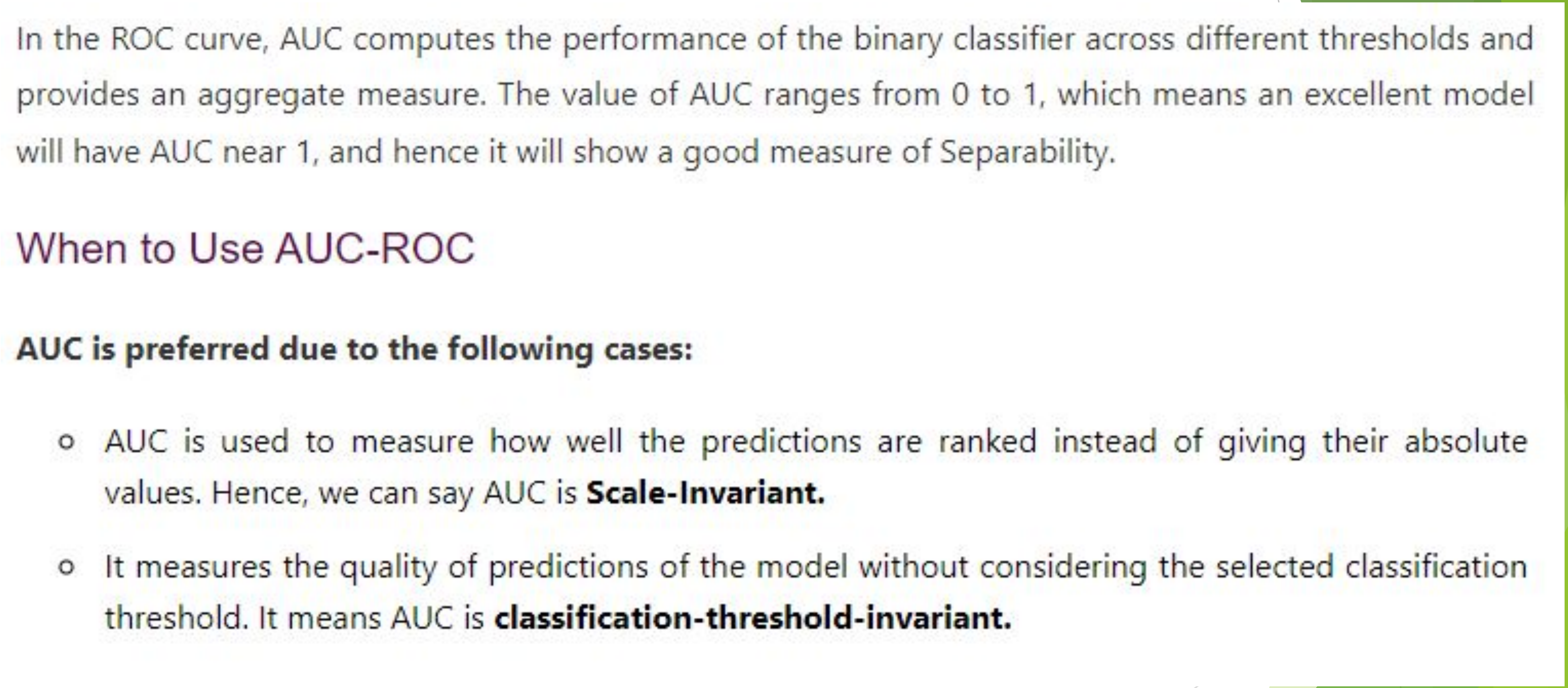
FN: False Negative

Now, to efficiently calculate the values at any threshold level, we need a method, which is AUC.

AUC: Area Under the ROC curve

AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve ranging from (0,0) to (1,1), as shown below image:





In the ROC curve, AUC computes the performance of the binary classifier across different thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1, which means an excellent model will have AUC near 1, and hence it will show a good measure of Separability.

When to Use AUC-ROC

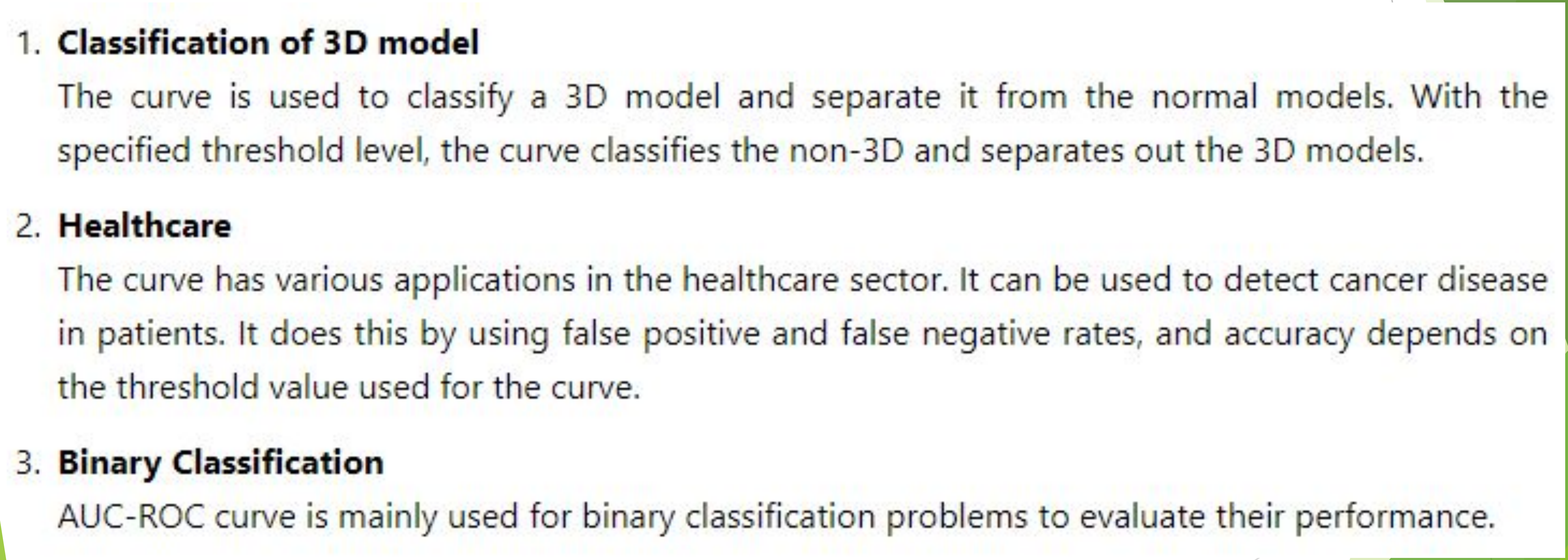
AUC is preferred due to the following cases:

- AUC is used to measure how well the predictions are ranked instead of giving their absolute values. Hence, we can say AUC is **Scale-Invariant.**
- It measures the quality of predictions of the model without considering the selected classification threshold. It means AUC is **classification-threshold-invariant.**

Applications of AUC-ROC Curve

Although the AUC-ROC curve is used to evaluate a classification model, it is widely used for various applications.

Some of the important applications of AUC-ROC are given below:



1. **Classification of 3D model**

   The curve is used to classify a 3D model and separate it from the normal models. With the specified threshold level, the curve classifies the non-3D and separates out the 3D models.

2. **Healthcare**

   The curve has various applications in the healthcare sector. It can be used to detect cancer disease in patients. It does this by using false positive and false negative rates, and accuracy depends on the threshold value used for the curve.

3. **Binary Classification**

   AUC-ROC curve is mainly used for binary classification problems to evaluate their performance.

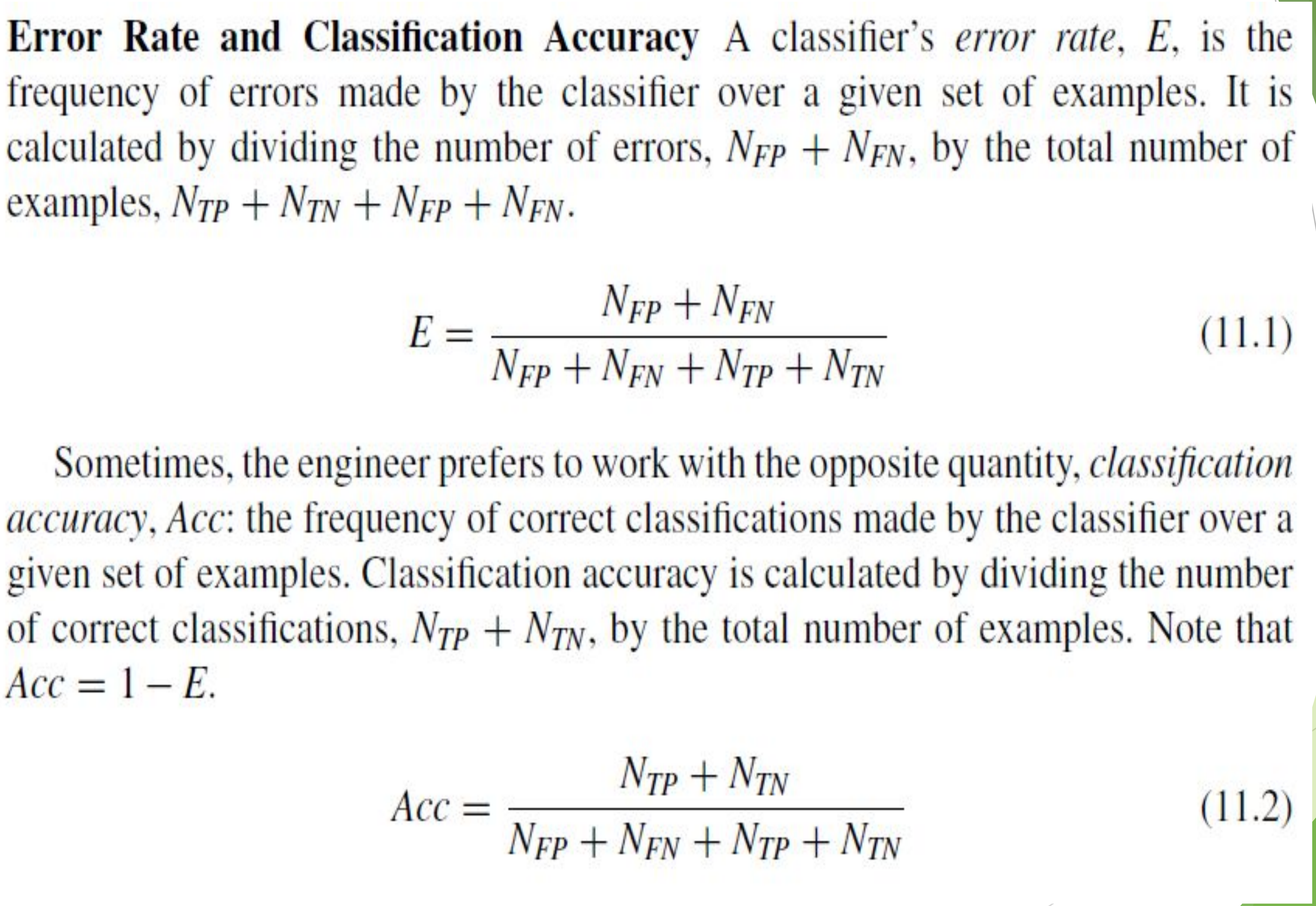
Sensitivity and Specificity The choice of the concrete criterion is often influenced by the given application field—with its specific needs and deep-rooted traditions that should not be ignored. Thus the medical domain has become accustomed to assessing performance of their "classifiers" (not necessarily developed by machine learning) by *sensitivity* and *specificity*. In essence, these quantities are nothing but *recall* measured on the positive and negative examples, respectively. Let us be more concrete:

Sensitivity is *recall* measured on positive examples:

$$Se = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{11.7}$$

Specificity is *recall* measured on negative examples:

$$Sp = \frac{N_{TN}}{N_{TN} + N_{FP}} \tag{11.8}$$



Error Rate and Classification Accuracy A classifier's *error rate*, E, is the frequency of errors made by the classifier over a given set of examples. It is calculated by dividing the number of errors, $N_{FP} + N_{FN}$, by the total number of examples, $N_{TP} + N_{TN} + N_{FP} + N_{FN}$.

$$E = \frac{N_{FP} + N_{FN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}} \tag{11.1}$$

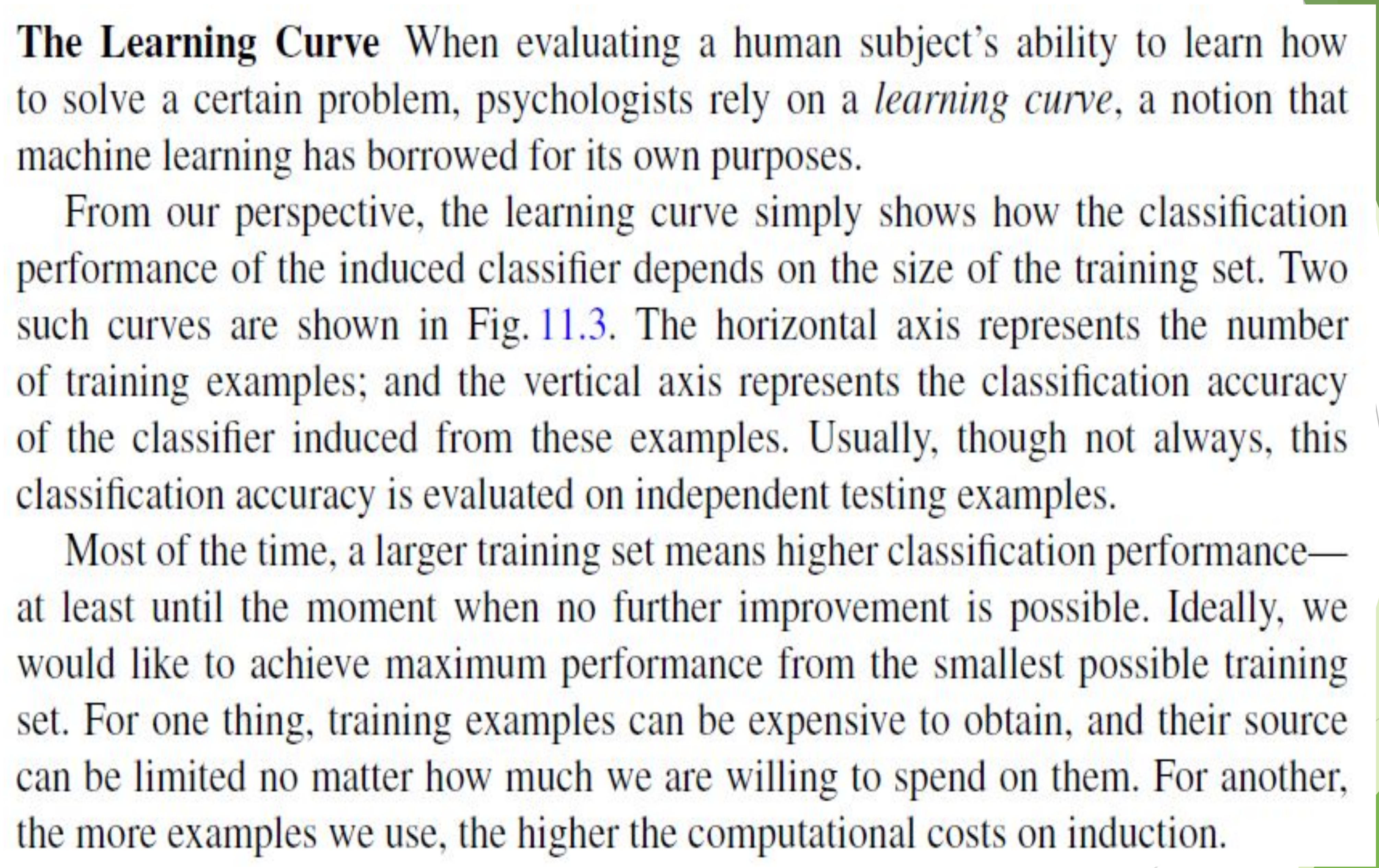
Sometimes, the engineer prefers to work with the opposite quantity, *classification accuracy*, Acc: the frequency of correct classifications made by the classifier over a given set of examples. Classification accuracy is calculated by dividing the number of correct classifications, $N_{TP} + N_{TN}$, by the total number of examples. Note that $Acc = 1 - E$.

$$Acc = \frac{N_{TP} + N_{TN}}{N_{FP} + N_{FN} + N_{TP} + N_{TN}} \tag{11.2}$$

Learning Curves and Computational Costs

Let us now turn our attention to the evaluation of the learning algorithm itself. How efficient is the given induction technique computationally? And how good are the classifiers it induces? Will better results be achieved if we choose some other machine-learning framework?

.

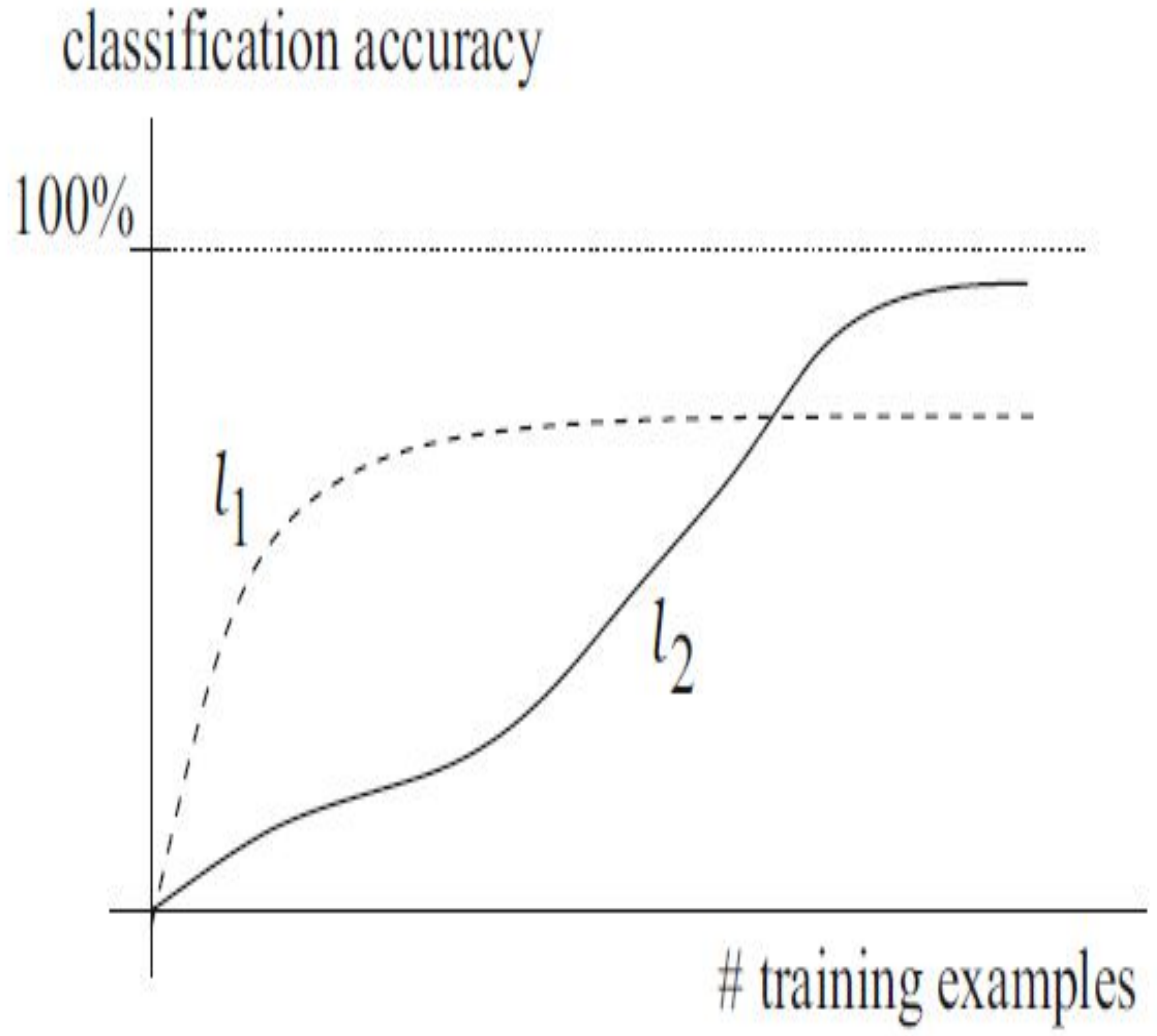


The Learning Curve When evaluating a human subject's ability to learn how to solve a certain problem, psychologists rely on a *learning curve*, a notion that machine learning has borrowed for its own purposes.

From our perspective, the learning curve simply shows how the classification performance of the induced classifier depends on the size of the training set. Two such curves are shown in Fig. 11.3. The horizontal axis represents the number of training examples; and the vertical axis represents the classification accuracy of the classifier induced from these examples. Usually, though not always, this classification accuracy is evaluated on independent testing examples.

Most of the time, a larger training set means higher classification performance—at least until the moment when no further improvement is possible. Ideally, we would like to achieve maximum performance from the smallest possible training set. For one thing, training examples can be expensive to obtain, and their source can be limited no matter how much we are willing to spend on them. For another, the more examples we use, the higher the computational costs on induction.

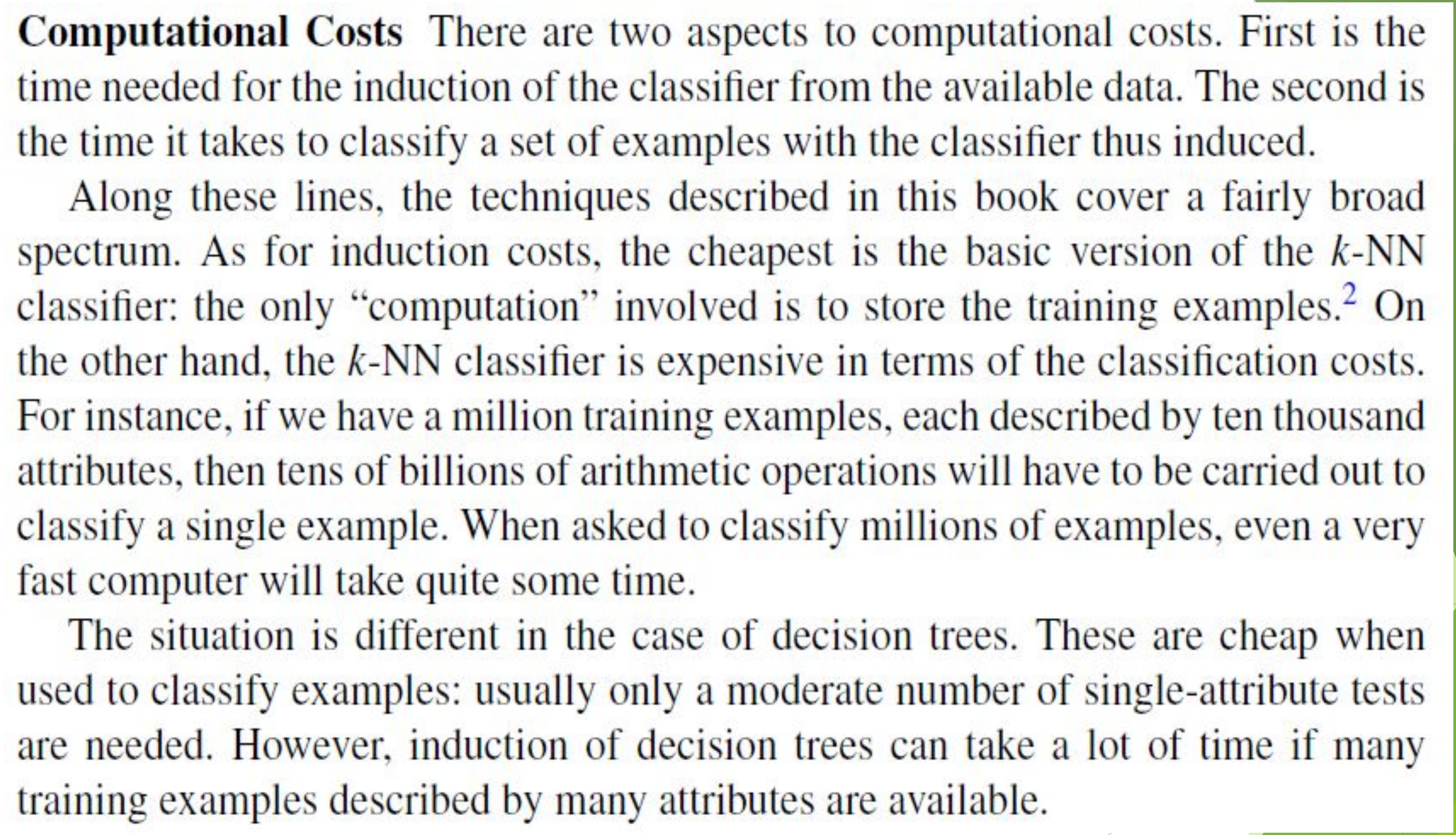
Fig. 11.3 A learning curve shows how the achieved classification accuracy depends on the number of examples used during learning



Comparing Learners with Different Learning Curves Figure 11.3 shows the learning curves of two learners, $l1$ and $l2$. The reader can see that the learning curve of the former, $l1$, rises very quickly, only to level off at a point beyond which virtually no improvement is possible—the limitation may be imposed by an incorrect bias.

By contrast, the learning curve of the second learner, $l2$, does not grow so fast, but in the end achieves higher levels of accuracy than $l1$.

Which of the two curves indicates a preferable learner depends on the circumstances of the given application. When the source of the training examples is limited, the first learner is clearly more appropriate. If the examples are abundant, we will prefer the other learner, assuming of course that the computational costs are not prohibitive.



Computational Costs There are two aspects to computational costs. First is the time needed for the induction of the classifier from the available data. The second is the time it takes to classify a set of examples with the classifier thus induced.

Along these lines, the techniques described in this book cover a fairly broad spectrum. As for induction costs, the cheapest is the basic version of the *k*-NN classifier: the only “computation” involved is to store the training examples.[2] On the other hand, the *k*-NN classifier is expensive in terms of the classification costs. For instance, if we have a million training examples, each described by ten thousand attributes, then tens of billions of arithmetic operations will have to be carried out to classify a single example. When asked to classify millions of examples, even a very fast computer will take quite some time.

The situation is different in the case of decision trees. These are cheap when used to classify examples: usually only a moderate number of single-attribute tests are needed. However, induction of decision trees can take a lot of time if many training examples described by many attributes are available.

Mean Square Error

- The Mean Square Error(MSE) or L2 loss is a loss function that quantifies the magnitude of the error between a machine learning algorithm prediction and an actual output by taking the average of the squared difference between the predictions and the target values. Squaring the difference between the predictions and actual target values results in a higher penalty assigned to more significant deviations from the target value.

- MSE = $(1/n) * \Sigma(y_i - \bar{y})^2$
- Where:
- n is the number of samples in the dataset
- y_i is the predicted value for the i-th sample
- $\bar{y}$ is the target value for the i-th sample

Mean Absolute Error (MAE)

- Mean Absolute Error (MAE), also known as L1 Loss, is a loss function used in regression tasks that calculates the average absolute differences between predicted values from a machine learning model and the actual target values. Unlike Mean Squared Error (MSE), MAE does not square the differences, treating all errors with equal weight regardless of their magnitude.
- $\text{MAE} = (1/n) * \Sigma|y_i - \bar{y}|$

Where:

- n is the number of samples in the dataset
- y_i is the predicted value for the i-th sample
- $\bar{y}$ is the target value for the i-th sample

Root Mean Square Error

Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

Differences among these evaluation metrics

- Mean Squared Error(MSE) and Root Mean Square Error penalizes the large prediction errors vi-a-vis Mean Absolute Error (MAE). However, RMSE is widely used than MSE to evaluate the performance of the regression model with other random models as it has the same units as the dependent variable (Y-axis).
- MSE is a differentiable function that makes it easy to perform mathematical operations in comparison to a non-differentiable function like MAE. Therefore, in many models, RMSE is used as a default metric for calculating Loss Function despite being harder to interpret than MAE.
- The lower value of MAE, MSE, and RMSE implies higher accuracy of a regression model. However, a higher value of R square is considered desirable.
- R Squared & Adjusted R Squared are used for explaining how well the independent variables in the linear regression model explains the variability in the dependent variable. R Squared value always increases with the addition of the independent variables which might lead to the addition of the redundant variables in our model. However, the adjusted R-squared solves this problem.
- Adjusted R squared takes into account the number of predictor variables, and it is used to determine the number of independent variables in our model. The value of Adjusted R squared decreases if the increase in the R square by the additional variable isn’t significant enough.
- For comparing the accuracy among different linear regression models, RMSE is a better choice than R Squared.