Overview of Machine Learning and Pattern Recognition

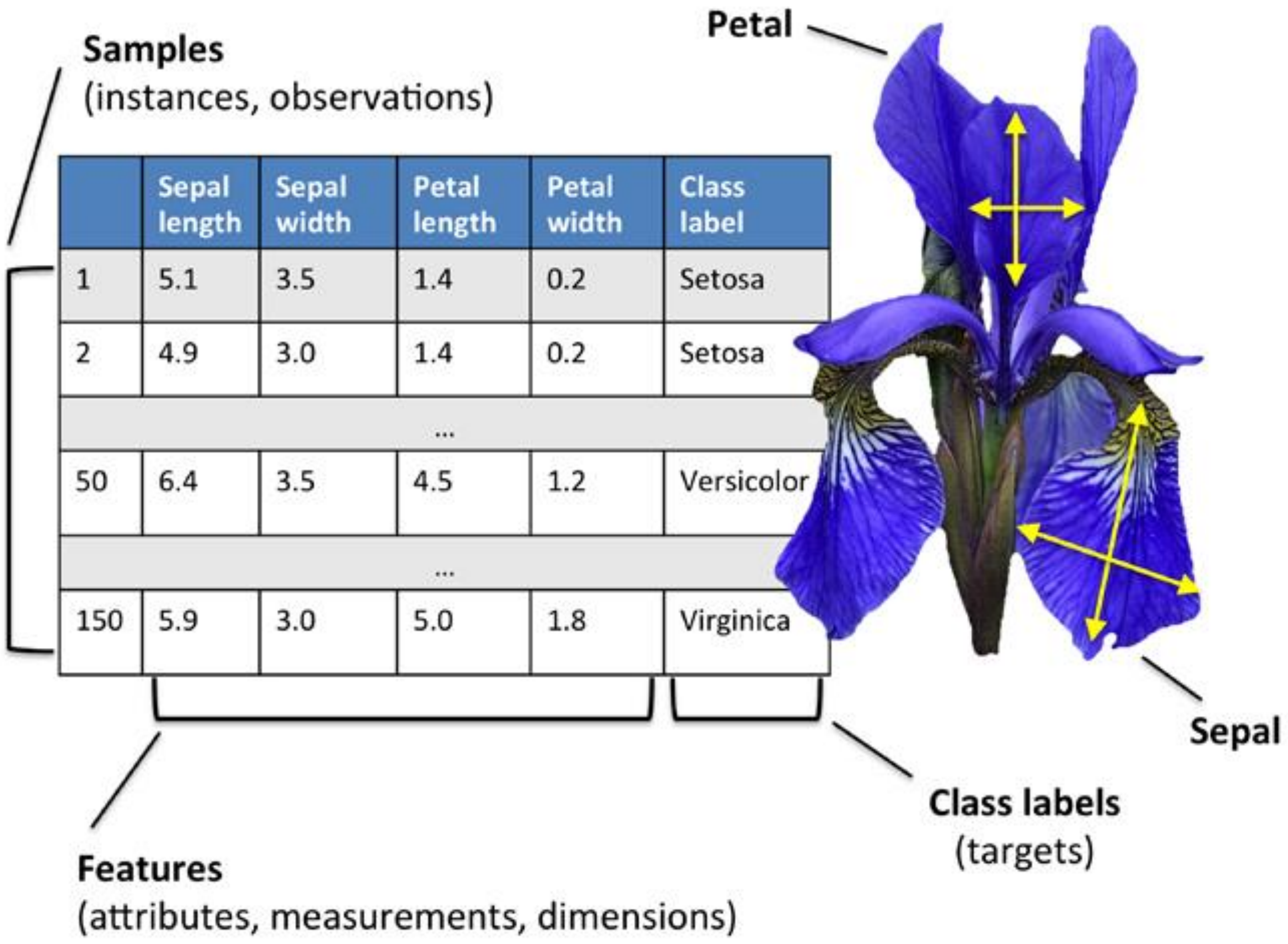
Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, ITALY

2. Classification problems: main concepts

Given a classification problem…

- How should the objects to be classified be represented?
- What algorithm can be used?
- How should learning (training) be done?
- How can classification performance be evaluated?

The most classic example: iris dataset



Samples
(instances, observations)
Petal
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

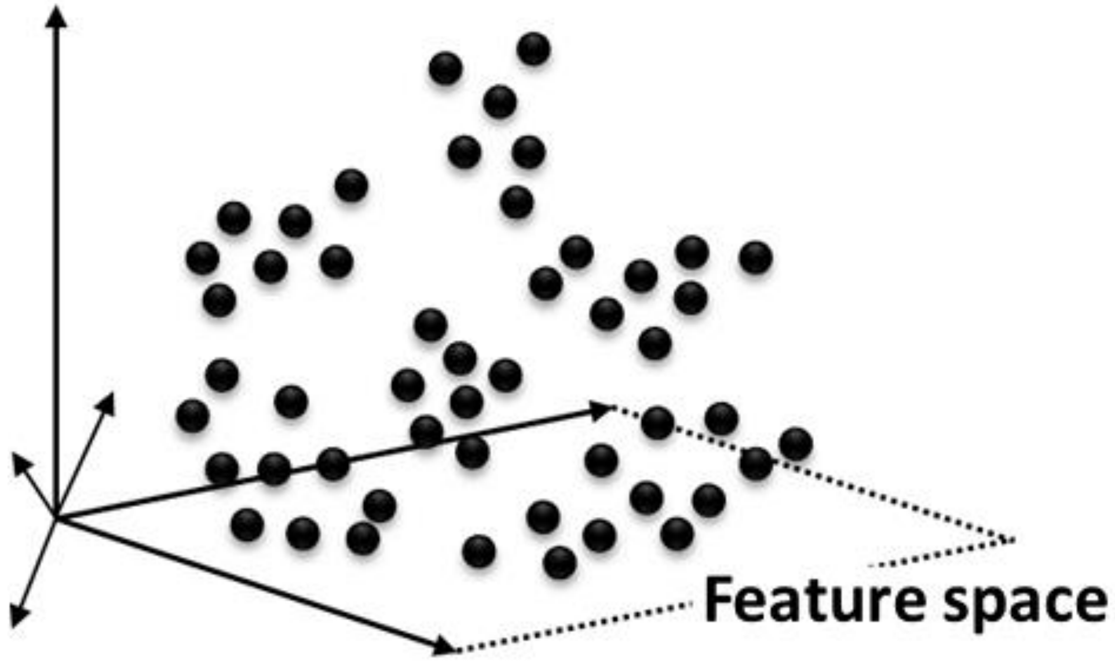
Feature vector

- An object to be classified is represented by a limited set of **features** (attributes or measurements), generally grouped into a numerical vector
- Example: the Iris dataset consists of 150 samples and 4 features → it can be represented as a set of 150 4-dimensional feature vectors, or as a 150x4 matrix:

$$
\begin{array}{cc}
\begin{bmatrix}
x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\
x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\
\vdots & \vdots & \vdots & \vdots \\
x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)}
\end{bmatrix} & \text{Sample 1} \\
\text{Feature 2} &
\end{array}
$$

Feature space

we can think of each sample represented as a point in a *n*-dimensional feature space where *n* is the number of features



Feature vector

- Features may be not be coherent w.r.t. each other
  - May have different units of measure
  - May be numerical or not
  - May be ordered (e.g. size є {small, medium, large}) or not ordered (e.g. color є {blu, red, green, etc.})
  - May be discrete, or continuous
  - May have different ranges
- In most of the cases, it makes sense to rescale the feature vector

e.g. remap all the features to [0,1] range:

```
for each feature $i$
    calculate $[min_i , max_i]$ over all the samples
    $\tilde{X}_i = \frac{X_i - min_i}{max_i - min_i}$
end
```

The concept of class

- A ***class*** is a set of objects, sharing some important properties. A class can be identified by
  - A **label**: a common categorical identifier (it can be a positive integer, a character, a string, etc.)

    e.g. {“Setosa”, “Versicolor”, “Virginica”}

    e.g. {1, 2, 3, …}

    e.g. {“Class 1”, “Class 2”, ….}

    e.g. {“Cancerous”, “Healthy”}
  - A **prototype**: an object representative of a specific class
    - It can either be a real object, or just an abstraction
    - It can be computed as the “center of gravity”
    - Typical approach for clustering problems

Class labels

- Each sample (i.e. feature vector) is associated to a class label or class prototype.
- If N is the number of samples of a dataset, the class labels can be represented as a N-dimensional vector
- For example, in the Iris dataset (150 samples, 3 classes):

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} \Big(y \in \{\text{Setosa},\ \text{Versicolor},\ \text{Virginica}\}$$

Representing data in a ML system

- Representing classification data in the form of matrices or vectors is just a convention. Each software might adopt its own reprepresentation form.
- For example, weka uses ARFF (Attribute-Relation File Format) files:

```
@RELATION iris
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}
```

The **header** contains the name of the relation, a list of the attributes and their types, the class labels

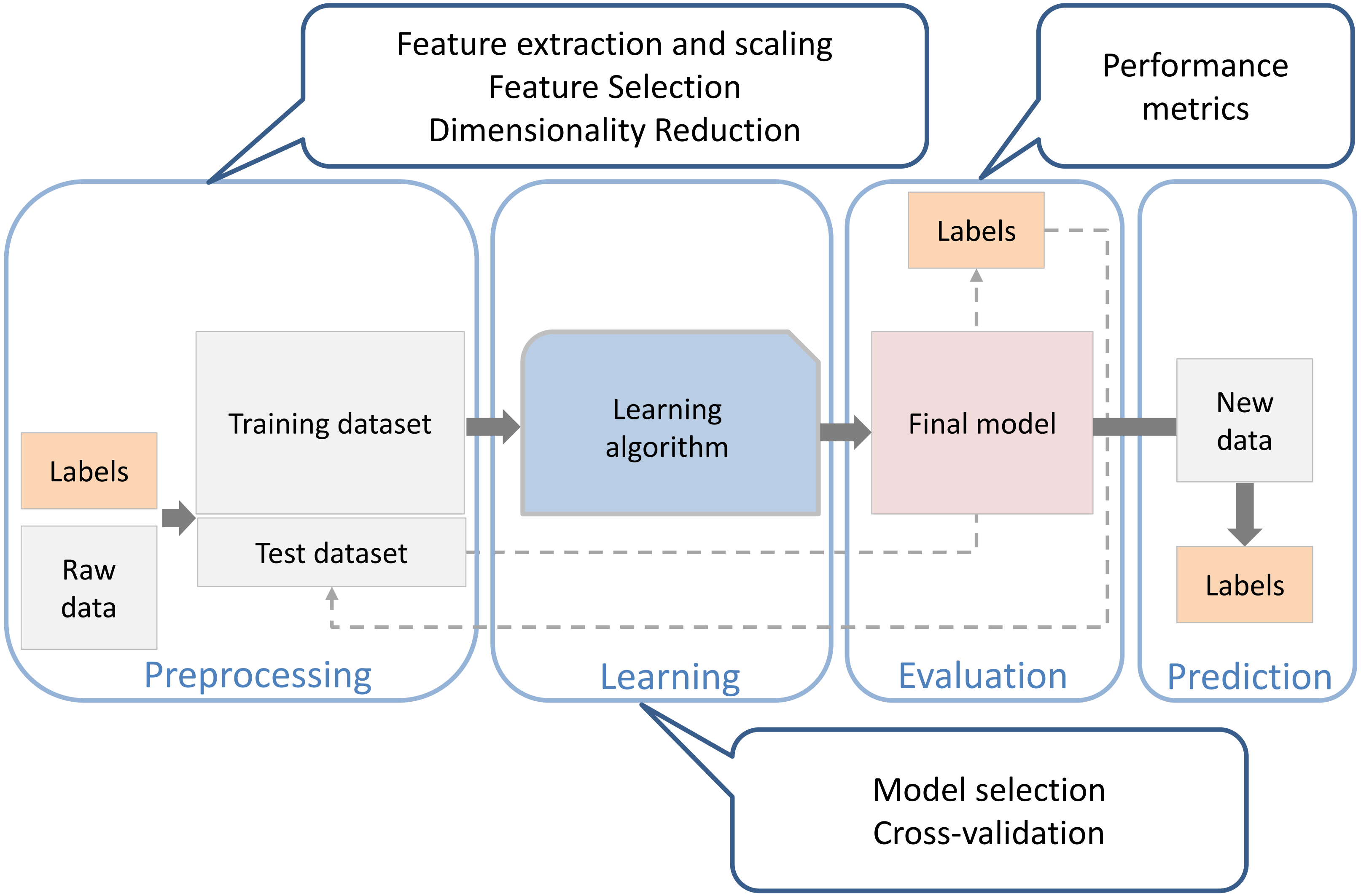
```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
(...)
```

Data: each feature vector is followed by corresponding class label

Representation CAVEAT

- Feature vector format has proved very convenient and “workable”
- Unfortunately, much real world data doesn’t arrive in neatly aligned feature vectors!
  - Sequences: events in time, genomes, books
  - Graphs: social networks, logistics, comms
  - Relational databases: patient’s health data distributed over many tables
- Each application requires preprocessing to
  - Extract relevant features
  - Represent features in a way that is most convenient to build an effective ML system

Building a ML system



Evaluating classification and predicting performance. Why?

- Multiple methods are available to classify or predict
- For each method, multiple choices are available for settings
- To choose best model, need to assess each model's performance

Evaluating classification and predicting performance. How?

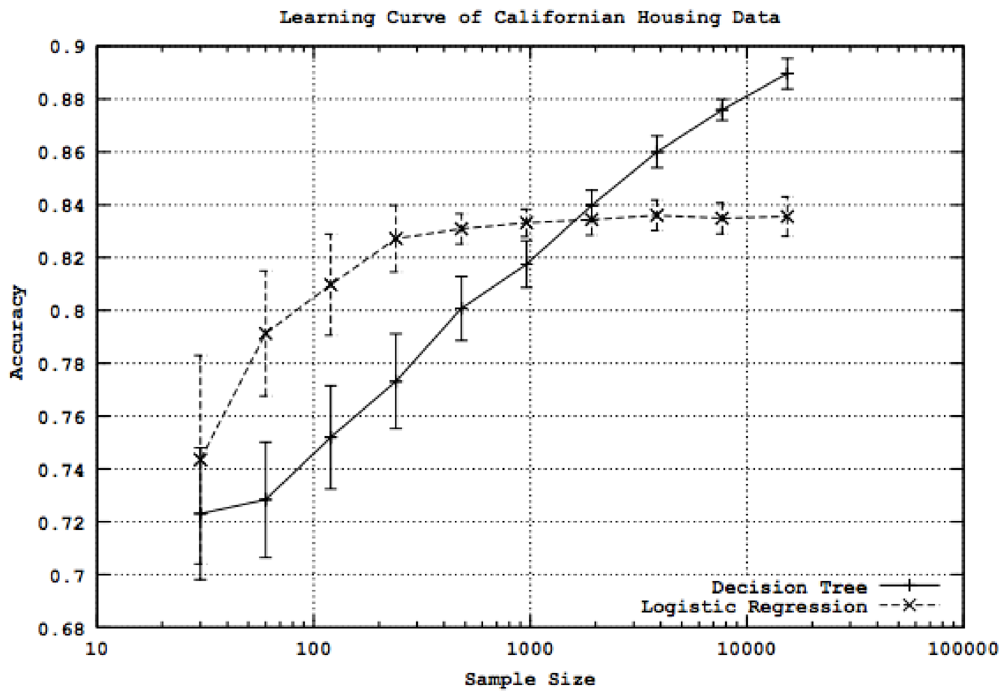
- How can we get an unbiased estimate of the accuracy of a learned model?
- when learning a model, you should pretend that you don’t have the test data yet
- if the test-set labels influence the learned model in any way, accuracy estimates will be biased!

Misclassification error

- **Error** = classifying a record as belonging to one class when it belongs to another class.
- **Error rate** = percent of misclassified records out of the total records in the test dataset
- **Accuracy rate** = 100 – Error rate

Learning curves

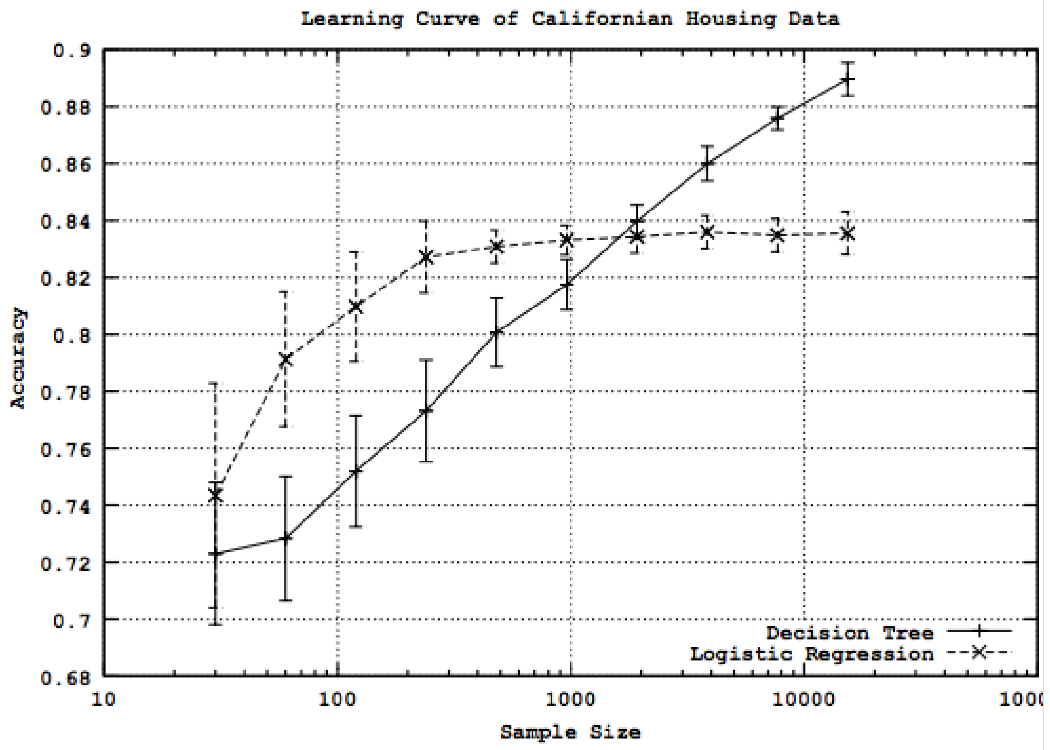
- How does the accuracy of a learning method change as a function of the training-set size?
- This can be assessed by plotting **learning curves**



Learning curves

- given training/test set partition

- for each sample size *s* on learning curve
  - (optionally) repeat n times
    - randomly select s instances from training set
    - learn model
    - evaluate model on test set to determine accuracy *a*
    - plot *(s, a)* or (*s, avg. accuracy and std bars)*

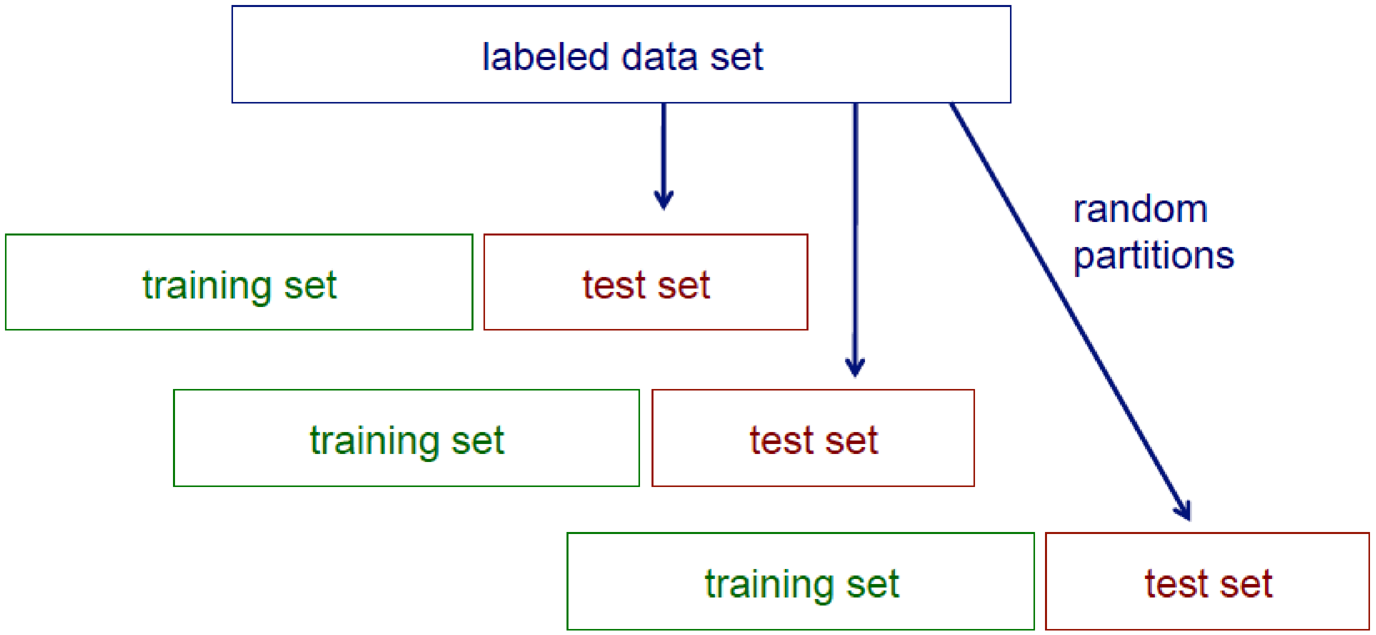


Training/Test set partitions

- We may not have enough data to make sufficiently large training and test sets
  - a larger test set gives us more **reliable** estimate of accuracy
  - but… a larger training set will be more **representative** of how much data we actually have for learning process
- A single training set doesn't tell us how sensitive accuracy is to a particular training sample (i.e. we migh not know how **robust** the ML system is)

Random resampling

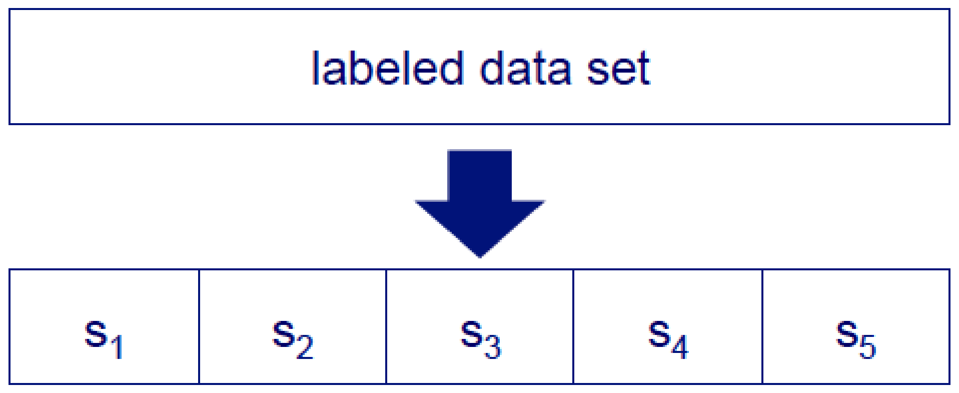
- The issue can be addressed by reapeatedly partitioning the available data into training and test sets



- **Stratified sampling**: When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set (randomly select instances from each class proportionally)

Crossvalidation

Partition the data (features + labels) into *n* subsets



Iteratively leave one subsample out as a test set, while you train the ML model on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Crossvalidation: example

- Suppose we have 100 instances, and we want to estimate with crossvalidation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

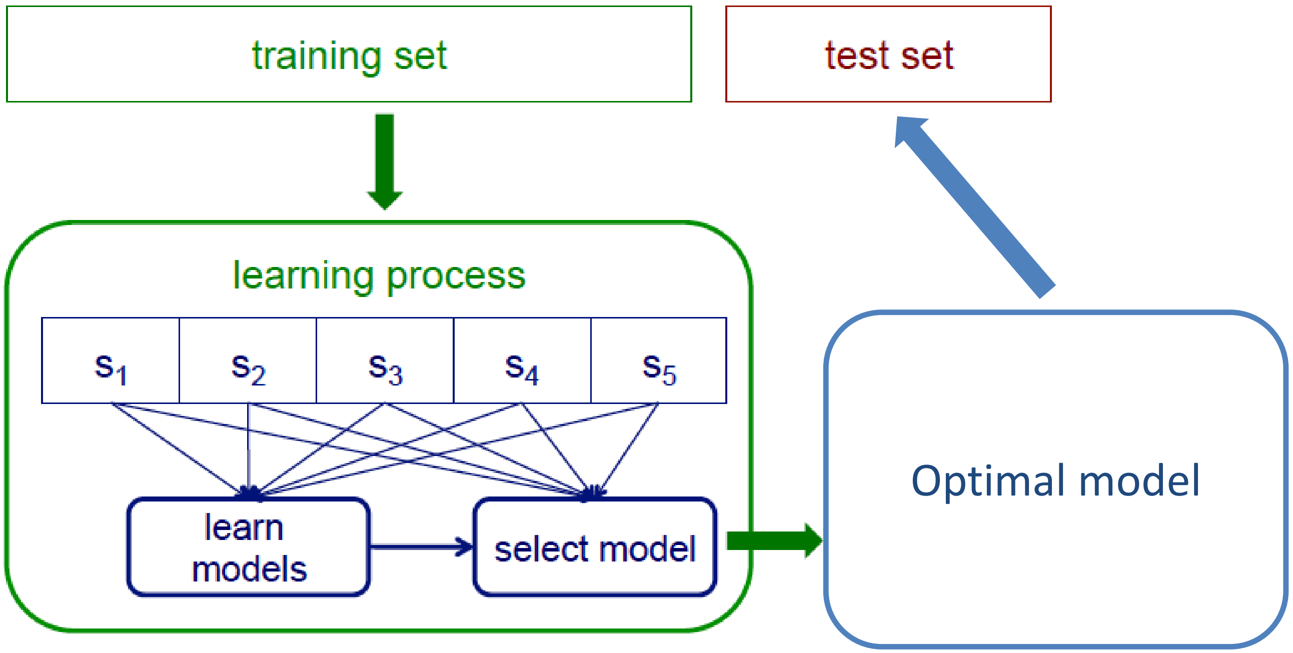
accuracy = 73/100 = 73%

Crossvalidation: summary of concepts

- 10-fold CV is the most common, but smaller values of *n* are often used when learning takes a lot of time
- in **leave-one-out CV**, *n* = # instances
- in **stratified CV**, data are partitioned sampling data homogeneously over classes
- CV makes efficient use of the available data for testing
- whenever we use multiple training sets, as in CV and random resampling, we are evaluating **a learning method**, not just an individual learned model!

Internal crossvalidation

We can use crossvalidation **within a training set** to select a model (e.g. to tune the parameters of a classifier)



Internal crossvalidation

- Suppose we have a ML model depending on a parameter’s value (*k*)

- Given a training set
  1. Partition training set into n folds: $s_1, s_2, \ldots, s_n$
  2. For each value of *k* considered:

     ```
     For i=1 to n
        learn ML model using all folds except for $s_i$
        evaluate the accuracy on si
     ```
  3. Select optimal *k* that obtained the best accuracy for $s_1, s_2, \ldots, s_n$
  4. Learn ML model with optimal k, this time using the entire traning set

- The steps are run independently for each training set (i.e. if we are using 10-fold CV to measure the overall accuracy of the model, then the procedure will be executed 10 times)

Naïve Rule

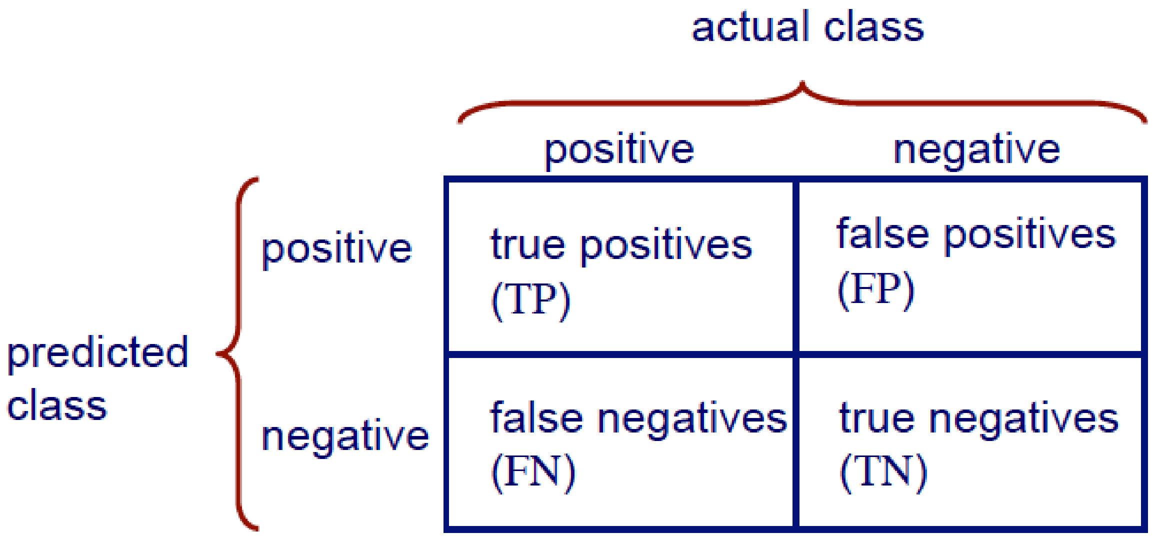
How can we understand how good is and what types of mistakes a ML model makes?

Naïve rule classifier: hypothetical classifier that classifies all the objects as belonging to the most prevalent class

- Often used as benchmark: we hope to do better than that!
  Exception: if our goal is to identify rare outcomes, we may do well by doing worse than the naïve rule

- "High separation of records" means that using predictor variables attains low error
- "Low separation of records" means that using predictor variables does not improve much on naïve rule

Confusion matrix for binary classification



		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

The definition of "positive" or "negative" class is conventional!
Usually, we define as "positive" the class we are most interested in

Quantifying performance

- **True Positive (TP)** a.k.a. *hit*: positive cases correctly identified
- **True Negative (NG)** a.k.a. *correct rejection*: negative cases correctly identified
- **False Positive (FP)** a.k.a. *false alarm,* a.k.a. *Type I error*: negative cases identified as positive
- **False Negative (FN)** a.k.a. *miss,* a.k.a. *Type II error*: positive cases identified as negative

- **Overall Error Rate =** 1 – Accuracy
- If multiple classes, error rate is:

  (sum of misclassified records)/(total records)

Error rate – binary classification

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	201	85
0	25	2689

Overall error rate = (25+85)/3000 = 3.67%

Accuracy = 1 – err = (201+2689) = 96.33%

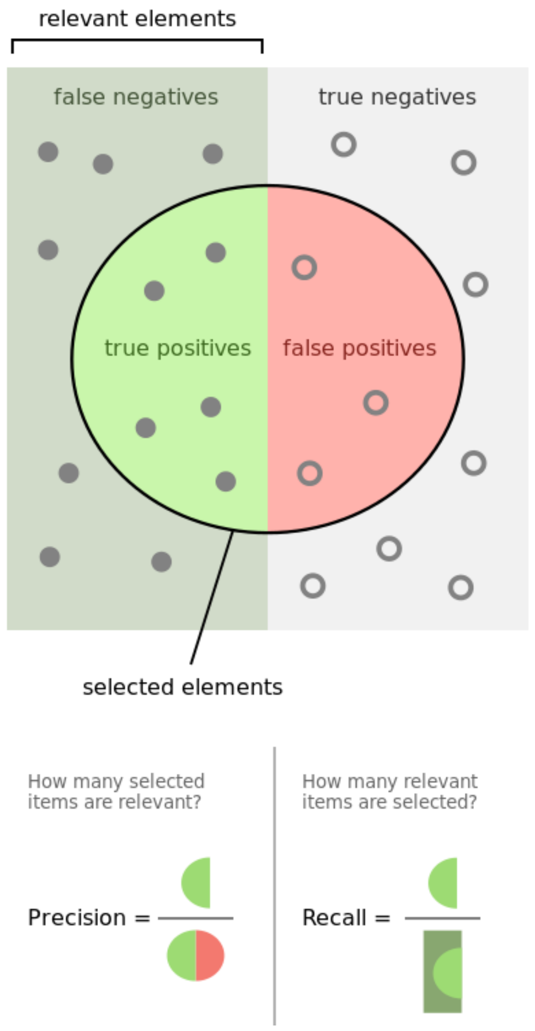
Quantifying performance

- Accuracy alone may not be useful measure in cases where:
  - There is a large class skew
    - Is 98% accuracy good, if 97% of the instance are negative?
  - There are differential misclassification costs – say, getting a negative wrong costs much more than getting a positive wrong
    - In the clinical domain, a false positive leads to unnecessary extra-testing, while a false negative may lead to not treating a disease!

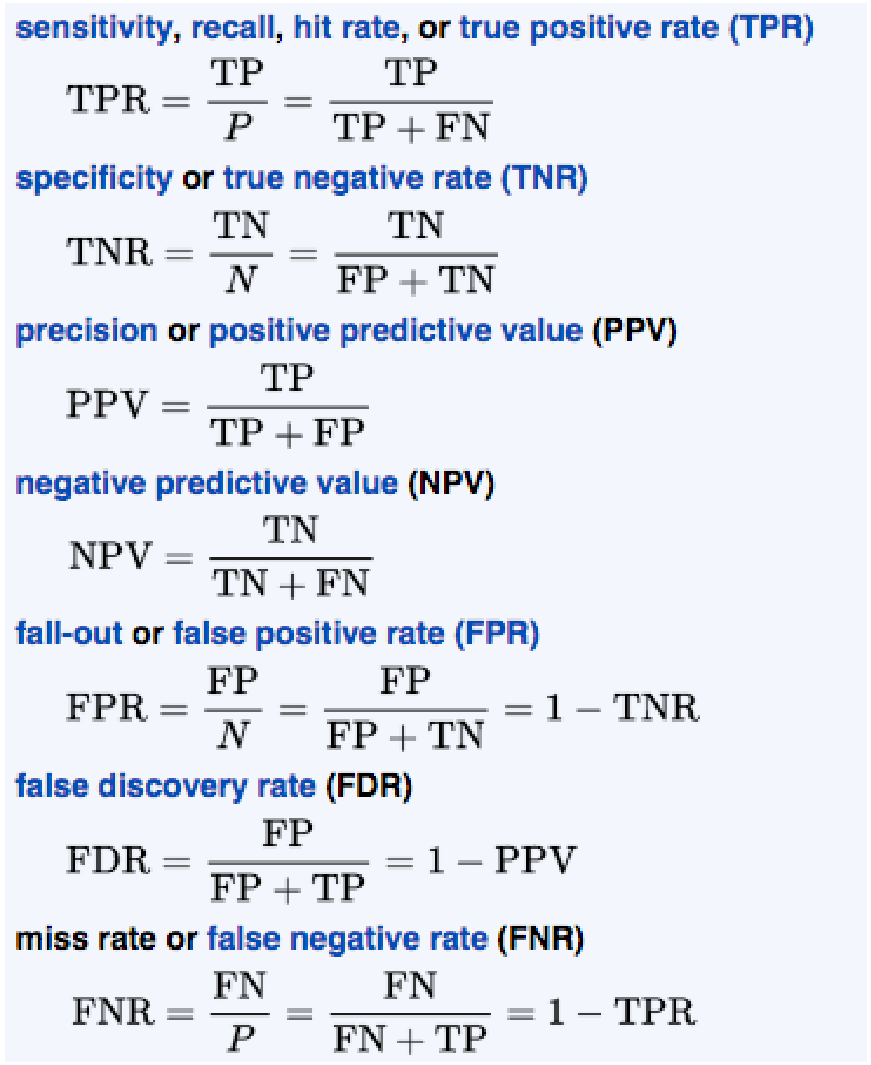
Precision and Recall

- Assuming that the positive class is the relevant one:
  - **precision** (also called **positive predictive value** PPV) is the fraction of selected instances that are actually relevant
  - **recall** (also known as **sensitivity**) is the fraction of relevant instances that are selected by the classifier

- Precision is a measure of "how useful the results are", and recall is "how complete the results are"



Quantifying performance



sensitivity, recall, hit rate, or true positive rate (TPR)

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

specificity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}}$$

fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} = 1 - \mathrm{TNR}$$

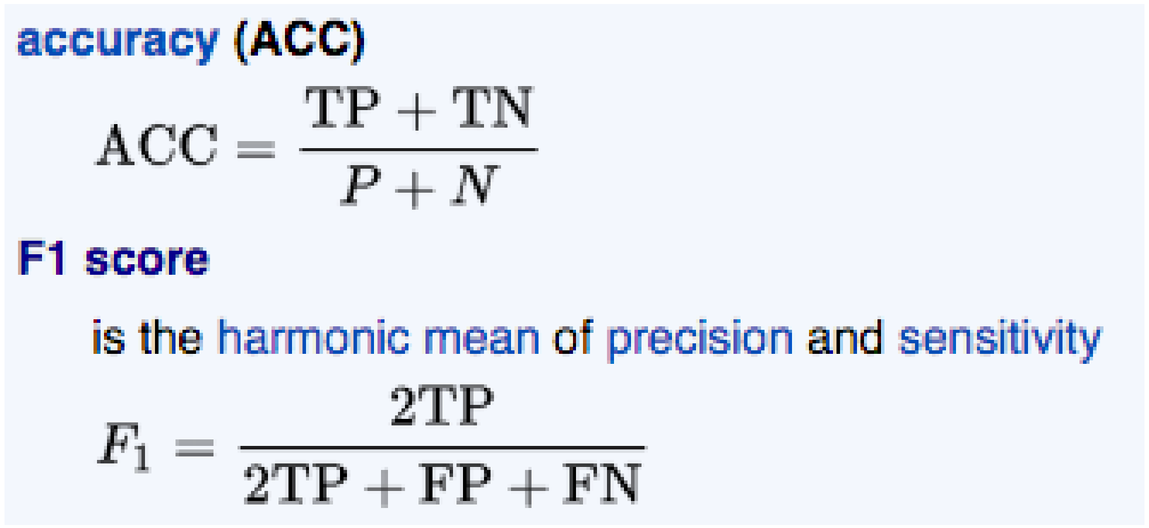
false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TP}} = 1 - \mathrm{PPV}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{P} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}} = 1 - \mathrm{TPR}$$

Quantifying performance



accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N}$$

F1 score

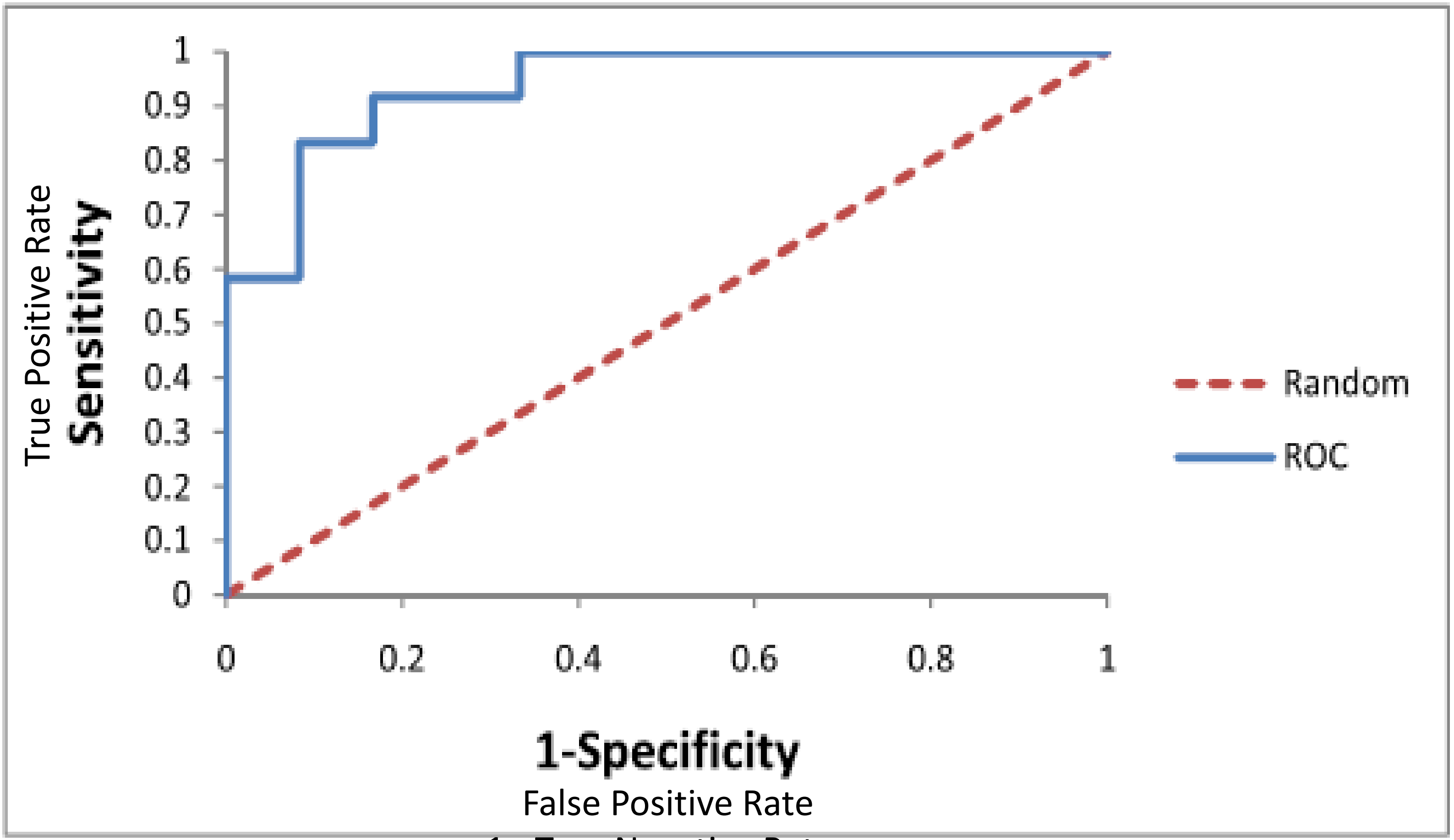
is the harmonic mean of precision and sensitivity

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Classification parameters

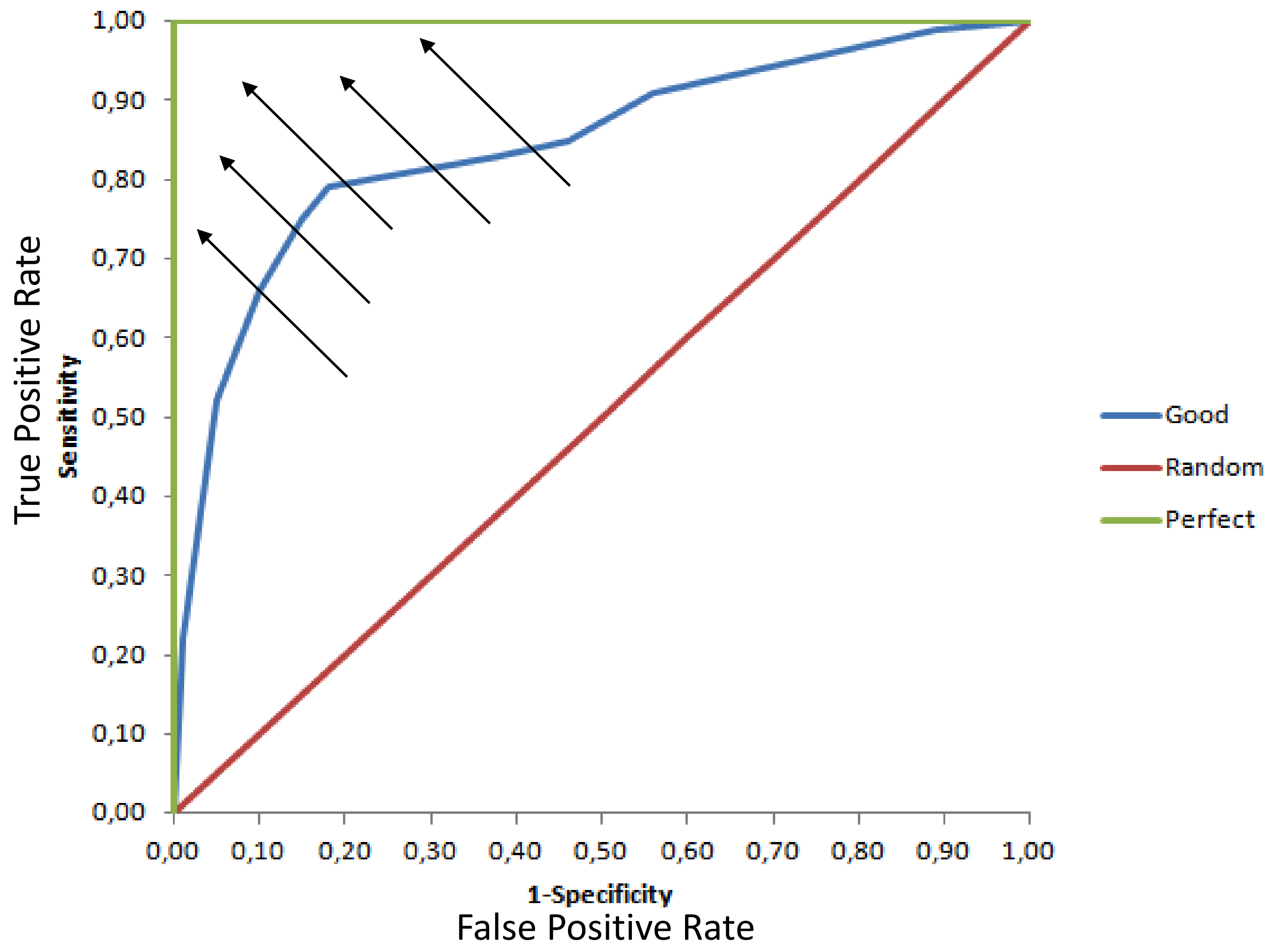
- Binary classification is often performed based on a cutoff value or a parameter's threshold (sometimes, based on more than one parameter)
- Simple example: cutoff value is 0.50
  - If $X >= 0.50$, classify as "1"
  - If $X < 0.50$, classify as "0"
- Can use different cutoff values
- The best cutoff needs to be empirically determined

Receiver Operating Curve (ROC)

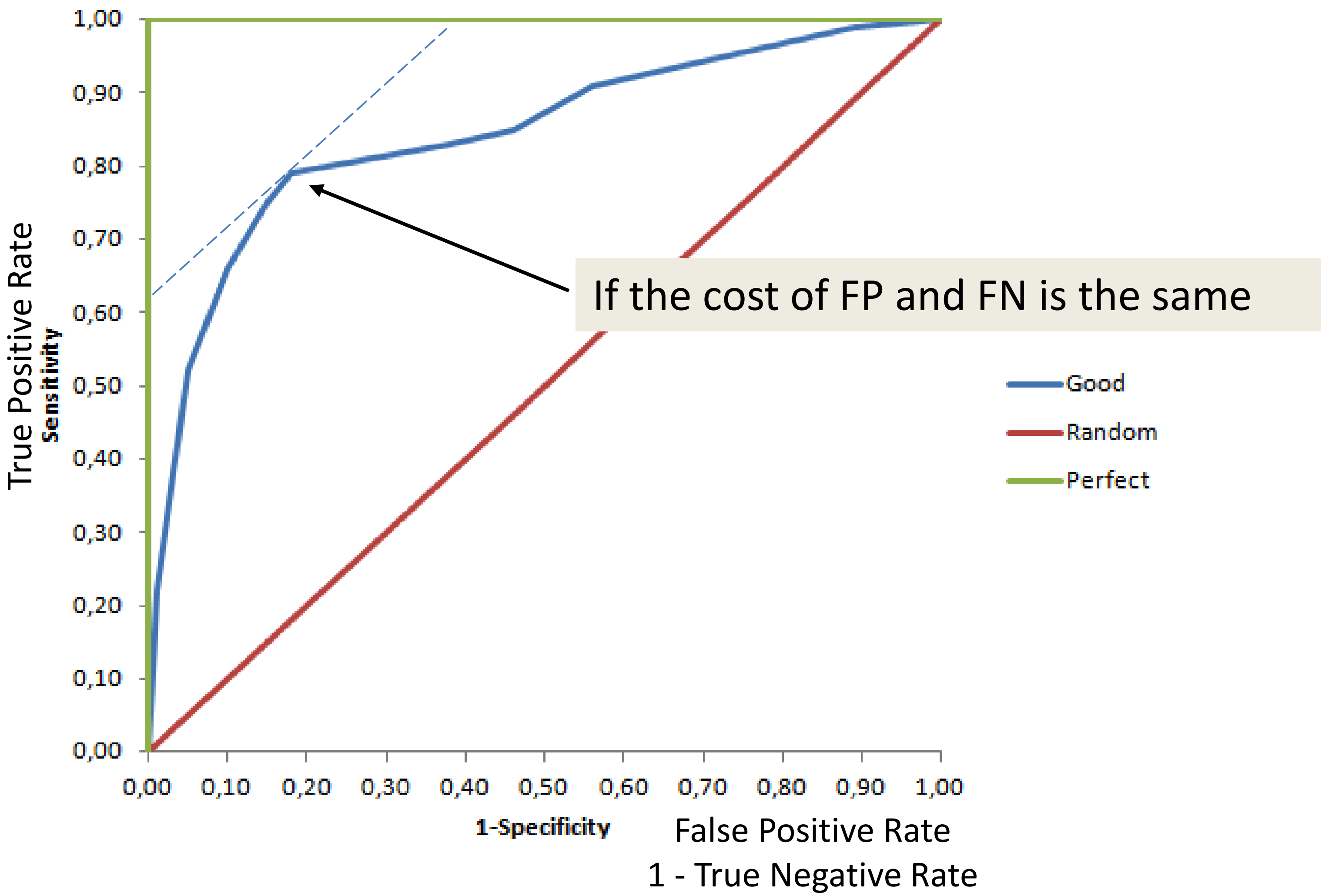


1 - True Negative Ratec

ROC to compare different classifiers



Best operating point of a classifier


True Positive Rate
Sensitivity
1-Specificity
False Positive Rate
1 - True Negative Rate
If the cost of FP and FN is the same
Good
Random
Perfect