Introduction to supervized ML

RCP209 courses 1/15 and 2/15

Arnaud Breloy, 2025

le cnam

Machine learning

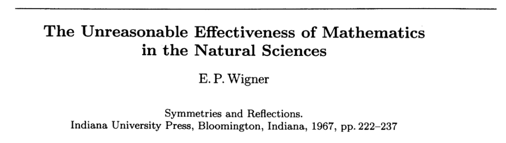
AI: field of computer science that studies or develops “intelligent” software

ML: develop **algorithms** to **solve problems** by **automatically processing data**
or “statistical learning”

A **broad field** that emerged from:

- **Informatics** computational science, data science
- **Applied mathematics** statistics, information theory, optimization
- **Applications** bio-informatics, signal processing, computer vision

A new relationship to data (1/3)



The Unreasonable Effectiveness of Mathematics in the Natural Sciences

E. P. Wigner

Symmetries and Reflections.
Indiana University Press, Bloomington, Indiana, 1967, pp. 222–237

Traditionally: **data** was made for **experts**

- **Scientific question** → **Experiments** → answer to an **hypothesis**

  The IPCC asking "Is global warming due to human activities?"

- **Mathematical models** → **Measures** → **Inversion**

  Meteorological data → yield forecasting

- Automated **classification** through **expert rules**

  Algorithmic transcription of "If # petals ≥ 5, then..."

Wigner (Nobel in φ), "*The unreasonable effectiveness of mathematics in the natural sciences*," Symmetries and Reflections, 1967.

A new relationship to data (2/3)

The Unreasonable Effectiveness of Data

Alon Halevy, Peter Norvig, and Fernando Pereira, *Google*

Current explosion of

- Available **data** sensors, measurements, experiments
- Data **dimension** pixels, monitored genes, sampling rate
- **Computing power**

Paradigm shift

- **Learn models** directly from the **data**
- Gather **data first**, ask **questions later**

Halevy, Norvig, & Pereira (Google) "*The unreasonable effectiveness of data*," IEEE intelligent systems, 2009

A new relationship to data (3/3)

Expert system

Machine learning

Tools: **statistics**, **informatics**, **linear algebra**, **optimization**

Notation: supervised dataset

Supervised data appends a **labels** $\{\mathbf{y}_i\}_{i=1}^n \in (\mathcal{Y})^n$ to the **sample set** $\{\mathbf{x}_i\}_{i=1}^n \in (\mathbb{R}^d)^n$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \leftarrow i^{\text{th}} \text{ sample} \quad \text{is paired with} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_i^\top \\ \vdots \\ \mathbf{y}_n^\top \end{bmatrix} \leftarrow i^{\text{th}} \text{ label}$$

- **Classification**: $\mathcal{Y} = \{1, \ldots, m\}$ and $\mathbf{Y}$ stores the **class labels**
- **Regression**: $\mathcal{Y} = \mathbb{R}^{d'}$ and $\mathbf{Y}$ stores the **latent variables** of a relationship $\mathbf{x} = g(\mathbf{y})$

Some examples

- **Classification**
  - $\mathbf{x}$ is an image (vector stores all values of the pixels)
  - $\mathbf{y}$ encodes the classes (cat, dog, car, ...)
- **Detection** (classification with 2-classes)
  - $\mathbf{x}$ contains values of physical constants of a patient
  - $\mathbf{y}$ is 0 (healthy) or 1 (sick)
- **Regression**
  - $\mathbf{x}$ gathers microphones measurements in a room
  - $\mathbf{y}$ is the position of the acoustic source
- **Prediction** in time series
  - $\mathbf{x}$ stores values of temperatures over d days
  - $\mathbf{y}$ stores the values of temperatures in the next d' days

Classification

Learn a **decision function** $f_\theta : \mathbb{R}^d \rightarrow \mathcal{Y}^1$ from **data** $\mathbf{X} \in \mathbb{R}^{n \times d}$ with **class labels** $\mathbf{Y} \in \mathcal{Y}^1$

Attribute classes to **new samples** $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$

"**Learning the model**" is finding θ so that f_θ produces the right boundaries

Regression

We assume an underlying relationship between **measurement** $\mathbf{x}$ and **hidden variables** $\mathbf{y}$

$$\mathbf{x} = g(\mathbf{y}) + \text{“noise”}$$

Learn a **regression function** $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ from **data** $\mathbf{X} \in \mathbb{R}^{n \times d}$ to **labels** $\mathbf{Y} \in \mathbb{R}^{n \times d'}$

Attribute estimates to **new samples** $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$

without noise we should find $f_\theta \simeq g^{-1}$

Example: fitting a curve

"Models" in ML

A **model** is a **function**

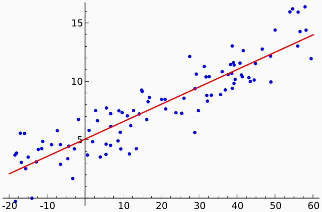
$$f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$$
$$\mathbf{x} \mapsto \hat{\mathbf{y}} = f_\theta(\mathbf{x})$$

with tunable set of **parameters** θ

Example: 1D linear function

$$\hat{y} = ax + b$$

with parameters $\theta = \{a, b\}$

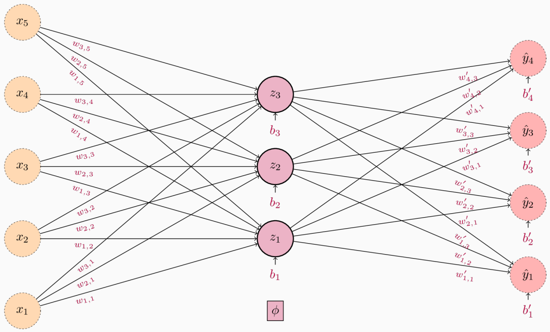


How to chose θ?

Neural networks are models

Multi layer preceptron (MLP):

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\text{out}}(\mathbf{W}^{\text{out}}\phi_{\text{in}}(\mathbf{W}^{\text{in}}\mathbf{x} + \mathbf{b}^{\text{in}}) + \mathbf{b}^{\text{out}}) \quad \text{with} \quad \boldsymbol{\theta} = \{\mathbf{W}^{\text{in}}, \mathbf{b}^{\text{in}}, \mathbf{W}^{\text{out}}, \mathbf{b}^{\text{out}}\}$$



ϕ: **activation function**

e.g., ReLU

Generalizes to **more layers**

$$\mathbf{h}^{(k+1)} = \phi^{(k)}(\mathbf{W}^{(k)}\mathbf{h}^{(k)} + \mathbf{b}^{(k)})$$

Other neural networks are models suited to some specific data

Signals and **Images**

- Convolutional neural networks (**CNN**)
- Vision transformers (**ViT**)

Data on graphs

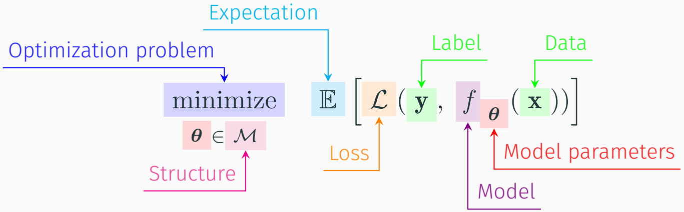
- Graph neural networks (**GNNs**)

Time series

- Residual neural networks (**RNN**)
- **LSTM**, **GRU**
- **Transformers**

Still, how to chose θ?

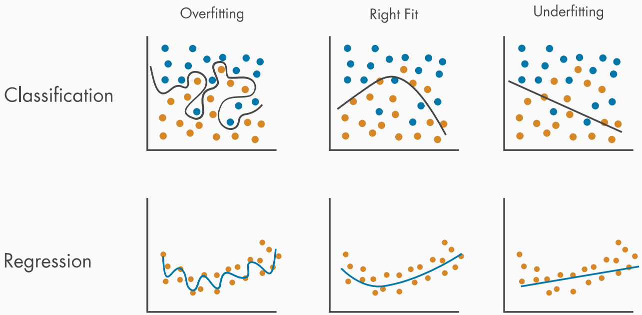
Supervized ML optimization philosophy



- **Design** prediction model $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$ and loss $\mathcal{L}$
- **Learn** the model parameters θ approx. $\mathbb{E}$ with data at hand + solve the optimization problem
- **Apply** model to new data

Capacity, generalization, over-fitting, ...

Expected performance is evaluated on a training set, does it work on **new unseen data** ?



Possibility of **over-fitting**: trade off between the model **capacity** and **generalization**

ML literature provides **methodologies** to **properly control this**

In this course

We assume to have several models/techniques at hand

Can theory explain what will happen?

How to **validate models** individually?

How to **compare models** properly?

Discuss proper **validation methods** and **best practices**

Notation: supervised dataset

Supervised data appends a **labels** $\{\mathbf{y}_i\}_{i=1}^n \in (\mathcal{Y})^n$ to the **sample set** $\{\mathbf{x}_i\}_{i=1}^n \in (\mathbb{R}^p)^n$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \leftarrow i^{\text{th}} \text{ sample} \quad \text{is paired with} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_i^\top \\ \vdots \\ \mathbf{y}_n^\top \end{bmatrix} \leftarrow i^{\text{th}} \text{ label}$$

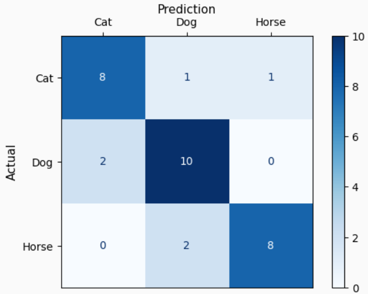
- **Classification**: $\mathcal{Y} = \{1, \ldots, m\}$ and $\mathbf{Y}$ stores the **class labels**
- **One-hot encoding**: $\mathbf{y}_i = k$ "sample $\mathbf{x}_i$ is in class k" is stored as $[0, \cdots, \underbrace{1}_{k^{th} \text{ elt.}}, 0, \cdots, 0]$

Classification results

f_θ is a trained model, and $\hat{\mathbf{y}}_i = f_\theta(\mathbf{x}_i)$ gives a prediction for each sample

Confusion matrix $\mathbf{C}$

- $C_{q\ell}$ stores the number of samples that are from class q and are predicted as class ℓ
- sometimes normalized per lines to read fractions per classes



Metrics for classification

Most of them can be constructed constructed from $\mathbf{C}$:

Accuracy: total ratio of correctly classified samples, $\sum_k C_{kk}/n$

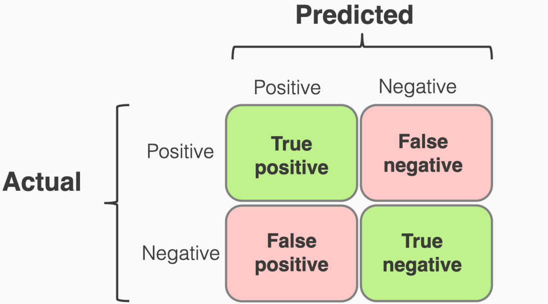
Recall (per class): “accuracy per class”, $C_{kk}/\sum_q C_{qk}$

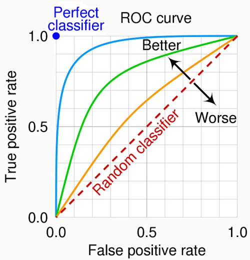
Precision (per class): correct prediction over number of prediction of class k, $C_{kk}/\sum_q C_{kq}$

Balanced accuracy: the (weighted) average of recall obtained on each class

other exist, e.g., F1-score

Focus on metrics for binary classification (detection)





Probability of detection (**PD**): TP/(TP+FN)

Probability of false alarm (**PFA**): FP/(FP+TN)

ROC: PD vs PFA when varying the threshold on the score function

AUC: area of the ROC (1 being optimal)

Worst practice in action

A classical procedure

- Gather a labeled dataset $\{\mathbf{X}, \mathbf{Y}\}$
- Choose and train a model f_θ on this dataset optimize θ w.r.t. loss $\mathcal{L}$
- Achieve 99.5% accuracy
- Deliver the model to production with confidence
- After all, what could go wrong?

Caveat

Be careful what you wish for: optimizing one criterion can yield unexpected results

Best performance after training does not means it will translate in practice

Empirical risk minimization

Expected risk: we aim for generality

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}\left[\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))\right]$$

Empirical risk: we apply in practice

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^{n} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i))$$

There might be issues:

- if n is too low, poor approximation of the expectation
- available samples might be not representative of future ones
- we will see that choosing f_θ just to minimize the empirical risk is not enough

What happens here ?

f_θ belongs to a family of fucntion $\mathcal{F} = \{f_\theta, \mid \theta \in \Theta\}$

$f^*_{\mathcal{D}_N}$ is the learned function: it minimizes the empirical risk $R_{\mathcal{D}_N}$ over $\mathcal{F}$

f^* is the ideal function of $\mathcal{F}$ that minimizes the expected risk R

$$R(f^*_{\mathcal{D}_N}) = R^* + [R(f^*) - R^*] + [R(f^*_{\mathcal{D}_N}) - R(f^*)]$$

R^* is the Bayes risk (ideal one)

$[R(f^*) - R^*]$ is the approximation error

- ≥ 0 because $\mathcal{F}$ might not contain the ideal function
- $= 0$ if R^* can be reached by a function of $\mathcal{F}$

$[R(f^*_{\mathcal{D}_N}) - R(f^*)]$ is the estimation error

- ≥ 0 because $f^*_{\mathcal{D}_N}$ is most likely not f^*

Capacity of a model

Capacity: refers to the literal "capacity" of f_θ to produce complex decision boundaries

will be formally defined

Generalization: ability to produce similar results on future (unseen) samples

There is a **trade-off** regarding capacity

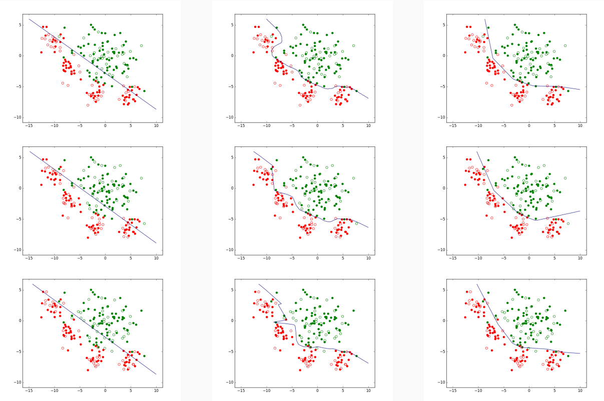
- **too large** : good performance on learning set, bad generalization
  "loss when applied in practice"
- **too low**: good generalization, but subpar performance
  "no loss when applied in practice"

Overfitting: poor generalization happening when the models exactly describes $\{\mathbf{x}\}_{i=1}^{n}$

instead of "any expected $\mathbf{x}$"

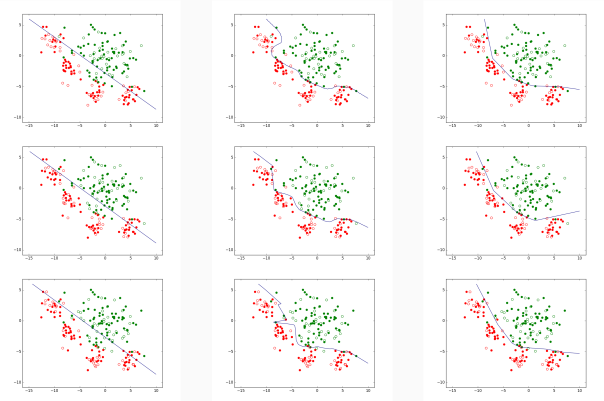
Capacity and overfitting

	LDA	MLP $\alpha = 10^{-5}$	MLP $\alpha = 1$
Sample 1			
Sample 2			
Sample 3			



Capacity and overfitting

	LDA	MLP $\alpha = 10^{-5}$	MLP $\alpha = 1$
Sample 1			
Sample 2			
Sample 3			



	LDA	MLP $\alpha = 10^{-5}$	MLP $\alpha = 1$
Err. train.	14,4%	1,5%	4%
Err. test	9,5%	7%	5,5%
Std. test	0,038	0,026	0,017

Defining capacity

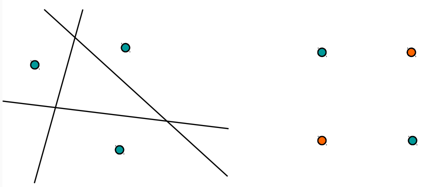
Take n points $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d \rightarrow$ there are 2^N possible partitions in 2 sets

Définition: the family $\mathcal{F}$ of functions $f : \mathbb{R}^p \rightarrow \{-1, 1\}$ shatters $\{\mathbf{x}_i\}_{1 \leq i \leq N}$ if all 2^N partitions can be constructed with functions from $\mathcal{F}$

Définition (Vapnik-Chervonenkis): $\mathcal{F}$ is of VC-dimension h if it shatters at least a set of h points but no sets of $h + 1$ points

Exemple: the VC-dimension of hyperplmanes of $\mathbb{R}^p$ is $h = p + 1$

In $\mathbb{R}^2$ lines can shatter triplets but not quadruplets of points



Linking capacity to generalization

VC-dimension measures the capacity in some way

It allows to bound the difference between empirical and expected risk

Theorem: let $R_{\mathcal{D}_N}(f)$ be the empirical risk defined by $L_{01}(\mathbf{x}, y, f) = \mathbf{1}_{f(\mathbf{x}) \neq y}$; if $\mathcal{F}$ has VC-dimension $h < \infty$, then $\forall f \in \mathcal{F}$, with probability $> 1 - \delta$ $(0 < \delta < 1)$, we have

$$R(f) \leq R_{\mathcal{D}_N}(f) + \underbrace{\sqrt{\frac{h\left(\log \frac{2n}{h} + 1\right) - \log \frac{\delta}{4}}{N}}}_{B(n,\mathcal{F})} \quad \text{for} \quad n > h$$

$B(N, \mathcal{F})$ decreases when $n \uparrow$, $h \downarrow$, and $\delta \uparrow$
$B(N, \mathcal{F})$ does not depends on the number of variables
$B(N, \mathcal{F})$ does not depend on the underlying law

The actual value is not useful in practice, but provides an interesting intuition!

Intuition gained from the bound

$$R(f^*_{\mathcal{D}_N}) = R^* + \underbrace{[R(f^*) - R^*]}_{\text{approx. error}} + \underbrace{[R(f^*_{\mathcal{D}_N}) - R(f^*)]}_{\text{estim. error}} \quad \text{and} \quad R(f) \leq R_{\mathcal{D}_N}(f) + \underbrace{\sqrt{\frac{h\left(\log\frac{2n}{h}+1\right)-\log\frac{\delta}{4}}{N}}}_{B(n,\mathcal{F})}$$

$\mathcal{F}$ has low capacity e.g., linear model

$\Rightarrow$ $B(N, \mathcal{F})$ low, but $R_{\mathcal{D}_N}(f)$ large $\Rightarrow$ no interesting guarantee for $R(f)$

$\mathcal{F}$ has large capacity e.g., MLP $\alpha = 10^-5$

$\Rightarrow$ $R_{\mathcal{D}_N}(f)$ low but $B(N, \mathcal{F})$ large $\Rightarrow$ no interesting guarantee for $R(f)$

$\mathcal{F}$ has "adequate" capacity e.g., MLP $\alpha = 1$

$\Rightarrow$ $R_{\mathcal{D}_N}(f)$ low, $B(N, \mathcal{F})$ low $\Rightarrow$ interesting guarantee for $R(f)$!

Parameters vs hyper-parameters

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \sum_{i=1}^{n} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)) + \lambda \rho(\theta)$$

Parameters are denoted theta θ: what is optimized at training

Hyper-parameters are choices around this

- Choice of the **model family** and **architecture** e.g. number of parameters
- Regularization parameter λ
- Choices of losses $\mathcal{L}$ and regularization penalty ρ

These do not move at training

However these have a direct impact on the **model capacity**

Controlling capacity

In practice neural networks can be too expressive, we can **control the capacity** by:

- Regularization: avoids fluctuation of the parameters

$$\underset{\theta \in \Theta}{\text{minimize}} \quad \sum_{i=1}^{n} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)) + \lambda \rho(\theta)$$

e.g., $\rho(\theta) = ||\theta||^2$

- Sequentially increase capacity e.g. number of layers or neurons in MLP

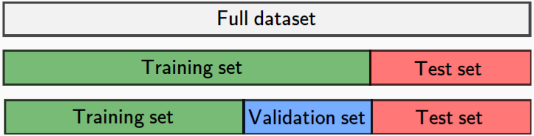
We need to tune **hyper-parameters** (architecture and loss)

Different models correspond to different optimization problems

$\longrightarrow$ Comparing loss values is not relevant

$\longrightarrow$ How to *properly* compare models after the learning step ?

Splitting



Comparison procedure:

Split the training data in a training and validation sets (non overlapping)
Train different models (methods and/or parameters) on the training set
Evaluate performance on the validation set

ideally, averaged on several splits (K-fold cross-validation)

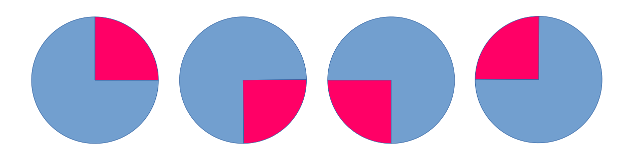
Hyperparameter selection:

Selecting hyper-parameters to maximize score on test set is cheating!
Create a sub-split for validation from the training set
Test hyperparameters on a grid cf. GridSearchCV in scikit-learn

Different splittings

Creating multiple data splits

K-folds: n samples split into K, training on $K-1$, evaluation on last one



Shuffle-and-split: validation set selected at random, K times

Special cases requiring attention

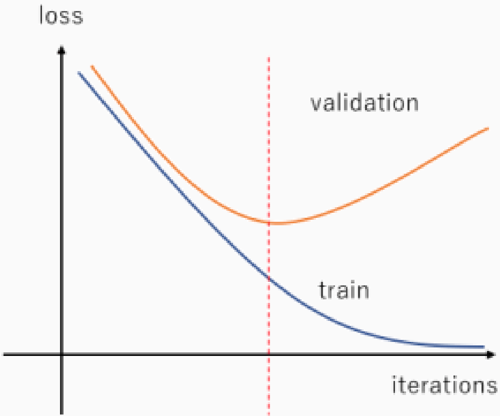
Unbalanced classes: use stratification to preserve proportions

Non-independence of observations

- groups of observations: test set should not contain samples correlated with train
- time series: split should be done by sequences

StratifiedKFold, StratifiedShuffleSplit, TimeSeriesSplit is scikitlearn

Evidencing overfitting in training



Conclusion

Supervised ML requires **labeled datasets**

We need to get the best **metrics**, while **ensuring generalization**

Estimating the generalization **cannot be done from training only**

There is a trade-off between **capacity** and **generalization**

Given a set of possible models (different hyper-parameters): test a grid methodically!

Last warnings

Best performance (number, value, criterion, ...) does not mean that the model is better

Caraful when defining what we want: "AI" is dumb and performs malicious compliance

Example1: A model that returns "true" has the best detection probability (100%)

Example2: predictions will shift in accordance with results the majority class

as it counts more into the standard average

Can completely render some classes invisible (inducing biases and unfairness)