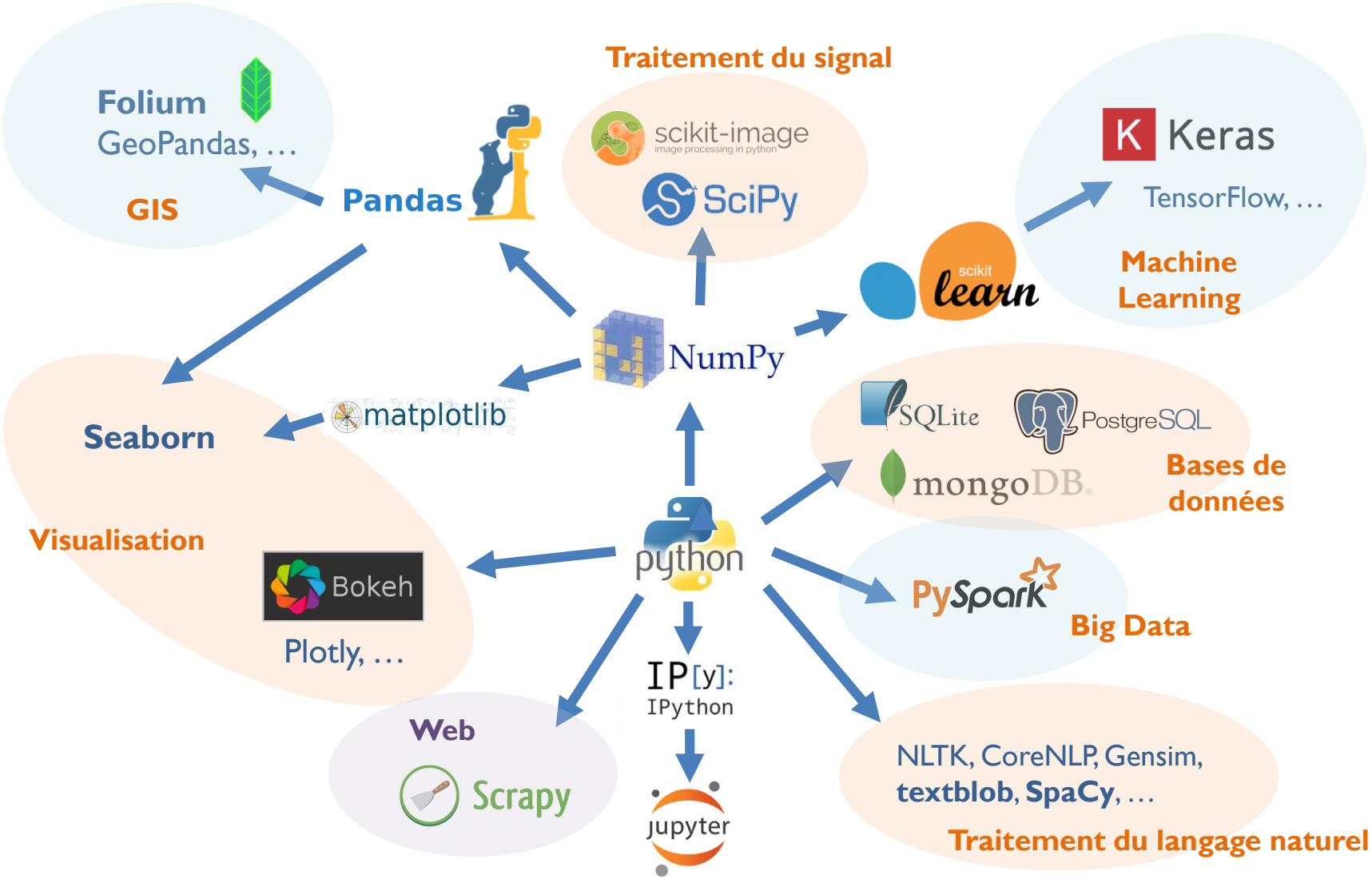
Chapitre 4

OUTILS PYTHON POUR LA DATA SCIENCE

L'écosystème Python pour les data scientists



Scikit learn

Données

- X : tableau 2D numpy de taille (n. exemples) x (n. variables)
- Y : tableau 1D numpy de taille (n.exemples)

Induction du modèle

- Création d'un estimateur/modèle (hyper paramètres passés au constructeur)
- Méthode **fit(Xa,Ya)** apprend les paramètres du modèle à partir de données d'apprentissage

Prédiction

- Méthode **predict(X)** prédit un vecteur de sortie Yp

Evaluation

- Méthode **score(Xtest,Ytest)** utilise la fonction associée au classifieur
- Nombreuses fonctions de score/coût dans **sklearn.metrics**
- Nombreuses techniques d'évaluation (CV k-folds, etc) dans **sklearn.cross_validation**

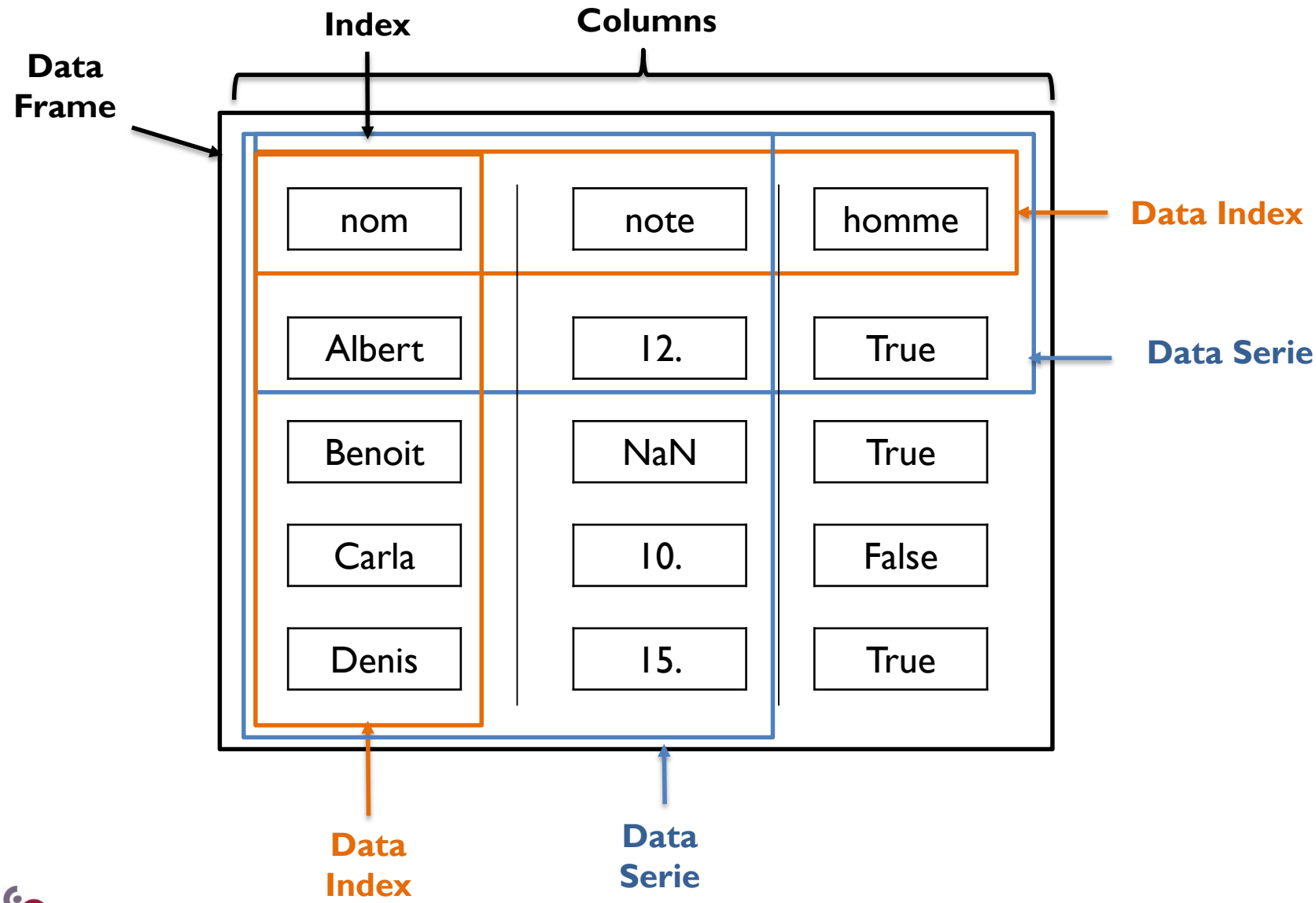
Modèle

```
import numpy as np

from sklearn import datasets
data = datasets.load_iris()
X,Y = data.data,data.target

# Mélange le jeu de données
from sklearn.utils import shuffle
X,Y = shuffle(X,Y)

# Crée une forêt de 10 arbres
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 10)

# Découpe en jeu d'apprentissage et jeu de test
Xa,Xt = X[:0.8*len(X)], X[0.8*len(X):]
Ya,Yt = Y[:0.8*len(Y)], Y[0.8*len(Y):]

# Apprend le modèle
forest.fit(Xa,Ya)

# Prédit la sortie
Yp = forest.predict(Xt)
```

Validation croisée

```
# Evalue le modèle (score par défaut)
print("score     = ", forest.score(Xt,Yt))

# Utilise une métrique
from sklearn.metrics import zero_one_loss
print("zero loss = ", zero_one_loss(Yt,Yp))

# Validation croisée avec fonction prédéfinie
forest = RandomForestClassifier(n_estimators = 10)
from sklearn import cross_validation as cv
acc = cv.cross_val_score(forest,X,Y, cv = 10, scoring = "accuracy")
print("accuracy = ", acc.mean())

# Validation croisée avec fonction de perte spécifique
from sklearn.metrics import make_scorer

cout = np.array([3,1,2])
def myloss(yreel, ypred):
  return cout[yreel[yreel != ypred]].sum()

myloss = cv.cross_val_score(forest,X,Y,cv = 10, scoring = make_scorer(myloss))
print("myloss = ", myloss.mean())
```

Pandas


Index
Columns
Data Frame
nom
note
homme
Albert
12.
True
Benoit
NaN
True
Carla
10.
False
Denis
15.
True
Data Index
Data Serie
Data Index
Data Serie

Pandas

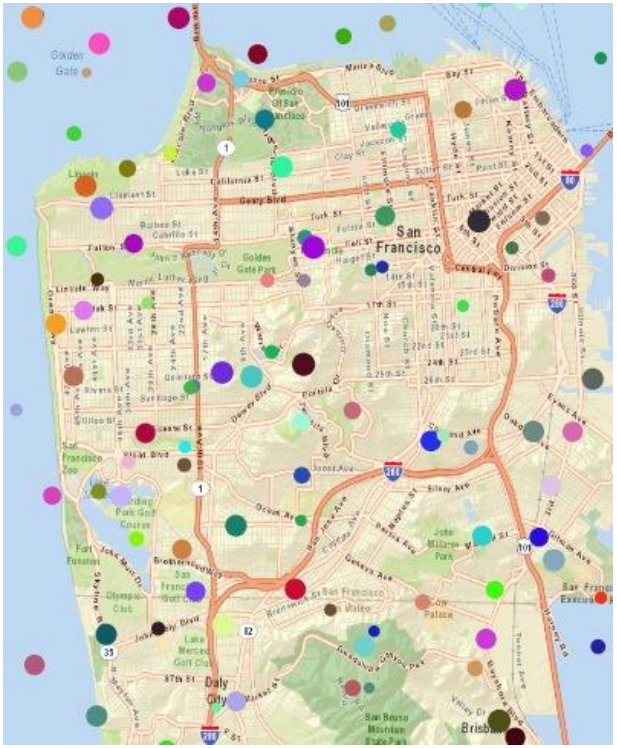
```
import pandas as pd
import numpy as np

# Création d'un "DataFrame"
data = {
"site"   : [ "Musée d'Art Moderne", "Musée du Louvres", "Musée d'Orsay",
"Centre Pompidou", "Centre Pompidou", "Musée des Beaux-Arts" ],
"type"   : [ "Contemporain", "Classique", "Impressionisme", "Contemporain",
"Contemporain", "Classique" ],
"ville"  : [ "Strasbourg", "Paris", "Paris", "Paris", "Metz", "Nancy" ],
"visite" : [ 2011, 2012, 2012, 2012, 2015, 2011 ]
}
frame = pd.DataFrame(data, index = ["Str1", "Par1", "Par2", "Par3", "Mtz1",
"Ncy1"])

# Sélections de colonnes et de lignes
print(frame[[1,2]].ix[1:3])
print(frame[['site','ville']].ix['Str1':'Mtz1'])
sample = frame.take(np.random.randint(0, len(frame), size = 2))

# Ajout, suppression de lignes, de colonnes
frame['nbr'] = [2,2,3,3,4,1,]
del frame['nbr']
frame.index.drop('Mtz1')
```

Pandas

```
# Statistiques
print(frame.describe())
print(frame['visite'].mean())

# Histogrammes et discrétisation
pd.cut(frame.visite,2)
frame['periode']=pd.cut(frame.visite,2)

# Aggrégation
groups = frame.groupby(frame['type'])
print(groups.mean())
print(groups.visite.agg(lambda g : (g.min(), g.max())))

# Nettoyage et préparation
frame.drop_duplicates()
frame['code'] = frame['type'].map({ "Contemporain": 1, "Classique" : 2 })
frame = frame.dropna()

# Jointure
frame2 = pd.DataFrame({
"ville"  : [ "Strasbourg", "Paris","Metz", "Nancy" ],
"dept"   : [ 67, 75, 57, 54 ]
})
frame3 = pd.merge(frame, frame2, on='ville')
```

Matplotlib et les projections cartographiques

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.basemap import Basemap

# Crée une projection centrée sur San Francisco
frame = ((-122.52, 37.68), (-122.38,37.82))
map = Basemap(
  llcrnrlon=frame[0][0], llcrnrlat=frame[0][1],
  urcrnrlon=frame[1][0], urcrnrlat=frame[1][1],
  epsg=3493)

# Charge le plan associé à la projection
map.arcgisimage(
  service='ESRI_StreetMap_World_2D',
  xpixels = 1000, verbose= True)

latitudes = np.array(...); longitudes = ...
couleurs = ...; rayons = ..
# Convertit (long,lat) en position image
(X,Y)=map(longitudes, latitudes)
# Appel classique aux fonctions matplotlib
plt.scatter(X, Y, rayons, marker = 'o', color = couleurs,
alpha = 0.4)
# Lance l'affichage
plt.show()
```




```
import sqlite3
villes = [
  { "nom" : "Metz", "pays" : 1, "popu" : 120.7 },
  ...
]
pays = [
  { "code" : 1, "nom" : "France" },
  { "code" : 2, "nom" : "Luxembourg" }
]
# Connexion à la base (simple fichier)
connection = sqlite3.connect('mabase.db')
# Création de tables
connection.execute('CREATE TABLE IF NOT EXISTS villes (nom text, pays integer,
population real)')
connection.execute('CREATE TABLE IF NOT EXISTS pays (nom text, code integer)')
# Suppression des entrées existantes
connection.execute('DELETE FROM villes')
connection.execute('DELETE FROM pays')
# Remplissage des tables à partir d'un tableau de dictionnaires
connection.executemany('INSERT INTO villes VALUES (:nom, :pays, :popu)', villes)
connection.executemany('INSERT INTO pays VALUES (:nom, :code)', pays)
# Confirme la mise à jour
connection.commit()
```

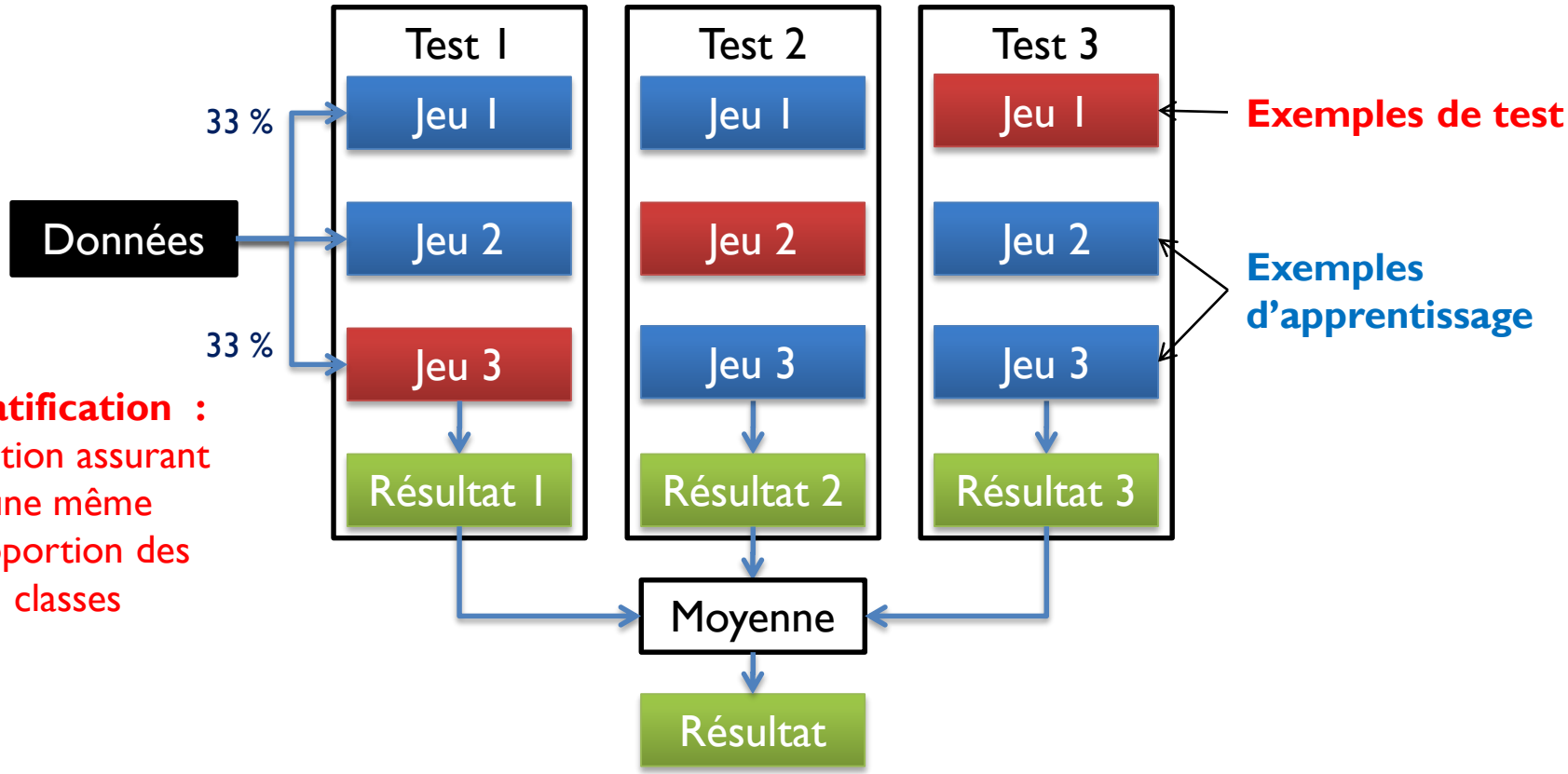
Une base SQL très légère (un seul fichier) mais locale.

Validation croisée

But : évaluer la capacité à généraliser

- Risque de sur-apprentissage → besoin d'un jeu d'exemples et de test indépendant
- Nombre d'échantillons limité

Solution : validation croisée 3-fold, 10-fold, …, m-fold (leave-one-out)



Evaluation d'une méthode de régression

Problème : mesurer les écarts entre prédictions et valeurs cibles réelles

Mesure	Définition
Corrélation entre $\hat{Y}$ et Y	$corr(\hat{Y}, Y) = \frac{\sum_{1\le i\le m}\left(\hat{y}_i - \hat{E}(\hat{y}_i)\right)\left(y_i - \hat{E}(y_i)\right)}{\sqrt{\sum_{1\le i\le m}\left(\hat{y}_i - \hat{E}(\hat{y}_i)\right)^2}\sqrt{\sum_{1\le i\le m}\left(y_i - \hat{E}(y_i)\right)^2}}$
Erreur absolue (L1) moyenne	$\varepsilon_1 = \frac{1}{m}\cdot\sum_{1\le i\le m}\lvert\hat{y}_i - y_i\rvert$
Erreur quadratique (L2) moyenne	$\varepsilon_2 = \sqrt{\frac{1}{m}\cdot\sum_{1\le i\le m}(\hat{y}_i - y_i)^2}$
Erreurs absolue et quadratique relatives	$\frac{\varepsilon_{1/2}}{\varepsilon_{1/2}\ pour\ \hat{y}_i = E(y_i)}$

Evaluation d'une méthode de classification binaire :

Les 4 catégories de prédiction d'un classifieur binaire :

Réalité \ Prédiction	**positif**	**négatif**
positif (P)	vrai **positif** (VP)	faux **négatif** (FN)
négatif (N)	faux **positif** (FP)	vrai **négatif** (VN)

Taux d'erreur : estimation du risque moyen $\hat{E}(\varepsilon)$ où ε est l'erreur valant 0 ou 1

$$t_\varepsilon = \frac{FN + FP}{P + N}$$

Taux de réussite / de prédiction (accuracy) :

$$t_r = \frac{VP + VN}{P + N} = 1 - t_\varepsilon$$

Généralisation au problème multi-classe : Matrice de confusion

Prédiction / Réalité	Classe 1	Classe 2	Classe 3	Classe 4
Classe 1	**n11**	n12	n13	n14
Classe 2	n21	**n22**	n23	n24
Classe 3	n31	n32	**n33**	n22
Classe 4	n41	n42	n43	**n44**

Taux de réussite :

$$t_r = \frac{\sum_i n_{i,i}}{\sum_{i,j} n_{i,j}}$$

Confiance sur les mesures (1)

Problème 1 : quelle confiance accorder à une mesure calculée sur un ensemble fini d'exemples de test ?

Réponse : calculer un **intervalle de confiance**

Exemple du taux d'erreur t_e calculé sur m exemples :

1. Hypothèse : exemples indép. ident. distribués (apprent. et test)

→ Si la probabilité de commettre une erreur est p, le nombre d'erreurs $\mathrm{N_e} = \sum_{i=1}^{m} |y_i \neq \hat{y}_i|$ suit une distribution binomiale de prob. p, telle que

$$E(N_e) = mp \; et \; E\left((N_e - mp)^2\right) = m \times p(1-p)$$

2. t_e converge vers une loi normale car m >> 100 (loi des grands nombres)

$$t_e = 1 - t_r = \frac{\mathrm{N_e}}{m} \xrightarrow[m \to +\infty]{} N\left(p, \frac{p(1-p)}{m}\right)$$

Confiance sur les mesures (2)

3. Normalisation de la distribution :

$$t_e \xrightarrow[m\to+\infty]{} N\left(p, \frac{p(1-p)}{m}\right) \Leftrightarrow \frac{t_e - p}{\sqrt{\frac{p(1-p)}{m}}} \xrightarrow[m\to+\infty]{} N(0,1)$$

4. Calcul d'un intervalle de confiance de risque δ sur $\mathcal{N}(0,1)$:

$$P(|X| > \sigma | X \sim \mathcal{N}(0,1)) = \delta \Leftrightarrow \sigma = table_{\mathcal{N}(0,1)}(\delta)$$

5. Déduction d'un intervalle de confiance sur p fonction de δ et m :

$$P(p_{min} \leq p \leq p_{max}) = 1 - \delta \; avec \; p_{max/min} = \frac{t_e + \frac{\sigma^2}{2m} \pm \sigma\sqrt{\frac{t_e}{m} - \frac{t_e^2}{m} + \frac{\sigma^2}{4m^2}}}{1 + \frac{\sigma^2}{m}}$$

Application numérique :

Exemple des données « Glass » (identification de verres en criminologie)

1. Nbr d'exemples : $m = 214$
2. Marge d'erreur de $\delta = 1\% \Rightarrow \sigma = 2{,}58$, $\delta = 5\% \Rightarrow \sigma = 1{,}96$
3. Test en validation croisée. Mesure de t_e. Ex C4.5 : $t_e = 66\ \%$
4. Calcul de l'intervalle de confiance

t_e	$\delta = 1\ \%$		$\delta = 5\ \%$	
	p_{min}	p_{max}	p_{min}	p_{max}
66 %	57 %	73 %	59 %	72 %
90 %	83 %	94 %	85 %	93 %
99 %	95 %	99,8 %	96,6 %	99,7 %

Confiance sur les mesures (3)

Problème :

comment comparer deux méthodes (A et B) selon une mesure calculée sur différents jeux de données ?

Exemple d'un score s calculé sur k jeux de m exemples :

1. Hypothèse : exemples indép. ident. distribués et m élevé >> 100

→ s_i^A et s_i^B suivent des lois normales (loi des grands nombres) pour le jeu i

→ $d_i = s_i^A - s_i^B$ suit une loi normale (loi des grands nombres) indépendante de i :

$$d_i = s_i^A - s_i^B \sim \mathcal{N}(\mu_i, \sigma_i)$$

2. Distribution de la mesure globale :

$$d = \frac{\sum_{i=1}^{k} d_i}{k} \sim \mathcal{N}(\mu, \sigma)$$

3. Estimation des paramètres :

$$\hat{\mu} = \frac{\sum_{i=1}^{k} d_i}{k}, \hat{\sigma}^2 = \frac{\sum_{i=1}^{k} (d_i - \hat{\mu})^2}{k-1}$$

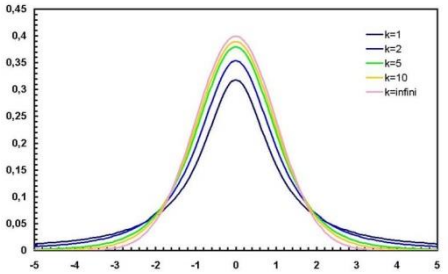
Confiance sur les mesures (4)

Problème : la variance n'est pas connue (loi du χ^2) :

$$d \sim \mathcal{N}(\mu, \sigma) \Rightarrow \frac{d - \hat{\mu}}{\sqrt{\widehat{\sigma^2}}} \sim ?$$

1. La distribution normalisée suit une loi de Student S_{k-1} de degré k-1 :

$$\frac{d - \hat{\mu}}{\sqrt{\widehat{\sigma^2}}} \sim S_{k-1}$$



2. Test d'hypothèse nulle : A et B sont équivalentes si $\mu = 0$

3. Calcul d'un intervalle de confiance de risque δ sur S(k) :

$$P(|X| > \sigma) = \delta \Leftrightarrow \sigma = table_{S_{k-1}}(\delta)$$

4. Test statistique :
   - Si $\sigma < \frac{d - \hat{\mu}}{\sqrt{\widehat{\sigma^2}}} < \sigma$ alors A et B sont statistiquement équivalents
   - Sinon si $d > 0$, A est meilleur que B
   - Sinon B est meilleur que A

Problème des classes déséquilibrées

- Compromis optimal entre sensibilité et spécificité :

$$\boxed{sensibilité(sen) = \frac{VP}{P} \qquad spécificité(spé) = \frac{VN}{N}}$$

Exemples :

- classifieur « toujours positif » $sen = 1, spé = 0$
- classifieur « toujours négatif » $sen = 0, spé = 1$
- classifieur aléatoire « positif de prob. α » $sen = \alpha, spé = 1 - \alpha$

- Biais du taux de réussite (accuracy) par le taux de positif :

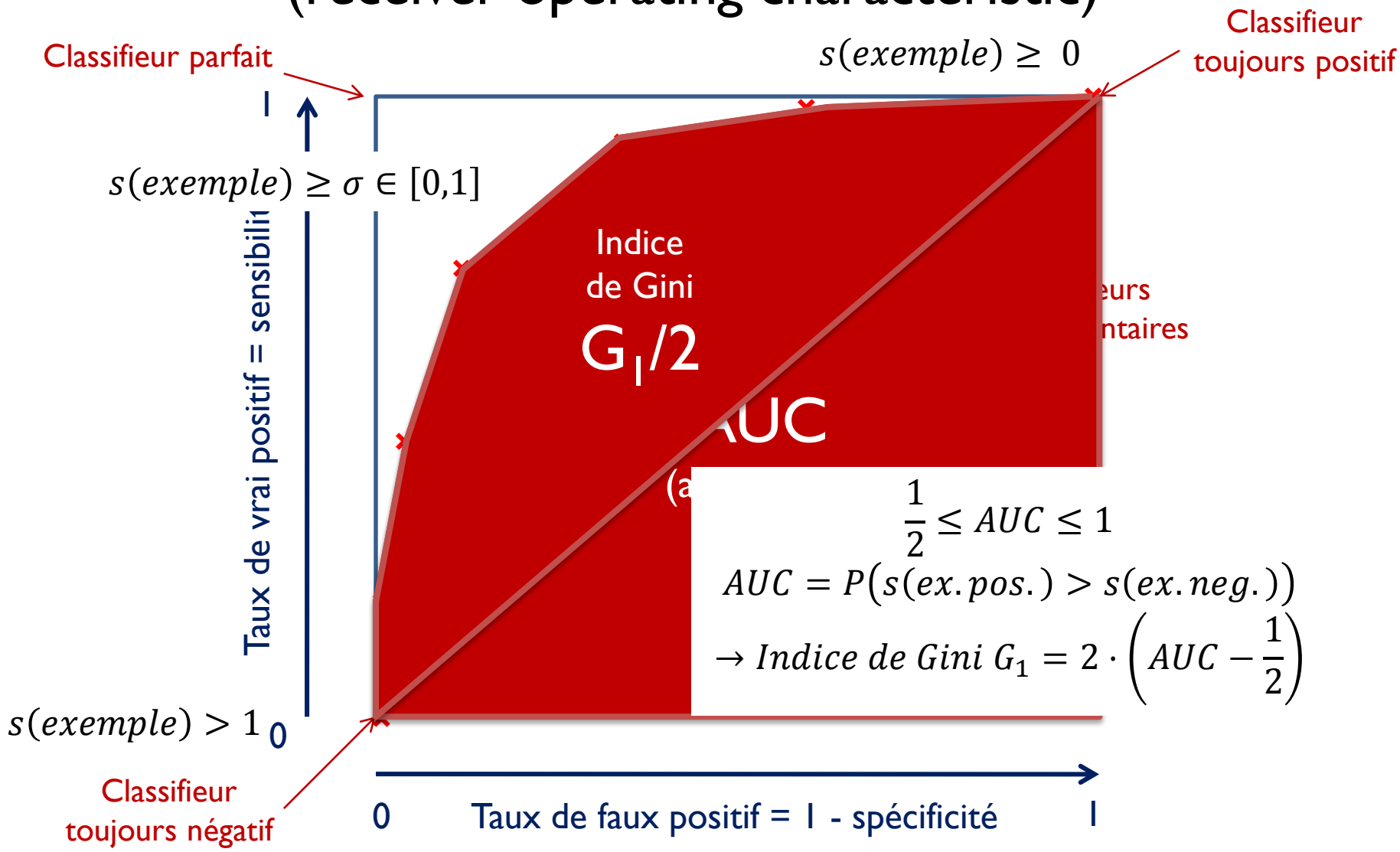
$$t_r = \frac{VP + VN}{P + N} = \frac{P}{P + N} \cdot sen + \frac{N}{P + N} \cdot spé = t_p \cdot sen + \left(1 - \mathrm{t_p}\right) \cdot spé$$

$$t_r(\text{"tjrs positif"}) = \mathrm{t_p} \quad t_n(\text{"tjrs négatif"}) = \mathrm{t_n}$$

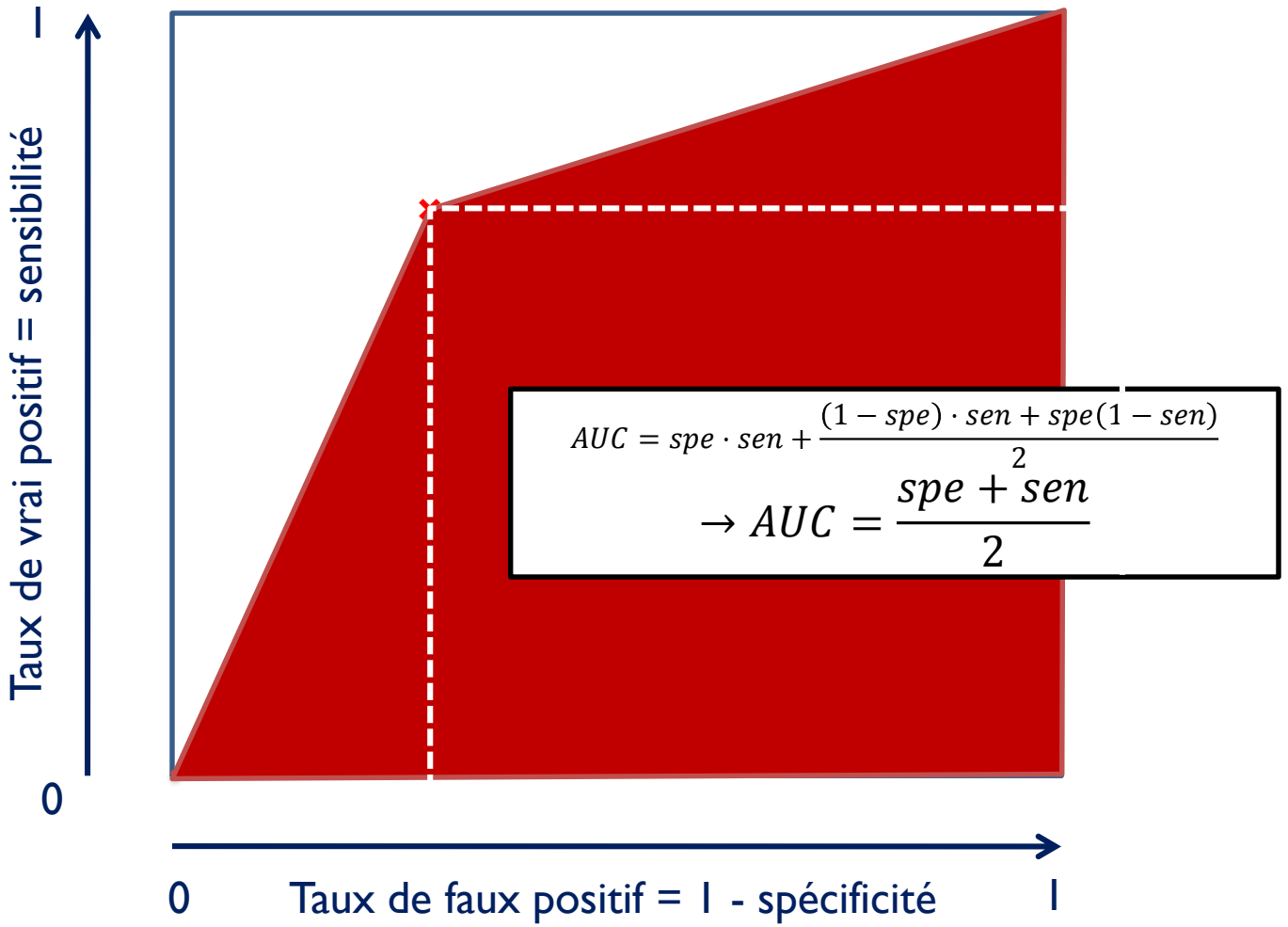
- Taux de réussite non biaisé :

$$t_r = \frac{1}{2} \cdot sen + \frac{1}{2} \cdot spé$$

Courbe ROC
(receiver operating characteristic)

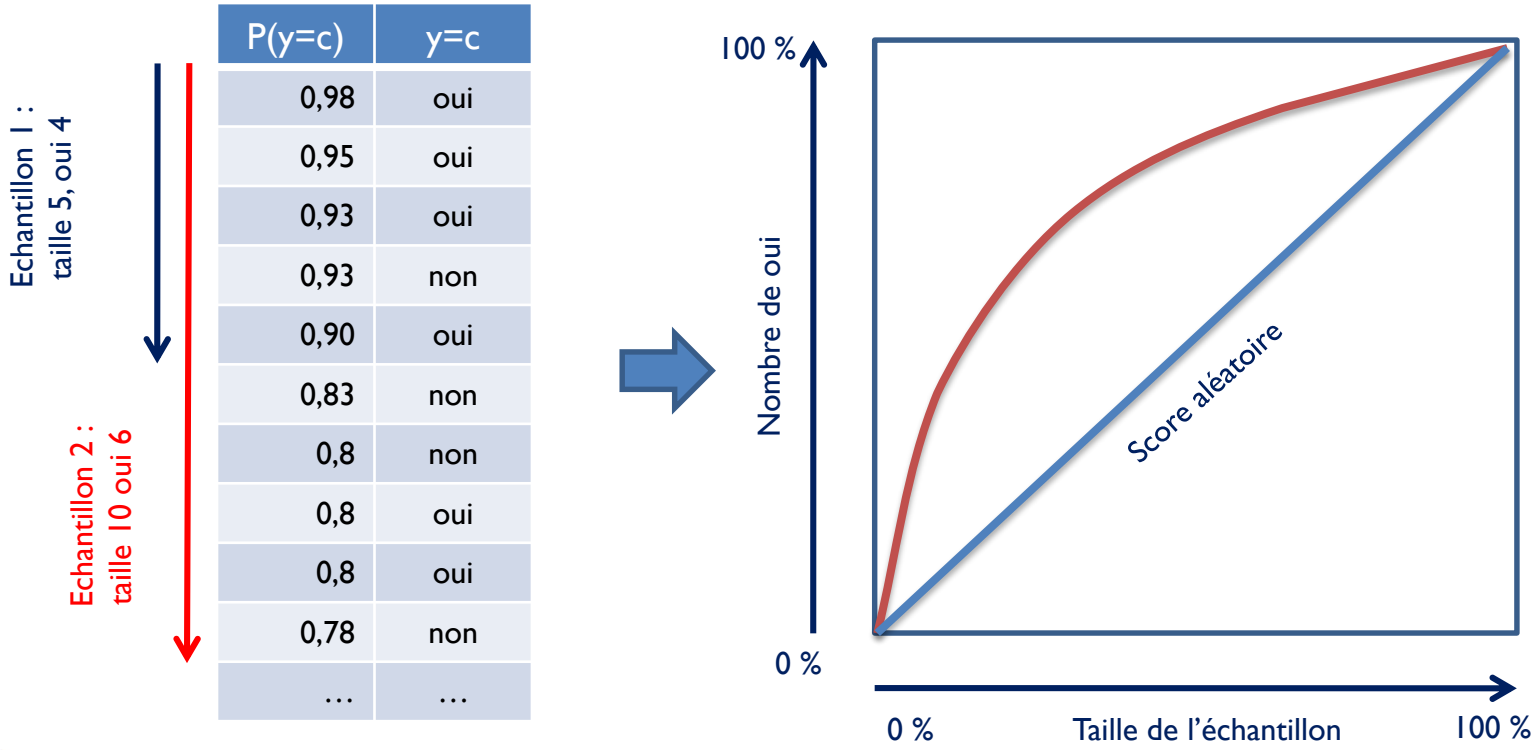


Courbe ROC
(receiver operating characteristic)



Lien avec les courbes lift (lift chart)

- Outil utilisé en « marketing » (démarchage, etc)
  - Tri des exemples par score décroissant
  - Abstraction du score
  - Lift chart : nombre de oui versus taille de l'échantillon



Erreur de classification à coût variable et aide à la décision

Application où le coût de l'erreur est asymétrique

Exemples : diagnostic médical, diagnostic de pannes, etc…

Matrice de coût :

Etat des réacteurs d'un avion

Réalité \ Prédiction	Bon état	Défaillant (dépannage)	Critique (arrêt d'exploitation)
Bon état	**100 €**	1000 €	10000 €
Défaillant (aggravation)	100000 €	**10000 €**	19000 €
Critique (risque d' accident)	100000000 €	100000000 €	**100000 €**

Critère à minimiser : $E(C) = \frac{\sum_{i,j} c_{i,j} \cdot n_{i,j}}{\sum_{i,j} n_{i,j}}$

Si $C = J - I = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ alors $E(C) = \frac{\sum_{i \neq j} n_{i,j}}{\sum_{i,j} n_{i,j}}$ vaut le taux d'erreur

Cas des classifieurs probabilistes

- Sortie : probabilité d'appartenance à chaque classe

$$\hat{P} = \left(\hat{P}(y = c_i|X)\right)_i$$

- Minimisation de l'espérance du coût :

$$\hat{E}(C|\hat{y} = c_i, X) \quad = \quad \sum_j \hat{P}(y = c_i|X) \cdot c_{j,i}$$

$$\left(\hat{E}(C|\hat{y} = c_i, X)\right)_i \quad = \quad C^T \times P$$

$$\Longrightarrow \hat{y} = \underset{c_i}{\operatorname{argmin}} \left(\hat{E}(C|\hat{y} = c_i, X)\right)$$

- Paradoxe : la plus grande probabilité n'est pas forcément la meilleure

$$C^T P = \begin{bmatrix} 10^2 & 10^3 & 10^4 \\ 10^5 & 10^4 & 10^4 \\ 10^8 & 10^8 & 10^5 \end{bmatrix}^T \cdot \begin{bmatrix} 0{,}95 \\ 0{,}049 \\ 0{,}001 \end{bmatrix} = 10^5 \cdot \begin{bmatrix} 1 \\ 1 \\ 0{,}1 \end{bmatrix} \Longrightarrow \hat{y} = \text{critique}$$

Mesures entropiques

Entropie :

$$H(p) = E_p(\log(p))$$

- Positive ou nulle. Nulle si p nulle p.p
- Quantifie l'incertitude d'une distribution

Distance ou divergence de Kullback-Leibler

$$D_{KL}(p, q) = E_p\left(\log\left(\frac{p}{q}\right)\right)$$

- Positive ou nulle. Nulle si $p = q$ p.p
- Pseudodistance entre distributions

Entropie croisée

$$H(p, q) = -E_p(\log(q)) = H(p) + D_{KL}(p, q) \geq H(p)$$

- Quantifie l'incertitude qui s'ajoute à représenter p par q

Logloss :

Estimation de l'entropie croisée entre les classes réelles (p) et la probabilité des classes prédites (q)

$$logloss = -\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{c=1}^{C}\mathbb{I}(y_i = c)\log(\hat{p}_{i,c})\right)$$