Cours de Data Science

Marc Tommasi

10 novembre 2020

Outline

Outline

Probabilités conditionnelles et Bayes

- Notations :
  - $P(A)$ la probabilité d'un événement A
  - $P(A \cap B)$ ou $P(A, B)$ la probabilité d'avoir à la fois les événements A et B.
  - $P(A \mid B)$ la probabilité d'avoir l'évenement A sachant B.

Definition des probabilités conditionnelles

$$P(A|B) = \frac{P(A,B)}{P(B)} \text{ ou encore par symétrie } P(B|A) = \frac{P(A,B)}{P(A)}$$

Donc

$$P(A,B) = P(A|B)P(B) = P(B|A)P(A)$$

Théorème de Bayes

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Petit abus de notations

- Abus de notation : je note donc $P(X = \boldsymbol{x})$ simplement par

$P(\boldsymbol{x})$. (Soit : $P(\boldsymbol{x})$ pour dire qu'une variable aléatoire X prend comme valeur $\boldsymbol{x}$ donc la probabilité de l'événement $X = \boldsymbol{x}$)

- *Exemple* : Soit X soit la variable sur deux attributs x_1 et x_2 qui peuvent prendre uniquement des valeurs binaires. Alors

$P(X)$ désigne t $P(X = (0, 0))$, $P(X = (0, 1))$, etc.. notés simplement $P(0, 0)$, $P(0, 1)$, etc. . .

- On va noter aussi $P(\boldsymbol{x})$ ou $P(x_1, x_2)$ pour désigner l'un de ces 4 cas.

Revenant à l'apprentissage

Rappels

- Les données sont générées par une probabilité jointe fixée mais inconnue d'avoir une description des données $\boldsymbol{x}$ et une classe y, écrite $P(\boldsymbol{x}, y)$.
- On veut chercher à résoudre le problème : trouver le meilleur y quand on observe un $\boldsymbol{x}$

La règle de Bayes

- C'est la meilleure règle qu'on puisse imaginer

$$\text{argmax}_y P(y \mid \boldsymbol{x})$$

- Erreur de Bayes : Erreur de cette règle
- c'est la plus petite erreur qu'on puisse faire pour cet apprentissage si les exemples sont décrits par $\boldsymbol{x}$.

Difficile à calculer

- $P(y \mid \boldsymbol{x})$ ne peut être calculée car P est inconnue.
- Si on applique le principe ERM, par la règle de Bayes on a

$$P(y \mid \boldsymbol{x}) = \frac{P(y)P(\boldsymbol{x} \mid y)}{P(\boldsymbol{x})}$$

- On cherche la valeur de y qui maximise cette quantité. Mais $P(\boldsymbol{x})$ ne dépend pas de y. Il suffit de résoudre

$$\mathrm{argmax}_y P(y)P(\boldsymbol{x} \mid y)$$

- Problème : le calcul ne peut être fait efficacement $\boldsymbol{x}$ et des valeurs qu'il peut prendre. Exemple : Dans le cas binaire avec 5 attributs, on a 2^5 possibilités et donc 2 fois 2^5 quantités à estimer pour tous les cas de $P(\boldsymbol{x} \mid y)$.

La chain rule

$$P(x_1, x_2, \ldots x_n) = P(x_1 \mid x_2, \ldots x_n)P(x_2, \ldots x_n)$$
$$= P(x_1 \mid x_2, \ldots x_n)P(x_2, \mid x_3, \ldots x_n)P(x_3, \ldots x_n)...$$

- Obtenu en appliquant $P(A, B) = P(A|B)P(B)$ de façon répétée quand B est un événement qui peut être une conjonction d'événements
- Par récursion

$$P(\boldsymbol{x} \mid y) = \prod_{k=1}^{n} P(x_k \mid x_{k-1}, ..., x_1, y)$$

Indépendance Conditionnelle et Naive Bayes

- Si A et B sont indépendants alors $P(A \mid B) = P(A)$.
- Si on fait l'hypothèse que tous les attributs sont indépendants,

c'est-à-dire si x_i et x_j sont indépendants pour tout $i \neq j$, alors le produit s'écrit :

$$P(\boldsymbol{x} \mid y) = \prod_{k=1}^{n} P(x_k \mid y)$$

- Pour calculer chacun de ces $P(x_k \mid y)$ il suffit de compter !
- C'est une approximation forte !
- Mais le calcul est de faible complexité

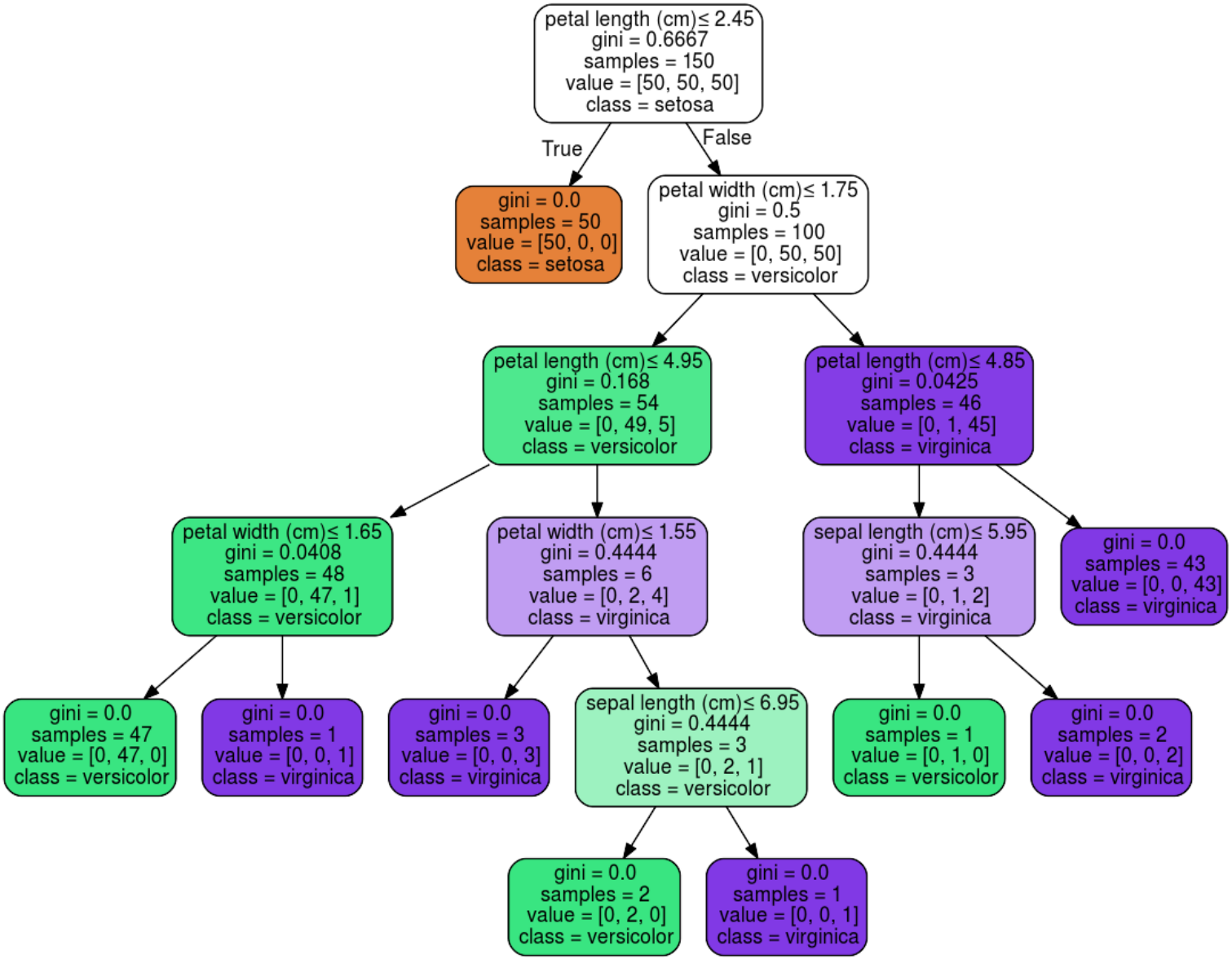
$$\text{argmax}_y P(y) \prod_{k=1}^{n} P(x_k \mid y)$$

Outline

Principes

- Classifieur de $\mathcal{X}$ dans $\mathcal{Y}$
- Modèle sous forme d'un ensemble de règles de décision successives, représentées dans un arbre
- Modèle simple à interpréter quand l'arbre est petit
- Exemple sur le jeu de données iris, 3 classes : setosa, virginica, versicolor ; 150 exemples ; attributs sepal length, sepal width, petal length, petal width.

Exemple


petal length (cm)≤ 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm)≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm)≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm)≤ 4.85
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm)≤ 1.65
gini = 0.0408
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm)≤ 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm)≤ 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm)≤ 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

Fonctionnement

- On part de la racine, avec un exemple (e.g. 3, 2, 3, 2)
- On passe à travers les tests, chaque test coupe l'espace en 2 parties.
- On désigne donc un partitionnement récursif de l'espace de description
- Chaque feuille donne une étiquette à une partie.
- Bonne visualisation dans le livre de Jake VandenPlas.

Algorithmes

- C'est une méthode qui introduit deux biais : choix de la classe de fonctions + biais algorithmique
- Le biais algorithmique provient d'une heuristique gourmande :
  - on construit l'arbre de la racine aux feuilles,
  - à chaque étape on développe un noeud correspondant à une partie des données
  - on sélectionne le meilleur test selon un critère de gain
  - on ne remet plus en cause ce choix
- Il existe plusieurs algorithmes : ID3, C4.5, C5, CART,...
- Sklearn implante CART

Explications avec ID3

- Les attributs $A = \{x_0, x_1, \ldots, x_p\}$ sont tous binaires

```
def id3(S,A):
  """ S : echantillon, A: attributs """
  if tous les éléments de S sont de même classe ou
     A est vide:
    retourner une feuille contenant la classe majoritaire

  Soit j l'attribut qui maximise le gain
    t_l = id3({(x,y) de S tq x_j=0}, A\{j})
    t_r = id3({(x,y) de S tq x_j=1}, A\{j})
    retourner l'arbre de noeud qui teste x_j=1
              avec les fils t_l (cas False) et t_r (cas True).
```

Fonctions de gain

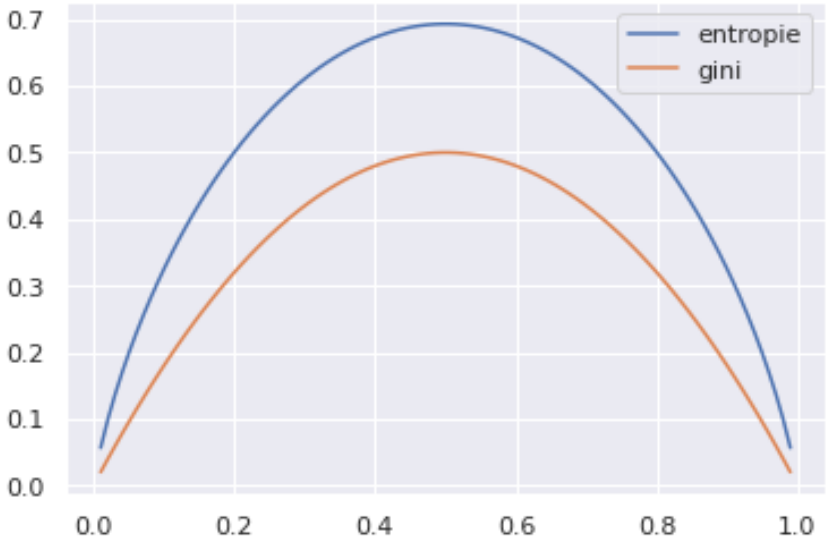
- Gain : différence observée entre absence et présence du test dans l'arbre.
- Notation : $\mathbb{P}_S[F]$ est la probabilité de F quand on tire uniformément dans S
  - (si le noeud a m exemples et m_l passent à gauche et m_r passent à droite avec un test x_i alors $\mathbb{P}_S[x_i = 1] = m_r/m$)
- le gain sera calculé avec une fonction C à définir :

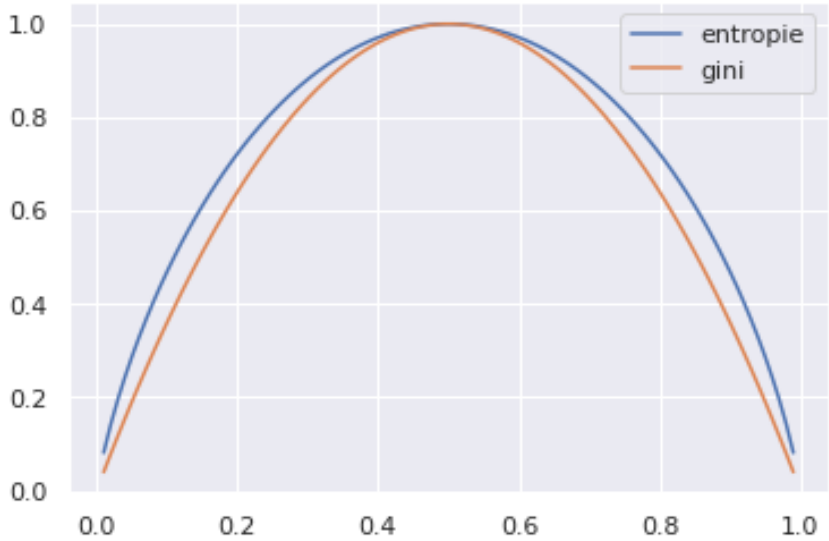
$$\text{Gain}(S, i) = C((\mathbb{P}_S[y = 1]) - \big(\mathbb{P}_S[x_i = 1]C(\mathbb{P}_S[y = 1 \mid x_i = 1]) + \mathbb{P}_S[x_i = 0]C(\mathbb{P}_S[y = 1 \mid x_i = 0])\big)$$

- Erreur d'apprentissage : différence entre les erreurs faites par l'arbre avant et après l'introduction de ce noeud. $C(a) = \min(a, 1 - a)$
- Gain en information : différence entre l'entropie avant et après le test. $C(a) = -a\log(a) - (1 - a)\log(1 - a)$
- Gini, pureté : $C(a) = 2a(1 - a)$ (facteur 2 pour avoir un maximum à 1)

Entropie et gini

- Vont favoriser les tests qui réalisent la meilleure séparation : $\mathbb{P}_S[y \mid x_i]$ proche de 0 ou de 1.





- $-a\log(a) - (1-a)\log(1-a)$ et $2a(1-a)$
- $-a\log_2(a) - (1-a)\log_2(1-a)$ et $4a(1-a)$

Limiter l'overfitting

- Compromis biais/complexité : plus la profondeur est grande, plus la classe de fonctions est complexe, plus le biais est faible mais plus les arbres de décision auront tendance à l'overfitting.
- borne sur la profondeur (comme dans sklearn, `max_depth`)
- l'élagage (pruning) consiste à supprimer des branches : remplacer un noeud par une étiquette de classe
  - Approche bottom-up avec un test statistique : (ξ^2 ou évaluation de l'erreur).

Le cas des attributs continus

- discrétisation considérant tous les seuils possibles observés sur l'échantillon d'apprentissage
- le calcul pour ces m tests possibles pourrait être $O(dm^2)$ mais peut être réduit à $O(dm\log(m))$.

Arbres de régression

- on fait la moyenne des exemples qui arrivent dans une feuille pour déterminer la valeur à prédire
- la fonction de coût pour la construction utilisée est par exemple MSE

Avantages et inconvénients

- lisibilité du modèle
- complexité algorithmique élevée pour trouver le meilleur arbre, mais approche heuristique rapide.
- difficulté de régler la profondeur, les critères qui évitent le sur-apprentissage

Outline

Principales mesures

- Taux d'erreurs (accuracy) : nombre d'exemples mal classés sur le nombre total d'exemples.
- Matrice de confusion : répartition des exemples selon la prédiction et leur vraie classe.

	Positifs	Négatifs
Prédiction Positive	TP	FP
Prédiction Négative	FN	TN

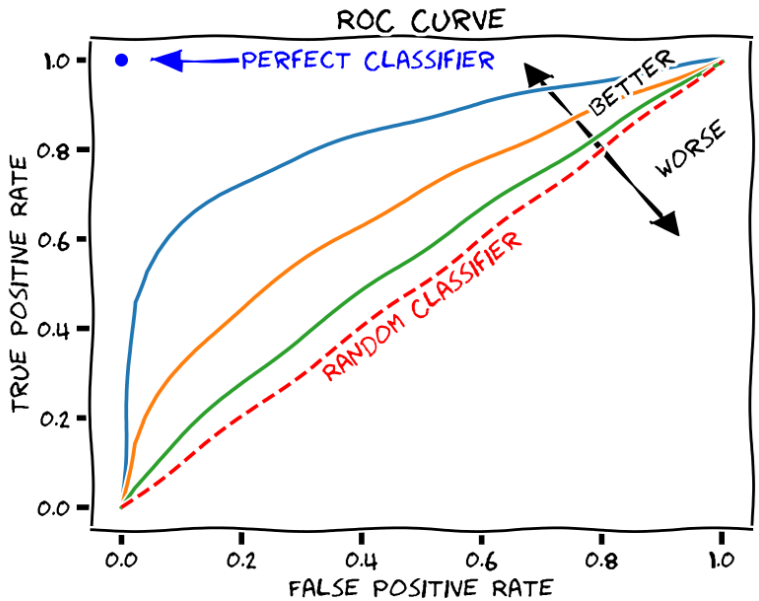
- Précision ($\frac{\mathrm{TP}}{\mathrm{TP+FP}}$) et Rappel ($\frac{\mathrm{TP}}{\mathrm{TP+FN}}$)
- Précision et rappels sont liés en proportionnalité inverse. Pour avoir un seul score moyen, on fait la moyenne harmonique F1 ($2 \times \frac{\mathrm{Precision} \times \mathrm{Rappel}}{\mathrm{Precision+Rappel}}$)
- On s'intéresse aussi aux taux de vrai positifs (TPR) et faux positifs (FPR) ($\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ et $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$)
- Voir Wikipedia pour toutes les mesures (vocabulaire) qui en sont dérivées.

Fonction de décision

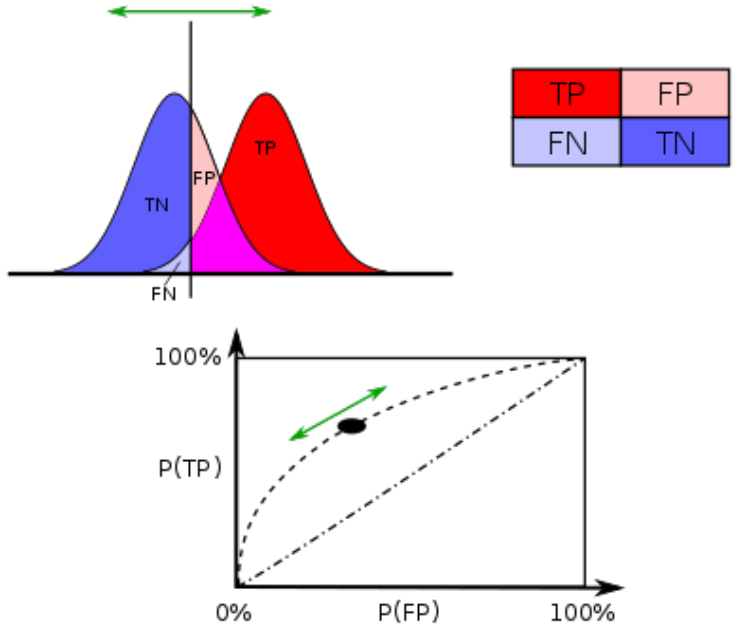
- Pour obtenir les prédictions, un score est calculé puis le score est comparé à un seuil
- Le seuil peut être changé et vu comme un hyper-paramètre

Courbe ROC

- La courbe ROC (Receiver operating characteristic) trace FPR et TPR dans un diagramme en fonction du seuil.



- Quand la sortie du classeur est vue comme une variable aléatoire



- La surface sous la courbe ROC peut être interprétée comme la probabilité du classeur de donner un score plus élevé à un exemple positif qu'à un exemple négatif (`roc_auc_score`)

Pour les problèmes multi-classes

- Les mesure marchent aussi dans le cadre multi-classes, avec certaines restrictions.
- Les taux de FPR, TPR, etc sont calculés par classe
- On est amenés à faire des moyennes sur les classes
- La moyenne n'est pas un bon indicateur
- On peut corriger par des techniques de macro ou micro average selon les problèmes réels
- Exemple :
  - Classe A : 1 TP et 1 FP ; Classe B : 10 TP et 90 FP ; Classe C : 1 TP et 1 FP ; Classe D : 1 TP et 1 FP.
  - macro-average : $(0.5 + 0.1 + 0.5 + 0.5)/4 = 0.4$
  - micro-average : $(1 + 10 + 1 + 1)/(2 + 100 + 2 + 2) = 0.123$
- on peut aussi pondérer les moyennes par les proportions dans chaque classe.

En sklearn

- Voir les méthodes dans `sklearn.metrics`
  - `confusion_matrix`
  - `precision_score`, `recall_score` et `f1_score`,
  - `classification_report` : un tableau avec les principales mesures
- les classifieurs ont une méthode `decision_function` ou une fonction `predict_proba`
- on peut tracer les courbes `precision_recall_curve` et `roc_curve` et matrice de confusion par `plot_confusion_matrix`, `plot_precision_recall_curve`, `plot_roc_curve`