Apprentissage supervisé

Quentin Fortier

January 11, 2026

Science des données

La science des données (*data science*) a pour objectif d'extraire de l'information à partir de données brutes.

Science des données

La science des données (*data science*) a pour objectif d'extraire de l'information à partir de données brutes.

Exemples :

- Données sur des fleurs : longueur et largeur des pétales et des sépales.
- Données sur des élèves : moyenne, classe...
- Données sur les clients d'une banque : âge, épargne, ...

Science des données : Représentation

Pour pouvoir avoir une notion de distance entre deux données, on représente chaque donnée comme un vecteur de $\mathbb{R}^p$.

Exemple : chaque donnée de fleur peut être représentée par un quadruplet de $\mathbb{R}^4$ correspondant à la longueur et largeur des pétales et des sépales.

Les composantes de ce vecteur sont appelées les attributs.

Science des données : Représentation

Parfois il est moins évident de représenter une donnée par un vecteur :

Science des données : Représentation

Parfois il est moins évident de représenter une donnée par un vecteur :

- Variable catégorielle (non numérique : genre, couleur...) : on utilise souvent un vecteur avec un 1 et que des 0 (*one-hot vector*).
  Exemple : on peut représenter les classes MP2I/MPI/MPSI/MP par des vecteurs $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$ ...

Science des données : Représentation

Parfois il est moins évident de représenter une donnée par un vecteur :

- Variable catégorielle (non numérique : genre, couleur...) : on utilise souvent un vecteur avec un 1 et que des 0 (*one-hot vector*). Exemple : on peut représenter les classes MP2I/MPI/MPSI/MP par des vecteurs $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$...
- Image : On passe d'une matrice de pixels avec n lignes, p colonnes à un vecteur de taille np.

Science des données : Représentation

Parfois il est moins évident de représenter une donnée par un vecteur :

- Variable catégorielle (non numérique : genre, couleur...) : on utilise souvent un vecteur avec un 1 et que des 0 (*one-hot vector*). Exemple : on peut représenter les classes MP2I/MPI/MPSI/MP par des vecteurs $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$ ...
- Image : On passe d'une matrice de pixels avec n lignes, p colonnes à un vecteur de taille np.
- Son : Transformée de Fourier discrète.

Science des données : Représentation

On représente classiquement l'ensemble des données (donc de vecteurs de $\mathbb{R}^p$) par une matrice X dont chaque ligne est une donnée et chaque colonne est un attribut.

OCaml	Matrice	Données
`x.(i)`	$i$ème ligne	$i$ème donnée
`Array.length`	nombre de lignes	nombre de données
`x.(i).(j)`	élément ligne i, colonne j	$j$ème attribut de la $i$ème donnée
`Array.length x.(0)`	nombre de colonnes	nombre d'attributs

Science des données : Distance

Pour savoir si deux données sont « proches » l'une de l'autre, on utilise une distance sur les données, c'est-à-dire sur $\mathbb{R}^p$.

Exemples :

- Distance euclidienne :

$$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

- Distance de Manhattan :

$$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

Science des données : Standardisation

Quand les attributs n'ont pas la même échelle, un attribut peut avoir beaucoup plus d'importance qu'un autre dans les calculs de distance.

Science des données : Standardisation

Quand les attributs n'ont pas la même échelle, un attribut peut avoir beaucoup plus d'importance qu'un autre dans les calculs de distance.

Pour que les attributs aient la même importance, on peut standardiser les données, c'est-à-dire les modifier pour avoir une moyenne de 0 et un écart-type de 1.

Science des données : Standardisation

Quand les attributs n'ont pas la même échelle, un attribut peut avoir beaucoup plus d'importance qu'un autre dans les calculs de distance.

Pour que les attributs aient la même importance, on peut standardiser les données, c'est-à-dire les modifier pour avoir une moyenne de 0 et un écart-type de 1.

La plupart des algorithmes de science des données fonctionnent mieux avec des données standardisées.

Science des données : Standardisation

Théorème

Si Z est une variable aléatoire d'écart-type $\sigma \neq 0$ alors $\dfrac{Z - \mathbb{E}(Z)}{\sigma}$ a une espérance nulle et un écart-type égal à 1.

Exercice

Écrire une fonction
`float standardiser(float** X, int n, int p)` qui standardise les données `X` de taille $n \times p$.

Apprentissage supervisé

Problème d'apprentissage supervisé

Inconnu : $f : X \longrightarrow Y$, où X un ensemble de données et Y un ensemble d'étiquettes (ou classes).
Entrée : des données d'entraînement $x_1, ..., x_n \in X$ et leurs étiquettes $f(x_1), ..., f(x_n) \in Y$.
Sortie : une fonction $g : X \longrightarrow Y$ approximant f.

À partir de données d'entraînement dont on connaît la classe, on veut prédire la classe de nouvelles données.

Apprentissage supervisé

Problème d'apprentissage supervisé

Inconnu : $f : X \longrightarrow Y$, où X un ensemble de données et Y un ensemble d'étiquettes (ou classes).
Entrée : des données d'entraînement $x_1, ..., x_n \in X$ et leurs étiquettes $f(x_1), ..., f(x_n) \in Y$.
Sortie : une fonction $g : X \longrightarrow Y$ approximant f.

À partir de données d'entraînement dont on connaît la classe, on veut prédire la classe de nouvelles données.

Suivant l'ensemble possible d'étiquettes, on parle de :

- Classification : Y est fini, par exemple $Y = \{1, ..., k\}$.
  Exemples : k plus proches voisins, arbre de décision, réseau de neurones...

Apprentissage supervisé

Problème d'apprentissage supervisé

Inconnu : $f : X \longrightarrow Y$, où X un ensemble de données et Y un ensemble d'étiquettes (ou classes).
Entrée : des données d'entraînement $x_1, ..., x_n \in X$ et leurs étiquettes $f(x_1), ..., f(x_n) \in Y$.
Sortie : une fonction $g : X \longrightarrow Y$ approximant f.

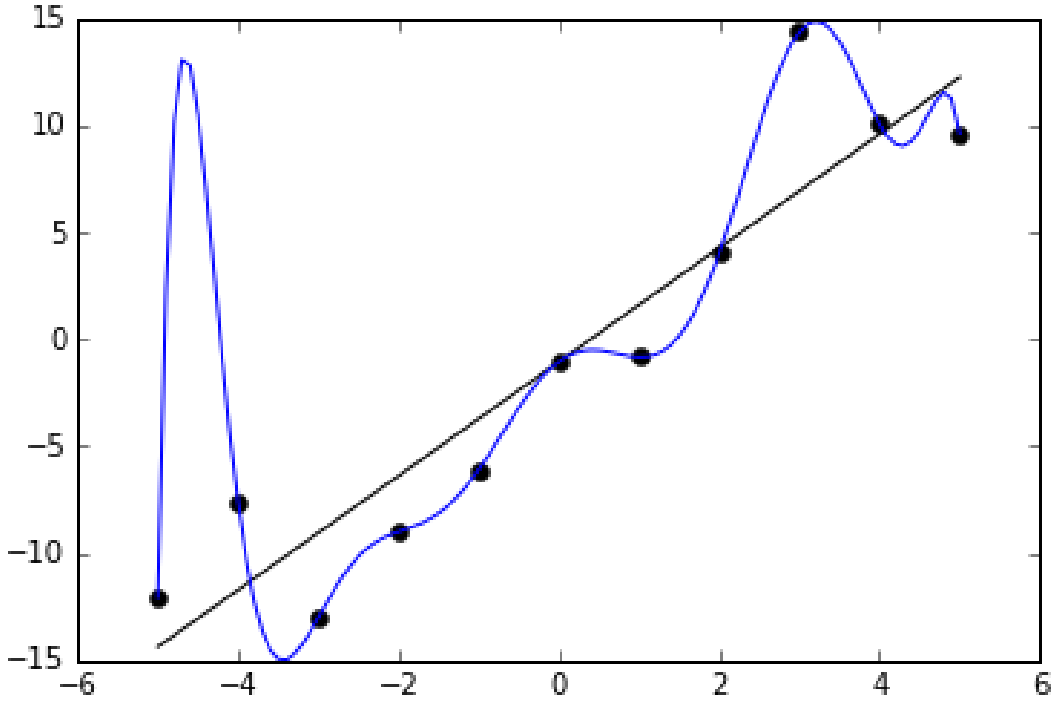
À partir de données d'entraînement dont on connaît la classe, on veut prédire la classe de nouvelles données.

Suivant l'ensemble possible d'étiquettes, on parle de :

- Classification : Y est fini, par exemple $Y = \{1, ..., k\}$.
  Exemples : k plus proches voisins, arbre de décision, réseau de neurones...
- Régression : Y est un ensemble continu, par exemple $Y = \mathbb{R}$.
  Exemples : régression linéaire, modèle linéaire généralisé...

Apprentissage supervisé

On souhaite éviter le surapprentissage (*overfitting*) où l'algorithme approxime trop bien les données d'entraînement et ne généralise mal sur de nouvelles données.



Le polynôme de Lagrange passe par tous les points d'entraînement mais généralise moins bien que la régression linéaire.

Apprentissage supervisé

Problème d'apprentissage supervisé

Inconnu : $f : X \longrightarrow Y$, où X un ensemble de données et Y un ensemble d'étiquettes (ou classes).
Entrée : des données d'entraînement $x_1, ..., x_n \in X$ et leurs étiquettes $f(x_1), ..., f(x_n) \in Y$.
Sortie : une fonction $g : X \longrightarrow Y$ approximant f.

Exemples de problèmes de classification :

données X	classes Y	$f(x)$
Tailles de tumeurs	Maligne, Bénigne	Gravité de x
Mails	Spam, Non-spam	Ce mail est-il un spam ?
Images	$[\![0, 9]\!]$	Chiffre représenté sur x ?
Musiques	classique, rap, rock...	Genre musical de x

Algorithme des k plus proches voisins (KNN)

Soit $k \in \mathbb{N}$.
L'algorithme des k plus proches voisins prédit la classe d'une nouvelle donnée x de la façon suivante :

1. Trouver les k données d'entraînement les plus proches de x (en termes de distance).

Algorithme des k plus proches voisins (KNN)

Soit $k \in \mathbb{N}$.
L'algorithme des k plus proches voisins prédit la classe d'une nouvelle donnée x de la façon suivante :

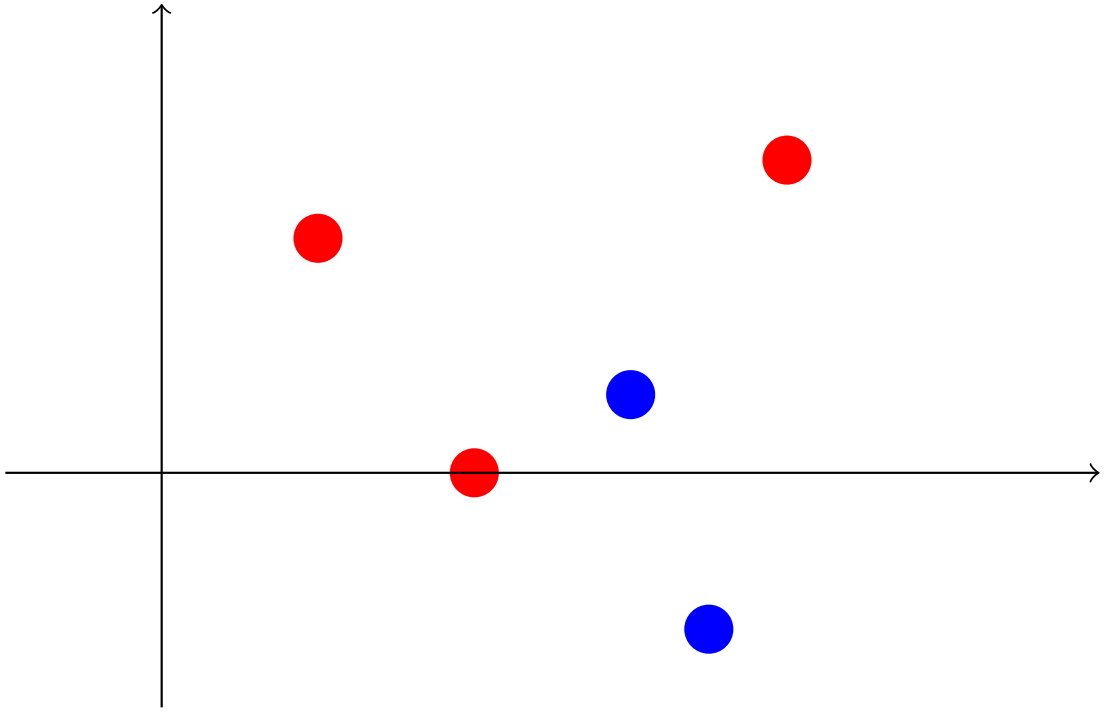
1. Trouver les k données d'entraînement les plus proches de x (en termes de distance).
2. Trouver la classe majoritaire $c \in Y$ parmi de ces k données.

Algorithme des k plus proches voisins (KNN)

Soit $k \in \mathbb{N}$.
L'algorithme des k plus proches voisins prédit la classe d'une nouvelle donnée x de la façon suivante :

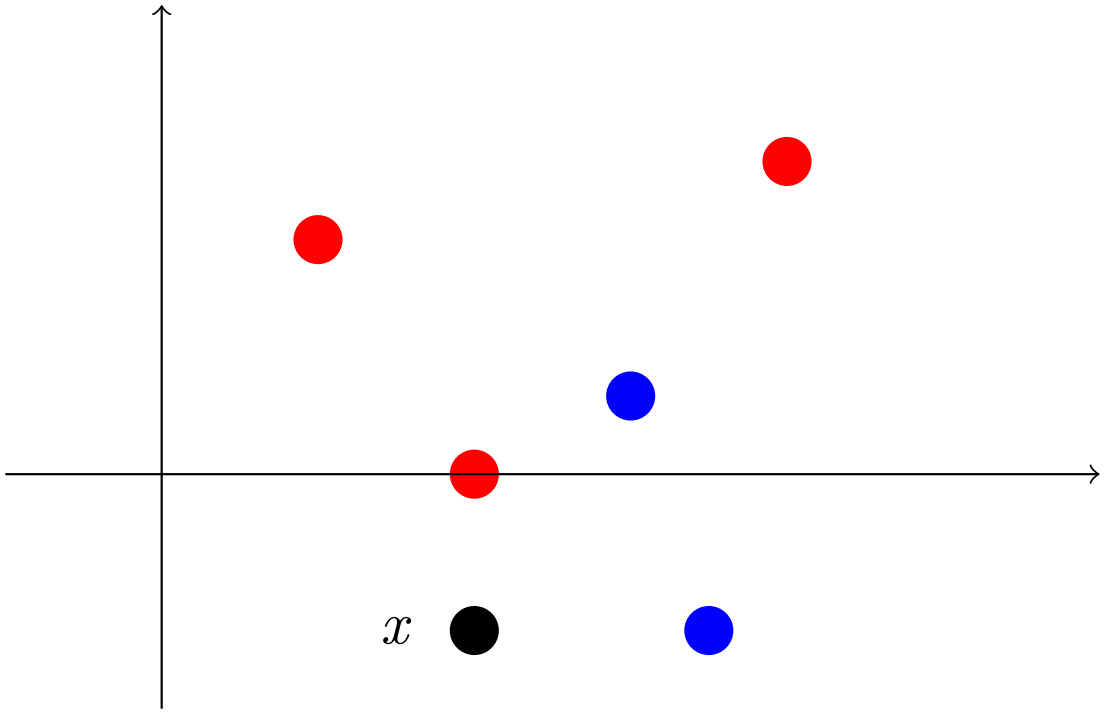
1. Trouver les k données d'entraînement les plus proches de x (en termes de distance).
2. Trouver la classe majoritaire $c \in Y$ parmi de ces k données.
3. Prédire que x est de classe c.

Algorithme des k plus proches voisins (KNN)



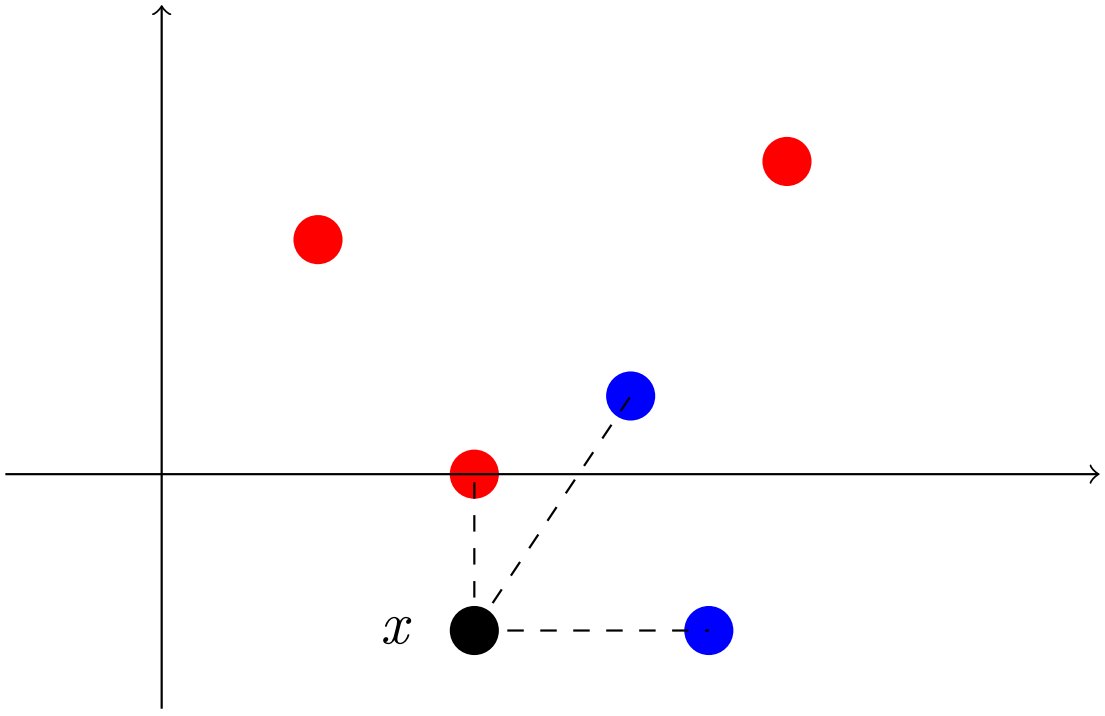
Des données dont les classes (rouge ou bleues) sont connues.

Algorithme des k plus proches voisins (KNN)



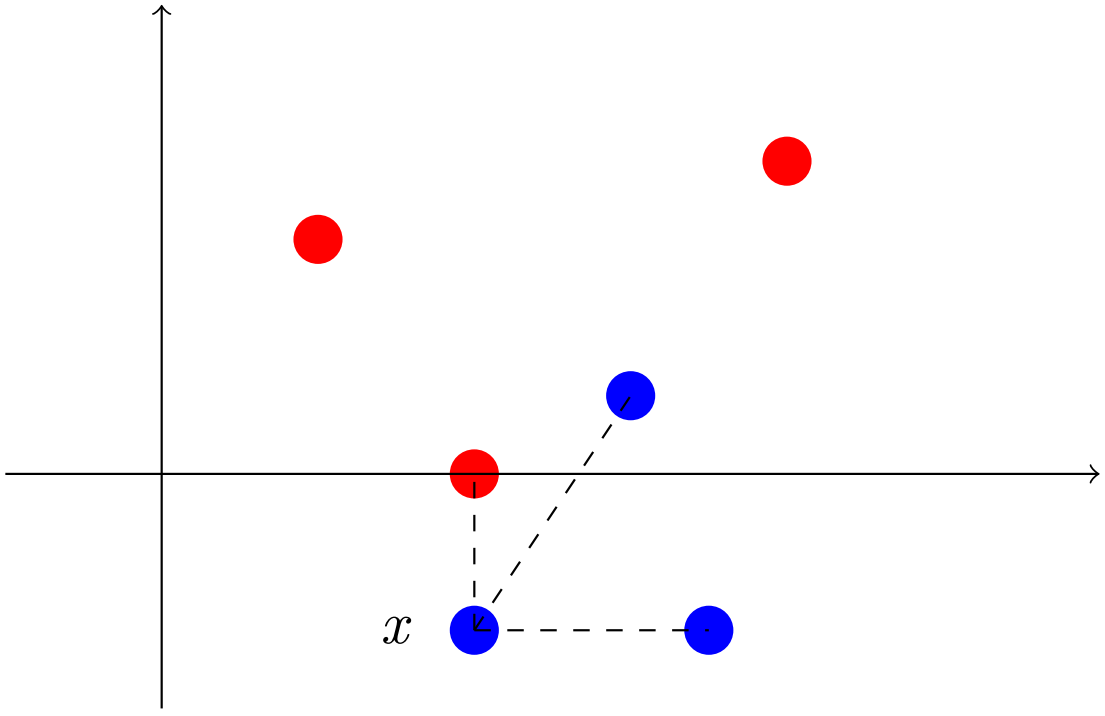
On veut prédire la classe d'une nouvelle donnée x.

Algorithme des k plus proches voisins (KNN)



On trouve les k plus proches voisins.

Algorithme des k plus proches voisins (KNN)



On associe à x la classe majoritaire de ses plus proches voisins.

Étape 1 : Trouver les k plus proches voisins.

Question

Soit `x` un vecteur sous forme de tableau, `X` une matrice de données, `k` un entier et `d` une fonction de distance supposée définie.
Écrire une fonction `int* voisins(float* x, float** X, int k)` renvoyant un tableau des indices des k plus proches voisins de `x` dans `X`.

```
bool in(int* T, int n, int x) {
    for(int i = 0; i < n; i++)
        if(T[i] == x) return true;
    return false;
}
int* voisins(float* x, float** X, int k, int n, int p) {
    int* I = malloc(k * sizeof(int));
    for(int i = 0; i < k; i++) { // ajout du ième minimum
        int jmin = -1;
        for(int j = 0; j < n; j++)
            if(jmin == -1 || d(x, X[j], p) < d(x, X[jmin], p))
                if(!in(I, k, j))
                    jmin = j;
        I[i] = jmin;
    }
    return I;
}
```

Complexité : $O(kn(p+k))$, si d est en $O(p)$.

Étape 1 : Trouver les k plus proches voisins.

Autres solutions :

- Trier les données d'entraînement par ordre croissant de distance à x et prendre les k premières en $O(np + n\log(n))$.
- Utiliser une file de priorité (tas min) en $O(np + n\log(k))$.

Étape 2 : Trouver la classe majoritaire

Question

Écrire une fonction `int maj(int* T, int n, int k)` renvoyant en $O(n)$ l'élément le plus fréquent d'un tableau `T` de taille `n` dont les éléments sont compris entre 0 et $k-1$.

Étape 2 : Trouver la classe majoritaire

Question

Écrire une fonction `int maj(int* T, int n, int k)` renvoyant en $O(n)$ l'élément le plus fréquent d'un tableau `T` de taille `n` dont les éléments sont compris entre 0 et $k-1$.

```
int maj(int* T, int n, int k) {
    int* compte = malloc(n, sizeof(int));
    for(int i = 0; i < n; i++) compte[i] = 0;
    for(int i = 0; i < n; i++) compte[T[i]]++;
    int c = 0;
    for(int i = 1; i < k; i++)
        if(compte[i] > compte[c])
            c = i;
    free(compte);
    return c;
}
```

Complexité : $O(n)$.

Étape 3 : Prédire la classe de x

Exercice

Écrire une fonction

```
int knn(float* x, float** X, int* Y, int k, int n, int p)
```

qui prédit la classe de `x` en utilisant l'algorithme KNN, où :

- `x` est la donnée à prédire,
- `X` est la matrice des données d'entraînement,
- `Y[i]` est la classe de la donnée `X[i]`,
- `k` est le nombre de voisins à considérer,
- `n` est le nombre de données d'entraînement,
- `p` est le nombre d'attributs.

Étape 3 : Prédire la classe de x

```
int knn(float* x, float** X, int* Y, int k, int n, int p) {
    int* I = voisins(x, X, k, n, p);
    int* classes = malloc(k * sizeof(int));
    for(int i = 0; i < k; i++)
        classes[i] = Y[I[i]];
    int c = maj(classes, k);
    free(I);
    free(classes);
    return c;
}
```

Évaluation d'un algorithme d'apprentissage

Supposons posséder des données X avec des étiquettes Y et qu'on veuille savoir si KNN est un bon classifieur.

Pour cela, on partitionne X en deux ensembles :

- Ensemble d'entraînement X_{train} (de classes Y_{train}) : données parmi lesquelles on va chercher les k plus proches voisins.
- Ensemble de test X_{test} (de classes Y_{test}) : données utilisées pour évaluer l'algorithme, en comparant les classes prédites par KNN avec les classes réelles.

Évaluation d'un algorithme d'apprentissage

Définition

- La précision d'un algorithme d'apprentissage est la proportion de données de test bien classées par rapport au nombre total de données.
- L'erreur est égale à $1 -$ précision.

Évaluation d'un algorithme d'apprentissage

Définition

La matrice de confusion est une matrice carrée dont les lignes et les colonnes sont les classes possibles. La case (i, j) contient le nombre de données de test de classe i qui ont été prédites comme appartenant à la classe j.

Évaluation d'un algorithme d'apprentissage

Définition

La matrice de confusion est une matrice carrée dont les lignes et les colonnes sont les classes possibles. La case (i, j) contient le nombre de données de test de classe i qui ont été prédites comme appartenant à la classe j.

Exemple : Dans la matrice suivante, on voit que toutes les données de classe 0 ont été prédites correctement, une donnée de classe 1 a été prédite à tord comme appartenant à la classe 2.

$$\begin{pmatrix} 21 & 0 & 0 \\ 0 & 29 & 1 \\ 0 & 2 & 23 \end{pmatrix}$$

La précision est la somme des éléments diagonaux divisée par la somme de tous les éléments.

Évaluation d'un algorithme d'apprentissage

Question

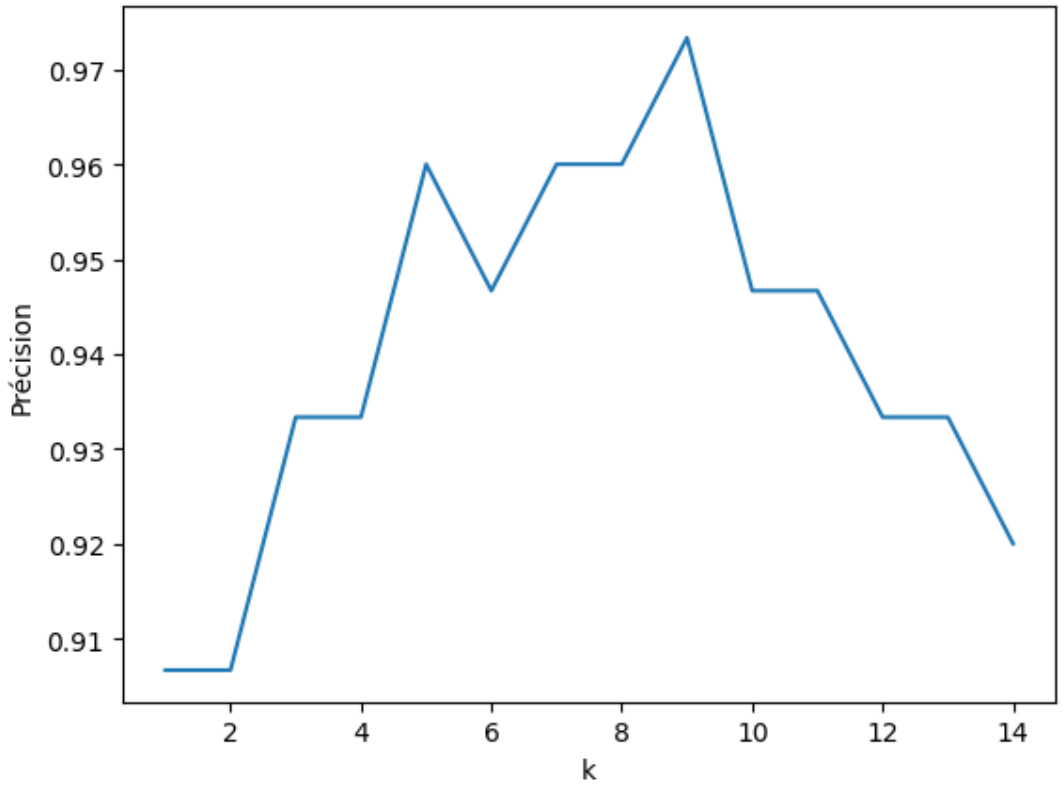
Comment choisir la valeur de k dans l'algorithme des k plus proches voisins ?

Évaluation d'un algorithme d'apprentissage

Question

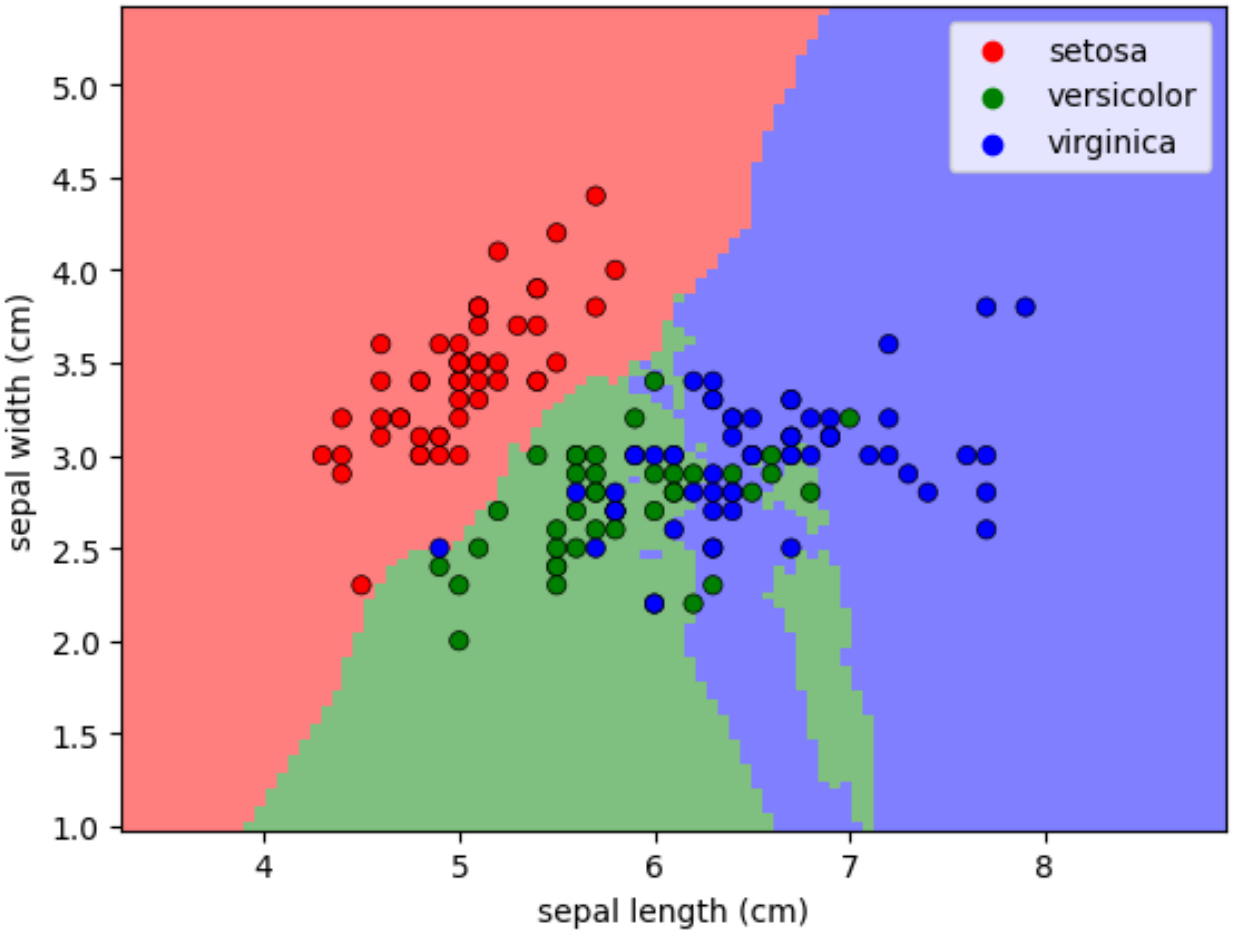
Comment choisir la valeur de k dans l'algorithme des k plus proches voisins ?

On peut afficher la précision en fonction de k pour choisir la valeur de k qui donne la meilleure précision.



Évaluation d'un algorithme d'apprentissage

On peut aussi visualiser la frontière de décision (*decision boundary*) permettant de voir à quelle classe est associée chaque point de l'espace des données :



Exemple complet : classification d'iris

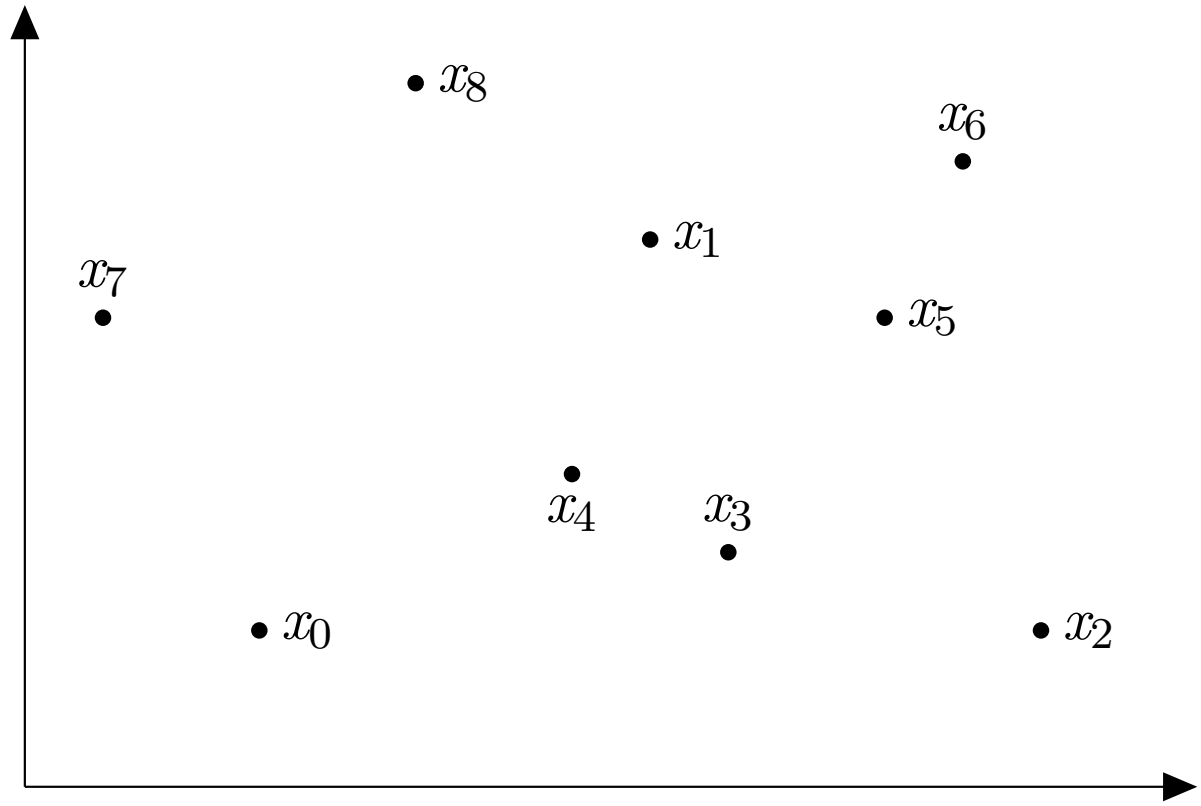
Exemples en Python :

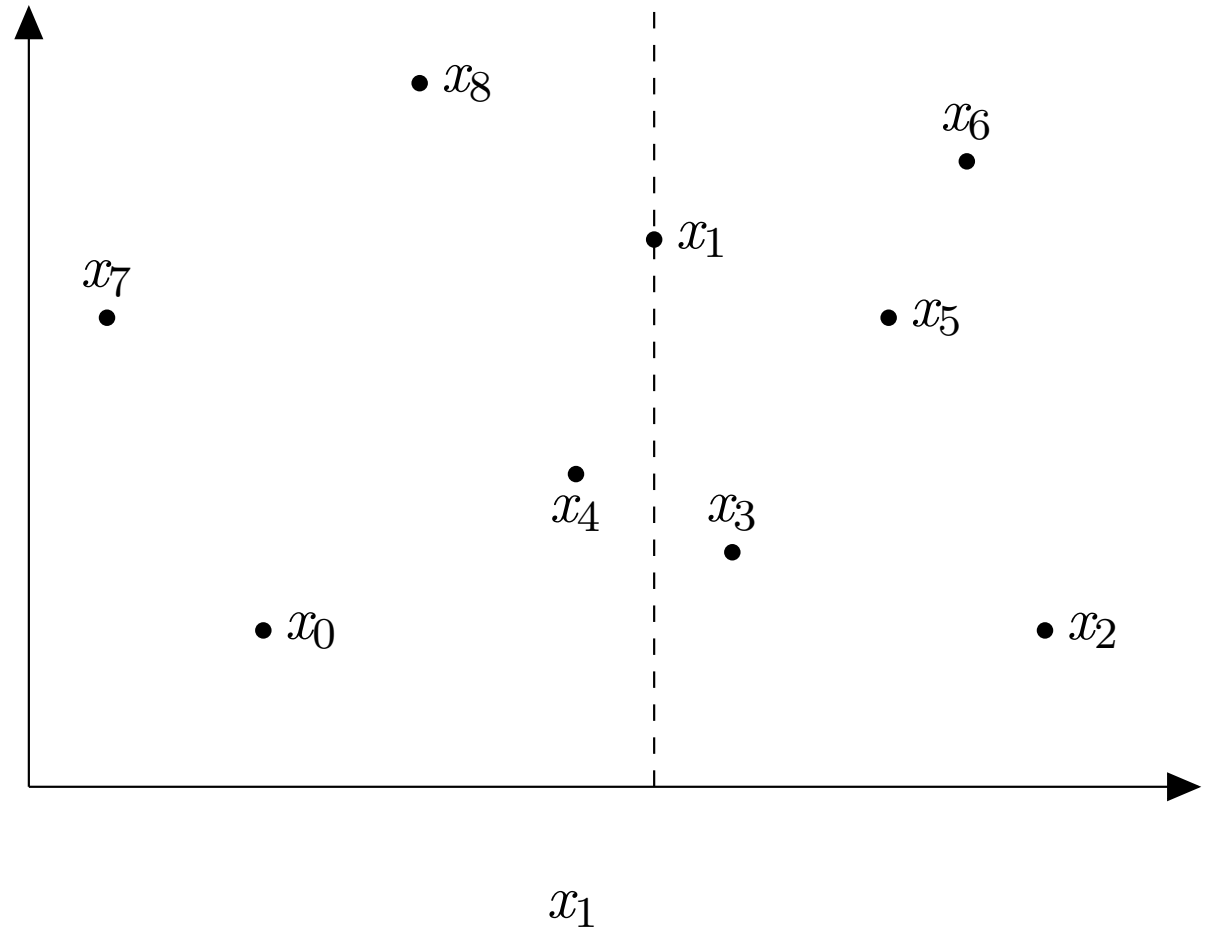
- Classification de fleurs
- Classification de chiffres manuscrits (MNIST).

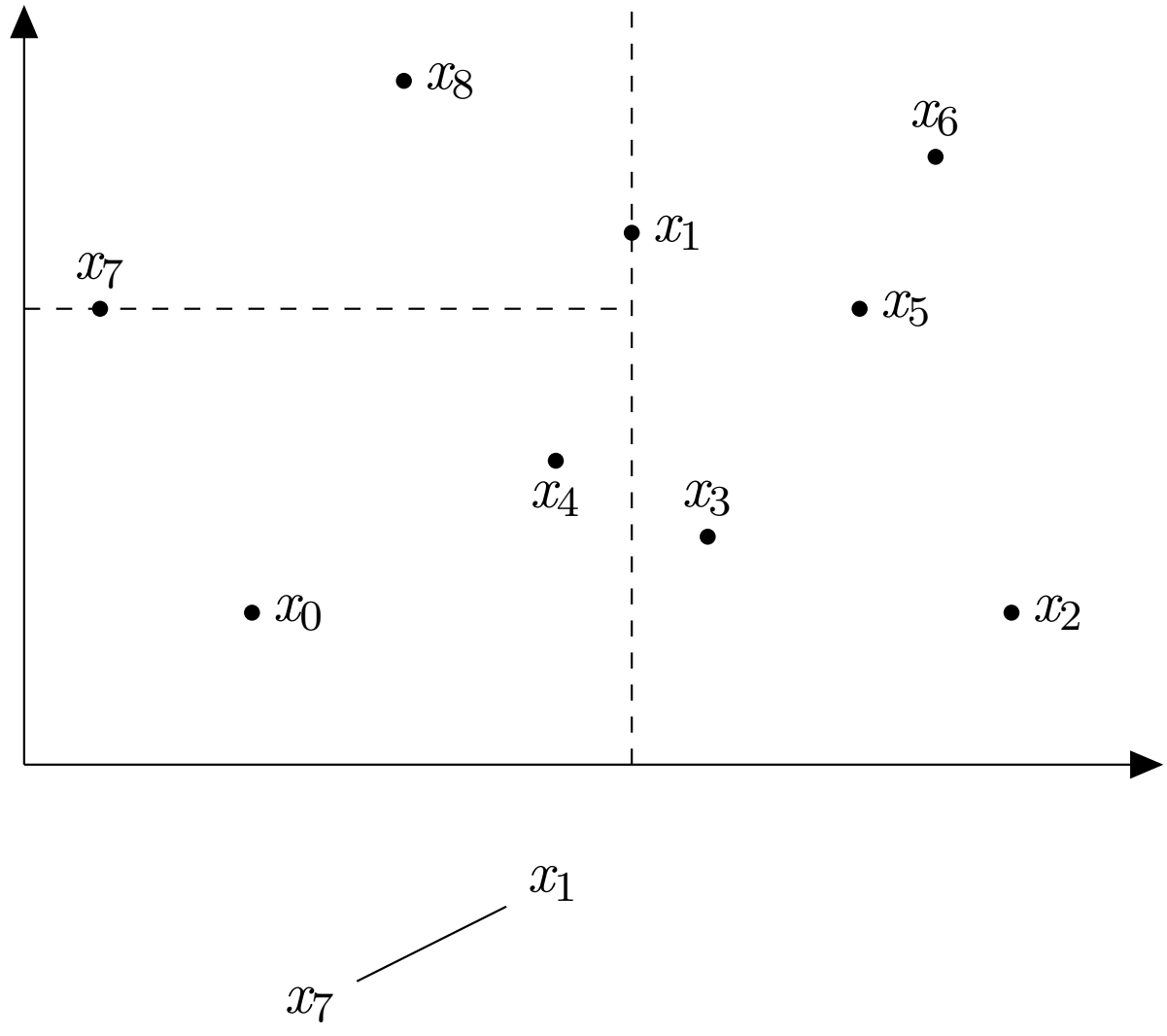
Arbre $k-d$

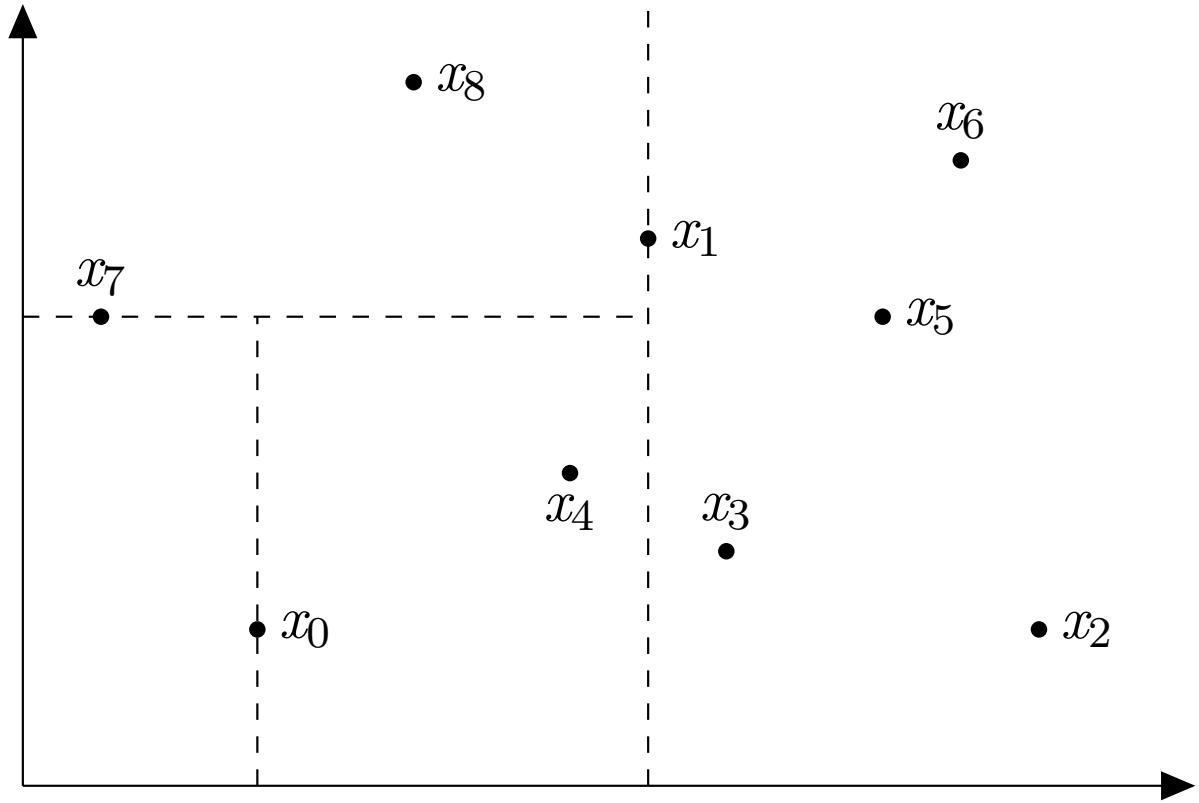
Un arbre $k-d$ est une structure de données permettant de calculer plus rapidement les plus proches voisins parmi des points de $\mathbb{R}^p$.

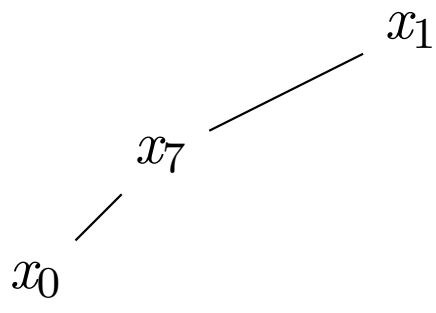
Construction de l'arbre : à la profondeur i, on divise les points de $\mathbb{R}^p$ en deux parties égales (± 1) en prenant comme hyperplan de division l'axe i modulo p.

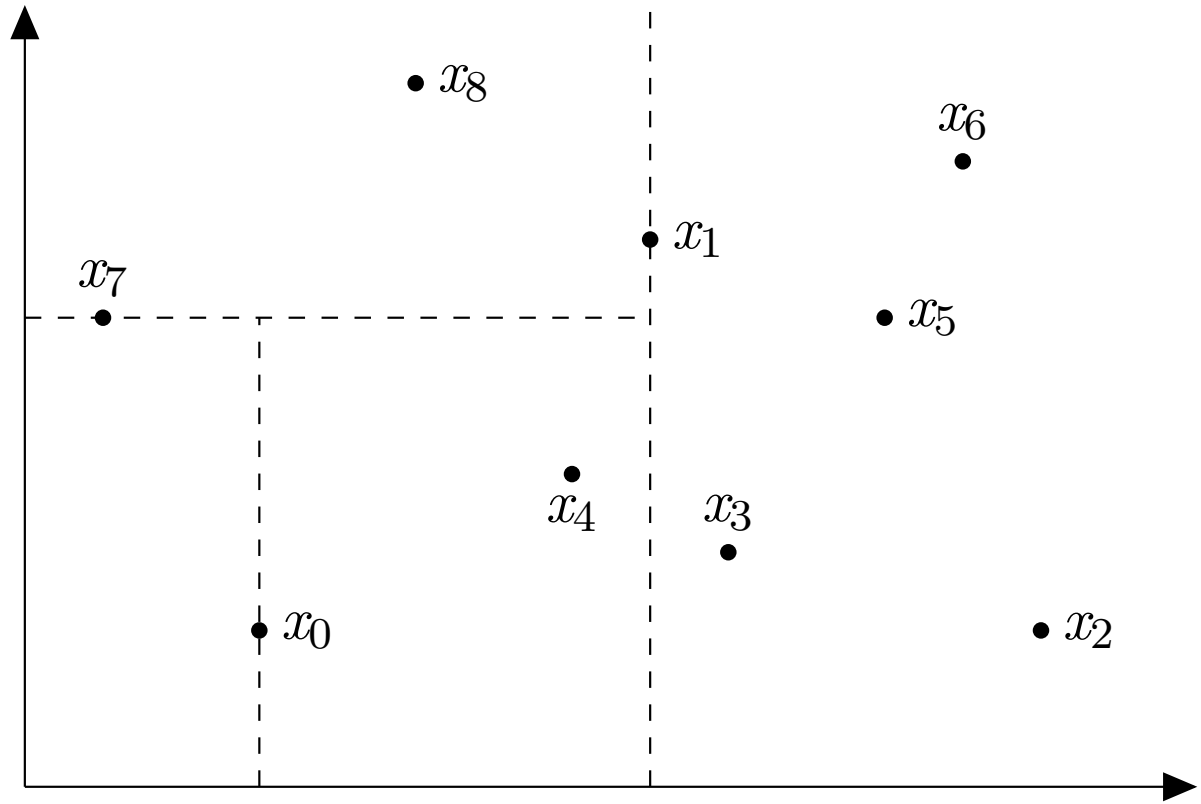

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2

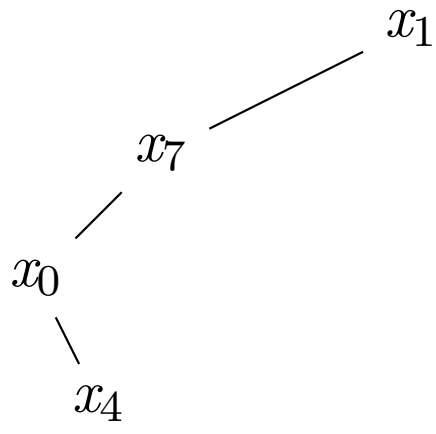

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1

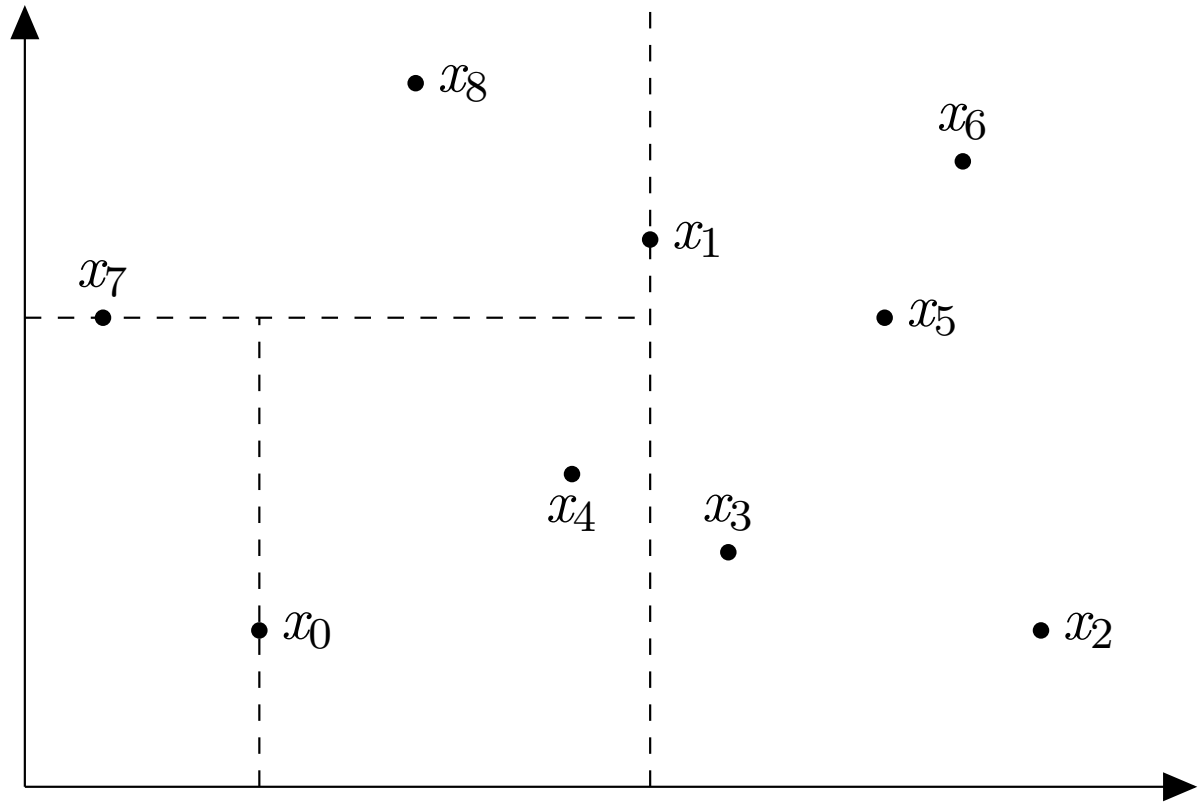

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1
x_7

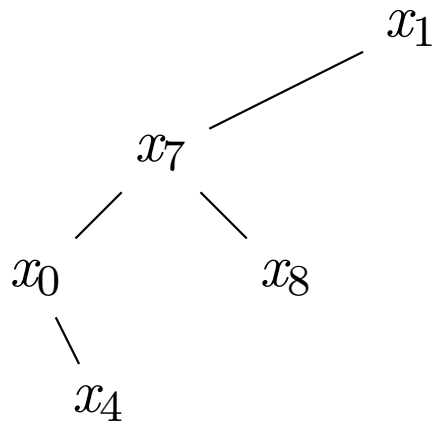

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2

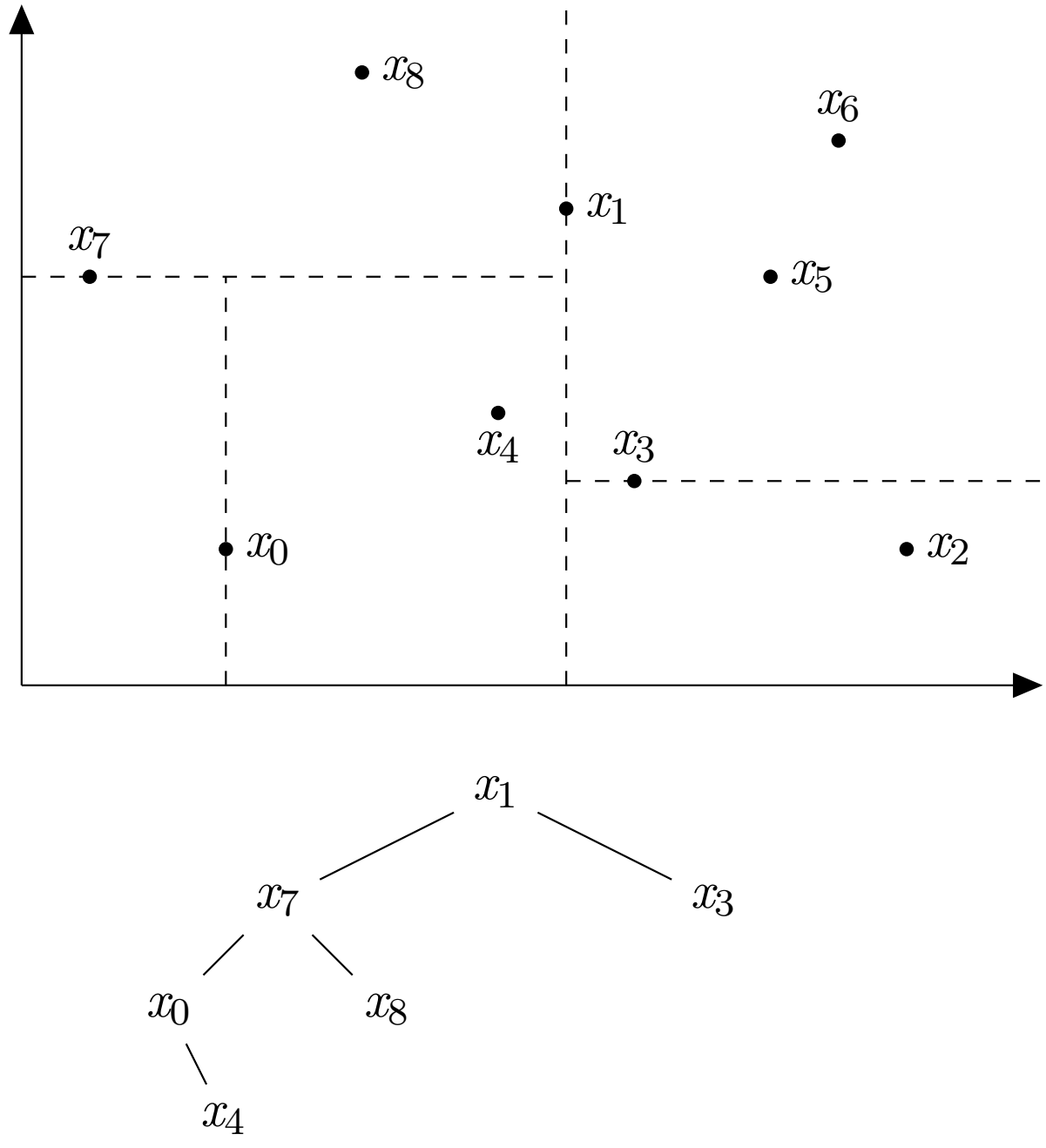

x_1
x_7
x_0

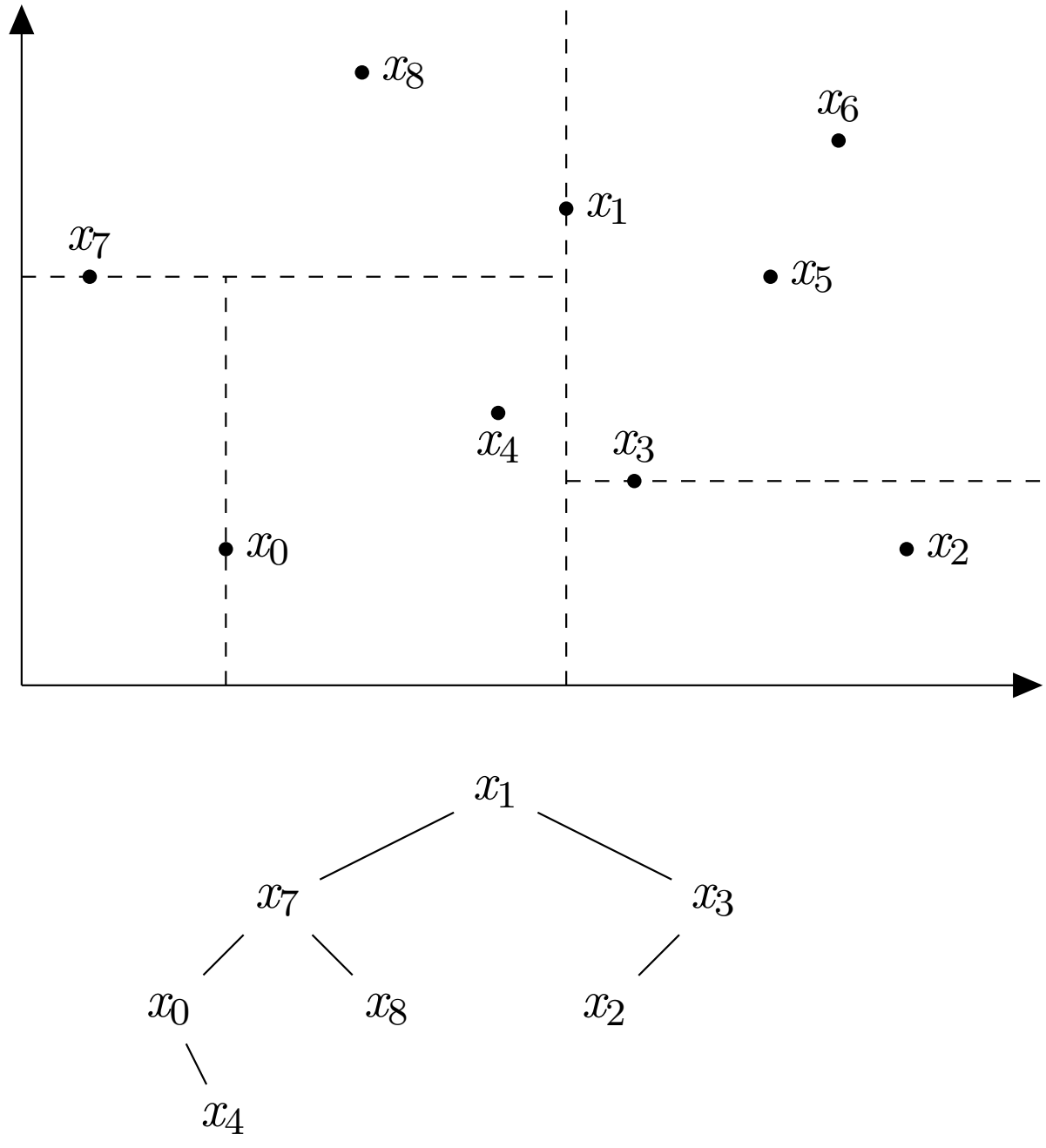

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2

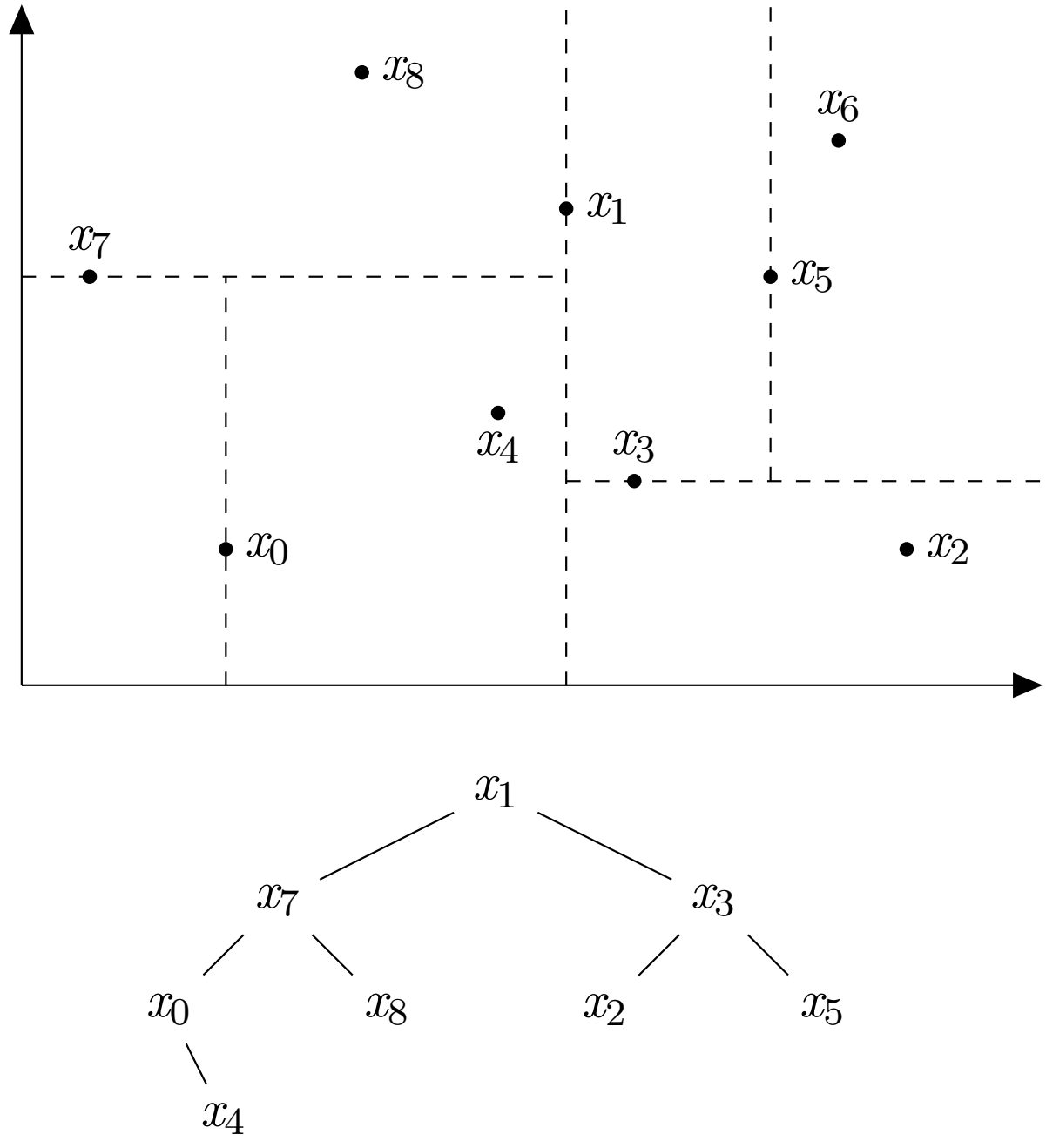

x_1
x_7
x_0
x_4

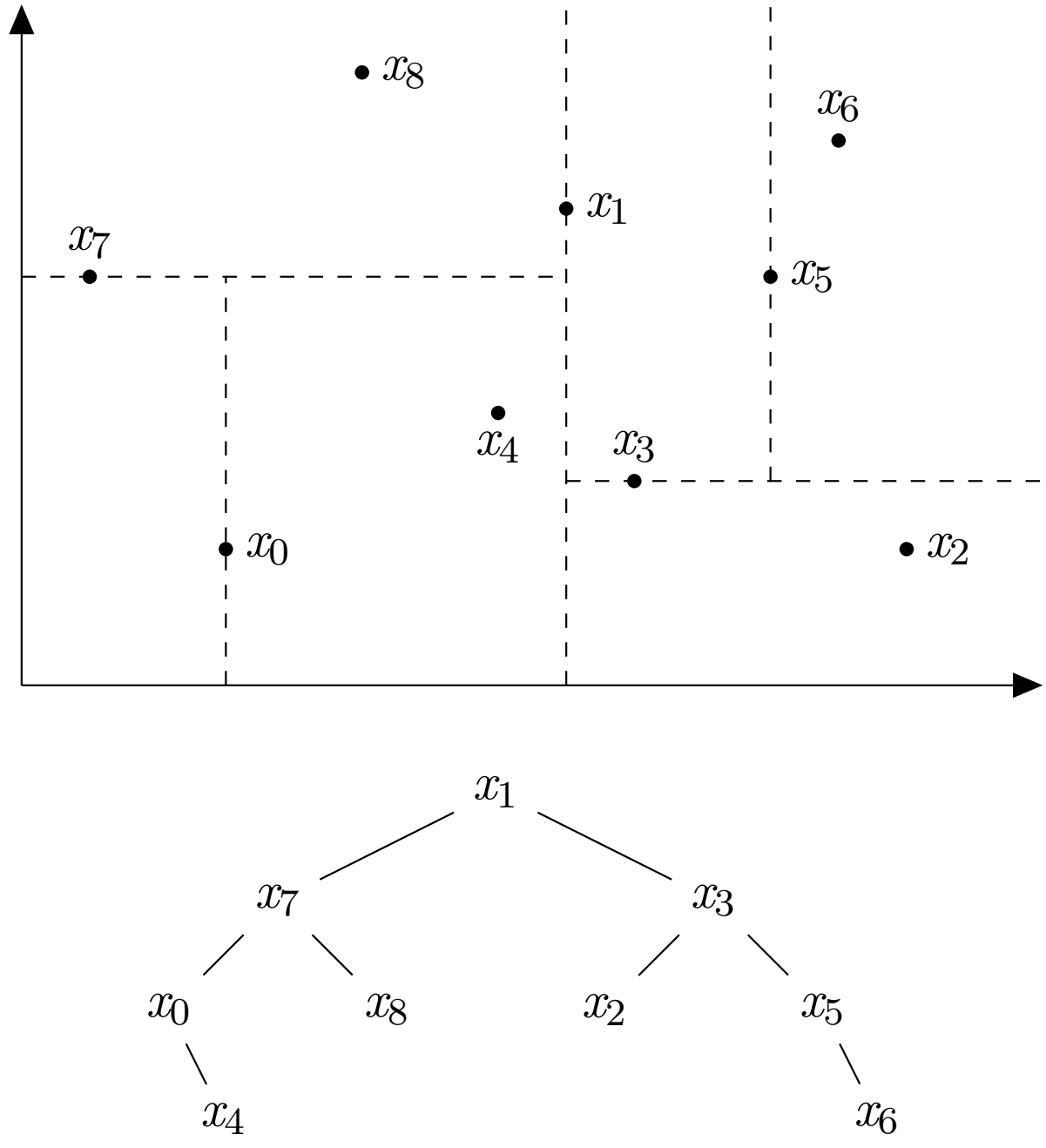

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2


x_1
x_7
x_0
x_8
x_4


x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1
x_7
x_3
x_0
x_8
x_4


x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1
x_7
x_3
x_0
x_8
x_2
x_4

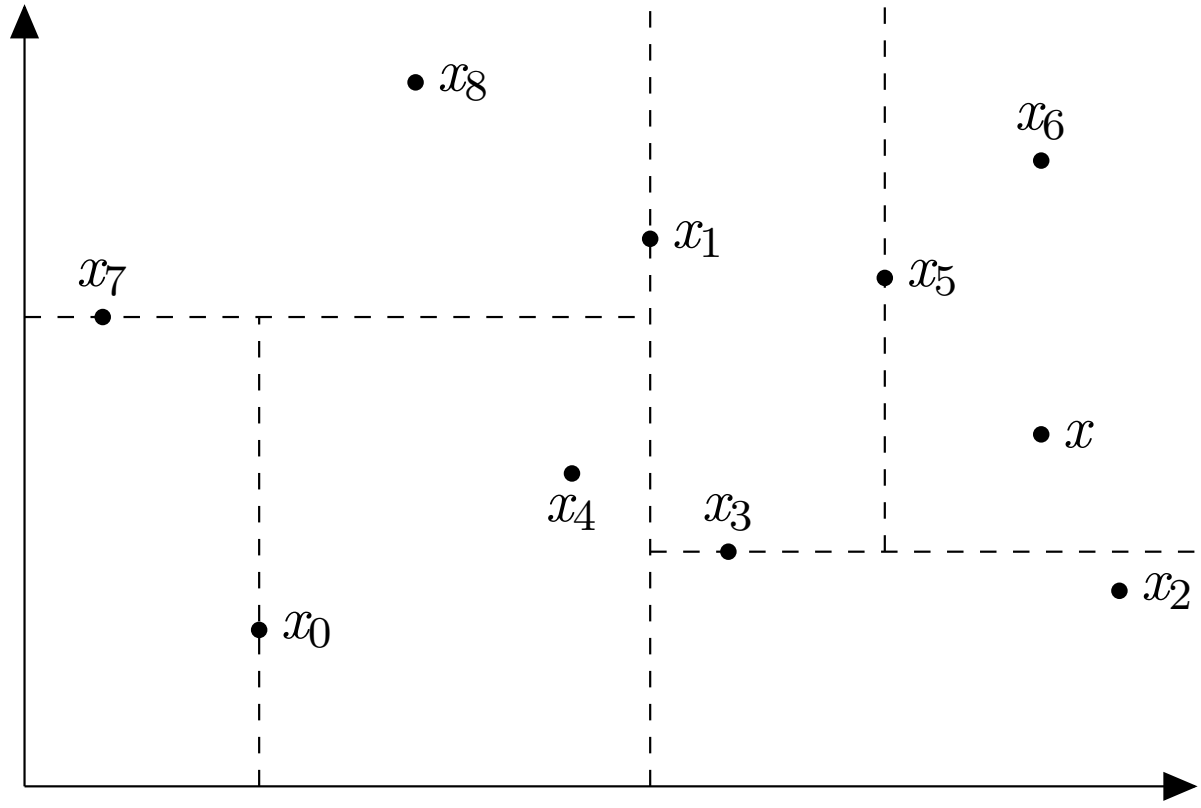

x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1
x_7
x_3
x_0
x_8
x_2
x_5
x_4


x_8
x_6
x_1
x_7
x_5
x_4
x_3
x_0
x_2
x_1
x_7
x_3
x_0
x_8
x_2
x_5
x_4
x_6

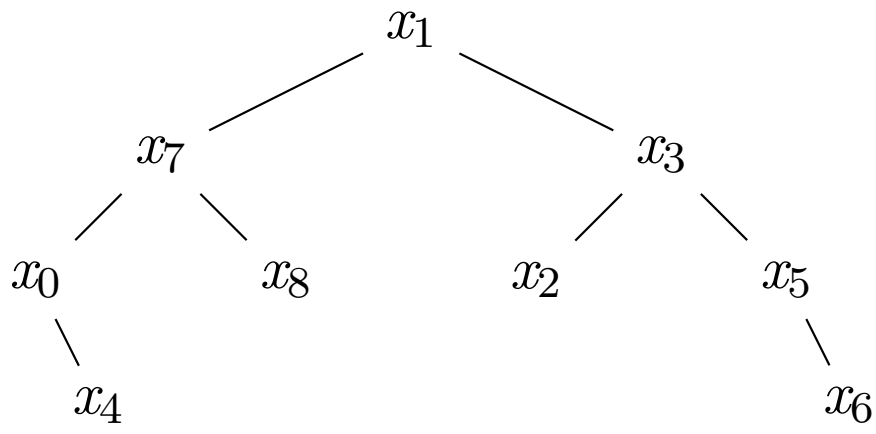
Arbre $k-d$

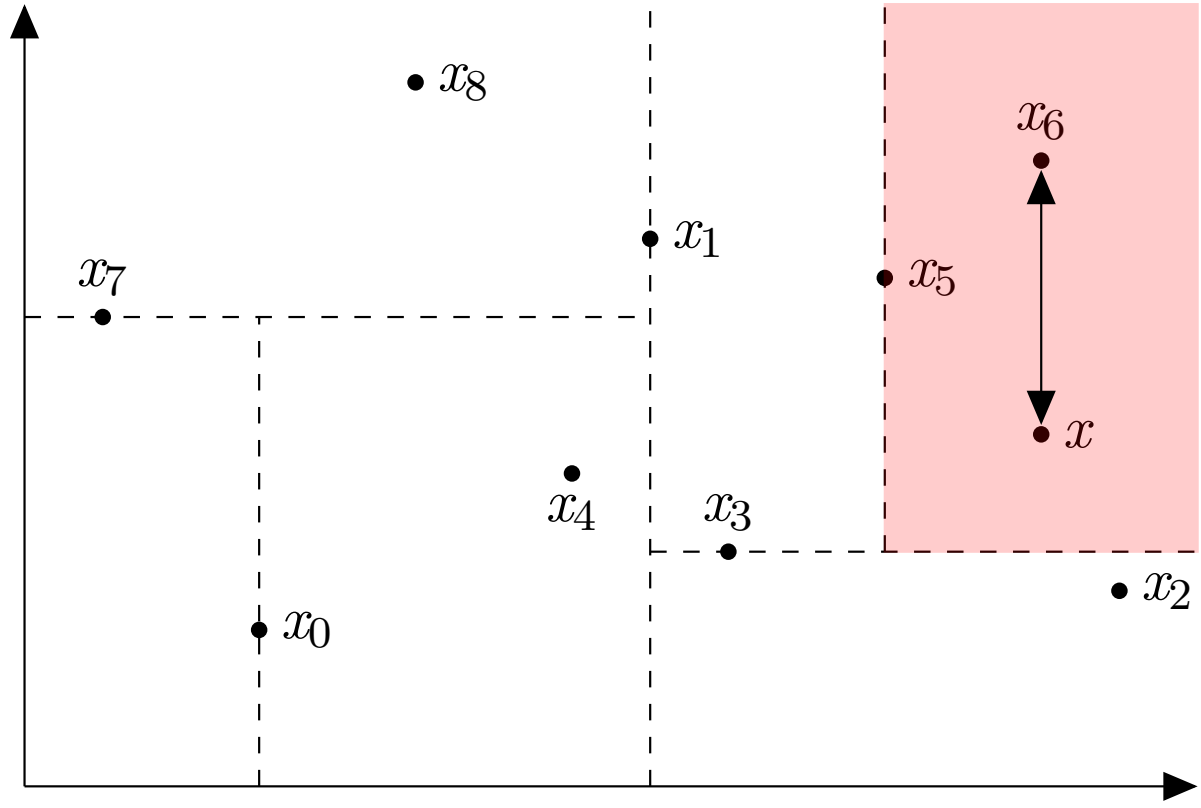
Pour trouver le point le plus proche de $y \in \mathbb{R}^p$:

1. On trouve la feuille de l'arbre correspondant à la zone contenant y.
2. On remonte l'arbre en conservant la distance minimum trouvée et en explorant l'autre sous-arbre si nécessaire.

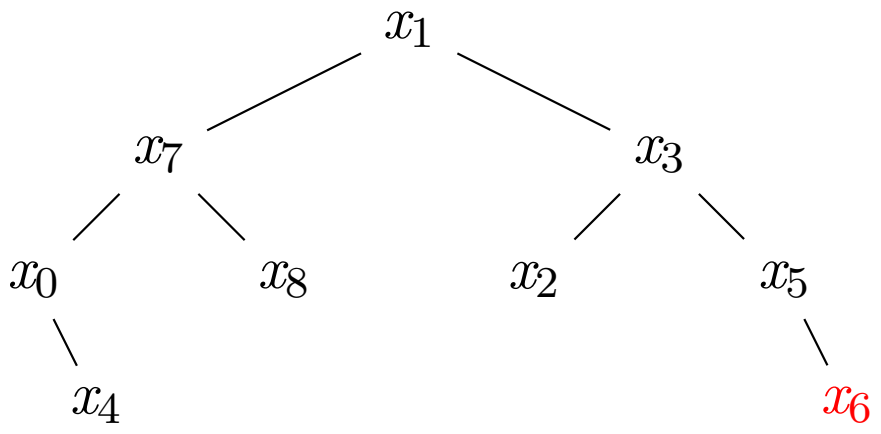


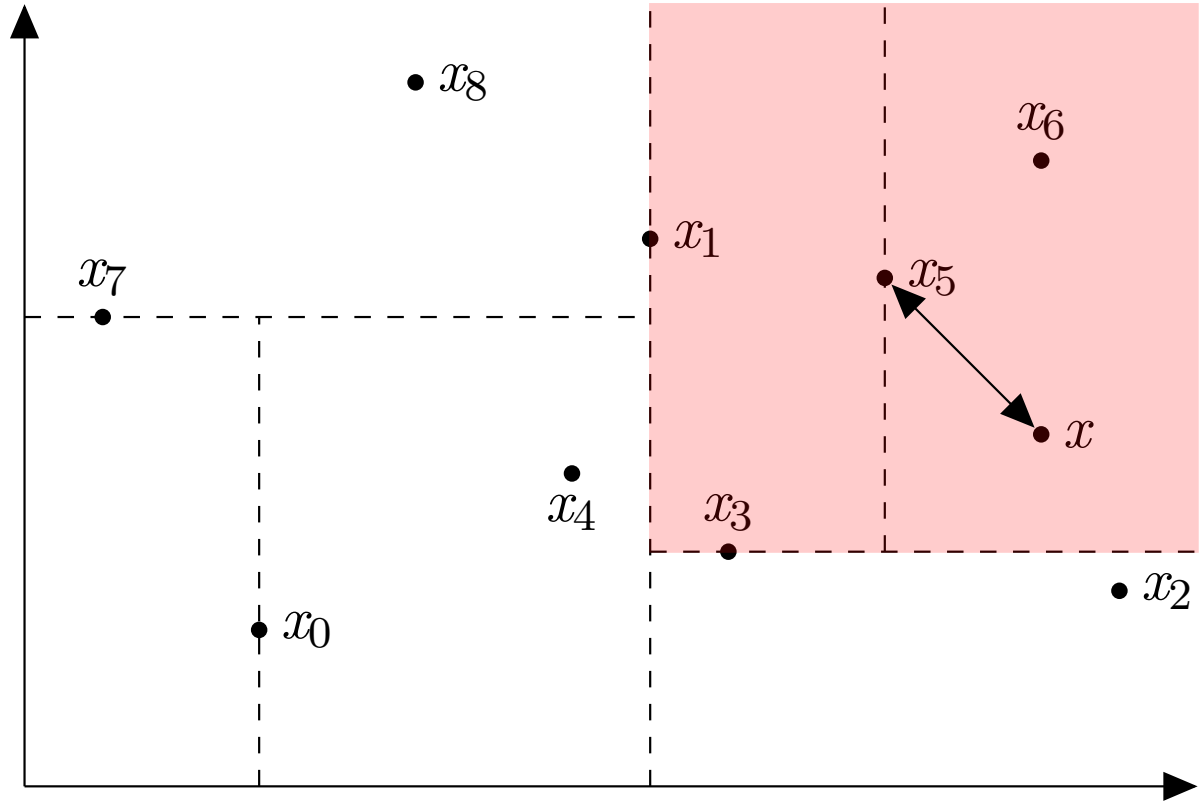
Recherche du plus proche voisin de x



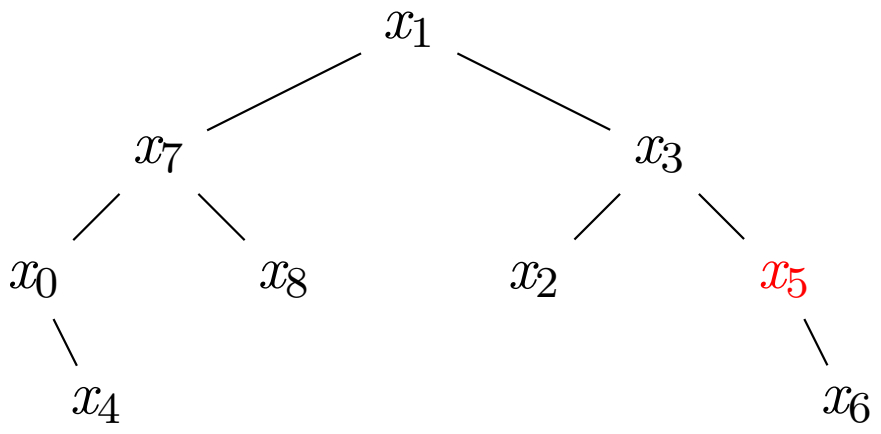


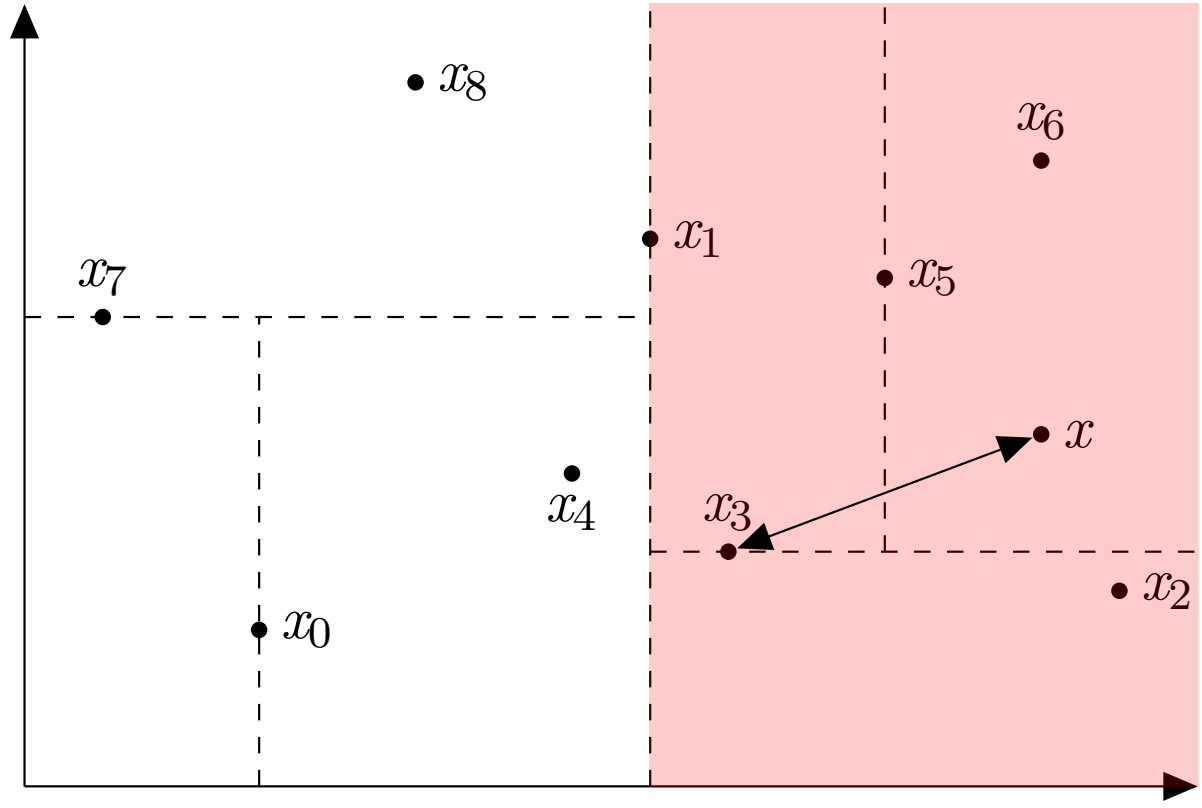
Plus proche voisin de x : x_6



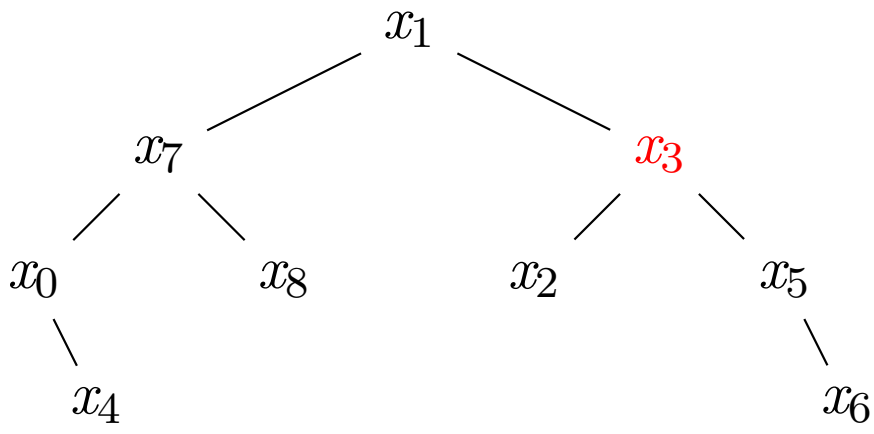


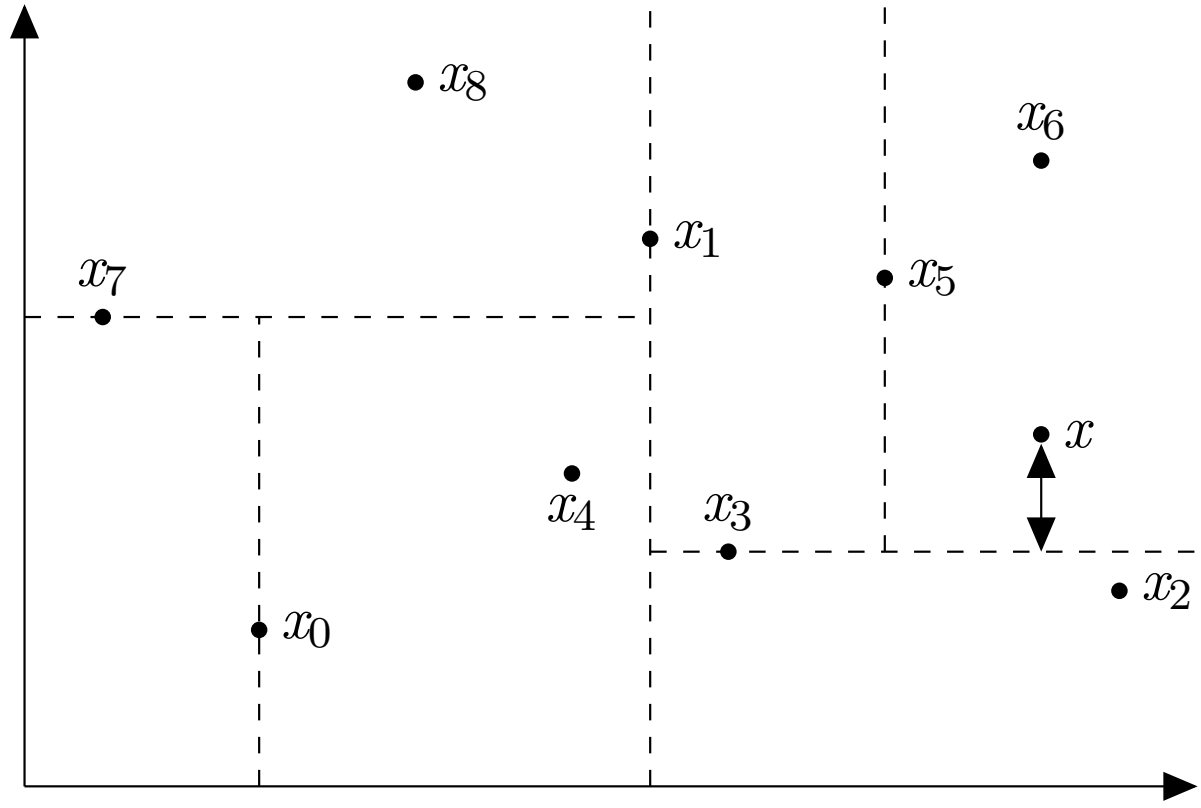
Plus proche voisin de x : x_5



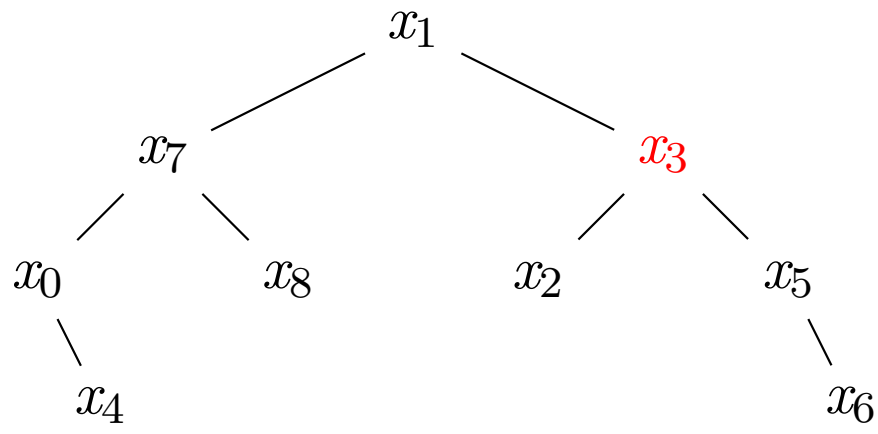


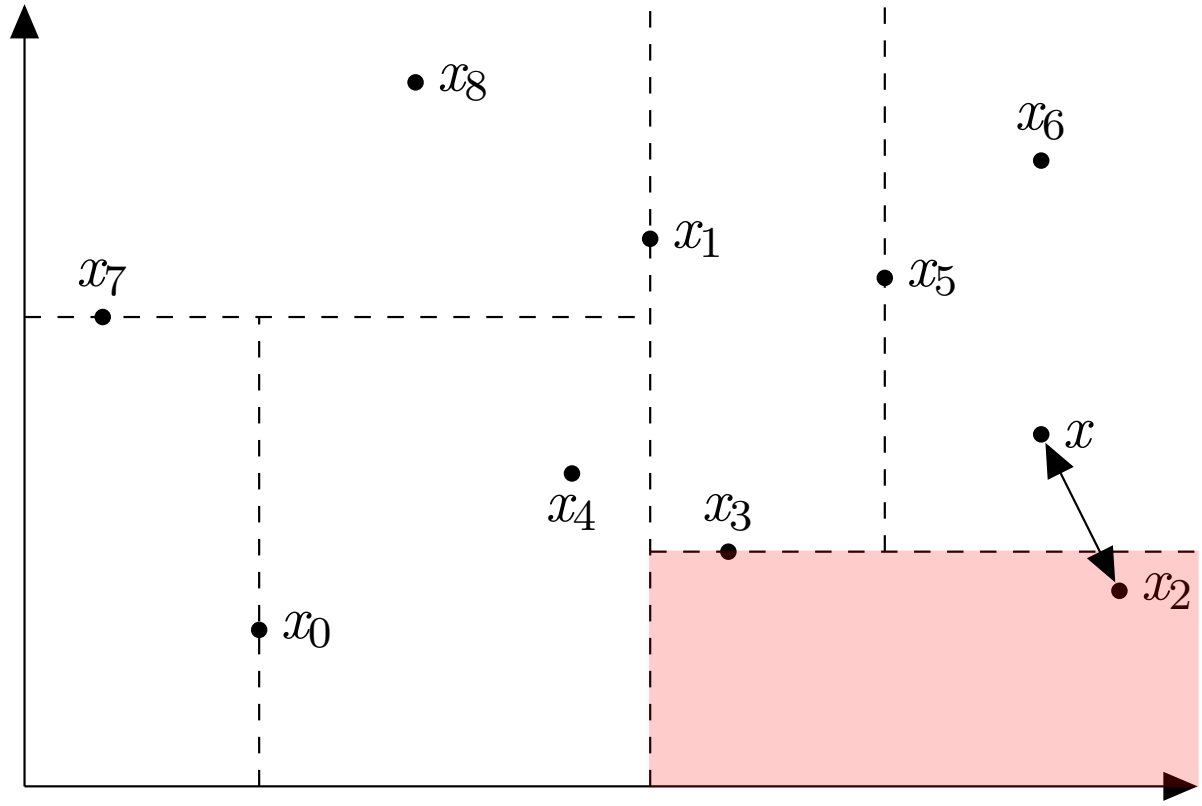
Plus proche voisin de x : x_5



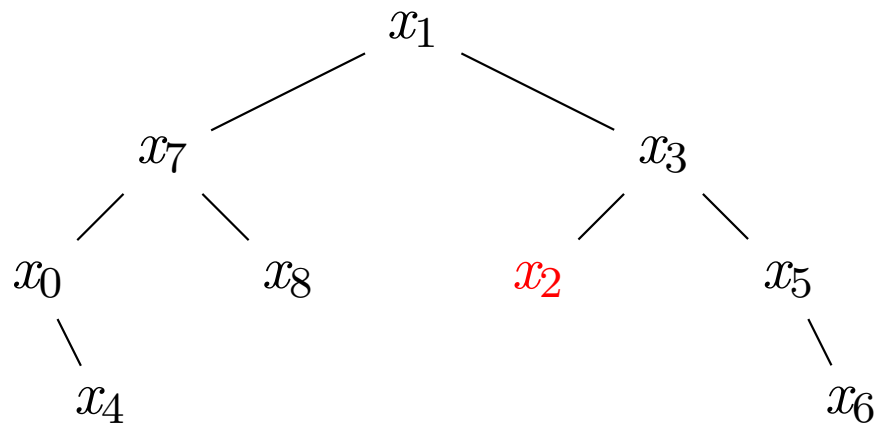


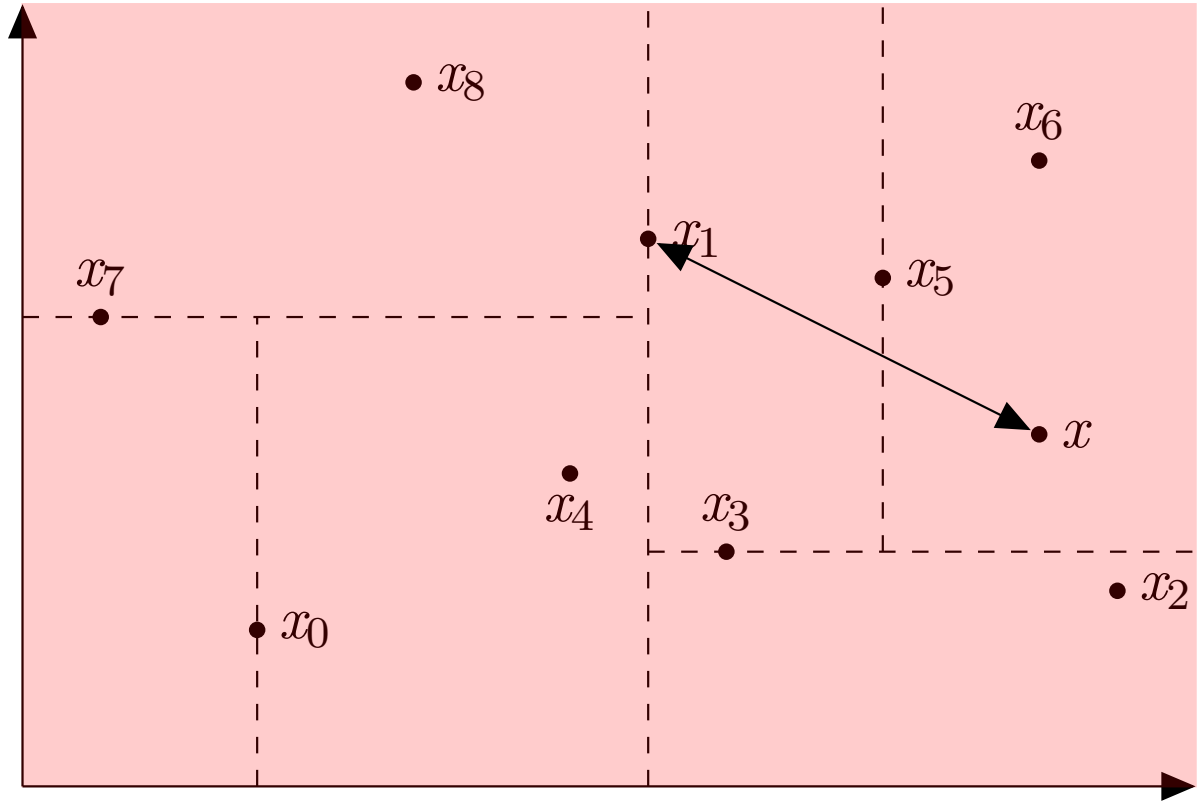
Distance à l'hyperplan inférieure à $d(x, x_6)$



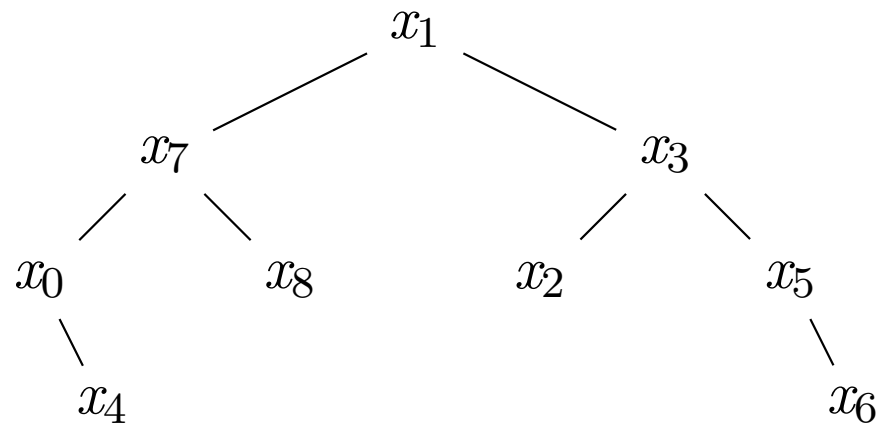


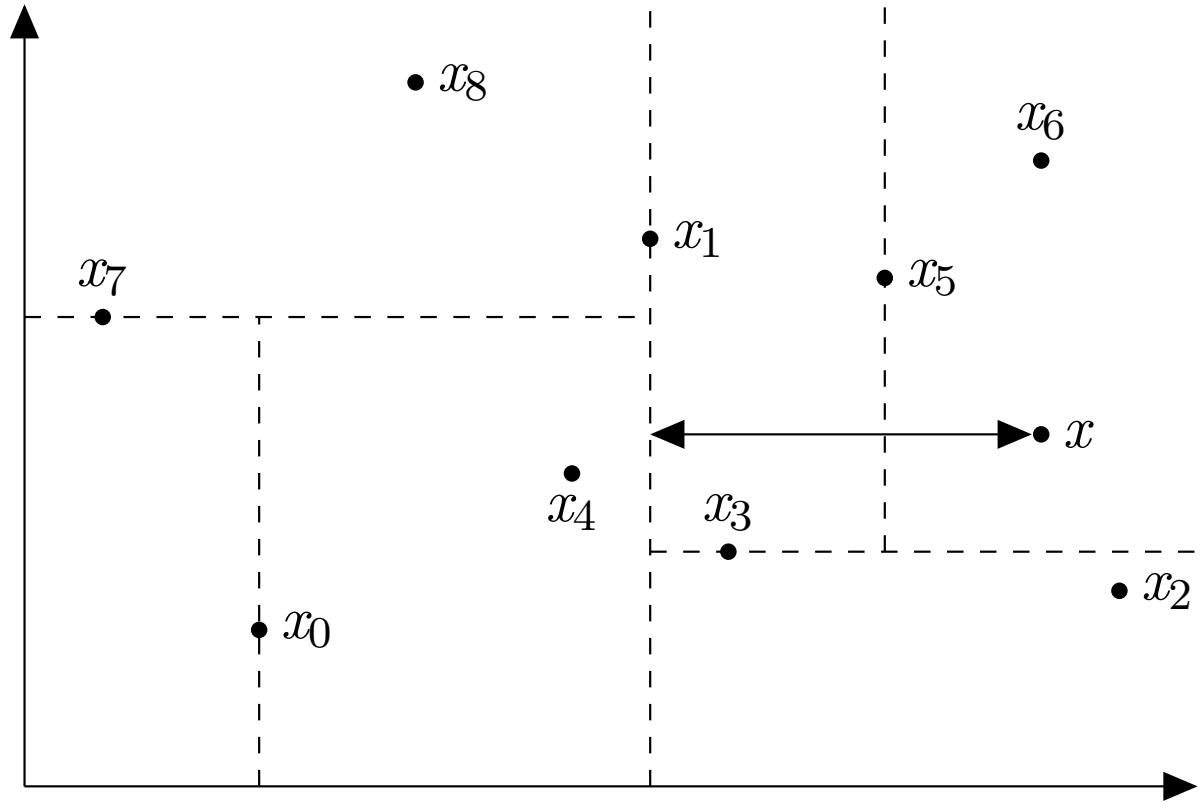
Plus proche voisin de x : x_2



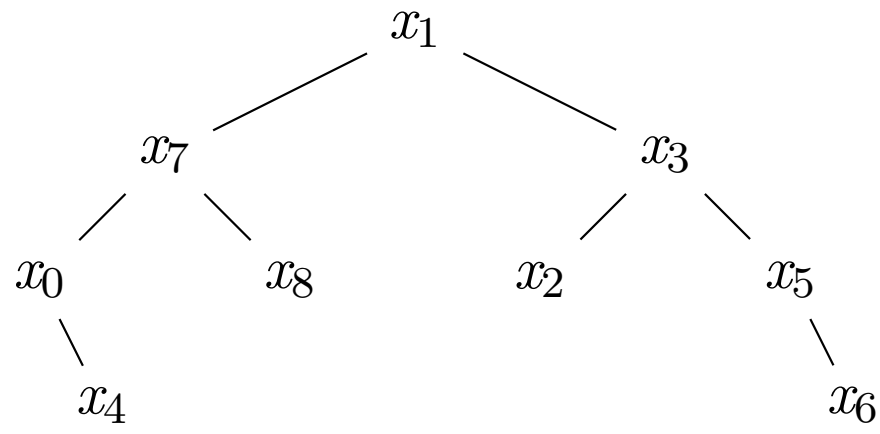


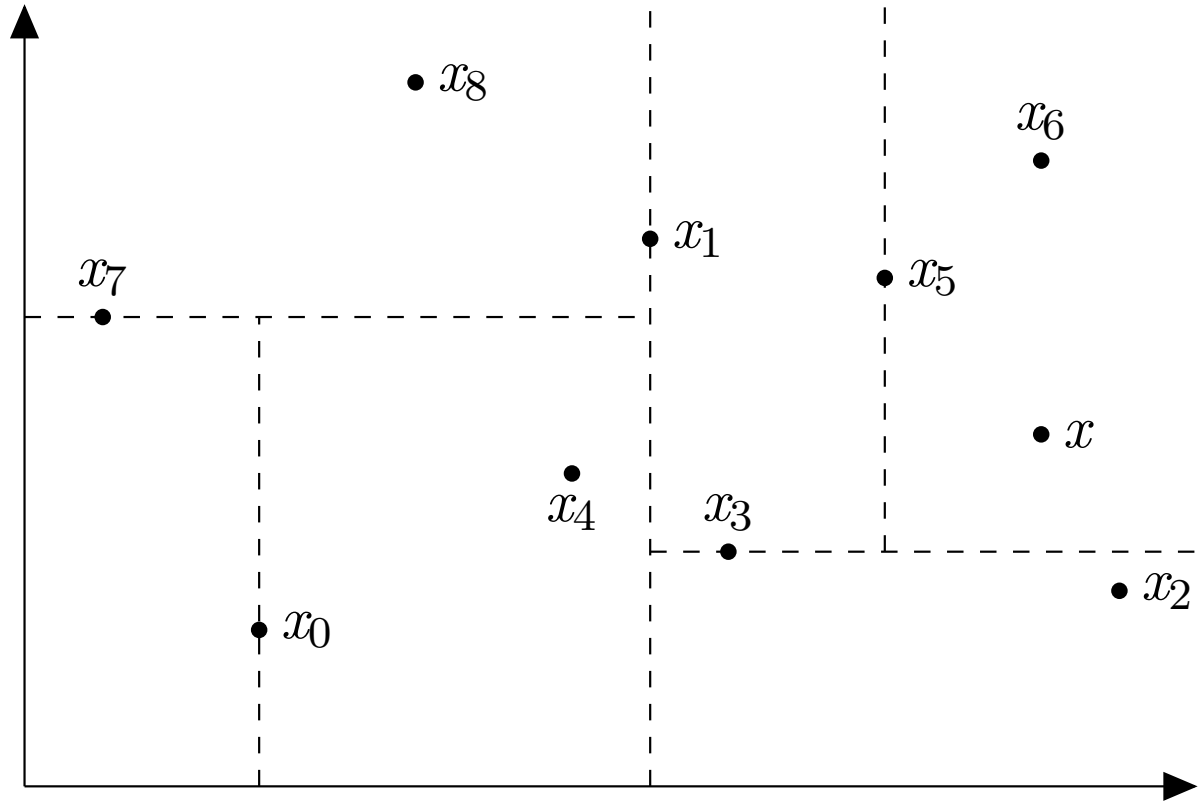
Plus proche voisin de x : x_2



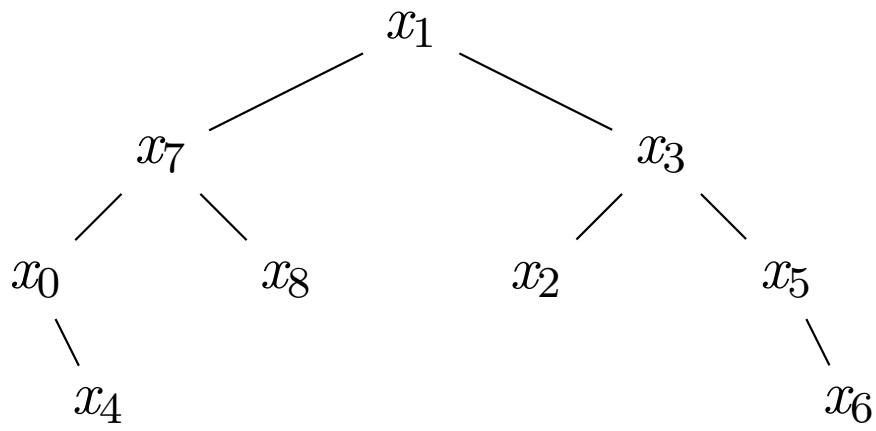


Distance à l'hyperplan supérieure à $d(x, x_2)$





On renvoie x_2



Arbre $k-d$

On peut adapter cette méthode pour trouver les k plus proches voisins parmi n points efficacement et accélerer l'algorithme des k plus proches voisins.

Classification non supervisée

Dans le cas de la classification non supervisée, il n'y a pas de donnée d'entraînement et l'ensemble des classes possibles n'est pas connue à l'avance. On cherche à regrouper les données en fonction de leur proximité.

Algorithme ID3

On suppose dans cette partie que les attributs et les classes sont des booléens (0 ou 1).

Définition

Un arbre de décision est un arbre binaire où :

- Chaque nœud interne est associé à un attribut a et correspond à la question : « a est-il vrai ? ».
- Chaque feuille est associée à une classe.

Algorithme ID3

L'algorithme de classification supervisée ID3 comporte deux étapes :

1. Entraînement : construction d'un arbre de décision.
2. Prédiction : classification d'une nouvelle donnée en parcourant l'arbre.

Algorithme ID3

Définition

Si X est un ensemble de données de classes Y on définit son entropie :

$$H(X) = -\sum_{c \in Y} p(c) \log_2(p(c))$$

où $p(c)$ est la proportion de données de classe c dans X.

L'entropie mesure la dispersion des valeurs : plus $H(X)$ est élevée, plus les données de X sont dispersées dans les classes de Y. Si toutes les données sont dans la même classe, $H(X) = 0$.

Algorithme ID3

Exercice

Une fonction $f : [0, 1] \longrightarrow \mathbb{R}^+$ mesurant la surprise d'un évènement ($f(p)$ étant la surprise de voir un évènement avec probabilité p se réaliser) doit raisonnablement vérifier :

1. f continue
2. $f(1) = 0$ (un évènement certain n'est pas surprenant)
3. Si $p < q$ alors $f(p) > f(q)$ (un évènement rare est plus surprenant qu'un évènement fréquent)
4. $f(pq) = f(p) + f(q)$ (la surprise de deux évènements indépendants est la somme des surprises)
5. $f(1/2) = 1$ (normalisation)

Montrer que $f(p) = -\log_2(p)$.

Algorithme ID3

On met en racine de l'arbre l'attribut qui sépare le mieux les données d'entraînement en deux classes :

Définition

Si a est un attribut de X, on définit :

$$H(x|a) = \sum_{v \in \{0,1\}} p(a = v) H(X|a = v)$$

où $p(a = v)$ est la proportion de données de X pour lesquelles $a = v$ et $X|a = v$ est l'ensemble des données de X pour lesquelles $a = v$.

L'algorithme ID3 choisit alors comme racine l'attribut a qui minimise l'entropie $H(X|a)$ après avoir séparé les données en fonction de a (on veut diminuer l'entropie).

Algorithme ID3

Exemple : On considère un étudiant qui a laissé des avis (positifs ou négatifs) sur un certains nombre de livre d'informatique. Voici les attributs de chaque livre :

- E : Le livre contient des exercices.
- C : le livre contient les corrigés de ces exercices.
- F : le livre est écrit en français.
- O : le livre utilise OCaml.
- Y : l'étudiant a laissé un avis positif.

On souhaite prédire si un nouveau livre sera apprécié ou non.

Algorithme ID3

E	C	F	O	Y
1	1	1	1	1
1	1	1	0	0
1	1	0	1	1
1	1	0	0	0
1	0	1	1	1
1	0	1	0	0
1	0	0	1	0
1	0	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0

Algorithme ID3

On calcule :

$$H(X|E=1) = -\frac{3}{8}\log_2(\frac{3}{8}) - \frac{5}{8}\log_2(\frac{5}{8}) \approx 0.95$$

$$H(X|E=0) = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) \approx 0.81$$

$$H(X|E) = \frac{8}{12}H(X|E=1) + \frac{4}{12}H(X|E=0) \approx 0.9$$

Algorithme ID3

On calcule :

$$H(X|E=1) = -\frac{3}{8}\log_2(\frac{3}{8}) - \frac{5}{8}\log_2(\frac{5}{8}) \approx 0.95$$

$$H(X|E=0) = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) \approx 0.81$$

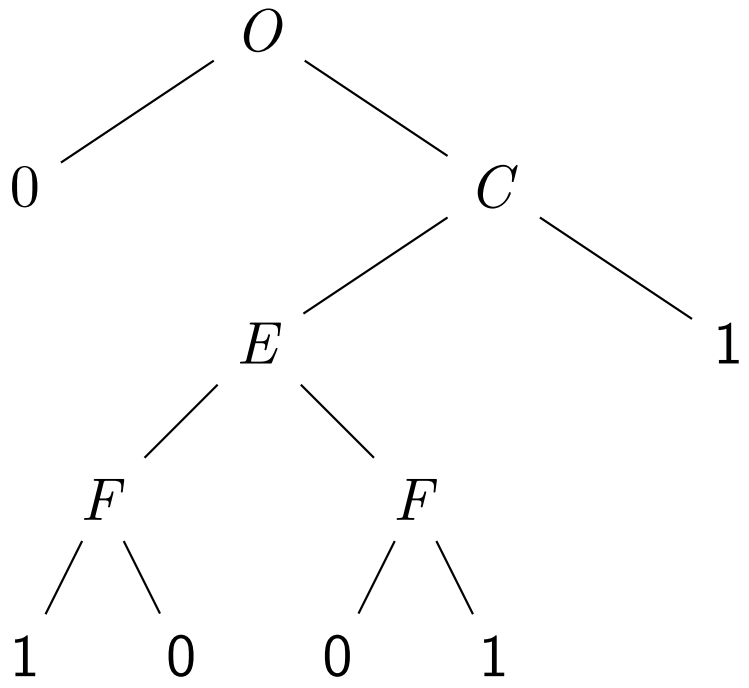
$$H(X|E) = \frac{8}{12}H(X|E=1) + \frac{4}{12}H(X|E=0) \approx 0.9$$

De même, $H(X|C) \approx 0.876$, $H(X|F) \approx 0.92$ et $H(X|O) \approx 0.46$.

On choisit donc O comme attribut de la racine de l'arbre, puis on s'applique récursivement sur $X|O=1$ et $X|O=0$.

Algorithme ID3

On obtient l'arbre de décision suivant (où le fils gauche correspond à 0 et le fils droit à 1) :



Exercice

Selon l'algorithme ID3, quel avis obtiendrait un livre $(E, C, F, O) = (0, 1, 1, 1)$? $(1, 0, 1, 1)$?