Machine Learning avec Python

K Nearest Neighbors

Niveau: IAU: 2025/2026
Manel SEKMA
Ing2_GL

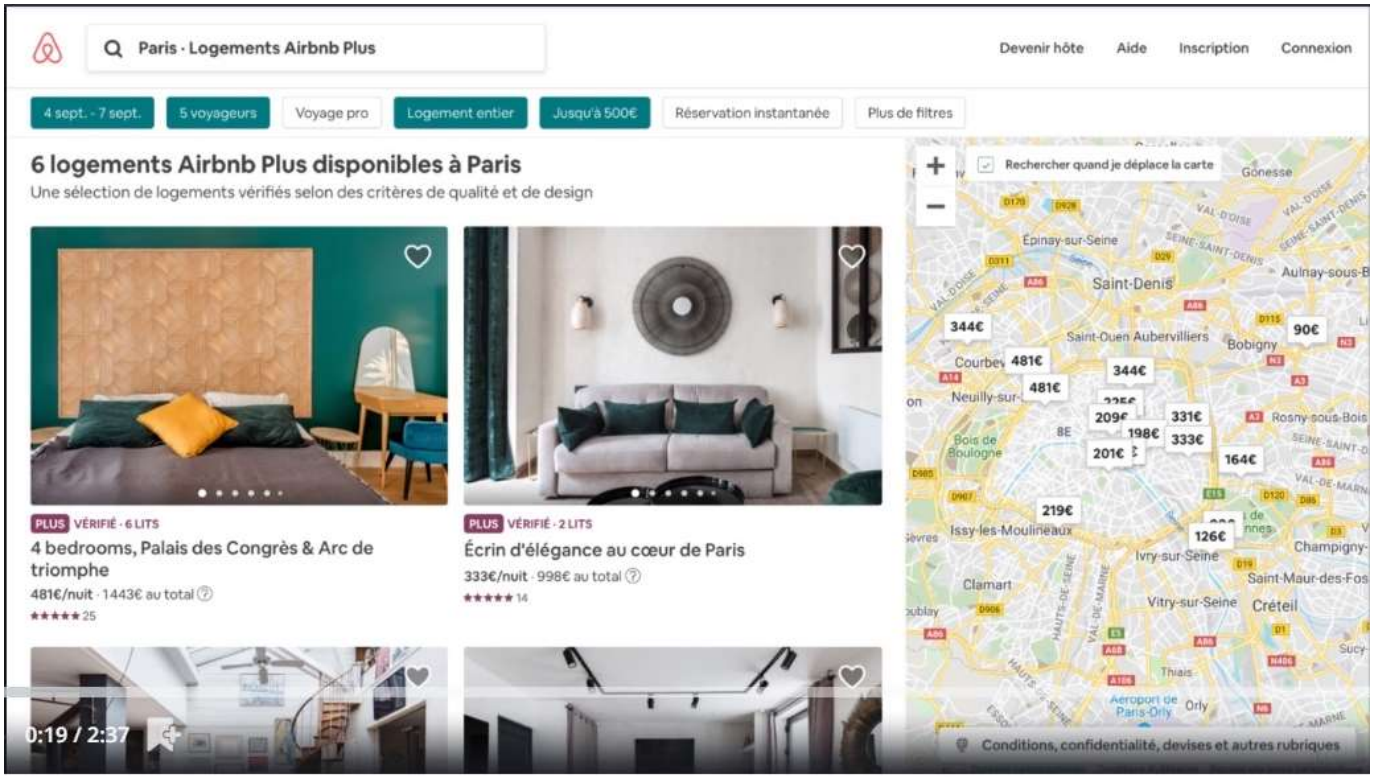
Objectif du cours

Comprendre le fonctionnement de l'algorithme **KNN**, savoir l'appliquer sur un ensemble de données, mesurer la similarité avec des distances adaptées, prendre les précautions d'usage, évaluer ses performances et ajuster ses hyperparamètres.

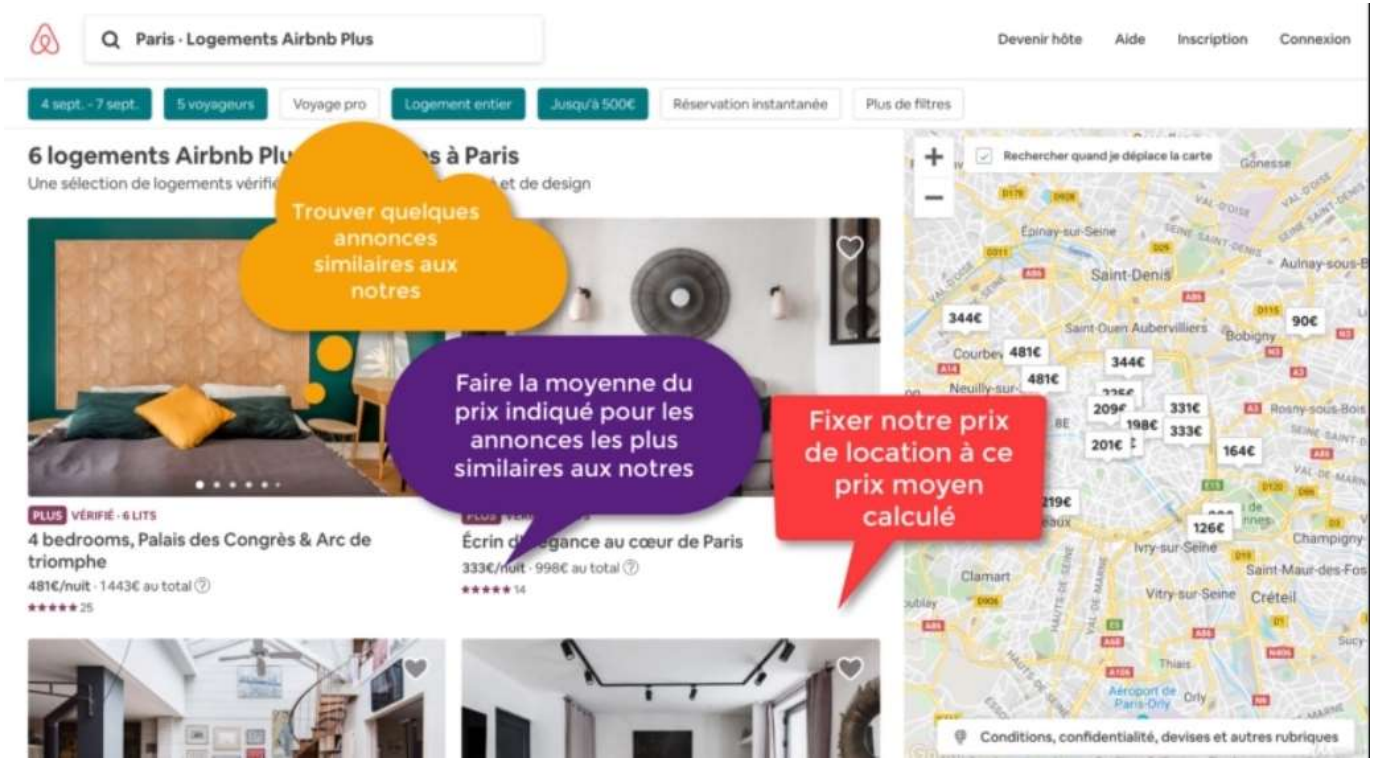
Plan

1. K Nearest Neighbors
    1. Pseudo-code knn
    2. Principe de l'algorithme knn
    3. Étude d'un exemple dataset airbnb
    4. Distance pour calculer la similarité
    5. Précautions
2. Evaluer un modèle
3. Les Hyperparamètres en Apprentissage Automatique

Étude d'un exemple avec dataset airbnb

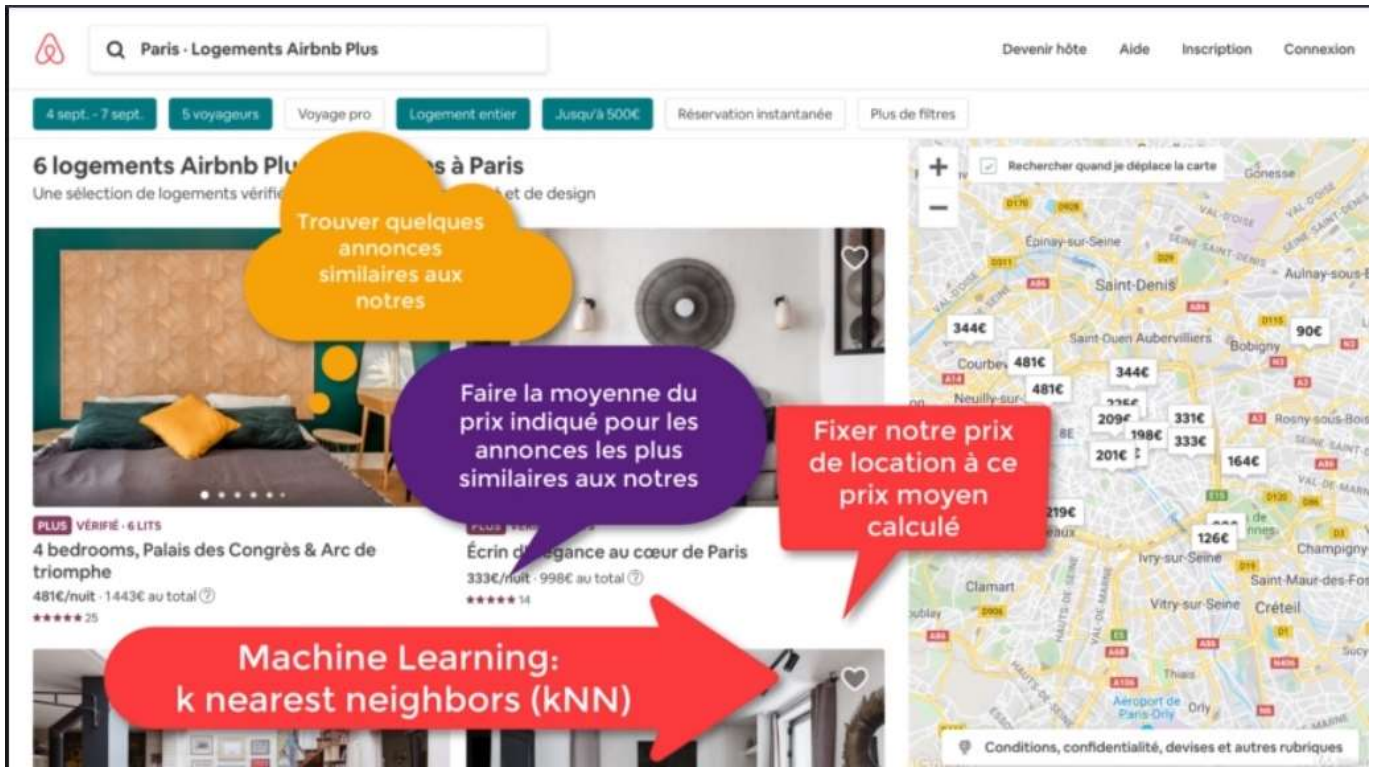


Étude d'un exemple

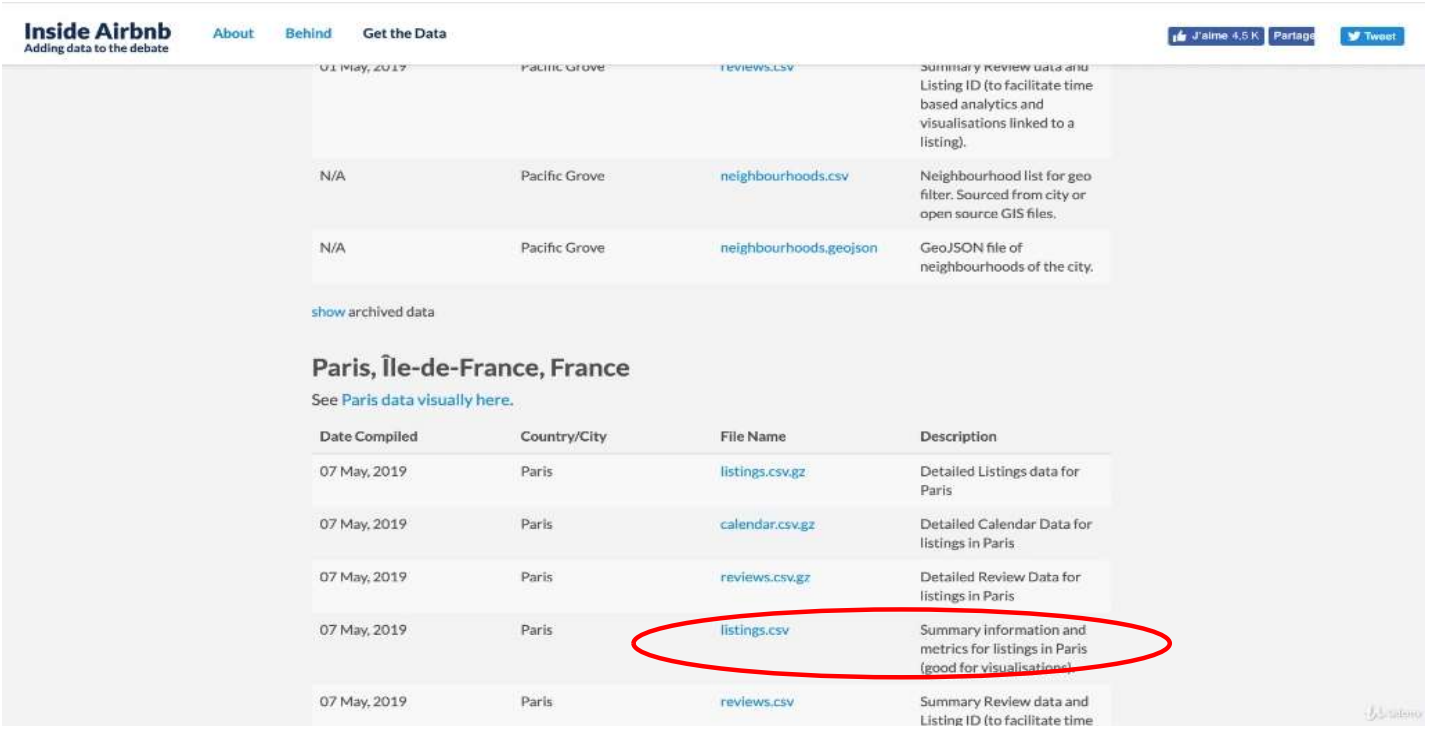


Étude d'un exemple

Définition du problème



Introduction au dataset aibnb



Inside Airbnb
Adding data to the debate

About Behind Get the Data

J'aime 4,5 K Partage Tweet

01 May, 2019	Pacific Grove	reviews.csv	Summary Review data and Listing ID (to facilitate time based analytics and visualisations linked to a listing).
N/A	Pacific Grove	neighbourhoods.csv	Neighbourhood list for geo filter. Sourced from city or open source GIS files.
N/A	Pacific Grove	neighbourhoods.geojson	GeoJSON file of neighbourhoods of the city.

show archived data

Paris, Île-de-France, France

See Paris data visually here.

Date Compiled	Country/City	File Name	Description
07 May, 2019	Paris	listings.csv.gz	Detailed Listings data for Paris
07 May, 2019	Paris	calendar.csv.gz	Detailed Calendar Data for listings in Paris
07 May, 2019	Paris	reviews.csv.gz	Detailed Review Data for listings in Paris
07 May, 2019	Paris	listings.csv	Summary information and metrics for listings in Paris (good for visualisations).
07 May, 2019	Paris	reviews.csv	Summary Review data and Listing ID (to facilitate time

Lien vers airbnb:

http://insideairbnb.com/get-the-data.html

Introduction au dataset aibnb

- host_response_rate: le taux de réponse de l'hôte
- host_acceptance_rate: nombre de requêtes/demandes à l'hôte qui convertissent
- host_listings_count: nombre d'autres logements de l'hôte
- latitude: latitude (coordonnée du logement)
- longitude: longitude
- city: la ville de localisation du logement
- zipcode: le code postal du logement
- state: la région du logement
- accommodates: le nombre d'invités que peut accueillir le logement
- room_type: le type de logement (Private room, Shared room ou Entire home/apt)
- bedrooms: nombre de chambres inclus dans le logement
- bathrooms: nombre de salles de bain inclus dans le logement

Plan

1. K Nearest Neighbors
    1. Pseudo-code knn
    2. Principe de l'algorithme knn
    3. Étude d'un exemple dataset airbnb
    4. Distance pour calculer la similarité
    5. Précautions
2. Evaluer un modèle
3. Les Hyperparamètres en Apprentissage Automatique

Introduction de knn

- C'est un algorithme d'apprentissage supervisé , non paramétrique,
- Son fonctionnement peut être assimilé à l'analogie suivante *"dis moi qui sont tes voisins, je te dirais qui tu es…"*.
- Il peut être utilisé aussi bien pour la régression que pour la classification

Pseudo-code knn

Début Algorithme

Données en entrée :

un ensemble de données D

une fonction de définition distance d

Un nombre entier K

Pour une nouvelle observation X dont on veut prédire sa variable de sortie y Faire :

1 Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D

2 Retenir les observations du jeu de données les proches de X en utilisation la fonction de calcul de distance d

3 Prendre les valeurs de y des K observations retenues :

Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenues

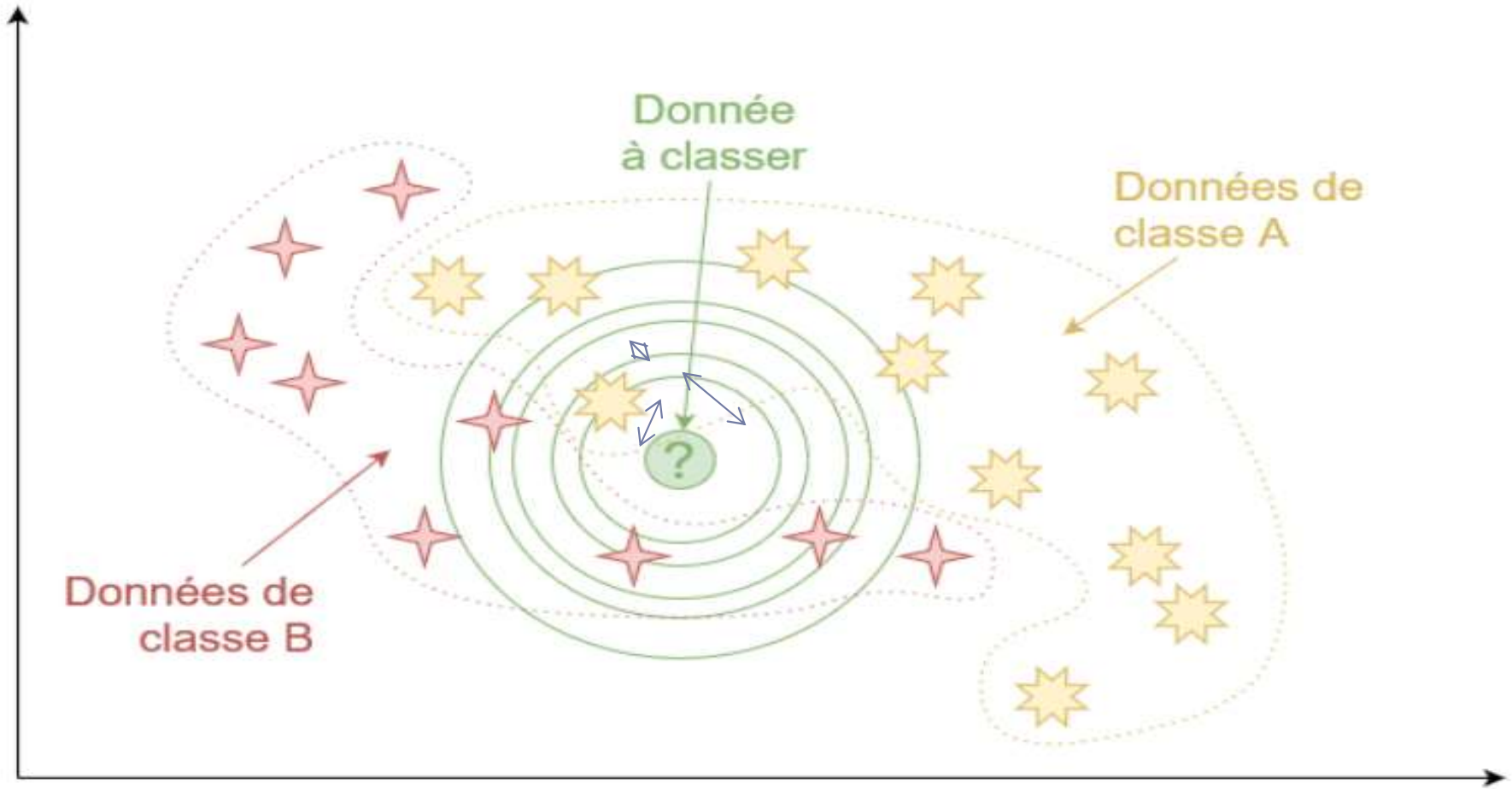
Si on effectue une classification , calculer le mode de y retenues

4 Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X

Fin Algorithme

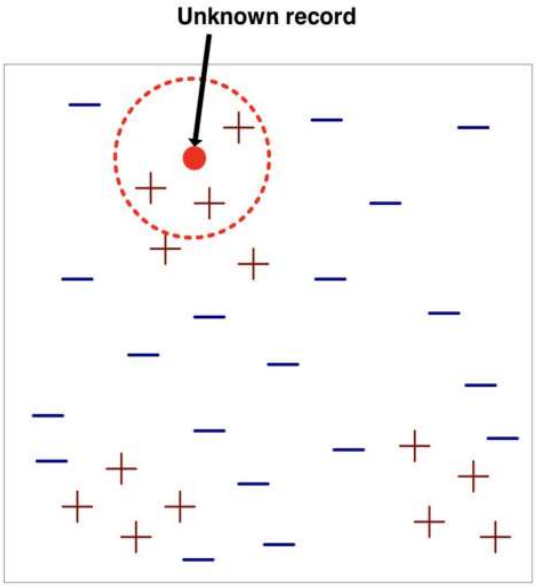
Principe de l'algorithme knn

Soit un entier k inférieur ou égal à n.



k plus proches voisins - k Nearest Neigbors (kNN)

Nearest-Neighbor Classifiers



On a besoin de trois choses :

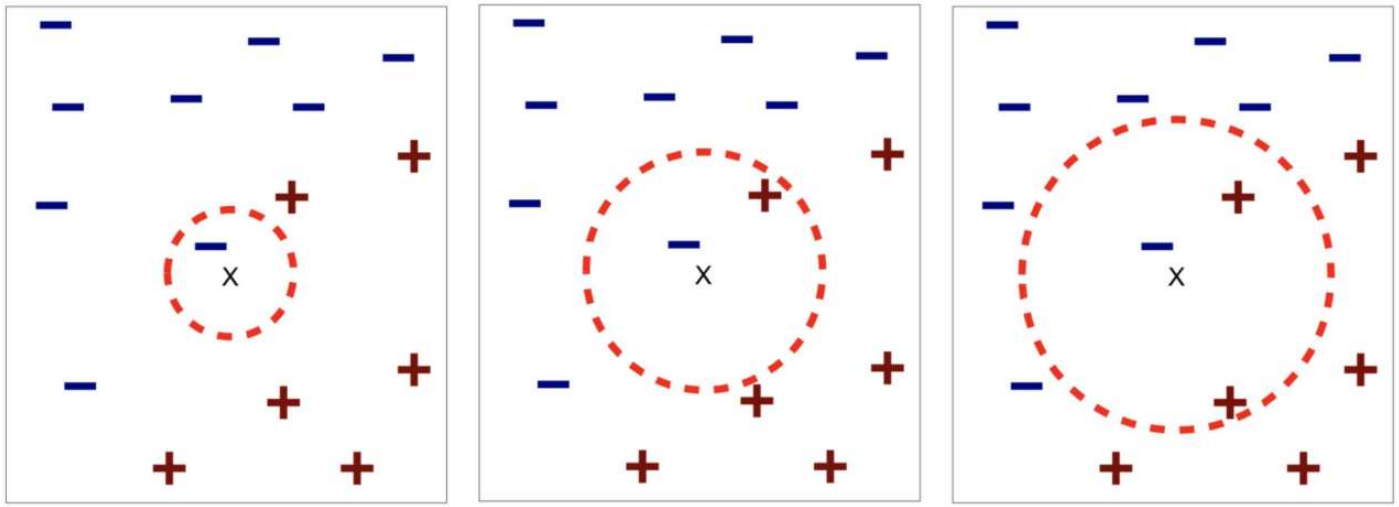
- Un ensemble d'entrainement.
- Une mesure de distance.
- La valeur de k, le nombre de voisins à interroger.

Pour classifier un nouvel enregistrement :

- Calculer la distance vers les autres enregistrements (de l'ensemble d'entraînement).
- Identifier k plus proches voisins.
- Utiliser la classe des k voisins les plus proches pour déterminer la classe du nouvel enregistrement (par un vote majoritaire par exemple).

k plus proches voisins - k Nearest Neigbors (kNN)

Nearest-Neighbor ?



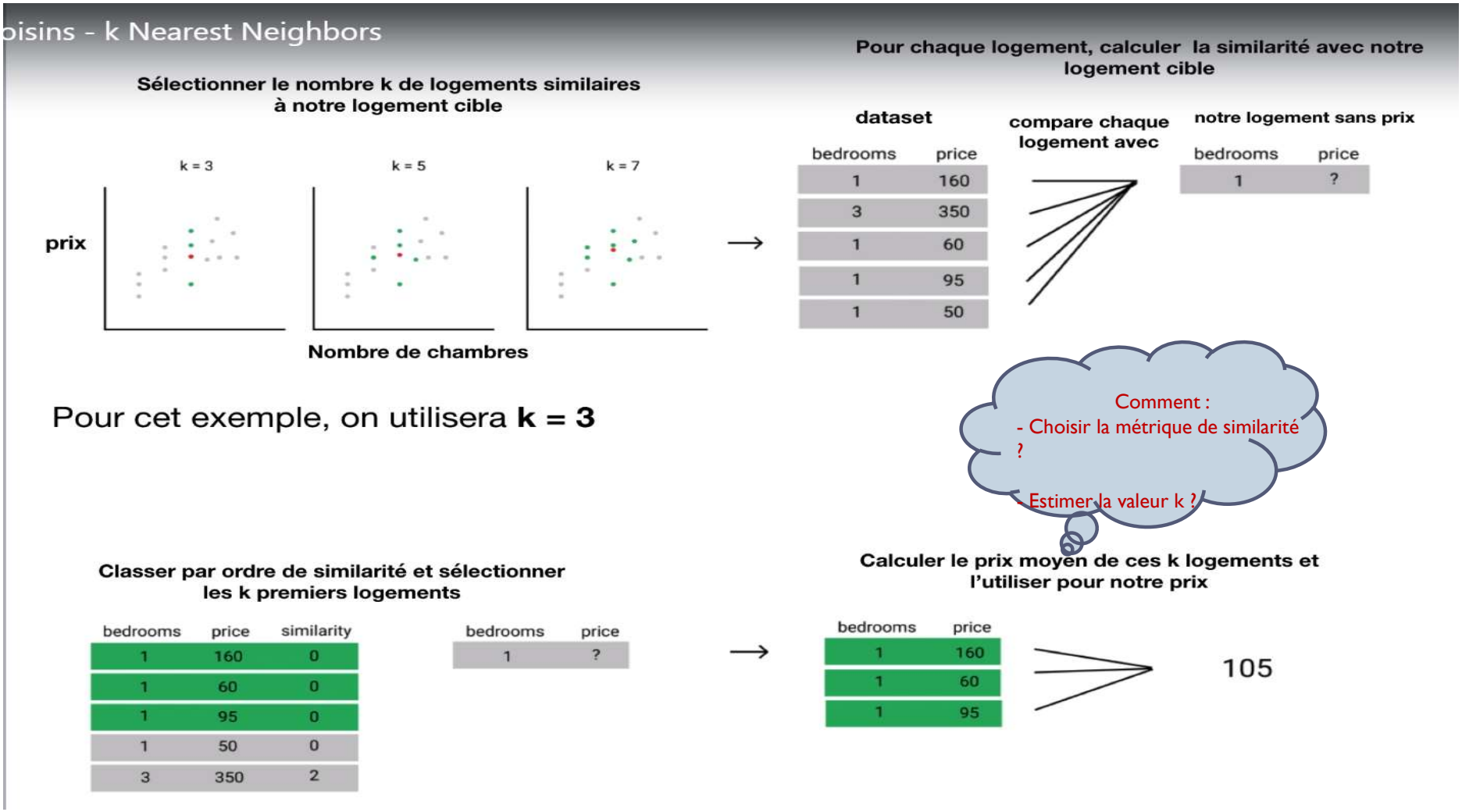
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Calcule des k plus proches voisins - k Nearest Neighbors

Stratégie

- Trouver quelques logements similaires
- Calculer le prix moyen par nuit de ces logements
- Définir ce prix pour notre logement

Calculer les k plus proches voisins



Choix de valeur de k

Choix de le valeur de *k* :

- Si *k* est trop petit, la classification sera trop sensible au "bruit".
- Si *k* est trop grand, le voisinage peut contenir des éléments d'autres classes.

K plus proches voisins--précautions

Quelques précautions à prendre :

- Les attributs doivent être normalisés pour éviter que les distances soient faussées par des attributs à grande valeur.
- Exemple : Taille (H), Poids (W)
  avec :
  - $H \in [1.5m, 1.8m]$
  - $W \in [60kg, 100kg]$

K plus proches voisins--précautions

Quelques précautions à prendre :

Attention à la distance euclidienne ...

- Vecteurs de features à grande dimension

  $\rightarrow$ presque tous les vecteurs sont à la même distance de l'exemple qu'on veut classifier.

- Solution : réduire la dimension des vecteurs (ACP par exemple)

Plan

1. Introduction
2. Bibliothèques en python pour l'analyse de données
3. K Nearest Neighbors
    1. Pseudo-code knn
    2. Principe de l'algorithme knn
    3. Distance pour calculer la similarité
    4. Précautions
4. Evaluer un modèle
5. Les Hyperparamètres en Apprentissage Automatique

Evaluer un modèle

MAE: Mean Absolute Error

- L'erreur absolue moyenne (Mean Absolute Error ou MAE) est la moyenne de toutes les erreurs de prédiction absolues, où l'erreur de prédiction est la différence entre la valeur réelle et la valeur prédite

- L'utilisation de la valeur absolue des erreurs de prédiction empêche l'annulation mutuelle des erreurs.

- La valeur MAE est exprimée dans la même unité que celle de la valeur cible.

Evaluer un modèle

MAE: Mean Absolute Error

- Supposons on cherche a prédire la température pour trois jours consécutifs. les prédictions et les valeurs réelles :
  - Jour 1 : Prédiction = 20°C, Réel = 25°C (Erreur = 20 - 25 = -5)
  - Jour 2 : Prédiction = 30°C, Réel = 28°C (Erreur = 30 - 28 = +2)
  - Jour 3 : Prédiction = 18°C, Réel = 16°C (Erreur = 18 - 16 = +2)

Sans valeur absolue

Somme des erreurs = (-5) + 2 + 2 = -1.

→ l'erreur totale semble faible (-1), mais cela ne reflète pas la réalité, car les erreurs positives et négatives se compensent partiellement.

→ Le modèle a clairement des erreurs importantes sur certains jours, mais elles sont masquées par cette somme.

Avec valeur absolue

Somme des erreurs absolues =
| -5 | + | 2 | + | 2 | = 5 + 2 + 2 = 9

→ une meilleure idée de l'ampleur réelle des erreurs,

→ l'erreur totale est 9, ce qui montre bien que le modèle a eu des prédictions incorrectes de manière plus marquée.

Evaluer un modèle

MSE: Mean Squared Error

- L'erreur quadratique moyenne élève chaque erreur au **carré**, puis calcule la moyenne.
- L'élévation au carré fait que l'erreur est maintenant dans l'**unité au carré**.
  - Si la cible est en degrés Celsius, la MSE sera en **degrés Celsius carrés**.

Evaluer un modèle

Mean Squared Error VS Mean Absolute Error

Comparaison des unités :

MAE : même unité que les cibles (facile à interpréter).

MSE : unité carrée des cibles (moins intuitive à interpréter directement).

→ Pour rendre l'unité de la MSE comparable à celle des cibles, on prend souvent la racine carrée de la MSE, ce qui donne la Root Mean Squared Error (RMSE), qui revient à l'unité d'origine, tout en préservant la sensibilité aux grandes erreurs.

Comparaison et choix selon le contexte :

- **MAE** est souvent préférée lorsque toutes les erreurs doivent être pondérées de manière égale, sans surpondérer les grandes erreurs.
  - Elle convient bien à des modèles robustes, moins sensibles aux anomalies ou aux valeurs aberrantes.

- **MSE** est plus adaptée lorsque l'on veut accorder une importance particulière aux grandes erreurs,
  - par exemple dans des applications sensibles où de grandes erreurs sont beaucoup plus coûteuses ou risquées. (exp: des prévisions financières où une erreur importante peut avoir des conséquences lourdes)

- **RMSE** combine les avantages de la MSE (pondération des grandes erreurs) et de la MAE (unité intuitive), ce qui en fait un compromis souvent utilisé en pratique.

Plan

1. Introduction
2. Bibliothèques en python pour l'analyse de données
3. K Nearest Neighbors
    1. Pseudo-code knn
    2. Principe de l'algorithme knn
    3. Distance pour calculer la similarité
    4. Précautions
4. Evaluer un modèle
5. Les Hyperparamètres en Apprentissage Automatique

Les Hyperparamètres en Apprentissage Automatique

Définition des Hyperparamètres

Les **hyperparamètres** sont des paramètres de configuration d'un modèle d'apprentissage automatique qui ne sont pas appris à partir des données, mais qui influencent le comportement et la performance du modèle.

≠

Les **paramètres** sont les inconnus que le modèle a pour objectif de déterminer dans le but de résoudre un problème.

Les Hyperparamètres en Apprentissage Automatique

Importance des Hyperparamètres

- Les choix d'hyperparamètres peuvent avoir un impact significatif sur la performance du modèle.
- Ils déterminent la complexité du modèle, la vitesse d'apprentissage, etc.

Réglage des Hyperparamètres

- Le réglage des hyperparamètres implique de trouver les meilleures valeurs pour ces paramètres afin d'optimiser la performance du modèle.

Exemple d'Hyperparamètres avec l'Algorithme KNN

- **Nombre de voisins (k)** : Détermine combien de voisins les données doivent être comparées pour la classification ou la régression. Un choix inapproprié de k peut entraîner un sous-ajustement ou un sur-ajustement.
- **Métrique de distance** : La distance utilisée pour mesurer la similarité entre les points, généralement la distance euclidienne, mais d'autres métriques telles que la distance de Manhattan sont également possibles.
- **Poids des voisins** : Permet de donner plus d'importance aux voisins proches en affectant des poids différents à chaque voisin. Par exemple, les voisins les plus proches peuvent avoir un poids plus élevé que les voisins plus éloignés.

Exemple d'Hyperparamètres avec l'Algorithme KNN

Réglage des Hyperparamètres

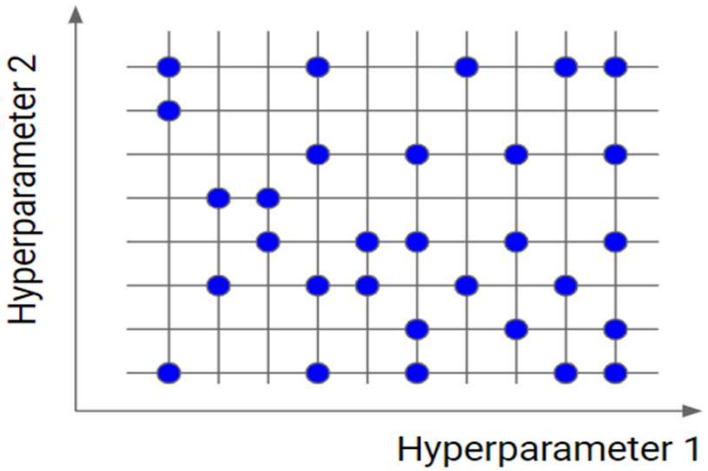
- Le choix de **k** peut être optimisé à l'aide de techniques telles que **la validation croisée** pour trouver la meilleure valeur.

- La métrique de distance et le poids des voisins peuvent également être ajustés pour obtenir les meilleures performances.

Optimisation des hyper paramètres

La technique recherche en Grille (Grid Search)

- Vise à automatiser le processus de réglage des hyperparamètres en testant systématiquement un ensemble prédéfini de valeurs pour chaque hyperparamètre.
- C'est une technique d'o



Recherche en Grille (Grid Search)

Méthode :

1. Spécifier un ensemble de valeurs possibles pour chaque hyperparamètre que l'on souhaite optimiser.
2. Enumèrer toutes les combinaisons possibles de ces valeurs (d'où le terme "grille")
3. Evaluer le modèle pour chaque combinaison en utilisant une technique d'évaluation comme la validation croisée.

Optimisation des hyper paramètres

Validation Croisée (Cross-Validation)

- Vise à évaluer la performance d'un modèle et à estimer comment il généralisera sur de nouvelles données.
- C'est une technique d'évaluation/une technique d'échantillonnage
- Principe : utiliser un ensemble de données pour ensuite les diviser en deux catégories. Ce sont :
  - les **données d'entraînement** utilisées pour entraîner le modèle,
  - les **données test** utilisées pour la prédiction.

Validation Croisée (Cross-Validation)

On dénote plusieurs techniques de validation croisée. Les principales sont :

- le **train-test split**,
- la **méthode k-folds**.

1- Le train-test split

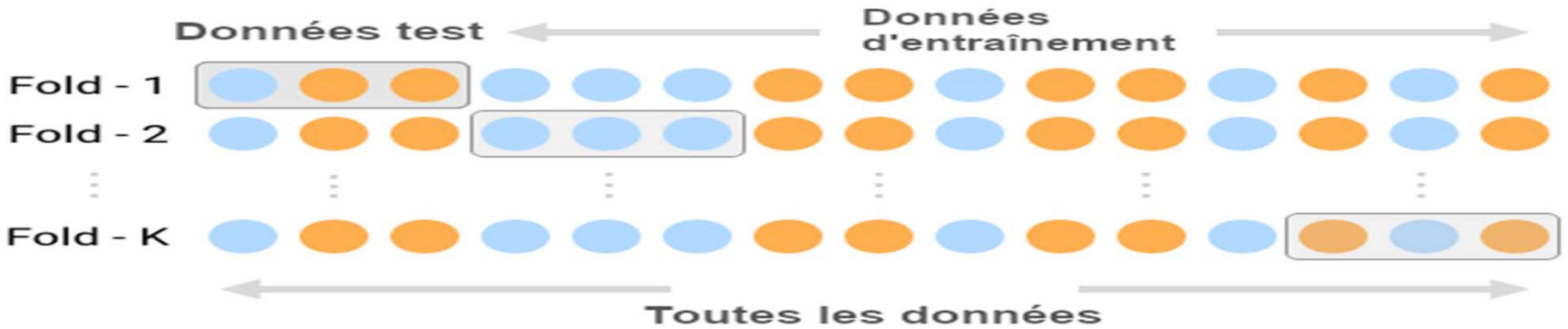
- Le principe de base du train-test split est de **décomposer l'ensemble des données de manière aléatoire**.
  - Une partie servira à entraîner le modèle de Machine Learning. L'autre partie, quant à elle, permet de réaliser le test de validation.
- En règle générale, 70 à 80 % des données seront destinés à l'entraînement. Le reste, c'est-à-dire les 20 à 30 %, seront exploités pour le cross validation.

Subject	*t*	Feature 1	Feature 2	Target
Paul	1	1000	male	0
Paul	2	1100	male	0
Paul	3	1200	male	1
Paul	4	1300	male	1
Crista	4	20	female	0
Crista	5	100	female	0
Paulina	1	10000	female	0
Paulina	2	100000	female	1
Paulina	3	95000	female	1
Paulina	4	97000	female	1
Paulina	5	99000	female	1
Paulina	6	101000	female	1
George	1	50000	male	1
George	2	50000	male	1
George	3	50000	male	1
George	4	50000	male	1
George	5	50000	male	1
George	6	50000	male	1

Subject	*t*	Feature 1	Feature 2	Target
Paul	1	1000	male	0
Paulina	1	10000	female	0
George	1	50000	male	1
Paul	2	1100	male	0
Paulina	2	100000	female	1
George	2	50000	male	1
Paul	3	1200	male	1
Paulina	3	95000	female	1
George	3	50000	male	1
Paul	4	1300	male	1
Crista	4	20	female	0
Paulina	4	97000	female	1
George	4	50000	male	1
Crista	5	100	female	0
Paulina	5	99000	female	1
George	5	50000	male	1
Paulina	6	101000	female	1
George	6	50000	male	1

2- La méthode k-folds.

1. L'ensemble de données est divisé en k folds (ou sous-ensembles) de taille égale.
2. Le modèle est entraîné k fois, chaque fois en utilisant **k-1** folds comme ensemble d'entraînement et le fold restant comme ensemble de validation.
3. L'opération est répétée k fois, de sorte que chaque fold soit utilisé comme ensemble de **validation exactement une fois**.
4. Les performances (comme l'erreur moyenne absolue ou l'erreur quadratique moyenne) sont mesurées pour chaque itération.



2- La méthode k-folds.

Algorithme des K-Fold:

- Diviser le DataFrame en **k** partitions égales
- Sélectionner **k-1** partitions pour le set de training
- Sélectionner la partition restante pour le set de test
- Entrainer le modèle sur le set de training
- Utiliser la modèle entrainé pour prédire les résultats sur le set de test
- Calculer la métrique d'erreur de ce fold
- Répéter toutes les étapes **k-1** fois, jusqu'à ce que chaque partition ait été utiliser en set de test pour une itération
- Calculer la moyenne des **k** valeurs d'erreur