mettre iris.zip sur site

Prédire la variété d'un iris

Christophe Viroulaud

Première NSI

En 1936, le biologiste *Ronald Fisher* a rassemblé les mesures de trois espèces d'iris.


Iris setosa


Iris versicolor


Iris virginica

Il est possible d'utiliser ces données pour pouvoir classifier un iris inconnu.
Comment produire déduction à partir de données existantes ? → prémisse
IA : machine learning

Comment prédire une information nouvelle à partir de données brutes ?

Comment prédire une information nouvelle à partir de données brutes ?

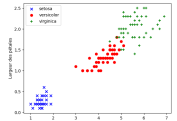


FIGURE – Variétés d'iris en fonction de leurs mesures

1. Une représentation graphique des informations apporte une compréhension plus éclairante.
2. Il apparaît que les mesures d'un iris peuvent permettre de déterminer leur variété.

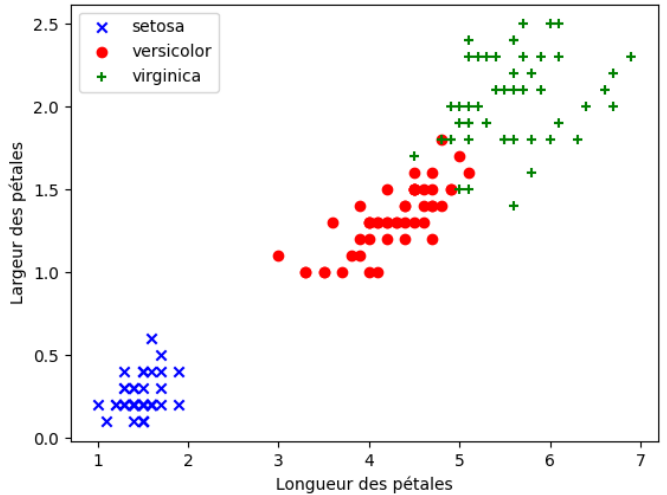


FIGURE – Variétés d'iris en fonction de leurs mesures

Utiliser les données pour prédire

Activité 1 :

1. Déterminer la variété des iris suivants :

longueur	1	6	5.1	2.5
largeur	0.5	2.5	1.55	0.85

2. Proposer une méthode pour effectuer un choix dans les cas ambigus.

Utiliser les données pour prédire

Activité 1 :

1. Déterminer la variété des iris suivants :

longueur	1	6	5.1	2.5
largeur	0.5	2.5	1.55	0.85

2. Proposer une méthode pour effectuer un choix dans les cas ambigus.

Correction

longueur	1	6	5.1	2.5
largeur	0.5	2.5	1.55	0.85
variété	setosa	virginica	ambigu	ambigu

Correction

longueur	1	6	5.1	2.5
largeur	0.5	2.5	1.55	0.85
variété	setosa	virginica	ambigu	ambigu

Méthode des **k plus proches voisins**

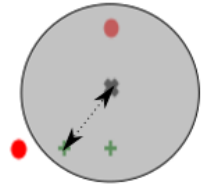
Pour déterminer la variété d'un iris inconnu :
- regarder la variété d'un nombre *k* de voisins,

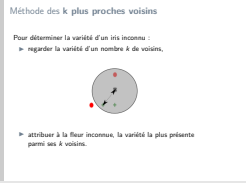
k **N**earest **N**eighbors

Méthode des **k plus proches voisins**

Pour déterminer la variété d'un iris inconnu :

- regarder la variété d'un nombre *k* de voisins,



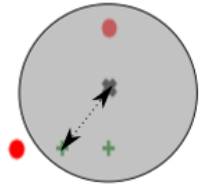


k **N**earest **N**eighbors

Méthode des **k plus proches voisins**

Pour déterminer la variété d'un iris inconnu :

- regarder la variété d'un nombre k de voisins,



- attribuer à la fleur inconnue, la variété la plus présente parmi ses k voisins.

Choix de k

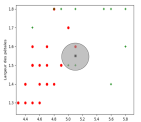


Figure – Détermination de l'iris (5.05, 1.5) pour $k = 3$

Choix de k

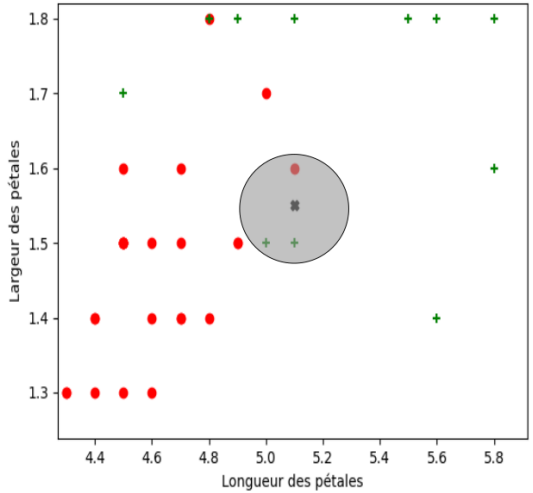


Figure – Détermination de l'iris (5.05, 1.5) pour $k = 3$

Choix de k

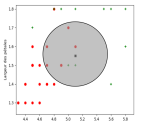


Figure – Détermination de l'iris (5.05, 1.5) pour $k = 7$

1. avantages de kNN : simple d'implémentation et résultats avec bon taux de réussite quand on a un gros échantillon
2. inconvénients : sensible au bruit (données mal étiquetées), sensible à l'échelle de chaque dimension

Choix de k

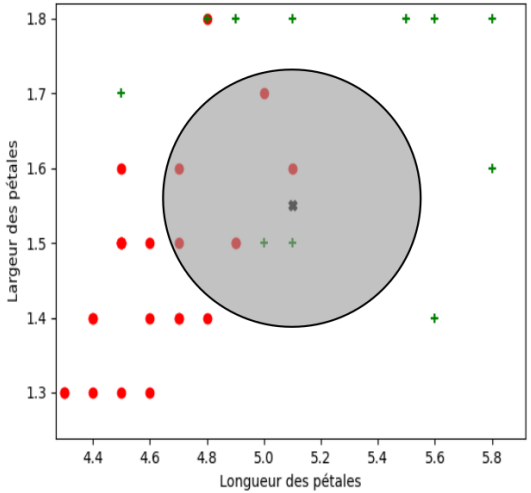


Figure – Détermination de l'iris (5.05, 1.5) pour $k = 7$

Complément
L'algorithme *kNN* est une méthode d'apprentissage *supervisé* : l'algorithme reçoit un ensemble de données déjà étiquetées sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction.

1. non supervisé : données pas étiquetées
2. par renforcement : par récompense

Complément

L'algorithme *kNN* est une méthode d'apprentissage *supervisé* : l'algorithme reçoit un ensemble de données déjà étiquetées sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction.

Calcul de la distance

Le plus naturel ici est de prendre la distance *à vol d'oiseau* ou plus formellement la **distance euclidienne**.

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

FIGURE – distance euclidienne

Calcul de la distance

Le plus naturel ici est de prendre la distance *à vol d'oiseau* ou plus formellement la **distance euclidienne**.

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

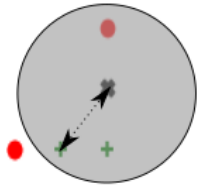


FIGURE – distance euclidienne

Calcul de la distance
$d = |x_A - x_B| + |y_A - y_B|$
FIGURE – distance de Manhattan

Calcul de la distance

$$d = |x_A - x_B| + |y_A - y_B|$$

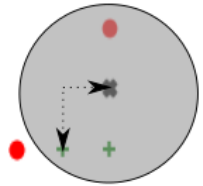


FIGURE – distance de Manhattan

Activité 2 : Écrire *en langage naturel*, l'algorithme kNN.

> **Activité 2 :** Écrire *en langage naturel*, l'algorithme kNN.

Correction

- Charger les données dans le programme.
- Choisir k.
- Stocker les mesures de la fleur inconnue.
- Calculer la distance euclidienne entre la fleur inconnue et tous les autres iris.
- Sélectionner les k plus proches iris (en distance) de la fleur inconnue.
- Affecter la variété majoritaire des k plus proches iris (en distance) à la fleur inconnue.

Correction

- Charger les données dans le programme.
- Choisir k.
- Stocker les mesures de la fleur inconnue.
- Calculer la distance euclidienne entre la fleur inconnue et tous les autres iris.
- Sélectionner les k plus proches iris (en distance) de la fleur inconnue.
- Affecter la variété majoritaire des k plus proches iris (en distance) à la fleur inconnue.

Pour charger les données on utilisera la bibliothèque *csv*.

Pour charger les données on utilisera la bibliothèque *csv*.


Activité 3 :
1. Télécharger le dossier compressé *iris.zip* sur le site https://cviroulaud.github.io
2. Ouvrir le fichier *data-iris.csv* avec un tableur pour observer les données.
3. Ouvrir le fichier *iris-eleve.py*.

Activité 3 :

1. Télécharger le dossier compressé *iris.zip* sur le site `https://cviroulaud.github.io`
2. Ouvrir le fichier *data-iris.csv* avec un tableur pour observer les données.
3. Ouvrir le fichier *iris-eleve.py*.

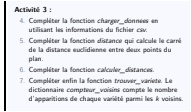
Correction

petal_length	petal_width	species
1.4	0.2	setosa
1.4	0.2	setosa
1.3	0.2	setosa

Correction

3 attributs

petal_length	petal_width	species
1.4	0.2	setosa
1.4	0.2	setosa
1.3	0.2	setosa


Activité 3 :
4. Compléter la fonction *charger_donnees* en utilisant les informations du fichier *csv*.
5. Compléter la fonction *distance* qui calcule le carré de la distance euclidienne entre deux points du plan.
6. Compléter la fonction *calculer_distances*.
7. Compléter enfin la fonction *trouver_variete*. Le dictionnaire *compteur_voisins* compte le nombre d'apparitions de chaque variété parmi les *k* voisins.

Activité 3 :

4. Compléter la fonction *charger_donnees* en utilisant les informations du fichier *csv*.
5. Compléter la fonction *distance* qui calcule le carré de la distance euclidienne entre deux points du plan.
6. Compléter la fonction *calculer_distances*.
7. Compléter enfin la fonction *trouver_variete*. Le dictionnaire *compteur_voisins* compte le nombre d'apparitions de chaque variété parmi les *k* voisins.

Correction

Correction

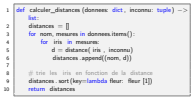
```
1  def charger_donnees(nom_fichier: str) -> dict:
2      fichier = open(nom_fichier)
3      data_iris = csv.DictReader( fichier , delimiter =",")
4      dico_varietes = {"setosa": [], " versicolor ": [], "
           virginica ": []}
5      # Pour chaque ligne de données
6      for iris in data_iris :
7          # Stocke la longueur et la largeur sous forme de
               tuple de flottants
8          dico_varietes [ iris ["species"]]. append(
9              ( float ( iris ["petal_length"]), float ( iris ["
                   petal_width"])))
10     fichier . close ()
11     return dico_varietes
```

Correction

Correction

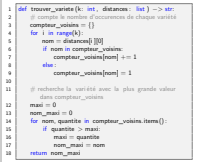
```
1 def distance(connu: tuple, inconnu: tuple) -> float:
2     return (connu[0]-inconnu[0])**2+(connu[1]-inconnu
          [1])**2
```

Correction



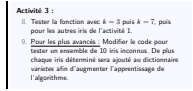
Correction

```python
def calculer_distances(donnees: dict, inconnu: tuple) -> list:
    distances = []
    for nom, mesures in donnees.items():
        for iris in mesures:
            d = distance(iris, inconnu)
            distances.append((nom, d))

    # trie les iris en fonction de la distance
    distances.sort(key=lambda fleur: fleur[1])
    return distances
```



Correction

```python
def trouver_variete(k: int, distances: list) -> str:
    # compte le nombre d'occurences de chaque variété
    compteur_voisins = {}
    for i in range(k):
        nom = distances[i][0]
        if nom in compteur_voisins:
            compteur_voisins[nom] += 1
        else:
            compteur_voisins[nom] = 1

    # recherche la variété avec la plus grande valeur
      dans compteur_voisins
    maxi = 0
    nom_maxi = 0
    for nom, quantite in compteur_voisins.items():
        if quantite > maxi:
            maxi = quantite
            nom_maxi = nom
    return nom_maxi
```


Activité 3 :
8. Tester la fonction avec $k = 3$ puis $k = 7$, puis pour les autres iris de l'activité 1.
9. Pour les plus avancés : Modifier le code pour tester un ensemble de 10 iris inconnus. De plus chaque iris déterminé sera ajouté au dictionnaire varietes afin d'augmenter l'apprentissage de l'algorithme.

Activité 3 :

8. Tester la fonction avec $k = 3$ puis $k = 7$, puis pour les autres iris de l'activité 1.
9. Pour les plus avancés : Modifier le code pour tester un ensemble de 10 iris inconnus. De plus chaque iris déterminé sera ajouté au dictionnaire *varietes* afin d'augmenter l'apprentissage de l'algorithme.

Correction


```
k = 3
cible = (5.1, 1.55)

varietes = charger_donnees("data-iris.csv")
distances_cible = calculer_distances( varietes , cible )
variete = trouver_variete(k, distances_cible )

print ("La variété est ", variete )
```


Correction

```
k = 3
cible = (5.1, 1.55)

varietes = charger_donnees("data-iris.csv")
distances_cible = calculer_distances( varietes , cible )
variete = trouver_variete(k, distances_cible )

print ("La variété est ", variete )
```

Correction

Correction

```
k = 3
cibles = [(1,0.5) ,(6,2.5) ,(5.1, 1.55) ,(2.5,0.85) ,(3,2) ,
          (6,1.2) ,(2.1,1) ,(3.2,1.5) ,(3.5,2.5) ,(4,1) ]

varietes = charger_donnees("data-iris.csv")
for iris_inconnu in cibles :
    # trouve la variété
    distances_cible = calculer_distances( varietes ,
        iris_inconnu )
    variete = trouver_variete(k, distances_cible )
    print(f"La variété de {iris_inconnu} est {variete}.")
    # ajout de cible au dictionnaire des données
    varietes [ variete ].append(iris_inconnu)
```