TP Classification des iris par k-NN

L'algorithme des k plus proches voisins (k-NN pour k-nearest neighbors) appartient à la famille des algorithmes d'apprentissage automatique (*machine learning*), dans la catégorie « apprentissage supervisé » : il est donc nécessaire d'avoir des données labellisées.

À partir d'un ensemble E de données labellisées, il sera possible de classer (déterminer le label) d'une nouvelle donnée (donnée n'appartenant pas à E)

Le fichier ***iris.csv*** contient les données de plusieurs dizaines d'iris. Et pour chaque iris :

- la longueur des pétales
- la largeur des pétales
- l'espèce de l'iris : au lieu d'utiliser les noms des espèces, on utilisera des chiffres, 0 pour « iris setosa », 1 pour « iris virginica » et 2 pour « iris versicolor »)

iris setosa

iris versicolor

iris virginica

Le but de ce TP est donc de créer un programme Python permettant d'inférer l'espèce d'un iris à partir de la longueur et de la largeur de ses pétales. Autrement dit, si Alexandre cueille un iris en forêt, le programme pourra lui prédire l'espèce à partir de la mesure de ses pétales.

Classification des iris par k-NN

I – fonction d'import du CSV

Implémenter la fonction `readCSV` dont le prototype est le suivant :

`readCSV(dossier:str, fichier:str) -> list`

La fonction lit le fichier `fichier` se trouvant dans le dossier `dossier` et renvoie une liste de dictionnaires, chaque dictionnaire représentant une ligne.

Format de retour :

```
[
    {'petal_length': '1.4', 'petal_width': '0.2', 'species': '0'},
    {'petal_length':'1.4', 'petal_width': '0.2', 'species': '0'},
    {'petal_length': '1.3', 'petal_width': '0.2', 'species': '0'},
    …
]
```

Dans la suite du document, cette liste de dictionnaires sera appelée **`fleurs`**

On utilisera :

- le module `os` pour le chargement des fichiers, afin d'éviter les erreurs d'import
  ```
  import os
  f = os.path.join(dossier, fichier)
  ```
  La doc est ici
- l'objet `DictReader` du module `csv`, histoire de ne pas ré-inventer la roue
  voir l'exemple de la doc python

II – Calcul de distance

2.1 fonction distance

Implémenter une fonction `distance` calculant la distance euclidienne entre deux points :

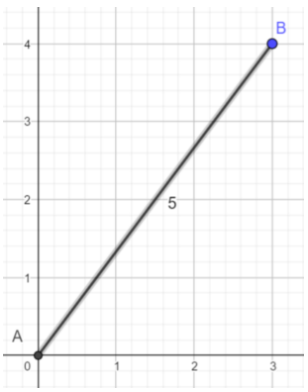
prototype :

`distance(x1:float, y1:float, x2:float, y2:float) → float`

on utilisera la fonction `sqrt` du module `math`

exemple d'utilisation :

```
>>> distance(0,0,3,4)
5.0
```



2.2 distance à une fleur

Implémenter une fonction `aj_distance` dont le prototype est le suivant :

```
aj_distance(new_x:float, new_y:float, data:list) -> None
```

`aj_distance` permet d'ajouter dans chaque dictionnaires de la liste `fleurs` ci-dessus la « distance entre les pétales », c'est-à-dire la distance euclidienne entre chaque couples ('petal_length', 'petal_width') et les données d'une nouvelle fleur (new_x, new_y)

exemple d'utilisation :

```
>>> fleurs = readCSV("E:\\1ere_NSI\\Algo\\KNN", "iris.csv")
>>> aj_distance(1,2,fleurs)
>>> fleurs
[
      {'petal_length': '1.4', 'petal_width': '0.2', 'species': '0', 'distance': 1.84},
      {'petal_length': '1.3', 'petal_width': '0.2', 'species': '0', 'distance': 1.82},
      …
]
```

III – Tri des données

Implémenter une fonction `trier` dont le prototype est le suivant :

```
trier(data:list, cle:str) -> None
```

Cette fonction implémente un algorithme de **tri en place** de la liste `data` en argument, en fonction de la clef `cle` fournie. Vous implémenterez un algorithme de tri de votre choix : tri par sélection, tri par insertion, tri fusion….
On l'utilisera pour trier les « lignes » de la liste `fleurs` ci-dessus en fonction d'une des entrées des `petal_length, petal_width, species ou distance`

IV – Visualisation

Écrire une fonction permettant de visualiser le nuage de points : ces points représentent les fleurs dont l'abscisse est `petal_length` et l'ordonnée `petal_width`. Une couleur différente est utilisée pour chaque variété de fleur.

Nous utiliserons la bibliothèque `matplotlib` à cet effet. Voici un exemple fictif d'utilisation de la bibliothèque pour produire un nuage de points (méthode `scatter`) :

```
import matplotlib.pyplot as plt

#définition des axes
x = [1, 2, 3, 4, 5]
plt.xlabel('mes abscisses')
plt.ylabel('mes ordonnées')
plt.title("Test d'affichage d'un nuage de points")
#ordonnées des points à placer
y1 = [1, 2, 3, 4, 5]
y2 = [2, 4, 9, 16, 25]

#affichage
plt.scatter(x, y1, c = 'red')
plt.scatter(x, y2, c = 'yellow')
plt.show()
```

V – classification d'un nouvel iris

Écrire une fonction `knn` dont le prototype est le suivant :

```
knn(new_x:float, new_y:float, data:list, k:int) -> int
```

`new_x` et `new_y` représentent le `petal_length` et le `petal_width` respectivement du nouvel iris dont nous souhaitons déterminer l'espèce par l'algorithme des k plus proches voisins.
Pour `data`, nous utiliserons la liste `fleurs`
`k` est le nombre de voisins pris en compte par l'algorithme k-NN pour déterminer l'espèce.
La fonction renvoie 0, 1 ou 2 représentant l'estimation de l'espèce pour l'iris ayant des pétales de mesures `new_x` et `new_y`

VI – détermination d'un k optimal

Le but est de :

1. retirer aléatoirement un sous-ensemble des données de la liste `fleurs` (par exemple 5%)
2. tester ces 5% en utilisant l'algorithme k-NN basé sur les 95% restant avec plusieurs valeurs de k et calculer un indicateur du nombre de bonnes réponses pour chaque valeur k testée.
3. Réitérer les étapes 1 et 2 pour différents sous-ensembles.

Le nom des fonctions et les méthodes sont libres...