Rapport classification challenge de Paul-Adrien LU-YEN-TUNG, Merwane MALLEM, et Antoine LIN (groupe i)

Pseudo KAGGLE : pol-adr1 , merwane_mallem, Antoine Lin

- *Les fonctions annexes*

```python
## fonction annexe
def distance(L1,L2):
    a=0
    for i in range(len(L1)):
        a=a+(L1[i]-L2[i])**2
    return math.sqrt(a)

def indice_max(L):
    a=0
    b=L[0]
    for i in range(len(L)):
        if b<=L[i]:
            a=i
            b=L[i]
    return a
```

Nous avons crée deux fonctions annexes. Une fonction distance qui calcule la distance euclidienne en prenant comme argument deux listes. Et une autre fonction qui renvoie l'indice du maximum dans une liste d'entiers.

- *Lecture des fichiers train et test*

```python

# Ouvrir le fichier et lire les données
with open('train_txt.txt', 'r') as file:
    data = []
    for line in file:
        row = line.strip().split(',')

        data.append(row)
    for i in range(len(data)):
        for j in range(9):
            data[i][j]=float(data[i][j])

#ouvrir le fichier test pour tester l'algo knn

with open('test_txt.txt', 'r') as file:
    data1 = []
    for line in file:
        row = line.strip().split(',')
        data1.append(row)
    for i in range(len(data1)):
        for j in range(8):
            data1[i][j]=float(data1[i][j])
```

Stockage des données du fichier trains sous forme de liste (data), stockage des données du fichier test dans la liste data1.

```
50
51  #création d'une liste qui prend que les paramètres numériques afin d'appliquer la fonction distance
52  data_float=[]
53  for i in range(len(data)):
54      data_float.append(data[i][1:-1])
55
56  data1_float=[]
57  for i in range(len(data1)):
58      data1_float.append(data1[i][1:])
59
```

Puis on a crée deux nouvelles listes : data_float et data1_float qui sont des listes contenant uniquement les paramètres numériques des listes data et data1, afin de pouvoir appliquer la fonction *distance*.

- *Algorithme KNN*

```
60
61  ##algorithme knn
62
63  def knn(k,L):
64      dist_list=[[] for i in range(len(data))]
65      for i in range(len(data)):
66          dist_list[i].append(distance(L,data_float[i]))
67          dist_list[i].append(data[i][-1])
68      dist_list=sorted(dist_list, key=lambda x: x[0])
69      label=set()
70      for i in range(k):
71          label.add(dist_list[i][1])
72      label_list=list(label)
73      frequence=[]
74      for i in range(len(label_list)):
75          b=0
76          for j in range(k):
77              if dist_list[j][1]==label_list[i]:
78                  b=b+1
79          frequence.append(b)
80      return label_list[indice_max(frequence)]
81
```

Ensuite, nous avons codé la fonction knn qui prend en argument k=le nombre de voisin, et L la liste contenant des paramètres du label à trouver.

On a donc introduit une liste dist_list contenant les distances euclidiennes entre L et les labels du fichier train cad les listes dans data_float.

Puis, on ordonne par ordre croissant les distances euclidiennes.

On sélectionne les k plus proches voisins.

Et enfin, on regarde quel label revient le plus fréquemment afin de déterminer la nature du label dont les paramètres sont dans la liste L.

```
82   ## tester les resultats du knn avec les données du fichier test_fleur
83  A=[]
84  for i in range(len(data1)):
85      A.append(knn(1,data1_float[i]))
86
```

Et enfin on crée une liste A qui contient toutes les prédictions de notre fonction knn.

Après plusieurs tentatives de k, nous avons obtenu le meilleur résultat avec k=1.

Nous avons testé différentes valeurs de k comme : 40,25,10,5,3,2,1.

Pour k=1, nous obtenons notre meilleur score : 0.98731.

L'algorithme que nous avons mis en place a été vu en TD data science IA (wp2)

- Charger les données
- Initialiser la valeur de k
- Pour obtenir la classe prédite, itérer de 1 jusqu'au nombre total de points d'apprentissage:
  - Calculer la distance entre la donnée de test et chaque données d'apprentissage. (Nous pouvons utiliser une distance Euclidienne)
  - ordonner les distances calculées par ordre croissant
  - Prendre les top k trouvées
  - retourner la classe la plus fréquente parmis les top k

- *Création du fichier csv (utilisation de la bibliothèque pandas)*

```
90  import pandas as pd
91
92  A=[int(A[i]) for i in range(len(A))]
93  # Données pour les deux colonnes
94  data = {
95      "Id": [i for i in range(1030,5030)],
96      "Label": A
97  }
98
99  # Créer un DataFrame avec les données
100 df = pd.DataFrame(data)
101
102 # Écrire le DataFrame dans un fichier Excel
103 df.to_csv('fichier.csv', index=False)
104
105 print("Fichier Excel 'fichier.xlsx' créé avec succès.")
106
```

Et enfin, on a crée à partir de la bibliothèque pandas le fichier CSV que l'on a soumis sur Kaggle.