Formation Machine Learning

Atelier Pratique AP-ML3

ALIASE

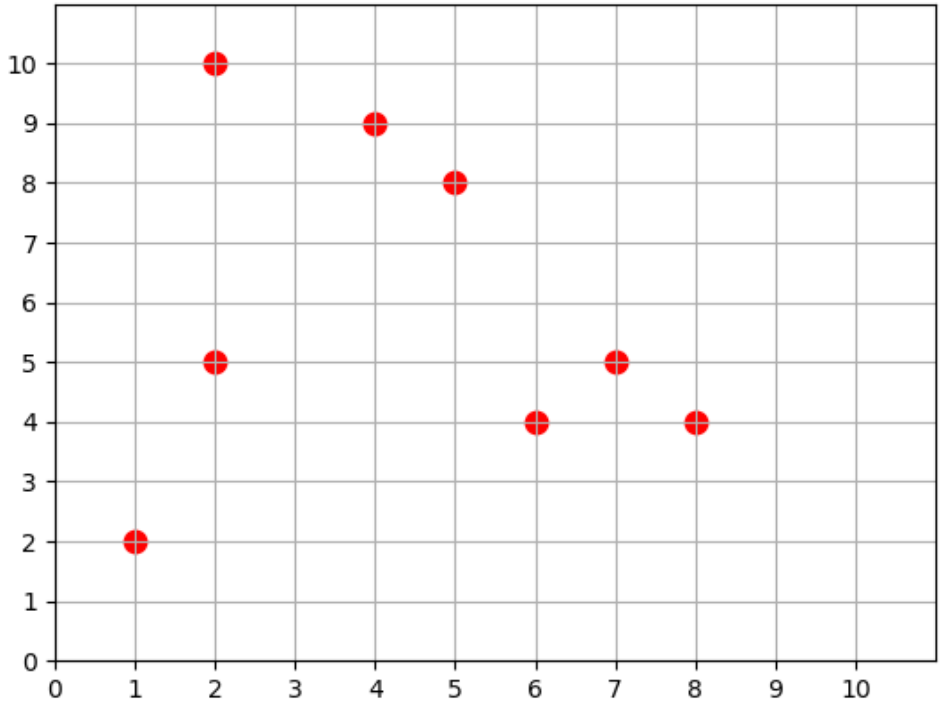
Atelier Pratique AP-ML3 : Exercice ML3.4

ALIASE
IA - SERVICE & FORMATION

Objectif : Programmer l'algorithme K-Means

a) Charger le dataset à partir du fichier ***dataset2-KMeans.csv*** dans un dataframe ***df***

b) Afficher les données sur un plan en utilisant la fonction ***scatter .*** La figure suivante devrait s'afficher :



Atelier Pratique AP-ML3 : Exercice ML3.4

ALIASE
IA - SERVICE & FORMATION

c) Ajouter K colonnes colonne **'dist-0' 'dist-1' 'dist-2'** . . . au dataframe

d) Définir une liste ***Li_centres*** et tirer K centres **aléatoirement** puis les insérer dans la liste.

Pour cela, on définit une fonction ***initialiser_centres()*** qui retourne **l'état initial** de la liste ***Li_centres***

Clusters			0	1	2	
Centres			(3.67, 9)	(7, 4.3)	(1.5, 3.5)	
Points	(x1 ,	x2)	dist-0	dist-1	dist-2	y (Cluster)
A	2	10	2.67	10.7	7	0
B	2	5	5.67	5.7	2	2
C	8	4	9.33	1.3	7	1
D	5	8	2.33	5.7	8	0
E	7	5	7.33	0.7	7	1
F	6	4	7.33	1.3	5	1
G	1	2	9.67	8.3	2	2
H	4	9	0.33	7.7	8	0

Li_centres

dist-0, dist-1, dist-2

Atelier Pratique AP-ML3 : Exercice ML3.4

ALIASE
IA - SERVICE & FORMATION

e) Définir une fonction ***distance (pt1, pt2)*** qui calcule la distance de **Manhattan** entre deux points pt1 et pt2.

pt1 et pt2 sont des tuples : pt1 (a1, a2) et pt2 (b1, b2)

$$d(A,B) = \sum_{i=1}^{n} | ai - bi |$$

f) Définir une fonction ***Rallier_points(Li_centres)*** qui rallie chaque point au centre le plus proche :

Pour chacun des points dans la dataset :

- calculer la distance entre le point et chacun des **K centres**
- stocker la distance calculée dans la colonne ***dist-i*** (i = 0, 1, 2, … K)
- calculer la distance minimale
- mettre à jour la colonne cluster avec le k pour lequel ***dist-k*** = distance minimale

Atelier Pratique AP-ML3 : Exercice ML3.4

ALIASE
IA - SERVICE & FORMATION

g) Définir une fonction ***c_gravite(cluster)*** qui retourne ***un tuple*** = centre de gravité d'un cluster (df[« cluster »])

h) Définir une fonction ***calculer_new_centres()*** qui recalcule et retourne la **liste des nouveaux centres**

i) Définir une fonction ***stabilite(cluster1, cluster2)*** qui retourne True si les 2 clusters sont identiques

j) Définit une fonction ***MyKmeans()*** qui boucle jusqu'à stabilisation des clusters :

. Tirer K centres aléatoirement

. Rallier chaque point au centre le plus proche

. Tester si stabilité atteinte. Si oui, arrêt de la boucle

. Recalculer des nouveaux centres de gravité des clusters

k) Lancer la fonction ***MyKmeans()***

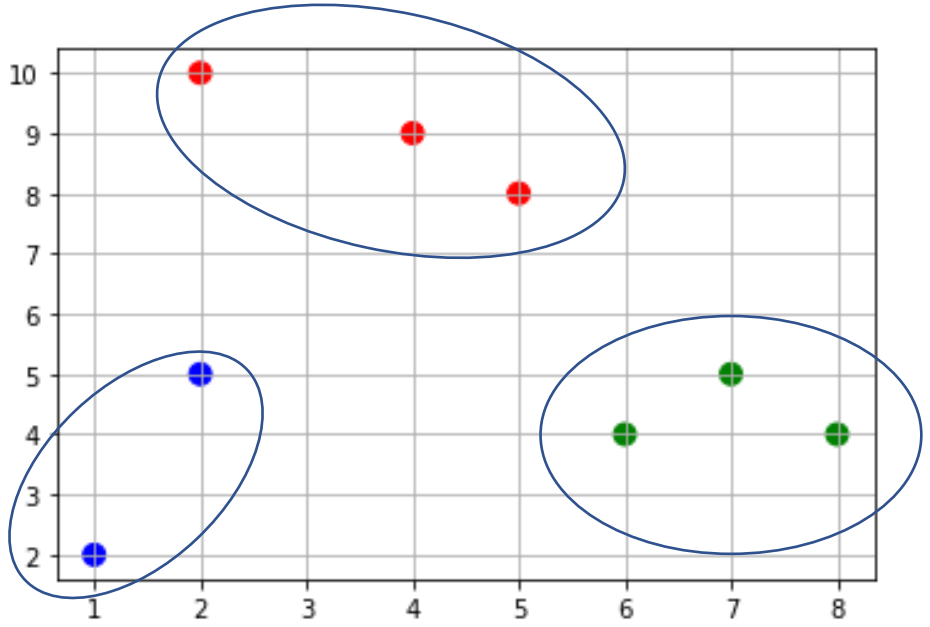
Atelier Pratique AP-ML3 : Exercice ML3.4

ALIASE
IA - SERVICE & FORMATION

l) Modifier la fonction ***MyKMeans()*** pour afficher avec scatter l'état courant des clusters en 3 couleurs différentes à chaque passage dans la boucle

m) Afficher l'état FINAL des clusters en 3 couleurs différentes (K=3) sur un plan en utilisant la fonction ***scatter .***

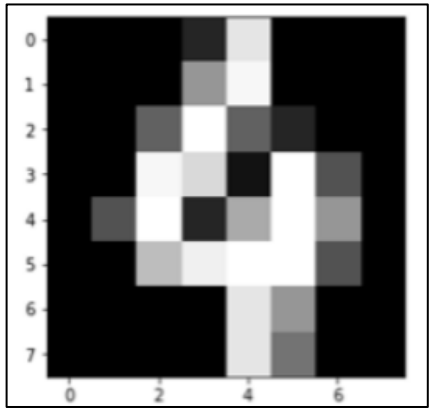
n) Faire les prédictions avec ***SKLEARN*** et afficher les clusters en 3 couleurs différentes (K=3) sur un plan.
Constater que les clusters générés avec SKLEARN sont les mêmes que ceux obtenus avec ***MyKMeans()***



SOLUTION

Atelier Pratique AP-ML3 : Exercice ML3.5

Objectif : Utiliser KMeans pour regrouper les images manuscrites dans 10 clusters



Atelier Pratique AP-ML3 : Exercice ML3.5

ALIASE
IA - SERVICE & FORMATION

Démarche :

1) Importer la base de données de chiffre

digits = **load_digits**()

X = digits.data ; **Y** = digits.target

2) Créer le model : mymodel = **KMeans**(n_clusters=**10**)

3) Entrainer le modèle : mymodel.**fit**(X)

4) Lancer le modèle (predict) et afficher les Clusters crées : predictions = mymodel.**predict**(X)

Atelier Pratique AP-ML3 : Exercice ML3.5

ALIAS
IA - SERVICE & FORMATION

5) Créer un dictionnaire : ***d_clusters = { k : v }***

k = *no cluster* , ***v*** = *liste des labels d'images* qui ont été regroupées dans le cluster [***label_im1, label_im2, …***]

via une fonction **create_dico_clusters( ):**

→ Pour cela, utiliser :

- **Y** : liste des labels d'images dans le dataset
- predictions : liste de no clusters prédits pour chaque image dans Y

```
d_clusters = {
    0 : [ label_im1,   label_im2, … ] ,      # images regroupées dans le cluster 0
    1 : [label_im1,   label_im2, … ] ,       # images regroupées dans le cluster 1
    2 : [label_im1,   label_im2, … ] ,       # images regroupées dans le cluster 2

    9 : [label_im1,   label_im2, … ]         # images regroupées dans le cluster 9
}
```

Atelier Pratique AP-ML3 : Exercice ML3.5

ALIAS
IA - SERVICE & FORMATION

6) A partir du dico **d_clusters**, créer un dico : ***d_distribution = { k : v }***

k = *no cluster* ,

v = *dico* ***{ im0:count0, im1:count1, ... }*** pour chacune des images dans ce cluster ,

où count est le nombre d'occurrences de l'image im dans le cluster no_cluster

via une fonction **create_dico_distribution(d_clusters)**

```
d_distribution = {

    0 : { im0:count0,  im1:count1, ... } ,      # nb d'occurrences de chaque image im dans le cluster 0
    1 : { im0:count0,  im1:count1, ... } ,      # nb d'occurrences de chaque image im dans le cluster 1


    9 : { im0:count0,  im1:count1, ... } ,      # nb d'occurrences de chaque image im dans le cluster 9
}
```

Atelier Pratique AP-ML3 : Exercice ML3.5

ALIASE
IA - SERVICE & FORMATION

7) Définir une fonction *affiche_distribution(d_distribution):*

Pour chacun des clusters, afficher le no cluster et son dico de distribution

```
Cluster : 0
Distribution : {  4: 163,    5: 2,    0: 1  }                    # L'image 4 est majoritaire dans le cluster 0

Cluster : 1
Distribution : {  3: 155,   9: 6,   5: 1,   2: 13,   1: 1,   8: 4  }    # L'image 3 est majoritaire dans le cluster 1

. . .

Cluster : 9
Distribution : {  1: 99,   2: 8,   8: 102,   9: 2,   6: 2,   4: 4,   7: 2,   3: 7  }  # L'image 8 est majoritaire dans le cluster 9
```

Atelier Pratique AP-ML3 : Exercice ML3.5

ALIAS
IA - SERVICE & FORMATION

8) A partir du dico d_distribution, créer un dico : ***d_resultat_final = { k : v }***

k = no cluster ,

v = ***dico {"image" : no_image, "pct" : pct}***

où no_image = l'image représentée par ce cluster (image majoritaire)

pct = pourcentage de présence de l'image dans ce cluster

via une fonction create_dico_resultat_final(d_distribution, d_resultat_final):

```
d_resultat_final = {

    0 : { "image":im0,   "pct": 98 } ,      # pourcentage de présence de l'image im0 dans le cluster 0
    1 : { "image":im1,   "pct": 95 } ,      # pourcentage de présence de l'image im1 dans le cluster 1


    9 : { "image":im9,   "pct": 97 }       # pourcentage de présence de l'image im9 dans le cluster 9
}
```

Atelier Pratique AP-ML3 : Exercice ML3.5

ALIASE
IA - SERVICE & FORMATION

9) Définir une *fonction* *affiche_resultat_final(d_resultat_final):*

pour chaque cluster dans le dico *d_resultat_final* afficher :

Cluster : 0
--> image représentée par le cluster : 4
--> % de présence dans le cluster : 90.71

Cluster : 1
--> image représentée par le cluster : 3
--> % de présence dans le cluster : 86.25

. . .

Cluster : 9
--> image représentée par le cluster : 8
--> % de présence dans le cluster : 56.76

SOLUTION

Atelier Pratique AP-ML3 : Quiz

Quiz : Machine Learning, Objectif pédagogique 3