Formation DIU EIL 2021

Python et analyse des données

Zaouche Djaouida

1. Récupérer le fichier users.csv sur le drive. Utiliser ce fichier pour construire une liste python users contenant un dictionnaire pour chaque utilisateur du fichier. Dans ce TP, un utilisateur est décrit par un identificateur, un nom et d'autres attributs que l'on rajoutera par la suite. La liste *users* peut être obtenue comme suit :

```
# Read CSV file
        import pandas as pd
        # Read the CSV into a pandas data frame (df)
        #   With a df you can do many things
          #   most important: visualize data with Seaborn
          df = pd.read_csv(pathOfYourCsvFile/users.csv', sep=',')
           users=[ ]
           for i in range(len(df)):
                  users.append({"id":df["id"][i],"name":df["name"][i]})
```

Ainsi :

```
>>> print(users)
    [{'id': 0, 'name': 'Alex'}, {'id': 1, 'name': 'Paul'}, {'id': 2, 'name': 'Charle'}, {'id': 3, 'name':
    'Anne'}, {'id':     4, 'name': 'Chi'}, {'id': 5, 'name': ' Amine'}]
```

2. Ecrire une fonction python qui, accepte une liste d'utilisateurs en entrée, visualise les informations des utilisateurs. Par exemple :

```
>>> displayInfo(users)
  id : 0   name : Alex
  id : 1   name : Paul
  id : 2   name : Charle
  id : 3   name : Anne
  id : 4   name : Chi
  id : 5   name :  Amine
```

3. Dans cette partie, nous définissons une relation d'amitié entre les utilisateurs (voir le graphe de la figure 1). Créer un fichier friendships.csv qui permettrait de construire la liste suivante : friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (5, 4)]. Ecrire le code python qui permet de construire la liste friendships à partir du fichier .csv.

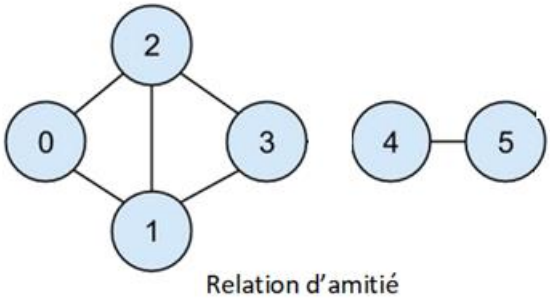

Figure 1

4. Ecrire une fonction addFriends qui accepte deux paramètres : une liste d'utilisateurs et une liste de couples définissant des relations d'amitié binaires entre les utilisateurs. La fonction ajoute un attribut friends à chaque utilisateur, contenant les identificateurs de ses amis. Par exemple :

```
>>> addFriends(users, friendships)
>>> displayInfo(users)
        id : 0 name : : Alex   friends : [1, 2]
        id : 1  name : Paul  friends : [0, 2, 3]
        id : 2   name : Charles friends : [0, 1, 3]
        id : 3   name : Anne friends : [1, 2]
        id : 4   name : Chi friends : [5]
        id : 5   name : Amine friends : [4]
```

5. Ecrire une fonction UserProj qui accepte deux paramètres : une liste d'utilisateurs et une liste d'attributs. Elle retourne une liste représentant la projection de la liste initiale sur les attributs. Par exemple :

```
>>> userProj=projUsers(users,["id","friends"])
>>> DisplayInfo(userProj)
  id : 0   friends : [1, 2]
  id : 1   friends : [0, 2, 3]
  id : 2   friends : [0, 1, 3]
  id : 3   friends : [1, 2]
  id : 4   friends : [5]
  id : 5   friends : [4]
```

6. Ecrire une fonction *numberFriends* qui accepte une liste d'utilisateurs et retourne une liste de couples associés aux utilisateurs. Pour chaque utilisateur, il existe un couple composé de l'identificateur de l 'utilisateur et du nombre de ses amis. La liste retournée doit être triée. Par exemple :

```
>>>numberFriends(users):
    [(1, 3), (2, 3), (0, 2), (3, 2), (4, 1), (5, 1)]
```

7. Ecrire une fonction connectedFriendsWith qui accepte une liste d'utilisateurs et l'identificateur d'un utilisateur. Elle retourne les identificateurs des utilisateurs qui sont des amis de l'utilisateur, ou des amis de ses amis, …. En termes de graphes, il s'agit de trouver les utilisateurs qui sont dans la même composante connexe que l'utilisateur. Par exemple :

```
>>> connectedFriendsWith(users(1])
    [0 ,1, 2, 3]
>>>connectedFriendsWith(users[4])
    [4,5]
>>>connectedFriendsWith(users[5])
    [4,5]
```

8. On considère la liste *interests* donnant les préférences des utilisateurs. On souhaite définir des recommendations aux utilisateurs en se basant sur les thèmes prédominants des utilisateurs qui lui sont connectés. Bien évidemment, les recommendations, pour un utilisateur donné, portent sur des thèmes non encore attachés à l'utilisateur. Par exemple, la liste interests pourrait ressembler à cela :

```
>>>interests = [ (0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"), (0, "Spark"), (0,
"Storm"), (0, "Cassandra"), (1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"), (1,
"Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"), (2, "numpy"), (2, "statsmodels"), (2,
"pandas"), (3, "R"), (3, "Python"), (3, "statistics"), (3, "regression"), (3, "probability"), (4, "machine
learning"), (4, "regression"), (4, "decision trees"), (4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5,
"C++"), (5, "Haskell"), (5, "programming languages")]
```

9. Proposer des fonctions pour faire de telles recommandations.

10. Implémenter l'affichage des Figures données ci-dessous :

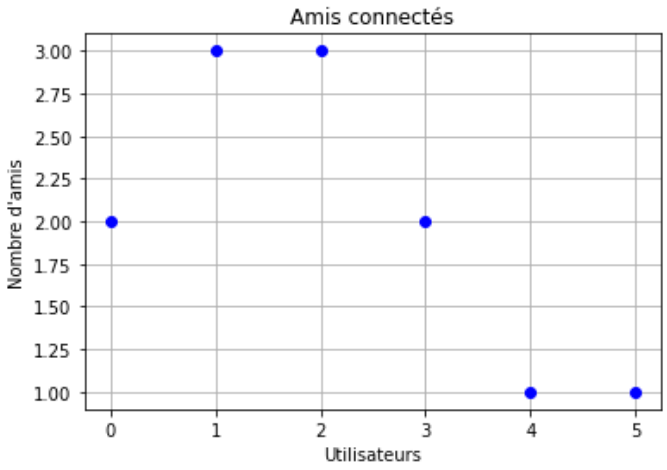

Figure 2

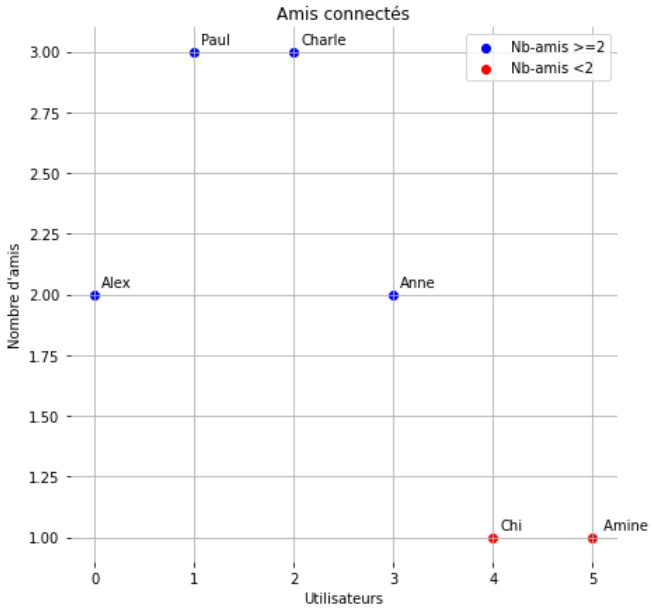

Figure 3

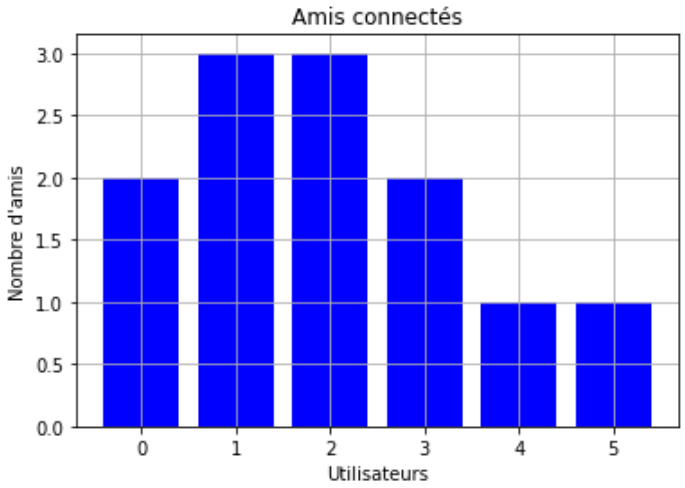

Figure 4

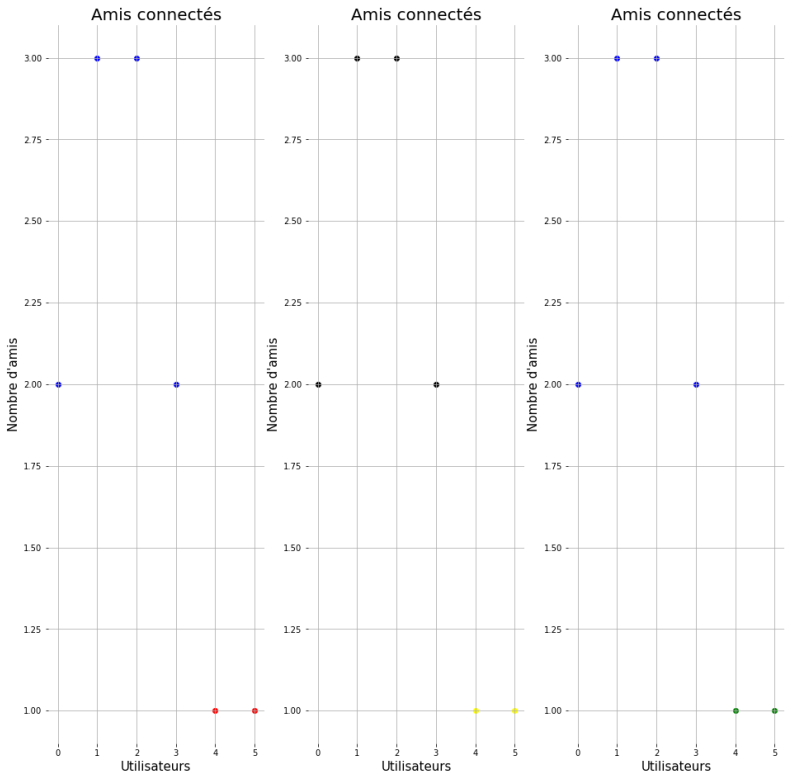
Amis connectés
Amis connectés
Amis connectés
Nombre d'amis
Utilisateurs
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
1
2
3
4
5

Figure 5