TP1 : FILTRAGE COLLABORATIF EN PYTHON

Le filtrage collaboratif est une technique de personnalisation sur les sites marchands qui permet de faire des recommandations d'achat à un internaute en comparant ses préférences sous forme d'évaluations produits (livres, disques, films, jeux,...) aux jugements portés par des internautes au profil similaire sur des produits qui n'ont pas encore été achetés par le prospect à qui est faite la recommandation.

Le terme "collaboratif" est utilisé car la technique nécessite que l'internaute ait déclaré préalablement ses goûts à travers une évaluation de produits (livres lus, films vus, etc.). Le principe est de dire "vous devriez aimer ce livre ou ce film car les internautes ayant les mêmes goûts que vous l'ont aimé".

Quel film proposé à Anne ?

Les évaluations de sept critiques cinématographiques, données entre 0 et 5 sur six films, sont reportées dans le tableau ci-dessous. Les films évalués ont pour titre : "Lady in the waters" (Lady), "Snakes on the Plane" (Snakes), "Just My Luck" (Luck), "Superman Returns" (Superman), "You, Me and Dupree" (Dupree) et "The Night Listener" (Night). Une case vide signifie que le critique n'a pas vu le film en question et donc est incapable de l'évaluer.

	Lady	Snakes	Luck	Superman	Dupree	Night
Lisa Rose	2.5	3.5	3.0	3.5	2.5	3.0
Gene Seymour	3.0	3.5	1.5	5.0	3.5	3.0
Michael Phillips	2.5	3.0		3.5		4.0
Claudia Puig		3.5	3.0	4.0	2.5	4.5
Mick Lasalle	3.0	4.0	2.0	3.0	2.0	3.0
Jack Matthews	3.0	4.0		5.0	3.5	3.0
Toby		4.5		4.0	1.0	

Anne, une étudiante dauphinoise souhaite aller au cinéma et hésite entre les trois films "Snakes", "Superman" et "Night". Ses préférences parmi les films déjà vus sont résumées par le tableau suivant :

	Lady	Snakes	Luck	Superman	Dupree	Night
Anne	1.5		4.0		2.0	

L'objectif de cet exercice est de faire des recommandations à Anne en tenant compte des notes des sept critiques.

1. Construire un dictionnaire `critiques` contenant les critiques des films et leurs notes (Le dictionnaire n'est ici qu'une suggestion de représentation des "préférences" des critiques. Vous pouvez donc choisir une autre structure de représentation de ces données).

2. Après avoir réuni les données sur les préférences des personnes, nous avons besoin d'un mécanisme permettant de déterminer celles ayant des goûts similaires à Anne. Pour cela, nous allons comparer chaque personne à toutes les autres en calculant un "score de similarité" ou "score de similitude".

   (a) Pour calculer simplement un score de similarité, nous utiliserons la distance de Manhattan ou la distance euclidienne.

   Ainsi, si n représente le nombre de films pour lesquels les critiques x et y ont attribué une note, alors le score de similarité entre x et y sera assimilé à :

   - leur distance de Manhattan $d(x, y)$ définie par la formule

   $$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

   - leur distance euclidienne $d(x, y)$ définie par la formule

   $$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

   où $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont les vecteurs notes des n films pour lesquels x et y ont attribué une note (les films pour lesquels aucune note n'a été attribuée ne seront pas pris en compte dans cette formule).

   i. Construire les fonctions `sim_distanceManhattan` et `sim_distanceEuclidienne` qui retournent respectivement un score de similarité, basées sur la distance de Manhattan et la distance euclidienne, pour deux personnes données.

   Indication : On pourra par exemple construire une fonction
   `def sim_distanceEuclidienne(person1,person2)`
   où `person1` et `person2` sont des dictionnaires contenant des notes attribuées par deux utilisateurs.

   Par exemple, le dictionnaire associé aux évaluations de Lisa Rose sera obtenu en tapant :

```
>>> critiques['Lisa Rose']
{'Lady': 2.5, 'Snake':3.5, 'Luck': 3.0, 'Superman': 3.5,
 'Dupree': 2.5, 'Night': 3.0}.
```

   Ainsi, nous obtiendrons

```
>>> sim_distanceEuclidienne(critiques['Lisa Rose'],
critiques['Gene Seymour'])
2.3979157616563596
```

   ii. Pour chacune des distances ci-dessus, construire la fonction
   `recommend(nouveauCritique, Critiques)` qui retourne la liste des films à recommander à l'utilisateur nouveauCritique en fonction des goûts des autres critiques.
   Indication : On pourra utiliser la fonction suivante, basée sur la distance de Manhattan,
   `computeNearestNeighbor(nouveauCritique, critiques)`, qui retourne une liste triée des critiques proches de nouveauCritique.

```
def computeNearestNeighbor(nouveauCritique, Critiques):
    distances=[]
    for critique in Critiques:
        if critique!=nouveauCritique:
        distance=sim\_distanceManhattan(Critiques[critique],
                          Critiques[nouveauCritique])
        distances.append((distance,critique))
    distances.sort()
    return distances
```

   En testant ces fonctions, vous obtiendrez par exemple :

```
>>> computeNearestNeighbor('Lisa Rose', Critiques)
[(1.5, 'Michael Phillips'), (2.0, 'Claudia Puig'), (2.5, 'Anne'),
(3.0, 'Mick LaSalle'), (3.0, 'Toby'), (3.5, 'Jack Matthews'),
(4.5, 'Gene Seymour')]
 >>> recommend('Lisa Rose', Critiques)
 []
 >>> recommend('Toby', Critiques)
 [('Lady', 1.5), ('Luck', 4.0)]
```

(b) La procédure de recommandation précédente permet de consulter les critiques de la personne ayant des goûts similaires à ceux de Anne et de sélectionner un film qu'elle n'a pas encore vu à partir de ces goûts, mais cette approche n'est pas très juste. En effet, elle pourrait mener à des critiques qui n'ont pas donné d'avis sur les films que Anne souhaiterait voir. Elle pourrait également désigner quelqu'un qui a aimé un film ayant eu mauvaise presse chez les autres personnes.

Pour résoudre ce problème, nous devons évaluer les éléments en produisant un score global qui classe les critiques. La procédure qui déterminera les recommandations à faire à Anne se déroulera en deux étapes. Pour la décrire, notons pour un film a donné, $\mathcal{C}(a)$ la liste des critiques ayant attribué une note à a, et $x(a)$ la note attribuée à a par le critique x.

i. **Etape 1** : Pour chaque film a non vu par Anne, calculer les quantités

$$total(a) = \sum_{x \in \mathcal{C}(a)} \frac{1}{1 + d(x, Anne)} \times x(a)$$

$$s(a) = \sum_{x \in \mathcal{C}(a)} \frac{1}{1 + d(x, Anne)}$$

$$s'(a) = \frac{total(a)}{s(a)}$$

La quantité $s'(a)$ permettra de conclure qu'une personne similaire à Anne contribuera de manière plus importante au score global qu'une personne qui lui est différente.

En utilisant la distance de Manhattan, on obtient pour le film "Night" non vu par Anne :

- $\mathcal{C}(a) = \{$Lisa Rose, Gene Seymour, Michael Phillips, Claudia Puig, Mick LaSalle, Jack Matthews$\}$
- $total(a) = \frac{4}{1+1} + \frac{4.5}{1+1.5} + \frac{3}{1+2.5} + \frac{3}{1+3} + \frac{3}{1+3.5} + \frac{3}{1+5.5} = 6.53534799$
- $s(a) = 1.81178266$
- $s'(a) = 3.6071$

ii. **Etape 2** : Le film à recommander à Anne sera le film qui aura la plus grande somme $s'(a)$.

⋆ A partir de ces explications, écrire la fonction `Bestrecommend` qui permettra de proposer à Anne une recommandation entre les trois films "Snakes", "Superman" et "Night".

⋆ En utilisant d'autres "poids" liés aux distances (poids différents de $\frac{1}{1 + d(x, Anne)}$), Écrire une autre fonction `OtherBestrec` qui permettra de proposer à Anne une recommandation entre les trois films "Snakes", "Superman" et "Night".

3. Une autre manière plus formelle de déterminer la similarité entre les préférences des personnes est d'utiliser le coefficient de correlation de Pearson. Il donne généralement de meilleures résultats que la distance euclidienne lorsque les données ne sont pas parfaitement normalisées, par exemple lorsque les critiques des films sont plus sévères que la moyenne. Ainsi, si n représente le nombre de films pour lesquels les critiques x et y ont attribué une note, alors le score de similarité entre deux critiques x et y sera assimilé à leur coefficient de correlation de Pearson $p(x, y)$ donné par la formule

$$p(x,y) = \frac{(\sum_{i=1}^{n} x_i\, y_i) - \frac{(\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n}}{\sqrt{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}} \times \sqrt{\sum_{i=1}^{n} y_i^2 - \frac{(\sum_{i=1}^{n} y_i)^2}{n}}}$$

où $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont les vecteurs notes des n films pour lesquels x et y ont attribué une note (les films pour lesquels aucune note n'a été attribuée ne seront pas pris en compte dans cette formule).

Pour visualiser cette approche, vous pouvez, par exemple, placez dans un graphique, au brouillon, les notes des deux critiques Mick LaSalle (axe x) et Gene Seymour (axe y). Par exemple le film "Superman" qui a reçu la note de 3 par Mick LaSalle et 5 par Gene Seymour, sera considéré dans le graphique comme un point de coordonnées $(3, 5)$. On peut donc remarquer que si deux critiques attribuent les mêmes notes à tous les films, alors la *droite de regression* (la droite qui se trouve aussi près que possible de tous les éléments du graphique) correspondra à une diagonale, ce qui se traduira par un coefficient de corrélation de pearson égal à 1.

Attention : Le coefficient de Pearson est une fonction croissante.

Écrire la fonction `PearsonRecommend` qui permettra de proposer à Anne, en utilisant le coefficient de Pearson, une recommandation entre les trois films "Snakes", "Superman" et "Night".

Indication : On pourra utiliser la fonction suivante qui détermine le coefficient de Pearson entre deux utilisateurs :

```
def pearson(person1, person2):
    sum_xy=0
    sum_x=0
    sum_y=0
    sum_x2=0
    sum_y2=0
    n=0
    for key in person1:
        if key in person2:
            n += 1
            x=person1[key]
            y=person2[key]
            sum_xy +=x*y
            sum_x += x
            sum_y += y
            sum_x2 += x**2
             sum_y2 += y**2
    denominator = sqrt(sum_x2 - (sum_x**2) / n) *
                  sqrt(sum_y2 - (sum_y**2) / n)
    if denominator == 0:
        return 0
    else:
        return (sum_xy - (sum_x * sum_y) /n ) / denominator
```

4. En utilisant comme mesure de similarité la similarité cosinus, écrire la fonction `CosinusRecommend` qui permettra de proposer à Anne, une recommandation entre les trois films "Snakes", "Superman" et "Night". On rappelle :

$$cos(x, y) = \frac{\sum_{i=1}^{n} x_i \, y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}$$

où $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont les vecteurs notes des n films pour lesquels x et y ont attribué une note. La fonction cosinus est une fonction de similarité croissante.

5. Construire un exemple avec au moins 20 films et 15 critiques où la même recommandation (pour une personne qui n'aura pas vu au moins la moitié des films) est obtenue à partir d'au moins 6 mesures de similarité distinctes. Le nombre de cases (cellules) vides de votre matrice d'évaluation sera compris entre 30% et 60% .

   Une fonction python, `pourcentagecasesvides`, indiquant le pourcentage de cases vides obtenue sera implémentée.

Vous proposerez, à travers un texte court généré automatiquement, à la personne (critique) supposée être ni informaticienne, ni mathématicienne, une explication d'une de ces recommandations (la recommandation qui lui sera proposée à la fin du processus).

6. Est-il possible de construire un exemple avec au moins 20 films et 15 critiques où les recommandations (pour une personne qui n'aura pas vu au moins la moitié des films) faites, à partir d'au moins 6 mesures de similarité distinctes, sont toutes différentes entre elles ? Le nombre de cases (cellules) vides de votre matrice d'évaluation étant compris entre 30% et 60% .

   La fonction `pourcentagecasesvides` implémentée à la question précédente indiquera le pourcentage de cases vides obtenue.

   Vous proposerez, à travers un texte court généré automatiquement, à la personne (critique) supposée être ni informaticienne, ni mathématicienne, une explication d'une de ces recommandations (la recommandation qui lui sera proposée à la fin du processus).

N.B : Autres précisions

⋆ Le TP1 sera rendu, **sous forme de fichier(s) .py**, dans le canal teams de chaque groupe, au plus tard le **10/10/24 à 23h59.**

⋆ Chaque groupe présentera à la séance 3, sur un ou plusieurs slides, l' exemple construit (les 20 films et 15 critiques), les recommandations obtenues et l'explication proposée pour la question 6.

La démarche utilisée pour générer l'exemple devra être précisée, ainsi que le pourcentage de données manquantes dans le tableau généré.

⋆ La résolution des questions 5 et 6, comptera pour 75% de la note du TP1.