UNIVERSITÉ DE SHERBROOKE

Faculté des sciences

Département d'informatique

IFT799 – Science des données

Rapport

TP2_3 : Segmentation des données

JEU DE DONNÉES ABR

RÉALISÉ PAR:

KAOUTAR L'HASSNAOUI 22 148 702

ZINEB EL YAMANI 22 142 129

ENCADRÉ PAR: SHENGRUI WANG

ETIENNE GAEL TAJEUNA

Abstract

Sur Amazon, les clients peuvent laisser leurs opinions sur des livres. Ce feedback s'avère être une source d'information avec beaucoup de potentiel. En effet, comparé au marketing de masse, proposer une expérience personnalisée peut se trouver très utile pour augmenter les ventes.

A partir des reviews effectuées par un internaute sur des livres donnés, on veut savoir quel serait l'opinion de cet internaute sur un autre livre. Ainsi, on devrait réaliser une prédiction basée sur les méthodes de système de recommandation et des algos de clustering.

Ce TP2_3 a pour but de segmenter nos données et d'en visualiser les résultats pour les preparer pour notre analyse predictive.

Table de matières

1.Introduction

2.Dataset

3.Méthodologie proposée dans le TP

4.Implémentation et analyse

4.1. partie (a)

4.2. partie (b)

5.Conclusion

6.Ressources

1.Introduction

Maintenant qu'on a une idée générale de nos données, on veut pousser notre exploration à la recherche de sous structures pertinentes.

Le TP1 nous a permis de comprendre nos données de façon globale grâce aux outils statistiques et à des visualisations après une projection en composantes principales.

Pour ce TP, on va utiliser différentes approches de clustering tout en visualisant et en évaluant les segmentations obtenues pour enfin sortir avec celle la plus représentative et utile pour notre future analyse prédictive.

2.Dataset

Comme pour le TP1 on travaille avec les 50000 premières lignes de la base données 'ABR.json'. Notons que l'on garde cet aspect aléatoire puisque la base de donnée initiale ne présente aucun ordre sur ces lignes.

Elle est composée de 9 colonnes dont la description est la suivante :

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A10000012B7CGYKOMPQ4L	000100039X	Adam	[0, 0]	Spiritually and mentally inspiring! A book tha...	5	Wonderful!	1355616000	12 16, 2012
1	A2S166WSCFIFP5	000100039X	adead_poet@hotmail.com "adead_poet@hotmail.com"	[0, 2]	This is one my must have books. It is a master...	5	close to god	1071100800	12 11, 2003
2	A1BM81XB4QHOA3	000100039X	Ahoro Blethends "Seriously"	[0, 0]	This book provides a reflection that you can a...	5	Must Read for Life Afficianados	1390003200	01 18, 2014
3	A1MOSTXNIO5MPJ	000100039X	Alan Krug	[0, 0]	I first read THE PROPHET in college back in th...	5	Timeless for every good and bad time in your l...	1317081600	09 27, 2011
4	A2XQ5LZHTD4AFT	000100039X	Alaturka	[7, 9]	A timeless classic. It is a very demanding an...	5	A Modern Rumi	1033948800	10 7, 2002
...	...	...	...	...	...	...	...	...	...
49995	A2R2VN5X77D66O	0028633873	Donna H.	[0, 0]	i gave this as a gift so i don't know how good...	5	bought as a gift	1402704000	06 14, 2014
49996	A38UCPTYS6LBHE	0028633873	Lindsay Harrison "film and book aficionado"	[22, 23]	I have been attempting to learn Hebrew, but I ...	5	For the Jew or the Goy	1051488000	04 28, 2003
49997	A2UENE1PINAKCT	0028633873	Marian E. Wells	[0, 0]	Wanted to know more about Yiddish and this boo...	5	YIDDISH for a Gentile?	1388620800	01 2, 2014
49998	ALA77HERW2U0J	0028633873	Michael Peterson	[1, 3]	While searching for a book that explains the Y...	5	This Yiddish book is a Feast for the Senses	1198627200	12 26, 2007
49999	A1LH5914M6CLTH	0028633873	Piet F. Van Allen	[2, 14]	It's a series - not to worry.Say, there's a "B...	3	Complete Idiot?	1181606400	06 12, 2007

50000 rows × 9 columns

3.Méthodologie proposée dans le TP:

Ce TP va nous permettre de voir la variation des segmentations avec la variation des méthodes de clustering et des métriques utilisées.

Le TP suit les étapes suivantes :

1.Clustering avec k-moyennes
    1.1.Distance euclidienne
    1.2.Similarite cosinus

2.ACP + K means
    2.1.Distance euclidienne
    2.2.Similarite cosinus

3.Clustering spectale
    2.1.Distance euclidienne
    2.2.Similarite cosinus

3.Evaluation des resultats

4.Echantillonage stratifié

a)

4.Implementation et analyse

Question 1 :

On prend notre échantillon de données comportant 50000 lignes (reviews) et on regroupe par les livres. On se retrouve ainsi avec 955 livres. En se référant aux critères d'appréciation donné dans l'énoncé du TP, la répartition est la suivante :

	000100039X	0001055178	0001473123	0001473727	0001473905	0001712772	000171287X	0001714538
Nbr Non Apprécié	10.0	4.0	1.0	0.0	0.0	1.0	0.0	2.0
Nbr Neutre	8.0	2.0	0.0	0.0	1.0	0.0	2.0	0.0
Nbr Apprécié	188.0	12.0	15.0	7.0	5.0	14.0	10.0	5.0
Nbr de votes	206.0	18.0	16.0	7.0	6.0	15.0	12.0	7.0

4 rows × 955 columns

Pour nous aider a mieux voir cette repartition, on calcule la moyenne, l'ecart type et la mediane des scores pour chaque livre :

	mean	std	median
asin			
000100039X	4.674757	0.875712	5.0
0001055178	3.555556	0.983524	4.0
0001473123	4.625000	1.024695	5.0
0001473727	5.000000	0.000000	5.0
0001473905	4.666667	0.816497	5.0
...	...	...	...

En concaténant ces deux matrices, on se retrouve avec la matrice X demandée dans la première question avec c =7 caractéristiques statistiques :

	Nbr_Non_Apprécié	Nbr_Neutre	Nbr_Apprécié	Nbr_de_votes	Moyenne	Écart_type	Médiane
000100039X	10.0	8.0	188.0	206.0	4.674757	0.875712	5.0
0001055178	4.0	2.0	12.0	18.0	3.555556	0.983524	4.0
0001473123	1.0	0.0	15.0	16.0	4.625000	1.024695	5.0
0001473727	0.0	0.0	7.0	7.0	5.000000	0.000000	5.0
0001473905	0.0	1.0	5.0	6.0	4.666667	0.816497	5.0
...	...	...	...	...	...	...	...
0028632613	3.0	4.0	11.0	18.0	3.722222	1.363626	4.0
0028632753	3.0	2.0	0.0	5.0	2.400000	0.547723	2.0
0028633504	2.0	0.0	22.0	24.0	4.625000	0.875388	5.0
0028633784	0.0	1.0	6.0	7.0	4.428571	0.786796	5.0
0028633873	0.0	1.0	7.0	8.0	4.625000	0.744024	5.0

955 rows × 7 columns

Question 2 :

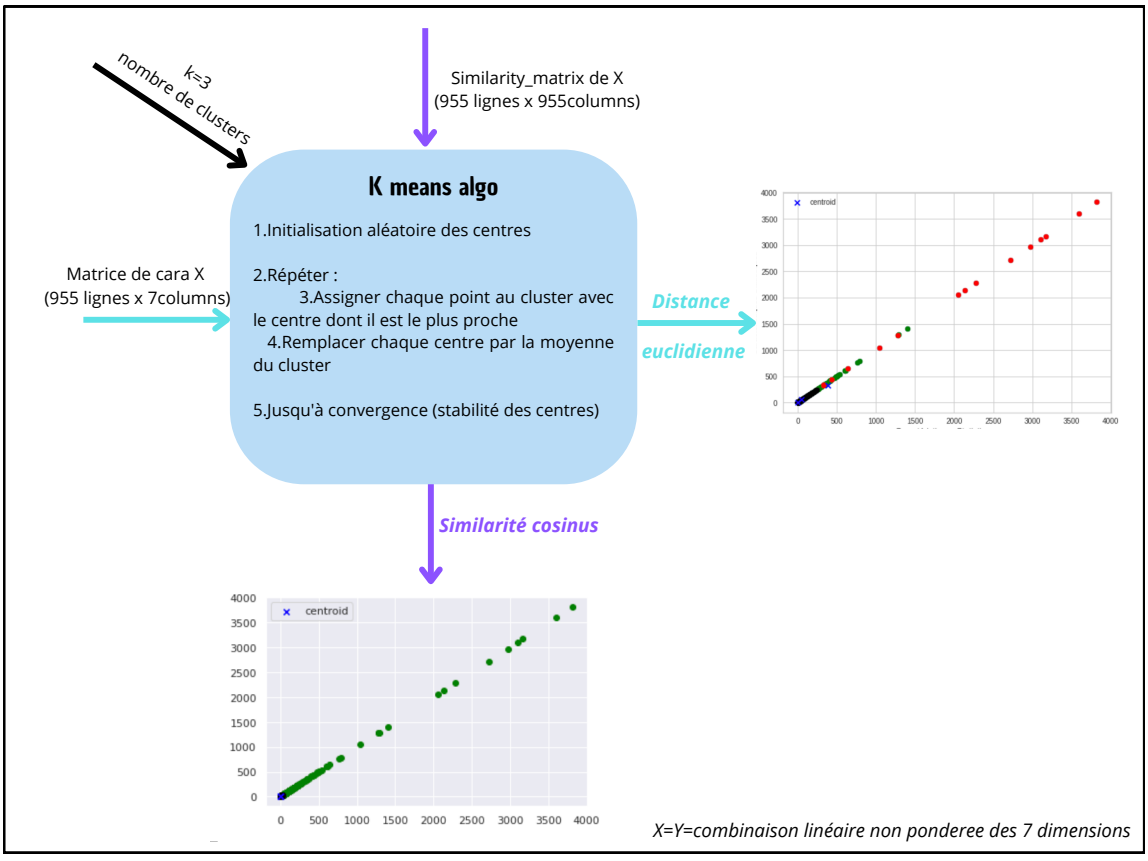
Vient l'étape du clustering où on va chercher à construire des groupes de livres à partir de nos 955 livres.

Dans cette question on adopte l'algorithme des **k-moyennes**.
Intuitivement, choisir k=3 clusters va avec notre analyse (3 types d'appréciations).
En outre, pour mesurer la similarités entre nos livres, on utilise la distance euclidienne dans un premier temps puis on utilise la similarité cosinus.

Etant donne k=3 clusters, l'algorithme des k-moyennes va attribuer à chaque livre 0 ou 1 ou 2. Chaque cluster est représenté par son centre tel que chaque livre est dans le cluster du centre dont il est le plus proche.

Comment l'algorithme determine-t il les 3 centres ?

L'agorithme au début choisit aleatoirement 3 points aleatoires comme centres. Ainsi, on construit 3 clusters initiaux, en attribuant a chaque livre le cluster avec le centre dont il est le plus proche. Pour chacun des clusters qu'on vient de former, on calcule la moyenne. Celle-ci devient le nouveau centroïde (notons qu'elle n'est pas nécessairement un point du jeu de donnée). On recommence jusqu'à ce qu'à ce qu'il y ait convergence : La convergence correspond au fait que les centroïdes ne changent pas après une mise à jour.
la distance utilisée par kmeans par défaut est la distance euclidienne



Analyse des résultats:

Distance euclidienne:

La mesure de similarité implémenté par défaut par la fonction KMeans() est la distance euclidienne. Ainsi, on peut directement l'appliquer.

Cluster 0

Centre :

```
[5.18000000e+01 5.32400000e+01 3.81120000e+02 4.86160000e+02
 4.21525756e+00 1.03760338e+00 4.64000000e+00]
```

Nombre d'elements : 934

Cluster 1

Centre :

```
[3.87000000e+02 3.29800000e+02 2.35620000e+03 3.07300000e+03
 4.14675880e+00 1.08470192e+00 4.60000000e+00]
```

Nombre d'elements : 4

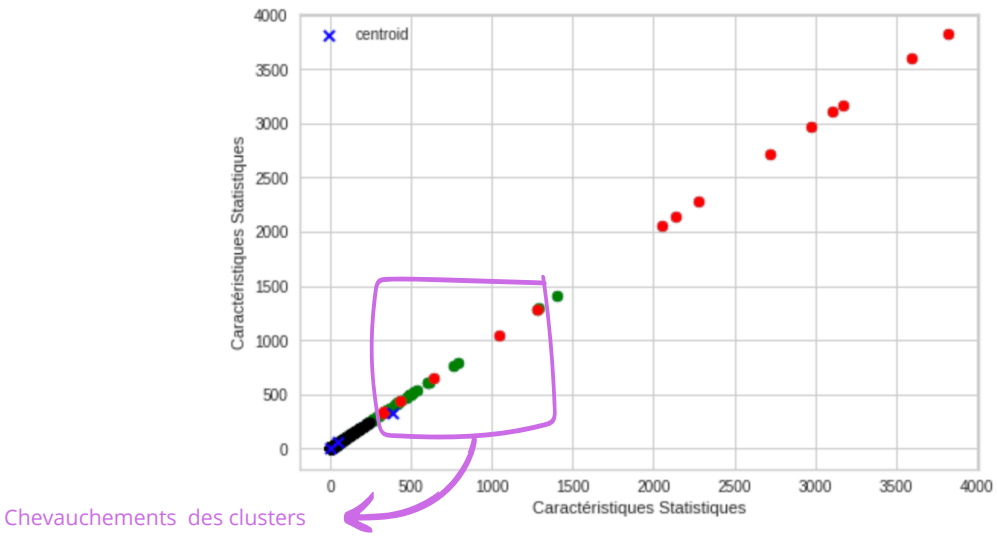
Cluster 2

Centre :

```
[2.10918919e+00 2.66810811e+00 1.95264865e+01 2.43037838e+01
 4.25460786e+00 9.12165402e-01 4.53729730e+00]]
```

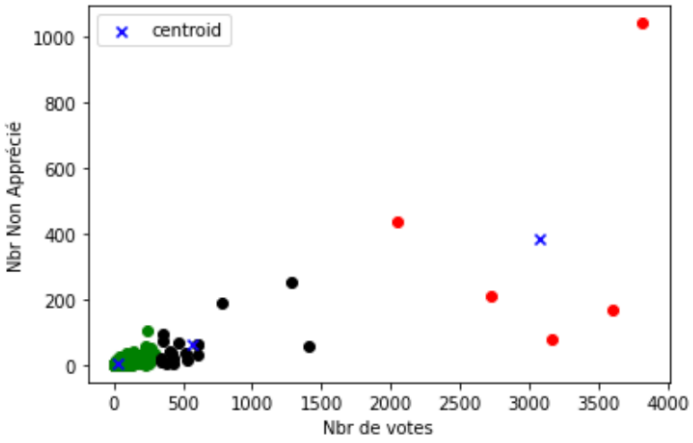
Nombre d'elements : 17

Que l'on peut visualiser grace a ce nuage de points coloré :



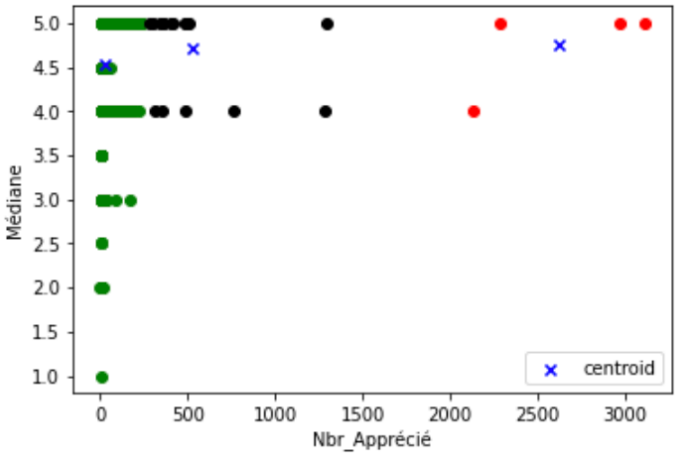
On remarque que nos clusters se chevauchent et les centres sont proches. Notons aussi que le cluster 0 comporte 934 points sur 955 points. Ce cluster est trop dense, on peut même le voir sur la figure.

Question de découvrir comment varie la segmentation en fonction de nos caractéristiques statistiques:



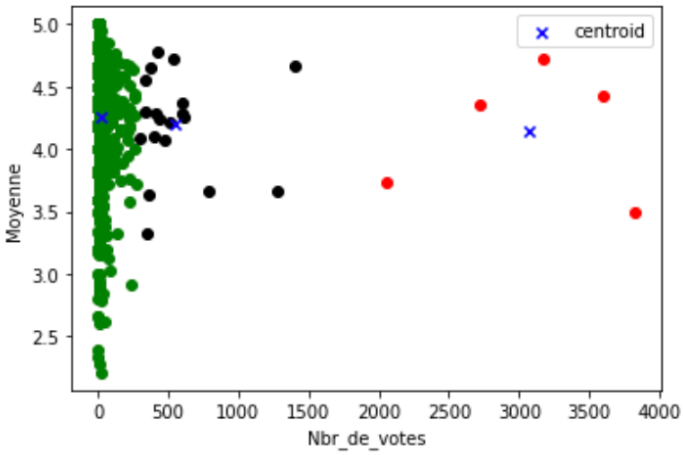
K-means en utilisant la distance euclidienne avec
(Nbre de votes, Nbr Non Apprecie)

On remarque que les chevauchements entre clusters a diminué mais est toujours présent.



K-means en utilisant la distance euclidienne avec
(Nbre Apprécié, médiane)

Moins de chevauchements



K-means en utilisant la distance euclidienne avec
(Nbr de votes, Moyenne)

On remarque que le chevauchement entre clusters a beaucoup diminué et nos données sont plus visibles.

On remarque que la projection selon chaque deux dimensions différentes donne des résultats différents. La visualisation du nombre de fois que le livre a été apprécié en fonction de la médiane des appréciations n'a pas donné un résultat de clustering intéressant. Par contre celui de la moyenne, nbr de votes est plus significatif vu la corrélation entre ces deux variables.

Similarité cosinus :

Pour appliquer l'algorithme des K-moyennes avec une similarité cosinus, on devra calculer la matrice de similarité par nous même en utilisant la similarite cosinus:

similarity_matrix = cosine_similarity(Car_statF.T.iloc[0:7].T)

On se retrouve donc avec la matrice 955x955 comme suit :

	0	1	2	3	4	5	6	7	8	9	...	945	946	947	948	949	950	951	952	953	
0	1.000000	0.953382	0.960727	0.826020	0.763804	0.954758	0.921034	0.831921	0.910293	0.697518	...	0.967331	0.940599	0.978499	0.870416	0.832924	0.939450	0.566980	0.982620	0.820325	0.8
1	0.953382	1.000000	0.973181	0.895221	0.871368	0.972541	0.965574	0.940466	0.963296	0.825946	...	0.982206	0.969839	0.985109	0.953092	0.901042	0.994066	0.780309	0.977481	0.907503	0.9
2	0.960727	0.973181	1.000000	0.947133	0.907375	0.999774	0.984940	0.942467	0.946283	0.862542	...	0.998528	0.997759	0.995536	0.967469	0.952399	0.959684	0.656123	0.995158	0.941953	0.9
3	0.826020	0.895221	0.947133	1.000000	0.984779	0.953079	0.966938	0.959043	0.903740	0.967375	...	0.933760	0.965018	0.918062	0.980244	0.998948	0.886661	0.675853	0.912704	0.991240	0.9
4	0.763804	0.871368	0.907375	0.984779	1.000000	0.915604	0.952465	0.961218	0.906842	0.995185	...	0.893878	0.931900	0.876928	0.978031	0.985921	0.876905	0.739705	0.863996	0.995538	0.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
950	0.939450	0.994066	0.959684	0.886661	0.876905	0.959353	0.967915	0.930721	0.983378	0.834699	...	0.967829	0.957122	0.973903	0.951788	0.892888	1.000000	0.802929	0.962179	0.910219	0.9
951	0.566980	0.780309	0.656123	0.675853	0.739705	0.663945	0.717956	0.807254	0.774020	0.742199	...	0.674322	0.679831	0.670974	0.769877	0.676717	0.802929	1.000000	0.638626	0.731077	0.7
952	0.982620	0.977481	0.995158	0.912704	0.863996	0.992987	0.969791	0.915444	0.939623	0.811248	...	0.997285	0.986912	0.998775	0.942517	0.918405	0.962179	0.638626	1.000000	0.906534	0.9
953	0.820325	0.907503	0.941953	0.991240	0.995538	0.948339	0.976028	0.969776	0.933028	0.981743	...	0.930707	0.960719	0.917239	0.991150	0.993469	0.910219	0.731077	0.906534	1.000000	0.9
954	0.852198	0.925676	0.959730	0.992151	0.988482	0.964963	0.986346	0.970691	0.944776	0.968947	...	0.949722	0.974831	0.938431	0.994544	0.994503	0.926337	0.722097	0.929423	0.998192	1.0

955 rows × 955 columns

Maintenant a l'entrée de notre fonction KMeans(), on mettra cette similarity_matrix.
On obtient 3 clusters définis comme suit :

Cluster 0

Centre :

```
[0.97585507 0.97010758 0.95900046 ... 0.97374537 0.85378694 0.87956948]
```

Nombre d'elements : 293

Cluster 1

Centre :

```
[0.75706813 0.87008576 0.89495502 ... 0.85468503 0.9742303  0.96798992]
```

Nombre d'elements : 296

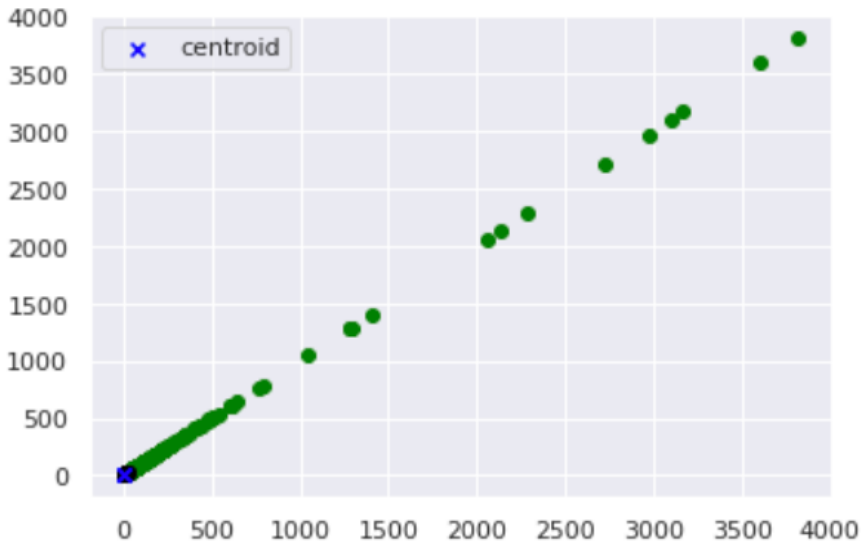
Cluster 2

Centre :

```
[0.91304389 0.9615315  0.9783395  ... 0.96405286 0.96298044 0.97360481]
```

Nombre d'elements : 366

Les coordonnees sont donnees dans la similarity_matrix ->955 coordonnees/colonne pour chaque point



on remarque que ces clusters sont chevauchés

On rajoute a notre matrice X les résultats :

Les clusters avec la similarite cosinus

Les clusters avec la distance euclidien

	Nbr_Non_Apprécié	Nbr_Neutre	Nbr_Apprécié	Nbr_de_votes	Moyenne	Écart_type	Médiane	Cluster_eu	Cluster_cos
000100039X	10.0	8.0	188.0	206.0	4.674757	0.875712	5.0	0	0
0001055178	4.0	2.0	12.0	18.0	3.555556	0.983524	4.0	0	0
0001473123	1.0	0.0	15.0	16.0	4.625000	1.024695	5.0	0	0
0001473727	0.0	0.0	7.0	7.0	5.000000	0.000000	5.0	0	1
0001473905	0.0	1.0	5.0	6.0	4.666667	0.816497	5.0	0	1
...	...	...	...	...	...	...	...	...	...
0028632613	3.0	4.0	11.0	18.0	3.722222	1.363626	4.0	0	0
0028632753	3.0	2.0	0.0	5.0	2.400000	0.547723	2.0	0	2
0028633504	2.0	0.0	22.0	24.0	4.625000	0.875388	5.0	0	0
0028633784	0.0	1.0	6.0	7.0	4.428571	0.786796	5.0	0	1
0028633873	0.0	1.0	7.0	8.0	4.625000	0.744024	5.0	0	1

955 rows × 10 columns

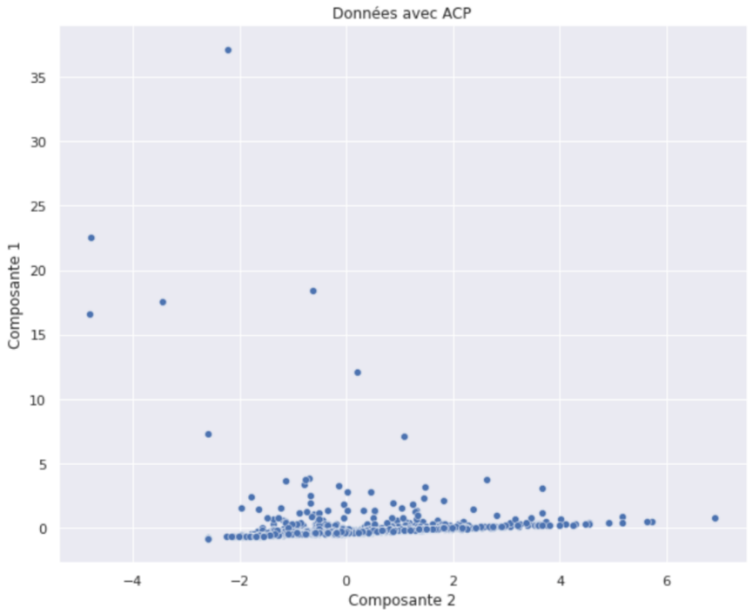
La segmentation differe en utilisant des mesures de similarites differentes pour le meme algorithme

Question 3:

Vu que nos données sont multivariées, utiliser une ACP va nous permettre de prendre en consideration l'apport de chaque dimension.
L'ACP va extraire les informations importantes et les synthétiser en un plus petit nombre de dimensions appelées composantes principales. Ces nouvelles variables correspondent à une combinaison linéaire des variables initiales.

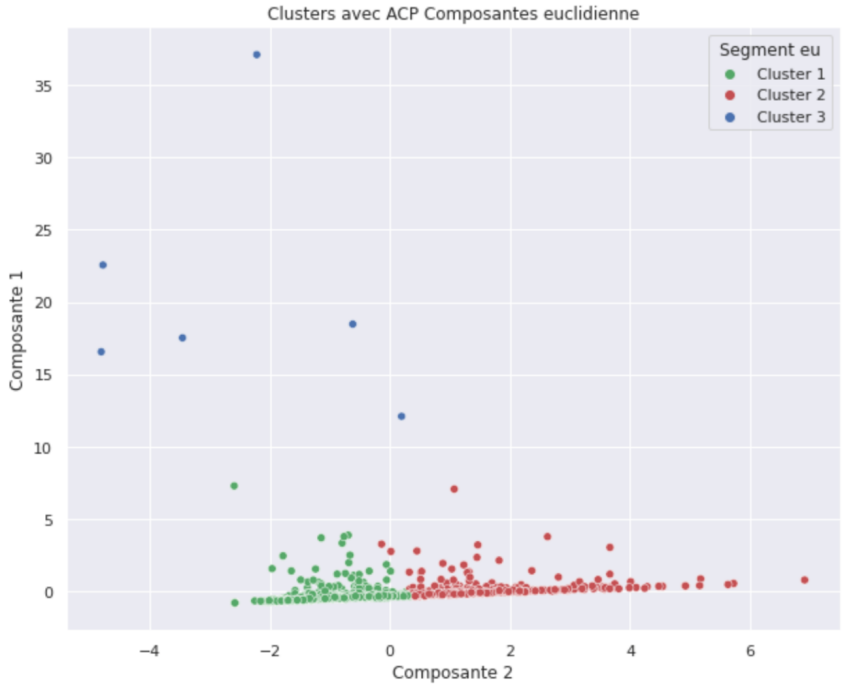
La projection de nos données dans les deux composantes principales est comme suit :

Composante 1	Composante 2
0.645914	-1.186084
-0.017748	1.529514
-0.385971	-0.706228
-0.775024	-2.582763
-0.488315	-1.034011
...	...
0.066736	1.823155
0.226189	4.242402
-0.377949	-0.917351
-0.443526	-0.760768
-0.489327	-1.079477

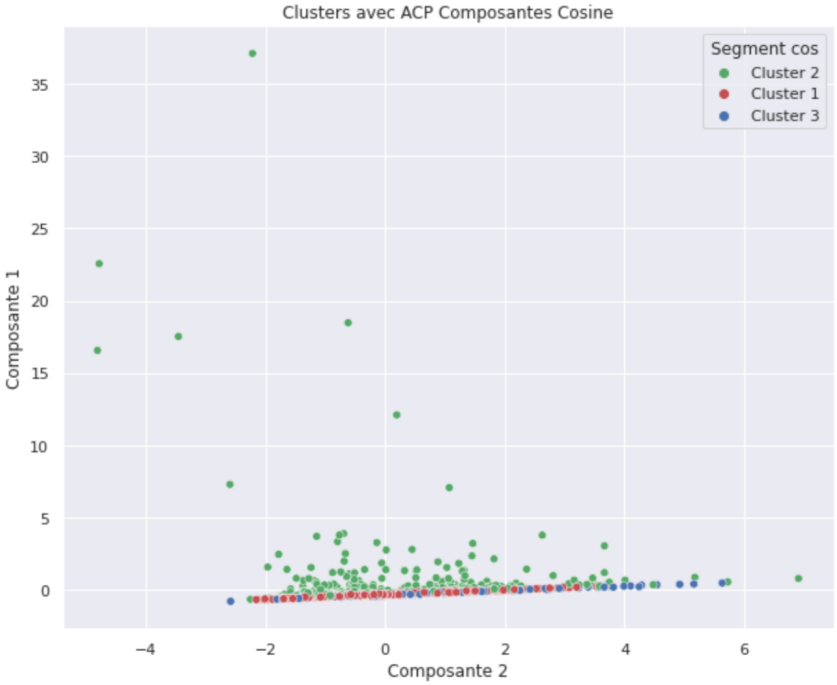


On applique ensuite l'algorithme des K-moyennes:

K-moyennes avec distance euclidienne



K-moyennes avec similarite cosinus



On confirme encore une fois que la segmentation est sensible à la mesure de similarité. Pour le même algorithme, en changeant de métriques, on retrouve deux segmentations différentes. L'ACP nous a permis de prendre en compte les l'apport de chaque dimension pour la segmentation et de mieux visualiser nos clusters. Toutefois, la segmentation avec la distance cosine représente un chevauchement entre les clusters.

Question 4:

Comme on a vu précédemment, l'algorithme des K-moyennes va construire des clusters centrés. Le spectral clustering est plus flexible. En effet il ne va pas imposer la forme des clusters.
Il va utiliser les valeurs propres de la matrice de similarité pour faire une réduction de dimensions avant de faire du clustering en moins de dimensions.

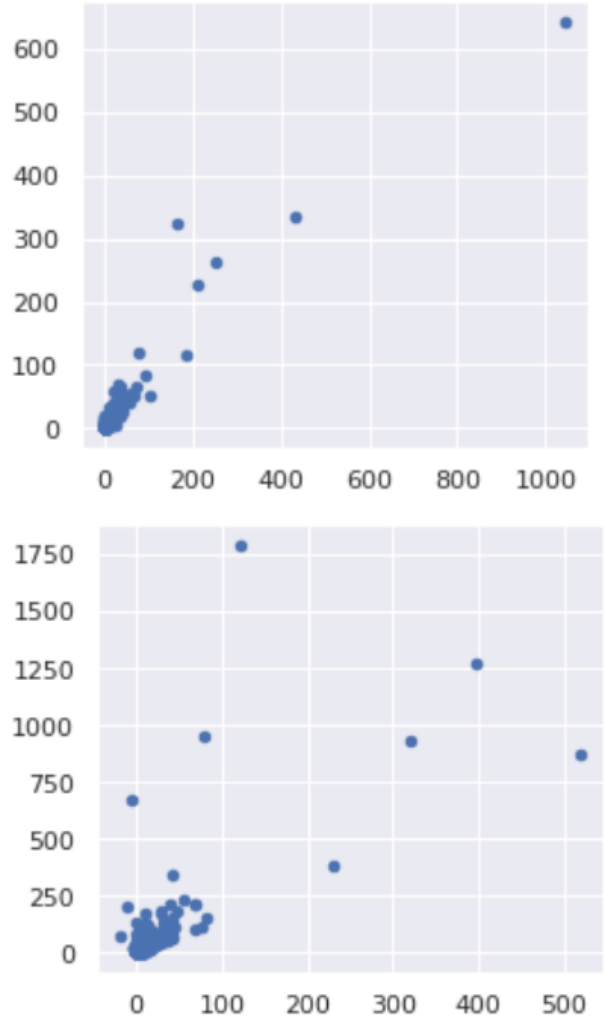
On commence par réduire les dimensions à 2 en forçant nos données en une distribution anisotropique :

```
X= Car_statF.iloc[:,0:6]
print(X) # composée de 2 dimensions et 955 lignes de livres
transformation = [[0.60,-0.63],[-0.40,0.85]]*3
plt.figure(figsize=(4,8))
plt.subplot(211)
plt.scatter(X.T.iloc[0],X.T.iloc[1],s=20)
X_aniso=np.dot(X,transformation) #distribution anisotropique
print(X_aniso)
plt.figure(figsize=(4,8))
plt.subplot(211)
plt.scatter(X_aniso[:,0],X_aniso[:,1],s=20)
```

```
            Nbr_Non_Apprécié  Nbr_Neutre  Nbr_Apprécié  Nbr_de_votes
000100039X              10.0         8.0         188.0         206.0
0001055178               4.0         2.0          12.0          18.0
0001473123               1.0         0.0          15.0          16.0
0001473727               0.0         0.0           7.0           7.0
0001473905               0.0         1.0           5.0           6.0
...                      ...         ...           ...           ...
0028632613               3.0         4.0          11.0          18.0
0028632753               3.0         2.0           0.0           5.0
0028633504               2.0         0.0          22.0          24.0
0028633784               0.0         1.0           6.0           7.0
0028633873               0.0         1.0           7.0           8.0

             Moyenne  Écart_type
000100039X  4.674757    0.875712
0001055178  3.555556    0.983524
0001473123  4.625000    1.024695
0001473727  5.000000    0.000000
0001473905  4.666667    0.816497
...              ...         ...
0028632613  3.722222    1.363626
0028632753  2.400000    0.547723
0028633504  4.625000    0.875388
0028633784  4.428571    0.786796
0028633873  4.625000    0.744024

[955 rows x 6 columns]
[[35.65456948 54.9592583 ]
 [ 3.33992357  5.51599575]
 [ 5.56512197  1.47724082]
 ...
 [ 7.22484475  3.1103299 ]
 [ 2.74242454  0.89877642]
 [ 3.07739048  0.95867024]]
<matplotlib.collections.PathCollection at 0x7f2d34c7cc90>
```



Les étapes du clustering spectral :

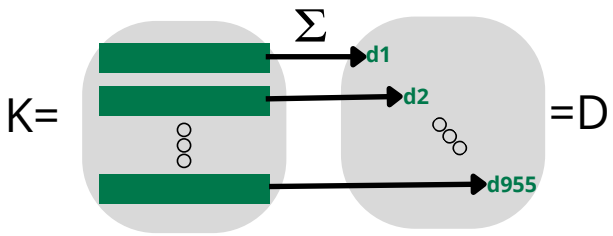
1.La matrice de similarité :

$$K(i,j) = \exp\left(-\frac{|x_i - x_j|^2}{2\sigma}\right), \quad K(i,i) = 0$$

```
rbf_param =7.6 #hyperparapetre pour accélerer la convergence de 2 point
#trouver M
from scipy.spatial import distance
K =np.exp(-rbf_param*distance.cdist(X,X,metric='sqeuclidean')) #similarity between each 2 nodes
```

2.The degree matrix D:

D=K.sum(axis=1)



3.The normalized graph Laplacien :

$$M = D^{-\frac{1}{2}} K D^{-\frac{1}{2}}$$

```
D=np.sqrt(1/D)
M=np.multiply(D[np.newaxis,:],np.multiply(K,D[:,np.newaxis]))
```

4.Trouver les 3 vecteurs propres

```
P,D,Pt=linalg.svd(M, full_matrices=True, lapack_driver='gesvd')
Psubset = P[:,0:3] #nbr of eigenvectors = nbr of clusters
```

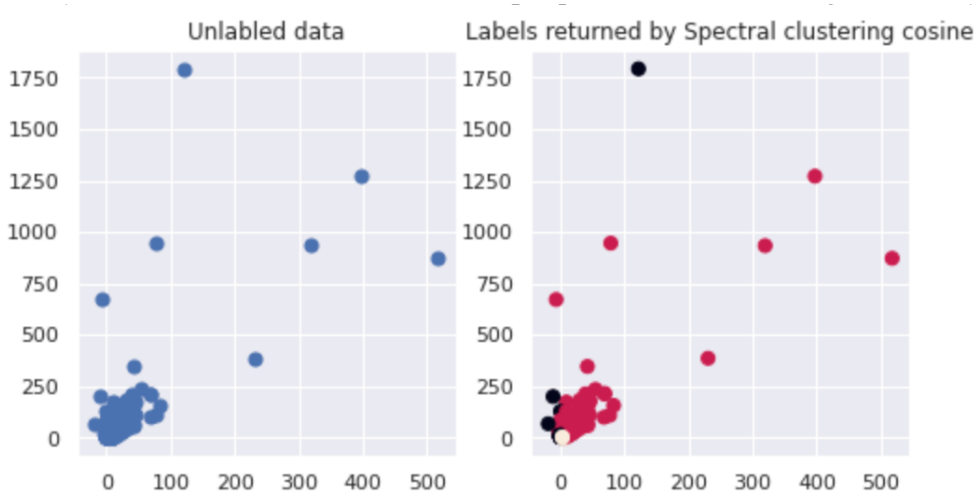
5.Normalisation $x_i \leftarrow \frac{x_i}{|x_i|}$

6.Treating each row as a data point and cluster the data into k clusters via kmeans

```
y_pred_sc=KMeans(n_clusters=3).fit_predict(normalize(Psubset))
```

Distance euclidienne :

Similarité cosinus :



Pour obtenir la matrice de similarité pour la distance cosinus, il suffit de changer de métrique dans la fonction distance qui est utilisée dans le calcul de cette matrice.

On remarque donc que le clustering spectral est visiblement plus flexible à la forme des clusters .
En utilisant les deux distances, on a pu récupéré des clusters allongés par contre au KMeans qui se restreint à des formes circulaires.

Toutefois la segmentation obtenue n'est pas trop representative; Il apparait que le clustering spectral dans ce cas ne donne pas plus d'information correcte que KMeans; chose qu'on va peut être confirmer avec le calcul de l'information mutuelle.

Question 5:

Pour évaluer les résultats de clustering vus jusqu'ici, nous allons utiliser 2 mesures :

- la silhouette
- l'information mutuelle

sklearn.metrics.silhouette_score()

Le score de silhouette prend en compte la distance intra-groupe entre l'échantillon et les autres points de données dans le même groupe (a) et la distance inter-groupe entre l'échantillon et le groupe le plus proche (b). Il est situe dans l'intervalle [-1, 1] de façon a ce que plus on s'approche de 1 plus nos clusters sont denses et bien séparés. Un score nul signifie que nos clusters se chevauchent. Un score negatif signife que les données appartenant aux clusters peuvent être fausses/incorrectes.

sklearn.metrics.mutual_info_score()

L'information mutuelle est une mesure de la similarité entre deux étiquettes des mêmes données. Le score obtenu est entre 0 et 1 tel que plus l'information mutuelle est grande, plus la segmentation est meilleure. Elle est symétrique.

La table de scores obtenue est la suivante :

	distance euclidienne silhouette	distance euclidienne info. mut	similarité cosinus silhouette	similarité cosinus info. mut
KMeans	0.931572	0.010142	0.668322	0.200047
Spectrale	0.713882	0.006375	0.084081	0.161104

On remarque qu'avec les deux mesures de similarité, K means a les plus hauts scores au niveau de l'info mutuelle qu'il offre et de la silhouette.

b)

Question 1:

L'échantillonnage stratifié est une technique d'échantillonnage utilisée pour obtenir des échantillons qui représentent le mieux la population. Il réduit le biais dans la sélection des échantillons en divisant la population en sous-groupes homogènes appelés strates et en échantillonnant au hasard les données de chaque strate .

On s'attendait à une question pareille vu que la proportionnalité des catégories dans notre clustering précédant diffèrent et ceci cause un déséquilibre au niveau des clusters.
Et donc pour répondre le choix aléatoire des données n'est pas une bonne méthode à suivre.

```
(df['overall'].value_counts())/ len(df)*100
```

```
5    56.416
4    22.326
3    10.896
2     5.706
1     4.656
Name: overall, dtype: float64
```

```
(T['Classe'].value_counts()) / len(T) * 100
```

```
0.0    91.727749
2.0     7.853403
1.0     0.418848
Name: Classe, dtype: float64
```

Ci dessus les proportionnalités de chaque score dans notre dataset ainsi que la proportionnalité de chaque catégorie dans notre dataset tel que :
2 : apprécié
1 : neutre
0 : Non Apprécié

Question 2:

En suivant les instructions du prof on se trouve donc en face de 3 étapes:

1.Construire un DataFrame qui comptabilise le nombre de fois qu'un livre est apprécié dans une catégorie comme suit :

	Nbr_Non_Apprécié	Nbr_Neutre	Nbr_Apprécié
000100039X	10.0	8.0	188.0
0001055178	4.0	2.0	12.0
0001473123	1.0	0.0	15.0
0001473727	0.0	0.0	7.0
0001473905	0.0	1.0	5.0
...	...	...	...
0028632613	3.0	4.0	11.0
0028632753	3.0	2.0	0.0
0028633504	2.0	0.0	22.0
0028633784	0.0	1.0	6.0
0028633873	0.0	1.0	7.0

955 rows × 3 columns

2. Ensuite prendre le maximum et associé la catégorie correspondante (tel que dans un vote de majorité) comme suit :

	Catégorie
000100039X	Nbr_Apprécié
0001055178	Nbr_Apprécié
0001473123	Nbr_Apprécié
0001473727	Nbr_Apprécié
0001473905	Nbr_Apprécié
...	...
0028632613	Nbr_Apprécié
0028632753	Nbr_Non_Apprécié
0028633504	Nbr_Apprécié
0028633784	Nbr_Apprécié
0028633873	Nbr_Apprécié

955 rows × 1 columns

0006481353	Nbr_Apprécié
000648204X	Nbr_Neutre
0006482791	Nbr_Apprécié
0006482821	Nbr_Apprécié
0006485944	Nbr_Apprécié
0006490603	Nbr_Apprécié
0006492460	Nbr_Apprécié
000649319X	Nbr_Apprécié
0006493599	Nbr_Apprécié
0006496911	Nbr_Apprécié
000649692X	Nbr_Apprécié
0006497802	Nbr_Apprécié
0006498000	Nbr_Apprécié
0006498051	Nbr_Apprécié
0006498604	Nbr_Apprécié
000649885X	Nbr_Apprécié
0006499163	Nbr_Apprécié
0006499317	Nbr_Apprécié
0006499708	Nbr_Apprécié
0006499945	Nbr_Apprécié
0006510019	Nbr_Neutre
0006511252	Nbr_Apprécié
0006511775	Nbr_Apprécié
0006512070	Nbr_Apprécié
0006512275	Nbr_Apprécié
0006512666	Nbr_Apprécié
0006513379	Nbr_Apprécié
0006513778	Nbr_Apprécié
0006514006	Nbr_Apprécié
000651412X	Nbr_Apprécié
0006514219	Nbr_Apprécié
0006514359	Nbr_Non_Apprécié

3.Vu que nous avons initialement commencer à un nombre de p livres qui correspondrait à une proportion de x% de la totalité des livres, nous devrons choisir dans chacune des catégories générales x% de livres comme suit :

	Nbr_Non_Apprécié	Nbr_Neutre	Nbr_Apprécié	Nbr_de_votes	Moyenne	Écart_type	Médiane	Catégori
0007175175	1.0	1.0	23.0	25.0	4.68	0.748331	5.0	Nbr_Appréci
0004724526	0.0	1.0	4.0	5.0	4.2	0.83666	4.0	Nbr_Appréci
0002154129	0.0	0.0	6.0	6.0	4.833333	0.408248	5.0	Nbr_Appréci
0028633784	0.0	1.0	6.0	7.0	4.428571	0.786796	5.0	Nbr_Appréci
0007103999	3.0	1.0	5.0	9.0	3.333333	1.658312	4.0	Nbr_Appréci
0007198299	0.0	1.0	4.0	5.0	4.2	0.83666	4.0	Nbr_Appréci
0007398433	0.0	0.0	8.0	8.0	4.75	0.46291	5.0	Nbr_Appréci
0007362544	1.0	0.0	16.0	17.0	4.294118	0.771744	4.0	Nbr_Appréci
0020425759	2.0	3.0	14.0	19.0	3.842105	0.898342	4.0	Nbr_Appréci
0002007770	212.0	227.0	2282.0	2721.0	4.352444	1.018398	5.0	Nbr_Appréci
0007300441	3.0	5.0	5.0	13.0	3.153846	1.463224	3.0	Nbr_Neutr
0028614518	0.0	3.0	3.0	6.0	3.833333	0.983192	3.5	Nbr_Neutr
0007119070	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutr
0006510019	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutr
000752417X	15.0	35.0	22.0	72.0	3.125	0.918319	3.0	Nbr_Neutr
0020826621	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutr
000648204X	5.0	7.0	5.0	17.0	2.882353	1.166316	3.0	Nbr_Neutr
0028617401	0.0	3.0	2.0	5.0	3.8	1.095445	3.0	Nbr_Neutr
0007328230	0.0	4.0	3.0	7.0	3.571429	0.786796	3.0	Nbr_Neutr
0027858405	0.0	4.0	3.0	7.0	3.714286	0.95119	3.0	Nbr_Neutr
002073610X	13.0	1.0	5.0	19.0	2.210526	1.685854	1.0	Nbr_Non_Appréci
0007432852	2.0	2.0	2.0	6.0	3.333333	1.36626	3.0	Nbr_Non_Appréci
0002311216	17.0	9.0	14.0	40.0	2.85	1.494434	3.0	Nbr_Non_Appréci
0025631403	3.0	1.0	2.0	6.0	2.666667	1.632993	2.5	Nbr_Non_Appréci
000712614X	4.0	3.0	4.0	11.0	3.272727	1.272078	3.0	Nbr_Non_Appréci
0007204396	12.0	9.0	8.0	29.0	2.793103	1.235756	3.0	Nbr_Non_Appréci
0007133073	5.0	2.0	3.0	10.0	2.6	1.837873	2.0	Nbr_Non_Appréci
0007190360	3.0	2.0	2.0	7.0	2.857143	1.676163	3.0	Nbr_Non_Appréci
0007278675	2.0	1.0	2.0	5.0	3.2	1.788854	3.0	Nbr_Non_Appréci
0007175337	6.0	2.0	6.0	14.0	3.214286	1.57766	3.0	Nbr_Non_Appréci

Petite visualisation où l'on sélectionne au hasard 10 échantillons dans chaque strate

Notre matrice d'entrée dans ce qui suit sera donc la matrice suivante sans inclure la catégorie du livre:

	Nbr_Non_Apprécié	Nbr_Neutre	Nbr_Apprécié	Nbr_de_votes	Moyenne	Écart_type	Médiane	Catégorie
0007175175	1.0	1.0	23.0	25.0	4.68	0.748331	5.0	Nbr_Apprécié
0004724526	0.0	1.0	4.0	5.0	4.2	0.83666	4.0	Nbr_Apprécié
0002154129	0.0	0.0	6.0	6.0	4.833333	0.408248	5.0	Nbr_Apprécié
0028633784	0.0	1.0	6.0	7.0	4.428571	0.786796	5.0	Nbr_Apprécié
0007103999	3.0	1.0	5.0	9.0	3.333333	1.658312	4.0	Nbr_Apprécié
0007198299	0.0	1.0	4.0	5.0	4.2	0.83666	4.0	Nbr_Apprécié
0007398433	0.0	0.0	8.0	8.0	4.75	0.46291	5.0	Nbr_Apprécié
0007362544	1.0	0.0	16.0	17.0	4.294118	0.771744	4.0	Nbr_Apprécié
0020425759	2.0	3.0	14.0	19.0	3.842105	0.898342	4.0	Nbr_Apprécié
0002007770	212.0	227.0	2282.0	2721.0	4.352444	1.018398	5.0	Nbr_Apprécié
0007300441	3.0	5.0	5.0	13.0	3.153846	1.463224	3.0	Nbr_Neutre
0028614518	0.0	3.0	3.0	6.0	3.833333	0.983192	3.5	Nbr_Neutre
0007119070	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutre
0006510019	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutre
000752417X	15.0	35.0	22.0	72.0	3.125	0.918319	3.0	Nbr_Neutre
0020826621	1.0	2.0	2.0	5.0	3.2	1.48324	3.0	Nbr_Neutre
000648204X	5.0	7.0	5.0	17.0	2.882353	1.166316	3.0	Nbr_Neutre
0028617401	0.0	3.0	2.0	5.0	3.8	1.095445	3.0	Nbr_Neutre
0007328230	0.0	4.0	3.0	7.0	3.571429	0.786796	3.0	Nbr_Neutre
0027858405	0.0	4.0	3.0	7.0	3.714286	0.95119	3.0	Nbr_Neutre
002073610X	13.0	1.0	5.0	19.0	2.210526	1.685854	1.0	Nbr_Non_Apprécié
0007432852	2.0	2.0	2.0	6.0	3.333333	1.36626	3.0	Nbr_Non_Apprécié
0002311216	17.0	9.0	14.0	40.0	2.85	1.494434	3.0	Nbr_Non_Apprécié
0025631403	3.0	1.0	2.0	6.0	2.666667	1.632993	2.5	Nbr_Non_Apprécié
000712614X	4.0	3.0	4.0	11.0	3.272727	1.272078	3.0	Nbr_Non_Apprécié
0007204396	12.0	9.0	8.0	29.0	2.793103	1.235756	3.0	Nbr_Non_Apprécié
0007133073	5.0	2.0	3.0	10.0	2.6	1.837873	2.0	Nbr_Non_Apprécié
0007190360	3.0	2.0	2.0	7.0	2.857143	1.676163	3.0	Nbr_Non_Apprécié
0007278675	2.0	1.0	2.0	5.0	3.2	1.788854	3.0	Nbr_Non_Apprécié
0007175337	6.0	2.0	6.0	14.0	3.214286	1.57766	3.0	Nbr_Non_Apprécié

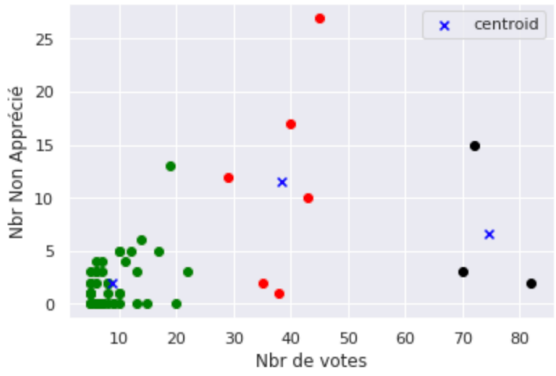
Question 3:

On reprend les mêmes étapes vues dans la partie (a).

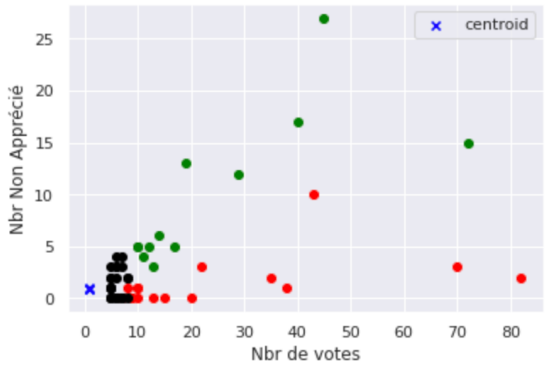
Résultats pour K-means :

Cosinus similarity matrix

	0	1	2	3	4	5	6	7	8	9	...	44	45	46	47	48	49	50	51	52	53
0	1.000000	0.983286	0.992192	0.981616	0.928118	0.988202	0.996233	0.996511	0.928118	0.985047	...	0.875835	0.938814	0.874584	0.892870	0.844804	0.915645	0.876044	0.912773	0.877897	0.836853
1	0.983286	1.000000	0.997675	0.999338	0.862254	0.959822	0.995349	0.993966	0.862254	0.959837	...	0.914534	0.935861	0.870526	0.937294	0.822210	0.870663	0.905475	0.888785	0.910457	0.832932
2	0.992192	0.997675	1.000000	0.995613	0.880021	0.968512	0.999081	0.999035	0.880021	0.966666	...	0.893217	0.930041	0.860621	0.916262	0.816883	0.877929	0.886270	0.887898	0.890642	0.820995
3	0.981616	0.999338	0.995613	1.000000	0.868173	0.962802	0.993993	0.991607	0.868173	0.963880	...	0.928431	0.946790	0.886678	0.948723	0.839706	0.881462	0.920303	0.901581	0.924852	0.851405
4	0.928118	0.862254	0.880021	0.868173	1.000000	0.969732	0.899569	0.895730	1.000000	0.968760	...	0.827450	0.936034	0.915659	0.815982	0.929803	0.986488	0.850080	0.961524	0.843364	0.897150
5	0.988202	0.959822	0.968512	0.962802	0.969732	1.000000	0.978223	0.975414	0.969732	0.999488	...	0.899571	0.969922	0.927209	0.904160	0.911751	0.966393	0.907744	0.961352	0.906422	0.898864
6	0.996233	0.995349	0.999081	0.993993	0.899569	0.978223	1.000000	0.999607	0.899569	0.976444	...	0.896205	0.939842	0.874179	0.916357	0.835229	0.896451	0.891858	0.903384	0.895272	0.836074
7	0.996511	0.993966	0.999035	0.991607	0.895730	0.975414	0.999607	1.000000	0.895730	0.972765	...	0.883610	0.930520	0.861124	0.905398	0.821746	0.888441	0.878836	0.893313	0.882479	0.821356
8	0.928118	0.862254	0.880021	0.868173	1.000000	0.969732	0.899569	0.895730	1.000000	0.968760	...	0.827450	0.936034	0.915659	0.815982	0.929803	0.986488	0.850080	0.961524	0.843364	0.897150
9	0.985047	0.959837	0.966666	0.963880	0.968760	0.999488	0.976444	0.972765	0.968760	1.000000	...	0.912124	0.977214	0.938370	0.915269	0.922603	0.970580	0.920326	0.968543	0.918904	0.911796
10	0.935352	0.873432	0.890002	0.879341	0.999719	0.974970	0.908724	0.904772	0.999719	0.974269	...	0.839063	0.943276	0.921694	0.828695	0.933119	0.988262	0.860447	0.965720	0.854180	0.902643
11	0.992011	0.952473	0.969102	0.949994	0.954569	0.987965	0.977373	0.978903	0.954569	0.982567	...	0.830355	0.921134	0.858346	0.843386	0.841777	0.927403	0.836845	0.909894	0.836553	0.821003
12	0.999070	0.988324	0.994405	0.987842	0.925218	0.989049	0.998014	0.997249	0.925218	0.987211	...	0.895377	0.949707	0.889196	0.911428	0.857478	0.919791	0.894866	0.921756	0.896874	0.853189
13	0.989196	0.997594	0.996529	0.998561	0.893266	0.975858	0.997096	0.994737	0.893266	0.976487	...	0.926564	0.955904	0.898275	0.944122	0.857205	0.902665	0.921558	0.917577	0.925005	0.863890
14	0.966503	0.908022	0.927581	0.909066	0.988425	0.985322	0.942248	0.941746	0.988425	0.981212	...	0.820105	0.927780	0.884162	0.820976	0.885299	0.962832	0.835787	0.937464	0.831956	0.855202
15	0.926600	0.962432	0.946202	0.971347	0.844089	0.931805	0.946416	0.937506	0.844089	0.940393	...	0.989592	0.973576	0.946570	0.996551	0.902369	0.893964	0.983794	0.933235	0.986772	0.926585
16	0.960239	0.994070	0.984365	0.995871	0.828837	0.938503	0.980109	0.976459	0.828837	0.941608	...	0.941868	0.938767	0.881553	0.963357	0.827092	0.854317	0.929929	0.885128	0.935680	0.847976
17	0.949348	0.882084	0.903662	0.883941	0.993574	0.975672	0.920715	0.919885	0.993574	0.971549	...	0.801711	0.918059	0.880358	0.798514	0.888725	0.965998	0.820906	0.936066	0.815752	0.854031



euclidienne

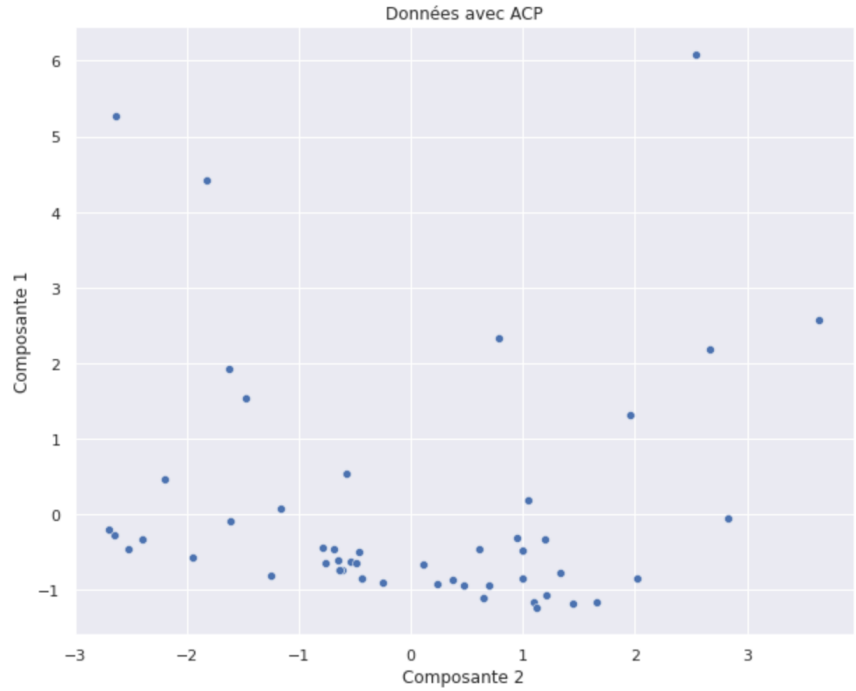


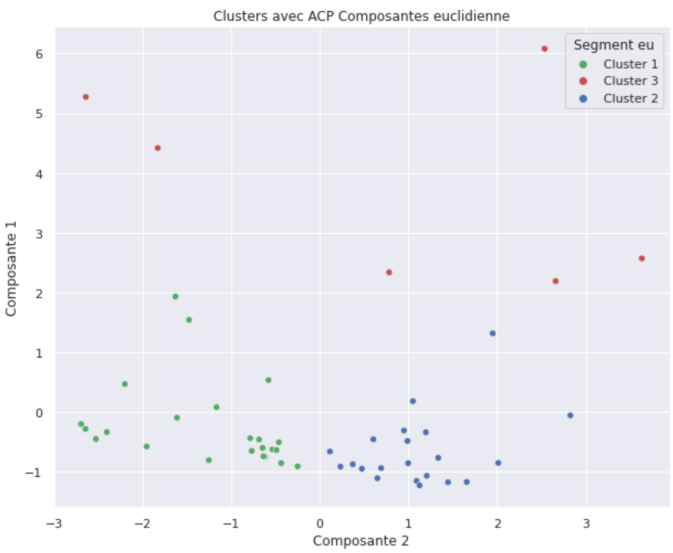
cosinus

Ces segmentations sont plus parlantes ; les trois couleurs sont présentes de manières équitable chose qui prouve que l'on retrouve tous nos clusters avec des tailles a peu près identiques dans la visualisation et **donc** la répartition est plus au moins équilibrée, surtout pour la similarité cosinus.

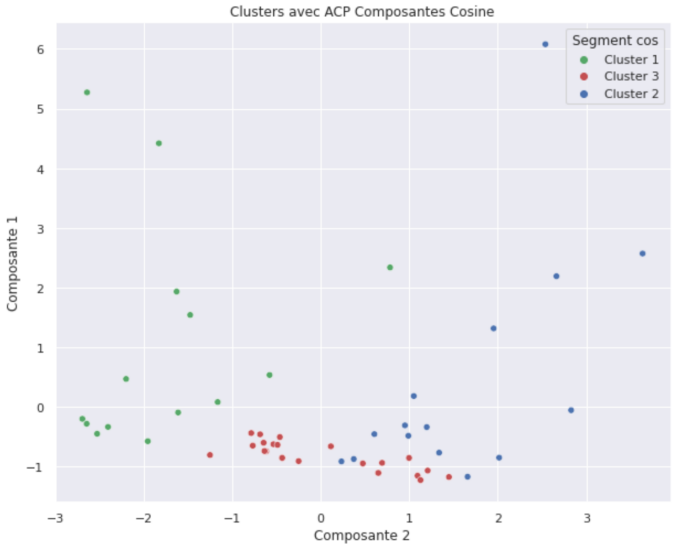
Ceci peut être expliqué par le fait que l'on a assez de données de toutes les catégories et nulle ne domine sur les autres.

Résultats pour ACP + K-means :





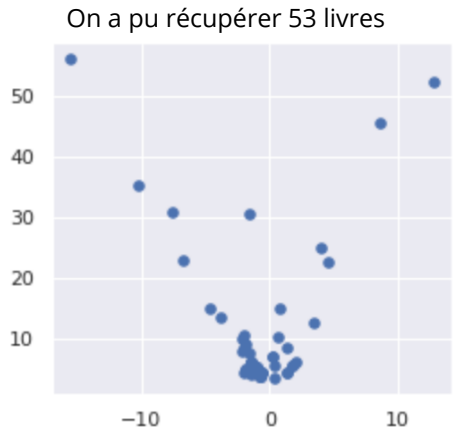
La segmentation est plus au moins équilibrée et il n'y a aucun chevauchement visuel.

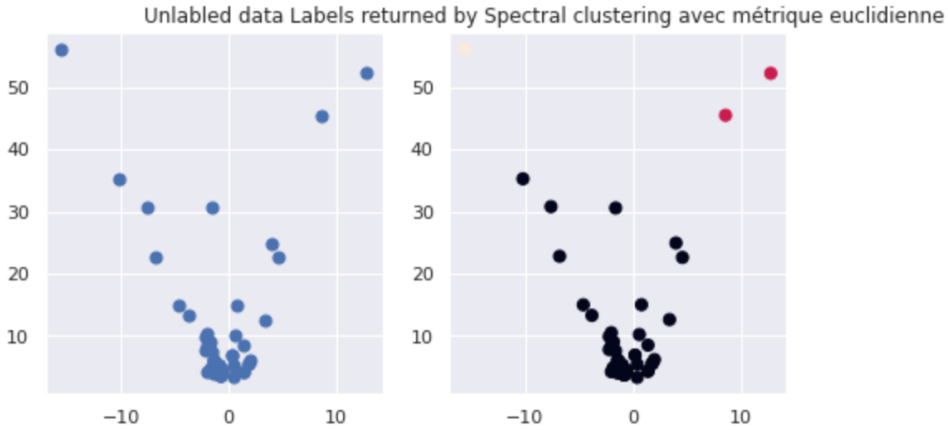


La segmentation est plus équilibrée mais il y a un chevauchement entre les clusters 2 et 3.

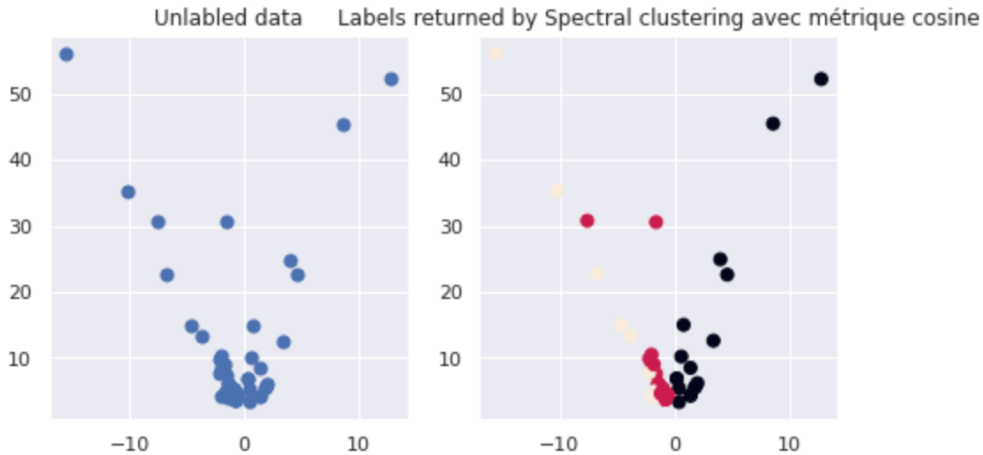
L'ACP donne une représentation plus saine des données à l'aide de 2 composantes principales et celles ci présentent des segmentations meilleures et plus équilibrées malgré le chevauchement léger avec la distance cosine

Résultats pour spectral clustering :





Pas de chevauchements, les 3 clusters sont présents mais la couleur blanche est minime; il n'y a qu'un point qui apparait dans le cluster.



Segmentation équilibrée entre clusters et il n'y a pas de chevauchements.
La segmentation semble avoir plus de sens avec la similarité cosinus, et le clustering spectral donne des clusters plus significatifs.

Résultats de l'évaluation :

	distance euclidienne silhouette	distance euclidienne info. mut	similarité cosinus silhouette	similarité cosinus info. mut
KMeans	0.809028	0.018037	0.667334	0.398677
Spectrale	0.562769	0.313901	0.474648	0.449523

C'est vrai que K-means a le plus haut score silhouette mais ne représente pas une bonne information sur nos livres.
Or le clustering spectral a des valeurs considérables dans le score silhouette et représente un bon score élevé avec l'Info mutuelle ; **il est donc meilleur dans ce cas**.

Comparaison des évaluations de clustering sans échantillonnage stratifié et avec échantillonnage stratifié :

Sans échantillonnage stratifié

	distance euclidienne silhouette	distance euclidienne info. mut	similarité cosinus silhouette	similarité cosinus info. mut
KMeans	0.931572	0.010142	0.668322	0.200047
Spectrale	0.713882	0.006375	0.084081	0.161104

Avec échantillonnage stratifié

	distance euclidienne silhouette	distance euclidienne info. mut	similarité cosinus silhouette	similarité cosinus info. mut
KMeans	0.809028	0.018037	0.667334	0.398677
Spectrale	0.562769	0.313901	0.474648	0.449523

- KMeans n'a pas été très sensible à la stratification des données , par contre le clustering spectral a très évolué et représente des scores plus élevés dans la partie b.
- Il y a une nette augmentation de l'information mutuelle pour les deux clustering avec l'échantillonnage stratifié. On s'y attendait d'après la définition de l'information mutuelle.
- Il y a une nette augmentation de de l'information mutuelle et de la silhouette avec la distance cosinus pour le clustering spectral.

Question 4:

Vu que maintenant, et grâce au clustering, on aura des données étiquetées, l'analyse prédictive sera basée sur un algorithme d'apprentissage supervisé. Le but serait de trouver un modèle (une fonction) qui minimise la différence entre notre prédiction et la cible pour les données déjà étiquetées tout en évitant un sur-apprentissage. Finalement, on va appliquer ce modèle sur nos nouvelles données pour les étiqueter.

5.Conclusion

Lors de ce TP, on a pu implémenter différentes méthodes de clustering tout en variant la métrique de similarité utilisée. Visualiser ces segmentations était très intéressant puisque c'était très sensible à la distance utilisée.

De plus on a pu évaluer ces segmentations grâce a deux métriques d'évaluation : l'information mutuelle et la silhouette.

Un aspect particulièrement pertinent était la variation de la qualité de la segmentation en fonction des données utilisées. En effet, l'échantillonnage stratifié a pu nous aider a dépasser la multi-variance interne de nos données en prenant un échantillon de chaque catégorie, et en offrant ainsi de meilleurs segmentations de nos données et par conséquent une augmentation de l'information mutuelle.

6.Ressources

https://365datascience.com/tutorials/python-tutorials/pca-k-means/

https://matplotlib.org/stable/gallery/shapes_and_collections/scatter.html

https://scikitlearn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

https://ledatascientist.com/faire-du-clustering-avec-lalgorithme-k-means/

https://www.youtube.com/watch?v=OQdVuxTsLOQ

Fin