Rapport de projet Data Mining

Participants:

Minh Luan Hoa P1520026

Mouhamed Loum P1514727

Mai Ngoc Linh Nguyen P1520025

Gaultier Parain P1808350

Introduction

Dans le cadre de l'UE data mining, nous avons été amenés à réaliser un projet implémentant une approche de fouilles de données. Dans cette perspective, nous avons décidé de recueillir des données sur les restaurants de la ville de Lyon afin de mener à bout ce projet.

Nos objectifs de base étaient dans un premier temps d'effectuer de la fouille de motifs sur nos données et dans un second temps d'implémenter une méthode de clustering sur nos données après traitement de ces dernières.Ainsi on voulait pouvoir extraire des informations qui en fin de compte nous permettrait par exemple de déterminer les caractéristiques des restaurants cinq étoiles...

Présentation des données

Les données sont issues du site Tripadvisor et contiennent pour chaque restaurant de Lyon et de ses alentours présent sur le site : son nom, le nombre d'avis, le type de nourriture, sa note moyenne, la présence du wifi, la possibilité de livraison, le prix, sa latitude, sa longitude et son district.

Il y a en tout 3068 restaurants dans notre corpus de données donc suffisamment de quoi extraire assez d'informations par des méthodes de fouilles de données.

Etat de l'art

Une analyse des données a déjà été faite https://www.kaggle.com/launay10christian/the-best-restaurant-in-lyon

Dans cette analyse, on a cherché quelles étaient les catégories de restaurants les plus fréquentes et les mieux notées. Cependant un seul facteur est pris en compte à chaque fois et la combinaison de plusieurs facteurs n'est pas étudiée. On peut donc rechercher les combinaisons associées à une meilleure note ou à un district en particulier. On peut également regrouper les restaurants en cluster selon différents facteurs.

Preprocessing

Dans cette étape, on supprime d'abord les espaces dans le nom des colonnes et les colonnes inutilisables ("ranking" et "average_price").

Ensuite, on traite les lignes contenant les informations manquantes en supprimant les lignes ayant la valeur "no information". Après les suppressions, le jeu de données se réduit à 1758 restaurants et 9 variables considérables.

2.3. Supprimer les lignes contenant la valeur 'no information' dans quelques colonnes

```
cols = ['review_number', 'food_type', 'overallRating', 'lat', 'lng', 'district']

for c in cols:
    data = data[~data[c].str.contains('no')]
data
```

	restaurant_name	review_number	food_type	overallRating	wifi	livraison	lat	lng
0	le Neuvième Art	626 avis	Française Européenne Végétariens bienvenus	5.0	yes	yes	45.76849365234375	4.856463909149
1	Aromatic	601 avis	Française Européenne Végétariens bienvenus	5.0	yes	no	45.77438735961914	4.83096122741€
2	Le Boeuf d'Argent	482 avis	Française Européenne Végétariens bienvenus	4.5	no	no	45.76266098022461	4.82690000534(
3	Le Comptoir des Cousins	205 avis	Française Européenne Sud-américaine	5.0	yes	no	45.76849365234375	4.856463909149
5	L'Archange	946 avis	Française Européenne	4.5	no	no	45.76845932006836	4.827991008758
...	...	...	...	...	...	...	...	...
2795	Le pop rock	49 avis	Française	1.5	no	no	45.75033950805664	4.826126098632
2797	Burger King	45 avis	Américaine	1.5	no	no	45.71208953857422	4.967150211334
2798	Les Chandelles	861 avis	Française	2.0	no	no	45.76337432861328	4.827271938323
2799	La Carretta	47 avis	Pizza	1.5	no	no	45.76569366455078	4.827927112579
2800	Pizzeria Volcano	12 avis	Pizza	1.0	no	no	45.75016784667969	4.830166816711

1758 rows × 9 columns

Puis, on traite la colonne de type de nourriture, chaque ligne de cette colonne contient plusieurs types de nourriture, ainsi, on doit les diviser. Pour cela, on implémente la fonction "split(food_type)" qui reçoit un String des "food_type" et renvoie la liste des types de nourriture. On utilise cette fonction et obtient une liste des types de nourriture. Ensuite, on convertit la liste des types de nourriture de chaque restaurant en une liste de booléens dont la valeur correspond à la présence chaque type de nourriture et on ajoute ces colonnes au jeu de données on se retrouve finalement avec un corpus de données de 78 colonnes .

	Française	Européenne	Végétariens bienvenus	Sud-américaine	Méditerranéenne	Internationale	Barbecue/Grillades	Bar à vins	Moc
0	1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0
3	1	1	0	1	0	0	0	0	0
4	1	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
1753	1	0	0	0	0	0	0	0	0
1754	0	0	0	0	0	0	0	0	0
1755	1	0	0	0	0	0	0	0	0
1756	0	0	0	0	0	0	0	0	0
1757	0	0	0	0	0	0	0	0	0

1758 rows × 78 columns

Enfin, on convertit les colonnes numériques de type String à Float, les colonnes de booléens de (yes/no) à (0/1) et enfin on transforme le jeu de données nettoyé à un fichier csv.

```
data['wifi'].replace(inplace=True,to_replace='yes',value=1.0)
data['wifi'].replace(inplace=True,to_replace='no',value=0.0)
data['livraison'].replace(inplace=True,to_replace='yes',value=1.0)
data['livraison'].replace(inplace=True,to_replace='no',value=0.0)
data
```

	restaurant_name	review_number	overallRating	wifi	livraison	lat	lng	distric
0	le Neuvième Art	626.0	5.0	1.0	1.0	45.76849365234375	4.85646390914917	69006 Lyon
1	Aromatic	601.0	5.0	1.0	0.0	45.77438735961914	4.830961227416992	69004 Lyon
2	Le Boeuf d'Argent	482.0	4.5	0.0	0.0	45.76266098022461	4.826900005340576	69005 Lyon
3	Le Comptoir des Cousins	205.0	5.0	1.0	0.0	45.76849365234375	4.85646390914917	69006 Lyon
4	L'Archange	946.0	4.5	0.0	0.0	45.76845932006836	4.827991008758545	69001 Lyon
...	...	...	...	...	...	...	...	...
1753	Le pop rock	49.0	1.5	0.0	0.0	45.75033950805664	4.8261260986328125	69002 Lyon
1754	Burger King	45.0	1.5	0.0	0.0	45.71208953857422	4.9671502113342285	69800 Saint-Priest Lyon
1755	Les Chandelles	861.0	2.0	0.0	0.0	45.76337432861328	4.827271938323975	69005 Lyon
1756	La Carretta	47.0	1.5	0.0	0.0	45.76569366455078	4.827927112579346	69005 Lyon
1757	Pizzeria Volcano	12.0	1.0	0.0	0.0	45.75016784667969	4.830166816711426	Lyon

1747 rows × 86 columns

Détection de motifs

Pour la détection de motifs, on utilise la bibliothèque pysubgroup https://github.com/flemmerich/pysubgroup

On recherche d'abord les motifs associés à une note de 5.0 pour l'espace de recherche, on ignore le nom du restaurant, le nombre d'avis, la latitude et la longitude car ils ne sont pas pertinents pour faire de la détection de motifs.

On prend une profondeur maximale de 2 car prendre une profondeur plus grandes rendrait le temps d'exécution trop grand.

On utilise l'algorithme beamsearch, implémenté dans cette bibliothèque.

On obtient les 10 motifs dont le score pour le test du chi² est le plus grand, c'est à dire les combinaisons de catégories de restaurants liées à une note de 5.0 ou à la présence dans un district en particulier.

```
target = ps.NominalTarget(' overallRating  ', '5.0 ')
#type(target)
searchspace = ps.create_selectors(resto, ignore=[' overallRating  ','restaurant_name ','review_number
',' food_type ',' average_price ',' lat ',' lng ',' ranking '])
task = ps.SubgroupDiscoveryTask (resto, target, searchspace, depth=2, qf=ps.ChiSquaredQF())
result = ps.BeamSearch().execute(task)
for (q, sg) in result:
    print (str(q) + ":\t" + str(sg.subgroup_description))
```

```
28.757646722379988:     Saine=True AND Restauration rapide=True
26.487461447292095:      district =69001  Lyon   AND Café=True
21.587440931129525:     Tunisienne=False AND Créole=False
21.587440931129525:     Tunisienne=False AND Cajun =False
21.515523653225245:     Européenne=False AND Japonaise=False
20.74788446125862:      Européenne=False AND Sushi=False
20.373981321125374:     Café=False AND Moderne=False
19.337701353141753:     Café=False AND Cajun =False
19.337701353141753:     Café=False AND Créole=False
18.83392794717037:      Café=True AND Bar à bières=True
```

On reproduit ensuite l'opération pour les différents districts pour voir quelles combinaisons de types de restaurants a le plus de chance d'être liée à la présence dans chaque district.

```
target = ps.NominalTarget(' district ', '69002  Lyon  ')
searchspace = ps.create_selectors(resto, ignore=[' district ',' overallRating  ','restaurant_name ','
review_number ',' food_type ',' average_price ',' lat ',' lng ',' ranking '])
task = ps.SubgroupDiscoveryTask (resto, target, searchspace, depth=2, qf=ps.ChiSquaredQF())
result = ps.BeamSearch().execute(task)
for (q, sg) in result:
    print (str(q) + ":\t" + str(sg.subgroup_description))
```

```
38.20515489461216:      Française=False AND  livraison =no
36.25015597938105:      Végétariens bienvenus=False AND Française=False
36.108645724176625:     Européenne=False AND Américaine=False
35.73399522744213:      Européenne=False AND  livraison =no
34.84840598945876:      Végétariens bienvenus=False AND Européenne=False
31.586115706280818:     Française=False AND Américaine=False
31.126227757476666:     Française=False AND Italienne=False
30.82591761203981:      Française=False AND Méditerranéenne=False
30.490860551861175:     Soupes=False AND Européenne=False
30.228653354434158:     Européenne=False AND Café=False
```

```
target = ps.NominalTarget(' district ', '69003  Lyon  ')
searchspace = ps.create_selectors(resto, ignore=[' district ',' overallRating  ','restaurant_name ','
review_number ',' food_type ',' average_price ',' lat ',' lng ',' ranking '])
task = ps.SubgroupDiscoveryTask (resto, target, searchspace, depth=2, qf=ps.ChiSquaredQF())
result = ps.BeamSearch().execute(task)
for (q, sg) in result:
    print (str(q) + ":\t" + str(sg.subgroup_description))
```

```
12.923633129020589:     Libanaise=False AND Fruits de mer =False
12.923633129020589:     Libanaise=False AND Poisson=False
12.632709032755558:     Fruits de mer =True AND Méditerranéenne=False
12.632709032755558:     Poisson=True AND Méditerranéenne=False
12.399310277373052:     Poisson=False AND Russe=False
12.399310277373052:     Fruits de mer =False AND Russe=False
12.205992040593271:     Saine=False AND Fruits de mer =False
12.205992040593271:     Saine=False AND Poisson=False
12.083076586928495:     Fruits de mer =False AND Africaine=False
12.083076586928495:     Poisson=False AND Africaine=False
```

Clustering

Pour le clustering, on applique l'algorithme K-means de la bibliothèque Scikit-learn avec le jeu de données nettoyé. On fait le regroupement 3 fois: basé sur les attributs du restaurant (nombres de commentaires, rating, avoir wifi et livraison ou non).

Avant de procéder au clustering, on commence par extraire les colonnes nécessaires. Ensuite, on normalise le training set et réduit ses dimensions avec StandardScaler et PCA de Scikit-learn.

1.1. Extraire les colonnes necessaires

In [3]:
```
df1 = df[df.columns[:4]]
df1.dtypes
```

Out[3]:
```
restaurant_name     object
review_number      float64
overallRating      float64
wifi               float64
dtype: object
```

1.2. Extraire et normaliser le training set

In [4]:
```
X = df1.values[:, 1:]
Xss = StandardScaler().fit_transform(X)
```

Puis, on lance K-means plusieurs fois pour trouver la valeur optimale pour le nombre de clusters (K) grâce à la méthode “coude”.

1.3. Appliquer K-means avec k entre 2 et 20 pour trouver la meilleure valeur k

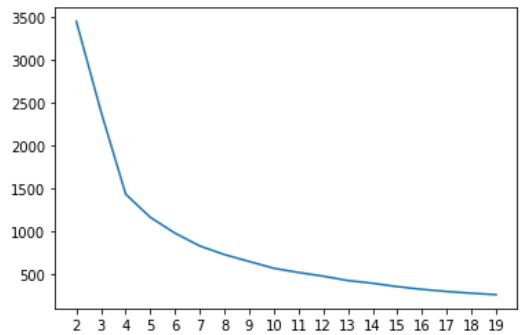
```
]: scores = []
max_k = 20
for i in range(2, max_k):
    kmeans = KMeans(n_clusters=i, random_state=0).fit(Xss)
    sc = kmeans.inertia_
    scores.append(sc)
    print('k = {0}, SSE = {1:.4f}'.format(i, sc))
```

```
k = 2, SSE = 3454.5131
k = 3, SSE = 2397.9363
k = 4, SSE = 1431.6823
k = 5, SSE = 1159.8584
k = 6, SSE = 975.4295
k = 7, SSE = 825.9201
k = 8, SSE = 725.0197
k = 9, SSE = 644.3009
k = 10, SSE = 564.1973
k = 11, SSE = 515.2932
k = 12, SSE = 471.9180
k = 13, SSE = 421.0376
k = 14, SSE = 389.1855
k = 15, SSE = 349.8451
k = 16, SSE = 317.9933
k = 17, SSE = 292.8907
k = 18, SSE = 272.4156
k = 19, SSE = 255.9120
```

1.4. Appliquer le methode "coude" pour choisir la valeur k

```
plt.plot(range(2, max_k), scores)
plt.xticks(range(2, max_k))
```

```
([<matplotlib.axis.XTick at 0x7f9c308e7350>,
  <matplotlib.axis.XTick at 0x7f9c308e9d10>,
  <matplotlib.axis.XTick at 0x7f9c308ceb50>,
  <matplotlib.axis.XTick at 0x7f9c2fc4a590>,
  <matplotlib.axis.XTick at 0x7f9c2fc4a750>,
  <matplotlib.axis.XTick at 0x7f9c2fc56250>,
  <matplotlib.axis.XTick at 0x7f9c2fc56350>,
  <matplotlib.axis.XTick at 0x7f9c2fc569d0>,
  <matplotlib.axis.XTick at 0x7f9c2fbdd050>,
  <matplotlib.axis.XTick at 0x7f9c2fbdd690>,
  <matplotlib.axis.XTick at 0x7f9c2fbdd850>,
  <matplotlib.axis.XTick at 0x7f9c2fc4ab50>,
  <matplotlib.axis.XTick at 0x7f9c2fbe4310>,
  <matplotlib.axis.XTick at 0x7f9c2fbe4510>,
  <matplotlib.axis.XTick at 0x7f9c2fbe4210>,
  <matplotlib.axis.XTick at 0x7f9c2fbeb1d0>,
  <matplotlib.axis.XTick at 0x7f9c2fbeb810>,
  <matplotlib.axis.XTick at 0x7f9c2fbf22d0>],
 <a list of 18 Text xticklabel objects>)
```



Après d'avoir eu la valeur K, on lance K-means et assigne le groupe pour chaque ligne dans le jeu de données.

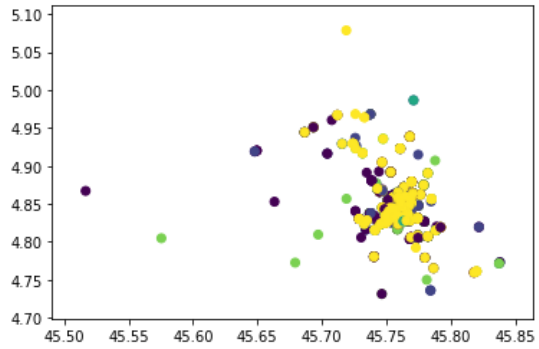
```
df['cluster'] = clusters
df
```

ɔéenne	...	Roumaine	Tunisienne	Suisse	Albanaise	Canadienne	Amérique centrale	Arabe	Vénézuélienne	Chilienne	cluster
	...	0	0	0	0	0	0	0	0	0	3
	...	0	0	0	0	0	0	0	0	0	3
	...	0	0	0	0	0	0	0	0	0	2
	...	0	0	0	0	0	0	0	0	0	1
	...	0	0	0	0	0	0	0	0	0	3
	...	...	...	...	...	...	...	...	...	...	...
	...	0	0	0	0	0	0	0	0	0	5
	...	0	0	0	0	0	0	0	0	0	5
	...	0	0	0	0	0	0	0	0	0	3
	...	0	0	0	0	0	0	0	0	0	5
	...	0	0	0	0	0	0	0	0	0	5

Finalement, on visualise les clusters en fonction de la position géographique des restaurant ensuite par 2 attributs "nombres de commentaires", "rating".

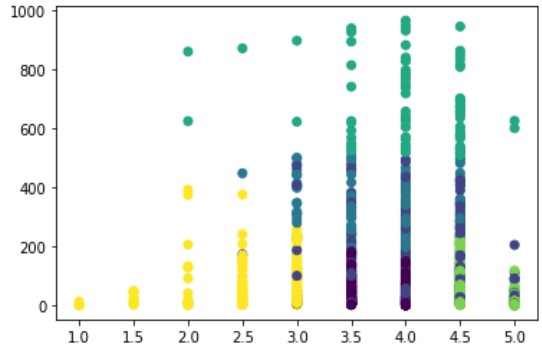
```
plt.scatter(df['lat'], df['lng'], c=df['cluster'])
```

```
<matplotlib.collections.PathCollection at 0x7f9c2fb0ddd0>
```

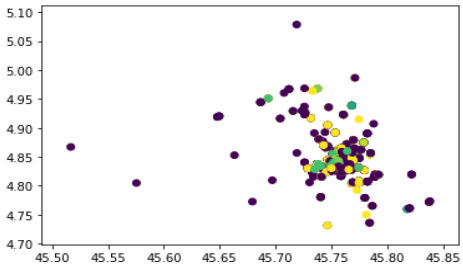



```
plt.scatter(df['overallRating'], df['review_number'], c=df['cluster'])
```

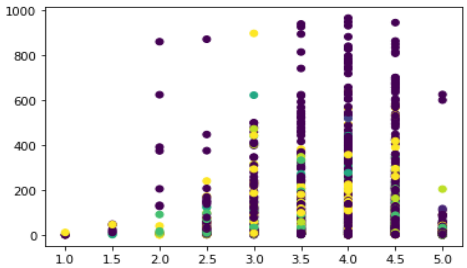
```
<matplotlib.collections.PathCollection at 0x7f9c2fa9c3d0>
```



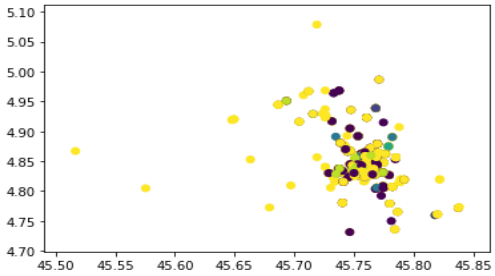
On a fait le même procédé pour faire du clustering à nouveau mais cette fois à partir du type de plat servi par les restaurants.



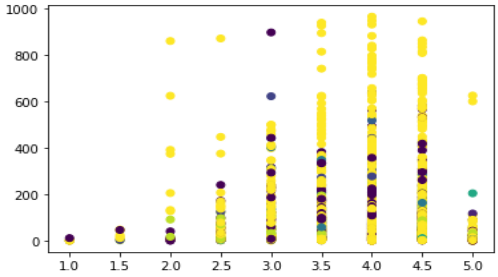

```
In [21]: plt.scatter(df['overallRating'], df['review_number'], c=df['cluster'])
Out[21]: <matplotlib.collections.PathCollection at 0x7f9c2ed28890>
```



Et enfin pour finir, nous avons fait une dernière itération de notre algorithme de clustering sur tous les attributs de notre jeu de données.




```
In [30]: plt.scatter(df['overallRating'], df['review_number'], c=df['cluster'])
Out[30]: <matplotlib.collections.PathCollection at 0x7f9c2e20c790>
```



Conclusion

Pour conclure, ce projet nous a montré que la détection de motifs pouvait nous apporter des informations supplémentaires pour déterminer quelles sont les caractéristiques des restaurants associés à une bonne note ou à la présence dans un district en particulier. On observe des caractéristiques très différentes associées à chaque district.

Le clustering apporte lui aussi des informations supplémentaires quant aux regroupements possibles entre restaurants mais leur nombre important ainsi que le nombre important de features rend ces regroupements difficiles à visualiser et interpréter.