INSTITUT FRANCOPHONE INTERNATIONAL

VNU ĐẠI HỌC QUỐC GIA HÀ NỘI Vietnam National University, Hanoi

RAPPORT GROUPE 10

Rapport Final

Fouille de Donnée et de Recherche d'informations
Data set : ***ensemble de données sur les avis des voyageurs***
Source data set:
http://archive.ics.uci.edu/ml/datasets/Travel+Reviews

Étudiants

- OUEDRAOGO Wend-Panga Jérémie
- NGUYEN Troung Thinh
- KAMBU MBUANGI Fabien

Enseignant

Nguyen Thi Minh Huyen

Promotion 24 - Système Intelligent Multimédia – 2019-2021

INTRODUCTION

Au cours de notre formation, nous avons eu le module de fouille de données, au cours duquel nous avons eu des séries de TP et nous avons terminer le module par une présentation générale de notre travail. Le présent document constitue le rapport final de ces travaux. Il est structuré selon l'ordre chronologiques des travaux pratiques effectués.

TABLE DES MATIÈRES

I - PREMIÈRE PARTIE : Data set

I.1 – Description du data set

Notre étude est faite sur un ensemble de données en provenance du site d'entrepôt de données http://archive.ics.uci.edu/, dont l'accès direct est http://archive.ics.uci.edu/ml/datasets/Travel+Reviews. Cet ensemble de données contient le recueille des avis des 980 voyageurs sur différents endroits qu'ils visitent à travers l'Asie de l'Est (*cf. I.2*) . Les avis recueillis pour chaque voyageur sont classés de 0 à 4, détaillés comme suit :

- (0) : Terrible
- (1) : Médiocre
- (2) : Moyen
- (3) : Très bon
- (4) :Excellent

I.2 – Description des variables

Notre data set est constitué de onze (11) attributs au totale détaillés comme suit :

1. **`UserID`***: identifiant utilisateur unique – sert à identifier l'utilisateur*
2. **`Galerie_Art`***: rétroaction moyenne des utilisateurs sur les galeries d'art*
3. **`Club_Danse`***: rétroaction moyenne des utilisateurs sur les clubs de danse*
4. **`Bar_Jus`***: rétroaction moyenne des utilisateurs sur les bars à jus*
5. **`Restaurant`***: rétroaction moyenne des utilisateurs sur les restaurants*
6. **`Musées`***: rétroaction moyenne des utilisateurs sur les musées*
7. **`Station_Vacances`***: Rétroaction moyenne des utilisateurs sur les stations de vacances*
8. **`Parc_Pique-nique`***: Rétroaction moyenne des utilisateurs sur les parcs / aires de pique-nique*
9. **`Plage`** *Rétroaction moyenne des utilisateurs sur les plages*

10. **`Theatre`***: Rétroaction moyenne des utilisateurs sur les théâtres*
11. **`Religieux`***: Rétroaction moyenne des utilisateurs sur les institutions religieuses*

3– Relation entre deux variables

Nous avons un ensemble de données de variables quantitatives. Et pour faire la relation entre deux variables, nous utiliserons une régression simple dont la formule est $Y = aX + b + e$, **avec** $e \sim N(0, \sigma)$.

II. DEUXIÈME PARTIE : Étude détaillée des attributs

II-1. Introduction

Cette partie est consacrée à une étude détaillée des différents attributs de notre jeux de données décrit ci-haut. Sur ce, nous effectuerons certaines opérations statistiques dont l'analyse des résultats nous permettra afin de nous faciliter les tâches dans les prochaines analyses.
Cette partie est consacrée à une étude détaillée des différents attributs de notre jeux de données décrit ci-haut. Sur ce, nous effectuerons certaines opérations statistiques dont l'analyse des résultats nous permettra afin de nous faciliter les tâches dans les prochaines analyses.

II-2. Les Outils utilisés

Pour notre étude, nous avons utilisé le langage de programmation python pour notre étude avec les librairies et outils suivants :

-**Anaconda** : c'est une distribution libre et open source3 des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique.
-**Jupiter Notbook**: c'est un éditeur de code python intégré à anaconda.
-Python : qui est un langage de programmation orientée objet et très adapté au domaine de fouille de données
- **Pandas** est une librairie python qui permet de manipuler facilement des données à analyser dans des tableaux qui sont appelés Data Frames.

-**NumPy** est une extension Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
-**Matplotlib** est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.
-***Seaborn*** est une librairie qui vient s'ajouter à Matplotlib pour remplacer certains réglages par défaut et fonctions.

II.3 Import du data set et exploration des données

-II.3.1. Importation

Nous avons utiliser la bibliothèque pandas de python pour cette opérations

La capture ci-haut présente les codes qui nous ont permis de charger notre jeux de données à partir de l'éditeur Jupiter Notbook de python.

Nous pouvons remarquer que ce notre data set possède 980 lignes d'individus et 11 colonnes des attributs de ces derniers.

```
uploaded = files.upload()
```

Sélect. fichiers Aucun fichier choisi Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving data_set_travel.csv to data_set_travel.csv
```

```
data = pd.read_csv(io.BytesIO(uploaded['data_set_travel.csv']))
data
```

	UserID	Galerie_Art	Club_Danse	Bar_Jus	Restaurant	Musees	Station_Vacances	Parc_Pique-nique	Plage	Theatre	Religieux
0	User1	0.93	1.80	2.29	0.62	0.80	2.42	3.19	2.79	1.82	2.42
1	User2	1.02	2.20	2.66	0.64	1.42	3.18	3.21	2.63	1.86	2.32
2	User3	1.22	0.80	0.54	0.53	0.24	1.54	3.18	2.80	1.31	2.50
3	User4	0.45	1.80	0.29	0.57	0.46	1.52	3.18	2.96	1.57	2.86
4	User5	0.51	1.20	1.18	0.57	1.54	2.02	3.18	2.78	1.18	2.54
...	...	...	...	...	...	...	...	...	...	...	...
975	User976	0.74	1.12	0.30	0.53	0.88	1.38	3.17	2.78	0.99	3.20
976	User977	1.25	0.92	1.12	0.38	0.78	1.68	3.18	2.79	1.34	2.80
977	User978	0.61	1.32	0.67	0.43	1.30	1.78	3.17	2.81	1.34	3.02
978	User979	0.93	0.20	0.13	0.43	0.30	0.40	3.18	2.98	1.12	2.46
979	User980	0.93	0.56	1.13	0.51	1.34	2.36	3.18	2.87	1.34	2.40

980 rows × 11 columns

Pour nous assurer que notre data set ne contient aucune données manquantes, nous avons exécuter la commande python suivantes :

```
Entrée [24]:  1  sum(data.isnull().values.ravel())
Out[24]: 0
```

Le résultat affirme que notre jeux de données ne possède aucune des valeurs manquantes.

-II.3.2. Exploration des données

a) **La moyenne :** nous allons afficher la moyenne de chaque attribut

```
moyenne=data.mean()
moyenne
```

```
Galerie_Art          0.893194
Club_Danse           1.352612
Bar_Jus              1.013306
Restaurant           0.532500
Musees               0.939735
Station_Vacances     1.842898
Parc_Pique-nique     3.180939
Plage                2.835061
Theatre              1.569439
Religieux            2.799224
dtype: float64
```

Nous pouvons interpréter ces valeurs des moyennes comme suit en se basant sur les critère des notation des avis par lieu qui va de 0 à 4, nous avons,
la moyenne des appréciations des:

- Galeries d'Arts est médiocre ;
- Clubs de Danses est moyen ;
- Bars Jus est moyen ;
- Restaurants est médiocre ;
- Musées est médiocre ;
- Stations de vacances est moyen ;
- Parcs Pique-nique est excellent ;
- Plages est très bon;
- Théâtres est moyen ;
- Lieux Religieux est Très bon ;

b) **La médiane :**nous affichons la moyenne de chaque attribut

```
mediane=data.median()
mediane
```

```
Galerie_Art          0.83
Club_Danse           1.28
Bar_Jus              0.82
Restaurant           0.50
Musees               0.90
Station_Vacances     1.80
Parc_Pique-nique     3.18
Plage                2.82
Theatre              1.54
Religieux            2.78
dtype: float64
```

En partant de nos résultats ci-dessus, nous pouvons affirmé que :
- Plus de la moitié des voyageurs ont donné un avis médiocre sur les Galeries d'arts ;
- Plus de la moitié des voyageurs ont donné un avis moyen sur les Clubs de danses ;
- Plus de la moitié des voyageurs ont donné un avis médiocre sur les Bars de Jus ;
- Plus de la moitié des voyageurs ont donné un avis médiocre sur les Restaurants ;
- Plus de la moitié des voyageurs ont donné un avis médiocre sur les Musées ;
- Plus de la moitié des voyageurs ont donné un avis moyen sur les Stations des vacances;
- Plus de la moitié des voyageurs ont donné un avis excellent sur les Parcs pique-nique ;
- Plus de la moitié des voyageurs ont donné un avis très bon sur les plages ;
- Plus de la moitié des voyageurs ont donné un avis moyen sur les Théâtres ;
- Plus de la moitié des voyageurs ont donné un avis très bon sur les sites Religieux;

c) **L'écart Type**

```
ecart_type=data.std()
ecart_type
```

```
Galerie_Art          0.326912
Club_Danse           0.478280
Bar_Jus              0.788607
Restaurant           0.279731
Musees               0.437430
Station_Vacances     0.539538
Parc_Pique-nique     0.007824
Plage                0.137505
Theatre              0.364629
Religieux            0.321380
dtype: float64
```

L’image ci-dessus présente la dispersion des variables dans chacune de nos attributs. Ces valeurs représentent les moyennes quadratiques des écarts par rapport à la moyenne.

d) **L’amplitude :**

```
maximum=data.max()
maximum
```

```
UserID              User99
Galerie_Art           3.22
Club_Danse            3.64
Bar_Jus               3.62
Restaurant            3.44
Musees                 3.3
Station_Vacances      3.76
Parc_Pique-nique      3.21
Plage                 3.39
Theatre               3.17
Religieux             3.66
dtype: object
```

L’image ci-haut nous donne la valeur maximum dans chacune de nos attributs.

e) **Le minimum :** L’image les valeurs ci-dessous nous donne la valeur minimum dans chacun de nos attributs. Nous pouvons constater les Parc_Pic-nique, les Plage et les religieux sont apprécié car leurs sont supérieures à la moyenne qui est de 2

```
minimum=data.min()
minimum
```

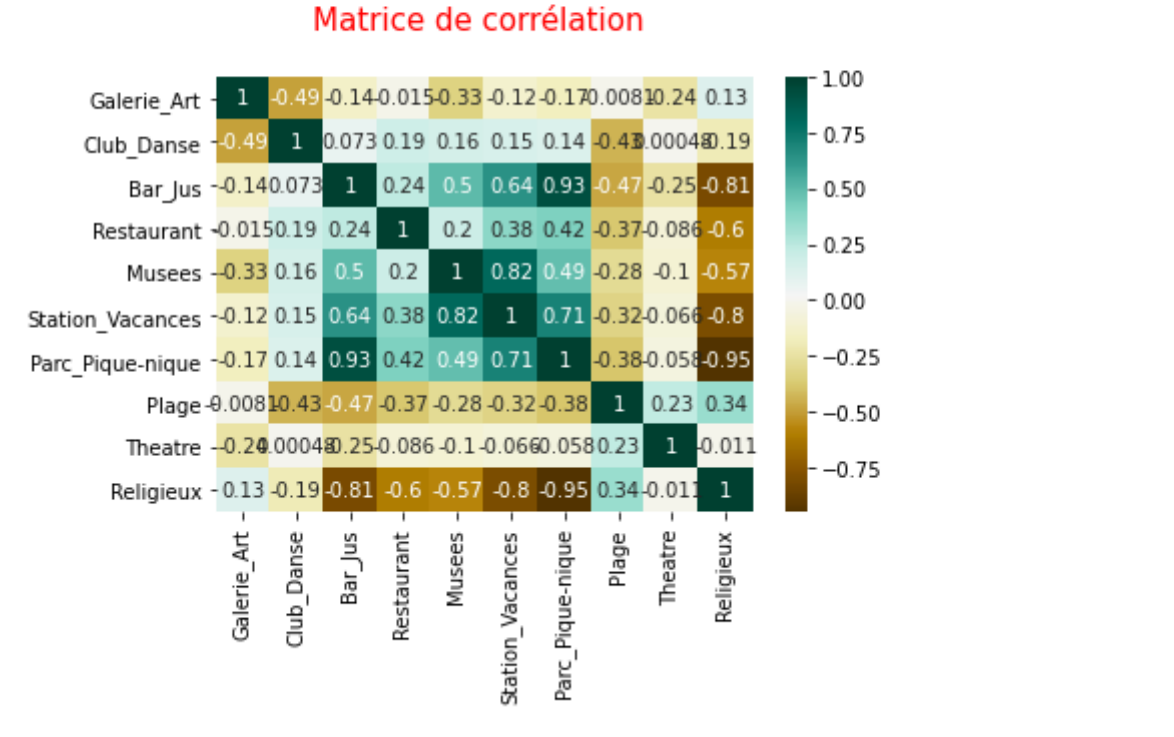
```
UserID              User1
Galerie_Art          0.34
Club_Danse              0
Bar_Jus              0.13
Restaurant           0.15
Musees               0.06
Station_Vacances     0.14
Parc_Pique-nique     3.16
Plage                2.42
Theatre              0.74
Religieux            2.14
dtype: object
```

f) La matrice de corrélation

La **matrice** des **corrélations** est tout simplement la **matrice** des coefficients de **corrélation** statistiques ou de coefficients de **corrélation** stochastiques calculés sur plusieurs variables prises deux à deux. En général, il s'agit des coefficients de Pearson.
Donc, sa taille est égale au nombre de variables. Ci-dessous nous représentons le résultat de notre matrice.

```
import seaborn as sns
MatrCorr=data.corr(method ='pearson')
plt.rcParams["font.size"]=10
sns.heatmap(MatrCorr.corr(), annot=True, cmap="BrBG")
plt.title("Matrice de corrélation "+"\n",fontsize=15,color='red')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:
  import pandas.util.testing as tm
Text(0.5, 1.0, 'Matrice de corrélation \n')
```



Partant de notre résultat ci-haut nous pouvons affirmer ce qui suit :
- La corrélation d'un attribut par rapport à lui même est 1, ce qui prouve qu'un attribut est fortement corrélé à lui même. Par exemple Galerie d'art par rapport à Galerie d'art la corrélation est 1.
Nous continuons notre interprétation par rapport à notre target qui est Station_Vacances. Partant de nos résultats nous affirmons :

- La corrélation entre Station de Vacances et Galerie d'art est de **-0.12** ce qui prouve que le deux valeurs ne sont pas fortement corrélé.

- La corrélation entre Station de Vacances et Clubs de danses est de **0.15** ce qui prouve que le deux valeurs ne sont pas fortement corrélé.
- La corrélation entre Station de Vacances et Bars Jus est de **0.64** ce qui prouve que le deux valeurs sont fortement corrélé.

- La corrélation entre Station de Vacances et Restaurant est de **0.38** ce qui prouve que le deux valeurs sont faiblement corrélé.

- La corrélation entre Station de Vacances et Musées est de **0.82** ce qui prouve que le deux valeurs sont assez corrélé.

- La corrélation entre Parc Pique-nique et Station des vacances est de **0.71** ce qui prouve que le deux valeurs sont fortement corrélé.

- La corrélation entre Station de Vacances et plage est de **-0.32** ce qui prouve que le deux valeurs ne sont pas fortement corrélé.

- La corrélation entre Station de Vacances et Théâtre est de **-0.066** ce qui prouve que le deux valeurs ne sont pas corrélés.

- La corrélation entre Station de Vacances et Religieux est de **-0.8** ce qui prouve que le deux valeurs sont fortement corrélé.

II – 4. Conclusion

Nous avons à présent effectuer l'étude de notre jeux de données et nous allons poursuivre notre étude dans la troisième partie en faisant l'analyse de composant principales.

TROISIÈME PARTIE : l'Analyse en Composant Principal (ACP)

III .1. Introduction

l'Analyse de Composant Principales (ACP) qui est une méthode de la famille de l'analyse des données et plus généralement de la statistique multivariée, qui consiste à transformer des variables liées entre elles (dites « corrélées » en statistique) en nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales », ou axes principaux. Elle permet au praticien de réduire le nombre de variables et de rendre l'information moins redondante.

III.2 Séparation de données de notre cible

Comme variable cible, nous avons choisi l'attribut Station_Vacances. Et pour commencer l'Analyse en composant Principal, nous devons séparer notre variable cible des autres variables

```
# separation en données features et en données cibles
columns_for_differencing = ['Station_Vacances']
X = data.copy()[data.columns.difference(columns_for_differencing)]
X
```

	Bar_Jus	Club_Danse	Galerie_Art	Musees	Parc_Pique-nique	Plage	Religieux	Restaurant	Theatre	UserID
0	2.29	1.80	0.93	0.80	3.19	2.79	2.42	0.62	1.82	User1
1	2.66	2.20	1.02	1.42	3.21	2.63	2.32	0.64	1.86	User2
2	0.54	0.80	1.22	0.24	3.18	2.80	2.50	0.53	1.31	User3
3	0.29	1.80	0.45	0.46	3.18	2.96	2.86	0.57	1.57	User4
4	1.18	1.20	0.51	1.54	3.18	2.78	2.54	0.57	1.18	User5
...	...	...	...	...	...	...	...	...	...	...
975	0.30	1.12	0.74	0.88	3.17	2.78	3.20	0.53	0.99	User976
976	1.12	0.92	1.25	0.78	3.18	2.79	2.80	0.38	1.34	User977
977	0.67	1.32	0.61	1.30	3.17	2.81	3.02	0.43	1.34	User978
978	0.13	0.20	0.93	0.30	3.18	2.98	2.46	0.43	1.12	User979
979	1.13	0.56	0.93	1.34	3.18	2.87	2.40	0.51	1.34	User980

980 rows × 10 columns

Nous avons extrait ci-dessus notre attribut cible de nos données.

III-3. Les Librairies utilisés

- **Pandas** est une librairie python qui permet de manipuler facilement des données à analyser dans des tableaux qui sont appelés Data Frames.

-**NumPy** est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

-**Matplotlib** est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.

-**Seaborn** est une librairie qui vient s'ajouter à Matplotlib pour remplancait certains réglages par défaut et fonctions.

-Sklearn est une bibliothèque libre Python destinée à l'apprentissage automatique.

-StandardScaler fournit plusieurs fonctions d'utilité communes et classes de transformateurs pour changer les vecteurs de caractéristiques bruts en une représentation qui convient mieux aux estimateurs en aval.

-PCA est une application de la visualisation des données.

III.4 Préparation de données

Nous devons explicitement centrer et réduire les variables pour réaliser une ACP normée avec PCA. Nous utilisons la classe StandardScaler pour ce faire.

```
#scikit-learn
import sklearn
```

Nous instancions l'objet et nous l'appliquons sur la matrice X. Nous obtenons une matrice Z.

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j}$$

Où $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ est la moyenne de la variable X_j, $\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}$ son écart-type.

```
#classe pour standardisation
from sklearn.preprocessing import StandardScaler

#instanciation
sc = StandardScaler()
#transformation – centrage-réduction
Z = sc.fit_transform(X)
print(Z)
dataX.describe()
```

```
[[ 1.6197497   0.93588703  0.11264465 ...  0.31295986  1.07016867
   0.68751758]
 [ 2.08917107  1.77264397  0.38808847 ...  0.38449354  2.47950035
   0.797274  ]
 [-0.60048651 -1.15600533  1.00018585 ... -0.00894171 -0.56168906
  -0.71187679]
 ...
 [-0.43555468 -0.06822131 -0.86671115 ... -0.36661012 -0.11663696
  -0.62955947]
 [-1.12065614 -2.41114075  0.11264465 ... -0.36661012 -2.67568658
  -1.23321978]
 [ 0.14805027 -1.6580595   0.11264465 ... -0.08047539  0.95890564
  -0.62955947]]
```

III-5 .Lancement de l'Analyse en Composant Principal(ACP)

Pour lancer l'ACP, nous commençons par les import de bibliothèques et de fonction nécessaire

```
#lancement de l'ACP
#classe pour l'ACP
from sklearn.decomposition import PCA

#instanciation
acp = PCA(svd_solver='full')

#affichage des paramètres
print(acp)
```

```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
    svd_solver='full', tol=0.0, whiten=False)
```

Brève Interprétation du code et du résultat

Le paramètre (svd_solver = 'full') indique l'algorithme utilisé pour la décomposition en valeurs singulières. Nous choisissons la méthode ''exacte'', sélectionnée de toute manière par défaut pour l'appréhension des bases de taille réduite. D'autres approches sont disponibles pour le traitement des grands ensembles de données. Le nombre de composantes (K) n'étant pas spécifié (n_components = None), il est par défaut égal au nombre de variables (K = p). Nous pouvons lancer les traitements dans un second temps. La fonction fit_transform() renvoie en sortie les coordonnées factorielles Fik que nous collectons dans la variable coord [TUTO_R, page 4 pour lancer l'ACP, page 7 pour la récupération du champs $scores des coordonnées factorielles]. Nous affichons le nombre de composantes générées (K), il est bien égal à p = 6.

III-6. Valeurs propres et scree plot

Affichons la variance expliquée

```
#variance expliquée
print(acp.explained_variance_)
n
```

```
[2.35990983 1.2505265  1.22150854 1.06206656 0.86563709 0.79096856
 0.67078011 0.45924778 0.32854809]
980
```

En comparant nos valeurs avec, nous rendons compte que nous avons pas les mêmes valeurs d'où il nous faut appliquer une correction.

III- 7 . Correction des valeurs formules et résultats

```
eigval = (n-1)/n*acp.explained_variance_
print(eigval)
```

```
[2.35750176 1.24925045 1.2202621  1.06098282 0.86475378 0.79016145
 0.67009564 0.45877916 0.32821284]
```

Notons que le PCA fournit également les proportions de variance associées aux axes. Il n'est pas nécessaire d'effectuer une correction dans ce cas.

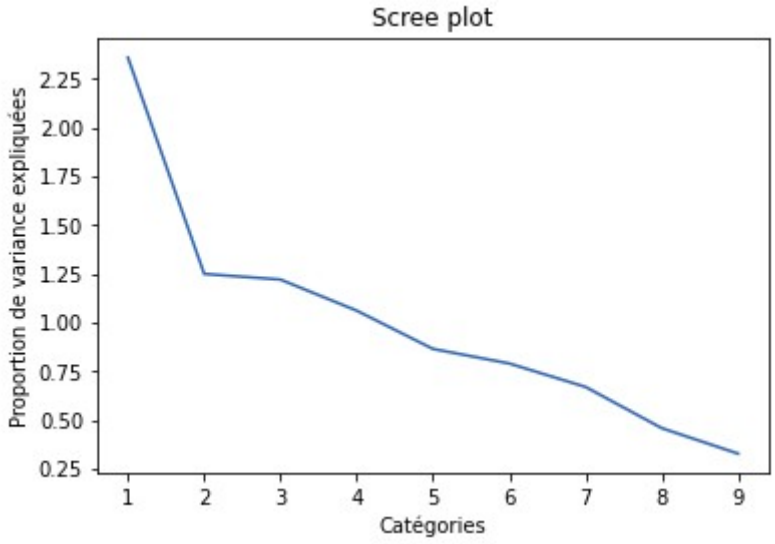
Proportion de variance expliquée

```
#proportion de variance expliquée
print(acp.explained_variance_ratio_)
```

```
[0.26194464 0.13880561 0.13558468 0.11788698 0.09608375 0.08779572
 0.07445507 0.05097546 0.03646809]
```

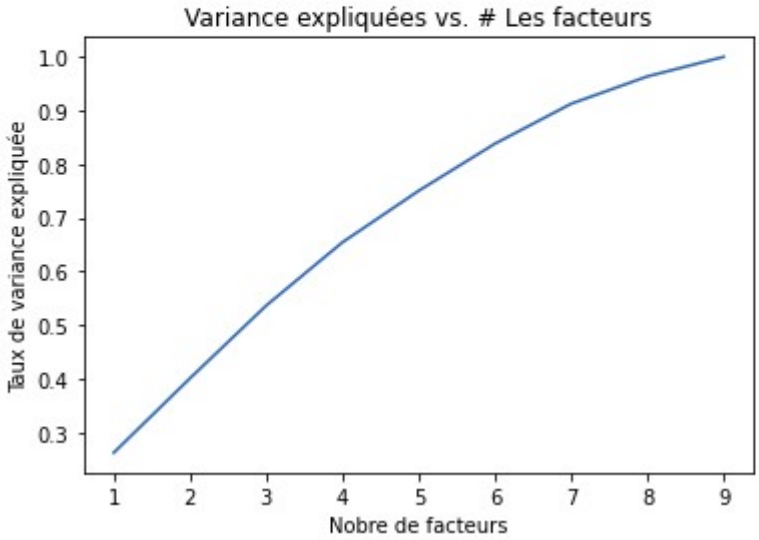
Il Notons que la première composante s'accapare de 26.19 % de l'information disponible et avec le deux premiers nous disposons de 40 %. Les suivants semblent anecdotiques.

III-8. Graphique Scree Plot



Le graphique du cumul de variance restituée selon le nombre de facteurs peut être intéressant également, variance expliquées contre le nombre de facteur.

```
plt.plot(np.arange(1,p+1),np.cumsum(acp.explained_variance_ratio_))
plt.title("Variance expliquées vs. # Les facteurs")
plt.ylabel("Taux de variance expliquée")
plt.xlabel("Nombre de facteurs")
plt.show()
```



III-9. Détermination du nombre de facteur à retenir

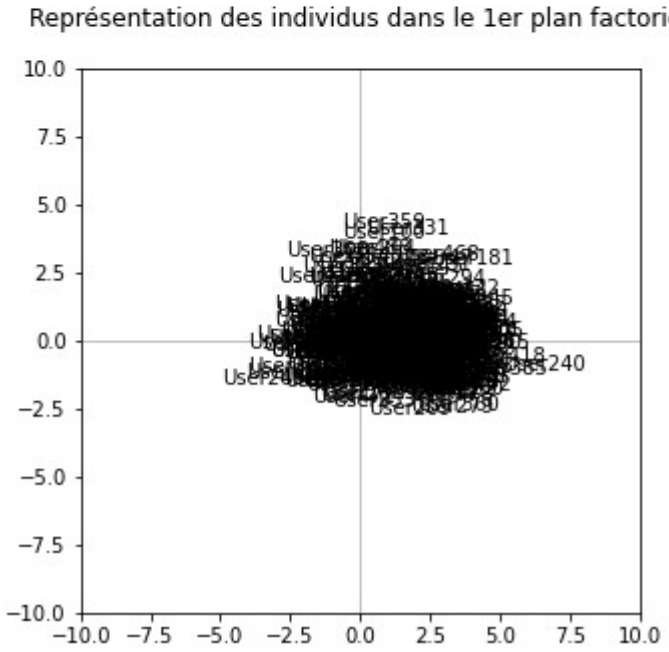
```
#seuils pour test des bâtons brisés
bs = 1/np.arange(p,0,-1)
bs = np.cumsum(bs)
bs = bs[::-1]
#test des bâtons brisés
print(pd.DataFrame({'Val.Propre':eigval,'Seuils':bs}))
```

```
   Val.Propre    Seuils
0    2.357502  2.828968
1    1.249250  1.828968
2    1.220262  1.328968
3    1.060983  0.995635
4    0.864754  0.745635
5    0.790161  0.545635
6    0.670096  0.378968
7    0.458779  0.236111
8    0.328213  0.111111
```

Avec cette procédure, seul le quatre premiers facteurs sont valides. Le cercle des corrélations que nous construirons par la suite semble aller dans le même sens. Néanmoins, par commodité (pas seulement en réalité, cette étude est plus subtile qu'elle n'en a l'air, nous choisissons K* = 2 pour pouvoir représenter les individus et les variables dans le plan.

III-10 Représentation des individus dans le plan

```
fig, axes = plt.subplots(figsize=(5,5))
axes.set_xlim(-10,10) #même limites en abscisse
axes.set_ylim(-10,10) #et en ordonnée
#placement des étiquettes des observations
for i in range(n//2):
    plt.annotate(X.index[i],(coord[i,0],coord[i,1]))
#ajouter les axes
plt.plot([-10,10],[0,0],color='silver',linestyle='-',linewidth=1)
plt.plot([0,0],[-10,10],color='silver',linestyle='-',linewidth=1)
#affichage
plt.title("Représentation des individus dans le 1er plan factoriel\n")
plt.show()
#
```



Qualité de représentation – Les COS² (cosinus carré). Pour calculer la qualité de représentation des individus sur les axes, nous devons d'abord calculer les carrés des distances à l'origine des individus, qui correspondent également à leur contribution dans l'inertie totale.

$$d_i^2 = \sum_{j=1}^{p} z_{ij}^2$$

III-11 Contribution des individus dans l'inertie totale

```
di = np.sum(Z**2,axis=1)
print(pd.DataFrame({'Id User':X.index,'d_i':di}))
```

```
     Id User        d_i
0      User1   6.831492
1      User2  20.247611
2      User3   7.013983
3      User4   6.000525
4      User5   5.486511
..       ...        ...
975  User976   6.276252
976  User977   3.055867
977  User978   2.674554
978  User979  20.265183
979  User980   6.553556

[980 rows x 2 columns]
```

Concrètement, users 2et 979 sont les deux individus qui se démarquent le plus des autres, et on les retrouve aux deux extrémités du premier axe factoriel qui porte 59.68% de l'information disponible.
Nous pouvons alors déduire la qualité de représentation des individus sur l'axe :

```
#qualité de représentation des individus - COS2
cos2 = coord**2
for j in range(p):
    cos2[:,j] = cos2[:,j]/di
print(pd.DataFrame({'id':X.index,'COS2_1':cos2[:,0],'COS2_2':cos2[:,1]}))
```

```
          id    COS2_1    COS2_2
0      User1  0.576030  0.002114
1      User2  0.794762  0.005051
2      User3  0.149903  0.177866
3      User4  0.219756  0.040555
4      User5  0.294075  0.053494
..       ...       ...       ...
975  User976  0.390421  0.209983
976  User977  0.113106  0.218526
977  User978  0.034846  0.002784
978  User979  0.367143  0.031622
979  User980  0.208557  0.002035

[980 rows x 3 columns]
```

Conformément à la théorie, pour chaque individu, la somme des COS² sur l’ensemble des facteurs est égale à 1.
vérifions la théorie - somme en ligne des cos2 = 1

```python
#Conformément à la théorie, pour chaque individu, la somme des COS² sur l’ensemble des
#facteurs est égale à 1.
#vérifions la théorie - somme en ligne des cos2 = 1
print(np.sum(cos2,axis=1))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
```

III-12.Contribution des individus aux axes

```
#contributions aux axes
ctr = coord**2
for j in range(p):
    ctr[:,j] = ctr[:,j]/(n*eigval[j])

print(pd.DataFrame({'id':X.index,'CTR_1':ctr[:,0],'CTR_2':ctr[:,1]}))
```

```
          id     CTR_1     CTR_2
0      User1  0.001703  0.000012
1      User2  0.006965  0.000084
2      User3  0.000455  0.001019
3      User4  0.000571  0.000199
4      User5  0.000698  0.000240
..       ...       ...       ...
975  User976  0.001061  0.001076
976  User977  0.000150  0.000545
977  User978  0.000040  0.000006
978  User979  0.003220  0.000523
979  User980  0.000592  0.000011

[980 rows x 3 columns]
```

Elles permettent de déterminer les individus qui pèsent le plus dans la définition de chaque facteur.

III-13. Contribution aux axes

```
ctr = coord**2
for j in range(p):
    ctr[:,j] = ctr[:,j]/(n*eigval[j])

print(pd.DataFrame({'id':X.index,'CTR_1':ctr[:,0],'CTR_2':ctr[:,1]}))
```

```
          id     CTR_1     CTR_2
0      User1  0.001703  0.000012
1      User2  0.006965  0.000084
2      User3  0.000455  0.001019
3      User4  0.000571  0.000199
4      User5  0.000698  0.000240
..       ...       ...       ...
975  User976  0.001061  0.001076
976  User977  0.000150  0.000545
977  User978  0.000040  0.000006
978  User979  0.003220  0.000523
979  User980  0.000592  0.000011

[980 rows x 3 columns]
```

Sans surprises, ce sont User 2 et User 979 qui sont déterminants pour le premier axe ; pour le second, nous avons user 3 et user et user 975.

Les sommes en ligne sont égales à l'unité :

```
print(np.sum(ctr,axis=0))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

III-14. Représentation des variables – Outils pour l'aide à l'interprétation

```
#le champ components_ de l'objet ACP
print(acp.components_)
```

```
[[-0.4163454  -0.18137721  0.02477384 -0.43830046  0.13880592  0.47940828
  -0.28627037 -0.51812832 -0.03912446]
 [-0.2255815  -0.02102377 -0.08548347  0.2397793   0.64667015  0.07074891
  -0.09830377  0.22699372  0.63205378]
 [ 0.14980448 -0.64537701  0.67099371 -0.02591534  0.23991104 -0.08574923
   0.01728753  0.14649927 -0.15320521]
 [-0.31476433  0.18724912  0.3089702  -0.35025399 -0.06674714 -0.15730476
   0.72263099 -0.08674679  0.29799366]
 [-0.26481745  0.43450108  0.51302881  0.33046448 -0.15184435  0.48436374
  -0.09989168  0.27974768 -0.15090775]
 [-0.34286005 -0.07115964 -0.29236999  0.22779889  0.41398839  0.02320228
   0.41268085  0.04560206 -0.62904767]
 [-0.36797958 -0.55016778 -0.18198536  0.38418075 -0.54131252  0.13005592
   0.13563137  0.03785147  0.22678299]
 [ 0.56926037 -0.0449495  -0.03938952  0.2047249   0.09577559  0.58170441
   0.42292369 -0.29618656  0.12772801]
 [ 0.08428462 -0.12855607 -0.25106716 -0.52577837 -0.06247268  0.37765517
   0.09136243  0.69392531 -0.02204858]]
```

Nous avons besoin des vecteurs propres pour l'analyse des variables. Ils sont fournis par le champ .components_

```
#racine carrée des valeurs propres
sqrt_eigval = np.sqrt(eigval)

#corrélation des variables avec les axes
corvar = np.zeros((p,p))
for k in range(p):
    corvar[:,k] = acp.components_[k,:] * sqrt_eigval[k]

#afficher la matrice des corrélations variables x facteurs
print(corvar)
```

```
[[-0.63926331 -0.25213216  0.16548222 -0.32421991 -0.24625941 -0.30477182
  -0.30122577  0.38557848  0.04828655]
 [-0.27848943 -0.02349824 -0.71291878  0.19287412  0.40405185 -0.06325453
  -0.45036387 -0.03044575 -0.0736496 ]
 [ 0.03803814 -0.09554477  0.74121638  0.31825173  0.47707646 -0.25989068
  -0.14897207 -0.02667979 -0.14383603]
 [-0.67297346  0.26800102 -0.02862751 -0.36077569  0.30730598  0.20249277
   0.31448794  0.13866681 -0.30121771]
 [ 0.21312481  0.72278241  0.26501887 -0.06875224 -0.14120331  0.36799853
  -0.44311502  0.06487191 -0.03579051]
 [ 0.73609106  0.07907596 -0.0947233  -0.16203023  0.4504202   0.02062475
   0.10646296  0.39400724  0.21635813]
 [-0.43954405 -0.109874    0.01909675  0.74433897 -0.09289141  0.36683624
   0.11102698  0.28645991  0.05234141]
 [-0.79554242  0.25371059  0.16183111 -0.08935268  0.26014335  0.04053614
   0.03098498 -0.20061675  0.39754885]
 [-0.06007232  0.7064457  -0.16923886  0.30694546 -0.14033235 -0.55916693
   0.18564313  0.08651432 -0.0126316 ]]
```

III-15. Affichage pour les deux premiers axes

```
    #on affiche pour les deux premiers axes
print(pd.DataFrame({'id':X.columns,'COR_1':corvar[:,0],'COR_2':corvar[:,1]}))
```

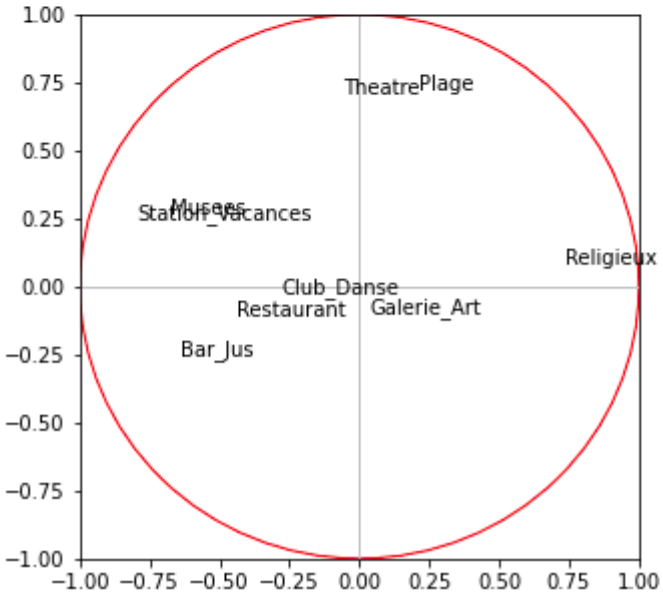
```
                id     COR_1     COR_2
0          Bar_Jus -0.639263 -0.252132
1       Club_Danse -0.278489 -0.023498
2      Galerie_Art  0.038038 -0.095545
3           Musees -0.672973  0.268001
4            Plage  0.213125  0.722782
5         Religieux  0.736091  0.079076
6       Restaurant -0.439544 -0.109874
7  Station_Vacances -0.795542  0.253711
8          Theatre -0.060072  0.706446
```

De ce point de vue, les résultats sont complètement cohérents. Nous pouvons dessiner maintenant le cercle des corrélations.

III-16.Cercle de Corrélation

```
#cercle des corrélations
fig, axes = plt.subplots(figsize=(5,5))
axes.set_xlim(-1,1)
axes.set_ylim(-1,1)
#affichage des étiquettes (noms des variables)
for j in range(p):
    plt.annotate(X.columns[j],(corvar[j,0],corvar[j,1]))

#ajouter les axes
plt.plot([-1,1],[0,0],color='silver',linestyle='-',linewidth=1)
plt.plot([0,0],[-1,1],color='silver',linestyle='-',linewidth=1)
#ajouter un cercle
cercle = plt.Circle((0,0),1,color='red',fill=False)
axes.add_artist(cercle)
#affichage
plt.show()
```



On s’aperçoit que Musées et station de vacances sont corrélé, théâtre et plage, Club de danse,restaurant, gallérie d’art et Bar jus le sont aussi, tandis que Religieux n’est le pas.

III-17.Qualité de représentation des variables (COS^2)

```
#cosinus carré des variables
cos2var = corvar**2
print(pd.DataFrame({'id':X.columns,'COS2_1':cos2var[:,0],'COS2_2':cos2var[:,1]}))
```

```
                id    COS2_1    COS2_2
0          Bar_Jus  0.408658  0.063571
1       Club_Danse  0.077556  0.000552
2      Galerie_Art  0.001447  0.009129
3           Musees  0.452893  0.071825
4            Plage  0.045422  0.522414
5         Religieux  0.541830  0.006253
6       Restaurant  0.193199  0.012072
7  Station_Vacances  0.632888  0.064369
8          Theatre  0.003609  0.499066
```

de danse,restaurant, gallérie d'art et Bar jus le sont aussi, tandis que Religieux n'est le pas.

On peut calculer la qualité de représentation des variables en montant la corrélation au carré : $COS^2jk = r^2jk$.

```
#cosinus carré des variables
cos2var = corvar**2
print(pd.DataFrame({'id':X.columns,'COS2_1':cos2var[:,0],'COS2_2':cos2var[:,1]}))
```

```
                id    COS2_1    COS2_2
0          Bar_Jus  0.408658  0.063571
1       Club_Danse  0.077556  0.000552
2      Galerie_Art  0.001447  0.009129
3           Musees  0.452893  0.071825
4            Plage  0.045422  0.522414
5         Religieux  0.541830  0.006253
6       Restaurant  0.193199  0.012072
7  Station_Vacances  0.632888  0.064369
8          Theatre  0.003609  0.499066
```

VI. QUATRIÈME PARTIE : l'Apprentissage automatique supervisé

IV.1.Introduction au problème

De nos jours, l'apprentissage automatique est une technique beaucoup utilisée pour résoudre des problèmes et prévoir des solutions dans la prédiction, et ce dans plusieurs domaines d'application. Dans notre cas nous sommes face à un domaine de sondage des avis de plusieurs voyageurs.

Nous voudrions utilisé l'apprentissage automatique pour prédire les avis sur un type de lieu visité par les voyages. Pour ce faire nous allons nous suivrons les points suivants :

- ➔ Présentation de la méthode étudiée
- ➔ Préparation de données
- ➔ Choix des paramètres et application de la méthode étudiée au jeu de données et évaluation
- ➔ Comparaison avec le résultat d'application d'une autre méthode d'apprentissage supervisé

IV. 2. Présentation de la méthode étudiée

L'apprentissage supervisé est une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir d'exemples annotés. On distingue les problèmes de régression des problèmes de classement1. L'apprentissage automatique supervisé utilise plusieurs méthodes tel que le Boosting, Machine à vecteurs de support (SVM), Réseau de neurones artificiels ,Méthode des k plus proches voisins, Arbre de décision,…

Dans l'apprentissage supervisé, on considère que les problèmes de prédiction d'une variable quantitative sont des problèmes de régression tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification.

Dans notre étude, vus la nature de nos données (variables quantitatives), nous utiliserons la méthode des K plus proches voisins (KNN - k Nearest Neighbors) ?

IV. 2.a) Présentation de la méthode étudiée : l'algorithme des K plus proche voisins(KNN)

L'algorithme K-NN (K-nearest neighbors) est une méthode d'apprentissage supervisé. Il peut être utilisé aussi bien pour la régression que pour la classification. Son fonctionnement peut être assimilé à l'analogie suivante "dis moi qui sont tes voisins, je te dirais qui tu es…".

Fonctionnement et principe pour la prédiction : Pour effectuer une prédiction, l'algorithme K-NN :

- ne calcule pas un modèle prédictif à partir d'un Training Set comme c'est le cas pour la régression logistique ou la régression linéaire.
- n'a pas besoin de construire un modèle prédictif. Ainsi, pour K-NN il n'existe pas de phase d'apprentissage proprement dite. C'est pour cela qu'on le catégorise parfois dans le Lazy Learning. ,
- K-NN se base sur le jeu de données pour produire un résultat.
-

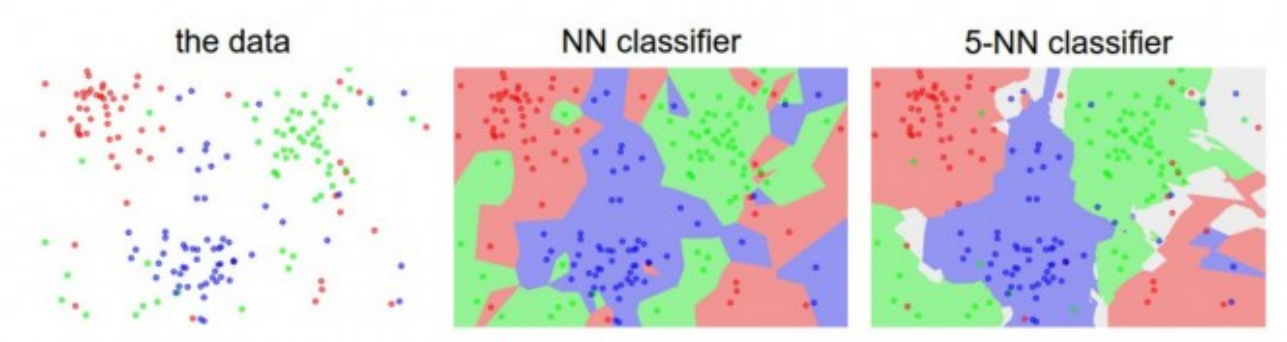
Le Principe de K-NN est basé sur: « ***dis moi qui sont tes voisins, je te dirais qui tu es !*** »

Comment K-NN effectue une prédiction ? Pour effectuer une prédiction, l'algorithme K-NN va se baser sur le jeu de données en entier. En effet, pour une observation, qui ne fait pas parti du jeu de données, qu'on souhaite prédire, l'algorithme va chercher les K instances du jeu de données les plus proches de notre observation. Ensuite pour ces K voisins, l'algorithme se basera sur leurs variables de sortie *(output variable) y pour calculer la valeur de la variable y de l'observation qu'on souhaite prédire.*

NB :

- Si K-NN est utilisé pour la régression(comme dans notre cas), c'est la moyenne (ou la médiane) des variables y des K plus proches observations qui servira pour la prédiction.
- Si K-NN est utilisé pour la classification, c'est le mode des variables y des K plus proches observations qui servira pour la prédiction

Comment choisir la valeur K: Le choix de la valeur K à utiliser pour effectuer une prédiction avec K-NN, varie en fonction du jeu de données. En règle générale, moins on utilisera de voisins (un nombre K petit) plus on sera sujette au sous apprentissage (underfitting). Par ailleurs, plus on utilise de voisins (un nombre K grand) plus, sera fiable dans notre prédiction. Toutefois, si on utilise K nombre de voisins avec K=N et N étant le nombre d'observations, on risque d'avoir du overfitting et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu.



Écriture algorithmique :

On peut schématiser le fonctionnement de K-NN en l'écrivant en pseudo-code suivant :

Début Algorithme
Données en entrée :

- un ensemble de données **D** .
- une fonction de définition distance **d**.
- Un nombre entier **K**

Pour une nouvelle observation **X** dont on veut prédire sa variable de sortie **y**
Faire :

1. Calculer toutes les distances de cette observation **X** avec les autres observations du jeu de données **D**
2. Retenir les **K** observations du jeu de données **D** les proches de **K** en utilisation le fonction de calcul de distance **d**
3. Prendre les valeurs de **y** des **K** observations retenues :
   1. Si on effectue une régression, calculer la moyenne (ou la médiane) de **y** retenues
   2. Si on effectue une classification , calculer le mode de **y** retenues
4. Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation **X**.

Fin Algorithme

Quelques limites de KNN

K-NN est un algorithme assez simple à appréhender. Principalement, grâce au fait qu'il n'a pas besoin de modèle pour pouvoir effectuer une prédiction. Le contre coût est qu'il doit garder en mémoire l'ensemble des observations pour pouvoir effectuer sa prédiction. Ainsi il faut faire attention à la taille du jeu d'entraînement.

Également, le choix de la méthode de calcul de la distance ainsi que le nombre de voisins K peut ne pas être évident. Il faut essayer plusieurs combinaisons et faire du tuning de l'algorithme pour avoir un résultat satisfaisant.

IV. 3. Préparation des données

La préparation des données est une étape clé avant la prédiction car la fiabilité de l'analyse des données dépend en très grande partie de la qualité des données. En effet elle consiste à :

- la collecte de données : consiste à acquérir les données
- le nettoyage de données:ce processus est fondamental à la préparation des données. Il permet d'améliorer la qualité des données en supprimant ou en modifiant les données erronées et a pour but est d'éviter de retrouver dans la base de données des données incorrectes ;
- l'enrichissement de données
- ou encore la fusion de données.

Au cours de la préparation des données, les données dites "brutes" sont soumises à différents traitements afin de les rendre exploitables pour l'étape d'Exploration de données, au cours de laquelle le but sera d'extraire des connaissances à partir des données via la construction de modèles.
Les données brutes peuvent comporté les des anomalies pour les raisons suivantes :

- ✗ Erreurs de saisies
- ✗ Erreurs lexicales
- ✗ Erreurs de formats
- ✗ Doublons
- ✗ Données manquante
- ✗ Erreurs sémantiques

Dans notre cas en fonction du type de nos données, nous nous sommes focalisés sur les données manquantes. Du fait que nos données contiennent des identifiants uniques, le risque de doublons ne se pose pas.

- -***Acquisition et description du data set*** : Notre étude est faite sur un ensemble de données en provenance du site d'entrepôt de données http://archive.ics.uci.edu/, dont l'accès direct est http://archive.ics.uci.edu/ml/datasets/Travel+Reviews. Cet ensemble de données contient le recueille des avis des 980 voyageurs sur différents endroits qu'ils visitent à travers l'Asie de l'Est. Les avis recueillis pour chaque voyageur sont classés de 0 à 4, détaillés comme suit

-***Traitement des données manquantes :*** Pour vérifier si notre data set contient de données manquantes nous avons exécuter le code ci-dessous :

```
Entrée [24]:  1  sum(data.isnull().values.ravel())
Out[24]: 0
```

Le résultat nous dit qu'il a 0 donnée manquante. Elles sont alors bonnes pour la suite des opérations.

IV. 4. Choix des paramètres et application de la méthode utilisée

A cette étape, nous avons à définir les paramètres que nous avons choisi pour l'étude.
Variable cible : Comme variable cible nous avons l'attribut Station_Vacance
Variable explicative : les autres attributs du data set.
Ce qui à motiver le choix de la variable à prédire:
En rappel : la valeur de chaque attribut est la moyenne des avis sur chaque lieu visité :

(0) :Terrible (1) : Médiocre (2) :Moyen (3) :Très bon (4) :Excellent

Par example : Si la valeur de Théâtre 3,42 pour un individu, cela signifie que ce sur tout les lieux de théâtre qu'il a visité on accueilli ces avis et la valeur 3,42 est la moyen de ces avis de cet individu là.
Analyse des valeurs moyenne de chaque attribut :

```
minimum=data.min()
minimum
```

```
UserID              User1
Galerie_Art          0.34
Club_Danse              0
Bar_Jus              0.13
Restaurant           0.15
Musees               0.06
Station_Vacances     0.14
Parc_Pique-nique     3.16
Plage                2.42
Theatre              0.74
Religieux            2.14
dtype: object
```

```
mediane=data.median()
mediane
```

```
Galerie_Art          0.83
Club_Danse           1.28
Bar_Jus              0.82
Restaurant           0.50
Musees               0.90
Station_Vacances     1.80
Parc_Pique-nique     3.18
Plage                2.82
Theatre              1.54
Religieux            2.78
dtype: float64
```

```
moyenne=data.mean()
moyenne
```

```
Galerie_Art          0.893194
Club_Danse           1.352612
Bar_Jus              1.013306
Restaurant           0.532500
Musees               0.939735
Station_Vacances     1.842898
Parc_Pique-nique     3.180939
Plage                2.835061
Theatre              1.569439
Religieux            2.799224
dtype: float64
```

```
maximum=data.max()
maximum
```

```
UserID              User99
Galerie_Art           3.22
Club_Danse            3.64
Bar_Jus               3.62
Restaurant            3.44
Musees                 3.3
Station_Vacances      3.76
Parc_Pique-nique      3.21
Plage                 3.39
Theatre               3.17
Religieux             3.66
dtype: object
```

Pour les besoins de l'étude, nous avons scindé les données en deux parties : Données d'entraînement constituent 80 % du data set et données de test : constitue 20 % du data set.

Le données de test nous permettra de tester notre modèle, en comparant les valeurs existantes et celles prédites.

Pour la validation de la performance du modèle, nous utiliserons une autre la méthode de l'arbre de décision pour et nous comparerons le résultat à travers les taux d'erreur de chacune des méthodes utilisées

IV. 5. Comparaison avec le résultat d'application d'une autre méthode d'apprentissage supervisé

- ***-Methode étudiée : KNN***

-Code du KNN pour la prédiction :

```
#Methode du KNN
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#import des données
dataset = pd.read_csv('data_set/data_set_travel.csv')
#afficcher un apercu descritif du datasets
dataset.describe();


#suppression de la variable cible deu data set
x = dataset.drop('Station_Vacances', axis=1)

#suppression de l'ideé user
X = x.drop('UserID', axis=1)

# definition de la variable cible
y = dataset['Station_Vacances']

from sklearn.model_selection import train_test_split
from sklearn import tree
#definition des données d'entrainement de des données de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

```

from sklearn.neighbors import KNeighborsRegressor
KNR = KNeighborsRegressor(20)
KNR.fit(X_train, y_train)

# prediction
y_pred = KNR.predict(X_test)

# affichage des valeur prédite avec celles existantes
df=pd.DataFrame({'Valeur actuelle':y_test, 'Valeur prédite':y_pred})
print(df)

#Affichage des taux d'erreur
from sklearn import metrics
print('Taux d\'erreur')
print('Moyennne absolue:', metrics.mean_absolute_error(y_test, y_pred))
print('Myenne carré:', metrics.mean_squared_error(y_test, y_pred))
print('Moyenne quadratique:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

-Résultat obtenu de KNN

```
     Valeur actuelle  Valeur prédite
440              1.50          1.5660
412              1.78          2.1205
331              1.86          2.0320
625              1.06          1.6280
578              1.26          1.5430
..                ...             ...
37               1.22          1.7720
644              3.06          2.3195
877              1.30          1.5950
817              1.74          1.6590
158              2.18          2.3335

[196 rows x 2 columns]
Taux d'erreur
Moyennne absolue: 0.27601785714285715
Myenne carré: 0.12344146811224489
Moyenne quadratique: 0.35134238018241537
```

- ***-Comparaison avec une autre Méthode : arbre de décision***

```
# prediction
y_pred = regressor.predict(X_test)

# affichage
df=pd.DataFrame({'Valeur Actuelle':y_test, 'Valeur Prédite':y_pred})
print(df)

#taux d'erreur
from sklearn import metrics
print('Taux d\'erreur')
print('Moyenne absolue:', metrics.mean_absolute_error(y_test, y_pred))
print('La moyenne carré:', metrics.mean_squared_error(y_test, y_pred))
print('Moyenne quadratique :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

text_representation = tree.export_text(clf)
print(text_representation)
```

```
# entrainement avec 80% des donnees
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
clf = regressor.fit(X_train, y_train)
```

```
# prediction
y_pred = regressor.predict(X_test)

# affichage
df=pd.DataFrame({'Valeur Actuelle':y_test, 'Valeur Prédite':y_pred})
print(df)

#taux d'erreur
from sklearn import metrics
print('Taux d\'erreur')
print('Moyenne absolue:', metrics.mean_absolute_error(y_test, y_pred))
print('La moyenne carré:', metrics.mean_squared_error(y_test, y_pred))
print('Moyenne quadratique :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

text_representation = tree.export_text(clf)
print(text_representation)
```

-Résultat obtenu de l'arbre de décision

```
     Valeur Actuelle  Valeur Prédite
440             1.50            1.38
412             1.78            2.12
331             1.86            2.66
625             1.06            1.70
578             1.26            0.98
..               ...             ...
37              1.22            1.62
644             3.06            2.74
877             1.30            1.80
817             1.74            2.08
158             2.18            2.17

[196 rows x 2 columns]
Taux d'erreur
Moyenne absolue: 0.33867346938775506
La moyenne carré: 0.20628163265306124
Moyenne quadratique : 0.45418237818420615
```

- ***-Comparaison des résultats obtenus***

Taux d'erreur	***KNN***	***Arbre de décision***
La moyenne absolue	*0.27601785714285715*	*0.35714285714285726*
La moyenne carré	*0.12344146811224489*	*0.22096326530612248*
Moyenne quadratique	*0.35134238018241537*	*0.47006729869894426*

Selon les taux d'erreur, la méthode KNN est renvoie mieux de bon résultat que l'arbre de décision

Conclusion générale

Au therme de notre étude, nous avons pu apprendre à manipuler un data set, de son acquisition à la préparation des données. Nous avons les techniques pour cela et que ces techniques d'apprentissage supervisé et non supervisé. Et chaque technique est utilisé en fonction du type de données à prédire. Aussi nous avons appris à faire la comparaison des résultats obtenus en utilisant une autre méthode d'apprentissage automatique.

Tout en étant conscient des difficultés rencontrés d'ordre technique, vu que nous venons que le domaine de fouille de données est nouveau pour nous, nous restons dans la perspective de d'approfondir nos recherches dans ce domaine pour plus d'efficacité, et de bon résultat.