AVIGNON UNIVERSITÉ

TP 2 : Panorama des méthodes de fouille de données

13 octobre 2023

Master 2
Intelligence Artificielle
UE Business Intelligence
ECUE Data Analytics & Big Data

Réalisé par :

AIT-ABBOU Samir

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE
ceri.univ-avignon.fr

Introduction

L'objectif de cette séance est de passer en revue les différents types de méthodes utilisées en fouille de données, que ce soit en analyse exploratoire ou en analyse prédictive

1 Analyse exploratoire

1.1 Classification / Clustering

Dans cette première partie, nous allons à étudier les données covid19 établies à partir des fichiers téléchargées depuis le site Web du Centre européen de prévention et de contrôle des maladies 1, une agence de l'Union européenne. Ces données fournissent des indicateurs de santé liées à la pandémie de COVID-19 sur 30 pays de l'espace économique européen. nous trouverons un descriptif des variables étudiées dans le fichier covid19.v2.txt

1- Ouvrez le fichier covid19.v2.csv à l'aide de la bibliothèque pandas. Combien y a-t-il d'attributs ? De quels types sont-ils ?

```
Nombre d'attributs : 18

Types d'attributs :
country                  object
country_code             object
population                int64
vaccine_2021-winter       int64
vaccine_2021-spring       int64
vaccine_2021-summer       int64
vaccine_2021-fall         int64
vaccine_2022-winter       int64
testing_rate_2020       float64
testing_rate_2021       float64
deaths_rate-2020-spring float64
deaths_rate-2020-summer float64
deaths_rate-2020-fall   float64
deaths_rate-2021-winter float64
deaths_rate-2021-spring float64
deaths_rate-2021-summer float64
deaths_rate-2021-fall   float64
deaths_rate-2022-winter float64
dtype: object
```

2- Y a-t-il des valeurs manquantes dans les données ?

```
Nombre de valeurs manquantes par attribut :
country                  0
country_code             0
population               0
vaccine_2021-winter      0
vaccine_2021-spring      0
vaccine_2021-summer      0
vaccine_2021-fall        0
vaccine_2022-winter      0
testing_rate_2020        0
testing_rate_2021        0
deaths_rate-2020-spring  0
deaths_rate-2020-summer  0
deaths_rate-2020-fall    0
deaths_rate-2021-winter  0
deaths_rate-2021-spring  0
deaths_rate-2021-summer  0
deaths_rate-2021-fall    0
deaths_rate-2022-winter  0
dtype: int64
```

Donc pas de valeurs manquantes !

Comment ces valeurs sont-elles représentées dans le ficher ?

Dans le fichier CSV, les valeurs manquantes sont généralement représentées par un champ vide (c'est-à-dire deux délimiteurs de virgule consécutifs) ou par "NaN" (Not-a-Number) ou "None", qui est une représentation courante des valeurs manquantes dans les données numériques en pandas.

Le nombre de doses pour 1 000 habitants

	country	country_code	population	vaccine_2021-winter	vaccine_2021-spring	vaccine_2021-summer	vaccine_2021-fall	vaccine_2022-winter
0	Austria	AT	8901064	711326	5070308	4528419	3686962	2744560
1	Belgium	BE	11522440	870773	6802244	7639306	2677905	4549303
2	Bulgaria	BG	6951482	212445	1251970	877732	1026556	818245
3	Croatia	HR	4058165	243767	1770881	1336909	893535	793016
4	Cyprus	CY	888005	75022	564483	436540	232256	305687

```python
# Liste des années de vaccination
annees_vaccination = ["2021-winter", "2021-spring", "2021-summer", "2021-fall", "2022-winter"]

for annee in annees_vaccination:
    # Calculer le nombre de doses pour 1 000 habitants dans la colonne existante
    covid[f'vaccine_{annee}'] = (covid[f'vaccine_{annee}'] / covid["population"]) * 1000

# Supprimer la colonne "population"
covid = covid.drop("population", axis=1)
```

	country	country_code	vaccine_2021-winter	vaccine_2021-spring	vaccine_2021-summer	vaccine_2021-fall	vaccine_2022-winter
0	Austria	AT	79.914716	569.629429	508.750302	414.215874	308.340666
1	Belgium	BE	75.571927	590.347531	662.993776	232.407806	394.821149
2	Bulgaria	BG	30.561109	180.101164	126.265450	147.674410	117.707994
3	Croatia	HR	60.068282	436.374815	329.436827	220.182028	195.412459
4	Cyprus	CY	84.483759	635.675475	491.596331	261.548077	344.240179

Les valeurs dans les colonnes "vaccine" ont été modifiées pour représenter le nombre de doses pour 1 000 habitants, et la colonne "population" a été supprimée. Les données sont prêtes pour la suite de l'analyse.

3- Quel est l'effet du filtre StandardScaler sur les données ?

Le filtre StandardScaler (également connu sous le nom de Standardization) est couramment utilisé pour normaliser les données avant d'appliquer des méthodes de clustering. Son effet sur les données est de les transformer de telle sorte que chaque attribut (caractéristique) ait une moyenne de 0 et un écart-type de 1. Cela signifie que les données sont centrées autour de zéro et ont des écarts types égaux.

Les avantages de l'utilisation de StandardScaler avant le clustering sont les suivants :

- **Égalisation d'échelle**: Lorsque on effectue des opérations de clustering, la distance entre les points de données est souvent utilisée comme mesure de similarité. Si les échelles de différentes caractéristiques varient considérablement, certaines caractéristiques peuvent avoir plus d'influence sur la mesure de distance que d'autres. En standardisant les données, on égalise les échelles et on traite toutes les caractéristiques de manière équitable.

- **Moyenne zéro et écart-type 1**: La standardisation permet de centrer les données autour de zéro, ce qui peut faciliter l'interprétation des résultats du clustering. Un écart-type de 1 signifie que la variabilité des données est conservée tout en les mettant à la même échelle.

- **Meilleure convergence**: Lorsque on effectue des méthodes de clustering qui dépendent de la distance, telles que K-means, une échelle équilibrée peut contribuer à une meilleure convergence de l'algorithme.

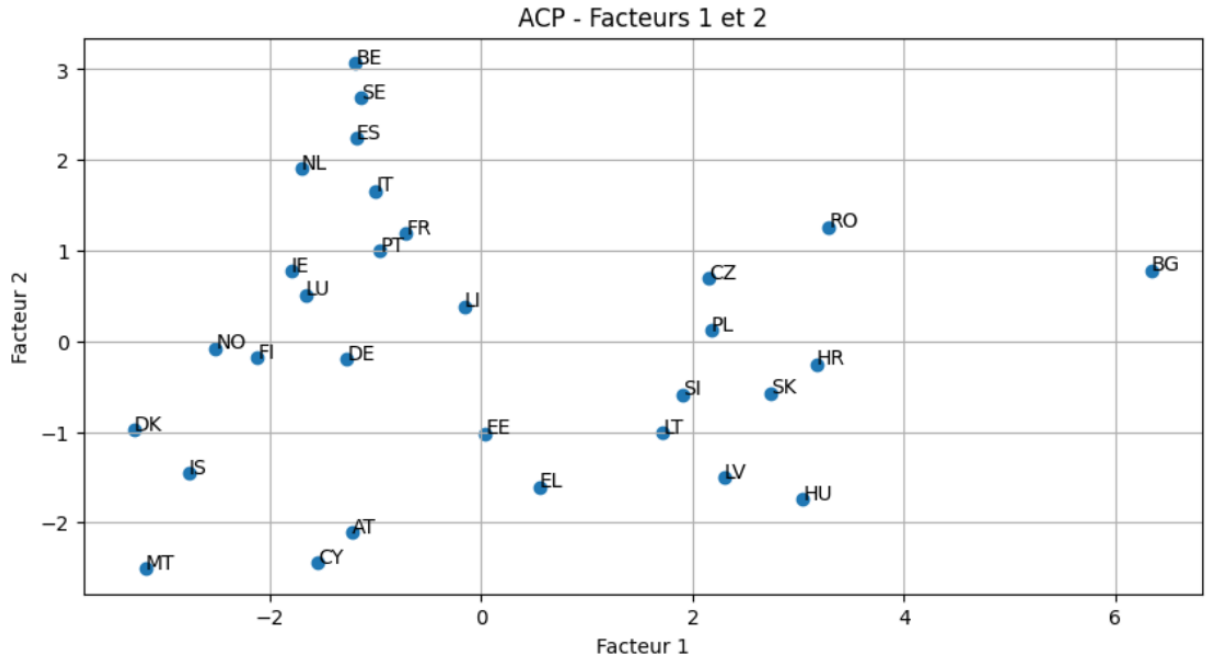
A-t-on besoin ici d'employer ce Filtre sur les données étudiées ?

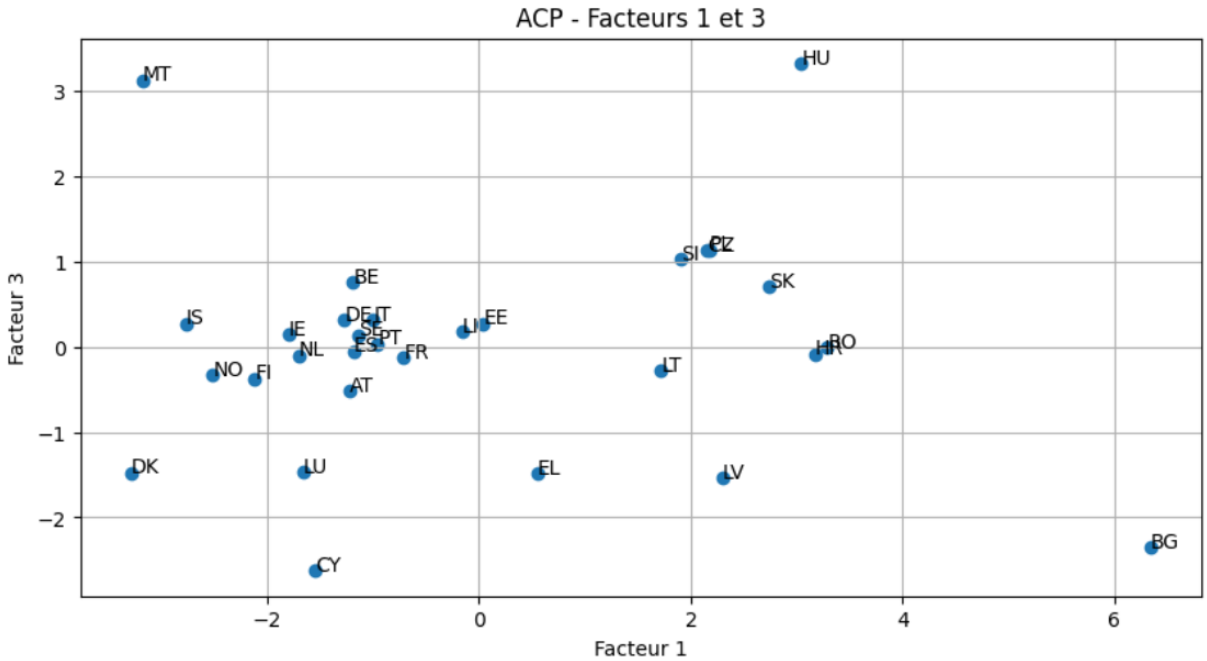
Les échelles des attributs varient considérablement, cela peut affecter la performance des méthodes de clustering, en particulier celles qui dépendent de distances.

Donc la Standarisation est néccessaire dans ce cas.

4- Effectuez une analyse en composantes principales (ACP) sur les données à l'aide de la bibliothèque scikit-learn

Joignez au rapport l'affichage des instances étiquetées par le code du pays suivant les facteurs 1 et 2, puis suivant les facteurs 1 et 3 de l'ACP.

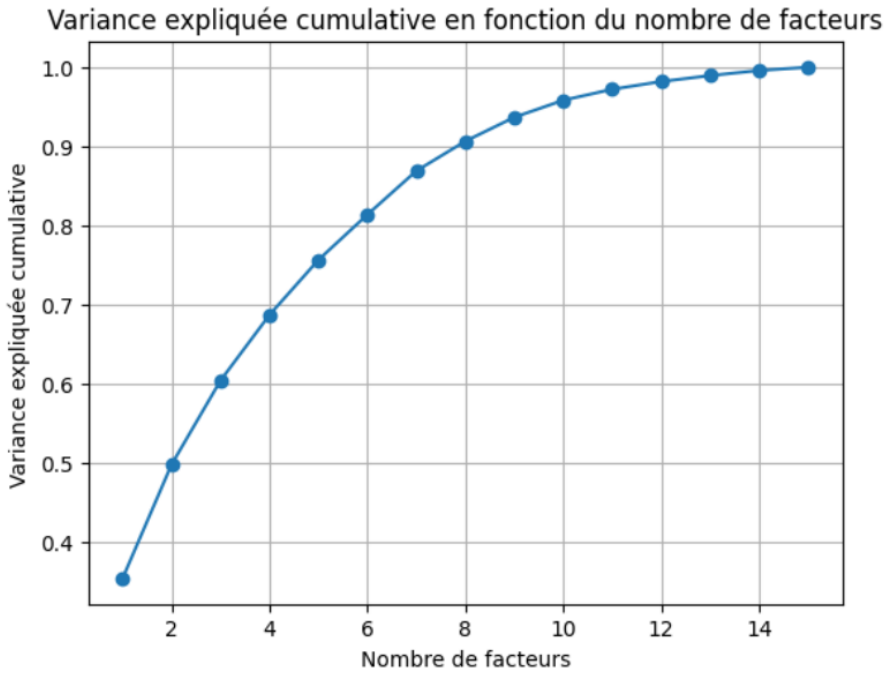




5- Tracer le graphique de la variance expliquée en fonction du rang du facteur (composante principale) :

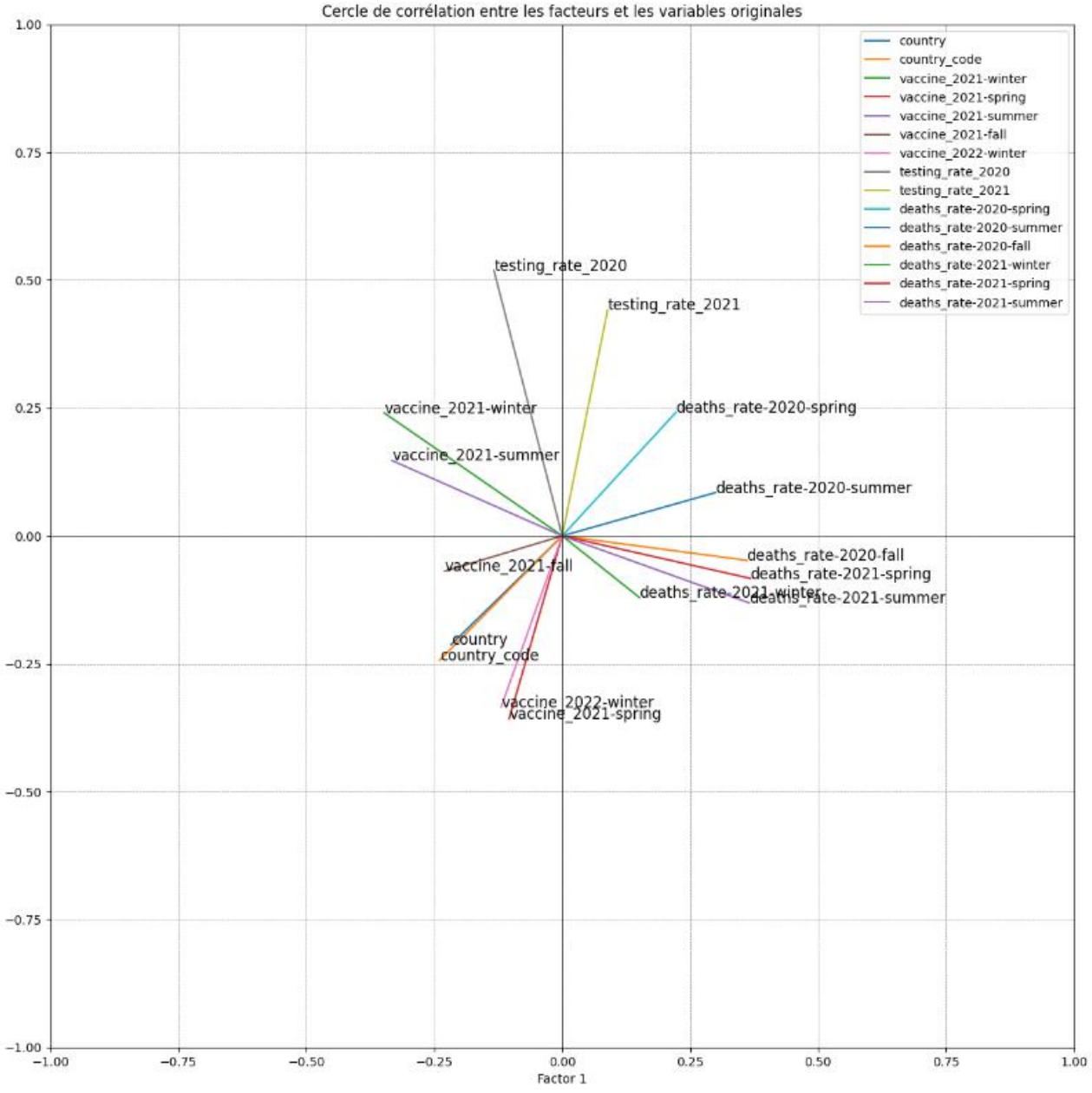
La variance expliquée est représentée en calculant les valeurs propres.

Ici j'ai représenté la variance expliquée cumulative :



Tracer le cercle de corrélation entre les facteurs et les variables originales. Pour cela, on peut utiliser les charges factorielles.

Le cercle de corrélation permet aux analystes de mieux comprendre quelles variables originales sont fortement corrélées aux facteurs de l'ACP, ce qui peut aider à interpréter la signification des facteurs eux-mêmes. Cela peut également aider à identifier des groupes de variables qui sont liées les unes aux autres et qui ont un impact similaire sur les facteurs. En fin de compte, cela facilite l'interprétation des résultats de l'ACP et la compréhension des relations sous-jacentes dans les données



Commentez les deux graphiques obtenus à la question précédente (a-chage dans les deux premiers plans de l'ACP) en discutant la proximité entre les pays suivant les 3 facteurs

Pour commenter les deux graphiques obtenus dans les deux premiers plans de l'ACP et discuter la proximité entre les pays en fonction des trois premiers facteurs, nous allons analyser les informations fournies par ces graphiques.

Graphique de l'ACP - Facteurs 1 et 2 :

- Si deux pays sont proches dans ce graphique, cela signifie qu'ils ont des caractéristiques similaires en termes du premier facteur (PC1) et du deuxième facteur (PC2).
- Les pays situés dans le même quadrant partagent des similitudes en termes du premier facteur (PC1), tandis que leur position verticale reflète leurs différences par rapport au deuxième facteur (PC2) c'est le cas du **PL** et **CZ**.

Graphique de l'ACP - Facteurs 1 et 3 :

- Si deux pays sont proches dans ce graphique, cela signifie qu'ils ont des caractéristiques similaires en termes du premier facteur (PC1) et du troisième facteur (PC3).
- Les pays situés dans le même quadrant partagent des similitudes en termes du premier facteur (PC1), tandis que leur position verticale reflète leurs différences par rapport au troisième facteur (PC3) c'est le cas du **BE** et **DE**.

❖ Par contre des Pays comme BG, HU et MT sont très loin des autres Pays suivant PC1, PC2 et PC3

meme en revenant sur les données on observe que BG par exemple a un profil diffèrent des autres pays ;elle marquée un très grand nombre de décès même s'elle a peu d'habitas par rapport aux autres Pays (entre 2020-fall et 2022-winter) et ça revient probablement au nombre démunie de Vaccins et de Test de Covid. par rapport auw autre Pays. (79.914716 vaccins_per_1000_Per dans AT Vs 30.561109 30.561109 à 2021-winter) et (**489.46** testing_rate_2020 VS **902.69** testing_rate_2020)

6- Clustering avec la méthode des k-moyennes

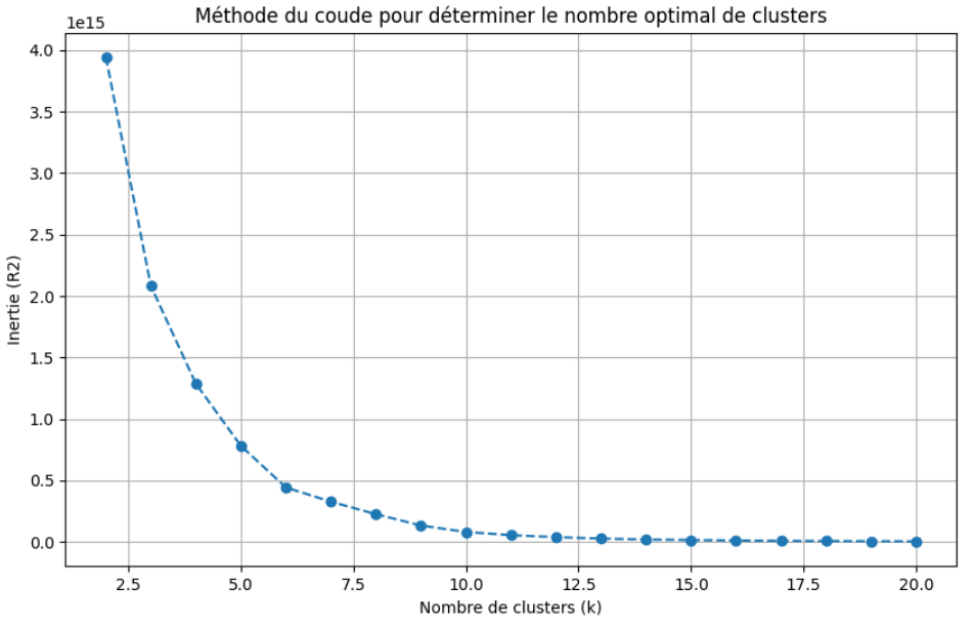
Je vais procéder à une analyse K-Means en utilisant un nombre de clusters (k) de 5 pour cette démonstration

```
Out[90]: 0     0
         1     0
         2     3
         3     3
         4     3
         5     0
         6     3
         7     3
         8     3
         9     4
         10    1
         11    0
         12    0
         13    3
         14    3
         15    4
         16    3
         17    3
         18    3
         19    3
         20    3
         21    0
         22    3
         23    2
         24    0
         25    0
         26    3
         27    3
         28    2
         29    0
         Name: cluster, dtype: int32
```

Tracez la variation du R2 pour k variant de 2 à 20.

L'attribut inertia_ de l'objet KMeans dans scikit-learn représente l'inertie du modèle de clustering. L'inertie mesure la somme des carrés des distances entre les points de données et leur centre de gravité (centroid) dans chaque cluster. Plus précisément, l'inertie est la somme des carrés des distances euclidiennes entre chaque point de données et le centroïde de son cluster assigné.

L'inertie est une mesure de la compacité des clusters. En général, les clusters sont considérés comme de meilleure qualité lorsque l'inertie est plus faible, car cela signifie que les points de données sont plus proches de leur centre de gravité respectif.



D'après cette courbe, quelle valeur de k retiendriez-vous ?

À partir de K=8 le **inertia** commence à prendre des valeurs minimales et stable, donc on peut prendre 8 clusters.

En fixant k à 8, commentez les profils des groupes contenant la France, le Danemark et la Bulgarie.

Donc après le clustring voici les Pays dans les meme clusterd que la France le Danemark et la Bulgarie

```
france cluster:  3

Denmark cluster:  1

Bulgaria cluster:  0

Pays du même cluster que la France:
['Belgium', 'Estonia', 'France', 'Ireland', 'Italy', 'Latvia', 'Liechtenstein', 'Lithuania', 'Netherlands', 'Norway', 'Portu
gal', 'Slovakia']
Pays du même cluster que le Danemark:
['Denmark']
Pays du même cluster que la Bulgarie:
['Bulgaria', 'Croatia', 'Finland', 'Germany', 'Hungary', 'Poland', 'Romania', 'Spain', 'Sweden']
```

- Donc Danemark est seule ne correspond à aucun pays
- La France partage le même profil que plusieurs Pays a savoir Ireland, Italy,Norway, Portu et d'autre
- Bulgarie aussi partage le même profil que Finland, Germany, Poland et d'autres

Et voici une description de ces trois clusters en calculant le mean :

```
Cluster 0:
vaccine_2021-winter          76.106085
vaccine_2021-spring         489.857452
vaccine_2021-summer         407.203644
vaccine_2021-fall           228.103141
vaccine_2022-winter         241.454082
testing_rate_2020           701.288889
testing_rate_2021          1588.733333
deaths_rate-2020-spring      23.837333
deaths_rate-2020-summer       7.064889
deaths_rate-2020-fall        42.496000
deaths_rate-2021-winter     105.112222
deaths_rate-2021-spring      80.687667
deaths_rate-2021-summer       8.268222
deaths_rate-2021-fall        72.975000
deaths_rate-2022-winter      81.435333
cluster                       0.000000
dtype: float64
```

```
Cluster 1:
vaccine_2021-winter          98.573134
vaccine_2021-spring         533.186908
vaccine_2021-summer         757.946872
vaccine_2021-fall           215.124160
vaccine_2022-winter         466.937947
testing_rate_2020          3215.000000
testing_rate_2021         27700.480000
deaths_rate-2020-spring       4.677000
deaths_rate-2020-summer       1.453000
deaths_rate-2020-fall         4.848000
deaths_rate-2021-winter      40.768000
deaths_rate-2021-spring       4.109000
deaths_rate-2021-summer       1.797000
deaths_rate-2021-fall         8.561000
deaths_rate-2022-winter      28.509000
cluster                       1.000000
dtype: float64
```

```
Cluster 3:
vaccine_2021-winter          77.732608
vaccine_2021-spring         514.897046
vaccine_2021-summer         560.899837
vaccine_2021-fall           244.247792
vaccine_2022-winter         343.130322
testing_rate_2020           973.212500
testing_rate_2021          3325.208333
deaths_rate-2020-spring      33.642750
deaths_rate-2020-summer       2.878583
deaths_rate-2020-fall        33.725333
deaths_rate-2021-winter     104.105083
deaths_rate-2021-spring      44.806750
deaths_rate-2021-summer       7.343167
deaths_rate-2021-fall        43.203667
deaths_rate-2022-winter      51.703000
cluster                       3.000000
dtype: float64
```

Donc d'après cette description il apparut que Danemark (Cluster 1) est toute seule différente des autres cluster parceque ella met en place plusieurs Test (moy = 27700 en 2021 vs 1550 et 3325 pour les pays du cluster 0 et 3)

Ce qui ramenir a un nombre de morts plus bas (40 vs 105 et 104 pour les pays du clusters 0 et 3 en Spring 2021)

1.2 Recherche de règles d'association

Cette partie aborde la recherche de règles d'association, en illustrant son principe par l'algorithme

Apriori, disponible dans la bibliothèque mlxtend.

Dans cet exercice, il est demandé d'étudier le fichier supermarket.csv, qui fournit des informations sur les achats dans un supermarché de Nouvelle-Zélande. L'objectif est ici de découvrir des règles s'appliquant sur le panier de consommation.

Ouvrez les données à l'aide de la bibliothèque pandas :

	department1	department2	department3	department4	department5	department6	department7	department8	department9	grocery misc	...
4622	?	?	?	?	?	?	?	?	?	?	...
4623	?	?	?	t	?	?	?	?	?	?	...
4624	?	?	?	?	?	?	?	?	?	?	...
4625	?	?	?	?	?	?	?	?	?	?	...
4626	t	?	?	?	?	?	?	?	?	?	...

5 rows × 217 columns

```
print(supermarket_data.columns)
```

```
Index(['department1', 'department2', 'department3', 'department4',
       'department5', 'department6', 'department7', 'department8',
       'department9', 'grocery misc',
       ...
       'department208', 'department209', 'department210', 'department211',
       'department212', 'department213', 'department214', 'department215',
       'department216', 'total'],
      dtype='object', length=217)
```

Quels sont les types des attributs ?

```
# Afficher les types des attributs
supermarket_data.dtypes
```

```
department1      object
department2      object
department3      object
department4      object
department5      object
                  ...
department213    object
department214    object
department215    object
department216    object
total            object
Length: 217, dtype: object
```

```
print(supermarket_data.dtypes.unique())
```

```
[dtype('O')]
```

Donc tous les attrributs sont objects !

Que représentent les valeurs associées aux ` ?'

Les valeurs associées aux ?' dans le DataFrame vaut dire que le client n'a pas pris ce produit.

Comment agit les commandes du fichier squelette, permettant la discrétisation ?

nous n'avons pas d'attributs numériques, donc la discrétisation n'est pas nécessaire. La discrétisation est généralement utilisée pour transformer des données continues en données discrètes, mais dans notre cas, nos attributs semblent déjà être nominaux avec des valeurs comme 't', '?', 'low', et 'high'.

par contre on peut procéder par un one hot encoding :

Établissez l'ensemble des itemsets fréquents à l'aide de la fonction apriori() de mlxtend en fixant un support minimal à 0,1

	support	itemsets
0	0.226281	(department1)
1	0.133780	(baby needs)
2	0.719689	(bread and cake)
3	0.604063	(baking needs)
4	0.532310	(juice-sat-cord-ms)
...	...	...
10277	0.101145	(total_high, fruit, frozen foods, vegetables, ...
10278	0.100713	(fruit, party snack foods, frozen foods, veget...
10279	0.103307	(total_high, fruit, party snack foods, frozen ...
10280	0.102658	(total_high, tissues-paper prd, fruit, frozen ...
10281	0.105036	(total_high, fruit, frozen foods, vegetables, ...

[10282 rows x 2 columns]

Expliquez ce que signifie cette valeur de 0,1

La valeur de 0,1 utilisée comme seuil de support (min_support = 0,1) dans l'algorithme Apriori signifie que seules les règles d'association qui ont un support égal ou supérieur à 10 % des transactions seront considérées comme fréquentes. En d'autres termes, un ensemble d'articles (ou items) doit apparaître dans au moins 10 % des transactions pour être considéré comme un itemset fréquent.

Le support d'un itemset est la proportion d'occurrences de cet itemset par rapport à l'ensemble total des transactions. C'est une mesure de la fréquence de l'itemset dans les données. Plus le support est élevé, plus l'itemset est courant dans les transactions.

Combien d'itemsets obtenez-vous ?

`[10282 rows x 2 columns]`: donc 10282 itemsets

Donnez leur fréquence en fonction du nombre d'items qu'ils contiennent.

	Taille de l'ensemble	Nombre d'itemsets fréquents
5	1	52
3	2	634
1	3	2598
0	4	3950
2	5	2470
4	6	558
6	7	20

Générez les règles à partir des itemsets fréquents avec une confiance minimale de 0,7

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(department1)	(bread and cake)	0.226281	0.719689	0.171601	0.758357	1.053729	0.008750	1.160023	0.065902
1	(baby needs)	(bread and cake)	0.133780	0.719689	0.100929	0.754443	1.048290	0.004649	1.141531	0.053180
2	(baking needs)	(bread and cake)	0.604063	0.719689	0.473525	0.783900	1.089221	0.038788	1.297136	0.206882
3	(juice-sat-cord-ms)	(bread and cake)	0.532310	0.719689	0.403933	0.758831	1.054387	0.020836	1.162301	0.110291
4	(tea)	(bread and cake)	0.193646	0.719689	0.153231	0.791295	1.099496	0.013866	1.343095	0.112224
...	...	...	...	...	...	...	...	...	...	...
24565	(fruit, bread and cake, biscuits, milk-cream, ...	(vegetables, frozen foods)	0.143289	0.406743	0.105036	0.733032	1.802198	0.046754	2.222200	0.519571
24566	(frozen foods, vegetables, biscuits, milk-crea...	(bread and cake, fruit)	0.129674	0.502485	0.105036	0.810000	1.611987	0.039877	2.618498	0.436213
24567	(bread and cake, vegetables, frozen foods, mil...	(biscuits, fruit)	0.145883	0.397018	0.105036	0.720000	1.813522	0.047118	2.153509	0.525205
24568	(bread and cake, frozen foods, biscuits, milk-...	(vegetables, fruit)	0.147828	0.476983	0.105036	0.710526	1.489626	0.034524	1.806786	0.385709
24569	(bread and cake, vegetables, biscuits, milk-cr...	(frozen foods, fruit)	0.141128	0.402204	0.105036	0.744257	1.850445	0.048273	2.337488	0.535108

24570 rows × 10 columns

Combien de règles obtenez-vous ?

D'après le tableau ci-dessus on a 24570 règles à partir de l'ensemble des itemsets fréquents avec un seuil de confiance minimal de

Commentez la première règle obtenue ?

La première règle obtenue est :

- Antécédents (antecedents) : (department1)
- Conséquents (consequents) : (bread and cake)
- Support de l'antécédent : 0.226281
- Support du conséquent : 0.719689
- Support de la règle : 0.171601
- Confiance : 0.758357
- Lift : 1.053729
- Gain d'information (leverage) : 0.008750
- Conviction : 1.160023
- Mesure de Zhang : 0.065902

- La première règle indique que si un client achète du département 1, il a une confiance de 75,84% à acheter du pain et des gâteaux (bread and cake).

- Le lift de 1.05 indique qu'il y a une légère dépendance positive entre ces deux achats, ce qui signifie que l'achat du department1 augmente légèrement la probabilité d'achat de bread and cake par rapport à ce que l'on pourrait attendre au hasard.

- Le gain d'information positif (leverage) de 0.008750 suggère que l'achat du department1 et bread and cake est plus fréquent que ce à quoi on pourrait s'attendre au hasard.

- La conviction de 1.160023 indique que les clients qui achètent department1 sont 1,16 fois plus susceptibles d'acheter bread and cake que de ne pas l'acheter.

Globalement, cette règle suggère une certaine association entre l'achat du département 1 et l'achat de pain et de gâteaux, avec une confiance modérément élevée. Cependant, la valeur du lift est proche de 1, ce qui signifie que l'association n'est pas très forte.

La meilleure règle pour l'ensemble ' supermarket ' pour chacune des métriques confidence et lift ? Commentez les scores et règles obtenus.

la meilleure règle pour l'ensemble ' supermarket ' pour la confidence: ¶

```
best_rule_by_confidence = rules.sort_values(by='confidence', ascending=False).iloc[0]
print("\nMeilleure règle en termes de confiance :\n")
best_rule_by_confidence.antecedents
```

```
Meilleure règle en termes de confiance :

frozenset({'biscuits',
           'frozen foods',
           'fruit',
           'party snack foods',
           'total_high',
           'vegetables'})
```

```
best_rule_by_confidence
```

```
antecedents          (fruit, party snack foods, frozen foods, veget...
consequents                                          (bread and cake)
antecedent support                                          0.110223
consequent support                                          0.719689
support                                                     0.103307
confidence                                                  0.937255
lift                                                        1.302306
leverage                                                    0.023981
conviction                                                   4.46746
zhangs_metric                                               0.260887
Name: 24519, dtype: object
```

Cette règle a une confiance élevée, ce qui signifie que lorsqu'on observe l'antécédent, la probabilité d'observer le conséquent est de 93,73%. Le lift de 1,302306 indique qu'il y a une relation positive entre l'antécédent et le conséquent. Le leverage positif (0,023981) indique que l'observation de l'antécédent a un effet positif sur l'observation du conséquent. La conviction élevée (4,46746) indique que l'antécédent a un fort effet sur le conséquent, et la métrique de Zhang est également positive (0,260887).

la meilleure règle pour l'ensemble ' supermarket ' pour le lift :

```
# Pour trouver la meilleure règle en termes de lift
best_rule_by_lift = rules.sort_values(by='lift', ascending=False).iloc[0]
print("\nMeilleure règle en termes de lift :")
best_rule_by_lift.antecedents
```

```
Meilleure règle en termes de lift :

frozenset({'biscuits',
           'frozen foods',
           'fruit',
           'tissues-paper prd',
           'vegetables'})
```

```
best_rule_by_lift
```

```
antecedents          (tissues-paper prd, fruit, frozen foods, veget...
consequents                                (total_high, bread and cake)
antecedent support                                          0.143289
consequent support                                          0.305381
support                                                     0.102658
confidence                                                   0.71644
lift                                                        2.346051
leverage                                                      0.0589
conviction                                                  2.449639
zhangs_metric                                               0.669715
Name: 24552, dtype: object
```

Cette règle a un lift élevé (2,346051), ce qui signifie qu'il y a une forte association entre l'antécédent et les conséquents. La confiance est de 71,64%, indiquant une probabilité élevée d'observer les conséquents lorsque l'antécédent est vrai. Le leverage positif (0,0589) indique un effet positif de l'antécédent sur les conséquents. La conviction (2,449639) est également élevée, ce qui suggère une forte influence de l'antécédent sur les conséquents, et la métrique de Zhang est positive (0,669715).

En résumé, la première règle a une confiance élevée mais un lift plus faible, tandis que la seconde règle a un lift très élevé, ce qui suggère une forte association entre les éléments de l'antécédent et du conséquent. Cependant, il est essentiel de noter que le choix de la "meilleure" règle dépend des objectifs spécifiques de l'analyse. La confiance met l'accent sur la fiabilité de la règle, tandis que le lift met l'accent sur la force de l'association.

Un dirigeant du supermarché souhaite améliorer ses ventes en réorganisant le positionnement des produits dans ses rayons. D'après votre analyse du panier de consommation fourni, quels produits sont souvent achetés conjointement avec des biscuits et des aliments pour animaux domestiques ?

Pour identifier quels produits sont souvent achetés conjointement avec des biscuits et des aliments pour animaux domestiques, nous devons revoir la règle d'association en ajustant le seuil de support minimal. Augmenter le seuil de support minimal signifie que nous ne tiendrons compte que des règles avec un support plus élevé, ce qui rendra les associations plus fortes et plus significatives. Le seuil précédemment fixé à 0,1 était assez bas, donc nous ré exécuterons l'analyse d'Apriori avec un seuil de support minimal plus élevé (0,2) pour identifier des règles plus spécifiques et significatives.

Voilà le résultat :

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
611	(biscuits, pet foods)	(bread and cake)	0.256105	0.719689	0.215907	0.843038	1.171392	0.03159	1.785854	0.196688

D'après l'analyse des règles d'association, on peut observer que les produits "biscuits" et "aliments pour animaux domestiques" sont souvent achetés conjointement avec "pain et gâteaux" (bread and cake) Voici les détails de la règle correspondante :

- Antécédents (produits souvent achetés ensemble) : "biscuits" et "aliments pour animaux domestiques"
- Conséquents (produit associé) : "pain et gâteaux"
- Support de l'antécédent (pourcentage d'occurrences de l'antécédent) : 21.59%
- Support du conséquent (pourcentage d'occurrences du conséquent) : 71.97%
- Support de la règle (pourcentage d'occurrences des antécédents et des conséquents ensemble) : 21.59%
- Confiance (probabilité conditionnelle que les conséquents se produisent lorsque les antécédents se produisent) : 84.30%
- Lift (mesure de l'importance de la règle) : 1.17
- Leverage (mesure de l'importance de la règle, prenant en compte les occurrences attendues) : 3.16%
- Conviction (mesure de la force de la règle) : 1.79
- Zhang's metric (une autre mesure de la force de la règle) : 19.67%

Cela signifie que les produits "biscuits" et "aliments pour animaux domestiques" et "pain et gâteaux" ont une forte probabilité d'être achetés ensemble, et cette association est significative. Par conséquent, réorganiser le positionnement de ces produits dans les rayons peut être une stratégie efficace pour augmenter les ventes

2 Analyse prédictive

2.1 Classement

Ouvrez les données d'apprentissage et de test à l'aide de la bibliothèque pandas

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Combien y a-t-il d'attributs dans ces fichiers ?

```
columns = train_data.columns
columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'class'],
      dtype='object')
```

```
len(columns)
```

```
15
```

Quels sont leurs types ?

```
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
class             object
dtype: object
```

Quelle est la variable à prédire ?

La variable à prédire (variable cible) dans le jeu de données "Adult earnings" est généralement la variable "class". Elle indique la classe de revenu d'une personne et détermine si cette personne gagne plus de 50 000 dollars par an (">50K") ou 50 000 dollars ou moins par an ("<=50K"). Elle est souvent utilisée pour des tâches de classification binaire, où l'objectif est de prédire si le revenu d'une personne dépasse ou non le seuil de 50 000 dollars.

Les classes sont-elles équilibrées dans les fichiers ?

```
# Comptez les occurrences de chaque classe
class_counts = train_data['class'].value_counts()

# Affichez le nombre d'occurrences de chaque classe
print(class_counts)
```

```
<=50K    24720
>50K      7841
Name: class, dtype: int64
```

D'après les résultats, les classes ne sont pas équilibrées dans le jeu de données d'apprentissage. Il y a 24 720 occurrences de la classe "<=50K" (revenu inférieur ou égal à 50 000 dollars) et seulement 7 841 occurrences de la classe ">50K" (revenu supérieur à 50 000 dollars).

L'ensemble de données est déséquilibré, avec une majorité d'individus ayant un revenu inférieur ou égal à 50 000 dollars. Cela peut avoir un impact sur la performance des modèles de classification, en particulier sur leur capacité à bien prédire la classe minoritaire (">50K"). Il peut être nécessaire de prendre en compte cet aspect lors de l'entraînement des modèles et de l'évaluation de leur performance, en utilisant des métriques appropriées pour les ensembles de données déséquilibrés.

Y a-t-il beaucoup de valeurs manquantes dans les données ?

```
train_data.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
class             0
dtype: int64
```

```
Nombre de valeurs manquantes dans train datset :
age                0
workclass       1836
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation      1843
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   583
class              0
dtype: int64
```

```
Nombre de valeurs manquantes dans test dataset :
age                0
workclass        963
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation       966
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   274
class              0
dtype: int64
```

Donc on a des valeurs manquantes sur les colonnes ‘Workclass’,’Occupation’ et ‘native-country’ dans trainset et testset.

Comment ces valeurs sont-elles représentées dans le fichier ?

Les valeurs manquantes sont représentées saus la forme de ‘?’.

Expliquez en une phrase en quoi consiste la méthode élémentaire de classement DummyClassifier utilisant la stratégie most_frequent.

Le DummyClassifier est un classifieur de base qui sert principalement comme outil de comparaison pour évaluer la performance des autres classifieurs. Il fonctionne en suivant une stratégie simple pour attribuer des étiquettes de classe aux exemples d'un ensemble de données. Il existe plusieurs stratégies possibles, dont l'une des plus courantes est la stratégie most_frequent. Voici comment fonctionne DummyClassifier avec cette stratégie :

Apprentissage initial : Lors de la phase d'apprentissage, le DummyClassifier analyse l'ensemble d'entraînement pour déterminer la classe majoritaire, c'est-à-dire la classe la plus fréquente parmi les exemples.

Prédiction : Lorsqu'il est temps de faire des prédictions, le classifieur attribue la classe majoritaire à toutes les instances de l'ensemble de test. Peu importe les caractéristiques ou les attributs de chaque exemple, ils se voient tous attribuer la même classe majoritaire.

En résumé, le DummyClassifier ne tient pas compte des caractéristiques ou des relations entre les attributs pour prendre ses décisions. Il se contente de prédire la classe majoritaire dans l'ensemble de données d'apprentissage.

Dans quel cas ce classifieur pourrait donner les meilleurs résultats parmi ceux testés ?

Ce classifieur pourrait donner les meilleurs résultats dans un scénario où les classes sont déséquilibrées, et la classe majoritaire est prédominante, car il prédit simplement cette classe pour chaque instance.

Dans un premier temps, seules les données numériques sont considérées pour prédire la classe. Remplacez chaque valeur manquante par la moyenne obtenue par chaque attribut en utilisant SimpleImputer. Normalisez ensuite les valeurs numériques à l'aide de StandardScaler

Tous d'abord j'ai filtré que les colonnes numériques après j'ai converti les valeurs manquantes représentées par ` ?' par NAN pour quelles soit détectable par SimpleImputer, et après j'ai remplacé ces valeurs manquantes par la moyennes de chaque colonne, apès j'ai appliqué la normalisation avec StandardScaler, et voilà donc les données numériques résultants :

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

```
array([[ 0.03067056, -1.06361075,  1.13473876,  0.1484529 , -0.21665953,
        -0.03542945],
       [ 0.83710898, -1.008707  ,  1.13473876, -0.14592048, -0.21665953,
        -2.22215312],
       [-0.04264203,  0.2450785 , -0.42005962, -0.14592048, -0.21665953,
        -0.03542945],
       ...,
       [ 1.42360965, -0.35877741, -0.42005962, -0.14592048, -0.21665953,
        -0.03542945],
       [-1.21564337,  0.11095988, -0.42005962, -0.14592048, -0.21665953,
        -1.65522476],
       [ 0.98373415,  0.92989258, -0.42005962,  1.88842434, -0.21665953,
        -0.03542945]])
```

Et j'ai fait la même chose avec les données de test.

Calculez les performances des cinq méthodes de classement étudiées sur les données 'adult earnings' en réalisant une validation croisée en 5 blocs sur les données d'apprentissage ?

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.80 (+/- 0.01)
Accuracy of Decision tree classifier on cross-validation: 0.77 (+/- 0.01)
Accuracy of Random Forest classifier on cross-validation: 0.81 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.82 (+/- 0.01)
```

```
Accuracy of Dummy classifier on cross-validation: 0.76
[[24720     0]
 [ 7841     0]]
Accuracy of Naive Bayes classifier on cross-validation: 0.80
[[23493  1227]
 [ 5411  2430]]
Accuracy of Decision tree classifier on cross-validation: 0.77
[[20867  3853]
 [ 3624  4217]]
Accuracy of Random Forest classifier on cross-validation: 0.81
[[22273  2447]
 [ 3841  4000]]
Accuracy of Logistic regression classifier on cross-validation: 0.82
[[23452  1268]
 [ 4750  3091]]
Accuracy of SVM classifier on cross-validation: 0.82
[[23895   825]
 [ 4912  2929]]
```

- Dummy Classifier : Ce classifieur est basé sur la stratégie "most_frequent" et prédit la classe majoritaire (<=50K) pour toutes les instances. Il atteint une précision de 0.76. Cependant, la matrice de confusion montre qu'il prédit toutes les instances comme étant "<=50K", ce qui n'est pas très utile pour la classification réelle, surtout pour prédire les ">50K".

- Naive Bayes Classifier : Le classifieur Naive Bayes atteint une précision de 0.80. Sa matrice de confusion montre qu'il effectue des prédictions plus équilibrées entre les deux classes, mais il y a toujours des erreurs significatives.

- Decision Tree Classifier : Le classifieur basé sur l'arbre de décision atteint une précision de 0.77. Sa matrice de confusion montre qu'il a du mal à distinguer les deux classes, en particulier les ">50K".

- Random Forest Classifier : Le classifieur de forêt d'arbres décisionnels obtient une précision de 0.81, ce qui est légèrement meilleur que le Decision Tree. Cependant, il a encore des difficultés à bien classifier les ">50K".

- Logistic Regression Classifier : Le classifieur de régression logistique atteint une précision de 0.82, ce qui est relativement bon. Il prédit de manière plus équilibrée les deux classes.

- SVM Classifier : Le classifieur SVM atteint également une précision de 0.82, ce qui est similaire à la régression logistique. Sa matrice de confusion montre une meilleure distinction entre les classes par rapport aux classifieurs précédents.

- En résumé, la régression logistique et le SVM semblent être les meilleurs classifieurs parmi ceux testés, avec des précisions de 0.82. Cependant, il est important de noter que ces performances pourraient être améliorées en ajustant les hyperparamètres des classifieurs ou en utilisant des techniques de prétraitement des données plus avancées.

Parmi les méthodes testées, quelle est la plus performante par rapport au taux de classification ?

Parmi les méthodes testées, la régression logistique et le SVM sont les plus performants en termes de taux de classification, avec une précision de 0.82. Cela signifie que ces deux classifieurs parviennent à prédire correctement la classe de 82 % des instances dans les données "Adult earnings". Ces deux méthodes sont donc les plus performantes en matière de classification parmi celles testées.

Dorénavant, on se propose de n'employer que les attributs catégoriels. Remplacez chaque valeur manquante par le mode (valeur la plus fréquente) de chaque attribut en recourant à SimpleImputer. Réalisez une disjonction des variables catégorielles à l'aide de la classe OneHotEncoder de la bibliothèque scikit-learn.

Voilà les résultats après remplacer les valeurs nul par le mode et la disjonction des variables catégorielles:

	workclass	education	marital-status	occupation	relationship	race	sex	native-country
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba

```
encoded_train_categorical_data
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

Calculez les performances de classement des cinq méthodes étudiants, en réalisant à nouveau une validation croisée en 5 bocs

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.49 (+/- 0.06)
Accuracy of Decision tree classifier on cross-validation: 0.82 (+/- 0.00)
Accuracy of Random Forest classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.83 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.83 (+/- 0.01)

Accuracy of Dummy classifier on cross-validation: 0.76
[[24720     0]
 [ 7841     0]]
Accuracy of Naive Bayes classifier on cross-validation: 0.49
[[ 8396 16324]
 [  375  7466]]
Accuracy of Decision tree classifier on cross-validation: 0.82
[[22363  2357]
 [ 3603  4238]]
Accuracy of Random Forest classifier on cross-validation: 0.82
[[22526  2194]
 [ 3522  4319]]
Accuracy of Logistic regression classifier on cross-validation: 0.83
[[22920  1800]
 [ 3731  4110]]
Accuracy of SVM classifier on cross-validation: 0.83
[[22937  1783]
 [ 3781  4060]]
```

Observez-vous des performances similaires à ce que vous aviez à la question précédente ?

Il est intéressant de noter que les performances des classifieurs avec uniquement les attributs catégoriels semblent meilleurs que celles obtenues précédemment avec l'ensemble des attributs. En particulier, les classifieurs "**Logistic Regression**" et "**SVM**" semblent avoir des performances significativement plus élevées, atteignant une précision d'environ 83%, ce qui est plus élevé que la meilleure précision précédemment obtenue.

Cela peut s'expliquer par le fait que, pour certaines méthodes, l'utilisation de variables catégorielles peut être plus appropriée ou qu'il peut y avoir des attributs numériques qui contiennent du bruit et affectent négativement les performances. Cela dépend de la nature des données et de l'algorithme de classification utilisé.

Utilisez enfin l'ensemble des colonnes, c-à-d en prenant en compte les attributs numériques et catégoriels. Calculez à nouveau les performances de classement en validation croisée à 5 blocs ?

Dans un premier lieu j'ai utilisé tous les données (numériques et catégorial), j'ai discrétisé les attributs catégoriaux et j'ai enlevé les valeurs manquantes après j'ai normalisé le tous.

Voilà les données finales :

```
array([[ 0.03067056, -1.06361075,  1.13473876, ...,  1.        ,
         0.        ,  0.        ],
       [ 0.83710898, -1.008707  ,  1.13473876, ...,  1.        ,
         0.        ,  0.        ],
       [-0.04264203,  0.2450785 , -0.42005962, ...,  1.        ,
         0.        ,  0.        ],
       ...,
       [ 1.42360965, -0.35877741, -0.42005962, ...,  1.        ,
         0.        ,  0.        ],
       [-1.21564337,  0.11095988, -0.42005962, ...,  1.        ,
         0.        ,  0.        ],
       [ 0.98373415,  0.92989258, -0.42005962, ...,  1.        ,
         0.        ,  0.        ]])
```

Et voilà les résultats finaux des 5 modèles :

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.52 (+/- 0.03)
Accuracy of Decision tree classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of Random Forest classifier on cross-validation: 0.85 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.85 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.86 (+/- 0.01)

Accuracy of Dummy classifier on cross-validation: 0.76
[[24720     0]
 [ 7841     0]]
Accuracy of Naive Bayes classifier on cross-validation: 0.52
[[ 9556 15164]
 [  350  7491]]
Accuracy of Decision tree classifier on cross-validation: 0.81
[[21610  3110]
 [ 2933  4908]]
Accuracy of Random Forest classifier on cross-validation: 0.85
[[22927  1793]
 [ 2980  4861]]
Accuracy of Logistic regression classifier on cross-validation: 0.85
[[23042  1678]
 [ 3161  4680]]
Accuracy of SVM classifier on cross-validation: 0.86
[[23294  1426]
 [ 3275  4566]]
```

Performance des classifieurs en utilisant les données numériques :

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.80 (+/- 0.01)
Accuracy of Decision tree classifier on cross-validation: 0.77 (+/- 0.01)
Accuracy of Random Forest classifier on cross-validation: 0.81 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.82 (+/- 0.01)
```

Performance des classifieurs en utilisant les données catégoriales :

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.49 (+/- 0.06)
Accuracy of Decision tree classifier on cross-validation: 0.82 (+/- 0.00)
Accuracy of Random Forest classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.83 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.83 (+/- 0.01)
```

Performance des classifieurs en utilisant tous les données

```
Accuracy of Dummy classifier on cross-validation: 0.76 (+/- 0.00)
Accuracy of Naive Bayes classifier on cross-validation: 0.52 (+/- 0.03)
Accuracy of Decision tree classifier on cross-validation: 0.82 (+/- 0.01)
Accuracy of Random Forest classifier on cross-validation: 0.85 (+/- 0.01)
Accuracy of Logistic regression classifier on cross-validation: 0.85 (+/- 0.01)
Accuracy of SVM classifier on cross-validation: 0.86 (+/- 0.01)
```

Il est important de noter que les performances du classifieur Naive Bayes semblent s'être détériorées. Cela peut être dû à l'utilisation de l'encodage One-Hot des caractéristiques catégorielles, qui peut augmenter la dimensionnalité de l'ensemble de données et rendre la méthode Naive Bayes moins efficace. D'un autre côté, des classifieurs plus complexes tels que le Decision Tree, Random Forest, Logistic Regression et SVM ont bien performé avec l'inclusion de toutes les caractéristiques.

Dans l'ensemble, l'utilisation de toutes les caractéristiques semble être une approche efficace pour améliorer la performance de vos classifieurs.

Quelle est la performance mesurée avec ce modèle sur le corpus de test ?

```python
# la méthode score pour évaluer la précision du modèle SVM sur les données de test :
accuracy = best_model.score(X_test, y_test)
print(f"Précision du modèle SVM sur les données de test : {accuracy:.2f}")
```

```
Précision du modèle SVM sur les données de test : 0.86
```

Quelle est la classe pour laquelle la prédiction fait le plus d'erreurs ?

```
Matrice de confusion :
[[11731   704]
 [ 1605  2241]]
```

La classe pour laquelle la prédiction fait le plus d'erreurs est celle représentée par les faux positifs et les faux négatifs dans la matrice de confusion. Dans la matrice de confusion que nous avons fournie, nous pouvons voir que la classe avec le plus grand nombre d'erreurs est la deuxième classe (la classe positive), qui comprend 1605 faux négatifs (instances réelles appartenant à cette classe mais prédites à tort comme n'appartenant pas à la classe) et 704 faux positifs (instances réelles n'appartenant pas à cette classe mais prédites à tort comme appartenant à la classe).

En d'autres termes, le modèle a du mal à prédire correctement la deuxième classe. Cela peut être un domaine d'amélioration réduire les erreurs pour cette classe spécifique.