Repte de Ciència de Dades per a la Detecció de Zones Sorolloses a Barcelona en el marc d'un repte proposat als alumnes

Projecte curs 2023-2024 Introducció a la ciència de dades amb Python (B2) Ed.2

Implementació d'un Repte per als alumnes de DAW2 que forma part de l'assignatura M0704 Serveis Web

Sergi Grau

Un projecte de clustering, o agrupament, passa per diverses fases. A continuació es descriuen les fases generals:

1. **Entendre el problema i definir l'objectiu**: Aquesta és la fase inicial on es defineix el problema i es determina com el clustering pot ajudar a resoldre'l. També es defineixen els objectius del projecte.

2. **Recopilació de dades**: En aquesta fase, es recopilen les dades necessàries per al projecte. Les dades poden provenir de diverses fonts, com ara bases de dades, fitxers, APIs, web scraping, etc.

3. **Preprocessament de dades**: Les dades recopilades es netegen i preprocessen. Això pot incloure la gestió de valors perduts, la normalització de dades, la codificació de variables categòriques, la reducció de la dimensionalitat, etc.

4. **Selecció de característiques**: En aquesta fase, es seleccionen les característiques (variables) que es faran servir per al clustering. Això pot implicar tècniques d'anàlisi de correlació, importància de característiques, etc.

5. **Selecció del model de clustering**: Es selecciona l'algoritme de clustering que es farà servir. Això pot dependre de la naturalesa de les dades i dels objectius del projecte. Alguns exemples d'algoritmes de clustering són K-means, DBSCAN, clustering jeràrquic, etc.

6. **Entrenament del model**: L'algoritme de clustering seleccionat s'entrena amb les dades. Això implica l'ajust dels paràmetres de l'algoritme per obtenir els millors resultats possibles.

7. **Avaluació del model**: El model de clustering es valida i s'avalua. Això pot implicar l'ús de mètriques com la silueta o l'índex de Davies-Bouldin.

8. **Interpretació dels resultats**: Finalment, es interpreten els resultats del clustering. Això pot implicar l'anàlisi de les característiques dels grups, la visualització dels resultats, etc.

9. **Implementació i seguiment**: El model es pot implementar en un entorn de producció, i es pot fer un seguiment del seu rendiment al llarg del temps.

Aquestes són les fases generals d'un projecte de clustering. Els detalls específics poden variar en funció de les necessitats del projecte.

Instal·lem les biblioteques necessaries pel notebook

In [ ]:
```
!pip3 install matplotlib
!pip3 install seaborn
!pip3 install np
!pip3 install pandas
```

```
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.fr
amework/Versions/3.11/lib/python3.11/site-packages (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib) (0.
12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotli
b) (4.50.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotli
b) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib)
(1.26.4)
Requirement already satisfied: packaging>=20.0 in /Users/imac2/Library/Pyt
hon/3.11/lib/python/site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from matplotlib) (10.2.
0)
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib)
(3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Framework
s/Python.framework/Versions/3.11/lib/python3.11/site-packages (from matplo
tlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.11/lib/python3.11/site-packages (from python-dateutil>=2.
7->matplotlib) (1.16.0)
Requirement already satisfied: seaborn in /Library/Frameworks/Python.frame
work/Versions/3.11/lib/python3.11/site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /Library/Framework
s/Python.framework/Versions/3.11/lib/python3.11/site-packages (from seabor
n) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /Library/Frameworks/Python.f
ramework/Versions/3.11/lib/python3.11/site-packages (from seaborn) (2.2.1)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /Library/Framewo
rks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from seab
orn) (3.8.3)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib!=3.
6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (4.50.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Py
thon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /Users/imac2/Library/Pyt
hon/3.11/lib/python/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(24.0)
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.fra
mework/Versions/3.11/lib/python3.11/site-packages (from matplotlib!=3.6.1,
>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Pyt
```

```
hon.framework/Versions/3.11/lib/python3.11/site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Framework
s/Python.framework/Versions/3.11/lib/python3.11/site-packages (from matplo
tlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from pandas>=1.2->se
aborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from pandas>=1.2->
seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.11/lib/python3.11/site-packages (from python-dateutil>=2.
7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Requirement already satisfied: np in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (1.0.2)
Requirement already satisfied: pandas in /Library/Frameworks/Python.framew
ork/Versions/3.11/lib/python3.11/site-packages (2.2.1)
Requirement already satisfied: numpy<2,>=1.23.2 in /Library/Frameworks/Pyt
hon.framework/Versions/3.11/lib/python3.11/site-packages (from pandas) (1.
26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /Library/Framewor
ks/Python.framework/Versions/3.11/lib/python3.11/site-packages (from panda
s) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from pandas) (2024.
1)
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Pytho
n.framework/Versions/3.11/lib/python3.11/site-packages (from pandas) (202
4.1)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.11/lib/python3.11/site-packages (from python-dateutil>=2.
8.2->pandas) (1.16.0)
```

1. Entendre el problema i definir l'objectiu: Aquesta és la fase inicial on es defineix el problema i es determina com el clustering pot ajudar a resoldre'l. També es defineixen els objectius del projecte. Aquest punt està especificat en el notebook del repte proposat als alumnes

2.Recopilació de dades: En aquesta fase, es recopilen les dades necessàries per al repte. Les dades poden provenir de diverses fonts, com ara bases de dades, fitxers, APIs, web scraping, etc. En el nostre cas venen del propi ajuntament amb OpenData.

In [ ]:
```python
import pandas as pd

#  llegim el fitxer CSV
df = pd.read_csv('2017_tramer_mapa_estrategic_soroll_bcn.csv')
# mostrem les primeres fileres del DataFrame
df.head()
```

Out[ ]:

	TRAM	TOTAL_D	TOTAL_E	TOTAL_N	TOTAL_DEN	TRANSIT_D	TRANSIT_E
0	T04719W	70 - 75 dB(A)	65 - 70 dB(A)	60 - 65 dB(A)	70 - 75 dB(A)	70 - 75 dB(A)	65 - 70 dB(A)
1	T19941Z	45 - 50 dB(A)	45 - 50 dB(A)	< 40 dB(A)	45 - 50 dB(A)	40 - 45 dB(A)	40 - 45 dB(A)
2	T18111R	55 - 60 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)
3	T03222Y	60 - 65 dB(A)	55 - 60 dB(A)	60 - 65 dB(A)	65 - 70 dB(A)	60 - 65 dB(A)	55 - 60 dB(A)
4	T17625I	55 - 60 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	60 - 65 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)

5 rows × 27 columns

Explicació dels camps del Dataframe

Les dades que es mostren contenen el tram **TRAM** de carrer i els nivells sonors de diferents tipus de fonts sonores i també el nivell de soroll global (la suma de totes les fonts). El tipus de font sonora el podem classificar en dos grups.

- Soroll de trànsit viari. **TRANSIT**
- Soroll de grans infraestructures viàries. **GI**
- Soroll de trànsit ferroviari i tramvia. **FFCC**
- Soroll industrial. **INDUST**

- Soroll en carrers de vianants. **VIANANTS**
- Soroll d'oci i aglomeració de persones. **OCI PATIS**

> A més cadascun d'ells s'indiquen per nivells sonors mitjans anuals per a cada període horari. Així doncs:

> Fonts sonores addicionals, que són de rellevància a la ciutat:

> Fonts sonores demanades per normativa:

- Ld és el nivell sonor mitjà anual pel període de dia (7-21h). **D**
- Le és el nivell sonor mitjà anual pel període de vespre (21-23h). **E**
- Ln és el nivell sonor mitjà anual pel període de nit (23-7h). **N**

A més, hi ha un quart índex acústic, el ponderat dia-vespre-nit (Lden), que correspon a una mitjana ponderada dels índexs Ld, Le i Ln, penalitzant els períodes vespre i nit.

L'índex Lden no esta inclòs als mapes de dades ambientals ja que l'Ordenança de Medi Ambient de la ciutat no l'utilitza. No obstant, degut a que és un paràmetre exigit

per normativa europea, es poden baixar els mapes d'aquest índex acústic a la pagina web d'Open Data BCN.

In [ ]:
```
# Mida del dataframe, files i nombre de camps
df.shape
```

Out[ ]: (17958, 27)

In [ ]:
```
# Mostrem un resum del mateix
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17958 entries, 0 to 17957
Data columns (total 27 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   TRAM         17958 non-null  object
 1   TOTAL_D      17958 non-null  object
 2   TOTAL_E      17958 non-null  object
 3   TOTAL_N      17958 non-null  object
 4   TOTAL_DEN    17958 non-null  object
 5   TRANSIT_D    17958 non-null  object
 6   TRANSIT_E    17958 non-null  object
 7   TRANSIT_N    17958 non-null  object
 8   TRANSIT_DEN  17958 non-null  object
 9   GI_TR_D      17958 non-null  object
 10  GI_TR_E      17958 non-null  object
 11  GI_TR_N      17958 non-null  object
 12  GI_TR_DEN    17958 non-null  object
 13  FFCC_D       17958 non-null  object
 14  FFCC_E       17958 non-null  object
 15  FFCC_N       17958 non-null  object
 16  FFCC_DEN     17958 non-null  object
 17  INDUST_D     17958 non-null  object
 18  INDUST_E     17958 non-null  object
 19  INDUST_N     17958 non-null  object
 20  INDUST_DEN   17958 non-null  object
 21  VIANANTS_D   17958 non-null  object
 22  VIANANTS_E   17958 non-null  object
 23  OCI_N        17958 non-null  object
 24  PATIS_D      17958 non-null  object
 25  PATIS_E      17958 non-null  object
 26  GEOM_WKT     17958 non-null  object
dtypes: object(27)
memory usage: 3.7+ MB
```

In [ ]:
```
# Mostrem alguns estadístics del DataFrame
df.describe()
```

Out[ ]:

	TRAM	TOTAL_D	TOTAL_E	TOTAL_N	TOTAL_DEN	TRANSIT_D	TRANS
count	17958	17958	17958	17958	17958	17958	1
unique	17958	9	8	7	9	9	
top	T04719W	60 - 65 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)	60 - 65 dB(A)	60 - 65 dB(A)	55 d
freq	1	3895	3847	3641	3936	3555	

4 rows × 27 columns

3. Preprocessament de dades: Les dades recopilades es netegen i preprocessen. Això pot incloure la gestió de valors perduts, la normalització de dades, la codificació de variables categòriques, la reducció de la dimensionalitat, etc.

Comencem per la Neteja de les dades (tractament de valors perduts, eliminació de duplicats,tractament de les dades categoriques).

In [ ]:
```
# verifiquem els valors que son NA
df = df.dropna()
```

In [ ]:
```
# Idemtifiquem les files duplicades i les eliminem
df = df.drop_duplicates()
```

In [ ]:
```
# conversió de les dades categòriques, totes les columnes
# Mirem quants intervals de DB hi ha
df
```

Out[ ]:

	TRAM	TOTAL_D	TOTAL_E	TOTAL_N	TOTAL_DEN	TRANSIT_D	TRANSI
0	T04719W	70 - 75 dB(A)	65 - 70 dB(A)	60 - 65 dB(A)	70 - 75 dB(A)	70 - 75 dB(A)	65 d
1	T19941Z	45 - 50 dB(A)	45 - 50 dB(A)	< 40 dB(A)	45 - 50 dB(A)	40 - 45 dB(A)	40 d
2	T18111R	55 - 60 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)	55 d
3	T03222Y	60 - 65 dB(A)	55 - 60 dB(A)	60 - 65 dB(A)	65 - 70 dB(A)	60 - 65 dB(A)	55 d
4	T17625I	55 - 60 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	60 - 65 dB(A)	55 - 60 dB(A)	55 d
...	...	...	...	...	...	...	
17953	T08868Y	55 - 60 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	55 - 60 dB(A)	55 - 60 dB(A)	55 d
17954	P98700	50 - 55 dB(A)	40 - 45 dB(A)	< 40 dB(A)	50 - 55 dB(A)	40 - 45 dB(A)	40 d
17955	P98690	40 - 45 dB(A)	< 40 dB(A)	< 40 dB(A)	40 - 45 dB(A)	40 - 45 dB(A)	< 40 d
17956	P27312	50 - 55 dB(A)	40 - 45 dB(A)	< 40 dB(A)	50 - 55 dB(A)	40 - 45 dB(A)	40 d
17957	P28659	55 - 60 dB(A)	45 - 50 dB(A)	40 - 45 dB(A)	55 - 60 dB(A)	50 - 55 dB(A)	45 d

17958 rows × 27 columns

In [ ]:
```
#reemplacem els valors de les columnes que contenen intervals de dB < 40
columns = ["TOTAL_D","TOTAL_E","TOTAL_N","TOTAL_DEN","TRANSIT_D","TRANSIT

for col in columns:
    df[col] = df[col].replace('< 40 dB(A)', '0 - 40 dB(A)')
```

In [ ]:
```
# obtdenim els valors únics de la columna TOTAL_D
import np
cat = np.unique(df['TOTAL_D'])
cat
```

Out[ ]:
```
array(['0 - 40 dB(A)', '40 - 45 dB(A)', '45 - 50 dB(A)', '50 - 55 dB
(A)',
       '55 - 60 dB(A)', '60 - 65 dB(A)', '65 - 70 dB(A)', '70 - 75 dB
(A)',
       '75 - 80 dB(A)'], dtype=object)
```

In [ ]:
```
# transformem les dades categòriques a codi
from sklearn.preprocessing import LabelEncoder

columnes = ["TOTAL_D","TOTAL_E","TOTAL_N","TOTAL_DEN","TRANSIT_D","TRANSI

# Inicialitzem el LabelEncoder
le = LabelEncoder()
# transformen categoriques a codi
le.fit(cat)

equivalences = {label: index for index, label in enumerate(le.classes_)}
equivalences
```

Out[ ]:
```
{'0 - 40 dB(A)': 0,
 '40 - 45 dB(A)': 1,
 '45 - 50 dB(A)': 2,
 '50 - 55 dB(A)': 3,
 '55 - 60 dB(A)': 4,
 '60 - 65 dB(A)': 5,
 '65 - 70 dB(A)': 6,
 '70 - 75 dB(A)': 7,
 '75 - 80 dB(A)': 8}
```

In [ ]:
```
for col in columnes:
    # Transforma la columna utilitzant l'LabelEncoder ajustat i afegeix-l
    df[col + '_codis'] = le.transform(df[col])
```

In [ ]:
```
# Mostrem només les columnes que hem afegit
df_codis = df.filter(regex='_codis$')
df_codis = pd.concat([df['TRAM'], df_codis], axis=1)
df_codis
```

Out[ ]:

	TRAM	TOTAL_D_codis	TOTAL_E_codis	TOTAL_N_codis	TOTAL_DEN_co
0	T04719W	7	6	5	
1	T19941Z	2	2	0	
2	T18111R	4	4	3	
3	T03222Y	5	4	5	
4	T17625I	4	4	3	
...	...	...	...	...	
17953	T08868Y	4	4	3	
17954	P98700	3	1	0	
17955	P98690	1	0	0	
17956	P27312	3	1	0	
17957	P28659	4	2	1	

17958 rows × 26 columns

Per mostrar la correlació entre les columnes del DataFrame `df_codis`, pots utilitzar el mètode `corr()` de pandas. Aquest mètode retorna un nou DataFrame on cada

entrada (i, j) és la correlació entre les columnes i i j del DataFrame original.

Aquí tens un exemple de com fer-ho:

In [ ]:
```
# Asumim que df_codis és el teu DataFrame
codis_columns = [col for col in df_codis.columns if col.endswith('codis')]

correlation = df_codis[codis_columns].corr()

# Mostrar la correlació
correlation
```

Out[ ]:

	TOTAL_D_codis	TOTAL_E_codis	TOTAL_N_codis	TOTAL_DEN
TOTAL_D_codis	1.000000	0.956883	0.912683	0.9
TOTAL_E_codis	0.956883	1.000000	0.947950	0.
TOTAL_N_codis	0.912683	0.947950	1.000000	0.
TOTAL_DEN_codis	0.955622	0.953742	0.943917	1.(
TRANSIT_D_codis	0.957116	0.928012	0.915865	0.
TRANSIT_E_codis	0.929699	0.957771	0.943398	0.
TRANSIT_N_codis	0.923549	0.948828	0.955662	0.
TRANSIT_DEN_codis	0.931506	0.924877	0.918799	0.9
GI_TR_D_codis	0.510541	0.508306	0.500959	0.4
GI_TR_E_codis	0.512250	0.514157	0.508215	0.4
GI_TR_N_codis	0.513593	0.515484	0.513613	0.
GI_TR_DEN_codis	0.508534	0.506106	0.497100	0.
FFCC_D_codis	0.104873	0.110205	0.106558	0.
FFCC_E_codis	0.102635	0.109141	0.104412	0.
FFCC_N_codis	0.095235	0.100915	0.101102	0.(
FFCC_DEN_codis	0.107602	0.113002	0.112361	0.
INDUST_D_codis	0.020080	0.021978	0.055643	0.(
INDUST_E_codis	-0.002685	0.007340	0.046852	0.(
INDUST_N_codis	-0.002474	0.007548	0.047077	0.(
INDUST_DEN_codis	0.009606	0.014533	0.050565	0.
VIANANTS_D_codis	0.006578	0.023184	0.030487	0.(
VIANANTS_E_codis	0.010487	0.033643	0.059405	0.(
OCI_N_codis	0.112620	0.135290	0.263743	0.
PATIS_D_codis	0.010112	-0.040402	-0.181165	-0.(
PATIS_E_codis	0.036158	0.053980	-0.121834	0.(

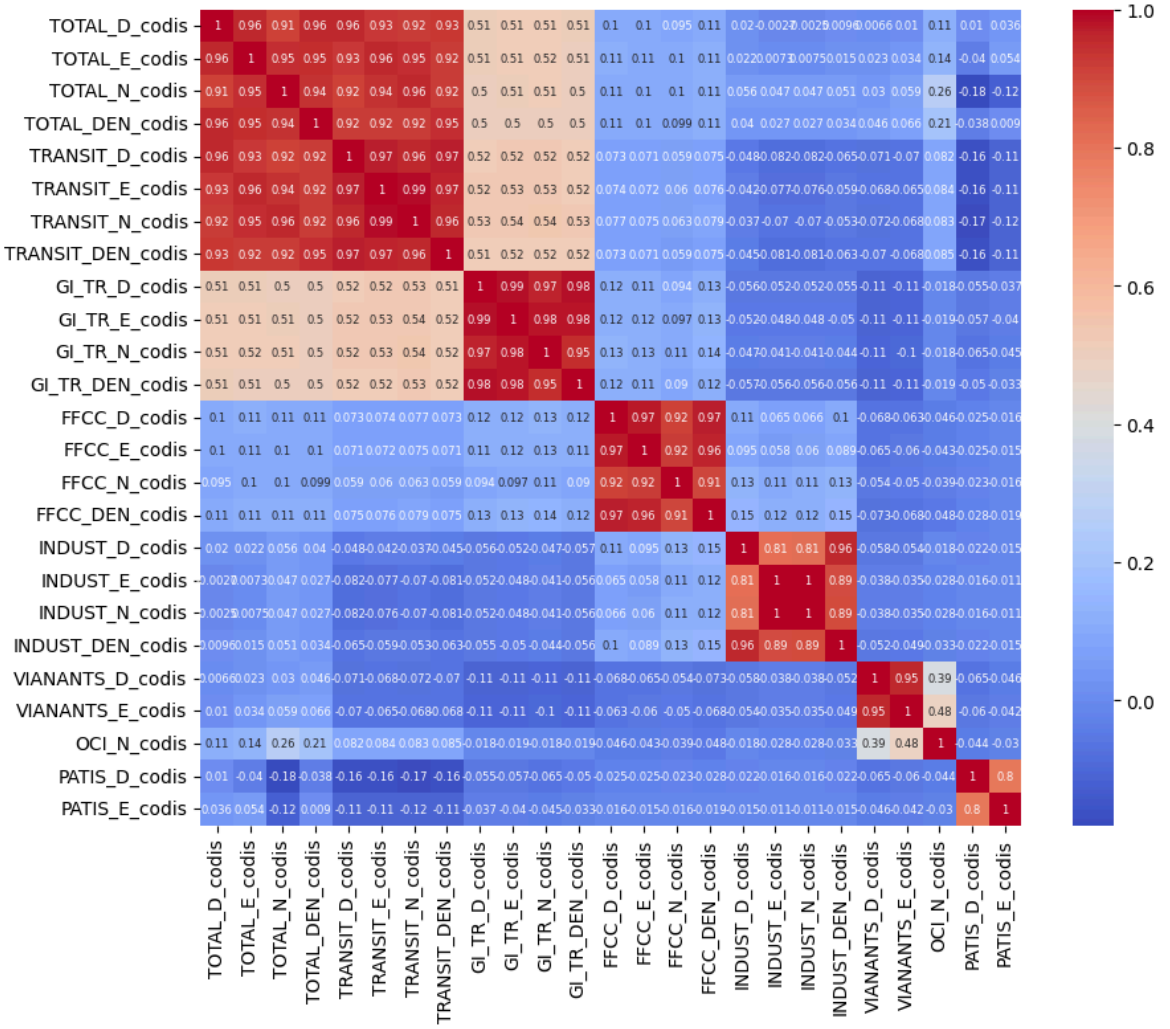
25 rows × 25 columns

4. Selecció de característiques: En aquesta fase, es seleccionen les característiques (variables) que es faran servir per al clustering. Això pot implicar tècniques d'anàlisi de correlació, importància de característiques, etc.

Tenim les correlacions de `df_codis`. Les correlacions varien entre -1 i 1. Un valor de 1 indica una correlació positiva perfecta, un valor de -1 indica una correlació negativa perfecta, i un valor de 0 indica cap correlació.

Per crear un mapa de correlacions (també conegut com a heatmap de correlacions) pots utilitzar la biblioteca seaborn. Observem el gran impacte que té el trànsit en el total de soroll amb una alta correlació.

In [ ]:
```
# Importem les biblioteques necessàries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Asumim que df_codis és el teu DataFrame
codis_columns = [col for col in df_codis.columns if col.endswith('codis')]

correlation = df_codis[codis_columns].corr()

# Crear el mapa de correlacions
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', annot_kws={"size":

# Mostrar el mapa
plt.show()
```



Aquest codi crearà un mapa de correlacions de les columnes de `df_codis` que acaben amb 'codis'. Les colors del mapa varien segons el valor de la correlació: les correlacions positives es mostren en colors càlids (cap a l'extrem vermell) i les correlacions negatives es mostren en colors freds (cap a l'extrem blau).

En aquest apartat fem la normalització de les dades (escalat de les característiques perquè tinguin rangs similars) i possiblement la reducció de la dimensionalitat si el conjunt de dades té un gran nombre de característiques.

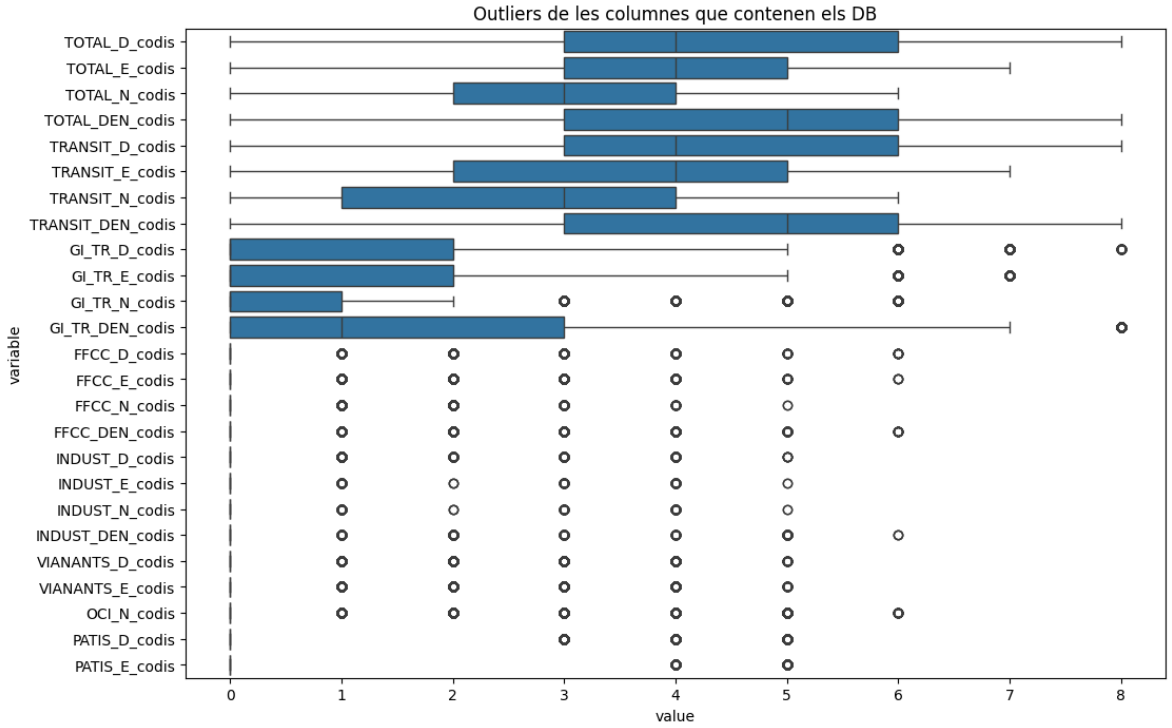
In [ ]:
```
#mostrem una descripció estadística de les dades
df_codis.describe()
```

Out[ ]:

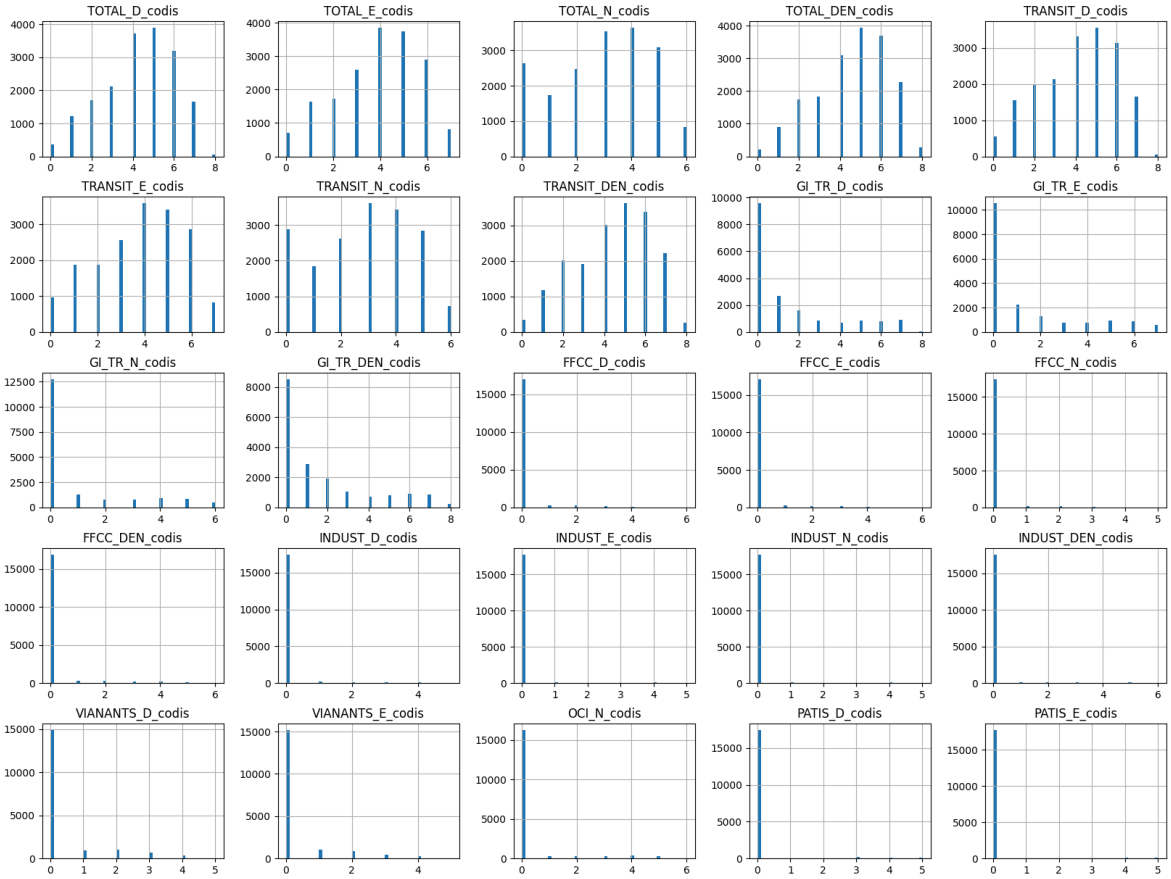
	TOTAL_D_codis	TOTAL_E_codis	TOTAL_N_codis	TOTAL_DEN_codis	TRANS
count	17958.000000	17958.000000	17958.000000	17958.000000	17
mean	4.262947	3.901659	2.914356	4.580187	
std	1.783585	1.796867	1.773596	1.795250	
min	0.000000	0.000000	0.000000	0.000000	
25%	3.000000	3.000000	2.000000	3.000000	
50%	4.000000	4.000000	3.000000	5.000000	
75%	6.000000	5.000000	4.000000	6.000000	
max	8.000000	7.000000	6.000000	8.000000	

8 rows × 25 columns

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Melt the DataFrame to long-form
df_codis_long = df_codis.melt()

# Filter the DataFrame to include only columns that end with '_codis'
df_codis_long = df_codis_long[df_codis_long['variable'].str.endswith('_co

# Create a boxplot
plt.figure(figsize=(12, 8))
sns.boxplot(x='value', y='variable', data=df_codis_long)
plt.title('Outliers de les columnes que contenen els DB')
plt.show()
```



In [ ]:
```
# mostrem les normals
df_codis.hist(bins=50, figsize=(20,15))
plt.show()
```



Per identificar i eliminar els valors atípics (outliers) d'un DataFrame de pandas, pots utilitzar diverses tècniques. Una de les més comunes és el mètode del rang interquartílic (IQR). Aquí tens un exemple de com podríes fer-ho:

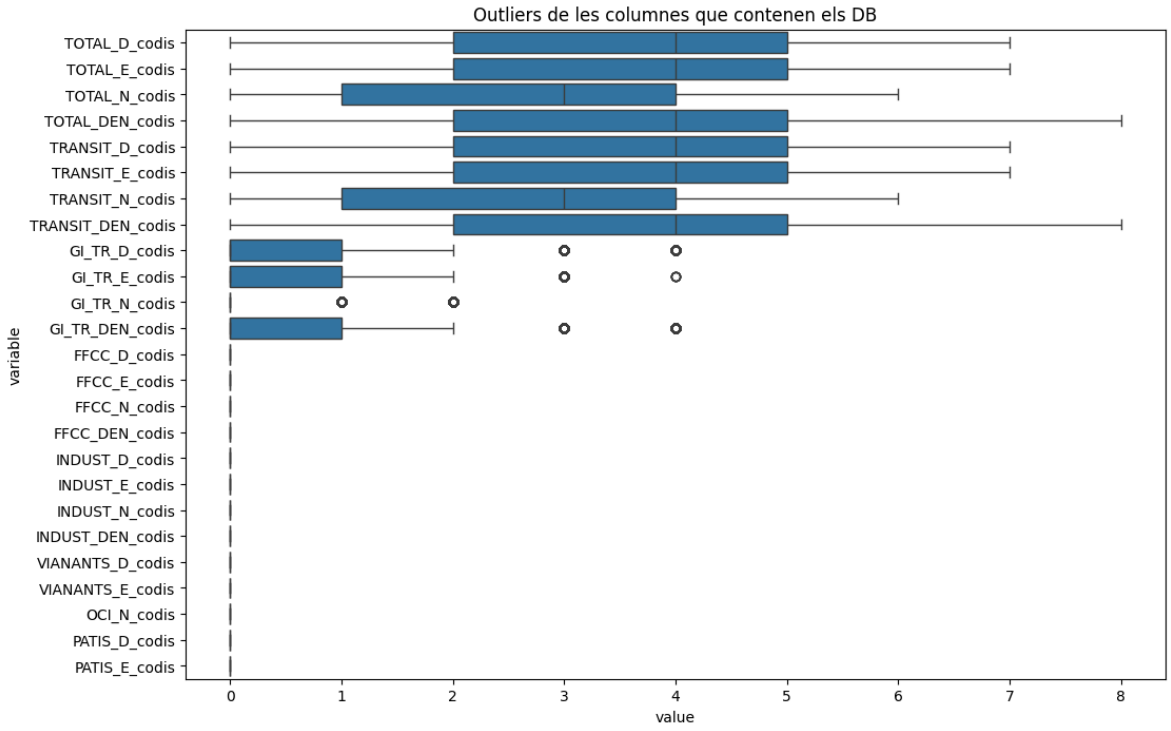
Primer, calculem l'IQR per a cada columna en el DataFrame. Després, definim els límits superior i inferior per als valors atípics. Finalment, filtrem el DataFrame per eliminar aquests valors atípics.

Aquest codi eliminarà qualsevol fila a `df_codis` que contingui almenys un valor atípic. Tingues en compte que aquest mètode pot no ser apropiat per a tots els conjunts de dades, especialment si els valors atípics són significatius per a la teva anàlisi.

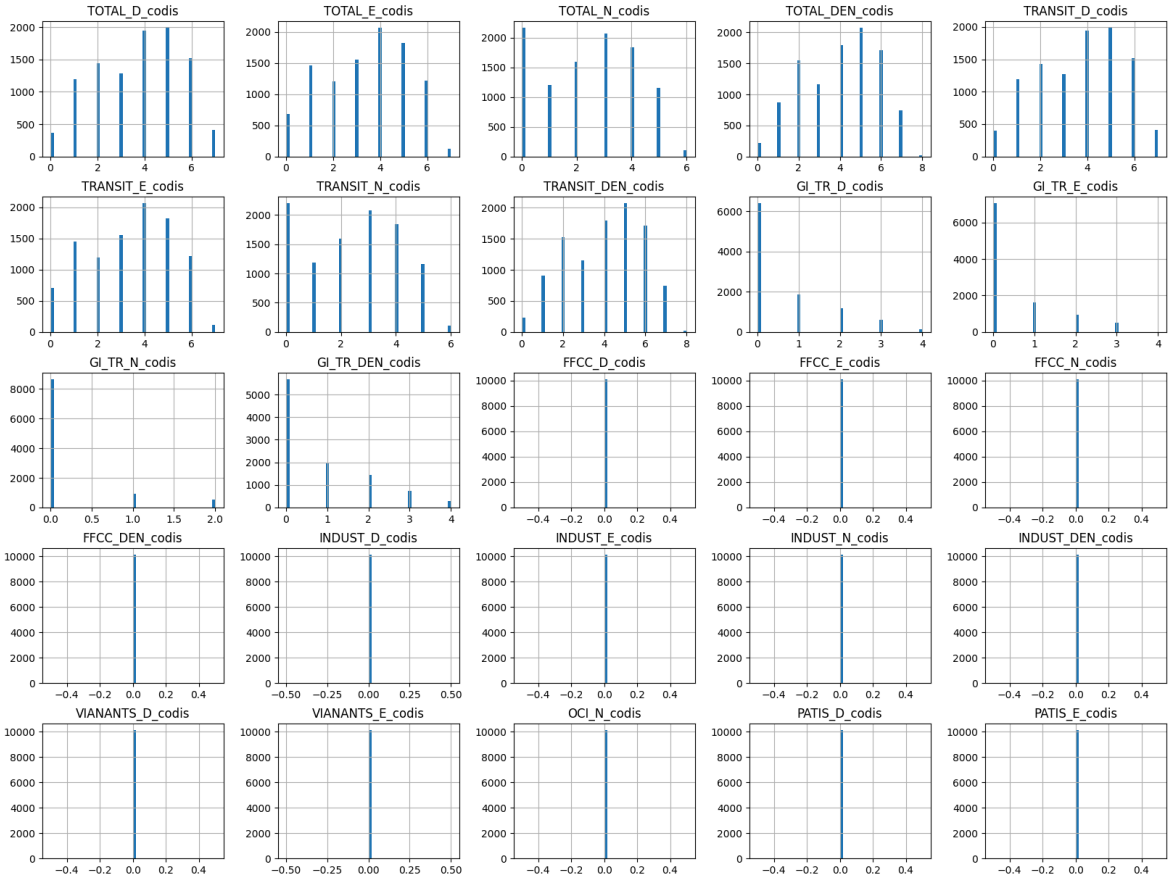
In [ ]:
```
# calculem IQR per cada columna numèrica
Q1 = df_codis[codis_columns].quantile(0.25)
Q3 = df_codis[codis_columns].quantile(0.75)
IQR = Q3 - Q1

# filtren els valors
filter = (df_codis[codis_columns] >= (Q1 - 1.5 * IQR)) & (df_codis[codis_

#Apliqueu el filtre a les dades
df_codis_no_outliers = df_codis.copy()
df_codis_no_outliers[codis_columns] = df_codis[codis_columns][filter]
df_codis_no_outliers.dropna(inplace=True)
```

De nou mostrem els outliers en les columnes

In [ ]:
```
# Melt the DataFrame to long-form
df_codis_long = df_codis_no_outliers.melt()

# Filter the DataFrame to include only columns that end with '_codis'
df_codis_long = df_codis_long[df_codis_long['variable'].str.endswith('_co

# Create a boxplot
plt.figure(figsize=(12, 8))
sns.boxplot(x='value', y='variable', data=df_codis_long)
plt.title('Outliers de les columnes que contenen els DB')
plt.show()
```



In [ ]:
```
# mostramos también las normales
df_codis_no_outliers.hist(bins=50, figsize=(20,15))
plt.show()
```



In [ ]:
```
#En un primer nivell anem a reduir manualment la dimensionalitat del data
df_codis_no_outliers['TOTAL_tarda_vespre'] = df_codis_no_outliers['TOTAL_
df_codis_no_outliers = df_codis_no_outliers[['TRAM', 'TOTAL_D_codis', 'TO
df_codis_no_outliers
```

Out[ ]:

	TRAM	TOTAL_D_codis	TOTAL_tarda_vespre
0	T04719W	7	11
2	T18111R	4	7
5	T05360P	4	7
6	T08863T	3	5
7	T00236S	2	3
...	...	...	...
17949	T17103N	5	7
17950	T07428A	5	9
17951	T02455J	5	9
17953	T08868Y	4	7
17955	P98690	1	0

10139 rows × 3 columns

5. Selecció del model de clustering: Es selecciona l'algoritme de clustering que es farà servir. Això pot dependre de la naturalesa de les dades i dels objectius del projecte. Alguns

exemples d'algoritmes de clustering són K-means, DBSCAN, clustering jeràrquic, etc.

En el nostre cas utilitzarem el K-means.

Recursos que expliquen com funciona K-means.

1. Documentació de Scikit-Learn sobre K-means: https://scikit-learn.org/stable/modules/clustering.html#k-means
2. Viquipèdia: https://ca.wikipedia.org/wiki/K-means
3. Coursera (Machine Learning by Andrew Ng): https://www.coursera.org/learn/machine-learning
4. Medium (Understanding the K-means Clustering Algorithm): https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
5. YouTube (StatQuest with Josh Starmer): https://www.youtube.com/watch?v=4b5d3muPQmA

6. Entrenament del model: L'algoritme de clustering seleccionat s'entrena amb les dades. Això implica l'ajust dels paràmetres de l'algoritme per obtenir els millors resultats possibles.

Per crear un model K-means a partir de les dades de `df_codis_no_outliers`, utilitzem la biblioteca sklearn

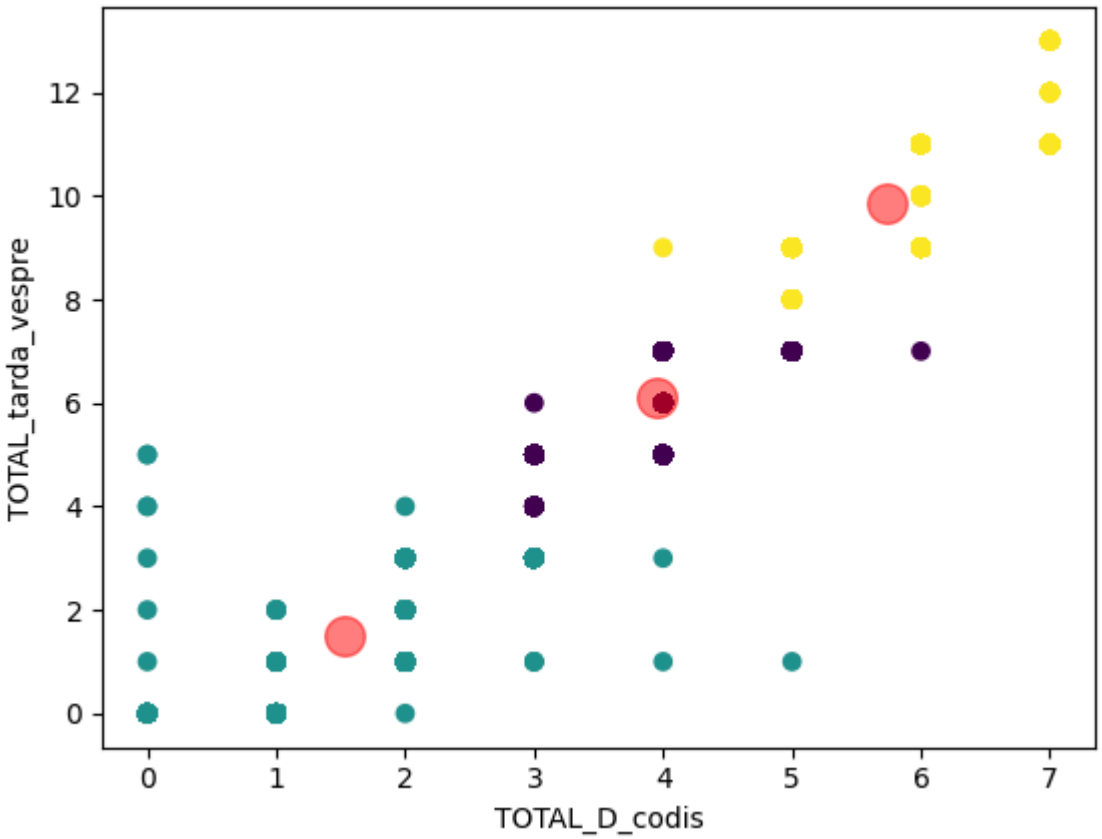
In [ ]:
```
# Importem les biblioteques necessàries
from sklearn.cluster import KMeans

# Asumim que df_codis_no_outliers és el teu DataFrame
# i que vols crear un model K-means amb 3 clústers
kmeans = KMeans(n_clusters=3)

# Ajustem el model a les dades
X = df_codis_no_outliers[['TOTAL_D_codis', 'TOTAL_tarda_vespre']]
kmeans.fit(X)

# Afegim les etiquetes dels clústers al DataFrame
df_codis_no_outliers['cluster'] = kmeans.labels_
```

Aquest codi crea un model K-means amb 3 clústers a partir de les columnes 'TOTAL_D_codis' i 'TOTAL_tarda_vespre' de `df_codis_no_outliers`. Després, afegim una nova columna al DataFrame que contindrà l'etiqueta del clúster assignada a cada fila.

Mostren un gràfic de punts del K-means amb la identificació dels centroïdes en vermell.

In [ ]:
```
# Creem el gràfic de punts
plt.scatter(df_codis_no_outliers['TOTAL_D_codis'], df_codis_no_outliers['
```

```
# Mostrem els centroïdes
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, alpha=0.5)

# Afegim les etiquetes dels eixos
plt.xlabel('TOTAL_D_codis')
plt.ylabel('TOTAL_tarda_vespre')

# Mostrem el gràfic
plt.show()
```



Aquest codi crearà un gràfic de punts on cada punt representa una fila de `df_codis_no_outliers` i el color del punt indica el clúster al qual pertany. També mostrarà els centroïdes dels clústers com a punts vermells.

7. Avaluació del model: El model de clustering es valida i s'avalua. Això pot implicar l'ús de mètriques com la silueta o l'índex de Davies-Bouldin.

Per avaluar el model K-means, pots utilitzar diverses mètriques com la inèrcia, el coeficient de silueta i l'índex de Davies-Bouldin.

In [ ]:
```
# Importem les biblioteques necessàries
from sklearn.metrics import silhouette_score, davies_bouldin_score

# Asumim que df_codis_no_outliers és el teu DataFrame
# i que kmeans és el teu model K-means

# Calculem la inèrcia
inertia = kmeans.inertia_
print(f'Inèrcia: {inertia}')
```

```
# Calculem el coeficient de silueta
silhouette = silhouette_score(df_codis_no_outliers[['TOTAL_D_codis', 'TOT
print(f'Coeficient de silueta: {silhouette}')

# Calculem l'índex de Davies-Bouldin
dbi = davies_bouldin_score(df_codis_no_outliers[['TOTAL_D_codis', 'TOTAL_
print(f'Índex de Davies-Bouldin: {dbi}')
```

```
Inèrcia: 17677.542611032382
Coeficient de silueta: 0.629756554400301
Índex de Davies-Bouldin: 0.5437357491110887
```

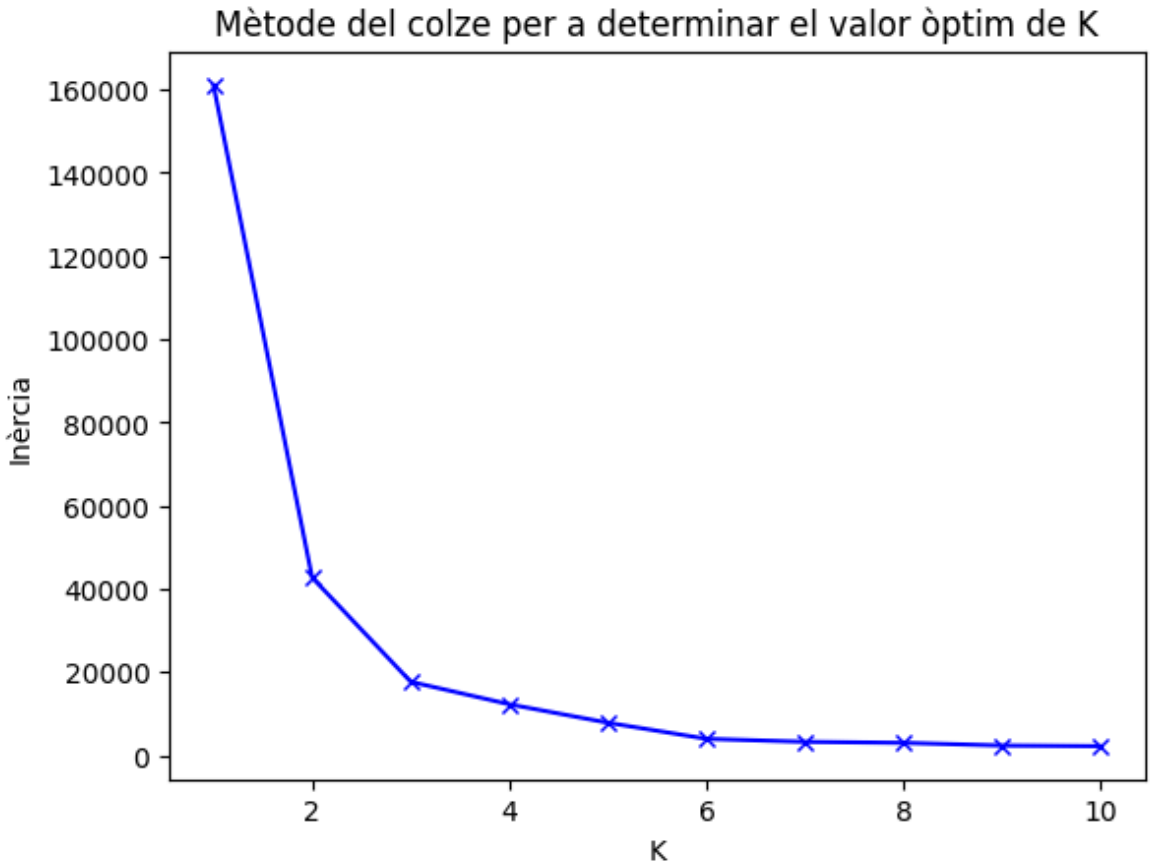
La inèrcia és la suma de les distàncies quadrades de cada mostra al seu centre de clúster més proper. És una mesura de la compactació dels clústers.

El coeficient de silueta és una mesura de com de semblants són les mostres dins del mateix clúster comparades amb les mostres d'altres clústers. Els valors varien entre -1 i 1, on un valor més alt indica que les mostres estan ben agrupades.

L'índex de Davies-Bouldin és una mesura de la similitud mitjana entre clústers, on un valor més baix és millor.

Per aplicar el mètode del colze (o codo) per determinar el valor òptim de K en el K-means, pots utilitzar el següent codi:

In [ ]:
```
# Creem una llista per a les inèrcies
inertias = []

# Provem diferents valors de K
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df_codis_no_outliers[['TOTAL_D_codis', 'TOTAL_tarda_vespre
    inertias.append(kmeans.inertia_)

# Creem el gràfic del mètode del colze
plt.plot(range(1, 11), inertias, 'bx-')
plt.xlabel('K')
plt.ylabel('Inèrcia')
plt.title('Mètode del colze per a determinar el valor òptim de K')
plt.show()
```



Com veiem en 2 canvia el colze, agafem doncs el seguent valor que és 3. És el que hem determinat d'un principi.

8. Interpretació dels resultats: Finalment, es interpreten els resultats del clustering. Això pot implicar l'anàlisi de les característiques dels grups, la visualització dels resultats, etc.

Visualment podem veure que hi ha una forta correlació entre el index de soroll diurn i nocturn, i que Barcelona mostra zones més tranquil·les i zones molt sorolloses. Algunes poden tenir un nivell baix de nivell soroll nocturn, però de dia tenir nivells elevats. En canvi les zones amb soroll nocturn elevat de dia també ho són, i per tant la població està molt penalitzada.

9. Implementació i seguiment: El model es pot implementar en un entorn de producció, i es pot fer un seguiment del seu rendiment al llarg del temps.

Aquesta és la part que has de fer com alumne, traspassar aquest model de clustering a una aplicació REST