

Explicació Dataset

Aquest dataset conté 2.240 files (clients) i 29 columnes, amb una combinació de dades numèriques, categòriques i temporals. És ric però brut, amb moltes inconsistències i soroll, per la qual cosa la prioritat és la neteja.

1. Estructura general:

- **Dades numèriques:** Ingressos, despeses en productes, nombre de visites web, any de naixement, etc.
- **Dades categòriques:** Nivell educatiu i estat civil.
- **Dades temporals:** Data d'alta del client (es convertirà en antiguitat en dies si es necessita en el futur).

2. Anàlisi de les variables categòriques:

- **Nivell educatiu:** Està descompensat amb un gran nombre d'educats universitaris. El grup **Basic** és molt petit, per la qual cosa podríem agrupar-lo amb **2n Cycle** i **Master** per simplificar-lo.
- **Estat civil:** La columna **Marital_Status** té valors inusuals com "**Alone**", "**Absurd**", i "**YOLO**" que s'han eliminat, ja que no són vàlids.

3. Anàlisi de les variables numèriques:

- **Ingressos:** És la variable més rellevant per segmentar els clients segons el seu poder adquisitiu.
- **Despesa:** Algunes persones tenen **valors de despesa zero** en productes, la qual cosa indica que no han comprat aquests productes.
- **Famílies:** Les variables **Kidhome** i **Teenhome** són útils per identificar si el client té fills o adolescents.

4. Qualitat de les dades: Nulls i Outliers:

- **Valors nuls:** Hi ha **24 files amb ingressos nuls**, que es decideix eliminar perquè representen només un **1% de les dades**.
- **Outliers:**
 - **Edat:** Alguns clients tenen **edats impossibles** (més de 120 anys). S'elimina qualsevol client nascut abans de 1920.

- **Ingressos extrems:** Es detecta un client amb **666.666 d'ingressos**, que es considera un outlier i es elimina per evitar distorsionar el clustering.

5. Altres Dades

Products

- MntWines: Amount spent on wine in last 2 years
- MntFruits: Amount spent on fruits in last 2 years
- MntMeatProducts: Amount spent on meat in last 2 years
- MntFishProducts: Amount spent on fish in last 2 years
- MntSweetProducts: Amount spent on sweets in last 2 years
- MntGoldProds: Amount spent on gold in last 2 years

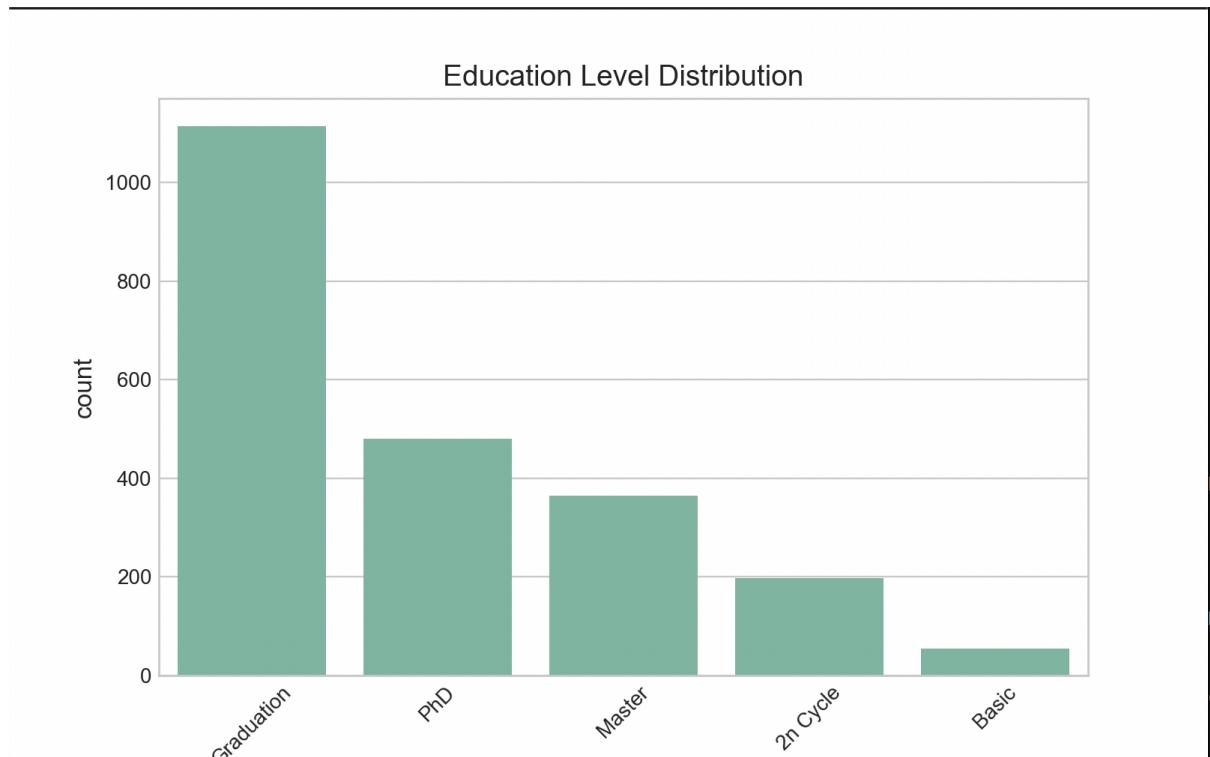
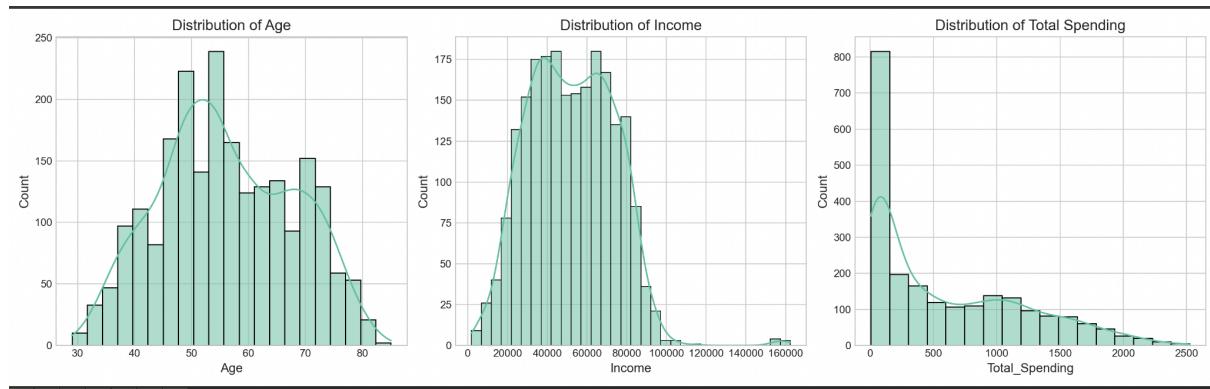
Promotion

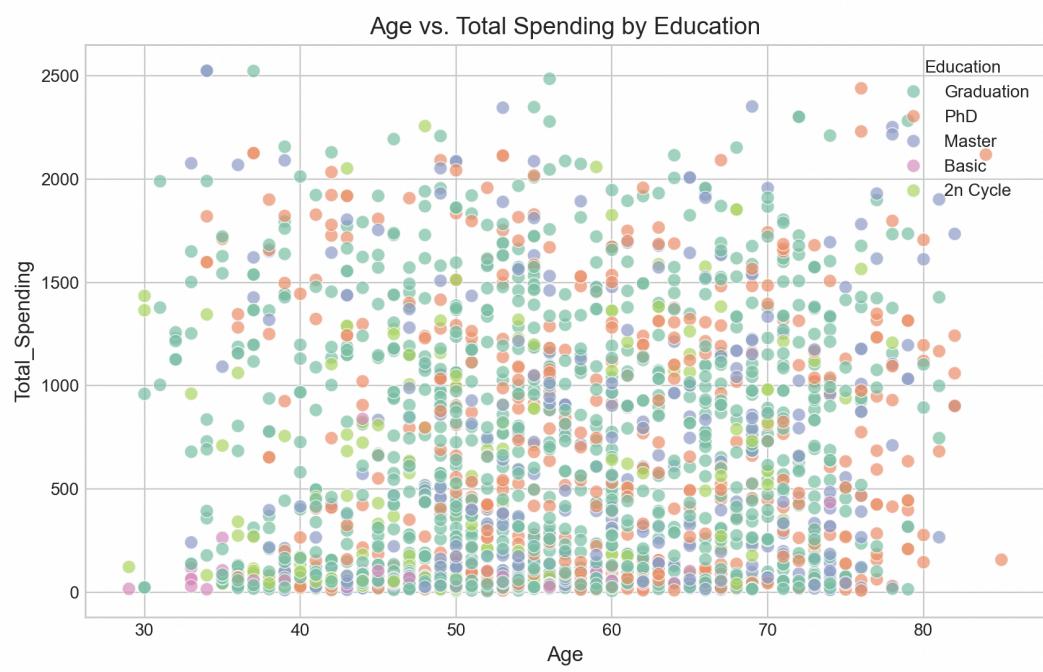
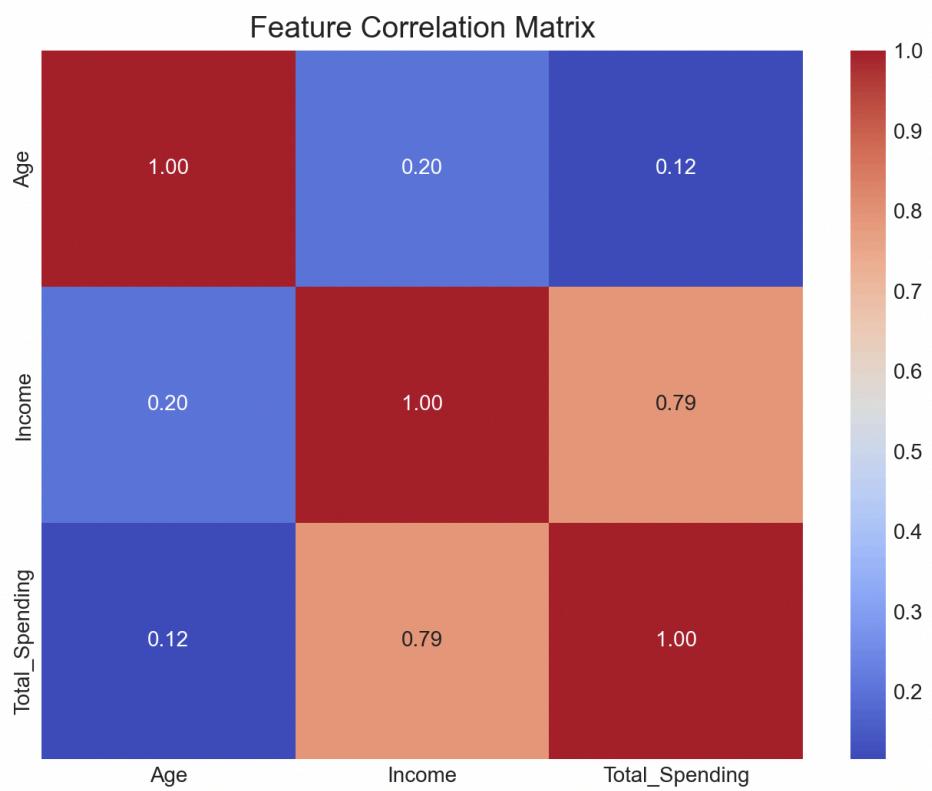
- NumDealsPurchases: Number of purchases made with a discount
- AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise
- AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
- AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
- AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise
- AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise
- Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

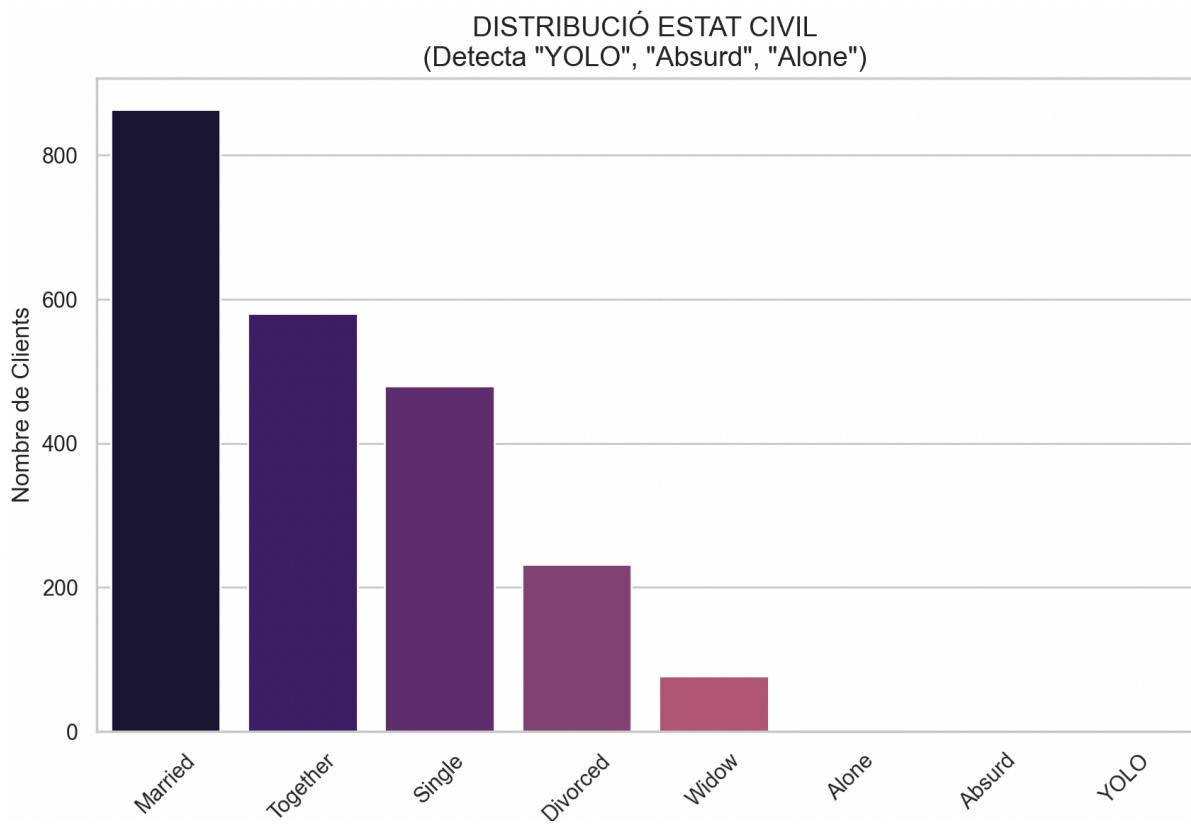
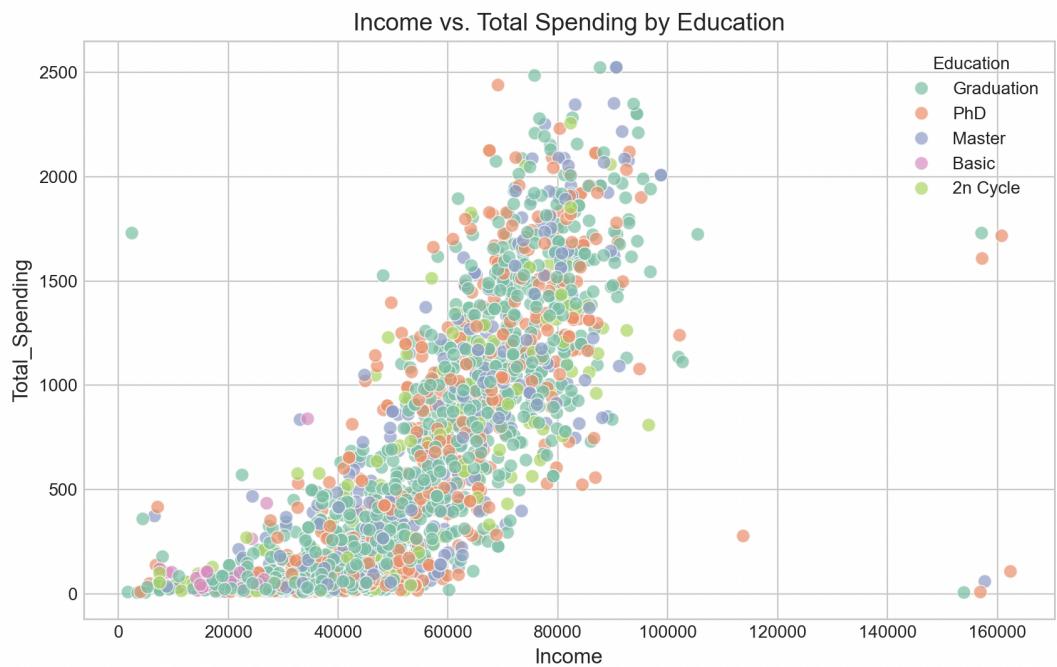
Place

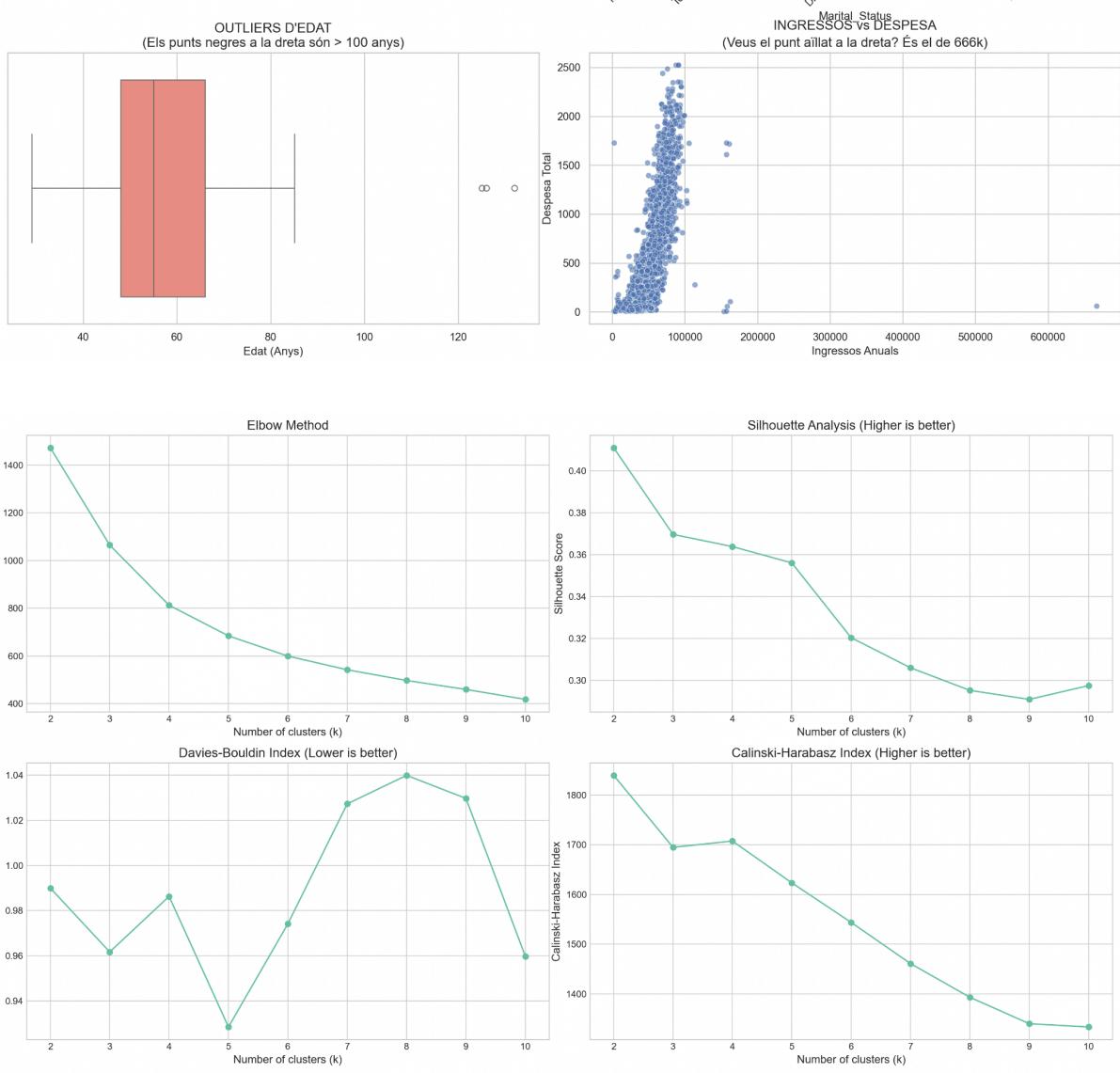
- NumWebPurchases: Number of purchases made through the company's website
- NumCatalogPurchases: Number of purchases made using a catalogue
- NumStorePurchases: Number of purchases made directly in stores
- NumWebVisitsMonth: Number of visits to company's website in the last month

Gràfics d'interpretació









Validar clustering (mètriques):

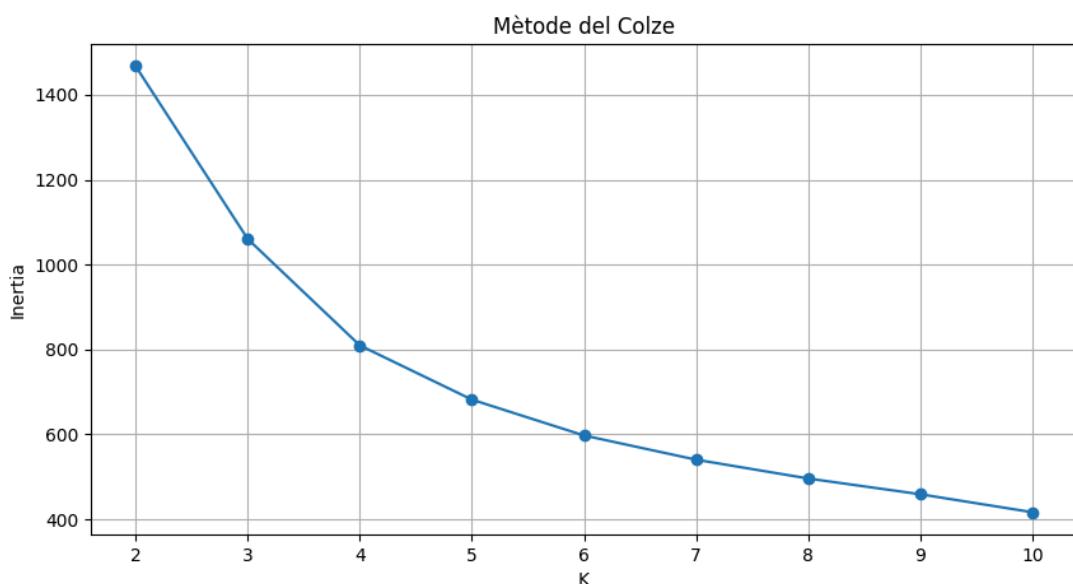
Cohesió (SSE): Calcula quant de "compactes" són els grups. Com més baix, millor (els punts estan molt a prop del seu centre).

Separació (BSS): Hem implementat una versió robusta basada en la distància entre els centres dels clústers. Com més alt, millor (els grups estan ben separats entre ells).

Correlació: Fa servir una mostra de 1.000 punts (perquè no trigui una eternitat) i mira si els punts que l'algorisme ha posat junts realment estan a prop en l'espai original. Un valor proper a -1 és perfecte (vol dir que incidència 1 es correspon amb distància petita).

Mapa de Calor (Heatmap): Reordena els clients segons el seu grup i pinta la distància. Si el clustering és bo, hauries de veure quadrats foscos (baixa distància) al llarg de la línia diagonal.

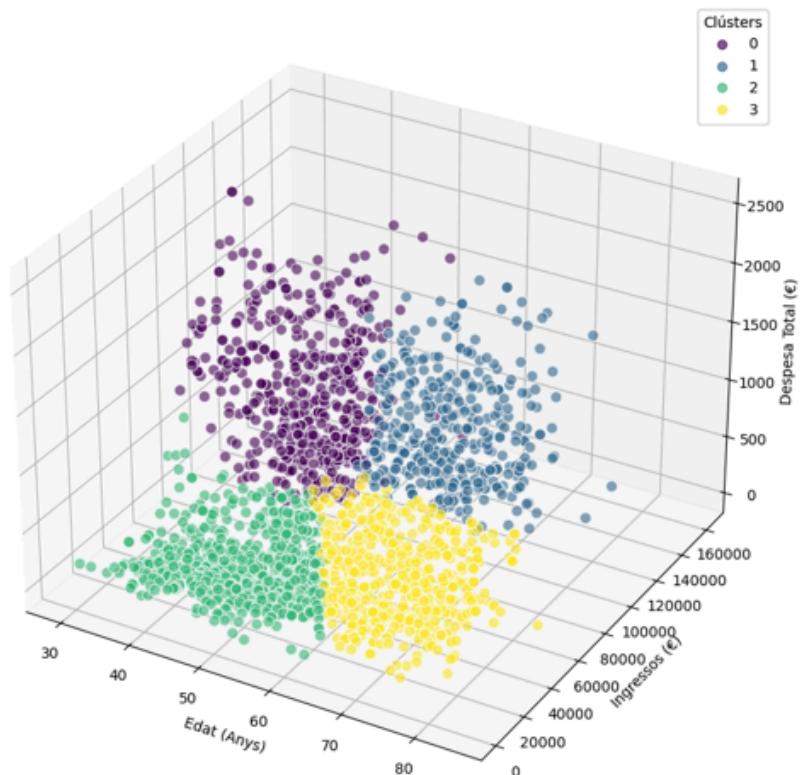
CONCLUSIÓ INICIAL



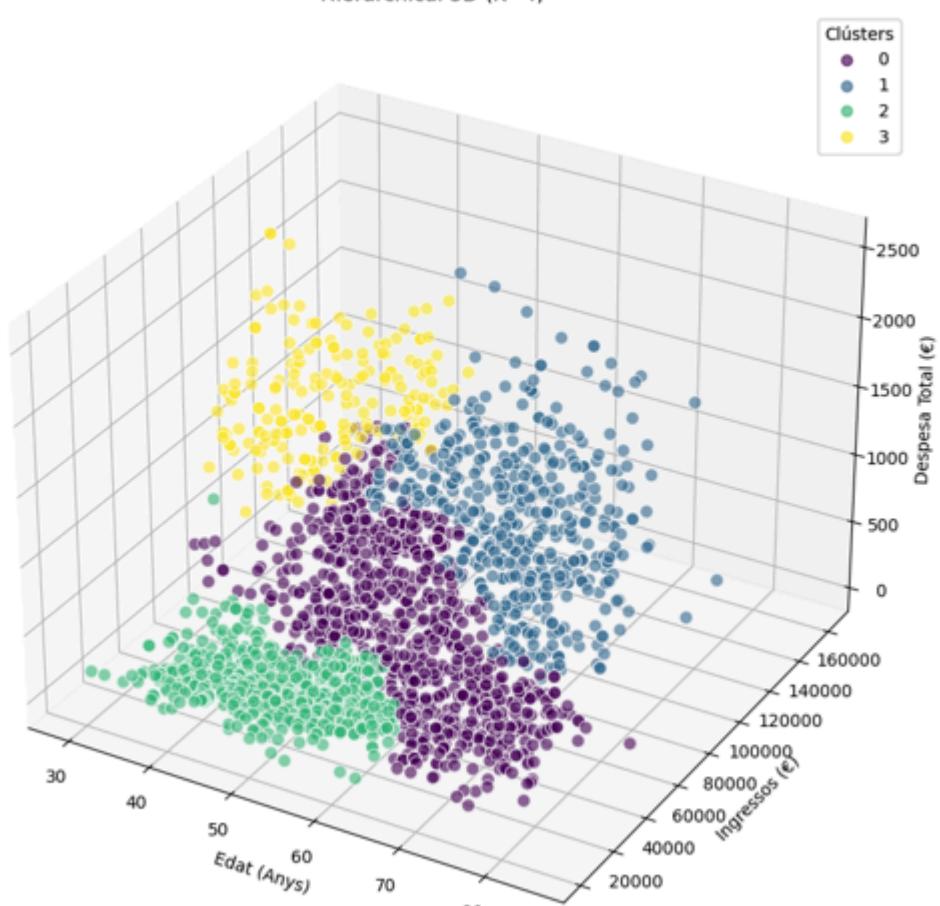
```
--- Preprocessing & Cleaning ---  
--- Cerca de K Òptima ---  
La K òptima seleccionada és: 4  
--- Executant i Guardant Models (K=4) ---  
Processant K-Means...  
-> Gràfic 3D guardat: 01_kmeans_3d.png  
Processant Jeràrquic...  
-> Gràfic 3D guardat: 02_hierarchical_3d.png  
Processant GMM...  
-> Gràfic 3D guardat: 03_gmm_3d.png  
--- Resum de Mètriques ---  
K-Means: SSE(Cohesió)=809.74, BSS(Separació)=8.88, Correlació=-0.5681  
Jeràrquic: SSE(Cohesió)=1032.59, BSS(Separació)=9.28, Correlació=-0.5131  
GMM: SSE(Cohesió)=1404.60, BSS(Separació)=7.34, Correlació=-0.4190  
CSV guardat a: resultats_clustering/marketing_campaign_final.csv  
Processament completat.
```

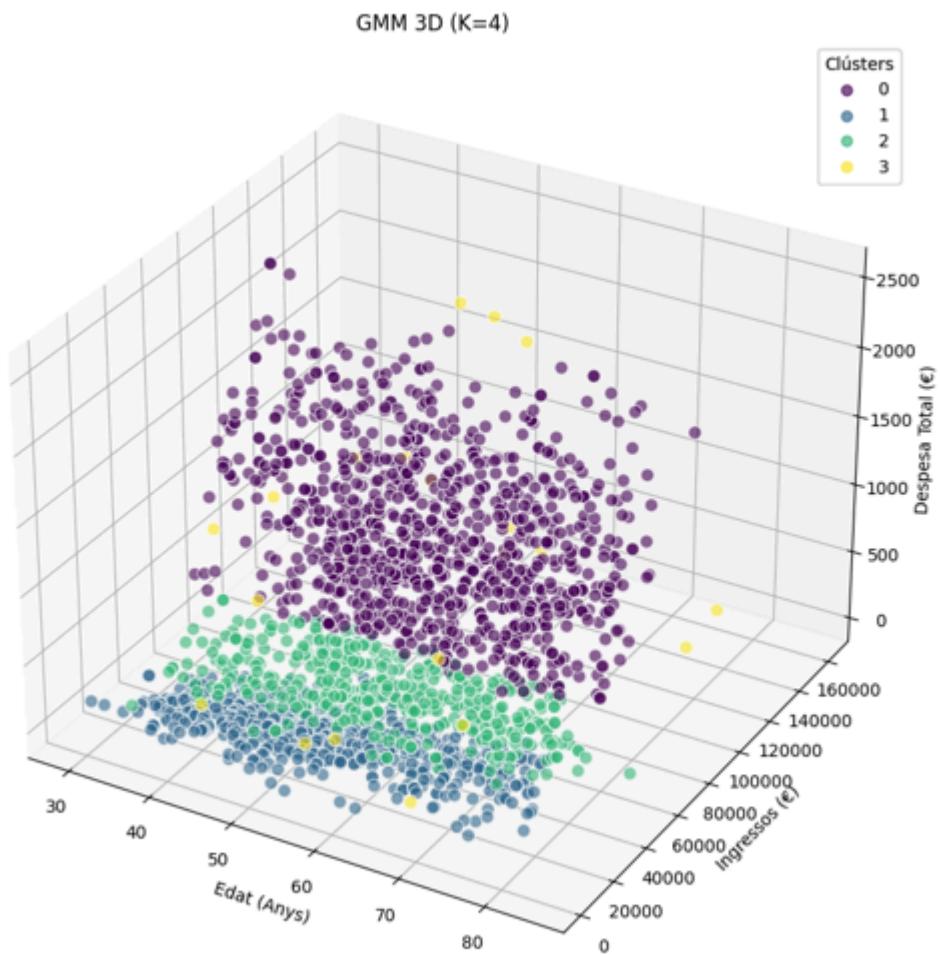
Amb els valors obtinguts de cada mètode podem apreciar com k-means té un SSE menor indicant que és el mètode que genera uns clústers amb major cohesió, la distància entre els punts d'un mateix clúster és més baixa. En canvi veiem com el mètode jeràrquic té una millor separació entre clústers, però no gaire més gran que k-means, per últim La correlació entre la matriu d'incidència i la matriu de proximitat és negativa tant en k-means com en el mètode jeràrquic, la qual cosa suggereix que hi ha una certa desconformitat entre els clusters i la proximitat dels punts dins de la matriu. K-Means sembla ser el millor algorisme per a aquest conjunt de dades, ja que mostra una bona cohesió i una separació raonable. Tot i la correlació negativa, aquest mètode és el més adequat per a la segmentació de clients. Gaussian Mixture Models no és adequat per a aquest conjunt de dades, ja que presenta la cohesió més feble, la separació més baixa i la correlació més negativa.

K-Means 3D (K=4)



Hierarchical 3D (K=4)





Enllaç github

Projecte de github d'on hem extret la idea inicial del codi:

<https://github.com/yuliverseML/Customer-Segmentation-Clustering/tree/main>

Sessió 2:

Automatitzar Mètode del colze

```
# --- 3. TROBAR K ÒPTIMA ---
# Utilitza el mètode del colze per determinar el nombre òptim de clústers
print("\n--- Cerca de K Òptima ---")

# Creem model base de KMeans (sense indicar K)
model = KMeans(random_state=42, n_init=10)

# Visualitzador del colze amb rang de K entre 2 i 10
visualizer = KElbowVisualizer(
    model,
    k=(2,10),
    metric='distortion', # equivalent a inertie
    timings=False
)

# Ajustem el visualitzador a les dades escalades
visualizer.fit(X_scaled)

# Guardem la figura al teu directori de resultats
visualizer.show(outpath=f"{output_folder}/00_metode_colze.png")

# Recuperem la K òptima automàticament
optimal_k = visualizer.elbow_value_

print(f"K Òptima trobada automàticament: {optimal_k}")
```

Creació de variables

Family_Size

```
partner_status = ['Married', 'Together']
data['Has_Partner'] = data['Marital_Status'].apply(lambda x: 1 if x in partner_status else 0)
data['Family_Size'] = 1 + data['Has_Partner'] + data['Kidhome'] + data['Teenhome']
```

Age

```
df['Age'] = 2025 - df['Year_Birth']
```

Total spending

```
df['Total_Spending'] = (df['MntWines'] + df['MntFruits'] +  
df['MntMeatProducts'] + df['MntFishProducts'] +  
df['MntSweetProducts'] + df['MntGoldProds'])
```

Neteja de dades

```
# Neteja Manual (La teva lògica)  
df_a = df_a.dropna(subset=['Income']) # Nuls fora  
df_a = df_a[df_a['Age'] < 100] # Outliers Edat fora  
df_a = df_a[df_a['Income'] < 600000] # Outliers Ingressos fora  
df_a = df_a[~df_a['Marital_Status'].isin(['YOLO', 'Absurd', 'Alone'])] # Status rars fora
```

RESUM FINAL (com a taula)

Escalador	Millor quan...	Evitar quan...	⋮
StandardScaler	Distribució normal, PCA, KMeans	Outliers	
MinMaxScaler	Models sensibles a rangs fixos, RNA	Outliers	
MaxAbsScaler	Dades esparses, molts zeros	Distribucions estranyes	
RobustScaler	Hi ha outliers, dades econòmiques	Vols distribució exacta	
Normalizer	Similaritat coseno, vectors	Dataset numèric normal	
QuantileTransformer	Distribucions rares, outliers	Vols mantenir estructura global	
PowerTransformer	Distribucions skewed	Molts zeros/negatius	

1. StandardScaler

- escala les dades de manera que tinguin mitjana 0 i desviació estàndard 1.
- Per cada variable x, transforma el valor x en

$$\frac{x - \text{mitjana}}{\text{desviació estàndard}}$$

- Per tant, les variables **queden centrades i distribuïdes uniformement**.
 - **Adequat per dades amb distribució normal** (gaussiana).
 - Ideal quan les dades estan **normalitzades** i no tenen *outliers*.
 - Molt útil per **algoritmes basats en distàncies** com **K-Means** i **PCA**, que assumeixen que les dades tenen una distribució semblant.
 - **Sensibilitat als outliers**: els outliers poden **alterar significativament** la mitjana i la desviació estàndard.
-

2. MinMaxScaler

Què fa?

- **MinMaxScaler** escala les dades a un interval definit, normalment **[0, 1]**.
$$\frac{x - \text{min}}{\text{max} - \text{min}}$$
 - Ideal quan es vol que les dades **tinguin un rang fix** (com per a models que requereixen un rang específic).
 - **Evita la distorsió** en el rang de les dades, així que **no es veurà afectat per outliers en la mateixa mesura** que en altres escaladors.
 - No és robust als **outliers**. Si tens un **outlier extrem**, pot comprimir molt les altres dades.
-

3. RobustScaler

Què fa?

- **RobustScaler** utilitza la **mediana** i el **rang interquartílic (IQR)** en lloc de la mitjana i la desviació estàndard.
- Això el fa més **robust als outliers**.
- Per cada variable `xxx`, es calcula la mediana i el IQR i es normalitza de la següent manera:
$$\frac{x - \text{mediana}}{\text{IQR}}$$

- Ideal per a dades amb outliers, ja que utilitza estadístiques robustes.
 - Útil quan les teves dades tenen **distribucions asimètriques** o **valors extrems** (com **ingressos** o **despeses extremes**).
 - Pot **no ser adequat** per a models com **PCA**, que depenen d'una distribució normal per reduir la dimensionalitat.
-

4. PowerTransformer

- **PowerTransformer** transforma les dades per fer-les **gaussianes**.
 - Utilitza transformacions com **Yeo-Johnson** o **Box-Cox** per **normalitzar les dades** i fer-les més **gaussianes** (són útils quan les dades no segueixen una distribució normal).
 - Pot transformar les dades **asimètriques** en dades **gaussianes**, reduint el **skewness**.
 - Ideal per a **dades asimètriques** o **fortament sesgades**.
 - Molt útil per a **modeling probabilístic** com **GMM**, ja que aquest tipus de model assumeix **normalitat** en les dades.
 - **No és tan senzill d'interpretar** com altres transformacions, ja que les dades es transformen per ajustar-se a una distribució normal.
-

Per que PowerTransformer són millors per a GMM?

1. **GMM** és un **model probabilístic** que assumeix que les dades són **gaussianes** (és a dir, segueixen una distribució normal).
2. **PowerTransformer**:
 - **Transforma les dades per ajustar-les a una distribució normal**, el que **millora la qualitat de la modelització amb GMM**.

Rellevància dels atributs

Hem usat 2 metriques per decidir la importància dels atributs, la primera ha sigut la desviació standard i la segona el PCA:

1. Desviació Standard:

--- DESVIACIÓ ESTÀNDAR ---	
Income	21544.431675
Total_Spending	602.967649
Age	11.695828
Family_Size	0.906056
Education_Graduation	0.500091
Marital_Status_Married	0.487557
Marital_Status_Together	0.438161
Education_PhD	0.411833
Marital_Status_Single	0.409627
Education_Master	0.370927
Marital_Status_Widow	0.182467
Education_Basic	0.154599
dtype: float64	

- La desviació estàndard mesura la variabilitat o dispersió d'un conjunt de dades respecte a la seva mitjana. Una desviació estàndard alta significa que els valors de la variable estan molt dispersos, mentre que una desviació estàndard baixa indica que els valors estan més concentrats al voltant de la mitjana. Quan fem clustering, aquestes variabilitats són importants perquè poden influir en quins atributs seran més útils per separar els clústers. Income i Total_Spending seran molt importants, ja que tenen una alta dispersió. Tot i això, per obtenir clústers més significatius, és possible que hagis d'incloure atributs com Age i Family_Size per afegir més profunditat a la segmentació. Les variables com Marital_Status i Education poden ser menys influents per clústers molt diferenciats, però poden ajudar a afinar la segmentació final.

2. PCA

	--- PCA (LOADINGS) ---		
	PC1	PC2	PC3
Age	-0.038533	-0.020271	-0.049728
Total_Spending	0.003166	-0.017082	0.034583
Income	-0.006309	-0.006756	0.008871
Family_Size	-0.046513	0.098117	-0.249197
Education_Basic	-0.018423	-0.009171	0.012100
Education_Graduation	0.807307	0.207180	-0.040006
Education_Master	-0.246308	-0.084253	-0.026980
Education_PhD	-0.472359	-0.098332	0.065621
Marital_Status_Married	-0.203015	0.800382	-0.101682
Marital_Status_Single	0.110640	-0.267293	0.698782
Marital_Status_Together	0.083360	-0.466246	-0.654554
Marital_Status_Widow	-0.004771	-0.020178	0.017218

- Les càrregues de PCA ens indiquen la contribució de cada variable a les components principals (PC1, PC2, PC3). Els valors més alts en una component indiquen que aquesta variable contribueix més a la variància explicada per aquesta component. A partir de l'anàlisi de PCA, podem concloure que les variables més importants per al clustering segons la variància explicada són les següents: Education_Graduation és la variable més influent, ja que contribueix en gran mesura a la separació de les dades en PC1 i PC2. Marital_Status_Married és una altra variable clau, ja que també influeix significativament en PC1 i PC2, Marital_Status_Together i Marital_Status_Single tenen una influència destacada en PC3, la qual cosa indica que les diferències en l'estat civil també poden ser un factor decisiu en la formació dels clústers. Total_Spending i Income poden ser útils per millorar la separació econòmica entre els clústers, però tenen menys importància en la projecció de PCA.

Inits dels metodes de clustering

1. KMeans

```
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
```

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++',  
n_init='auto', max_iter=300, tol=0.0001, verbose=0, random_state=None,  
copy_x=True, algorithm='lloyd') #
```

Paràmetres utilitzats:

- **n_clusters**: nombre de clústers que volem obtenir.
- **random_state**: valor per inicialitzar la llavor aleatòria i garantir resultats reproductibles.
- **n_init**: número de vegades que l'algorisme s'inicialitza amb diferents centroides per triar la millor solució.

Altres paràmetres importants (no presents però habituals):

- **max_iter**: iteracions màximes per al refinament dels centroides.
- **init**: mètode d'inicialització dels centroides (per defecte "k-means++").

Funció: Determina com es seleccionen els punts inicials per als centroides.

'**k-means++**' selecciona els centroides inicials de manera que maximitza la distància entre ells, millorant la convergència i minimitzant la possibilitat de quedar atrapats en un mínim local.

'**random**' selecciona els centroides de manera aleatòria.

- **tol**

Descripció: Tolerància per a la convergència. Si el canvi en la funció de cost (la distància total dels punts als centroides) és menor que tol, el model es considera com a convergit.

Funció: Estableix el límit de canvi que el model ha de complir per aturar les iteracions. Si no hi ha canvis importants entre iteracions (segons el valor de tol), el model para.

Exemple: tol=0.0001 és un valor per defecte que normalment funciona bé per la majoria de casos.

2. Agglomerative Clustering (Jeràrquic)

```
hierarchical = AgglomerativeClustering(n_clusters=optimal_k, linkage='ward')
```

```
class sklearn.cluster.AgglomerativeClustering(n_clusters=2, *,  
metric='euclidean', memory=None, connectivity=None, compute_full_tree='auto',  
linkage='ward', distance_threshold=None, compute_distances=False) [source]
```

Paràmetres utilitzats:

- **n_clusters**: nombre de clústers finals desitjats.
- **linkage**: criteri d'enllaç entre grups.
En el teu cas: '**ward**', que minimitza la variància dins dels clústers.

Altres paràmetres habituals:

- **metric** Descripció: Mètode utilitzat per calcular les distàncies entre mostres durant l'aglomeració.

Funció: El valor per defecte és 'euclidean', però també pots utilitzar altres mètodes com: 'manhattan': Distància Manhattan (útil en certs casos com espais més esparsos).'cosine': Distància coseno (per dades que representen vectors de característiques, com embeddings).'precomputed': Quan ja tens la matriu de distàncies prèviament calculada.

Exemple: El valor 'euclidean' és el més comú per a clústers basats en distància en espais numèrics.

- **distance_threshold**: alternativa a n_clusters quan es vol tallar l'arbre per distància.
- **compute_full_tree**

Descripció: Controla si es calcula l'arbre aglomeratiu complet. Si es posa a '**auto**', es calcularà l'arbre complet només si és necessari.

Funció: Si tens **moltíssimes dades**, potser no volguesss en la construcció de l'arbre, i en aquests casos, pots establir aquest paràmetre a **False** per

reduir el temps de càcul.

Exemple: En general, 'auto' funciona bé per a la majoria de casos.

- **memory**

Descripció: Permet emmagatzemar la **memòria** per a operacions intermitges del clustering, útil en conjunts de dades **molt grans**.

Funció: Si tens un conjunt de dades gran i vols **optimitzar la memòria**, pots establir aquesta opció amb una ruta per emmagatzemar les dades temporals (per exemple, a un disc).

Exemple: Normalment, no cal tocar aquest paràmetre a menys que treballis amb **grans volums de dades**.

3. Gaussian Mixture Model (GMM)

```
gmm = GaussianMixture(n_components=optimal_k, random_state=42)
```

```
class sklearn.mixture.GaussianMixture(n_components=1, *,  
covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1,  
init_params='kmeans', weights_init=None, means_init=None,  
precisions_init=None, random_state=None, warm_start=False, verbose=0,  
verbose_interval=10) [source]
```

Paràmetres utilitzats:

- **n_components**: nombre de components gaussianes (equivalent a nombre de clústers).
- **random_state**: llavor per generar les inicialitzacions repetibles.

Altres paràmetres habituals:

- **covariance_type**: tipus de matriu de covariàncies ('full', 'tied', 'diag', 'spherical').
- **max_iter**: iteracions màximes de l'algorisme EM.
- **n_init**: execucions inicials amb diferents punts de partida.
- **tol**: Tolerància per a la convergència.
Funció: Aquest paràmetre controla quan el model atura l'iteració. Si el canvi en la funció de cost entre dues iteracions consecutives és menor que **tol**, el model es considera convergit
- **reg_covar**: Valor que s'afegeix a la matriu de covariància per evitar que sigui singular.

Funció: Aquest paràmetre ajuda a **regularitzar** les matrius de covariància per evitar problemes numèrics. És especialment útil en casos amb **dades molt correlacionades** o quan les dades tenen un **nombre de mostres limitat**.

- **Init_params:** **Descripció:** Defineix el mètode per inicialitzar els paràmetres dels components.

Funció: El valor per defecte és '**kmeans**', que inicialitza els **mèdis dels clústers** mitjançant el **mètode K-Means**.

- **weights_init, means_init, precisions_init**

Descripció: Permet inicialitzar manualment els **pesos, mitjanes i precisions** de cada component. S'utilitza per ajustar el model amb valors inicials.

Funció: Normalment no cal modificar aquests paràmetres. Són útils quan tens informació prèvia sobre els **clústers**.

4. PCA (Principal Component Analysis)

```
pca = PCA(n_components=2)
```

```
class sklearn.decomposition.PCA(n_components=None, *, copy=True,
whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto',
n_oversamples=10, power_iteration_normalizer='auto', random_state=None)
```

Paràmetres utilitzats:

- **n_components:** nombre de components principals a mantenir.
En aquest cas **2**, per reduir les dades a 2 dimensions.

Altres paràmetres importants (no definits però habituals):

- **svd_solver:** mètode de descomposició (auto, full, arpack, randomized).

- **whiten**: normalitza les components perquè tinguin variància unitària.
- **random_state**: només afecta alguns solvers com randomized.

2. t-SNE (t-Distributed Stochastic Neighbor Embedding)

```
tsne = TSNE(n_components=2, random_state=42)
```

```
class sklearn.manifold.TSNE(n_components=2, *, perplexity=30.0,
early_exaggeration=12.0, learning_rate='auto', max_iter=1000,
n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean',
metric_params=None, init='pca', verbose=0, random_state=None,
method='barnes_hut', angle=0.5, n_jobs=None)
```

Paràmetres utilitzats:

- **n_components**: dimensions de sortida desitjades (en aquest cas, 2D).
- **random_state**: llavor per garantir resultats reproductibles.

Altres paràmetres importants del t-SNE (molt rellevants):

- **perplexity**: controla l'equilibri entre veïns propers i llunyans (valor típic: 5–50).
- **learning_rate**: velocitat d'aprenentatge, afecta la qualitat de l'embedding.
- **n_iter**: número d'iteracions (per defecte 1000).
- **metric**: mètrica de distància (euclidean per defecte).
- **init**: inicialització (pca o random).
- **early_exaggeration**: controla la separació inicial dels punts.