Apendice_1_Resultados_del_Proyecto

February 5, 2024

1 Proyecto final - Capstone - Jupyter Notebook (Google Colab)

1.0.1 *Clusterización y reducción de dimensiones - Satisfacción de pacientes en el IMSS*

- Ejecución del código en Python relativo a la **clusterización y reducción dimensional** de un dataset relacionado con la calidad de la atención brindada a diferentes pacientes del Instituto Mexicano del Seguro Social.
- Se evaluarán diferentes aspectos y se realizará el preprocesado del dataset para poder **arrojar resultados útiles** para la correcta realización de la clusterización y mostrar conclusiones con respecto a la información que nos interesa obtener (**grupos de pacientes** de acuerdo a la calidad entregada, segmentación geográfica, entre otros).
- *Para el caso de Google Colab es necesario **instalar las librerías K-Modes, UMAP Learn y SHAP** mediante pip*

[17]:
```
#pip install kmodes umap-learn shap
```

Código inicial utilizado únicamente para realizar el reseteo total de las variables del notebook

[ ]:
```
%reset -f
```

Se realiza la importación de las librerías respectivas que se utilizarán para trabajar con nuestro dataset y preprocesar y procesar la información.

- **Pandas**: Manejo del DataFrame exportado desde nuestro dataset en formato CSV, así como parte del preprocesado del mismo.
- **NumPy**: Preprocesado del DataFrame de dicho sistema.
- **Matplotlib**: Graficación de los datos generados por el código, necesario para el uso de la librería Seaborn.
- **Seaborn**: Librería de apoyo basada en Matplotlib, que permite una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos.
- **Scikit-learn (sklearn)**: Librería utilizada para Machine Learning, supervisado y no supervisado. En esta ocasión dicha librería se utilizará para realizar la clusterización del modelo así como su reducción dimensional haciendo uso de diferentes algoritmos.
- Necesario para el preprocesado de las variables (DataEncoder, numerificación de las variables), así como para el uso de la librería K-Modes, UMAP Learn y SHAP.
- **K-Modes**: Variación de K-Means, es un algoritmo de clusterización utilizado para agrupar puntos de datos en clusters basados en atributos de categoría (variables categóricas). Por el tipo de variables que agrupa, utiliza las diferencias (desigualdades totales) entre cada punto

de datos, y dependiendo de que tan grandes sean estos, tomará similaridades o diferencias para agrupar centroides.
- Como su nombre lo indica, utilizará **modas** en lugar de medias, para realizar esta agrupación.
- La librería también cuenta con el algoritmo **K-Prototypes**, que también se usará dentro del proyecto y del cual se hablará más abajo.
- **UMAP**: Acrónimo del algoritmo de reducción de dimensiones *Uniform Manifold Approximation and Projection (Aproximación y proyección de colectores uniformes)*. Es una técnica de reducción de dimensiones que se puede utilizar para visualización de manera similar a t-SNE, pero también para reducción de dimensiones no lineal general.
- **LightGBM**: Clasificador utilizado para entrenar los modelos de clusterización generados por los algoritmos respectivos, que permite comprobar la precisión con la que cada modelo de clusterizado se acerca a los datos reales. Es un clasificador basado en el principio del **árbol de decisiones (decision tree)**, y que se considera un mejor algoritmo de clasificación que otros similares, como XGBOOST.
- **SHAP**: (Shapley Additive Explanations), librería de refuerzo que utiliza valores Shapley para comprender la **importancia de cada variable** en la clasificación de nuestros segundos clusters, permitiendonos visualizar la importancia de cada variable dentro de nuestros clusters.

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from kmodes.kmodes import KModes
from kmodes import kprototypes
import umap
from lightgbm import LGBMClassifier
import shap
```

Esta parte solo aplica para su uso en Google Colab, o al montar archivos en la nube de Google Drive. Omitir en caso de utilizar archivos locales en Jupyter Notebook

- Se realiza el **montado del disco en la nube de Google Drive**, para utilizar los datasets que se encuentren guardados en dicha ubicación, guardar los cambios y/o interaccionar con los archivos provenientes de Drive.

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Se utiliza la librería pandas para **crear el DataFrame inicial** (sin procesar) de datos directamente de la información de nuestro archivo CSV, en la ubicación en Google Drive o de un archivo local en nuestro disco.

- El archivo cuenta con 123 columnas de datos de tipo mixto (String, Enteros, Float). Se utiliza la función **head** para mostrar **las primeras 5 filas** del documento como muestra del contenido del DataFrame generado.

Los datos fueron obtenidos de fuentes oficiales, provenientes del Instituto Mexicano del Seguro Social (México), generados y actualizados a **Noviembre de 2022**.

[4]:
```
#Dependiendo del entorno, se cambiará la ruta del dataset
df = pd.read_csv("/content/drive/MyDrive/2022_Nov_ENCal_Nacional_CSV.csv")
df.head()
```

```
<ipython-input-4-18feee75cb01>:2: DtypeWarning: Columns (17,18,22,23,24,25) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("/content/drive/MyDrive/2022_Nov_ENCal_Nacional_CSV.csv")
```

[4]:

	Folio	deleg	unidmed	id_unid	fecha_d	fecha_m	fecha_a	\
0	1	1	UMF 10 Aguascalientes	3	31	10	2022	
1	4	1	UMF 10 Aguascalientes	3	31	10	2022	
2	6	1	UMF 10 Aguascalientes	3	31	10	2022	
3	9	1	UMF 10 Aguascalientes	3	31	10	2022	
4	10	1	UMF 10 Aguascalientes	3	31	10	2022	

	hr_ini_h	hr_ini_m	hr_fin_h	…	id_unidad	Unidad	\
0	8	25	8	…	3	UMF 10 Aguascalientes	
1	8	59	9	…	3	UMF 10 Aguascalientes	
2	8	38	8	…	3	UMF 10 Aguascalientes	
3	8	30	8	…	3	UMF 10 Aguascalientes	
4	8	45	8	…	3	UMF 10 Aguascalientes	

	Cve_Delegación	Delegación	Promedio_Diario2021	estrato	MOSCondia	\
0	1	Aguascalientes	1248	1	0.209115	
1	1	Aguascalientes	1248	1	0.209115	
2	1	Aguascalientes	1248	1	0.209115	
3	1	Aguascalientes	1248	1	0.209115	
4	1	Aguascalientes	1248	1	0.209115	

	FE_FinalNR	fecha	entidad
0	45.431635	31/10/2022	1
1	43.529844	31/10/2022	1
2	39.005015	31/10/2022	1
3	43.529844	31/10/2022	1
4	39.005015	31/10/2022	1

[5 rows x 123 columns]

Se coloca la **información del DataFrame**, identificando los tipos de datos mixtos que se mencionan anteriormente, así como la cantidad de entradas (filas) que se tienen en el mismo, y el tamaño en memoria.

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12575 entries, 0 to 12574
```

```
Columns: 123 entries, Folio to entidad
dtypes: float64(4), int64(58), object(61)
memory usage: 11.8+ MB
```

Como iniciado del preprocesado, se va a trabajar de inicio con las columnas **“deleg”**, **“sat1”**, **“probsal”**, **“cal1”**, **“edad”** y **“sexo”** que cuentan con una codificación numérica (strings) donde cada número representa un valor diferente en la estadística, y será el punto que se usará como base para comparar nuestra hipótesis, de que cada grupo de pacientes presentará características diferentes respecto a la **percepción del servicio de atención que recibieron**, dentro del universo de pacientes del IMSS:

Los padecimientos (probsal) que se estarán cuantificando son los siguientes: - 4 - ***Diabetes mellitus o pie diabético*** - 7 - ***Control, supervisión o seguimiento de personas sanas*** - 6 - ***Traumatismos y envenenamientos*** - 34 - ***Hipertensión*** - 5 - ***Infecciones respiratorias agudas*** - 3 - ***Enfermedades del corazón*** - 1 - ***Embarazos***

Como paso inicial se tomarán para nuestro DataFrame únicamente **los valores de nuestro interés para el estadístico** (aquellos padecimientos mencionados anteriormente, que son los padecimientos más comunes arrojados por nuestro dataset), también se eliminarán de la tabla las **respuestas vacías ” “** (que no permiten el procesado adecuado) y los **”no respondió/sin respuesta” (valor 99, 998, 999)** de las columnas”*probsal*“,”*sat1*” y “*cal1*”, evitando así modificar el estadístico innecesariamente. - Adicionalmente, y para mejorar la precisión de nuestro cluster, se concentrarán las edades de nuestra muestra en categorías, que permitan clasificar de mejor manera estas variables.

Se coloca la función describe para propósitos del EDA (análisis estadístico de datos).

[6]:
```
df["sat1"] = df["sat1"].astype(str)
df["cal1"] = df["cal1"].astype(str)
labels = ["{0} - {1}".format(i, i + 9) for i in range(0, 100, 10)]
df["edad_cat"] = pd.cut(df.edad, range(0, 105, 10), right=False, labels=labels)
serie = ["4","7","6","34","5","3","1"]
df_clean_list = []
df_clean = df[(df.sat1 != "99") & (df.probsal != " ") & (df.probsal != "998") &␣
 ↪(df.probsal != "999") & (df.cal1 != "99")]
for numero in serie:
  df_clean_list.append(df_clean[df_clean.probsal == int(numero)])

df_clean = pd.concat(df_clean_list, ignore_index=True)
df_clean["probsal"] = df_clean["probsal"].astype(str)
pd.set_option('display.max_rows', None)
#print(len(df_clean))
df_clean.describe().T
#df_clean.to_csv("/content/drive/MyDrive/Dataset_IMSS_Nov2022.csv")
```

[6]:

	count	mean	std	min	\
Folio	4846.0	5317.696451	3064.546412	9.000000	
deleg	4846.0	17.373298	9.250886	1.000000	
id_unid	4846.0	911.288898	567.606087	3.000000	
fecha_d	4846.0	9.023318	9.149874	1.000000	

fecha_m	4846.0	10.867107	0.339494	10.000000
fecha_a	4846.0	2022.000000	0.000000	2022.000000
hr_ini_h	4846.0	13.082542	3.303217	8.000000
hr_ini_m	4846.0	29.605241	17.011511	0.000000
hr_fin_h	4846.0	13.240817	3.306121	8.000000
hr_fin_m	4846.0	30.335947	17.244607	0.000000
tipopac	4846.0	1.040239	0.209749	1.000000
sat3	4846.0	3.724515	12.529324	1.000000
filtrosaux1	4846.0	1.947379	5.146112	1.000000
filtrosaux2	4846.0	2.071605	4.753016	1.000000
recmedhoy	4846.0	1.292612	2.838155	1.000000
btratou	4846.0	2.157656	5.790629	1.000000
calfatna_a	4846.0	1.958316	8.749791	1.000000
calfatna_c	4846.0	1.258976	4.224944	1.000000
calfatna_e	4846.0	2.228436	7.780876	1.000000
calfatna_g	4846.0	2.797152	9.424444	1.000000
calfatna_h	4846.0	2.890012	9.514861	1.000000
calfatna_n	4846.0	1.749690	6.584238	1.000000
calfatna_k	4846.0	2.883409	9.618364	1.000000
calfatna_l	4846.0	2.226372	8.270013	1.000000
calfatna_j	4846.0	3.090177	10.560217	1.000000
calfatna_m	4846.0	2.905902	9.615101	1.000000
calfatna_f	4846.0	3.269707	11.164632	1.000000
escuchar	4846.0	1.294057	4.873592	1.000000
responder	4846.0	1.250310	4.450609	1.000000
amable	4846.0	1.353900	5.446012	1.000000
tiemuv	4846.0	74.397235	189.871156	1.000000
atn1fam	4846.0	17.237722	35.625364	1.000000
atnpref	4846.0	6.574494	22.191625	1.000000
atnpref2_a	4846.0	12.854932	27.973166	1.000000
corrup1	4846.0	2.877218	9.202278	1.000000
imagen	4846.0	4.555922	14.425899	1.000000
calfinmb_f	4846.0	2.843582	7.639956	1.000000
limp1	4846.0	5.582130	13.809995	1.000000
recomej_a	4846.0	237.967189	420.691394	1.000000
turno	4846.0	2.425093	9.673219	1.000000
antdhu	4846.0	32.873492	103.820469	0.020000
antusuario	4846.0	30.644806	127.736505	0.020000
tiempollegar	4846.0	33.534461	44.745209	1.000000
filtroaten	4846.0	1.799629	3.454329	1.000000
porbsal_pais	4846.0	16.892076	29.170035	1.000000
contagio_covid	4846.0	2.177672	7.127644	1.000000
vacunados	4846.0	1.102352	2.444987	1.000000
edad	4846.0	52.170863	17.006079	18.000000
sexo	4846.0	1.333058	0.471356	1.000000
escolar	4846.0	5.479777	5.015796	1.000000
ocupa	4846.0	7.496698	49.490480	1.000000

trabajoaño	4846.0	2.412712	8.936429	1.000000
Nivel	4846.0	1.000000	0.000000	1.000000
id_unidad	4846.0	911.288898	567.606087	3.000000
Cve_Delegación	4846.0	17.373298	9.250886	1.000000
Promedio_Diario2021	4846.0	848.004746	528.003295	50.000000
estrato	4846.0	1.569955	0.750890	1.000000
MOSCondia	4846.0	0.219124	0.220180	0.019962
FE_FinalNR	4846.0	48.699059	40.962596	3.385742
entidad	4846.0	16.114527	8.430196	1.000000

	25%	50%	75%	max
Folio	2752.250000	5412.000000	7704.000000	11036.000000
deleg	9.000000	17.000000	25.000000	33.000000
id_unid	395.000000	958.000000	1304.000000	2108.000000
fecha_d	3.000000	7.000000	10.000000	31.000000
fecha_m	11.000000	11.000000	11.000000	11.000000
fecha_a	2022.000000	2022.000000	2022.000000	2022.000000
hr_ini_h	10.000000	13.000000	16.000000	20.000000
hr_ini_m	15.000000	30.000000	44.000000	59.000000
hr_fin_h	10.000000	13.000000	16.000000	20.000000
hr_fin_m	15.000000	31.000000	45.000000	59.000000
tipopac	1.000000	1.000000	1.000000	3.000000
sat3	2.000000	2.000000	2.000000	99.000000
filtrosaux1	1.000000	2.000000	2.000000	99.000000
filtrosaux2	2.000000	2.000000	2.000000	99.000000
recmedhoy	1.000000	1.000000	1.000000	99.000000
btratou	1.000000	2.000000	2.000000	99.000000
calfatna_a	1.000000	1.000000	1.000000	99.000000
calfatna_c	1.000000	1.000000	1.000000	99.000000
calfatna_e	1.000000	2.000000	2.000000	99.000000
calfatna_g	2.000000	2.000000	2.000000	99.000000
calfatna_h	2.000000	2.000000	2.000000	99.000000
calfatna_n	1.000000	1.000000	2.000000	99.000000
calfatna_k	2.000000	2.000000	2.000000	99.000000
calfatna_l	1.000000	2.000000	2.000000	99.000000
calfatna_j	2.000000	2.000000	2.000000	99.000000
calfatna_m	2.000000	2.000000	2.000000	99.000000
calfatna_f	2.000000	2.000000	2.000000	99.000000
escuchar	1.000000	1.000000	1.000000	99.000000
responder	1.000000	1.000000	1.000000	99.000000
amable	1.000000	1.000000	1.000000	99.000000
tiemuv	15.000000	30.000000	30.000000	999.000000
atn1fam	2.000000	2.000000	2.000000	99.000000
atnpref	1.000000	1.000000	2.000000	99.000000
atnpref2_a	4.000000	4.000000	4.000000	99.000000
corrup1	2.000000	2.000000	2.000000	99.000000
imagen	2.000000	2.000000	3.000000	99.000000

```
calfinmb_f              2.000000     2.000000     2.000000     99.000000
limp1                   2.000000     3.000000     6.000000     99.000000
recomej_a               1.000000     4.000000    13.000000    999.000000
turno                   1.000000     1.000000     2.000000     99.000000
antdhu                  8.000000    20.000000    35.000000    999.000000
antusuario              3.000000    10.000000    20.000000    999.000000
tiempollegar           15.000000    30.000000    40.000000    999.000000
filtroaten              1.000000     2.000000     2.000000     99.000000
porbsal_pais            5.000000     5.000000     9.000000     99.000000
contagio_covid          1.000000     2.000000     2.000000     99.000000
vacunados               1.000000     1.000000     1.000000     99.000000
edad                   38.000000    54.000000    65.000000     93.000000
sexo                    1.000000     1.000000     2.000000      2.000000
escolar                 3.000000     5.000000     7.000000     99.000000
ocupa                   2.000000     5.000000     8.000000    999.000000
trabajoaño              1.000000     2.000000     2.000000     99.000000
Nivel                   1.000000     1.000000     1.000000      1.000000
id_unidad             395.000000   958.000000  1304.000000   2108.000000
Cve_Delegación          9.000000    17.000000    25.000000     33.000000
Promedio_Diario2021   413.000000   689.000000  1286.000000   2252.000000
estrato                 1.000000     1.000000     2.000000      4.000000
MOSCondia               0.105073     0.167289     0.231727      1.000000
FE_FinalNR             19.622273    33.626257    61.009658    239.680025
entidad                 9.000000    15.000000    23.000000     30.000000
```

Se muestra la columna "edad_cat" recien creada, con los **rangos de edades** generados por cada fila dentro de nuestro dataset, para principios prácticos de facilitar la clusterización, con el método inicial, que es mediante clusterización categórica.

[7]:
```
df_clean["edad_cat"].head()
```

[7]:
```
0    50 - 59
1    50 - 59
2    40 - 49
3    60 - 69
4    60 - 69
Name: edad_cat, dtype: category
Categories (10, object): ['0 - 9' < '10 - 19' < '20 - 29' < '30 - 39' … '60 -
69' < '70 - 79' <
                          '80 - 89' < '90 - 99']
```

Se muestra también la correlación de la variables numéricas con cada columna, utilizando la **función corr()** de Pandas.

[8]:
```
df_clean.corr()
```

```
<ipython-input-8-798b28326b09>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
```

```
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df_clean.corr()
```

[8]:

	Folio	deleg	id_unid	fecha_d	fecha_m	\
Folio	1.000000	0.816464	0.995390	-0.056396	0.030689	
deleg	0.816464	1.000000	0.781484	-0.002027	-0.020083	
id_unid	0.995390	0.781484	1.000000	-0.060882	0.034050	
fecha_d	-0.056396	-0.002027	-0.060882	1.000000	-0.940387	
fecha_m	0.030689	-0.020083	0.034050	-0.940387	1.000000	
fecha_a	NaN	NaN	NaN	NaN	NaN	
hr_ini_h	0.038200	0.036748	0.030894	-0.038982	0.030765	
hr_ini_m	-0.019590	-0.004164	-0.022589	0.013481	-0.022487	
hr_fin_h	0.038660	0.038334	0.031186	-0.036804	0.027783	
hr_fin_m	-0.021846	-0.015612	-0.023033	0.006074	-0.009894	
tipopac	-0.019664	-0.016146	-0.016869	0.033280	-0.035031	
sat3	-0.073251	-0.091676	-0.073876	0.009719	-0.019769	
filtrosaux1	0.030139	0.026391	0.029790	-0.011187	0.017262	
filtrosaux2	0.009065	0.006090	0.010269	-0.022771	0.025085	
recmedhoy	0.022530	0.029854	0.019882	-0.008934	0.003736	
btratou	0.006031	0.008638	0.006859	0.035898	-0.035326	
calfatna_a	0.031499	0.033892	0.029167	0.029490	-0.041609	
calfatna_c	0.021151	0.030642	0.019965	0.012909	-0.016868	
calfatna_e	0.056285	0.073096	0.046814	-0.014799	0.017902	
calfatna_g	0.076586	0.095540	0.064252	-0.021286	0.034020	
calfatna_h	0.076410	0.096768	0.064148	-0.015615	0.027230	
calfatna_n	0.033927	0.047772	0.029105	-0.002592	0.006260	
calfatna_k	0.066524	0.086339	0.054567	-0.010710	0.021991	
calfatna_l	0.060041	0.077222	0.050228	-0.008697	0.018142	
calfatna_j	0.081770	0.104317	0.068891	-0.022216	0.038231	
calfatna_m	0.075932	0.096176	0.063610	-0.020691	0.033537	
calfatna_f	0.069942	0.089224	0.060048	-0.023873	0.031349	
escuchar	0.014779	0.021746	0.015154	0.026539	-0.030142	
responder	0.040353	0.046858	0.041570	-0.001598	-0.009535	
amable	0.000564	0.004383	0.001460	0.004693	-0.000345	
tiemuv	0.018003	0.011465	0.017019	0.003430	-0.001618	
atn1fam	0.043865	0.062902	0.040713	0.029327	-0.030272	
atnpref	0.037847	0.040401	0.039889	0.028309	-0.023589	
atnpref2_a	0.019866	0.026895	0.020417	0.029939	-0.021156	
corrup1	-0.012097	-0.018021	-0.009750	-0.009472	0.024638	
imagen	0.002138	0.009800	0.002164	0.019029	-0.024106	
calfinmb_f	0.036796	0.050040	0.035357	0.012232	-0.026637	
limp1	-0.028698	-0.019902	-0.023153	0.016720	-0.004055	
recomej_a	0.001940	0.005352	0.006056	-0.018279	0.019551	
turno	0.045699	0.064208	0.039498	-0.011186	-0.009380	
antdhu	-0.008261	0.004491	-0.009800	0.016615	-0.012105	
antusuario	0.000047	0.002015	-0.000298	0.032308	-0.037244	

```
tiempollegar          0.001299 -0.005469  0.002177  0.013240 -0.008680
filtroaten            0.023119  0.002600  0.024685 -0.014421  0.016185
porbsal_pais          0.003667  0.017753  0.004342 -0.009440  0.008514
contagio_covid       -0.008835 -0.000677 -0.010629  0.007678 -0.002949
vacunados             0.008044  0.013814  0.006614 -0.014582  0.010920
edad                  0.000323  0.021154 -0.003968 -0.011795  0.016625
sexo                 -0.016661 -0.034862 -0.013673 -0.013430  0.004501
escolar               0.004277  0.010640  0.005043 -0.030816  0.012603
ocupa                 0.017726  0.026376  0.016223 -0.031856  0.019555
trabajoaño            0.026343  0.043573  0.020520 -0.002003 -0.003076
Nivel                      NaN       NaN       NaN       NaN       NaN
id_unidad             0.995390  0.781484  1.000000 -0.060882  0.034050
Cve_Delegación        0.816464  1.000000  0.781484 -0.002027 -0.020083
Promedio_Diario2021  -0.099782 -0.006744 -0.103468  0.020407 -0.059822
estrato               0.105911  0.074281  0.094402 -0.029092  0.094770
MOSCondia             0.044756  0.094911  0.051454 -0.034246  0.003238
FE_FinalNR           -0.060517 -0.085205 -0.075770 -0.042604  0.033234
entidad               0.805082  0.987788  0.768087 -0.004300 -0.011124

                      fecha_a  hr_ini_h  hr_ini_m  hr_fin_h  hr_fin_m  … \
Folio                     NaN  0.038200 -0.019590  0.038660 -0.021846  …
deleg                     NaN  0.036748 -0.004164  0.038334 -0.015612  …
id_unid                   NaN  0.030894 -0.022589  0.031186 -0.023033  …
fecha_d                   NaN -0.038982  0.013481 -0.036804  0.006074  …
fecha_m                   NaN  0.030765 -0.022487  0.027783 -0.009894  …
fecha_a                   NaN       NaN       NaN       NaN       NaN  …
hr_ini_h                  NaN  1.000000 -0.073935  0.993900 -0.011640  …
hr_ini_m                  NaN -0.073935  1.000000 -0.006144  0.209249  …
hr_fin_h                  NaN  0.993900 -0.006144  1.000000 -0.080745  …
hr_fin_m                  NaN -0.011640  0.209249 -0.080745  1.000000  …
tipopac                   NaN -0.023860  0.004395 -0.024097  0.002710  …
sat3                      NaN -0.034310  0.007377 -0.034512  0.014467  …
filtrosaux1               NaN  0.019270  0.023644  0.021186  0.000695  …
filtrosaux2               NaN  0.008931  0.022545  0.008767  0.021748  …
recmedhoy                 NaN -0.022347 -0.019007 -0.023855  0.001947  …
btratou                   NaN -0.017136  0.025448 -0.016387  0.017863  …
calfatna_a                NaN  0.017658  0.001031  0.018199 -0.005843  …
calfatna_c                NaN  0.000997  0.001707  0.001711 -0.003744  …
calfatna_e                NaN  0.035178 -0.011235  0.034360 -0.001880  …
calfatna_g                NaN  0.034802  0.000622  0.035106  0.003433  …
calfatna_h                NaN  0.032040 -0.002268  0.032329  0.000903  …
calfatna_n                NaN  0.030141  0.004948  0.031357 -0.005767  …
calfatna_k                NaN  0.026022  0.005977  0.027521 -0.004972  …
calfatna_l                NaN  0.045140  0.005502  0.045299  0.011725  …
calfatna_j                NaN  0.039306  0.000195  0.040299 -0.007161  …
calfatna_m                NaN  0.026681 -0.000483  0.026931  0.003608  …
calfatna_f                NaN  0.039177  0.004496  0.040586 -0.008681  …
```

escuchar	NaN	-0.006329	-0.000312	-0.008521	0.026328	…
responder	NaN	-0.002838	-0.006521	-0.002106	-0.015233	…
amable	NaN	-0.005410	-0.007492	-0.006798	0.013204	…
tiemuv	NaN	-0.000395	0.028024	-0.000492	0.023873	…
atn1fam	NaN	0.014850	-0.012904	0.015078	-0.010377	…
atnpref	NaN	0.002281	-0.010957	0.001020	0.006872	…
atnpref2_a	NaN	0.009375	-0.017269	0.006986	0.011934	…
corrup1	NaN	0.013282	0.011077	0.013285	0.008809	…
imagen	NaN	0.026515	0.002405	0.026494	0.003977	…
calfinmb_f	NaN	0.013033	-0.008438	0.012384	0.005644	…
limp1	NaN	0.011498	-0.000639	0.011440	-0.002273	…
recomej_a	NaN	0.043579	-0.001648	0.042121	-0.002925	…
turno	NaN	0.046599	0.020525	0.048087	0.000824	…
antdhu	NaN	0.028905	-0.006393	0.026919	0.011758	…
antusuario	NaN	0.016920	0.008545	0.016119	0.017787	…
tiempollegar	NaN	-0.003138	0.004475	-0.003545	0.007202	…
filtroaten	NaN	-0.003904	0.000062	-0.002190	-0.023096	…
porbsal_pais	NaN	-0.008367	-0.001747	-0.008520	-0.006451	…
contagio_covid	NaN	-0.003051	0.015771	-0.002377	0.008856	…
vacunados	NaN	-0.013824	-0.000378	-0.012267	-0.023158	…
edad	NaN	0.011727	0.001547	0.011771	0.013420	…
sexo	NaN	0.014685	-0.026869	0.013949	-0.019558	…
escolar	NaN	0.009593	0.001332	0.009299	0.001653	…
ocupa	NaN	0.007824	-0.012060	0.008059	-0.012218	…
trabajoaño	NaN	-0.012733	-0.006957	-0.013641	0.005632	…
Nivel	NaN	NaN	NaN	NaN	NaN	…
id_unidad	NaN	0.030894	-0.022589	0.031186	-0.023033	…
Cve_Delegación	NaN	0.036748	-0.004164	0.038334	-0.015612	…
Promedio_Diario2021	NaN	0.055219	0.003958	0.055161	0.001279	…
estrato	NaN	-0.022050	0.001104	-0.021462	-0.011108	…
MOSCondia	NaN	0.057893	-0.015448	0.055304	0.004460	…
FE_FinalNR	NaN	-0.012801	0.011821	-0.010885	-0.003461	…
entidad	NaN	0.035178	-0.004541	0.036430	-0.013913	…

	ocupa	trabajoaño	Nivel	id_unidad	Cve_Delegación	\
Folio	0.017726	0.026343	NaN	0.995390	0.816464	
deleg	0.026376	0.043573	NaN	0.781484	1.000000	
id_unid	0.016223	0.020520	NaN	1.000000	0.781484	
fecha_d	-0.031856	-0.002003	NaN	-0.060882	-0.002027	
fecha_m	0.019555	-0.003076	NaN	0.034050	-0.020083	
fecha_a	NaN	NaN	NaN	NaN	NaN	
hr_ini_h	0.007824	-0.012733	NaN	0.030894	0.036748	
hr_ini_m	-0.012060	-0.006957	NaN	-0.022589	-0.004164	
hr_fin_h	0.008059	-0.013641	NaN	0.031186	0.038334	
hr_fin_m	-0.012218	0.005632	NaN	-0.023033	-0.015612	
tipopac	0.009368	0.012280	NaN	-0.016869	-0.016146	
sat3	-0.008201	-0.010176	NaN	-0.073876	-0.091676	

filtrosaux1	-0.001736	-0.005344	NaN	0.029790	0.026391
filtrosaux2	-0.001930	-0.004559	NaN	0.010269	0.006090
recmedhoy	-0.004445	-0.002321	NaN	0.019882	0.029854
btratou	-0.003157	-0.008170	NaN	0.006859	0.008638
calfatna_a	-0.003942	-0.007493	NaN	0.029167	0.033892
calfatna_c	-0.001471	-0.003805	NaN	0.019965	0.030642
calfatna_e	0.000149	-0.006800	NaN	0.046814	0.073096
calfatna_g	-0.003751	-0.009012	NaN	0.064252	0.095540
calfatna_h	-0.003423	-0.009467	NaN	0.064148	0.096768
calfatna_n	-0.001837	-0.006614	NaN	0.029105	0.047772
calfatna_k	-0.003586	-0.009626	NaN	0.054567	0.086339
calfatna_l	-0.004091	-0.009132	NaN	0.050228	0.077222
calfatna_j	-0.004392	-0.010573	NaN	0.068891	0.104317
calfatna_m	-0.002723	-0.008731	NaN	0.063610	0.096176
calfatna_f	-0.003978	-0.009935	NaN	0.060048	0.089224
escuchar	-0.004154	0.084450	NaN	0.015154	0.021746
responder	0.088904	0.045114	NaN	0.041570	0.046858
amable	-0.003219	-0.004766	NaN	0.001460	0.004383
tiemuv	-0.015265	-0.027382	NaN	0.017019	0.011465
atn1fam	-0.021284	0.017179	NaN	0.040713	0.062902
atnpref	-0.010182	-0.000036	NaN	0.039889	0.040401
atnpref2_a	-0.014495	0.017511	NaN	0.020417	0.026895
corrup1	-0.003055	-0.008763	NaN	-0.009750	-0.018021
imagen	-0.009084	0.031217	NaN	0.002164	0.009800
calfinmb_f	-0.004495	0.019891	NaN	0.035357	0.050040
limp1	-0.006760	0.017936	NaN	-0.023153	-0.019902
recomej_a	0.043064	0.058297	NaN	0.006056	0.005352
turno	0.038602	-0.008343	NaN	0.039498	0.064208
antdhu	-0.005764	0.016985	NaN	-0.009800	0.004491
antusuario	-0.006679	0.007777	NaN	-0.000298	0.002015
tiempollegar	0.029039	-0.005476	NaN	0.002177	-0.005469
filtroaten	0.000027	-0.002870	NaN	0.024685	0.002600
porbsal_pais	-0.007470	0.045895	NaN	0.004342	0.017753
contagio_covid	-0.005293	0.027254	NaN	-0.010629	-0.000677
vacunados	0.002364	-0.000989	NaN	0.006614	0.013814
edad	0.002843	0.037115	NaN	-0.003968	0.021154
sexo	-0.000094	-0.030337	NaN	-0.013673	-0.034862
escolar	0.154838	0.018849	NaN	0.005043	0.010640
ocupa	1.000000	0.044101	NaN	0.016223	0.026376
trabajoaño	0.044101	1.000000	NaN	0.020520	0.043573
Nivel	NaN	NaN	NaN	NaN	NaN
id_unidad	0.016223	0.020520	NaN	1.000000	0.781484
Cve_Delegación	0.026376	0.043573	NaN	0.781484	1.000000
Promedio_Diario2021	-0.009967	-0.008720	NaN	-0.103468	-0.006744
estrato	0.005727	0.010368	NaN	0.094402	0.074281
MOSCondia	0.015969	-0.024249	NaN	0.051454	0.094911
FE_FinalNR	-0.028906	0.009383	NaN	-0.075770	-0.085205

```
entidad             0.024742    0.041791    NaN   0.768087        0.987788

                  Promedio_Diario2021   estrato  MOSCondia  FE_FinalNR  \
Folio                       -0.099782  0.105911   0.044756   -0.060517
deleg                       -0.006744  0.074281   0.094911   -0.085205
id_unid                     -0.103468  0.094402   0.051454   -0.075770
fecha_d                      0.020407 -0.029092  -0.034246   -0.042604
fecha_m                     -0.059822  0.094770   0.003238    0.033234
fecha_a                           NaN       NaN        NaN         NaN
hr_ini_h                     0.055219 -0.022050   0.057893   -0.012801
hr_ini_m                     0.003958  0.001104  -0.015448    0.011821
hr_fin_h                     0.055161 -0.021462   0.055304   -0.010885
hr_fin_m                     0.001279 -0.011108   0.004460   -0.003461
tipopac                     -0.006519 -0.023774   0.006704    0.003405
sat3                        -0.052063 -0.039886  -0.056553   -0.017894
filtrosaux1                 -0.018864  0.017271   0.006905    0.000982
filtrosaux2                 -0.007339 -0.008083   0.008265   -0.016749
recmedhoy                   -0.020349  0.024387  -0.000375   -0.017654
btratou                     -0.014887 -0.010749   0.003540    0.003632
calfatna_a                  -0.021314  0.006947  -0.017342   -0.001069
calfatna_c                   0.011657 -0.014202   0.019706   -0.002585
calfatna_e                  -0.023968  0.027733  -0.024546    0.026684
calfatna_g                  -0.032796  0.036757  -0.034628    0.035882
calfatna_h                  -0.035188  0.032782  -0.042961    0.035454
calfatna_n                   0.001182 -0.006498   0.025272    0.003267
calfatna_k                  -0.031743  0.029864  -0.045065    0.040465
calfatna_l                  -0.024099  0.029008  -0.023290    0.030780
calfatna_j                  -0.006776  0.015590  -0.042531    0.038497
calfatna_m                  -0.030444  0.040220  -0.035924    0.039147
calfatna_f                  -0.036291  0.038384  -0.004459    0.014773
escuchar                    -0.022486 -0.004917  -0.016604    0.001564
responder                   -0.016802 -0.021329  -0.017740   -0.018293
amable                      -0.016168  0.012241  -0.020849   -0.002275
tiemuv                      -0.029793  0.028130   0.036643   -0.013113
atn1fam                      0.000562 -0.004927  -0.007978   -0.027846
atnpref                     -0.021409 -0.019159  -0.015850   -0.033238
atnpref2_a                  -0.001947  0.001294   0.010738   -0.038300
corrup1                     -0.034431  0.008606   0.004205   -0.021494
imagen                      -0.000929 -0.012451   0.002186   -0.030895
calfinmb_f                   0.016568 -0.001654  -0.020635    0.030645
limp1                       -0.017154  0.058023  -0.010859   -0.006149
recomej_a                   -0.028431  0.021705   0.017375   -0.031035
turno                        0.032296 -0.020889  -0.003887    0.002563
antdhu                       0.013702  0.003114  -0.004486    0.000410
antusuario                  -0.003095  0.027115  -0.010999    0.003142
tiempollegar                 0.029037 -0.059441   0.016974   -0.044588
filtroaten                  -0.021529  0.023428  -0.010239   -0.014015
```

porbsal_pais	0.000553	-0.019976	-0.020120	-0.004432
contagio_covid	-0.006361	0.032481	-0.028322	0.019289
vacunados	-0.004050	0.004306	-0.006495	-0.013587
edad	0.030177	-0.010715	0.045130	-0.101792
sexo	-0.021264	0.003553	-0.020730	0.028880
escolar	0.003461	-0.005543	0.001128	0.012158
ocupa	-0.009967	0.005727	0.015969	-0.028906
trabajoaño	-0.008720	0.010368	-0.024249	0.009383
Nivel	NaN	NaN	NaN	NaN
id_unidad	-0.103468	0.094402	0.051454	-0.075770
Cve_Delegación	-0.006744	0.074281	0.094911	-0.085205
Promedio_Diario2021	1.000000	-0.584384	0.406992	0.121846
estrato	-0.584384	1.000000	-0.210534	0.176162
MOSCondia	0.406992	-0.210534	1.000000	-0.356032
FE_FinalNR	0.121846	0.176162	-0.356032	1.000000
entidad	0.008580	0.054539	0.072020	-0.076907

	entidad
Folio	0.805082
deleg	0.987788
id_unid	0.768087
fecha_d	-0.004300
fecha_m	-0.011124
fecha_a	NaN
hr_ini_h	0.035178
hr_ini_m	-0.004541
hr_fin_h	0.036430
hr_fin_m	-0.013913
tipopac	-0.018015
sat3	-0.072063
filtrosaux1	0.026116
filtrosaux2	0.006378
recmedhoy	0.028162
btratou	0.009249
calfatna_a	0.031508
calfatna_c	0.030599
calfatna_e	0.073678
calfatna_g	0.096553
calfatna_h	0.097952
calfatna_n	0.047499
calfatna_k	0.087347
calfatna_l	0.078045
calfatna_j	0.104768
calfatna_m	0.097212
calfatna_f	0.088406
escuchar	0.022329
responder	0.046298

```
amable                0.007897
tiemuv                0.010228
atn1fam               0.064799
atnpref               0.036009
atnpref2_a            0.019067
corrup1              -0.021468
imagen                0.011783
calfinmb_f            0.050334
limp1                -0.027300
recomej_a            -0.005723
turno                 0.065782
antdhu                0.003961
antusuario            0.000129
tiempollegar         -0.002271
filtroaten            0.002184
porbsal_pais          0.008029
contagio_covid        0.000664
vacunados             0.011448
edad                  0.021411
sexo                 -0.031469
escolar               0.009678
ocupa                 0.024742
trabajoaño            0.041791
Nivel                      NaN
id_unidad             0.768087
Cve_Delegación        0.987788
Promedio_Diario2021   0.008580
estrato               0.054539
MOSCondia             0.072020
FE_FinalNR           -0.076907
entidad               1.000000

[60 rows x 60 columns]
```

Como parte del EDA, se realizará la graficación de las variables de nuestro interés, generando varios histogramas con diversas frecuencias de nuestro interés:

- *Gráfica 1:* **probsal** - Histograma de Enfermedades evaluadas por frecuencia
- 4 - ***Diabetes mellitus o pie diabético***
- 7 - ***Control, supervisión o seguimiento de personas sanas***
- 6 - ***Traumatismos y envenenamientos***
- 34 - ***Hipertensión***
- 5 - ***Infecciones respiratorias agudas***
- 3 - ***Enfermedades del corazón***

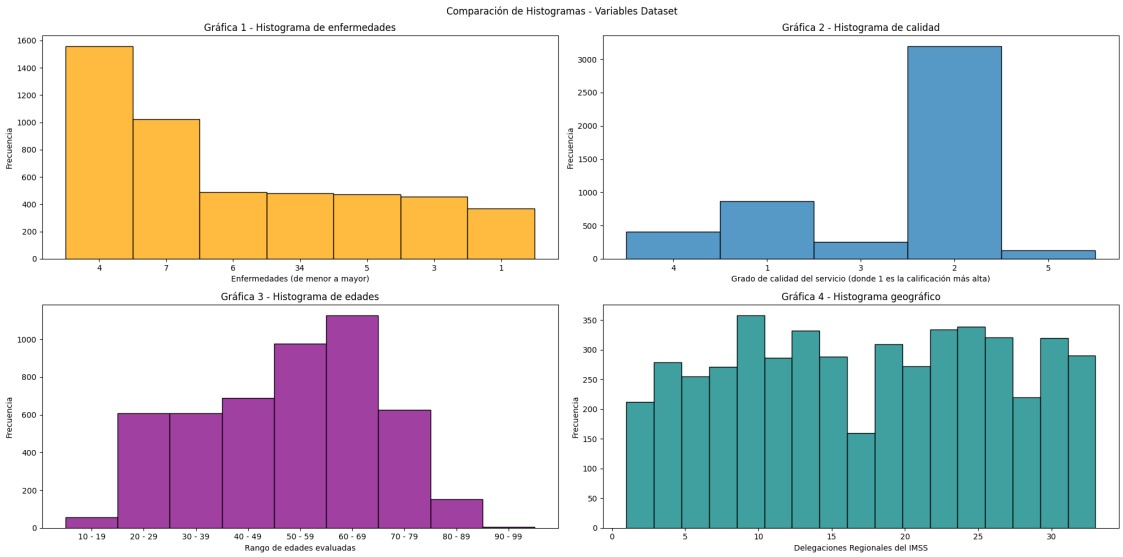
- 1 - ***Embarazos***
- *Gráfica 2:* **sat1**: Histograma de Calidad
- Escala de calificación, donde 1 es la **mejor calificación** y 5 es la **peor calificación**.
- *Gráfica 3:* **edad_cat**: Histograma de Edades
- Rango de edades **de los 0 a los 99 años**.
- *Gráfica 4:* **deleg**: Histograma Geográfico (por delegación regional), dividido en **una de las 35 delegaciones geográficas** que maneja el IMSS a nivel México.
- 01 Aguascalientes
- 02 Baja California
- 03 Baja California Sur
- 04 Campeche
- 05 Chiapas
- 06 Chihuahua
- 07 Coahuila
- 08 Colima
- 09 D.F. Norte
- 10 D.F. Sur
- 11 Durango
- 12 México Oriente
- 13 México Poniente
- 14 Guanajuato
- 15 Guerrero
- 16 Hidalgo
- 17 Jalisco
- 18 Michoacán
- 19 Morelos
- 20 Nayarit
- 21 Nuevo León
- 22 Oaxaca
- 23 Puebla
- 24 Querétaro
- 25 Quintana Roo

- 26 San Luis Potosí
- 27 Sinaloa
- 28 Sonora
- 29 Tabasco
- 30 Tamaulipas
- 31 Tlaxcala
- 32 Veracruz Norte
- 33 Veracruz Sur
- 34 Yucatán
- 35 Zacatecas

Cada histograma nos dará **una primicia inicial** de los grupos que se podrán clusterizar y las agrupaciones que podrían surgir de esta información.

[ ]:
```
X2 = df_clean["sat1"].astype('str')

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(20,10))
axes = axes.flatten()

sns.histplot(df_clean["probsal"], color="orange", ax=axes[0], kde=False)
axes[0].set_title("Gráfica 1 - Histograma de enfermedades")
axes[0].set(xlabel='Enfermedades (de menor a mayor)')
axes[0].set(ylabel="Frecuencia")
axes[0].set_xticks(range(0,7));

sns.histplot(X2, ax=axes[1], kde=False)
axes[1].set_title("Gráfica 2 - Histograma de calidad")
axes[1].set(xlabel='Grado de calidad del servicio (donde 1 es la calificación␣
 ↪más alta)')
axes[1].set(ylabel="Frecuencia")

sns.histplot(df_clean["edad_cat"], color="purple", ax=axes[2], kde=False)
axes[2].set_title("Gráfica 3 - Histograma de edades")
axes[2].set(xlabel='Rango de edades evaluadas')
axes[2].set(ylabel="Frecuencia")

sns.histplot(df_clean["deleg"], color="teal", ax=axes[3], kde=False)
axes[3].set_title("Gráfica 4 - Histograma geográfico")
axes[3].set(xlabel='Delegaciones Regionales del IMSS')
axes[3].set(ylabel="Frecuencia")

fig.suptitle("Comparación de Histogramas - Variables Dataset")
plt.tight_layout()
```

```
plt.show()
```



1.1 *Método 1: Clusterización por K-Modes*

El primer algoritmo de clusterización que probaremos será K-Modes, que está directamente relacionado con la posibilidad de **clusterizar variables categóricas**, que son las que vienen incluídas en la mayor parte de nuestro dataset.

- Una vez realizado el primer preprocesado del DataFrame y el EDA, procedemos a **realizar un preprocesado final de los datos**, con miras a realizar una codificación de los datos en variables numéricas.
- Se separarán **las variables de nuestro interés** para realizar la clusterización, de las columnas del DataFrame original (X_clean)
- Se crea una copia exacta original del DataFrame preprocesado, para poder trabajar con el mismo posteriormente.
- Se utilizará el LabelEncoder de Scikit-learn para poder **codificar las variables categóricas en números**, de tal forma que se pueda trabajar con este tipo de variables y se mejore el resultado de la clusterización.

[ ]:
```
df_clean["sat1"] = df_clean["sat1"].astype(int)
df_clean["cal1"] = df_clean["cal1"].astype(int)
X_clean = df_clean[["deleg","sat1","probsal","cal1","edad_cat","sexo"]]
X_clean_copy = X_clean.copy()
le = preprocessing.LabelEncoder()
X_clean = X_clean.apply(le.fit_transform)
X_clean.head()
```

[ ]:

	deleg	sat1	probsal	cal1	edad_cat	sexo
0	0	3	3	1	4	0

1	0	0	3	0	4	0
2	0	2	3	1	3	0
3	0	1	3	1	5	0
4	0	1	3	1	5	0

Posteriormente, se llevará a cabo la aplicación del **algoritmo de agrupamiento K-Modes** para diferentes valores de k (número de clusters) con el fin de **determinar el número óptimo de clusters** de nuestro conjunto de datos representado por la matriz X_clean. - Se utiliza un bucle "for" para iterar sobre varios valores de k (desde 2 hasta 9), y en cada iteración, se ajusta el modelo K-Modes con el **método de inicialización "Cao"**. La métrica de costo (cost_) asociada a cada valor de k se registra en la lista "cost". - Posteriormente, se visualiza el **"método del codo" (Elbow method)** en la *Gráfica 5*, que muestra el número de clusters en el eje x y el costo asociado en el eje y. Esta gráfica facilita la identificación del punto de inflexión, representando **el número óptimo de clusters**.

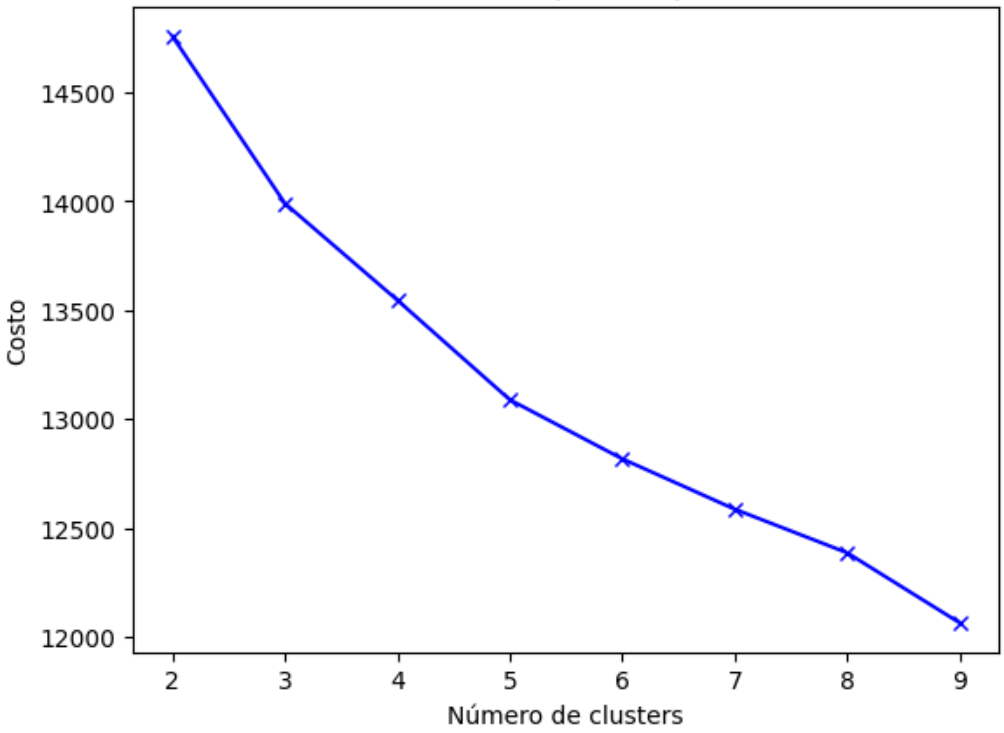
En este contexto, el código proporciona una herramienta para la toma de decisiones sobre el **número adecuado de clusters en nuestro conjunto de datos** ya preprocesados mediante el método K-Modes, permitiendo una comprensión visual de la relación entre el número de clusters y la variación del costo asociado.

[ ]:
```
cost = []
K = range(2,10)
for num_clusters in list(K):
  kmode = KModes(n_clusters = num_clusters, init = "Cao", n_init = 1, verbose =␣
 ↪1)
  kmode.fit_predict(X_clean)
  cost.append(kmode.cost_)

plt.plot(K,cost,'bx-')
plt.xlabel('Número de clusters')
plt.ylabel("Costo")
plt.title("Gráfica 5: Método del codo (Elbow) para k óptima, usando K-Modes -␣
 ↪Cao")
plt.show()
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 14756.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 13990.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 13546.0
Init: initializing centroids
```

```
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 13089.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 12818.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 12587.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 12387.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 12066.0
```

Gráfica 5: Método del codo (Elbow) para k óptima, usando K-Modes - Cao



Se realiza el mismo proceso, pero **utilizando la inicialización "Huang"** como comparativa de ambos métodos, donde cada una difiere de la elección de centroides iniciales. El resultado se plasma en nuestra *Gráfica 6*.

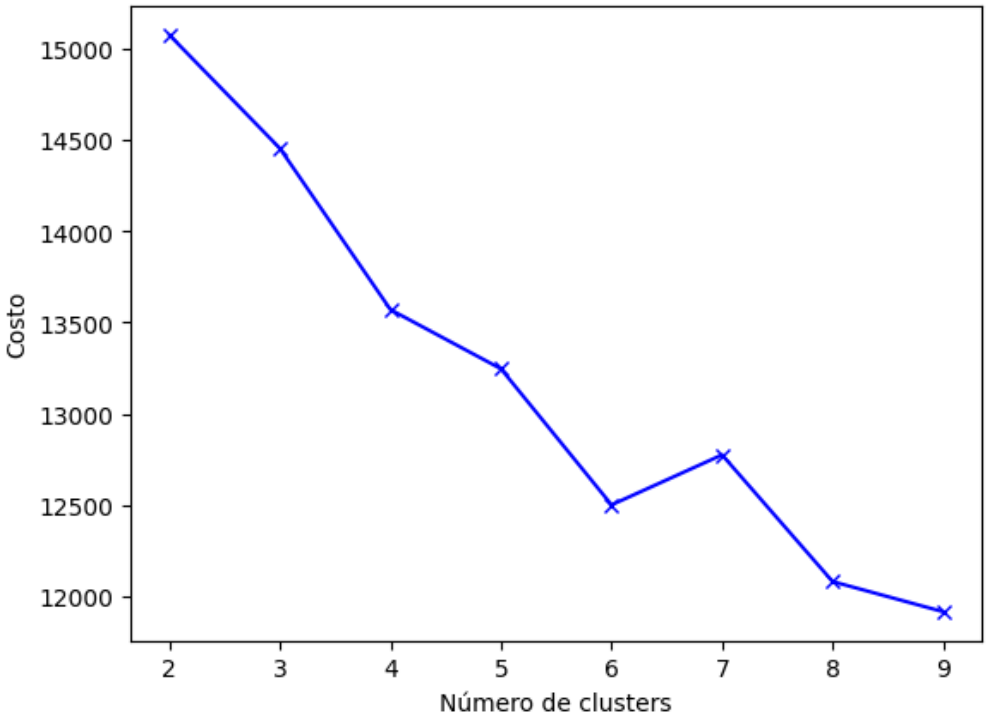
- **Cao (primer inicialización):** Utiliza un método basado en el análisis de frecuencias de los datos categóricos. Calcula la moda de cada característica categórica en el conjunto de datos y utiliza estas modas como centroides iniciales.
- **Huang (segunda inicialización):** Se basa en una métrica de similitud entre las instancias de datos. Selecciona el primer centroide de forma aleatoria y luego elige los centroides restantes de acuerdo con su distancia y similitud con los datos existentes.

[ ]:
```
cost = []
K = range(2,10)
for num_clusters in list(K):
  kmode = KModes(n_clusters = num_clusters, init = "Huang", n_init = 1, verbose␣
 ↪= 1)
  kmode.fit_predict(X_clean)
  cost.append(kmode.cost_)

plt.plot(K,cost,'bx-')
plt.xlabel('Número de clusters')
plt.ylabel("Costo")
plt.title("Gráfica 6: Método del codo (Elbow) para k óptima, usando K-Modes -␣
 ↪Huang")
plt.show()
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 1084, cost: 15075.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 2159, cost: 14454.0
Run 1, iteration: 2/100, moves: 32, cost: 14454.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 1578, cost: 13571.0
Run 1, iteration: 2/100, moves: 105, cost: 13571.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 582, cost: 13247.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 1928, cost: 12502.0
Run 1, iteration: 2/100, moves: 246, cost: 12502.0
Init: initializing centroids
Init: initializing clusters
```

```
Starting iterations…
Run 1, iteration: 1/100, moves: 583, cost: 12776.0
Run 1, iteration: 2/100, moves: 19, cost: 12776.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 875, cost: 12082.0
Run 1, iteration: 2/100, moves: 410, cost: 12082.0
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 528, cost: 11917.0
```

Gráfica 6: Método del codo (Elbow) para k óptima, usando K-Modes - Huang



Como podemos ver, al utilizar la inicialización "Huang", el resultado del método del codo **no nos permite arrojar un resultado concluyente** de que k sería nuestra K ideal, por lo que para nuestros propósitos se utilizará Cao como inicialización, con k=5, de acuerdo con lo arrojado por el método del codo (aunque de acuerdo a la muestra, también podría seleccionarse k=4).

Se realizará una **clusterización inicial** que se mostrará en la variable "fitClusters_cao" utilizando las características anteriormente señaladas.

- Se ajustará nuestro conjunto de datos al modelo de clustering, y se obtienen las **asignaciones de cluster** para cada observación a través de la función fit_predict.

[ ]:
```
km_cao = KModes(n_clusters=5, init="Cao", n_init=1, verbose=1)
fitClusters_cao = km_cao.fit_predict(X_clean)
#fitClusters_cao
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 13089.0
```

Posteriormente, los centroides de nuestros clusters resultantes se almacenan en un DataFrame llamado clusterCentroidsDf, donde cada columna representa **una característica de los datos originales**. Este DataFrame proporciona una visión consolidada de los centroides de los clusters, lo que puede ser útil para interpretar y analizar las **características clave de cada cluster** en el contexto de nuestro proyecto final.

[ ]:
```
clusterCentroidsDf = pd.DataFrame(km_cao.cluster_centroids_)
clusterCentroidsDf.columns = X_clean.columns
clusterCentroidsDf
```

[ ]:

	deleg	sat1	probsal	cal1	edad_cat	sexo
0	12	1	3	1	5	0
1	8	1	6	1	4	1
2	7	0	6	1	3	0
3	23	1	6	0	6	0
4	30	1	6	1	2	0

Como paso siguiente, se realiza la preparación y combinación de datos después de aplicar nuestro algoritmo 1 de clustering al conjunto de datos.

- Primero, **se reinicia el índice del DataFrame X_clean_copy** y se almacena en X_clean, eliminando el índice original.
- Luego, se genera un DataFrame clustersDf que contiene **las asignaciones de clusters resultantes** del proceso de clustering, etiquetadas como "cluster_predicted".
- Posteriormente, ambos DataFrames, X_clean y clustersDf, se combinan a lo largo de las columnas para **formar un nuevo DataFrame llamado combinedDf**
- Finalmente, **se restablece el índice del DataFrame resultante**

Este fragmento de nuestro código proporciona una estructura organizada para analizar y visualizar nuestros datos junto con las asignaciones de clusters respectivas generadas por K-Modes.

[ ]:
```
X_clean = X_clean_copy.reset_index(drop=True)
clustersDf = pd.DataFrame(fitClusters_cao)
clustersDf.columns = ["cluster_predicted"]
combinedDf = pd.concat([X_clean,clustersDf],axis=1).reset_index(drop=True)
#combinedDf = combinedDf.drop(["index","level_0"],axis = 1)
```

Se muestra el DataFrame resultante con la función combinedDf.head()

[ ]:
```
combinedDf.head()
```

[ ]:

	deleg	sat1	probsal	cal1	edad_cat	sexo	cluster_predicted
0	1	4	4	2	50 - 59	1	0
1	1	1	4	1	50 - 59	1	0
2	1	3	4	2	40 - 49	1	0
3	1	2	4	2	60 - 69	1	0
4	1	2	4	2	60 - 69	1	0

Se separa cada cluster obtenido para realizar una pequeña EDA con la función .info() y obtener las muestras de cada uno de estos, dependiendo el tamaño de las muestras obtenidas.

[ ]:
```
cluster0 = combinedDf[combinedDf["cluster_predicted"] == 0]
cluster1 = combinedDf[combinedDf["cluster_predicted"] == 1]
cluster2 = combinedDf[combinedDf["cluster_predicted"] == 2]
cluster3 = combinedDf[combinedDf["cluster_predicted"] == 3]
cluster4 = combinedDf[combinedDf["cluster_predicted"] == 4]
```

- **Cluster 0**: El cluster más grande de los 5 clasificados, abarca en su mayoría (como se puede ver en las gráficas) pacientes diabéticos (**probsal = 4**), y sectores poblacionales con **mayor cantidad de derechohabientes del IMSS** (D.F. y el centro de México). Abarcan también rangos de edad más altos (tercera edad) y pacientes del sexo femenino.

[ ]:
```
cluster0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2467 entries, 0 to 4845
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   deleg              2467 non-null   int64
 1   sat1               2467 non-null   int64
 2   probsal            2467 non-null   object
 3   cal1               2467 non-null   int64
 4   edad_cat           2467 non-null   category
 5   sexo               2467 non-null   int64
 6   cluster_predicted  2467 non-null   uint16
dtypes: category(1), int64(4), object(1), uint16(1)
memory usage: 123.2+ KB
```

- **Cluster 1**: El segundo más grande de los 5 clusters generados, abarca también pacientes diabéticos (**probsal = 4**) y pacientes sanos (**probsal = 7**). Su rango de edades se encuentra, en su mayoría, entre los 50 y 59 años, y su mayor delegación es la D.F. Norte (**deleg = 9**). Es el único cluster donde **los hombres son mayoría**.

[ ]:
```
cluster1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1081 entries, 15 to 4839
```

```
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   deleg              1081 non-null   int64
 1   sat1               1081 non-null   int64
 2   probsal            1081 non-null   object
 3   cal1               1081 non-null   int64
 4   edad_cat           1081 non-null   category
 5   sexo               1081 non-null   int64
 6   cluster_predicted  1081 non-null   uint16
dtypes: category(1), int64(4), object(1), uint16(1)
memory usage: 54.2+ KB
```

- **Cluster 2**: Uno de los tres clusters más pequeños de nuestra muestra. Cuenta con pacientes mujeres en su mayoría, con una alta proporción de pacientes del estado de Colima (**deleg = 8**), y con un rango de edad entre 40 y 49 años. Pacientes sanos (**probsal = 7**) en su mayoría, con pequeñas proporciones de pacientes con enfermedad crónica o infecciosa. Son el cluster que mejor atención refleja en nuestro dataset (**sat1 = 1**).

[ ]:
```
cluster2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 533 entries, 11 to 4827
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   deleg              533 non-null    int64
 1   sat1               533 non-null    int64
 2   probsal            533 non-null    object
 3   cal1               533 non-null    int64
 4   edad_cat           533 non-null    category
 5   sexo               533 non-null    int64
 6   cluster_predicted  533 non-null    uint16
dtypes: category(1), int64(4), object(1), uint16(1)
memory usage: 26.9+ KB
```

- **Cluster 3**: El cluster más pequeño de toda la muestra. Cuenta con pacientes mujeres en su mayoría, con una alta proporción de pacientes del estado de Querétaro (**deleg = 24**), y con un rango de edad entre 70 y 79 años, con un porcentaje menor de personas entre 20 y 29 años. Pacientes sanos (**probsal = 7**) en gran mayoría, con muy pequeña población con enfermedades.

[ ]:
```
cluster3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 368 entries, 9 to 4838
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   deleg              368 non-null    int64
 1   sat1               368 non-null    int64
 2   probsal            368 non-null    object
 3   cal1               368 non-null    int64
 4   edad_cat           368 non-null    category
 5   sexo               368 non-null    int64
 6   cluster_predicted  368 non-null    uint16
dtypes: category(1), int64(4), object(1), uint16(1)
memory usage: 18.7+ KB
```

- **Cluster 4**: El último cluster generado por el algoritmo K-Modes. Cuenta con pacientes mujeres prácticamente en su totalidad, así como una totalidad de pacientes entre 30 y 39 años, con una pequeña proporción de pacientes entre 20 y 29 años. Una gran parte de su muestra proviene del estado de Tlaxcala (**deleg = 31**). La muestra poblacional son prácticamente en su totalidad mujeres embarazadas (**probsal = 1**) y mujeres sanas (**probsal = 7**).

[ ]:
```
cluster4.info()
```

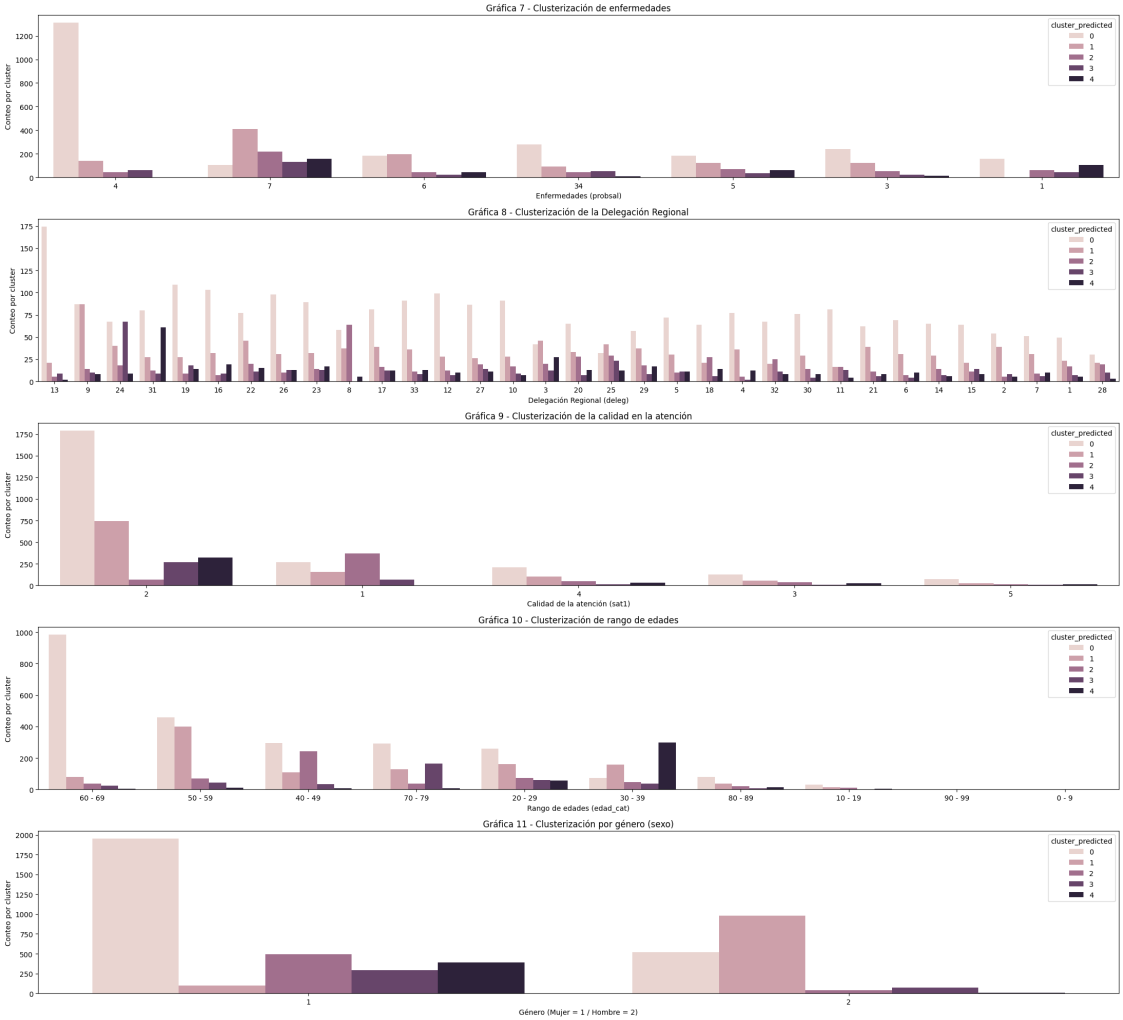
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397 entries, 1395 to 4840
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   deleg              397 non-null    int64
 1   sat1               397 non-null    int64
 2   probsal            397 non-null    object
 3   cal1               397 non-null    int64
 4   edad_cat           397 non-null    category
 5   sexo               397 non-null    int64
 6   cluster_predicted  397 non-null    uint16
dtypes: category(1), int64(4), object(1), uint16(1)
memory usage: 20.1+ KB
```

Para poder graficar nuestra clusterización 1, utilizamos las librerías **Matplotlib y Seaborn** para crear un conjunto de cinco gráficos de barras, cada uno representando la distribución de las variables "probsal" (*Gráfica 7*), "deleg" (*Gráfica 8*), "sat1" (*Gráfica 9*), "edad_cat" (*Gráfica 10*) y "sexo" (*Gráfica 11*). Cada gráfico de barras muestra **la frecuencia de las categorías de estas variables**, ordenadas según su frecuencia descendente. Los colores de las barras en cada gráfico están codificados por las asignaciones de clusters obtenidas previamente.

- La visualización proporciona una comparación clara de cómo las categorías de cada variable **se distribuyen en los diferentes clusters identificados** durante el análisis de clustering, lo que puede ayudar a identificar patrones y tendencias dentro de nuestro Dataset.

[ ]:
```
fig2, axes2 = plt.subplots(nrows=5, figsize=(22,20))
axes2 = axes2.flatten()

sns.countplot(x=combinedDf["probsal"],order=combinedDf["probsal"].
 ↪value_counts().index,hue=combinedDf['cluster_predicted'],ax=axes2[0])
```

```
axes2[0].set_title("Gráfica 7 - Clusterización de enfermedades")
axes2[0].set(xlabel='Enfermedades (probsal)')
axes2[0].set(ylabel="Conteo por cluster")
sns.countplot(x=combinedDf["deleg"],order=combinedDf["deleg"].value_counts().
 ↪index,hue=combinedDf['cluster_predicted'],ax=axes2[1])
axes2[1].set_title("Gráfica 8 - Clusterización de la Delegación Regional")
axes2[1].set(xlabel='Delegación Regional (deleg)')
axes2[1].set(ylabel="Conteo por cluster")
sns.countplot(x=combinedDf["sat1"],order=combinedDf["sat1"].value_counts().
 ↪index,hue=combinedDf['cluster_predicted'],ax=axes2[2])
axes2[2].set_title("Gráfica 9 - Clusterización de la calidad en la atención")
axes2[2].set(xlabel='Calidad de la atención (sat1)')
axes2[2].set(ylabel="Conteo por cluster")
sns.countplot(x=combinedDf["edad_cat"],order=combinedDf["edad_cat"].
 ↪value_counts().index,hue=combinedDf['cluster_predicted'],ax=axes2[3])
axes2[3].set_title("Gráfica 10 - Clusterización de rango de edades")
axes2[3].set(xlabel='Rango de edades (edad_cat)')
axes2[3].set(ylabel="Conteo por cluster")
sns.countplot(x=combinedDf["sexo"],order=combinedDf["sexo"].value_counts().
 ↪index,hue=combinedDf['cluster_predicted'],ax=axes2[4])
axes2[4].set_title("Gráfica 11 - Clusterización por género (sexo)")
axes2[4].set(xlabel='Género (Mujer = 1 / Hombre = 2)')
axes2[4].set(ylabel="Conteo por cluster")
plt.tight_layout()
plt.show()
```



Como siguiente paso, evaluaremos la calidad de los clusters obtenidos con K-Modes mediante un **clasificador LightGBM**.

- Primero, preparamos los datos **convirtiendo las variables categóricas a un formato adecuado para el clasificador**.
- Luego, **se entrena un modelo LightGBM** con muestreo de columnas para evitar sobreajuste y calcular la puntuación F1 ponderada mediante validación cruzada (cross-val-score), la cual refleja el **rendimiento del modelo** en la tarea de predecir los clusters a los que pertenecen los datos.
- Finalmente, **imprime la media de las puntuaciones F1** obtenidas en cada iteración de la validación cruzada, proporcionando un estimador del rendimiento de los clusters.

Para considerar fiables nuestros clusters generados, el valor arrojado F1 debe ser lo más cercano a 1. Entre más cercano sea este valor a 1, **nuestros clusters serán más fiables a los datos reales**, y viceversa, entre más alejados estén de 1, serán menos confiables.

[ ]:
```
kmtst_data = X_clean.copy()
for c in kmtst_data.select_dtypes(include='object'):
    kmtst_data[c] = kmtst_data[c].astype('category')

clf_km = LGBMClassifier(colsample_bytree=0.8)
cv_scores_km = cross_val_score(clf_km, kmtst_data, clustersDf,␣
 ↪scoring='f1_weighted')
print(f'La media de la puntuación F1 obtenida para los clusters K-Modes es {np.
 ↪mean(cv_scores_km)}')
```

```
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000152 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 67
[LightGBM] [Info] Number of data points in the train set: 3876, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675248
[LightGBM] [Info] Start training from score -1.499829
[LightGBM] [Info] Start training from score -2.208120
[LightGBM] [Info] Start training from score -2.575584
[LightGBM] [Info] Start training from score -2.503657

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000152 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 67
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675506
[LightGBM] [Info] Start training from score -1.500087
[LightGBM] [Info] Start training from score -2.208378
[LightGBM] [Info] Start training from score -2.575842
[LightGBM] [Info] Start training from score -2.500766

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
```

```
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000134 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 67
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675000
[LightGBM] [Info] Start training from score -1.500087
[LightGBM] [Info] Start training from score -2.208378
[LightGBM] [Info] Start training from score -2.579237
[LightGBM] [Info] Start training from score -2.500766

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000168 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 67
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675000
[LightGBM] [Info] Start training from score -1.501244
[LightGBM] [Info] Start training from score -2.206033
[LightGBM] [Info] Start training from score -2.579237
[LightGBM] [Info] Start training from score -2.500766

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000160 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 66
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675000
[LightGBM] [Info] Start training from score -1.500087
[LightGBM] [Info] Start training from score -2.206033
[LightGBM] [Info] Start training from score -2.579237
[LightGBM] [Info] Start training from score -2.503915
```

```
La media de la puntuación F1 obtenida para los clusters K-Modes es
0.8459270982061879
```

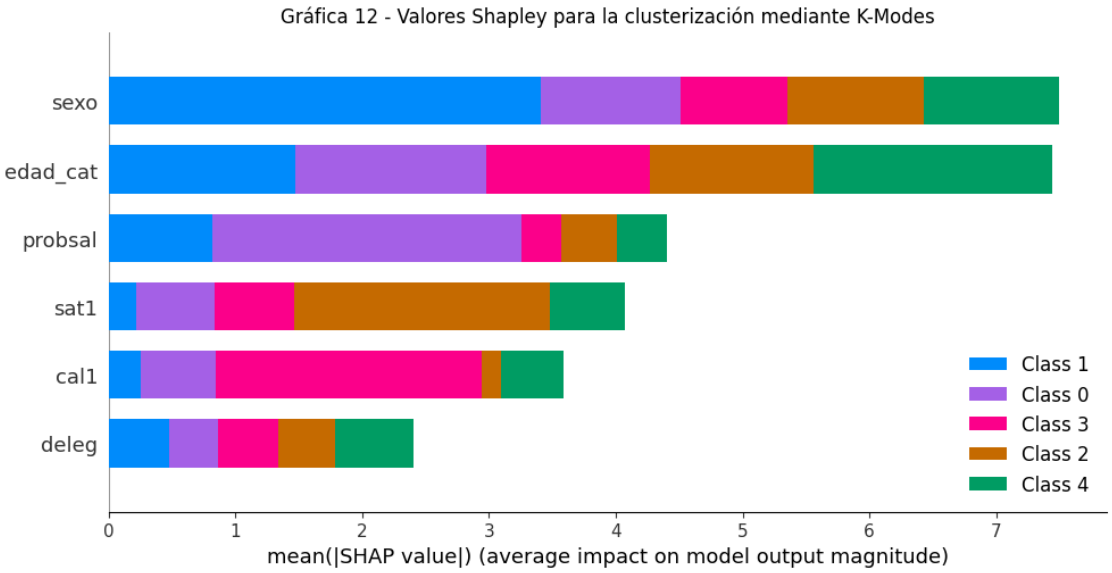
Como análisis final de la clusterización, utilizaremos la técnica de explicación de **modelos Shapley Values (SHAP)** para comprender la importancia de las variables en la clasificación de los clusters.

- Primero, entrenamos un modelo con el clasificador LightGBM para predecir los clusters.
- Luego, utilizamos el modelo entrenado para **calcular los valores Shapley** para cada variable en el conjunto de datos.
- Finalmente, genera una visualización de los valores Shapley, que muestra la importancia relativa de cada variable en la clasificación de los clusters (*Gráfica 12*).

Cada gráfica Shapley tendrá una variación de acuerdo con el algoritmo utilizado para realizar la clusterización. Esta variación entre las dos gráficas de Valores Shapley **se discutirá en la presentación** del proyecto.

[ ]:
```
clf_km.fit(kmtst_data, clustersDf)
explainer_km = shap.TreeExplainer(clf_km)
shap_values_km = explainer_km.shap_values(kmtst_data)
shap.summary_plot(shap_values_km, kmtst_data, plot_type="bar", plot_size=(10,␣
 ↪5), show=False)
plt.title("Gráfica 12 - Valores Shapley para la clusterización mediante␣
 ↪K-Modes")
plt.show()
```

```
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.000748 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 67
[LightGBM] [Info] Number of data points in the train set: 4846, number of used
features: 6
[LightGBM] [Info] Start training from score -0.675151
[LightGBM] [Info] Start training from score -1.500267
[LightGBM] [Info] Start training from score -2.207387
[LightGBM] [Info] Start training from score -2.577826
[LightGBM] [Info] Start training from score -2.501973
```



1.2 *Método 2: Clusterización por K-Prototypes + reducción de dimensiones por UMAP*

Como segundo algoritmo a probar, se presentará la variante **K-Prototypes** del algoritmo, que se probará con un algoritmo especial de reducción de dimensiones, el **Uniform Manifold Approximation and Projection (UMAP - Aproximación y proyección de colectores uniformes)**. Es una técnica de reducción de dimensiones que se puede utilizar para visualización de manera similar a t-SNE, pero también para reducción de dimensiones no lineal general.

- En la primera parte, se prepara **el conjunto de datos para el análisis**. Se seleccionan las variables de interés ("deleg", "sat1", "probsal", "cal1", "edad" y "sexo") y se escalan utilizando la función StandardScaler().
- En la segunda parte, se realiza el análisis de reducción de dimensionalidad. Se utiliza el algoritmo UMAP para **crear un espacio de representación de dos dimensiones** para los datos.

El resultado del análisis se almacena en la variable embedding, que es un array de dos dimensiones.

[ ]:
```
X_proto = df_clean[["deleg","sat1","probsal","cal1","edad","sexo"]]
X_proto_scaled = preprocessing.StandardScaler().fit_transform(X_proto)

reducer = umap.UMAP()
embedding = reducer.fit_transform(X_proto_scaled)
embedding.shape
```

[ ]:
```
(4846, 2)
```

Al igual que en la clusterización anterior, se puede realizar el **método del codo** para conocer la cantidad de k idónea para clusterizar nuestros datos, modificando únicamente el algoritmo a

utilizar. El resultado se muestra en la *Gráfica 13*.

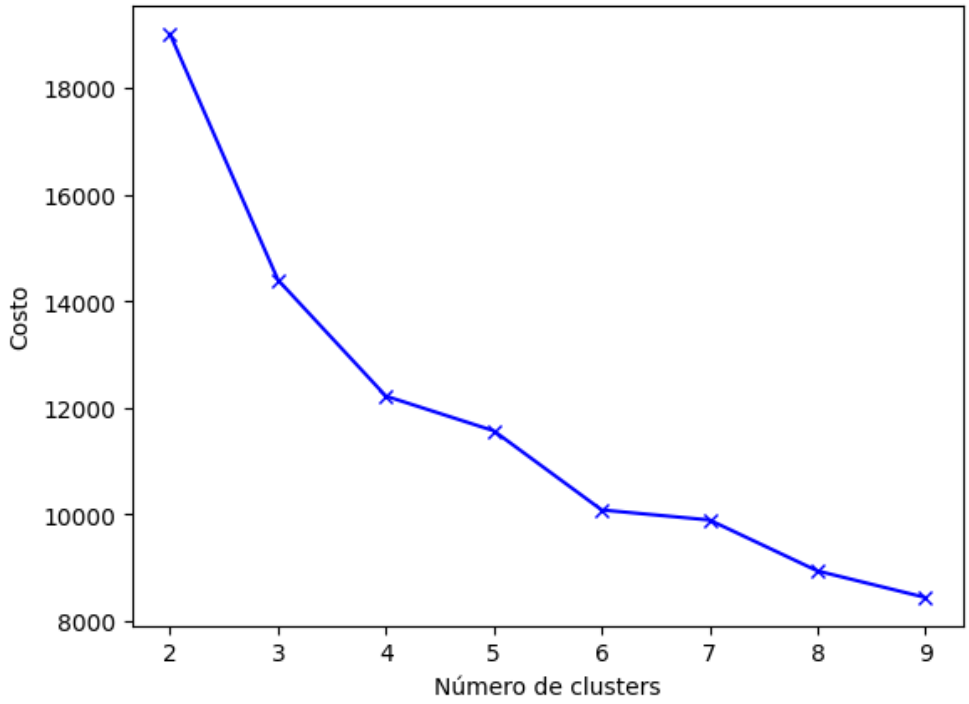
[ ]:
```
cost_prot = []
K_prot = range(2,10)
categorical_columns = [0, 2]
for num_clusters_prot in list(K_prot):
  kprot = kprototypes.KPrototypes(n_clusters = num_clusters_prot, init = "Cao",␣
 ↪n_init = 1, verbose = 1)
  kprot.fit_predict(X_proto_scaled, categorical=categorical_columns)
  cost_prot.append(kprot.cost_)

plt.plot(K_prot,cost_prot,'bx-')
plt.xlabel('Número de clusters')
plt.ylabel("Costo")
plt.title("Gráfica 13: Método del codo (Elbow) para k óptima, usando␣
 ↪K-Prototypes - Cao")
plt.show()
```

```
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 864, ncost: 19352.343643932898
Run: 1, iteration: 2/100, moves: 405, ncost: 19072.730019778723
Run: 1, iteration: 3/100, moves: 172, ncost: 19041.962083640876
Run: 1, iteration: 4/100, moves: 91, ncost: 19030.65731635582
Run: 1, iteration: 5/100, moves: 60, ncost: 19024.336676986135
Run: 1, iteration: 6/100, moves: 58, ncost: 19019.499921847055
Run: 1, iteration: 7/100, moves: 46, ncost: 19016.74307285987
Run: 1, iteration: 8/100, moves: 15, ncost: 19016.358798029276
Run: 1, iteration: 9/100, moves: 0, ncost: 19016.358798029276
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1962, ncost: 15803.845267502631
Run: 1, iteration: 2/100, moves: 853, ncost: 14448.783748544953
Run: 1, iteration: 3/100, moves: 93, ncost: 14403.34700051256
Run: 1, iteration: 4/100, moves: 0, ncost: 14403.34700051256
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1782, ncost: 14754.60564634751
Run: 1, iteration: 2/100, moves: 925, ncost: 12995.500571738216
Run: 1, iteration: 3/100, moves: 622, ncost: 12293.724308638033
Run: 1, iteration: 4/100, moves: 117, ncost: 12269.411473236105
Run: 1, iteration: 5/100, moves: 56, ncost: 12255.90273378023
```

```
Run: 1, iteration: 6/100, moves: 77, ncost: 12230.862216534239
Run: 1, iteration: 7/100, moves: 54, ncost: 12222.891280631178
Run: 1, iteration: 8/100, moves: 4, ncost: 12222.849530817093
Run: 1, iteration: 9/100, moves: 0, ncost: 12222.849530817093
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1001, ncost: 11792.96266671384
Run: 1, iteration: 2/100, moves: 310, ncost: 11576.364007110724
Run: 1, iteration: 3/100, moves: 42, ncost: 11571.676861077638
Run: 1, iteration: 4/100, moves: 7, ncost: 11571.540651685189
Run: 1, iteration: 5/100, moves: 2, ncost: 11571.502888815301
Run: 1, iteration: 6/100, moves: 1, ncost: 11571.479954343831
Run: 1, iteration: 7/100, moves: 1, ncost: 11571.475096581371
Run: 1, iteration: 8/100, moves: 0, ncost: 11571.475096581371
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1522, ncost: 10614.54187047314
Run: 1, iteration: 2/100, moves: 578, ncost: 10131.08324866834
Run: 1, iteration: 3/100, moves: 154, ncost: 10093.39352966762
Run: 1, iteration: 4/100, moves: 50, ncost: 10086.786977528436
Run: 1, iteration: 5/100, moves: 18, ncost: 10086.537051523079
Run: 1, iteration: 6/100, moves: 0, ncost: 10086.537051523079
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1861, ncost: 10381.959113958317
Run: 1, iteration: 2/100, moves: 510, ncost: 9999.346482548925
Run: 1, iteration: 3/100, moves: 69, ncost: 9994.893200134371
Run: 1, iteration: 4/100, moves: 41, ncost: 9969.278148247537
Run: 1, iteration: 5/100, moves: 77, ncost: 9900.584220595827
Run: 1, iteration: 6/100, moves: 14, ncost: 9897.798508073713
Run: 1, iteration: 7/100, moves: 1, ncost: 9897.792358591472
Run: 1, iteration: 8/100, moves: 0, ncost: 9897.792358591472
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1465, ncost: 9741.318587868745
Run: 1, iteration: 2/100, moves: 714, ncost: 9123.34586221446
Run: 1, iteration: 3/100, moves: 333, ncost: 9019.124341899153
Run: 1, iteration: 4/100, moves: 226, ncost: 8961.979493674657
Run: 1, iteration: 5/100, moves: 105, ncost: 8942.533027665786
Run: 1, iteration: 6/100, moves: 23, ncost: 8942.028726303739
```

```
Run: 1, iteration: 7/100, moves: 1, ncost: 8942.015654750487
Run: 1, iteration: 8/100, moves: 0, ncost: 8942.015654750487
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/100, moves: 1409, ncost: 9295.539103708204
Run: 1, iteration: 2/100, moves: 759, ncost: 8655.295857590847
Run: 1, iteration: 3/100, moves: 424, ncost: 8474.181724071515
Run: 1, iteration: 4/100, moves: 175, ncost: 8453.553844870785
Run: 1, iteration: 5/100, moves: 84, ncost: 8444.9249140551
Run: 1, iteration: 6/100, moves: 26, ncost: 8444.119122033311
Run: 1, iteration: 7/100, moves: 3, ncost: 8444.103377125382
Run: 1, iteration: 8/100, moves: 0, ncost: 8444.103377125382
```

Gráfica 13: Método del codo (Elbow) para k óptima, usando K-Prototypes - Cao



Generamos la **visualización de los datos** en el espacio de representación de dos dimensiones generado por el algoritmo UMAP.

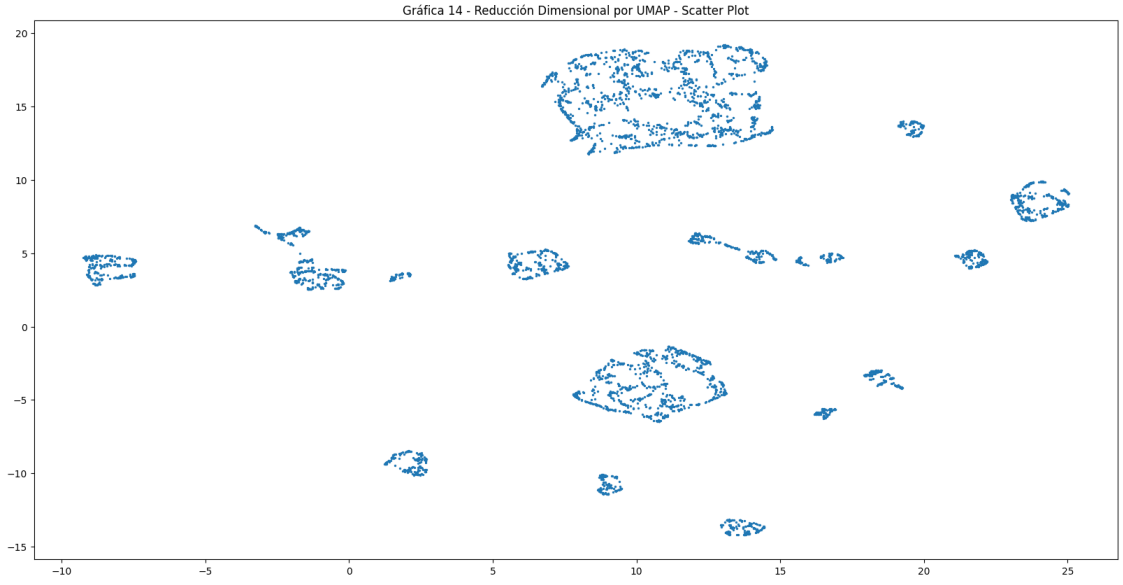
- Se genera un **scatter plot de los datos**, con cada punto representado por un círculo, con base a la reducción dimensional generada en la variable **"embedding"**, y mostrada en la *Gráfica 14*.

Esta visualización se utilizará para comprender las relaciones entre las variables y a identificar los clusteres respectivos en los datos.

- De inicio, y gracias al scatter plot (dispersión de puntos) de tipo espectral, podemos **identificar una mayor cantidad de clusteres** que los generados anteriormente. Esta afirmación puede y debe respaldarse con alguno de los algoritmos generados para k vistos anteriormente.

[ ]:
```
plt.figure(figsize=(20, 10))
plt.scatter(*embedding.T, s=2, cmap='Spectral', alpha=1.0)
plt.title("Gráfica 14 - Reducción Dimensional por UMAP - Scatter Plot")
plt.show()
```

```
No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
```



Posteriormente, aplicamos el **algoritmo de clustering K-Prototypes** para agrupar los datos en 5 grupos, considerando tanto variables numéricas como categóricas.

- La primera parte del código prepara los datos **transformando las variables numéricas mediante la transformación de potencia** para mejorar la normalidad y estabilidad, y luego ejecuta el algoritmo con la inicialización de centroides de Cao y empleando 4 núcleos para cómputo paralelo.
- Finalmente, muestra la **distribución de los datos en cada cluster** para comprender el tamaño y composición de los grupos resultantes.

K-Prototypes, al igual que K-Modes, **está basado en K-Means** y es muy similar en función al mismo, sin embargo, su gran diferencia es la posibilidad de procesar tanto variables categóricas como variables numéricas, utilizando una combinación de **distancias de la moda** (para datos categóricos) y **distancias euclidianas** (para datos numéricos).

[ ]:
```
X_proto_data = X_proto.copy()

for c in X_proto.select_dtypes(exclude='object').columns:
```

```
    pt = preprocessing.PowerTransformer()
    X_proto_data[c] =  pt.fit_transform(np.array(X_proto_data[c]).reshape(-1,␣
 ↪1))

categorical_columns = [0, 2]

#Algoritmo para realizar la clusterización mediante K-Prototypes
kproto = kprototypes.KPrototypes(n_clusters= 5, init='Cao', n_jobs = 4)
clusters = kproto.fit_predict(X_proto_data, categorical=categorical_columns)

pd.Series(clusters).value_counts()
```

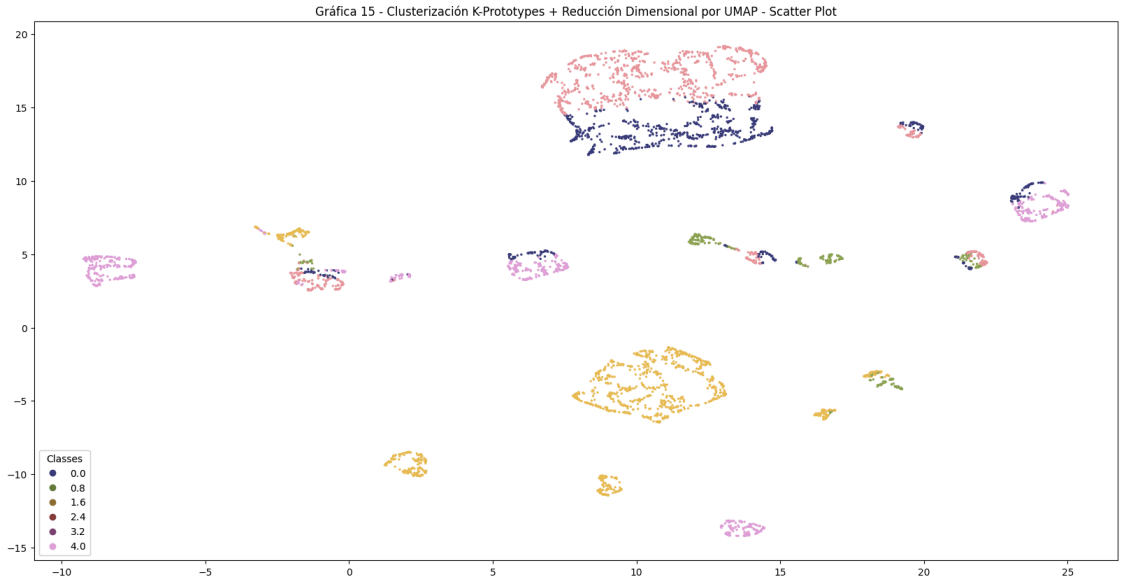
[ ]:
```
2    1353
3    1242
0     952
4     888
1     411
dtype: int64
```

Posteriormente se genera **una visualización de los datos en dos dimensiones**, coloreando los puntos según la asignación a los clusters resultantes del algoritmo K-Prototypes:

- Para ello, se utiliza la representación de los datos obtenida con UMAP y la combina con la información de cada uno de los clusters generados. Además, crea una leyenda que indica **la correspondencia entre los colores y los clusters**, facilitando la interpretación de los grupos identificados en la visualización realizada con base al algoritmo K-Prototypes. Esta visualización se muestra en la *Gráfica 15*.

[ ]:
```
fig, ax = plt.subplots()
fig.set_size_inches((20, 10))
scatter = ax.scatter(embedding[:, 0], embedding[:, 1], s=2, c=clusters,␣
 ↪cmap='tab20b', alpha=1.0)

#Se incorpora un color diferente a la dispersión de puntos, de acuerdo con el␣
 ↪cluster identificado
legend1 = ax.legend(*scatter.legend_elements(num=5),
                    loc="lower left", title="Classes")
ax.add_artist(legend1)
ax.set_title("Gráfica 15 - Clusterización K-Prototypes + Reducción Dimensional␣
 ↪por UMAP - Scatter Plot")
```

[ ]:
```
Text(0.5, 1.0, 'Gráfica 15 - Clusterización K-Prototypes + Reducción Dimensional
por UMAP - Scatter Plot')
```



Como siguiente paso, y al igual que en el caso de K-Modes, evaluaremos la calidad de los clusters obtenidos con K-Prototypes mediante un **clasificador LightGBM**.

- Primero, preparamos los datos **convirtiendo las variables categóricas a un formato adecuado para el clasificador**.
- Luego, **se entrena un modelo LightGBM** con muestreo de columnas para evitar sobreajuste y calcular la puntuación F1 ponderada mediante validación cruzada (cross-val-score), la cual refleja el **rendimiento del modelo** en la tarea de predecir los clusters a los que pertenecen los datos.
- Finalmente, **imprime la media de las puntuaciones F1** obtenidas en cada iteración de la validación cruzada, proporcionando un estimador del rendimiento de los clusters.

Para considerar fiables nuestros clusters generados, el valor arrojado F1 debe ser lo más cercano a 1. Entre más cercano sea este valor a 1, **nuestros clusters serán más fiables a los datos reales**, y viceversa, entre más alejados estén de 1, serán menos confiables.

[ ]:
```
lgbm_data = X_proto.copy()
for c in lgbm_data.select_dtypes(include='object'):
    lgbm_data[c] = lgbm_data[c].astype('category')

clf_kp = LGBMClassifier(colsample_bytree=0.8)
cv_scores_kp = cross_val_score(clf_kp, lgbm_data, clusters,␣
 ↪scoring='f1_weighted')
print(f'La media de la puntuación F1 obtenida para los clusters K-Prototypes es␣
 ↪{np.mean(cv_scores_kp)}')
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.000334 seconds.
```

```
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 132
[LightGBM] [Info] Number of data points in the train set: 3876, number of used
features: 6
[LightGBM] [Info] Start training from score -1.627926
[LightGBM] [Info] Start training from score -2.469545
[LightGBM] [Info] Start training from score -1.275069
[LightGBM] [Info] Start training from score -1.361828
[LightGBM] [Info] Start training from score -1.695887
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000094 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 132
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -1.628184
[LightGBM] [Info] Start training from score -2.466759
[LightGBM] [Info] Start training from score -1.275327
[LightGBM] [Info] Start training from score -1.362086
[LightGBM] [Info] Start training from score -1.696145
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000092 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 131
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -1.626870
[LightGBM] [Info] Start training from score -2.466759
[LightGBM] [Info] Start training from score -1.276250
[LightGBM] [Info] Start training from score -1.361080
[LightGBM] [Info] Start training from score -1.697552
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000095 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 132
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
```

```
[LightGBM] [Info] Start training from score -1.626870
[LightGBM] [Info] Start training from score -2.466759
[LightGBM] [Info] Start training from score -1.276250
[LightGBM] [Info] Start training from score -1.361080
[LightGBM] [Info] Start training from score -1.697552
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000120 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 130
[LightGBM] [Info] Number of data points in the train set: 3877, number of used
features: 6
[LightGBM] [Info] Start training from score -1.626870
[LightGBM] [Info] Start training from score -2.466759
[LightGBM] [Info] Start training from score -1.276250
[LightGBM] [Info] Start training from score -1.361080
[LightGBM] [Info] Start training from score -1.697552
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
La media de la puntuación F1 obtenida para los clusters K-Prototypes es
0.9851590742828957
```

Para finalizar nuestra última clusterización, utilizaremos, de nueva cuenta, la técnica de explicación de **modelos Shapley Values (SHAP)** para comprender la importancia de las variables en la clasificación de los clusters.

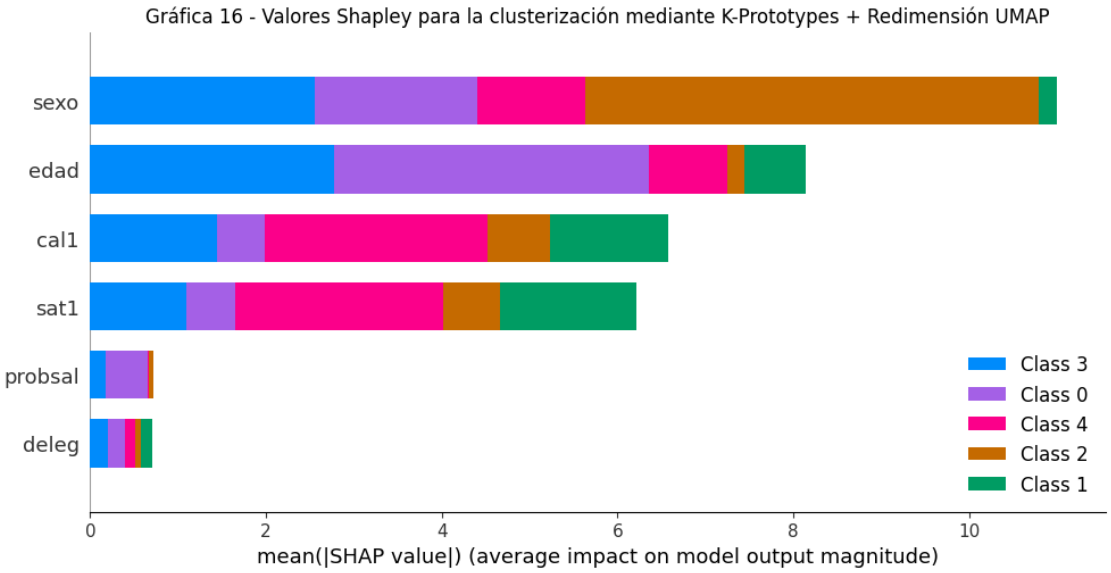
- Primero, entrenamos un modelo con el clasificador LightGBM para predecir los clusters.
- Luego, utilizamos el modelo entrenado para **calcular los valores Shapley** para cada variable en el conjunto de datos.
- Finalmente, genera una visualización de los valores Shapley (*Gráfica 16*), que muestra la importancia relativa de cada variable en la clasificación de los clusters.

[ ]:
```
clf_kp.fit(lgbm_data, clusters)
explainer_kp = shap.TreeExplainer(clf_kp)
shap_values_kp = explainer_kp.shap_values(lgbm_data)
shap.summary_plot(shap_values_kp, lgbm_data, plot_type="bar", plot_size=(10,␣
↪5), show=False)
plt.title("Gráfica 16 - Valores Shapley para la clusterización mediante␣
↪K-Prototypes + Redimensión UMAP")
plt.show()
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000121 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
```

```
[LightGBM] [Info] Total Bins 132
[LightGBM] [Info] Number of data points in the train set: 4846, number of used
features: 6
[LightGBM] [Info] Start training from score -1.627344
[LightGBM] [Info] Start training from score -2.467316
[LightGBM] [Info] Start training from score -1.275829
[LightGBM] [Info] Start training from score -1.361431
[LightGBM] [Info] Start training from score -1.696937
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

Gráfica 16 - Valores Shapley para la clusterización mediante K-Prototypes + Redimensión UMAP



1.2.1 Explicación de resultados

Para explicar estos resultados, sabemos que la satisfacción de los pacientes es una medida de cómo los pacientes **perciben la calidad de la atención** que reciben dentro de las clínicas del IMSS.

UMAP (Uniform Manifold Approximation and Projection) es un algoritmo de reducción de dimensionalidad no lineal que **mapea datos de alta dimensión a un espacio de menor dimensión** preservando la estructura subyacente de los datos. Utiliza una combinación de técnicas de optimización y geometría Riemanniana para modelar la relación entre puntos en el espacio original y el espacio reducido, minimizando la energía potencial en el espacio de menor dimensión. UMAP se destaca por su capacidad para preservar la topología local, capturar estructuras no lineales y **ser eficiente en el procesamiento de grandes conjuntos de datos**.

A través de la reducción dimensional por UMAP, **simplificamos los datos** sin perder mucha información, lo cual mejora considerablemente el rendimiento de nuestro modelo de aprendizaje al trabajar los datos, a diferencia de lo que sucedió con K-Modes, donde el score F1 **es considerablemente menor**, donde los datos de prueba también influirán en el rendimiento de nuestro

modelo.

En este caso, los datos representan las muestras de satisfacción de pacientes en el IMSS.

- La reducción dimensional por UMAP redujo los datos **a 2 dimensiones** conservando las características del dataset.
- La clusterización **K-Prototypes** utilizó esta reducción de dimensiones como ventaja, contra la clusterización por **K-Modes**, que no redimensionó los datos, únicamente realizó la codificación de los mismos.
- Los colores de los puntos en el **gráfico de dispersión (Gráfica 15)** representan los clusters a los que pertenecen los datos. Los puntos del mismo color pertenecen al mismo cluster. UMAP tiende a preservar la topología local de los datos, lo que significa que la relación entre puntos cercanos **se mantiene** en el espacio de menor dimensión.
- Para K-Modes, se realizó una graficación por barras de la característica representada por su grupo en cada cluster **(Gráficas 7 a 11)**

Los clusters representan **grupos de pacientes con características similares**.

- Si seguimos el ejemplo anterior, y **para el caso del Método 2**, tomando en cuenta lo dicho en los valores Shapley, podemos deducir que la clusterización agrupa (de manera principal) pacientes por su sexo, edad, y la calidad de la atención recibida, con una mejor consideración de enfermedades y geografía. **El cluster más grande (cluster "0")** que se ve en el scatter plot es el que está **satisfecho con el servicio (calificación "2")**, sin que sea la calificación máxima de nuestra escala.
- **Para el caso del Método 1**, los grupos son similares, pero en el caso de los valores Shapley, el peso de cada variable cambia en la clusterización, tomando **las enfermedades (probsal) un mayor peso** a la hora de realizar este proceso.

Podemos utilizar esta información para **identificar áreas de mejora en la atención al paciente en el IMSS**. Por ejemplo, si un cluster representa a pacientes que están insatisfechos con la atención que reciben, podemos investigar las causas de esta insatisfacción. Una vez que hayamos identificado las causas, podemos tomar medidas para mejorar la atención al paciente en estas áreas.