Tecnológico de Monterrey

Maestría en Inteligencia Artificial Aplicada

Materia | Proyecto Integrador

Laura Elena Hernández Mata | A01169213

Evelyn Aylin Rendon Medina | A01748750

Samara García González | A01273001

Proyecto | Islas de Calor y Justicia Social

Avance 3 | Baseline

Asesor: Prof. Dr. Roberto Ponce López

Tutora: María de la Paz Rico Fernández

13 de octubre de 2024

Índice

1. Algoritmo

La selección de un algoritmo como base para las siguientes entregas es clave, ya que nos servirá como referencia con respecto al rendimiento de futuras ejecuciones. En este sentido, nos aseguramos de que el algoritmo elegido estuviera tanto alineado con el objetivo del proyecto, como que contara con un resultado satisfactorio.

- **Justificación de uso para el tipo de problema**

El algoritmo que utilizaremos se llama k-means, el cual, de manera inicial, nos permitirá saber cuáles son los parámetros mínimos aceptables para el proyecto. La finalidad de este es clusterizar o agrupar los datos con base en sus características, siendo este un modelo no supervisado. Estamos utilizando el método del codo para definir el nivel óptimo de k, es decir, de cuántos clústeres serán manejados.

Según Ramírez (2023), los pasos que sigue este algoritmo inician con especificar el número de clústeres deseados, para luego obtener los centroides de cada clúster. Esto permite clasificar al conjunto de datos cuando su distancia cuadrática es menor con respecto a la posición del centroide. Luego, se recalculan los centroides como media de los puntos previamente clasificados; estos últimos dos pasos se vuelven iterativos hasta que los clústeres no cambien.

El método del codo es bastante popular y sirve, como se mencionó anteriormente, para encontrar el valor óptimo de k. Adicionalmente, aplicamos el valor de *silhouette* o silueta, esto con la finalidad de determinar qué tan similar es un punto con su propio grupo y así conocer su cohesión contrastado con el resto de los clústeres y su separación de estos. Por tanto, un valor alto indicará que el punto pertenece al clúster que debería.

Consideramos que este es viable ya que nos ayuda a segmentar los datos, que es el tipo de aplicación que se requiere para conocer cuántos grupos hay en los sectores poblacionales. Dadas las variables seleccionadas que corresponden a la vulnerabilidad o sus atenuantes, pretendemos agrupar a aquellas AGEBs que sean similares entre sí. Más adelante, esto nos permitirá determinar la criticidad de la vulnerabilidad para cada uno de los grupos obtenidos.

- **Análisis de aspectos clave**

El tipo de datos que se tiene como fuente de información del SCITEL (INEGI, 2020) son estructurados. Esto quiere decir que se encuentran en una tabla y siguen un estándar para su clasificación; en las filas y columnas encontramos los datos ya

codificados, lo que facilita su uso, especialmente el de las etiquetas cualitativas como Estado.

Del diccionario de datos, seleccionamos aquellas variables que consideramos relevantes para determinar el grado de vulnerabilidad por Estado y AGEB. Como sabemos, el DataFrame requiere tener un mismo tipo de datos por columna, los cuales inicialmente fueron objetos, pero fueron procesados para convertirse en datos numéricos enteros (int) que hace concordancia con el tipo de datos que se maneja.

En cuanto a la cantidad de datos, contamos con un total de 235,243 registros en un total de 108 columnas, una vez que el set de datos ha sido procesado, tal como se explica anteriormente.

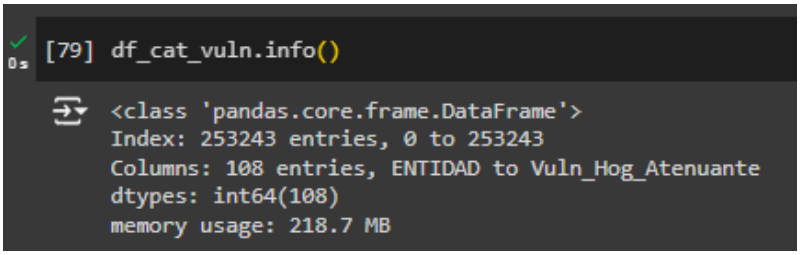

Imagen 1. Análisis de cantidad de datos. *Elaboración propia*

Las columnas que contiene nuestro set de datos está relacionado con lo que la población es y tiene, esto nos permite interpretar su nivel de vulnerabilidad y los factores que podrían atenuar dicha condición. Por tanto, es necesario interpretar la magnitud de personas vulnerables en determinada región, lo que nos permitirá contrastar contra las islas de calor. Los datos seleccionados sabemos que poseen los datos de interés para dicho objetivo; esta interpretación es parte del procesamiento.

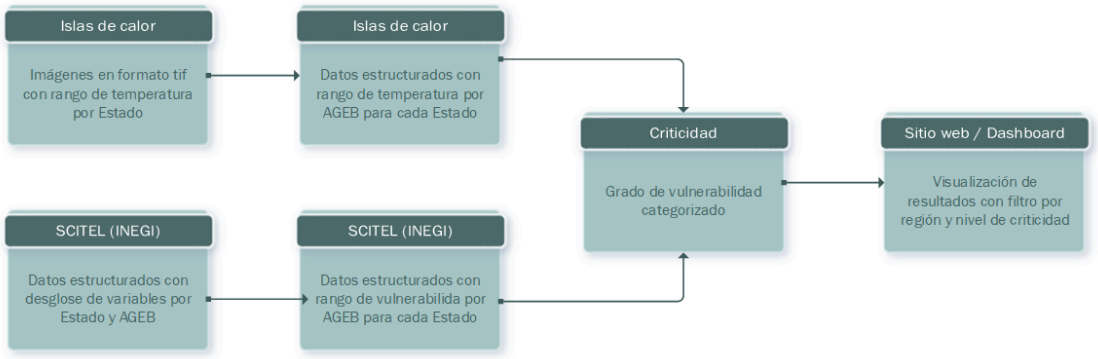

Imagen 2. Diagrama de entradas y salidas. *Elaboración propia*.

2. Características importantes

En esta sección se realiza el análisis de relevancia de las características observadas. Esto a través de métodos de selección y de extracción de características.

- **Relevancia y/o extracción de características**

En el análisis de esta semana nos percatamos que había algunas filas totalizadoras que podrían no ser de utilidad para el análisis subsecuente, ya que podrían afectar directamente los niveles de agregación. En la siguiente imagen se puede apreciar que estos totalizadores no tenían una localidad, AGEB, y manzana asignada.

```
In [61]:
df_concat_3copy = df_concat_3estados.copy()
label_encoder = LabelEncoder()
df_concat_3copy['NOM_ENT'] = label_encoder.fit_transform(df_concat_3copy['NOM_ENT'])
df_concat_3copy.head()
```

Out[61]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_RADIO	VPH_TV	VPH_PC	VPI
0	13.0	1	0.0	0000	0.0	3082841.0	1601462	1481379	134738	66770	...	579898	759456	261093	
1	13.0	1	0.0	0000	0.0	22268.0	11563	10705	1241	593	...	3835	5303	1045	
2	13.0	1	1.0	0000	0.0	439.0	229	210	21	6	...	76	116	39	
3	13.0	1	1.0	0043	0.0	439.0	229	210	21	6	...	76	116	39	
4	13.0	1	1.0	0043	1.0	92.0	54	38	6	*	...	12	21	7	

5 rows × 102 columns

Imagen 3. Contenido del DataFrame previo a la extracción. *Elaboración propia*.

Por tanto, decidimos eliminar esa información para así volver a crear las características que se trabajaron anteriormente. A continuación, se muestra el código utilizado para este fin, así como el resultado, donde sólo se encuentran los datos necesarios para continuar con nuestro análisis.

```
df_cat1 = df_concat_3estados.copy()

df_cat1 = df_cat1[~df_cat1['NOM_LOC'].str.contains('Total|total', case=False, na=False)]
df_cat1.head()
```

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VPH_RADIO	VPH_TV	VPH_PC	VPH
4	13.0	Hidalgo	1.0	Acatlán	0043	1.0	92.0	54	38	6	...	12	21	7	
5	13.0	Hidalgo	1.0	Acatlán	0043	2.0	7.0	4	3	0	...	*	*	*	
6	13.0	Hidalgo	1.0	Acatlán	0043	3.0	0.0	0	0	0	...	0	0	0	
7	13.0	Hidalgo	1.0	Acatlán	0043	6.0	47.0	20	27	*	...	11	10	6	
8	13.0	Hidalgo	1.0	Acatlán	0043	7.0	44.0	21	23	*	...	8	12	3	

5 rows × 103 columns

Imagen 4. Contenido del DataFrame posterior a la extracción. *Elaboración propia*.

En cuanto a las columnas contenidas en nuestro DataFrame, ya no era necesario mantener el nombre de la localidad, ya que esta sólo se necesitaba para el paso anterior, que fue quitar los totalizadores. Por tanto, se procedió con la eliminación de dicha característica.

```
df_cat2 = df_cat1.copy()

df_cat2 = df_cat2.drop(columns=['NOM_LOC'])
df_cat2.head()
```

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_RADIO	VPH_TV	VPH_PC	VPH_
4	13.0	Hidalgo	1.0	0043	1.0	92.0	54	38	6	*	...	12	21	7	
5	13.0	Hidalgo	1.0	0043	2.0	7.0	4	3	0	0	...	*	*	*	
6	13.0	Hidalgo	1.0	0043	3.0	0.0	0	0	0	0	...	0	0	0	
7	13.0	Hidalgo	1.0	0043	6.0	47.0	20	27	*	0	...	11	10	6	
8	13.0	Hidalgo	1.0	0043	7.0	44.0	21	23	*	0	...	8	12	3	

Imagen 5. Contenido del DataFrame posterior a la segunda extracción. *Elaboración propia*.

Procedemos a revisar la correlación de las variables, esto para entender cuál es su relación entre sí. Igualmente, nos permitirá comprender los resultados obtenidos más adelante, ya que nos contextualiza con base en los datos que se tienen en el Data Frame.

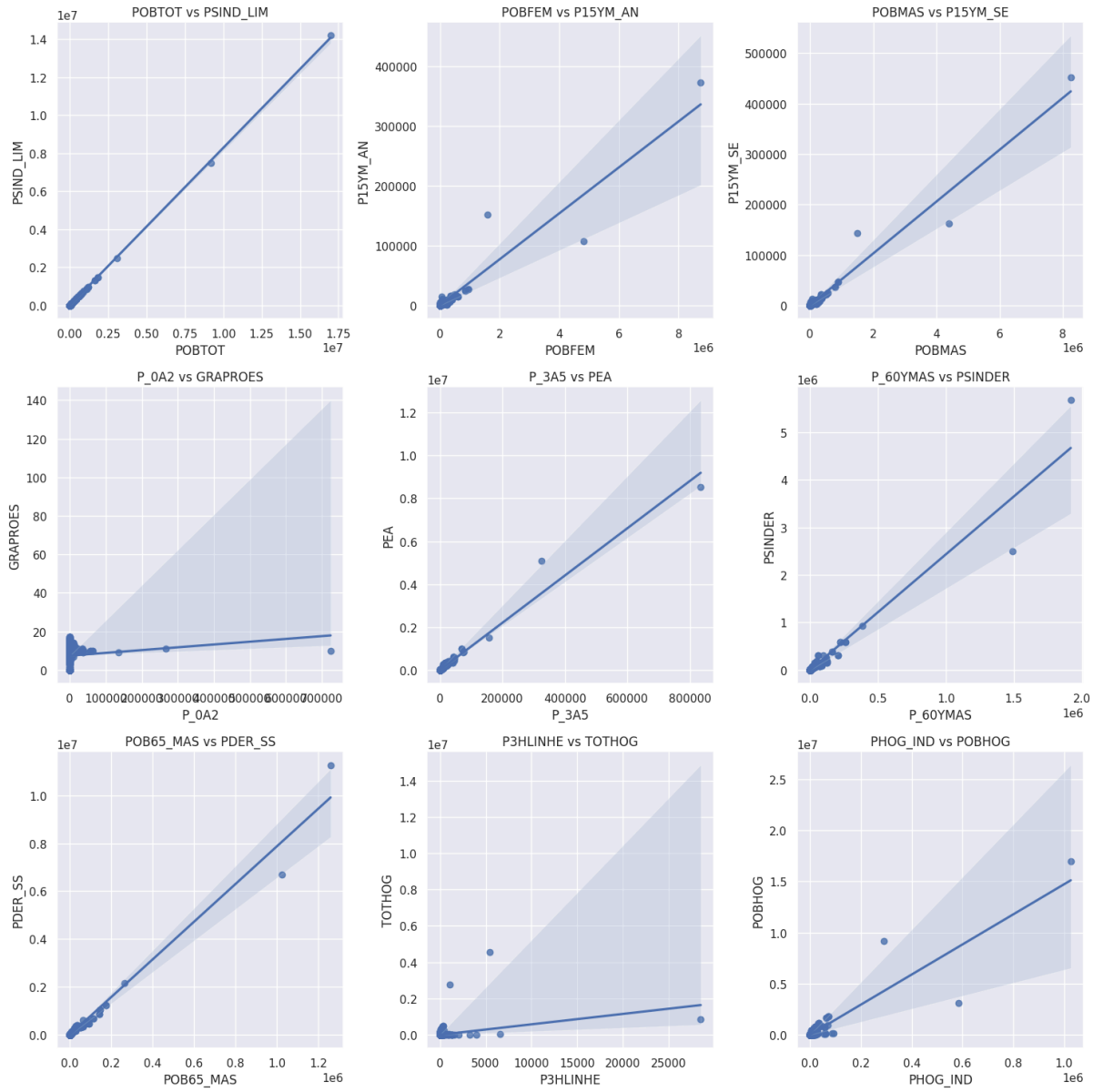

Imagen 6. Análisis de correlación posterior a la extracción. *Elaboración propia*.

Posteriormente, procedimos a repetir el análisis de PCA para determinar los componentes más relevantes en nuestro proyecto. Como parte de los resultados, también imprimimos el resultado obtenido.

```
============PC1================
** Más importante:
POBFEM         0.136129
POBHOG         0.136073
OCUPVIVPAR     0.136073
Name: PC1, dtype: float64
** Menos importante:
ENTIDAD    -0.015315
LOC        -0.007141
MZA        -0.003650
Name: PC1, dtype: float64
============PC2================
** Más importante:
P15YM_AN       0.243883
VPH_LETR       0.236292
P15YM_AN_F     0.234966
Name: PC2, dtype: float64
** Menos importante:
VPH_PC       -0.128372
VPH_TELEF    -0.117825
VPH_INTER    -0.110911
Name: PC2, dtype: float64
============PC3================
** Más importante:
P15YM_SE_M      0.330423
P15YM_SE        0.330241
PAFIL_OTRAI     0.318513
Name: PC3, dtype: float64
** Menos importante:
GRAPROES      -0.274927
GRAPROES_M    -0.272336
GRAPROES_F    -0.272066
Name: PC3, dtype: float64
============PC4================
** Más importante:
P15YM_SE_M      0.41251
P15YM_SE        0.40317
PAFIL_OTRAI     0.40187
Name: PC4, dtype: float64
** Menos importante:
PCDISC_MEN     -0.100449
PCDISC_VIS     -0.094782
PCDISC_LENG    -0.093049
Name: PC4, dtype: float64
============PC5================
** Más importante:
PCDISC_MEN      0.250725
PCDISC_MOT2     0.230696
PCDISC_AUD      0.216129
Name: PC5, dtype: float64
** Menos importante:
ENTIDAD       -0.171745
VIVPAR_DES    -0.145929
P_0A2         -0.111345
Name: PC5, dtype: float64
============PC6================
** Más importante:
P3HLINHE       0.429848
P3HLINHE_M     0.424994
P3HLINHE_F     0.423416
Name: PC6, dtype: float64
** Menos importante:
PROM_OCUP     -0.281461
GRAPROES_F    -0.263184
GRAPROES      -0.262702
Name: PC6, dtype: float64
============PC7================
** Más importante:
VPH_NDEAED     0.322533
VPH_S_ELEC     0.305707
VPH_SNBIEN     0.255704
Name: PC7, dtype: float64
** Menos importante:
PROM_OCUP     -0.256624
P3HLINHE_M    -0.186484
P3HLINHE      -0.186136
Name: PC7, dtype: float64
============PC8================
** Más importante:
PCLIM_MOT2      0.418149
PCLIM_HACO      0.394742
PCLIM_RE_CO     0.291773
Name: PC8, dtype: float64
** Menos importante:
PCDISC_MEN     -0.232103
PCDISC_LENG    -0.228140
PCDISC_MOT2    -0.223193
Name: PC8, dtype: float64
```

Imagen 7. Resultados del análisis PCA. *Elaboración propia*.

Cabe destacar que la vulnerabilidad puede verse reflejada en una calificación o *score* que nos permita distinguir cuáles son las zonas más afectadas por las variables elegidas en el proyecto. En la siguiente tabla se aprecia dicha calificación, lo cual nos permitió entender el comportamiento del fenómeno.

Score

In [47]:
```
vulnerable_df = df_cat_vul1.copy()

# Define the columns to check
columns_to_check = ['Vuln_Indigena_Discreta_Int', 'Vuln_Edad', 'Vuln_Disc_Lim',
                    'Vuln_Disc_Discreta_Int', 'Vuln_Esc_Salud', 'Vuln_Salud_Discreta_Int',
                    'Vuln_Hog_Serv', 'Vuln_Hog_Connect']

# Create a new column 'Vuln_Score' which sums 1 for each column that is not equal to 0
vulnerable_df['Vuln_Score'] = (vulnerable_df[columns_to_check] != 0).sum(axis=1)

# Display the updated DataFrame
vulnerable_df.head()
```

Out[47]:

...	VPH_PC	VPH_TELEF	VPH_CEL	VPH_INTER	VPH_SINRTV	VPH_SINLTC	VPH_SINCINT	VPH_SINTIC	Vuln_Hog_Connect	Vuln_Score
...	7	5	21	8	4	3	14	0	21.0	7
...	0	0	0	0	0	0	0	0	NaN	5
...	0	0	0	0	0	0	0	0	NaN	5
...	6	9	11	9	0	0	4	0	4.0	7
...	3	9	9	4	0	0	6	0	6.0	8

Imagen 8. Análisis de PCA posterior a la extracción. *Elaboración propia*.

- **Representación visual del resultado**

A continuación, se muestra el resultado obtenido del PCA, en la cual se puede observar la variabilidad a través de los componentes. Así como, la conformación de los componentes principales. Esta es una forma sencilla de entender el resultado arrojado que maximiza la variabilidad y disminuye la dimensionalidad; lo cual permite un uso más eficiente de los recursos.

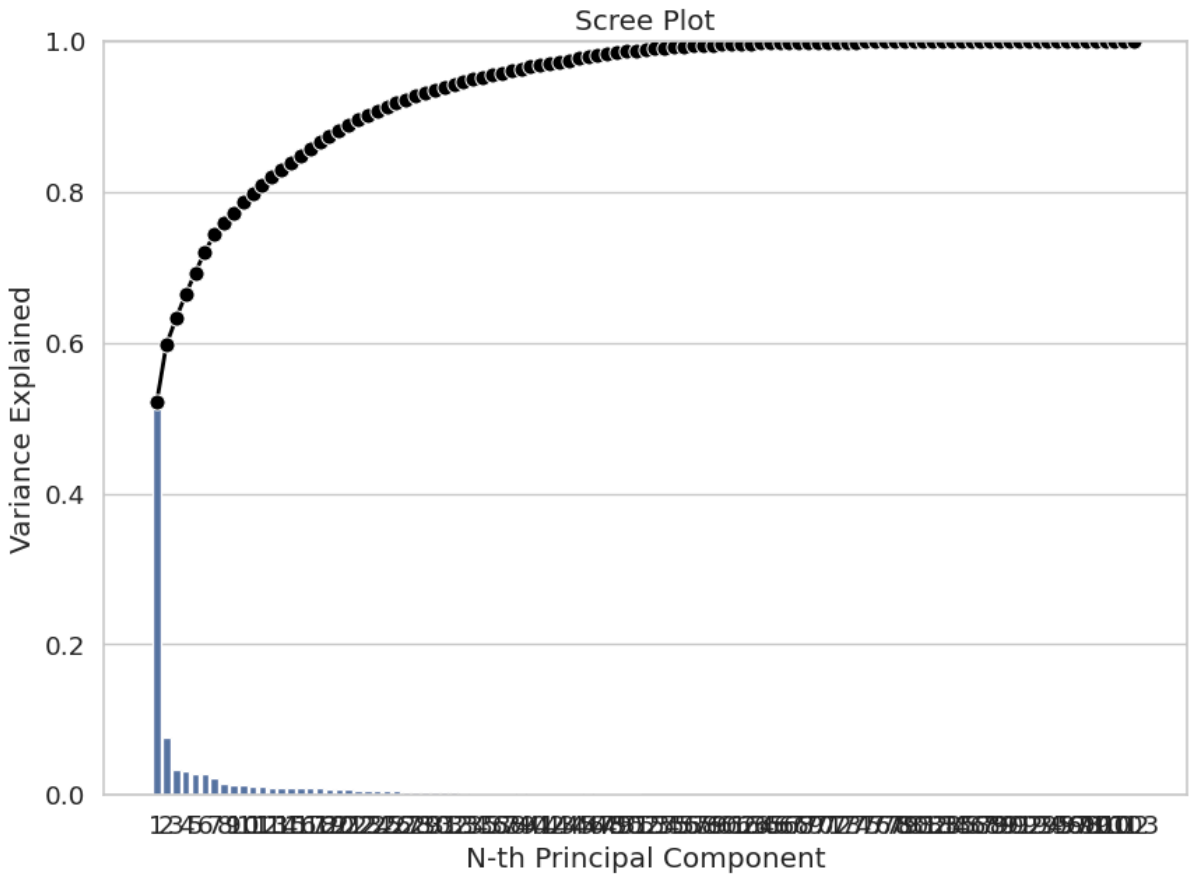

Imagen 9. Análisis de PCA posterior a la extracción. *Elaboración propia*.

3. Sub/sobre ajuste

En términos de sub o sobre entrenamiento del modelo, los resultados pueden ser relativamente menos evidentes en clasificación, contra lo que observamos en un modelo de regresión, sobre todo cuando contemplamos un entrenamiento supervisado, pues en dicho caso podemos sencillamente evaluar cómo se comporta el algoritmo con el conjunto de datos de entrenamiento, contra cómo lo hace con los de prueba y, por tanto determinar si efectivamente entendió los datos o simplemente

memorizó a los que tuvo acceso, teniendo un rendimiento inadecuado con datos nuevos.

- **Métricas de evaluación del modelo**

Sin embargo, en nuestro caso y dado el modelo k-means que seleccionamos, para corroborar que se tuviera un comportamiento adecuado, se evaluó la selección óptima de K empleando el puntaje de la silueta, que nos ayuda a determinar si un punto está adecuadamente asignado a su grupo, dada la distancia.

Por otro lado, como segundo método también empleamos el método del codo, que nos ayuda de igual forma a hacer una selección óptima de K o corroborar la cantidad de clusters a emplear. De esta forma, se observa cómo disminuye la inercia conforme aumentan los grupos, es decir, cómo cambia la suma de los cuadrados del punto contra el centroide, a través de diferentes cantidades de grupos.

- **Inspección gráfica del modelo**

El uso del método del codo nos permite conocer la cantidad de grupos que se pueden formar con nuestra base de datos. Una vez realizado el paso de selección y extracción de características, procedimos a calcular el valor de k. Esto será utilizado más adelante cuando se definan los parámetros del k-means.

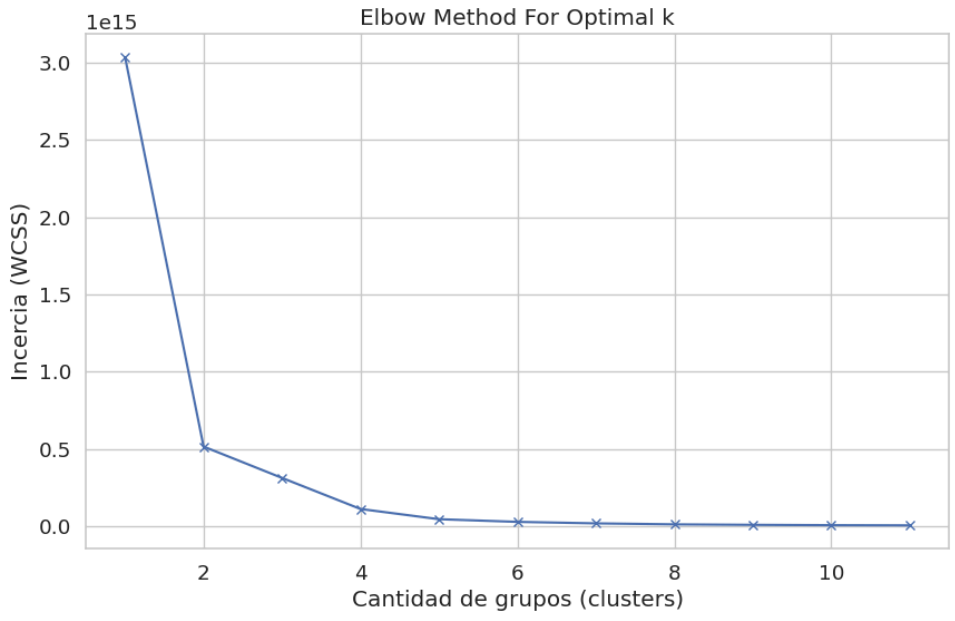

Imagen 10. Aplicación del método del codo. *Elaboración propia*.

En el resultado obtenido, observamos que la inercia disminuye casi en su totalidad

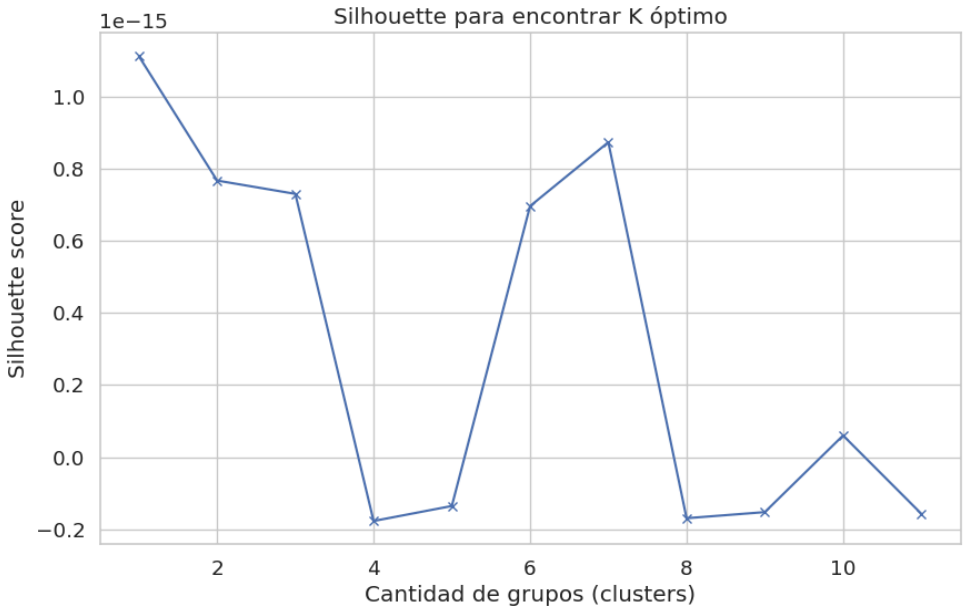

Imagen 11. Aplicación del método de silueta. *Elaboración propia*.

Al visualizar los resultados obtenidos para el método de codo y silueta, nos fue posible determinar un valor K óptimo igual a 4

4. Métrica

En esta sección hablamos de las métricas que nos ayudan a entender el comportamiento del modelo.

- **Alineación con el objetivo del problema y contexto**

Las métricas utilizadas para entender el comportamiento de k-means son las mismas que se emplean para determinar el valor óptimo de k. Es decir, la cantidad de clústeres que se manejarán con el algoritmo de clasificación. Cabe mencionar

que se está utilizando el método de silueta y de codo; igualmente, los datos procesados son poblacionales.

En nuestro caso particular, no se utilizan técnicas de desempeño convencionales como la matriz de confusión, ya que al no tratarse de un problema de regresión con entrenamiento supervisado, no hay contra qué valor contrastar los resultados obtenidos. Por lo cual sólo podemos corroborar que la selección de parámetros sea la correcta previa a entrenar al modelo.

- **Problemas específicos de los datos**

En nuestro caso particular, entender y ser capaces de visualizar el desbalance de datos es imperativo para una de las preguntas de negocio, puesto que, dentro de nuestros objetivos, es visualizar la población vulnerable por Estado y AGEB, de forma que se pueda establecer un rango de criticidad con base en los criterios seleccionados. Sin embargo, los problemas específicos de nuestro datos, se destacan los siguientes

```
[4] df_list_3e = [df_hidalgo,df_cdmx1,df_cdmx2,df_edomex1,df_edomex2,df_edomex3,df_edomex4]
    df_list_3e = pd.concat(df_list_3e, ignore_index=True)
    df_list_3e.head()
```

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2
0	13.0	Hidalgo	0.0	Total de la entidad	0000	0.0	3082841.0	1601462	1481379	134738
1	13.0	Hidalgo	0.0	Total del municipio	0000	0.0	22268.0	11563	10705	1241
2	13.0	Hidalgo	1.0	Total de la localidad urbana	0000	0.0	439.0	229	210	21
3	13.0	Hidalgo	1.0	Total AGEB urbana	0043	0.0	439.0	229	210	21
4	13.0	Hidalgo	1.0	Acatlán	0043	1.0	92.0	54	38	6

Imagen 12. Señalización de totalizadores. *Elaboración propia*.

Totalizadores indicados en la columna NOM_LOC, lo anterior fue tratado de la siguiente manera.

```
[7] df_cat1 = df3e.copy()

    df_cat1 = df_cat1[~df_cat1['NOM_LOC'].str.contains('Total|total', case=False, na=False)]
    df_cat1.head()
```

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VPH_RADIO
4	13.0	Hidalgo	1.0	Acatlán	0043	1.0	92.0	54	38	6	...	12
5	13.0	Hidalgo	1.0	Acatlán	0043	2.0	7.0	4	3	0	...	*
6	13.0	Hidalgo	1.0	Acatlán	0043	3.0	0.0	0	0	0	...	0
7	13.0	Hidalgo	1.0	Acatlán	0043	6.0	47.0	20	27	*	...	11
8	13.0	Hidalgo	1.0	Acatlán	0043	7.0	44.0	21	23	*	...	8

Imagen 13. Corroboración de eliminación de totalizadores. *Elaboración propia*.

Buscando dentro de dicha columna, aquellos valores con las palabras clave “Total” o “total”, lo que nos permitió eliminar las filas que contenían totales y alteraban el condensado de los datos. Posteriormente, fue posible eliminar la columna, puesto que la identificación se hacía con etiqueta numérica a través de “LOC”

Por otro lado, también se hicieron los pasos que en otras etapas se han descrito, respecto al tratamiento de los valores “*”, que no corresponden con datos numéricos relativos a la población, permitiendo convertir los datos de tipo objeto a entero, como era requerido para el análisis.

```
[57] df_cat_vul2 = df_cat2.copy()

     scaler = StandardScaler()
     df_cat_vul2 = pd.DataFrame(scaler.fit_transform(df_cat_vul2), columns=df_cat_vul2.columns)
     df_cat_vul2.head()
```

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F
0	-0.012754	0.0	-0.388157	-0.834597	-0.575405	-0.096278	-0.012323	-0.170730	0.417182	-0.444645
1	-0.012754	0.0	-0.388157	-0.834597	-0.551097	-0.665641	-0.676301	-0.628151	-0.568055	-0.444645
2	-0.012754	0.0	-0.388157	-0.834597	-0.526790	-0.712530	-0.729419	-0.667358	-0.568055	-0.444645
3	-0.012754	0.0	-0.388157	-0.834597	-0.453867	-0.397705	-0.463828	-0.314491	-0.568055	-0.444645
4	-0.012754	0.0	-0.388157	-0.834597	-0.429559	-0.417800	-0.450549	-0.366768	-0.568055	-0.444645

Imagen 14. Escalamiento. *Elaboración propia*.

Finalmente, se realizó un escalamiento para visualizar y entender de manera más adecuada la proporción de personas con características que las podrían colocar en una clasificación vulnerable o, en su defecto, que tendrían atenuantes para dicha condición. De esta manera, podríamos no perder de vista a las personas vulnerables en una región cuyo total poblacional fuese menor.

5. Desempeño

En esta sección ahondaremos en cuál es el desempeño mínimo para nuestro modelo. Es decir, determinaremos cuál es la base para tener un nivel aceptable más adelante en el proyecto. Conforme se revisó el valor óptimo de K por el método del codo, como también el de silueta, determinamos que la cantidad indicada de grupos sería igual a 4, que son justamente los que observamos en la gráfica.

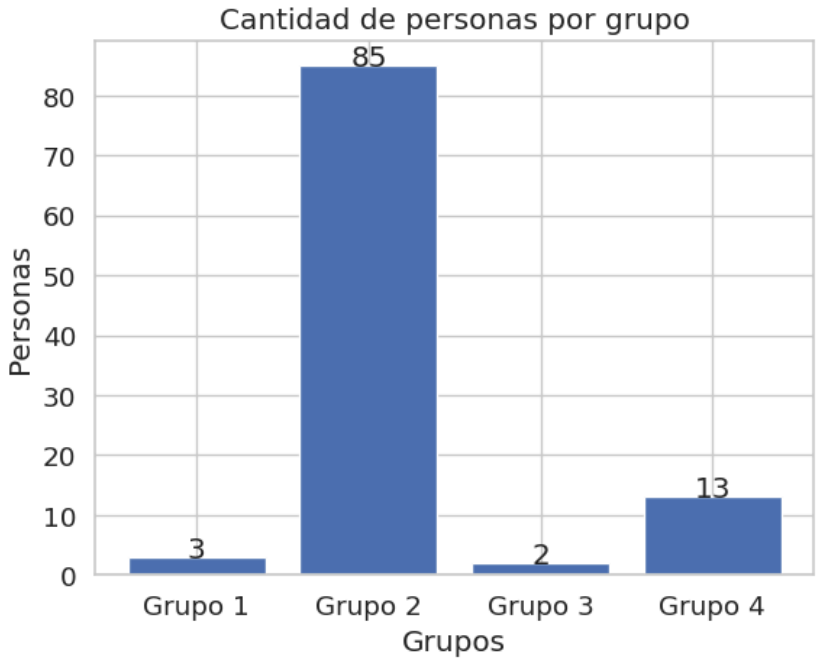

Imagen 15. Grupos resultado de Kmeans. *Elaboración propia*.

Por tanto, podemos también observar la distribución de los centroides en un plano X,Y, resaltados por figuras. Los mismos son los que corresponden a la cantidad de registros que se pueden observar en la captura siguiente.

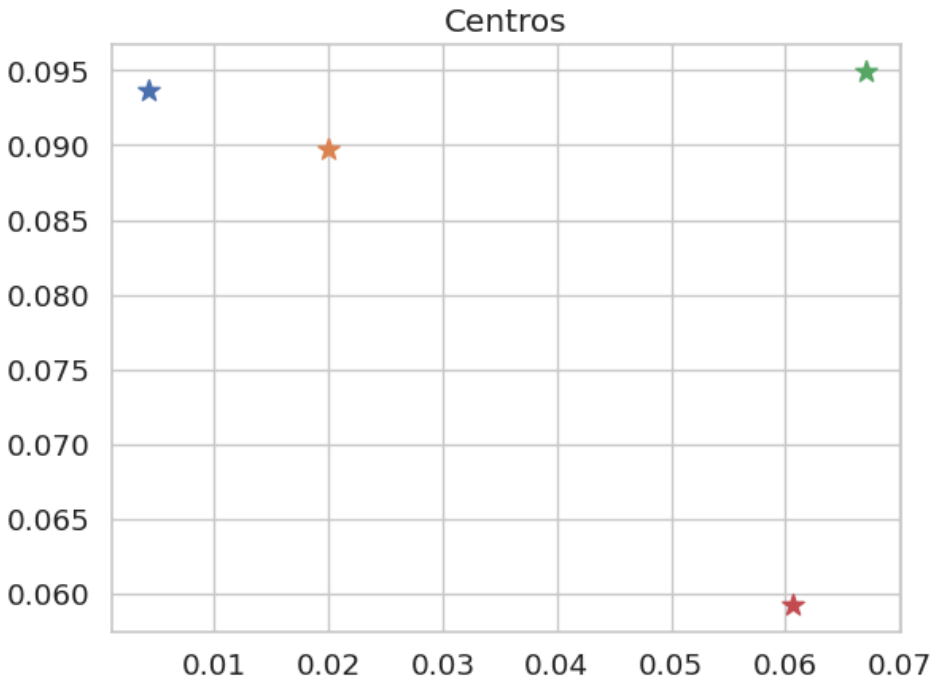

Imagen 16. Distribución de los grupos. *Elaboración propia*.

Adicionalmente, obtuvimos la inercia para el modelo elegido, la cual nos ayuda a medir qué tan bien nos ha realizado la agrupación de los puntos para cada clúster. Su resultado viene del cálculo de la suma de las distancias cuadradas entre cada punto y el centroide al cual pertenece. Es decir, posee una fórmula matemática para generar un resultado.

Cabe mencionar que un valor numérico alto de inercia sugiere que los puntos están dispersos fuera del clúster, lo cual indica que la formación de los grupos no es la adecuada. En su defecto, una inercia baja implica que los clústeres son más compactos, lo que es un resultado más deseable en el comportamiento. En la siguiente imagen se muestra el resultado para el algoritmo elegido.

```
In [18]: kmeans.inertia_

Out[18]: 98.99999999999989
```

Imagen 17. Cálculo de la inercia. *Elaboración propia*.

Como podemos visualizar, la paquetería nos ofrece una función para calcular de manera sencilla la inercia, sin tener que definir la fórmula.

6. Conclusiones

Durante el desarrollo de esta entrega, lógicamente retomamos el procesamiento realizado en las etapas previas, donde se exploraron los datos, generaron características e hicimos limpieza en general. A partir del tratamiento que se detalla en el desarrollo, fuimos capaces de definir una escala relativa al puntaje de vulnerabilidad, con base en las variables ya seleccionadas.

Por tanto, una vez que contábamos con un puntaje de vulnerabilidad de referencia, realizamos una escalación de los datos, para entender de manera más evidente la proporción de vulnerabilidad de cada región, independientemente del tamaño de la población. Así pues, se podrían reducir la dimensionalidad de forma posterior e iniciar con el set listo, nuestro algoritmo de clasificación, que en este caso fue Kmeans.

En cuanto a la evaluación de los resultados provenientes del modelo Kmeans, se emplearon las métricas asociadas al algoritmo, donde la inercia nos ayudó a medir qué tan bien se forman los clusters a partir de los datos. Además, el puntaje de la silueta nos ayudó a determinar qué tan similar es un punto para su propio clúster asignado, en contraste con los demás grupos.

Cabe destacar que el método del codo y silueta fueron empleados previos a determinar el valor óptimo de K o, expresado en otras palabras, cuántos clusters formamos. Por tanto, dicha verificación nos permitió tomar una decisión consciente en la cantidad de grupos que se seleccionaron, garantizando un comportamiento adecuado de la clasificación.

Finalmente, es este primer aproximamiento por medio del algoritmo Kmeans resulta pertinente a la pregunta que deseamos responder, porque nuestra necesidad en cuanto al set de datos SCITEL es segmental el concentrado de población con cierto grado de vulnerabilidad, con un factor de criticidad que permita definir las áreas de mayor interés para plantear acciones en materia de justicia social.

7. Referencias

Bech, J. (2019). Análisis Multivariado. Universidad Autónoma de Aguascalientes. ISBN 978-607-8652-68-6. https://editorial.uaa.mx/docs/analisis_multivariado.pdf

Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc." VanderPlas, J. Python data science handbook: Essential tools for working with data. " O'Reilly Media, Inc."

INEGI. (2020). Sistema de consulta de integración territorial (SCITEL). Principales resultados por AGEB y manzana urbana. INEGI. https://www.inegi.org.mx/app/scitel/Default?ev=10

INEGI. (s. f.). *Publicaciones y mapas*.

https://www.inegi.org.mx/app/biblioteca/ficha.html?upc=889463807469

Kumar Mukhiya, S., y Ahmed, U. (2020). Hands-On Exploratory Data Analysis with Python. Packt Publishing. https://learning.oreilly.com/library/view/hands-on-exploratory-data/9781789537253/0957090f-fa4d-4145-95dd-6d3782e5c04d.xhtml

Mas, J. (2019). Análisis univariante. Universitat Oberta de Catalunya. PID_00268326. https://openaccess.uoc.edu/bitstream/10609/148455/3/AnalisisUnivariante.pdf

Mahendry K., (2017, Junio), How to Determine the Optimal K for K-Means?, Medium, Recuperado en noviembre 2022 de https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

Ramírez, L. (2023). Algoritmo k-means: ¿Qué es y cómo funciona?. IEBS. https://www.iebschool.com/blog/algoritmo-k-means-que-es-y-como-funciona-big-data/

Studer, S., *et. al*. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. Preprints 2021, 1, 0. https://doi.org/10.48550/arXiv.2003.05155

Sham K., (2020, Junio), Find the best customer service centre location by using K-means clustering, medium, Recuperado en noviembre 2022 de https://medium.com/analytics-vidhya/find-the-best-customer-service-centre-location-by-using-k-means-clustering-fcc05eb7ab0f

Torre, J., *et. al.* (2023). Metodología para identificar y cuantificar islas de calor en entornos urbanos con imágenes satelitales. Centro para el Futuro de las Ciudades, Tecnológico de Monterrey. https://drive.google.com/drive/folders/1p-hPh6o_heBx-HAEKY1CsAioUi1XuRcS?hl=es

Visengeriyeva, L., Kammer, A., Bär, I., Kniesz, A., y Plöd, M. (2023). CRISP-ML(Q). The ML Lifecycle Process. MLOps. INNOQ. https://ml-ops.org/content/crisp-ml

8. Anexos

Anexo - Repositorio en GitHub