IslasDeCalor / EDA_SCITEL_INEGI_Avance_4.ipynb

LH-1169213 Creado con Colab c32c027 · 5 minutes ago

5.84 MB

Open in Colab

Proyecto - Islas de Calor y Justicia Social

Avance 4 - Ingeniería de características

Equipo 24:

Laura Elena Hernández Mata | A01169213

Evelyn Aylin Rendon Medina | A01748750

Samara García González | A01273001

Datos de la materia:

Proyecto integrador

TC5035.10

Asesor:

Roberto Ponce López

Tutora:

María de la Paz Rico Fernández

Fecha de entrega:

20 de octubre de 2024

Data Set Information:

El Sistema de Consulta de Integración Territorial (SCITEL) consta de un conjunto de 222 indicadores que permiten conocer las características de la población, los hogares censales y las viviendas para las AGEB y manzanas urbanas del país. De acuerdo con SCITEL de INEGI (2020), el conjunto de indicadores para conocer la estructura de la población Sexo y Edad, Fecundidad, Migración, Etnicidad, Discapacidad, Características Educativas, Características Económicas, Derechohabiencia a Servicios de Salud, Situación Conyugal y Religión; y también permite conocer el número de hogares y su población; así como descripción de la Vivienda, en las que destaca: Viviendas y Ocupantes, Material de pisos, Número de cuartos, Servicios de que disponen (Energía eléctrica, Agua entubada, Sanitario, Drenaje) y Bienes en la vivienda. Contiene un diccionario de datos que incluye la explicación de las variables, y también el nombre del indicador, su descripción, el mnemónico, los rangos de respuesta y la longitud

de cada campo.

Avance 1 (Entrega: 29 de septiembre)

Parte 1: Análisis exploratorio

Entender la pregunta de negocio, es decir, resumir el objeto de estudio a la respuesta que plantea es esencial para cualquier proyecto relacionado con datos. Sin embargo, es solo el primer paso del planteamiento, pues será indispensable conocer la materia prima que se dispone para lograrlo, es decir, el estado actual de nuestros datos en calidad, cantidad y en general, qué compone nuestro set que alimentará o entrenará el modelo.

Por tanto, es de suma relevancia para este proyecto tener un entendimiento profundo de nuestro conjunto de datos, es decir, conocer nuestras variables, cómo se comportan, qué valores atípicos poseen, cómo se distribuyen y la forma en que están sesgadas las clases. Solo a partir de lo anterior, de realizar un análisis exploratorio y definición de tratamiento de acuerdo con lo hallado, será posible cimentar la construcción de un modelo que permita responder la pregunta planteada.

Si bien lo anterior se define como esencial, es importante destacar que el análisis y la preparación de los datos es lo que más tiempo puede llevar en un proyecto de esta naturaleza, pues debemos tener certeza que la selección y tratamiento son los adecuados para el siguiente proceso que se ha planteado llevar a cabo, lo que será, por tanto, parte de un proceso iterativo de mejora; hasta ser capaces de alcanzar el comportamiento adecuado de nuestro modelo.

¿Se identifican tendencias temporales? (En caso de que el conjunto incluya una dimensión de tiempo).

En nuestro caso de análisis particular no se analiza ninguna variable en función del tiempo, puesto que, si bien para obtener el resultado de temperatura promedio para las imágenes de las islas de calor, se analizan muestras alrededor de un año, 2022, el resultado es una imagen o fotografía estática que emplea el promedio con un intervalo de confianza, sin variación conforme el tiempo. Por otro lado, los datos provenientes del INEGI también corresponden al estudio realizado en un año particular, en este caso, 2020.

Parte 2: Estructura de los datos

En esta sección se encuentra lo relacionado con la estructura y análisis de los datos, así como la identificación de las variables faltantes. Esta parte de la entrega tiene la finalidad de contener los aspectos básicos previos al análisis un/bi/multivariable y el procesamiento de la información del proyecto.

2.1 Descripción general de la forma y los tipos de datos

A continuación explicaremos la forma en que están compuestos nuestros datos, tanto los relativos a las islas de calor, como también los datos obtenidos del INEGI.

1. Dataset de Imágenes Raster de Islas de Calor en México

Forma: Consta de imágenes en formato raster, donde cada píxel representa un valor de temperatura (en grados Celsius) correspondiente a diferentes áreas geográficas. Así mismo una segunda sección de imágenes raster pero enfocadas a zonas rurales. Las imágenes están georreferenciadas, lo que permite su ubicación precisa en un sistema de coordenadas geográficas.

Tipos de Datos:

- **Valor de Temperatura:** Datos continuos que representan las temperaturas en diferentes áreas.
- **SCITEL:** Primordialmente datos numéricos, correspondientes a las variables de la población, en particular, las que consideramos
- **Clave de entidad federativa:** Código que identifica a la entidad federativa. El código 00 identifica a los registros con los totales a nivel nacional.
- **Entidad federativa:** Nombre oficial de la entidad federativa.
- **Clave de municipio o demarcación territorial:** Código que identifica al municipio o demarcación territorial al interior de una entidad federativa, conforme al Marco Geoestadístico. El código 000 identifica a los registros con los totales a nivel de entidad federativa.
- **Municipio o demarcación territorial:** Nombre oficial del municipio o demarcación territorial en el caso de la Ciudad de México.
- **Clave de localidad:** Código que identifica a la localidad al interior de cada municipio o demarcación territorial conforme al Marco Geoestadístico. El código 0000 identifica a los registros con los totales a nivel de municipio o demarcación territorial.
- **Localidad:** Nombre con el que se reconoce a la localidad dado por la ley o la costumbre.
- **Clave del AGEB:** Clave que identifica al AGEB urbana, al interior de una localidad, de acuerdo con la desagregación del Marco Geoestadístico.
- **Clave de manzana:** Clave que identifica a la manzana, al interior de una AGEB, de acuerdo a la desagregación del Marco Geoestadístico.
- **Población total:** Total de personas que residen habitualmente en el país, la entidad federativa, el municipio o la demarcación territorial y la localidad. Incluye la estimación del número de personas en viviendas particulares sin información de ocupantes. Incluye a la población que no especificó su edad.
- **Población femenina:** Total de mujeres que residen habitualmente en el país, la entidad federativa, el municipio o la demarcación territorial y la localidad. Incluye la estimación del número de mujeres en viviendas particulares sin información de ocupantes. Incluye a la población que no especificó su edad.
- **Población masculina:** Total de hombres que residen habitualmente en el país, la entidad federativa, el municipio o la demarcación territorial y la localidad. Incluye la estimación del número de hombres en viviendas particulares sin información de ocupantes. Incluye a la población que no especificó su edad.

2. Base de Datos del INEGI (Censo de Población y Vivienda 2020 - SCITEL)

A continuación mostramos un resumen de los datos en forma tabular.

Bases de datos para el proyecto			

Nombre de base de datos	SCITEL (INEGI, 2020)	Islas de calor (centígrados)	Islas de calor (categorizado)
Tipo de datos	Estructurados	Sin estructurar	Estructurados
Formato	xlsx	tif	dataframe
Fuente de información	Censo del INEGI	Información satelital provista para el proyecto	Información satelital previamente procesada para la realización del proyecto

Tabla 1. Bases de datos para el proyecto. (*Elaboración propia, 2024*).

Forma: Estructurada en formato Excel, con filas y columnas que organizan la información de manera tabular. Cada fila representa un municipio o una unidad geográfica (AGEB), y cada columna representa diferentes variables e indicadores.

Tipos de Datos:

- **Municipio:** Identificador categórico que indica el nombre o código del municipio.
- **Estado:** Categórico, indicando a qué estado pertenece cada municipio.
- **AGEB (Área Geoestadística Básica):** Identificador categórico que clasifica áreas para fines estadísticos.
- **Población Total:** Datos numéricos que representan el número total de habitantes en cada municipio.
- **Indicadores Socioeconómicos:** Incluyen variables como el ingreso promedio, el nivel de educación, la vivienda, el acceso a servicios básicos, entre otros.

¿Existen distribuciones sesgadas en el conjunto de datos? ¿Necesitamos aplicar alguna transformación no lineal?

De primera instancia, sabemos que se tiene sesgo en la distribución de los datos relativos a las temperaturas encontradas en el análisis. Lo anterior puede jugarnos en contra al generar clasificaciones, dependiendo de la sensibilidad del modelo seleccionado, el tratamiento específico que se le brinden a los datos y los hiperparámetros.

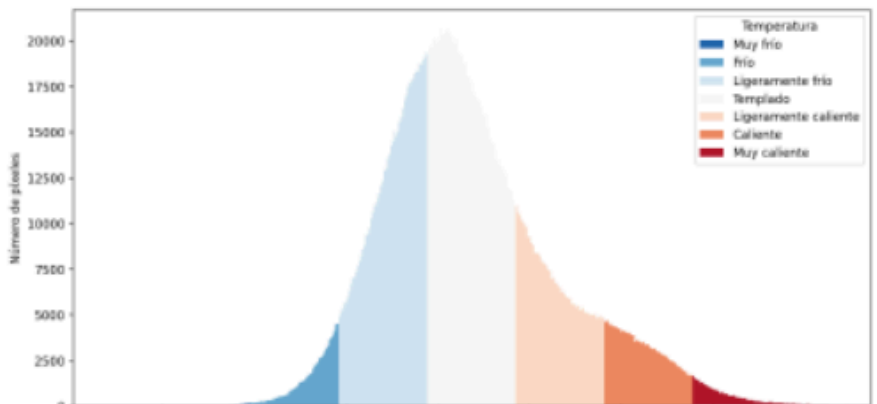


Imagen 1. Distribución de temperatura basadas en las estadísticas de la imagen. *(Torre, et. al. 2023)*

Sin embargo, es importante entender y asociar este desbalanceo de clases, porque es precisamente la búsqueda de las zonas intensas de calor lo que más nos interesa, es decir, la vulnerabilidad contra el calor y, como podemos observar, las zonas de mayor calor son las áreas con menor cantidad de observaciones.

Así pues, en este primer análisis exploratorio se observa dicho comportamiento, pero aún es necesario evaluar cómo afectará al modelo de clasificación, pues el tratamiento es también un proceso iterativo y adaptativo.

¿Se deberían normalizar las imágenes para visualizarlas mejor?

Sí, se planea realizar una normalización de las imágenes del dataset , el método considerado para esto es la normalización por percentiles , ya que nos permite escalar los valores de temperatura usando los percentiles 1 y 99 y disminuir los valores extremos o atípicos . Además es importante recalcar que se deben redimensionar las imágenes a un solo tamaño, ya que actualmente vienen en diferentes tamaños y esto dificulta su análisis y comparación visual.

¿Hay desequilibrio en las clases de la variable objetivo?

Sí, como se observa en la gráfica superior (Torre, et. al. 2023), el enfoque en las zonas de calor intenso revela que estas áreas, que representan mayor vulnerabilidad, tienen menos observaciones. Este desbalance puede afectar el rendimiento del modelo por lo que se debe evaluar su impacto y considerar técnicas de ajuste.

2.2 Estadísticas descriptivas para las variables del conjunto

Importación y visualización Datos de Hidalgo en Python

1.- Importamos las librerías necesarias

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.- Cargamos en la variable df el archivo csv e indicamos con cuales columnas estaremos trabajando, al final mostramos los primero 5 registros

In [ ]:
```
#No mover_no correr
#Hidalgo
#Contiene todas las columnas
#df = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                 #usecols=['ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', 'POBMAS',
           # 'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           # 'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_
           # 'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT
          # 'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE',
```

```
            #'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'P
          #  'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPI
          #  'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB
          #  'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUA
          # 'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ
          # 'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFI
          #  'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL
          # 'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC'])
#df.head()
```

In [ ]:
```
# Importación de datos de las columnas de interés
# Hidalgo
df = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/mair
                 usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
            'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
            'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG
0	0000	3082841	1601462	1481379	134738	156486	383675	264746	28497	58
1	0000	22268	11563	10705	1241	1239	2625	1847	0	
2	0000	439	229	210	21	18	64	53	0	
3	0043	439	229	210	21	18	64	53	0	
4	0043	92	54	38	6	5	11	10	0	

5 rows × 22 columns

4.- Reemplazamos los caracteres tipo * que no aportan a la BD

In [ ]:
```
df.replace('*', 0, inplace=True)
df.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG
0	0000	3082841	1601462	1481379	134738	156486	383675	264746	28497	58
1	0000	22268	11563	10705	1241	1239	2625	1847	0	
2	0000	439	229	210	21	18	64	53	0	
3	0043	439	229	210	21	18	64	53	0	
4	0043	92	54	38	6	5	11	10	0	

5 rows × 22 columns

5.- Observamos el tipo de dato por columna

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42088 entries, 0 to 42087
Data columns (total 22 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   AGEB       42088 non-null  object
 1   POBTOT     42088 non-null  int64 
 2   POBFEM     42088 non-null  object
 3   POBMAS     42088 non-null  object
 4   P_0A2      42088 non-null  object
 5   P_3A5      42088 non-null  object
 6   P_60YMAS   42088 non-null  object
 7   POB65_MAS  42088 non-null  object
 8   P3HLINHE   42088 non-null  object
 9   PHOG_IND   42088 non-null  object
 10  PCON_DISC  42088 non-null  object
 11  PCON_LIMI  42088 non-null  object
 12  PSIND_LIM  42088 non-null  object
 13  P15YM_AN   42088 non-null  object
 14  P15YM_SE   42088 non-null  object
 15  GRAPROES   42088 non-null  object
 16  PEA        42088 non-null  object
 17  PSINDER    42088 non-null  object
 18  PDER_SS    42088 non-null  object
 19  TOTHOG     42088 non-null  object
 20  POBHOG     42088 non-null  object
 21  VIVTOT     42088 non-null  int64 
dtypes: int64(2), object(20)
memory usage: 7.1+ MB
```

5.1- Notamos que los datos están como objetos, por lo que hacemos un cast

In [ ]:
```
df_int= df.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)

df_int.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42088 entries, 0 to 42087
Data columns (total 22 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   AGEB       42088 non-null  int64
 1   POBTOT     42088 non-null  int64
 2   POBFEM     42088 non-null  int64
 3   POBMAS     42088 non-null  int64
 4   P_0A2      42088 non-null  int64
 5   P_3A5      42088 non-null  int64
 6   P_60YMAS   42088 non-null  int64
 7   POB65_MAS  42088 non-null  int64
 8   P3HLINHE   42088 non-null  int64
 9   PHOG_IND   42088 non-null  int64
 10  PCON_DISC  42088 non-null  int64
 11  PCON_LIMI  42088 non-null  int64
 12  PSIND_LIM  42088 non-null  int64
 13  P15YM_AN   42088 non-null  int64
 14  P15YM_SE   42088 non-null  int64
 15  GRAPROES   42088 non-null  int64
 16  PEA        42088 non-null  int64
 17  PSINDER    42088 non-null  int64
 18  PDER_SS    42088 non-null  int64
 19  TOTHOG     42088 non-null  int64
 20  POBHOG     42088 non-null  int64
 21  VIVTOT     42088 non-null  int64
```

```
dtypes: int64(22)
memory usage: 7.1 MB
```

5.2 -El método isnull().values.any(), puede verificar si un DataFrame de Pandas contiene valores NaN/None en cualquier celda (todas las filas y columnas). Este método devuelve True si encuentra NaN/None en cualquier celda de un DataFrame, devuelve False cuando no se encuentra

In [ ]:
```
df_int.isnull().values.any()
```

Out[ ]:
```
False
```

6.-El método df.isnull().any() muestra si una columna tiene o no tiene valores nulos

In [ ]:
```
df_int.isnull().any()
```

Out[ ]:

	0
AGEB	False
POBTOT	False
POBFEM	False
POBMAS	False
P_0A2	False
P_3A5	False
P_60YMAS	False
POB65_MAS	False
P3HLINHE	False
PHOG_IND	False
PCON_DISC	False
PCON_LIMI	False
PSIND_LIM	False
P15YM_AN	False
P15YM_SE	False
GRAPROES	False
PEA	False
PSINDER	False
PDER_SS	False
TOTHOG	False
POBHOG	False
VIVTOT	False

dtype: bool

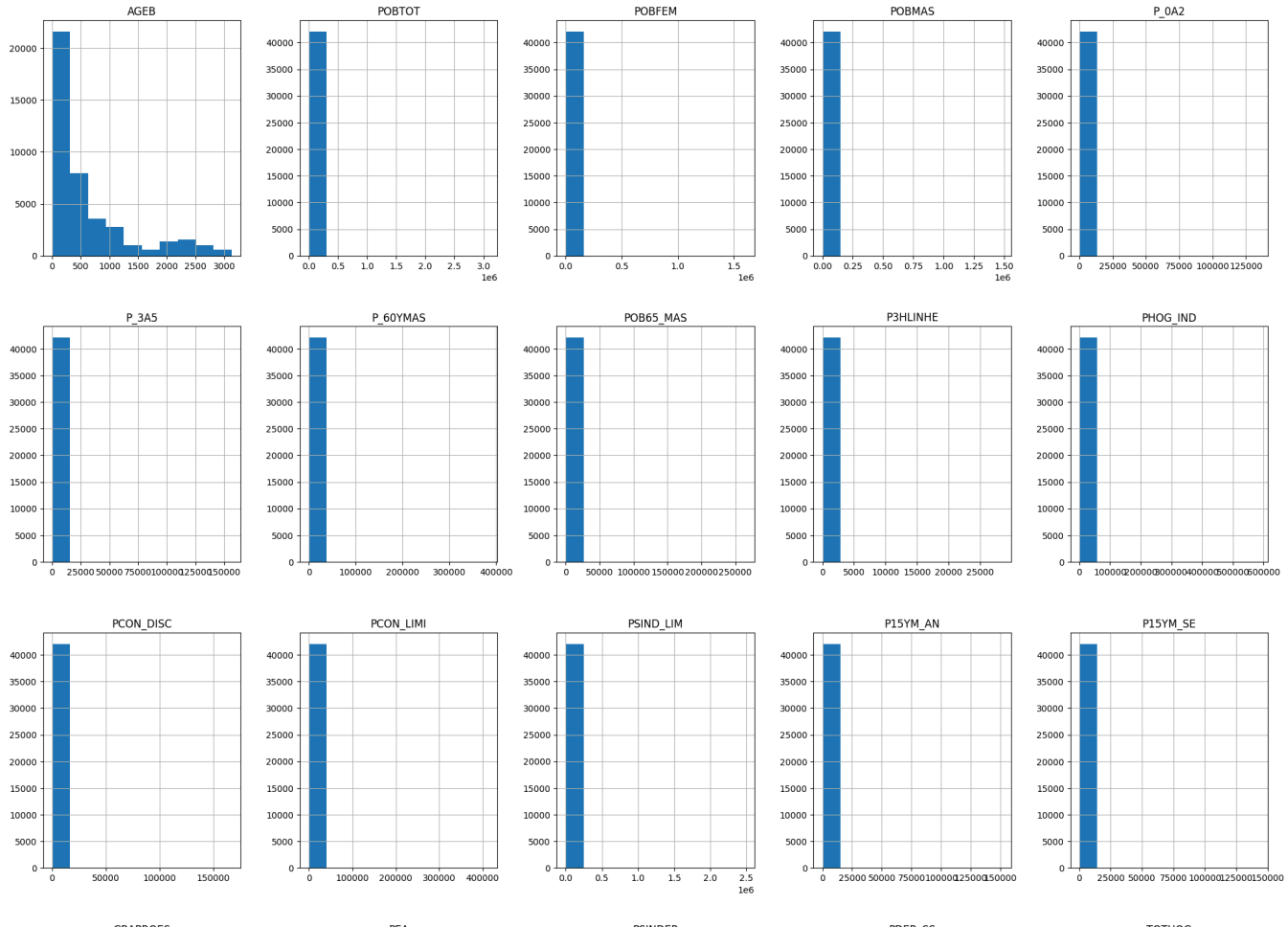
dtype: bool

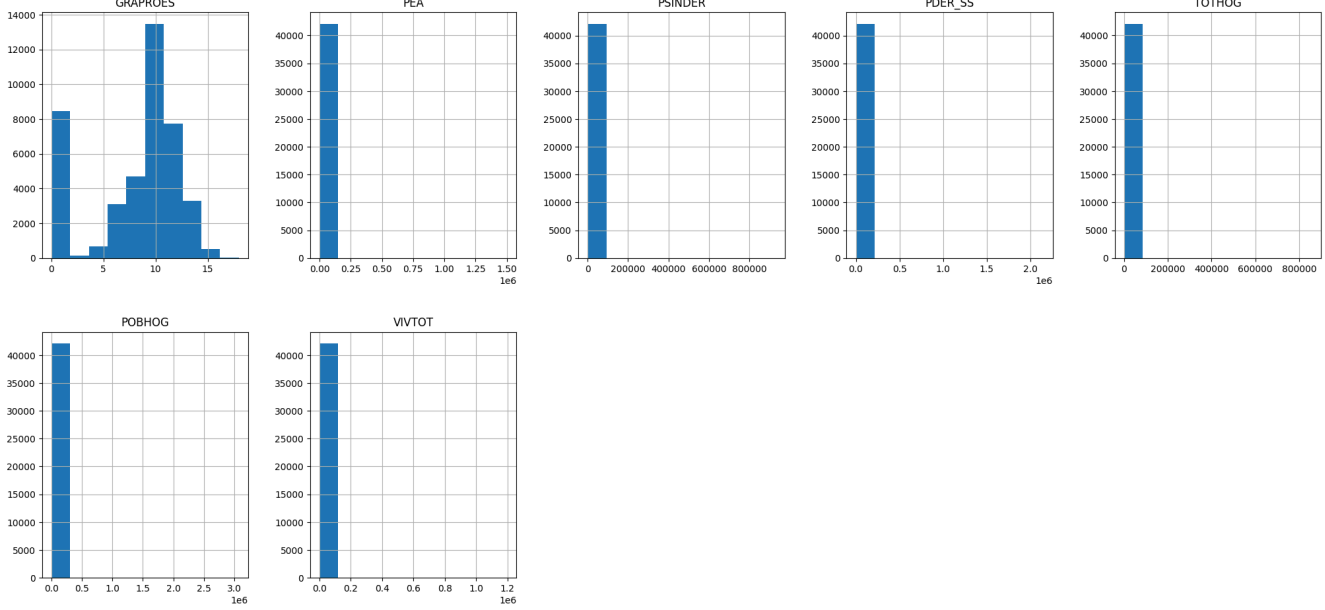
7.- Observamos la distrubución de cada columna o característica

In [ ]:
```
fig = plt.figure(figsize = (25,30))
ax = fig.gca()
df_int.hist(ax = ax)
```

```
<ipython-input-9-6820ffd6d359>:3: UserWarning: To output multiple subplots, the figure containing the passed axes is being cleared.
  df_int.hist(ax = ax)
```

Out[ ]:
```
array([[<Axes: title={'center': 'AGEB'}>,
        <Axes: title={'center': 'POBTOT'}>,
        <Axes: title={'center': 'POBFEM'}>,
        <Axes: title={'center': 'POBMAS'}>,
        <Axes: title={'center': 'P_0A2'}>],
       [<Axes: title={'center': 'P_3A5'}>,
        <Axes: title={'center': 'P_60YMAS'}>,
        <Axes: title={'center': 'POB65_MAS'}>,
        <Axes: title={'center': 'P3HLINHE'}>,
        <Axes: title={'center': 'PHOG_IND'}>],
       [<Axes: title={'center': 'PCON_DISC'}>,
        <Axes: title={'center': 'PCON_LIMI'}>,
        <Axes: title={'center': 'PSIND_LIM'}>,
        <Axes: title={'center': 'P15YM_AN'}>,
        <Axes: title={'center': 'P15YM_SE'}>],
       [<Axes: title={'center': 'GRAPROES'}>,
        <Axes: title={'center': 'PEA'}>,
        <Axes: title={'center': 'PSINDER'}>,
        <Axes: title={'center': 'PDER_SS'}>,
        <Axes: title={'center': 'TOTHOG'}>],
       [<Axes: title={'center': 'POBHOG'}>,
        <Axes: title={'center': 'VIVTOT'}>, <Axes: >, <Axes: >, <Axes: >]],
      dtype=object)
```





Importación de Datos de cada Estado en Python

Leemos los dataset de cada Estado

Hidalgo

In [ ]:
```
# Leemos los datos por Estado
# Hidalgo
df1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
            'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
            'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df1.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG
0	0000	3082841	1601462	1481379	134738	156486	383675	264746	28497	58
1	0000	22268	11563	10705	1241	1239	2625	1847	0	
2	0000	439	229	210	21	18	64	53	0	
3	0043	439	229	210	21	18	64	53	0	
4	0043	92	54	38	6	5	11	10	0	

5 rows × 22 columns

CDMX

In [ ]:
```
# Leemos los datos por Estado
# CDMX 1
df2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
            'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
            'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df2.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	9209944.0	4805017	4404927	267151	324942	1491619	1022105	1032	2
1	0000	432205.0	227255	204950	11784	14191	78650	54863	21	
2	0000	432205.0	227255	204950	11784	14191	78650	54863	21	
3	0010	3183.0	1695	1488	60	73	816	671	0	
4	0010	159.0	86	73	*	*	43	37	0	

5 rows × 22 columns

In [ ]:

```
# Leemos los datos por Estado
# CDMX 2
df3 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
             'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
             'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df3.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_I
0	5179	42	19	23	*	*	7	6	0	
1	5179	27	15	12	3	0	4	4	0	
2	5179	23	13	10	*	0	*	*	0	
3	5179	23	10	13	0	*	0	0	0	
4	5179	5	*	*	*	*	*	*	*	

5 rows × 22 columns

Estado de México

In [ ]:

```
# Leemos los datos por Estado
# EdoMex 1
df4 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
             'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
             'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df4.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0	16992418	8741123	8251295	724133	834611	1919454	1258354	5422	10
1	0	67872	35255	32617	3782	3985	7502	5303	121	
2	0	5988	3148	2840	300	308	674	437	0	
3	127	3373	1796	1577	180	175	463	301	0	

4	127	0	0	0	0	0	0	0	0

5 rows × 22 columns

In [ ]:
```
# Leemos los datos por Estado
# EdoMex 2
df5 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
              'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
              'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df5.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_I
0	2502	58	31	27	3	5	3	*	0	
1	2502	43	22	21	3	*	5	4	*	
2	2502	55	28	27	0	*	11	6	0	
3	2502	38	24	14	*	3	4	*	0	
4	2502	49	31	18	*	*	10	9	0	

5 rows × 22 columns

In [ ]:
```
# Leemos los datos por Estado
# EdoMex 3
df6 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
              'P_0A2','P_3A5', 'P_60YMAS', 'POB65_MAS', 'P3HLINHE', 'PHOG_IND', 'PCON_DISC',
              'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df6.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_I
0	132	191	104	87	8	6	33	28	0	
1	132	27	16	11	*	*	3	3	0	
2	132	83	39	44	3	5	7	3	0	
3	132	118	63	55	3	7	5	3	0	
4	132	116	56	60	4	4	3	*	0	

5 rows × 22 columns

In [ ]:
```
# Leemos los datos por Estado
# EdoMex 4
df7 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heads/ma
                  usecols=[ 'AGEB', 'POBTOT', 'POBFEM', 'POBMAS',
```

```
            'PSIND_LIM', 'P15YM_AN', 'P15YM_SE', 'GRAPROES', 'PEA', 'PSINDER', 'PDER_SS',
df7.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_I
0	453	24	14	10	0	0	4	4	0	
1	453	52	35	17	3	*	15	13	0	
2	453	0	0	0	0	0	0	0	0	
3	453	30	15	15	0	0	8	8	0	
4	453	11	4	7	0	0	5	3	0	

5 rows × 22 columns

Concatenación de los df

In [ ]:

```
dataframes_list = []

for i in range(1, 7):  # Loop from 1 to 7 df
    df_name = f'df{i}'  # Create the DataFrame name as a string
    dataframes_list.append(eval(df_name))  # Append the DataFrame to the list

# Print the list of DataFrames
print(dataframes_list)
```

```
[       AGEB   POBTOT   POBFEM   POBMAS   P_0A2   P_3A5 P_60YMAS POB65_MAS  \
0      0000  3082841  1601462  1481379  134738  156486   383675    264746
1      0000    22268    11563    10705    1241    1239     2625      1847
2      0000      439      229      210      21      18       64        53
3      0043      439      229      210      21      18       64        53
4      0043       92       54       38       6       5       11        10
...     ...      ...      ...      ...     ...     ...      ...       ...
42083  0313      346      190      156      14      20        8         *
42084  0313      102       58       44       4       3        0         0
42085  0313       68       32       36       *       4        *         0
42086  0313       72       41       31       3       5        *         *
42087  0313      104       59       45       6       8        5         *

      P3HLINHE PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES      PEA  \
0        28497   584693  ...   2482249   151311   143099     9.37  1502364
1            0      203  ...     18608     1755     1894     7.35    11462
2            0       12  ...       300       10       14     9.48      221
3            0       12  ...       300       10       14     9.48      221
4            0        0  ...        62        *        3     9.31       43
...        ...      ...  ...       ...      ...      ...      ...      ...
42083        0       88  ...       316        5        7    10.05      135
42084        0       32  ...        92        *        *    10.41       41
42085        0       15  ...        63        0        0    10.47       30
42086        0        8  ...        64        0        *    10.06       33
42087        0       33  ...        97        4        4     9.35       31

      PSINDER  PDER_SS  TOTHOG   POBHOG   VIVTOT
0      928550  2149373  857174  3075177  1198017
1        9798    12464    5869    22267     8021
2         227      212     130      439      161
3         227      212     130      439      161
```

```
4          47       45       25       92       27
...       ...      ...      ...      ...      ...
42083     115      231       91      346      114
42084      26       76       25      102       28
42085      26       42       20       68       25
42086      19       53       22       72       32
42087      44       60       24      104       29

[42088 rows x 22 columns],        AGEB     POBTOT   POBFEM   POBMAS   P_0A2   P_3A5 P_60YMAS
POB65_MAS  \
0      0000  9209944.0  4805017  4404927  267151  324942  1491619   1022105
1      0000   432205.0   227255   204950   11784   14191    78650     54863
2      0000   432205.0   227255   204950   11784   14191    78650     54863
3      0010     3183.0     1695     1488      60      73      816       671
4      0010      159.0       86       73       *       *       43        37
...     ...        ...      ...      ...     ...     ...      ...       ...
34477  5179       44.0       25       19       *       *        6         3
34478  5179       35.0       22       13       4       *        *         0
34479  5179       64.0       41       23       *       *        9         8
34480  5179       72.0       40       32       *       0       11         5
34481   NaN        NaN      NaN      NaN     NaN     NaN      NaN       NaN

      P3HLINHE PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES      PEA  \
0         1032   289139  ...   7489519   107444   162484    11.48  5099957
1           21     7261  ...    348220     3675     4863    11.91   239069
2           21     7261  ...    348220     3675     4863    11.91   239069
3            0       16  ...      2619       18       29    11.52     1582
4            0        0  ...       125        4        5    11.75       81
...        ...      ...  ...       ...      ...      ...      ...      ...
34477        0        0  ...        27        0        0    10.83       21
34478        0        0  ...        31        0        0    12.22       22
34479        0       13  ...        50        3        3    10.62       34
34480        0        4  ...        60        *        *     9.25       34
34481      NaN      NaN  ...       NaN      NaN      NaN      NaN      NaN

       PSINDER  PDER_SS   TOTHOG   POBHOG     VIVTOT
0      2502789  6689012  2756319  9159392  3036239.0
1        90370   341302   134168   431347   149381.0
2        90370   341302   134168   431347   149381.0
3          603     2580      867     3183      894.0
4           27      132       48      159       49.0
...        ...      ...      ...      ...        ...
34477       20       24       17       44       20.0
34478       16       19       10       35       10.0
34479       28       36       15       64       15.0
34480       40       32       16       72       17.0
34481      NaN      NaN      NaN      NaN        NaN

[34482 rows x 22 columns],        AGEB  POBTOT POBFEM POBMAS P_0A2 P_3A5 P_60YMAS POB65_MAS P
3HLINHE  \
0      5179      42     19     23     *     *        7         6        0
1      5179      27     15     12     3     0        4         4        0
2      5179      23     13     10     *     0        *         *        0
3      5179      23     10     13     0     *        0         0        0
4      5179       5      *      *     *     *        *         *        *
...     ...     ...    ...    ...   ...   ...      ...       ...      ...
34455  1524     106     58     48     *     4       25        12        0
34456  1524     107     55     52     *     *       27        17        0
34457  1524     246    116    130    13     8       27        15        0
34458  1524     158     78     80     *     5       40        22        0
34459  1524     474    247    227     9    25       69        45        0

      PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES  PEA PSINDER PDER_SS  \
0            6  ...        35        0        0     9.91   26      16      26
1            3  ...        22        *        *     9.77   20      13      14
```

```
2           3  ...        17        0        0     8.88    8       18       5
3           *  ...        16        0        0     8.31    8       14       9
4           *  ...         *        *        *        *    *        *       *
...       ...  ...       ...      ...      ...      ...  ...      ...     ...
34455       5  ...        81        *        *    10.18   58       21      85
34456       0  ...        99        0        *     9.76   70       17      90
34457       0  ...       183       10       13    10.12  130       91     155
34458       4  ...       107        *        0    10.89   77       45     112
34459      45  ...       359        7        7    10.34  270       93     381

      TOTHOG POBHOG VIVTOT
0         11     42     11
1          7     27      8
2          5     23      6
3          6     23      6
4          *      *      1
...      ...    ...    ...
34455     31    106     32
34456     40    107     42
34457     73    246     76
34458     47    158     48
34459    130    474    137

[34460 rows x 22 columns],        AGEB    POBTOT   POBFEM   POBMAS   P_0A2   P_3A5 P_60YMAS P
OB65_MAS  \
0         0  16992418  8741123  8251295  724133  834611  1919454   1258354
1         0     67872    35255    32617    3782    3985     7502      5303
2         0      5988     3148     2840     300     308      674       437
3       127      3373     1796     1577     180     175      463       301
4       127         0        0        0       0       0        0         0
...     ...       ...      ...      ...     ...     ...      ...       ...
34756  2502        22       10       12       *       *        3         *
34757  2502        31       14       17       0       3        3         *
34758  2502        42       21       21       *       *        7         *
34759  2502       100       44       56       3       4       13         4
34760  2502        31       15       16       *       *        8         4

      P3HLINHE PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES      PEA  \
0         5422  1026540  ...  14186323   372454   452522    10.08  8544416
1          121    19713  ...     52686     3791     3726      8.3    35834
2            0      297  ...      4871       81       92    11.63     3117
3            0      162  ...      2786       41       51    11.86     1749
4            0        0  ...         0        0        0        0        0
...        ...      ...  ...       ...      ...      ...      ...      ...
34756        0        0  ...        14        0        0    10.94       10
34757        0        4  ...        27        0        0    11.86       14
34758        0        0  ...        34        *        *      9.5       21
34759        0        0  ...        80        *        *    10.92       49
34760        0        0  ...        28        0        *    11.15       21

       PSINDER   PDER_SS   TOTHOG    POBHOG   VIVTOT
0      5672574  11267677  4568635  16943520  5403492
1        20961     46861    17393     67872    23788
2         1988      3997     1583      5988     2078
3         1104      2269      921      3373     1172
4            0         0        0         0        0
...        ...       ...      ...       ...      ...
34756        0        22        7        22       10
34757        4        27        9        31       20
34758       21        21       11        42       21
34759       39        53       37       100       41
34760        9        22       11        31       18

[34761 rows x 22 columns],        AGEB  POBTOT POBFEM POBMAS P_0A2 P_3A5 P_60YMAS POB65_MAS P
3HLINHE  \
```

```
HLINHE  \
0      2502      58     31     27     3     5        3         *        0
1      2502      43     22     21     3     *        5         4        *
2      2502      55     28     27     0     *       11         6        0
3      2502      38     24     14     *     3        4         *        0
4      2502      49     31     18     *     *       10         9        0
...     ...     ...    ...    ...   ...   ...      ...       ...      ...
40307   132      20      7     13     *     *        3         3        0
40308   132      94     48     46     4     0       10         4        0
40309   132       0      0      0     0     0        0         0        0
40310   132      32     16     16     3     *        0         0        0
40311   132      20     11      9     3     *        *         *        0

      PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES PEA PSINDER PDER_SS  \
0            0  ...        42        0        0    11.13  30      25      32
1            6  ...        34        0        0     9.63  19      12      31
2            0  ...        41        0        *    10.31  28      13      42
3            0  ...        28        0        0    11.81  23       5      33
4            0  ...        32        *        *     9.68  27       5      44
...        ...  ...       ...      ...      ...      ...  ..     ...     ...
40307        0  ...        11        *        *     8.69  10      17       3
40308        0  ...        84        *        3     8.96  41      62      32
40309        0  ...         0        0        0        0   0       0       0
40310       20  ...        29        *        *      7.1  17      26       6
40311        5  ...        15        0        0     8.54   6       9      11

      TOTHOG POBHOG VIVTOT
0         17     58     20
1         10     43     20
2         16     55     20
3         12     38     20
4         14     49     20
...      ...    ...    ...
40307      5     20      5
40308     24     94     24
40309      0      0      0
40310      6     32      6
40311      5     20      5

[40312 rows x 22 columns],       AGEB  POBTOT POBFEM POBMAS P_0A2 P_3A5 P_60YMAS POB65_MAS P3
HLINHE  \
0      132     191    104     87     8     6       33        28        0
1      132      27     16     11     *     *        3         3        0
2      132      83     39     44     3     5        7         3        0
3      132     118     63     55     3     7        5         3        0
4      132     116     56     60     4     4        3         *        0
...    ...     ...    ...    ...   ...   ...      ...       ...      ...
34962  449      31     14     17     0     0        8         6        0
34963  453    1529    853    676    44    53      311       217        0
34964  453      37     18     19     *     3        6         4        0
34965  453      18      9      9     0     0        6         5        0
34966  453      18      9      9     0     *        5         4        0

      PHOG_IND  ... PSIND_LIM P15YM_AN P15YM_SE GRAPROES  PEA PSINDER PDER_SS  \
0            0  ...       125        4        5     9.49   92      96      95
1            *  ...        19        0        *     9.28   11       9      18
2            7  ...        71        *        3     7.79   38      30      53
3            8  ...       101        *        *     9.58   65      39      79
4            6  ...       109        0        *        9   49      33      83
...        ...  ...       ...      ...      ...      ...  ...     ...     ...
34962        0  ...        27        0        0    14.04   21       4      27
34963        3  ...      1354        6        5    14.04  933     482    1046
34964        0  ...        35        0        0     14.9   23      10      27
34965        0  ...        17        0        0     13.8   10       3      15
34966        0  ...        15        0        0    14.18   10       8      10
```

```
      TOTHOG POBHOG VIVTOT
0         43    191     47
1          9     27     11
2         24     83     24
3         30    118     32
4         26    116     28
...      ...    ...    ...
34962      8     31      9
34963    578   1517    745
34964     11     37     12
34965      8     18      8
34966      6     18      8

[34967 rows x 22 columns]]
```

In [ ]:
```
df_concat = pd.concat(dataframes_list, ignore_index=True)
df_concat.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

In [ ]:
```
# Observamos la estructura
df_concat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 221070 entries, 0 to 221069
Data columns (total 22 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   AGEB       221069 non-null  object
 1   POBTOT     221069 non-null  float64
 2   POBFEM     221069 non-null  object
 3   POBMAS     221069 non-null  object
 4   P_0A2      221069 non-null  object
 5   P_3A5      221069 non-null  object
 6   P_60YMAS   221069 non-null  object
 7   POB65_MAS  221069 non-null  object
 8   P3HLINHE   221069 non-null  object
 9   PHOG_IND   221069 non-null  object
 10  PCON_DISC  221069 non-null  object
 11  PCON_LIMI  221069 non-null  object
 12  PSIND_LIM  221069 non-null  object
 13  P15YM_AN   221069 non-null  object
 14  P15YM_SE   221069 non-null  object
 15  GRAPROES   221069 non-null  object
 16  PEA        221069 non-null  object
 17  PSINDER    221069 non-null  object
 18  PDER_SS    221069 non-null  object
 19  TOTHOG     221069 non-null  object
 20  POBHOG     221069 non-null  object
```

```
 21  VIVTOT     221069 non-null  float64
dtypes: float64(2), object(20)
memory usage: 37.1+ MB
```

In [ ]:
```
# Eliminamos los caracteres de tipo *
df_concat.replace('*', np.nan, inplace=True)
df_concat.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

In [ ]:
```
print("Número de renglones: ", len(df_concat))
```

```
Número de renglones:  221070
```

Con base en lo anterior, es posible realizar un primer abordaje para conocer cómo se realiza la lectura de nuestras variables, describiendo cada columna.

Tal como se aprecia en la imagen anterior, la mayoría de los datos no están catalogados como numéricos, esto a pesar que al abrir las bases de datos sí son números. Esto puede pasar por diversas circunstancias, entre las que destaca, por ejemplo, el uso de caracteres especiales.

Selección y limpieza de los Datos en Python

Una vez que se analizaron los datos, encontramos el caracter *"*"* como parte de la información, por esto es que no aparecían los datos como numéricos, tal como se explicó anteriormente. Por tanto, decidimos optar por distintas formas de tratamiento de los datos no congruentes, como eran en este caso los caracteres *"*"* .

Decidimos realizar distintas estrategias para conocer cuál sería mejor en el tratamiento de la información. Cada una de éstas se caracteriza por la copia del DataFrame original y la realización de distintos cambios en dichas copias.

In [ ]:
```
# Hacemos el cambio para que todas las variables sean numéricas
dfs= df_concat.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
dfs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 221070 entries, 0 to 221069
Data columns (total 22 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   AGEB       221070 non-null  int64
 1   POBTOT     221070 non-null  int64
 2   POBFEM     221070 non-null  int64
```

```
 3   POBMAS      221070 non-null  int64
 4   P_0A2       221070 non-null  int64
 5   P_3A5       221070 non-null  int64
 6   P_60YMAS    221070 non-null  int64
 7   POB65_MAS   221070 non-null  int64
 8   P3HLINHE    221070 non-null  int64
 9   PHOG_IND    221070 non-null  int64
 10  PCON_DISC   221070 non-null  int64
 11  PCON_LIMI   221070 non-null  int64
 12  PSIND_LIM   221070 non-null  int64
 13  P15YM_AN    221070 non-null  int64
 14  P15YM_SE    221070 non-null  int64
 15  GRAPROES    221070 non-null  int64
 16  PEA         221070 non-null  int64
 17  PSINDER     221070 non-null  int64
 18  PDER_SS     221070 non-null  int64
 19  TOTHOG      221070 non-null  int64
 20  POBHOG      221070 non-null  int64
 21  VIVTOT      221070 non-null  int64
dtypes: int64(22)
memory usage: 37.1 MB
```

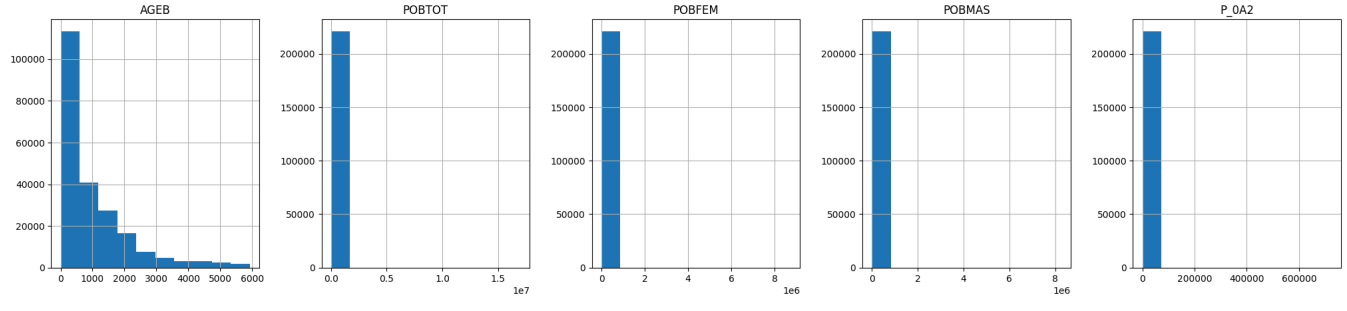
In [ ]:

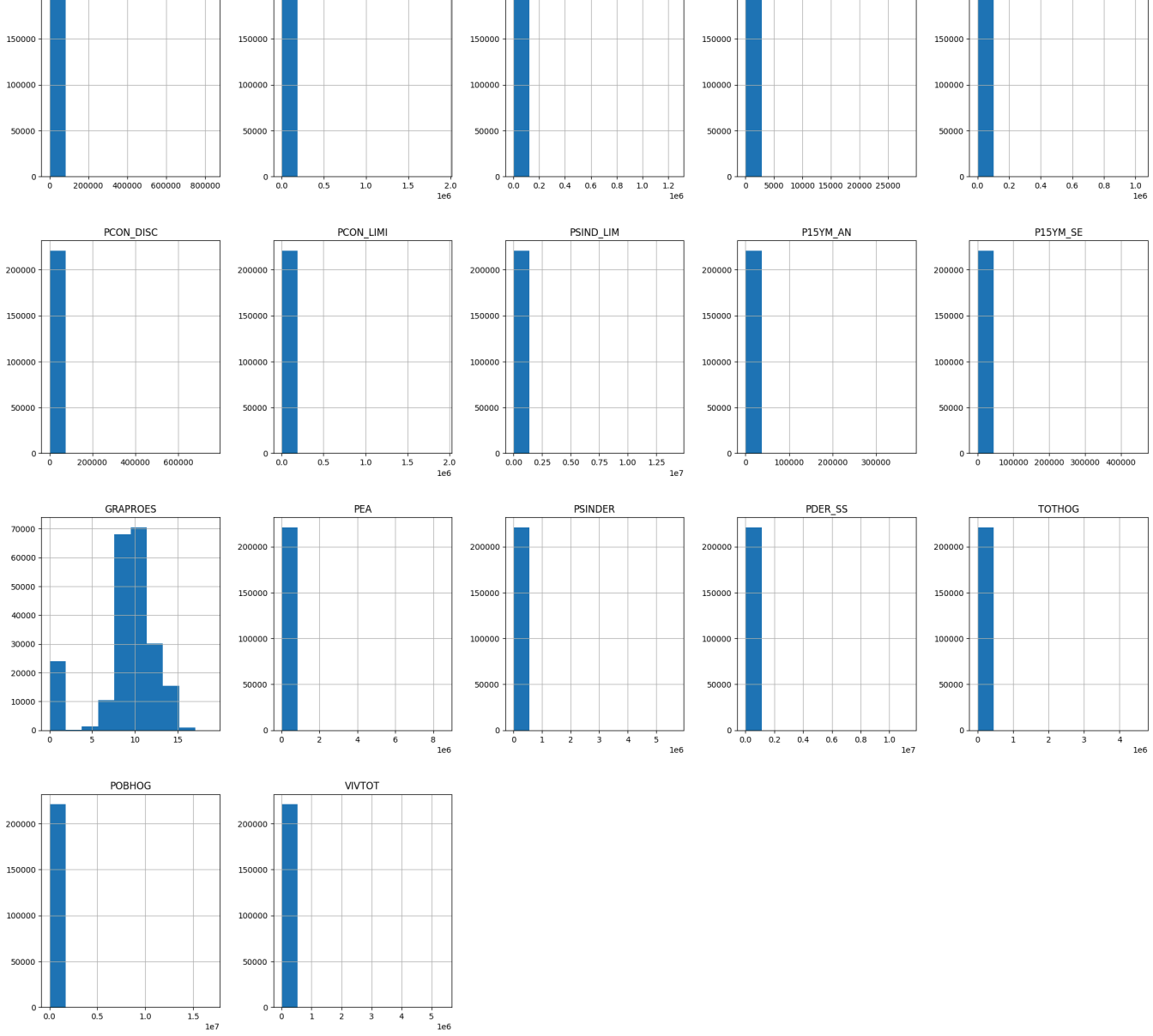
```
fig = plt.figure(figsize = (25,30))
ax = fig.gca()
dfs.hist(ax = ax)
```

```
<ipython-input-23-b5849d95ada3>:3: UserWarning: To output multiple subplots, the figure conta
ining the passed axes is being cleared.
  dfs.hist(ax = ax)
```

Out[ ]:

```
array([[<Axes: title={'center': 'AGEB'}>,
        <Axes: title={'center': 'POBTOT'}>,
        <Axes: title={'center': 'POBFEM'}>,
        <Axes: title={'center': 'POBMAS'}>,
        <Axes: title={'center': 'P_0A2'}>],
       [<Axes: title={'center': 'P_3A5'}>,
        <Axes: title={'center': 'P_60YMAS'}>,
        <Axes: title={'center': 'POB65_MAS'}>,
        <Axes: title={'center': 'P3HLINHE'}>,
        <Axes: title={'center': 'PHOG_IND'}>],
       [<Axes: title={'center': 'PCON_DISC'}>,
        <Axes: title={'center': 'PCON_LIMI'}>,
        <Axes: title={'center': 'PSIND_LIM'}>,
        <Axes: title={'center': 'P15YM_AN'}>,
        <Axes: title={'center': 'P15YM_SE'}>],
       [<Axes: title={'center': 'GRAPROES'}>,
        <Axes: title={'center': 'PEA'}>,
        <Axes: title={'center': 'PSINDER'}>,
        <Axes: title={'center': 'PDER_SS'}>,
        <Axes: title={'center': 'TOTHOG'}>],
       [<Axes: title={'center': 'POBHOG'}>,
        <Axes: title={'center': 'VIVTOT'}>, <Axes: >, <Axes: >, <Axes: >]],
      dtype=object)
```





Las estrategias abordadas fueron:

Estrategía 1.- Eliminación de renglones/columnas con datos faltantes

En esta parte mostraremos el paso a paso y cuáles son los resultados de cada una de las modificaciones. Iniciamos con la eliminación de las columnas con valores nulos y mantenemos un seguimiento de los renglones. En este primer paso, notamos que el número de renglones es de 1,640,722; esto se aprecia en la siguiente imagen.

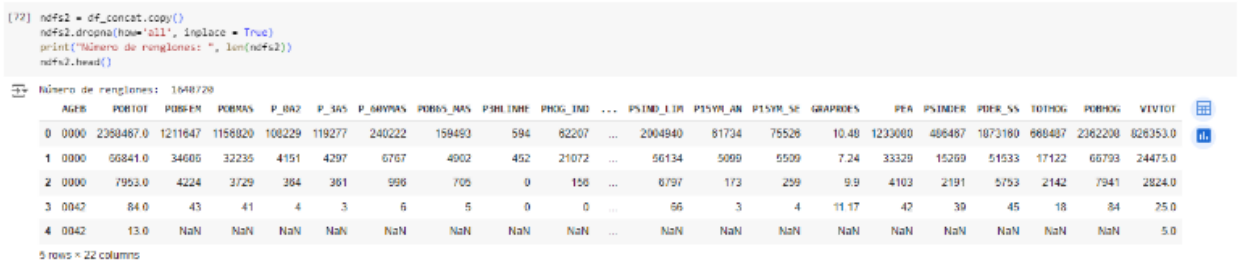
```
[71] ndfs1 = df_concat.copy()
     ndfs1.dropna(axis = 1, inplace = True) # axis 1 son columnas / axis 0 son renglones.
     print("Número de renglones: ", len(ndfs1))
     ndfs1.head()

     Número de renglones:  1640722
```

0	
1	
2	
3	
4	

Imagen 4. Eliminación de columnas con valores nulos. *(Elaboración propia, 2023)*

Imagen 4. Eliminación de columnas con valores nulos. *(Elaboración propia, 2023)*

Posteriormente, realizamos una eliminación de todos los renglones donde haya valores nulos. Podemos observar que en el resultado tenemos aún el total de 1,640,720 renglones. Es importante mencionar que nos aseguramos de mantener los DataFrames originales intactos y sólo trabajar en nuevos.



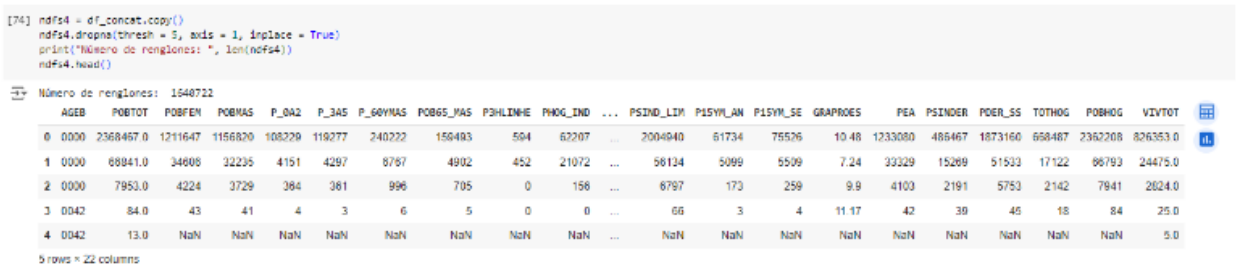
```
[72] ndfs2 = df_concat.copy()
     ndfs2.dropna(how='all', inplace = True)
     print("Número de renglones: ", len(ndfs2))
     ndfs2.head()

→ Número de renglones:  1640720
```

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_IND	...	PSIND_LIM	P15YM_AN	P15YM_SE	GRAPROES	PEA	PSINDER	PDER_SS	TOTHOG	POBHOG	VIVTOT
0	0000	2368467.0	1211647	1156820	108229	119277	240222	156493	594	62207	...	2004940	61734	75526	10.48	1233080	486467	1873160	668487	2362208	826353.0
1	0000	68841.0	34608	32235	4151	4297	6767	4902	452	21072	...	56134	5099	5509	7.24	33329	15269	51533	17122	66793	24475.0
2	0000	7953.0	4224	3729	364	361	996	705	0	156	...	6797	173	259	9.9	4103	2191	5753	2142	7941	2824.0
3	0042	84.0	43	41	4	3	6	5	0	0	...	66	3	4	11.17	42	39	45	18	84	25.0
4	0042	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0

5 rows × 22 columns

Imagen 5. Eliminación de renglones con valores nulos. *(Elaboración propia, 2024)*

Aplicamos otro método de eliminación para nulos. Observamos que la cantidad de filas se mantiene intacta con un total de 1,640,722.



```
[74] ndfs4 = df_concat.copy()
     ndfs4.dropna(thresh = 5, axis = 1, inplace = True)
     print("Número de renglones: ", len(ndfs4))
     ndfs4.head()

→ Número de renglones:  1640722
```

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_IND	...	PSIND_LIM	P15YM_AN	P15YM_SE	GRAPROES	PEA	PSINDER	PDER_SS	TOTHOG	POBHOG	VIVTOT
0	0000	2368467.0	1211647	1156820	108229	119277	240222	156493	594	62207	...	2004940	61734	75526	10.48	1233080	486467	1873160	668487	2362208	826353.0
1	0000	68841.0	34608	32235	4151	4297	6767	4902	452	21072	...	56134	5099	5509	7.24	33329	15269	51533	17122	66793	24475.0
2	0000	7953.0	4224	3729	364	361	996	705	0	156	...	6797	173	259	9.9	4103	2191	5753	2142	7941	2824.0
3	0042	84.0	43	41	4	3	6	5	0	0	...	66	3	4	11.17	42	39	45	18	84	25.0
4	0042	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0

5 rows × 22 columns

Imagen 6. Eliminación de renglones con valores nulos. *(Elaboración propia, 2024)*

Procedimiento paso a paso:

1.1.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente utilizamos el método dropna para eliminar los registros con valores nulos, observamos que una vez eliminados ya no poseemos valores nulos en nuestro dataset

In [ ]:
```
ndfs = df_concat.copy()
ndfs.dropna(inplace = True)
print("¿Existen valores nulos?", ndfs.isna().values.any())
print("Número de registros eliminados:", len(dfs) - len(ndfs))
```

```
¿Existen valores nulos? False
Número de registros eliminados: 165106
```

1.2.- Como siguiente paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente utilizamos el método dropna con el valor axis=1 para eliminar las columnas con valores nulos

In [ ]:
```
ndfs1 = df_concat.copy()
```

```
ndfs1.dropna(axis = 1, inplace = True) # axis 1 son columnas / axis 0 son renglones.
print("Número de renglones: ", len(ndfs1))
ndfs1.head()
```

```
Número de renglones:  221070
```

Out[ ]:

0

1

2

3

4

1.3.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente utilizamos el método dropna con el valor how=all para eliminar los renglones donde todos los elementos son nulos, dado que no se cumple esta condición, se conservan todos

In [ ]:

```
ndfs2 = df_concat.copy()
ndfs2.dropna(how='all', inplace = True)
print("Número de renglones: ", len(ndfs2))
ndfs2.head()
```

```
Número de renglones:  221069
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

1.4.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente utilizamos el método dropna con el valor thresh=4 para eliminar los renglones donde al menos se tengan "n" elementos no nulos para ser conservados

In [ ]:

```
ndfs3 = df_concat.copy()
ndfs3.dropna(thresh=25, inplace = True)
print("Número de renglones: ", len(ndfs3))
ndfs3.head()
```

```
Número de renglones:  0
```

Out[ ]:

AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_IN

0 rows × 22 columns

1.5.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente utilizamos el método dropna con el valor thresh=n y axis=1 para eliminar las columnas donde al menos se tengan "n" elementos no nulos para ser conservados

In [ ]:
```
ndfs4 = df_concat.copy()
ndfs4.dropna(thresh = 5, axis = 1, inplace = True)
print("Número de renglones: ", len(ndfs4))
ndfs4.head()
```

```
Número de renglones:  221070
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

1.6.- Eliminación de los totalizadores

In [ ]:
```
ndfs5 = df_concat.copy()
ndfs5.drop(ndfs5[ndfs5['AGEB'] == 0].index, inplace=True)
print("Número de renglones: ", len(ndfs5))
ndfs5.head()
```

```
Número de renglones:  221070
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

Estrategía 2.- Imputación de datos faltantes

Decidimos también realizar la imputación de datos faltantes a través de diversas técnicas tales como la moda, la media y la mediana. Si bien esto lo hicimos para experimentar, cabe mencionar que no elegimos ninguna de estas técnicas, ya que podrían ensuciar los datos poblacionales. Igualmente, al final lo que decidimos es tratar los nulos como datos faltantes y, posteriormente, esos fueron eliminados de la base para los análisis subsecuentes.

De forma que, una vez determinada la estrategia final, se lograron tratar los caracteres "*" y poder convertir las columnas de tipo object a int, puesto que correspondía que el tipo de datos que contienen

Procedimiento paso a paso:

2.1.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente obtenemos la media del campo a imputar

In [ ]:
```
ndfs6 = df_concat.copy()
TotalViviendas = ndfs6.VIVTOT.mean()
print("Total de viviendas:", TotalViviendas)
```

```
Total de viviendas: 185.11044063165798
```

In [ ]:
```
# Una vez que tenemos la media, imputamos el valor en la columna
ndfs6['VIVTOT'].fillna(value = TotalViviendas, inplace = True)
ndfs6.head()
```

```
<ipython-input-31-e8ad7cab591f>:2: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.
```

```
ndfs6['VIVTOT'].fillna(value = TotalViviendas, inplace = True)
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

In [ ]:
```
# Verificamos que ya no tenemos valores nulos en la columna imputada
print(ndfs6['VIVTOT'].isnull().values.any())
```

```
False
```

2.2.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente imputamos con la mediana

In [ ]:
```
ndfs7 = df_concat.copy()
ndfs7['VIVTOT'].fillna(value = ndfs7.VIVTOT.median(), inplace = True)
ndfs7.head()
```

```
<ipython-input-33-e84d352a8e19>:2: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.
```

```
ndfs7['VIVTOT'].fillna(value = ndfs7.VIVTOT.median(), inplace = True)
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

In [ ]:
```
# Verificamos que ya no tenemos valores nulos en la columna imputada
print(ndfs7['VIVTOT'].isnull().values.any())
```

```
False
```

2.3.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente imputamos con la moda

In [ ]:
```
ndfs8 = df_concat.copy()
ndfs8['VIVTOT'].fillna(value = ndfs8.VIVTOT.median(), inplace = True)
ndfs8.head()
```

```
<ipython-input-35-421a0b0b576c>:2: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.
```

```
ndfs8['VIVTOT'].fillna(value = ndfs8.VIVTOT.median(), inplace = True)
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

5 rows × 22 columns

In [ ]:
```
# Verificamos que ya no tenemos valores nulos en la columna imputada
print(ndfs8['VIVTOT'].isnull().values.any())
```

```
False
```

2.4.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, posteriomente eliminamos los renglones específicos donde hay nulos

In [ ]:
```
ndfs9 = df_concat.copy()
ndfs9.dropna(subset=['VIVTOT', 'POBTOT'], inplace = True)
len(ndfs9)
```

Out[ ]:
```
221069
```

2.5.- Como primer paso realizamos una copia del dataset original para no perder los cambios realizados hasta ahora, imputamos dos columnas con dos estrategías diferentes

In [ ]:
```
ndfs10 = df_concat.copy()
ndfs10.VIVTOT.fillna(ndfs10.VIVTOT.mode()[0], inplace=True)
ndfs10.POBTOT.fillna(ndfs10.POBTOT.mean(), inplace=True)
ndfs10.head()
print(ndfs10['VIVTOT'].isnull().values.any())
print(ndfs10['POBTOT'].isnull().values.any())
```

```
False
False
<ipython-input-38-32484cd0a34a>:2: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.


  ndfs10.VIVTOT.fillna(ndfs10.VIVTOT.mode()[0], inplace=True)
<ipython-input-38-32484cd0a34a>:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.


  ndfs10.POBTOT.fillna(ndfs10.POBTOT.mean(), inplace=True)
```

2.6.- Eliminamos la clave del AGEB que no necesita ser analizada así como los nulos

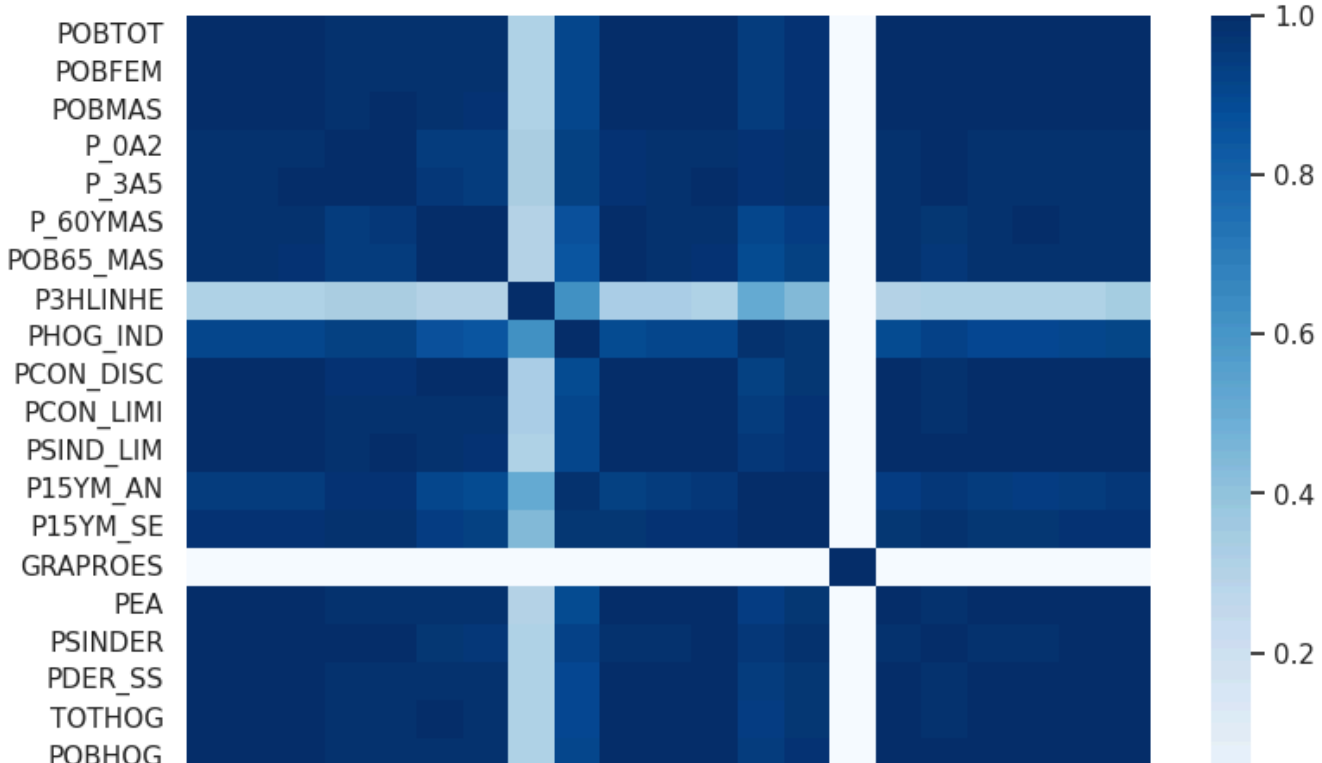
In [ ]:
```
ndfs11 = df_concat.copy()
ndfs11= ndfs11.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
ndfs11.info()
ndfs11 = ndfs11.drop('AGEB', axis=1)
```

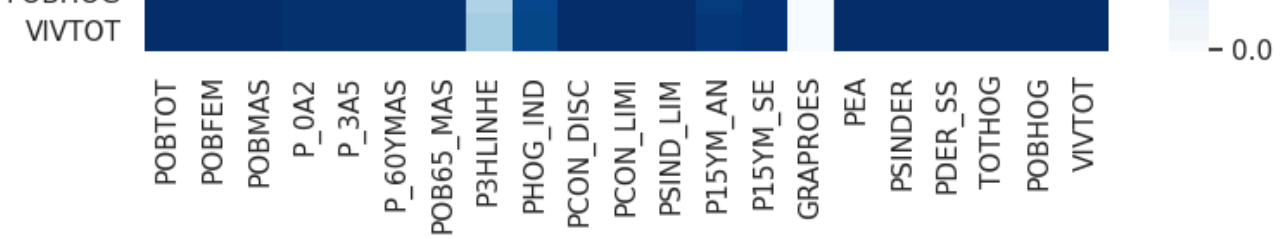
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 221070 entries, 0 to 221069
Data columns (total 22 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   AGEB       221070 non-null  int64
 1   POBTOT     221070 non-null  int64
 2   POBFEM     221070 non-null  int64
 3   POBMAS     221070 non-null  int64
 4   P_0A2      221070 non-null  int64
 5   P_3A5      221070 non-null  int64
 6   P_60YMAS   221070 non-null  int64
 7   POB65_MAS  221070 non-null  int64
 8   P3HLINHE   221070 non-null  int64
 9   PHOG_IND   221070 non-null  int64
 10  PCON_DISC  221070 non-null  int64
 11  PCON_LIMI  221070 non-null  int64
 12  PSIND_LIM  221070 non-null  int64
 13  P15YM_AN   221070 non-null  int64
 14  P15YM_SE   221070 non-null  int64
 15  GRAPROES   221070 non-null  int64
 16  PEA        221070 non-null  int64
 17  PSINDER    221070 non-null  int64
 18  PDER_SS    221070 non-null  int64
 19  TOTHOG     221070 non-null  int64
 20  POBHOG     221070 non-null  int64
 21  VIVTOT     221070 non-null  int64
dtypes: int64(22)
memory usage: 37.1 MB
```

2.7.- Evaluamos la correlación con Y

In [ ]:

```
sns.set(rc={'figure.figsize':(9,6)})

pearson_correlation_map = ndfs11.corr(method="pearson")
sns.heatmap(pearson_correlation_map,cmap="Blues")

plt.show()
```





Estrategía Final.- Eliminación de totalizadores e imputación de nulos en AGEB

In [ ]:
```
ndfs_filtered = df_concat.copy()
ndfs_final = df_concat.copy()
ndfs_filtered = ndfs_final[ndfs_final['AGEB'] != 0]
ndfs_filtered.head()
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
4	0043	92.0	54	38	6	5	11	10	0	

5 rows × 22 columns

In [ ]:
```
ndfs_filtered['AGEB'].fillna(0, inplace=True)
ndfs_filtered.dropna(inplace=True)
ndfs_filtered.head()
```

```
<ipython-input-42-6757994bb328>:1: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.
```

```
ndfs_filtered['AGEB'].fillna(0, inplace=True)
```

Out[ ]:

	AGEB	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	0000	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	5
1	0000	22268.0	11563	10705	1241	1239	2625	1847	0	
2	0000	439.0	229	210	21	18	64	53	0	
3	0043	439.0	229	210	21	18	64	53	0	
6	0043	0.0	0	0	0	0	0	0	0	

5 rows × 22 columns

In [ ]:
```
df2  = ndfs_filtered.copy()
df2 = df2.drop('AGEB', axis=1)
```

In [ ]:
```
df2.head()
```

Out[ ]:

	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHOG_IND
0	3082841.0	1601462	1481379	134738	156486	383675	264746	28497	584693
1	22268.0	11563	10705	1241	1239	2625	1847	0	203
2	439.0	229	210	21	18	64	53	0	12
3	439.0	229	210	21	18	64	53	0	12
6	0.0	0	0	0	0	0	0	0	0

5 rows × 21 columns

2.3 Frecuencia de las clases en variables categóricas

Respecto a los sets provenientes del INEGI, nuestra primordial categoría corresponde a los Estados, teniendo por tanto al nivel más superficial, datos para cada uno de los 32 Estados de la República. Sin embargo, ya que se determinó que el estudio sería realizado por AGEB, es dicho nivel de detalles relativo a las categorías que podríamos considerar como el primario.

¿Cuál es la cardinalidad de las variables categóricas?

Una de las principales categorías en nuestra base de datos de islas de calor está dada por los rangos de temperatura, que están determinados de acuerdo con la información compartida. Tal como se muestra en la siguiente imagen, donde se consideran los rangos de temperatura y se clasifican con nomenclatura nominal.

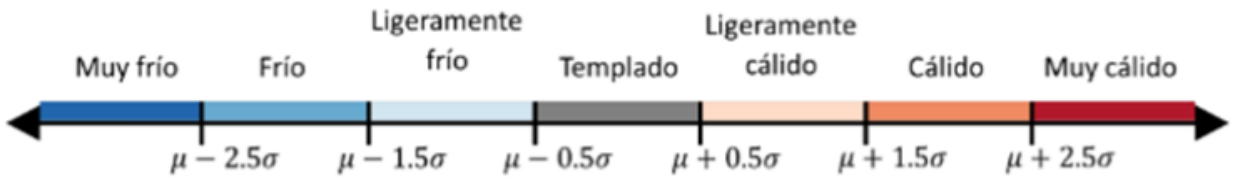


Imagen 10. Categorías de temperatura basadas en las estadísticas de la imagen. *(Torre, et. al. 2023)*

2.4 Identificación de valores faltantes

Los valores faltantes son aquellos datos que no están presentes en un conjunto de datos, ya sea porque no se registraron, se perdieron o no se recopilaron. En el análisis de datos, los valores faltantes pueden aparecer en cualquier variable y pueden afectar la calidad y precisión del análisis, causando sesgos o errores en los resultados.

¿Hay valores faltantes en el conjunto de datos? ¿Se pueden identificar patrones de ausencia?

En las bases de datos del INEGI y en el dataset de imágenes raster con las islas de calor por estado, no hay valores faltantes per se. Sin embargo, nos encontramos con una encrucijada, ya que el dataset del INEGI no contaba con las coordenadas de las AGEB. Esto representa un desafío, ya que la falta de información espacial puede limitar el análisis y la integración con los datos de temperatura. Sobre todo tomando en cuenta que los datos de INEGI están en formato tabular, y se necesitará crear una geometría como en este caso centroides de los AGEB para poder relacionarlos espacialmente. Por esta razón fue necesario recurrir al Marco Geoestadístico. Censo de Población y Vivienda 2020 (INEGI, s.f.) y descargar los archivos .shp por estado.

Parte 3: Análisis univariante

El análisis univariante es aquel que permite estudiar, como su nombre lo indica, una variable a través de distintos instrumentos; el cual se abordará en esta sección. Cabe mencionar que se distingue del análisis multivariante ya que en este punto lo que se busca es describir, y, a pesar de que se estudien distintas variables, no se determinará cómo están asociadas, sino más bien conocer sus características. (Mas, 2019)

¿Cuáles son las estadísticas resumidas del conjunto de datos?

Al ser un conjunto de datos a total país, se puede apreciar que las estadísticas de la base de datos no muestran una tendencia como tal. Más bien, esto se apreciará de mejor manera cuando se trabajen las mallas por AGEB, donde también se utilizará la información de temperatura.

¿Hay valores atípicos en el conjunto de datos?

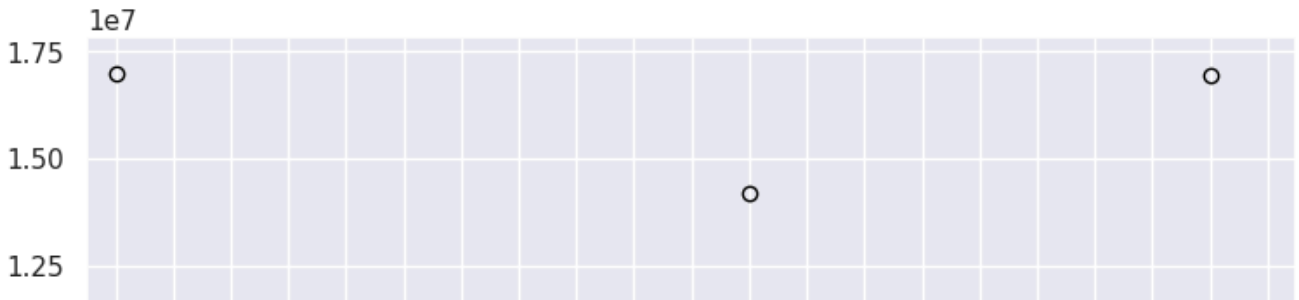
Sí, encontramos que hay valores atípicos en la información de SCITEL (INEGI, 2020). Tal como se aprecia en la siguiente imagen; cabe mencionar que para el procesamiento de datos, decidimos normalizarlos.

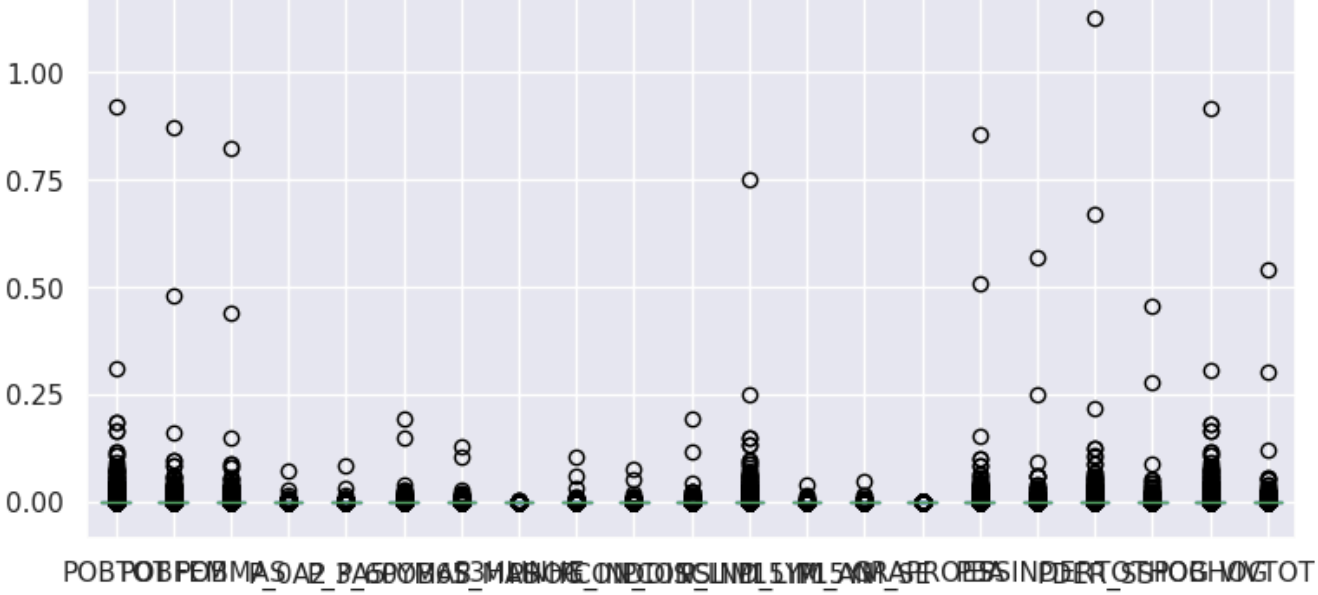
In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

Análisis por boxplot

In [ ]:
```
ndfs11= ndfs11.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
ndfs11.plot(kind='box')
```

Out[ ]: <Axes: >





Normalización del dataframe para correlación

In [ ]:
```
scaler = StandardScaler()
df_transformed = pd.DataFrame(scaler.fit_transform(ndfs11), columns=ndfs11.columns)
df_transformed.head()
```

Out[ ]:

	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	71.239303	71.677528	70.768385	78.562350	78.469231	69.580124	71.829724	437.984402	219.6
1	0.501643	0.504563	0.498525	0.711854	0.609295	0.462537	0.487569	-0.006283	0.0
2	-0.002880	-0.002811	-0.002934	0.000396	-0.003065	-0.001996	0.000736	-0.006283	-0.0
3	-0.002880	-0.002811	-0.002934	0.000396	-0.003065	-0.001996	0.000736	-0.006283	-0.0
4	-0.010901	-0.010645	-0.011152	-0.008352	-0.009585	-0.011609	-0.010933	-0.006283	-0.0

5 rows × 21 columns

Parte 4: Análisis bi/multivariante

El análisis bivariable nos sirve para conocer la relación entre dos variables, y con ello determinar si el comportamiento de una afecta a la otra. Es decir, si el proceder de una variable está en función de otra. Naturalmente, el análisis multivariante nos sirve para determinar esas relaciones entre más de dos variables. (Bech, 2019)

¿Hay correlación entre las variables dependientes e independientes?

Sí, observamos correlación en algunas de las variables, tal como se muestra en la siguiente imagen. Recordemos que aquellas variables que tienen una correlación perfecta son las que tienen un 1; que normalmente esto ocurre con la misma variable, que es lo que se espera. Sin embargo, se puede apreciar que hay, por ejemplo, una alta correlación cuando hablamos de personas de 60 años o más con alguna discapacidad.

Procedemos a importar las librerías, a leer el archivo con la información y a tener una copia de

la data

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [ ]:
```
df2.describe()
```

Out[ ]:

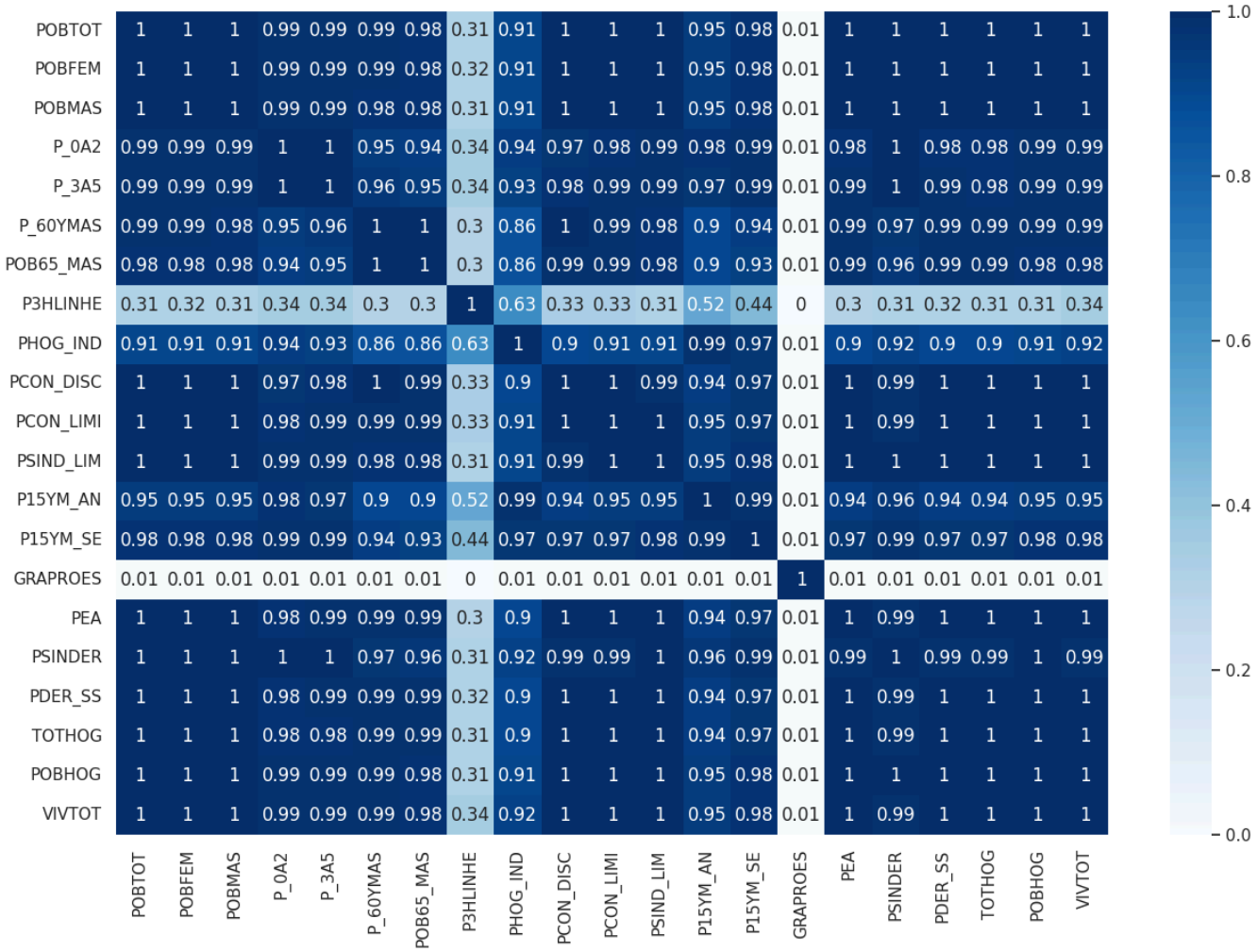
	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POI
count	5.596400e+04	5.596400e+04	5.596400e+04	55964.000000	55964.000000	5.596400e+04	5.596
mean	1.816706e+03	9.412028e+02	8.755028e+02	67.006361	79.080141	2.429248e+02	1.633
std	8.596102e+04	4.438182e+04	4.158104e+04	3406.716529	3961.371372	1.095377e+04	7.321
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000
25%	2.300000e+01	1.100000e+01	1.100000e+01	0.000000	0.000000	1.750000e+00	0.000
50%	1.630000e+02	8.400000e+01	7.900000e+01	6.000000	7.000000	1.900000e+01	1.200
75%	3.160000e+02	1.630000e+02	1.520000e+02	13.000000	15.000000	4.200000e+01	2.800
max	1.699242e+07	8.741123e+06	8.251295e+06	724133.000000	834611.000000	1.919454e+06	1.258

8 rows × 21 columns

In [ ]:
```
sns.set(rc={'figure.figsize':(15,10)})
df2= df2.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
df2.info()
df2.head()
sns.heatmap(df2.corr().round(2), annot=True, cmap="Blues")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 55964 entries, 0 to 221066
Data columns (total 21 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   POBTOT     55964 non-null  int64
 1   POBFEM     55964 non-null  int64
 2   POBMAS     55964 non-null  int64
 3   P_0A2      55964 non-null  int64
 4   P_3A5      55964 non-null  int64
 5   P_60YMAS   55964 non-null  int64
 6   POB65_MAS  55964 non-null  int64
 7   P3HLINHE   55964 non-null  int64
 8   PHOG_IND   55964 non-null  int64
 9   PCON_DISC  55964 non-null  int64
 10  PCON_LIMI  55964 non-null  int64
 11  PSIND_LIM  55964 non-null  int64
 12  P15YM_AN   55964 non-null  int64
 13  P15YM_SE   55964 non-null  int64
 14  GRAPROES   55964 non-null  int64
 15  PEA        55964 non-null  int64
 16  PSINDER    55964 non-null  int64
```

```
 17  PDER_SS     55964 non-null  int64
 18  TOTHOG      55964 non-null  int64
 19  POBHOG      55964 non-null  int64
 20  VIVTOT      55964 non-null  int64
dtypes: int64(21)
memory usage: 9.4 MB
```



Observamos la información de las columnas

Nota: en el conteo de nulos podemos apreciar que hay algunas columnas que tienen faltantes, por ejemplo género y nivel eductativo, entre otros

¿Cómo se distribuyen los datos en función de diferentes categorías?

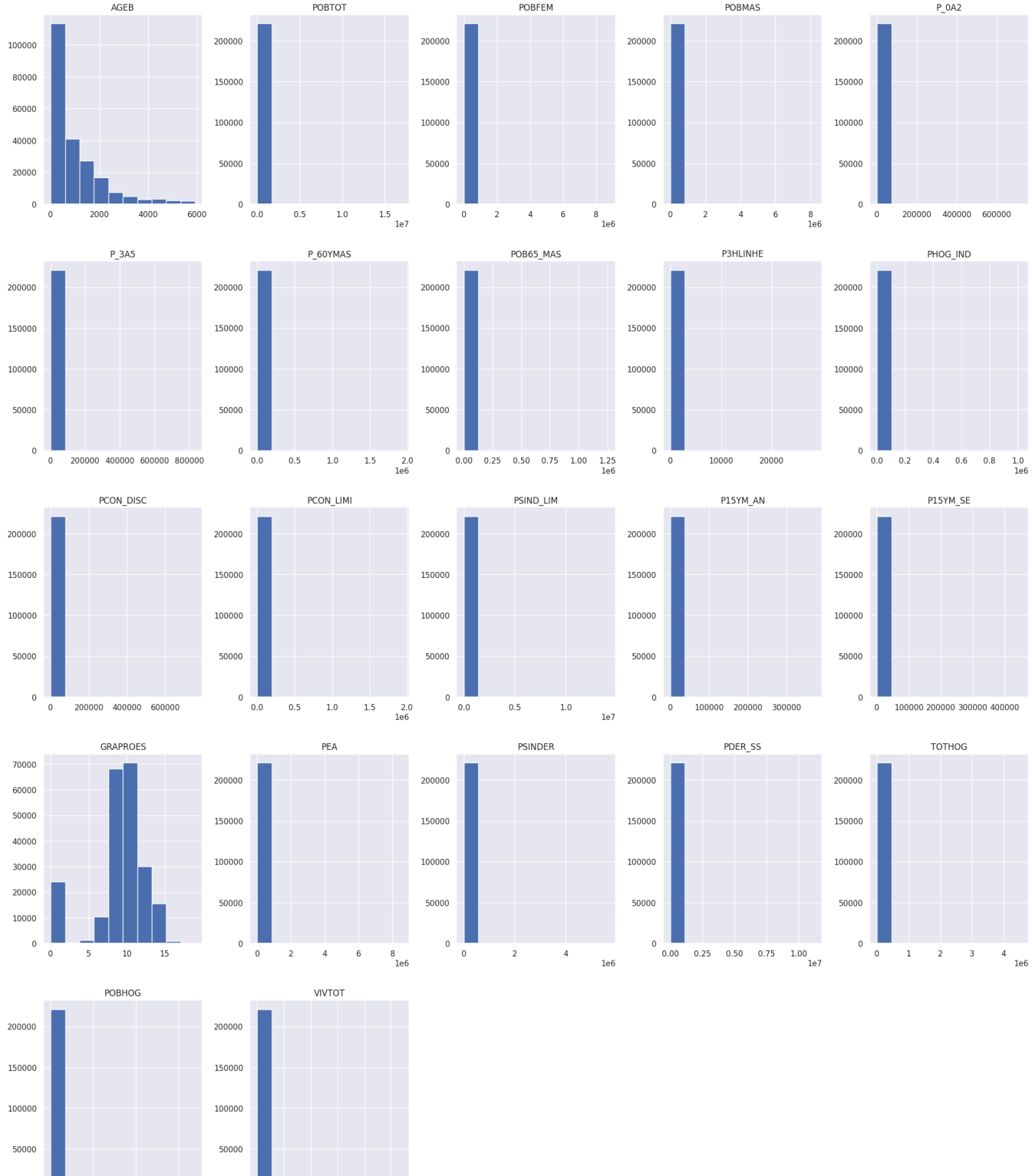
En la siguiente imagen observamos a grandes rasgos la distribución de nuestras variables, esto por medio de histogramas. La función de esto es poder conocer cómo se comportan cada una de las variables.

In [ ]:
```
fig = plt.figure(figsize = (25,30))
ax = fig.gca()
dfs.hist(ax = ax)
```

```
<ipython-input-69-b5849d95ada3>:3: UserWarning: To output multiple subplots, the figure conta
ining the passed axes is being cleared.
  dfs.hist(ax = ax)
```

Out[ ]:
```
array([[<Axes: title={'center': 'AGEB'}>,
        <Axes: title={'center': 'POBTOT'}>,
        <Axes: title={'center': 'POBFEM'}>,
        <Axes: title={'center': 'POBMAS'}>,
        <Axes: title={'center': 'P_0A2'}>],
```

```
 [<Axes: title={'center': 'P_3A5'}>,
  <Axes: title={'center': 'P_60YMAS'}>,
  <Axes: title={'center': 'POB65_MAS'}>,
  <Axes: title={'center': 'P3HLINHE'}>,
  <Axes: title={'center': 'PHOG_IND'}>],
 [<Axes: title={'center': 'PCON_DISC'}>,
  <Axes: title={'center': 'PCON_LIMI'}>,
  <Axes: title={'center': 'PSIND_LIM'}>,
  <Axes: title={'center': 'P15YM_AN'}>,
  <Axes: title={'center': 'P15YM_SE'}>],
 [<Axes: title={'center': 'GRAPROES'}>,
  <Axes: title={'center': 'PEA'}>,
  <Axes: title={'center': 'PSINDER'}>,
  <Axes: title={'center': 'PDER_SS'}>,
  <Axes: title={'center': 'TOTHOG'}>],
 [<Axes: title={'center': 'POBHOG'}>,
  <Axes: title={'center': 'VIVTOT'}>, <Axes: >, <Axes: >, <Axes: >]],
dtype=object)
```



Cabe destacar que en algunas de las gráficas se observa un único valor, como sumatoria que nos resume en realidad el total, puesto que la segmentación por AGEB se hará en etapas posteriores.

¿Existen patrones o agrupaciones (clusters) en los datos con características similares?

Inicialmente, realizamos un análisis PCA para poder conocer cómo se comportan las variables entre sí y, de esta manera, conocer los escenarios a los cuales nos podremos enfrentar más adelante cuando trabajemos con clústeres más avanzados. En la siguiente imagen se aprecia el resumen del análisis PCA, así como la proporción acumulada de cada variable en la explicación del fenómeno.

Para comprender mejor la varianza explicada, decidimos graficar los resultados. En la siguiente imagen se puede observar que a través de los primeros tres componentes se tendría la totalidad de la varianza explicada.

PCA y análisis de componentes

In [ ]:
```
pca = PCA()
pca.fit(df_transformed)
pcaSummary = pd.DataFrame({'Standard deviation': np.round(np.sqrt(pca.explained_variance_)
                           'Proportion of variance': np.round(pca.explained_variance_ratio_
                           'Cumulative proportion': np.round(np.cumsum(pca.explained_varian
                           })

pcaSummary = pcaSummary.transpose()
pcaSummary.columns = ['PC{}'.format(i) for i in range(1, len(pcaSummary.columns) + 1)]
pcaSummary
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC12	PC13
Standard deviation	4.33	1.03	1.00	0.44	0.09	0.06	0.05	0.03	0.03	0.02	...	0.01	0.01
Proportion of variance	89.17	5.05	4.76	0.93	0.04	0.02	0.01	0.00	0.00	0.00	...	0.00	0.00
Cumulative proportion	89.17	94.23	98.99	99.92	99.95	99.97	99.99	99.99	100.00	100.00	...	100.00	100.00

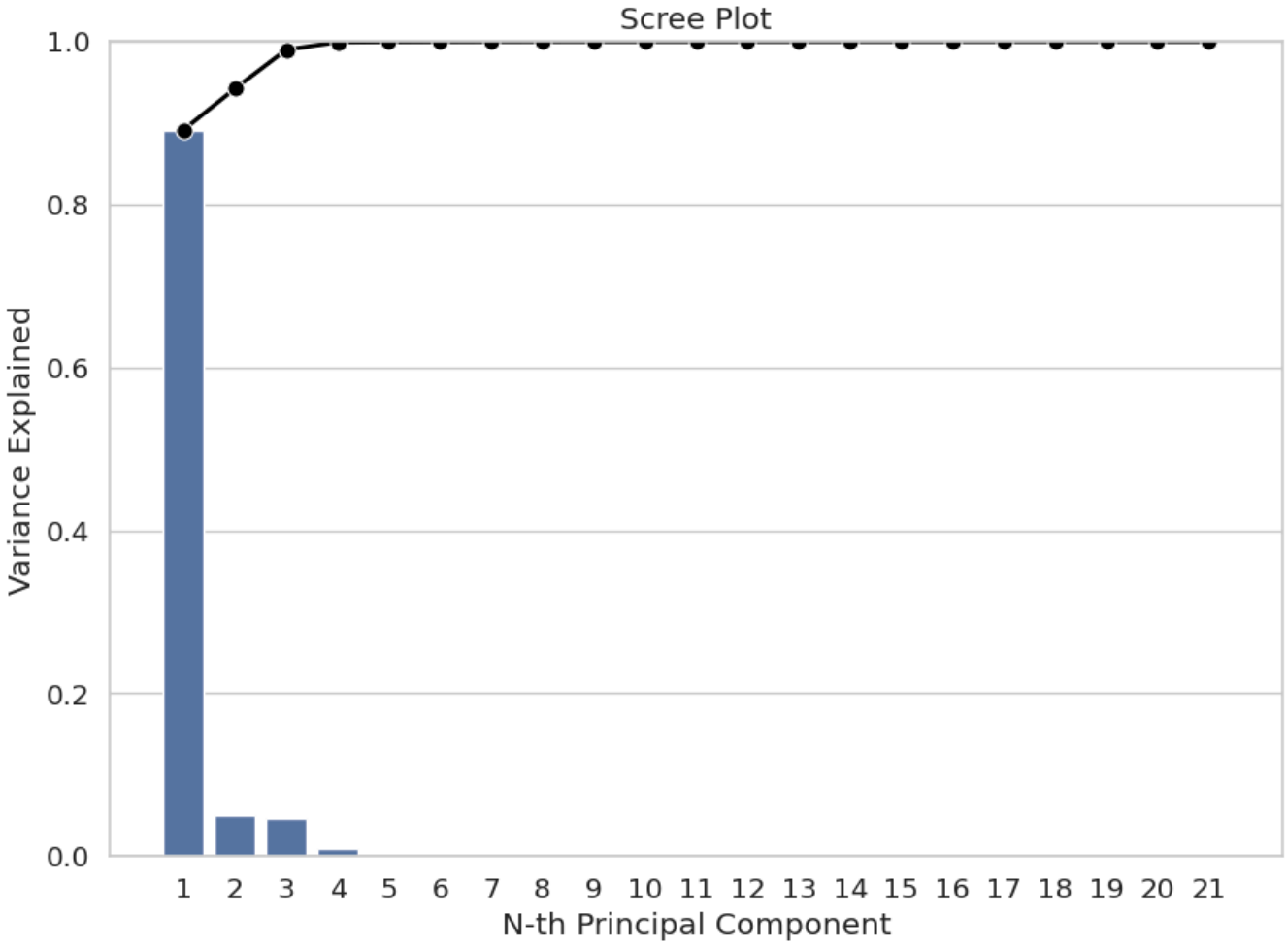
3 rows × 21 columns

Graficamos los resultados

In [ ]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [ ]:
```
PC_components = np.arange(pca.n_components_) + 1

scree = sns.set(style = 'whitegrid', font_scale = 1.2)

fig, ax = plt.subplots(figsize=(10, 7))
```

```
scree = sns.barplot(x = PC_components, y = pca.explained_variance_ratio_, color = 'b')

scree = sns.lineplot(x = PC_components-1,
                     y = np.cumsum(pca.explained_variance_ratio_),
                     color = 'black',
                     linestyle = '-',
                     linewidth = 2,
                     marker = 'o',
                     markersize = 8
                    )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()
```



In [ ]:
```
pca.components_
```

Out[ ]:
```
array([[ 2.30633238e-01,  2.30617978e-01,  2.30639229e-01,
         2.28903551e-01,  2.29428767e-01,  2.26675845e-01,
         2.25737688e-01,  8.57215787e-02,  2.15366537e-01,
         2.29653343e-01,  2.30484067e-01,  2.30598434e-01,
         2.23444203e-01,  2.28134132e-01,  8.32395238e-04,
         2.30144352e-01,  2.29941293e-01,  2.30334707e-01,
         2.30165591e-01,  2.30639046e-01,  2.30755234e-01],
       [-5.96551282e-02, -5.99821992e-02, -5.93072619e-02,
        -1.28125276e-02, -1.78413649e-02, -9.33654189e-02,
        -9.22673247e-02,  8.87161664e-01,  3.17656651e-01,
```

```
 -5.50721170e-02, -4.72842187e-02, -6.12332153e-02,
  1.90448236e-01,  9.69732278e-02, -8.50662820e-02,
 -7.54364979e-02, -5.18499235e-02, -6.30554114e-02,
 -6.84753512e-02, -5.92902571e-02, -3.41104651e-02],
[-5.15230521e-03, -5.07908875e-03, -5.18107563e-03,
 -2.32497043e-03, -2.55762672e-03, -6.70853893e-03,
 -6.44659908e-03,  7.72678857e-02,  2.55243075e-02,
 -4.59332377e-03, -3.84223813e-03, -5.31637879e-03,
  1.43499339e-02,  6.07948774e-03,  9.96364775e-01,
 -6.18874277e-03, -5.24011733e-03, -5.04842751e-03,
 -5.29699353e-03, -5.08653589e-03, -2.35342580e-03],
[ 1.18981287e-02,  2.53777186e-02, -2.49066364e-03,
 -3.03806723e-01, -2.61632017e-01,  3.78574101e-01,
  4.28012301e-01,  3.25003272e-01, -3.03296860e-01,
  2.01236163e-01,  1.06596475e-01, -1.46817460e-02,
 -3.57581453e-01, -2.61167980e-01, -4.29547133e-03,
  9.86459857e-02, -1.71717112e-01,  9.99556986e-02,
  1.15764151e-01,  9.52413233e-03,  5.88419205e-02],
[-6.17412759e-02, -4.88681331e-02, -7.54718340e-02,
 -2.03242123e-01, -2.11657015e-01,  1.70684062e-01,
  2.21200949e-01, -2.82218154e-01,  8.23896195e-01,
  2.65820221e-02, -4.48450786e-02, -7.02960151e-02,
 -1.46007318e-02, -8.44470796e-02,  3.40756864e-04,
  9.54300670e-03, -2.14577246e-01,  1.27596530e-02,
  2.91390234e-03, -6.35181561e-02, -1.17811084e-02],
[-7.77345292e-02, -8.84678170e-02, -6.62525084e-02,
 -1.57480858e-01, -1.46009863e-01,  1.73905106e-01,
  2.08937151e-01, -8.76804665e-02, -1.63996665e-01,
  3.49132856e-01,  9.32645241e-02, -1.24622942e-01,
  2.85467131e-01,  6.28501439e-01,  1.25960929e-03,
 -1.19242453e-01,  2.51165364e-02, -1.23072947e-01,
 -2.25389101e-01, -8.75793227e-02, -3.40653543e-01],
[-2.31098168e-02, -1.07295325e-02, -3.63249849e-02,
  3.60507281e-02, -2.24343608e-02,  1.32195074e-01,
  2.35668786e-01, -4.66705275e-02, -2.18318571e-01,
 -4.08010180e-01, -3.21047397e-01,  4.00940893e-02,
  5.13079017e-01,  8.50869089e-02, -1.29952555e-04,
 -9.62936701e-03, -4.77358808e-01,  1.85361796e-01,
  1.11520159e-01, -2.52515914e-02,  2.37346186e-01],
[-9.00470372e-02, -6.73882555e-02, -1.14189043e-01,
 -5.02698456e-02, -4.26836644e-02, -1.14033529e-01,
 -1.08786404e-01, -5.94831876e-02, -6.85457144e-02,
  3.88139374e-01,  4.12951555e-01, -1.86986785e-01,
  5.12556785e-01, -4.33654052e-01, -1.26698631e-04,
 -8.36937730e-02, -6.33429947e-02, -1.01009377e-01,
 -1.06062610e-02, -7.43404576e-02,  3.21112084e-01],
[-5.08434564e-02, -3.86173356e-02, -6.38713417e-02,
  2.57213395e-01,  2.66611631e-01,  3.11884318e-01,
  4.09030026e-01,  2.00777640e-02,  6.74308615e-03,
 -1.73925775e-01, -1.09299247e-01, -3.50919741e-02,
  1.85522704e-01, -4.06994130e-01, -2.38820143e-04,
 -1.84881465e-01,  3.67306097e-01, -2.54756402e-01,
 -1.52170412e-01, -2.56153921e-02, -2.97117814e-01],
[-1.16880263e-01, -1.19222794e-01, -1.14235621e-01,
 -2.03325568e-01, -1.09510037e-01,  5.68630147e-02,
  1.02876752e-01, -1.76785303e-02, -7.08775085e-03,
 -2.95019317e-01,  4.79000395e-02, -1.34302393e-01,
 -6.74744995e-02,  2.55709389e-01, -9.62260859e-05,
 -1.81252650e-02,  4.41296911e-01, -3.87943444e-01,
  2.93795480e-01, -1.32274750e-01,  5.14654807e-01],
[-1.10365955e-01, -1.37563810e-01, -8.10987579e-02,
  3.84748944e-01,  3.91237548e-01,  1.08938216e-01,
  1.10736955e-01,  3.69248152e-03,  3.83921001e-02,
  2.29666748e-02,  2.89796854e-01, -1.73445931e-01,
 -3.57043028e-01,  1.94760105e-01, -2.65115451e-06,
```

```
 -3.80556409e-01, -3.66618627e-01,  5.76457181e-03,
 -7.25763859e-02, -1.11045012e-01,  2.42350630e-01],
[ 1.05969334e-01,  8.77169575e-02,  1.25011730e-01,
 -4.15437754e-01,  1.02840081e-01,  2.07045862e-02,
 -9.33000480e-03, -1.13206023e-02,  7.33297645e-03,
 -5.26542453e-01,  6.06833425e-01,  7.17046052e-02,
  6.79593173e-02,  3.66396559e-03, -9.59036984e-05,
 -1.31611883e-01, -1.35896260e-03,  1.44006109e-01,
 -2.40466301e-01,  1.14608666e-01, -1.31523153e-01],
[-1.01215691e-01, -1.01604183e-01, -9.96242962e-02,
  4.27426742e-01, -2.79648376e-01,  1.56400913e-01,
 -1.29445351e-01,  3.67801312e-03, -1.67134149e-02,
 -2.37405419e-01,  3.84140685e-01, -1.70966529e-01,
  3.03255310e-02,  1.91656549e-02,  4.93332521e-05,
  4.51756807e-01, -1.21130709e-01, -1.00945584e-01,
  3.01837668e-01, -8.76052231e-02, -3.23929758e-01],
[-7.05264393e-02, -8.79466490e-02, -5.04839080e-02,
 -3.71267489e-01,  5.50823566e-01, -8.73130263e-02,
 -8.07735155e-03, -5.70163651e-03,  5.10500470e-04,
  9.22045156e-02, -2.78665281e-02, -1.03771760e-01,
  4.74519566e-02, -8.20238848e-03,  4.49762457e-05,
  8.85032457e-03, -1.27428216e-01, -5.87940695e-02,
  6.25724590e-01, -1.03319075e-03, -3.19537074e-01],
[-7.55393024e-02, -6.64990547e-02, -8.21006232e-02,
 -1.68231140e-01,  4.20554746e-01,  1.30997099e-01,
 -4.28671348e-02,  4.67895058e-03, -8.63929339e-03,
  9.64677116e-03, -7.25807503e-02, -1.18085415e-01,
 -2.89724125e-02,  2.00033550e-02,  4.16239007e-05,
  6.85405908e-01, -8.64601189e-02, -1.38259622e-01,
 -4.54773988e-01, -8.91955852e-02,  1.64852999e-01],
[-1.86761057e-02,  6.60079928e-02, -1.08108425e-01,
 -8.67582651e-02,  2.78795329e-02,  7.25852542e-01,
 -6.15270960e-01, -4.61035083e-03,  2.67916946e-03,
  2.58664117e-02, -1.12697426e-01, -8.11556861e-03,
  3.83166622e-02, -1.06307705e-02, -3.15185961e-05,
 -1.92497740e-01,  3.50418347e-02,  4.92699466e-02,
  3.78906887e-02,  1.05089975e-01,  4.05768656e-02],
[ 9.40393492e-02, -5.40742333e-01,  7.60870238e-01,
 -1.55035392e-02, -1.96188960e-02,  1.25024367e-01,
 -8.69156801e-02, -6.53043760e-04,  4.81421227e-03,
  2.58885143e-02, -2.10584353e-02,  1.15746505e-01,
  3.70090287e-02, -5.13467868e-02,  4.29086991e-06,
 -1.47859604e-02, -6.42627384e-02, -1.48215465e-01,
  1.13480446e-02, -2.25468840e-01,  1.40821751e-02],
[-4.69215021e-01, -5.23493080e-01, -5.67984764e-02,
  1.60817355e-03, -5.26738620e-03, -3.08692840e-02,
  3.86896765e-02,  2.49974120e-04, -5.33181687e-04,
 -1.16392420e-02, -1.73921256e-02, -1.07254673e-01,
  1.71074730e-03, -2.87674257e-03, -1.00364609e-05,
  4.43900992e-02,  2.59175834e-01,  5.44915222e-01,
 -3.17165773e-02,  3.47301728e-01,  2.08173074e-02],
[ 1.42306048e-01, -3.12743650e-02,  1.66569930e-01,
  2.17591934e-02, -6.09396697e-02, -3.44094347e-02,
  2.85181787e-02, -2.89023413e-04,  1.09386785e-03,
 -1.58548260e-02, -6.28304874e-02, -3.67540191e-01,
 -9.78815100e-03,  1.48181269e-02,  1.51678313e-06,
 -1.21411359e-02, -1.78677233e-01, -3.88331336e-01,
 -3.04333906e-02,  7.87820230e-01,  2.78991608e-02],
[ 7.49037158e-01, -2.93312924e-01, -2.11648879e-01,
 -6.15746527e-04,  6.41239668e-03, -1.26956049e-03,
  3.03412913e-03,  1.11135436e-05, -3.57031997e-04,
 -2.70777737e-02, -5.86395590e-02, -4.33591881e-01,
  4.33740122e-03, -5.73991982e-03,  1.70472016e-05,
  8.28762288e-04,  1.32206989e-01,  2.80129337e-01,
  6.45171781e-04, -1.40989225e-01, -2.73718201e-03],
```

```
[ 2.30956065e-01, -4.52833562e-01, -4.40323633e-01,
 -1.34769500e-03, -2.36779530e-03, -2.08644647e-03,
  2.09296201e-03,  8.76910504e-05,  2.03807591e-04,
  3.88339424e-02,  8.96810944e-02,  6.55207392e-01,
 -5.17277737e-03,  6.54775021e-03,  7.41090655e-06,
  1.32456707e-02, -1.09446919e-01, -2.31399289e-01,
 -5.55966211e-03,  2.07405669e-01,  5.92407392e-03]])
```

Cálculo de la importancia de los componentes

In [ ]:
```
pcsComponents_df = pd.DataFrame(pca.components_.transpose(),
                                columns=pcaSummary.columns,
                                index=df_transformed.columns
                                )
pcsComponents_df
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	
POBTOT	0.230633	-0.059655	-0.005152	0.011898	-0.061741	-0.077735	-0.023110	-0.090047	-0
POBFEM	0.230618	-0.059982	-0.005079	0.025378	-0.048868	-0.088468	-0.010730	-0.067388	-0
POBMAS	0.230639	-0.059307	-0.005181	-0.002491	-0.075472	-0.066253	-0.036325	-0.114189	-0
P_0A2	0.228904	-0.012813	-0.002325	-0.303807	-0.203242	-0.157481	0.036051	-0.050270	0
P_3A5	0.229429	-0.017841	-0.002558	-0.261632	-0.211657	-0.146010	-0.022434	-0.042684	0
P_60YMAS	0.226676	-0.093365	-0.006709	0.378574	0.170684	0.173905	0.132195	-0.114034	0
POB65_MAS	0.225738	-0.092267	-0.006447	0.428012	0.221201	0.208937	0.235669	-0.108786	0
P3HLINHE	0.085722	0.887162	0.077268	0.325003	-0.282218	-0.087680	-0.046671	-0.059483	0
PHOG_IND	0.215367	0.317657	0.025524	-0.303297	0.823896	-0.163997	-0.218319	-0.068546	0
PCON_DISC	0.229653	-0.055072	-0.004593	0.201236	0.026582	0.349133	-0.408010	0.388139	-0
PCON_LIMI	0.230484	-0.047284	-0.003842	0.106596	-0.044845	0.093265	-0.321047	0.412952	-0
PSIND_LIM	0.230598	-0.061233	-0.005316	-0.014682	-0.070296	-0.124623	0.040094	-0.186987	-0
P15YM_AN	0.223444	0.190448	0.014350	-0.357581	-0.014601	0.285467	0.513079	0.512557	0
P15YM_SE	0.228134	0.096973	0.006079	-0.261168	-0.084447	0.628501	0.085087	-0.433654	-0
GRAPROES	0.000832	-0.085066	0.996365	-0.004295	0.000341	0.001260	-0.000130	-0.000127	-0
PEA	0.230144	-0.075436	-0.006189	0.098646	0.009543	-0.119242	-0.009629	-0.083694	-0
PSINDER	0.229941	-0.051850	-0.005240	-0.171717	-0.214577	0.025117	-0.477359	-0.063343	0
PDER_SS	0.230335	-0.063055	-0.005048	0.099956	0.012760	-0.123073	0.185362	-0.101009	-0
TOTHOG	0.230166	-0.068475	-0.005297	0.115764	0.002914	-0.225389	0.111520	-0.010606	-0
POBHOG	0.230639	-0.059290	-0.005087	0.009524	-0.063518	-0.087579	-0.025252	-0.074340	-0
VIVTOT	0.230755	-0.034110	-0.002353	0.058842	-0.011781	-0.340654	0.237346	0.321112	-0

21 rows × 21 columns

In [ ]:
```
print("=============PC1================")
print("** Más importante:")
```

```
print(pcsComponents_df.PC1.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC1.nsmallest(3))
print("=============PC2================")
print("** Más importante:")
print(pcsComponents_df.PC2.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC2.nsmallest(3))
print("=============PC3================")
print("** Más importante:")
print(pcsComponents_df.PC3.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC3.nsmallest(3))
print("=============PC4================")
print("** Más importante:")
print(pcsComponents_df.PC4.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC4.nsmallest(3))
print("=============PC5================")
print("** Más importante:")
print(pcsComponents_df.PC5.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC5.nsmallest(3))
print("=============PC6================")
print("** Más importante:")
print(pcsComponents_df.PC6.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC6.nsmallest(3))
print("=============PC7================")
print("** Más importante:")
print(pcsComponents_df.PC7.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC7.nsmallest(3))
print("=============PC8================")
print("** Más importante:")
print(pcsComponents_df.PC8.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC8.nsmallest(3))
```

```
=============PC1================
** Más importante:
VIVTOT    0.230755
POBMAS    0.230639
POBHOG    0.230639
Name: PC1, dtype: float64
** Menos importante:
GRAPROES    0.000832
P3HLINHE    0.085722
PHOG_IND    0.215367
Name: PC1, dtype: float64
=============PC2================
** Más importante:
P3HLINHE    0.887162
PHOG_IND    0.317657
P15YM_AN    0.190448
Name: PC2, dtype: float64
** Menos importante:
P_60YMAS    -0.093365
POB65_MAS   -0.092267
GRAPROES    -0.085066
Name: PC2, dtype: float64
=============PC3================
```

```
** Más importante:
GRAPROES    0.996365
P3HLINHE    0.077268
PHOG_IND    0.025524
Name: PC3, dtype: float64
** Menos importante:
P_60YMAS    -0.006709
POB65_MAS   -0.006447
PEA         -0.006189
Name: PC3, dtype: float64
=============PC4=================
** Más importante:
POB65_MAS    0.428012
P_60YMAS     0.378574
P3HLINHE     0.325003
Name: PC4, dtype: float64
** Menos importante:
P15YM_AN   -0.357581
P_0A2      -0.303807
PHOG_IND   -0.303297
Name: PC4, dtype: float64
=============PC5=================
** Más importante:
PHOG_IND     0.823896
POB65_MAS    0.221201
P_60YMAS     0.170684
Name: PC5, dtype: float64
** Menos importante:
P3HLINHE   -0.282218
PSINDER    -0.214577
P_3A5      -0.211657
Name: PC5, dtype: float64
=============PC6=================
** Más importante:
P15YM_SE     0.628501
PCON_DISC    0.349133
P15YM_AN     0.285467
Name: PC6, dtype: float64
** Menos importante:
VIVTOT     -0.340654
TOTHOG     -0.225389
PHOG_IND   -0.163997
Name: PC6, dtype: float64
=============PC7=================
** Más importante:
P15YM_AN     0.513079
VIVTOT       0.237346
POB65_MAS    0.235669
Name: PC7, dtype: float64
** Menos importante:
PSINDER     -0.477359
PCON_DISC   -0.408010
PCON_LIMI   -0.321047
Name: PC7, dtype: float64
=============PC8=================
** Más importante:
P15YM_AN     0.512557
PCON_LIMI    0.412952
PCON_DISC    0.388139
Name: PC8, dtype: float64
** Menos importante:
P15YM_SE    -0.433654
PSIND_LIM   -0.186987
POBMAS      -0.114189
Name: PC8, dtype: float64
```

Conclusiones al 29 de septiembre

Al terminar este análisis exploratorio de los datos EDA , corroboramos que es una etapa crucial que permite tener una comprensión más profunda de la estructura y características de nuestros datos , a través de técnicas y visualizaciones como el análisis univariante y bi/multivariante, el EDA ayuda a identificar patrones, tendencias y relaciones significativas que pueden influir en los resultados del modelado.

Además, este proceso nos resultó de utilidad para detectar y justificar operaciones de preprocesamiento, como el manejo de valores faltantes y atípicos, así como la reducción de la alta cardinalidad, lo que contribuye a mejorar la calidad de los datos y la eficacia de nuestro modelo y aplicación.

Cabe destacar que el análisis exploratorio es una de las fases fundamentales y que más tiempo lleva a cualquier analista de datos, ya sea en proyectos de IA, Machine Learning, Aprendizaje Profundo, etc. En nuestro caso particular, pudimos observar que de inicio, las bases de datos parecían limpias.

Sin embargo, a través de este proceso, al observar las medidas de distribución central y la descripción de tipos de datos, pudimos percibir que eran “objetos”, lo cual implicó ahondar en las razones por las cuales se obtenía esta lectura.

Así pues, identificamos caracteres no coincidentes con el rango de los datos y exploramos diferentes formas de tratarlo, apegándonos al caso de estudio. En este sentido, al trabajar con datos poblacionales, no fue posible realizar la imputación de datos. Ya que buscamos apegarnos a la información que realmente se pudo obtener de las bases de datos, esto al ser de caracter poblacional.

Finalmente, el análisis exploratorio nos permitirá clusterizar la información poblacional. Para luego comprender cómo estos datos tienen un impacto en las Islas de Calor y así determinar el nivel de criticidad, esto considerando los factores que nos permitirán realizar planes de acción sustentados en datos fehacientes.

Fuentes consultadas al 29 de septiembre

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

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

Torre, L. et. al. (2023). Metodología para identificar y cuantificar islas de calor en entornos urbanos con

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

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

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

Avance 2 (Entrega: 6 octubre 2024)

Parte 5: Construcción - Procesamiento por Codificación y Creación de nuevas características

1. Procesamiento de datos

Aplicamos operaciones comunes para convertir los datos crudos en un conjunto de variables útiles para el aprendizaje automático.

1.1 Codificación

De forma ilustrativa, se genera una codificación del tipo Label Encoding, para el nombre de la entidad, que era nuestra única variable categórica. Sin embargo, solamente se hace para seguir la lógica del proceso, lo que sería necesario de no contar con la columna “ENTIDAD”, que ya contiene de forma numérica un equivalente para los nombres de la entidad. A continuación, se muestra la codificación realizada en nuestro DataFrame.

Asimismo, observamos los primeros registros obtenidos y que ya están codificados, sin embargo, dicho cálculo será descartado posteriormente por lo ya comentado. En este punto cabe destacar que seleccionamos la codificación conocida como Label Encoding porque no aumenta la dimensionalidad del set de datos, contrario a One Hot Encoder. Lo anterior es relevante dada la cantidad de registros y columnas de nuestro conjunto.

Armamos los DF completos para cada entidad

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# Leemos los datos por Estado
# Hidalgo
df_hidalgo = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2'
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
```

```
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
df_hidalgo.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH
0	13	Hidalgo	0	0000	0	3082841	1601462	1481379	134738	66770	...	
1	13	Hidalgo	0	0000	0	22268	11563	10705	1241	593	...	
2	13	Hidalgo	1	0000	0	439	229	210	21	6	...	
3	13	Hidalgo	1	0043	0	439	229	210	21	6	...	
4	13	Hidalgo	1	0043	1	92	54	38	6	*	...	

5 rows × 102 columns

In [ ]:

```
# Leemos los datos por Estado
# CDMX 1
df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/hea
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
           'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
           'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2'
          'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
           'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
           'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
          'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
          'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
           'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
          'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC','P_0A2', 'P_0A2_F', 'P_0A
df_cdmx1.head()
```

```
<ipython-input-78-097fdbaa06da>:3: DtypeWarning: Columns (178) have mixed types. Specify dtyp
e option on import or set low_memory=False.
  df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/head
s/main/CDMX.csv",
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VP
0	9.0	Ciudad de México	0.0	0000	0.0	9209944.0	4805017	4404927	267151	131720	...	
1	9.0	Ciudad de México	0.0	0000	0.0	432205.0	227255	204950	11784	5702	...	
2	9.0	Ciudad de México	1.0	0000	0.0	432205.0	227255	204950	11784	5702	...	
3	9.0	Ciudad de México	1.0	0010	0.0	3183.0	1695	1488	60	32	...	

4	9.0	Ciudad de México	1.0	0010	1.0	159.0	86	73	*	*	...

5 rows × 102 columns

In [ ]:
```
# Leemos los datos por Estado
# CDMX 2
df_cdmx2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/hea
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRI
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUAD
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_
df_cdmx2.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_
0	9	Ciudad de México	1	5179	25	42	19	23	*	0	...	
1	9	Ciudad de México	1	5179	27	27	15	12	3	3	...	
2	9	Ciudad de México	1	5179	28	23	13	10	*	0	...	
3	9	Ciudad de México	1	5179	29	23	10	13	0	0	...	
4	9	Ciudad de México	1	5179	30	5	*	*	*	*	...	

5 rows × 102 columns

In [ ]:
```
# Leemos los datos por Estado
# EdoMex 1
df_edomex1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/h
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRI
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUAD
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
```

```
            'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
             'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
            'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_
df_edomex1.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VP
0	15	México	0	0	0	16992418	8741123	8251295	724133	357766	...	
1	15	México	0	0	0	67872	35255	32617	3782	1768	...	
2	15	México	1	0	0	5988	3148	2840	300	132	...	
3	15	México	1	127	0	3373	1796	1577	180	86	...	
4	15	México	1	127	1	0	0	0	0	0	...	

5 rows × 102 columns

In [ ]:

```
# Leemos los datos por Estado
# EdoMex 2
df_edomex2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_
df_edomex2.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_
0	15	México	1	2502	70	58	31	27	3	*	...	
1	15	México	1	2502	71	43	22	21	3	*	...	
2	15	México	1	2502	72	55	28	27	0	0	...	
3	15	México	1	2502	73	38	24	14	*	0	...	
4	15	México	1	2502	74	49	31	18	*	*	...	

5 rows × 102 columns

In [ ]:

```
# Leemos los datos por Estado
# EdoMex 3
df_edomex3 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
```

```
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'P
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
df_edomex3.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_
0	15	México	1	132	58	191	104	87	8	3	...	
1	15	México	1	132	59	27	16	11	*	0	...	
2	15	México	1	132	60	83	39	44	3	*	...	
3	15	México	1	132	61	118	63	55	3	*	...	
4	15	México	1	132	62	116	56	60	4	*	...	

5 rows × 102 columns

In [ ]:

```
# Leemos los datos por Estado
# EdoMex 4
df_edomex4 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                 usecols=['NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'POBFEM', 'P
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'P
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
df_edomex4.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VPH_
0	15	México	1	453	5	24	14	10	0	0	...	
1	15	México	1	453	6	52	35	17	3	*	...	
2	15	México	1	453	7	0	0	0	0	0	...	
3	15	México	1	453	8	30	15	15	0	0	...	
4	15	México	1	453	9	11	4	7	0	0	...	

5 rows × 102 columns

Concatenamos los DF

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:
```
df_list_3estados = [df_hidalgo,df_cdmx1,df_cdmx2,df_edomex1,df_edomex2,df_edomex3,df_edomex
df_concat_3estados = pd.concat(df_list_3estados, ignore_index=True)
df_concat_3estados.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VP
0	13.0	Hidalgo	0.0	0000	0.0	3082841.0	1601462	1481379	134738	66770	...	
1	13.0	Hidalgo	0.0	0000	0.0	22268.0	11563	10705	1241	593	...	
2	13.0	Hidalgo	1.0	0000	0.0	439.0	229	210	21	6	...	
3	13.0	Hidalgo	1.0	0043	0.0	439.0	229	210	21	6	...	
4	13.0	Hidalgo	1.0	0043	1.0	92.0	54	38	6	*	...	

5 rows × 102 columns

In [ ]:
```
#Verificamos el tipo de variables del DF
df_concat_3estados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253244 entries, 0 to 253243
Columns: 102 entries, ENTIDAD to VPH_SINTIC
dtypes: float64(5), object(97)
memory usage: 197.1+ MB
```

De forma ilustrativa, codificamos la variable "NOM_ENT"

In [ ]:
```
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[ ]:

	ENTIDAD	NOM_ENT	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	...	VP
0	13.0	1	0.0	0000	0.0	3082841.0	1601462	1481379	134738	66770	...	
1	13.0	1	0.0	0000	0.0	22268.0	11563	10705	1241	593	...	
2	13.0	1	1.0	0000	0.0	439.0	229	210	21	6	...	
3	13.0	1	1.0	0043	0.0	439.0	229	210	21	6	...	
4	13.0	1	1.0	0043	1.0	92.0	54	38	6	*	...	

5 rows × 102 columns

In [ ]:
```
#Verificamos el tipo de variables del DF
df_concat_3copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253244 entries, 0 to 253243
Columns: 102 entries, ENTIDAD to VPH_SINTIC
dtypes: float64(5), int64(1), object(96)
memory usage: 197.1+ MB
```

Sin embargo, seguiremos utilizando "ENTIDAD" que es una variable codificada desde la BD original de INEGI

1.2 Creación de nuevas características

Para la ingeniería de características, consideramos necesario analizar variables específicas en el conjunto original de datos, contemplando cuatro diferentes particiones, donde las tres primeras nos ayudan a identificar factores de vulnerabilidad y la última, atenuantes. A continuación explicaremos con mayor detalle el proceso que seguimos

En esta primera generación de nuevas características partimos de la concatenación de los tres Estados de interés, su respectiva limpieza e imputación designada en la etapa anterior

1 - Población indígena

En este primer listado, tomamos las columnas:

P3HLINHE: Población de 3 años y más que habla alguna lengua indígena y no habla español

PHOG_IND: Población en hogares censales indígenas

Donde contemplamos que dicha población, para cada registro en el set de datos, tendría cierto factor de vulnerabilidad adicional al resto del total, que sería especialmente importante considerar. Así pues, si al menos alguno de los dos (condición lógica OR) registros contienen datos diferentes a cero, se sumarán; obteniendo el total de ambos como un valor del tipo entero.

2 - Edad

Contemplando las columnas:

P_0A2: Población de 0 a 2 años

P_3A5: Población de 3 a 5 años

P_60YMAS: Población de 60 años y más

Seguimos una lógica parecida al caso anterior, donde identificamos cierto rango de edades que podrían posicionar a esa proporción de la población total como vulnerable sobre el resto. Por tanto, si hay valores en al menos alguna de las tres columnas, se sumarán las personas que yacen en dicha condición, en su defecto, el valor será cero.

3 - Discapacidad

Donde definimos como relevantes las columnas:

PCON_DISC: Población con discapacidad

PCON_DISC: Población con discapacidad

PCON_LIMI: Población con limitación

Por tanto, contemplando un proceder similar al anteriormente expuesto, seccionamos las personas que poseen alguna discapacidad en general o también cierta limitación. Se abarca, por tanto, el aspecto físico y cognitivo que, sin duda podría resultar en un mayor índice respecto a la vulnerabilidad

4 - Analfabetismo

En este punto consideramos las columnas:

P15YM_AN: Población de 15 años y más analfabeta

P15YM_SE: Población de 15 años y más sin escolaridad

PSINDER: Población sin afiliación a servicios de salud

Dada la condición y procesamiento lógico ya manifestado, realizamos lo mismo para esta partición de la población total, considerando la vulnerabilidad relativa a falta de educación y servicios de salud

5 - Vivienda

En términos de vivienda, consideramos en esta primera partición las columnas:

VPH_PISOTI: Viviendas particulares habitadas con piso de tierra

VPH_NDEAED: Viviendas particulares habitadas que no disponen de energía eléctrica, agua entubada, ni drenaje

Se destacan estas variables porque nos ayudan a entender las condiciones de la vivienda para las personas en cuestión. Además, nos ayuda a revelar cierto indicio de su situación socioeconómica, donde, de pertenecer a dicha partición, podría considerar como un factor adicional de vulnerabilidad

6 - Posesión de bienes

Para este punto, respecto a lo que se puede poseer en la vivienda, consideramos:

VPH_SINRTV: Viviendas particulares habitadas sin radio ni televisor

VPH_SINLTC: Viviendas particulares habitadas sin línea telefónica fija ni teléfono celular

VPH_SINCINT: Viviendas particulares habitadas sin computadora ni Internet

En este punto nos referimos más a la carencia de bienes, los cuales si bien podrían resultar un atenuante en el grado de vulnerabilidad, al no disponerlos, resultaría en un efecto contrario

7 - Atenunante de vulnerabilidad

Finalmente, para esta última característica atenuante en cuanto a la vulnerabilidad, consideramos:

VPH_REFRI: Viviendas particulares habitadas que disponen de refrigerador

VPH_AUTOM: Viviendas particulares habitadas que disponen de automóvil o camioneta

VPH_MOTO: Viviendas particulares habitadas que disponen de motocicleta o motoneta

VPH_BICI: Viviendas particulares habitadas que disponen de bicicleta como medio de transporte

En esta última sumarización se entienden los factores que podrían atenuar una situación vulnerable, como también revelar más datos sobre una posición socioeconómica más saludable.

1.3 Procedimiento para la generación de nuevas características

Creación de característica de vulnerabilidad: "Población de 3 años y más que habla alguna lengua indígena y no habla español" o "Población en hogares censales indígenas"

In [ ]:
```
df_cat = df_concat_3estados.copy()

# Eliminamos los caracteres de tipo *
df_cat.replace('*', np.nan, inplace=True)
df_cat.head()
df_cat.dropna(how='all', inplace = True)

# Hacemos el cambio para que todas las variables sean numéricas
df_cat= df_cat.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
df_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 253243 entries, 0 to 253243
Columns: 102 entries, ENTIDAD to VPH_SINTIC
dtypes: int64(102)
memory usage: 199.0 MB
```

In [ ]:
```
df_indigena = df_cat.copy()
for index, row in df_indigena.iterrows():
    if row['P3HLINHE'] != 0 or row['PHOG_IND'] != 0:
        df_indigena.at[index, 'Vuln_Indigena'] = (int(df_indigena.at[index,'P3HLINHE']) + 

print(df_indigena[['Vuln_Indigena', 'P3HLINHE', 'PHOG_IND']].head())

bins = [0, 1000, 3000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_indigena['Vuln_Indigena_Discreta'] = pd.cut(df_indigena['Vuln_Indigena'], bins=bins, lab
df_indigena['Vuln_Indigena_Discreta_Int'] = df_indigena['Vuln_Indigena_Discreta'].cat.code

print(df_indigena[['Vuln_Indigena', 'Vuln_Indigena_Discreta','Vuln_Indigena_Discreta_Int']
```

```
   Vuln_Indigena  P3HLINHE  PHOG_IND
0       613190.0     28497    584693
1          203.0         0       203
2           12.0         0        12
3           12.0         0        12
4            NaN         0         0
   Vuln_Indigena Vuln_Indigena_Discreta  Vuln_Indigena_Discreta_Int
0       613190.0                   Alto                           3
1          203.0                   Bajo                           1
2           12.0                   Bajo                           1
3           12.0                   Bajo                           1
4            NaN                    NaN                           0
```

In [ ]:
```
df_indigena = df_indigena['Vuln_Indigena'].apply(pd.to_numeric, errors='coerce').fillna(0)
df_indigena.head()
```

Out[ ]:

	Vuln_Indigena
0	613190
1	203
2	12
3	12
4	0

dtype: int64

Creamos la característica de vulnerabilidad: "Población de 0 a 2 años" o "Población de 3 a 5 años" o "Población de 60 años y más"

In [ ]:
```
df_edad = df_cat.copy()
for index, row in df_edad.iterrows():
    if row['P_0A2'] != 0 or row['P_3A5'] != 0 or row['P_60YMAS'] != 0:
        df_edad.at[index, 'Vuln_Edad'] = (int(df_edad.at[index,'P_0A2']) + int(df_edad.at[

print(df_edad[['Vuln_Edad', 'P_0A2', 'P_3A5', 'P_60YMAS']].head())
```

```
   Vuln_Edad   P_0A2   P_3A5  P_60YMAS
0   674899.0  134738  156486    383675
1     5105.0    1241    1239      2625
2      103.0      21      18        64
3      103.0      21      18        64
4       22.0       6       5        11
```

In [ ]:
```
df_edad = df_edad['Vuln_Edad'].apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
df_edad.head()
```

Out[ ]:

	Vuln_Edad
0	674899
1	5105
2	103
3	103
4	22

dtype: int64

Creamos la característica de vulnerabilidad: "Población con discapacidad" o "Población con limitación"

In [ ]:
```
df_disc_lim = df_cat.copy()
for index, row in df_disc_lim.iterrows():
    if row['PCON_DISC'] != 0 or row['PCON_LIMI'] != 0:
        df_disc_lim.at[index, 'Vuln_Disc_Lim'] = (int(df_disc_lim.at[index,'PCON_DISC']) +

print(df_disc_lim[['Vuln_Disc_Lim', 'PCON_DISC', 'PCON_LIMI']].head())
```

```
bins = [0, 1000, 3000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_disc_lim['Vuln_Disc_Discreta'] = pd.cut(df_disc_lim['Vuln_Disc_Lim'], bins=bins, labels=
df_disc_lim['Vuln_Disc_Discreta_Int'] = df_disc_lim['Vuln_Disc_Discreta'].cat.codes + 1

print(df_disc_lim[['Vuln_Disc_Lim', 'Vuln_Disc_Discreta','Vuln_Disc_Discreta_Int']].head()
```

```
   Vuln_Disc_Lim  PCON_DISC  PCON_LIMI
0       580203.0     166965     413238
1         3545.0        974       2571
2          139.0         43         96
3          139.0         43         96
4           30.0          9         21
   Vuln_Disc_Lim Vuln_Disc_Discreta  Vuln_Disc_Discreta_Int
0       580203.0               Alto                       3
1         3545.0               Alto                       3
2          139.0               Bajo                       1
3          139.0               Bajo                       1
4           30.0               Bajo                       1
```

In [ ]:
```
df_disc_lim = df_disc_lim['Vuln_Disc_Lim'].apply(pd.to_numeric, errors='coerce').fillna(0)
df_disc_lim.head()
```

Out[ ]:

	Vuln_Disc_Lim
0	580203
1	3545
2	139
3	139
4	30

dtype: int64

Creamos la característica de vulnerabilidad: "Población de 15 años y más analfabeta" o "Población de 15 años y más sin escolaridad" o "Población sin afiliación a servicios de salud"

In [ ]:
```
df_esc_salud = df_cat.copy()
for index, row in df_esc_salud.iterrows():
    if row['P15YM_AN'] != 0 or row['P15YM_SE'] != 0 or row['PSINDER']:
        df_esc_salud.at[index, 'Vuln_Esc_Salud'] = (int(df_esc_salud.at[index,'P15YM_AN'])

print(df_esc_salud[['Vuln_Esc_Salud', 'P15YM_AN', 'P15YM_SE', 'PSINDER']].head())


# Bajo 0-1000, Medio 1000-15000, Alto 15000<
bins = [0, 1000, 15000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_esc_salud['Vuln_Salud_Discreta'] = pd.cut(df_esc_salud['Vuln_Esc_Salud'], bins=bins, lab
df_esc_salud['Vuln_Salud_Discreta_Int'] = df_esc_salud['Vuln_Salud_Discreta'].cat.codes +
```

```
print(df_esc_salud[['Vuln_Esc_Salud', 'Vuln_Salud_Discreta','Vuln_Salud_Discreta_Int']].he
```

```
   Vuln_Esc_Salud  P15YM_AN  P15YM_SE  PSINDER
0       1222960.0    151311    143099   928550
1         13447.0      1755      1894     9798
2           251.0        10        14      227
3           251.0        10        14      227
4            50.0         0         3       47
   Vuln_Esc_Salud Vuln_Salud_Discreta  Vuln_Salud_Discreta_Int
0       1222960.0                Alto                        3
1         13447.0               Medio                        2
2           251.0                Bajo                        1
3           251.0                Bajo                        1
4            50.0                Bajo                        1
```

In [ ]:
```
df_esc_salud = df_esc_salud['Vuln_Esc_Salud'].apply(pd.to_numeric, errors='coerce').fillna
df_esc_salud.head()
```

Out[ ]:

	Vuln_Esc_Salud
0	1222960
1	13447
2	251
3	251
4	50

dtype: int64

Creamos la característica de vulnerabilidad: "Viviendas particulares habitadas con piso de tierra" o "Viviendas particulares habitadas que no disponen de energía eléctrica, agua entubada, ni drenaje"

In [ ]:
```
df_hog_serv = df_cat.copy()
for index, row in df_hog_serv.iterrows():
    if row['VPH_PISOTI'] != 0 or row['VPH_NDEAED'] != 0 :
        df_hog_serv.at[index, 'Vuln_Hog_Serv'] = (int(df_hog_serv.at[index,'VPH_PISOTI']) +

print(df_hog_serv[['Vuln_Hog_Serv', 'VPH_PISOTI', 'VPH_NDEAED']].head())
```

```
   Vuln_Hog_Serv  VPH_PISOTI  VPH_NDEAED
0        26472.0       24402        2070
1          115.0         102          13
2            NaN           0           0
3            NaN           0           0
4            NaN           0           0
```

In [ ]:
```
df_hog_serv = df_hog_serv['Vuln_Hog_Serv'].apply(pd.to_numeric, errors='coerce').fillna(0)
df_hog_serv.head()
```

Out[ ]:

	Vuln_Hog_Serv
0	26472

1	115
2	0
3	0
4	0

dtype: int64

Creamos la característica de vulnerabilidad: "Viviendas particulares habitadas sin radio ni televisor" o "Viviendas particulares habitadas sin línea telefónica fija ni teléfono celular" o "Viviendas particulares habitadas sin computadora ni Internet"

In [ ]:
```
df_hog_connect = df_cat.copy()
for index, row in df_hog_connect.iterrows():
    if row['VPH_SINRTV'] != 0 or row['VPH_SINLTC'] != 0 or row['VPH_SINCINT']:
        df_hog_connect.at[index, 'Vuln_Hog_Connect'] = (int(df_hog_connect.at[index,'VPH_S

print(df_hog_connect[['Vuln_Hog_Connect', 'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT']].head
```

	Vuln_Hog_Connect	VPH_SINRTV	VPH_SINLTC	VPH_SINCINT
0	621072.0	51691	102005	467376
1	4707.0	322	680	3705
2	77.0	9	8	60
3	77.0	9	8	60
4	21.0	4	3	14

In [ ]:
```
df_hog_connect = df_hog_connect['Vuln_Hog_Connect'].apply(pd.to_numeric, errors='coerce').
df_hog_connect.head()
```

Out[ ]:

	Vuln_Hog_Connect
0	621072
1	4707
2	77
3	77
4	21

dtype: int64

Creamos la característica atenuante: "Viviendas particulares habitadas que disponen de refrigerador" o "Viviendas particulares habitadas que disponen de automóvil o camioneta" o "Viviendas particulares habitadas que disponen de motocicleta o motoneta" o "Viviendas particulares habitadas que disponen de bicicleta como medio de transporte"

In [ ]:
```
df_hog_atenuante = df_cat.copy()
for index, row in df_hog_atenuante.iterrows():
    if row['VPH_REFRI'] != 0 or row['VPH_AUTOM'] != 0 or row['VPH_MOTO'] != 0 or row['VPH_
        df_hog_atenuante.at[index, 'Vuln_Hog_Atenuante'] = (int(df_hog_atenuante.at[index,

print(df_hog_atenuante[['Vuln_Hog_Atenuante', 'VPH_REFRI', 'VPH_AUTOM', 'VPH_MOTO', 'VPH_B
```

	Vuln_Hog_Atenuante	VPH_REFRI	VPH_AUTOM	VPH_MOTO	VPH_BICI
0	1365712.0	708258	378990	86059	192405
1	12162.0	4986	3572	1273	2331
2	238.0	114	71	10	43
3	238.0	114	71	10	43
4	33.0	22	8	0	3

In [ ]:
```
df_hog_atenuante = df_hog_atenuante['Vuln_Hog_Atenuante'].apply(pd.to_numeric, errors='coe
df_hog_atenuante.head()
```

Out[ ]:

	Vuln_Hog_Atenuante
0	1365712
1	12162
2	238
3	238
4	33

dtype: int64

1.4 Generación de grupos

El procesamiento y generación de grupos anteriormente expuestos fue de relevancia para poder particionar del total poblacional, aquellos que se podrían ver especialmente vulnerables, generando un totalizador para lo designado. Dicho resumen nos permitirá agrupar y cuantificar las personas que designamos vulnerables, lo que se detalla más adelante.

	Vuln_Indigena	Vuln_Edad	Vuln_Disc_Lim	Vuln_Esc_Salud	Vuln_Hog_Serv	Vuln_Hog_Connect	Vuln_Hog_Atenuante
count	2.532430e+05	2.532430e+05	2.532430e+05	2.532430e+05	253243.000000	2.532430e+05	2.532430e+05
mean	2.997133e+01	1.134401e+02	8.965603e+01	1.893087e+02	1.988793	5.853404e+01	2.520060e+02
std	2.529480e+03	8.533440e+03	6.654507e+03	1.480060e+04	206.042487	4.919776e+03	1.854559e+04
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000e+00
25%	0.000000e+00	4.000000e+00	0.000000e+00	7.000000e+00	0.000000	0.000000e+00	1.200000e+01
50%	0.000000e+00	1.400000e+01	1.000000e+01	2.200000e+01	0.000000	5.000000e+00	3.200000e+01
75%	5.000000e+00	3.200000e+01	2.500000e+01	5.100000e+01	0.000000	1.400000e+01	6.700000e+01
max	1.031962e+06	3.478198e+06	2.685381e+06	6.497550e+06	96753.000000	2.232460e+06	7.509537e+06

Imagen 9. Medidas de distribución de las variables generadas. *(Elaboración propia, 2024)*

Consecuentemente, después de añadir o concatenar el total de población vulnerable y el atenuante considerado en el set de datos ya procesado, obtenemos las medidas de distribución central para entender mejor el comportamiento de los datos y cuántas personas podrían considerarse vulnerables, que es el enfoque que se tiene en el proyecto integrador, respecto a la justicia social y establecer un nivel de criticidad en las acciones.

Agregamos las características en un DF

In [ ]:
```
df_cat_vuln=df_cat.copy()
df_cat_vuln = df_cat_vuln.drop(columns=['NOM_ENT'])
df_cat_vuln = pd.concat([df_cat_vuln, df_indigena, df_edad, df_esc_salud, df_disc_lim, df_h
df_cat_vuln.head()
```

Out[ ]:

	ENTIDAD	LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	P_0A2_F	P_0A2_M	...	VPH_
0	13	0	0	0	3082841	1601462	1481379	134738	66770	67968	...	
1	13	0	0	0	22268	11563	10705	1241	593	648	...	
2	13	1	0	0	439	229	210	21	6	15	...	
3	13	1	43	0	439	229	210	21	6	15	...	
4	13	1	43	1	92	54	38	6	0	4	...	

5 rows × 108 columns

Analizamos el contenido del DF

In [ ]:
```
df_cat_vuln.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 253243 entries, 0 to 253243
Columns: 108 entries, ENTIDAD to Vuln_Hog_Atenuante
dtypes: int64(108)
memory usage: 218.7 MB
```

In [ ]:
```
df_cat_vuln.describe()
```

Out[ ]:

	ENTIDAD	LOC	AGEB	MZA	POBTOT	POBFEM	
count	253243.000000	253243.000000	253243.000000	253243.000000	2.532430e+05	2.532430e+05	2.5
mean	13.034212	13.511651	1018.056250	23.737864	5.383219e+02	2.784972e+02	2.5
std	2.569214	32.132339	1173.825449	40.627002	4.049589e+04	2.090770e+04	1.9
min	9.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.0
25%	9.000000	1.000000	168.000000	8.000000	2.900000e+01	1.500000e+01	1.4
50%	15.000000	1.000000	601.000000	18.000000	7.100000e+01	3.700000e+01	3.4
75%	15.000000	7.000000	1449.000000	31.000000	1.460000e+02	7.600000e+01	7.0
max	15.000000	300.000000	5925.000000	800.000000	1.699242e+07	8.741123e+06	8.2

8 rows × 108 columns

Verificamos los valores de las columnas añadidas y sus medidas de distribución

In [ ]:
```
df_cat_vuln[['Vuln_Indigena', 'Vuln_Edad', 'Vuln_Disc_Lim', 'Vuln_Esc_Salud', 'Vuln_Hog_Se
```

Out[ ]:

	Vuln_Indigena	Vuln_Edad	Vuln_Disc_Lim	Vuln_Esc_Salud	Vuln_Hog_Serv	Vuln_Hog_Connect	Vul
0	613190	674899	580203	1222960	26472	621072	
1	203	5105	3545	13447	115	4707	
2	12	103	139	251	0	77	
3	12	103	139	251	0	77	
4	0	22	30	50	0	21	

In [ ]:

```
df_cat_vuln[['Vuln_Indigena', 'Vuln_Edad', 'Vuln_Disc_Lim', 'Vuln_Esc_Salud', 'Vuln_Hog_Se
```

Out[ ]:

	Vuln_Indigena	Vuln_Edad	Vuln_Disc_Lim	Vuln_Esc_Salud	Vuln_Hog_Serv	Vuln_Hog_Conne
count	2.532430e+05	2.532430e+05	2.532430e+05	2.532430e+05	253243.000000	2.532430e+(
mean	2.997133e+01	1.134401e+02	8.965603e+01	1.893087e+02	1.988793	5.853404e+(
std	2.529480e+03	8.533440e+03	6.654507e+03	1.480060e+04	206.042487	4.919776e+(
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000e+(
25%	0.000000e+00	4.000000e+00	0.000000e+00	7.000000e+00	0.000000	0.000000e+(
50%	0.000000e+00	1.400000e+01	1.000000e+01	2.200000e+01	0.000000	5.000000e+(
75%	5.000000e+00	3.200000e+01	2.500000e+01	5.100000e+01	0.000000	1.400000e+(
max	1.031962e+06	3.478198e+06	2.685381e+06	6.497550e+06	96753.000000	2.232460e+(

1.4 Discretización o binning

Agregamos una columna adicional a nuestro resultado anterior para representar los rangos discretizados por rangos como valores enteros. Esto nos permitirá realizar análisis cuantitativos más fácilmente. En este caso realizamos la discretización en 3 grupos dependiendo de los valores obtenidos en 'Vuln_Esc_Salud' , utilizando los siguientes bines = [0, 1000, 15000, np.inf] siendo:

- **Alto:** Valores en 'Vuln_Esc_Salud' de 0 a 1000, sin contemplar 1000
- **Medio:** Valores en 'Vuln_Esc_Salud' de 0 a 15000, sin contemplar 15000
- **Bajo:** Valores en 'Vuln_Esc_Salud' de 15000 al infinito

```
# Bajo 0-1000, Medio 1000-15000, Alto 15000<
bins = [0, 1000, 15000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_esc_salud['Vuln_Salud_Discreta'] = pd.cut(df_esc_salud['Vuln_Esc_Salud'], bins=bins, labels=labels)
df_esc_salud['Vuln_Salud_Discreta_Int'] = df_esc_salud['Vuln_Salud_Discreta'].cat.codes + 1

print(df_esc_salud[['Vuln_Esc_Salud', 'Vuln_Salud_Discreta','Vuln_Salud_Discreta_Int']].head())
```

Imagen 10. Discretización de variables. *(Elaboración propia, 2024)*

Además se colocó una columna extra en la que se muestra el grupo al que pertenece pero en forma de número entero para facilitar su uso en futuras operaciones.

	Vuln_Esc_Salud	Vuln_Salud_Discreta	Vuln_Salud_Discreta_Int
0	1222960.0	Alto	3
1	13447.0	Medio	2
2	251.0	Bajo	1
3	251.0	Bajo	1
4	50.0	Bajo	1

Imagen 11. Resultados de discretización. *(Elaboración propia, 2024)*

Parte 6: Normalización - Procesamiento por transformación y escalamiento

A continuación, se muestra el escalamiento realizado al DataFrame; esto a través de StandardScaler.

Es importante realizar escalamiento o al menos tener un abordaje y verificar que los resultados sean los esperados puesto que, posteriormente dado el modelo de clusterización seleccionado, podría haber cierta sensibilidad a valores que no estén normalizados, afectando el rendimiento general del mismo.

Asimismo, normalizar puede ser relevante en datos poblacionales para entender mejor el conjunto de datos, en perspectiva de comparar las distintas densidades poblacionales y que se mantenga una proporción que pueda ser representativa.

In [ ]:
```
scaler = StandardScaler()
df_transformed = pd.DataFrame(scaler.fit_transform(df2), columns=df2.columns)
df_transformed.head()
```

Out[ ]:

	POBTOT	POBFEM	POBMAS	P_0A2	P_3A5	P_60YMAS	POB65_MAS	P3HLINHE	PHO
0	35.842429	36.062851	35.605571	39.531377	39.483376	35.004876	36.137033	220.373257	110.5
1	0.237916	0.239330	0.236396	0.344615	0.292810	0.217468	0.229951	-0.012210	0.0
2	-0.016027	-0.016047	-0.016005	-0.013505	-0.015419	-0.016335	-0.015075	-0.012210	-0.0
3	-0.016027	-0.016047	-0.016005	-0.013505	-0.015419	-0.016335	-0.015075	-0.012210	-0.0
4	-0.021134	-0.021207	-0.021056	-0.019669	-0.019963	-0.022177	-0.022314	-0.012210	-0.0

5 rows × 21 columns

Parte 7: Selección y extracción - Métodos de filtrado

Se utilizaron métodos de filtrado para la selección de características y técnicas de extracción de características. Esto nos permitirá reducir los requerimientos de almacenamiento, la complejidad del modelo y el tiempo de entrenamiento.

Correlación

En términos de correlación en nuestras variables, lógicamente encontramos valores altos, sobre todo hablando de una correlación positiva, tal como se muestra en la siguiente imagen. Cabe destacar que consideramos el comportamiento observado como racional puesto que, básicamente el set de datos contiene variables relativas a la población, destacando un totalizador y particiones de la misma.

Recordemos que aquellas variables que tienen una correlación perfecta son las que tienen un 1; lo que ocurrirá siempre que se analiza una variable consigo misma, siendo lo que observamos en la diagonal de la imagen. Sin embargo, se puede apreciar que hay, por ejemplo, una alta correlación cuando hablamos de personas de 60 años o más con alguna discapacidad.

El mismo comportamiento lo podemos observar si calculamos e imprimimos en forma de texto el cálculo de la correlación entre las variables, que es lo que observamos a continuación.

Lo anterior nos puede permitir deducir, por ejemplo, en qué proporción aumenta cierta condición de la población contra el total, es decir, generar estadísticas de incidencias o relativos al nivel socioeconómico contra un total para cada región.

Por otro lado, para ahondar un poco más en la correlación, sabemos que un comportamiento lineal es necesario, por lo cual decidimos generar un diagrama de dispersión que nos permitiese entender la distribución de los datos contra una línea de referencia, que es lo que se observa en la imagen que se presenta a continuación.

Como se observa, los datos que contiene nuestro set, -o la concatenación de nuestros conjuntos originales-, presenta un comportamiento, en su mayoría, lineal. En algunos casos se destacan algunos datos atípicos.

Procedimiento

In [ ]:
```
print(df2.corr())
```

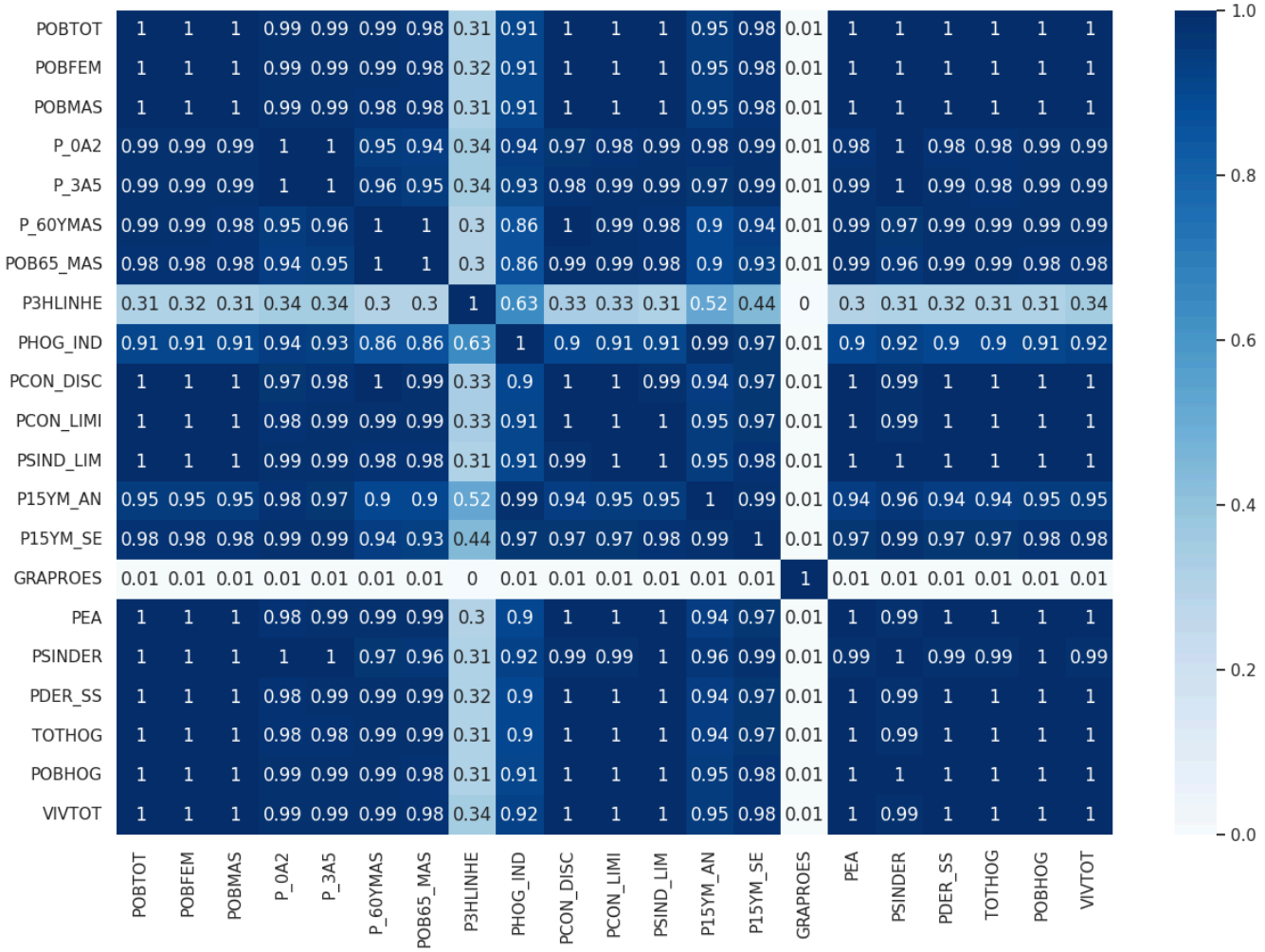
```
              POBTOT    POBFEM    POBMAS     P_0A2     P_3A5  P_60YMAS  \
POBTOT     1.000000  0.999980  0.999977  0.988880  0.991567  0.985677
POBFEM     0.999980  1.000000  0.999914  0.988007  0.990801  0.986654
POBMAS     0.999977  0.999914  1.000000  0.989768  0.992340  0.984590
P_0A2      0.988880  0.988007  0.989768  1.000000  0.999762  0.950209
P_3A5      0.991567  0.990801  0.992340  0.999762  1.000000  0.956023
P_60YMAS   0.985677  0.986654  0.984590  0.950209  0.956023  1.000000
POB65_MAS  0.981622  0.982737  0.980388  0.943158  0.949355  0.999710
P3HLINHE   0.314569  0.315068  0.314022  0.336454  0.335232  0.298974
PHOG_IND   0.908959  0.908071  0.909866  0.935688  0.933555  0.861122
PCON_DISC  0.995731  0.996195  0.995191  0.972942  0.977191  0.995252
PCON_LIMI  0.998698  0.998913  0.998424  0.982272  0.985751  0.990796
PSIND_LIM  0.999914  0.999827  0.999963  0.990384  0.992846  0.983713
P15YM_AN   0.952003  0.950954  0.953082  0.976324  0.974511  0.903514
P15YM_SE   0.978583  0.977759  0.979419  0.991928  0.991533  0.939928
GRAPROES   0.013019  0.013121  0.012909  0.011100  0.011432  0.014743
PEA        0.999072  0.999297  0.998788  0.981726  0.985171  0.991620
```

```
PSINDER    0.996131  0.995596  0.996657  0.996873  0.998108  0.968215
PDER_SS    0.999116  0.999343  0.998829  0.982341  0.985716  0.991345
TOTHOG     0.998757  0.999031  0.998420  0.980804  0.984347  0.992253
POBHOG     0.999998  0.999973  0.999981  0.989055  0.991722  0.985483
VIVTOT     0.998988  0.999112  0.998811  0.986299  0.989190  0.987020

           POB65_MAS  P3HLINHE  PHOG_IND  PCON_DISC  ...  PSIND_LIM  P15YM_AN  \
POBTOT      0.981622  0.314569  0.908959   0.995731  ...   0.999914  0.952003
POBFEM      0.982737  0.315068  0.908071   0.996195  ...   0.999827  0.950954
POBMAS      0.980388  0.314022  0.909866   0.995191  ...   0.999963  0.953082
P_0A2       0.943158  0.336454  0.935688   0.972942  ...   0.990384  0.976324
P_3A5       0.949355  0.335232  0.933555   0.977191  ...   0.992846  0.974511
P_60YMAS    0.999710  0.298974  0.861122   0.995252  ...   0.983713  0.903514
POB65_MAS   1.000000  0.301520  0.855037   0.993027  ...   0.979410  0.896552
P3HLINHE    0.301520  1.000000  0.625664   0.329035  ...   0.311363  0.516372
PHOG_IND    0.855037  0.625664  1.000000   0.895834  ...   0.909804  0.986591
PCON_DISC   0.993027  0.329035  0.895834   1.000000  ...   0.994459  0.935758
PCON_LIMI   0.987612  0.331994  0.907087   0.998799  ...   0.997959  0.947297
PSIND_LIM   0.979410  0.311363  0.909804   0.994459  ...   1.000000  0.953387
P15YM_AN    0.896552  0.516372  0.986591   0.935758  ...   0.953387  1.000000
P15YM_SE    0.933556  0.441492  0.967731   0.965983  ...   0.979557  0.993222
GRAPROES    0.014927  0.001304  0.008049   0.013424  ...   0.012927  0.008733
PEA         0.988404  0.304191  0.896975   0.997926  ...   0.998568  0.940665
PSINDER     0.962277  0.309527  0.918882   0.985597  ...   0.996901  0.963075
PDER_SS     0.988203  0.316301  0.901789   0.997884  ...   0.998618  0.944126
TOTHOG      0.989240  0.311968  0.898369   0.997990  ...   0.998182  0.941041
POBHOG      0.981404  0.314783  0.909241   0.995631  ...   0.999925  0.952268
VIVTOT      0.983500  0.341941  0.915697   0.996090  ...   0.998657  0.954627

           P15YM_SE  GRAPROES       PEA   PSINDER   PDER_SS    TOTHOG  \
POBTOT     0.978583  0.013019  0.999072  0.996131  0.999116  0.998757
POBFEM     0.977759  0.013121  0.999297  0.995596  0.999343  0.999031
POBMAS     0.979419  0.012909  0.998788  0.996657  0.998829  0.998420
P_0A2      0.991928  0.011100  0.981726  0.996873  0.982341  0.980804
P_3A5      0.991533  0.011432  0.985171  0.998108  0.985716  0.984347
P_60YMAS   0.939928  0.014743  0.991620  0.968215  0.991345  0.992253
POB65_MAS  0.933556  0.014927  0.988404  0.962277  0.988203  0.989240
P3HLINHE   0.441492  0.001304  0.304191  0.309527  0.316301  0.311968
PHOG_IND   0.967731  0.008049  0.896975  0.918882  0.901789  0.898369
PCON_DISC  0.965983  0.013424  0.997926  0.985597  0.997884  0.997990
PCON_LIMI  0.974625  0.013375  0.999124  0.991961  0.999205  0.999117
PSIND_LIM  0.979557  0.012927  0.998568  0.996901  0.998618  0.998182
P15YM_AN   0.993222  0.008733  0.940665  0.963075  0.944126  0.941041
P15YM_SE   1.000000  0.009437  0.970336  0.985944  0.972409  0.970055
GRAPROES   0.009437  1.000000  0.013486  0.011985  0.013478  0.013791
PEA        0.970336  0.013486  1.000000  0.991840  0.999801  0.999853
PSINDER    0.985944  0.011985  0.991840  1.000000  0.991558  0.990850
PDER_SS    0.972409  0.013478  0.999801  0.991558  1.000000  0.999807
TOTHOG     0.970055  0.013791  0.999853  0.990850  0.999807  1.000000
POBHOG     0.978736  0.013024  0.999021  0.996224  0.999069  0.998705
VIVTOT     0.978737  0.013598  0.998544  0.993199  0.999023  0.998896

             POBHOG    VIVTOT
POBTOT     0.999998  0.998988
POBFEM     0.999973  0.999112
POBMAS     0.999981  0.998811
P_0A2      0.989055  0.986299
P_3A5      0.991722  0.989190
P_60YMAS   0.985483  0.987020
POB65_MAS  0.981404  0.983500
P3HLINHE   0.314783  0.341941
PHOG_IND   0.909241  0.915697
PCON_DISC  0.995631  0.996090
PCON_LIMI  0.998658  0.998771
PSIND_LIM  0.999925  0.998657
```

```
P15YM_AN    0.952268  0.954627
P15YM_SE    0.978736  0.978737
GRAPROES    0.013024  0.013598
PEA         0.999021  0.998544
PSINDER     0.996224  0.993199
PDER_SS     0.999069  0.999023
TOTHOG      0.998705  0.998896
POBHOG      1.000000  0.998978
VIVTOT      0.998978  1.000000

[21 rows x 21 columns]
```

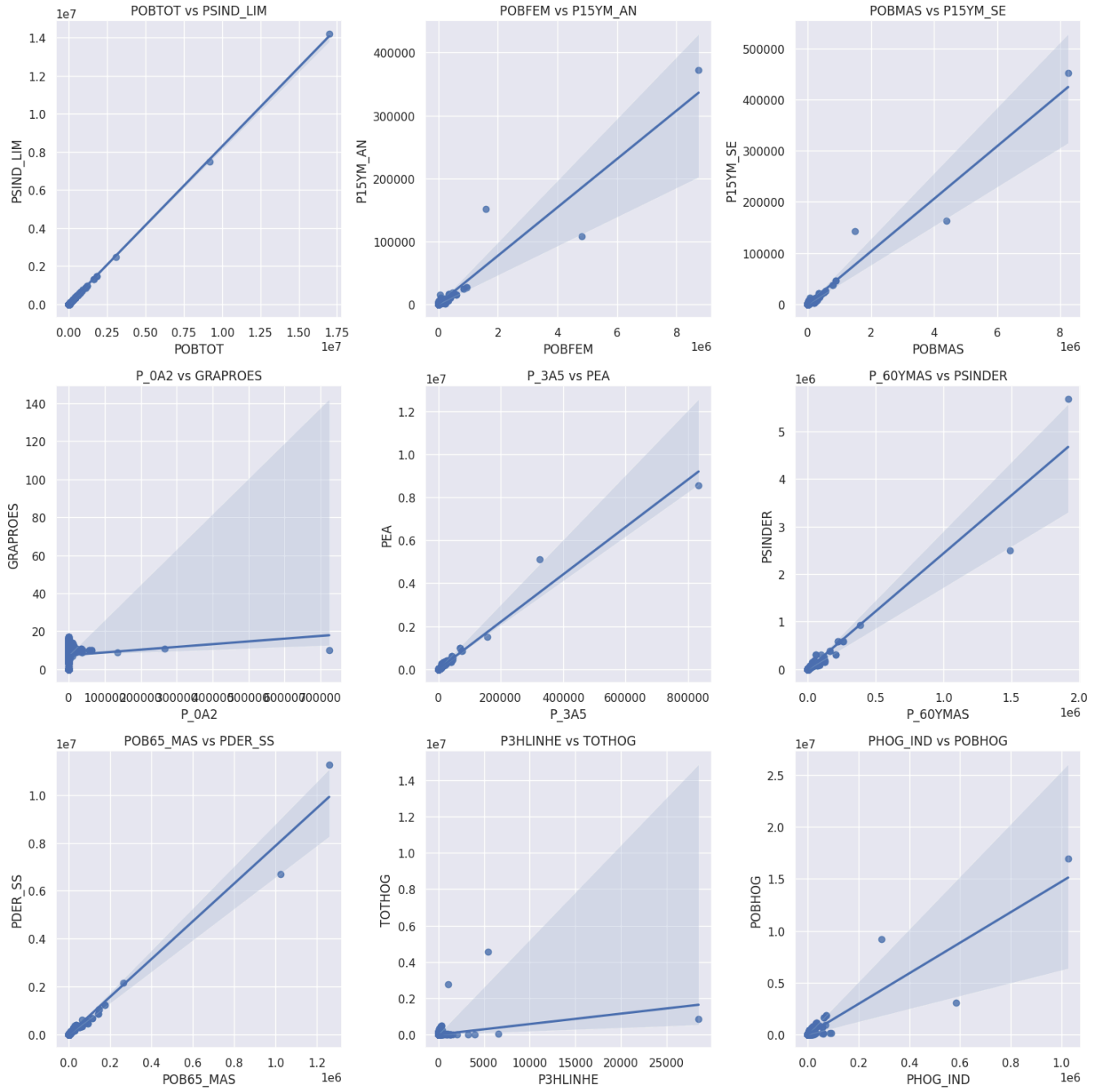
In [ ]:

```
sns.set(rc={'figure.figsize':(15,10)})
df2= df2.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
sns.heatmap(df2.corr().round(2), annot=True, cmap="Blues")
plt.show()
```



In [ ]:

```
x_subset_cols = df2.iloc[:, :10]
y_subset_cols = df2.iloc[:, 11:20]

fig, axes = plt.subplots(3, 3, figsize=(15, 15))
axes = axes.flatten()

for i, (x_col, y_col) in enumerate(zip(x_subset_cols, y_subset_cols)):
    sns.regplot(data=df2, x=x_col, y=y_col, ax=axes[i])

    axes[i].set_title(f'{x_col} vs {y_col}')

plt.tight_layout()
plt.show()
```



PCA y análisis de componentes

Como parte de esta y la anterior entrega, se realizó un análisis PCA para poder entender mejor cómo se comportan las variables entre sí y, de esta manera, conocer los escenarios a los cuales nos podremos enfrentar más adelante cuando trabajemos con clústeres más avanzados. Así pues, en la siguiente imagen podemos apreciar el resumen del análisis PCA, así como también la proporción acumulada de cada variable en la explicación del fenómeno.

Cabe destacar que para generar una mejor comprensión sobre el resultado anteriormente expresado, se grafican los componentes principales. En la imagen que se presenta a continuación es posible observar que a través de los primeros tres componentes se tendría básicamente la totalidad de la varianza explicada.

Realizar un análisis PCA en este caso particular nos resultó relevante porque se estaban contemplando diversas variables relativas a la población y era necesario destacar aquellas que realmente generan variabilidad o resultan componentes principales para entender en su generalidad al set de datos.

Procedimiento

In [ ]:
```
pca = PCA()
pca.fit(df_transformed)
pcaSummary = pd.DataFrame({'Standard deviation': np.round(np.sqrt(pca.explained_variance_),
                           'Proportion of variance': np.round(pca.explained_variance_ratio_
                           'Cumulative proportion': np.round(np.cumsum(pca.explained_varian
                           })

pcaSummary = pcaSummary.transpose()
pcaSummary.columns = ['PC{}'.format(i) for i in range(1, len(pcaSummary.columns) + 1)]
pcaSummary
```

Out[ ]:

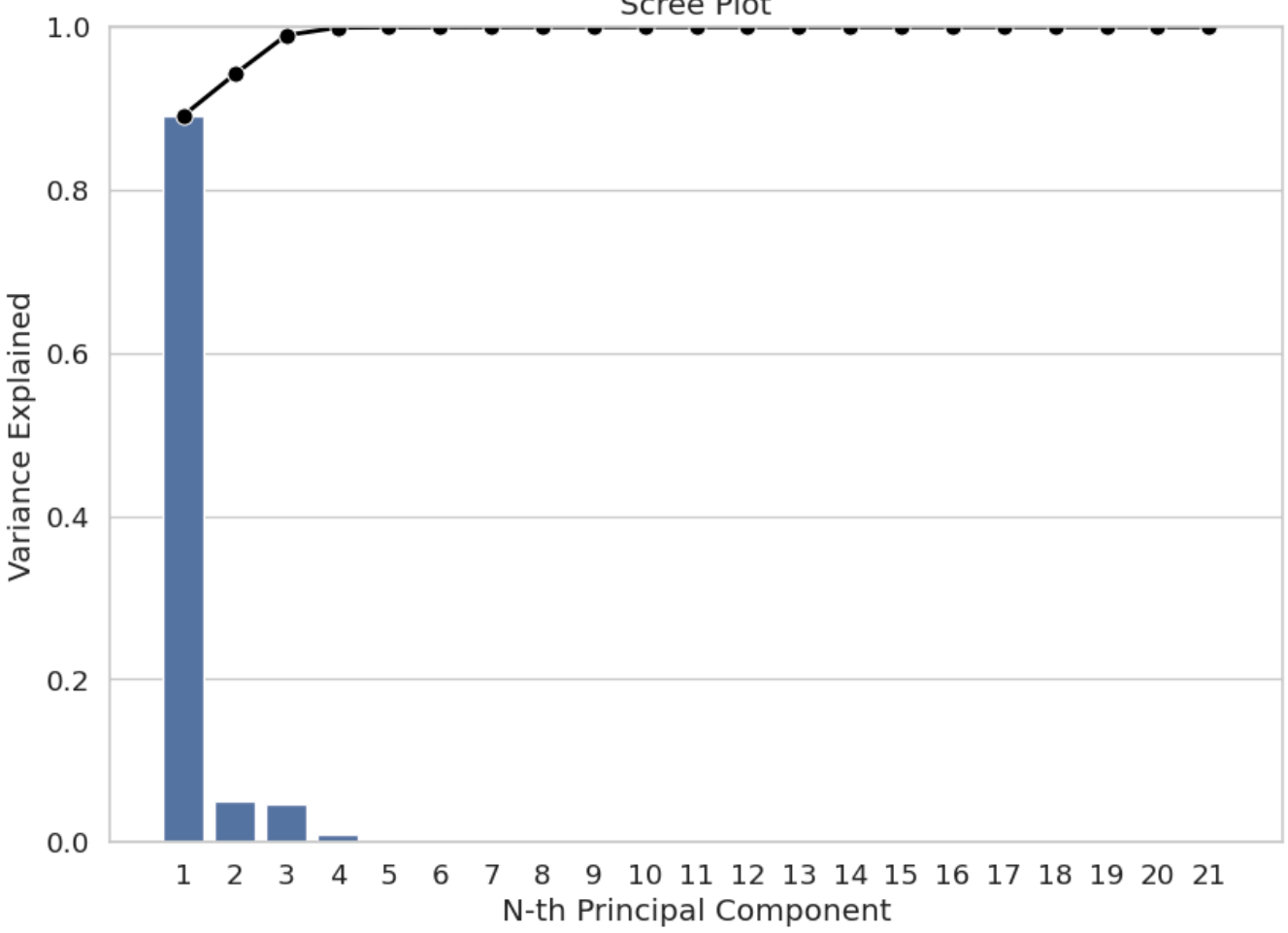
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC12	PC13
Standard deviation	4.33	1.03	1.00	0.44	0.09	0.06	0.05	0.03	0.03	0.02	...	0.01	0.01
Proportion of variance	89.18	5.05	4.76	0.93	0.04	0.02	0.01	0.00	0.00	0.00	...	0.00	0.00
Cumulative proportion	89.18	94.23	98.99	99.92	99.96	99.98	99.99	99.99	100.00	100.00	...	100.00	100.00

3 rows × 21 columns

Graficamos los resultados

In [ ]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [ ]:
```
PC_components = np.arange(pca.n_components_) + 1

scree = sns.set(style = 'whitegrid', font_scale = 1.2)

fig, ax = plt.subplots(figsize=(10, 7))

scree = sns.barplot(x = PC_components, y = pca.explained_variance_ratio_, color = 'b')

scree = sns.lineplot(x = PC_components-1,
                     y = np.cumsum(pca.explained_variance_ratio_),
                     color = 'black',
                     linestyle = '-',
                     linewidth = 2,
                     marker = 'o',
                     markersize = 8
                    )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()
```



In [ ]:
```
pca.components_
```

Out[ ]:
```
array([[ 2.30627720e-01,  2.30612540e-01,  2.30633703e-01,
         2.28899965e-01,  2.29423942e-01,  2.26671568e-01,
         2.25733595e-01,  8.57139692e-02,  2.15381413e-01,
         2.29648961e-01,  2.30480811e-01,  2.30593095e-01,
         2.23455314e-01,  2.28167390e-01,  3.06060280e-03,
         2.30138818e-01,  2.29937585e-01,  2.30329792e-01,
         2.30160557e-01,  2.30633620e-01,  2.30752283e-01],
       [-5.95478589e-02, -5.98781117e-02, -5.91927225e-02,
        -1.26291805e-02, -1.76740388e-02, -9.33671351e-02,
        -9.22895296e-02,  8.86069192e-01,  3.17396637e-01,
        -5.50445056e-02, -4.72262915e-02, -6.11122418e-02,
         1.90532482e-01,  9.70438370e-02, -9.71074324e-02,
        -7.53474647e-02, -5.16924684e-02, -6.29711224e-02,
        -6.83998629e-02, -5.91803779e-02, -3.40369674e-02],
       [-6.35535457e-03, -6.26734047e-03, -6.44901545e-03,
        -3.74839331e-03, -3.91113226e-03, -7.40556293e-03,
        -6.98762646e-03,  8.86318330e-02,  2.81601839e-02,
        -5.17164193e-03, -4.58490086e-03, -6.64353665e-03,
         1.51634732e-02,  6.04097609e-03,  9.95244463e-01,
        -7.37723603e-03, -6.78871964e-03, -6.11269123e-03,
        -6.32907191e-03, -6.31544856e-03, -3.07221700e-03],
       [ 1.18645385e-02,  2.53274808e-02, -2.50575564e-03,
        -3.03853722e-01, -2.61744565e-01,  3.78615623e-01,
         4.28100871e-01,  3.25155973e-01, -3.03881380e-01,
         2.01308603e-01,  1.06495499e-01, -1.47132222e-02,
        -3.57117525e-01, -2.60523174e-01, -6.87823402e-03,
         9.86249994e-02, -1.71756616e-01,  9.99134018e-02,
         1.15706399e-01,  9.46660432e-03,  5.88280615e-02],
       [-6.38510355e-02, -5.09744820e-02, -7.75920883e-02,
        -2.06555168e-01, -2.17201628e-01,  1.75528163e-01,
         2.28690387e-01, -2.84724440e-01,  8.15939581e-01,
```

```
  2.78193680e-02, -4.83572932e-02, -7.24036407e-02,
  2.78037737e-03, -7.26226623e-02,  3.32589414e-04,
  8.12301533e-03, -2.21707893e-01,  1.27786158e-02,
  1.02319546e-03, -6.61221114e-02, -1.15898288e-02],
[-7.44252173e-02, -8.56549889e-02, -6.24357418e-02,
 -1.51733329e-01, -1.32817645e-01,  1.60662489e-01,
  1.86040187e-01, -7.85794587e-02, -1.57577940e-01,
  3.92386039e-01,  1.27180074e-01, -1.27664241e-01,
  2.50187378e-01,  5.99377579e-01,  1.28843343e-03,
 -1.23790588e-01,  7.47155735e-02, -1.40974128e-01,
 -2.42369581e-01, -8.26213802e-02, -3.69452697e-01],
[-2.81649610e-02, -1.71031337e-02, -3.99706341e-02,
  2.91494513e-02, -2.47721991e-02,  1.46861620e-01,
  2.52931851e-01, -4.86420163e-02, -2.47995257e-01,
 -3.76657624e-01, -3.10180328e-01,  3.10237549e-02,
  5.33803606e-01,  1.45232294e-01,  1.23390224e-04,
 -2.49759731e-02, -4.72053303e-01,  1.75966895e-01,
  8.57358216e-02, -3.05820501e-02,  1.96596806e-01],
[-6.12976542e-02, -4.65740319e-02, -7.70103023e-02,
 -1.58045194e-01, -1.52745056e-01, -2.39194474e-01,
 -2.76529994e-01, -6.13586654e-02, -6.87667928e-02,
  4.10898839e-01,  4.21821331e-01, -1.55528359e-01,
  3.71145349e-01, -1.97322337e-01, -8.19611987e-05,
  3.35963848e-03, -2.20960506e-01,  1.84347427e-02,
  6.47888820e-02, -5.89777676e-02,  4.39501026e-01],
[ 1.00882577e-01,  8.20359984e-02,  1.20994139e-01,
 -2.12573985e-01, -2.38395710e-01, -2.33172854e-01,
 -3.29611386e-01,  7.77533052e-03,  1.92306547e-02,
  2.36552308e-02, -1.14450952e-01,  1.35051320e-01,
 -3.55469802e-01,  5.31023271e-01,  3.38183519e-04,
  2.22849622e-01, -3.15259088e-01,  3.03741137e-01,
  1.17120421e-01,  7.34440809e-02,  4.82418483e-02],
[-1.01925072e-01, -1.04488204e-01, -9.91847774e-02,
 -2.46315420e-01, -1.51362801e-01,  3.79659514e-02,
  7.50015848e-02, -1.71046037e-02, -1.21198411e-02,
 -3.00245198e-01,  8.73001658e-03, -1.10658726e-01,
 -7.72161179e-02,  3.07418490e-01, -7.67947851e-05,
  2.20731281e-02,  4.36859226e-01, -3.62791011e-01,
  3.14921844e-01, -1.19045426e-01,  4.88397858e-01],
[ 1.13756546e-01,  1.44030391e-01,  8.14384994e-02,
 -3.76417102e-01, -3.60876528e-01, -1.04437654e-01,
 -9.30948527e-02, -3.92422925e-03, -4.31588793e-02,
 -2.81589252e-02, -2.48377862e-01,  1.72769698e-01,
  3.98266311e-01, -2.53819343e-01,  6.61000094e-05,
  3.67208395e-01,  3.52828451e-01,  8.08917876e-03,
  2.96103889e-02,  1.18685896e-01, -2.76723963e-01],
[ 1.03950056e-01,  8.27581891e-02,  1.26564741e-01,
 -4.05840190e-01,  1.20356312e-01,  2.42650517e-02,
 -7.65066574e-03, -1.10172901e-02,  1.23475811e-02,
 -5.22726463e-01,  6.10672896e-01,  6.85248479e-02,
  3.97109955e-02,  2.17884155e-02, -6.32285695e-05,
 -1.56358497e-01, -1.59636944e-02,  1.46829266e-01,
 -2.51018528e-01,  1.11374333e-01, -1.07823327e-01],
[-9.95763423e-02, -1.01842940e-01, -9.71526604e-02,
  4.20100733e-01, -2.75036554e-01,  1.69595364e-01,
 -1.37302709e-01,  3.39280731e-03, -1.70095007e-02,
 -2.46941237e-01,  3.93676790e-01, -1.71046317e-01,
  3.07997278e-02,  2.14907903e-02,  6.40572936e-05,
  4.57185445e-01, -1.23422993e-01, -9.91188941e-02,
  2.81954904e-01, -8.47684922e-02, -3.20685890e-01],
[-6.70275162e-02, -9.33978484e-02, -3.88779822e-02,
 -3.66710650e-01,  5.48957382e-01, -9.43093710e-02,
 -2.94519420e-03, -5.74259920e-03,  1.75666773e-03,
  8.92099253e-02, -1.75964007e-02, -1.03479052e-01,
  4.95510901e-02, -1.40395943e-02,  2.76804649e-05,
```

```
        1.46374278e-02, -1.33893688e-01, -6.07632416e-02,
        6.23443704e-01, -1.92457653e-03, -3.29172708e-01],
       [-7.28600442e-02, -5.90637534e-02, -8.75823859e-02,
        -1.76373689e-01,  4.27626973e-01,  1.47768411e-01,
        -5.68519741e-02,  4.53473238e-03, -8.03510941e-03,
        1.35673941e-02, -8.21855552e-02, -1.18794980e-01,
        -3.40432630e-02,  2.59996463e-02,  2.72034883e-05,
        6.69817876e-01, -8.70603337e-02, -1.38702113e-01,
        -4.56656094e-01, -8.25776114e-02,  1.75195895e-01],
       [-1.55087692e-02,  6.56253160e-02, -1.02107132e-01,
        -9.24460469e-02,  2.44510840e-02,  7.19150887e-01,
        -6.12658506e-01, -4.84896979e-03,  2.85776696e-03,
        3.20393034e-02, -1.19761212e-01, -2.11921584e-03,
        4.49240192e-02, -1.79898485e-02, -1.78869260e-05,
        -2.13407156e-01,  4.07139579e-02,  5.35594298e-02,
        5.56623240e-02,  1.06831646e-01,  3.20464277e-02],
       [ 8.08366032e-02, -5.55382677e-01,  7.59906251e-01,
        -1.37932285e-02, -2.46765972e-02,  1.21533263e-01,
        -8.29990149e-02, -5.26352748e-04,  3.99088079e-03,
        2.63608153e-02, -2.57302466e-02,  1.10623579e-01,
        3.85418826e-02, -5.23902279e-02,  1.73427843e-05,
        -7.67946150e-03, -5.16991545e-02, -1.28343593e-01,
        4.00068043e-04, -2.18324152e-01,  1.97300643e-02],
       [ 3.38121651e-01,  5.58167242e-01,  1.03239429e-01,
        -3.72704254e-03,  8.72884324e-03,  3.91909688e-02,
        -4.53915586e-02, -2.26113767e-04,  3.60995995e-04,
        1.58965271e-02,  2.87684178e-02,  1.87722041e-01,
        1.43194641e-03, -9.40432313e-04, -4.21006734e-06,
        -4.43593295e-02, -2.43496265e-01, -5.12508477e-01,
        3.63507833e-02, -4.44540405e-01, -2.43847475e-02],
       [ 1.29966156e-01,  6.28472258e-02,  2.01600265e-01,
        2.25921832e-02, -6.43891708e-02, -2.84460095e-02,
        2.16229817e-02, -3.44826711e-04,  1.19741441e-03,
        -1.15182131e-02, -5.69747819e-02, -3.16396580e-01,
        -9.27426555e-03,  1.43032422e-02,  3.33569695e-06,
        -2.19003405e-02, -2.18029608e-01, -4.73804641e-01,
        -2.60894487e-02,  7.44940526e-01,  2.60978650e-02],
       [-2.45915137e-01, -2.37542753e-01, -2.54840585e-01,
        -1.80366392e-03, -3.34198704e-03, -4.94988851e-04,
        1.73445687e-04,  9.18982140e-05,  3.42188128e-04,
        4.75199809e-02,  1.08237946e-01,  7.93375602e-01,
        -6.07563570e-03,  7.59217623e-03, -2.25351444e-06,
        1.32473753e-02, -1.47106906e-01, -3.11239218e-01,
        -5.59634448e-03,  2.36364889e-01,  6.49402056e-03],
       [ 8.16346328e-01, -4.21481008e-01, -3.94882809e-01,
        -9.70386168e-12,  1.22179008e-10,  2.12382945e-10,
        -2.04933097e-10, -5.80183301e-13, -4.14073682e-12,
        -3.35654477e-10, -7.07587557e-10, -5.44857585e-09,
        7.13121030e-11, -8.94871865e-11,  1.39806672e-15,
        -1.97745290e-10,  8.79386903e-10,  1.88451795e-09,
        1.72237668e-10, -4.35357059e-09, -1.59531437e-10]])
```

Cálculo de la importancia de los componentes

In [ ]:

```
pcsComponents_df = pd.DataFrame(pca.components_.transpose(),
                                columns=pcaSummary.columns,
                                index=df_transformed.columns
                                )
pcsComponents_df
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	
POBTOT	0.230628	-0.059548	-0.006355	0.011865	-0.063851	-0.074425	-0.028165	-0.061298	0

POBFEM	0.230613	-0.059878	-0.006267	0.025327	-0.050974	-0.085655	-0.017103	-0.046574	0
POBMAS	0.230634	-0.059193	-0.006449	-0.002506	-0.077592	-0.062436	-0.039971	-0.077010	0
P_0A2	0.228900	-0.012629	-0.003748	-0.303854	-0.206555	-0.151733	0.029149	-0.158045	-0
P_3A5	0.229424	-0.017674	-0.003911	-0.261745	-0.217202	-0.132818	-0.024772	-0.152745	-0
P_60YMAS	0.226672	-0.093367	-0.007406	0.378616	0.175528	0.160662	0.146862	-0.239194	-0
POB65_MAS	0.225734	-0.092290	-0.006988	0.428101	0.228690	0.186040	0.252932	-0.276530	-0
P3HLINHE	0.085714	0.886069	0.088632	0.325156	-0.284724	-0.078579	-0.048642	-0.061359	0
PHOG_IND	0.215381	0.317397	0.028160	-0.303881	0.815940	-0.157578	-0.247995	-0.068767	0
PCON_DISC	0.229649	-0.055045	-0.005172	0.201309	0.027819	0.392386	-0.376658	0.410899	0
PCON_LIMI	0.230481	-0.047226	-0.004585	0.106495	-0.048357	0.127180	-0.310180	0.421821	-0
PSIND_LIM	0.230593	-0.061112	-0.006644	-0.014713	-0.072404	-0.127664	0.031024	-0.155528	0
P15YM_AN	0.223455	0.190532	0.015163	-0.357118	0.002780	0.250187	0.533804	0.371145	-0
P15YM_SE	0.228167	0.097044	0.006041	-0.260523	-0.072623	0.599378	0.145232	-0.197322	0
GRAPROES	0.003061	-0.097107	0.995244	-0.006878	0.000333	0.001288	0.000123	-0.000082	0
PEA	0.230139	-0.075347	-0.007377	0.098625	0.008123	-0.123791	-0.024976	0.003360	0
PSINDER	0.229938	-0.051692	-0.006789	-0.171757	-0.221708	0.074716	-0.472053	-0.220961	-0
PDER_SS	0.230330	-0.062971	-0.006113	0.099913	0.012779	-0.140974	0.175967	0.018435	0
TOTHOG	0.230161	-0.068400	-0.006329	0.115706	0.001023	-0.242370	0.085736	0.064789	0
POBHOG	0.230634	-0.059180	-0.006315	0.009467	-0.066122	-0.082621	-0.030582	-0.058978	0
VIVTOT	0.230752	-0.034037	-0.003072	0.058828	-0.011590	-0.369453	0.196597	0.439501	0

21 rows × 21 columns

In [ ]:
```
print("=============PC1================")
```

```
print("** Más importante:")
print(pcsComponents_df.PC1.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC1.nsmallest(3))
print("=============PC2================")
print("** Más importante:")
print(pcsComponents_df.PC2.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC2.nsmallest(3))
print("=============PC3================")
print("** Más importante:")
print(pcsComponents_df.PC3.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC3.nsmallest(3))
print("=============PC4================")
print("** Más importante:")
print(pcsComponents_df.PC4.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC4.nsmallest(3))
print("=============PC5================")
print("** Más importante:")
print(pcsComponents_df.PC5.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC5.nsmallest(3))
print("=============PC6================")
print("** Más importante:")
print(pcsComponents_df.PC6.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC6.nsmallest(3))
print("=============PC7================")
print("** Más importante:")
print(pcsComponents_df.PC7.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC7.nsmallest(3))
print("=============PC8================")
print("** Más importante:")
print(pcsComponents_df.PC8.nlargest(3))
print("** Menos importante:")
print(pcsComponents_df.PC8.nsmallest(3))
```

```
=============PC1================
** Más importante:
VIVTOT    0.230752
POBMAS    0.230634
POBHOG    0.230634
Name: PC1, dtype: float64
** Menos importante:
GRAPROES    0.003061
P3HLINHE    0.085714
PHOG_IND    0.215381
Name: PC1, dtype: float64
=============PC2================
** Más importante:
P3HLINHE    0.886069
PHOG_IND    0.317397
P15YM_AN    0.190532
Name: PC2, dtype: float64
** Menos importante:
GRAPROES    -0.097107
P_60YMAS    -0.093367
POB65_MAS   -0.092290
Name: PC2, dtype: float64
=============PC3================
```

```
** Más importante:
GRAPROES    0.995244
P3HLINHE    0.088632
PHOG_IND    0.028160
Name: PC3, dtype: float64
** Menos importante:
P_60YMAS    -0.007406
PEA         -0.007377
POB65_MAS   -0.006988
Name: PC3, dtype: float64
=============PC4=================
** Más importante:
POB65_MAS    0.428101
P_60YMAS     0.378616
P3HLINHE     0.325156
Name: PC4, dtype: float64
** Menos importante:
P15YM_AN   -0.357118
PHOG_IND   -0.303881
P_0A2      -0.303854
Name: PC4, dtype: float64
=============PC5=================
** Más importante:
PHOG_IND     0.815940
POB65_MAS    0.228690
P_60YMAS     0.175528
Name: PC5, dtype: float64
** Menos importante:
P3HLINHE   -0.284724
PSINDER    -0.221708
P_3A5      -0.217202
Name: PC5, dtype: float64
=============PC6=================
** Más importante:
P15YM_SE     0.599378
PCON_DISC    0.392386
P15YM_AN     0.250187
Name: PC6, dtype: float64
** Menos importante:
VIVTOT     -0.369453
TOTHOG     -0.242370
PHOG_IND   -0.157578
Name: PC6, dtype: float64
=============PC7=================
** Más importante:
P15YM_AN     0.533804
POB65_MAS    0.252932
VIVTOT       0.196597
Name: PC7, dtype: float64
** Menos importante:
PSINDER     -0.472053
PCON_DISC   -0.376658
PCON_LIMI   -0.310180
Name: PC7, dtype: float64
=============PC8=================
** Más importante:
VIVTOT       0.439501
PCON_LIMI    0.421821
PCON_DISC    0.410899
Name: PC8, dtype: float64
** Menos importante:
POB65_MAS   -0.276530
P_60YMAS    -0.239194
PSINDER     -0.220961
Name: PC8, dtype: float64
```

Conclusiones al 6 de octubre 2024

En la entrega anterior del proyecto integrador se inició con el análisis exploratorio de los datos, para poder comprender mejor los registros y cómo se comportan, dado el contexto de la pregunta que deseamos resolver. En nuestro caso, nos enfocamos en las islas de calor con enfoque en la justicia social, para lo cual hemos dispuesto de diferentes orígenes de datos.

Como primera instancia, en términos poblacionales es donde ha sido más exigente el procesamiento de datos, pues era inicialmente necesario unificar todas las entidades, además de limpiar y tratar valores fuera de lo establecido, como lo fueron los asteriscos y valores faltantes. Con este conjunto de datos, la pregunta de negocio que deseábamos responder es si dicho sector poblacional es o no vulnerable; además, el grado de vulnerabilidad dadas sus propias condiciones, para definir un nivel de criticidad. Por tanto, para este punto habíamos ya logrado satisfacer las primeras dos etapas que define la metodología CRISP-ML.

Consecuentemente, continuamos preparando de manera más detallada nuestro conjunto de datos definiendo aquellas variables que nos podrían ayudar a determinar cuando la población se encontrarse en condiciones vulnerables o, inherentemente a su persona podría representar un factor de vulnerabilidad en condiciones desfavorables. Así pues, fue necesario implementar ingeniería de características, hacer escalamiento o normalización y transformaciones, como también explorar la codificación, y emplear diversos métodos de filtrado, tales como obtención de la correlación, análisis de componentes principales o PCA y análisis factorial o FA.

Por tanto, en esta segunda entrega hemos sido capaces de cubrir las primeras tres fases que define la metodología, creando así los cimientos que nos ayudarán a modelar conforme lo establecido, que es segmentar la vulnerabilidad contra un grado de criticidad, lo que posteriormente contrastaremos contra las islas de calor que también se han ido procesando a nivel de AGEB.

Fuentes consultadas al 6 de octubre 2024

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

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

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

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

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

Avance 3 (Entrega: 13 octubre 2024)

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.

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.

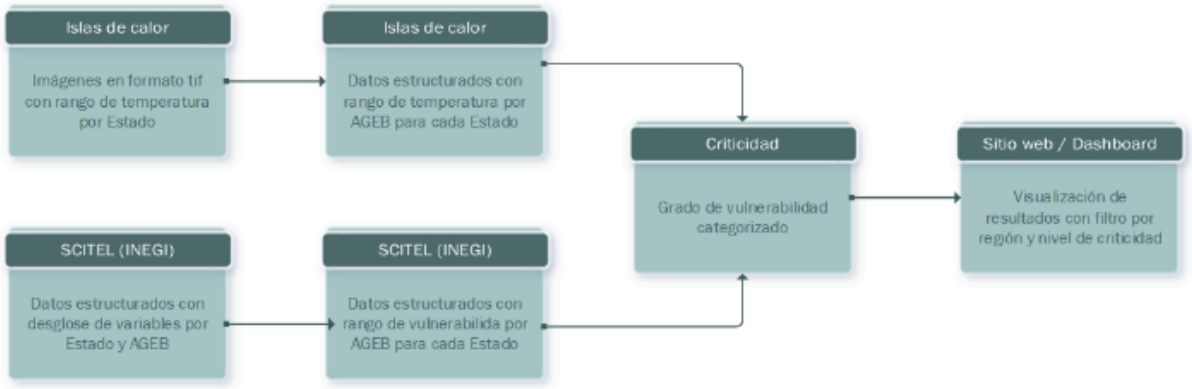


Imagen 2. Diagrama de entradas y salidas. *Elaboración propia*.

Parte 6: Algoritmo

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

In [ ]:
```
# Leemos los datos por Estado
# Hidalgo
df_hidalgo = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                 usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL'
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# CDMX 1
df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/hea
```

```
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC','P_0A2', 'P_0A2_F', 'P_0A
# CDMX 2
df_cdmx2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heac
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 1
df_edomex1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 2
df_edomex2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 3
df_edomex3 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
```

```
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'P
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_0
# EdoMex 4
df_edomex4 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'P
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'P
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_0
```

```
<ipython-input-121-90ee3e741790>:18: DtypeWarning: Columns (178) have mixed types. Specify dt
ype option on import or set low_memory=False.
  df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/head
s/main/CDMX.csv",
```

In [ ]:
```
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()
```

Out[ ]:

	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	...

5 rows × 103 columns

Parte 7. 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.

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.

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.

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.

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.

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.

In [ ]:
```
df3e = [df_hidalgo,df_cdmx1,df_cdmx2,df_edomex1,df_edomex2,df_edomex3,df_edomex4]
df3e = pd.concat(df3e, ignore_index=True)
df3e.head()
```

Out[ ]:

	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	...

5 rows × 103 columns

Limpieza

In [ ]:
```
df_cat1 = df3e.copy()

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

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VP
4	13.0	Hidalgo	1.0	Acatlán	0043	1.0	92.0	54	38	6	...	
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	...	
7	13.0	Hidalgo	1.0	Acatlán	0043	6.0	47.0	20	27	*	...	
8	13.0	Hidalgo	1.0	Acatlán	0043	7.0	44.0	21	23	*	...	

5 rows × 103 columns

In [ ]:
```
df_cat2 = df_cat1.copy()

df_cat2.replace('*', np.nan, inplace=True)
df_cat2.head()
df_cat2.dropna(how='all', inplace = True)

# Hacemos el cambio para que todas las variables sean numéricas
df_cat2= df_cat2.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
df_cat2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 243657 entries, 4 to 253243
Columns: 103 entries, ENTIDAD to VPH_SINTIC
dtypes: int64(103)
memory usage: 193.3 MB
```

Escalamiento

In [ ]:
```
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()
```

Out[ ]:

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

5 rows × 103 columns

PCA

In [ ]:
```
pca = PCA()
pca.fit(df_cat_vul2)
pcaSummary = pd.DataFrame({'Standard deviation': np.round(np.sqrt(pca.explained_variance_),
                           'Proportion of variance': np.round(pca.explained_variance_ratio_
                           'Cumulative proportion': np.round(np.cumsum(pca.explained_varian
                           })

pcaSummary = pcaSummary.transpose()
pcaSummary.columns = ['PC{}'.format(i) for i in range(1, len(pcaSummary.columns) + 1)]
pcaSummary
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC94	PC95	
Standard deviation	7.26	2.77	1.86	1.80	1.71	1.67	1.53	1.23	1.18	1.16	...	0.01	0.01	
Proportion of variance	52.25	7.57	3.41	3.20	2.89	2.77	2.31	1.51	1.39	1.32	...	0.00	0.00	
Cumulative proportion	52.25	59.82	63.23	66.43	69.32	72.10	74.41	75.92	77.31	78.63	...	100.00	100.00	1

3 rows × 103 columns

In [ ]:
```
pca.components_
```

Out[ ]:
```
array([[-1.53153444e-02, -4.44089210e-16, -7.14073209e-03, ...,
         6.88672095e-02,  1.04256902e-01,  2.88028585e-02],
       [ 6.96552661e-02,  2.22044605e-16,  3.08580507e-02, ...,
         2.12965002e-01,  1.62332955e-01,  2.13399066e-01],
       [ 3.03873964e-02,  9.71445147e-17, -1.75181000e-02, ...,
        -2.91119883e-02,  1.52897971e-02, -1.24835115e-01],
       ...,
       [-0.00000000e+00,  5.98995245e-01,  8.59035065e-15, ...,
        -1.05748743e-14, -3.85684540e-13,  9.86016824e-15],
       [-0.00000000e+00, -7.79558483e-02,  1.56819002e-15, ...,
        -1.66325287e-14,  6.15188456e-13, -1.18238752e-14],
       [ 0.00000000e+00,  1.00794923e-01, -2.62984079e-15, ...,
        -3.88578059e-15, -9.80604487e-14,  1.44328993e-15]])
```

In [ ]:
```
df_pca_vuln = pd.DataFrame(pca.components_.transpose(),
                           columns=pcaSummary.columns,
                           index=df_cat_vul2.columns
                           )
df_pca_vuln
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	
ENTIDAD	-1.531534e-02	6.965527e-02	3.038740e-02	-4.178437e-02	-1.717454e-01	-4.183483e-02	-1.83
NOM_ENT	-4.440892e-16	2.220446e-16	9.714451e-17	-4.440892e-16	-6.106227e-16	4.163336e-16	4.16

LOC	-7.140732e-03	3.085805e-02	-1.751810e-02	-4.011052e-03	-1.106479e-01	-2.433692e-02	-9.54
NOM_LOC	-0.000000e+00	-1.110223e-16	-4.857226e-17	1.387779e-16	-0.000000e+00	-9.714451e-17	-1.11
AGEB	1.079837e-03	-1.620167e-02	-1.481996e-02	2.479652e-02	1.758005e-02	-2.999701e-02	-2.35
...	...	...	...	...	...	...	
VPH_INTER	1.260484e-01	-1.109114e-01	-5.221334e-02	1.713801e-02	-3.695757e-02	7.609884e-02	7.56
VPH_SINRTV	6.730050e-02	1.686195e-01	-8.596397e-02	-6.638657e-03	-5.704332e-02	-8.052921e-03	1.39
VPH_SINLTC	6.886721e-02	2.129650e-01	-2.911199e-02	-3.460645e-02	-5.986976e-02	-9.682028e-02	2.74
VPH_SINCINT	1.042569e-01	1.623330e-01	1.528980e-02	-5.491218e-02	-5.702478e-02	-1.077887e-01	-1.29
VPH_SINTIC	2.880286e-02	2.133991e-01	-1.248351e-01	2.305288e-02	-2.398401e-02	-2.608159e-02	2.32

103 rows × 103 columns

In [ ]:

```
print("=============PC1================")
print("** Más importante:")
print(df_pca_vuln.PC1.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC1.nsmallest(3))
print("=============PC2================")
print("** Más importante:")
print(df_pca_vuln.PC2.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC2.nsmallest(3))
print("=============PC3================")
print("** Más importante:")
print(df_pca_vuln.PC3.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC3.nsmallest(3))
print("=============PC4================")
print("** Más importante:")
print(df_pca_vuln.PC4.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC4.nsmallest(3))
print("=============PC5================")
print("** Más importante:")
print(df_pca_vuln.PC5.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC5.nsmallest(3))
print("=============PC6================")
print("** Más importante:")
print(df_pca_vuln.PC6.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC6.nsmallest(3))
print("=============PC7================")
print("** Más importante:")
print(df_pca_vuln.PC7.nlargest(3))
print("** Menos importante:")
```

```
print("** Menos importante:")
print(df_pca_vuln.PC7.nsmallest(3))
print("=============PC8================")
print("** Más importante:")
print(df_pca_vuln.PC8.nlargest(3))
print("** Menos importante:")
print(df_pca_vuln.PC8.nsmallest(3))
```

```
=============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:
```

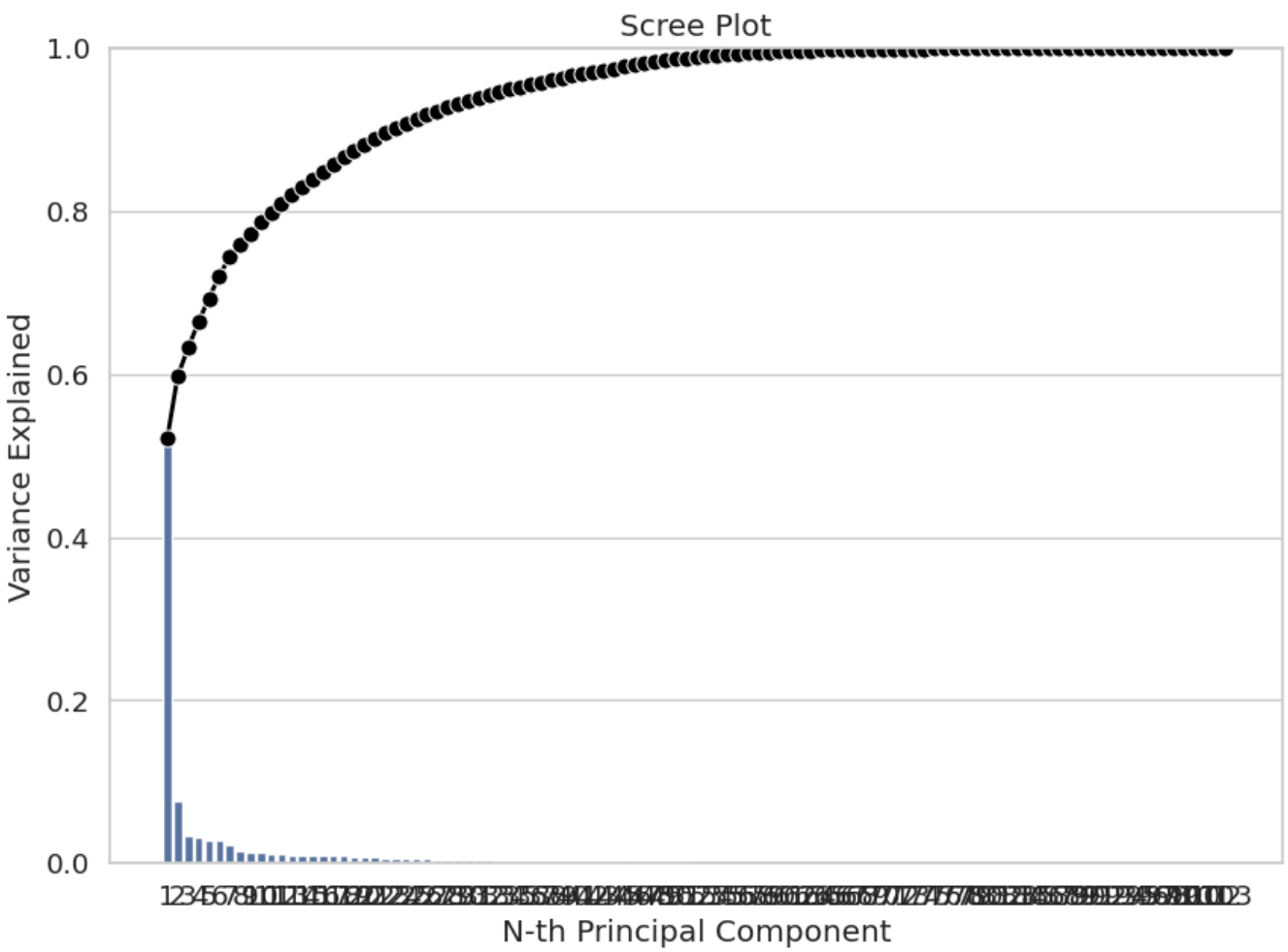
```
P3HLINHE      0.429848
P3HLINHE_M    0.424994
P3HLINHE_F    0.423416
Name: PC6, dtype: float64
** Menos importante:
PROM_OCUP    -0.281461
GRAPROES_F   -0.263184
GRAPROES     -0.262702
Name: PC6, dtype: float64
=============PC7=================
** Más importante:
VPH_NDEAED    0.322533
VPH_S_ELEC    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
```

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.

In [ ]:

```
PC_components = np.arange(pca.n_components_) + 1

scree = sns.set(style = 'whitegrid', font_scale = 1.2)

fig, ax = plt.subplots(figsize=(10, 7))

scree = sns.barplot(x = PC_components, y = pca.explained_variance_ratio_, color = 'b')

scree = sns.lineplot(x = PC_components-1,
                     y = np.cumsum(pca.explained_variance_ratio_),
                     color = 'black',
                     linestyle = '-',
                     linewidth = 2,
                     marker = 'o',
                     markersize = 8
                    )

plt.title('Scree Plot')
plt.xlabel('N-th Principal Component')
plt.ylabel('Variance Explained')
plt.ylim(0, 1)
plt.show()
```



Analisis Factorial

El Análisis Factorial (AF) es una técnica estadística multivariante que busca identificar grupos de variables altamente correlacionadas, denominados factores. Estos factores representan conceptos latentes o constructos subyacentes que explican la variabilidad observada en las variables originales.

Una vez identificadas las principales características , decidimos aplicar un análisi factorial (AF) que es una técnica estadística multivariante que busca identificar grupos de variables altamente correlacionadas (factores) , esto debido a que al tener una gran cantidad de variables socioeconómicas en nuestro data frame, agrupar las variables altamente relacionadas nos permite simplificar la interpretación de los datos haciendo más fácil identificar patrones y tendencias.

Para esto utilizamos el resultado obtenido del paso anterior, es decir de los componentes principales (PCA), usado para crear nuevas variables (componentes principales) que son combinaciones lineales de las variables originales:

In [ ]:
```
 #Análisis Factorial
!pip install factor_analyzer
from factor_analyzer import FactorAnalyzer

corr_matrix = df_pca_vuln.corr()
print("** CORR MATRIX:" )
print(corr_matrix)
if np.isnan(corr_matrix).any().any():
    print("La matriz de correlación contiene valores NaN. Revisa tus datos.")
else:
# Bartlett
```

```
    from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
    chi_square_value,p_value=calculate_bartlett_sphericity(corr_matrix)
    print(chi_square_value, p_value)
# Correlación entre las variables- Barlett - chi-square
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value,p_value=calculate_bartlett_sphericity(corr_matrix)
print("** Barlett:")
print(chi_square_value, p_value)
# KMO
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(corr_matrix)
print("** KMO:")
print(kmo_all)

fa = FactorAnalyzer(n_factors=2, rotation='varimax')
fa.fit(corr_matrix)

print("** Cargas factoriales:")
print(fa.loadings_)
print("** Varianza por cada factor:")
print(fa.get_factor_variance())
```

```
Requirement already satisfied: factor_analyzer in /usr/local/lib/python3.10/dist-packages (0.
5.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from factor
_analyzer) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from factor_
analyzer) (1.13.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from factor_
analyzer) (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from
factor_analyzer) (1.5.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packa
ges (from pandas->factor_analyzer) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from
pandas->factor_analyzer) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (fro
m pandas->factor_analyzer) (2024.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from
scikit-learn->factor_analyzer) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-package
s (from scikit-learn->factor_analyzer) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from pyth
on-dateutil>=2.8.2->pandas->factor_analyzer) (1.16.0)
** CORR MATRIX:
            PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
PC1    1.000000 -0.483069  0.158049 -0.380450 -0.160724 -0.001849 -0.032050
PC2   -0.483069  1.000000  0.023640 -0.056905 -0.024040 -0.000277 -0.004794
PC3    0.158049  0.023640  1.000000  0.018618  0.007865  0.000090  0.001568
PC4   -0.380450 -0.056905  0.018618  1.000000 -0.018933 -0.000218 -0.003775
PC5   -0.160724 -0.024040  0.007865 -0.018933  1.000000 -0.000092 -0.001595
...         ...       ...       ...       ...       ...       ...       ...
PC99   0.002828  0.000423 -0.000138  0.000333  0.000141  0.000002  0.000028
PC100 -0.110786 -0.016571  0.005422 -0.013050 -0.005513 -0.000063 -0.001099
PC101 -0.112223 -0.016786  0.005492 -0.013220 -0.005585 -0.000064 -0.001114
PC102  0.044273  0.006622 -0.002167  0.005215  0.002203  0.000025  0.000439
PC103 -0.190751 -0.028531  0.009335 -0.022470 -0.009493 -0.000109 -0.001893

            PC8       PC9      PC10  ...      PC94      PC95      PC96  \
PC1   -0.097578 -0.294683  0.145667  ...  0.001671  0.001978  0.003009
PC2   -0.014595 -0.044077  0.021788  ...  0.000250  0.000296  0.000450
PC3    0.004775  0.014421 -0.007128  ... -0.000082 -0.000097 -0.000147
```

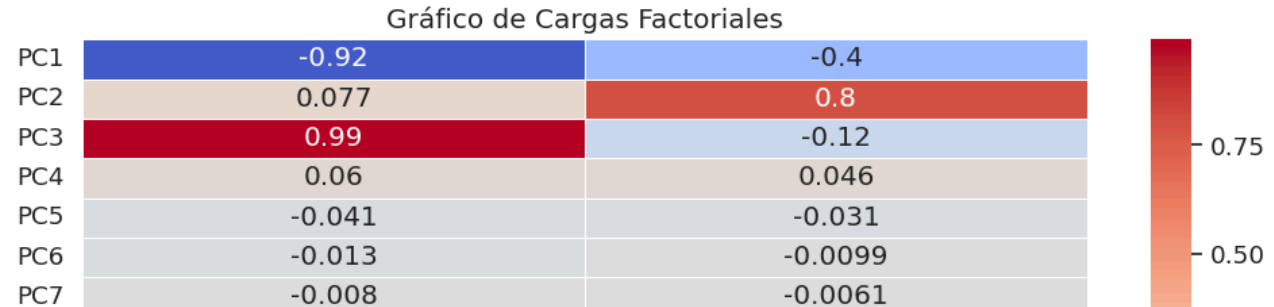
```
PC4   -0.011495 -0.034713  0.017159  ...  0.000197  0.000233  0.000354
PC5   -0.004856 -0.014665  0.007249  ...  0.000083  0.000098  0.000150
...         ...       ...       ...  ...       ...       ...       ...
PC99   0.000085  0.000258 -0.000128  ... -0.000001 -0.000002 -0.000003
PC100 -0.003347 -0.010108  0.004997  ...  0.000057  0.000068  0.000103
PC101 -0.003391 -0.010240  0.005062  ...  0.000058  0.000069  0.000105
PC102  0.001338  0.004040 -0.001997  ... -0.000023 -0.000027 -0.000041
PC103 -0.005763 -0.017405  0.008603  ...  0.000099  0.000117  0.000178

                PC97      PC98      PC99     PC100     PC101     PC102  \
PC1   -7.746680e-04  0.006216  0.002828 -0.110786 -0.112223  0.044273
PC2   -1.158695e-04  0.000930  0.000423 -0.016571 -0.016786  0.006622
PC3    3.790983e-05 -0.000304 -0.000138  0.005422  0.005492 -0.002167
PC4   -9.125526e-05  0.000732  0.000333 -0.013050 -0.013220  0.005215
PC5   -3.855146e-05  0.000309  0.000141 -0.005513 -0.005585  0.002203
...              ...       ...       ...       ...       ...       ...
PC99   6.783941e-07 -0.000005  1.000000  0.000097  0.000098 -0.000039
PC100 -2.657325e-05  0.000213  0.000097  1.000000 -0.003850  0.001519
PC101 -2.691796e-05  0.000216  0.000098 -0.003850  1.000000  0.001538
PC102  1.061941e-05 -0.000085 -0.000039  0.001519  0.001538  1.000000
PC103 -4.575380e-05  0.000367  0.000167 -0.006543 -0.006628  0.002615

          PC103
PC1   -0.190751
PC2   -0.028531
PC3    0.009335
PC4   -0.022470
PC5   -0.009493
...         ...
PC99   0.000167
PC100 -0.006543
PC101 -0.006628
PC102  0.002615
PC103  1.000000

[103 rows x 103 columns]
2500.4389828527355 1.0
** Barlett:
2500.4389828527355 1.0
** KMO:
[0.4167016  0.46498125 0.03618737 0.58011524 0.13058377 0.00950941
 0.01087442 0.03826371 0.55080611 0.0335284  0.2435731  0.02198908
 0.01896132 0.13270264 0.04573061 0.0966525  0.00991803 0.07081415
 0.0098755  0.01101712 0.00945873 0.01494122 0.01247403 0.01290317
 0.00945292 0.00955176 0.02110677 0.01363366 0.01005542 0.01215001
 0.01264684 0.02526834 0.00989243 0.00943839 0.01415087 0.00993265
 0.01625738 0.00946968 0.01176961 0.01094726 0.00967452 0.00944988
 0.01016865 0.0094422  0.00954463 0.00979924 0.00947014 0.00974434
 0.01255571 0.00948245 0.01324628 0.00989966 0.0105564  0.00969207
 0.01098664 0.01324007 0.01081015 0.00949579 0.00945529 0.01000452
 0.00949051 0.01012827 0.00951041 0.00981423 0.01139682 0.00985899
 0.00997952 0.01023459 0.00944176 0.00944092 0.00981401 0.01135584
 0.01023039 0.00984994 0.01152988 0.00957885 0.00951931 0.01061366
 0.01590153 0.00998911 0.00961504 0.00956983 0.00983983 0.01083455
 0.00946717 0.00944066 0.01014804 0.00988681 0.01896558 0.00963057
 0.00945582 0.00944781 0.00966081 0.00961652 0.00962805 0.00966921
 0.00953717 0.00982112 0.00966172 0.05051302 0.05203452 0.01381762
 0.21073389]
/usr/local/lib/python3.10/dist-packages/factor_analyzer/utils.py:244: UserWarning: The invers
e of the variance-covariance matrix was calculated using the Moore-Penrose generalized matrix
inversion, due to its determinant being at or very close to zero.
  warnings.warn(
** Cargas factoriales:
[[-9.42160646e-01 -5.11134441e-01]
 [ 3.82037109e-02  1.00342630e+00]
```

```
[-1.67455031e-01 -3.92329238e-02]
[ 3.68616940e-01  7.56489865e-02]
[ 1.47392711e-01  3.60663884e-02]
[-7.78281093e-03 -1.71419251e-03]
[ 2.18863154e-02  5.62987107e-03]
[ 8.59307705e-02  2.14066113e-02]
[ 2.79072294e-01  6.33567936e-02]
[-1.54867146e-01 -3.65087458e-02]
[ 1.86595635e-01  4.49396809e-02]
[ 6.07723625e-02  1.52422279e-02]
[ 5.38589832e-02  1.35385536e-02]
[ 1.48306362e-01  3.62782668e-02]
[ 9.41454982e-02  2.34037953e-02]
[ 1.30975653e-01  3.22242875e-02]
[ 1.22013135e-02  3.23062172e-03]
[ 1.15096033e-01  2.84514628e-02]
[ 1.15815599e-02  3.07711645e-03]
[-3.10675918e-02 -7.44967276e-03]
[-5.07054577e-03 -1.04391901e-03]
[ 4.20254943e-02  1.06157445e-02]
[ 3.17289965e-02  8.06776981e-03]
[ 3.38104204e-02  8.58309392e-03]
[ 5.57337735e-04  3.47890453e-04]
[ 5.04869246e-03  1.45942968e-03]
[ 5.88811935e-02  1.47765199e-02]
[ 3.70260142e-02  9.37899135e-03]
[ 1.40222649e-02  3.68167678e-03]
[-4.11283057e-02 -9.91369603e-03]
[-4.49967554e-02 -1.08582533e-02]
[ 6.71202215e-02  1.68033484e-02]
[-1.70029934e-02 -3.98979790e-03]
[-2.03604633e-03 -2.93624452e-04]
[-5.55536407e-02 -1.34266019e-02]
[-1.76827730e-02 -4.15736606e-03]
[ 4.63657744e-02  1.16886021e-02]
[-5.81702872e-03 -1.22842932e-03]
[ 2.78973158e-02  7.11887381e-03]
[-3.03634067e-02 -7.27683544e-03]
[-1.26908548e-02 -2.92614821e-03]
[-4.30293729e-03 -8.54161049e-04]
[-2.12502467e-02 -5.03623538e-03]
[-6.85404217e-04  4.04474953e-05]
[-9.09932666e-03 -2.03940937e-03]
[ 1.03829412e-02  2.78024975e-03]
[ 1.76624449e-03  6.47014086e-04]
[ 9.43368054e-03  2.54515945e-03]
[ 3.21380053e-02  8.16904125e-03]
[-6.53847156e-03 -1.40672601e-03]
[-4.93793520e-02 -1.19262011e-02]
[ 1.19374042e-02  3.16525396e-03]
[-2.61379809e-02 -6.23880934e-03]
[ 8.44110352e-03  2.29935939e-03]
[-3.07619321e-02 -7.37465627e-03]
[ 3.53393143e-02  8.96155028e-03]
[-2.89406876e-02 -6.92750202e-03]
[-7.18983103e-03 -1.56768117e-03]
[-4.79678863e-03 -9.76247385e-04]
[-1.88376494e-02 -4.44197538e-03]
[ 2.81062375e-03  9.05466322e-04]
[-2.06814325e-02 -4.89616565e-03]
[-7.82404734e-03 -1.72438039e-03]
[ 1.06286425e-02  2.84110174e-03]
[-3.46903100e-02 -8.33810645e-03]
[-1.64164434e-02 -3.84518642e-03]
[ 1.30471684e-02  3.44013757e-03]
```

```
[-2.21501173e-02 -5.25777835e-03]
[-7.63177813e-04  2.12088747e-05]
[-3.07687936e-03 -5.51018661e-04]
[-1.55929305e-02 -3.64211479e-03]
[-3.43140213e-02 -8.24588471e-03]
[ 1.60455384e-02  4.18288659e-03]
[-1.62542034e-02 -3.80518282e-03]
[-3.58907910e-02 -8.63222991e-03]
[-1.01878876e-02 -2.30824697e-03]
[-8.17956340e-03 -1.81221092e-03]
[-2.67923342e-02 -6.39965968e-03]
[ 4.52476088e-02  1.14122893e-02]
[-1.85958446e-02 -4.38239220e-03]
[-1.12086711e-02 -2.56028766e-03]
[-9.91458250e-03 -2.24075579e-03]
[-1.60708097e-02 -3.75996132e-03]
[-2.91980907e-02 -6.99071811e-03]
[-5.65909279e-03 -1.18939398e-03]
[-9.83212475e-04 -3.32192365e-05]
[-2.09617564e-02 -4.96519807e-03]
[-1.69057609e-02 -3.96582712e-03]
[-8.36937660e-02 -2.01912992e-02]
[-1.16158212e-02 -2.66080064e-03]
[-4.84012604e-03 -9.86960452e-04]
[ 6.23089709e-05  2.25418293e-04]
[-1.23664723e-02 -2.84608875e-03]
[-1.12480902e-02 -2.57001944e-03]
[-1.15509502e-02 -2.64478659e-03]
[-1.25663688e-02 -2.89542506e-03]
[-8.84056658e-03 -1.97549523e-03]
[-1.57259074e-02 -3.67490880e-03]
[-1.23883198e-02 -2.85148107e-03]
[ 9.87981351e-02  2.45307934e-02]
[ 1.00197611e-01  2.48691556e-02]
[-5.33313312e-02 -1.28871236e-02]
[ 1.76627375e-01  4.27292321e-02]]
** Varianza por cada factor:
(array([1.39547972, 1.29514728]), array([0.01354835, 0.01257425]), array([0.01354835, 0.02612
259]))
```

In [ ]:

```
#Graficando
fa.fit(pcsComponents_df)
loadings = fa.loadings_
df_loadings = pd.DataFrame(loadings, index=pcsComponents_df.columns, columns=['Factor 1',


plt.figure(figsize=(12, 8))
sns.heatmap(df_loadings, annot=True, cmap='coolwarm', center=0, linewidths=.5)
plt.title('Gráfico de Cargas Factoriales')
plt.xlabel('Factores')
plt.ylabel('Variables')
plt.show()
```



Variables	Factor 1	Factor 2
PC8	-0.015	-0.012
PC9	-0.0043	-0.0033
PC10	-0.0038	-0.0029
PC11	-0.00076	-0.00058
PC12	-0.0032	-0.0024
PC13	0.0015	0.0011
PC14	-0.00078	-0.0006
PC15	0.0013	0.00097
PC16	-0.001	-0.00078
PC17	0.00013	0.0001
PC18	-0.00054	-0.00042
PC19	-0.00068	-0.00052
PC20	-0.00018	-0.00013
PC21	-5.9e-06	-4.5e-06

Factores

Partes 8 a 10. Algoritmo Kmeans, Sub/sobreajuste, Métricas y Desempeño

Kmeans

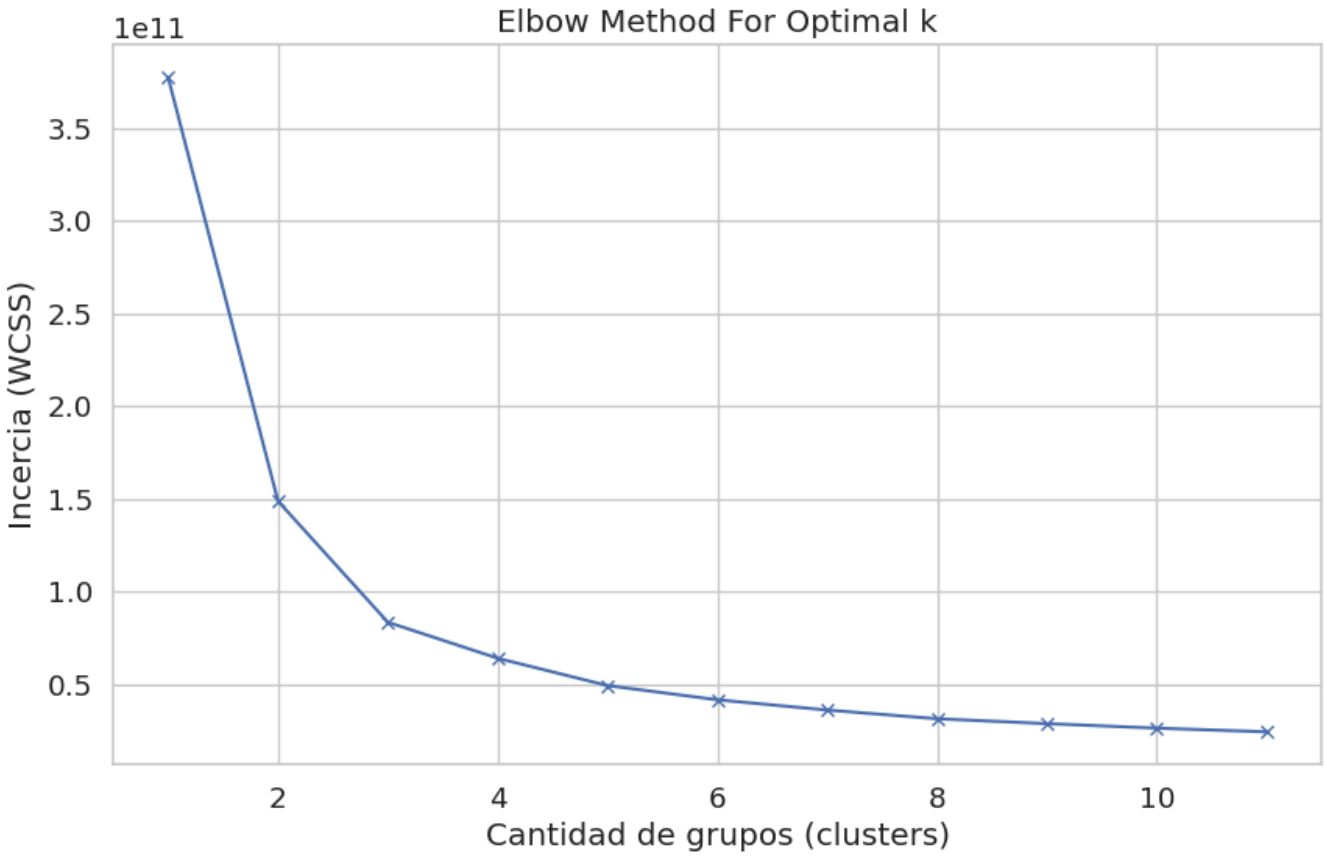
In [ ]:
```
from sklearn.cluster import KMeans
```

Modelo y análisis de 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.

In [ ]:
```
K = range(1,12)
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitud
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(df_cat2)
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()
```

```
1 2 3 4 5 6 8 9 10 11
```



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

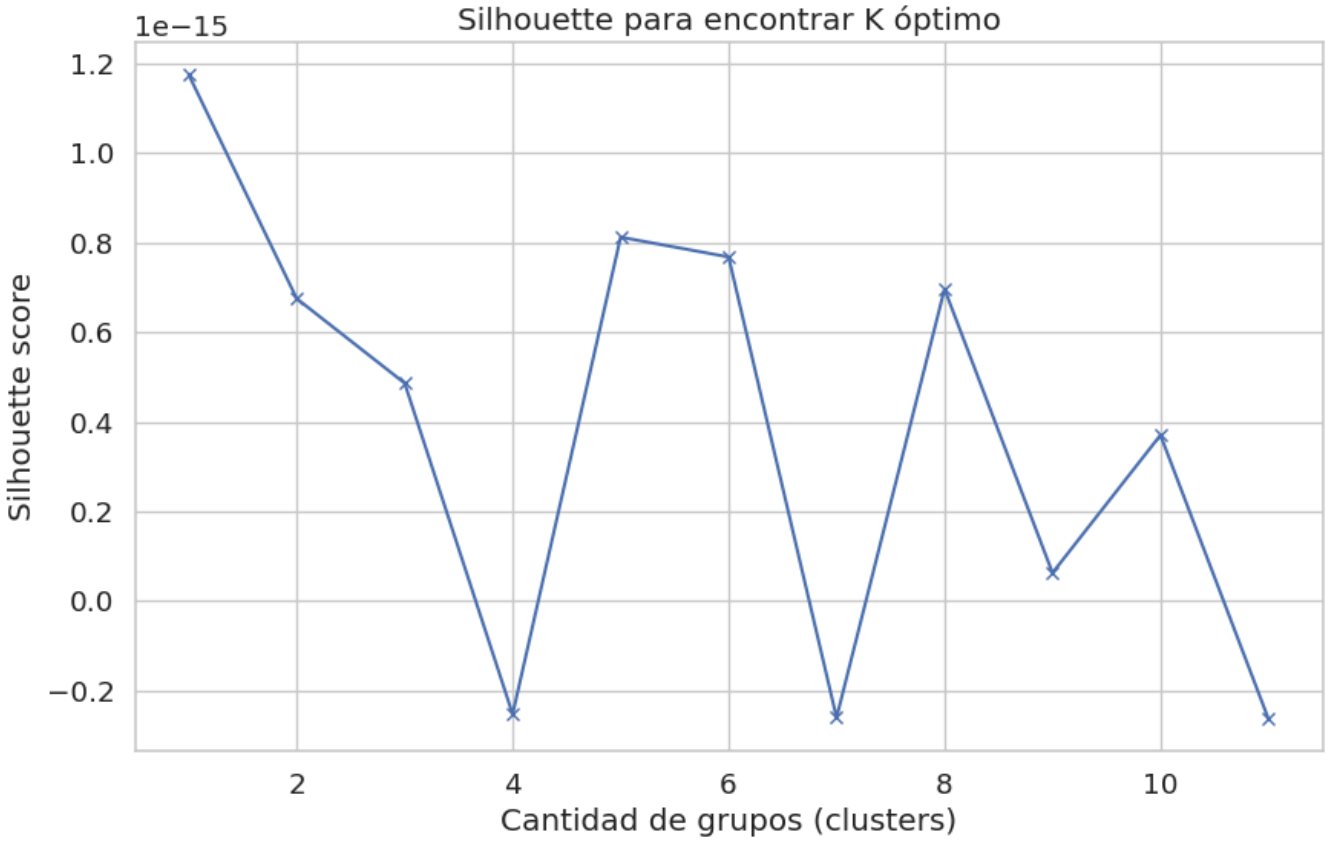
El uso del método del codo nos permite conocer la cantidad de grupos que se pueden formar con nuestra base de datos. Una vez realizado el paso de selección y extracción de características, procedimos a calcular el valor de k. Esto será utilizado más adelante cuando se definan los parámetros del k-means. En el resultado obtenido, observamos que la inercia disminuye casi en su totalidad. Al visualizar los resultados obtenidos para el método de codo y silueta, nos fue posible determinar un valor K óptimo igual a 4.

In [ ]:

```
from sklearn.metrics import silhouette_score
#El valor de la silueta mide qué tan similar es un punto a su propio grupo (cohesión)
#en contraste con otros grupos (separación). Un valor alto indica el clúster correcto.
#Silhouette Score alcanza su máximo global en el k óptimo, lo que idealmente se vería
#como un pico en el gráfico de valores Silhouette contra k (cantidad de grupos/clusters)
sil=[]
kmax=12
#La disimilitud no se definiría para un solo grupo, por lo tanto, el número mínimo de grup
```

```
#La disimilitud no se definiría para un solo grupo, por lo tanto, el número mínimo de grupo
for k in range(2, kmax+1):
  kmeans=KMeans(n_clusters=k).fit(df_pca_vuln)
  print(k, end=' ')
  labels=kmeans.labels_
  #Incoporar el resultado a una lista
  sil.append(silhouette_score(df_pca_vuln, labels, metric='euclidean'))
#Personalización de la gráfica
plt.figure(figsize=(10, 6))
plt.plot(K, sil, 'bx-')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Silhouette score')
plt.title('Silhouette para encontrar K óptimo')
plt.show()
```

```
2 3 4 5 6 7 8 9 10 11 12
```



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

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.

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.

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.

In [ ]:
```
kmeans = KMeans(n_clusters=4, random_state=1)
y_kmeans = kmeans.fit_predict(df_pca_vuln)
```

In [ ]:
```
#Observamos los centros de los grupos que el modelo nos ha generado
centros=kmeans.cluster_centers_
for i in range(0,4):
  print('Grupo',i,'Centro:',centros[i-1][0].round(2),)
```

```
Grupo 0 Centro: 0.03
Grupo 1 Centro: 0.1
Grupo 2 Centro: 0.09
Grupo 3 Centro: 0.09
```

In [ ]:
```
clusterCount=np.bincount(y_kmeans) #np.bincount(kmeans.labels_)
for i in range(0,4):
  print('Grupo ',i,', registros del grupo:',clusterCount[i-1])
```

```
Grupo  0 , registros del grupo: 1
Grupo  1 , registros del grupo: 6
Grupo  2 , registros del grupo: 79
Grupo  3 , registros del grupo: 17
```

In [ ]:
```
#Creamos una función para calcular la distancia del centro con los puntos
def calculoDistancia(centro,X,Y):
  #Creamos una lista vacía para añadir los resultados
  distancias=[]

  cx, cy=centro
#Iteramos en los puntos para calcular la distancia con base en la fórmula ya establecida
  for x,y in list(zip(X,Y)):
    difX=(x-cx)**2
    difY=(y-cy)**2
    distancia=np.sqrt(difX+difY)
    #Añadimos los resultados a una lsita
```

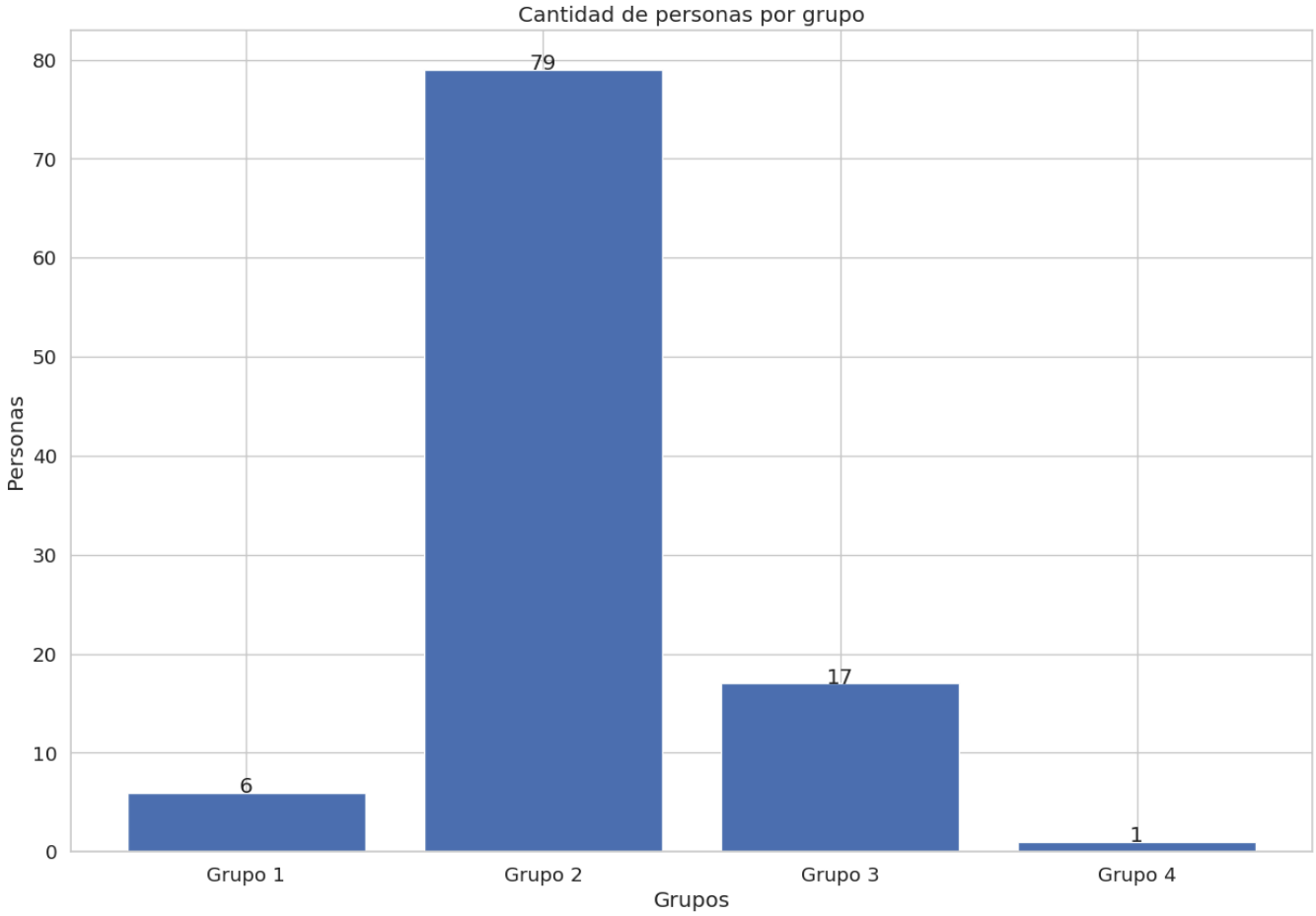
```
    distancias.append(distancia)

#Regresamos los valores obtenidos
  return distancias
```

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.

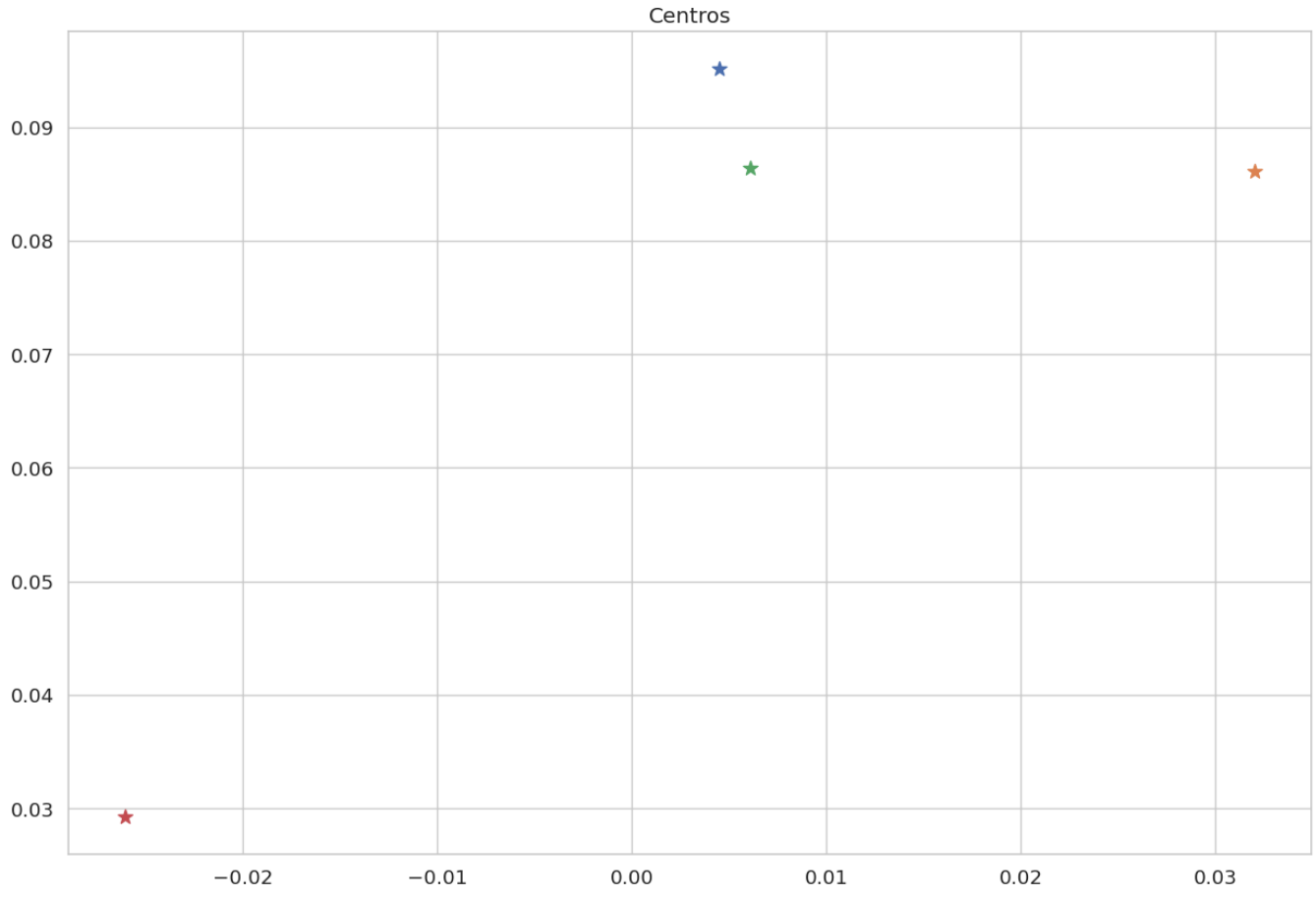
In [ ]:
```
#Se comienza en cero para evitar un problema con los índices de Python
for i in range(0,4):
  exec("c" + str(i) + '=kmeans.cluster_centers_[i]')
```

In [ ]:
```
grupos=['Grupo 1','Grupo 2','Grupo 3','Grupo 4']
plt.bar(grupos,clusterCount)
plt.xlabel('Grupos')
plt.ylabel('Personas')
plt.title('Cantidad de personas por grupo')
for i in range(len(clusterCount)):
        plt.text(i, clusterCount[i], clusterCount[i], ha = 'center')
```



In [ ]:
```
#Creamos una lista que contiene todas las variables correspondientes a los centros
varsCentros=[c0,c1,c2,c3]

#Recorremos la lista anterior para graficar los centros en función de la longitud y latitud
for i in range(0,4):
  plt.scatter(varsCentros[i][1], varsCentros[i][0],marker='*',s=100)
```

```
plt.title('Centros')
```



Evaluación y desempeño

In [ ]:
```
kmeans.inertia_
```

Out[ ]:
```
99.00000000000004
```

Conclusiones al 13 de octubre 2024

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

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.

Fuentes consultadas al 13 de octubre 2024

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

Avance 4 (Entrega: 20 octubre 2024)

En esta sección mostraremos los algoritmos que consideramos apropiados para el proyecto. Se podrá notar que en todos los casos se trabajó con no supervisados ya que los datos poblacionales no están etiquetados. Sin embargo, es posible realizar un análisis supervisado una vez creada la variable "Vuln_Score", ya que esa sería nuestra salida para comparar los resultados.

Cabe destacar que se obtuvieron dos gráficas para entender el comportamiento de los datos en comparación con el índice de vulnerabilidad obtenido. Por tanto, como se puede observar en la siguiente imagen, se realizó en primera instancia el análisis de la relación entre las variables "Población Total" y "AGEB"; y para la asignación de color, se utilizó la variable que construimos, denominada "Vuln_Score", que es el índice de vulnerabilidad.

Sin embargo, una de las limitaciones que encontramos en dicha gráfica es que no es posible comprender si hay un comportamiento de clases para los diferentes tipos de vulnerabilidad. Por tanto, decidimos graficar las variables "Población Total" y "Vuln_Score", usando esta última nuevamente para colorear los resultados. En la siguiente imagen se puede observar que el índice de vulnerabilidad se encuentra presente para todos los tamaños de población.

Parte 11. Preparación de algoritmos

Copiamos el resumen de las etapas anteriores, para correr por bloque el avance 4

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

In [ ]:
```
# Leemos los datos por Estado
# Hidalgo
df_hidalgo = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# CDMX 1
df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/hea
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_M
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
```

```
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
           'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
           'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
          'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
          'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
           'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
          'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC','P_0A2', 'P_0A2_F', 'P_0A
# CDMX 2
df_cdmx2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/heac
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
           'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_MO
           'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
          'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
           'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
           'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
          'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
          'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
           'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
          'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 1
df_edomex1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
           'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_MO
           'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
          'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
           'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
           'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
          'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
          'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
           'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
          'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 2
df_edomex2 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
           'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_MO
           'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
          'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
           'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
           'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
          'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
          'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
           'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
          'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 3
df_edomex3 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', 'I
           'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
           'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_MO
           'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
          'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', 'I
            'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PI
           'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
```

```
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
# EdoMex 4
df_edomex4 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/he
                 usecols=['NOM_LOC','NOM_ENT','ENTIDAD',  'LOC', 'AGEB', 'MZA', 'POBTOT', '
            'P_60YMAS', 'P_60YMAS_F', 'P_60YMAS_M', 'POB65_MAS', 'P3HLINHE', 'P3HLINHE_F',
            'PCON_DISC', 'PCDISC_MOT', 'PCDISC_VIS', 'PCDISC_LENG', 'PCDISC_AUD', 'PCDISC_MO
            'PCON_LIMI', 'PCLIM_CSB', 'PCLIM_VIS', 'PCLIM_HACO', 'PCLIM_OAUD', 'PCLIM_MOT2',
           'PCLIM_PMEN', 'PSIND_LIM', 'P15YM_AN', 'P15YM_AN_F', 'P15YM_AN_M', 'P15YM_SE', '
             'GRAPROES', 'GRAPROES_F', 'GRAPROES_M', 'PEA', 'PEA_F', 'PEA_M', 'PSINDER', 'PE
            'PDER_ISTE', 'PDER_ISTEE', 'PAFIL_PDOM', 'PDER_SEGP', 'PDER_IMSSB', 'PAFIL_IPRIV
            'TOTHOG', 'POBHOG', 'VIVTOT', 'TVIVHAB', 'TVIVPAR', 'VIVPAR_HAB', 'TVIVPARHAB',
            'PROM_OCUP', 'VPH_PISODT', 'VPH_PISOTI', 'VPH_C_ELEC', 'VPH_S_ELEC', 'VPH_AGUADV
           'VPH_AGUAFV', 'VPH_TINACO', 'VPH_CISTER', 'VPH_EXCSA', 'VPH_LETR', 'VPH_DRENAJ',
           'VPH_C_SERV', 'VPH_NDEAED', 'VPH_DSADMA', 'VPH_NDACMM', 'VPH_SNBIEN', 'VPH_REFRI
            'VPH_MOTO', 'VPH_BICI', 'VPH_RADIO', 'VPH_TV', 'VPH_PC', 'VPH_TELEF', 'VPH_CEL',
           'VPH_SINRTV', 'VPH_SINLTC', 'VPH_SINCINT', 'VPH_SINTIC', 'P_0A2', 'P_0A2_F', 'P_(
```

```
<ipython-input-146-90ee3e741790>:18: DtypeWarning: Columns (178) have mixed types. Specify dt
ype option on import or set low_memory=False.
  df_cdmx1 = pd.read_csv("https://raw.githubusercontent.com/LH-1169213/IslasDeCalor/refs/head
s/main/CDMX.csv",
```

In [ ]:
```
df3e = [df_hidalgo,df_cdmx1,df_cdmx2,df_edomex1,df_edomex2,df_edomex3,df_edomex4]
df3e = pd.concat(df3e, ignore_index=True)
df3e.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	V
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	...	

5 rows × 103 columns

In [ ]:
```
df_cat1 = df3e.copy()

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

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VP
4	13.0	Hidalgo	1.0	Acatlán	0043	1.0	92.0	54	38	6	...	

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	...
7	13.0	Hidalgo	1.0	Acatlán	0043	6.0	47.0	20	27	*	...
8	13.0	Hidalgo	1.0	Acatlán	0043	7.0	44.0	21	23	*	...

5 rows × 103 columns

In [ ]:
```
df_cat2 = df_cat1.copy()

df_cat2.replace('*', np.nan, inplace=True)
df_cat2.head()
df_cat2.dropna(how='all', inplace = True)

# Hacemos el cambio para que todas las variables sean numéricas
df_cat2= df_cat2.apply(pd.to_numeric, errors='coerce').fillna(0).astype(int)
df_cat2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 243657 entries, 4 to 253243
Columns: 103 entries, ENTIDAD to VPH_SINTIC
dtypes: int64(103)
memory usage: 193.3 MB
```

In [ ]:
```
df_indi = df_cat2.copy()
for index, row in df_indi.iterrows():
    if row['P3HLINHE'] != 0 or row['PHOG_IND'] != 0:
        df_indi.at[index, 'Vuln_Indigena'] = (int(df_indi.at[index,'P3HLINHE']) + int(df_i


bins = [0, 1000, 3000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_indi['Vuln_Indigena_Discreta'] = pd.cut(df_indi['Vuln_Indigena'], bins=bins, labels=lab
df_indi['Vuln_Indigena_Discreta_Int'] = df_indi['Vuln_Indigena_Discreta'].cat.codes + 1


df_ed = df_cat2.copy()
for index, row in df_ed.iterrows():
    if row['P_0A2'] != 0 or row['P_3A5'] != 0 or row['P_60YMAS'] != 0:
        df_ed.at[index, 'Vuln_Edad'] = (int(df_ed.at[index,'P_0A2']) + int(df_ed.at[index,



df_disc = df_cat2.copy()
for index, row in df_disc.iterrows():
    if row['PCON_DISC'] != 0 or row['PCON_LIMI'] != 0:
        df_disc.at[index, 'Vuln_Disc_Lim'] = (int(df_disc.at[index,'PCON_DISC']) + int(df_


bins = [0, 1000, 3000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_disc['Vuln_Disc_Discreta'] = pd.cut(df_disc['Vuln_Disc_Lim'], bins=bins, labels=labels)
df_disc['Vuln_Disc_Discreta_Int'] = df_disc['Vuln_Disc_Discreta'].cat.codes + 1
```

```
df_escs = df_cat2.copy()
for index, row in df_escs.iterrows():
    if row['P15YM_AN'] != 0 or row['P15YM_SE'] != 0 or row['PSINDER']:
        df_escs.at[index, 'Vuln_Esc_Salud'] = (int(df_escs.at[index,'P15YM_AN']) + int(df_


# Bajo 0-1000, Medio 1000-15000, Alto 15000<
bins = [0, 1000, 15000, np.inf]
labels = ['Bajo', 'Medio', 'Alto']

# Añadiendo  las columnas
df_escs['Vuln_Salud_Discreta'] = pd.cut(df_escs['Vuln_Esc_Salud'], bins=bins, labels=labels
df_escs['Vuln_Salud_Discreta_Int'] = df_escs['Vuln_Salud_Discreta'].cat.codes + 1

df_hog = df_cat2.copy()
for index, row in df_hog.iterrows():
    if row['VPH_PISOTI'] != 0 or row['VPH_NDEAED'] != 0 :
        df_hog.at[index, 'Vuln_Hog_Serv'] = (int(df_hog.at[index,'VPH_PISOTI']) + int(df_h


df_con = df_cat2.copy()
for index, row in df_con.iterrows():
    if row['VPH_SINRTV'] != 0 or row['VPH_SINLTC'] != 0 or row['VPH_SINCINT']:
        df_con.at[index, 'Vuln_Hog_Connect'] = (int(df_con.at[index,'VPH_SINRTV']) + int(d
```

In [ ]:
```
df_cat_vul1=df_cat2.copy()
df_cat_vul1 = pd.concat([df_indi, df_ed, df_disc, df_escs, df_hog, df_con],axis=1)
df_cat_vul1.head()
```

Out[ ]:

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VP
4	13	0	1	0	43	1	92	54	38	6	...	
5	13	0	1	0	43	2	7	4	3	0	...	
6	13	0	1	0	43	3	0	0	0	0	...	
7	13	0	1	0	43	6	47	20	27	0	...	
8	13	0	1	0	43	7	44	21	23	0	...	

5 rows × 630 columns

In [ ]:
```
df_cat_vul1[['Vuln_Indigena_Discreta_Int','Vuln_Edad','Vuln_Disc_Lim','Vuln_Disc_Discreta_
```

Out[ ]:

	Vuln_Indigena_Discreta_Int	Vuln_Edad	Vuln_Disc_Lim	Vuln_Disc_Discreta_Int	Vuln_Esc_Salud
4	0	22.0	30.0	1	50.0
5	0	NaN	NaN	0	NaN
6	0	NaN	NaN	0	NaN
7	0	7.0	11.0	1	26.0

8	1	6.0	10.0	1	14.0
...	...	...	...	...	...
253239	0	NaN	NaN	0	NaN
253240	0	NaN	NaN	0	4.0
253241	0	NaN	NaN	0	NaN
253242	0	NaN	3.0	1	NaN
253243	0	NaN	NaN	0	NaN

243657 rows × 8 columns

In [ ]:
```
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[ ]:

	ENTIDAD	NOM_ENT	LOC	NOM_LOC	AGEB	MZA	POBTOT	POBFEM	POBMAS	P_0A2	...	VP
4	13	0	1	0	43	1	92	54	38	6	...	
5	13	0	1	0	43	2	7	4	3	0	...	
6	13	0	1	0	43	3	0	0	0	0	...	
7	13	0	1	0	43	6	47	20	27	0	...	
8	13	0	1	0	43	7	44	21	23	0	...	

5 rows × 631 columns

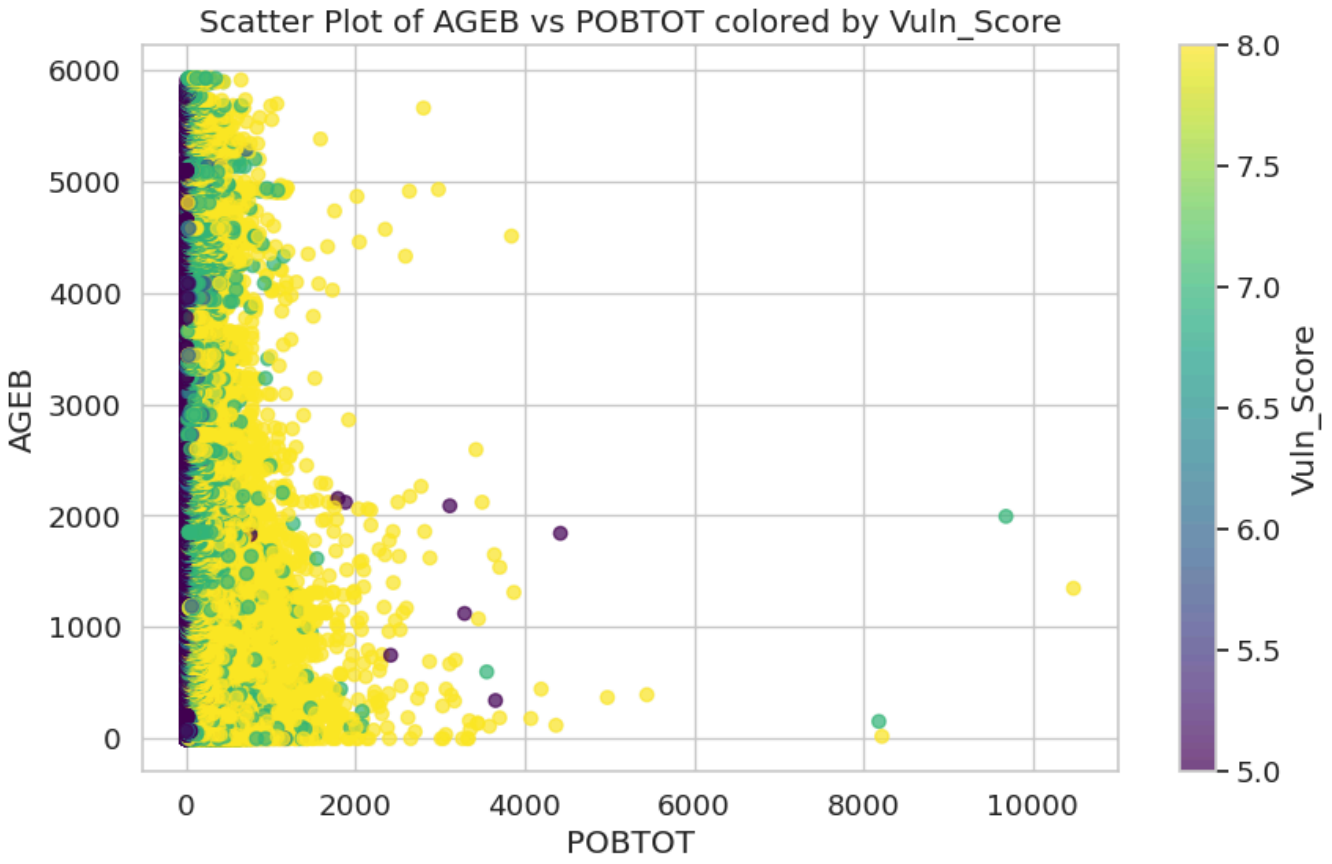
In [ ]:
```
vuln_df = df_cat2.copy()

vuln_df['Vuln_Score'] = vulnerable_df['Vuln_Score']
print(vuln_df.head())
```

```
   ENTIDAD  NOM_ENT  LOC  NOM_LOC  AGEB  MZA  POBTOT  POBFEM  POBMAS  P_0A2  \
4       13        0    1        0    43    1      92      54      38      6
5       13        0    1        0    43    2       7       4       3      0
6       13        0    1        0    43    3       0       0       0      0
7       13        0    1        0    43    6      47      20      27      0
8       13        0    1        0    43    7      44      21      23      0

   ...  VPH_TV  VPH_PC  VPH_TELEF  VPH_CEL  VPH_INTER  VPH_SINRTV  VPH_SINLTC  \
4  ...      21       7          5       21          8           4           3
5  ...       0       0          0        0          0           0           0
6  ...       0       0          0        0          0           0           0
```

```
7  ...      10       6          9       11          9          0          0
8  ...      12       3          9        9          4          0          0

   VPH_SINCINT  VPH_SINTIC  Vuln_Score
4           14           0           7
5            0           0           5
6            0           0           5
7            4           0           7
8            6           0           8

[5 rows x 104 columns]
```

In [ ]:

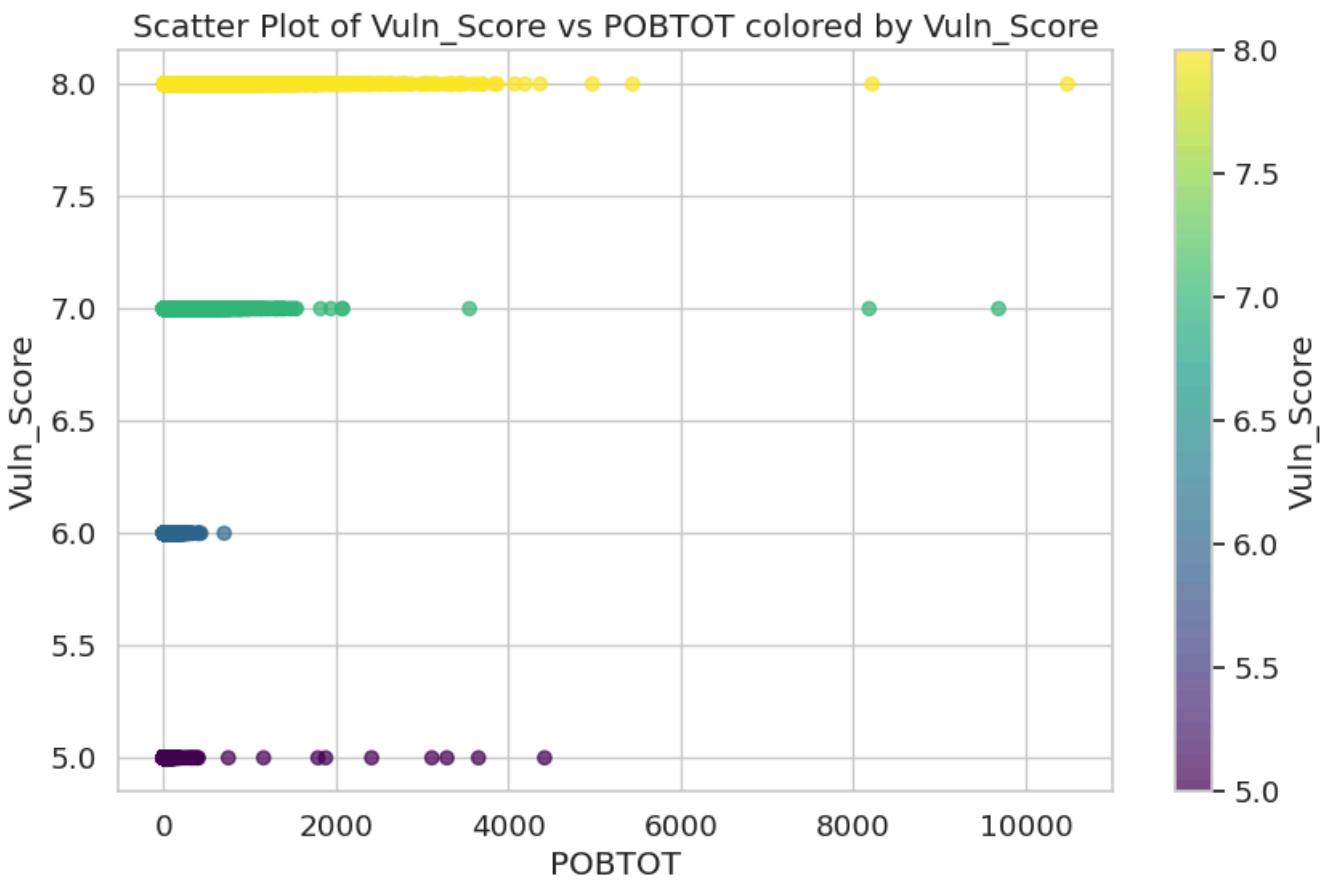
```
plt.figure(figsize=(10, 6))
sc = plt.scatter(vuln_df['POBTOT'], vuln_df['AGEB'], c=vuln_df['Vuln_Score'], cmap='viridis

# Add color bar to show the color scale
plt.colorbar(sc, label='Vuln_Score')

# Add labels and title
plt.xlabel('POBTOT')
plt.ylabel('AGEB')
plt.title('Scatter Plot of AGEB vs POBTOT colored by Vuln_Score')

# Show the plot
plt.show()
```



In [ ]:

```
plt.figure(figsize=(10, 6))
sc = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=vuln_df['Vuln_Score'], cmap='

# Add color bar to show the color scale
plt.colorbar(sc, label='Vuln_Score')

# Add labels and title
plt.xlabel('POBTOT')
```

```
plt.ylabel('Vuln_Score')
plt.title('Scatter Plot of Vuln_Score vs POBTOT colored by Vuln_Score')

# Show the plot
plt.show()
```



Parte 12. Construcción de modelos

No supervisados

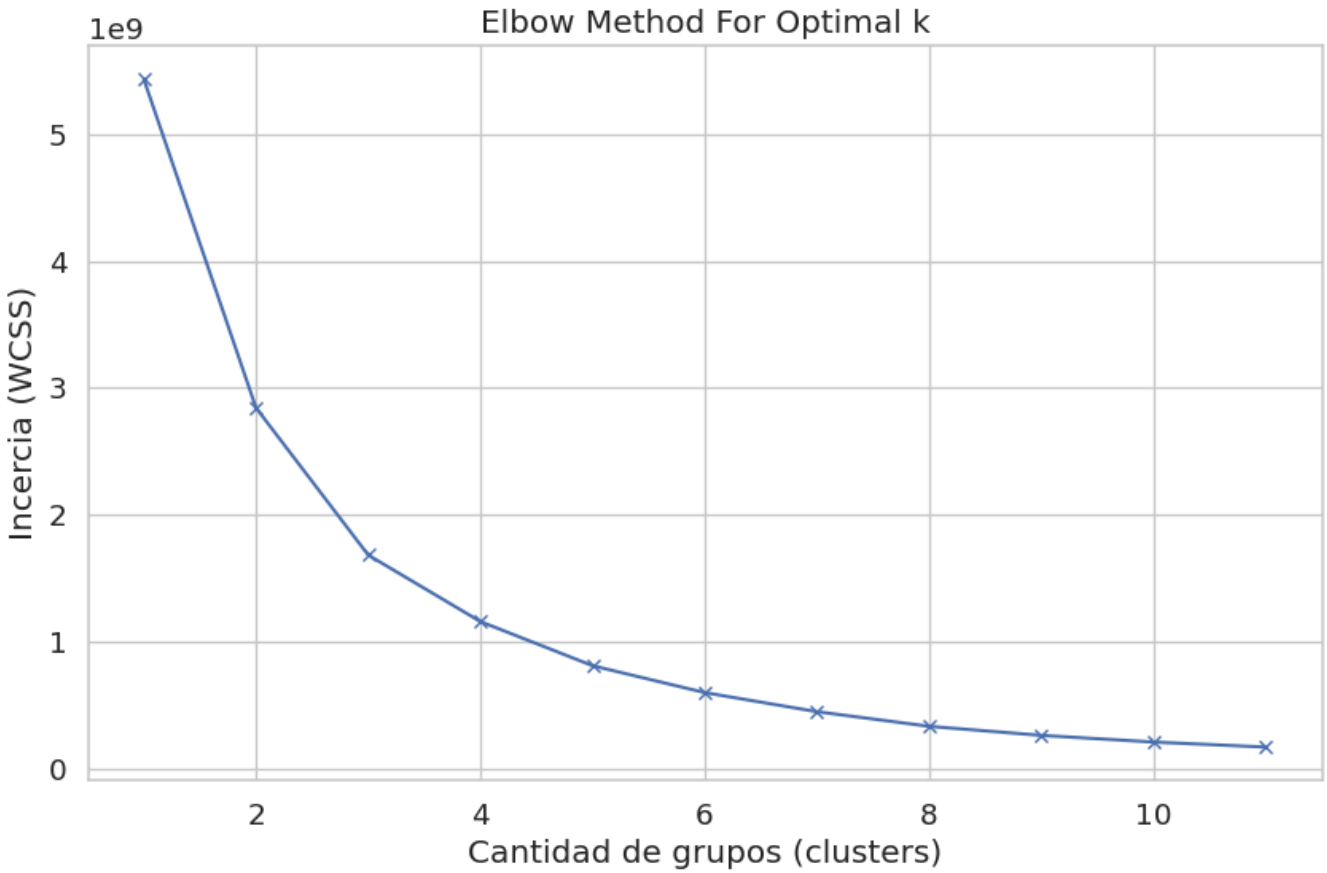
Modelo 1: Kmeans

In [ ]:
```
from sklearn.cluster import KMeans
```

In [ ]:
```
K = range(1,12)
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitudT
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(vuln_df[['Vuln_Score','POI
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
```

```
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()
```

```
1 2 3 4 5 6 7 8 9 10 11
```



In [ ]:
```
kmeans = KMeans(n_clusters=3, random_state=1)
y_kmeans = kmeans.fit_predict(vuln_df[['Vuln_Score','POBTOT']])
```

In [ ]:
```
centros=kmeans.cluster_centers_
for i in range(0,3):
  print('Grupo',i,'Centro:',centros[i-1][0].round(2),)
```

```
Grupo 0 Centro: 7.9
Grupo 1 Centro: 6.75
Grupo 2 Centro: 7.63
```

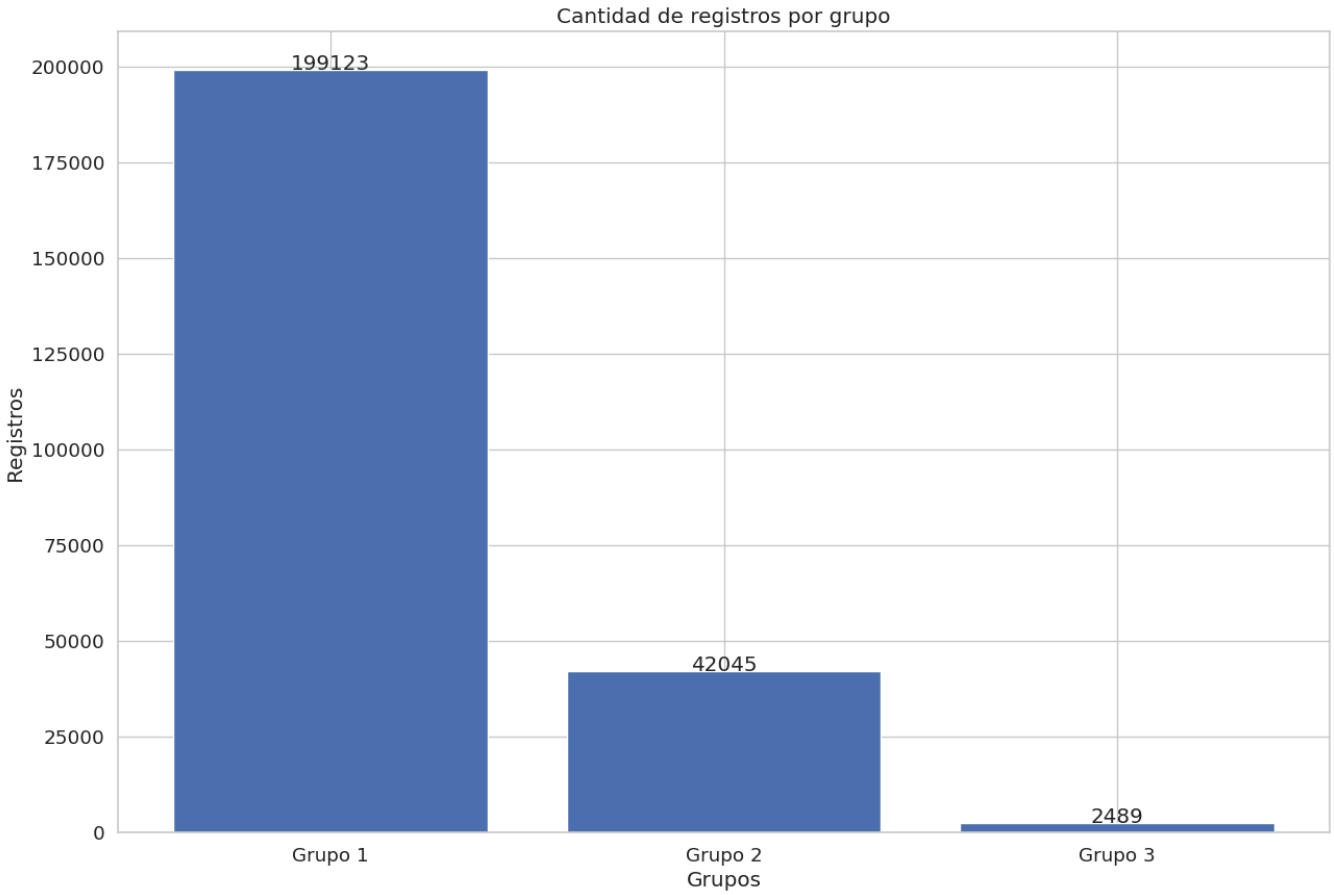
In [ ]:
```
clusterCount=np.bincount(y_kmeans) #np.bincount(kmeans.labels_)
for i in range(0,3):
  print('Grupo ',i,', registros del grupo:',clusterCount[i-1])
```

```
Grupo  0 , registros del grupo: 2489
Grupo  1 , registros del grupo: 199123
Grupo  2 , registros del grupo: 42045
```

In [ ]:
```
#Se comienza en cero para evitar un problema con los índices de Python
for i in range(0,3):
  exec("c" + str(i) + '=kmeans.cluster_centers_[i]')
```

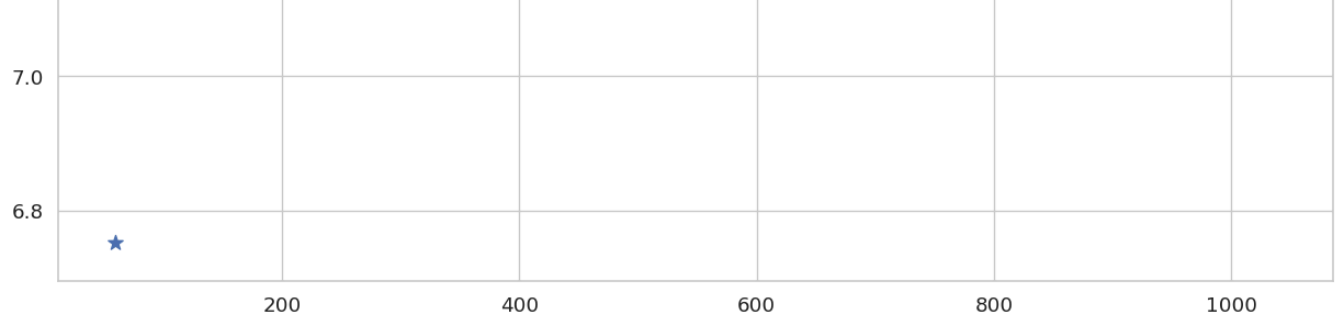
In [ ]:
```
grupos=['Grupo 1','Grupo 2','Grupo 3']
```

```
plt.bar(grupos,clusterCount)
plt.xlabel('Grupos')
plt.ylabel('Registros')
plt.title('Cantidad de registros por grupo')
for i in range(len(clusterCount)):
        plt.text(i, clusterCount[i], clusterCount[i], ha = 'center')
```



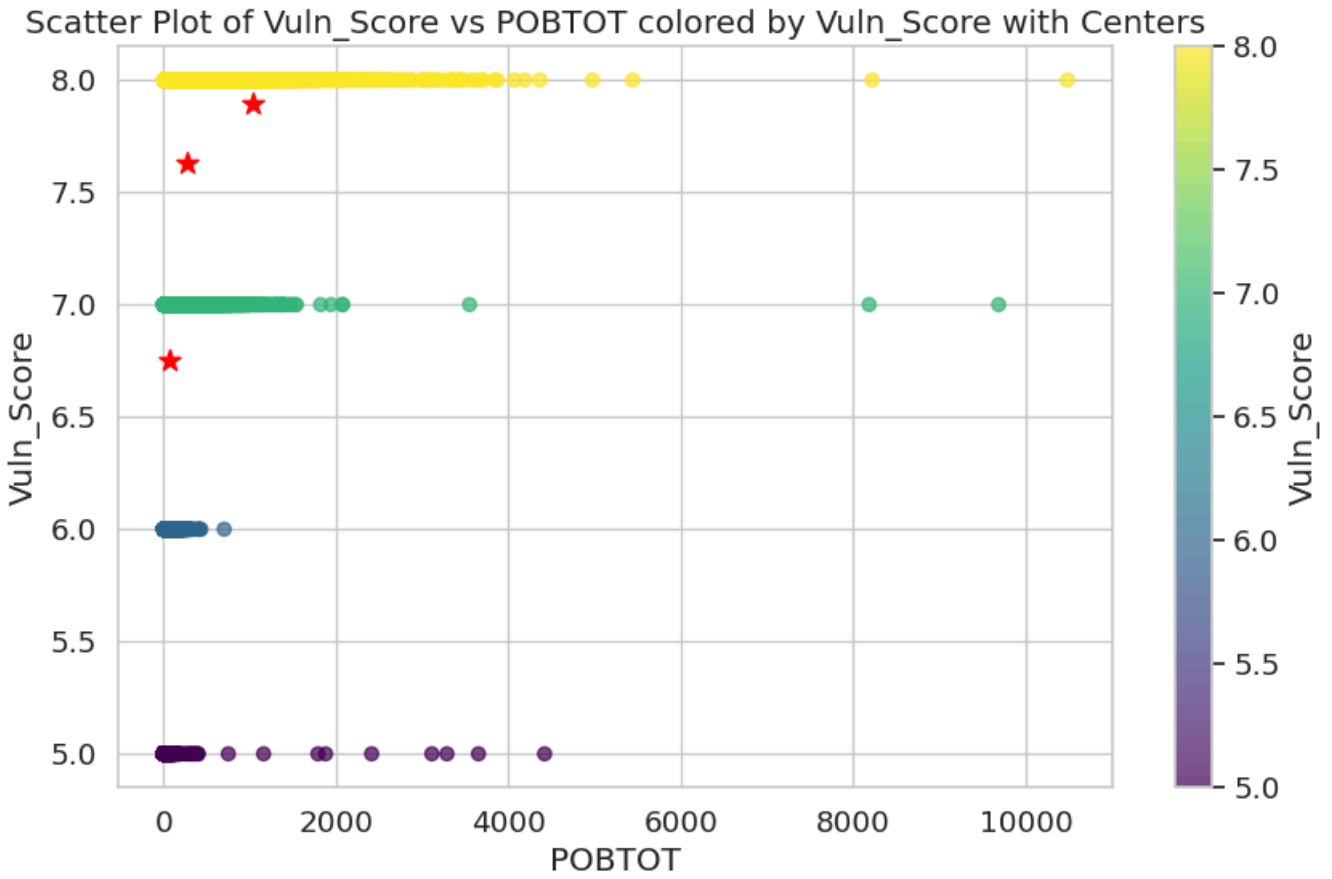
In [ ]:

```
#Creamos una lista que contiene todas las variables correspondientes a los centros
varsCentros=[c0,c1,c2]

#Recorremos la lista anterior para graficar los centros en función de la longitud y latitu
for i in range(0,3):
  plt.scatter(varsCentros[i][1], varsCentros[i][0],marker='*',s=100)
  plt.title('Centros')
```



In [ ]:

```
plt.figure(figsize=(10, 6))
sc = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=vuln_df['Vuln_Score'], cmap='

plt.colorbar(sc, label='Vuln_Score')

varsCentros = [c0, c1, c2]

for i in range(3):
    plt.scatter(varsCentros[i][1], varsCentros[i][0], marker='*', s=100, color='red', label

plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Scatter Plot of Vuln_Score vs POBTOT colored by Vuln_Score with Centers')

plt.show()
```



Modelo 2: DBSCAN

El algoritmo Density-Based Spatial Clustering of Applications with Noise (DBSCAN) está orientado en agrupamiento, el cual está basado en la densidad de puntos en el espacio, como su mismo nombre lo indica.

En cuanto a su funcionamiento, primeramente elige un punto aleatorio, del cual observará la cantidad dentro de una distancia determinada, cuántos puntos adicionales se encuentran. Se establecerá como punto central si, a partir de dicho punto y en la distancia determinada, se tiene una cantidad mínima de puntos; en su defecto, se considerará como ruido.

Finalmente, si se encontró el punto central, se procede con la expansión del cluster, buscando todos los puntos dentro de la distancia. Lo anterior se repetirá para todos los puntos que no han sido analizado hasta formar los clústeres.

In [ ]:
```
from numpy import unique
from numpy import where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import DBSCAN
```

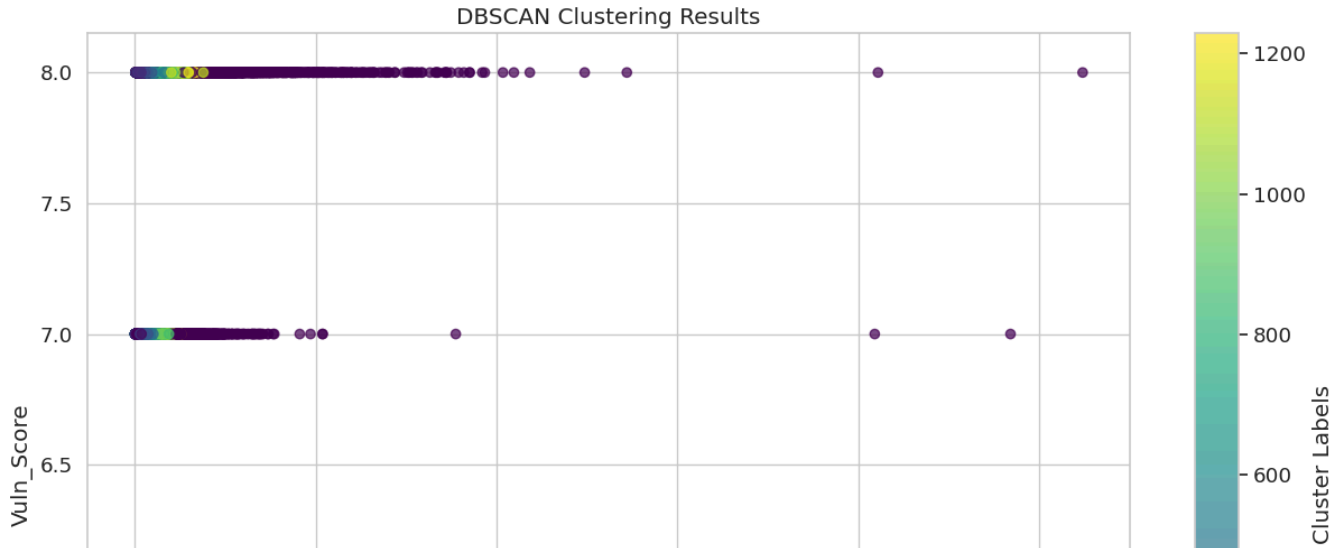
In [ ]:
```
dbscan_model = DBSCAN(eps=0.25, min_samples=9)
dbscan_model.fit(vuln_df[['Vuln_Score','POBTOT']],)
```

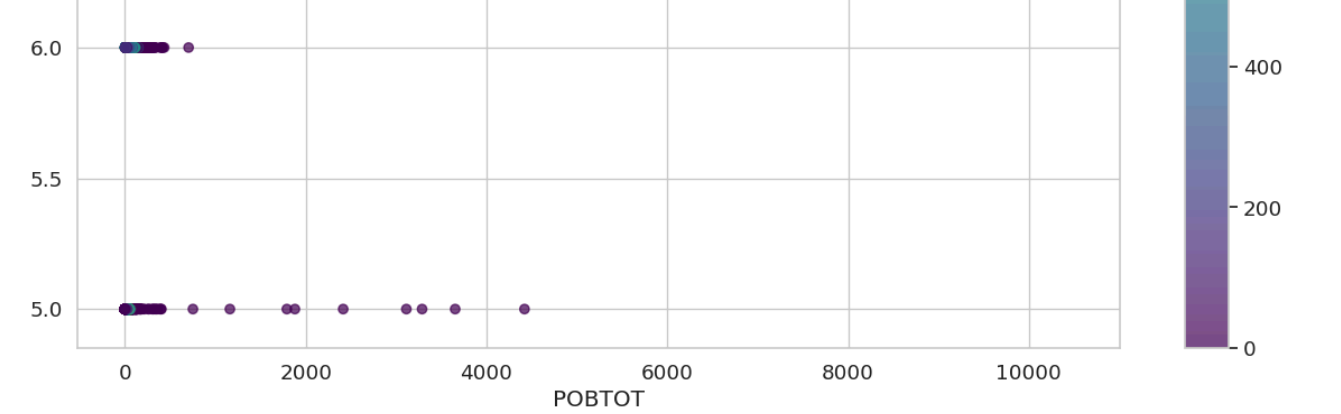
Out[ ]:
```
DBSCAN(eps=0.25, min_samples=9)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
dbscan_result = dbscan_model.fit_predict(vuln_df[['Vuln_Score','POBTOT']])
dbscan_clusters = unique(dbscan_result)
print(dbscan_clusters)
```

```
[  -1    0    1 ... 1227 1228 1229]
```

In [ ]:
```
plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=dbscan_result, cmap='viridis', alpl
plt.colorbar(label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('DBSCAN Clustering Results')

plt.show()
```

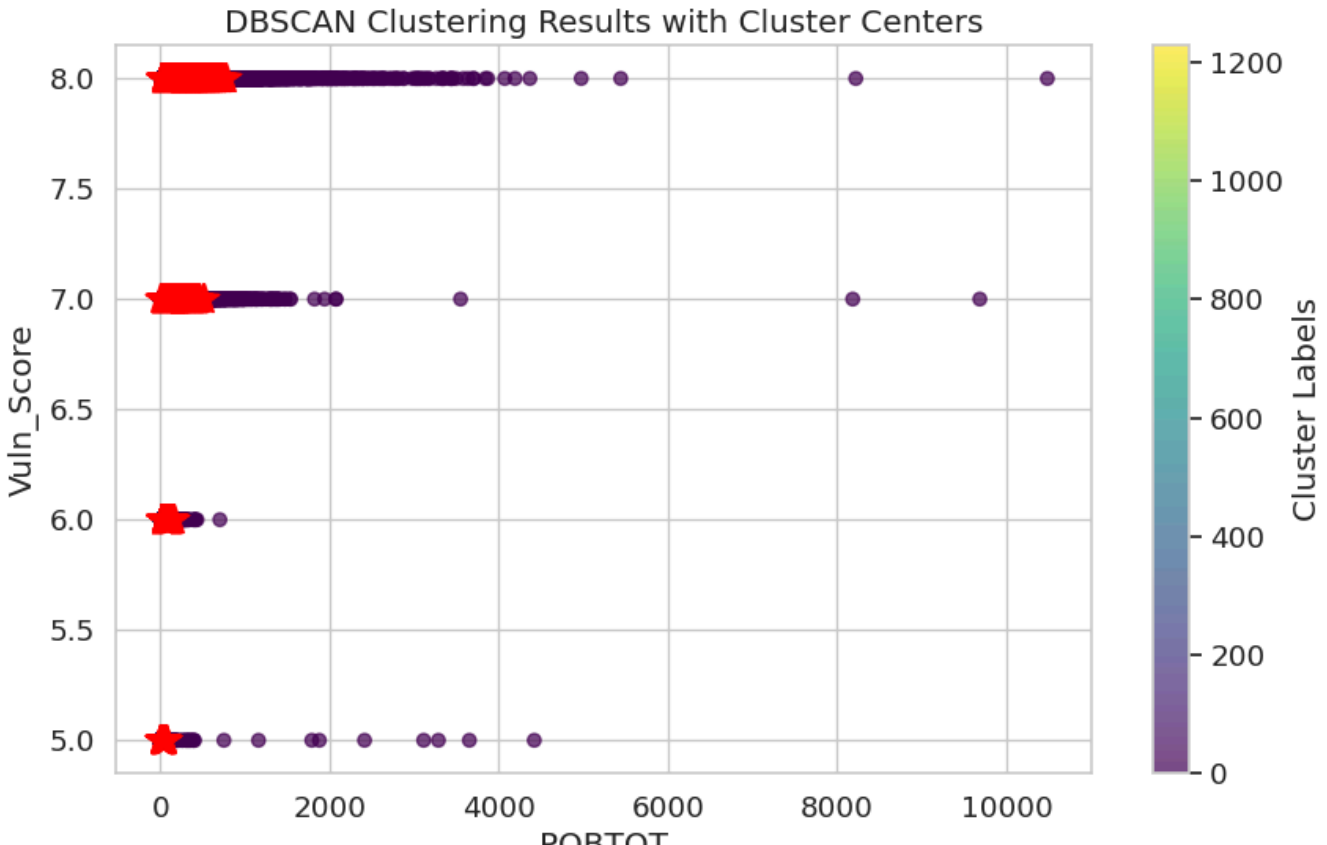




In [ ]:

```
dbscan_clusters = np.unique(dbscan_result)
print("Unique clusters found:", dbscan_clusters)

plt.figure(figsize=(10, 6))

scatter = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=dbscan_result, cmap='vir

for cluster in dbscan_clusters:
    if cluster != -1:
        cluster_points = vuln_df[dbscan_result == cluster]
        center_x = cluster_points['POBTOT'].mean()
        center_y = cluster_points['Vuln_Score'].mean()
        plt.scatter(center_x, center_y, marker='*', s=200, color='red', label=f'Cluster {c

plt.colorbar(scatter, label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('DBSCAN Clustering Results with Cluster Centers')

plt.show()
```

```
Unique clusters found: [  -1    0    1 ... 1227 1228 1229]
```



POBTOT

Modelo 3: BIRCH

El algoritmo Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) está también orientado a la agrupación, sobre todo para conjunto grandes de datos. Se enfoca en crear grupos con clústeres jerárquicos compactos, de forma que también es posible reducir la representación por una versión más condensada.

Para entender el modelo, es importante entender el CF Tree (Clustering Feature Tree), la cual es una estructura jerárquica y compacta de los datos, donde se tendrán nodos del árbol, el cual albergará un resumen, incluyendo la cantidad de puntos dentro del cluster, la suma de las coordenadas de cada punto y la suma de los cuadrados de todos los puntos. Así pues, el análisis se realiza de manera incremental y se almacena en el CF Tree, construyendo el árbol desde las raíces hasta la copa (abajo a arriba), agrupando a su vez los puntos en grupos, de acuerdo con el umbral de dispersión y número máximo de nodos hijo. Adicionalmente, se pueden trabajar algunas fases de condensación, clustering global y refinamiento.

In [ ]:
```
from numpy import unique
from numpy import where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import Birch
```

In [ ]:
```
birch_model = Birch(threshold=0.03, n_clusters=2)
```

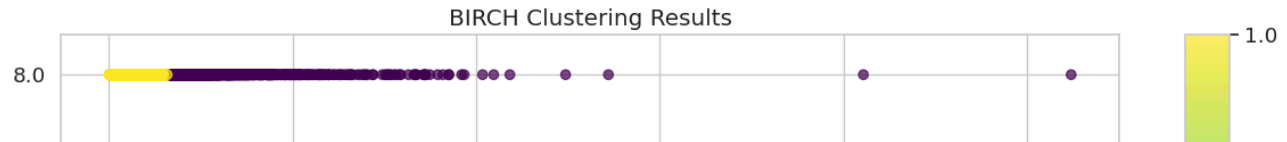
In [ ]:
```
birch_model.fit(vuln_df[['Vuln_Score','POBTOT']])
```

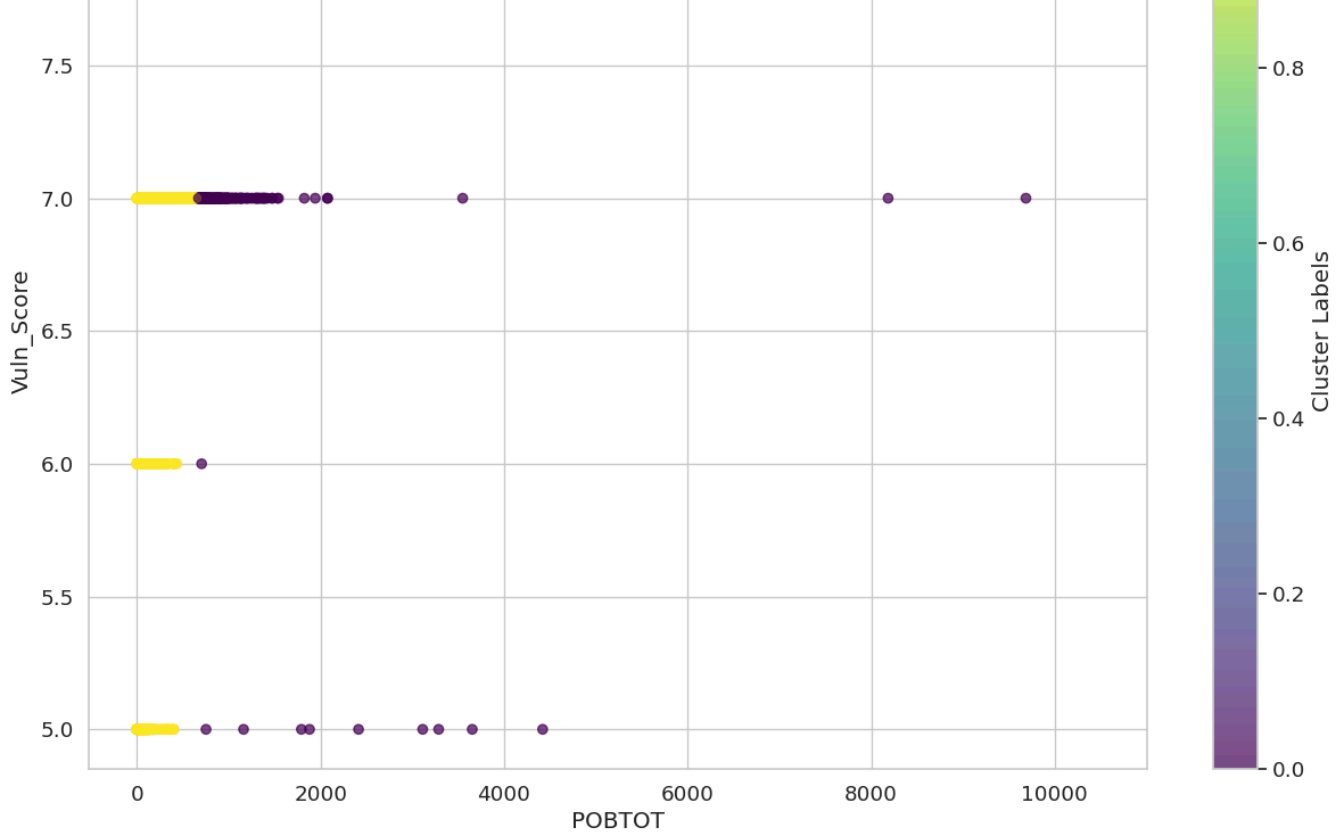
Out[ ]:
```
Birch(n_clusters=2, threshold=0.03)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
birch_result = birch_model.predict(vuln_df[['Vuln_Score','POBTOT']])
birch_clusters = unique(birch_result)
print(birch_clusters)
```

```
[0 1]
```

In [ ]:
```
plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=birch_result, cmap='viridis', alpha
plt.colorbar(label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('BIRCH Clustering Results')

plt.show()
```

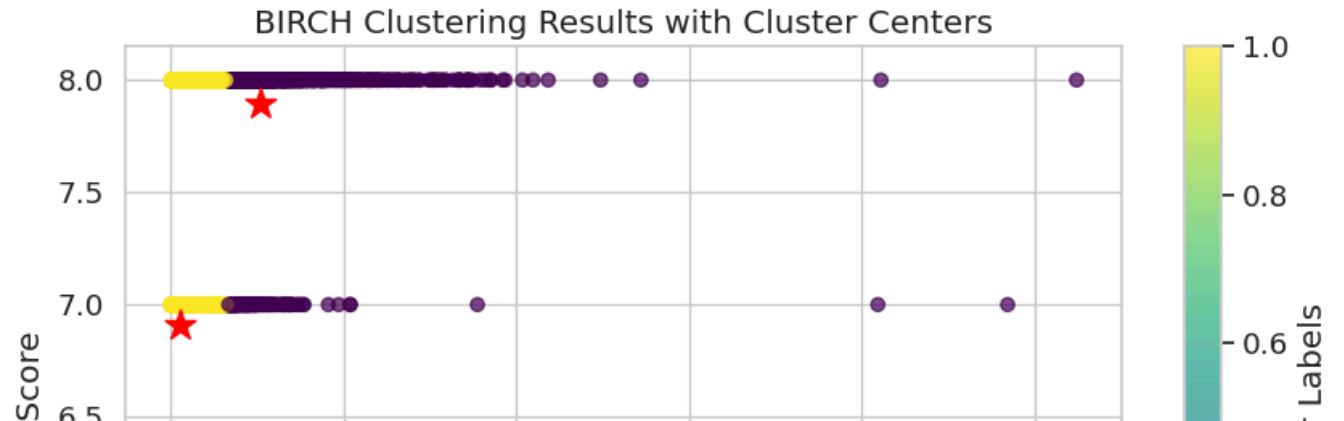


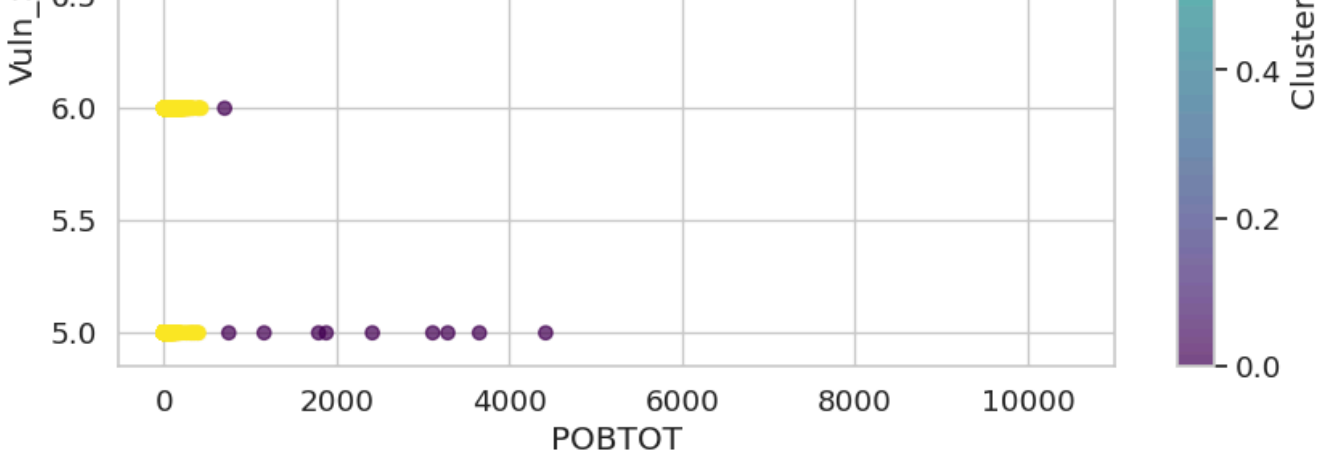


In [ ]:

```
birch_clusters = np.unique(birch_result)
print("Unique clusters found:", birch_clusters)

plt.figure(figsize=(10, 6))

scatter = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=birch_result, cmap='virid

for cluster in birch_clusters:
    if cluster != -1:
        cluster_points = vuln_df[birch_result == cluster]
        center_x = cluster_points['POBTOT'].mean()
        center_y = cluster_points['Vuln_Score'].mean()
        plt.scatter(center_x, center_y, marker='*', s=200, color='red', label=f'Cluster {cl

plt.colorbar(scatter, label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('BIRCH Clustering Results with Cluster Centers')

plt.show()
```

```
Unique clusters found: [0 1]
```





Modelo 4: Gaussian Mixture

El algoritmo Gaussian Mixture (GMM) posee un enfoque probabilístico que está igualmente orientado a agrupamiento, donde asumirá que los datos tienen origen de una combinación de diversas distribuciones gaussianas. Lo anterior hace muy identificable a este modelo, pues no comienza asumiendo de manera determinista si un punto corresponde a un grupo, sino en su defecto obtiene la probabilidad de pertenecer al cluster.

In [ ]:
```
from numpy import unique
from numpy import where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.mixture import GaussianMixture
```

In [ ]:
```
gaussian_model = GaussianMixture(n_components=2)
```

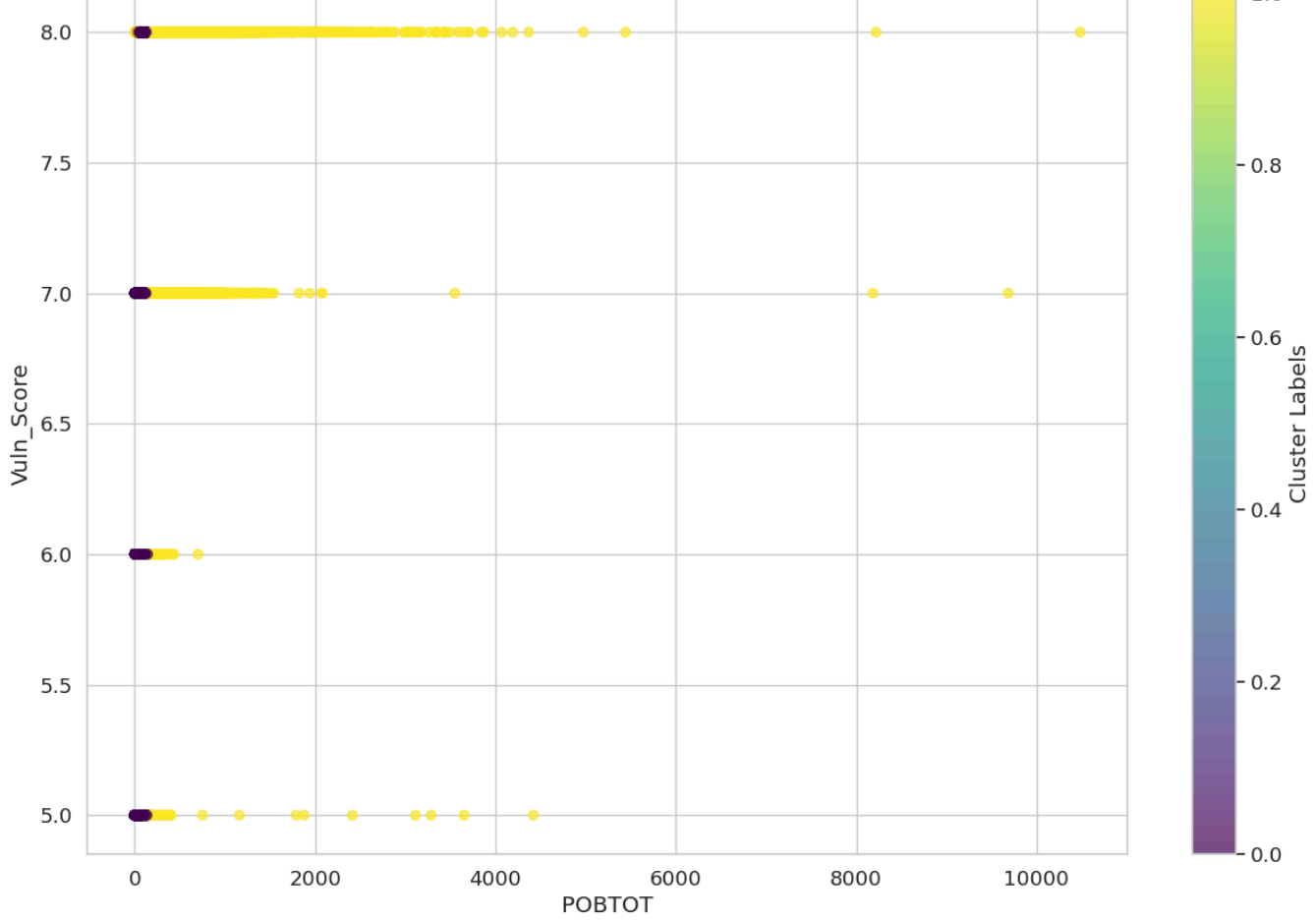
In [ ]:
```
gaussian_model.fit(vuln_df[['Vuln_Score','POBTOT']])
```

Out[ ]:
```
GaussianMixture(n_components=2)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
gaussian_result = gaussian_model.predict(vuln_df[['Vuln_Score','POBTOT']])
gaussian_clusters = unique(gaussian_result)
print(gaussian_clusters)
```

```
[0 1]
```

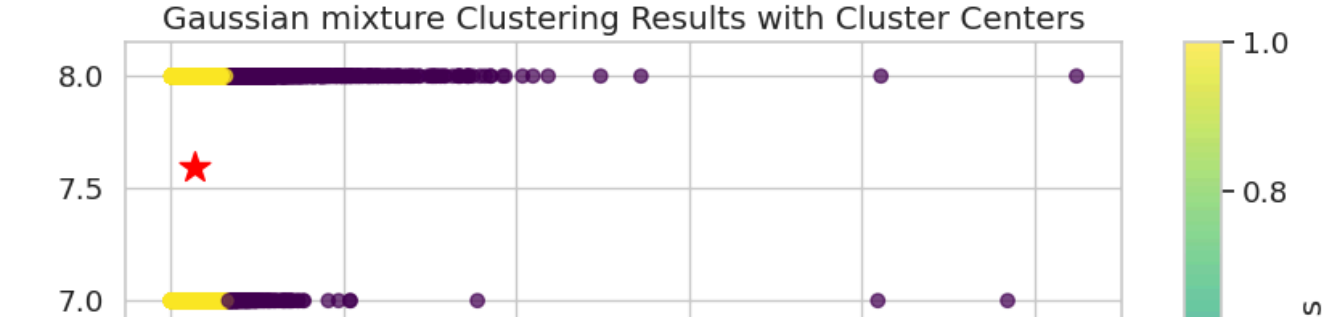
In [ ]:
```
plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=gaussian_result, cmap='viridis', a
plt.colorbar(label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Gaussian Mixture Clustering Results')

plt.show()
```

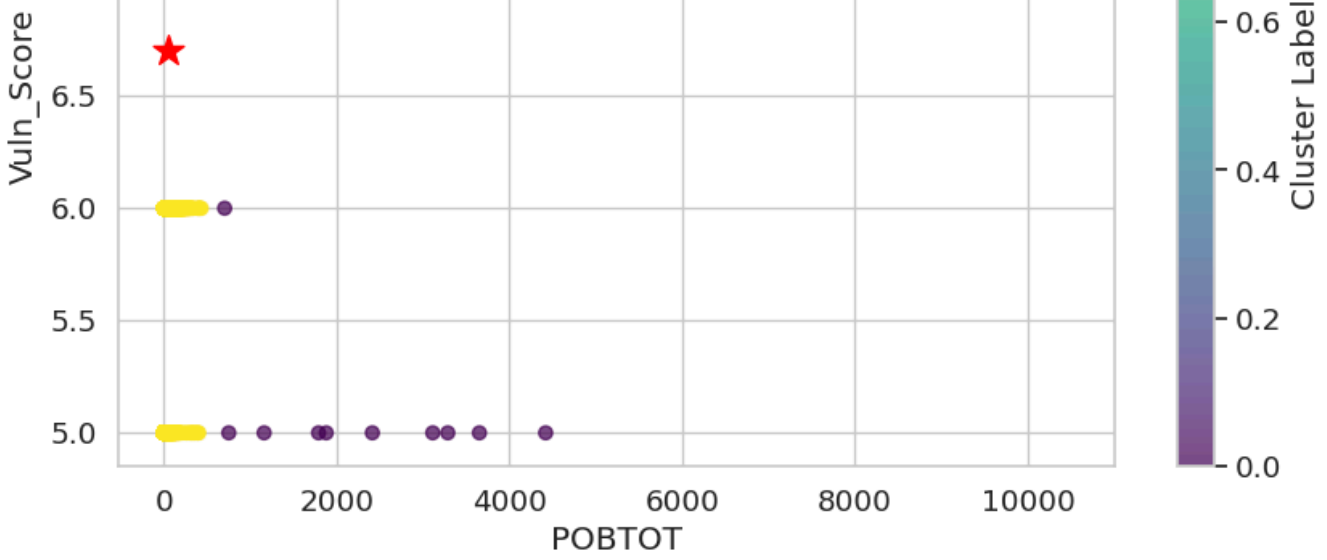


In [ ]:

```
gaussian_clusters = np.unique(birch_result)
print("Unique clusters found:", gaussian_clusters)

plt.figure(figsize=(10, 6))

scatter = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=birch_result, cmap='viri

for cluster in gaussian_clusters:
    if cluster != -1:
        cluster_points = vuln_df[gaussian_result == cluster]
        center_x = cluster_points['POBTOT'].mean()
        center_y = cluster_points['Vuln_Score'].mean()
        plt.scatter(center_x, center_y, marker='*', s=200, color='red', label=f'Cluster {c

plt.colorbar(scatter, label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Gaussian mixture Clustering Results with Cluster Centers')

plt.show()
```

```
Unique clusters found: [0 1]
```





Modelo 5: Validación cruzada

La validación cruzada es una técnica utilizada en el ámbito del aprendizaje automático para evaluar la capacidad de generalización de un modelo. En lugar de entrenar y evaluar el modelo en el mismo conjunto de datos, la validación cruzada divide los datos en múltiples subconjuntos llamados "folds" y realiza múltiples ciclos de entrenamiento y evaluación.

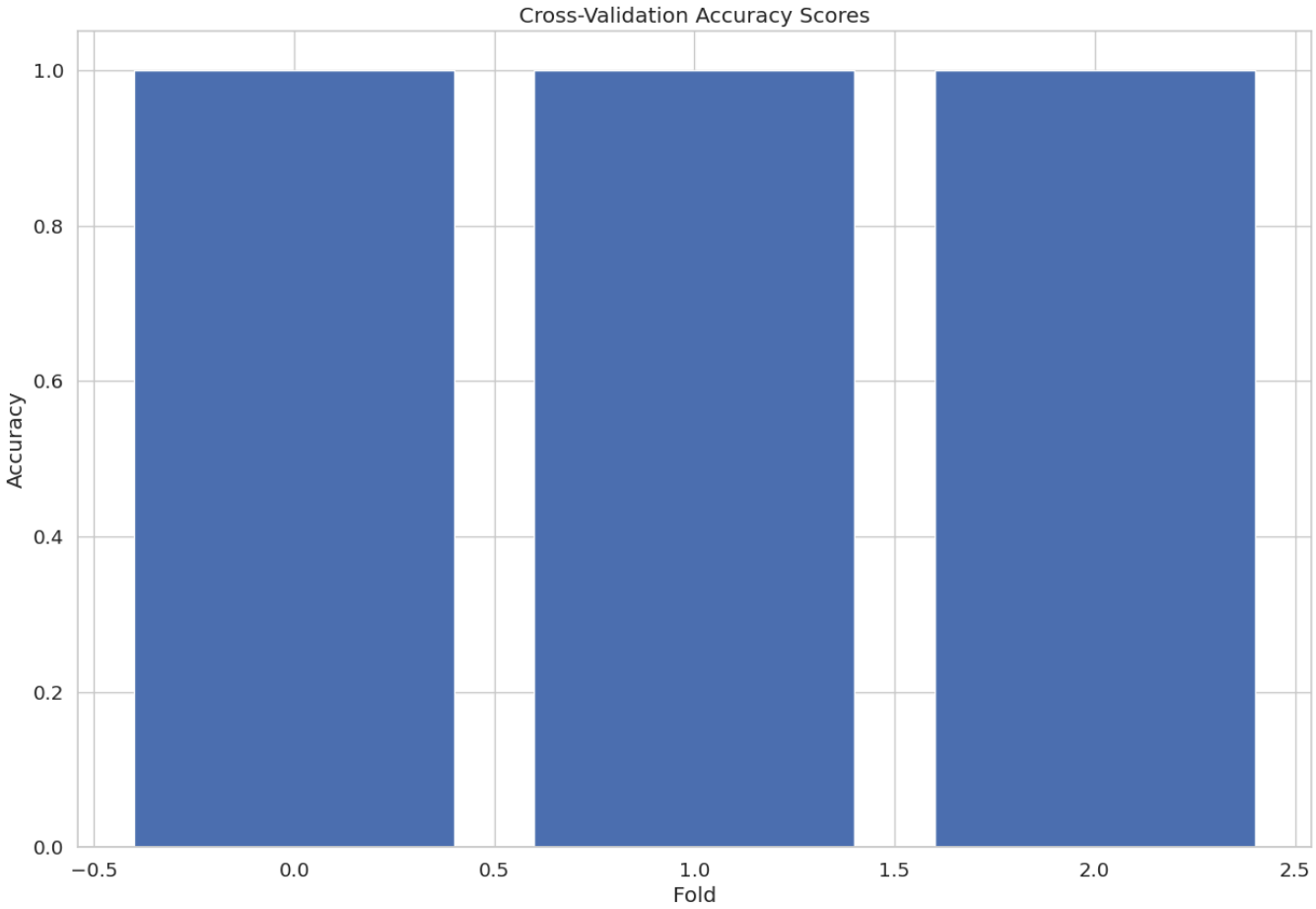
In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
# Cross-validation
cv = 3
scoring = 'accuracy'

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('pca', PCA()),
    ('model', LogisticRegression())
])

# Perform cross-validation
scores = cross_val_score(pipeline, vuln_df, y= vuln_df['Vuln_Score'], cv=cv, scoring=scori
# Cross-validation scores
print(f"Cross-validation scores: {scores}")
print(f"Mean score: {scores.mean()}")
```

```
Cross-validation scores: [0.99975375 0.99993844 0.99993844]
Mean score: 0.9998768761004199
```

In [ ]:
```
plt.bar(range(cv), scores)

# Add labels and title
plt.xlabel("Fold")
plt.ylabel("Accuracy")
plt.title("Cross-Validation Accuracy Scores")

# Show the plot
plt.show()
```

```
plt.show()
```



Supervisados

Modelo 6: Análisis discriminante

El análisis discriminante es una técnica estadística que se utiliza en el campo de la inteligencia artificial (IA) para clasificar observaciones en grupos predefinidos. En otras palabras, busca encontrar las mejores características que permitan distinguir entre diferentes categorías. Permite construir modelos de clasificación altamente precisos, lo que es esencial en muchas aplicaciones de la IA, como el reconocimiento de patrones, el diagnóstico médico y la segmentación de clientes. Ayuda a identificar las características más importantes para la clasificación, lo que reduce la dimensionalidad de los datos y mejora la eficiencia de los modelos.

In [ ]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score
```

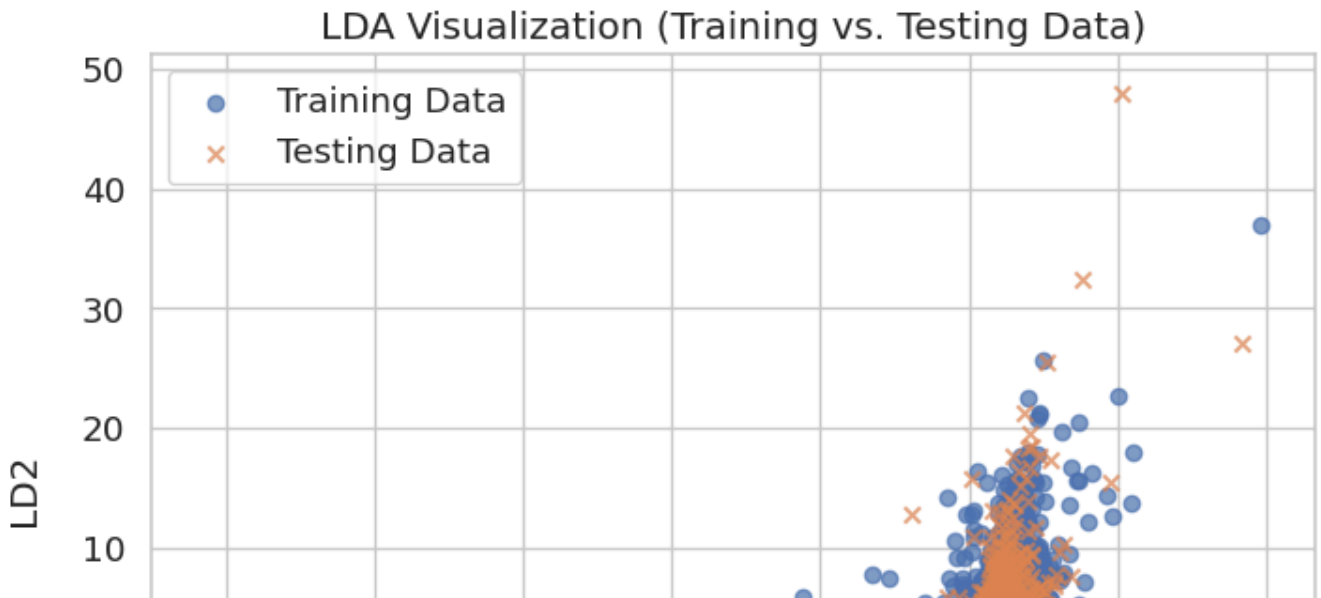
In [ ]:
```
vuln_df_copy = vuln_df.copy()

#Main y variable and dataframe
X = vuln_df_copy
y = vuln_df_copy['Vuln_Score']

# Train Test Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

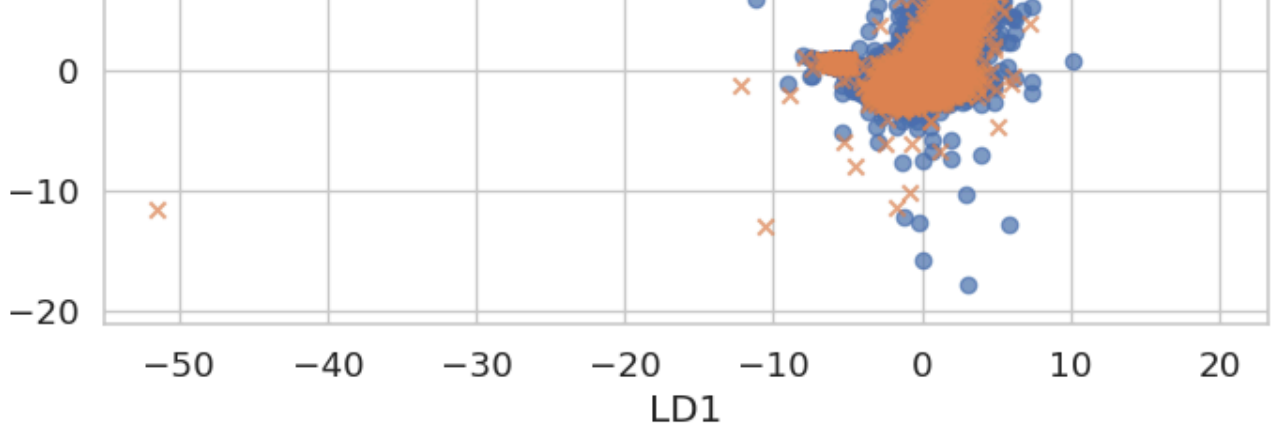
```
# LDA model
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
y_pred = lda.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

```
Accuracy: 0.7045
```

In [ ]:

```
lda = LinearDiscriminantAnalysis(n_components=2)
lda.fit(X_train, y_train)


X_train_transformed = lda.transform(X_train)
X_test_transformed = lda.transform(X_test)


train_df = pd.DataFrame(X_train_transformed, columns=['LD1', 'LD2'])
train_df['target'] = y_train

test_df = pd.DataFrame(X_test_transformed, columns=['LD1', 'LD2'])
test_df['target'] = y_test

# Create the scatter plot
plt.figure(figsize=(8, 6))

# Plot the training data with different markers
plt.scatter(train_df['LD1'], train_df['LD2'], label='Training Data', marker='o', alpha=0.7

# Plot the testing data with different markers
plt.scatter(test_df['LD1'], test_df['LD2'], label='Testing Data', marker='x', alpha=0.7)

# Add labels and title
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.title('LDA Visualization (Training vs. Testing Data)')
plt.legend()
plt.grid(True)

plt.show()
```





Parte 14. Comparativa entre modelos

A continuación, incluimos los resultados de los modelos presentados anteriormente:

In [ ]:
```
from sklearn.metrics import calinski_harabasz_score

#DBSCAN
print(f'Calinski-Harabasz Score: {calinski_harabasz_score(X, dbscan_result )}')

#Birch
print(f'Calinski-Harabasz Score: {calinski_harabasz_score(X, birch_result)}')

#Mezcla Gausianna
print(f'Calinski-Harabasz Score: {calinski_harabasz_score(X, gaussian_result)}')
```

```
Calinski-Harabasz Score: 18.818915703459613
Calinski-Harabasz Score: 10182.940961537983
Calinski-Harabasz Score: 9765.803480345672
```

K-means (2 a 15 clústeres):

In [ ]:
```
kmeans.inertia_
```

Out[ ]:
```
122948907.99234214
```

```
El Índice de Calinski-Harabasz es: 13
```

Validación cruzada:

```
Cross-validation scores: [0.99975375 0.99993844 0.99993844]
Mean score: 0.9998768761004199
```

Análisis discriminante:

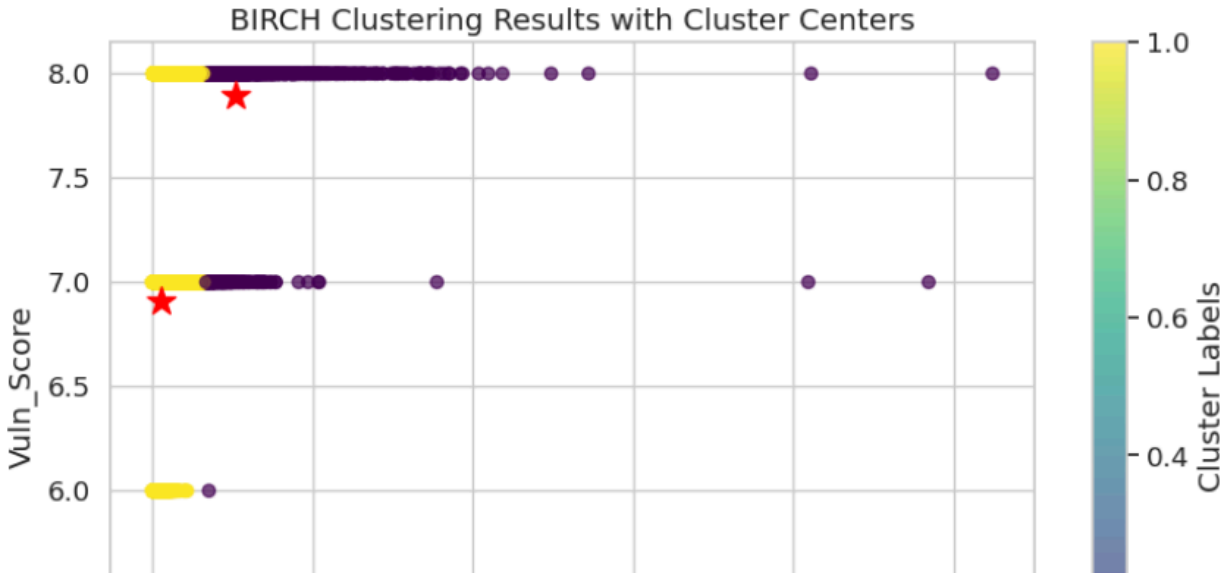
```
Accuracy: 0.7045
```

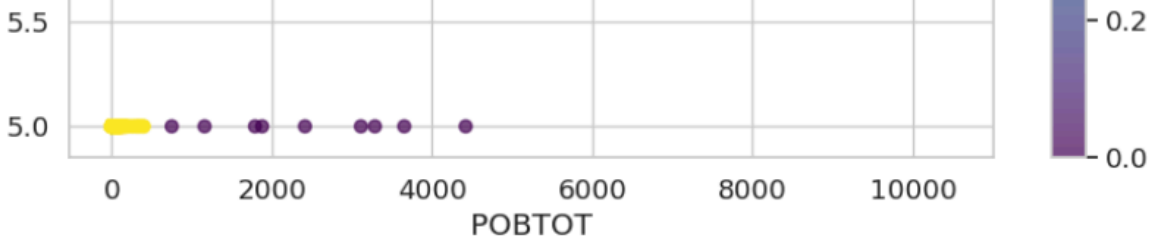
A continuación, presentamos la tabla con los datos ordenados por resultado.

Modelos	Métrica	Tiempo de	Hiperparámetros	Resultados

Modelos	Métrica	entrenamiento	Hiperparámetros	Resultados
K-means	Calinski-Harabasz	No supervisado	Cantidad de clústeres, método de iniciación, número máximo de iteraciones, tolerancia, número de iniciaciones, método de cálculo de distancia y módulo de ejecución.	13
DBSCAN	Calinski-Harabasz	No supervisado	Épsilon, mínimo de muestras, métrica de distancia, algoritmo de búsqueda, tamaño de la hoja, y parámetros de distancia.	18
Gaussian Mixture	Calinski-Harabasz	No supervisado	Componentes, varianza, inicialización de parámetros, cantidad de repeticiones, iteraciones, tolerancia y aleatoriedad.	9,765.80
BIRCH	Calinski-Harabasz	No supervisado	Número máximo de hijos (factor de ramificación), umbral o threshold, número de clústers, etiqueta de datos, copia de los datos y afinidad.	10,182.94
Análisis Discriminante	Accuracy	Supervisado	Probabilidad a priori, regularización de la matriz de covarianza, método para calcular el modelo, umbral de tolerancia y forma de la matriz de covarianza.	70.45%
Validación Cruzada	Media	No supervisado	Número de folds o conjuntos, aleatorización, métrica utilizada, retorno de puntuaciones.	99.98%

Dados la representación gráfica del modelo BIRCH y que las métricas no son tan significativas con un modelo no supervisado, ya que no se cuenta con comparabilidad histórica, también consideramos el resultado de dicho modelo.





Parte 15. Ajuste fino

Modelo 1: BIRCH

Búsqueda de los mejores parámetros

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.cluster import Birch
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import silhouette_score
```

In [ ]:
```
X = vuln_df[['Vuln_Score', 'POBTOT']]
birch_model = Birch()
```

In [ ]:
```
# Create a pipeline
pipeline = Pipeline([
    ('birch', birch_model)])

# Define the parameter grid for tuning
param_grid = {
    'birch__threshold': [0.01, 0.02, 0.03, 0.04, 0.05, 0.06],
    'birch__n_clusters': [2, 3, 4, 5, 6]}

# Define a scoring function for evaluation
def calinski_harabasz_scorer(estimator, X):
    # Fit the estimator
    labels = estimator.fit_predict(X)
    # Calculate the silhouette score
    return calinski_harabasz_score(X, labels)

# Set up the GridSearchCV
grid_search = GridSearchCV(
    pipeline,
    param_grid,
    scoring=calinski_harabasz_scorer,
    cv=5,  # Number of cross-validation folds
    n_jobs=-1)  # Use all available cores
```

In [ ]:
```
# Fit the GridSearchCV
grid_search.fit(X)

# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_
```

```
print("Best Parameters:", best_params)
print("Best Calinski Harabasz Score:", best_score)

# Use the best estimator to fit the model
best_birch_model = grid_search.best_estimator_
```

```
Best Parameters: {'birch__n_clusters': 6, 'birch__threshold': 0.01}
Best Calinski Harabasz Score: 46458.05659727363
```

Modelo mejorado con los parámetros encontrados

In [ ]:
```
best_birch_model.fit(X)
```

Out[ ]:
```
Pipeline(steps=[('birch', Birch(n_clusters=6, threshold=0.01))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

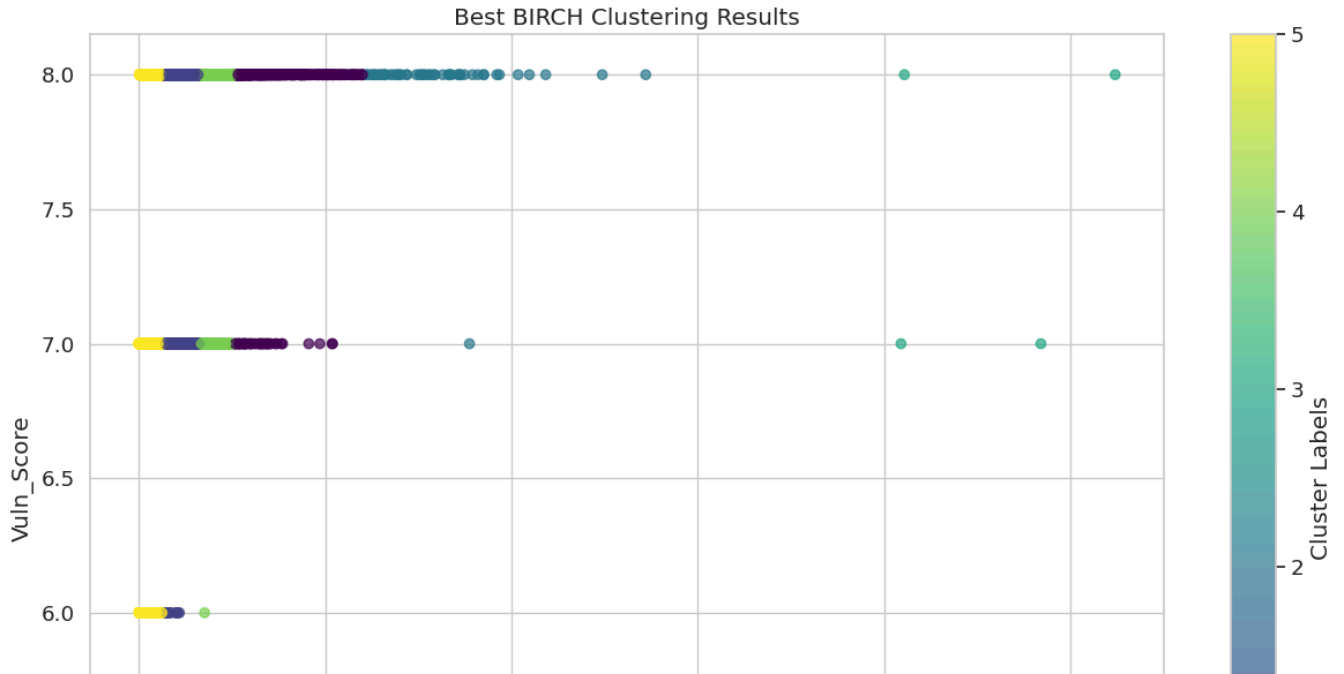
In [ ]:
```
birch_result_bm = best_birch_model.predict(X)
birch_clusters_bm = np.unique(birch_result_bm)
print(birch_clusters_bm)
```

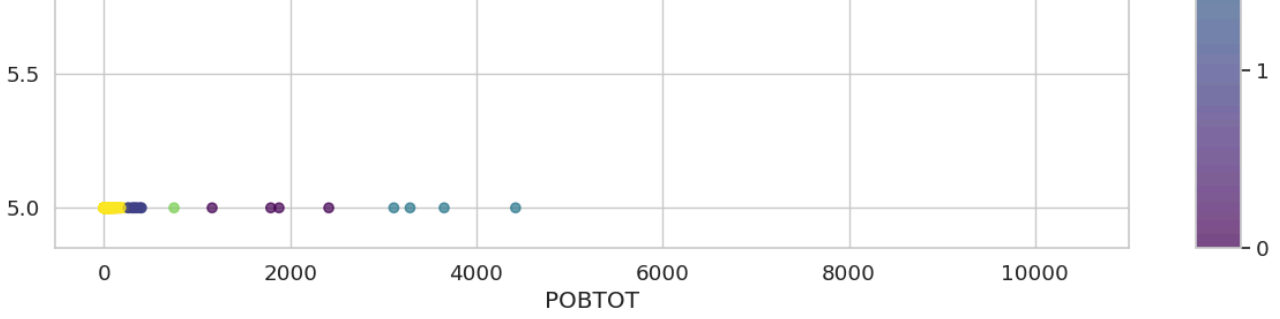
```
[0 1 2 3 4 5]
```

In [ ]:
```
#Métrica
print(f'Calinski-Harabasz Score: {calinski_harabasz_score(X, birch_result_bm)}')
```

```
Calinski-Harabasz Score: 179439.11832552517
```

In [ ]:
```
plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=birch_result_bm, cmap='viridis', a
plt.colorbar(label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Best BIRCH Clustering Results')

plt.show()
```

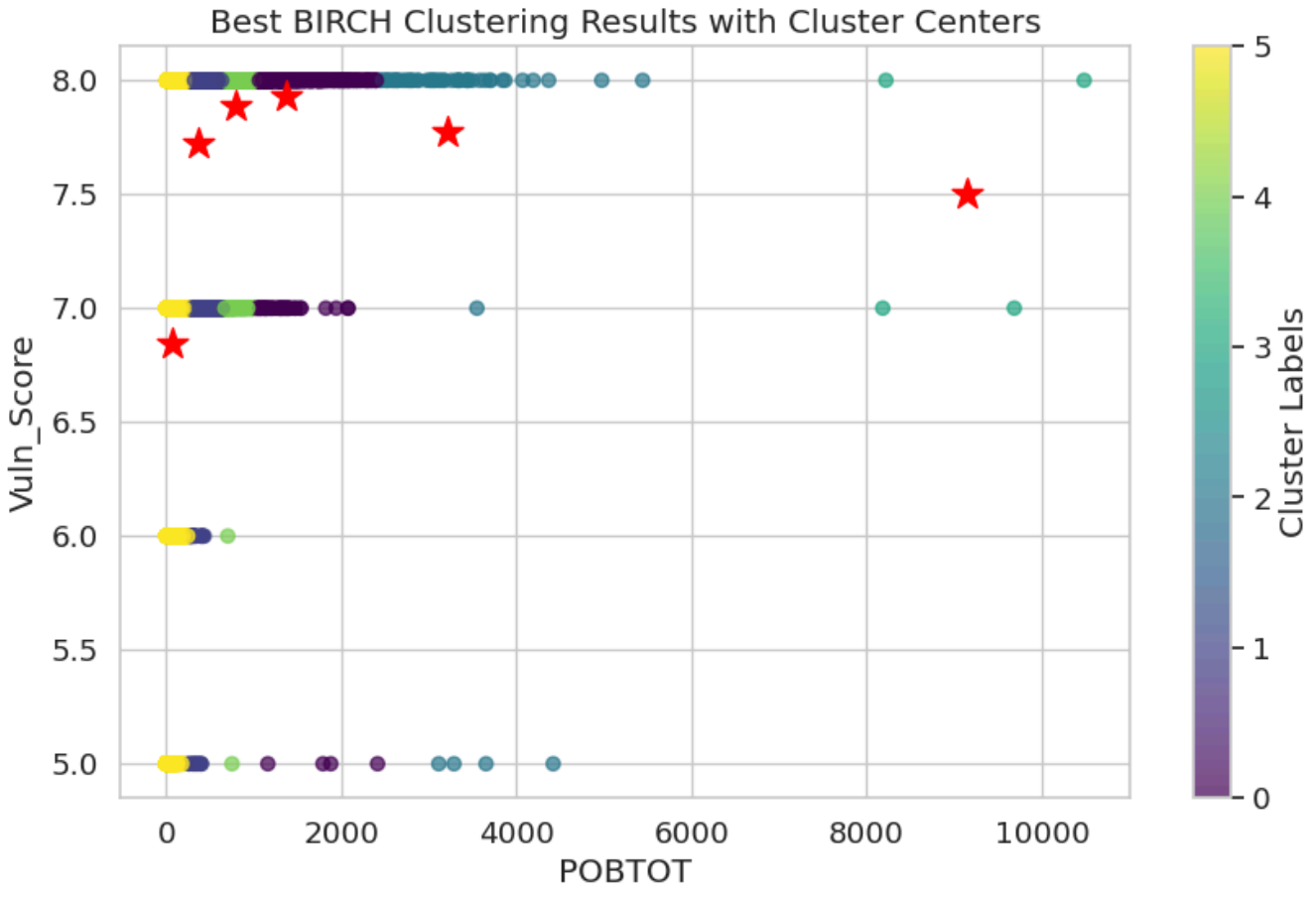




In [ ]:

```
birch_clusters_bm = np.unique(birch_result_bm)
print("Unique clusters found:", birch_clusters_bm)

plt.figure(figsize=(10, 6))

scatter = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=birch_result_bm, cmap='v

for cluster in birch_clusters_bm:
    if cluster != -1:
        cluster_points = vuln_df[birch_result_bm == cluster]
        center_x = cluster_points['POBTOT'].mean()
        center_y = cluster_points['Vuln_Score'].mean()
        plt.scatter(center_x, center_y, marker='*', s=200, color='red', label=f'Cluster {c

plt.colorbar(scatter, label='Cluster Labels')
plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Best BIRCH Clustering Results with Cluster Centers')

plt.show()
```

```
Unique clusters found: [0 1 2 3 4 5]
```



Comparación con resultado previo al ajuste:

Gráficamente podemos observar que los datos se ajustna conforme a la densidad encontrada y lógicamente, con el rango de vulnerabilidad. De forma que con esta selección de parámetros y el modelos, es posible realizar un agrupamiento más lógico en términos de concentración y pertenencia al clúster contra el rango de vulnerabilidad para la población.

Modelo 2: k-means

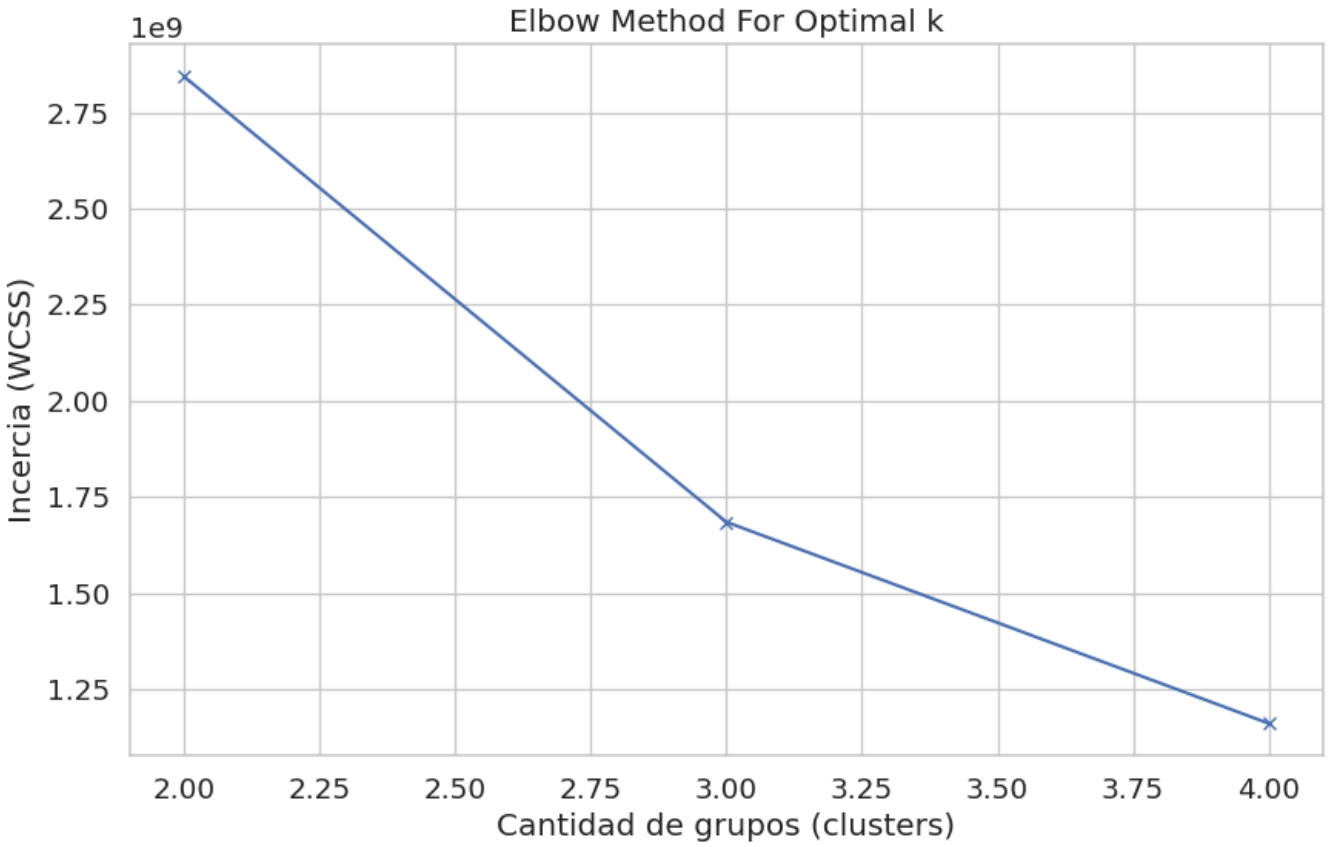
K-means Rango (2-5 clústers)

In [ ]:

```
from sklearn.metrics import calinski_harabasz_score
K = range(2,5) #Rango de 2 a 5 clusters
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitudT
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(vuln_df[['Vuln_Score','POB
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()
```

```
#Mejora
# Utilizando K-Means++ y calculando el Índice de Calinski-Harabasz
calinski_scores = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score'
    calinski_scores.append(calinski_harabasz_score(vuln_df[['Vuln_Score','POBTOT']], kmean

# Se debe seleccionar el valor de k con el mejor Índice de Calinski-Harabasz
best_k = np.argmax(calinski_scores) + 1
print("El Índice de Calinski-Harabasz es:", best_k)

# Aplicar K-Means con el mejor valor de k
kmeans = KMeans(n_clusters=best_k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score
```

2 3 4




```
El Índice de Calinski-Harabasz es: 3
```

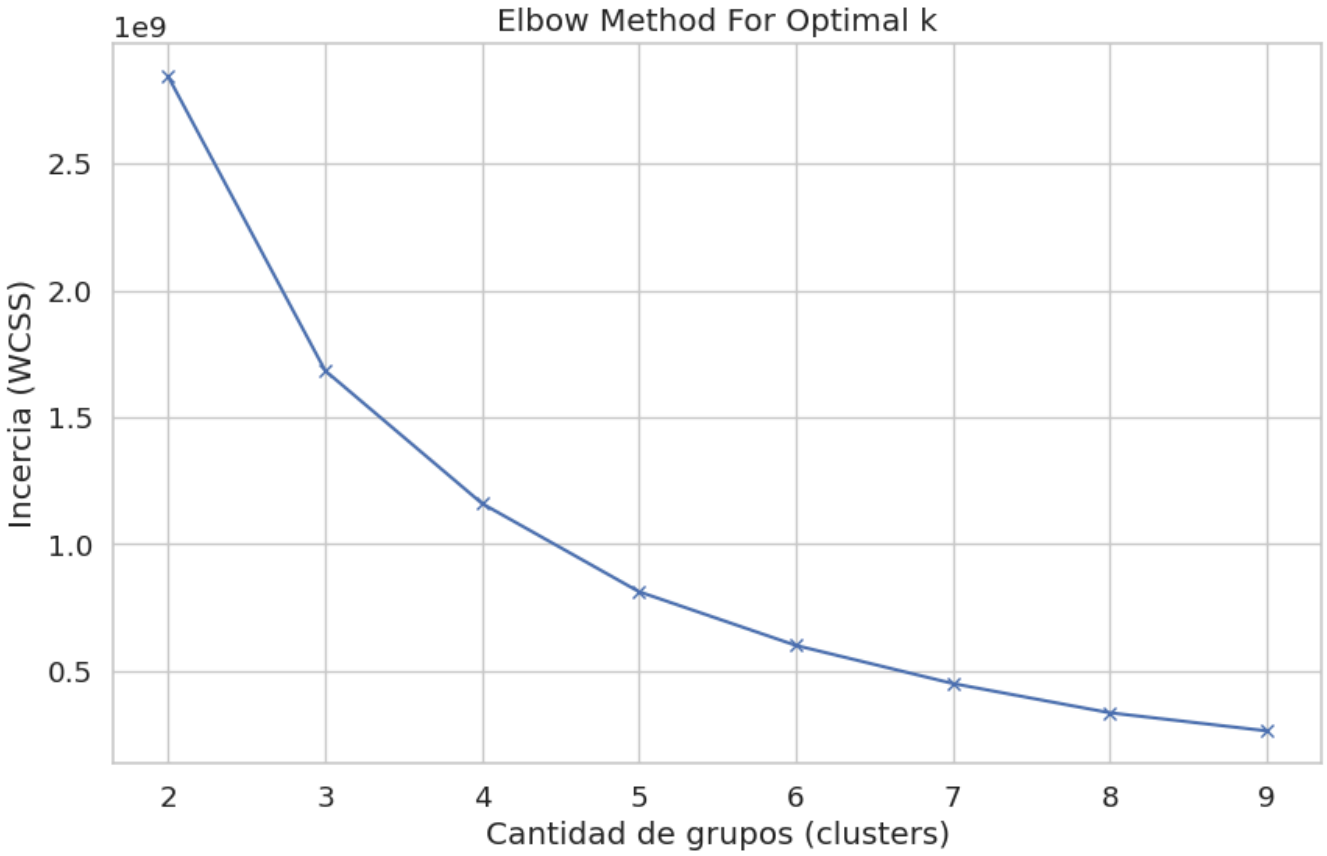
K-means Rango (2-10 clústeres)

In [ ]:

```
from sklearn.metrics import calinski_harabasz_score
K = range(2,10) #Rango de 2 a 15 clusters
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitudT
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(vuln_df[['Vuln_Score','POB
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
```

```
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()

#Mejora
# Utilizando K-Means++ y calculando el Índice de Calinski-Harabasz
calinski_scores = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score'
    calinski_scores.append(calinski_harabasz_score(vuln_df[['Vuln_Score','POBTOT']], kmean

# Se debe seleccionar el valor de k con el mejor Índice de Calinski-Harabasz
best_k = np.argmax(calinski_scores) + 1
print("El Índice de Calinski-Harabasz es:", best_k)

# Aplicar K-Means con el mejor valor de k
kmeans = KMeans(n_clusters=best_k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score
```

```
2 3 4 5 6 7 8 9
```




```
El Índice de Calinski-Harabasz es: 8
```

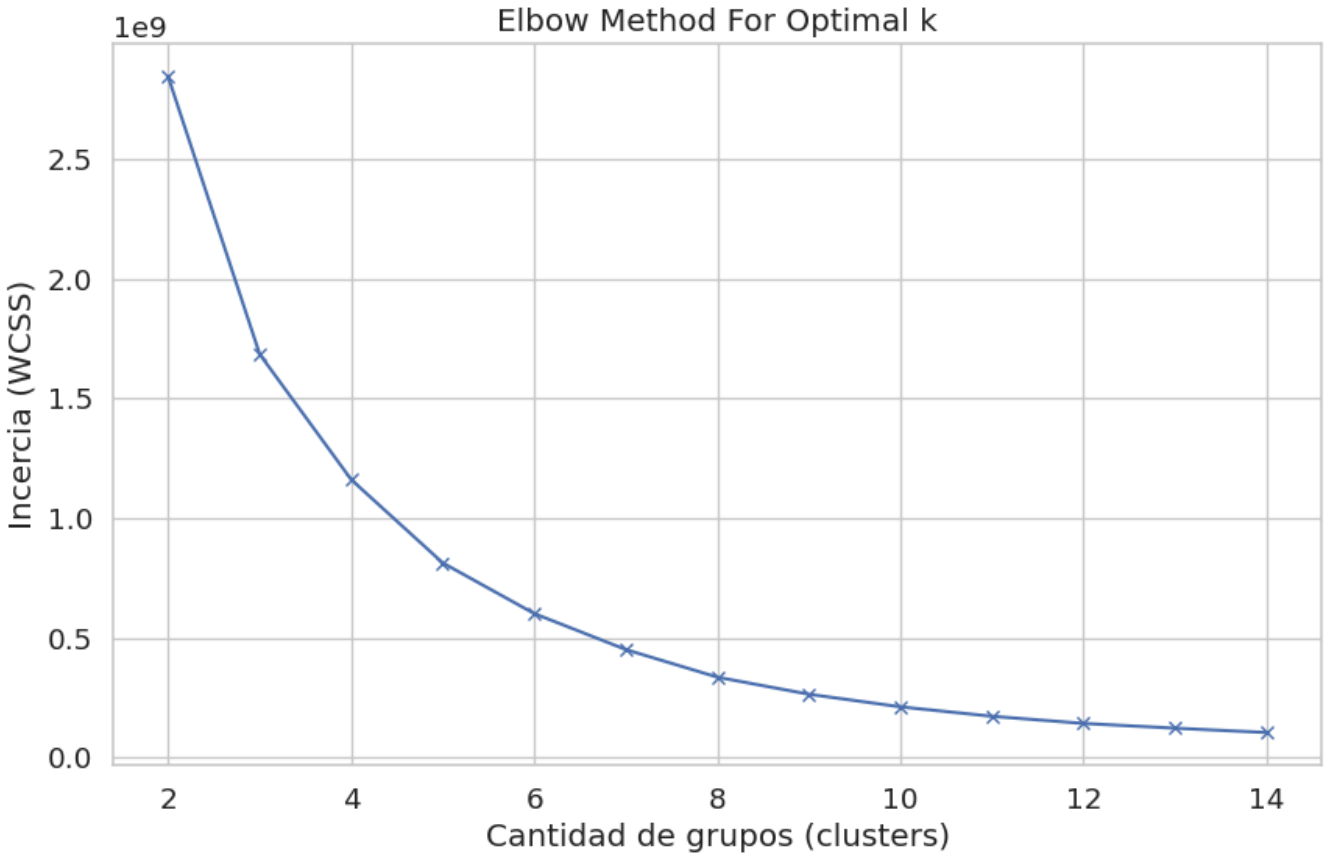
K Means Rango (2-15 clústers)

In [ ]:

```
from sklearn.metrics import calinski_harabasz_score
K = range(2,15) #Rango de 2 a 15 clusters
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitudT
```

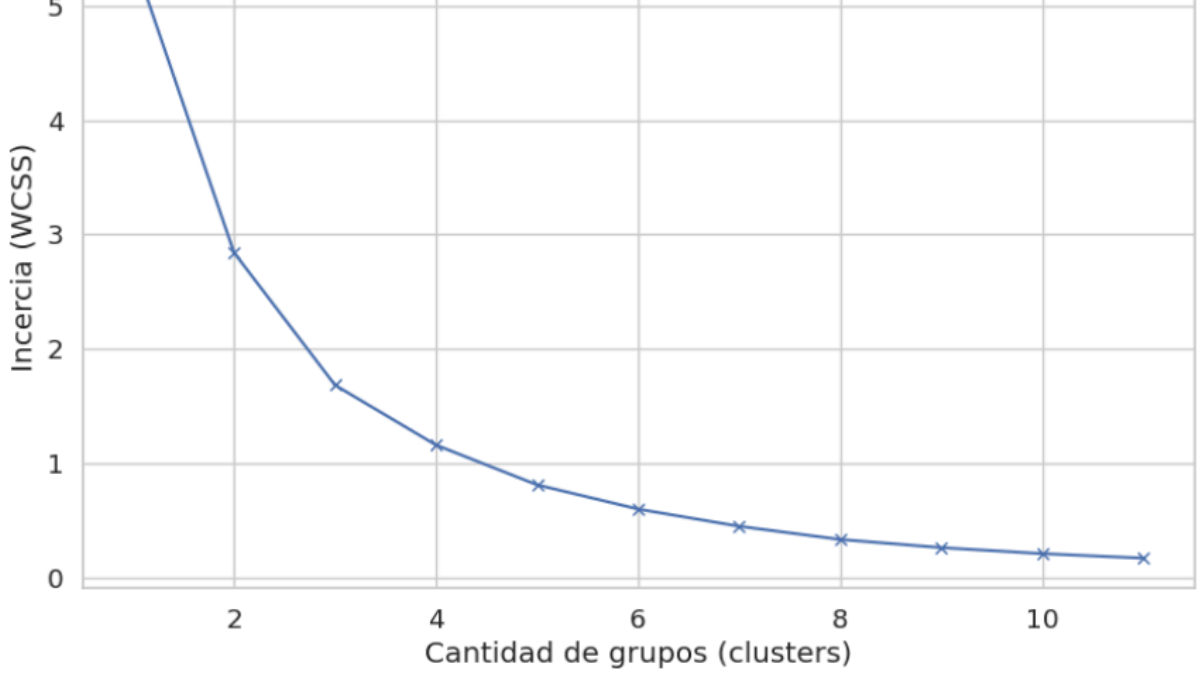
```
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(vuln_df[['Vuln_Score','POB
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()

#Mejora
# Utilizando K-Means++ y calculando el Índice de Calinski-Harabasz
calinski_scores = []
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score'
    calinski_scores.append(calinski_harabasz_score(vuln_df[['Vuln_Score','POBTOT']], kmean

# Se debe seleccionar el valor de k con el mejor Índice de Calinski-Harabasz
best_k = np.argmax(calinski_scores) + 1
print("El Índice de Calinski-Harabasz es:", best_k)

# Aplicar K-Means con el mejor valor de k
kmeans = KMeans(n_clusters=best_k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score
```

```
2 3 4 5 6 7 8 9 10 11 12 13 14
```




```
El Índice de Calinski-Harabasz es: 13
```

A continuación, se muestra el codo sin ajustes que se había obtenido inicialmente:



Se seleccionó el modelo con 2 a 15 clústeres dado que presenta un mejor rendimiento con base en Calinski-Harabasz, con un score de 13. Ahora, cuando se aprecia el resultado gráfico contra el obtenido inicialmente en esta semana, es posible visualizar una caída del codo un poco más suave y menos abrupta. En este sentido, consideramos que es la mejor opción entre los modelos generados.

Parte 16. Modelo final

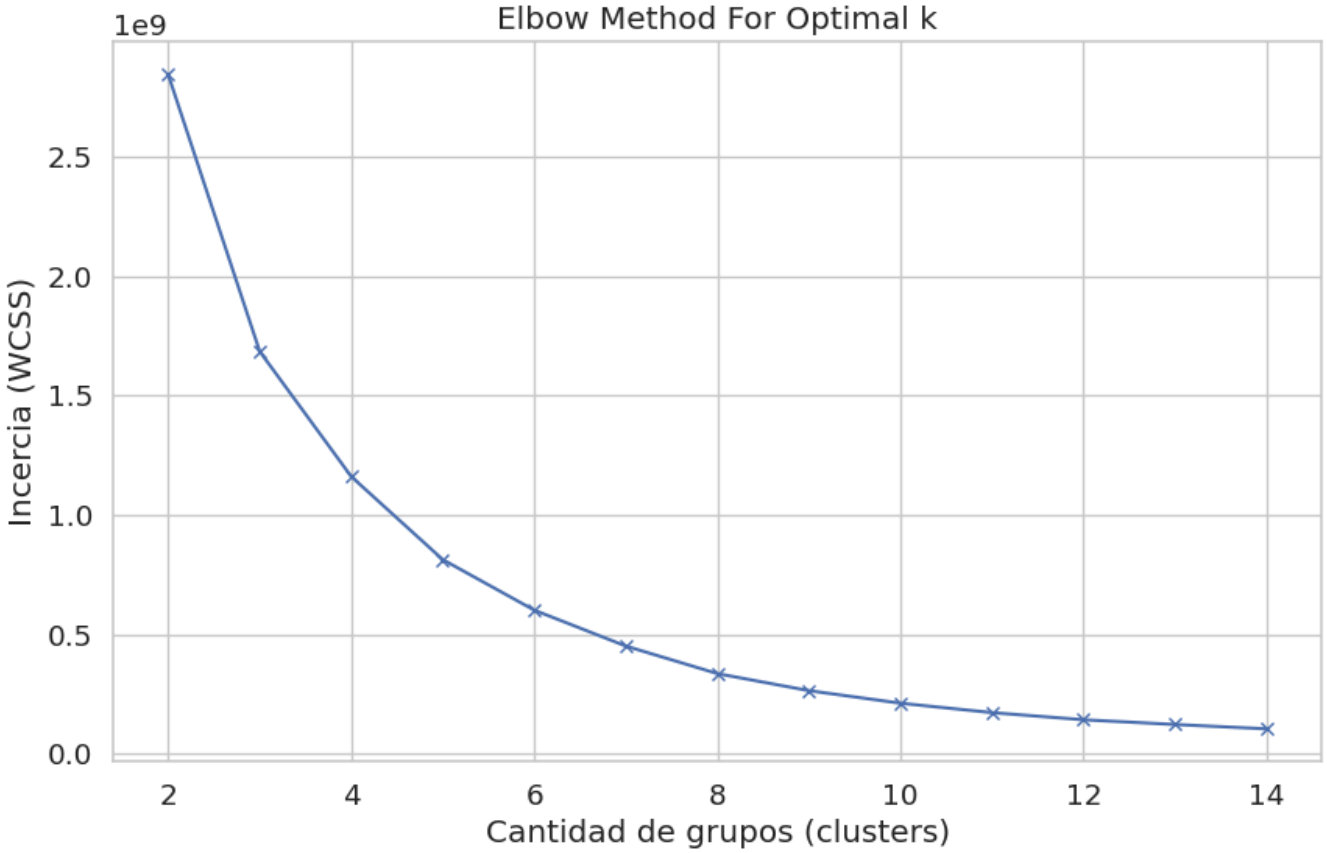
Tal como se menciona en la sección anterior, consideramos que el modelo a utilizar es k-means con 2 a 15 clústeres. El rendimiento desde el codo se aprecia con movimientos más suaves.

In [ ]:

```
from sklearn.metrics import calinski_harabasz_score
K = range(2,15) #Rango de 2 a 15 clusters
#Creamos una lista para la suma del cuadrado de las distancias
sumaCuadradoDistancias=[]

#Creamos un bucle que recorra el rango de grupos
for k in K:
    print(k, end=' ')
    #Ajustamos el rango de grupos a nuestra variable de latitud y longitudT
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=20).fit(vuln_df[['Vuln_Score','POI
    #.inertia_ nos computa la suma del cuadrado de las distancias
    sumaCuadradoDistancias.append(kmeans.inertia_)

#Ajustamos el tamaño de la gráfica para una mejor visualización
plt.figure(figsize=(10, 6))
#Se genera la gráfica con el rango de grupos y su suma cuadrada de distancia
plt.plot(K, sumaCuadradoDistancias, 'bx-')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Cantidad de grupos (clusters)')
plt.ylabel('Incercia (WCSS)')
plt.show()

#Mejora
# Utilizando K-Means++ y calculando el Índice de Calinski-Harabasz
calinski_scores = []
for k in K:
```

```
    kmeans = KMeans(n_clusters=k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score'
    calinski_scores.append(calinski_harabasz_score(vuln_df[['Vuln_Score','POBTOT']], kmean

# Se debe seleccionar el valor de k con el mejor Índice de Calinski-Harabasz
best_k = np.argmax(calinski_scores) + 1
print("El Índice de Calinski-Harabasz es:", best_k)

# Aplicar K-Means con el mejor valor de k
kmeans = KMeans(n_clusters=best_k, random_state=1, n_init='auto').fit(vuln_df[['Vuln_Score
```

```
2 3 4 5 6 7 8 9 10 11 12 13 14
```




```
El Índice de Calinski-Harabasz es: 13
```

In [ ]:
```
kmeans = KMeans(n_clusters=5, random_state=1)
y_kmeans = kmeans.fit_predict(vuln_df[['Vuln_Score','POBTOT']])
```

In [ ]:
```
centros=kmeans.cluster_centers_
for i in range(0,5):
  print('Grupo',i,'Centro:',centros[i-1][0].round(2),)
```

```
Grupo 0 Centro: 7.72
Grupo 1 Centro: 6.56
Grupo 2 Centro: 7.89
Grupo 3 Centro: 7.84
Grupo 4 Centro: 7.47
```

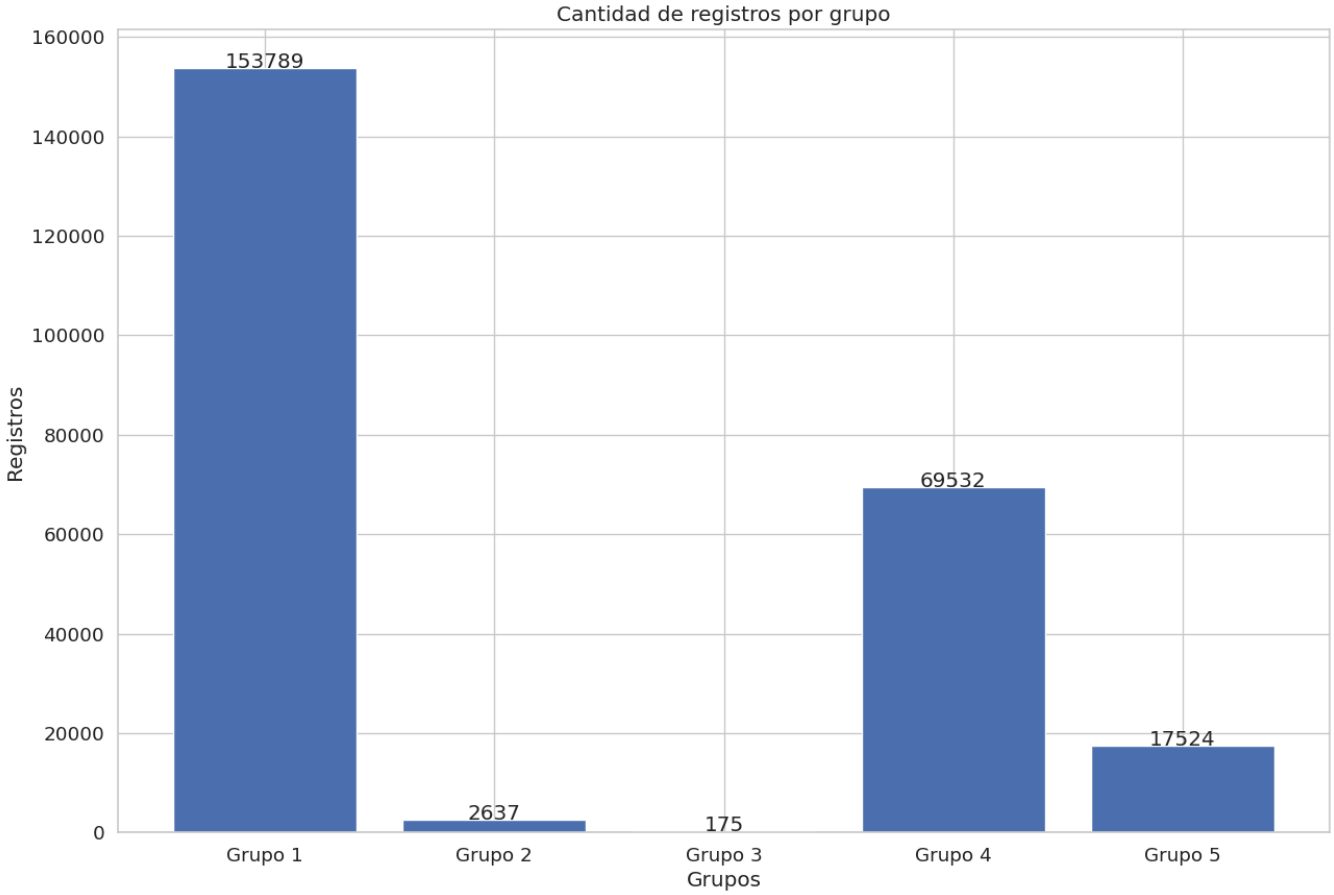
In [ ]:
```
clusterCount=np.bincount(y_kmeans)
for i in range(0,5):
  print('Grupo ',i,', registros del grupo:',clusterCount[i-1])
```

```
Grupo  0 , registros del grupo: 17524
Grupo  1 , registros del grupo: 153789
Grupo  2 , registros del grupo: 2637
Grupo  3 , registros del grupo: 175
```

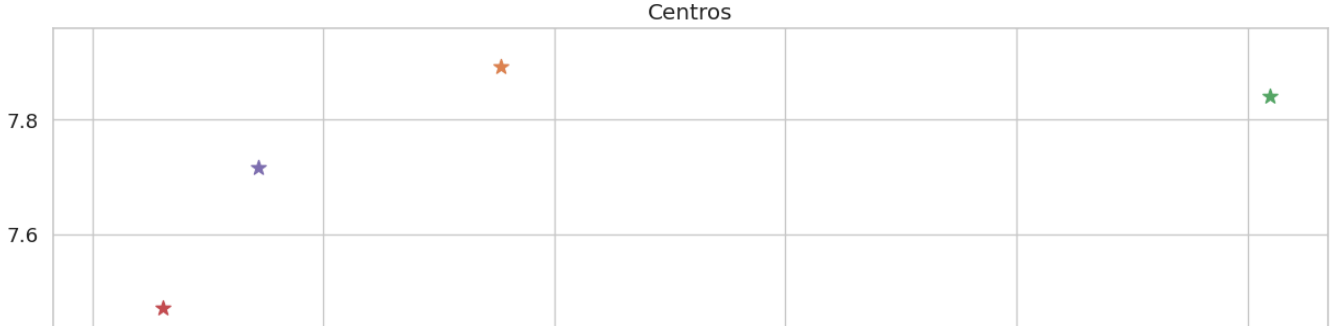
```
Grupo  4 , registros del grupo: 69532
```

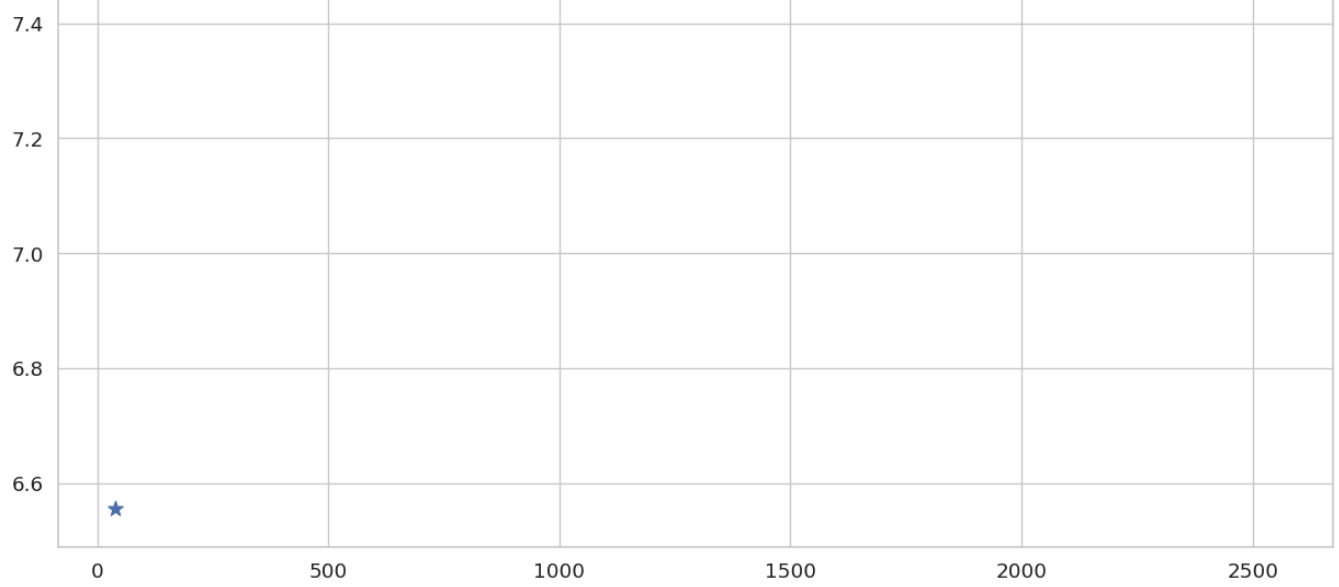
In [ ]:
```
for i in range(0,5):
  exec("c" + str(i) + '=kmeans.cluster_centers_[i]')
```

In [ ]:
```
grupos=['Grupo 1','Grupo 2','Grupo 3','Grupo 4','Grupo 5']
plt.bar(grupos,clusterCount)
plt.xlabel('Grupos')
plt.ylabel('Registros')
plt.title('Cantidad de registros por grupo')
for i in range(len(clusterCount)):
        plt.text(i, clusterCount[i], clusterCount[i], ha = 'center')
```



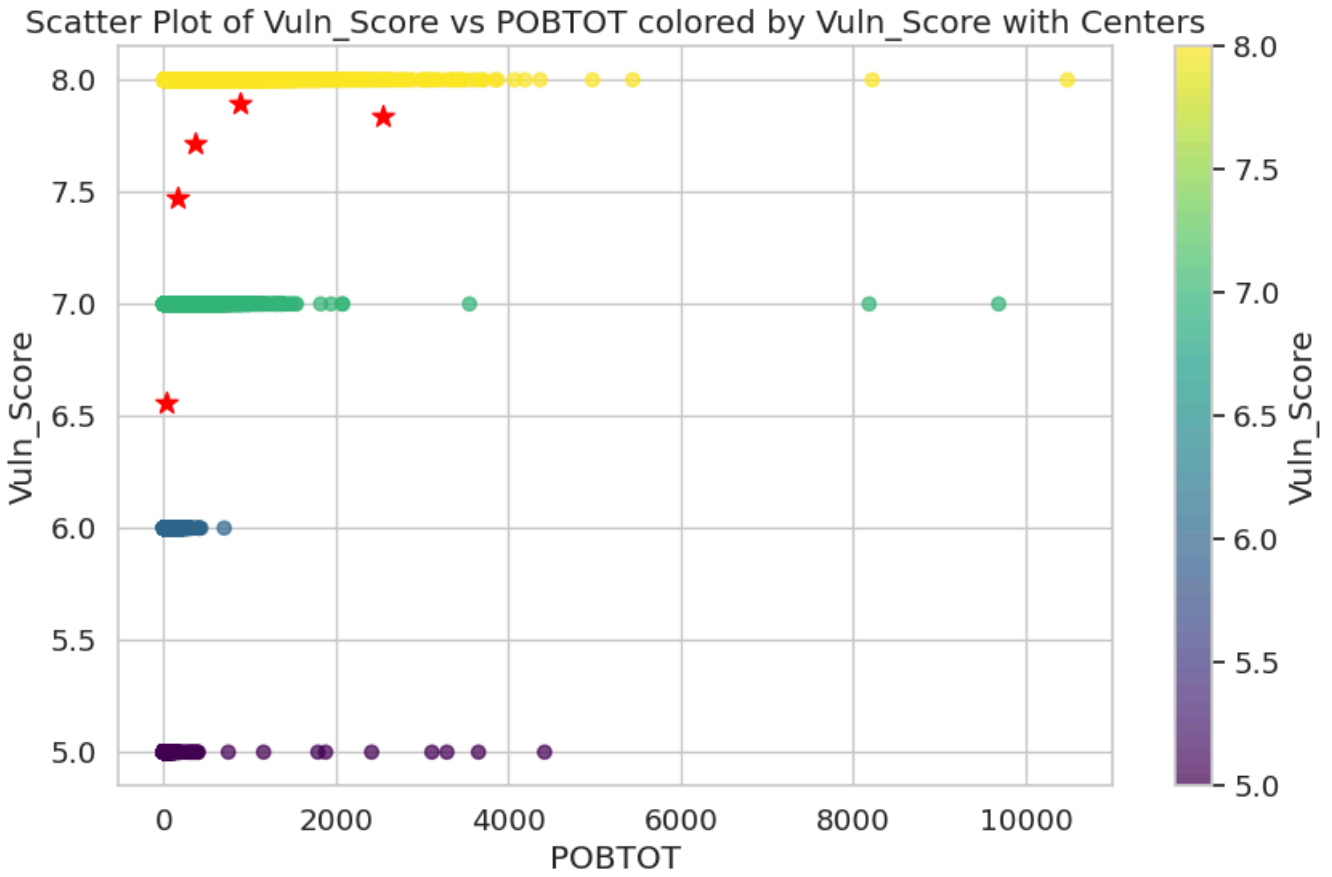
In [ ]:
```
#Creamos una lista que contiene todas las variables correspondientes a los centros
varsCentros=[c0,c1,c2,c3,c4]

#Recorremos la lista anterior para graficar los centros en función de la longitud y latitud
for i in range(0,5):
  plt.scatter(varsCentros[i][1], varsCentros[i][0],marker='*',s=100)
  plt.title('Centros')
```





In [ ]:

```
plt.figure(figsize=(10, 6))
sc = plt.scatter(vuln_df['POBTOT'], vuln_df['Vuln_Score'], c=vuln_df['Vuln_Score'], cmap='

plt.colorbar(sc, label='Vuln_Score')

varsCentros = [c0, c1, c2,c3,c4]

for i in range(5):
    plt.scatter(varsCentros[i][1], varsCentros[i][0], marker='*', s=100, color='red', labe

plt.xlabel('POBTOT')
plt.ylabel('Vuln_Score')
plt.title('Scatter Plot of Vuln_Score vs POBTOT colored by Vuln_Score with Centers')

plt.show()
```



Conclusiones al 20 de octubre

Durante el desarrollo de la práctica nos fue posible retomar el análisis de k-means. Adicionalmente, exploramos más algoritmos de clasificación, en específico de entrenamiento no supervisado. Esto nos permitió agrupar a la población con base en el índice de vulnerabilidad. Dentro de la comparativa realizada fue posible destacar que la mayoría de los algoritmos poseen una metodología similar ya que toman un punto de referencia y determinan a través de esto la pertenencia de los puntos contiguos. Dadas las distancias, se evalúa si se pertenece o no al grupo asignado.

Fue enriquecedor utilizar distintos formatos gráficos para comparar los resultados, así como centralizar el análisis en una métrica en particular para garantizar la comparabilidad. Cabe destacar que el algoritmo Gausian Mixture se diferencia de los otros métodos utilizados, dado que en lugar de asumir la pertenencia de un grupo de puntos a un clúster, lo estima de manera probabilística. Sin embargo, al analizar su rendimiento en comparación a los demás, preferimos utilizar k-means y BIRCH.

Los resultados obtenidos fueron enriquecedores, ya que nos fue posible obtener un mejor rendimiento de k-means, esto respecto con el baseline de la semana pasada. Es importante realizar un análisis con un ajuste de hiperparámetros para poder explorar cuál sería la selección que genere un mejor rendimiento del modelo. Esto se puede lograr al construir un pipeline donde se definirán los diferentes valores para analizar los hiperparámetros contra una métrica definida y así obtener el rendimiento esperado. Lo anterior se realizó para ambos mejores modelos seleccionados y fue gracias a dicho proceso que fue posible definir el moedelo final.

Fuentes consultadas

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