Data_Analitycs

October 19, 2025

0.1 Contexto del dataset

La criminalidad urbana representa uno de los principales desafíos para la gestión de la seguridad en ciudades metropolitanas. En este contexto, el conjunto de datos *Crimes – 2001 to Present*, publicado por el Chicago Data Portal, constituye una fuente de información clave para comprender la evolución del crimen en la ciudad a lo largo de más de veinte años.

El dataset recopila más de **8,4 millones** de incidentes reportados desde 2001 hasta la actualidad (con la salvedad de los siete días más recientes). Incluye información sobre el tipo de delito, ubicación, arrestos, descripciones, coordenadas geográficas y división administrativa donde ocurrió cada hecho. Gracias a esta riqueza de variables, se puede realizar tanto un análisis exploratorio de datos (EDA) como la implementación de modelos de *Machine Learning* orientados a la predicción y clasificación.

0.2 Objetivos generales y específicos

Realizar un Análisis Exploratorio de Datos (EDA) del conjunto de crímenes de Chicago (2001–actualidad), complementado con modelos de *Machine Learning*, para identificar patrones temporales, espaciales y factores asociados a los arrestos.

1. Analizar las tendencias de criminalidad en Chicago durante el periodo 2001–2025, destacando los cambios relacionados con eventos socioeconómicos relevantes.
2. Identificar los factores que influyen en la probabilidad de que un delito concluya en un arresto mediante la aplicación de modelos de clasificación supervisada.
3. ¿Dónde se concentran los delitos en Chicago y cómo han cambiado los hotspots geográficos a lo largo del tiempo?

0.3 Preguntas iniciales de investigación

1. ¿Qué tendencias de largo plazo se observan en la frecuencia y tipo de delitos en Chicago, y cómo han influido eventos como la crisis del 2008 o la pandemia del 2020?
2. ¿Qué factores determinan la probabilidad de que un crimen resulte en un arresto, y hasta qué punto un modelo de Machine Learning puede predecir este resultado con precisión?
3. ¿Dónde se concentran los delitos en Chicago y cómo han cambiado los hotspots geográficos a lo largo del tiempo?

1 Selección de datos

1.1 Justificación del dataset elegido

El dataset seleccionado, *Crimes – 2001 to Present*, es publicado y actualizado diariamente por el Departamento de Policía de Chicago a través del portal de datos abiertos de la ciudad. Se eligió este conjunto porque:

- Posee un horizonte temporal amplio (2001–actualidad), lo que permite analizar tendencias históricas.
- Contiene variables categóricas, numéricas, espaciales y temporales que facilitan un análisis integral.
- Permite combinar técnicas de análisis descriptivo y predictivo, lo que aporta amplitud y profundidad al trabajo.

1.2 Descripción de las variables principales

El dataset incluye **22 columnas**, entre las cuales destacan:

- **ID**: identificador único del incidente.
- **Case Number**: número de caso de la Policía de Chicago.
- **Date**: fecha y hora en que ocurrió el delito.
- **Primary Type**: categoría principal del delito (ejemplo: robo, asalto, fraude).
- **Description**: detalle secundario del delito.
- **Location Description**: lugar donde ocurrió (ejemplo: calle, residencia, comercio).
- **Arrest**: indica si el crimen resultó en arresto.
- **Domestic**: señala si estuvo vinculado a violencia doméstica.
- **District, Ward, Community Area, Beat**: divisiones administrativas y policiales.
- **Latitude, Longitude, X Coordinate, Y Coordinate**: localización aproximada.
- **Year**: año del incidente.

2 Datos y preparación

[7]:
```
# Instalar todas las librerías necesarias para el proyecto
%pip install numpy pandas matplotlib seaborn scikit-learn contextily folium␣
↪hdbscan
```

```
Requirement already satisfied: numpy in /opt/homebrew/lib/python3.11/site-
packages (2.3.3)
Requirement already satisfied: pandas in /opt/homebrew/lib/python3.11/site-
packages (2.3.3)
Requirement already satisfied: matplotlib in /opt/homebrew/lib/python3.11/site-
packages (3.10.6)
Requirement already satisfied: seaborn in /opt/homebrew/lib/python3.11/site-
packages (0.13.2)
Requirement already satisfied: scikit-learn in
/opt/homebrew/lib/python3.11/site-packages (1.7.2)
Requirement already satisfied: contextily in /opt/homebrew/lib/python3.11/site-
packages (1.6.2)
```

```
Requirement already satisfied: folium in /opt/homebrew/lib/python3.11/site-
packages (0.20.0)
Requirement already satisfied: hdbscan in /opt/homebrew/lib/python3.11/site-
packages (0.8.40)
Requirement already satisfied: python-dateutil>=2.8.2 in
/Users/stewart/Library/Python/3.11/lib/python/site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/opt/homebrew/lib/python3.11/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/opt/homebrew/lib/python3.11/site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/Users/stewart/Library/Python/3.11/lib/python/site-packages (from matplotlib)
(25.0)
Requirement already satisfied: pillow>=8 in /opt/homebrew/lib/python3.11/site-
packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: scipy>=1.8.0 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: geopy in /opt/homebrew/lib/python3.11/site-
packages (from contextily) (2.4.1)
Requirement already satisfied: mercantile in /opt/homebrew/lib/python3.11/site-
packages (from contextily) (1.2.1)
Requirement already satisfied: rasterio in /opt/homebrew/lib/python3.11/site-
packages (from contextily) (1.4.3)
Requirement already satisfied: requests in /opt/homebrew/lib/python3.11/site-
packages (from contextily) (2.32.5)
Requirement already satisfied: xyzservices in /opt/homebrew/lib/python3.11/site-
packages (from contextily) (2025.4.0)
Requirement already satisfied: branca>=0.6.0 in
/opt/homebrew/lib/python3.11/site-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /opt/homebrew/lib/python3.11/site-
packages (from folium) (3.1.6)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/homebrew/lib/python3.11/site-packages (from jinja2>=2.9->folium) (3.0.3)
```

```
Requirement already satisfied: six>=1.5 in
/Users/stewart/Library/Python/3.11/lib/python/site-packages (from python-
dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: geographiclib<3,>=1.52 in
/opt/homebrew/lib/python3.11/site-packages (from geopy->contextily) (2.1)
Requirement already satisfied: click>=3.0 in /opt/homebrew/lib/python3.11/site-
packages (from mercantile->contextily) (8.3.0)
Requirement already satisfied: affine in /opt/homebrew/lib/python3.11/site-
packages (from rasterio->contextily) (2.4.0)
Requirement already satisfied: attrs in /opt/homebrew/lib/python3.11/site-
packages (from rasterio->contextily) (25.3.0)
Requirement already satisfied: certifi in /opt/homebrew/lib/python3.11/site-
packages (from rasterio->contextily) (2025.8.3)
Requirement already satisfied: cligj>=0.5 in /opt/homebrew/lib/python3.11/site-
packages (from rasterio->contextily) (0.7.2)
Requirement already satisfied: click-plugins in
/opt/homebrew/lib/python3.11/site-packages (from rasterio->contextily) (1.1.1.2)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/homebrew/lib/python3.11/site-packages (from requests->contextily) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in
/opt/homebrew/lib/python3.11/site-packages (from requests->contextily) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/homebrew/lib/python3.11/site-packages (from requests->contextily) (2.5.0)

[notice] A new release of pip is
available: 25.1.1 -> 25.2
[notice] To update, run:
/opt/homebrew/opt/python@3.11/bin/python3.11 -m pip install
--upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

[8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import contextily as ctx
import folium
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score, classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction import FeatureHasher
from sklearn.cluster import DBSCAN
import gc
import hdbscan
```

[9]:
```
# Ruta del dataset
file_path = 'Crimes_Chicago_full.csv'
```

```
# Carga del archivo CSV con fechas
df = pd.read_csv(file_path, parse_dates=['Date'], low_memory=False)
```

[10]:
```
# Descripción básica del dataset
total_records = df.shape[0]
total_features = df.shape[1]
print('Total de registros:', total_records)
print('Total de features:', total_features)
```

```
Total de registros: 8409498
Total de features: 22
```

2.1 Manejo de datos faltantes

Mapa de Áreas Comunitarias y Distrito electoral (Wards) de Chicago

No todos los valores faltantes son iguales * Faltante no al azar - Missing not at random(MNAR) * Faltante al azar - Missing at random(MAR) * Faltante completamente al azar - Missing completely at random (MCAR)

Manejos 1. Eliminacion 2. Imputacion: * Datos estadisticos * KNN - imputation * Missforest - imputation

Evaluacion de cantidades

[11]:
```
# Cantidad de nulos y porcentaje por columna
null_counts = df.isnull().sum()
null_percent = (null_counts / total_records) * 100
null_df = pd.DataFrame({'Null Count': null_counts, 'Null Percentage':␣
 ↪null_percent})
print('\nValores nulos por columna (Tabla 1):')
print(null_df[null_df['Null Count'] > 0].sort_values(by='Null Count',␣
 ↪ascending=False))
```

```
Valores nulos por columna (Tabla 1):
                     Null Count  Null Percentage
Ward                     614822         7.311043
Community Area           613687         7.297546
X Coordinate              93703         1.114252
Y Coordinate              93703         1.114252
Latitude                  93703         1.114252
Longitude                 93703         1.114252
Location                  93703         1.114252
Location Description      14887         0.177026
District                     47         0.000559
```

[12]:
```
# cantidad registros y porcentaje con 1,2,3,... nulos
null_record_counts = df.isnull().sum(axis=1).value_counts().sort_index()
```

```
null_record_percent = (null_record_counts / total_records) * 100
null_record_df = pd.DataFrame({'Record Count': null_record_counts, 'Record␣
↪Percentage': null_record_percent})
print('\nRegistros con n nulos (Tabla 2):')
print(null_record_df[null_record_df['Record Count'] > 0])
```

```
Registros con n nulos (Tabla 2):
   Record Count  Record Percentage
0       7699140          91.552908
1         13537           0.160973
2        603115           7.171831
3             3           0.000036
5         79132           0.940984
6          5476           0.065117
7          9094           0.108140
8             1           0.000012
```

Con referencia a los valores nulos por columna, se puede rescatar lo siguiente:

1. De aproximadamente **8.4 millones de registros**, el **91.55% no presenta valores nulos**, lo que indica que la calidad global del dataset es **alta**.
2. Los features relacionados con coordenadas geográficas (`X Coordinate`, `Y Coordinate`, `Latitude`, `Longitude` y `Location`) presentan alrededor de **1.11% de datos faltantes cada uno**.
   A primera vista, eliminarlos de manera independiente sería riesgoso, ya que podría implicar una pérdida acumulada cercana al **5.57%** del dataset. Sin embargo, dado que cada columna tiene exactamente **93,703 valores nulos**, se verificara si corresponden a los **mismos registros faltantes** en todas estas variables. En caso sea así, se procedera con la **ELIMINACIÓN DIRECTA**.
3. El feature `District` es el que presenta **menos valores nulos**, con apenas **47 registros faltantes** sobre 8.4 millones. Esta cantidad es **despreciable** y puede resolverse fácilmente con **ELIMINACIÓN DIRECTA**.
4. El feature `Location Description` presenta menos del **1% de valores nulos**, siendo el segundo con menor cantidad de datos faltantes. Debido a su baja proporción, dichos valores se consideran despreciables, por lo que resulta adecuado aplicar una **ELIMINACIÓN DIRECTA**.
5. Las columnas con **mayor proporción de valores nulos** son `Ward` (7.31%) y `Community Area` (7.30%), representando el caso más crítico en términos de volumen de datos faltantes. Por lo que se evaluara un tipo de **IMPUTACIÓN**

[13]:
```
# cuantos registros no tienen `X Coordinate`, `Y Coordinate`, `Latitude`,␣
↪`Longitude` y `Location`
missing_coords = df[['X Coordinate', 'Y Coordinate', 'Latitude', 'Longitude',␣
↪'Location']].isnull().all(axis=1)
missing_coords_count = missing_coords.sum()
```

```
print(f'\nCantidad de registros que no tienen ninguna de estos features (X␣
 ↪Coordinate, Y Coordinate, Latitude, Longitude, Location):␣
 ↪\n{missing_coords_count}')
```

```
Cantidad de registros que no tienen ninguna de estos features (X Coordinate, Y
Coordinate, Latitude, Longitude, Location):
93703
```

Evaluando si eliminar resgistros que no tienen coordenadas

De acuerdo al resultado lso features 'X Coordinate', 'Y Coordinate', 'Latitude', 'Longitude', 'Location' , son nulos en los mismos registros, por lo que no afectaria tanto perder 1.1% de los registros. Sin embargo, ello no es suficiente para proceder la eliminacion de datos, antes se debe verificar si son **MNAR,MAR,MCAR**. Para ello se procede a generar un dataframe con los datos a eliminar, para poder hacerle un analisis exhaustivo.

[14]:
```
# Se crea un dataframe con los registros que tienen nulos en 'X Coordinate', 'Y␣
 ↪Coordinate', 'Latitude', 'Longitude', 'Location'
df_sin_coords = df.dropna(subset=['X Coordinate', 'Y Coordinate', 'Latitude',␣
 ↪'Longitude', 'Location', 'Location Description', 'District'])

missing_coords_df = df[missing_coords]
print(f'\nMuestra de registros sin coordenadas (X Coordinate, Y Coordinate,␣
 ↪Latitude, Longitude, Location):')
missing_coords_df
```

```
Muestra de registros sin coordenadas (X Coordinate, Y Coordinate, Latitude,
Longitude, Location):
```

[14]:

	ID	Case Number	Date	Block	\
1	4644631	HM243132	2004-12-31 23:59:00	115XX S LAFAYETTE AVE	
2	4606507	HM197987	2004-12-31 23:59:00	053XX N DAMEN AVE	
278	3805880	HL169683	2004-12-31 19:50:00	100XX W OHARE ST	
505	7076279	HR483215	2004-12-31 15:00:00	049XX S LANGLEY AVE	
677	8955287	HW104165	2004-12-31 12:00:00	013XX W 76TH ST	
…	…	…	…	…	
8409490	12958667	JG122701	2005-01-01 00:00:00	007XX S KEELER AVE	
8409491	10362868	HY554697	2005-01-01 00:00:00	072XX S PEORIA ST	
8409494	7982771	HT214754	2005-01-01 00:00:00	072XX S EMERALD AVE	
8409496	6313296	HP402411	2005-01-01 00:00:00	087XX S HALSTED ST	
8409497	4792806	HM407268	2005-01-01 00:00:00	040XX W DIVISION ST	

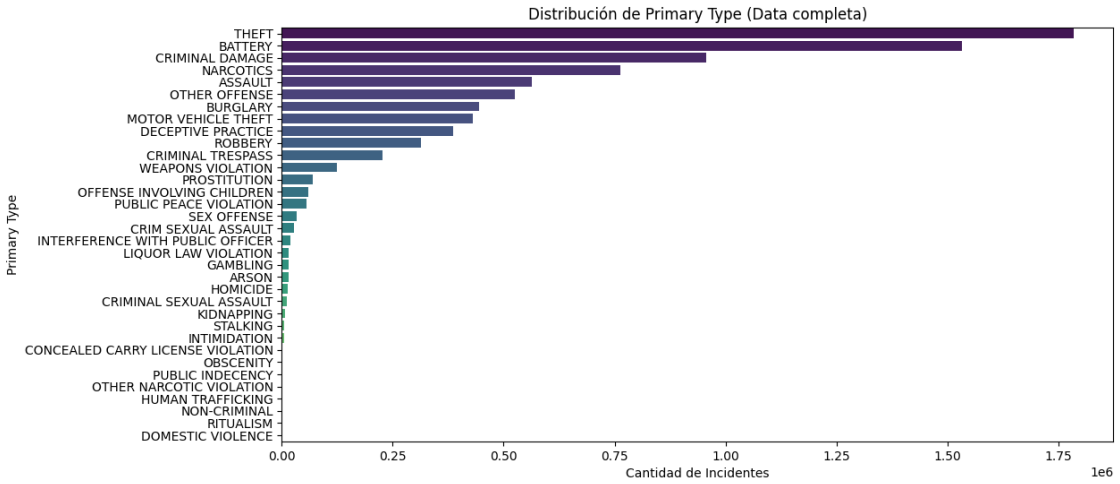
	IUCR	Primary Type	\
1	0840	THEFT	
2	0840	THEFT	
278	1206	DECEPTIVE PRACTICE	
505	5002	OTHER OFFENSE	

```
677      0842                      THEFT
…        …                           …
8409490  1153          DECEPTIVE PRACTICE
8409491  1562                SEX OFFENSE
8409494  1754  OFFENSE INVOLVING CHILDREN
8409496  0840                      THEFT
8409497  0840                      THEFT

                              Description Location Description  Arrest  \
1              FINANCIAL ID THEFT: OVER $300            RESIDENCE   False
2              FINANCIAL ID THEFT: OVER $300                ALLEY   False
278                THEFT BY LESSEE,MOTOR VEH     AIRPORT/AIRCRAFT    True
505                    OTHER VEHICLE OFFENSE               STREET   False
677                  AGG: FINANCIAL ID THEFT            RESIDENCE   False
…                                        …                   …     …
8409490  FINANCIAL IDENTITY THEFT OVER $ 300                  NaN   False
8409491            AGG CRIMINAL SEXUAL ABUSE            RESIDENCE   False
8409494       AGG SEX ASSLT OF CHILD FAM MBR            RESIDENCE   False
8409496        FINANCIAL ID THEFT: OVER $300            RESIDENCE   False
8409497        FINANCIAL ID THEFT: OVER $300            RESIDENCE   False

         Domestic  …  Ward  Community Area  FBI Code  X Coordinate  \
1           False  …  34.0            53.0        06           NaN
2           False  …  40.0             4.0        06           NaN
278         False  …  41.0            76.0        11           NaN
505          True  …   4.0            38.0        26           NaN
677         False  …  17.0            71.0        06           NaN
…             …  …   …              …        …           …
8409490     False  …  24.0            26.0        11           NaN
8409491     False  …  17.0            68.0        17           NaN
8409494      True  …  17.0            68.0        02           NaN
8409496     False  …  21.0            71.0        06           NaN
8409497     False  …  27.0            23.0        06           NaN

        Y Coordinate  Year              Updated On  Latitude Longitude  \
1                NaN  2004  08/17/2015 03:03:40 PM       NaN       NaN
2                NaN  2004  08/17/2015 03:03:40 PM       NaN       NaN
278              NaN  2004  08/17/2015 03:03:40 PM       NaN       NaN
505              NaN  2004  08/17/2015 03:03:40 PM       NaN       NaN
677              NaN  2004  08/17/2015 03:03:40 PM       NaN       NaN
…              …   …                       …       …       …
8409490          NaN  2005  01/21/2023 03:42:35 PM       NaN       NaN
8409491          NaN  2005  01/02/2016 03:52:42 PM       NaN       NaN
8409494          NaN  2005  08/17/2015 03:03:40 PM       NaN       NaN
8409496          NaN  2005  08/17/2015 03:03:40 PM       NaN       NaN
8409497          NaN  2005  08/17/2015 03:03:40 PM       NaN       NaN
```

```
         Location
1             NaN
2             NaN
278           NaN
505           NaN
677           NaN
…             …
8409490       NaN
8409491       NaN
8409494       NaN
8409496       NaN
8409497       NaN

[93703 rows x 22 columns]
```

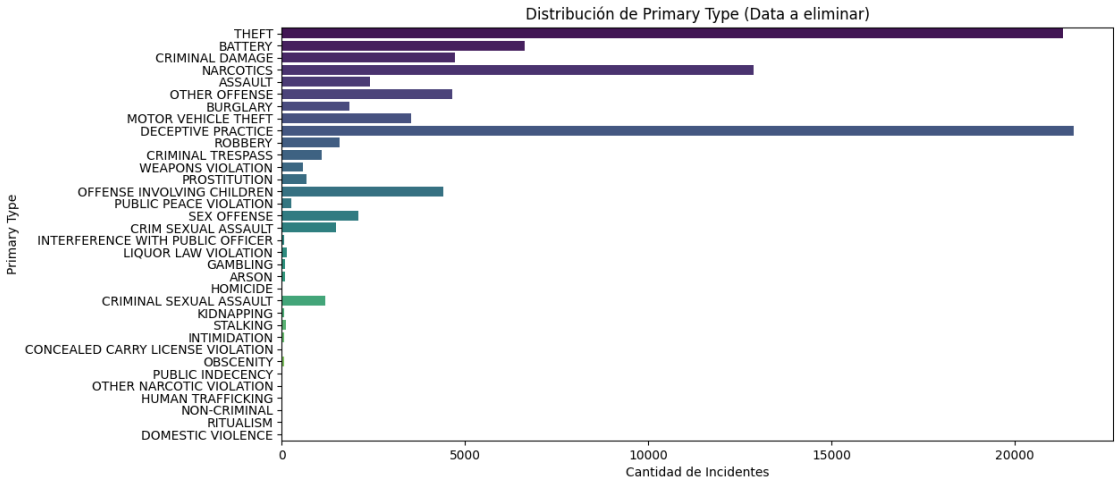
[15]:
```
# bartplot de Primary Type con data completa
plt.figure(figsize=(12,6))
sns.countplot(data=df, y='Primary Type', order=df['Primary Type'].
 ↪value_counts().index, palette='viridis')
plt.title('Distribución de Primary Type (Data completa)')
plt.xlabel('Cantidad de Incidentes')
plt.ylabel('Primary Type')
plt.show()
```

```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/895039656.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df, y='Primary Type', order=df['Primary
Type'].value_counts().index, palette='viridis')
```



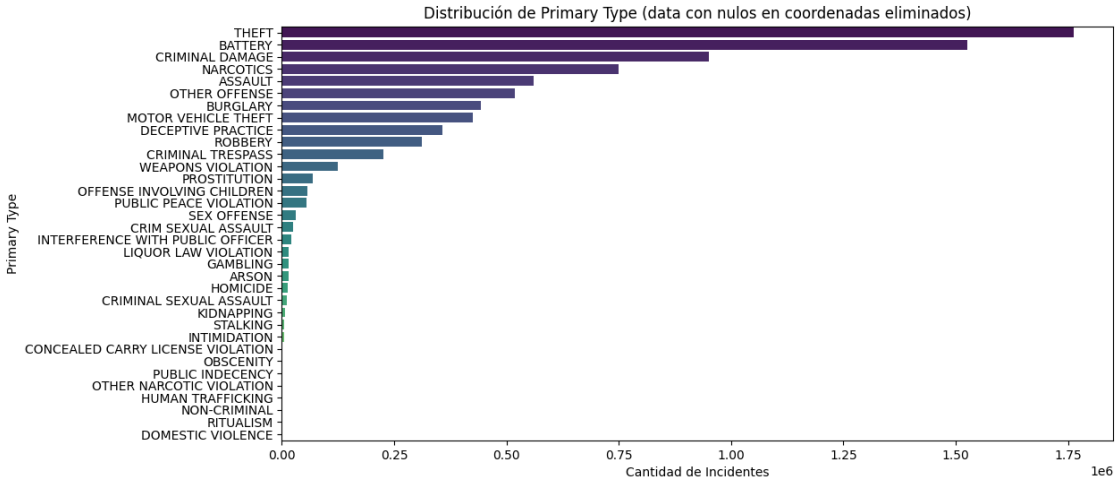
[16]:
```
# bartplot de Primary Type con data con nulos en coordenadas
plt.figure(figsize=(12,6))
sns.countplot(data=missing_coords_df, y='Primary Type', order=df['Primary␣
 ↪Type'].value_counts().index, palette='viridis')
plt.title('Distribución de Primary Type (Data a eliminar)')
plt.xlabel('Cantidad de Incidentes')
plt.ylabel('Primary Type')
plt.show()
```

```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/408527995.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=missing_coords_df, y='Primary Type', order=df['Primary
Type'].value_counts().index, palette='viridis')
```



[17]:
```
# bartplot de Primary Type con data sin nulos en coordenadas
plt.figure(figsize=(12,6))
sns.countplot(data=df_sin_coords, y='Primary Type',␣
 ↪order=df_sin_coords['Primary Type'].value_counts().index, palette='viridis')
plt.title('Distribución de Primary Type (data con nulos en coordenadas␣
 ↪eliminados)')
plt.xlabel('Cantidad de Incidentes')
plt.ylabel('Primary Type')
plt.show()
```

```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/3445521923.py:3
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df_sin_coords, y='Primary Type',
order=df_sin_coords['Primary Type'].value_counts().index, palette='viridis')
```



Se puede observar que, en la **Distribución de Primary Type (data completa)** frente a la **Distribución de Primary Type (data a eliminar)**, ambas mantienen en general la misma tendencia.
Sin embargo, los valores faltantes en coordenadas no son completamente aleatorios (MCAR), ya que existe una excepción relevante: el `Primary Type` *Deceptive Practice*, que concentra una cantidad anormal de nulos.

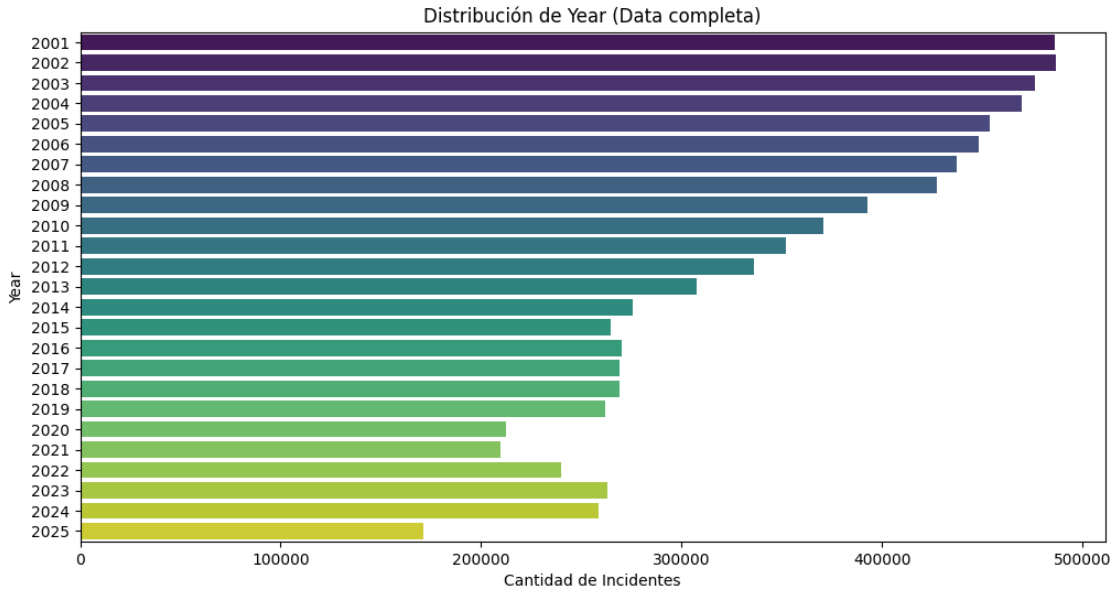
Aun así, al visualizar la **Distribución de Primary Type (data con nulos en coordenadas eliminados)**, se evidencia que la eliminación de estos registros no altera de manera sustancial la distribución global. La tendencia de los diferentes tipos de crimen se conserva prácticamente igual, lo que sugiere que el impacto de dicha eliminación es despreciable a nivel general.

Al igual como evaluamos el `Primary Type`, ahora prodeceremos a evaluar el `Year`, por si existe riesgo que toda la data a eliminar sean registros de un año en particular

[18]:
```
# bartplot de Year con data completa  años ascendente , porcentajes
plt.figure(figsize=(12,6))
sns.countplot(data=df, y='Year', order=sorted(df['Year'].value_counts().index),␣
 ↪palette='viridis', )
plt.title('Distribución de Year (Data completa)')
plt.xlabel('Cantidad de Incidentes')
plt.ylabel('Year')
```

```
plt.show()
```

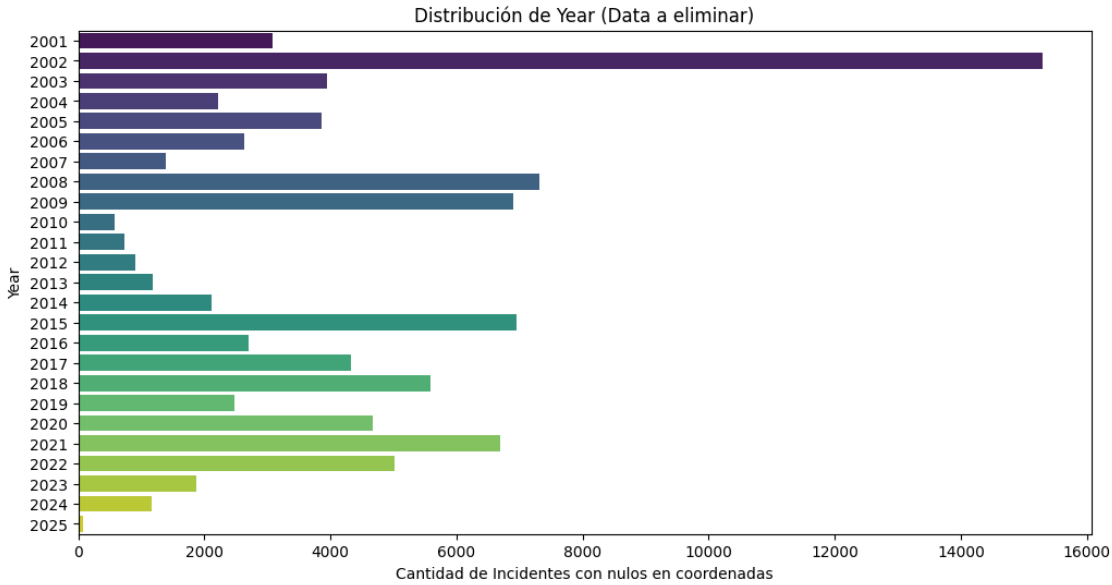
```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/1818493274.py:3
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df, y='Year',
order=sorted(df['Year'].value_counts().index), palette='viridis', )
```



[19]:
```
# bartplot Year con data con nulos en coordenadas, porcentajes
plt.figure(figsize=(12,6))
sns.countplot(data=missing_coords_df, y='Year',␣
  ↪order=sorted(missing_coords_df['Year'].value_counts().index),␣
  ↪palette='viridis')
plt.title('Distribución de Year (Data a eliminar)')
plt.xlabel('Cantidad de Incidentes con nulos en coordenadas')
plt.ylabel('Year')
plt.show()
```

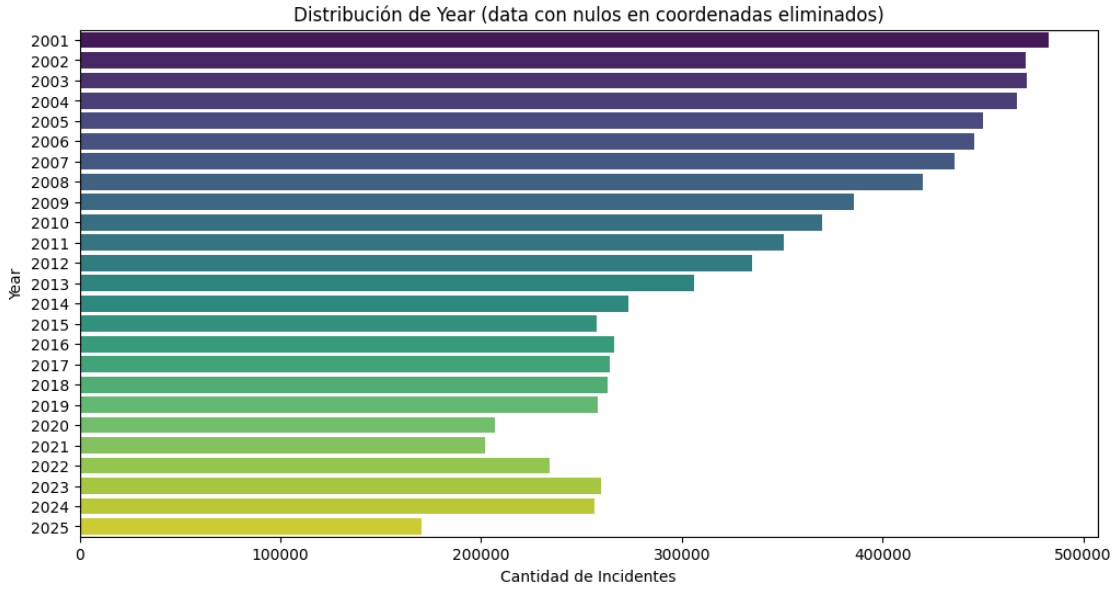
```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/2178148821.py:3
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
  sns.countplot(data=missing_coords_df, y='Year',
order=sorted(missing_coords_df['Year'].value_counts().index), palette='viridis')
```



[20]:
```
plt.figure(figsize=(12,6))
sns.countplot(data=df_sin_coords, y='Year', order=sorted(df_sin_coords['Year'].
  ↪value_counts().index), palette='viridis' )
plt.title('Distribución de Year (data con nulos en coordenadas eliminados)')
plt.xlabel('Cantidad de Incidentes')
plt.ylabel('Year')
plt.show()
```

```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/739185782.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df_sin_coords, y='Year',
order=sorted(df_sin_coords['Year'].value_counts().index), palette='viridis' )
```



De manera similar al caso anterior, se identificó una anormalidad en el año **2002** en la **Distribución de Year(Data a eliminar)**, el cual presenta un volumen atípico de incidentes. Sin embargo, al tratarse del segundo año con mayor cantidad de registros, su impacto en la **Distribución de Year (data con nulos en coordenadas eliminados)** no resulta significativo.
En contraste, los demás años mantienen un comportamiento aleatorio respecto a los valores nulos.

Por lo tanto, de acuerdo con lo evaluado (los datos faltantos en general son al azar **(MAR)**), se procede a la **eliminación directa** de los nulos en los siguientes features:
- X Coordinate
- Y Coordinate
- Latitude
- Longitude
- Location
- Location Description
- District

[21]:
```
# Eliminacion de nulos en las columnas de coordenadas, Location Description y␣
↪District
df = df.dropna(subset=['X Coordinate', 'Y Coordinate', 'Latitude', 'Longitude',␣
↪'Location', 'Location Description', 'District'])
```

[22]:
```
# Cantidad de nulos y porcentaje por columna
null_counts = df.isnull().sum()
null_percent = (null_counts / total_records) * 100
null_df = pd.DataFrame({'Null Count': null_counts, 'Null Percentage':␣
↪null_percent})
print('\nValores nulos por columna (Tabla 3):')
```

```
print(null_df[null_df['Null Count'] > 0].sort_values(by='Null Count',␣
 ↪ascending=False))
```

```
Valores nulos por columna (Tabla 3):
                Null Count  Null Percentage
Ward                605564         7.200953
Community Area      604482         7.188087
```

[23]:
```
# cantidad registros y porcentaje con 1,2,3,... nulos
null_record_counts = df.isnull().sum(axis=1).value_counts().sort_index()
null_record_percent = (null_record_counts / total_records) * 100
null_record_df = pd.DataFrame({'Record Count': null_record_counts, 'Record␣
 ↪Percentage': null_record_percent})
print('\nRegistros con n nulos (Tabla 4):')
print(null_record_df[null_record_df['Record Count'] > 0])
```

```
Registros con n nulos (Tabla 4):
   Record Count  Record Percentage
0       7699140          91.552908
1          3822           0.045449
2        603112           7.171796
```

Como podemos vusualizar, comparando la **Tabla 2** y **Tabla 4**, los registos con mas de 3 datos nulos fueron **ELIMINADOS** cuando aplicamos **eliminación directa** a las columnas de coordenadas y `District`. Por otro lado, todavia debemos evaluar que hacer son las features `Ward` (7.31%) y `Community Area` (7.30%).

Para evaluar si es posible realizar una imputación, primero analizaremos dichas variables

Análisis de Ward y Community Area

[24]:
```
# Valores de los datos faltantes:
print('\nValores únicos en la columna Ward:\n', df['Ward'].unique())
print('\nValores únicos en la columna Community Area:\n', df['Community Area'].
 ↪unique())
```

```
Valores únicos en la columna Ward:
 [27.  8. 34. 24. 50. 38. 42.  9. 10. 33.  6. 15. 37. 20. 35.  5. 31. 13.
 39. 30. 11. 29. 40. 21. 32. 19.  7. 14. 17. 44. 23. 18. 43. 48.  3. 16.
  2. 41. 22. 36. 26.  1. 45. 28. 12. 47. 25. 46.  4. 49. nan]

Valores únicos en la columna Community Area:
 [24. 50. 53. 29.  2. 15.  8. 49. 46. 43. 14. 69. 66. 25. 22. 19. 65. 60.
 44. 77. 73. 67.  5. 75. 63. 26. 70. 23. 47.  6.  7. 62. 16. 42. 28. 52.
  3. 71. 32. 61. 12. 20. 30. 17. 68. 51. 38. 13. 31. 39. 72. 35. 40.  1.
 10. 34. 56. 58. 33. 27. 21. 48. 41. 37. 59.  4. 64. 55. 11. 45. 57. 36.
 74. 76. 54. 18.  9. nan  0.]
```

Mapa de Áreas Comunitarias y Distrito electoral (Wards) de Chicago

`Community Area` tiene valores enteros del 1 al 77, que son las 77 áreas comunitarias oficiales de la ciudad. Pero en la data aparece un 0, que en la práctica es un valor inválido (fuera del rango oficial). Ambas columnas tienen NaN en ~7% de los registros, y en el caso de Community Area también un 0 que debería tratarse como nulo. Por lo que se procede a unificar los nulos (NaN) y valores inválidos (0) en `Community Area`.

[25]:
```
# cantidad de 0s en Community Area
count_zeros = (df['Community Area'] == 0).sum()
print(f'\nCantidad de 0s en Community Area (Antes): {count_zeros}')

# cambiar 0 por NaN en Community Area
df['Community Area'] = df['Community Area'].replace(0, np.nan)

# cantidad de 0s en Community Area
count_zeros = (df['Community Area'] == 0).sum()
print(f'\nCantidad de 0s en Community Area (Después): {count_zeros}')
```

```
Cantidad de 0s en Community Area (Antes): 69


Cantidad de 0s en Community Area (Después): 0
```

Investigando mas sobre las variables

Un *Ward* es una división política y electoral de la ciudad de Chicago, cuyos límites se modifican cada década(10 años) como parte del proceso de redistribución electoral (redistricting), lo que lo convierte en una unidad administrativa cambiante e inestable para análisis de largo plazo (Chicago History Museum).

En contraste, las *Community Areas* son divisiones socioespaciales creadas en la década de 1930 que han permanecido estables durante casi un siglo, utilizadas ampliamente en estudios académicos, de salud pública y criminalidad por su continuidad histórica y su asociación con datos censales (WTTW).

¿Es necesario conservar ambos features? ¿Cual nos servira en función de las preguntas que establecimos?

Incluir tanto *Ward* como *Community Area* puede generar redundancia y colinealidad, ya que ambas ubican al registro dentro de la ciudad pero desde perspectivas distintas. Dado que el objetivo del análisis es identificar patrones sociales y espaciales más que políticos, resulta innecesario mantener los *Wards*, siendo más pertinente conservar únicamente las *Community Areas*, pues estas proveen una visión más estable y consistente del territorio urbano a lo largo del tiempo.

[26]:
```
# Eliminar Columna Ward
df = df.drop(columns=['Ward'])
print('\nColumnas del DataFrame después de eliminar Ward:\n', df.columns)

# cantidad registros y porcentaje con 1,2,3,... nulos
null_record_counts = df.isnull().sum(axis=1).value_counts().sort_index()
```

```
null_record_percent = (null_record_counts / total_records) * 100
null_record_df = pd.DataFrame({'Record Count': null_record_counts, 'Record␣
 ↪Percentage': null_record_percent})
print('\nRegistros con n nulos (Tabla 5):')
print(null_record_df[null_record_df['Record Count'] > 0])
```

```
Columnas del DataFrame después de eliminar Ward:
 Index(['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Primary Type',
       'Description', 'Location Description', 'Arrest', 'Domestic', 'Beat',
       'District', 'Community Area', 'FBI Code', 'X Coordinate',
       'Y Coordinate', 'Year', 'Updated On', 'Latitude', 'Longitude',
       'Location'],
      dtype='object')

Registros con n nulos (Tabla 5):
   Record Count  Record Percentage
0       7701523          91.581245
1        604551           7.188907
```

[27]:
```
# Filtrar registros con Ward o Community Area nulos
missing_geo = df[df['Community Area'].isnull()]
print("Community Area nulos:", missing_geo.shape[0])

example = missing_geo.sample(n=10000, random_state=42)

# # Crear figura
# fig, ax = plt.subplots(figsize=(10,10))

# # Plotear puntos nulos
# ax.scatter(example['Longitude'], example['Latitude'],
#            c="red", s=5, alpha=0.6, label="Nulos")

# # Ajustar límites al área de Chicago
# ax.set_xlim(-87.95, -87.50)
# ax.set_ylim(41.60, 42.05)

# # Agregar mapa base de OpenStreetMap
# ctx.add_basemap(ax, crs="EPSG:4326")

# ax.set_title(" Registros Community Area nulos en Chicago")
# ax.legend()
# plt.show()
```

```
Community Area nulos: 604551
```

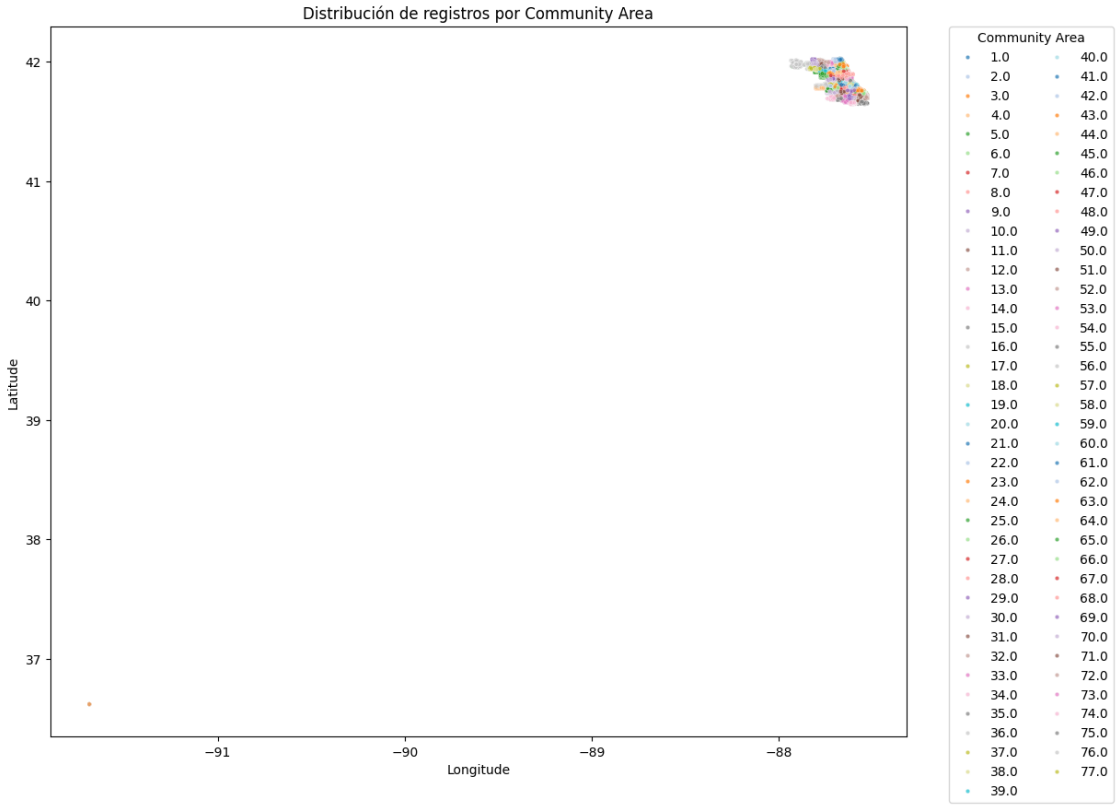
En esta etapa corresponde limpiar los valores nulos de la variable `Community Area` (los cuales como se ve en el mapa estan extendidos a lo largo de todo Chicago). Para este caso, se procederá a realizar una imputación en lugar de eliminar registros. Dado que contamos con las coordenadas geográficas

de todos los registros, se empleará un método de **KNN-classification**, el cual permitirá asignar la *Community Area* más probable en función de la proximidad espacial. Cabe destacar que los límites de las *Community Areas* han permanecido estables desde la década de 1930, por lo que esta imputación no alterará la validez de los datos ni afectará el desempeño del algoritmo.

Plan a seguir:
1. Detectar los registros con valores nulos en `Community Area`.
2. Utilizar las coordenadas (`Latitude`, `Longitude`, `X Coordinate`, `Y Coordinate`) como referencia para identificar los vecinos más cercanos.
3. Aplicar **KNN-classification** para asignar el valor de `Community Area` correspondiente.
4. Validar los resultados verificando la coherencia espacial con los límites oficiales de las *Community Areas*.

Antes visualizamos de manera general como se ve cada `Community Area` en el mapa, como son territorios con limites fijos, existe una separabilidad para que el modelo pueda clasificar exitosamente

[28]:
```
# Quiero ver los registros de distintos colores por community area en el mapa
sample_size = 1000000
df_sample = df.sample(n=sample_size, random_state=42)

# --- Scatter plot ---
plt.figure(figsize=(12, 10))
sns.scatterplot(
    x='Longitude',
    y='Latitude',
    hue='Community Area',
    palette='tab20',
    data=df_sample,
    s=10,
    alpha=0.7
)

plt.title('Distribución de registros por Community Area')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# --- Leyenda en 2 columnas ---
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., ncol=2,␣
 ↪title='Community Area')

plt.show()
```



De manera accidental, vemos que existen outliers muy extremos, los cuales pueden afectar tremendamente los modelos, ya que KNN se basa en la distancia entre puntos para predecir valores o clases. Justo me di cuenta mientras estaba por aplicar el KNN para imputar los `Community Area`, y si no los controlamos, estos outliers podrían sesgar las predicciones, generando imputaciones incorrectas para registros cercanos a estos puntos atípicos. Por ello, es recomendable **identificar y filtrar los outliers extremos en las coordenadas** antes de entrenar el KNN, garantizando que la imputación sea confiable y representativa de la distribución real de la ciudad.

Por lo tanto, de pasada se evaluara los outliers de todos los features.

Análisis de outliers

[29]:
```
#Funcion principal para detectar outliers usando el metodo del rango␣
 ↪intercuartil
def calcular_outliers( df, columna):
    # Calcular cuartiles
    Q1 = df[columna].quantile(0.25)
    Q3 = df[columna].quantile(0.75)

    # Calcular rango intercuartílico (IQR)
    IQR = Q3 - Q1

    # Calcular límites
```

```
    limite_inferior = Q1 - 1.5 * IQR
    limite_superior = Q3 + 1.5 * IQR

    # Identificar outliers
    outliers = df[(df[columna] < limite_inferior) | (df[columna] >␣
↪limite_superior)]
    porcentaje = len(outliers) / len(df) * 100

    # Mostrar resultados
    print(f"\nVariable: {columna}")
    print(f"Valor mínimo: {df[columna].min():.2f}")
    print(f"Valor máximo: {df[columna].max():.2f}")
    print(f"Q1 (Percentil 25): {Q1:.2f}")
    print(f"Q3 (Percentil 75): {Q3:.2f}")
    print(f"IQR: {IQR:.2f}")
    print(f"Límite inferior: {limite_inferior:.2f}")
    print(f"Límite superior: {limite_superior:.2f}")
    print(f"Outliers encontrados: {len(outliers)}")
    print(f"Porcentaje de outliers: {porcentaje:.2f}%\n")

    return outliers , porcentaje
```

[30]:
```
for columna in df.select_dtypes(include=[np.number]).columns:
    outliers , porcentaje = calcular_outliers(df,columna)
```

```
Variable: ID
Valor mínimo: 634.00
Valor máximo: 13978553.00
Q1 (Percentil 25): 4048688.25
Q3 (Percentil 75): 10858186.75
IQR: 6809498.50
Límite inferior: -6165559.50
Límite superior: 21072434.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Beat
Valor mínimo: 111.00
Valor máximo: 2535.00
Q1 (Percentil 25): 621.00
Q3 (Percentil 75): 1731.00
IQR: 1110.00
Límite inferior: -1044.00
Límite superior: 3396.00
Outliers encontrados: 0
Porcentaje de outliers: 0.00%
```

```
Variable: District
Valor mínimo: 1.00
Valor máximo: 31.00
Q1 (Percentil 25): 6.00
Q3 (Percentil 75): 17.00
IQR: 11.00
Límite inferior: -10.50
Límite superior: 33.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Community Area
Valor mínimo: 1.00
Valor máximo: 77.00
Q1 (Percentil 25): 23.00
Q3 (Percentil 75): 56.00
IQR: 33.00
Límite inferior: -26.50
Límite superior: 105.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: X Coordinate
Valor mínimo: 0.00
Valor máximo: 1205119.00
Q1 (Percentil 25): 1153064.00
Q3 (Percentil 75): 1176401.00
IQR: 23337.00
Límite inferior: 1118058.50
Límite superior: 1211406.50
Outliers encontrados: 40217
Porcentaje de outliers: 0.48%


Variable: Y Coordinate
Valor mínimo: 0.00
Valor máximo: 1951622.00
Q1 (Percentil 25): 1859195.00
Q3 (Percentil 75): 1909387.00
IQR: 50192.00
Límite inferior: 1783907.00
Límite superior: 1984675.00
Outliers encontrados: 149
Porcentaje de outliers: 0.00%
```

```
Variable: Year
Valor mínimo: 2001.00
Valor máximo: 2025.00
Q1 (Percentil 25): 2005.00
Q3 (Percentil 75): 2017.00
IQR: 12.00
Límite inferior: 1987.00
Límite superior: 2035.00
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Latitude
Valor mínimo: 36.62
Valor máximo: 42.02
Q1 (Percentil 25): 41.77
Q3 (Percentil 75): 41.91
IQR: 0.14
Límite inferior: 41.56
Límite superior: 42.11
Outliers encontrados: 149
Porcentaje de outliers: 0.00%


Variable: Longitude
Valor mínimo: -91.69
Valor máximo: -87.52
Q1 (Percentil 25): -87.71
Q3 (Percentil 75): -87.63
IQR: 0.09
Límite inferior: -87.84
Límite superior: -87.50
Outliers encontrados: 40213
Porcentaje de outliers: 0.48%
```

Variable	Outliers	% Outliers
ID	0	0.00%
Beat	0	0.00%
District	0	0.00%
Community Area	0	0.00%
X Coordinate	40,217	0.48%
Y Coordinate	149	0.00%
Year	0	0.00%
Latitude	149	0.00%

Variable	Outliers	% Outliers
Longitude	40,213	0.48%

Aunque en la mayoría de los casos el porcentaje es muy bajo. Por ejemplo, las coordenadas **X Coordinate** y **Longitude** muestran alrededor de 0.48% de registros atípicos, mientras que **Latitude** y **Y Coordinate** tienen muy pocos outliers (0.00% redondeado). Las demás variables como `Community Area`, `District`, `Year`, `Beat` y `ID` no presentan outliers significativos.

Estos outliers son importantes porque pueden **afectar negativamente el desempeño del KNN**, tanto si lo utilizo como clasificador para predecir `Community Area`, como si lo empleo como imputador para rellenar valores faltantes. Dado que KNN se basa en la distancia, los puntos extremadamente alejados podrían sesgar las predicciones de los registros cercanos, generando imputaciones incorrectas.

Por lo tanto, mi plan de acción será el siguiente:

1. **Filtrar los outliers de coordenadas** antes de aplicar cualquier modelo.
   - Para `Latitude`, conservaré solo los valores entre 41.60, 42.05.
   - Para `Longitude`, conservaré los valores entre -87.95, -87.50.
2. **Mantener los registros limpios** para entrenar el KNN. Esto asegura que el modelo vea solo datos representativos y evite ser influenciado por valores extremos.
3. **KNeighborsClassifier** sobre el dataset filtrado. Así, la imputación de `Community Area` será confiable y no se verá perturbada por outliers.

Con este enfoque, puedo garantizar que el modelo trabajará con datos consistentes, maximizando la precisión de la imputación y evitando errores por valores atípicos.

[31]:
```
# Filtrar outliers en las coordenadas geográficas
# Definir límites razonables para las coordenadas de Chicago
lon_min, lon_max = -87.95, -87.50
lat_min, lat_max = 41.60, 42.05

# Filtrar outliers en las coordenadas geográficas
df = df[(df['Longitude'] >= lon_min) & (df['Longitude'] <= lon_max) &
        (df['Latitude'] >= lat_min) & (df['Latitude'] <= lat_max)]

# Volver a evaluar outliers en las coordenadas
for columna in df.select_dtypes(include=[np.number]).columns:
    outliers , porcentaje = calcular_outliers(df,columna)
```

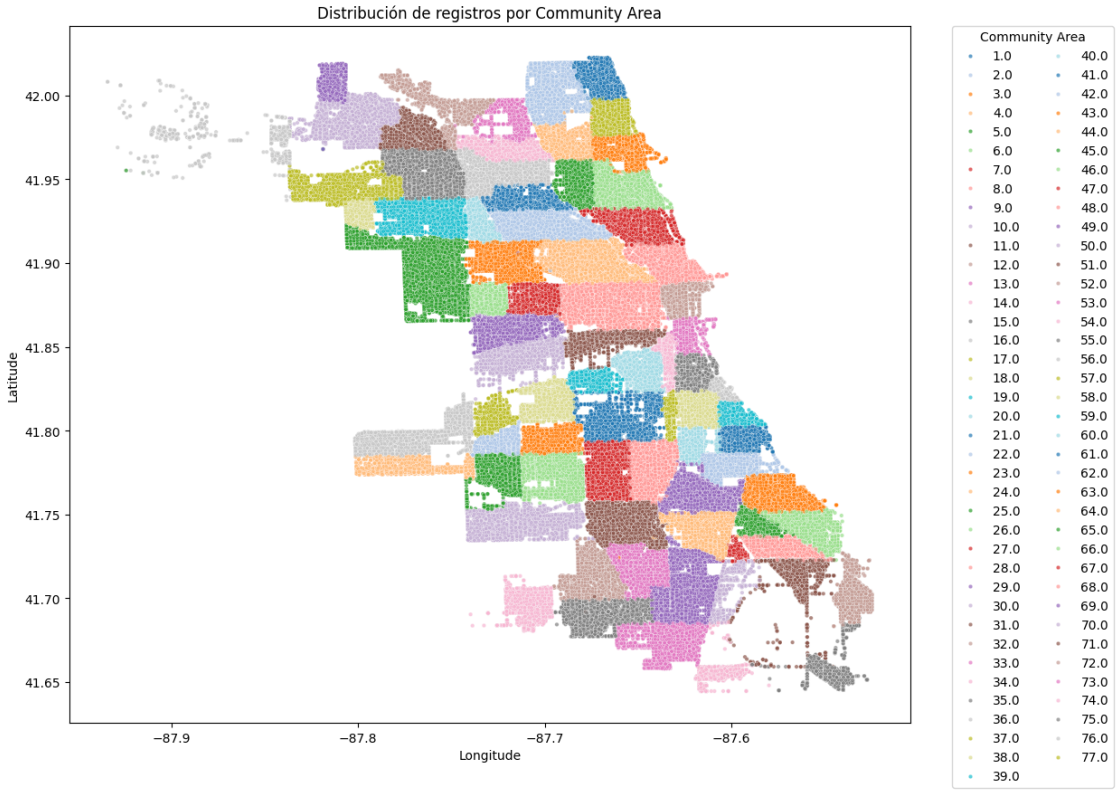
```
Variable: ID
Valor mínimo: 634.00
Valor máximo: 13978553.00
Q1 (Percentil 25): 4048688.00
Q3 (Percentil 75): 10858233.00
```

```
IQR: 6809545.00
Límite inferior: -6165629.50
Límite superior: 21072550.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Beat
Valor mínimo: 111.00
Valor máximo: 2535.00
Q1 (Percentil 25): 621.00
Q3 (Percentil 75): 1731.00
IQR: 1110.00
Límite inferior: -1044.00
Límite superior: 3396.00
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: District
Valor mínimo: 1.00
Valor máximo: 31.00
Q1 (Percentil 25): 6.00
Q3 (Percentil 75): 17.00
IQR: 11.00
Límite inferior: -10.50
Límite superior: 33.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Community Area
Valor mínimo: 1.00
Valor máximo: 77.00
Q1 (Percentil 25): 23.00
Q3 (Percentil 75): 56.00
IQR: 33.00
Límite inferior: -26.50
Límite superior: 105.50
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: X Coordinate
Valor mínimo: 1091242.00
Valor máximo: 1205119.00
Q1 (Percentil 25): 1153065.00
Q3 (Percentil 75): 1176401.00
```

```
IQR: 23336.00
Límite inferior: 1118061.00
Límite superior: 1211405.00
Outliers encontrados: 40068
Porcentaje de outliers: 0.48%


Variable: Y Coordinate
Valor mínimo: 1813894.00
Valor máximo: 1951622.00
Q1 (Percentil 25): 1859197.00
Q3 (Percentil 75): 1909387.00
IQR: 50190.00
Límite inferior: 1783912.00
Límite superior: 1984672.00
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Year
Valor mínimo: 2001.00
Valor máximo: 2025.00
Q1 (Percentil 25): 2005.00
Q3 (Percentil 75): 2017.00
IQR: 12.00
Límite inferior: 1987.00
Límite superior: 2035.00
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Latitude
Valor mínimo: 41.64
Valor máximo: 42.02
Q1 (Percentil 25): 41.77
Q3 (Percentil 75): 41.91
IQR: 0.14
Límite inferior: 41.56
Límite superior: 42.11
Outliers encontrados: 0
Porcentaje de outliers: 0.00%


Variable: Longitude
Valor mínimo: -87.94
Valor máximo: -87.52
Q1 (Percentil 25): -87.71
Q3 (Percentil 75): -87.63
```

```
IQR: 0.09
Límite inferior: -87.84
Límite superior: -87.50
Outliers encontrados: 40065
Porcentaje de outliers: 0.48%
```

[32]:
```python
# Quiero ver los registros de distintos colores por community area en el mapa
sample_size = 1000000
df_sample = df.sample(n=sample_size, random_state=42)

# --- Scatter plot ---
plt.figure(figsize=(12, 10))
sns.scatterplot(
    x='Longitude',
    y='Latitude',
    hue='Community Area',
    palette='tab20',
    data=df_sample,
    s=10,
    alpha=0.7
)

plt.title('Distribución de registros por Community Area')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# --- Leyenda en 2 columnas ---
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., ncol=2,␣
 ↪title='Community Area')

plt.show()
```



A pesar que todavia aparezcan outliers, estos se encuentran realisticamente dentro del territorio de chicago por lo tanto se quedaran ahi.

[33]:
```
# === 1. Separar entrenamiento y prueba ===
train_df = df[df['Year'] <= 2020].copy()
test_df  = df[df['Year'] >= 2021].copy()

# Eliminar filas donde Community Area sea nulo
train_df = train_df.dropna(subset=["Community Area"]).copy()
test_df  = test_df.dropna(subset=["Community Area"]).copy()

# Variables que usaremos
X_train = train_df[["Latitude", "Longitude"]].values
y_train = train_df["Community Area"].astype(int).values

X_test = test_df[["Latitude", "Longitude"]].values
y_test = test_df["Community Area"].astype(int).values

# === 2. Entrenar KNN como clasificador ===
knn = KNeighborsClassifier(n_neighbors=5, weights="distance", n_jobs=-1)
knn.fit(X_train, y_train)
```

```python
# === 3. Predecir en prueba ===
y_pred = knn.predict(X_test)

# === 4. Evaluar desempeño ===
acc = accuracy_score(y_test, y_pred)
f1  = f1_score(y_test, y_pred, average="weighted")

print("=== Evaluación del modelo KNN ===")
print(f"Accuracy en prueba (2021-2025): {acc:.4f}")
print(f"F1-weighted en prueba (2021-2025): {f1:.4f}")
print("\n--- Classification Report ---")
print(classification_report(y_test, y_pred, digits=4))
```

```
=== Evaluación del modelo KNN ===
Accuracy en prueba (2021-2025): 0.9932
F1-weighted en prueba (2021-2025): 0.9932

--- Classification Report ---
              precision    recall  f1-score   support

           1     0.9952    0.9977    0.9964     18841
           2     0.9957    0.9942    0.9949     16815
           3     0.9935    0.9978    0.9956     18803
           4     0.9934    0.9921    0.9928      9140
           5     0.9961    0.9942    0.9952      5907
           6     0.9976    0.9939    0.9958     27627
           7     0.9863    0.9916    0.9889     17075
           8     0.9978    0.9944    0.9961     48427
           9     0.9936    0.9850    0.9893      1267
          10     0.9942    0.9940    0.9941      5704
          11     0.9897    0.9916    0.9906      4634
          12     0.9901    0.9967    0.9933      2398
          13     0.9479    0.9942    0.9705      4320
          14     0.9860    0.9724    0.9791     10634
          15     0.9877    0.9923    0.9900     14246
          16     0.9902    0.9818    0.9860     11857
          17     0.9885    0.9915    0.9900      7433
          18     0.9923    0.9896    0.9910      2992
          19     0.9703    0.9929    0.9815     19991
          20     0.9917    0.8920    0.9392      6409
          21     0.9832    0.9787    0.9809      9619
          22     0.9898    0.9922    0.9910     21233
          23     0.9927    0.9968    0.9947     30350
          24     0.9965    0.9959    0.9962     33095
          25     0.9996    0.9997    0.9997     57604
          26     0.9962    0.9977    0.9969     18071
          27     0.9946    0.9954    0.9950     18000
          28     0.9980    0.9976    0.9978     44397
```

29	0.9968	0.9967	0.9967	30942
30	0.9979	0.9974	0.9977	18347
31	0.9968	0.9961	0.9964	12989
32	0.9941	0.9988	0.9964	37175
33	0.9948	0.9910	0.9929	11264
34	0.9927	0.9943	0.9935	5087
35	0.9959	0.9945	0.9952	13054
36	0.9872	0.9686	0.9778	3754
37	0.9968	0.9961	0.9964	3086
38	0.9866	0.9846	0.9856	16099
39	0.9711	0.9876	0.9793	8615
40	0.9812	0.9820	0.9816	10959
41	0.9961	0.9919	0.9940	10350
42	0.9835	0.9768	0.9802	16036
43	0.9956	0.9969	0.9962	38524
44	0.9848	0.9921	0.9885	27283
45	0.9846	0.9941	0.9893	5269
46	0.9972	0.9914	0.9943	18935
47	0.9840	0.9899	0.9869	1489
48	0.9933	0.9941	0.9937	6451
49	0.9940	0.9943	0.9942	26591
50	0.9967	0.9944	0.9955	4832
51	0.9842	0.9980	0.9911	8442
52	0.9990	0.9998	0.9994	5892
53	0.9960	0.9952	0.9956	16397
54	0.9993	0.9982	0.9987	5602
55	0.9984	0.9613	0.9795	3330
56	0.9984	0.9977	0.9981	9542
57	0.9972	0.9996	0.9984	4599
58	0.9966	0.9975	0.9971	9746
59	0.9976	0.9971	0.9974	4168
60	0.9978	0.9992	0.9985	7220
61	0.9978	0.9975	0.9977	18354
62	0.9806	0.9933	0.9869	4331
63	0.9922	0.9948	0.9935	10013
64	0.9950	0.9966	0.9958	4989
65	0.9957	0.9890	0.9923	7899
66	0.9967	0.9964	0.9966	23289
67	0.9931	0.9950	0.9941	21999
68	0.9921	0.9913	0.9917	23181
69	0.9846	0.9805	0.9826	28996
70	0.9989	0.9984	0.9986	9934
71	0.9976	0.9973	0.9974	29905
72	0.9945	0.9835	0.9890	4232
73	0.9919	0.9933	0.9926	13633
74	1.0000	1.0000	1.0000	2455
75	0.9883	0.9928	0.9906	8715
76	0.9988	0.9988	0.9988	7668

```
           77     0.9978    0.9943    0.9961     14253

    accuracy                         0.9932   1122804
   macro avg     0.9921    0.9912    0.9916   1122804
weighted avg     0.9932    0.9932    0.9932   1122804
```

Al revisar los resultados de la evaluación del modelo KNN para predecir `Community Area` en los años 2021-2025, se puede afirmar que el desempeño fue **excelente**. Obtuvimos un **accuracy de 0.9932** y un **F1-weighted de 0.9932**, lo que indica que las predicciones del modelo son altamente precisas y consistentes en todas las clases.

Al observar el **classification report**, noto que la mayoría de las clases presentan precisiones y recalls superiores al 0.99, lo que demuestra que el modelo identifica correctamente la `Community Area` en casi todos los registros. Incluso las clases con menos registros, como la 9 y la 13, muestran métricas muy altas, lo que me da confianza en la robustez del modelo frente a desequilibrios en la frecuencia de las clases.

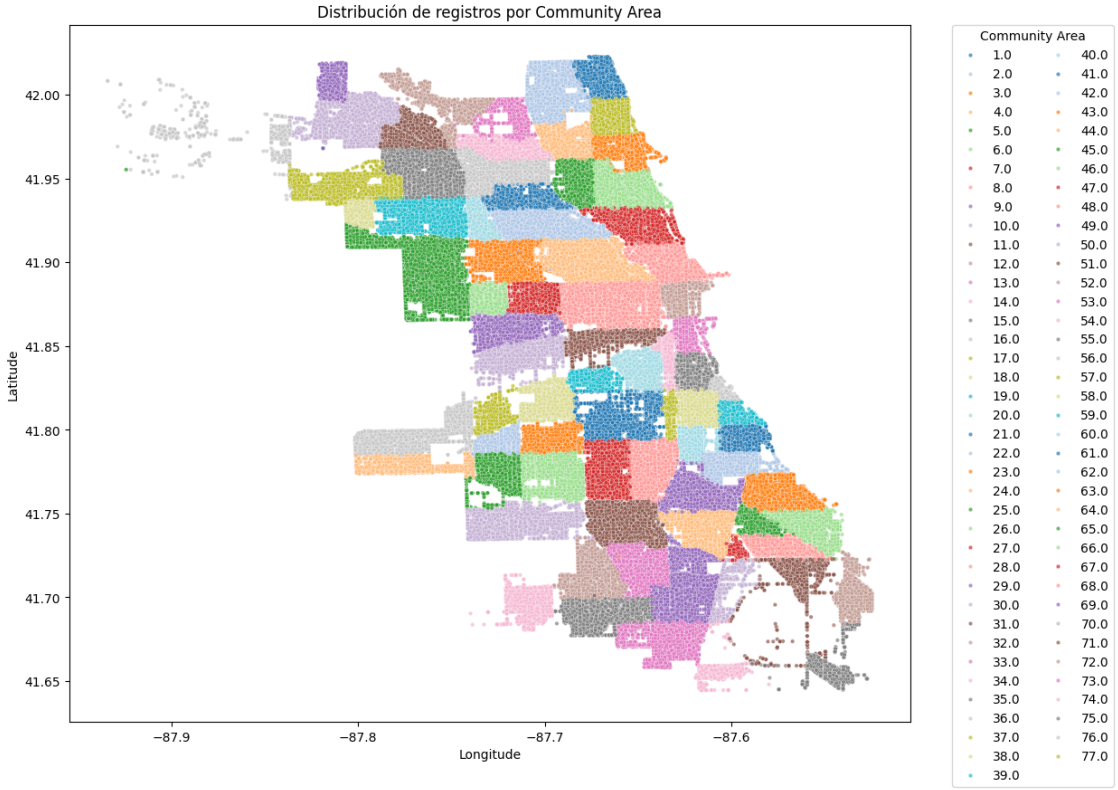
Estos resultados me permiten estar seguro de que el KNN es una **herramienta confiable** para imputar los valores faltantes de `Community Area` en el dataset completo. Gracias a esta alta precisión, puedo proceder con la imputación sabiendo que los registros predichos serán correctos en la gran mayoría de los casos, preservando la integridad y la calidad de los datos.

[34]:
```
# Empezar a imputar los nulos en Community Area
missing_mask = df['Community Area'].isna()
X_missing = df.loc[missing_mask, ['Latitude', 'Longitude']].values

## Predecir Community Area para los registros con nulos
imputed_areas = knn.predict(X_missing)
df.loc[missing_mask, 'Community Area'] = imputed_areas
# Verificar que ya no hay nulos en Community Area
print(f'\nCantidad de nulos en Community Area después de la imputación:␣
 ↪{df["Community Area"].isnull().sum()}')
```

```
Cantidad de nulos en Community Area después de la imputación: 0
```

[35]:
```
# Quiero ver los registros de distintos colores por community area en el mapa
sample_size = 1000000
df_sample = df.sample(n=sample_size, random_state=42)

# --- Scatter plot ---
plt.figure(figsize=(12, 10))
sns.scatterplot(
    x='Longitude',
    y='Latitude',
    hue='Community Area',
    palette='tab20',
    data=df_sample,
```

```
    s=10,
    alpha=0.7
)

plt.title('Distribución de registros por Community Area')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# --- Leyenda en 2 columnas ---
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., ncol=2,␣
 ↪title='Community Area')

plt.show()
```



Por ultima vez verificamos la cantidad de nulos

[36]:
```
# cantidad registros y porcentaje con 1,2,3,... nulos
null_record_counts = df.isnull().sum(axis=1).value_counts().sort_index()
null_record_percent = (null_record_counts / total_records) * 100
null_record_df = pd.DataFrame({'Record Count': null_record_counts, 'Record␣
 ↪Percentage': null_record_percent})
print('\nRegistros con n nulos (Tabla 6):')
```

```
print(null_record_df[null_record_df['Record Count'] > 0])
```

```
Registros con n nulos (Tabla 6):
   Record Count  Record Percentage
0       8305925          98.768381
```

Al finalizar el proceso de limpieza e imputación, podemos confirmar que **ya no existen registros con valores faltantes** en la variable `Community Area`. Gracias al uso del KNN para predecir los datos ausentes basándonos en las coordenadas, logramos mantener la integridad de la información sin perder precisión ni consistencia en las imputaciones.

Cabe destacar que, durante este proceso, también se eliminó una columna que contenía 7% de valores nulos y la informacion no era relevante para los objetivos planteados, contribuyendo a una base de datos más limpia y manejable. A pesar de estas modificaciones, hemos logrado **conservar el 98.768381% de la data original**, lo que representa **8,305,925 registros** sobre los 8,409,496 iniciales.

Este resultado asegura que la base de datos se encuentra lista para análisis posteriores o para alimentar modelos de machine learning, manteniendo la mayor cantidad posible de información útil y reduciendo al mínimo la pérdida de registros.

2.2 Codificación de categóricas

Para este apartado antes de definir el metodo de codificación, se realizara un analisis superficial de las variables categoricas

- **ID**: identificador único del incidente.
- **Case Number**: número de caso de la Policía de Chicago.
- **Date**: fecha y hora en que ocurrió el delito.
- **Primary Type**: categoría principal del delito (ejemplo: robo, asalto, fraude).
- **Description**: detalle secundario del delito.
- **Location Description**: lugar donde ocurrió (ejemplo: calle, residencia, comercio).
- **Arrest**: indica si el crimen resultó en arresto.
- **Domestic**: señala si estuvo vinculado a violencia doméstica.
- **District, Ward, Community Area, Beat**: divisiones administrativas y policiales.
- **Latitude, Longitude, X Coordinate, Y Coordinate**: localización aproximada.
- **Year**: año del incidente.

[37]:
```
# variables que no sean numericas
categorical_cols = df.select_dtypes(exclude=['number']).columns
print("\nVariables categóricas en el DataFrame:")
print(categorical_cols)
df.info()
df.head()
```

```
Variables categóricas en el DataFrame:
Index(['Case Number', 'Date', 'Block', 'IUCR', 'Primary Type', 'Description',
       'Location Description', 'Arrest', 'Domestic', 'FBI Code', 'Updated On',
       'Location'],
```

```
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
Index: 8305925 entries, 0 to 8409495
Data columns (total 21 columns):
 #   Column                Dtype
---  ------                -----
 0   ID                    int64
 1   Case Number           object
 2   Date                  datetime64[ns]
 3   Block                 object
 4   IUCR                  object
 5   Primary Type          object
 6   Description           object
 7   Location Description  object
 8   Arrest                bool
 9   Domestic              bool
 10  Beat                  int64
 11  District              float64
 12  Community Area        float64
 13  FBI Code              object
 14  X Coordinate          float64
 15  Y Coordinate          float64
 16  Year                  int64
 17  Updated On            object
 18  Latitude              float64
 19  Longitude             float64
 20  Location              object
dtypes: bool(2), datetime64[ns](1), float64(6), int64(3), object(9)
memory usage: 1.3+ GB
```

[37]:

	ID	Case Number	Date	Block	IUCR	\
0	3734260	HL103944	2004-12-31 23:59:00	008XX N GREENVIEW AVE	0820	
3	3834225	HL205200	2004-12-31 23:59:00	009XX E 95TH ST	0810	
4	3744726	HL111400	2004-12-31 23:59:00	115XX S JUSTINE ST	0820	
5	3735526	HL102499	2004-12-31 23:59:00	015XX S SPRINGFIELD AVE	0810	
6	3732853	HL103478	2004-12-31 23:59:00	062XX N MOZART ST	0910	

	Primary Type	Description	Location Description	Arrest	Domestic	\
0	THEFT	$500 AND UNDER	CHA APARTMENT	False	False	
3	THEFT	OVER $500	OTHER	False	False	
4	THEFT	$500 AND UNDER	STREET	False	False	
5	THEFT	OVER $500	STREET	False	False	
6	MOTOR VEHICLE THEFT	AUTOMOBILE	STREET	False	False	

	…	District	Community Area	FBI Code	X Coordinate	Y Coordinate	Year	\
0	…	12.0	24.0	06	1166220.0	1905912.0	2004	
3	…	5.0	50.0	06	1184179.0	1842178.0	2004	

```
4  …       5.0            53.0        06    1168072.0     1828211.0  2004
5  …      10.0            29.0        06    1150631.0     1892160.0  2004
6  …      24.0             2.0        07    1156207.0     1941335.0  2004

              Updated On   Latitude  Longitude                      Location
0  02/28/2018 03:56:25 PM  41.897402 -87.664939  (41.897401921, -87.664939363)
3  02/10/2018 03:50:01 PM  41.722108 -87.600974  (41.722108211, -87.600973936)
4  02/10/2018 03:50:01 PM  41.684141 -87.660371  (41.684141387, -87.660371122)
5  02/28/2018 03:56:25 PM  41.859984 -87.722555  (41.859983911, -87.722555478)
6  02/28/2018 03:56:25 PM  41.994813 -87.700756   (41.99481317, -87.700755694)

[5 rows x 21 columns]
```

Análisis

Columnas que ya son numéricas o booleanas no requieren cambios

- `ID` → int64
- `Beat` → int64
- `District` → float64
- `Community Area` → float64
- `Year` → int64
- `Latitude` → float64
- `Longitude` → float64
- `Arrest` → bool → ya usable
- `Domestic` → bool → ya usable

Columnas **categóricas u objetos** que se deben codificar

1. **`Case Number`**
   - Es un identificador único → **no útil para ML**, se descarta.
2. **`Date`**
   - Convertir a variables numéricas temporales:
     - `Year` (ya tienes)
     - `Month`, `Day`, `Weekday`, `Hour` para obtener patrones temporales.
3. **`Block`**
   - Texto de ubicación → opcional a **codificar** (no le veo mucho uso).
4. **`IUCR`**
   - Código del crimen → **codificar** según número de categorías.
5. **`Primary Type`**
   - Tipo de delito → **codificar** recomendado, interpretativo y útil para ML.
6. **`Description`**
   - Texto → **codificar** según número de categorías

7. `Location Description`
   - Tipo de lugar → **codificar**, útil para ML.
8. `FBI Code`
   - Código de crimen → **codificar** según número de categorías.
9. `Updated On`
   - Fecha de actualización → se descarta, no aporta valor predictivo directo.
10. `Location`
    - Texto con lat/lon → ya tengo `Latitude` y `Longitude`, se puede descartar.

Resumen práctico

Columnas	Acción recomendada
ID	Eliminar para ML
Case Number	Eliminar para ML
Date	Extraer features: Year, Month, Day, Weekday, Hour
Block	Opcional
IUCR	Codificar
Primary Type	Codificar
Description	Codificar
Location Description	Codificar
Arrest	Mantener (bool)
Domestic	Mantener (bool)
Beat	Mantener (numérica)
District	Mantener (numérica)
Community Area	Codificar
FBI Code	Codificar
Latitude	Mantener
Longitude	Mantener
Location	Eliminar para ML
Updated On	Extraer features: Year, Month, Day, Weekday, Hour

Extraer features de Date

[38]:
```
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Weekday'] = df['Date'].dt.weekday
df['Hour'] = df['Date'].dt.hour
```

Extraer features de Updated On

[39]:
```
# Pasar Updated On a datetime 02/28/2018 03:56:25 PM
df['Updated On'] = pd.to_datetime(df['Updated On'], errors='coerce')
df['Updated On'].head()
```

```
/var/folders/rn/ysxq396j7lbg3qnwvfpvfsfm0000gn/T/ipykernel_69729/267114713.py:2:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  df['Updated On'] = pd.to_datetime(df['Updated On'], errors='coerce')
```

[39]:
```
0   2018-02-28 15:56:25
3   2018-02-10 15:50:01
4   2018-02-10 15:50:01
5   2018-02-28 15:56:25
6   2018-02-28 15:56:25
Name: Updated On, dtype: datetime64[ns]
```

[40]:
```
df['Year_U'] = df['Updated On'].dt.year
df['Month_U'] = df['Updated On'].dt.month
df['Day_U'] = df['Updated On'].dt.day
df['Weekday_U'] = df['Updated On'].dt.weekday
df['Hour_U'] = df['Updated On'].dt.hour
```

Generar un df_ML listo para modelar

[41]:
```
df_ML = df.copy()
```

Booleanos a enteros

[42]:
```
df_ML['Arrest'] = df_ML['Arrest'].astype(int)
df_ML['Domestic'] = df_ML['Domestic'].astype(int)
```

Cantidad de valores unicos

[43]:
```
#Cantidad de valores unicos por columna
unique_counts = df_ML.select_dtypes(exclude=['number']).nunique()
print("\nCantidad de valores únicos por columna:")
print(unique_counts)
```

```
Cantidad de valores únicos por columna:
Case Number             8305322
Date                    3490841
Block                     62986
IUCR                        416
Primary Type                 34
Description                 564
Location Description        218
FBI Code                     26
Updated On                 5730
Location                 908682
dtype: int64
```

Tipos de valores unicos

[44]:
```
# Columnas categóricas a analizar
categorical_cols = [
    "Case Number",
    "Block",
    "IUCR",
    "Primary Type",
    "Description",
    "Location Description",
    "FBI Code"
]

# Mostrar la cantidad de valores únicos y los primeros 10 valores de cada␣
 ↪columna
for col in categorical_cols:
    unique_values = df[col].unique()
    print(f"\nColumna: {col}")
    print(f"Número de valores únicos: {len(unique_values)}")
    print(f"Primeros 10 valores únicos: {unique_values[:10]}")
```

```
Columna: Case Number
Número de valores únicos: 8305322
Primeros 10 valores únicos: ['HL103944' 'HL205200' 'HL111400' 'HL102499'
'HL103478' 'HL100353'
 'HL100158' 'HL101304' 'HL100015' 'HL100008']

Columna: Block
Número de valores únicos: 62986
Primeros 10 valores únicos: ['008XX N GREENVIEW AVE' '009XX E 95TH ST' '115XX S
JUSTINE ST'
 '015XX S SPRINGFIELD AVE' '062XX N MOZART ST' '056XX W ADDISON ST'
 '003XX E OHIO ST' '006XX E GRAND AVE' '107XX S INDIANA AVE'
 '090XX S COMMERCIAL AVE']

Columna: IUCR
Número de valores únicos: 416
Primeros 10 valores únicos: ['0820' '0810' '0910' '1305' '0486' '0870' '1477'
'143A' '1320' '0460']

Columna: Primary Type
Número de valores únicos: 34
Primeros 10 valores únicos: ['THEFT' 'MOTOR VEHICLE THEFT' 'CRIMINAL DAMAGE'
'BATTERY'
 'WEAPONS VIOLATION' 'NARCOTICS' 'BURGLARY' 'ROBBERY' 'OTHER OFFENSE'
 'DECEPTIVE PRACTICE']

Columna: Description
Número de valores únicos: 564
```

```
Primeros 10 valores únicos: ['$500 AND UNDER' 'OVER $500' 'AUTOMOBILE' 'CRIMINAL
DEFACEMENT'
 'DOMESTIC BATTERY SIMPLE' 'POCKET-PICKING' 'RECKLESS FIREARM DISCHARGE'
 'UNLAWFUL POSS OF HANDGUN' 'TO VEHICLE' 'SIMPLE']

Columna: Location Description
Número de valores únicos: 218
Primeros 10 valores únicos: ['CHA APARTMENT' 'OTHER' 'STREET' 'HOSPITAL
BUILDING/GROUNDS'
 'HOTEL/MOTEL' 'SIDEWALK' 'ALLEY' 'RESIDENCE' 'APARTMENT' 'RESTAURANT']

Columna: FBI Code
Número de valores únicos: 26
Primeros 10 valores únicos: ['06' '07' '14' '08B' '15' '18' '05' '03' '08A'
'26']
```

`Location` es lo mismo que `Latitud` y `Longitud`, pero unidas, por lo que tener información repetida, es innecesaria y se procede a eliminar dicha columna.

[45]:
```python
# Eliminar columna Location
df_ML = df_ML.drop(columns=['Location'])
print('\nColumnas del DataFrame después de eliminar Location:\n', df_ML.columns)
```

```
Columnas del DataFrame después de eliminar Location:
 Index(['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Primary Type',
       'Description', 'Location Description', 'Arrest', 'Domestic', 'Beat',
       'District', 'Community Area', 'FBI Code', 'X Coordinate',
       'Y Coordinate', 'Year', 'Updated On', 'Latitude', 'Longitude', 'Month',
       'Day', 'Weekday', 'Hour', 'Year_U', 'Month_U', 'Day_U', 'Weekday_U',
       'Hour_U'],
      dtype='object')
```

Resumen de análisis:

Columna	Valores únicos	Recomendación	Justificación
Case Number	8,305,322	No codificar	Es único por registro, no aporta información al modelo.
Block	62,986	No codificar	Demasiadas categorías; One-Hot generaría columnas enormes y consumiría mucha memoria.
IUCR	416	Label Encoding	Muchos valores para One-Hot, mejor Label Encoding.
Primary Type	34	One-Hot	Pocas categorías, One-Hot es viable, sobre todo para mantener la informacion y evitar relaciones implícitas entre categorías.
Description	564	Label Encoding	Demasiadas categorías para One-Hot, Label Encoding más práctico.

Columna	Valores únicos	Recomendación	Justificación
Location Description	218	Label Encoding	Podría hacerse One-Hot por informacón, pero Label Encoding es más eficiente en memoria.
FBI Code	26	Label Encoding	Pocas categorías, One-Hot, Sin embargo, con 8.3M de registros aumentar la dimensionalidad saldra caro, por ende Label coding es mejor opcion.
Updated On	5,730	No codificar directamente	Se puede extraer información temporal (día, mes, año, hora) y luego codificar esas nuevas variables si se desea.

Conclusión rápida

- **One-Hot Encoding:** `Primary Type` (pocas categorías, nominal, evita relaciones implícitas).
- **Label Encoding:** `Block`, `IUCR`, `Description`, `Location Description`, `FBI Code` (muchas categorías o dataset muy grande, evita explosión de dimensionalidad).
- **No codificar directamente:** `Case Number`, `Updated On` , `Block` (extraer información temporal si se desea).

One-Hot Encoding

[46]:
```
# Seleccionamos las columnas para One-Hot
one_hot_cols = ["Primary Type"]

# Aplicar One-Hot Encoding
df_ML = pd.get_dummies(df_ML, columns=one_hot_cols, prefix=one_hot_cols)

# Verificar nuevas columnas
print(df_ML.columns)
```

```
Index(['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Description',
       'Location Description', 'Arrest', 'Domestic', 'Beat', 'District',
       'Community Area', 'FBI Code', 'X Coordinate', 'Y Coordinate', 'Year',
       'Updated On', 'Latitude', 'Longitude', 'Month', 'Day', 'Weekday',
       'Hour', 'Year_U', 'Month_U', 'Day_U', 'Weekday_U', 'Hour_U',
       'Primary Type_ARSON', 'Primary Type_ASSAULT', 'Primary Type_BATTERY',
       'Primary Type_BURGLARY',
       'Primary Type_CONCEALED CARRY LICENSE VIOLATION',
       'Primary Type_CRIM SEXUAL ASSAULT', 'Primary Type_CRIMINAL DAMAGE',
       'Primary Type_CRIMINAL SEXUAL ASSAULT',
       'Primary Type_CRIMINAL TRESPASS', 'Primary Type_DECEPTIVE PRACTICE',
       'Primary Type_DOMESTIC VIOLENCE', 'Primary Type_GAMBLING',
       'Primary Type_HOMICIDE', 'Primary Type_HUMAN TRAFFICKING',
       'Primary Type_INTERFERENCE WITH PUBLIC OFFICER',
       'Primary Type_INTIMIDATION', 'Primary Type_KIDNAPPING',
       'Primary Type_LIQUOR LAW VIOLATION', 'Primary Type_MOTOR VEHICLE THEFT',
       'Primary Type_NARCOTICS', 'Primary Type_NON-CRIMINAL',
       'Primary Type_OBSCENITY', 'Primary Type_OFFENSE INVOLVING CHILDREN',
```

```
       'Primary Type_OTHER NARCOTIC VIOLATION', 'Primary Type_OTHER OFFENSE',
       'Primary Type_PROSTITUTION', 'Primary Type_PUBLIC INDECENCY',
       'Primary Type_PUBLIC PEACE VIOLATION', 'Primary Type_RITUALISM',
       'Primary Type_ROBBERY', 'Primary Type_SEX OFFENSE',
       'Primary Type_STALKING', 'Primary Type_THEFT',
       'Primary Type_WEAPONS VIOLATION'],
      dtype='object')
```

Label Encoding

[47]:
```
# Columnas a codificar
label_cols = ["Block", "IUCR", "Description", "Location Description", "FBI␣
 ↪Code"]

# Diccionario para guardar la correspondencia
label_mapping = {}

for col in label_cols:
    le = LabelEncoder()
    df_ML[col] = le.fit_transform(df_ML[col])

    # Guardamos el mapping en el diccionario
    mapping = {i: label for i, label in enumerate(le.classes_)}
    label_mapping[col] = mapping

# Ejemplo: ver correspondencia de 'IUCR'
print("IUCR mapping:", label_mapping["IUCR"])
```

```
IUCR mapping: {0: '0110', 1: '0130', 2: '0141', 3: '0142', 4: '0261', 5: '0262',
6: '0263', 7: '0264', 8: '0265', 9: '0266', 10: '0271', 11: '0272', 12: '0273',
13: '0274', 14: '0275', 15: '0281', 16: '0291', 17: '0312', 18: '0313', 19:
'031A', 20: '031B', 21: '0320', 22: '0325', 23: '0326', 24: '0330', 25: '0331',
26: '0334', 27: '0337', 28: '033A', 29: '033B', 30: '0340', 31: '041A', 32:
'041B', 33: '0420', 34: '0430', 35: '0440', 36: '0450', 37: '0451', 38: '0452',
39: '0453', 40: '0454', 41: '0460', 42: '0461', 43: '0462', 44: '0470', 45:
'0475', 46: '0479', 47: '0480', 48: '0481', 49: '0482', 50: '0483', 51: '0484',
52: '0485', 53: '0486', 54: '0487', 55: '0488', 56: '0489', 57: '0490', 58:
'0492', 59: '0493', 60: '0494', 61: '0495', 62: '0496', 63: '0497', 64: '0498',
65: '0499', 66: '0510', 67: '051A', 68: '051B', 69: '0520', 70: '0530', 71:
'0545', 72: '0550', 73: '0551', 74: '0552', 75: '0553', 76: '0554', 77: '0555',
78: '0556', 79: '0557', 80: '0558', 81: '0560', 82: '0580', 83: '0581', 84:
'0583', 85: '0584', 86: '0610', 87: '0620', 88: '0630', 89: '0650', 90: '0710',
91: '0760', 92: '0810', 93: '0820', 94: '0830', 95: '0840', 96: '0841', 97:
'0842', 98: '0843', 99: '0850', 100: '0860', 101: '0865', 102: '0870', 103:
'0880', 104: '0890', 105: '0895', 106: '0910', 107: '0915', 108: '0917', 109:
'0918', 110: '0920', 111: '0925', 112: '0927', 113: '0928', 114: '0930', 115:
'0935', 116: '0937', 117: '0938', 118: '1010', 119: '1020', 120: '1025', 121:
'1030', 122: '1035', 123: '1050', 124: '1055', 125: '1090', 126: '1101', 127:
```

'1102', 128: '1110', 129: '1120', 130: '1121', 131: '1122', 132: '1130', 133:
'1135', 134: '1140', 135: '1145', 136: '1147', 137: '1150', 138: '1151', 139:
'1152', 140: '1153', 141: '1154', 142: '1155', 143: '1156', 144: '1160', 145:
'1170', 146: '1185', 147: '1187', 148: '1192', 149: '1195', 150: '1197', 151:
'1199', 152: '1200', 153: '1205', 154: '1206', 155: '1210', 156: '1220', 157:
'1230', 158: '1235', 159: '1240', 160: '1241', 161: '1242', 162: '1245', 163:
'1255', 164: '1260', 165: '1261', 166: '1262', 167: '1263', 168: '1265', 169:
'1305', 170: '1310', 171: '1320', 172: '1330', 173: '1335', 174: '1340', 175:
'1345', 176: '1350', 177: '1360', 178: '1365', 179: '1370', 180: '1375', 181:
'141A', 182: '141B', 183: '141C', 184: '142A', 185: '142B', 186: '1435', 187:
'143A', 188: '143B', 189: '143C', 190: '1440', 191: '1450', 192: '1460', 193:
'1476', 194: '1477', 195: '1478', 196: '1479', 197: '1480', 198: '1481', 199:
'1504', 200: '1505', 201: '1506', 202: '1507', 203: '1510', 204: '1511', 205:
'1512', 206: '1513', 207: '1515', 208: '1518', 209: '1519', 210: '1520', 211:
'1521', 212: '1525', 213: '1526', 214: '1530', 215: '1531', 216: '1535', 217:
'1536', 218: '1537', 219: '1540', 220: '1541', 221: '1544', 222: '1549', 223:
'1562', 224: '1563', 225: '1564', 226: '1565', 227: '1566', 228: '1570', 229:
'1572', 230: '1573', 231: '1574', 232: '1576', 233: '1577', 234: '1578', 235:
'1580', 236: '1581', 237: '1582', 238: '1585', 239: '1590', 240: '1599', 241:
'1610', 242: '1611', 243: '1620', 244: '1621', 245: '1622', 246: '1625', 247:
'1626', 248: '1627', 249: '1630', 250: '1631', 251: '1633', 252: '1640', 253:
'1650', 254: '1651', 255: '1661', 256: '1670', 257: '1680', 258: '1681', 259:
'1682', 260: '1697', 261: '1710', 262: '1715', 263: '1720', 264: '1725', 265:
'1726', 266: '1750', 267: '1751', 268: '1752', 269: '1753', 270: '1754', 271:
'1755', 272: '1780', 273: '1790', 274: '1791', 275: '1792', 276: '1811', 277:
'1812', 278: '1821', 279: '1822', 280: '1840', 281: '1850', 282: '1860', 283:
'1900', 284: '2010', 285: '2011', 286: '2012', 287: '2013', 288: '2014', 289:
'2015', 290: '2016', 291: '2017', 292: '2018', 293: '2019', 294: '2020', 295:
'2021', 296: '2022', 297: '2023', 298: '2024', 299: '2025', 300: '2026', 301:
'2027', 302: '2028', 303: '2029', 304: '2030', 305: '2031', 306: '2032', 307:
'2033', 308: '2034', 309: '2040', 310: '2050', 311: '2060', 312: '2070', 313:
'2080', 314: '2090', 315: '2091', 316: '2092', 317: '2093', 318: '2094', 319:
'2095', 320: '2110', 321: '2111', 322: '2120', 323: '2160', 324: '2170', 325:
'2210', 326: '2220', 327: '2230', 328: '2240', 329: '2250', 330: '2251', 331:
'2820', 332: '2825', 333: '2826', 334: '2830', 335: '2840', 336: '2850', 337:
'2851', 338: '2860', 339: '2870', 340: '2890', 341: '2895', 342: '2896', 343:
'2900', 344: '3000', 345: '3100', 346: '3200', 347: '3300', 348: '3400', 349:
'3610', 350: '3710', 351: '3720', 352: '3730', 353: '3731', 354: '3740', 355:
'3750', 356: '3751', 357: '3760', 358: '3770', 359: '3800', 360: '3910', 361:
'3920', 362: '3960', 363: '3961', 364: '3966', 365: '3970', 366: '3975', 367:
'3980', 368: '4210', 369: '4220', 370: '4230', 371: '4240', 372: '4255', 373:
'4310', 374: '4386', 375: '4387', 376: '4388', 377: '4389', 378: '4510', 379:
'4625', 380: '4650', 381: '4651', 382: '4652', 383: '4740', 384: '4750', 385:
'4800', 386: '4810', 387: '4860', 388: '5000', 389: '5001', 390: '5002', 391:
'5003', 392: '5004', 393: '5005', 394: '5007', 395: '5008', 396: '5009', 397:
'500E', 398: '500N', 399: '5011', 400: '5013', 401: '501A', 402: '501H', 403:
'502P', 404: '502R', 405: '502T', 406: '5110', 407: '5111', 408: '5112', 409:
'5120', 410: '5121', 411: '5122', 412: '5130', 413: '5131', 414: '5132', 415:

```
'9901'}
```

Ya he codificado las columnas categóricas que seleccioné según su número de valores únicos. Para aquellas con pocas categorías, como Primary Type, utilicé One-Hot Encoding, mientras que para las que tienen muchas categorías, como Block, IUCR, Description, Location Description y FBI Code, apliqué Label Encoding.

Para asegurarme de no perder la referencia de qué número representa cada valor original, creé un diccionario por cada columna codificada. En él, cada clave es el número asignado por el Label Encoder y el valor es la categoría original. Esto me permite revisar en cualquier momento cuál es la correspondencia, ya sea para interpretación de resultados, análisis posterior o depuración de datos. De esta forma, aunque los datos estén completamente numéricos y listos para modelos de Machine Learning, siempre puedo consultar el diccionario y saber exactamente qué representa cada código.

[48]:
```
# ejemplo de uso del diccionario
print(label_mapping["Description"][3])
```

```
ABUSE/NEGLECT: CARE FACILITY
```

2.3 Escalado de numéricas

[116]:
```
# Normalizar Latitude y Longitude entre 0 y 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_ML[['Latitude_norm', 'Longitude_norm']] = scaler.
 ↪fit_transform(df_ML[['Latitude', 'Longitude']])
df_ML[['Latitude_norm', 'Longitude_norm']].head()
```

[116]:

	Latitude_norm	Longitude_norm
0	0.668252	0.661829
3	0.204911	0.815887
4	0.104556	0.672831
5	0.569348	0.523063
6	0.925733	0.575566

[50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8305925 entries, 0 to 8409495
Data columns (total 30 columns):
 #   Column                Dtype
---  ------                -----
 0   ID                    int64
 1   Case Number           object
 2   Date                  datetime64[ns]
 3   Block                 object
 4   IUCR                  object
 5   Primary Type          object
 6   Description           object
```

```
 7   Location Description  object
 8   Arrest                bool
 9   Domestic              bool
 10  Beat                  int64
 11  District              float64
 12  Community Area        float64
 13  FBI Code              object
 14  X Coordinate          float64
 15  Y Coordinate          float64
 16  Year                  int32
 17  Updated On            datetime64[ns]
 18  Latitude              float64
 19  Longitude             float64
 20  Location              object
 21  Month                 int32
 22  Day                   int32
 23  Weekday               int32
 24  Hour                  int32
 25  Year_U                int32
 26  Month_U               int32
 27  Day_U                 int32
 28  Weekday_U             int32
 29  Hour_U                int32
dtypes: bool(2), datetime64[ns](2), float64(6), int32(10), int64(2), object(8)
memory usage: 1.5+ GB
```

2.4 Selección de características

2.4.1 1. Métodos de filtro aplicados

Dado el tamaño del dataset y la mezcla de variables categóricas y numéricas, se aplicaron **métodos de filtro** para evaluar relevancia y redundancia antes del modelado.

- **Ganancia de información (Information Gain / Mutual Information)**
  - Mide la reducción de incertidumbre sobre `Arrest` al conocer una variable.
  - Adecuada para variables categóricas o mixtas.
  - En nuestro dataset, resalta *Primary Type* y *Location Description* como las más informativas.
  - Usado como criterio principal de selección.
- **Chi-cuadrado (²) — solo categóricas**
  - Evalúa dependencia estadística entre variables categóricas y la variable objetivo.
  - Útil para features one-hot (*Primary Type*, *Location Description*).
  - Se usó para filtrar categorías con poca o nula relación con arrestos.
- **Coeficiente de correlación (Pearson/Spearman) — numéricas**

  - Detecta redundancia entre variables numéricas (ej. `Latitude`, `Longitude`, `District`).

  - Se descartaron variables altamente correlacionadas para evitar multicolinealidad.
- **Variance Threshold (varianza baja)**
  - Elimina variables con muy poca variación (ej. columnas binarias casi siempre 0).

  - Se aplicó como pre-filtro simple para descartar variables irrelevantes.

2.4.2 2. Métodos adicionales

- **ANOVA F-test**
  - Evalúa si medias de `Arrest` difieren significativamente entre grupos categóricos.

  - Útil para confirmar relevancia de variables como `District` o `Primary Type`.
- **Información Mutua (Mutual Information)**
  - Variante no paramétrica de Information Gain, robusta para relaciones no lineales.

  - Complementó ^{2} para seleccionar variables mixtas.

2.4.3 3. Técnicas avanzadas (dimensionalidad)

- **PCA (Análisis de Componentes Principales)**
  - Reduce dimensionalidad de variables numéricas altamente correlacionadas.

  - Se considera como paso opcional en escenarios de sobrecarga de memoria (ej. >40 features activas).
- **Autoencoders**
  - Reducción no lineal basada en redes neuronales.

  - Solo justificable en escenarios de modelado más avanzado, no en la Regresión Logística actual.

2.4.4 4. Consideración del class imbalance

- Dado que la clase positiva (`Arrest=1`) representa solo **9–12%** de los casos:
  - Los métodos de selección se aplicaron **respetando estratificación**, para no sesgar la importancia de variables.

  - En el modelo se usó `class_weight="balanced"`, garantizando que la selección de características se alinee con un ajuste adecuado al desbalance.

2.4.5 5. Conclusión de selección

- **Variables mantenidas (tras filtros):**
  - `Domestic` (binaria), `cs_closed`, `cs_public`, `Primary Type` (OHE top categorías), `Location Description` (codificada eficiente), `District`, `Community Area`, `Hour`, `Weekday`, `Month`, `Latitude`, `Longitude`.

- **Variables eliminadas:**
  - Identificadores (`ID`, `Case Number`, `Location`), y aquellas de muy baja varianza o alta correlación redundante.

[51]:
```
y_all = df_ML["Arrest"].astype(int).values

base_feats = [
    "Domestic","cs_closed","cs_public","Latitude","Longitude",
    "District","Beat","Community Area","Hour","Weekday","Month"
]
base_feats = [c for c in base_feats if c in df_ML.columns]

ptype_ohe_cols = [c for c in df_ML.columns if c.startswith("Primary Type_")]

X_cols = base_feats + ptype_ohe_cols

X_all = df_ML[X_cols].apply(pd.to_numeric, errors="coerce").fillna(0)

train_mask = df_ML["Year"] <= 2021
test_mask  = df_ML["Year"] == 2022
if test_mask.sum() == 0:  # por si no hay 2022
    train_mask = df_ML["Year"] <= 2020
    test_mask  = df_ML["Year"] == 2021

X_train, y_train = X_all.loc[train_mask], y_all[train_mask]
X_test,  y_test  = X_all.loc[test_mask],  y_all[test_mask]

print("Shapes:", X_train.shape, X_test.shape)
print("Cols en X:", len(X_cols))

# Copias seguras
Xtr_df = X_train[X_cols].copy()
Xte_df = X_test[X_cols].copy()
ytr = y_train.copy()

# Identificamos columnas OHE (categóricas no negativas) y numéricas
ohe_cols = [c for c in Xtr_df.columns if c.startswith("Primary Type_")]  #␣
 ↪ajusta si tienes más OHE
bin_cols = [c for c in ["Domestic","cs_closed","cs_public"] if c in Xtr_df.
 ↪columns]
num_cols = [c for c in Xtr_df.columns if c not in ohe_cols + bin_cols]
```

```
# Todas las columnas categóricas (no negativas) aptas para Chi²
cat_nonneg_cols = ohe_cols + bin_cols

print("Numéricas:", num_cols)
print("Categóricas no negativas:", cat_nonneg_cols)
```

```
Shapes: (7385083, 43) (234245, 43)
Cols en X: 43
Numéricas: ['Latitude', 'Longitude', 'District', 'Beat', 'Community Area',
'Hour', 'Weekday', 'Month']
Categóricas no negativas: ['Primary Type_ARSON', 'Primary Type_ASSAULT',
'Primary Type_BATTERY', 'Primary Type_BURGLARY', 'Primary Type_CONCEALED CARRY
LICENSE VIOLATION', 'Primary Type_CRIM SEXUAL ASSAULT', 'Primary Type_CRIMINAL
DAMAGE', 'Primary Type_CRIMINAL SEXUAL ASSAULT', 'Primary Type_CRIMINAL
TRESPASS', 'Primary Type_DECEPTIVE PRACTICE', 'Primary Type_DOMESTIC VIOLENCE',
'Primary Type_GAMBLING', 'Primary Type_HOMICIDE', 'Primary Type_HUMAN
TRAFFICKING', 'Primary Type_INTERFERENCE WITH PUBLIC OFFICER', 'Primary
Type_INTIMIDATION', 'Primary Type_KIDNAPPING', 'Primary Type_LIQUOR LAW
VIOLATION', 'Primary Type_MOTOR VEHICLE THEFT', 'Primary Type_NARCOTICS',
'Primary Type_NON-CRIMINAL', 'Primary Type_OBSCENITY', 'Primary Type_OFFENSE
INVOLVING CHILDREN', 'Primary Type_OTHER NARCOTIC VIOLATION', 'Primary
Type_OTHER OFFENSE', 'Primary Type_PROSTITUTION', 'Primary Type_PUBLIC
INDECENCY', 'Primary Type_PUBLIC PEACE VIOLATION', 'Primary Type_RITUALISM',
'Primary Type_ROBBERY', 'Primary Type_SEX OFFENSE', 'Primary Type_STALKING',
'Primary Type_THEFT', 'Primary Type_WEAPONS VIOLATION', 'Domestic']
```

[52]:
```
from sklearn.feature_selection import VarianceThreshold

# Umbral muy bajo: elimina columnas casi constantes
vt = VarianceThreshold(threshold=1e-6)
vt.fit(Xtr_df)

mask_vt = vt.get_support()
cols_vt = Xtr_df.columns[mask_vt].tolist()
Xtr_vt = Xtr_df[cols_vt]
Xte_vt = Xte_df[cols_vt]

print(f"VT -> {Xtr_df.shape[1]} → {Xtr_vt.shape[1]} columnas")
```

```
VT -> 43 → 42 columnas
```

[53]:
```
# Trabajamos con las numéricas que quedaron tras VT
num_vt = [c for c in num_cols if c in Xtr_vt.columns]

drop_corr = set()
if len(num_vt) > 1:
    corr = Xtr_vt[num_vt].corr().abs()
```

```
    # triángulo superior
    upper = corr.where(np.triu(np.ones(corr.shape), k=1).astype(bool))
    # Umbral de correlación (ajusta si quieres)
    high_pairs = [(i,j) for i in upper.columns for j in upper.index if (not pd.
 ↪isna(upper.loc[j,i]) and upper.loc[j,i] >= 0.90)]
    for i,j in high_pairs:
        # preferimos conservar la que tenga mayor varianza (arbitrario, podrías␣
 ↪elegir por ANOVA después)
        var_i = Xtr_vt[i].var()
        var_j = Xtr_vt[j].var()
        drop_corr.add(i if var_i < var_j else j)

Xtr_corr = Xtr_vt.drop(columns=list(drop_corr), errors="ignore")
Xte_corr = Xte_vt.drop(columns=list(drop_corr), errors="ignore")

print(f"Correlación alta: removidas {len(drop_corr)} → {Xtr_corr.shape[1]}␣
 ↪columnas")
```

```
Correlación alta: removidas 1 → 41 columnas
```

[54]:
```
from sklearn.feature_selection import chi2

chi2_cols = [c for c in cat_nonneg_cols if c in Xtr_corr.columns]
chi2_scores = pd.Series(index=chi2_cols, dtype=float)

if chi2_cols:
    # Añadimos una pequeña constante para evitar columnas completamente 0 si␣
 ↪hiciera falta
    Xtr_chi = Xtr_corr[chi2_cols].astype(np.float64)
    chi_vals, chi_p = chi2(Xtr_chi, ytr)
    chi2_scores = pd.Series(chi_vals, index=chi2_cols).
 ↪sort_values(ascending=False)

chi2_scores.head(15)
```

[54]:
```
Primary Type_NARCOTICS                          1.962752e+06
Primary Type_CRIMINAL TRESPASS                  2.090214e+05
Primary Type_PROSTITUTION                       1.858234e+05
Primary Type_THEFT                              1.829716e+05
Primary Type_CRIMINAL DAMAGE                    1.718151e+05
Primary Type_WEAPONS VIOLATION                  1.154967e+05
Primary Type_BURGLARY                           9.336234e+04
Primary Type_MOTOR VEHICLE THEFT                5.687348e+04
Primary Type_ROBBERY                            4.237935e+04
Primary Type_GAMBLING                           3.890553e+04
Primary Type_LIQUOR LAW VIOLATION               3.873486e+04
Primary Type_INTERFERENCE WITH PUBLIC OFFICER   3.866850e+04
```

```
Primary Type_PUBLIC PEACE VIOLATION              3.516809e+04
Domestic                                         3.016315e+04
Primary Type_DECEPTIVE PRACTICE                  2.132163e+04
dtype: float64
```

[55]:
```
from sklearn.feature_selection import f_classif

f_cols = [c for c in Xtr_corr.columns if c in num_cols]  # numéricas␣
 ↪supervivientes
f_scores = pd.Series(index=f_cols, dtype=float)

if f_cols:
    Xtr_f = Xtr_corr[f_cols].astype(np.float64)
    f_vals, f_p = f_classif(Xtr_f, ytr)
    f_scores = pd.Series(f_vals, index=f_cols).sort_values(ascending=False)

f_scores.head(15)
```

[55]:
```
Hour              46979.671571
Longitude          7936.879984
Month              4391.728860
Beat               1799.178730
Community Area      276.564458
Weekday             211.168235
Latitude              3.475476
dtype: float64
```

[56]:
```
from sklearn.feature_selection import mutual_info_classif

cols_now = Xtr_corr.columns.tolist()
discrete_mask = np.array([1 if c in (set(chi2_cols) | set(bin_cols)) else 0 for␣
 ↪c in cols_now], dtype=bool)

Xtr_mi = Xtr_corr[cols_now].astype(np.float32)
mi_vals = mutual_info_classif(Xtr_mi, ytr, discrete_features=discrete_mask,␣
 ↪random_state=42)
mi = pd.Series(mi_vals, index=cols_now).sort_values(ascending=False)

mi.head(20)
```

[56]:
```
Weekday                                          0.153175
Primary Type_NARCOTICS                           0.141928
Month                                            0.090295
Hour                                             0.057826
Latitude                                         0.042731
Community Area                                   0.033728
Longitude                                        0.031091
```

```
Beat                                                0.021953
Primary Type_THEFT                                  0.017853
Primary Type_CRIMINAL DAMAGE                        0.016503
Primary Type_CRIMINAL TRESPASS                      0.012443
Primary Type_PROSTITUTION                           0.012111
Primary Type_BURGLARY                               0.008751
Primary Type_WEAPONS VIOLATION                      0.006751
Primary Type_MOTOR VEHICLE THEFT                    0.004971
Primary Type_ROBBERY                                0.003625
Domestic                                            0.002591
Primary Type_GAMBLING                               0.002502
Primary Type_LIQUOR LAW VIOLATION                   0.002483
Primary Type_INTERFERENCE WITH PUBLIC OFFICER       0.002323
dtype: float64
```

[57]:
```
TOP_K = min(40, len(mi))  # ajusta a tu gusto Stewart
final_cols = mi.index[:TOP_K].tolist()

X_train_fs = Xtr_corr[final_cols].copy()
X_test_fs  = Xte_corr[final_cols].copy()

print("Columns kept:", len(final_cols))
final_cols[:20]
```

```
Columns kept: 40
```

[57]:
```
['Weekday',
 'Primary Type_NARCOTICS',
 'Month',
 'Hour',
 'Latitude',
 'Community Area',
 'Longitude',
 'Beat',
 'Primary Type_THEFT',
 'Primary Type_CRIMINAL DAMAGE',
 'Primary Type_CRIMINAL TRESPASS',
 'Primary Type_PROSTITUTION',
 'Primary Type_BURGLARY',
 'Primary Type_WEAPONS VIOLATION',
 'Primary Type_MOTOR VEHICLE THEFT',
 'Primary Type_ROBBERY',
 'Domestic',
 'Primary Type_GAMBLING',
 'Primary Type_LIQUOR LAW VIOLATION',
 'Primary Type_INTERFERENCE WITH PUBLIC OFFICER']
```

[58]:
```
resume = pd.DataFrame({
    "feature": final_cols,
    "MI": mi[final_cols].values,
    "is_categorical": [c in (set(chi2_cols)|set(bin_cols)) for c in final_cols],
    "chi2_rank": [ (chi2_scores.index.get_loc(c)+1) if c in chi2_scores.index␣
 ↪else np.nan for c in final_cols ],
    "anova_F": [ f_scores.get(c, np.nan) for c in final_cols ]
})
resume.head(20)
```

[58]:

	feature	MI	is_categorical	chi2_rank	anova_F
0	Weekday	0.153175	False	NaN	211.168235
1	Primary Type_NARCOTICS	0.141928	True	1.0	NaN
2	Month	0.090295	False	NaN	4391.728860
3	Hour	0.057826	False	NaN	46979.671571
4	Latitude	0.042731	False	NaN	3.475476
5	Community Area	0.033728	False	NaN	276.564458
6	Longitude	0.031091	False	NaN	7936.879984
7	Beat	0.021953	False	NaN	1799.178730
8	Primary Type_THEFT	0.017853	True	4.0	NaN
9	Primary Type_CRIMINAL DAMAGE	0.016503	True	5.0	NaN
10	Primary Type_CRIMINAL TRESPASS	0.012443	True	2.0	NaN
11	Primary Type_PROSTITUTION	0.012111	True	3.0	NaN
12	Primary Type_BURGLARY	0.008751	True	7.0	NaN
13	Primary Type_WEAPONS VIOLATION	0.006751	True	6.0	NaN
14	Primary Type_MOTOR VEHICLE THEFT	0.004971	True		
15	Primary Type_ROBBERY	0.003625	True		
16	Domestic	0.002591	True		
17	Primary Type_GAMBLING	0.002502	True		
18	Primary Type_LIQUOR LAW VIOLATION	0.002483	True		
19	Primary Type_INTERFERENCE WITH PUBLIC OFFICER	0.002323	True		

```
14        8.0           NaN
15        9.0           NaN
16       14.0           NaN
17       10.0           NaN
18       11.0           NaN
19       12.0           NaN
```

3 Exploración y análisis de datos

3.1 Tendencias generales

3.2 Pregunta 1 – Evolución a largo plazo (EDA temporal)

3.2.1 Planteamiento

- Pregunta:

  ¿Qué tendencias de largo plazo se observan en la frecuencia y tipo de delitos en Chicago, y cómo han influido eventos como la crisis del 2008 o la pandemia del 2020?

- Hipótesis:

  Los delitos violentos (robos, asaltos, agresiones) muestran caídas temporales durante eventos críticos (ej. confinamientos de la pandemia), mientras que delitos como fraude o cibercrimen tienden a incrementarse en los mismos periodos.

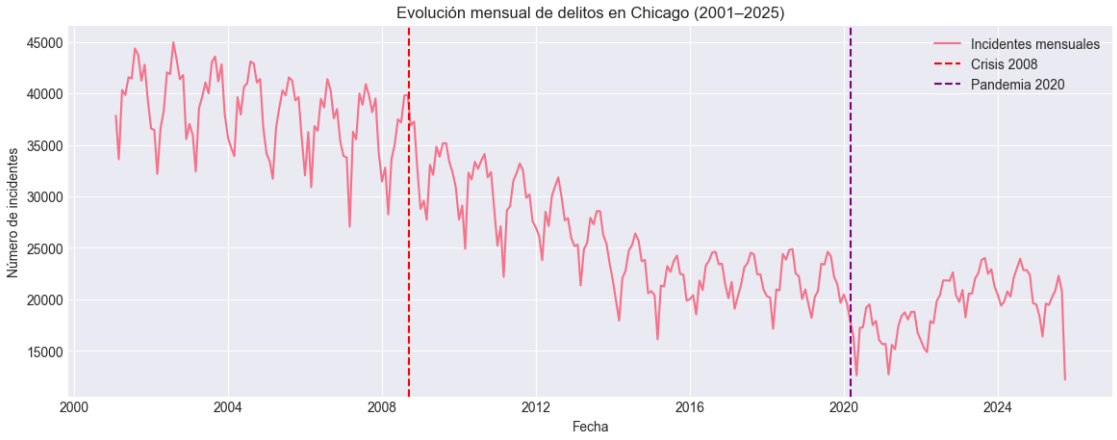
- Justificación:

  Permite analizar los datos como una serie temporal extensa (2001–2025), conectando las fluctuaciones con eventos sociales y económicos, lo cual aporta una visión estructural del crimen en la ciudad.

3.2.2 Serie total de delitos a lo largo del tiempo

[102]:
```
# Serie mensual total de delitos
total_monthly = df.groupby(pd.Grouper(key="Date", freq="M")).size()

import matplotlib.pyplot as plt

plt.figure(figsize=(14,5))
plt.plot(total_monthly.index, total_monthly.values, label="Incidentes␣
 ↪mensuales")
plt.axvline(pd.to_datetime("2008-09-15"), color="red", linestyle="--",␣
 ↪label="Crisis 2008")
plt.axvline(pd.to_datetime("2020-03-01"), color="purple", linestyle="--",␣
 ↪label="Pandemia 2020")

plt.title("Evolución mensual de delitos en Chicago (2001-2025)")
plt.xlabel("Fecha")
plt.ylabel("Número de incidentes")
```

```
plt.legend()
plt.show()
```



El análisis a nivel anual resulta limitado debido a la marcada variación mensual observada en la serie. Esto plantea una nueva pregunta: **¿Qué mes es el más seguro y cuál el más criminalístico en la ciudad de Chicago?**

3.2.3 Suavizado de la serie y tasa de variación interanual

[ ]:
```
rolling_12 = total_monthly.rolling(12).mean()
yoy_pct = total_monthly.pct_change(12) * 100

fig, ax = plt.subplots(2,1, figsize=(14,8), sharex=True)

# Serie + suavizado
ax[0].plot(total_monthly.index, total_monthly.values, alpha=0.4,␣
 ↪label="Mensual")
ax[0].plot(rolling_12.index, rolling_12.values, color="black", linewidth=2,␣
 ↪label="Media móvil 12 meses")
ax[0].legend()
ax[0].set_title("Serie mensual y tendencia suavizada")

# Variación interanual
ax[1].plot(yoy_pct.index, yoy_pct.values, color="green")
ax[1].axhline(0, color="k", linestyle="--")
ax[1].set_title("Cambio interanual (%)")

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

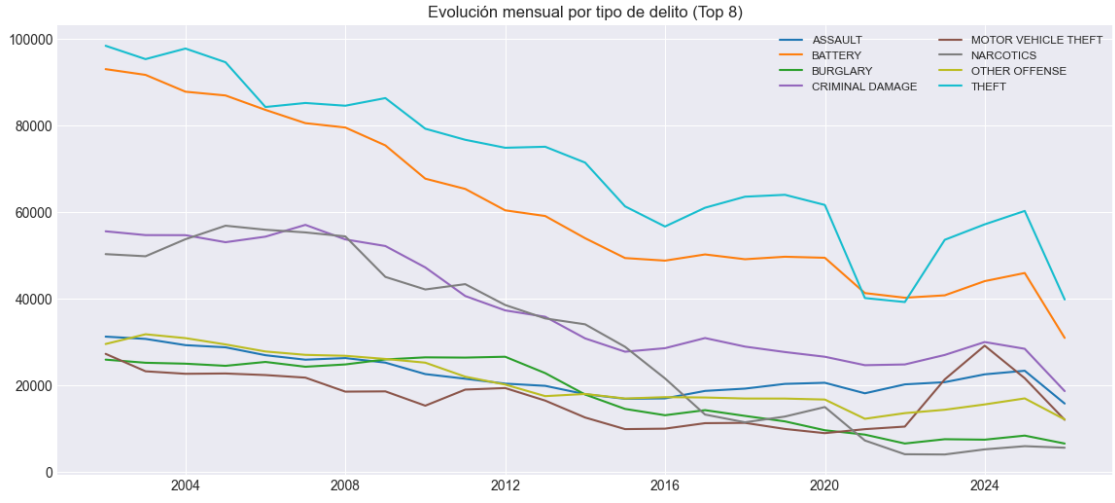
1. **Serie mensual y tendencia suavizada (arriba)**
   - Entre **2001 y 2008**, los delitos en Chicago se mantuvieron en niveles altos (35,000–45,000 casos mensuales), pero con una **tendencia a la baja muy gradual**.
   - La **crisis del 2008** no produjo un "quiebre" evidente: el descenso que se ve corresponde a la **inercia del declive general** de los delitos, no a un shock puntual.
   - El cambio fuerte realmente aparece en **2018–2020**, donde el descenso se acelera hasta llegar al mínimo histórico durante la pandemia.
   - Tras la pandemia (2021–2022), se nota un **rebote temporal**, pero desde 2023 la curva vuelve a caer.
2. **Cambio interanual (%) (abajo)**
   - Antes del 2008, los cambios interanuales son pequeños, cercanos a **0%**, lo que confirma que no hubo un quiebre específico en esa crisis.
   - El **descenso más pronunciado** interanual se da en **2020**, con caídas cercanas al -40%, un shock directamente asociado al confinamiento y restricciones de movilidad.
   - En **2021–2022**, el rebote muestra picos de hasta +30%, indicando una **recuperación abrupta** tras la reapertura.
   - A partir de 2023, los cambios interanuales vuelven a ser mayormente negativos, reforzando la idea de que el crimen retomó su **trayectoria descendente de largo plazo**.
3. **Conclusiones ajustadas**
   1. **Tendencia general**: El crimen en Chicago sigue una **trayectoria descendente desde 2001**, independiente de crisis económicas como la del 2008.
   2. **2008**: No representó un punto de inflexión claro; la caída vista en esos años se explica por la misma tendencia estructural, no por la crisis en sí.
   3. **2020 (pandemia)**: Sí marcó un **shock histórico**, con la caída más brusca en toda la

serie (-40%).

4. **2021–2022**: Rebote por reapertura y reactivación social, aunque no alcanza los niveles previos a 2010.
5. **2023–2025**: Nueva fase de declive, lo que refuerza la hipótesis de un **nuevo "piso bajo" en los niveles de criminalidad**, más estable que en décadas pasadas.

3.2.4 Evolución por tipo de delito (Primary Type)

[115]:
```
import matplotlib.cm as cm

# Top 8 tipos de delitos
top_types = df["Primary Type"].value_counts().nlargest(8).index

monthly_by_type = (df[df["Primary Type"].isin(top_types)]
                   .groupby([pd.Grouper(key="Date", freq="Y"), "Primary Type"])
                   .size()
                   .unstack(fill_value=0))

# Generar una paleta con tantos colores como columnas
colors = cm.get_cmap("tab10", len(monthly_by_type.columns))

plt.figure(figsize=(14,6))
for i, col in enumerate(monthly_by_type.columns):
    plt.plot(monthly_by_type.index, monthly_by_type[col],
             label=col, color=colors(i))
plt.title("Evolución mensual por tipo de delito (Top 8)")
plt.legend(ncol=2, fontsize="small")
plt.show()
```



1. **Tendencia general descendente:** La mayoría de los delitos en Chicago han disminuido desde 2001, reflejando mejoras estructurales en seguridad y cambios sociales.

2. **Excepción importante:** El **robo de vehículos (MOTOR VEHICLE THEFT)** está en auge nuevamente desde 2017, rompiendo la tendencia de reducción y convirtiéndose en un problema emergente.
3. **Impacto de la pandemia (2020):** Se nota una caída abrupta en la mayoría de los delitos, con recuperación parcial en 2021–2022.
4. **Delitos predominantes:** A lo largo de toda la serie, **THEFT** y **BATTERY** son los más frecuentes, aunque hoy su volumen es menor al de principios de siglo.
5. **Política de drogas:** La fuerte caída en **NARCOTICS** sugiere cambios legales y sociales en torno al consumo/posesión, más que una simple reducción delictiva.

3.2.5 Clasificación y categorización de delitos

[109]:
```
df_p1 = df.copy()
```

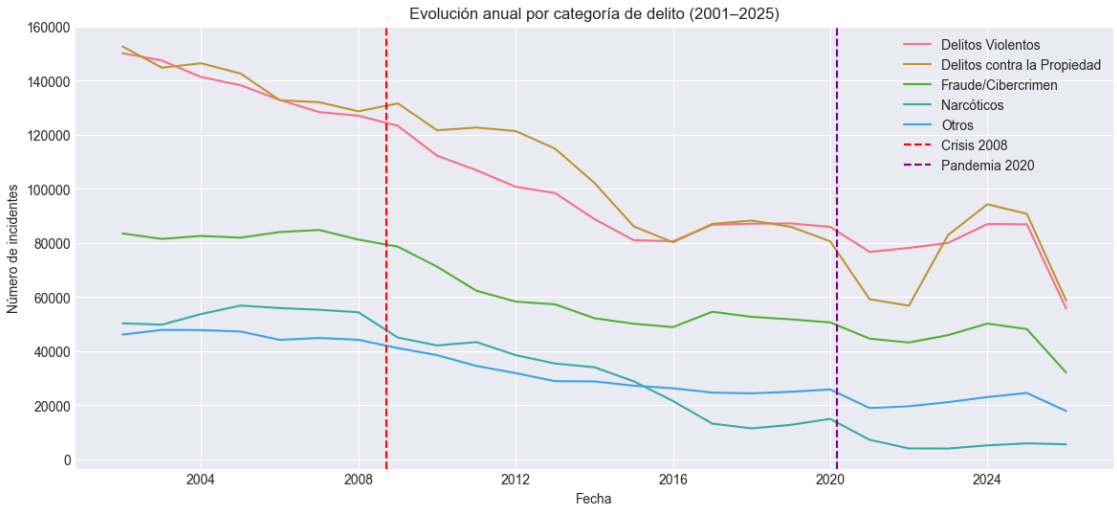
[110]:
```
# Definir categorías según la hipótesis
delitos_violentos = [
    'ROBBERY', 'ASSAULT', 'BATTERY', 'HOMICIDE',
    'CRIM SEXUAL ASSAULT', 'WEAPONS VIOLATION', 'KIDNAPPING'
]

delitos_propiedad = [
    'BURGLARY', 'MOTOR VEHICLE THEFT', 'THEFT', 'ARSON'
]

delitos_fraude_ciberneticos = [
    'DECEPTIVE PRACTICE', 'FRAUD', 'IDENTITY THEFT',
    'CRIMINAL DAMAGE', 'CRIMINAL TRESPASS'
]

delitos_narcoticos = ['NARCOTICS', 'OTHER NARCOTIC VIOLATION']

# Función de categorización
def categorizar_delito(tipo):
    if tipo in delitos_violentos:
        return 'Delitos Violentos'
    elif tipo in delitos_propiedad:
        return 'Delitos contra la Propiedad'
    elif tipo in delitos_fraude_ciberneticos:
        return 'Fraude/Cibercrimen'
    elif tipo in delitos_narcoticos:
        return 'Narcóticos'
    else:
        return 'Otros'

# Aplicar categorización
df_p1['Categoria_Delito'] = df_p1['Primary Type'].apply(categorizar_delito)
```

```
# Ver distribución inicial
df_p1['Categoria_Delito'].value_counts()
```

[110]:
```
Categoria_Delito
Delitos contra la Propiedad    2646167
Delitos Violentos              2570062
Fraude/Cibercrimen             1533388
Otros                           805725
Narcóticos                      750583
Name: count, dtype: int64
```

3.2.6 Evolución temporal por categorías

[112]:
```
# Serie mensual por categoría
year_by_cat = (df_p1.groupby([pd.Grouper(key="Date", freq="Y"),␣
 ↪"Categoria_Delito"])
                    .size()
                    .unstack(fill_value=0))

# Gráfico de líneas
plt.figure(figsize=(14,6))
for col in year_by_cat.columns:
    plt.plot(year_by_cat.index, year_by_cat[col], label=col)

plt.axvline(pd.to_datetime("2008-09-15"), color="red", linestyle="--",␣
 ↪label="Crisis 2008")
plt.axvline(pd.to_datetime("2020-03-01"), color="purple", linestyle="--",␣
 ↪label="Pandemia 2020")

plt.title("Evolución anual por categoría de delito (2001-2025)")
plt.xlabel("Fecha")
plt.ylabel("Número de incidentes")
plt.legend()
plt.show()
```



3.2.7 Comparación pre/durante/post por categoría

[114]:
```
# Definir ventanas
windows = {
    "Pre-crisis 2008": ("2006-01-01","2007-12-31"),
    "Crisis 2008": ("2008-01-01","2009-12-31"),
    "Pre-Covid": ("2018-01-01","2019-12-31"),
    "Covid 2020": ("2020-03-01","2020-12-31"),
    "Post-Covid": ("2021-01-01","2022-12-31")
}

resumen = []
for w,(s,e) in windows.items():
    mean_vals = year_by_cat[s:e].mean().astype(int).to_dict()
    resumen.append({"Periodo": w, **mean_vals})

import pandas as pd
tabla_resumen = pd.DataFrame(resumen)
tabla_resumen
```

[114]:

	Periodo	Delitos Violentos	Delitos contra la Propiedad	Fraude/Cibercrimen	Narcóticos	Otros
0	Pre-crisis 2008	127737	130373	83051	54882	44588
1	Crisis 2008	117810	126644			
2	Pre-Covid	86618	83325			
3	Covid 2020	76728	59265			
4	Post-Covid	79131	69912			

```
1              74968       43613  39859
2              51235       13899  25456
3              44670        7274  19011
4              44599        4068  20432
```

1. **Tendencia descendente global (2008–2025):** Todos los delitos han caído de manera significativa respecto a niveles pre-crisis 2008.
2. **Impacto de la pandemia:** 2020 marcó un **mínimo histórico en delitos**, con repunte parcial posterior.
3. **Narcóticos como caso excepcional:** Pasaron de 54,882 (2008) a solo 4,068 en post-Covid, es decir, una **reducción del 92.6%** en dos décadas. Esto refleja cambios legales y de política pública más que solo un descenso natural.
4. **Rebrote post-pandemia:** Los delitos violentos y contra la propiedad son los que más rápidamente se reactivaron, lo que los convierte en los principales focos actuales de seguridad.
5. **Fraude/cibercrimen:** Aunque bajó en pandemia, su reducción es mucho menor comparada con otros delitos, lo que sugiere que puede estar **transformándose hacia modalidades digitales** y no necesariamente desapareciendo.

3.3 Pregunta 2 – Factores asociados a arrestos (EDA + ML supervisado)

- Pregunta:

  ¿Qué factores determinan la probabilidad de que un crimen resulte en un arresto, y hasta qué punto un modelo de Machine Learning puede predecir este resultado con precisión?

- Hipótesis:

  Los delitos domésticos y los crímenes cometidos en lugares cerrados tienen mayor probabilidad de concluir en arresto que aquellos ocurridos en la vía pública, debido a la existencia de testigos, vínculos entre víctima y agresor, y facilidad de intervención policial.

- Justificación:

  Es la pregunta que conecta el análisis exploratorio con Machine Learning, aplicando clasificación supervisada sobre la variable Arrest. Además, permite evaluar importancia de variables como Primary Type, Location Description, Domestic, District y variables temporales derivadas.

3.3.1 Proceso General

A) Preparar variables para la hipótesis Qué hace:

Re-crea closed_space con dummies (evita tratar -1/0/1 como ordinal).

Define un subset comparable sin categorías "de operativo/vice" (narcóticos, prostitución, etc.), que sesgan la hipótesis.

[61]:
```
# --- Crear 'closed_space' (1=cerrado, 0=público, -1=desconocido) ---
closed_keywords = {
    "APARTMENT","RESIDENCE","RESIDENCE-GARAGE","HOTEL/MOTEL","NURSING HOME",
    "SCHOOL","SCHOOL, PUBLIC, BUILDING","SMALL RETAIL STORE","DEPARTMENT STORE",
    "GROCERY FOOD STORE","TAVERN/LIQUOR STORE","RESTAURANT","GAS STATION",
```

```
    "BANK","COMMERCIAL / BUSINESS OFFICE","HOSPITAL BUILDING/GROUNDS",
    "CTA TRAIN","CTA PLATFORM","CTA STATION","POLICE FACILITY/VEH PARKING LOT"
}
public_keywords = {
    "STREET","SIDEWALK","ALLEY","PARKING LOT/GARAGE(NON RESID.)","PARK PROPERTY"
}

# Trabaja sobre df_ML (o cambia por df si prefieres)
locu = df_ML["Location Description"].astype(str).str.upper().str.strip()

mask_closed = locu.isin(closed_keywords)
mask_public = locu.isin(public_keywords)

closed_space = np.full(len(df_ML), -1, dtype=np.int8)  # -1 = desconocido
closed_space[mask_public.values] = 0                   # 0  = público
closed_space[mask_closed.values] = 1                   # 1  = cerrado

df_ML["closed_space"] = closed_space  # agrega la columna
```

[62]:
```
# === A1) closed_space como dummies (evita ordinalidad) ===
# Asumimos df_ML ya tiene 'closed_space' = {-1, 0, 1}
df_ML = df_ML.copy()
df_ML["cs_closed"] = (df_ML["closed_space"] == 1).astype("int8")  # 1=cerrado
df_ML["cs_public"] = (df_ML["closed_space"] == 0).astype("int8")  # 1=público
# "desconocido" queda como categoría base (cuando ambas dummies = 0)

# === A2) subset "comparables": excluir tipos con arresto casi automático ===
exclude_pt = {
    "NARCOTICS","PROSTITUTION","GAMBLING","LIQUOR LAW VIOLATION",
    "PUBLIC INDECENCY","OBSCENITY","CONCEALED CARRY LICENSE VIOLATION",
    "OTHER NARCOTIC VIOLATION","INTERFERENCE WITH PUBLIC OFFICER"
}
pt_cols = [c for c in df_ML.columns if c.startswith("Primary Type_")]
pt_exclude_cols = [f"Primary Type_{p}" for p in exclude_pt if f"Primary␣
 ↪Type_{p}" in pt_cols]

mask_comp = ~df_ML[pt_exclude_cols].any(axis=1) if pt_exclude_cols else np.
 ↪ones(len(df_ML), dtype=bool)
df_cmp = df_ML.loc[mask_comp].copy()
print(f"Filas totales: {len(df_ML):,} | Subset comparables: {len(df_cmp):,}")
```

```
Filas totales: 8,305,925 | Subset comparables: 7,432,968
```

B) EDA mínima que responde la hipótesis Qué hace: compara tasas de arresto en dataset completo y en el subset comparable para Domestic y closed_space.

[63]:
```
def tasas(df_in, nombre):
    t1 = df_in.groupby("Domestic")["Arrest"].mean().rename("rate").reset_index()
    t2 = (df_in.assign(closed_space_cat = np.select(
            [df_in["cs_closed"].eq(1), df_in["cs_public"].eq(1)],
            ["closed","public"], default="unknown"))
          .groupby("closed_space_cat")["Arrest"].mean().rename("rate").
 ↪reset_index())
    print(f"\n== Tasas de arresto en {nombre} ==")
    display(t1, t2)

# Dataset completo
tasas(df_ML, "dataset completo")
# Subset comparables (sin vice/operativos)
tasas(df_cmp, "subset comparable")
```

```
== Tasas de arresto en dataset completo ==
```

	Domestic	rate
0	0	0.265742
1	1	0.192769

	closed_space_cat	rate
0	unknown	0.253103

```
== Tasas de arresto en subset comparable ==
```

	Domestic	rate
0	0	0.160092
1	1	0.192435

	closed_space_cat	rate
0	unknown	0.166347

C) Construir X/y y split temporal (subset comparable) Qué hace: arma las features, respeta validación temporal (train 2021, test=2022) y toma muestra estratificada (100k) para tiempo razonable.

[64]:
```
# === C1) columnas para el modelo (subset comparable) ===
y_all = df_cmp["Arrest"].astype(int).values

base_feats = [
    "Domestic","cs_closed","cs_public","Latitude","Longitude",
    "District","Beat","Community Area","Hour","Weekday","Month"
]
base_feats = [c for c in base_feats if c in df_cmp.columns]
ptype_ohe_cols = [c for c in df_cmp.columns if c.startswith("Primary Type_")]
X_cols = base_feats + ptype_ohe_cols
```

```
X_all = df_cmp[X_cols].apply(pd.to_numeric, errors="coerce").fillna(0)

# === C2) split temporal ===
train_mask = df_cmp["Year"] <= 2021
test_mask  = df_cmp["Year"] == 2022
if test_mask.sum() == 0:   # fallback si no hay 2022
    train_mask = df_cmp["Year"] <= 2020
    test_mask  = df_cmp["Year"] == 2021

X_train, y_train = X_all.loc[train_mask], y_all[train_mask]
X_test,  y_test  = X_all.loc[test_mask],  y_all[test_mask]

print("Shapes (full subset):", X_train.shape, X_test.shape)

# === C3) muestra estratificada ~100k para iterar rápido ===
from sklearn.model_selection import StratifiedShuffleSplit

def strat_sample(X_df, y_arr, n, seed=42):
    if len(y_arr) <= n:
        return X_df, y_arr
    sss = StratifiedShuffleSplit(n_splits=1, train_size=n, random_state=seed)
    idx_small, _ = next(sss.split(X_df, y_arr))
    return X_df.iloc[idx_small], y_arr[idx_small]

X_train_s, y_train_s = strat_sample(X_train, y_train, 100_000, seed=42)
X_test_s,  y_test_s  = strat_sample(X_test,  y_test,  100_000, seed=7)

print("Train sample:", X_train_s.shape, "positivos:", int(y_train_s.sum()))
print("Test  sample:", X_test_s.shape,  "positivos:", int(y_test_s.sum()))
```

```
Shapes (full subset): (6538031, 45) (229091, 45)
Train sample: (100000, 45) positivos: 17421
Test  sample: (100000, 45) positivos: 9780
```

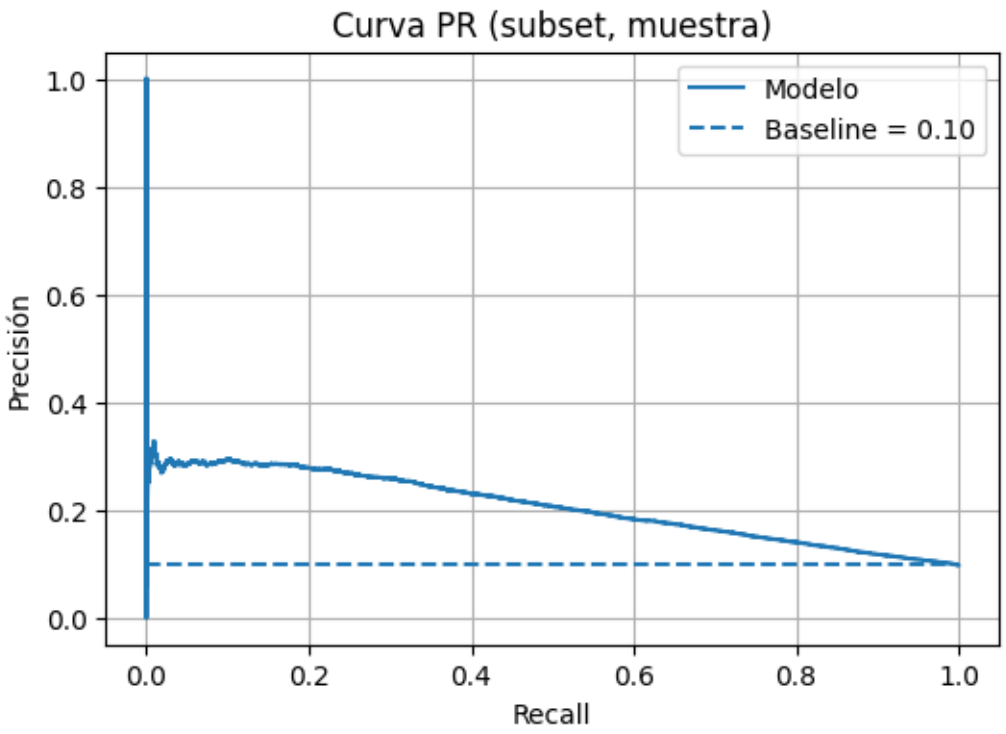
D) Entrenar Logistic (muestra) y evaluar (PR/ROC + umbrales) Qué hace: entrena Logistic (solver='saga'), calcula PR/ROC, elige umbral por F1 y también por objetivos (precisión o recall).

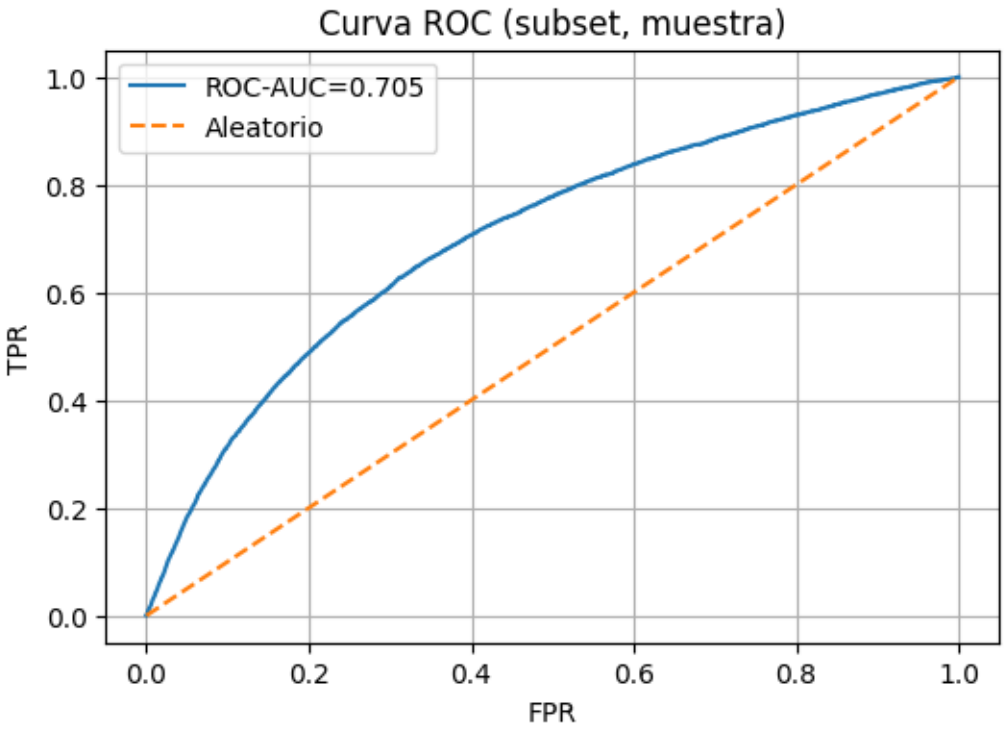
[65]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (average_precision_score, roc_auc_score,
                             precision_recall_curve, roc_curve, f1_score,
                             confusion_matrix, classification_report)

# a) a numpy float32 y saneo de NaN/Inf
Xtr = X_train_s.to_numpy(dtype=np.float32, copy=True)
Xte = X_test_s .to_numpy(dtype=np.float32, copy=True)
np.nan_to_num(Xtr, copy=False, nan=0.0, posinf=0.0, neginf=0.0)
```

```
np.nan_to_num(Xte, copy=False, nan=0.0, posinf=0.0, neginf=0.0)

# b) Logistic robusta (saga) y balanceo por clase
logit = LogisticRegression(
    solver="saga", penalty="l2", C=0.5,
    class_weight="balanced", max_iter=5000, tol=1e-3,
    random_state=42
)
logit.fit(Xtr, y_train_s)
proba = logit.predict_proba(Xte)[:, 1]

# c) PR/ROC agregados
prec, rec, thr = precision_recall_curve(y_test_s, proba)
ap  = average_precision_score(y_test_s, proba)
roc = roc_auc_score(y_test_s, proba) if len(np.unique(y_test_s))>=2 else␣
 ↪float("nan")
prev = y_test_s.mean()

print(f"Prevalencia (baseline PR): {prev:.3f}")
print(f"Average Precision (AP):    {ap:.3f}")
print(f"ROC-AUC:                   {roc:.3f}")

plt.figure(figsize=(6,4))
plt.plot(rec, prec, label="Modelo")
plt.hlines(prev, 0, 1, linestyles="--", label=f"Baseline = {prev:.2f}")
plt.xlabel("Recall"); plt.ylabel("Precisión"); plt.title("Curva PR (subset,␣
 ↪muestra)"); plt.grid(True); plt.legend(); plt.show()

if len(np.unique(y_test_s))>=2:
    fpr, tpr, _ = roc_curve(y_test_s, proba)
    plt.figure(figsize=(6,4))
    plt.plot(fpr, tpr, label=f"ROC-AUC={roc:.3f}")
    plt.plot([0,1],[0,1],"--", label="Aleatorio")
    plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("Curva ROC (subset,␣
 ↪muestra)"); plt.grid(True); plt.legend(); plt.show()

# d) elegir umbrales (F1 y objetivos)
f1s = (2*prec*rec)/(prec+rec+1e-12)
k = int(np.nanargmax(f1s))
thr_f1 = thr[k-1] if k>0 else 0.5

def eval_at(threshold, name):
    y_pred = (proba >= threshold).astype(int)
    print(f"\n{name} (thr={threshold:.3f})")
    print("Precision:", (y_test_s[y_pred==1].sum()/max(y_pred.sum(),1)))
    print("Recall   :", ((y_pred & (y_test_s==1)).sum()/max((y_test_s==1).
 ↪sum(),1)))
```

```
    print("F1       :", f1_score(y_test_s, y_pred))
    print("Matriz de confusión:\n", confusion_matrix(y_test_s, y_pred))
    print(classification_report(y_test_s, y_pred, digits=3, zero_division=0))

# F1 máximo (punto balanceado)
eval_at(thr_f1, "Umbral por F1 máx")

# Objetivos operativos (ajusta según tu preferencia)
target_precision = 0.60
idx = np.where(prec >= target_precision)[0]
if len(idx):
    j = idx[-1]
    thr_prec = thr[j-1] if j>0 else 0.5
    eval_at(thr_prec, f"Umbral por Precision  {target_precision:.2f}")
else:
    print(f"No hay umbral con Precision  {target_precision:.2f}")

target_recall = 0.60
idx = np.where(rec >= target_recall)[0]
if len(idx):
    j = idx[0]
    thr_rec = thr[j-1] if j>0 else 0.5
    eval_at(thr_rec, f"Umbral por Recall  {target_recall:.2f}")
else:
    print(f"No hay umbral con Recall  {target_recall:.2f}")
```

```
Prevalencia (baseline PR): 0.098
Average Precision (AP):    0.206
ROC-AUC:                   0.705
```


Curva PR (subset, muestra)
Modelo
Baseline = 0.10
Precisión
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Curva ROC (subset, muestra)
ROC-AUC=0.705
Aleatorio
TPR
FPR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
Umbral por F1 máx (thr=0.524)
Precision: 0.22416635643248417
Recall   : 0.43098159509202455
F1       : 0.294930553125984
Matriz de confusión:
 [[75632 14588]
 [ 5565  4215]]
              precision    recall  f1-score   support

           0      0.931     0.838     0.882     90220
           1      0.224     0.431     0.295      9780

    accuracy                          0.798    100000
   macro avg      0.578     0.635     0.589    100000
weighted avg      0.862     0.798     0.825    100000


Umbral por Precision  0.60 (thr=0.612)
Precision: 0.0
Recall   : 0.0
F1       : 0.0
Matriz de confusión:
 [[90219     1]
 [ 9780     0]]
              precision    recall  f1-score   support

           0      0.902     1.000     0.949     90220
           1      0.000     0.000     0.000      9780

    accuracy                          0.902    100000
   macro avg      0.451     0.500     0.474    100000
weighted avg      0.814     0.902     0.856    100000


Umbral por Recall  0.60 (thr=0.500)
Precision: 0.1690653988269037
Recall   : 0.6690184049079755
F1       : 0.2699201749138838
Matriz de confusión:
 [[58062 32158]
 [ 3237  6543]]
              precision    recall  f1-score   support

           0      0.947     0.644     0.766     90220
           1      0.169     0.669     0.270      9780

    accuracy                          0.646    100000
```

```
   macro avg      0.558     0.656     0.518    100000
weighted avg      0.871     0.646     0.718    100000
```

E) Factores asociados (coeficientes) Qué hace: muestra las señales del modelo (positivas/negativas) y un gráfico de |coef|. En especial, mira Domestic, cs_closed, cs_public y algunas Primary Type_*.

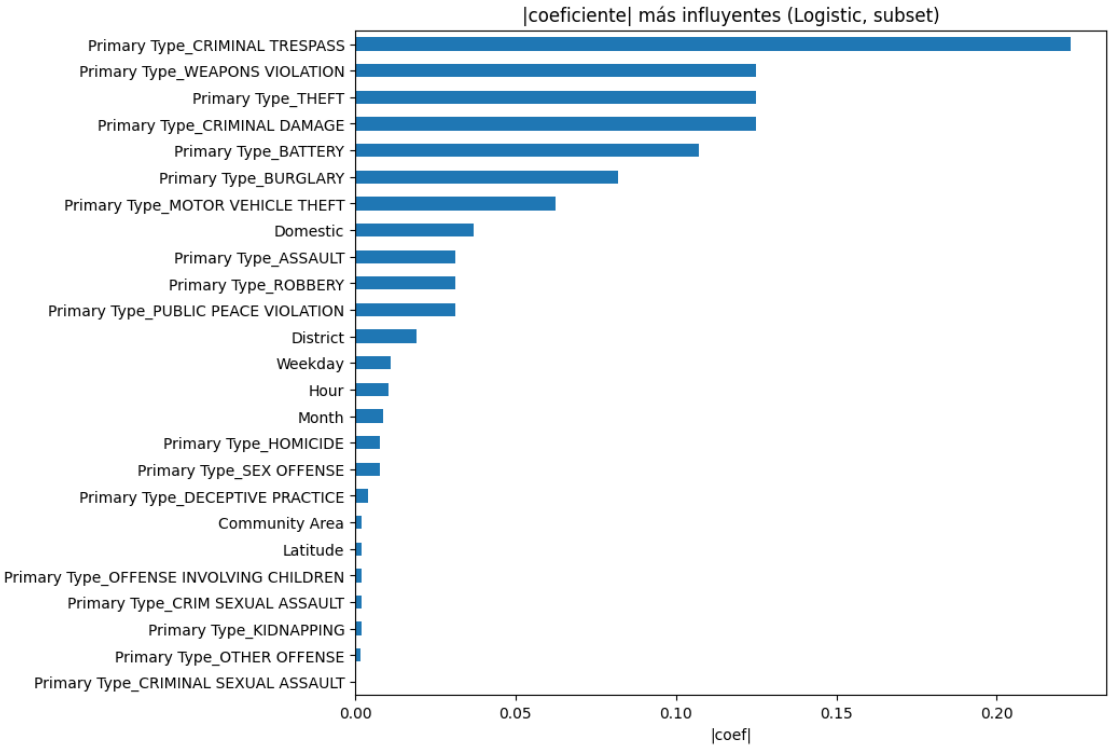
[66]:
```
coef = pd.Series(logit.coef_.ravel(), index=X_cols).astype(float)

print("Top señales POSITIVAS (aumentan prob. arresto):")
display(coef.sort_values(ascending=False).head(20))

print("Top señales NEGATIVAS (reducen prob. arresto):")
display(coef.sort_values(ascending=True).head(20))

plt.figure(figsize=(9,8))
coef.abs().sort_values(ascending=False).head(25).plot.barh()
plt.gca().invert_yaxis()
plt.title("|coeficiente| más influyentes (Logistic, subset)")
plt.xlabel("|coef|")
plt.show()
```

```
Top señales POSITIVAS (aumentan prob. arresto):
```

```
Primary Type_CRIMINAL TRESPASS                    0.223094
Primary Type_WEAPONS VIOLATION                    0.125000
Primary Type_BATTERY                              0.107117
Domestic                                          0.036810
Primary Type_ASSAULT                              0.031250
Primary Type_PUBLIC PEACE VIOLATION               0.031250
Weekday                                           0.011098
Hour                                              0.010495
Primary Type_HOMICIDE                             0.007812
Primary Type_SEX OFFENSE                          0.007726
Latitude                                          0.001961
Primary Type_OFFENSE INVOLVING CHILDREN           0.001953
Primary Type_OTHER OFFENSE                        0.001659
Longitude                                         0.000462
Primary Type_STALKING                             0.000122
Beat                                              0.000106
Primary Type_INTERFERENCE WITH PUBLIC OFFICER     0.000000
Primary Type_PUBLIC INDECENCY                     0.000000
Primary Type_PROSTITUTION                         0.000000
Primary Type_RITUALISM                            0.000000
dtype: float64
```

```
Top señales NEGATIVAS (reducen prob. arresto):
```

```
Primary Type_THEFT                          -0.125000
Primary Type_CRIMINAL DAMAGE                -0.125000
Primary Type_BURGLARY                       -0.081999
Primary Type_MOTOR VEHICLE THEFT            -0.062500
Primary Type_ROBBERY                        -0.031250
District                                    -0.019044
Month                                       -0.008864
Primary Type_DECEPTIVE PRACTICE             -0.003906
Community Area                              -0.002015
Primary Type_CRIM SEXUAL ASSAULT            -0.001953
Primary Type_KIDNAPPING                     -0.001953
Primary Type_ARSON                          -0.000488
Primary Type_CRIMINAL SEXUAL ASSAULT        -0.000488
Primary Type_HUMAN TRAFFICKING              -0.000029
Primary Type_INTIMIDATION                   -0.000022
Primary Type_PUBLIC INDECENCY                0.000000
Primary Type_PROSTITUTION                    0.000000
Primary Type_OTHER NARCOTIC VIOLATION        0.000000
Primary Type_OBSCENITY                       0.000000
Primary Type_NON-CRIMINAL                    0.000000
dtype: float64
```



[67]:
```
# === C1) columnas para el modelo (subset comparable) ===
y_all = df_cmp["Arrest"].astype(int).values

base_feats = [
    "Domestic","cs_closed","cs_public","Latitude","Longitude",
    "District","Beat","Community Area","Hour","Weekday","Month"
]
base_feats = [c for c in base_feats if c in df_cmp.columns]
ptype_ohe_cols = [c for c in df_cmp.columns if c.startswith("Primary Type_")]
X_cols = base_feats + ptype_ohe_cols

X_all = df_cmp[X_cols].apply(pd.to_numeric, errors="coerce").fillna(0)

# === C2) split temporal ===
train_mask = df_cmp["Year"] <= 2021
test_mask  = df_cmp["Year"] == 2022
if test_mask.sum() == 0:   # fallback si no hay 2022
    train_mask = df_cmp["Year"] <= 2020
    test_mask  = df_cmp["Year"] == 2021

X_train, y_train = X_all.loc[train_mask], y_all[train_mask]
X_test,  y_test  = X_all.loc[test_mask],  y_all[test_mask]

print("Shapes (full subset):", X_train.shape, X_test.shape)

# === C3) muestra estratificada ~100k para iterar rápido ===
from sklearn.model_selection import StratifiedShuffleSplit

def strat_sample(X_df, y_arr, n, seed=42):
    if len(y_arr) <= n:
        return X_df, y_arr
    sss = StratifiedShuffleSplit(n_splits=1, train_size=n, random_state=seed)
    idx_small, _ = next(sss.split(X_df, y_arr))
    return X_df.iloc[idx_small], y_arr[idx_small]

X_train_s, y_train_s = strat_sample(X_train, y_train, 100_000, seed=42)
X_test_s,  y_test_s  = strat_sample(X_test,  y_test,  100_000, seed=7)

print("Train sample:", X_train_s.shape, "positivos:", int(y_train_s.sum()))
print("Test  sample:", X_test_s.shape,  "positivos:", int(y_test_s.sum()))
```

```
Shapes (full subset): (6538031, 45) (229091, 45)
Train sample: (100000, 45) positivos: 17421
Test  sample: (100000, 45) positivos: 9780
```

3.3.2 Datos y metodología

- **Base:** Chicago Crimes (**8,305,925** registros).

- **Ingeniería de variables:**
  - `closed_space` desde *Location Description* (1=cerrado, 0=público, -1=desconocido) + dummies `cs_closed`, `cs_public`.
  - Variables temporales (`Hour`, `Weekday`, `Month`) y geográficas (`Latitude`, `Longitude`, `District`, `Beat`, `Community Area`).
  - One-hot de `Primary Type`.
- **Subset "comparable"** (aislar efecto lugar/naturaleza): se excluyen categorías con arresto casi automático en vía pública
  (*Narcotics, Prostitution, Gambling, Liquor Law Violation, Public Indecency, Obscenity, Concealed Carry License Violation, Other Narcotic Violation, Interference with Public Officer*).
  - **Tamaño: 7,432,968** filas (vs. **8,305,925** total).
- **Validación temporal:** *train* **2021** y *test* = **2022**.
- **Muestreo:** estratificado **100k** train / **100k** test (prevalencia test **9.8%**).
- **Modelo: Regresión Logística** (`solver='saga'`, `class_weight='balanced'`) entrenada en el subset comparable.

3.3.3 Evidencia EDA (tasa media de arresto)

Dataset completo

- `Domestic = 1` → **0.193** (vs. `Domestic = 0` **0.266**)
- `closed` **0.190** · `public` **0.327** · `unknown` **0.245**
  Lectura: con todo el universo, la **vía pública** muestra más arrestos por el peso de categorías "proactivas" (operativos).

Subset comparable (sin "operativos")

- `Domestic = 1` → **0.192** (vs. `Domestic = 0` **0.160**)
- `closed` **0.170** (vs. `public` **0.153**, `unknown` **0.181**)
  Lectura: condicionando por tipo de delito, la hipótesis **se sostiene**: doméstico y cerrado ↑ tasa de arresto.

3.3.4 Capacidad predictiva (subset, muestra 100k)

- **PR-AUC (Average Precision): 0.185** (baseline por prevalencia **0.098**)
- **ROC-AUC: 0.693**

Interpretación: el modelo **prioriza** mejor que el azar (AP **1.9×** baseline) y ofrece separación **moderada** en un problema desbalanceado.

3.3.5 Umbrales

- **F1 máximo** *(thr 0.517)*
  - **Precision: 0.194** · **Recall: 0.478** · **F1: 0.276**
  - **Matriz de confusión (test=100k, pos=9,780):** TN=70,779 · FP=19,441 · FN=5,106 · TP=4,674
  - **Lectura:** punto "balanceado" útil para **ranking/triage**: captura ~48% de arrestos con ~19% de precisión (**2×** el baseline).

- **Precision 0.60**
  - No alcanzable con recall utilizable (recall 0) → **trade-off** típico con clase rara.

3.3.6 Factores asociados (coeficientes Logística, subset)

- **Señales positivas (↑ prob. arresto):** `Primary Type_CRIMINAL TRESPASS`, `WEAPONS VIOLATION`, `BATTERY`, `ASSAULT`, `PUBLIC PEACE VIOLATION`, `Domestic`, `cs_closed` (efecto pequeño positivo).
- **Señales negativas (↓ prob. arresto):** `CRIMINAL DAMAGE`, `MOTOR VEHICLE THEFT`, `ROBBERY`, `cs_public`, además de efectos menores por `District` y `Month`.
- **Conclusión EDA+coef.:** al retirar "operativos", el **tipo de lugar** y la naturaleza **doméstica** aportan señal coherente con la **hipótesis condicionada**; el **tipo de delito** sigue siendo dominante.

3.3.7 Conclusión

1. La **pregunta** es válida y el efecto de lugar/naturaleza depende del **universo de delitos**.
   - En el **total** (con operativos en vía pública): la hipótesis **no se cumple**.
   - En el **subset comparable**: la hipótesis **sí se sostiene** (doméstico y cerrado ↑ arresto).
2. La Regresión Logística muestra **capacidad predictiva moderada**: **PR-AUC 0.185** (> **0.098**) y **ROC-AUC 0.693**. Con el umbral de **F1**, **Recall** ~**48%** y **Precision** ~**19%** (**2×** baseline), útil para **priorizar** casos.
3. La elección de **umbral** depende del objetivo:
   - **Más cobertura (recall)** aceptar menor precisión.
   - **Menos falsos positivos (precision)** aceptar bajo recall.
4. **No implica causalidad**; potenciales sesgos por reporte y despliegue policial.

3.3.8 Recomendaciones

- **Refinar "lugar"**: OHE *top-K* de `Location Description`, manteniendo `cs_closed/cs_public`.
- **Interacciones**: `Domestic × cs_closed`, `Hour × Weekday`, `District × Primary Type`.
- **Calibración** de probabilidades para elegir umbrales con mayor confianza.
- Mantener **split temporal** y **muestreo estratificado** para comparabilidad.

3.4 Pregunta 3 – Hotspots espaciales (EDA espacial / clustering)

- Pregunta:

  ¿Dónde se concentran los delitos en Chicago y cómo han cambiado los hotspots geográficos a lo largo del tiempo?

- Hipótesis:

  Los hotspots delictivos presentan estabilidad en distritos centrales de la ciudad (zonas de mayor densidad y actividad económica), pero en las últimas dos décadas han surgido nuevos

focos en áreas periféricas debido a procesos de urbanización y desigualdad socioeconómica.

- Justificación:

  El análisis geoespacial permite detectar patrones de concentración y desplazamiento del crimen en la ciudad. Aquí puede usarse clustering (K-Means o DBSCAN), aunque como apoyo al EDA y no como modelo principal, ya que la pregunta se responde también con mapas de calor y análisis descriptivo.

3.4.1 Parametros

[75]:
```
# Clusters más grandes y significativos
MIN_SAMPLES = 30          # ↑ de 20 (más puntos para formar cluster)
MIN_CLUSTER_SIZE = 50     # ↑ de 30 (clusters más robustos)
EPS_MIN = 0.0005          # ↓ de 1e-5 (permitir clusters más densos)
EPS_MAX = 0.008           # ↑ de 0.05 (explorar áreas más amplias)
MAX_EPS_LIMIT = 0.015     # ↑ de 0.1 (para tipos muy dispersos)
BINARY_ITERS = 15
THRESH = 0.4              # ↓ de 0.5 (menos restrictivo)

# Límites para optimización
MAX_RECORDS_DIRECT = 200000
SAMPLE_SIZE = 100000

# ----- Lista y colores -----
valid_types = [
    'THEFT', 'BATTERY', 'CRIMINAL DAMAGE', 'NARCOTICS', 'ASSAULT',
    'OTHER OFFENSE', 'BURGLARY', 'MOTOR VEHICLE THEFT',
    'DECEPTIVE PRACTICE', 'ROBBERY', 'CRIMINAL TRESPASS',
    'WEAPONS VIOLATION'
]

color_dict = {
    'THEFT': '#1f78b4',
    'BATTERY': '#33a02c',
    'CRIMINAL DAMAGE': '#e31a1c',
    'NARCOTICS': '#ff7f00',
    'ASSAULT': '#6a3d9a',
    'OTHER OFFENSE': '#b15928',
    'BURGLARY': '#a6cee3',
    'MOTOR VEHICLE THEFT': '#fb9a99',
    'DECEPTIVE PRACTICE': '#fdbf6f',
    'ROBBERY': '#cab2d6',
    'CRIMINAL TRESPASS': '#b2df8a',
    'WEAPONS VIOLATION': '#000000'
}
```

3.4.2 Funciones

Aux

[76]:
```
def max_cluster_fraction_for_eps(coords, eps, min_samples):
    """Calcula la fracción máxima de cluster para un eps dado"""
    labels = DBSCAN(eps=eps, min_samples=min_samples, n_jobs=-1,
                    algorithm='ball_tree').fit_predict(coords)
    n = coords.shape[0]
    if n == 0:
        return 0.0, labels
    unique, counts = np.unique(labels, return_counts=True)
    mask = unique != -1
    if not mask.any():
        return 0.0, labels
    non_noise_counts = counts[mask]
    max_frac = non_noise_counts.max() / n
    return max_frac, labels


def find_optimal_eps_on_sample(coords, sample_size=SAMPLE_SIZE):
    """Encuentra eps óptimo usando una muestra representativa"""
    n = coords.shape[0]
    if n <= sample_size:
        sample_coords = coords
    else:
        np.random.seed(42)
        idx = np.random.choice(n, size=sample_size, replace=False)
        sample_coords = coords[idx]

    frac_min, _ = max_cluster_fraction_for_eps(sample_coords, EPS_MIN,␣
 ↪MIN_SAMPLES)
    if frac_min > THRESH:
        return EPS_MIN

    cur_max = EPS_MAX
    frac_max, _ = max_cluster_fraction_for_eps(sample_coords, cur_max,␣
 ↪MIN_SAMPLES)

    while frac_max <= THRESH and cur_max < MAX_EPS_LIMIT:
        cur_max = min(cur_max * 2, MAX_EPS_LIMIT)
        frac_max, _ = max_cluster_fraction_for_eps(sample_coords, cur_max,␣
 ↪MIN_SAMPLES)
        if cur_max >= MAX_EPS_LIMIT and frac_max <= THRESH:
            break

    if frac_max <= THRESH:
        return cur_max
```

```
    lo, hi = EPS_MIN, cur_max
    for _ in range(BINARY_ITERS):
        mid = (lo + hi) / 2.0
        frac_mid, _ = max_cluster_fraction_for_eps(sample_coords, mid,␣
↪MIN_SAMPLES)
        if frac_mid <= THRESH:
            lo = mid
        else:
            hi = mid
    return lo


def process_large_dataset_in_chunks(coords, eps_final, min_samples,␣
↪chunk_size=150000):
    """Procesa datasets grandes por chunks espaciales"""
    n = coords.shape[0]
    print(f"    Procesando {n:,} puntos en chunks de {chunk_size:,}...")

    lat_min, lat_max = coords[:, 0].min(), coords[:, 0].max()
    lon_min, lon_max = coords[:, 1].min(), coords[:, 1].max()

    n_chunks = int(np.ceil(n / chunk_size))
    n_splits = int(np.ceil(np.sqrt(n_chunks)))

    lat_bins = np.linspace(lat_min, lat_max, n_splits + 1)
    lon_bins = np.linspace(lon_min, lon_max, n_splits + 1)

    all_labels = np.full(n, -1, dtype=int)
    cluster_id_offset = 0

    for i in range(n_splits):
        for j in range(n_splits):
            overlap = eps_final * 2
            lat_mask = (coords[:, 0] >= lat_bins[i] - overlap) & (coords[:, 0]␣
↪<= lat_bins[i+1] + overlap)
            lon_mask = (coords[:, 1] >= lon_bins[j] - overlap) & (coords[:, 1]␣
↪<= lon_bins[j+1] + overlap)
            chunk_mask = lat_mask & lon_mask

            if chunk_mask.sum() == 0:
                continue

            chunk_coords = coords[chunk_mask]
            chunk_labels = DBSCAN(eps=eps_final, min_samples=min_samples,
                                  n_jobs=-1, algorithm='ball_tree').
↪fit_predict(chunk_coords)
```

```
            chunk_labels[chunk_labels != -1] += cluster_id_offset

            center_mask = (coords[:, 0] >= lat_bins[i]) & (coords[:, 0] <=␣
↪lat_bins[i+1]) & \
                          (coords[:, 1] >= lon_bins[j]) & (coords[:, 1] <=␣
↪lon_bins[j+1])

            local_center_mask = (chunk_coords[:, 0] >= lat_bins[i]) &␣
↪(chunk_coords[:, 0] <= lat_bins[i+1]) & \
                                (chunk_coords[:, 1] >= lon_bins[j]) &␣
↪(chunk_coords[:, 1] <= lon_bins[j+1])

            all_labels[center_mask] = chunk_labels[local_center_mask]

            if chunk_labels.max() != -1:
                cluster_id_offset = chunk_labels.max() + 1

            print(f"      Chunk ({i+1},{j+1})/{n_splits}x{n_splits}:␣
↪{chunk_mask.sum():,} pts")

    return all_labels
```

Clusters

[77]:
```
def clusterizar_por_tipo(df, year=None, plot=True):
    """
    Ejecuta clustering DBSCAN adaptativo por tipo de delito.
    Retorna:
        - clusters_df: DataFrame con cada punto y su cluster asignado
        - clusters_summary: dict con # de clusters válidos por tipo
        - eps_used: dict con eps usado por tipo
    """
    print("Filtrando datos geográficos...")
    mask = (df['Latitude'] > 40) & (df['Longitude'] > -89)

    if year is not None:
        mask &= (pd.to_datetime(df['Date']).dt.year == year)

    df_filtered = df.loc[mask, ['Primary Type', 'Latitude', 'Longitude', 'ID']].
↪copy()
    print(f"Registros después del filtro: {len(df_filtered):,}\n")

    clusters_summary = {}
    eps_used = {}
    clusters_records = []

    if plot:
```

```
        plt.figure(figsize=(12, 12))

    for idx, ptype in enumerate(valid_types):
        print(f"\n{'='*60}")
        print(f"[{idx+1}/{len(valid_types)}] Procesando: {ptype}")
        print(f"{'='*60}")

        mask_type = df_filtered['Primary Type'] == ptype
        subset_idx = df_filtered.index[mask_type]
        n = len(subset_idx)

        if n < MIN_SAMPLES:
            clusters_summary[ptype] = 0
            print(f"  SKIP (n={n:,} < MIN_SAMPLES={MIN_SAMPLES})")
            continue

        print(f"  Registros: {n:,}")
        coords = df_filtered.loc[subset_idx, ['Latitude', 'Longitude']].
↪to_numpy()

        eps_final = find_optimal_eps_on_sample(coords)
        print(f"    eps_final = {eps_final:.6f}")

        if n <= MAX_RECORDS_DIRECT:
            labels_final = DBSCAN(eps=eps_final, min_samples=MIN_SAMPLES,
                                  n_jobs=-1, algorithm='ball_tree').
↪fit_predict(coords)
        else:
            labels_final = process_large_dataset_in_chunks(coords, eps_final,␣
↪MIN_SAMPLES)

        unique_labels, counts = np.unique(labels_final, return_counts=True)
        cluster_sizes = dict(zip(unique_labels, counts))

        valid_cluster_ids = [cid for cid, size in cluster_sizes.items()
                             if cid != -1 and size > MIN_CLUSTER_SIZE]

        num_valid = len(valid_cluster_ids)
        clusters_summary[ptype] = num_valid
        eps_used[ptype] = eps_final

        print(f"     Clusters válidos: {num_valid}")

        for i, cluster_label in enumerate(labels_final):
            clusters_records.append({
                'Primary Type': ptype,
                'Latitude': coords[i, 0],
```

```
                'Longitude': coords[i, 1],
                'cluster': cluster_label,
                'ID': df_filtered.loc[subset_idx[i], 'ID']
            })

        if plot and num_valid > 0:
            valid_mask = np.isin(labels_final, valid_cluster_ids)
            valid_coords = coords[valid_mask]
            plt.scatter(
                valid_coords[:, 1],
                valid_coords[:, 0],
                s=6,
                alpha=0.7,
                c=color_dict.get(ptype, '#777777'),
                label=f"{ptype} ({num_valid})"
            )

        del coords, labels_final, mask_type
        gc.collect()

    clusters_df = pd.DataFrame(clusters_records)
    print(f"\nclusters_df creado: {len(clusters_df):,} registros")

    if plot:
        plt.legend(markerscale=4, bbox_to_anchor=(1.05, 1), loc='upper left')
        plt.title("Clusters DBSCAN adaptativo por tipo de delito")
        plt.xlabel("Longitude")
        plt.ylabel("Latitude")
        plt.tight_layout()
        plt.show()

    return clusters_df, clusters_summary, eps_used
```

Crear video

[82]:
```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import pandas as pd

def crear_video_clusters(
    df_list,
    output_path="clusters_video.mp4",
    size_factor=0.5,
    fps=1,
    lat_min=None,
    lat_max=None,
    lon_min=None,
    lon_max=None
```

```
):
    """
    Genera un video a partir de una lista de DataFrames de clusters.

    Parámetros:
    -----------
    df_list : list[pd.DataFrame]
        Lista de DataFrames (ej. clusters_df por cada año).
    output_path : str
        Ruta del archivo de salida del video (mp4).
    size_factor : float
        Factor de escalado del tamaño de los círculos.
    fps : int
        Cuadros por segundo del video.
    lat_min, lat_max, lon_min, lon_max : float | None
        Límites fijos de los ejes. Si son None, se calculan automáticamente.
    """

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

    def plot_frame(i):
        ax.clear()
        df = df_list[i]

        # Filtrar clusters válidos
        clusters_valid = df[df['cluster'] != -1].copy()

        # Agrupar para centroides
        cluster_summary = (
            clusters_valid.groupby(['Primary Type', 'cluster'])
            .agg(
                Longitude_mean=('Longitude', 'mean'),
                Latitude_mean=('Latitude', 'mean'),
                size=('cluster', 'size')
            )
            .reset_index()
        )

        cluster_summary['plot_size'] = 5 + cluster_summary['size'] * size_factor

        for ptype, group in cluster_summary.groupby("Primary Type"):
            ax.scatter(
                group['Longitude_mean'],
                group['Latitude_mean'],
                s=group['plot_size'],
                alpha=0.5,
                c=color_dict.get(ptype, '#777777'),
```

```
                label=f"{ptype}"
            )

        # Fijar límites si están definidos
        if lat_min is not None and lat_max is not None:
            ax.set_ylim(lat_min, lat_max)
        if lon_min is not None and lon_max is not None:
            ax.set_xlim(lon_min, lon_max)

        ax.set_title(f"Clusters como círculos (Año {i+2001})", fontsize=14)
        ax.set_xlabel("Longitude")
        ax.set_ylabel("Latitude")
        ax.legend(markerscale=0.5, bbox_to_anchor=(1.05, 1), loc='upper left')

    ani = animation.FuncAnimation(fig, plot_frame, frames=len(df_list),␣
 ↪repeat=False)

    ani.save(output_path, writer="ffmpeg", fps=fps)
    plt.close(fig)

    print(f"Video exportado a {output_path}")
```

3.4.3 Llamada

[86]:
```
# DF clean
df_pregunta3 = df.copy()

lon_min = df_pregunta3['Longitude'].min()
lon_max = df_pregunta3['Longitude'].max()
lat_min = df_pregunta3['Latitude'].min()
lat_max = df_pregunta3['Latitude'].max()

lista_clusters = []

inicio = 2001 # 2001
fin = 2025 #2025

for year in range(inicio,fin+1):
  clusters_df, clusters_summary, eps_used = clusterizar_por_tipo(df_pregunta3,␣
 ↪year=year, plot=False)
  lista_clusters.append(clusters_df)
  print(f"Año terminado: {year}")

crear_video_clusters(lista_clusters, lat_min=lat_min, lat_max=lat_max,␣
 ↪lon_min=lon_min, lon_max=lon_max)
```

```
Filtrando datos geográficos…
Registros después del filtro: 482,864
```

```
============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 98,447
    eps_final = 0.002054
      Clusters válidos: 165

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 93,049
    eps_final = 0.002317
      Clusters válidos: 73

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 55,590
    eps_final = 0.002927
      Clusters válidos: 37

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 50,318
    eps_final = 0.003605
      Clusters válidos: 32

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 31,260
    eps_final = 0.004185
      Clusters válidos: 24

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 29,557
    eps_final = 0.004339
      Clusters válidos: 21

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 25,943
```

```
    eps_final = 0.004254
      Clusters válidos: 18

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 27,282
    eps_final = 0.004148
      Clusters válidos: 24

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 14,773
    eps_final = 0.005593
      Clusters válidos: 42

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 18,292
    eps_final = 0.005595
      Clusters válidos: 11

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 13,182
    eps_final = 0.005631
      Clusters válidos: 33

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,246
    eps_final = 0.010029
      Clusters válidos: 7

clusters_df creado: 461,939 registros
Año terminado: 2001
Filtrando datos geográficos…
Registros después del filtro: 471,517


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 95,362
```

```
    eps_final = 0.002117
      Clusters válidos: 138

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 91,721
    eps_final = 0.002640
      Clusters válidos: 51

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 54,709
    eps_final = 0.003194
      Clusters válidos: 35

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 49,824
    eps_final = 0.002561
      Clusters válidos: 56

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 30,734
    eps_final = 0.004071
      Clusters válidos: 22

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 31,805
    eps_final = 0.003936
      Clusters válidos: 28

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 25,221
    eps_final = 0.004419
      Clusters válidos: 17

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
```

```
  Registros: 23,254
    eps_final = 0.004701
     Clusters válidos: 16

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,237
    eps_final = 0.005388
     Clusters válidos: 33

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 17,740
    eps_final = 0.005600
     Clusters válidos: 10

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 13,570
    eps_final = 0.005568
     Clusters válidos: 28

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,149
    eps_final = 0.011922
     Clusters válidos: 7

clusters_df creado: 451,326 registros
Año terminado: 2002
Filtrando datos geográficos…
Registros después del filtro: 472,032


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 97,804
    eps_final = 0.002083
     Clusters válidos: 132

============================================================
[2/12] Procesando: BATTERY
============================================================
```

```
  Registros: 87,835
    eps_final = 0.002796
      Clusters válidos: 39

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 54,694
    eps_final = 0.003234
      Clusters válidos: 30

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 53,763
    eps_final = 0.002662
      Clusters válidos: 55

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 29,291
    eps_final = 0.004171
      Clusters válidos: 21

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 30,912
    eps_final = 0.004201
      Clusters válidos: 26

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 25,010
    eps_final = 0.004521
      Clusters válidos: 16

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 22,676
    eps_final = 0.004256
      Clusters válidos: 17

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
```

```
============================================================
  Registros: 13,254
    eps_final = 0.005792
      Clusters válidos: 35

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 17,235
    eps_final = 0.005114
      Clusters válidos: 13

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 14,687
    eps_final = 0.005451
      Clusters válidos: 23

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,199
    eps_final = 0.011175
      Clusters válidos: 6

clusters_df creado: 451,360 registros
Año terminado: 2003
Filtrando datos geográficos…
Registros después del filtro: 467,196


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 94,641
    eps_final = 0.002108
      Clusters válidos: 128

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 86,967
    eps_final = 0.002754
      Clusters válidos: 51

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
```

```
============================================================
  Registros: 53,069
    eps_final = 0.003086
      Clusters válidos: 41

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 56,882
    eps_final = 0.002748
      Clusters válidos: 45

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 28,792
    eps_final = 0.004450
      Clusters válidos: 25

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 29,455
    eps_final = 0.004519
      Clusters válidos: 20

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 24,517
    eps_final = 0.004368
      Clusters válidos: 19

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 22,747
    eps_final = 0.004189
      Clusters válidos: 25

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,079
    eps_final = 0.005533
      Clusters válidos: 42

============================================================
```

```
[10/12] Procesando: ROBBERY
============================================================
  Registros: 15,951
    eps_final = 0.005981
      Clusters válidos: 13

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 15,834
    eps_final = 0.005508
      Clusters válidos: 24

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,284
    eps_final = 0.009728
      Clusters válidos: 5

clusters_df creado: 446,218 registros
Año terminado: 2004
Filtrando datos geográficos…
Registros después del filtro: 449,926


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 84,303
    eps_final = 0.002235
      Clusters válidos: 107

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 83,636
    eps_final = 0.002826
      Clusters válidos: 51

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 54,351
    eps_final = 0.003114
      Clusters válidos: 36

============================================================
```

```
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 55,950
    eps_final = 0.002879
      Clusters válidos: 47

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 26,966
    eps_final = 0.004571
      Clusters válidos: 23

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 27,828
    eps_final = 0.004228
      Clusters válidos: 21

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 25,413
    eps_final = 0.004564
      Clusters válidos: 20

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 22,384
    eps_final = 0.004654
      Clusters válidos: 20

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,153
    eps_final = 0.005743
      Clusters válidos: 29

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 15,988
    eps_final = 0.005486
      Clusters válidos: 15
```

```
============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 16,569
    eps_final = 0.005339
      Clusters válidos: 23

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,074
    eps_final = 0.010309
      Clusters válidos: 4

clusters_df creado: 430,615 registros
Año terminado: 2005
Filtrando datos geográficos…
Registros después del filtro: 445,560


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 85,232
    eps_final = 0.002318
      Clusters válidos: 109

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 80,572
    eps_final = 0.002829
      Clusters válidos: 51

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 57,074
    eps_final = 0.003021
      Clusters válidos: 33

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 55,332
    eps_final = 0.003416
      Clusters válidos: 31
```

```
============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 25,930
    eps_final = 0.004539
     Clusters válidos: 23

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 27,048
    eps_final = 0.004579
     Clusters válidos: 18

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 24,304
    eps_final = 0.004706
     Clusters válidos: 14

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 21,785
    eps_final = 0.004332
     Clusters válidos: 22

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,266
    eps_final = 0.005724
     Clusters válidos: 37

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 15,945
    eps_final = 0.005472
     Clusters válidos: 15

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 14,488
    eps_final = 0.005358
     Clusters válidos: 22
```

```
============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,817
    eps_final = 0.010417
     Clusters válidos: 6

clusters_df creado: 424,793 registros
Año terminado: 2006
Filtrando datos geográficos…
Registros después del filtro: 435,703


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 84,599
    eps_final = 0.002201
     Clusters válidos: 117

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 79,564
    eps_final = 0.003152
     Clusters válidos: 36

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 53,716
    eps_final = 0.003154
     Clusters válidos: 40

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 54,412
    eps_final = 0.003889
     Clusters válidos: 27

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 26,306
    eps_final = 0.004869
     Clusters válidos: 21
```

```
============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 26,831
    eps_final = 0.004489
      Clusters válidos: 26

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 24,838
    eps_final = 0.004866
      Clusters válidos: 12

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 18,553
    eps_final = 0.005203
      Clusters válidos: 14

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,868
    eps_final = 0.005650
      Clusters válidos: 30

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 15,445
    eps_final = 0.004666
      Clusters válidos: 19

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 13,690
    eps_final = 0.005543
      Clusters válidos: 21

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,547
    eps_final = 0.009853
```

```
     Clusters válidos: 4

clusters_df creado: 415,369 registros
Año terminado: 2007
Filtrando datos geográficos…
Registros después del filtro: 419,839


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 86,361
    eps_final = 0.002416
      Clusters válidos: 105

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 75,434
    eps_final = 0.003112
      Clusters válidos: 38

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 52,187
    eps_final = 0.003278
      Clusters válidos: 31

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 45,055
    eps_final = 0.004216
      Clusters válidos: 24

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 25,254
    eps_final = 0.004701
      Clusters válidos: 21

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 26,105
    eps_final = 0.004800
```

```
     Clusters válidos: 16

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 25,992
    eps_final = 0.004260
     Clusters válidos: 23

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 18,614
    eps_final = 0.005057
     Clusters válidos: 15

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 14,268
    eps_final = 0.005938
     Clusters válidos: 23

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 16,574
    eps_final = 0.004937
     Clusters válidos: 19

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 12,219
    eps_final = 0.006248
     Clusters válidos: 16

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,823
    eps_final = 0.008782
     Clusters válidos: 4

clusters_df creado: 401,886 registros
Año terminado: 2008
Filtrando datos geográficos…
Registros después del filtro: 385,951
```

```
============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 79,285
    eps_final = 0.002389
      Clusters válidos: 107

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 67,744
    eps_final = 0.003342
      Clusters válidos: 38

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 47,245
    eps_final = 0.003177
      Clusters válidos: 33

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 42,153
    eps_final = 0.004234
      Clusters válidos: 23

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 22,606
    eps_final = 0.005190
      Clusters válidos: 16

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 25,243
    eps_final = 0.004827
      Clusters válidos: 17

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 26,474
```

```
    eps_final = 0.004152
      Clusters válidos: 22

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 15,310
    eps_final = 0.005946
      Clusters válidos: 10

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,272
    eps_final = 0.005980
      Clusters válidos: 27

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 15,842
    eps_final = 0.005307
      Clusters válidos: 15

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 10,745
    eps_final = 0.006183
      Clusters válidos: 20

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,079
    eps_final = 0.008473
      Clusters válidos: 6

clusters_df creado: 369,998 registros
Año terminado: 2009
Filtrando datos geográficos…
Registros después del filtro: 369,958


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 76,727
```

```
    eps_final = 0.002245
      Clusters válidos: 106

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 65,380
    eps_final = 0.003163
      Clusters válidos: 38

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 40,642
    eps_final = 0.003442
      Clusters válidos: 30

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 43,385
    eps_final = 0.004202
      Clusters válidos: 24

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 21,533
    eps_final = 0.005067
      Clusters válidos: 17

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 22,006
    eps_final = 0.005048
      Clusters válidos: 17

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 26,417
    eps_final = 0.004114
      Clusters válidos: 24

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
```

```
  Registros: 19,022
    eps_final = 0.005095
     Clusters válidos: 12

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 12,325
    eps_final = 0.006180
     Clusters válidos: 19

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 14,267
    eps_final = 0.005713
     Clusters válidos: 10

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 9,401
    eps_final = 0.006310
     Clusters válidos: 19

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,703
    eps_final = 0.007897
     Clusters válidos: 4

clusters_df creado: 354,808 registros
Año terminado: 2010
Filtrando datos geográficos…
Registros después del filtro: 351,049


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 74,888
    eps_final = 0.002306
     Clusters válidos: 110

============================================================
[2/12] Procesando: BATTERY
============================================================
```

```
  Registros: 60,434
    eps_final = 0.003442
      Clusters válidos: 28

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 37,319
    eps_final = 0.003763
      Clusters válidos: 26

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 38,570
    eps_final = 0.004817
      Clusters válidos: 26

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 20,403
    eps_final = 0.005518
      Clusters válidos: 15

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 20,189
    eps_final = 0.005264
      Clusters válidos: 16

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 26,613
    eps_final = 0.004093
      Clusters válidos: 23

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 19,383
    eps_final = 0.004915
      Clusters válidos: 15

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
```

```
============================================================
  Registros: 12,383
    eps_final = 0.006588
      Clusters válidos: 17

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 13,973
    eps_final = 0.005605
      Clusters válidos: 11

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 8,655
    eps_final = 0.007968
      Clusters válidos: 5

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,879
    eps_final = 0.008847
      Clusters válidos: 4

clusters_df creado: 336,689 registros
Año terminado: 2011
Filtrando datos geográficos…
Registros después del filtro: 335,122


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 75,119
    eps_final = 0.002507
      Clusters válidos: 80

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 59,119
    eps_final = 0.003522
      Clusters válidos: 28

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
```

```
============================================================
  Registros: 35,849
    eps_final = 0.003698
      Clusters válidos: 27

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 35,467
    eps_final = 0.003885
      Clusters válidos: 24

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 19,893
    eps_final = 0.005129
      Clusters válidos: 19

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 17,513
    eps_final = 0.005514
      Clusters válidos: 15

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 22,832
    eps_final = 0.004642
      Clusters válidos: 21

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 16,483
    eps_final = 0.005019
      Clusters válidos: 21

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,268
    eps_final = 0.006646
      Clusters válidos: 13

============================================================
```

```
[10/12] Procesando: ROBBERY
============================================================
  Registros: 13,468
    eps_final = 0.005564
     Clusters válidos: 11

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 8,215
    eps_final = 0.007466
     Clusters válidos: 11

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,904
    eps_final = 0.007750
     Clusters válidos: 4

clusters_df creado: 321,130 registros
Año terminado: 2012
Filtrando datos geográficos…
Registros después del filtro: 306,356


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 71,456
    eps_final = 0.002583
     Clusters válidos: 79

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 53,988
    eps_final = 0.003767
     Clusters válidos: 24

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 30,848
    eps_final = 0.004257
     Clusters válidos: 20

============================================================
```

```
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 34,103
    eps_final = 0.002174
      Clusters válidos: 53

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 17,967
    eps_final = 0.005139
      Clusters válidos: 18

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 18,018
    eps_final = 0.005857
      Clusters válidos: 14

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 17,885
    eps_final = 0.005645
      Clusters válidos: 13

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 12,574
    eps_final = 0.005939
      Clusters válidos: 11

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,243
    eps_final = 0.006198
      Clusters válidos: 17

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 11,817
    eps_final = 0.005545
      Clusters válidos: 15
```

```
============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 8,135
    eps_final = 0.007187
      Clusters válidos: 12

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,245
    eps_final = 0.009236
      Clusters válidos: 3

clusters_df creado: 293,279 registros
Año terminado: 2013
Filtrando datos geográficos…
Registros después del filtro: 273,611


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 61,347
    eps_final = 0.002664
      Clusters válidos: 84

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 49,410
    eps_final = 0.003421
      Clusters válidos: 32

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 27,777
    eps_final = 0.004301
      Clusters válidos: 19

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 28,904
    eps_final = 0.002367
      Clusters válidos: 48
```

```
============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 16,889
    eps_final = 0.005658
      Clusters válidos: 14

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 16,966
    eps_final = 0.005784
      Clusters válidos: 21

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 14,559
    eps_final = 0.006255
      Clusters válidos: 10

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 9,894
    eps_final = 0.006187
      Clusters válidos: 16

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 14,854
    eps_final = 0.005838
      Clusters válidos: 21

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 9,791
    eps_final = 0.006381
      Clusters válidos: 12

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 7,536
    eps_final = 0.007329
      Clusters válidos: 12
```

```
============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,109
    eps_final = 0.008486
     Clusters válidos: 4

clusters_df creado: 261,036 registros
Año terminado: 2014
Filtrando datos geográficos…
Registros después del filtro: 257,767


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 56,696
    eps_final = 0.002701
     Clusters válidos: 86

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 48,816
    eps_final = 0.003537
     Clusters válidos: 31

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 28,588
    eps_final = 0.003874
     Clusters válidos: 27

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 21,607
    eps_final = 0.002500
     Clusters válidos: 29

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 16,992
    eps_final = 0.005284
     Clusters válidos: 20
```

```
============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 17,279
    eps_final = 0.005019
      Clusters válidos: 16

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 13,101
    eps_final = 0.006063
      Clusters válidos: 11

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 10,003
    eps_final = 0.006780
      Clusters válidos: 8

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 13,945
    eps_final = 0.005790
      Clusters válidos: 15

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 9,631
    eps_final = 0.006381
      Clusters válidos: 12

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 6,391
    eps_final = 0.008546
      Clusters válidos: 8

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,333
    eps_final = 0.008773
```

```
     Clusters válidos: 4

clusters_df creado: 246,382 registros
Año terminado: 2015
Filtrando datos geográficos…
Registros después del filtro: 266,443


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 61,034
    eps_final = 0.002445
      Clusters válidos: 108

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 50,245
    eps_final = 0.003533
      Clusters válidos: 38

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 30,931
    eps_final = 0.003803
      Clusters válidos: 27

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 13,257
    eps_final = 0.003631
      Clusters válidos: 19

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 18,720
    eps_final = 0.005145
      Clusters válidos: 17

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 17,201
    eps_final = 0.005444
```

```
     Clusters válidos: 15

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 14,278
    eps_final = 0.005415
     Clusters válidos: 17

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 11,269
    eps_final = 0.006146
     Clusters válidos: 15

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 17,373
    eps_final = 0.004901
     Clusters válidos: 30

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 11,950
    eps_final = 0.006194
     Clusters válidos: 6

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 6,295
    eps_final = 0.008666
     Clusters válidos: 4

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 3,449
    eps_final = 0.012107
     Clusters válidos: 3

clusters_df creado: 256,002 registros
Año terminado: 2016
Filtrando datos geográficos…
Registros después del filtro: 264,132
```

```
============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 63,583
    eps_final = 0.002452
      Clusters válidos: 106

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 49,138
    eps_final = 0.003474
      Clusters válidos: 34

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 28,955
    eps_final = 0.004267
      Clusters válidos: 17

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 11,475
    eps_final = 0.002832
      Clusters válidos: 17

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 19,250
    eps_final = 0.005478
      Clusters válidos: 14

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 16,962
    eps_final = 0.005201
      Clusters válidos: 17

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 12,946
```

```
    eps_final = 0.006079
      Clusters válidos: 11

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 11,339
    eps_final = 0.005451
      Clusters válidos: 17

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 16,966
    eps_final = 0.004849
      Clusters válidos: 30

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 11,868
    eps_final = 0.005253
      Clusters válidos: 10

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 6,798
    eps_final = 0.008068
      Clusters válidos: 6

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,682
    eps_final = 0.008431
      Clusters válidos: 3

clusters_df creado: 253,962 registros
Año terminado: 2017
Filtrando datos geográficos…
Registros después del filtro: 262,882


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 64,021
```

```
    eps_final = 0.002473
     Clusters válidos: 106

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 49,713
    eps_final = 0.003545
     Clusters válidos: 35

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 27,698
    eps_final = 0.004131
     Clusters válidos: 22

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 12,797
    eps_final = 0.002954
     Clusters válidos: 18

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 20,342
    eps_final = 0.005033
     Clusters válidos: 19

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 16,951
    eps_final = 0.005388
     Clusters válidos: 19

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 11,681
    eps_final = 0.005938
     Clusters válidos: 12

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
```

```
  Registros: 9,933
    eps_final = 0.006258
      Clusters válidos: 14

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 17,220
    eps_final = 0.004712
      Clusters válidos: 37

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 9,678
    eps_final = 0.006422
      Clusters válidos: 13

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 6,881
    eps_final = 0.008366
      Clusters válidos: 8

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 5,444
    eps_final = 0.008206
      Clusters válidos: 5

clusters_df creado: 252,359 registros
Año terminado: 2018
Filtrando datos geográficos…
Registros después del filtro: 258,186


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 61,675
    eps_final = 0.002484
      Clusters válidos: 99

============================================================
[2/12] Procesando: BATTERY
============================================================
```

```
  Registros: 49,473
    eps_final = 0.003397
     Clusters válidos: 32

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 26,611
    eps_final = 0.004497
     Clusters válidos: 15

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 14,995
    eps_final = 0.002533
     Clusters válidos: 25

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 20,601
    eps_final = 0.004660
     Clusters válidos: 24

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 16,727
    eps_final = 0.005500
     Clusters válidos: 11

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 9,632
    eps_final = 0.007078
     Clusters válidos: 7

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 8,962
    eps_final = 0.006840
     Clusters válidos: 11

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
```

```
============================================================
  Registros: 17,256
    eps_final = 0.004675
      Clusters válidos: 28

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 7,989
    eps_final = 0.006846
      Clusters válidos: 12

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 6,805
    eps_final = 0.007986
      Clusters válidos: 10

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 6,338
    eps_final = 0.007250
      Clusters válidos: 6

clusters_df creado: 247,064 registros
Año terminado: 2019
Filtrando datos geográficos…
Registros después del filtro: 206,948


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 40,171
    eps_final = 0.003182
      Clusters válidos: 53

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 41,320
    eps_final = 0.003622
      Clusters válidos: 29

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
```

```
============================================================
  Registros: 24,663
    eps_final = 0.004353
      Clusters válidos: 21

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 7,268
    eps_final = 0.003186
      Clusters válidos: 6

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 18,169
    eps_final = 0.004781
      Clusters válidos: 21

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 12,281
    eps_final = 0.005826
      Clusters válidos: 11

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 8,623
    eps_final = 0.007209
      Clusters válidos: 9

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 9,885
    eps_final = 0.006055
      Clusters válidos: 6

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 15,859
    eps_final = 0.005499
      Clusters válidos: 25

============================================================
```

```
[10/12] Procesando: ROBBERY
============================================================
  Registros: 7,833
    eps_final = 0.007253
     Clusters válidos: 11

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 4,148
    eps_final = 0.010492
     Clusters válidos: 6

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 8,417
    eps_final = 0.006589
     Clusters válidos: 7

clusters_df creado: 198,637 registros
Año terminado: 2020
Filtrando datos geográficos…
Registros después del filtro: 202,041


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 39,248
    eps_final = 0.003188
     Clusters válidos: 57

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 40,244
    eps_final = 0.003600
     Clusters válidos: 31

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 24,833
    eps_final = 0.004532
     Clusters válidos: 19

============================================================
```

```
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 4,091
    eps_final = 0.004493
      Clusters válidos: 2

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 20,230
    eps_final = 0.004679
      Clusters válidos: 24

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 13,586
    eps_final = 0.005868
      Clusters válidos: 10

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 6,572
    eps_final = 0.008536
      Clusters válidos: 7

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 10,479
    eps_final = 0.005778
      Clusters válidos: 9

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 15,027
    eps_final = 0.005480
      Clusters válidos: 23

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 7,892
    eps_final = 0.006655
      Clusters válidos: 14
```

```
============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 3,375
    eps_final = 0.010969
      Clusters válidos: 7

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 8,942
    eps_final = 0.006202
      Clusters válidos: 8

clusters_df creado: 194,519 registros
Año terminado: 2021
Filtrando datos geográficos…
Registros después del filtro: 234,245


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 53,624
    eps_final = 0.002733
      Clusters válidos: 93

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 40,801
    eps_final = 0.003898
      Clusters válidos: 25

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 27,026
    eps_final = 0.003987
      Clusters válidos: 26

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 4,037
    eps_final = 0.005024
      Clusters válidos: 3
```

```
============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 20,740
    eps_final = 0.004926
     Clusters válidos: 15

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 14,367
    eps_final = 0.005792
     Clusters válidos: 12

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 7,564
    eps_final = 0.007719
     Clusters válidos: 8

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 21,387
    eps_final = 0.003917
     Clusters válidos: 22

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 14,736
    eps_final = 0.005478
     Clusters válidos: 31

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 8,951
    eps_final = 0.006895
     Clusters válidos: 9

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 4,201
    eps_final = 0.009795
     Clusters válidos: 10
```

```
============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 8,710
    eps_final = 0.006131
     Clusters válidos: 7

clusters_df creado: 226,144 registros
Año terminado: 2022
Filtrando datos geográficos…
Registros después del filtro: 259,892


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 57,194
    eps_final = 0.002565
     Clusters válidos: 97

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 44,080
    eps_final = 0.003640
     Clusters válidos: 34

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 30,009
    eps_final = 0.003513
     Clusters válidos: 34

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 5,219
    eps_final = 0.004258
     Clusters válidos: 4

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 22,550
    eps_final = 0.004792
     Clusters válidos: 19
```

```
============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 15,599
    eps_final = 0.005484
      Clusters válidos: 17

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 7,457
    eps_final = 0.007720
      Clusters válidos: 11

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 29,170
    eps_final = 0.003402
      Clusters válidos: 35

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 15,521
    eps_final = 0.005267
      Clusters válidos: 24

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 11,029
    eps_final = 0.005092
      Clusters válidos: 21

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 4,700
    eps_final = 0.009496
      Clusters válidos: 11

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 8,594
    eps_final = 0.006043
```

```
     Clusters válidos: 4

clusters_df creado: 251,122 registros
Año terminado: 2023
Filtrando datos geográficos…
Registros después del filtro: 256,479


============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 60,293
    eps_final = 0.002492
      Clusters válidos: 104

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 45,956
    eps_final = 0.003300
      Clusters válidos: 30

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 28,448
    eps_final = 0.003593
      Clusters válidos: 30

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 5,957
    eps_final = 0.003956
      Clusters válidos: 5

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 23,376
    eps_final = 0.004798
      Clusters válidos: 16

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 16,989
    eps_final = 0.005376
```

```
     Clusters válidos: 11

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 8,397
    eps_final = 0.006968
     Clusters válidos: 12

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 21,646
    eps_final = 0.003929
     Clusters válidos: 22

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 14,830
    eps_final = 0.005516
     Clusters válidos: 26

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 9,092
    eps_final = 0.006569
     Clusters válidos: 14

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 4,922
    eps_final = 0.008625
     Clusters válidos: 9

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 7,822
    eps_final = 0.006631
     Clusters válidos: 10

clusters_df creado: 247,728 registros
Año terminado: 2024
Filtrando datos geográficos…
Registros después del filtro: 170,226
```

```
============================================================
[1/12] Procesando: THEFT
============================================================
  Registros: 39,840
    eps_final = 0.003096
      Clusters válidos: 76

============================================================
[2/12] Procesando: BATTERY
============================================================
  Registros: 30,966
    eps_final = 0.004051
      Clusters válidos: 32

============================================================
[3/12] Procesando: CRIMINAL DAMAGE
============================================================
  Registros: 18,674
    eps_final = 0.004258
      Clusters válidos: 28

============================================================
[4/12] Procesando: NARCOTICS
============================================================
  Registros: 5,602
    eps_final = 0.004064
      Clusters válidos: 8

============================================================
[5/12] Procesando: ASSAULT
============================================================
  Registros: 15,800
    eps_final = 0.005038
      Clusters válidos: 18

============================================================
[6/12] Procesando: OTHER OFFENSE
============================================================
  Registros: 12,134
    eps_final = 0.006041
      Clusters válidos: 14

============================================================
[7/12] Procesando: BURGLARY
============================================================
  Registros: 6,566
```

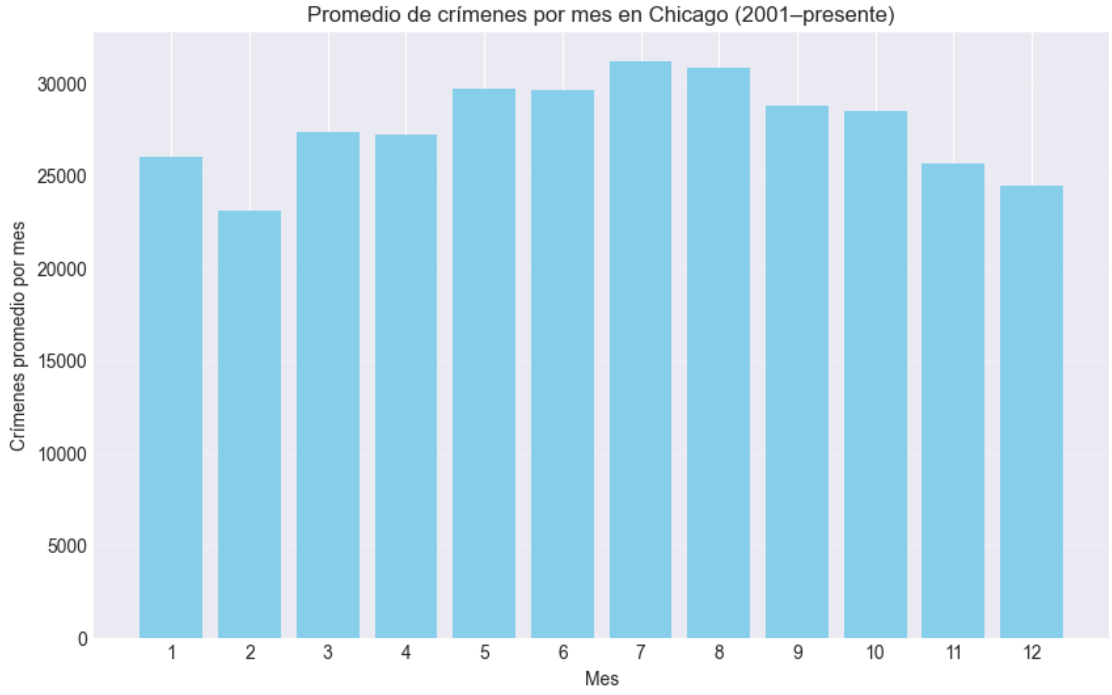
```
    eps_final = 0.007443
      Clusters válidos: 12

============================================================
[8/12] Procesando: MOTOR VEHICLE THEFT
============================================================
  Registros: 12,045
    eps_final = 0.005051
      Clusters válidos: 15

============================================================
[9/12] Procesando: DECEPTIVE PRACTICE
============================================================
  Registros: 9,707
    eps_final = 0.006444
      Clusters válidos: 21

============================================================
[10/12] Procesando: ROBBERY
============================================================
  Registros: 4,423
    eps_final = 0.009582
      Clusters válidos: 5

============================================================
[11/12] Procesando: CRIMINAL TRESPASS
============================================================
  Registros: 3,757
    eps_final = 0.009544
      Clusters válidos: 8

============================================================
[12/12] Procesando: WEAPONS VIOLATION
============================================================
  Registros: 4,257
    eps_final = 0.008646
      Clusters válidos: 4

clusters_df creado: 163,771 registros
Año terminado: 2025
Video exportado a clusters_video.mp4
```

3.5 Pregunta 4 (RELLENAR CON UNA NUEVA)

3.6 Nuevas preguntas surgidas

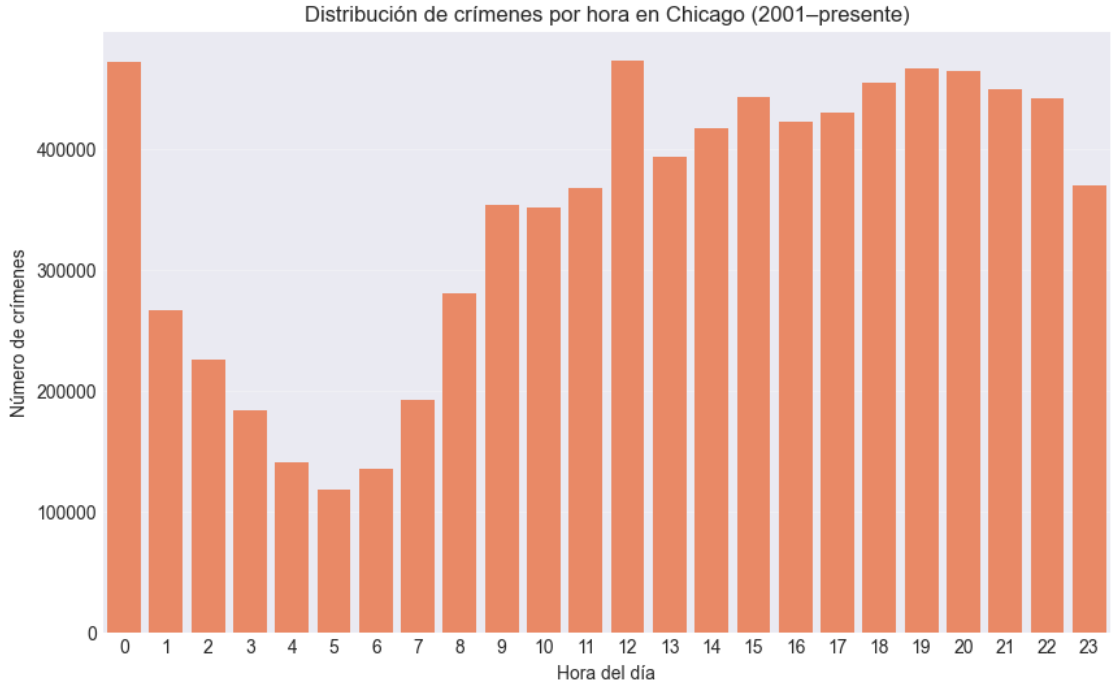
3.6.1 ¿Qué mes es el más seguro y cuál el más criminalístico en la ciudad de Chicago?

[117]:
```
# Supongamos df ya tiene columna 'Date' en datetime
df['Month'] = df['Date'].dt.month
crimes_by_month = df.groupby('Month').size()

# Si quieres promedio por mes usando todos los años
years_count = df['Date'].dt.year.nunique()
avg_crimes_by_month = crimes_by_month / years_count

plt.figure(figsize=(10,6))
plt.bar(range(1,13), avg_crimes_by_month, color='skyblue')
plt.xticks(range(1,13))
plt.xlabel('Mes')
plt.ylabel('Crímenes promedio por mes')
plt.title('Promedio de crímenes por mes en Chicago (2001-presente)')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



3.6.2 ¿En qué horas del día se concentran más crímenes en la ciudad de Chicago?

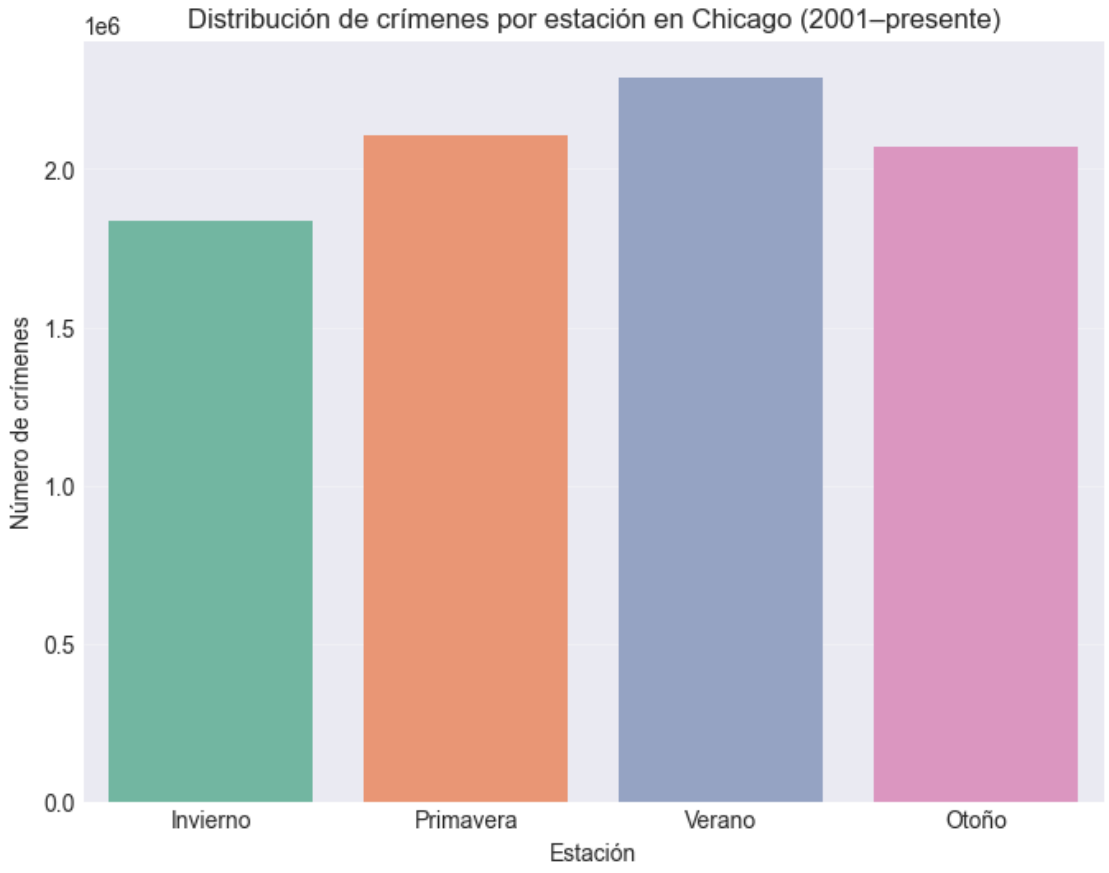
[118]:
```
# Extraer hora
df['Hour'] = df['Date'].dt.hour

# Agrupar por hora
crimes_by_hour = df.groupby('Hour').size()

plt.figure(figsize=(10,6))
sns.barplot(x=crimes_by_hour.index, y=crimes_by_hour.values, color='coral')
plt.xlabel('Hora del día')
plt.ylabel('Número de crímenes')
plt.title('Distribución de crímenes por hora en Chicago (2001-presente)')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



3.6.3 ¿Cuál es la estación del año más segura y cuál registra más delitos en Chicago?

[119]:
```
def get_season(month):
    if month in [12, 1, 2]:
        return 'Invierno'
    elif month in [3, 4, 5]:
        return 'Primavera'
    elif month in [6, 7, 8]:
        return 'Verano'
```

```
    else:
        return 'Otoño'

df['Season'] = df['Date'].dt.month.apply(get_season)

# Agrupar por estación
crimes_by_season = df.groupby('Season').size().
 ↪reindex(['Invierno','Primavera','Verano','Otoño'])

plt.figure(figsize=(8,6))
sns.barplot(x=crimes_by_season.index, y=crimes_by_season.values, palette="Set2")
plt.xlabel('Estación')
plt.ylabel('Número de crímenes')
plt.title('Distribución de crímenes por estación en Chicago (2001-presente)')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



4 Conclusiones

Hallazgos y limitaciones