NUMPY -MATPLOTLIB

El código combina de manera efectiva las capacidades de NumPy y Matplotlib para realizar un análisis completo de los datos de actuaciones de bomberos. Desde la carga de los datos, pasando por la normalización y el análisis estadístico, hasta llegar a la representación gráfica.

Carga de datos y preparación inicial

En esta primera sección se emplea la biblioteca NumPy para cargar el archivo CSV que contiene los datos de las actuaciones de bomberos del año 2025. Se utiliza la función np.loadtxt() con el delimitador de punto y coma (;), ya que este es el formato del archivo. La instrucción skiprows=1 permite omitir la fila de encabezados. Posteriormente, se seleccionan las columnas numéricas y se reemplazan los valores vacíos por ceros para evitar errores de conversión al tipo float.

```
import numpy as np
import matplotlib.pyplot as plt
# Cargar el archivo CSV con el delimitador correcto (punto y coma)
bombero = np.loadtxt( fname: './ActuacionesBomberos_2025.csv', skiprows=1, delimiter=';', dtype=str, encoding='utf-16')
columnas_numericas = bombero[:, 4:]
columnas_numericas = np.where(columnas_numericas == '', '0', columnas_numericas).astype(float)
```

Normalización de datos

Se define la función normalizar_min_max() para escalar los datos al rango [0, 1]. Este método permite comparar variables con diferentes unidades o escalas. Se calcula el mínimo y máximo de cada columna y se aplica la fórmula (x - min) / (max - min). Además, se controla el caso de rango nulo para evitar divisiones por cero. Luego se aplica la normalización a las columnas numéricas.

```
#funcion de normalizacion
def normalizar_min_max(datos):  5 usages
    minimos = np.min(datos, axis=0)
    maximos = np.max(datos, axis=0)
    rango = maximos - minimos
    rango = np.where(rango == 0, 1, rango)  # Evitar división por cero
    return (datos - minimos) / rango

datos_normalizados = normalizar_min_max(columnas_numericas)
```

Cálculo de estadísticas básicas

El código incluye varios cálculos estadísticos relevantes que permiten comprender la magnitud de las actuaciones y su distribución entre distritos: Se calcula la **media de actuaciones en el mes de octubre** para obtener una medida representativa del periodo. Se determina el **número total de actuaciones durante el año 2025.** Se calcula el **total de servicios varios registrados en el distrito de Puente de Vallecas**. Finalmente, se obtiene la **menor cantidad de actuaciones por fuego en el distrito de Usera.** Estos valores se obtienen utilizando funciones de NumPy como np.sum() y np.min(), que son altamente optimizadas para el manejo de matrices numéricas.

```
medoct = bombero[0:22, 4:]  # Filas de octubre, columna a partir de la columna numero de distrito

# Convertir a números para calcular la media
medoctpamedia = medoct.astype(float)
mediaactuacionesoctubre = np.mean(medoctpamedia)
print(f"\nLa media de actuaciones de bomeros en el mes de Octubre: {mediaactuacionesoctubre:.0f}")
print(f"\n Rangos normalizados: {normalizar_min_max(medoctpamedia)}")

# Actuaciones totales en lo que llevamos de año
sumatotaldeactuaciones2025 = bombero[:, 4:]
# Reemplazar vacios con '0' antes de convertir
sumatotaldeactuaciones2025 = np.where(sumatotaldeactuaciones2025 == '', '0', sumatotaldeactuaciones2025).astype(float)
total2025 = np.sum(sumatotaldeactuaciones2025)

print(f"\nEl numero total de actuaciones en 2025 es de: {total2025:.0f}")
print(f"\n Rangos normalizados: {normalizar_min_max(sumatotaldeactuaciones2025)}")

filaspuentevallecas = bombero[bombero[:, 2] == 'PUENTE VALLECAS']
# Extraer la columna SERVICIOS VARIOS (columna 10)
serviciosvariospv = filaspuentevallecas[:, 10]
# Convertir en float
serviciosvariospv = np.where(serviciosvariospv == '', '0', serviciosvariospv).astype(float)
# Sumar todos los servicios varios de Puente de Vallecas
totalserviciosvarios = np.sum(serviciosvariospv)

print(f"\nActuaciones con el pretexto de servicios varios en el distrito de Puente de Vallecas: {totalserviciosvarios:.0f}")
print(f"\n Rangos normalizados: {normalizar_min_max(sumatotaldeactuaciones2025)}")

# menor cantidad de actuaciones por fuego, utilizando el numero de distrito en vez del nombre, en usera
filasusera = bombero[bombero[:, 3] == '12']
intervencionfuegos = filasusera[:, 5]
intervencionfuegos = np.where(intervencionfuegos == '', '0', intervencionfuegos).astype(float)
menosfuegosusera= np.min(intervencionfuegos)
print(f"\nLa menor cantidad de actuaciones por fuego en Usera fue: {menosfuegosusera:.0f}")
print(f"\n Rangos normalizados: {normalizar_min_max(sumatotaldeactuaciones2025)}")
```

EJECUCIÓN ESTADÍSTICAS BÁSICAS CON NUMPY

```
La media de actuaciones de bomeros en el mes de Octubre: 159

 Rangos normalizados: [[0.68992248 1.         1.         1.         0.90444444 1.
  1.        ]
 [0.36175711 0.30314961 0.5511811  0.41025641 0.61777778 0.38888889
  0.34210526]
 [0.12661499 0.27165354 0.36220472 0.28937729 0.31555556 0.33333333
  0.15789474]
 [0.24031008 0.48818898 0.69094488 0.63003663 0.56222222 0.51111111
  0.35964912]
 [0.22997416 0.22440945 0.54527559 0.47252747 0.56888889 0.48888889
  0.29824561]
 [0.40310078 0.44488189 0.65354331 0.71428571 0.47111111 0.46666667
  0.28070175]
 [0.3255814  0.53937008 0.69291339 0.47985348 0.48666667 0.43333333
  0.25438596]
 [0.80361757 0.32283465 0.80905512 0.53479853 1.         0.7
  0.54385965]
 [0.28165375 0.21259843 0.54133858 0.42857143 0.78222222 0.48888889
  0.51754386]
 [0.57622739 0.32283465 0.87598425 0.6007326  0.80222222 0.54444444
  0.43859649]
 [0.83204134 0.40944882 0.88188976 0.52380952 0.77111111 0.54444444
  0.22807018]
 [0.55813953 0.18503937 0.48031496 0.32967033 0.41777778 0.37777778
  0.45614035]
 [0.79586563 0.44488189 0.89173228 0.52747253 0.76444444 0.68888889
  0.42105263]
 [0.12403101 0.08661417 0.32283465 0.22710623 0.28888889 0.31111111
  0.        ]
 [0.41860465 0.32283465 0.72834646 0.43956044 0.79777778 0.57777778
  0.89473684]
 [0.36950904 0.18503937 0.46259843 0.42490842 0.60666667 0.48888889
  0.07894737]
 [0.80878553 0.16929134 0.53543307 0.42124542 0.52       0.28888889
  0.35964912]
 [1.         0.07086614 0.49606299 0.24542125 0.38888889 0.56666667
```

```
El numero total de actuaciones en 2025 es de: 45956

 Rangos normalizados: [[0.71631206 1.         1.         ... 0.91451292 1.         1.        ]
 [0.41607565 0.30859375 0.57383178 ... 0.65805169 0.43877551 0.38016529]
 [0.20094563 0.27734375 0.39439252 ... 0.38767396 0.3877551  0.20661157]
 ...
 [0.03782506 0.01171875 0.04299065 ... 0.03777336 0.04081633 0.03305785]
 [0.0070922  0.         0.00373832 ... 0.027833   0.04081633 0.01652893]
 [0.00472813 0.         0.00186916 ... 0.02584493 0.01020408 0.07438017]]
```

```
Actuaciones con el pretexto de servicios varios en el distrito de Puente de Vallecas: 95

 Rangos normalizados: [[0.71631206 1.         1.         ... 0.91451292 1.         1.        ]
 [0.41607565 0.30859375 0.57383178 ... 0.65805169 0.43877551 0.38016529]
 [0.20094563 0.27734375 0.39439252 ... 0.38767396 0.3877551  0.20661157]
 ...
 [0.03782506 0.01171875 0.04299065 ... 0.03777336 0.04081633 0.03305785]
 [0.0070922  0.         0.00373832 ... 0.027833   0.04081633 0.01652893]
 [0.00472813 0.         0.00186916 ... 0.02584493 0.01020408 0.07438017]]
```

```
La menor cantidad de actuaciones por fuego en Usera fue: 2

 Rangos normalizados: [[0.71631206 1.         1.         ... 0.91451292 1.         1.        ]
 [0.41607565 0.30859375 0.57383178 ... 0.65805169 0.43877551 0.38016529]
 [0.20094563 0.27734375 0.39439252 ... 0.38767396 0.3877551  0.20661157]
 ...
 [0.03782506 0.01171875 0.04299065 ... 0.03777336 0.04081633 0.03305785]
 [0.0070922  0.         0.00373832 ... 0.027833   0.04081633 0.01652893]
 [0.00472813 0.         0.00186916 ... 0.02584493 0.01020408 0.07438017]]
```

Análisis por distritos

En esta parte se realiza una extracción selectiva de columnas del dataset, donde cada una representa un tipo de actuación (fuegos, daños en construcción, salvamentos e incidentes). Para evitar sobrecargar las gráficas, se limita el análisis a los primeros diez distritos. Esto facilita una visualización más clara y ordenada.

```python
# Extraer columnas necesarias
distritos = bombero[:, 2]
fuegos = bombero[:, 4].astype(float)
danos_construccion = bombero[:, 5].astype(float)
salvamentos = bombero[:, 6].astype(float)
incidentes = bombero[:, 8].astype(float)



# Tomamos los primeros 10 distritos para no saturar las gráficas
distritos = distritos[:10]
fuegos = fuegos[:10]
danos_construccion = danos_construccion[:10]
salvamentos = salvamentos[:10]
incidentes = incidentes[:10]
```

Visualización de datos con Matplotlib

El bloque de visualización utiliza la biblioteca Matplotlib para representar gráficamente la información. Se utilizan distintos tipos de gráficos para distintos propósitos:

Gráficos de líneas: muestran la evolución comparativa entre el número de fuegos y los salvamentos por distrito. Los marcadores y la cuadrícula permiten una mejor lectura de los valores.

```
# Gráfico Dos líneas
plt.plot( *args: distritos, fuegos, label="Fuegos", marker="o")
plt.plot( *args: distritos, salvamentos, label="Salvamentos y Rescates", marker="s")
plt.xlabel("Distrito")
plt.ylabel("Número de actuaciones")
plt.title("Comparación de Fuegos y Salvamentos por Distrito")
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
plt.close()
```

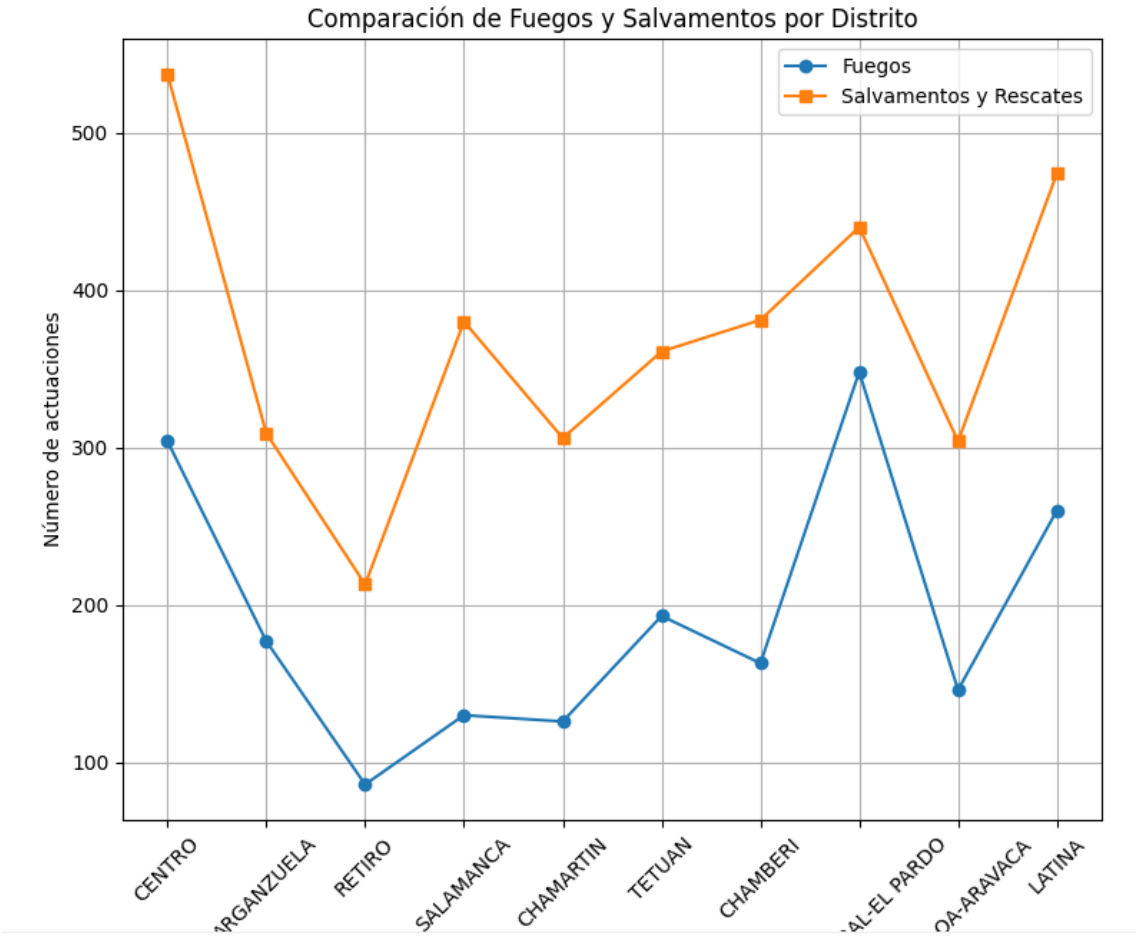


Gráfico de puntos : representa visualmente la relación entre el número de fuegos de cada distrito, permitiendo observar las relaciones entre ambos tipos de actuaciones y rotación de los nombres de los distritos para mejorar la presentación.

```
# Gráfico Puntos individuales
plt.scatter(distritos, fuegos)
plt.xlabel("Distrito")
plt.ylabel("Número de fuegos")
plt.title("Fuegos por Distrito (Gráfico de puntos)")
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```

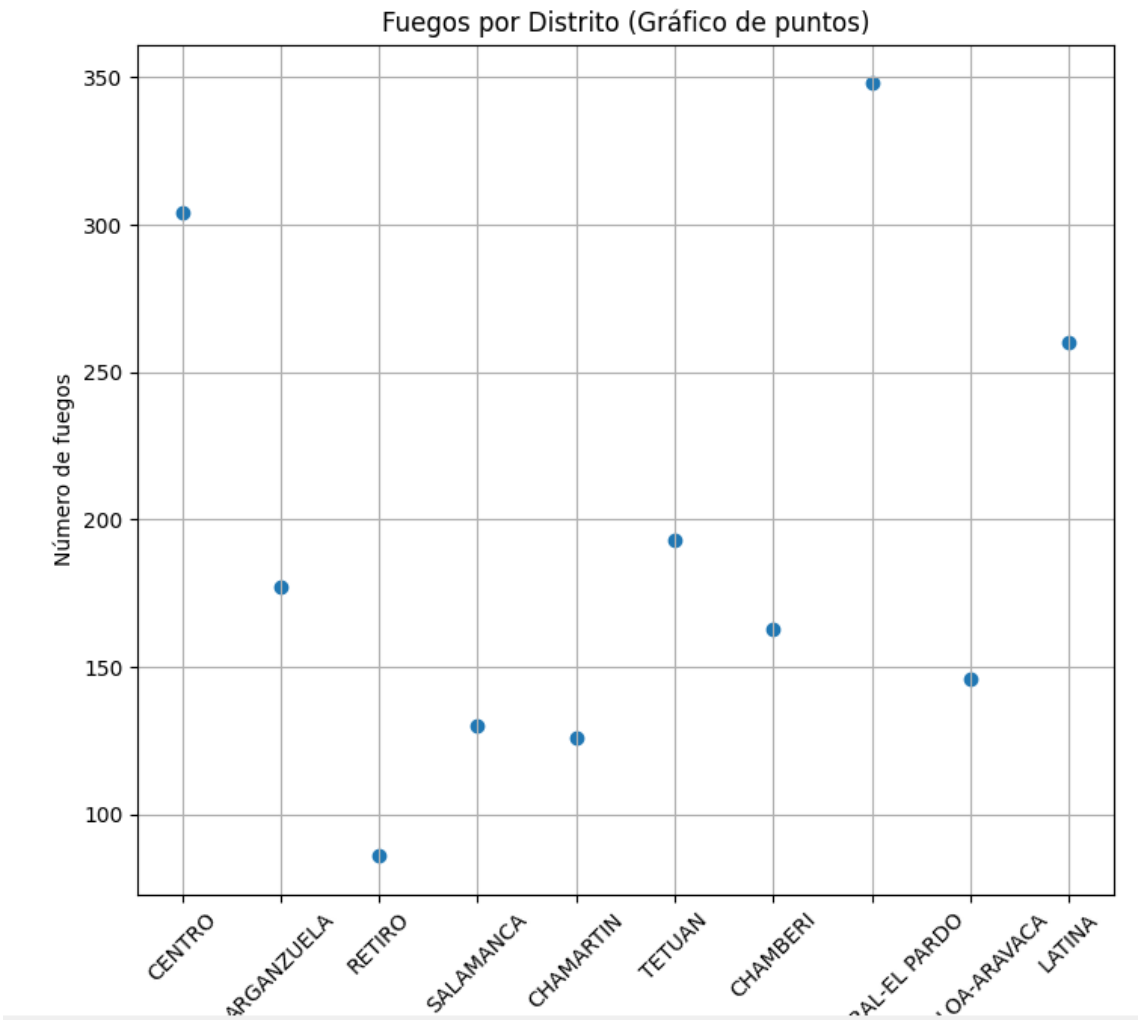


Gráfico de barras: se emplea para comparar el número total de fuegos en los diferentes distritos. Este tipo de gráfico es muy útil para destacar qué zonas presentan mayor incidencia de incendios.

```
#Gráfico de barras (bar)
plt.bar(distritos, fuegos)
plt.xlabel("Distrito")
plt.ylabel("Número de fuegos")
plt.title("Fuegos por distrito")
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

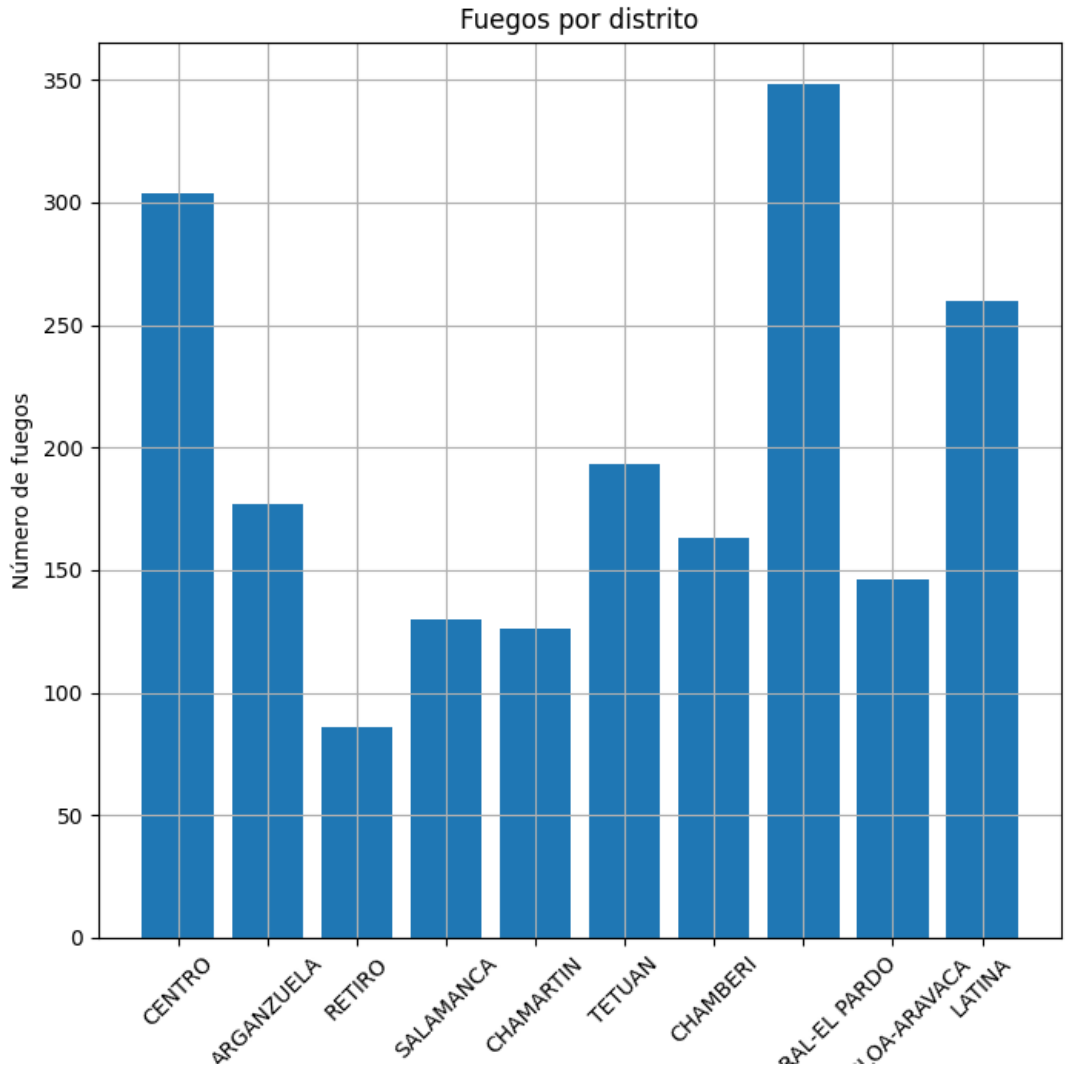


Gráfico combinado de tres elementos: integra tres variables (fuegos, salvamentos e incidentes) en un mismo eje, empleando diferentes colores y formas de marcadores. Esto ofrece una visión global de las principales actuaciones por distrito.

```
# Gráfico Tres Elementos
plt.plot( *args: distritos, fuegos, 'r--',              # Rojo discontinua → fuegos
         distritos, salvamentos, 'bs',          # Azul con marcadores cuadrados → salvamentos
         distritos, incidentes, 'g^')           # Verde con triángulos → daños por agua
plt.xlabel("Distrito")
plt.ylabel("Número de actuaciones")
plt.title("Actuaciones de Bomberos por Distrito")
plt.legend(["Fuegos", "Salvamentos", "Incidentes"])
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
plt.close()
```

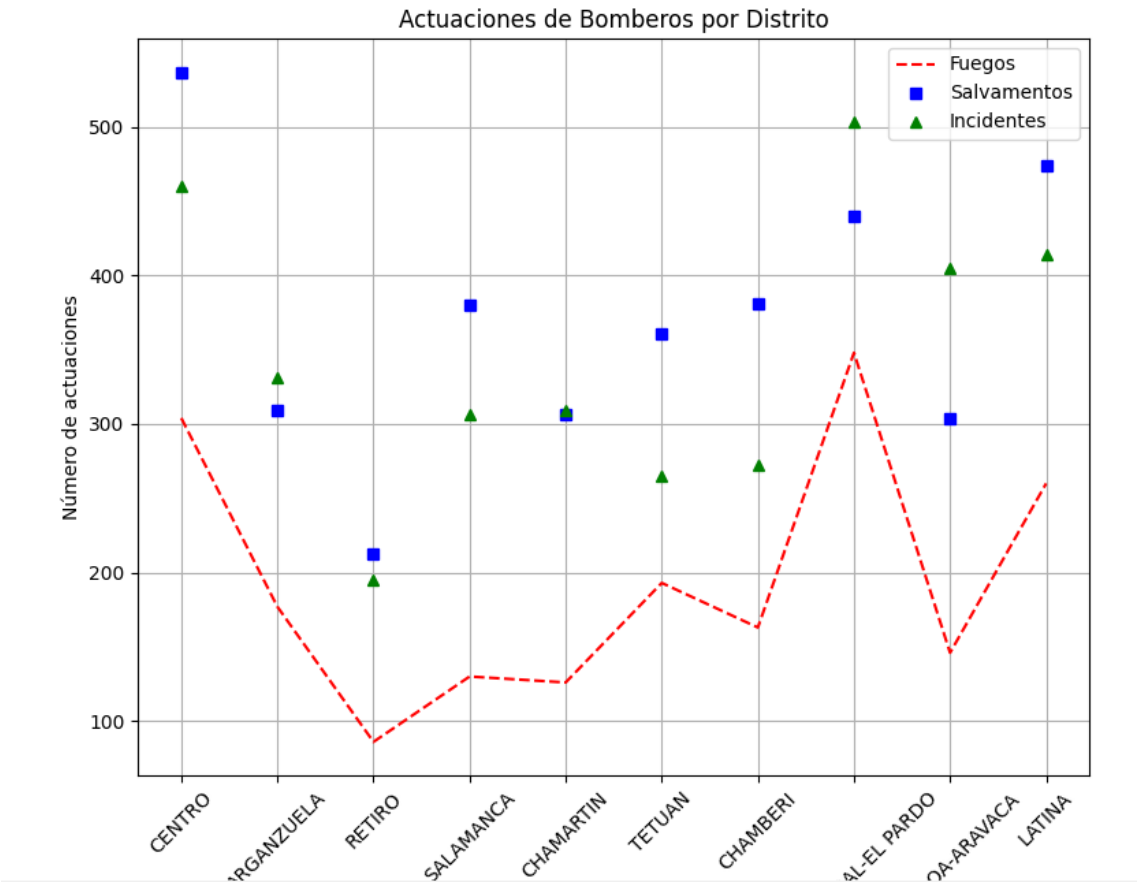
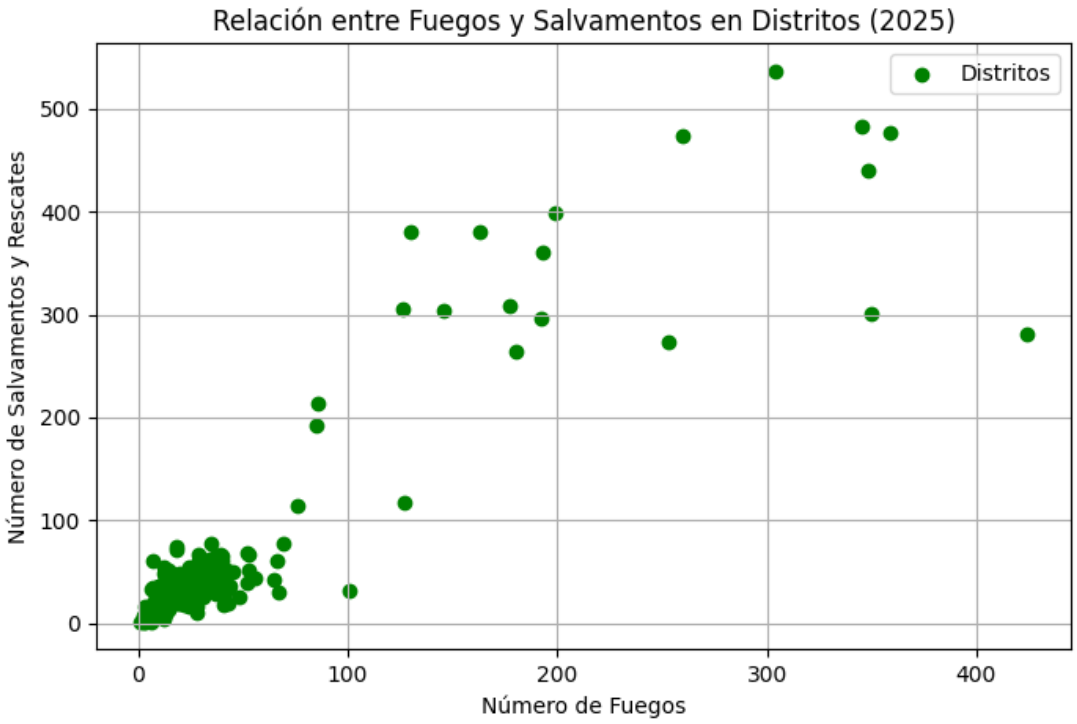


Gráfico Scatter: representa visualmente la relación entre el número de fuegos y salvamentos en cada distrito, permitiendo observar posibles correlaciones entre ambos tipos de actuaciones.

```
# Extraer las columnas que necesitamos
fuegos_total = bombero[:, 4].astype(float)
salvamentos_total = bombero[:, 6].astype(float)



# Gráfico scatter
plt.figure(figsize=(8, 5))
plt.scatter(fuegos_total, salvamentos_total, c='green', label="Distritos")
plt.xlabel("Número de Fuegos")
plt.ylabel("Número de Salvamentos y Rescates")
plt.title("Relación entre Fuegos y Salvamentos en Distritos (2025)")
plt.legend()
plt.grid(True)
plt.show()
```



Histograma: representa la distribución de los daños en construcción, permitiendo analizar la frecuencia de aparición de distintos niveles de daño. Esta visualización es útil para detectar concentraciones.

```
# Gráfica Histograma
plt.hist(danos_construccion, color='skyblue', edgecolor='black')
plt.xlabel("Número de daños en construcción")
plt.ylabel("Frecuencia")
plt.title("Distribución de daños en construcción (Histograma)")
plt.grid(True)
plt.show()
```