# Universidad Autónoma de Baja California
# Facultad de Ciencias químicas e Ingeniería
# Ingeniero en computación


## Proyecto Final
### Lenguaje de programación python

**INTEGRANTES:**

Mariano Perez Piña - 01271949
Tamara Rico Lopez - 001270673
Jorge Omar Torres Sosa - 00127090

**PROFESOR:** Karina Raya Diaz

Para comenzar python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

En python se pueden hacer una increíble cantidad de cosas si se saben utilizar las diversas herramientas que se nos son otorgadas, entre ellas la manipulación de información y variables. Dado esto podemos hacer uso de bases de datos para que sirvan como fuente de la cual manipularemos la información, teniendo la opción de organizarla en una diversa cantidad de maneras. Una base de datos es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora.

## Objetivo:

Este proyecto tiene como fin el visualizar e interpretar la información a partir de una base de datos mediante el uso de herramientas en python.

En cuanto a la base de datos que utilizaremos, nuestro fin es poder visualizar el número de incidentes que ocurren en Estados Unidos.

Para este caso la base de datos que utilizaremos será la de “Emergency - 911 calls” la cual contiene información acerca de las llamadas efectuadas hacia la línea de emergencia del 911 en Estados Unidos.

## Primer Avance:

## Histograma:

A partir de python, se pudo acceder a la base de datos acerca de las llamadas al 911 en Estados Unidos y se diseñó una gráfica de barras, es decir, un histograma que presenta el número de accidentes registrados según su categoría.

En este caso, las categorías son: incendios, accidentes de tráfico y emergencias médicas. Se calculó el número de ocurrencias a partir del código y es posible observar que la emergencia que registró más llamas fueron las emergencias médicas, seguidas de los accidentes de tráfico y de los incendios.

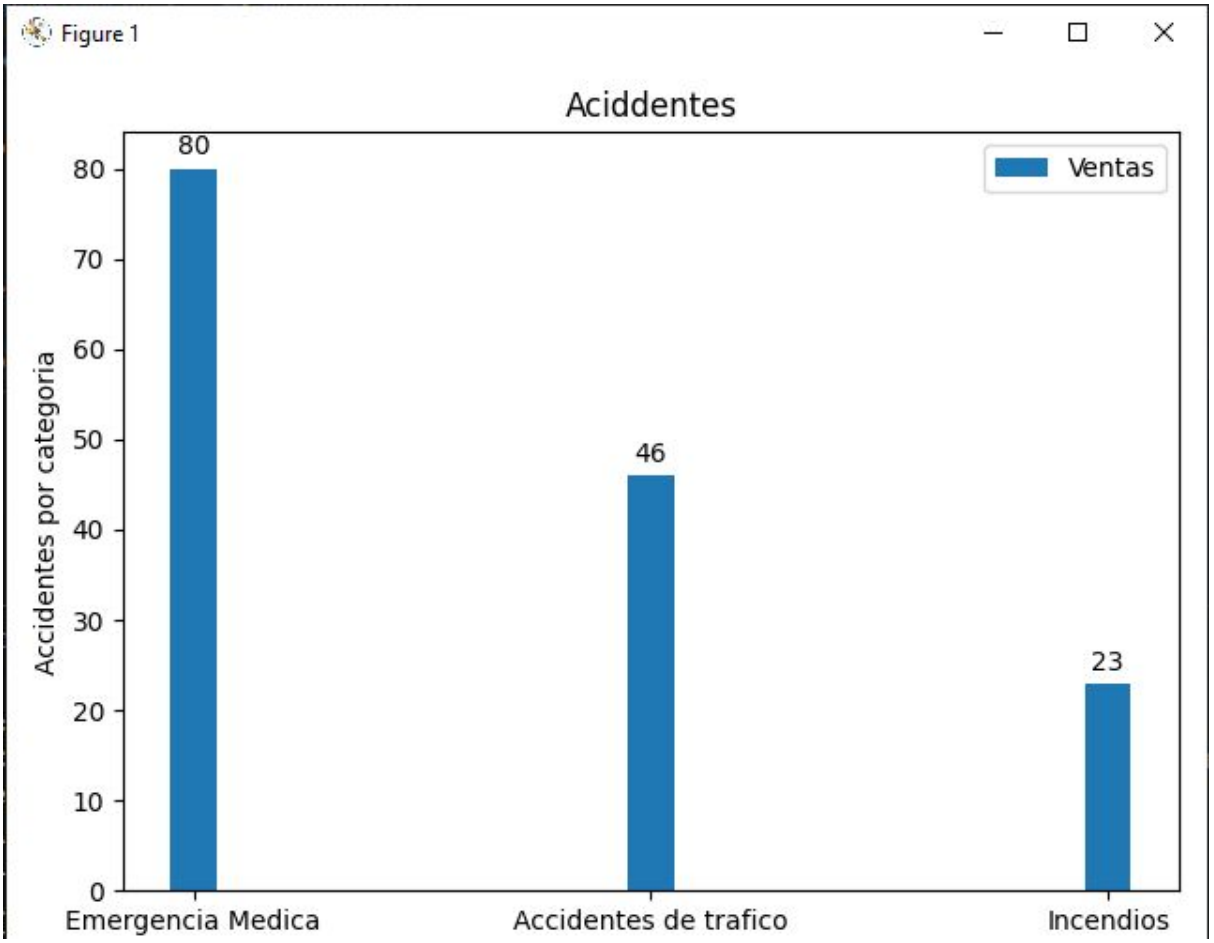


**Figura 1**. Histograma de los Accidentes reportados al 911 en Pennsylvania

**Código:**

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

cantProductos = {'ems': 0, 'trafico': 0, 'fire': 0}

filtro = pd.read_excel('911.xls')
i = 0
while (i < (filtro.shape[0])):
  tipo = filtro['title'][i]
  if tipo[:4] == 'EMS:':
    cantProductos['ems'] += 1
  elif tipo[:5] == 'Fire:':
    cantProductos['fire'] += 1
  elif tipo[:8] == 'Traffic:':
    cantProductos['trafico'] += 1
```

```
  i += 1

productos = ['Emergencia Medica', 'Accidentes de trafico', 'Incendios']

#Obtenemos la posición de cada etiqueta en el eje de X
x = np.arange(len(productos))
#tamaño de cada barra
width = 0.1

fig, ax = plt.subplots()

#Generamos las barras para el conjunto de hombres
barraproductos = ax.bar(x, cantProductos.values(), width, label='Ventas')
#Generamos las barras para el conjunto de mujeres

#Añadimos las etiquetas de identificación de valores en el gráfico
ax.set_ylabel('Accidentes por categoría')
ax.set_title('Aciddentes')
ax.set_xticks(x)
ax.set_xticklabels(productos)
#Añadimos un legen() esto permite mostrar con colores a que pertenece cada valor.
ax.legend()

def autolabel(rects):
    """Función para agregar una etiqueta con el valor en cada barra"""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

#Añadimos las etiquetas para cada barra
autolabel(barraproductos)

fig.tight_layout()
#Mostramos la gráfica con el método show()
plt.show()
```

## Red de nodos:

A partir de las ubicaciones de las ciudades, y de la cantidad de accidentes que provocaron las llamadas al 911, se representó dicha información mediante una red de nodos.

Dentro de la red, las conexiones representan similitudes en la cantidad de accidentes entre las ciudades de Pennsylvania, el tamaño representa a las ciudades con más accidentes y su ubicación es de acuerdo a la realidad; ya que se utilizaron las coordenadas de dichas ciudades.

Por lo tanto, este diagrama sirve para visualizar el número de emergencias por ciudades, y no sólo la clase de emergencia más presentada, con el objetivo de tener el panorama completo de la situación.

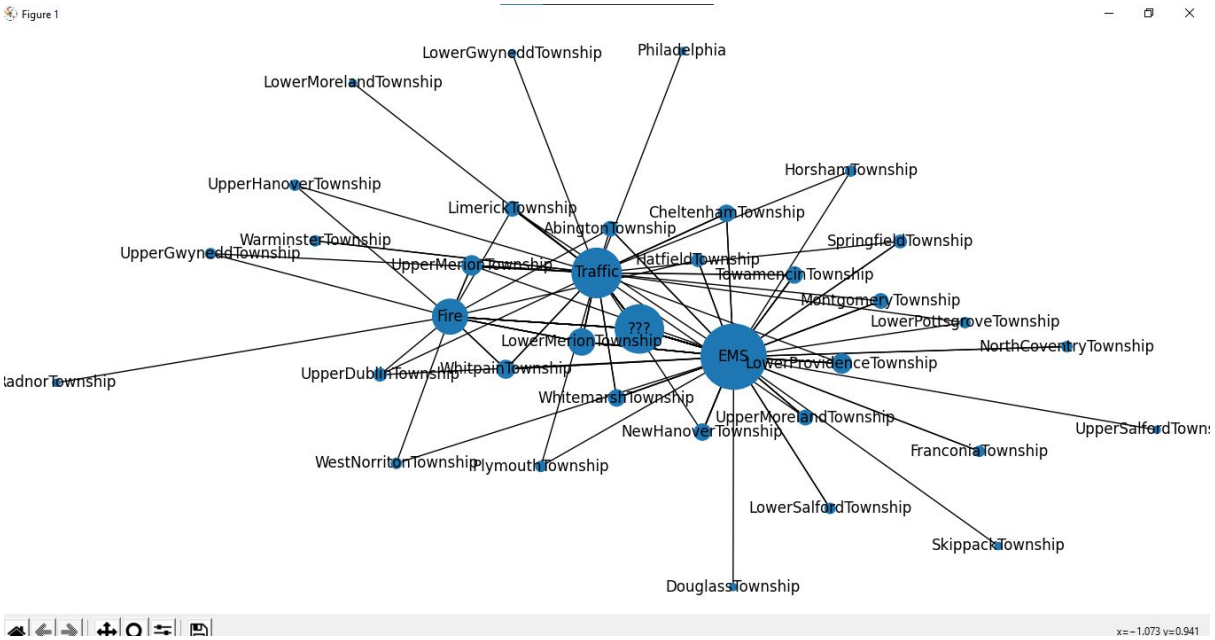


**Figura 2.** Red de nodos de emergencias entre las ciudades de Pennsylvania.

## Código:

```
from geopy.geocoders import Nominatim
import pandas as pd
import pylab
import networkx as nx
import pickle

geolocator = Nominatim(user_agent="Nominatim")

cantProductos = {'ems': 0, 'tráfico': 0, 'fire': 0}
```

```
filtro = pd.read_excel('911.xls')
i = 0
direcciones = []
incidentestipo = []
while (i < filtro.shape[0]):
  tipo = filtro['title'][i]
  raw = (geolocator.reverse(str(filtro['lat'][i])+','+str(filtro['lng'][i]))).raw
  try:
    direcciones.append(raw['address']['city'])
  except:
   direcciones.append( "???" )

  if tipo[:4] == 'EMS':
    cantProductos['ems'] += 1
    incidentestipo.append('EMS')
  elif tipo[:5] == 'Fire':
    incidentestipo.append('Fire')
    cantProductos['fire'] += 1
  elif tipo[:8] == 'Traffic':
    incidentestipo.append('Traffic')
    cantProductos['trafico'] += 1
  i += 1
  print(i)

for i in range(len(direcciones)):
  direcciones[i] = direcciones[i].replace(" ","")

arch = open("soda.txt", "w")
for i in range(len(direcciones)):
  arch.write(incidentestipo[i] + " " + direcciones[i])
  arch.write("\n")
arch.close()

G=nx.read_weighted_edgelist('soda.txt', create_using=nx.MultiGraph())

custom_node_color={}
custom_node_color['0'] = 'black'

nx.draw(G, with_labels = True, node_size = [G.degree(i) * 35 for i in G.nodes()])

#Archivo con el dato del promedio del grado de clusterización por nodo
avg_node_degree = nx.average_neighbor_degree(G)
vString = str(avg_node_degree)
pylab.show()
```

**Segundo Avance:**
**Dendograma:**

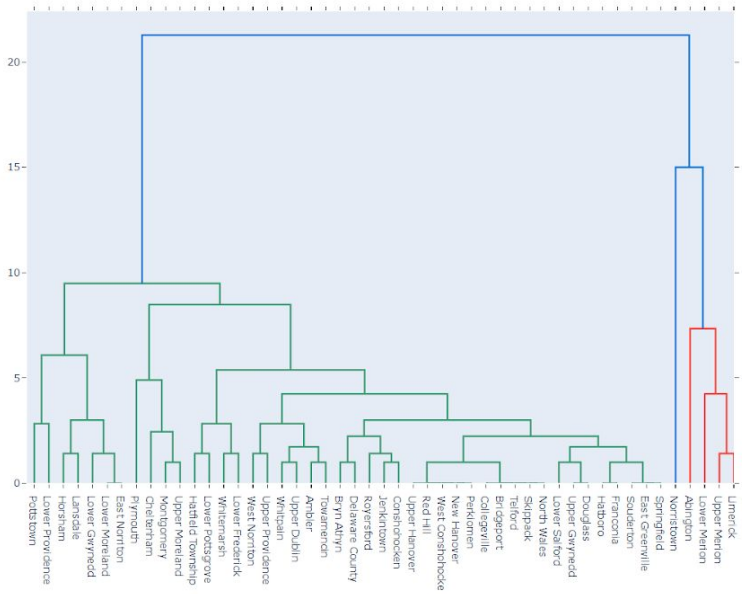


**Figura 3.** Dendograma de los incidentes basados en las ciudades (sin editar)

A partir de la base de datos principal, se diseñó una base de datos especialmente para el dendograma. En dicho documento se ordenaron los datos de acuerdo a las ciudades y la cantidad de accidentes de cada tipos que habían sucedido en la misma ciudad.

El resultado es el siguiente dendograma en el que se agrupan todas las ciudades según los incidentes, donde el color rojo representa los incendios, el color azul los accidentes de tráfico y, el color verde las emergencias médicas. Por otro lado, las agrupaciones se hacen de acuerdo a las similitudes en número de incidentes reportados en cada ciudad.

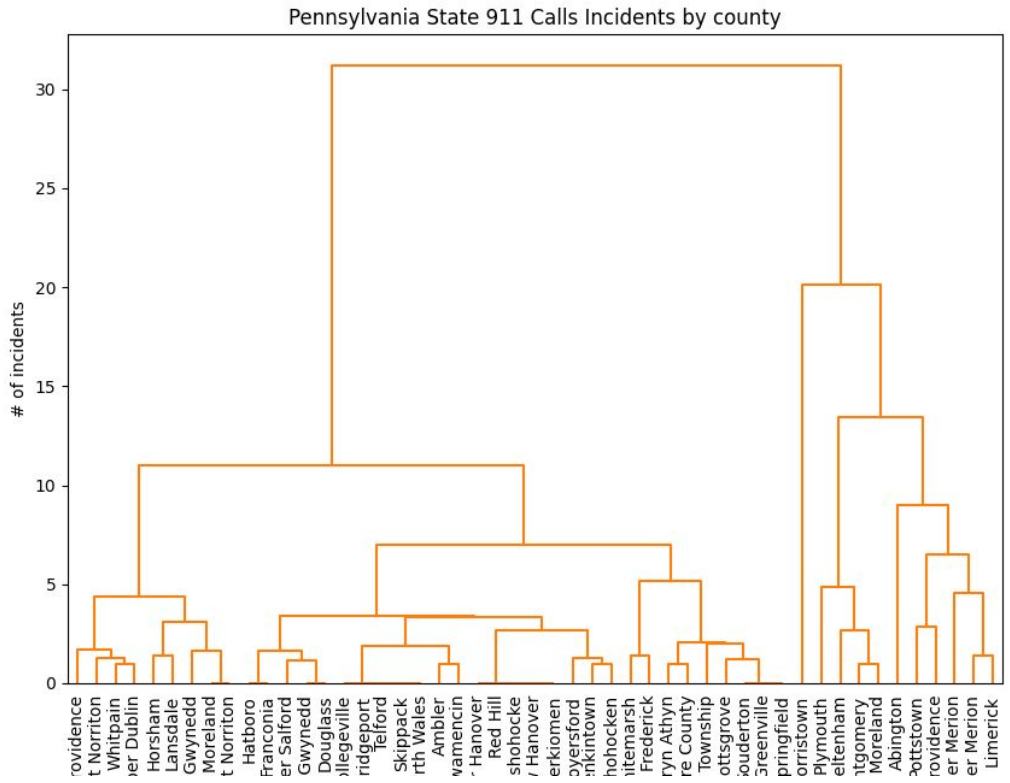


**Figura 4.** Dendograma de los incidentes por ciudad en Pennsylvania.

**Código:**

```
#Librerías necesarias para el dendograma
import plotly.figure_factory as ff
import pandas as pd
import numpy as np
import os

#Acceso al archivo con la base de datos
filepath = str(os.path.dirname(os.path.realpath(__file__))) + "\\Datos911.csv"

#Dendograma para la distribución de los tipos de emergencias en cada ciudad
X = pd.read_csv(filepath)
X.rename(columns={X.columns[0]:"# of incidents"}, inplace= True)
X.set_index("# of incidents")
X.replace(np.nan, 0, inplace= True)
incidentes = list(X["# of incidents"])
del X["# of incidents"]
#del X["Origin"]

#CREACIÓN DEL DENDOGRAMA
fig = ff.create_dendrogram(X, labels=incidentes)
fig.update_layout(autosize=False, width=1000, height=800)
fig.show()
```

**Tercer Avance:**
**Geolocalización:**

Con el fin de lograr una mejor visualización de los datos, se desarrolló una red de puntos mediante la localización de las ciudades de Pennsylvania en el espacio. El tamaño de cada punto de cada ciudad representa la cantidad de accidentes que ha presentado; de acuerdo a la información disponible en la base de datos.

En la imagen se muestran las ciudades con su respectivo nombre; sin embargo, algunos problemas con el código provocaron que no se pudieran observar los colores.

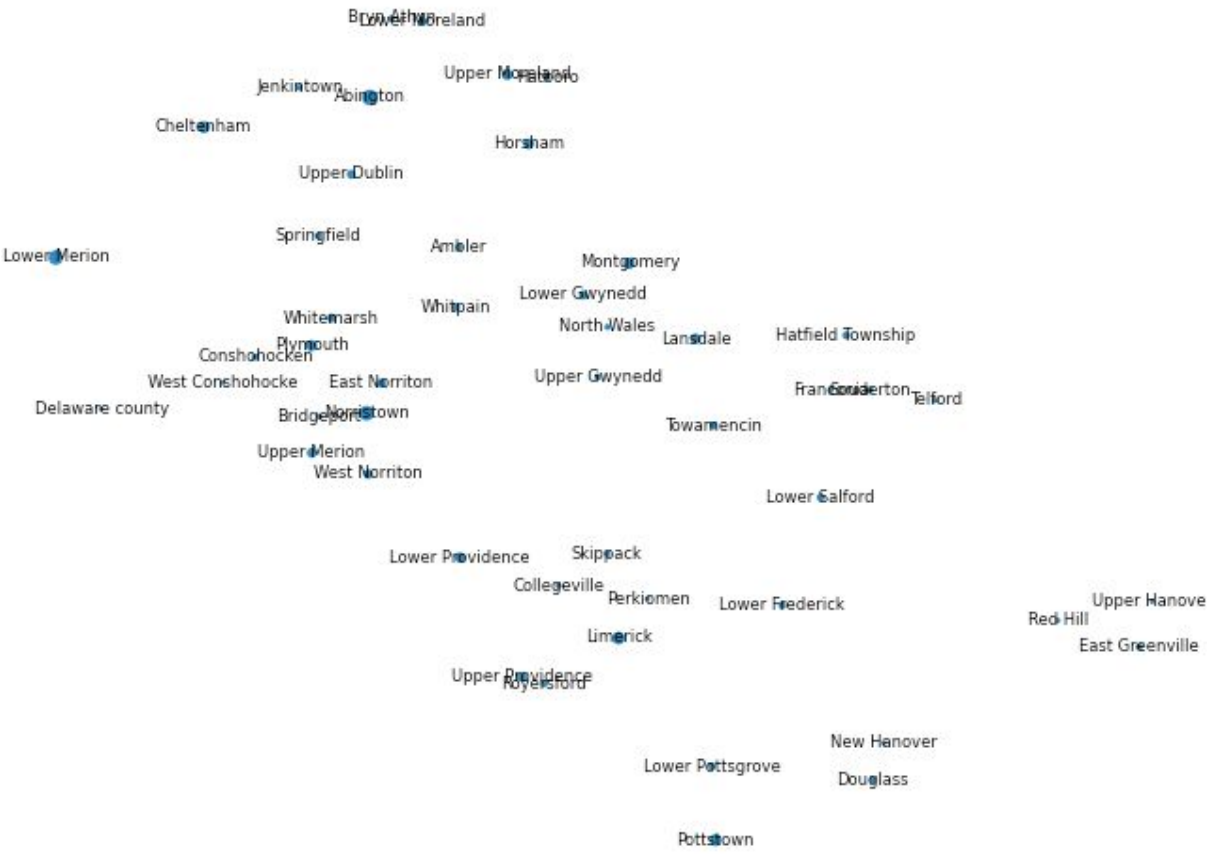


**Figura 5.** Geolocalización de las ciudades

**Código:**

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_node('Lower Merion', pos=(40.0086941, -75.2287561), score=21, ems=10)
G.add_node('Montgomery', pos=(40.2188701, -75.2328183), score=11, ems=11)
```

```
G.add_node('Norristown', pos=( 40.1226,-75.3396), score=22, ems=21)
G.add_node('Horsham', pos=(40.1821, -75.1479), score=9, ems=5)
G.add_node('East Greenville', pos=(40.4058, -75.5061), score=3, ems=1)
G.add_node('Abington', pos=(40.1238, -75.1148), score=25, ems=14)
G.add_node('Pottstown', pos=(40.2506,-75.6435 ), score=13, ems=9)
G.add_node('Ambler', pos=( 40.1564, -75.2214), score=3, ems=3)
G.add_node('Jenkintown', pos=(40.098, -75.1078), score=2, ems=1)
G.add_node('Skippack', pos=(40.210846, -75.4397957), score=2, ems=2)
G.add_node('Upper Merion', pos=(40.1025527, -75.3678382), score=8, ems=15)
G.add_node('Whitpain', pos=(40.1552833, -75.2642296), score=5, ems=3)
G.add_node('Lower Salford', pos=(40.2890267, -75.3995896), score=4, ems=2)
G.add_node('Plymouth', pos=(40.1023985, -75.2914577), score=11, ems=1)
G.add_node('Upper Moreland', pos=(40.1741312, -75.0984907), score=10, ems=4)
G.add_node('Cheltenham', pos=(40.062974, -75.135914), score=13, ems=4)
G.add_node('Lansdale', pos=(40.2432578, -75.2865516), score=9, ems=6)
G.add_node('New Hanover', pos=(40.3121807, -75.5742598), score=1, ems=1)
G.add_node('West Norriton', pos=(40.123193, -75.3829629), score=7, ems=4)
G.add_node('Whitemarsh', pos=(40.1098315, -75.2719513), score=6, ems=1)
G.add_node('Limerick', pos=(40.2149774, -75.4997122), score=15, ems=7)
G.add_node('Lower Moreland', pos=(40.1430862, -75.0605932), score=6, ems=4)
G.add_node('Towamencin', pos=(40.2497608, -75.3484526), score=4, ems=3)
G.add_node('Upper Providence', pos=(40.1795657, -75.5273502), score=7, ems=3)
G.add_node('Springfield', pos=(40.1049255, -75.2132496), score=3, ems=1)
G.add_node('Hatfield Township', pos=(40.2984302, -75.2841346), score=4, ems=0)
G.add_node('Lower Pottsgrove', pos=(40.249191, -75.5909541), score=4, ems=1)
G.add_node('Upper Gwynedd', pos=(40.2074035, -75.3137919), score=3, ems=2)
G.add_node('Perkiomen', pos=(40.2255371, -75.4716175), score=1, ems=1)
G.add_node('Lower Providence', pos=(40.156807, -75.4423234), score=9, ems=7)
G.add_node('Conshohocken', pos=(40.0817953, -75.2996401), score=3, ems=2)
G.add_node('North Wales', pos=(40.2109404, -75.2782317), score=2, ems=2)
G.add_node('Douglass', pos=(40.3082416, -75.6006932), score=3, ems=2)
G.add_node('Souderton', pos=(40.3071113, -75.3230319), score=3, ems=1)
G.add_node('Lower Gwynedd', pos=(40.2020579, -75.2554801), score=6, ems=3)
G.add_node('Upper Dublin', pos=(40.116985, -75.1694856), score=6, ems=4)
G.add_node('Royersford', pos=(40.1876645, -75.532277), score=3, ems=1)
G.add_node('Telford', pos=(40.3309982, -75.3304253), score=2, ems=2)
G.add_node('West Conshohocken', pos=(40.0700952, -75.3178667), score=1, ems=1)
G.add_node('Hatboro', pos=(40.188666, -75.1007192), score=2, ems=1)
G.add_node('Red Hill', pos=(40.3760733, -75.4870743), score=1, ems=1)
G.add_node('Lower Frederick', pos=(40.2750408, -75.476455), score=4, ems=0)
G.add_node('Bridgeport', pos=(40.105308, -75.341987), score=2, ems=2)
G.add_node('East Norriton', pos=(40.1280087, -75.3181112), score=6, ems=4)
G.add_node('Bryn Athyn', pos=(40.1317717, -75.0582548), score=2, ems=0)
G.add_node('Collegeville', pos=(40.1930534, -75.4623315), score=2, ems=2)
```

```
G.add_node('Franconia', pos=(40.2927598, -75.3230399), score=2, ems=1)
G.add_node('Delaware county', pos=(40.0252992, -75.3365305), score=1, ems=0)
G.add_node('Upper Hanover', pos=(40.4106035, -75.4736714), score=1, ems=1)

score = nx.get_node_attributes(G, 'score')
node_size = [float(score[v]) for v in G]
cmap = plt.cm.rainbow
nx.draw(G, nx.get_node_attributes(G, 'pos'), font_size=6, with_labels=True,
node_size=node_size, edge_cmap=cmap)

plt.show()
```

### Geolocalización en un mapa:

Para finalizar con el proyecto, se ubicaron los puntos, que representan a las ciudades de Pennsylvania, en un mapa del mismo estado. Las ciudades que se muestran son aquellas que han presentado más incidentes y; por lo tanto, llamadas al 911.

A partir del resultado, se puede llegar a la conclusión de que en el centro del estado ocurren más accidentes que en el resto del estado; así como que en las afueras, los resultados son más dispersos.

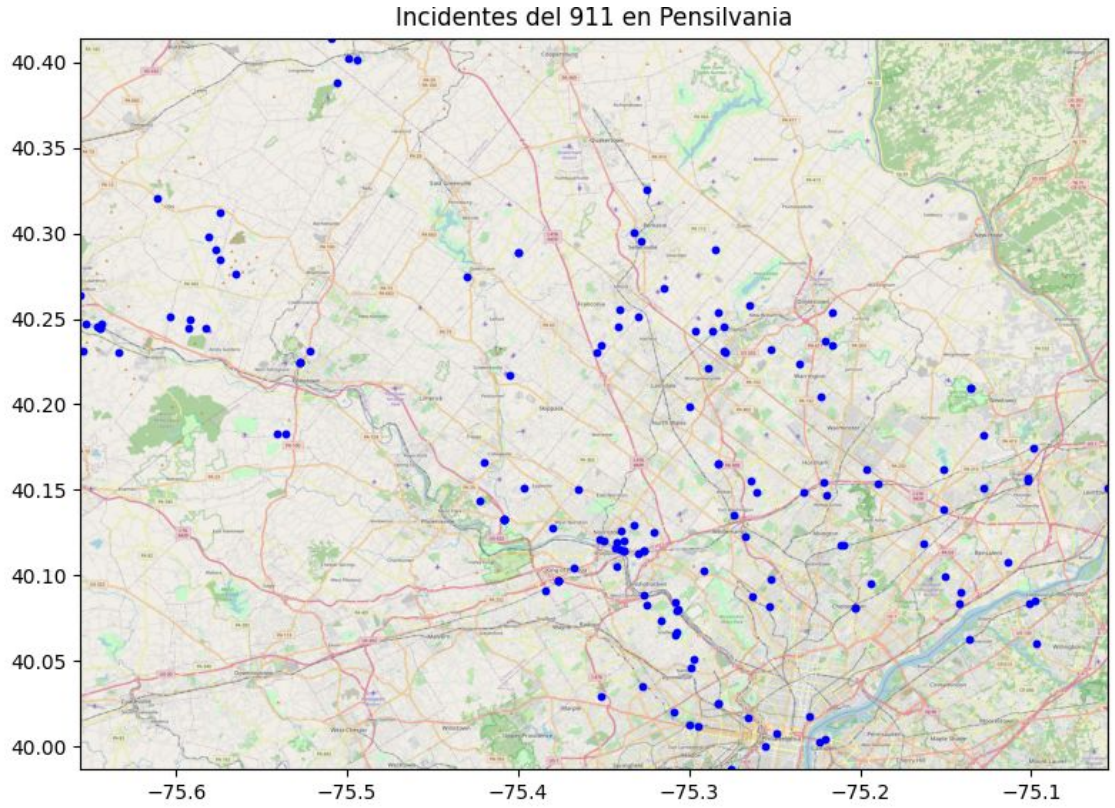


**Figura 6.** Mapa de incidentes en Pennsylvania

**Código:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv ('911.csv')
BBox = (df.lng.min(), df.lng.max(),df.lat.min(), df.lat.max())

ruh_m = plt.imread ('map.png')
fig, ax = plt.subplots(figsize = (8,7))
ax.scatter(df.lng, df.lat, zorder=1, alpha= 1, c='b', s=10)
ax.set_title('Incidentes del 911 en Pensilvania')
ax.set_xlim(BBox[0],BBox[1])
ax.set_ylim(BBox[2],BBox[3])
ax.imshow(ruh_m, zorder=0, extent = BBox, aspect= 'equal')
plt.show()
```

**Anexos:**

- Base de datos acerca de llamadas del 911 en EU. https://www.kaggle.com/mchirico/montcoalert
- Carpeta de google drive con los archivos utilizados a lo largo del proyecto. https://drive.google.com/drive/folders/1FNq4aAwo22yXgU2iL7JDlAy4LOZE88cE?usp=sharing

**Referencias:**


- Mike Chirico, “Emergency - 911 Calls.” Kaggle, 2020, doi: 10.34740/KAGGLE/DSV/1381403.