USO DE ARMAS EN LA CIUDAD DE TORONTO EN CANADÁ DEL 2004 AL 2022

Jarren Chaves Vizcaíno
Universidad Nacional de Costa Rica,
Heredia, Costa Rica
jarren.chaves.vizcaino@est.una.ac.cr

Andrea Chacón Chacón
Universidad Nacional de Costa Rica,
Heredia, Costa Rica
andrea.chacon.chacon@est.una.ac.cr

Santiago Azofeifa Benavides
Universidad Nacional de Costa Rica
Heredia, Costa Rica
santiago.azofeifa.benavides@est.una.ac.cr

Oscar Gutiérrez Rosales
Universidad Nacional de Costa Rica
Heredia, Costa Rica
oscar.gutierrez.rosales@est.una.ac.cr

I. INTRODUCCION

En el vasto universo de la programación, el lenguaje Python se destaca como una herramienta versátil y poderosa que ha revolucionado la forma en que abordamos una amplia gama de tareas. En este artículo, se explorarán las múltiples facetas en las que Python se ha convertido en un aliado indispensable en diversos campos, simplificando y agilizando procesos que, de otra manera, consumirían tiempo y recursos considerablemente.

Desde su creación a principios de la década de 1990 por Guido van Rossum, Python ha evolucionado para convertirse en uno de los lenguajes de programación más populares y ampliamente adoptados en todo el mundo. Su popularidad no solo se debe a su sintaxis simple y legible, sino también a su robusto ecosistema de bibliotecas y herramientas que hacen posible su aplicación en una amplia variedad de dominios, desde la ciencia de datos hasta el desarrollo web y la automatización de tareas.

A lo largo de este artículo, se demostrará cómo Python se ha convertido en una herramienta de mucha ayuda para temas y entornos de análisis de datos, además de que es indispensable para programadores, científicos de datos, ingenieros, investigadores y profesionales de diferentes campos. Se descubrirá cómo Python ha permitido la simplificación de tareas complejas, la optimización de procesos y el acceso a soluciones efectivas en una amplia gama de escenarios, demostrando que este lenguaje de programación se ha convertido en un compañero inestimable en el mundo digital actual.

II. MARCO TEORICO

Jupyter Notebooks son herramientas versátiles para enseñanza y colaboración en línea. El artículo propone una solución para profesores: un sistema multiusuario basado en VirtualBox y JupyterHub. Describe la creación de una máquina virtual y cómo los profesores pueden administrar usuarios y el sistema virtualizado [1].

Se destaca el papel de Python, junto con otros paradigmas de programación, en el ámbito de la tecnología de la información. Se resalta su rápida velocidad de desarrollo y su evolución desde la década de 1980 hasta convertirse en un lenguaje multiparadigma relevante en la era del Big Data. El artículo también enfatiza cómo Python se integra en diversas comunidades y su capacidad para abordar áreas y aplicaciones específicas de manera efectiva en comparación con otros lenguajes de programación [2].

Se destaca la eficacia de Python para abordar problemas como los cálculos manuales y los dibujos complicados en el análisis de circuitos y cómo su amplia gama de bibliotecas científicas puede mejorar el análisis. Se mencionan aplicaciones específicas de Python en la resolución de ecuaciones, la representación gráfica y el cálculo lógico en el análisis de circuitos. También describe cómo las bibliotecas Sympy y Matplotlib se utilizan para resolver ecuaciones y dibujar curvas de señales, respectivamente. Y se resalta cómo el uso de Python puede agilizar el diseño lógico y cómo la evolución tecnológica seguirá mejorando la eficiencia en el análisis de circuitos [3].

Python y MATLAB son usados en problemas electromagnéticos. El artículo compara su rendimiento en cálculos y tiempos de ejecución, y demuestra su eficacia en un problema realista [4].

La precisión del diagnóstico médico depende de imágenes precisas. El artículo propone usar métodos de mejora en imágenes médicas con Python, mejorando detalles y permitiendo a médicos enfocarse mejor [5].

Se presenta un motor de simulación desarrollado en Python para sistemas continuos. El enfoque se basa en características de CSMP y software de simulación continua. El objetivo es describir la necesidad y uso del motor, analizar conceptos de Python utilizados en su implementación (API, comunicación con sistemas inteligentes, procesamiento de datos en tiempo real, etc.) y mostrar la arquitectura del motor, sus casos de uso y la

aplicación de conceptos en el proceso de desarrollo. La conclusión discute resultados, mejoras futuras y actualizaciones del motor [6].

Este artículo presenta una herramienta educativa en Python para simular el control del ángulo de un motor de corriente continúa utilizando un controlador PID. La aplicación muestra gráficos y animaciones que ayudan a los estudiantes a entender mejor el concepto de control PID y a visualizar el comportamiento del sistema en la clase de control con retroalimentación [7].

Mayavi es un paquete de visualización científica en 3D de código abierto, integrado con el ecosistema de paquetes científicos de Python. Ofrece herramientas para desarrollar aplicaciones desde visualización interactiva de datos hasta soluciones personalizadas para usuarios finales [8].

Python se ha establecido como el estándar para la investigación científica interactiva y basada en cálculos. Se exploran las ventajas de Python en la investigación y destaca bibliotecas esenciales y herramientas utilizadas en este campo [9].

Con el auge de la tecnología y la importancia de la alfabetización en información, el artículo aborda la enseñanza de programación en Python en universidades. Describe la creación de una plataforma de enseñanza y un método mixto de enseñanza que combina la enseñanza en el aula con la enseñanza en línea. Los resultados demuestran mejoras en la eficiencia de aprendizaje de los estudiantes [10].

III. DESCRIPCIÓN DE PÁGINA

La página web es un recurso del Servicio de Policía de Toronto que ofrece información sobre el uso de los datos abiertos y las estadísticas del crimen en la ciudad. También contiene una sección de preguntas frecuentes que responde a algunas dudas comunes sobre la privacidad, la seguridad, el mapeo, la analítica y la experiencia de usuario de los datos abiertos. Esta proporciona enlaces a otros recursos y contactos relacionados con los datos abiertos y el Servicio de Policía de Toronto, así como los términos de uso y la licencia de los datos.

Entre los datos abiertos que ofrece la página web, hay una gran cantidad de archivos CSV que se pueden descargar y analizar. Todos estos archivos son de acceso libre y gratuito, y contienen información sobre diferentes aspectos del crimen en Toronto.[1] Uno de estos archivos es el llamado:

"Shooting_and_Firearm_Discharges_Open_Data", el cual contiene datos sobre los tiroteos y disparos de armas de fuego ocurridos en la ciudad desde el año 2004 hasta el 2023. Este archivo se tornó interesante debido a que permite observar y analizar que incluso en países que se consideran muy seguros, igualmente se dan este tipo de altercados.

IV. DESCRIPCIÓN DE PROCESAMIENTO DE LA INFORMACIÓN

Para la realización de los análisis, las funciones y las gráficas, se hizo uso de la librería de Python llamada Pandas, además, se utilizó también la librería Matplotlib, para la parte relacionada con los gráficos. A continuación, se presentarán las distintas funciones realizadas, junto con su sintaxis y sus respectivas salidas.

A. Obtener las tres primeras fechas con mayores muertes registradas.

Para la solución de este problema fue necesario primero filtrar todos los datos que tuvieran su columna de muertes con un valor mayor que 0, y luego solo se le solicita que muestre los primeros 3 datos que están en el CSV y muestre de estos: El año, el mes, el día, y la cantidad de muertes que ocurrieron.

```
def obtener_top_muertes_por_fecha(data):

    datos_filtrados = data.loc[data['DEATH'] >
0]

    datos_ordenados =
datos_filtrados.sort_values(by=['DEATH'],
ascending=False)

    top_muertes = datos_ordenados[['YEAR',
'MONTH', 'DAY','DEATH']].head(3)

    return top_muertes

resultado = obtener_top_muertes_por_fecha(data)

print(resultado)
```

YEAR	MONTH	DAY	DEATH
2005	September	16	3
2008	July	20	3
2010	September	29	2

Figura 1. Obtener las tres primeras fechas con mayores muertes registradas.

Con la salida que muestra el programa en la figura 1, se puede deducir que los años más agresivos en Toronto estuvieron entre 2005 hasta 2010.

B. Saber cuál fue el año con más muertes en el periodo de tiempo de estudio.

Como la salida anterior muestra las tres fechas con más muertes, ahora interesa confirmar si alguno de esos años fueron los más agresivos durante el periodo. Para este problema se decidió usar el método idmax (), que determina en este caso el año en el que se registraron más muertes, pero antes de poder aplicarlo se necesita un proceso, que es, sumar todas las muertes de los años específicos por medio del método sum().

[1] https://data.torontopolice.on.ca/datasets/shooting-and-firearm-discharges-open-data/explore

```
def anio_mas_muertes():

    muertes_por_anio =
df.groupby('YEAR')['DEATH'].sum()

    anio_con_mas_muertes =
muertes_por_anio.idxmax() #Evaluar

    return anio_con_mas_muertes

anio_con_mas_muertes = anio_mas_muertes()

print(f"Anio con mas muertes:
{anio_con_mas_muertes}")
```

```
Anio con mas muertes: 2005
```

Figura 2. Saber cuál fue el año con más muertes en el periodo de tiempo de estudio.

Como se muestra en la figura 2, se confirma que el año más agresivo fue el 2005.

C. Obtener los meses con más personas heridas en el año 2005.

Por ahora solo se han analizado los datos por medio de las muertes, pero interesa además saber la cantidad de heridos que tuvo ese año durante sus meses respectivos, y mostrar sus datos ordenados de mayor a menor, que para esto último se utiliza la función `sort_values(ascending=False)`.

```
def heridos_por_mes_2005():

    heridos_por_mes = df[df['YEAR'] ==
2005].groupby('MONTH')['INJURIES'].sum()

    heridos_por_mes =
heridos_por_mes.sort_values(ascending=False)

    return heridos_por_mes

heridos_por_mes_2005 = heridos_por_mes_2005()

print(heridos_por_mes_2005)
```

```
MONTH
June         24
August       22
July         20
April        16
December     15
May          15
October      14
November     13
March        12
February     11
January       8
September     8
```

Figura 3. Obtener los meses con más personas heridas en el 2005.

Como se observa gracias a la salida mostrada en la figura 3, se sabe que en todos los meses del año 2005 se registraron al menos 8 heridos.

D. Impresión de los meses en orden descendente en los cuales más altercados ocurrieron de todos los años dentro del periodo.

Para la resolución del problema que se propuso, primero se tomó un conjunto de datos filtrado por años, y luego se aplicó el `metodo value_counts()` para saber la cantidad de altercados que ocurrieron.

```
def contar_meses_desde_anio(data,
anio_minimo=2004):

    data_filtrada = data.loc[data['YEAR'] >=
anio_minimo]

    conteo_meses =
data_filtrada['MONTH'].value_counts()#Evaluar

    return conteo_meses

resultado = contar_meses_desde_anio(data)

print(resultado)
```

```
August       604
July         587
September    537
May          520
June         491
October      488
January      440
April        438
November     427
March        424
December     410
February     341
```

Figura 4. Impresión de los meses en orden descendiente en los cuales ocurrieron más altercados ocurrieron dentro de todos los años del periodo.

Con esta pequeña tabla mostrada en la figura 4, la cual corresponde a la salida de la función, se confirma que el mes de agosto fue el mes en el que más altercados ocurrieron durante el periodo.

E. ¿Cuál será la cantidad de altercados ocurridos en un tiempo especificado en el año 2010?

Como se sabe del punto A, el año 2010 tuvo una de las fechas más agresivas en cuanto a muertes se refiere del periodo, entonces se plantea la siguiente solución a la pregunta dada, se creó una función que filtra los datos en el año 2010 por medio de una de las tres opciones que el usuario digitó, y con esto por medio de varias estructuras condicionales como lo son el “If, Else”, se dan los datos filtrados y con eso se hace la suma de las muertes y heridos en el tiempo digitado.

```
def sumar_muertes_y_heridas_2010(data, tiempo):
       data_2010 = data[data['YEAR'] == 2010]

    if tiempo == 'Afternoon':
        data_filtrada =
data_2010[(data_2010['TIME_RANGE']) ==
'Afternoon' ]
    elif tiempo == 'Evening':
        data_filtrada =
data_2010[(data_2010['TIME_RANGE'])==
'Evening']
    elif tiempo == 'Night':
        data_filtrada =
data_2010[(data_2010['TIME_RANGE'] ) ==
'Night']

      suma_muertes =
data_filtrada['DEATH'].sum()
    suma_heridas =
data_filtrada['INJURIES'].sum()

    return suma_muertes, suma_heridas
```

```
Ingrese un periodo del dia para el filtrado: Night
Para el tiempo 'Night' en el año 2010,
la suma de muertes es 10 y la suma de personas heridas es 53
```

Figura 6. ¿Cuál será la cantidad de altercados ocurridos en un tiempo especificado en el año 2010?

Primero se solicita al usuario que digite un periodo de las tres opciones, una vez digitado el periodo podemos ver en la figura 5, de forma clara y dividida la suma de muertes y personas heridas que hubo en el periodo digitado y el año 2010.

F. Se sabe el año con más muertes registradas. ¿Pero y el año con más personas heridas y cuantos fueron registrados?

Para resolver la pregunta presentada, se hace uso del método sum() e idmax(), para tomar el mayor con su respectiva suma, y asi dar respuesta a la pregunta.

```
def anio_mas_heridos():
    heridas_por_anio =
df.groupby('YEAR')['INJURIES'].sum()
    anio_con_mas_heridos =
heridas_por_anio.idxmax()
    total_heridos = heridas_por_anio.max()
    return anio_con_mas_heridos, total_heridos

anio_con_mas_heridos, total_heridos =
anio_mas_heridos()
print(f"Anio con mas heridos:
{anio_con_mas_heridos}")
print(f"Total de heridos en ese año:
{total_heridos}")
```

```
Anio con mas heridos: 2019
Total de heridos en ese año: 240
```

Figura 8. Se sabe el año con más muertes registradas. ¿Pero y el año con más personas heridas y cuantos fueron registrados?

Y como se puede observar el año con más heridos fue 2019 con un total de 240.

G. Datos en el mes de septiembre.

Con base en el punto A, se sabe que una de las fechas con mayores mayor número de asesinatos se ubica en el año 2005, por lo que se quiere saber todos los hechos reportados de ese año.

```
def dataSetiembre():
  return data.loc[(data['MONTH']=='September')]
print(dataSetiembre())
```

	X	Y	OBJECTID	EVENT_UNIQUE_ID	DATE
7	-79.588530	43.738688	8	GO-2004646393	2004/09/08 04:00:00+00
20	-79.373917	43.653962	21	GO-2004649788	2004/09/11 04:00:00+00
22	-85.488744	0.000000	23	GO-2004591518	2004/09/03 04:00:00+00
38	-79.302434	43.797213	39	GO-2004497549	2004/09/30 04:00:00+00
42	-79.328071	43.667525	43	GO-2004668846	2004/09/24 04:00:00+00
...	...	...	...	...	...
5664	-79.385553	43.648863	5665	GO-20221766117	2022/09/11 04:00:00+00
5669	-79.489545	43.746807	5670	GO-20221757893	2022/09/10 04:00:00+00
5696	-79.255781	43.736698	5697	GO-20221839939	2022/09/21 04:00:00+00
5700	-79.537787	43.717190	5701	GO-20221758114	2022/09/10 04:00:00+00
5705	-79.443993	43.658083	5706	GO-20221765442	2022/09/11 04:00:00+00

Figura 5. Datos en el mes de septiembre.

En la figura 7, se pueden ver 10 filas y columnas de los hechos ocurridos, en ese mes, cabe resaltar que si se ejecuta el código que está arriba se desplegará toda la información de ese año.

H. Mes con más muertes durante el año 2005.

Gracias al punto A se ve que el año 2005 posee la fecha con más muertes registradas, pero ahora se quiere obtener el mes en el cual más muertes se reportaron en ese año. Para ello se usa el metodo sum() y junto al el método idmax() para obtener el mes con mayor número de muertes.

```
def mes_mas_muertes_2005(data_2005):
    muertes_por_mes =
data_2005.groupby('MONTH')['DEATH'].sum()
    mes_con_mas_muertes_2005 =
muertes_por_mes.idxmax()
    return mes_con_mas_muertes_2005

data_2005 = df[df['YEAR'] == 2005]
mes_con_mas_muertes =
mes_mas_muertes_2005(data_2005)
print(f"Mes con más muertes en el año 2005:
{mes_con_mas_muertes}")
```

```
Mes con más muertes en el año 2005: August
```

Figura 7. Mes con más muertes durante el año 2005.

Con la impresión mostrada en la figura 7, se confirma que el mes con más muertes registradas fue agosto.

I. Obtener todos los datos del año 2010.

Con el punto A, se sabe que otra de las fechas con mayor número de muertes ocurrió en el año 2010, por lo que también interesa saber que pasó durante todo ese año. Para esta inquietud solo se utiliza el metodo de la librería pandas .loc, y así se obtiene solo las filas que tengan el 2010.

```
def data2010():
  return data.loc[(data['YEAR']==2010)]
print(data2010())
```

	X	Y	OBJECTID	EVENT_UNIQUE_ID	DATE
1365	-79.432442	43.639686	1366	GO-20103719744	2010/11/11 05:00:00+00
1366	-79.240108	43.788863	1367	GO-20103842165	2010/05/12 04:00:00+00
1367	-79.428126	43.635975	1368	GO-20103778563	2010/08/27 04:00:00+00
1368	-79.554057	43.695335	1369	GO-20103801910	2010/12/31 05:00:00+00
1369	-79.567160	43.683330	1370	GO-20102384820	2010/09/22 04:00:00+00
...	...	...	...	...	...
1619	-79.185722	43.753568	1620	GO-20103762306	2010/06/01 04:00:00+00
1620	-79.495071	43.688950	1621	GO-20103812935	2010/09/04 04:00:00+00
1621	-79.500001	43.761057	1622	GO-20103446670	2010/11/03 04:00:00+00
1622	-79.405829	43.656852	1623	GO-20103885505	2010/11/01 04:00:00+00
1623	-79.603912	43.743634	1624	GO-20103859861	2010/07/03 04:00:00+00

Figura 9. Obtener todos los datos del año 2010.

Al igual que en la figura 5, solo se muestran 5 columnas de los datos, pero se insiste en que la lista completa puede ser observada por medio de la función definida en este punto.

J. Datos filtrados por el mes de enero en el año 2010.

Ya que en el punto A, el año 2010 tiene una de las fechas, interesa saber cómo fue que comenzó ese año, por lo tanto, como solución se hace uso del metodo .loc de la librería de pandas, y se escribe que se van a querer los datos con el año 2010 y el mes de enero al mismo tiempo.

```
def data2010_Enero():
  return data.loc[(data['YEAR']==2010) &
(data['MONTH']=='January')]
print(data2010_Enero())
```

	X	Y	OBJECTID	EVENT_UNIQUE_ID	DATE
1386	-79.508884	43.696808	1387	GO-20103651358	2010/01/29 05:00:00+00
1397	-79.213764	43.744333	1398	GO-20103133204	2010/01/07 05:00:00+00
1415	-79.366042	43.662164	1416	GO-20103648874	2010/01/29 05:00:00+00
1442	-79.448227	43.719106	1443	GO-20103665325	2010/01/07 05:00:00+00
1449	-79.499791	43.624881	1450	GO-20103669221	2010/01/30 05:00:00+00
1522	-79.322706	43.697241	1523	GO-20103720862	2010/01/14 05:00:00+00
1544	-79.217662	43.798116	1545	GO-20103688752	2010/01/14 05:00:00+00
1548	-79.297019	43.691738	1549	GO-20103658361	2010/01/24 05:00:00+00

Figura 10. Datos filtrados por el mes de enero en el año 2010.

Al igual que en funciones anteriores, la figura 10 solo muestra 5 columnas y 8 filas, se insiste en que, si se ejecuta la función, esta muestra todos los datos.

V. ANÁLISIS DE LOS DATOS

Para poder analizar los datos de una manera más explícita y representativa se hará uso de la librería Matplotlib para de esta manera generar gráficos y poder entender un poco mejor de todo lo que se ha hablado.

Por medio de la función .plot.barh(color="gray"), se generará un gráfico de barras horizontales que va a mostrar el promedio de muertes de los primeros 10 barrios del CSV escogido.

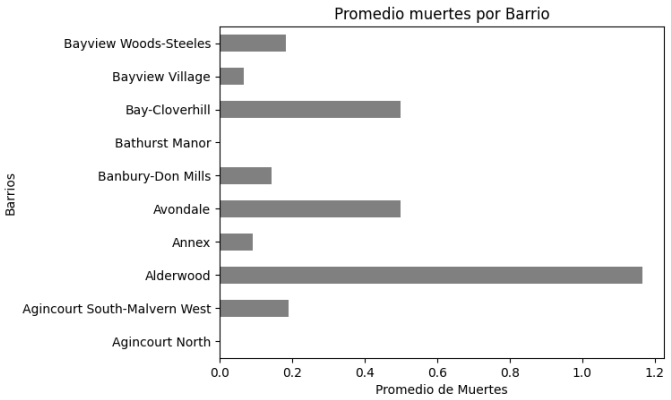

Gráfico 1. Promedio Muertes por Barrio.

Para un segundo caso de análisis, el cual corresponde a poder visualizar en que etapa del día (mañana, día y noche) ocurrieron los datos, para lo cual se realizó un histograma el cual muestra por la parte inferior, las distintas etapas del día y en la parte izquierda, muestra los diferentes rangos de cantidades de muertes. Para ilustrar esto se adjunta la siguiente imagen.

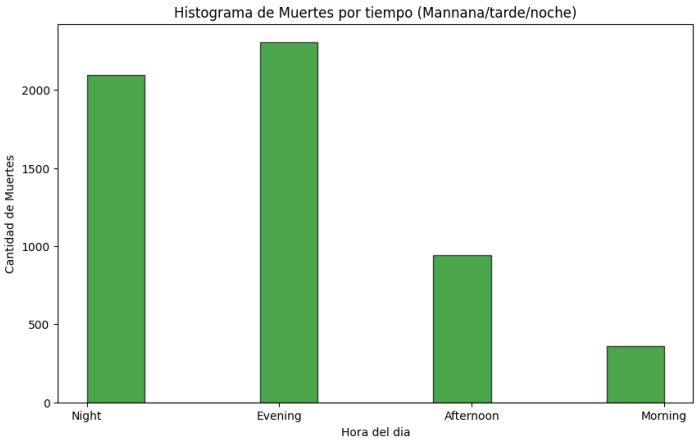

Gráfico 2. Histograma Muertes por Tiempo.

Para obtener una representación más fácil de entender de los datos, se decidió utilizar un gráfico de pastel, donde se representa el total de altercados del periodo de estudio entre los días de la semana, así se puede observar muy fácilmente que el día en el que más altercados ocurrieron fue el día domingo o "Sunday" en caso del gráfico 3 con un 19.2% del total de altercados.

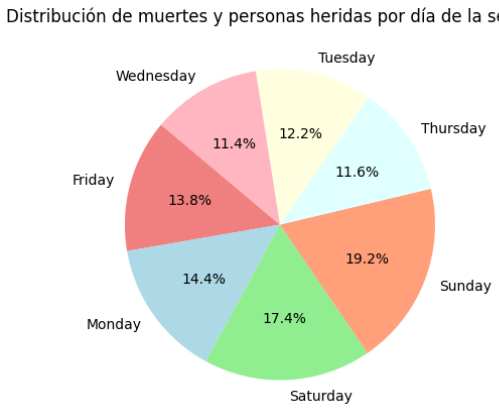

Gráfico 3. Distribución de Muertes y personas heridas por día de la semana.

Con el siguiente gráfico de barras se puede observar que todos los días están por encima de los 300 altercados totales. Para su construcción se utiliza el método groupby() de la librería Pandas, para conseguir en número exacto de muertes y personas heridas, para estos dos resultados sumarlos y llamar a la suma "total_combinado" y con esta suma utilizar la función plot() para asi mostrar el gráfico 4.

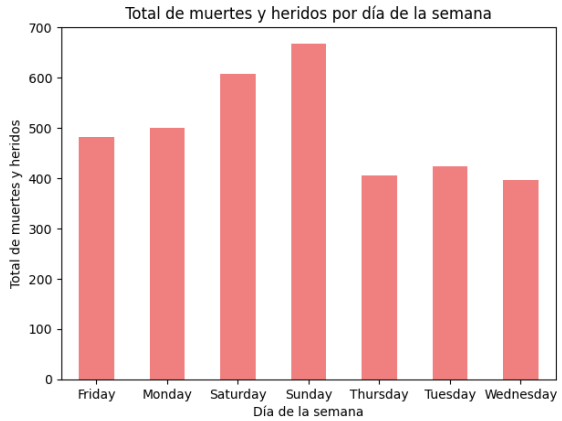

Gráfico 4. Total de muertes y heridos por día de la semana.

Para los dos siguientes mapas de calor se hizo uso de la librería Seaborn, como en la función D del punto IV, en la cual se saca el total de altercados por mes, ahora se quiere representar de una manera más visual, para ello se utilizan métodos como pyvot_table para poder crear una tabla que va a ser la base del mapa de calor, y luego se utiliza la función sns.heatmap de la librería Seaborn para construir el mapa de calor. A continuación, se presentarán los mapas de calor correspondientes a la cantidad de personas heridas por mes y año, y la cantidad de muertes por mes y año también.

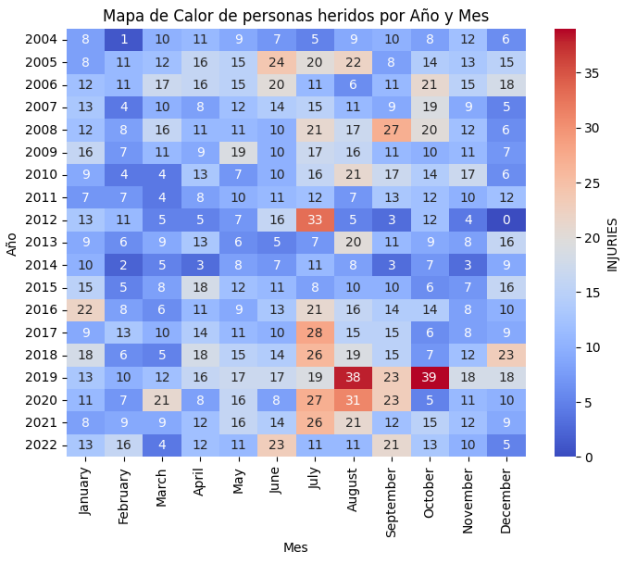

Gráfico 5. Mapa de Calor de personas heridas por Año y Mes.

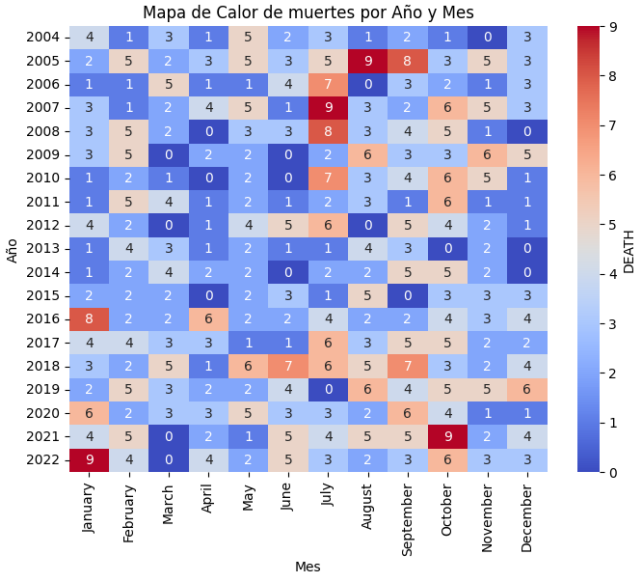

Gráfico 6. Mapa de Calor de muertes por Año y Mes.

VI. CONCLUSIONES Y RECOMENDACIONES

Disponibilidad de Datos Abiertos: La página web del Servicio de Policía de Toronto ofrece una amplia gama de datos abiertos en formato CSV que están disponibles de manera gratuita para el público. Esto demuestra un compromiso con la transparencia y la accesibilidad de la información relacionada con el crimen en la ciudad.

Acceso a Datos de Crimen: Los archivos CSV disponibles incluyen información detallada sobre diversos aspectos del crimen en Toronto, lo que brinda a investigadores, periodistas y ciudadanos la oportunidad de analizar y comprender mejor la situación de seguridad en la ciudad.

Incluso en Países Seguros Ocurren Tiroteos: La presencia de un archivo específico llamado:

"Shooting_and_Firearme_Discharges_Open_Data", el cual abarca tiroteos y disparos de armas de fuego desde 2004 hasta 2023, destaca que incluso en lugares considerados seguros, como Toronto, se producen incidentes relacionados con armas de fuego. Esto resalta la importancia de seguir monitoreando y abordando la violencia armada en todas las comunidades.

Recursos y Soporte: La página web también ofrece recursos adicionales, como una sección de preguntas frecuentes que aborda inquietudes comunes relacionadas con la privacidad, la seguridad y la analítica de datos abiertos. Esto demuestra un esfuerzo por brindar apoyo a aquellos que desean utilizar estos datos de manera efectiva.

Transparencia y Colaboración: La disponibilidad de enlaces a otros recursos y contactos relacionados con el Servicio de Policía de Toronto, así como los términos de uso y la licencia de los datos, refleja un enfoque de colaboración y transparencia en la divulgación de datos, lo que fomenta una mayor participación y comprensión de la comunidad en temas de seguridad y crimen.

Para la parte de las recomendaciones se tiene lo siguiente:

Recopilación y limpieza de datos es esencial para asegurarse de que los datos utilizados en la investigación sean de alta calidad y precisos. La calidad de los datos es fundamental para obtener resultados confiables y significativos. Esto implica la recopilación de datos de fuentes confiables y la implementación de un proceso de limpieza para eliminar errores, duplicados y valores atípicos. La limpieza de datos garantiza que los datos estén en un formato coherente y listos para el análisis.

El análisis temporal es una parte importante de la investigación, ya que permite comprender cómo cambian los datos con el tiempo. Identificar tendencias temporales, como aumentos o disminuciones en ciertos períodos, puede proporcionar información valiosa. Esto implica examinar años, meses o períodos específicos de interés y evaluar cómo se relacionan con los resultados de la investigación.

La visualización de datos desempeña un papel crucial en la comunicación de los hallazgos de la investigación. Utilizar herramientas de visualización, como Matplotlib, permite representar gráficamente los resultados. Los gráficos y las visualizaciones facilitan la comprensión de patrones y tendencias en los datos. Los gráficos pueden ayudar a identificar relaciones y patrones que pueden no ser evidentes en los datos sin procesar.

La interacción con el usuario es importante para adaptar la investigación a las necesidades individuales. Incorporar elementos de interacción, como la capacidad de seleccionar períodos de tiempo específicos o ajustar los parámetros del análisis, hace que la investigación sea más personalizada y útil para los usuarios finales. Esto permite a los usuarios explorar los datos de manera más efectiva y obtener información relevante para sus necesidades.

La documentación adecuada es esencial para garantizar la transparencia y la reproducibilidad de la investigación. Esto implica documentar de manera completa y precisa el proceso de investigación, incluyendo la metodología utilizada, las fuentes de datos, los pasos de limpieza y análisis, y cualquier suposición o decisión tomada durante el proceso. La documentación adecuada facilita la revisión y validación de los resultados por parte de otros investigadores y asegura la integridad de la investigación.

VII. REFERENCIAS

[1] A. P. Lorandi Medina, G. M. Ortigoza Capetillo, G. H. Saba, M. A. H. Perez, and P. J. Garcia Ramirez, “A Simple Way To Bring Python To The Classrooms,” in 2020 IEEE International Conference on Engineering Veracruz (ICEV), Boca del Rio, Mexico: IEEE, Oct. 2020, pp. 1–6. doi: 10.1109/ICEV50249.2020.9289692.

[2] A. Kumar and Supriya. P. Panda, “A Survey: How Python Pitches in IT-World,” in 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India: IEEE, 2019, pp. 248–251. doi: 10.1109/COMITCon.2019.8862251. C. Kaur and A. Sharma, "Social Issues Sentiment Analysis using Python," *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, 2020, pp. 1-6, doi: 10.1109/ICCCS49678.2020.9277251.

[3] H. Zhang, P. Yang, and Y. Niu, “Application of Python Scientific computing library and Simulation in Circuit Analysis,” in 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India: IEEE, Apr. 2023, pp. 892–898. doi: 10.1109/CSNT57126.2023.10134600.

[4] A. Weiss and A. Elsherbeni, “Computational Performance of MATLAB and Python for Electromagnetic Applications,” in 2020 International Applied Computational Electromagnetics Society Symposium (ACES), Monterey, CA, USA: IEEE, 2020, pp. 1–2. doi: 10.23919/ACES49320.2020.9196078.

[5] H. Zhou and S. Wu, “Design of medical image enhancement algorithm based on Python,” in 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China: IEEE, Jan. 2021, pp. 482–485. doi: 10.1109/ICPECA51329.2021.9362581.

[6] T. Naumovic, M. Despotovic-Zrakic, B. Radenkovic, L. Zivojinovic, and I. Jezdovic, “Development of a Continuous System Simulation Engine in Python Programing Language,” in 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina: IEEE, 2020, pp. 1–5. doi: 10.1109/INFOTEH48170.2020.9066334.

[7] W. Phutthanukun and P. Chayratsami, “Development of Angle Control System Application Using Python,” in 2019 IEEE 11th International Conference on Engineering Education (ICEED), Kanazawa, Japan: IEEE, 2019, pp. 128–132. doi: 10.1109/ICEED47294.2019.8994929.

[8] P. Ramachandran and G. Varoquaux, “Mayavi: 3D Visualization of Scientific Data,” Comput. Sci. Eng., vol. 13, no. 2, pp. 40–51, 2011, doi: 10.1109/MCSE.2011.35.

[9] J. Millman and M. Aivazis, “Python for Scientists and Engineers,” Comput. Sci. Eng., vol. 13, no. 2, pp. 9–12, 2011, doi: 10.1109/MCSE.2011.36.

[10] S. Zhou, Z. He, N. Xiong, and X. Liu, “Research and Application of Mixed Teaching Method of Python Programming Based on SPOC,” in 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China: IEEE, 2019, pp. 189–193. doi: 10.1109/ICISCAE48440.2019.221615.