



Pontificia Universidad Javeriana

Samuel Peña García

Tomás De Aza Márquez

Juan Sebastián Córdoba Valderrama

*Uso de Machine Learning y Big Data en la Toma de Decisiones Gubernamentales en
Nueva York*

Procesamiento de Datos a Gran Escala

2024-10

Bogotá, Colombia

Índice

Entendimiento del negocio	1
Selección de los datos a utilizar	1
Colección y descripción de datos	2
Tipos de datos	3
Comprensión del significado de cada atributo	3
Descripción general del contenido de los conjuntos de datos	4
Exploración de los datos	5
Reporte de calidad de datos	9
Planteamiento de preguntas sobre los datos	11
Filtros, limpieza y transformación inicial.....	11
Bono 1	13
Bono 2	15
Solución a las preguntas planteadas	17
Selección de técnicas de aprendizaje de máquina.....	23
Aplicación de los modelos de Machine Learning.....	24
Conclusiones.....	27
Bono	27

Entendimiento del negocio

El contexto general de la situación en Nueva York se centra en la necesidad de mejorar algunos indicadores territoriales, en particular, la cantidad de arrestos y la cantidad de accidentes viales. Para abordar estos problemas, se han proporcionado datos específicos como los de arrestos, pobreza, accidentes viales y niveles de educación en Nueva York.

Los indicadores macroeconómicos relevantes que podrían considerarse en este contexto incluyen tasas de desempleo, ingresos per cápita, niveles de pobreza, y distribución de la educación, ya que estos factores pueden estar correlacionados con las tasas de criminalidad y accidentes. Estos indicadores ayudarán a comprender las posibles causas o factores contribuyentes a los problemas identificados y a formular intervenciones efectivas.

El objetivo del equipo de consultoría, en este caso, es desarrollar un plan de acción basado en el procesamiento de datos a gran escala para mejorar los indicadores mencionados. Esto implica analizar los conjuntos de datos suministrados, identificar patrones o tendencias, y proponer soluciones o recomendaciones basadas en los hallazgos. La meta es utilizar el procesamiento de datos y el análisis para informar y guiar las decisiones políticas y las intervenciones del gobierno del estado de Nueva York, con el fin de abordar y mitigar las preocupaciones relacionadas con los arrestos y los accidentes viales.

Selección de los datos a utilizar

Para desarrollar el plan de acción basado en datos, es necesario definir de manera clara cuáles serán estos datos para utilizar, y ante nada, verificar la veracidad y la confiabilidad de las fuentes de donde se obtengan estos datos. De esta manera, se buscaron conjuntos de datos relacionados con el objetivo y los temas de estudio para poder realizar el respectivo procesamiento y análisis.

Se seleccionó el centro de datos abiertos de Nueva York, fuente muy confiable —primaria, teniendo en cuenta que tiene relación directa con el Ayuntamiento—. Es ahí donde se encontraron los siguientes conjuntos de datos:

Nombre	Detalle	Enlace
NYPD Arrest Data	Detalle de cada arresto efectuado en Nueva York por el cuerpo de policía.	Vínculo
NYCgov Poverty Measure Data	Datos de pobreza en Nueva York basados en una micro muestra de una encuesta.	Vínculo
Motor Vehicle Collisions – Vehicles	Detalle de cada vehículo envuelto en accidentes de tránsito en Nueva York desde 2016.	Vínculo
2016 – 2017 Health Education Report	Reporte del nivel educación proporcionado por el Departamento de educación de la ciudad de Nueva York	Vínculo

Teniendo en cuenta esto, se escogió el conjunto de datos “NYPD Arrest Data” para analizar, tal y como lo indica el nombre del conjunto, los datos de los arrestos de Nueva York, teniendo en cuenta el objetivo de negocio y la cantidad de datos relevantes y representativos que se encuentran allí —revisado con más detalle en el siguiente apartado—.

Colección y descripción de datos

Para iniciar, se importaron las librerías relevantes y se instanció PySpark —versión 3.5.1— en Databricks Community Edition:

```
1  ##Importar librerías a utilizar
2
3  from pyspark import SparkContext
4  from pyspark.sql import *
5  from pyspark.sql.functions import *
6  from pyspark.sql.types import *
7  import pandas as pd
8  import os
```

```
1  ##Se instancia Pyspark
2  sc = SparkContext.getOrCreate()
3  sql_sc = SQLContext(sc)
4  sc
```

/databricks/spark/python/pyspark/sql/context.py:117:
warnings.warn()

SparkContext

[Spark UI](#)

Version

v3.5.1

Master

local[8]

AppName

Databricks Shell

Posteriormente, se carga el conjunto de datos a un dataframe Spark. Finalmente, se muestran las primeras filas del DataFrame verificando la correcta carga de los datos.

```
1  path1 = 'https://raw.githubusercontent.com/TommyDS2005/Proyecto-Procesamiento-de-Datos/main/NYPD_Arrest_Data_Part1.csv'
2  path2 = 'https://raw.githubusercontent.com/TommyDS2005/Proyecto-Procesamiento-de-Datos/main/NYPD_Arrest_Data_Part2.csv'
3
4  df1 = pd.read_csv(path1, sep=',')
5  df2 = pd.read_csv(path2, sep=',')
6
7  dfpd = pd.concat([df1,df2])
8
9  df = spark.createDataFrame(dfpd)
10
11 # Muestra las primeras filas del DataFrame de Spark para verificar
12 df.show()
```

284	-73.73476	POINT (-73.73476 ...													
	261635779	01/10/2023 105.0	STRANGULATION 1ST 106.0	FELONY ASSAULT PL 1211200	F	Q	101	0	18-24	M	BLACK	1050620	157860	40.599	
718	-73.760999	POINT (-73.760999...													
	261739896	01/11/2023 205.0	TRESPASS 2, CRIMINAL 352.0	CRIMINAL TRESPASS PL 1401502	M	K	88	0	18-24	F	BLACK	991150	192509	40.695	
068	-73.975116	POINT (-73.975116...													
	264206386	02/25/2023 105.0	STRANGULATION 1ST 106.0	FELONY ASSAULT PL 1211200	F	K	63	0	45-64	M	BLACK	1000520	168264	40.628	

Tipos de datos

En el conjunto de datos se presentan los siguientes atributos, con su respectivo tipo de dato:

- ARREST_KEY: long (nullable = true)
- ARREST_DATE: date (nullable = true)
- PD_CD: double (nullable = true)
- PD_DESC: string (nullable = true)
- KY_CD: double (nullable = true)
- OFNS_DESC: string (nullable = true)
- LAW_CODE: string (nullable = true)
- LAW_CAT_CD: string (nullable = true)
- ARREST_BORO: string (nullable = true)
- ARREST_PRECINCT: long (nullable = true)
- JURISDICTION_CODE: long (nullable = true)
- AGE_GROUP: string (nullable = true)
- PERP_SEX: string (nullable = true)
- PERP_RACE: string (nullable = true)
- X_COORD_CD: long (nullable = true)
- Y_COORD_CD: long (nullable = true)
- Latitude: double (nullable = true)
- Longitude: double (nullable = true)
- New Georeferenced Column: string (nullable = true)

Comprensión del significado de cada atributo

- ARREST_KEY: Identificador único y persistente generado aleatoriamente para cada arresto. Tipo de dato texto plano.
- ARREST_DATE: Fecha exacta del arresto del evento reportado. Tipo de dato fecha y hora.
- PD_CD: Código de clasificación interna de tres dígitos (más detallado que el Código Clave). Tipo de dato número.
- PD_DESC: Descripción de la clasificación interna que corresponde con el código PD (más detallada que la Descripción de la Ofensa). Tipo de dato texto plano.
- KY_CD: Código de clasificación interna de tres dígitos (categoría más general que el código PD). Tipo de dato número.

- **OFNS_DESC:** Descripción de la clasificación interna que corresponde con el código KY (categoría más general que la descripción PD). Tipo de dato texto plano.
- **LAW_CODE:** Códigos de la ley correspondientes al Código Penal de Nueva York, VTL y otras leyes locales diversas. Tipo de dato texto plano.
- **LAW_CAT_CD:** Nivel del delito: delito mayor (felony), delito menor (misdemeanor), infracción (violation). Tipo de dato texto plano.
- **ARREST_BORO:** Barrio del arresto. B (Bronx), S (Staten Island), K (Brooklyn), M (Manhattan), Q (Queens). Tipo de dato texto plano.
- **ARREST_PRECINCT:** Comisaría donde ocurrió el arresto. Tipo de dato número.
- **JURISDICTION_CODE:** Código de jurisdicción responsable del arresto. Los códigos de jurisdicción 0 (Patrulla), 1 (Tránsito) y 2 (Vivienda) representan al NYPD, mientras que los códigos 3 en adelante representan jurisdicciones ajenas al NYPD. Tipo de dato número.
- **AGE_GROUP:** Edad del perpetrador dentro de una categoría establecida. Tipo de dato texto plano.
- **PERP_SEX:** Descripción del sexo del perpetrador. Tipo de dato texto plano.
- **PERP_RACE:** Descripción de la raza del perpetrador. Tipo de dato texto plano.
- **X_COORD_CD:** Coordenada X del punto medio de la cuadra para el Sistema de Coordenadas Planas del Estado de Nueva York, Zona de Long Island, NAD 83, unidades en pies (FIPS 3104). Tipo de dato número.
- **Y_COORD_CD:** Coordenada Y del punto medio de la cuadra para el Sistema de Coordenadas Planas del Estado de Nueva York, Zona de Long Island, NAD 83, unidades en pies (FIPS 3104). Tipo de dato número.
- **Latitude:** Coordenada de latitud para el Sistema de Coordenadas Global, WGS 1984, grados decimales (EPSG 4326). Tipo de dato número.
- **Longitude:** Coordenada de longitud para el Sistema de Coordenadas Global, WGS 1984, grados decimales (EPSG 4326). Tipo de dato número.

Descripción general del contenido de los conjuntos de datos

Este conjunto de datos contiene registros detallados de arrestos efectuados por el Departamento de Policía de Nueva York (NYPD) durante el año 2023, incluyendo información demográfica del detenido, el tipo de delito, la ubicación exacta y la fecha del arresto, así como el código y descripción de la ley asociada con el arresto. Con estos datos

podremos realizar un análisis adecuado a la altura de las necesidades de un gran cliente como lo es el ayuntamiento de Nueva York.

Exploración de los datos

Para este apartado se realizaron análisis preliminares que van desde lo más sencillo como saber cuántos datos tenemos, hasta ir más allá, entendiendo valores como la varianza o analizando de forma preliminar los datos a través de una perspectiva gráfica, para conocer bien los datos que tenemos antes de proceder con transformaciones sobre el conjunto de datos.

Primeramente, se realizó un análisis estadístico de los datos:

	summary	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC
1	count	226872	226872	226870	226872	226855	226872
2	mean	2.706479248400111E8	null	424.7544011989245	null	249.3451323532653	null
3	stddev	5304010.298148365	null	274.47538060486033	null	147.68673264760517	null
4	min	261180920	01/01/2023	1.0	(null)	101.0	(null)
5	max	279779734	12/31/2023	997.0	WEAPONS,MFR,TRANSPORT,ETC.	995.0	VEHICLE AND TRAFFIC LAWS

	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECINCT	JURISDICTION_CODE	AGE_GROUP	PERP_SEX	PERP_RACE
1	226872	225273	226872	226872	226872	226872	226872	226872
2	null	9.0	null	63.43052910892486	0.9285367960788462	null	null	null
3	null	0.0	null	34.63504525700395	7.538568508006559	null	null	null
4	(null)	(null)	B	1	0	18-24	F	AMERICAN INDIAN/ALASKAN NATIVE
5	VTL21300A5	V	S	123	97	<18	U	WHITE HISPANIC

X_COORD_CD	Y_COORD_CD	Latitude	Longitude	New Georeferenced Column
226872	226872	226872	226872	226872
1005786.7287192778	208289.0843206742	40.73815365744051	-73.92191484770171	null
21509.437648151736	29744.718872647278	0.11823655424556384	0.17333780170454147	null
0	0	0.0	-74.253256	POINT (-73.70059684703173 40.7390218775969)
1067220	271819	40.912714	0.0	POINT (0 0)

Posteriormente, se realizó una pequeña conversión del dataframe, dejando el atributo *ARREST_DATE* en formato datetime, para posteriormente observar el dataframe con la ayuda de Pandas, con el fin de buscar variables para realizar gráficas y demás análisis que contribuyeran a la exploración de datos.

Table

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE
1	261265483	2023-01-03	397	ROBBERY, OPEN AREA UNCLASSIFIED	105	ROBBERY	PL
2	261271301	2023-01-03	105	STRANGULATION 1ST	106	FELONY ASSAULT	PL
3	261336449	2023-01-04	397	ROBBERY, OPEN AREA UNCLASSIFIED	105	ROBBERY	PL
4	261328047	2023-01-04	105	STRANGULATION 1ST	106	FELONY ASSAULT	PL
5	261417496	2023-01-05	244	BURGLARY, UNCLASSIFIED, UNKNOWN	107	BURGLARY	PL
6	261583093	2023-01-08	109	ASSAULT 2,1, UNCLASSIFIED	106	FELONY ASSAULT	PL
7	261611604	2023-01-08	262	ASSAULT 2,2,4	114	ASSAULT	PL

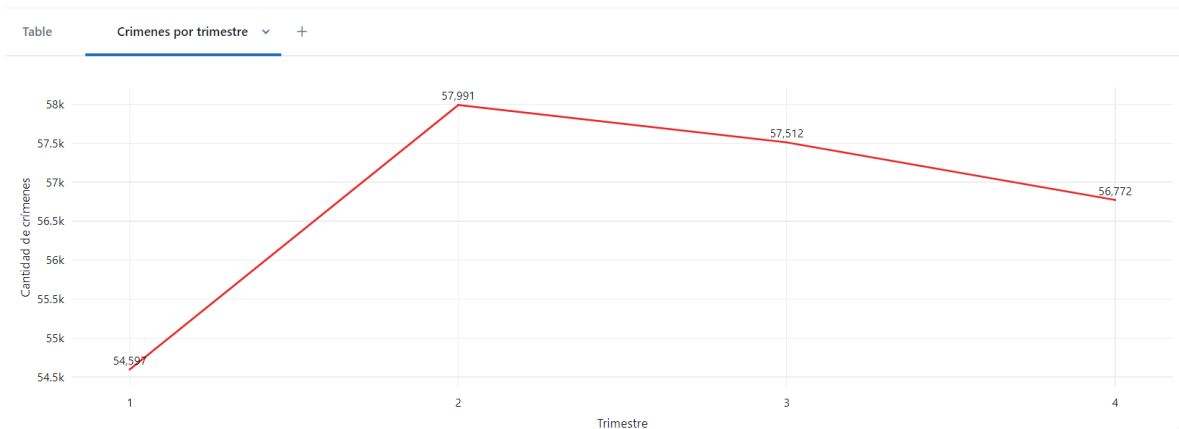
	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECINCT	JURISDICTION_CODE	AGE_GROUP	PERP_SEX	PERP_RACE
1	PL 1402000	F	K	75	0	18-24	F	BLACK
2	PL 215510B	F	K	79	0	25-44	F	BLACK
3	PL 215510B	F	K	79	0	25-44	F	BLACK
4	PL 265031B	F	M	23	0	25-44	M	WHITE HISPANIC
5	PL 215510B	F	K	94	0	25-44	M	WHITE
6	PL 2650303	F	K	77	0	18-24	M	BLACK

X_COORD_CD	Y_COORD_CD	Latitude	Longitude	New Georeferenced Column
1017119	183909	40.671404	-73.881509	POINT (-73.881509 40.671404)
999541	187345	40.680883	-73.944867	POINT (-73.944867 40.680883)
999541	187345	40.680883	-73.944867	POINT (-73.944867 40.680883)
999007	229814	40.79744999	-73.94670167	POINT (-73.94670167 40.79744999)
997245	204129	40.726956	-73.953115	POINT (-73.953115 40.726956)
1003508	185056	40.674593	-73.930572	POINT (-73.930572 40.674593)
007701	202541	40.722506	-73.951146	POINT (-73.951146 40.722506)

Teniendo en cuenta esto, se procedió a hacer las gráficas del análisis exploratorio de datos (EDA)

Primero, se buscó graficar la cantidad de arrestos a lo largo de las fechas en el dataframe — que representaban todo el año 2023—. No obstante, dada la cantidad de registros, que eran 226872, se tuvo que optar por una agrupación trimestral, para que la librería Pandas pudiera procesar todos los datos.

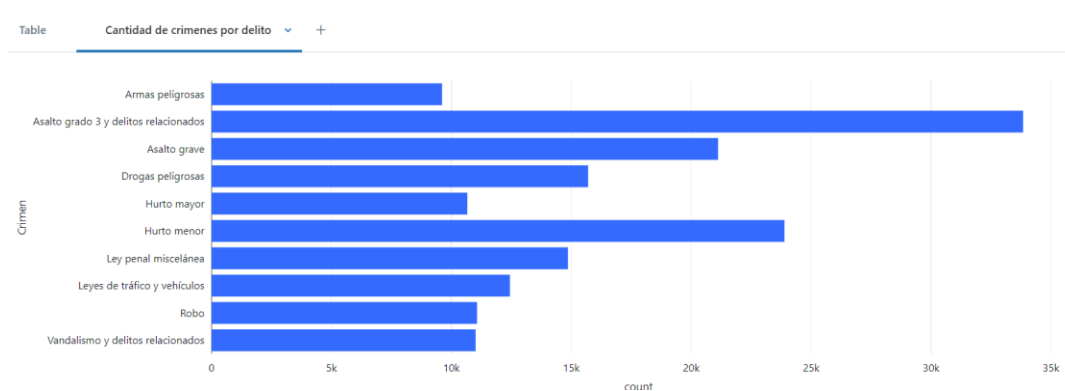
En ese orden de ideas, se creó un dataframe intermedio que incluyera el año y el trimestre del delito para posteriormente realizar una agrupación y conteo que facilitara la graficación.



Esto muestra cómo hubo un incremento significativo de los arrestos entre el primer y segundo trimestre, que posteriormente fue decreciendo, pero a una medida mucho menor que a la que se dio el crecimiento. Esto podría ser objeto de estudio a profundidad a futuro, tratando de buscar los motivos que pudieron surgir para este aumento de más de 3000 arrestos de un trimestre a otro.

Luego, se realizó una agrupación de cantidad de arrestos por delito para lograr identificar cuáles crímenes fueron los más recurrentes dentro del dataframe.

En este caso, se realizó una pequeña transformación, mapeando los nombres de los delitos para realizar una traducción al español que facilite el entendimiento de la gráfica.

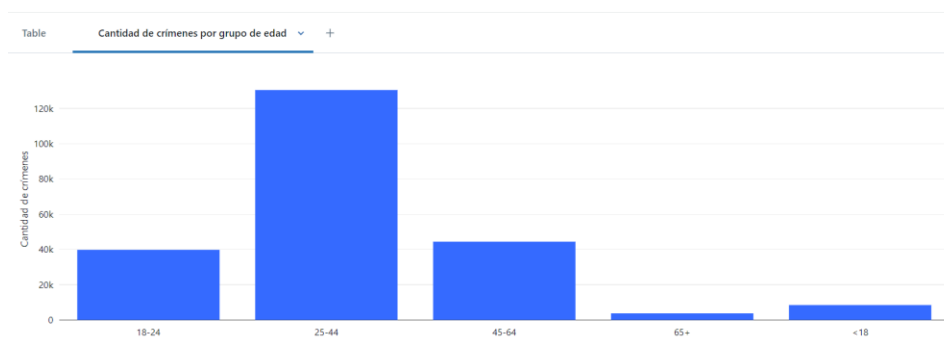


Estos resultados podrían ser objeto de un estudio más profundo para identificar qué frentes o qué estrategias se deben utilizar para hacer frente a estos crímenes que son más significativos, y que en este caso son graves, como el caso del asalto grado 3, el hurto menor y el asalto grave, que son los 3 crímenes con mayor tasa de aparición en el dataframe.

Después se realizó una tabla que permitiera analizar la cantidad de arrestos por trimestre por arrestos.

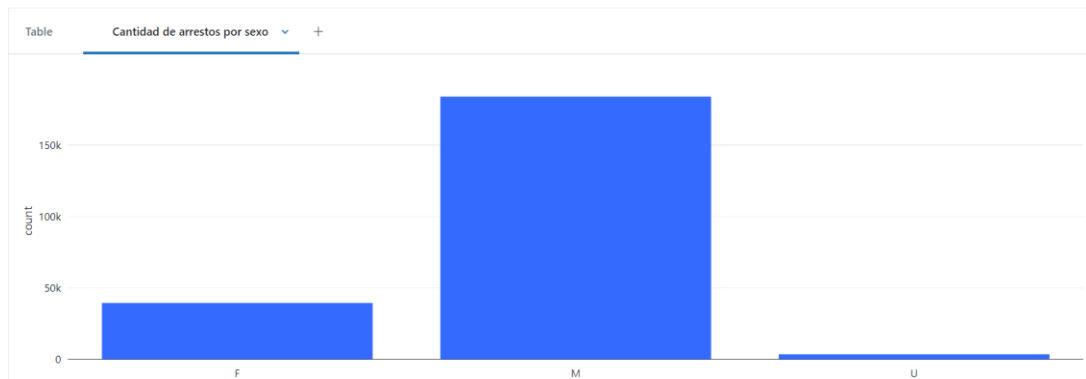
	OFNS_DESC	Year	Quarter	count
1	ASSAULT 3 & RELATED OFFENSES	2023	2	8730
2	ASSAULT 3 & RELATED OFFENSES	2023	4	8639
3	ASSAULT 3 & RELATED OFFENSES	2023	3	8581
4	ASSAULT 3 & RELATED OFFENSES	2023	1	7889
5	PETIT LARCENY	2023	2	6201
6	PETIT LARCENY	2023	3	5953
7	PETIT LARCENY	2023	1	5871
8	PETIT LARCENY	2023	4	5863
9	FELONY ASSAULT	2023	3	5554
10	FELONY ASSAULT	2023	2	5527
11	FELONY ASSAULT	2023	4	5090
12	FELONY ASSAULT	2023	1	4950

Posteriormente, se realizó un análisis de la cantidad de arrestos por grupo de edad.



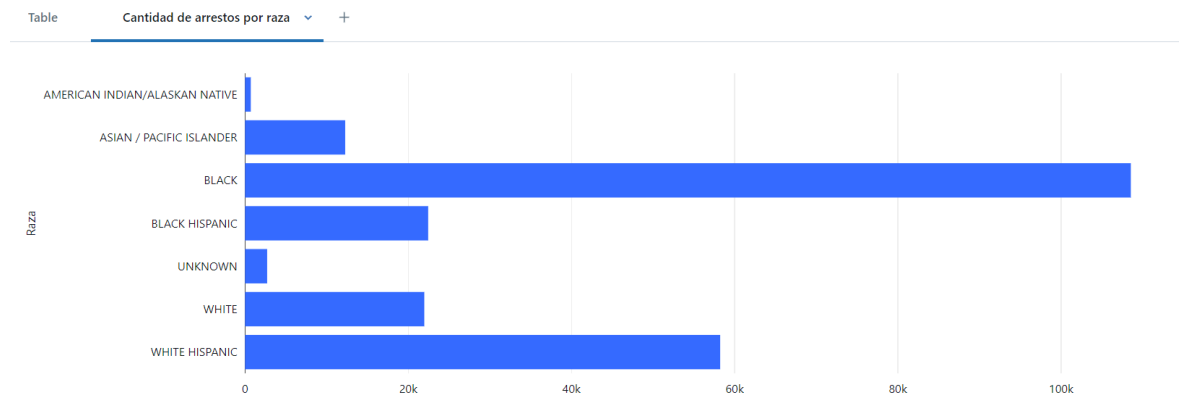
Esta gráfica es sumamente interesante, dado que muestra cómo las personas de mediana edad son aquellas que cometen más delitos. No obstante, es importante observar cómo el grupo de edad de 18-24 tiene 39,791 arrestos asociados, lo que podría ser importante para estudiar a futuro de cara a las causas de esa tasa de arrestos asociados, pudiendo buscar relaciones con factores clave como la educación y demás que puedan llevar a entender apropiadamente el porqué de esa cantidad de registros asociados a ese grupo de edad.

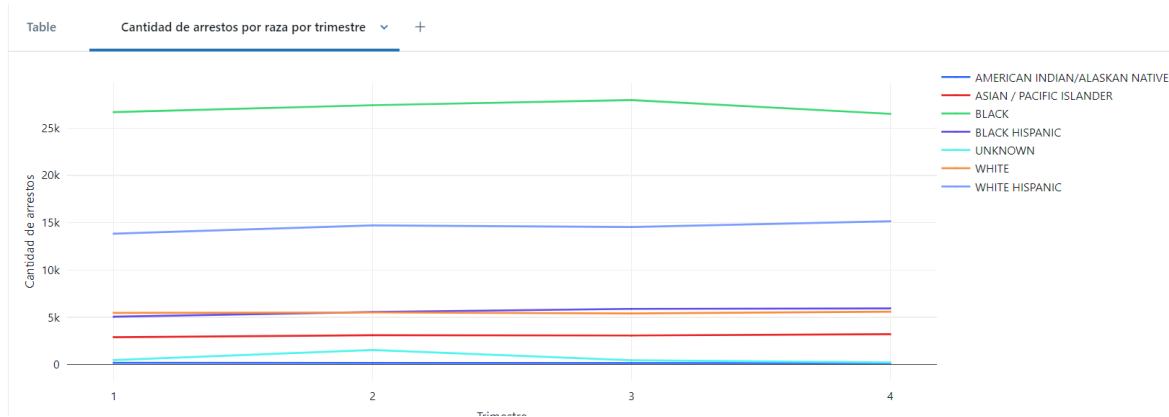
Aunado a esto, se realizó un análisis de la cantidad de arrestos por sexo.



Luego, se realizó un análisis de la cantidad de arrestos por raza, factor crítico en un país como Estados Unidos, donde es foco de polémica el aparente sesgo del cuerpo de policía hacia la raza de los ciudadanos.

Primero se realizó una agrupación por raza, que se repitió después agregando la variable temporal, nuevamente medida por trimestres.





Estos resultados muestran cómo la mayor cantidad de arrestos son hacia personas de raza negra. Ahora, este análisis debe realizarse con cuidado, siendo un posible objeto de estudio de cara a entender el porqué de todos estos arrestos hacia las personas de raza negra, siendo lo primero el tratar de saber si estos arrestos fueron realizados con fundamento y con una causa probable de por medio o con el hecho de haber sido atrapado en flagrancia. Asimismo, sería importante el entender el porqué de todos estos arrestos hacia la gente de raza negra más allá de si son justificados o no, dado que lo más probable es que, independientemente de la inquietud postulada previamente, las personas de raza negra sí tengan la mayor cantidad de delitos cometidos. En ese orden de ideas, sería importante buscar las causas de esto, ya sean asociadas a causas como la educación o el desempleo.

Reporte de calidad de datos

Se revisó cuántos valores faltantes o nulos se tenían para cada atributo, el resultado fue el siguiente:

```
2 df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

▶ (2) Spark Jobs

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|ARREST_KEY|ARREST_DATE|PD_CD|PD_DESC|KY_CD|OFNS_DESC|LAW_CODE|LAW_CAT_CD|ARREST_BORO|ARREST_PRECINCT|JURISDICTION_CODE|AGE_GROUP|PERP_SEX|PERP_RACE|X_COORD_CD|Y_COORD_CD|
|Latitude|Longitude|New Georeferenced Column|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|2|17|0|0|1599|0|0|0|0|0|0|0|0|0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Teniendo en cuenta esto, fue necesario utilizar técnicas para tratar los valores faltantes.

Primero, se buscó un patrón dentro de las variables directamente relacionadas en el dataframe para rellenar aquellos valores nulos de la variable *LAW_CAT_CD*, aquella con mayor cantidad de datos nulos y que representa el nivel del delito —categorizado en delito mayor o felony), delito menor o misdemeanor, e infracción o violation—.

```
1 #Se busca un patron dentro de las variables directamente relacionadas en el dataset para rellenar valores nulos de la variable "LAW_CAT_CD"
2
3 df.filter(isnan("LAW_CAT_CD") | col("LAW_CAT_CD").isNull()).select("PD_CD", "PD_DESC", "LAW_CODE", "LAW_CAT_CD").show()
```

```
+-----+-----+-----+
|PD_CD|      PD_DESC|  LAW_CODE|LAW_CAT_CD|
+-----+-----+-----+
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 15.0|FUGITIVE/OTHER JU...|FOA9000015|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 29.0|NYS PAROLE VIOLATION|FOA9000029|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 49.0|U.S. CODE UNCLASS...|FOA9000049|    NULL|
| 16.0|FUGITIVE/OTHER ST...|FOA9000016|    NULL|
```

Es aquí donde se observa que la mayoría de los registros con la variable *LAW_CAT_CD* en nulo tienen como *PD_DESC* o descripción del delito el valor **U.S. CODE UNCLASSIFIED**, lo que significa que el delito no está tipificado. De esta forma, para aquellos valores nulos en el atributo, se les asignará un nuevo valor NC, correspondiente a not classified.

```
1 df = df.withColumn("LAW_CAT_CD", when((isnan("LAW_CAT_CD") | col("LAW_CAT_CD").isNull()), "NC").otherwise(df["LAW_CAT_CD"]))
```

Los otros atributos que presentan valores nulos: *PD_CD* y *KY_CD*, tienen una tasa de nulidad extremadamente baja, por lo cual no es necesario hacer ninguna transformación y, en cambio, es mejor eliminar estos registros, teniendo en cuenta que los valores nulos no son una muestra representativa del dataframe:

```
df = df.na.drop(subset = ["PD_CD" , "KY_CD"])
```

Ahora, una vez realizado este procedimiento, se revisa nuevamente la cantidad de valores nulos en el dataframe, dando como resultado lo siguiente:

```
1 df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

▶ (2) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [ARREST_KEY: long, ARREST_DATE: string ... 17 more fields]

```
+-----+-----+-----+
|ARREST_KEY|ARREST_DATE|PD_CD|PD_DESC|KY_CD|OFNS_DESC|LAW_CODE|LAW_CAT_CD|ARREST_BORO|ARREST_PRECINCT|JURISDICTION_CODE|AGE_GROUP|PERP_SEX|PERP_RACE|X_COORD_CD|Y_COORD_CD|
|Latitude|Longitude|New Georeferenced Column|
+-----+-----+-----+
| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
```

Esto muestra cómo las técnicas aplicadas para la reducción y eliminación de registros nulos en el dataframe fueron exitosas.

Planteamiento de preguntas sobre los datos

Teniendo en cuenta todo esto, se plantearon las siguientes preguntas acerca de los datos, preguntas que serán respondidas en la segunda entrega del proyecto.

1. ¿Cuál es la tendencia anual de arrestos en Nueva York, y hay alguna temporada o mes específico donde los arrestos incrementan?
2. ¿Cómo se distribuyen los arrestos entre los diferentes delitos (delitos mayores, delitos menores, infracciones) y cómo ha evolucionado esta distribución en los últimos años?
3. ¿Cuáles son los tipos de delitos más comunes en cada barrio de Nueva York y cómo se compara esto con el promedio de la ciudad?
4. ¿Hay alguna correlación entre la edad o el sexo del perpetrador y el tipo de delito cometido?
5. ¿Cómo varían los arrestos entre los distintos precincts y cuáles presentan la mayor y menor cantidad de arrestos?
6. ¿Cuál es el impacto de las jurisdicciones no-NYPD en los arrestos y cómo se distribuyen estos arrestos por tipo de delito?
7. ¿Cuáles son las 5 zonas con mayor cantidad de arrestos en Nueva York?

Filtros, limpieza y transformación inicial

Analizando el dataframe, en primera instancia se encuentra un aparente error de digitación en la variable *LAW_CAT_CD*, donde se encuentran registros numéricos que no deberían existir, teniendo en cuenta los valores posibles de la columna, los cuáles se revisaron en un apartado previo. De esta forma, se deben buscar los registros con este error.

```
2 filtered_df = df.filter(df["LAW_CAT_CD"]=="9").select("PD_CD", "PD_DESC", "LAW_CODE", "LAW_CAT_CD")
3 display(filtered_df)
```

▶ (3) Spark Jobs

▶ filtered_df: pyspark.sql.dataframe.DataFrame = [PD_CD: double, PD_DESC: string ... 2 more fields]

	PD_CD	PD_DESC	LAW_CODE	LAW_CAT_CD
1	849	NY STATE LAWS, UNCLASSIFIED VIO	CPL5700600	9
2	849	NY STATE LAWS, UNCLASSIFIED VIO	CPL5700600	9
3	849	NY STATE LAWS, UNCLASSIFIED VIO	CPL5700600	9
4	849	NY STATE LAWS, UNCLASSIFIED VIO	CPL5700600	9
5	849	NY STATE LAWS, UNCLASSIFIED VIO	CPL5700600	9

Asimismo, se observa la cantidad de registros con error en esta columna:

```
1 #Se mira la cantidad de registros con el error
2 print("La cantidad de registros con error en la columna LAW_CAT_CD es: " +str(filtered_df.count()))
```

► (2) Spark Jobs

La cantidad de registros con error en la columna LAW_CAT_CD es: 611

Teniendo en cuenta el contexto y la documentación de los datos, se decide asignar el valor asociado a la descripción "UV:unclassified violence" ya que no está dentro de la misma serie de leyes que NC: not classified, y no se puede realizar una asociación directa de una con otra.

```
df = df.withColumn("LAW_CAT_CD", when(df["LAW_CAT_CD"] == "9", "UV").otherwise(df["LAW_CAT_CD"]))
```

Finalmente, se revisan los resultados, donde se evidencia que la transformación fue efectiva para cada uno de los 611 registros con el error encontrado.

```
1 display(df.filter(df["LAW_CAT_CD"]=="UV").select("PD_CD","PD_DESC","LAW_CODE","LAW_CAT_CD"))
```

► (3) Spark Jobs

Table ▾ +

	PD_CD	PD_DESC	LAW_CODE	LAW_CAT_CD
1	849	NY STATE LAWS,UNC CPL5700600	UV	
2	849	NY STATE LAWS,UNC CPL5700600	UV	
3	849	NY STATE LAWS,UNC CPL5700600	UV	
4	849	NY STATE LAWS,UNC CPL5700600	UV	
5	849	NY STATE LAWS,UNC CPL5700600	UV	

611 rows | 3.76 seconds runtime Refreshed now

Ahora, teniendo en cuenta que todos los datos del dataframe del 2023, se extraerá el mes para observar los arrestos en diferentes periodos del año, siendo esto mucho más específico que los resultados revisados de manera trimestral, primero transformando el atributo *ARREST_DATE* al formato correspondiente.

```
1 df = df.withColumn("ARREST_DATE", to_date(col("ARREST_DATE"), "MM/dd/yyyy"))
2
3 df = df.withColumn("Month", month(col("ARREST_DATE")))
4 display(df.select("ARREST_DATE","OFNS_DESC","Month"))
```

► (2) Spark Jobs

► df: pyspark.sql.dataframe.DataFrame = [ARREST_KEY: long, ARREST_DATE: date ... 18 more fields]

Table ▾ +

	ARREST_DATE	OFNS_DESC	Month
1	2023-01-03	ROBBERY	1
2	2023-01-03	FELONY ASSAULT	1
3	2023-01-04	ROBBERY	1
4	2023-01-04	FELONY ASSAULT	1
5	2023-01-05	BURGLARY	1



Finalmente, se evidencia la necesidad de realizar una imputación a variables categóricas como: *LAW_CAT_CD*, *ARREST_BORO*, *ARREST_PRECINT*, *AGE_GROUP*, *PERP_SEX* y *PERP_RACE*.

```
3 from pyspark.ml.feature import StringIndexer
4 from pyspark.ml import Pipeline
5
6
7 columnas = ["LAW_CAT_CD", "ARREST_BORO", "ARREST_PRECINT", "AGE_GROUP", "PERP_SEX", "PERP_RACE"]
8 for columna in columnas:
9     indexer = StringIndexer(inputCol=columna, outputCol=columna + "Ind")
10    model = indexer.fit(df)
11    df = model.transform(df)
12    print(f"Mapping de {columna} a indices: {model.labels}")
13
```

► (12) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [ARREST_KEY: long, ARREST_DATE: string ... 24 more fields]

Mapping de LAW_CAT_CD a índices: ['M', 'F', 'NC', 'V', 'UV', 'I']

Mapping de ARREST_BORO a índices: ['K', 'B', 'M', 'Q', 'S']

Mapping de ARREST_PRECINT a índices: ['14', '75', '44', '40', '103', '46', '52', '43', '73', '120', '113', '110', '47', '25', '109', '42', '48', '105', '67', '18', '114', '115', '41', '60', '79', '84', '102', '70', '5', '72', '45', '77', '106', '13', '34', '83', '49', '121', '90', '71', '32', '23', '1', '63', '107', '28', '62', '6', '68', '108', '33', '7', '9', '104', '61', '81', '19', '10', '24', '69', '101', '78', '30', '122', '112', '50', '88', '66', '26', '94', '76', '20', '100', '123', '17', '111', '22']

Mapping de AGE_GROUP a índices: ['25-44', '45-64', '18-24', 'c18', '65+']

Mapping de PERP_SEX a índices: ['M', 'F', 'U']

Mapping de PERP_RACE a índices: ['BLACK', 'WHITE HISPANIC', 'BLACK HISPANIC', 'WHITE', 'ASIAN / PACIFIC ISLANDER', 'UNKNOWN', 'AMERICAN INDIAN/ALASKAN NATIVE']

Bono 1

Para el bono número 1 se utilizaron las librerías Selenium, Pandas, Geopandas, Matplotlib y Mapclassify.

Primero se hizo la extracción de los datos a través de selenium y se guardaron los datos en un dataframe de Pandas.

```
[ ] ##En esta celda se realiza el proceso de recolección de la API a través de Selenium y del Xpath de la página web
headers = ["Borough", "region", "Males", "Females", "Total Population"]
data = []
for j in range(1, 56):
    union = []
    th = []
    td = []
    if j <= 10:
        th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[1]").text)
        if j >= 11 and j < 29:
            th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[1]").text)
        if j >= 29 and j < 39:
            th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[1]").text)
        if j >= 39 and j < 53:
            th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[1]").text)
        if j >= 53:
            th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[1]").text)
    for i in range(1, 4):
        if i != 3:
            if i != 1:
                th.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/th[{i}]").text)
                td.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/td[{i}]").text)
            else:
                td.append(driver.find_element(By.XPATH, f"/html/body/div[2]/div/div[2]/table/tbody/tr[{j}]/td[{i}]").text)
        union = th + td
        data.append(union)
dfBono = pd.DataFrame(data, columns=headers)
```

[] dfBono.head()

	Borough	region	Males	Females	Total Population
0	Bronx	Riverdale, Fieldston & Kingsbridge	52,133	61,937	114,070
1	Bronx	Wakefield, Williamsbridge & Woodlawn	65,087	77,848	142,935
2	Bronx	Co-op City, Pelham Bay & Schuylerville	55,615	65,929	121,544
3	Bronx	Pelham Parkway, Morris Park & Laconia	61,233	67,896	129,130
4	Bronx	Belmont, Crotona Park East & East Tremont	75,963	87,740	163,704

Luego, se actualizaron los nombres del dataframe que se tiene del web scrapping para que los nombres de los distritos de Nueva York sean iguales tanto en el df como en GeoJSON que servirán para la realización de las gráficas.

```
nombre_mapper = {
    'Richmond (Staten Island)': 'Staten Island',
    'New York (Manhattan)': 'Manhattan',
    'Kings (Brooklyn)': 'Brooklyn'
}

dfBono['Borough'] = dfBono['Borough'].replace(nombre_mapper)
```

Debido al formato de la tabla de la página web se tuvo que cambiar los valores de los atributos que son numéricos ya que al contener "," se identifica como un string y se tuvo que quitar para poder operar con ellos.

```
dfBono['Males'] = dfBono['Males'].str.replace(',', '').astype(float)
dfBono['Females'] = dfBono['Females'].str.replace(',', '').astype(float)

# Sumar las columnas 'Males' y 'Females' para obtener 'Total Population'
dfBono['Total Population'] = dfBono['Males'] + dfBono['Females']

# Ahora realiza la suma de población por 'Borough'
df_suma_poblacion = dfBono.groupby('Borough')['Total Population'].sum().reset_index()
```

Luego, se importaron las librerías Geopandas, Matplotlib y mapclassify, después se cargó un GeoJSON.

```
##Se usa el enlace al GeoJSON que se encuentra en el repositorio de GitHub y de ahí extraer las formas geométricas de los distritos de nueva york
url = 'https://raw.githubusercontent.com/TommyDS2005/Proyecto-Procesamiento-de-Datos/main/new-york-city-boroughs.geojson'
gdf = gpd.read_file(url)
```

Finalmente, se Juntan en un solo conjunto de datos los atributos de df_suma_poblacion y de gdf para que se tenga en una sola tabla los atributos de población y de forma geométrica, de forma semejante a una operación de unión, finalmente se grafica el GeoDataFrame.

```
gdf_merge = gdf.merge(df_suma_poblacion, left_on='name', right_on='Borough')

gdf_merge['Total Population'] = pd.to_numeric(gdf_merge['Total Population'], errors='coerce')

# Crear etiquetas con el nombre del distrito y la población total
gdf_merge['label'] = gdf_merge.apply(lambda x: f'{x["Borough"]}\n{int(x["Total Population"])}', axis=1)

# Graficar el GeoDataFrame con la columna 'Total Population'
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
gdf_merge.plot(column='Total Population', cmap='OrRd', linewidth=0.8, edgecolor='0.8', ax=ax)

# Añadir las etiquetas a la figura
for idx, row in gdf_merge.iterrows():
    # El color del texto depende de la luminosidad del color del fondo
    # Utilizamos una simple estimación de la luminosidad para decidir el color del texto
    # basado en la población total (esto podría no ser exacto y es una simplificación)
    if row['Total Population'] > 2200000:
        text_color = 'white'
    else:
        text_color = 'black'

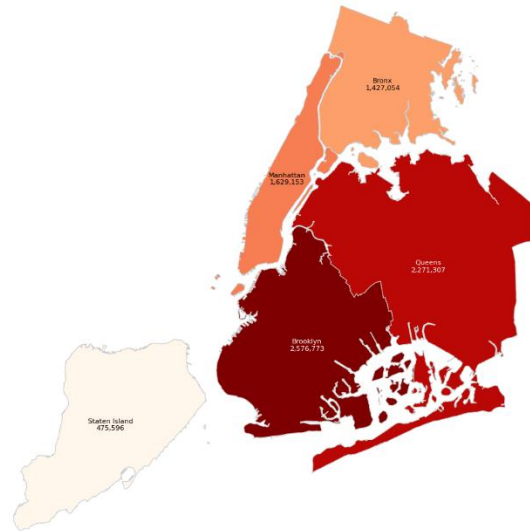
    plt.annotate(text=row['label'], xy=(row['geometry'].centroid.x, row['geometry'].centroid.y),
                ha='center', va='center', fontsize=8, color = text_color)

# Ajustar la posición del título
ax.set_title('Population Heat Map of New York City Boroughs', fontdict={'fontsize': 20}, loc='center')

# Desactivar los ejes
ax.set_axis_off()

# Ajustar el layout y mostrar la figura
plt.tight_layout()
plt.show()
```


Population Heat Map of New York City Boroughs



Bono 2

Para el segundo bono, se utilizaron las librerías Pandas y Folium. Aunado a esto, se realizó un llamado a la API de OpenWeather.

```

### How to make api call
import requests
import json

def make_api_call(latitude,longitude,apikey):
    url = f"https://api.openweathermap.org/data/2.5/weather?lat={latitude}&lon={longitude}&appid={apikey}"
    try:
        response = requests.get(url)
    except requests.exceptions.RequestException as e:
        print(f"Error recibiendo los datos: {e}")
        return None
    weather_data = response.json()
    return weather_data

```

```

apikey = '8b47fab30f52065a6468746c614f997f'
lat = 40.781722
lon = -73.973056

wheatherData = make_api_call(lat,lon,apikey)

###Imprimir JSON de informacion
print(wheatherData)

```

Luego, se importa Pandas y se carga el conjunto de datos de arrestos ya utilizado.

```

import pandas as pd

df = pd.read_csv("C:/Users/jusec/OneDrive/Documentos/GitHub/Proyecto-Procesamiento-de-Datos/NYPD_Arrest_Data__Year_to_Date__20240322.csv")

```

Se importa la librería Folium y se genera un mapa de calor interactivo acorde a la cantidad de arrestos por ubicación geográfica según los datos del conjunto de arrestos.

```
import folium as fl
from folium.plugins import HeatMap

m = fl.Map(location = [40.738154, -73.921915])

heatmap = HeatMap(list(zip(df['Latitude'], df['Longitude'])),
                  min_opacity=0.2,
                  radius=50, blur=50,
                  max_zoom=1)

heatmap.add_to(m)

m
```



Luego, se agrupa la cantidad de arrestos por barrios y se sacan promedios.

```
###Agrupar por barrios y sacar promedios

latdic = dict(df.groupby(['ARREST_BORO'])['Latitude'].mean())

londic = dict(df.groupby(['ARREST_BORO'])['Longitude'].mean())

print(latdic, londic, sep='\n')

{'B': 40.842336551776924, 'K': 40.65929330809888, 'M': 40.771409056657504, 'Q': 40.71504591174004, 'S': 40.61131624332839}
{'B': -73.8909266020233, 'K': -73.94753174177487, 'M': -73.96970222125222, 'Q': -73.830016519732, 'S': -74.117971159588}
```

Tras esto, crea un diccionario de los barrios donde se han realizado arrestos que aparezcan en el conjunto de datos y se grafica un mapa con marcadores correspondientes a cada barrio y los datos del clima actualizados.

```
###Crear los marcadores en el mapa
latitude = 40.7382
longitude = -73.9219

boro_dic = {
    'B': 'Bronx',
    'S': 'Satatn Island',
    'K': 'Brooklyn',
    'M': 'Manhattan',
    'Q': 'Queens'
}

m = fl.Map(location = [latitude, longitude], zoom_start=10)

for lat, lon in zip(latdic, londic):
    # Datos de ejemplo para un punto en el tiempo
    forecast_example = make_api_call(latdic[lat], londic[lon], apikey)

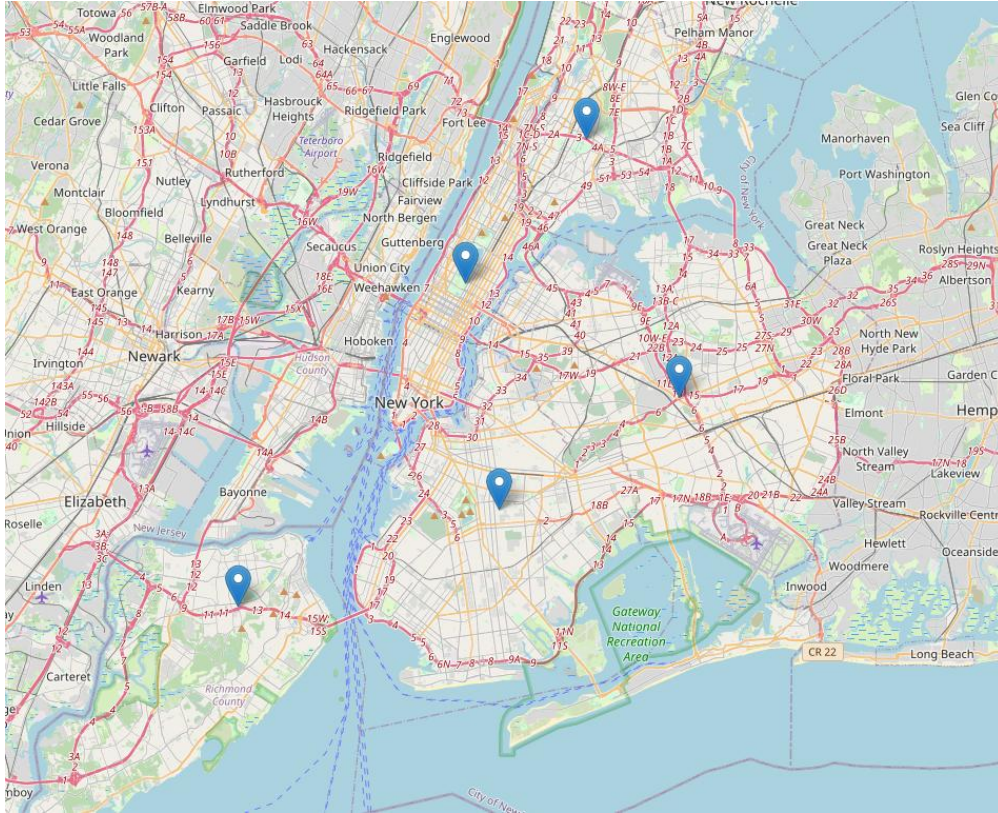
    # Formato del texto del marcador
    popup_text = f"""
    Fecha y hora: {forecast_example['dt']}<br>
    Temperatura: {forecast_example['main']['temp']} K<br>
    Presión: {forecast_example['main']['pressure']} hPa<br>
    Humedad: {forecast_example['main']['humidity']}%<br>
    Clima: {forecast_example['weather'][0]['main']} ({forecast_example['weather'][0]['description']})<br>
    Velocidad del viento: {forecast_example['wind']['speed']} m/s<br>
    Barrio: {boro_dic[lat]}
    """

    # Agregar un marcador al mapa
    fl.Marker(
        [latdic[lat], londic[lon]],
        popup=fl.Popup(html=popup_text, max_width=250)
    ).add_to(m)

# Muestra el mapa
m
```

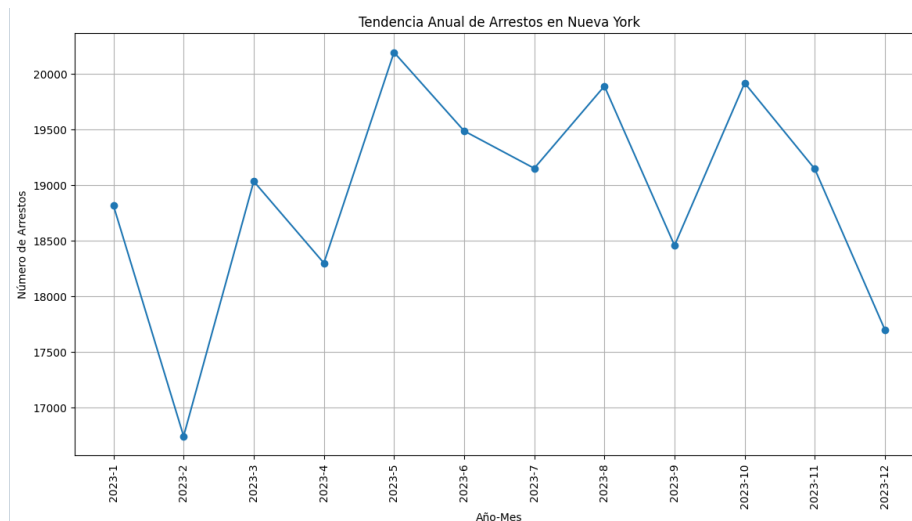


Finalmente, se muestra el resultado del mapa.



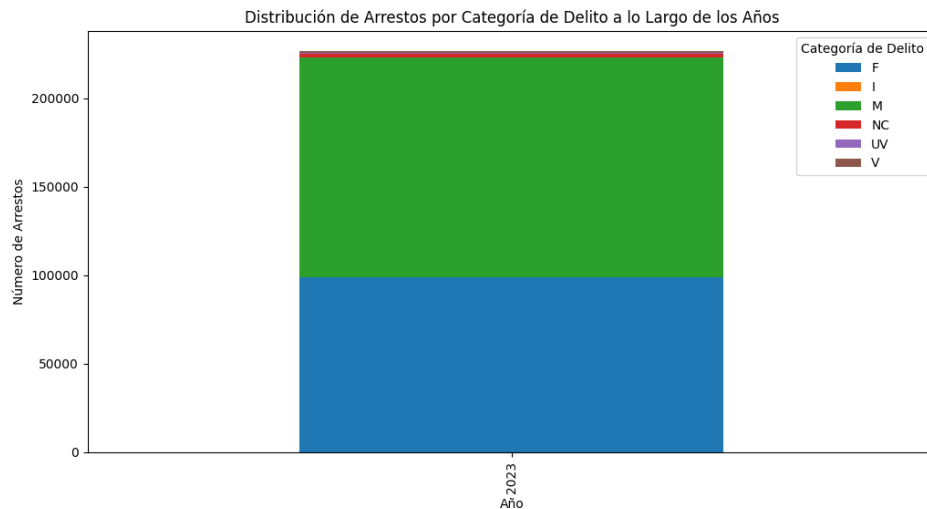
Solución a las preguntas planteadas

1. ¿Cuál es la tendencia anual de arrestos en Nueva York, y hay alguna temporada o mes específico donde los arrestos incrementan?



En este gráfico se puede apreciar cómo la cantidad de arrestos en Nueva York varía mucho según la época del año, no obstante, podemos remarcar el mes de febrero como el que tiene la menor cantidad de arrestos y el mes de mayo como el que presenta la mayor cantidad. En ese orden de ideas, no es posible hacer ninguna conclusión respecto a la temporalidad teniendo en cuenta lo mucho que varían los datos. No obstante, de cara al futuro sería interesante y apropiado buscar información de otros años para revisar si se mantiene esta tendencia.

2. **¿Cómo se distribuyen los arrestos entre los diferentes delitos (delitos mayores, delitos menores, infracciones) y cómo ha evolucionado esta distribución en los últimos años?**



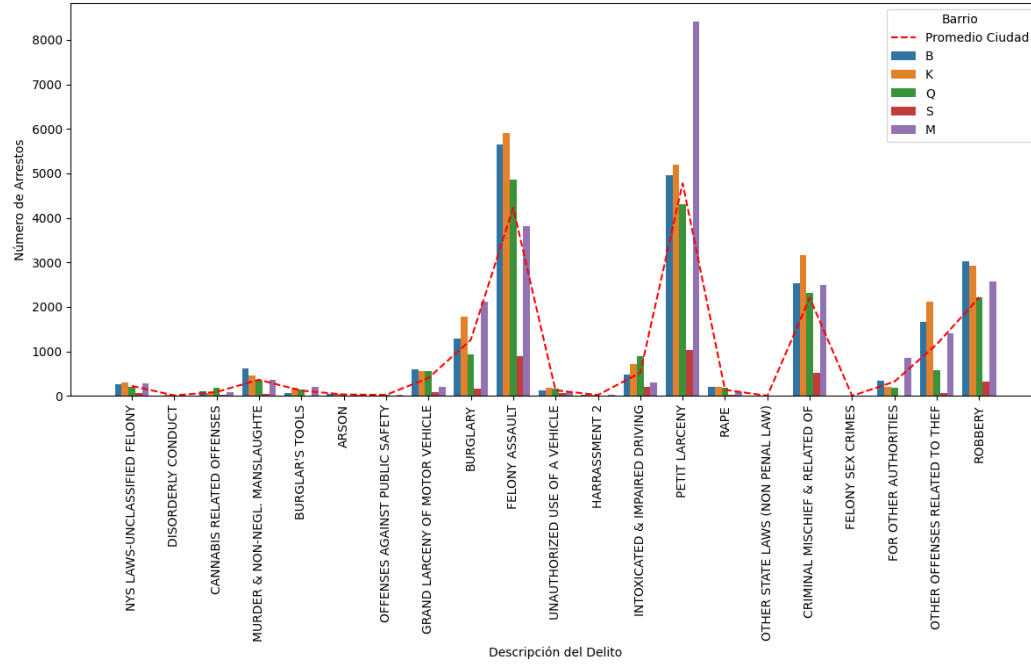
En este caso observamos cómo la mayor cantidad de arrestos se la llevan M = Misdemeanor (Delito menor) y F = Felony (Delito mayor) con una diferencia abismal, mostrando un contraste muy claro, donde sería importante revisar más específicamente cada tipo de delito.

3. **¿Cuáles son los tipos de delitos más comunes en cada barrio de Nueva York y cómo se compara esto con el promedio de la ciudad?**

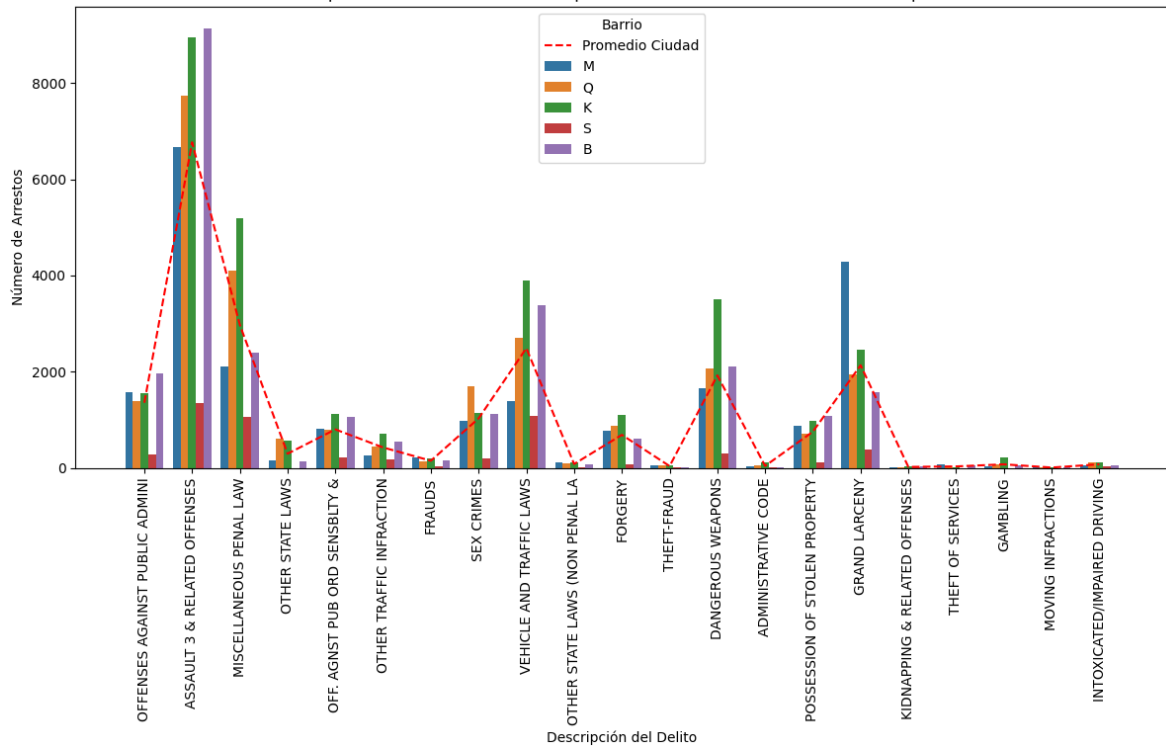
Las gráficas que se presentan a continuación quizá son las más importantes, estas sirven para marcar una pauta y, de cara al ente competente, podría ser un punto de partida para implementar estrategias sectorizadas en Nueva York que busquen reducir las tasas de criminalidad, teniendo en cuenta que cada sector necesita una estrategia diferente, teniendo en cuenta que cada zona tiene un delito mayor diferente.

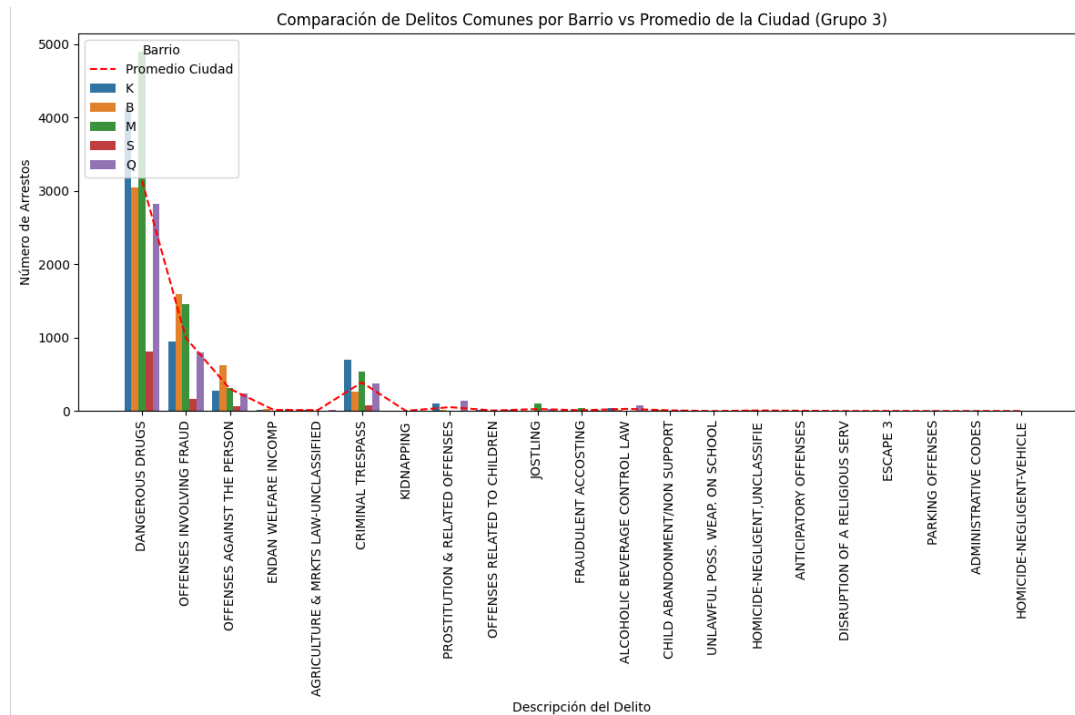


Comparación de Delitos Comunes por Barrio vs Promedio de la Ciudad (Grupo 1)

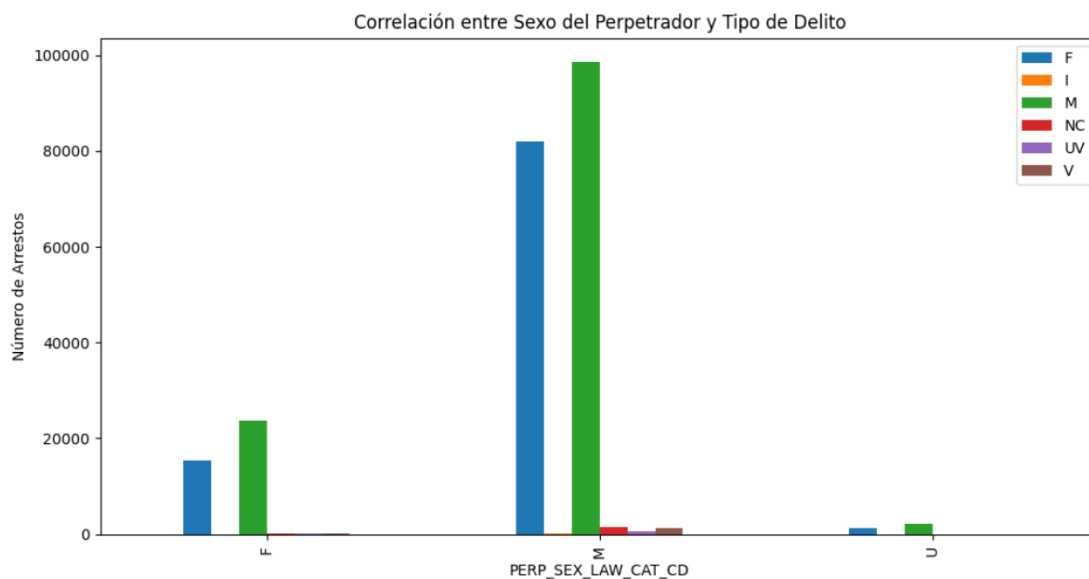


Comparación de Delitos Comunes por Barrio vs Promedio de la Ciudad (Grupo 2)

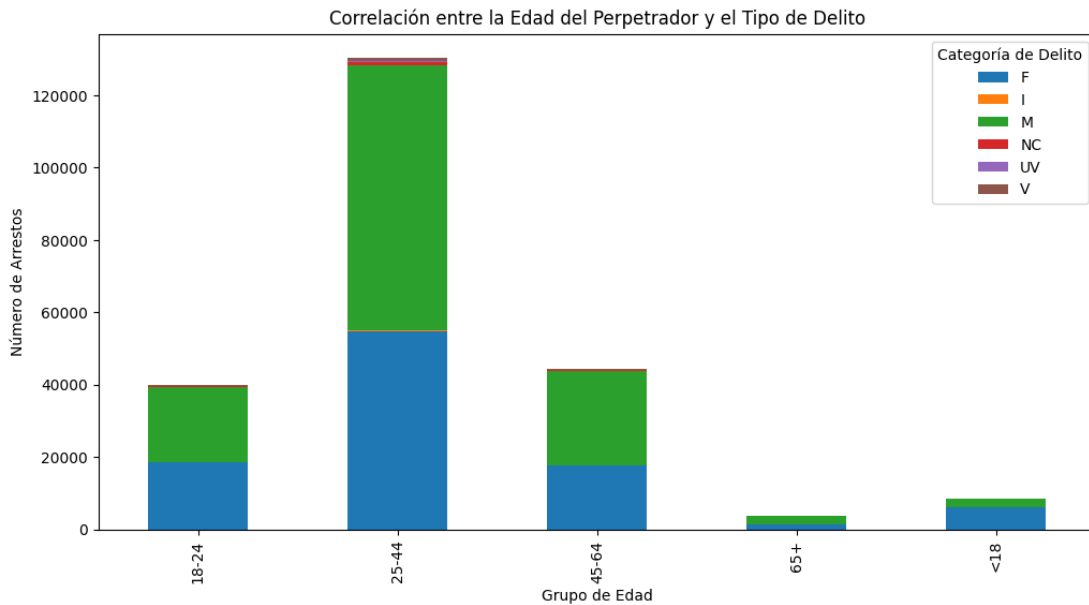




4. ¿Hay alguna correlación entre la edad o el sexo del perpetrador y el tipo de delito cometido?

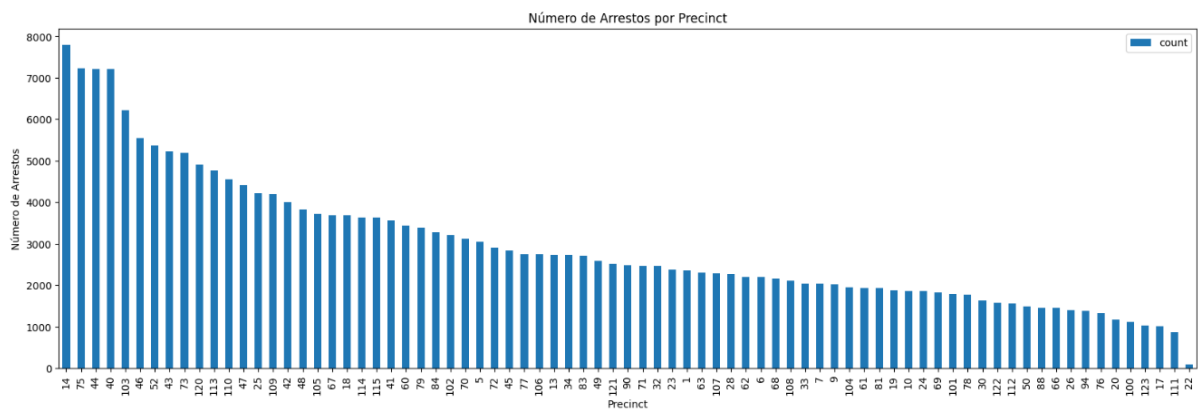


En este caso vemos que no existe correlación alguna, teniendo en cuenta que los géneros tipificados en el conjunto de datos siguen la misma tendencia vista anteriormente, por lo cual es muy complejo observar si existe correlación. No obstante, sería importante de cara al futuro revisar estos datos, pero por cada delito específicamente, buscando encontrar una correlación.



Lo mismo sucede con la edad, que mantiene la tendencia vista y que, a falta de un análisis específico de cada delito, no muestra correlación o tendencia alguna.

5. ¿Cómo varían los arrestos entre los distintos precincts y cuáles presentan la mayor y menor cantidad de arrestos?



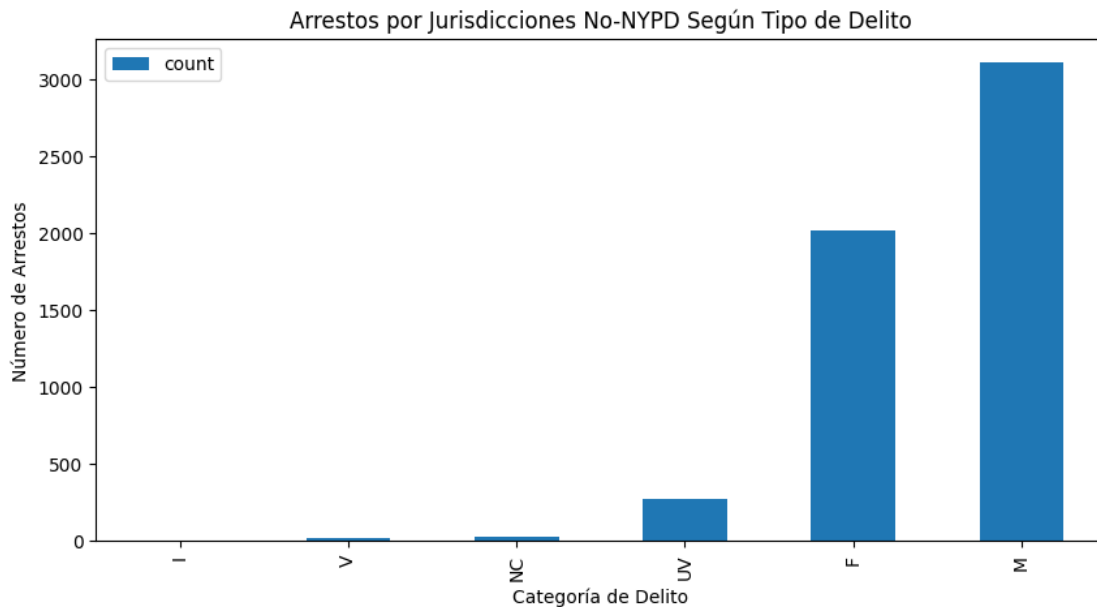
Esta gráfica muestra claramente los 5 precincts con más arrestos y los 5 con la menor cantidad. Esta gráfica es bastante específica y con un análisis de cada delito en estos 10 barrios especiales se podría gestionar mejor las estrategias para la reducción de delitos, buscando así evitar un gasto innecesario de recursos en una zona o la falta de recursos en otras. A continuación, se muestran cuáles son estos precincts:



Precincts con mayor cantidad de arrestos	
Precinct	Ubicación
4	Midtown South - Manhattan
75	Brooklyn
44	Bronx
40	Bronx
103	Queens

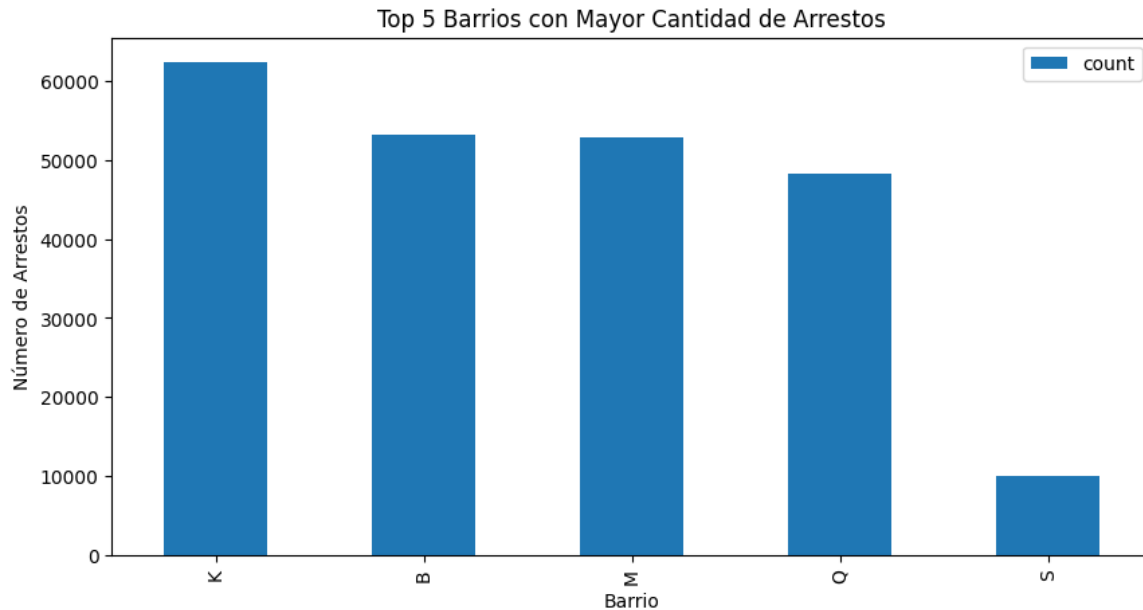
Precincts con menor cantidad de arrestos	
Precinct	Ubicación
22	Central Park - Manhattan
111	Queens
17	Manhattan
123	Staten Island
100	Queens

6. ¿Cuál es el impacto de las jurisdicciones no-NYPD en los arrestos y cómo se distribuyen estos arrestos por tipo de delito?



En este caso, vemos cómo se mantiene la tendencia vista, con el caso particular de que Unclassified Violence —que es el tipo de delito que fue añadido en la transformación— aparece con una tasa alta respecto a las demás categorías, lo cuál sería interesante de analizar de cara a un futuro.

7. ¿Cuáles son las 5 zonas con mayor cantidad de arrestos en Nueva York?



En este caso, vemos otra gráfica que aporta al objetivo de buscar información visual para lograr distribuir los esfuerzos de las fuerzas públicas de Nueva York. En este caso vemos que las zonas con mayor cantidad de arrestos son, en orden: K (Brooklyn), B (Bronx), M (Manhattan), Q (Queens), S (Staten Island).

Selección de técnicas de aprendizaje de máquina

Se seleccionaron 2 modelos uno de aprendizaje supervisado y uno de aprendizaje no supervisado, para este caso un problema de clasificación, se hizo uso de una Regresión Logística, junto con el método K-Means, con el propósito inicial de buscar clasificaciones múltiples dentro de la variable LAW_CAT_CD, tal como se explica en cada modelo a continuación.

Razón para usar Regresión Logística:

- **Clasificación:** La regresión logística es un modelo de clasificación muy utilizado cuando se tiene una variable dependiente categórica. En este caso, el objetivo es predecir LAW_CAT_CDInd (una categoría) basada en varias características (por ejemplo, ARREST_BOROInd, AGE_GROUPInd, etc.).
- **Interpretabilidad:** La regresión logística permite interpretar fácilmente las probabilidades de pertenencia a cada clase, lo cual es útil para entender las relaciones entre las características y la variable objetivo.

- **Eficiencia:** Es un modelo eficiente y rápido de entrenar, especialmente adecuado para grandes conjuntos de datos.

Modelo No Supervisado: K-Means

Razón para usar K-Means:

- **Clustering:** K-Means es uno de los algoritmos de clustering más comunes y es adecuado para agrupar datos en clusters basados en la similitud. En este caso, agrupa las ubicaciones de arrestos (Latitude y Longitude) en clusters.
- **Simplicidad:** Es fácil de entender e implementar. También es rápido y escalable para grandes conjuntos de datos.
- **Exploración de Datos:** Ayuda a identificar patrones o estructuras en los datos que no son inmediatamente obvios, proporcionando una comprensión más profunda de la distribución geográfica de los arrestos.

Aplicación de los modelos de Machine Learning

Aplicación Regresión Logisitica:

- **Entrenamiento y Prueba:** Se dividieron los datos en un 60% para entrenamiento y un 40% para prueba. Esto es fundamental para evaluar la capacidad del modelo para generalizar a nuevos datos.
- **VectorAssembler:** Se utilizo VectorAssembler para combinar múltiples columnas de características en un único vector, que es el formato requerido por muchos algoritmos de machine learning en PySpark.
- **Pipeline:** Se configuró un pipeline que facilita el flujo de trabajo al combinar la transformación de datos y el entrenamiento del modelo en una única estructura.
- **Evaluación:** Se utilizó MulticlassClassificationEvaluator para evaluar la precisión y el F1 Score del modelo, asegurando que tienes métricas relevantes para juzgar el rendimiento del modelo.



```
##70% Entrenamiento 30% Testeo
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

# Asegúrate de que solo se utilizan columnas numéricas en el VectorAssembler
assembler = VectorAssembler(
    inputCols=["LAW_CAT_CDInd", "ARREST_BOROInd", "ARREST_PRECINCTInd", "AGE_GROUPInd", "PERP_SEXInd", "PERP_RACEInd", "Latitude",
               "Longitude", "Month"],
    outputCol="features")

# Configura la regresión logística
lr = LogisticRegression(featuresCol="features", labelCol="LAW_CAT_CDInd", maxIter=10)

# Configura el pipeline
pipeline = Pipeline(stages=[assembler, lr])

# Dividir los datos
train_data, test_data = df.randomSplit([0.7, 0.3], seed=42)

# Entrenar el modelo
model = pipeline.fit(train_data)

# Predecir
predictions = model.transform(test_data)
predictions.select("features", "LAW_CAT_CDInd", "prediction").show(5)
```

```
# Evaluador para la precisión
evaluator_accuracy = MulticlassClassificationEvaluator(labelCol="LAW_CAT_CDInd", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator_accuracy.evaluate(predictions)
print(f"Precisión: {accuracy}")

# Evaluador para el F1 Score
evaluator_f1 = MulticlassClassificationEvaluator(labelCol="LAW_CAT_CDInd", predictionCol="prediction", metricName="f1")
f1_score = evaluator_f1.evaluate(predictions)
print(f"F1 Score: {f1_score}")

# Matriz de confusión
confusion_matrix = predictions.groupBy("LAW_CAT_CDInd").pivot("prediction").count() # Asegúrate de ajustar los valores de pivot según las
clases que tengas
confusion_matrix.show()
```

```
Precisión: 0.9917359024217783
F1 Score: 0.9895987049401818
+-----+-----+-----+-----+
|LAW_CAT_CDInd| 0.0| 1.0| 2.0| 3.0| 4.0|
+-----+-----+-----+-----+
|              0.0|37360| NULL| NULL| NULL| NULL|
|              1.0| NULL|29409| NULL| NULL| NULL|
|              4.0| NULL| NULL| 1| 5| 161|
|              3.0| NULL| NULL| 54| 302| 50|
|              2.0| NULL| 367| 91| 9| 8|
|              5.0| NULL| NULL| 6| 46| 15|
+-----+-----+-----+-----+
```

Aplicación K-Means:

- VectorAssembler: De nuevo, se utilizo VectorAssembler para combinar las columnas Latitude y Longitude en un solo vector de características.
- Pipeline: se configuro un pipeline que incluye la transformación de características y el algoritmo de clustering, haciendo que el proceso sea modular y fácil de reproducir.



- Visualización: Al transformar los datos y visualizar los resultados con matplotlib, se puede interpretar visualmente los clusters y obtener insights sobre la distribución espacial de los arrestos en Nueva York.

```
###K = 3 Seed = 4321
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler

# Asamblea de características para el clustering
assembler = VectorAssembler(
    inputCols=["Latitude", "Longitude"],
    outputCol="features")

# Configurar K-Means
kmeans = KMeans().setK(3).setSeed(4321).setFeaturesCol("features")

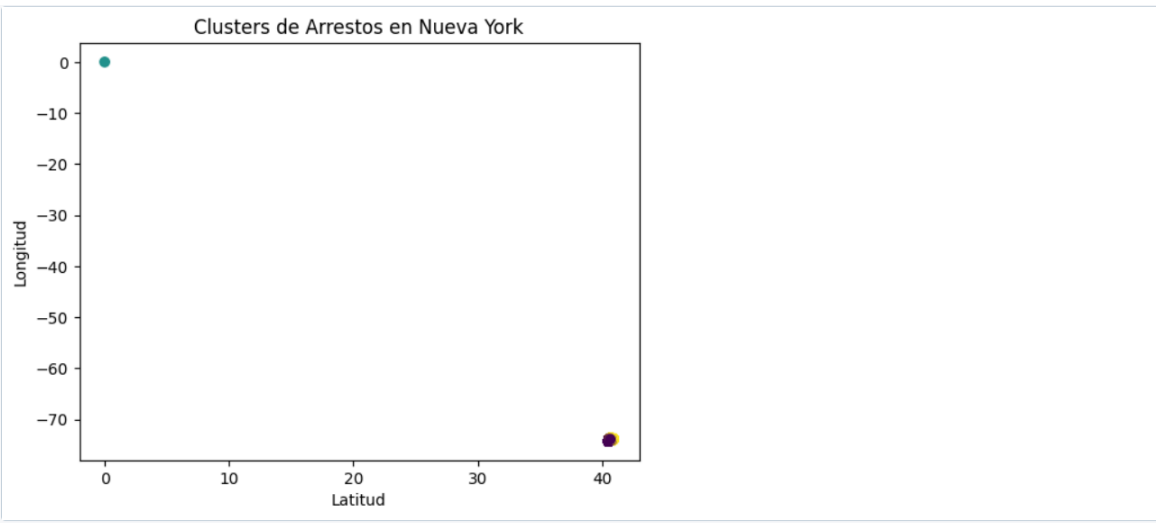
# Crear el Pipeline
pipeline = Pipeline(stages=[assembler, kmeans])

# Ajustar el modelo
model = pipeline.fit(df)

# Transformar los datos
clusters = model.transform(df)

# Visualizar los resultados del clustering
import matplotlib.pyplot as plt
import pandas as pd

pandas_df = clusters.toPandas()
plt.scatter(pandas_df['Latitude'], pandas_df['Longitude'], c=pandas_df['prediction'])
plt.title('Clusters de Arrestos en Nueva York')
plt.xlabel('Latitud')
plt.ylabel('Longitud')
plt.show()
```



```
from pyspark.ml.evaluation import ClusteringEvaluator

# Instanciar un evaluador de clustering
evaluator = ClusteringEvaluator()

# Evaluar el modelo y calcular el Silhouette score
silhouette = evaluator.evaluate(clusters)
print(f"Silhouette con distancia euclidiana cuadrada: {silhouette}")

# Mostrar los centros de los clusters
centers = model.stages[-1].clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

```
Silhouette con distancia euclidiana cuadrada: 0.5721958052192962
Cluster Centers:
[ 40.67838345 -73.97274938]
[0. 0.]
[ 40.79177456 -73.87722044]
```

En términos generales ninguno de los modelos preseleccionados establece métricas fiables, por lo tanto, se debería evaluar la opción de establecer una nueva técnica de aprendizaje de maquina eficaz para solventar el ya fijado problema de clasificación.

Conclusiones

- En este caso, en los primeros y últimos meses del año se presentan la menor cantidad de delitos. Sería pertinente revisar datos de otros años para corroborar si existe alguna tendencia que permita dar paso a un análisis extendido.
- La cantidad de delitos menores es mayor que la cantidad de delitos mayores, sin embargo, es preocupante la alta cantidad de ofensas mayores que se presentan. Esto debe ser revisado por cada delito específicamente y, conforme a eso se debe actuar con diversas políticas y regulaciones.
- Si bien es cierto que la distribución de los arrestos realizados fuera de la jurisdicción de la NYPD es similar, se presenta un aumento de los arrestos para UV: Unclassified Violence.
- Es evidente que cada sector de Nueva York necesita una estrategia diferente para bajar la criminalidad, teniendo en cuenta que cada sector tiene un crimen al cual le debe prestar más atención

Bono

A continuación, se detalla la construcción de la Red Neuronal solicitada:

```

22/5/2024 (9 s) 66
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer

# Si tienes columnas categóricas, necesitas convertirlas a numéricas
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index").fit(df) for column in ["PERP_SEX", "PERP_RACE", "ARREST_BORO", "LAW_CAT_CD"]]

# Aplicar los indexadores y añadir una columna de características
pipeline = Pipeline(stages=indexers)
df_r = pipeline.fit(df).transform(df)

assembler = VectorAssembler(inputCols=["PERP_SEX_index", "PERP_RACE_index", "ARREST_BORO_index", "Latitude", "Longitude"], outputCol="features")
df_r = assembler.transform(df_r)

# Asegúrate de que la columna de etiquetas es numérica y está indexada si es categórica
labelIndexer = StringIndexer(inputCol="LAW_CAT_CD", outputCol="label").fit(df_r)
df_r = labelIndexer.transform(df_r)

from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Dividir los datos en conjuntos de entrenamiento y prueba
train_data, test_data = df_r.randomSplit([0.8, 0.2], seed=4321)

# Especificar las capas de la red neuronal
# Número de entradas (features), dos capas ocultas de tamaños 20 y 15, y el número de clases de salida
layers = [5, 20, 15, len(labelIndexer.labels)]

# Crear el clasificador
nn = MultilayerPerceptronClassifier(layers=layers, blockSize=256, seed=1234, maxIter=200, featuresCol="features", labelCol="label")

# Entrenar el modelo
model = nn.fit(train_data)

# Predecir
predictions = model.transform(test_data)

(50) trabajos de Spark
train_data: pyspark.sql.dataframe.DataFrame
test_data: pyspark.sql.dataframe.DataFrame
predictions: pyspark.sql.dataframe.DataFrame

```

A continuación, la evaluación de la red neuronal:

```

22/5/2024 (36 s) 69
# Evaluador para la precisión
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Precisión: {:.2f}".format(accuracy))

# Evaluador para el F1 Score
evaluator_f1 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="f1")
f1_score = evaluator_f1.evaluate(predictions)
print("F1 Score: {:.2f}".format(f1_score))

# Matriz de confusión
confusion_matrix = predictions.groupBy("label").pivot("prediction").count()
confusion_matrix.show()

(9) trabajos de Spark
confusion_matrix: pyspark.sql.dataframe.DataFrame = [label: double, 0.0: long ... 1 campo adicional]

Precisión: 0.55
F1 Score: 0.46
+-----+
|label|  0.0|  1.0|
+-----+
|  0.0|22622|2284|
|  1.0|17426|2282|
|  4.0|  116|NULL|
|  3.0|  215|  51|
|  2.0|  306|  16|
|  5.0|   43|   4|
+-----+

```

Vemos cómo la red Neuronal presenta un rendimiento mediocre, a pesar de los intentos de una correcta implementación. Esto sugiere que la calidad de los datos no es la mejor y que si se desea presentar modelos de aprendizaje de máquina es requerido tener unos datos de muy alta calidad.