

Proyecto de Análisis de Datos

I. DEFINICIÓN DEL CASO DE ESTUDIO

Este proyecto se basa en la realización de distintos casos de estudio, que consisten en analizar diferentes temas para comprender mejor su situación y su importancia. En este trabajo se estudian cuatro temas: Netflix, la política en Ecuador, los Juegos Olímpicos y los accidentes vehiculares, haciendo referencia a la temática del tráfico vehicular. Para ello se utilizan archivos CSV obtenidos de la plataforma Kaggle y se accede a páginas de datos abiertos del Ecuador. Con esta información se busca descubrir tendencias o patrones y encontrar posibles soluciones para cada caso de estudio.

II. OBJETIVO GENERAL Y ESPECÍFICOS

Objetivo general:

Analizar datos de las distintas temáticas, aplicando las técnicas de manejo de datos desde la recolección y limpieza hasta la transformación y visualización aprendidas para descubrir patrones y generar conclusiones.

Objetivos específicos:

- Extracción y limpieza de datos mediante Python y paginas como Kaggle y datos abiertos.
- Integrar los datos en dos bases de datos relacional (MySQL, SQLite, SQL Server) y NoSQL (MongoDB).
- Crear dashboards en Power BI que permitan visualizar tendencias.

III. DESCRIPCIÓN DEL EQUIPO DE TRABAJO Y ACTIVIDADES REALIZADAS POR CADA UNO.

Camila Bueno: Encargada de la búsqueda de datos sobre las temáticas, la exportación a MySQL, el análisis de sentimientos y la elaboración de visualizaciones en Power BI relacionadas con la temática de tránsito.

Roger Grefa: Responsable de la exportación de datos a SQLite sobre la política en Ecuador, la creación de visualizaciones correspondientes a su temática y la elaboración de la presentación en PowerPoint.

Diego Montaluisa: Encargado de la conexión de la temática de Juegos Olímpicos a MongoDB, la integración de toda la data en SQL Server y la creación de visualizaciones en Power BI para sus temáticas.

IV. CRONOGRAMA DE ACTIVIDADES

Actividad	Fecha Inicio	Fecha Fin
Extracción de datos	25/07/2025	26/07/2025
Limpieza de datos	27/07/2025	29/07/2025
Análisis de datos	30/07/2025	1/01/2025
Visualización de datos	02/01/2025	02/08/2025
Redacción del informe	03/02/2025	04/02/2025

V. RECURSO Y HERRAMIENTAS UTILIZADAS

Para el desarrollo del proyecto se utilizaron las siguientes herramientas:






- **Python:** Para la extracción, limpieza y c conexión entre las bases de datos y transformación de archivos.
- **Power BI:** Para la creación de dashboards interactivos.
- **MySQL, SQLite, SQL Server:** Para el almacenamiento de los csv y json , datos relacionales.
- **MongoDB Compass, Mongo DB Atlas :** Para el almacenamiento de csv datos no relacionales.
- **Librerías de Python:** pandas, pymongo, sqlalchemy.

VI. ARQUITECTURA DE LA SOLUCIÓN.

La arquitectura del proyecto se compone de las siguientes etapas:

1. **Extracción de datos:** Se obtuvieron datos de Kaggle, y páginas con datos abiertos sobre nuestro país que proporciona el estado como datosabiertos.com o INEC.
2. **Limpieza:** Se utilizó Python y mediante código se realizó la limpieza de los csv y json.
3. **Almacenamiento:** Los datos se almacenaron en SQLite, MySQL, MongoDB y SQL Server.
4. **Análisis:** Se aplicó análisis de sentimientos al dataframe.

SQLite

- ✓  Tables (1)
 - >  provincias
-  Indices (0)
-  Views (0)
-  Triggers (0)

Y desde la interfaz grafica podemos ver que la tabla ya se encuentra dentro de SQLite.

Ademas para comprobar podemos realizar la conexión con esta base desde VS code

Iniciamos con la importación de los recursos necesarios para la conexión

```
import sqlite3
import pandas as pd
```

Procedemos con la conexión

```
# Conexión
conexion = sqlite3.connect("proyecto.db")
```

Verificamos cuantas tablas tenemos en SQLite con el siguiente código e imprimimos las tablas que hayan.

```
consulta = "SELECT name FROM sqlite_master WHERE type='table';"
tablas = pd.read_sql_query(consulta, conexion)
print("Tablas en la base de datos:")
print(tablas)
```

En este caso solo nos muestra una tabla llamada provincia

```
Tablas en la base de datos:
  name
0  provincias
```

Hacemos la conexión con la tabla y asignamos la primera fila como nombres de la columna y borramos el índice anterior.

```
df = pd.read_sql_query("SELECT * FROM provincias", conexion)
df.columns = df.iloc[0] # Asignar primera fila como nombres de columna
df = df[1:].reset_index(drop=True) # Eliminar la primera fila original y reiniciar índice
```

luego imprimimos los elementos de la tabla y finalizamos la conexión.

```
print("Contenido de la tabla 'provincias':")
print(df.head())

conexion.close()
```

Resultado:

```
Contenido de la tabla 'provincias':
0  Text filter  Country  Place  Date Retweet count \
0  Andres Arauz  Ecuador  Pichincha  1/1/2021 14:13  786
1  Andres Arauz  Ecuador  Pichincha  1/1/2021 14:14  9
2  Yaku Perez  Ecuador  Azuay  1/1/2021 14:14  96
3  Guillermo Lasso  Ecuador  Azuay  1/1/2021 14:14  90
4  Andres Arauz  Ecuador  Pichincha  1/1/2021 14:14  933

0  Reply count Favorites Timestamp
0  145 309 Yes
1  15 50 No
2  30 79 No
3  29 77 No
4  169 358 No
```

MySQL Workbench

Para importar los datos limpios a una base de datos relacional en **MySQL Workbench**, se utilizó la librería **SQLAlchemy** en Python, la cual facilita la creación de conexiones con bases de datos. Primero, se cargó el archivo *siniestros_limpio.csv* en un DataFrame usando Pandas. Luego, se definieron los parámetros de conexión (usuario, contraseña, host, puerto y nombre de la base de datos) y se creó el enlace mediante `create_engine()`. Finalmente, se utilizó el método `to_sql()` para enviar el contenido del DataFrame a MySQL, creando la tabla *siniestros* de forma automática.

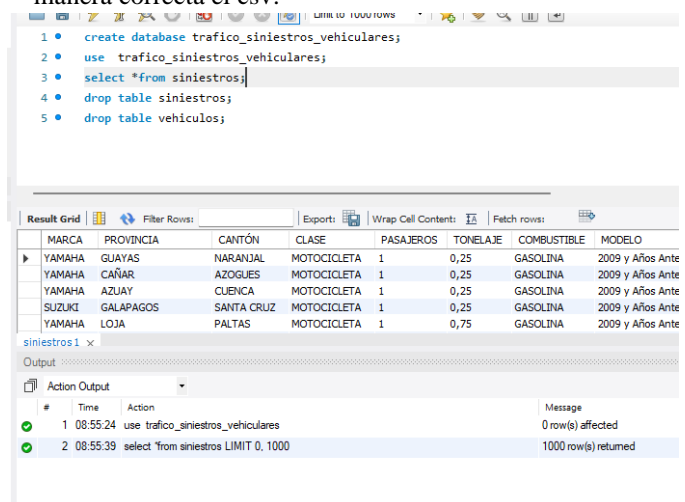
```
import pandas as pd
from sqlalchemy import create_engine
df = pd.read_csv('siniestros_limpio.csv')

# Datos de conexión a MySQL
usuario = 'root'
contrasena = '1234'
host = 'localhost'
puerto = '3306'
basee = 'trafico_siniestros_vehiculares'

# Crear conexión con SQLAlchemy
conexion = create_engine('mysql+pymysql://({usuario}):({contrasena})@({host}):({puerto})/({basee})')

# Subir el DataFrame a MySQL
df.to_sql('siniestros', con=conexion, index=False)
print("Datos importados a MySQL.")
```

Una vez importado podemos abrir el programa y ejecutar el comando de `select * from` para comprobar que se exporto de manera correcta el csv.



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following commands:

```
1 create database trafico_siniestros_vehiculares;
2 use trafico_siniestros_vehiculares;
3 select * from siniestros;
4 drop table siniestros;
5 drop table vehiculos;
```

The Results Grid shows the output of the `select * from siniestros;` query, displaying columns: MARCA, PROVINCIA, CANTÓN, CLASE, PASAJEROS, TONELAJE, COMBUSTIBLE, and MODELO. The first few rows are visible:

MARCA	PROVINCIA	CANTÓN	CLASE	PASAJEROS	TONELAJE	COMBUSTIBLE	MODELO
YAMAHA	GUAYAS	NARANJAL	MOTOCICLETA	1	0,25	GASOLINA	2009 y Años Ante
YAMAHA	CAÑAR	AZOQUES	MOTOCICLETA	1	0,25	GASOLINA	2009 y Años Ante
YAMAHA	AZUAY	CUENCA	MOTOCICLETA	1	0,25	GASOLINA	2009 y Años Ante
SUZUKI	GALAPAGOS	SANTA CRUZ	MOTOCICLETA	1	0,25	GASOLINA	2009 y Años Ante
YAMAHA	LOJA	PALTAS	MOTOCICLETA	1	0,75	GASOLINA	2009 y Años Ante

The Action Output pane shows the execution of the `select * from siniestros LIMIT 0, 1000;` query, indicating that 1000 row(s) were returned.

Con ello comprobamos que si se conecto correctamente y la exportación del csv utilizando Python si se dio.

MongoDB Atlas

Para la temática de los Juegos Olímpicos, se realizó la conexión a **MongoDB** utilizando la librería pymongo. Primero, se creó el cliente con la ruta de conexión mongodb://localhost:27017/ y se seleccionó la base de datos llamada *ProyectoDatos*. Dentro de esta base, se trabajó con la colección *Deportes* El DataFrame con la información previamente procesada se convirtió en una lista de diccionarios mediante el método to_dict('records') y se insertó en la colección con insert_many(). Finalmente, para verificar que los datos fueron cargados correctamente, se utilizó collection.find() para consultar todos los registros almacenados.

```
#Conectar a MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client['ProyectoDatos']
collection = db['Deportes']
# Insertar el DataFrame en MongoDB
collection.insert_many(df.to_dict('records'))

# Consultar los datos insertados
resultados = collection.find()
```

Asi desde la interfaz grafica podemos verificar que se cargo de manera correcta.



Procesos de migración entre bases de datos SQL y NoSQL

Migración de MySQL a MongoDB

Para realizar la migración entre estas dos bases se utilizo por código ya los archivos anteriormente limpiados y cargados se los subió a Mongo Atlas De la siguiente manera

```
import pandas as pd
from pymongo import MongoClient

df = pd.read_csv("vehiculos_siniestros.csv", encoding="utf-8")

client = MongoClient("mongodb://diagonalsal:Ar1es188@cluster0.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0")
db = client['ProyectoDatos']
coleccion = db['Vehiculos_siniestros']

datos = df.to_dict(orient="records")

if datos:
    coleccion.insert_many(datos)
    print(" Datos subidos correctamente a MongoDB Atlas.")
else:
    print("El archivo CSV está vacío.")
```

De esta forma, primero se estableció la conexión con **MongoClient**, indicando la ruta de conexión al servidor de MongoDB Atlas. Luego, se seleccionó la base de datos predeterminada que había sido creada previamente para el proyecto. A continuación, se subieron los archivos correspondientes a cada temática, en formato CSV o JSON, cargándolos en Python y convirtiéndolos a listas de diccionarios mediante el método to_dict('records'). Finalmente, se insertaron en las colecciones designadas dentro de la base de datos, quedando la información lista para consultas y análisis posteriores.

Migracion de SQLite a MongoDB

Para la exportación de esto se realizo de la misma manera por código ya que dado que la data estaba limpia.

```
import json
from pymongo import MongoClient

cliente = MongoClient("mongodb://diagonalsal:Ar1es188@cluster0.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0")

db = cliente['ProyectoDatos'] # base de datos
coleccion = db['Eleccion_2021'] # coleccion

with open('provincias_limpio.json', 'r', encoding='utf-8') as archivo:
    datos = json.load(archivo)

if isinstance(datos, list):
    coleccion.insert_many(datos)
else:
    coleccion.insert_one(datos)

print("Todos subidos correctamente a MongoDB Atlas.")
```

Realizamos lo mismo con otros archivos.
Migracion de MongoDB a SQ Server

Así comprobamos que todos los archivos provenientes de las bases de datos relacionales anteriores se encontraban correctamente almacenados en **MongoDB Atlas**, asegurando que la migración y carga de información se realizó de manera exitosa.

Collection Name	Documents	Logical Data Size	Arg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
Deportes_Olimpicos	2667	1,084KB	425B	476KB	1	100KB	100KB
Deportes	162804	57,799KB	373B	14,614KB	1	5,254KB	5,254KB
Eleccion_2021	3736	468,348B	184B	248KB	1	128KB	128KB
Pasatiempos	1000	235,678B	242B	120KB	1	56KB	56KB
Traslado	1204297	252,799KB	221B	55,949KB	1	31,994KB	31,994KB
Vehiculos_limpio	201074	42,129KB	220B	10,949KB	1	4,689KB	4,689KB
Vehiculos_siniestros	201074	42,129KB	220B	11,094KB	1	5,834KB	5,834KB

Conexion a CouchDB

En este proyecto, se utilizó Python para conectarse a una base de datos NoSQL Couchbase con el objetivo de extraer todos los documentos del bucket asignado. Para ello, se emplearon librerías específicas como couchbase, dns, y time, lo que permitió establecer conexión segura al clúster mediante credenciales configuradas. Luego se seleccionó la colección deseada y se ejecutó una consulta N1QL para obtener los datos. Finalmente, se incorporó control de errores para asegurar que el proceso se realice de forma correcta y estable. Este procedimiento facilitó el acceso directo a la información, permitiendo futuras tareas de limpieza, análisis y visualización.

```
import json
from couchbase.cluster import Cluster
from couchbase.options import ClusterOptions
from couchbase.auth import PasswordAuthenticator
from couchbase.exceptions import CouchbaseException

# Conexión
cluster = Cluster("couchbase://localhost", ClusterOptions(
    PasswordAuthenticator("Diago", "Ar1es188")))
bucket = cluster.bucket("noticias")
coleccion = bucket.default_collection()

# Leer lista de objetos
with open("noticias.json", "r", encoding="utf-8") as f:
    lista = json.load(f)

# Subir cada documento
for doc in lista:
    doc_id = doc.get("id")
    if doc_id:
        coleccion.upsert(doc_id, doc)
        print(f"🟢 Documento '{doc_id}' subido.")
    else:
        print(f"Documento sin ID, omitido.")
```

Análisis de sentimiento

```

import pandas as pd
df = pd.read_csv('netflix_titles1.csv')

# Filamos a live
positive_words = ['fun', 'happy', 'love', 'great', 'excellent', 'adventure', 'inspiring']
negative_words = ['murder', 'crime', 'death', 'sad', 'abuse', 'tragedy', 'drug']

# Clasificador simple
def sentiment(text):
    text = str(text).lower()
    for word in positive_words:
        if word in text:
            return 'positive'
    for word in negative_words:
        if word in text:
            return 'negative'
    return 'neutral'

df['sentiment'] = df['description'].apply(sentiment)
print(df[['title', 'description', 'sentiment']])

```

```

[22]
...
      title \
0    Dick Johnson Is Dead
1    Blood & Water
2    Ganglands
3    Jailbirds New Orleans
4    Kota Factory
...
8802   Zodiac
8803  Zombie Dumb
8804  ZombieLand
8805    Zoom
8806    7/19/96

```

Se realizo desde la interfaz grafica se exportaron los archivos csv en la cual cada csv representaba la colección previo a esto se exportaron desde Atlas las colecciones a csv ya que la data estaba previamente limpia

Podemos verificar que ya se abrió

Navigator

Display Options

localhost\SQLEXPRESS: DatosProyecto [7]

ProjectoDatos.Deportes_Olimpicos

ProjectoDatos.Deportistas

ProjectoDatos.Elecciones_2021

ProjectoDatos.Pasajeros

ProjectoDatos.Transito

ProjectoDatos.Vehiculos_Limpio

ProjectoDatos.Vehiculos_sinistros

ProjectoDatos.Vehiculos_sinistros

id	MARCA	PROVINCIA
68917eb631a7fe023214c0d	YAMAHA	GUIYAS
68917eb631a7fe023214c11	YAMAHA	CAÑAR
68917eb631a7fe023214c2	YAMAHA	AZUAY
68917eb631a7fe023214c3	SUZUKI	LAGOAGOS
68917eb631a7fe023214c4	YAMAHA	LOJA
68917eb631a7fe023214c5	YAMAHA	SANTA ELENA
68917eb631a7fe023214c6	YAMAHA	EL ORO
68917eb631a7fe023214c7	YAMAHA	CAÑAR
68917eb631a7fe023214c8	YAMAHA	LOJA
68917eb631a7fe023214c9	YAMAHA	ZAMORA CHINCHIPE
68917eb631a7fe023214ca	YAMAHA	CAÑAR
68917eb631a7fe023214cb	SUZUKI	GUIYAS
68917eb631a7fe023214cc	HONDA	TUNGURAHUA
68917eb631a7fe023214cd	MTM	ESMERALDAS
68917eb631a7fe023214ce	MTM	MANABI
68917eb631a7fe023214cf	HONDA	AZUAY
68917eb631a7fe023214d0	YAMAHA	AZUAY
68917eb631a7fe023214d1	KAWASAKI	AZUAY
68917eb631a7fe023214d2	HONDA	TUNGURAHUA
68917eb631a7fe023214d3	SUZUKI	AZUAY
68917eb631a7fe023214d4	HONDA	LOJA
68917eb631a7fe023214d5	GEELY	CAÑAR
68917eb631a7fe023214d6	KINGTON	CAÑAR

Select Related Tables

Load

Transform Data

Grafico 1: Suma de muertos por provincia

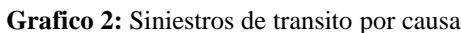


Grafico 2: Siniestros de transito por causa

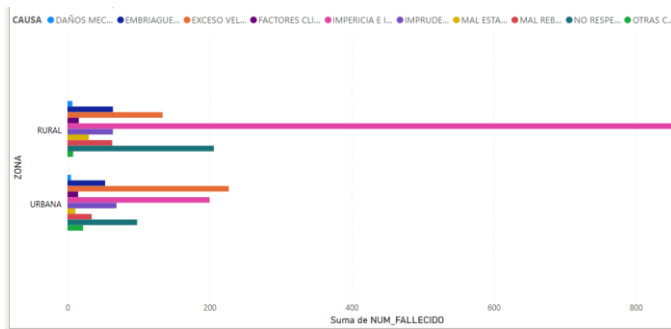


Grafico 3: Numero de fallecidos y lesionados por provincia

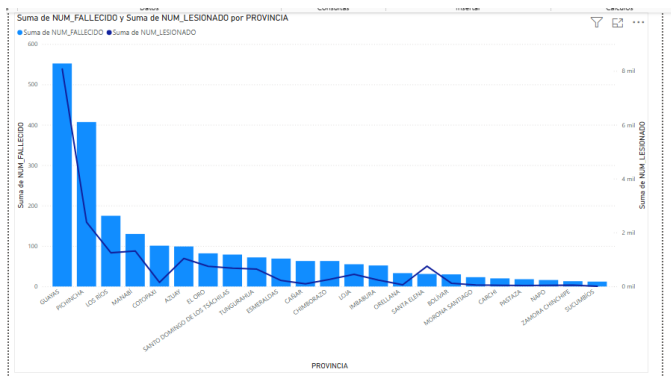


Grafico 4: Distribucion por cada tipo de vehículo

Grafico 7: En este caso todas las interacciones que hubo entre los candidatos fueron específicamente de Ecuador, y no hubo interacción de ecuatorianos en otros países



Grafico 5: Dominancia de países por categoría de películas

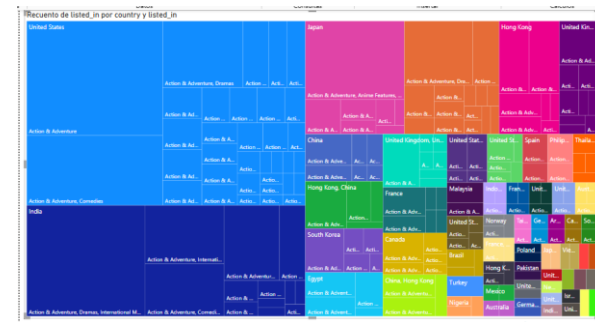
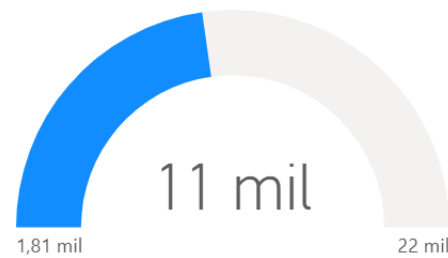


Grafico 6: Podemos ver que esta por debajo el promedio de lo retweet de la población. Máx. de Retweet count y Promedio de Retweet count



Recuento de Country por Place

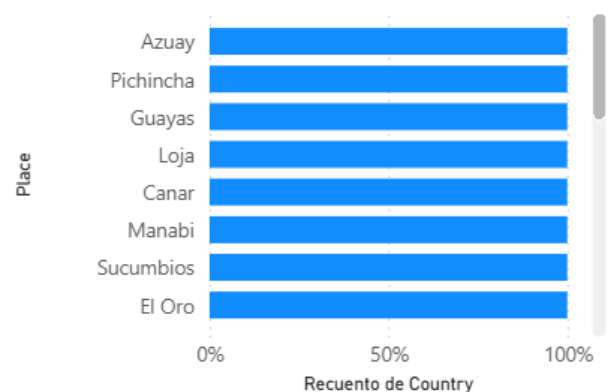


Grafico 8: La fecha con respecto a los retweet y se da la conclusión de que se da en el mes de enero y pocos días de febrero

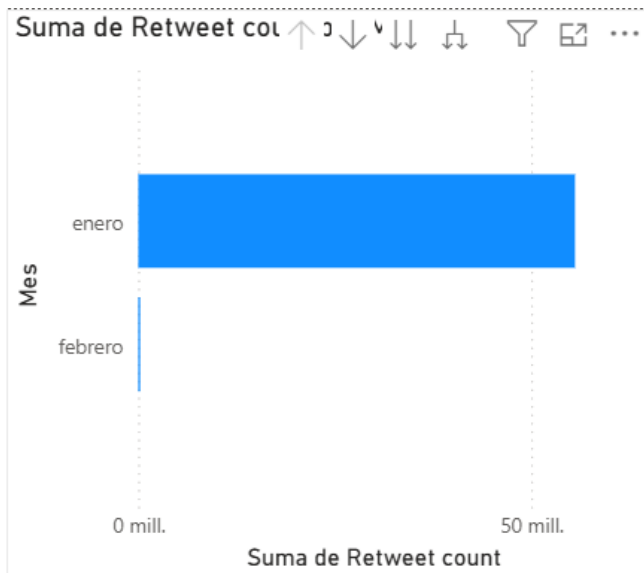


Grafico 9: En este estudio se puede apreciar que en la provincia de pichincha el candidato posee mas apoyo.

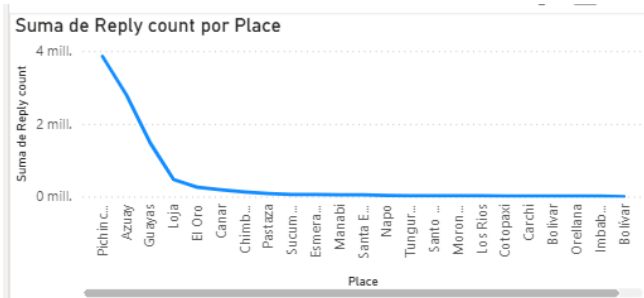


Grafico 9: En este estudio se puede apreciar que en la provincia de pichincha el candidato posee mas apoyo.

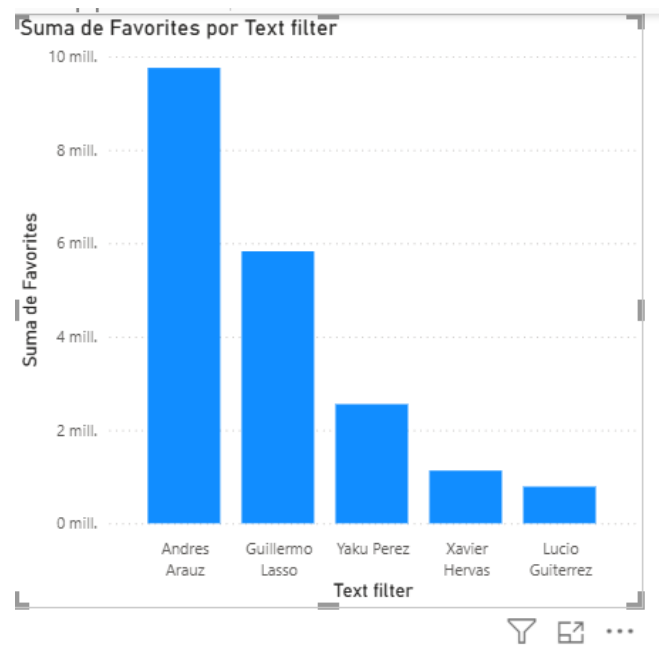


Grafico 10: en esta grafica se puede apreciar que Andres Arauz tiene un dice alto de ser de los favoritos en estas elecciones

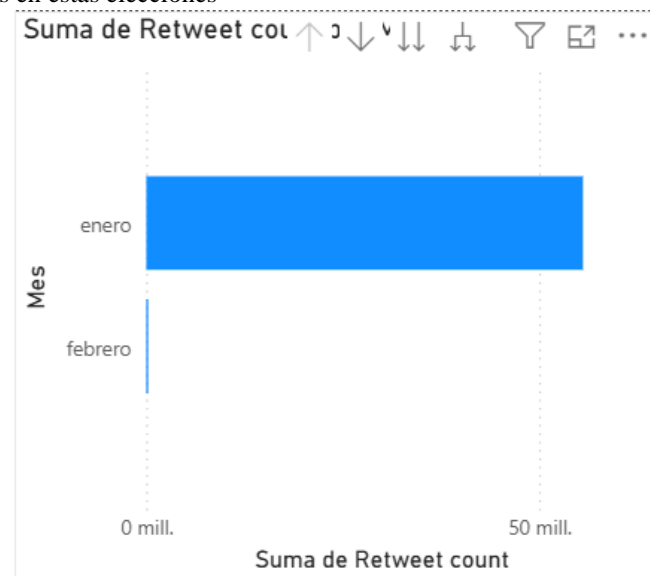
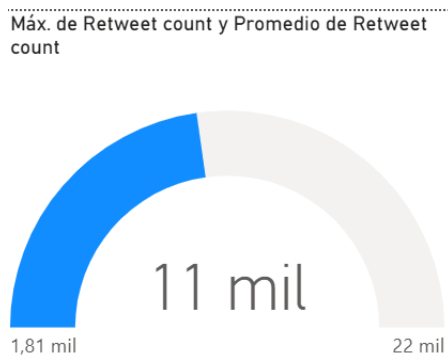


Grafico 6: Podemos ver que esta por debajo el promedio de lo retweet de la población



[1] **Kaggle Dataset** Kaggle.com, 2025.
<https://www.kaggle.com/datasets>

[2] “Conjunto de datos - Datos Abiertos Ecuador”, Datosabiertos.gob.ec , 2021.
<https://www.datosabiertos.gob.ec/dataset/>

X. RESULTADOS OBTENIDOS

1. Se podría determinar que la mayoría de muertes se dan en las dos provincias principales del país que son Quito y Guayaquil.
2. La mayoría de muertes se dan por fallas del mecánico o en si choques esto serviría para prevenir o buscar una entidad que se encargue de esto.
3. En Ecuador hay dos pulsos políticos que predominan en Ecuador.
4. En el país quienes predominan son los RC5 y la de Guillermo Lasso.

XI. CONCLUSIONES Y RECOMENDACIONES.

La extracción de datos desde la web representa una estrategia valiosa para fortalecer la calidad y actualidad del análisis de información. Mediante técnicas como web scraping o el uso de APIs abiertas, es posible acceder a datasets relevantes y representativos del contexto ecuatoriano. Esta práctica no solo permite enriquecer los modelos analíticos, sino también facilita la toma de decisiones informadas al integrar múltiples fuentes y temáticas. En definitiva, aprovechar datos dinámicos directamente desde la web impulsa una mejora sustancial en el desarrollo de soluciones tecnológicas orientadas al país

XII. DESAFÍOS Y PROBLEMAS ENCONTRADOS

Se intentó usar APIs para obtener datos, pero no dieron información útil o presentaron errores. También se trató de exportar datos desde el código hacia diferentes bases, pero hubo problemas como fallas del servidor o estructuras incorrectas. Por eso, se usaron otras fuentes como archivos CSV de Kaggle.

XIII. BIBLIOGRAFIA