PROYECTO INTEGRAL
Master en IA y Big Data

VIAD

Málaga Tech Park
Centro Público Integrado de FP

Autores:
Jesús Jiménez García
Jose Pérez Soler

2022-2023

ÍNDICE

1º Justificación y descripción del proyecto.

La aparición del Covid 19 ha demostrado la imperiosa necesidad del sistema sanitario de poseer herramientas para la predicción de picos de ingresos, escoger tratamientos adecuados o tener al personal específico en plantilla para poder afrontar las diferentes visicitudes a las que este mundo global e interconectado nos tiene últimamente acostumbrados.VIAD nace de esa necesidad, no sólo aporta la posibilidad de realizar predicciones sino que también implementa la consulta de todos los datos estadísticos disponibles en el INE, de forma fácil y amigable.VIAD es el resultado de un entrenamiento automático supervisado con grandes volúmenes de datos.

2º Objetivo

VIAD (vigilante inteligente atenuador de defunciones) tiene como objetivo predecir las defunciones, las causas de las mismas, por provincias y fechas. Este modelo usa técnicas avanzadas de aprendizaje automático supervisado y análisis de grandes conjuntos de datos históricos, para así proporcionar predicciones lo más precisas posibles. La información que produzca este modelo será de gran utilidad para los profesionales de la salud y para los políticos a la hora de tomar decisiones.
Predicción de defunciones por:
- Causas
- Fechas
- Por Provincia
- Comunidad autónoma.

3º Instalación-importación

Instalamos los paquetes y las liberías que vamos a usar.

```
import os
import glob
import time
import pickle
import sys
import math
import boto3 # Amazon AWS
from google.colab import files
import pandas as pd # Análisis de datos
import numpy as np # Funciones con matrices
import matplotlib.pyplot as plt # Gráficos
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns # Gráficos
import joblib # exportación de modelos
# Machine Learning
from statsmodels.tsa.arima.model import ARIMA
from pmdarima.arima import auto_arima
```

```
!pip install boto3
!pip install matplotlib
!pip install pmdarima
```

4º S3

Una vez tenemos las BBDDs localizadas procedemos a subirlas a S3, ya que allí hemos creado nuestro bucket y sus carpetas correspondientes donde tenemos almacenados las BBDDs.

```
# Creamos la sesión con nuestros datos de acceso
session = boto3.Session(aws_access_key_id, aws_secret_access_key, aws_session_token)

# Creamos el cliente que interactua con el S3
s3 = session.client('s3', region_name='us-east-1')

# Nombre del bucket y ruta de la carpeta a descargar
bucket_name = 'viad'
folder_path = 'Proyecto-Datalake/'

# Lista de las carpetas internas de las que obtendremos los datos
years = ['2015', '2016', '2017', '2018', '2019', '2020', '2021']

# Nombre de las carpetas donde guardaremos los datos
destination_csv = "csv/"

# Verificar si el directorio de destino existe y crearlo si es necesario
if not os.path.exists(destination_csv):
  os.makedirs(destination_csv)

# Recorremos cada carpeta y descargamos los ficheros
for year in years:

  # Descargar todos los ficheros de la carpeta
  response = s3.list_objects_v2(Bucket=bucket_name, Prefix=folder_path + year)

  for obj in response['Contents']:
    # Obtener la ruta y el nombre del archivo
    file_path = obj['Key']
    file_name = os.path.basename(file_path)

    if file_name.split(sep=".")[1] == "csv" and file_name[0] != "Z":
      s3.download_file(bucket_name, file_path, destination_csv + file_name)

print("Archivos descargados.")
```

5º Obtención de datos.

Los datos han sido obtenidos del INE, aunque se solicitó estadísticas por meses el organismo únicamente nos la facilitaría a cambio de un pago que no se podía subsanar para el proyecto, por lo que en lugar de realizar predicciones mensuales, las hemos tenido que realizar anuales.

6º Agrupación-Categorización BBDD

Como la columna Comunidad es el resultado de la suma de la columna Provincia, la borramos, luego agrupamos por años, renombramos columnas y concatenamos con el resto de dataframes.

```
# Se borra la columna comunidad por que es la suma de las provincias
community_to_delete = ["Andalucía", "Aragón", "Canarias", "Cataluña",
                       "Castilla y León", "Castilla-La Mancha", "Extremadura",
                       "Galicia", "País Vasco", "Comunitat Valenciana"]

df_year = []

# Leemos todos los CSV por años y lo añadimos a df_year
for year in years:
  file_list = glob.glob('csv/' + year + '*')
  dfs = []

  for filename in file_list:
    # Leemos el archivo
    df = pd.read_csv(filename, delimiter=';', dtype=str)

    # Comprobamos si tiene la columna provincia sino tiene columna provincia se la ponemos
    if 'Provincia de residencia' not in df.columns:
      df['Provincia de residencia'] = filename.split(sep='-')[1]
    else:
       df = df[~df['Provincia de residencia'].isin(community_to_delete)]

    # Cambiamos la columna a enteros
    df['Total'] = df['Total'].str.replace('.', '', regex=True).astype(int)

    # Cambiamos los nombre de las columnas
    df.rename(columns={
        'Causas (lista reducida)': 'Disease',
        'Provincia de residencia' : 'Province',
        'Sexo' : 'Sex',
        'Edad' : 'Age'
        }, inplace=True)

    # Añadimos a la lista de dataframes
    dfs.append(df)

  # Concatenamos con los otros dataframe del mismo año
  df_year.append(pd.concat(dfs, ignore_index=True))
  df_year[-1]['Year'] = year
```

7º Agrupamiento del Dataset

El siguiente paso sería fusionar todos los dataframes con todos los años, con las columnas eliminadas, desde el año 2000 hasta el 2021.

```
[ ]  # Juntamos todos los dataframes de todos los años comprobamos que están todos los años, desde el 2000 hasta 2021
     causes_death_df = pd.concat(df_year, ignore_index=True)

[ ]  causes_death_df
```

8º Limpieza de datos

Debido a que los datos no vienen limpios ni ordenados adecuadamente para trabajar con ellos hemos tenido que agrupar y categorizar por año añadiendo las provincias.
Hemos borrado los duplicados y reunificado aquellas que tenían nombres parecidos.

```
# Revisamos los valores únicos para la columna Province
causes_death_df['Province'].unique()
```

```
array(['Almería', 'Cádiz', 'Córdoba', 'Granada', 'Huelva', 'Jaén',
       'Málaga', 'Sevilla', 'Madrid', 'Melilla', 'Alicante/Alacant',
       'Castellón/Castelló', 'Valencia/València', 'Murcia', 'Coruña (A)',
       'Lugo', 'Ourense', 'Pontevedra', 'Cantabria', 'Asturias',
       'Albacete', 'Ciudad Real', 'Cuenca', 'Guadalajara', 'Toledo',
       'Huesca', 'Teruel', 'Zaragoza', 'Badajoz', 'Cáceres', 'Avila',
       'Burgos', 'León', 'Palencia', 'Salamanca', 'Segovia', 'Soria',
       'Valladolid', 'Zamora', 'Navarra', 'Rioja', 'Baleares',
       'Palmas (Las)', 'Santa Cruz de Tenerife', 'Ceuta', 'Alava',
       'Guipúzcoa', 'Vizcaya', 'Barcelona', 'Girona', 'Lleida',
       'Tarragona', 'Coruña, A', 'Palmas, Las', 'Araba/Alava', 'Bizkaia',
       'Gipuzkoa', 'Valencia/Val\x8ancia'], dtype=object)
```

```
# Reemplazamos los nombres duplicados o con erratas para unificarlos
causes_death_df['Province'] = causes_death_df['Province'].replace(
    [
        'Valencia/Val\x8ancia',
        'Alava',
        'Guipúzcoa',
        'Vizcaya',
        'Coruña (A)',
        'Palmas (Las)',
    ],
    [
        'Valencia/València',
        'Araba/Alava',
        'Gipuzkoa',
        'Bizkaia',
        'Coruña, A',
        'Palmas, Las',
    ])
```

```
causes_death_df['Sex'].unique()

array(['Ambos sexos', 'Hombres', 'Mujeres'], dtype=object)

causes_death_df['Year'].unique()

array(['2015', '2016', '2017', '2018', '2019', '2020', '2021'],
      dtype=object)

print(causes_death_df.isnull().sum())

Disease      0
Sex          0
Age          0
Total        0
Province     0
Community    0
Year         0
dtype: int64
```

También hemos creado diccionarios para transformar los strings en enteros para que luego al modelo le sea más sencillo ser entrenado.

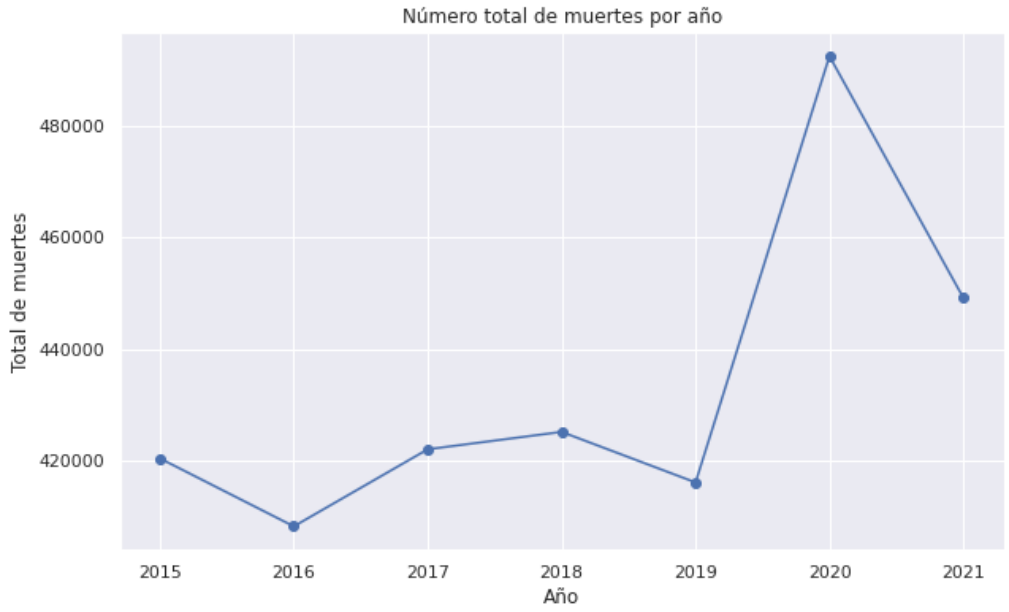
```
causes_death_df["Age"].replace(['Menores de 1 año', 'De 1 a 14 años',
                                'De 15 a 29 años', 'De 30 a 39 años', 'De 40 a 44 años',
                                'De 45 a 49 años', 'De 50 a 54 años', 'De 55 a 59 años',
                                'De 60 a 64 años', 'De 65 a 69 años', 'De 70 a 74 años',
                                'De 75 a 79 años', 'De 80 a 84 años', 'De 85 a 89 años',
                                'De 90 a 94 años', 'De 95 años y más'],
                                [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16],
                                inplace=True)
```

Aquí tenemos una parte del código

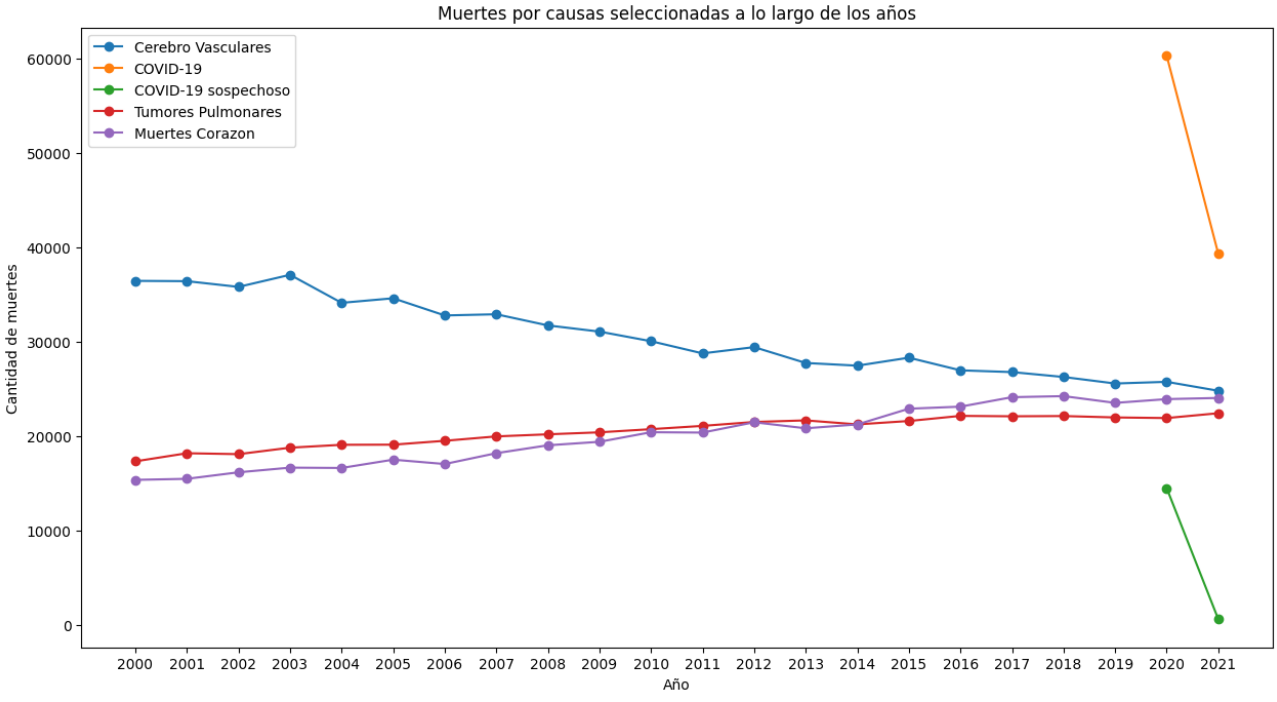
```
# Reemplazamos los nombres duplicados o con erratas para unificarlos
causes_death_df['Disease'] = causes_death_df['Disease'].replace(
    [
      '014 Tumor maligno del hígado y de las vías biliares intrahepáticas',
      '020 Tumor maligno del hueso y de los cartílagos articulares',
      '062 Influenza (gripe)',
      '099  Agresiones (Homicidios)',
      '086 Paro cardíaco, muerte sin asistencia y causa desconocida de mortali
      '064 Enfermedades crónicas de las vías respiratorias inferiores (excepto
      '042-043 III.Enf. de sangre y de órg. hematopoyéticos y ciertos trast. qu
```

9º Visualización

A continuación hemos realizado alguna gráficas para comprobar como ha sido la evolución de las defunciones a lo largos de los años, resulta llamativo que en el 2020 la tasa de defunciones se dispara, debido a la aparición del Covid 19.



Hemos realizado una gráfica con las causas más comunes de defunciones y hemos añadido el Covid19, justo en 2020 aparece un repunte. También podemos observar como va descendiendo los fallecidos por esa causa.



10º Exportación

Exportamos los datos tratados a CSV con el fin de poder incluir los datos de MongoDB y poder hacer consultas.

```
[ ]  causes_arima_df.to_csv('causes_death_categorized.csv', index=False)

     files.download('causes_death_categorized.csv')
```

11 Entrenamiento-ARIMA

Debido a que nuestra base de datos es una serie de tiempo con una gran cantidad de datos usaremos ARIMA.

ARIMA son las siglas de Autoregressive Integrated Moving Average, este modelo estadístico es usado para predicciones a futuro, se compone de tres partes:

- Componente Autoregresivo (AR), se refiere a la regresión de la serie del tiempo actual sobre los valores pasados.
- Componente de media móvil (MA), referido a la regresión de la serie de tiempo actual en términos de errores de predicción pasados.
- El componente integrado (I), se refiere a la diferenciación de la serie de tiempo para hacerla estacionaria, es decir, con una media y varianza constante en el tiempo.

Este modelo se usa mucho en campos como la economía, las finanzas ,la ingeniería y la meteorología. ARIMA tiene la capacidad de identificar patrones y tendencias en una serie de tiempo y analizar el impacto de eventos pasados en la serie de tiempo.

Básicamente se usa para :

- Predecir valores futuros en series de tiempo.
- Identificar patrones y tendencias en series de tiempo.
- Analizar el impacto de eventos pasados en una serie de tiempo

A continuación tenemos una muestra del dataframe con el que vamos a entrenar el modelo. Deberemos realizar una serie de modificaciones, entre ellas eliminar una serie de columnas.

```
# Este es dataframe al que le aplicaremos ARIMA
causes_arima_df
```

Borraremos las columnas Men y Women.

```
# Estamos borrando las columnas de Men y Women
df_both_genders = causes_arima_df.drop(['Men', 'Women'], axis=1)
df_both_genders
```

	Province	Disease	Both Genders	Age	Year
32	14	1	0	1	2000-01-01
33	14	1	0	2	2000-01-01
34	14	1	0	3	2000-01-01
35	14	1	0	4	2000-01-01
36	14	1	0	5	2000-01-01
...	...	...	...	...	...
2145723	36	102	0	12	2021-01-01
2145724	36	102	0	13	2021-01-01
2145725	36	102	0	14	2021-01-01
2145726	36	102	0	15	2021-01-01
2145727	36	102	0	16	2021-01-01

1871168 rows × 5 columns

Luego borramos las columnas Age, agrupamos las columnas restantes y sumamos las filas.

```
#Borramos la columna Age
df_both_genders_totalAge = df_both_genders.drop("Age", axis=1)

# Agrupar los datos por las columnas restantes y sumar las filas
df_both_genders_totalAge = df_both_genders_totalAge.groupby(["Province", "Disease", "Year"]).sum().reset_index()

# Verificamos el contenido y la suma de las columnas
df_both_genders_totalAge
```

Luego agrupamos por provincia, enfermedad, año y ambos generos.
Luego establecemos el año como indice para que ARIMA pueda trabajar con los datos.

```
# Establecemos el año como indice
df_both_genders_totalAge_province43_Disease18 = df_both_genders_totalAge_province43_Disease18.set_index('Year')

# verificamos que la columna de fecha es el índice del DataFrame
df_both_genders_totalAge_province43_Disease18
```

Luego seleccionamos un ejemplo para una provincia, causa de muerte y fecha en concreto, lo que tendríamos que realizar y se realiza, es un entrenamiento por cada una de estas consulta. En la web generaremos los entrenamientos a través de las consultas. Por lo que los entrenamientos pasan a ser dinámicos.

```
# Establecemos el año como indice
df_both_genders_totalAge_province43_Disease18 = df_both_genders_totalAge_province43_Disease18.set_index('Year')

# verificamos que la columna de fecha es el índice del DataFrame
df_both_genders_totalAge_province43_Disease18
```

```
# Convertir el dataframe en una serie de tiempo
df_both_genders_totalAge_province43_Disease18_ts = pd.Series(df_both_genders_totalAge_province43_Disease18['Both Genders'].values, index=df_both_genders_totalAge_province43_Disease18.index, dtype='int')

# Ajustamos el modelo ARIMA utilizando la función auto_arima
model = auto_arima(df_both_genders_totalAge_province43_Disease18_ts, start_p=1, start_q=1,
                   test='adf',       # Utilizamos el test de Dickey-Fuller aumentado
                   max_p=3, max_q=3, # Valores máximos de p y q
                   m=1,              # Frecuencia de la serie temporal (1 para anual)
                   d=None,           # d se estima automáticamente por la función
                   seasonal=False,   # No utilizamos componente estacional
                   trace=True)

# Mostramos los resultados
print(model.summary())
```

```
# Ajustar el modelo ARIMA
model = ARIMA(df_both_genders_totalAge_province43_Disease18_ts, order=model.order)
print("Empezando a entrenar")
model_fit = model.fit()
print("Entrenamiento acabado")
```

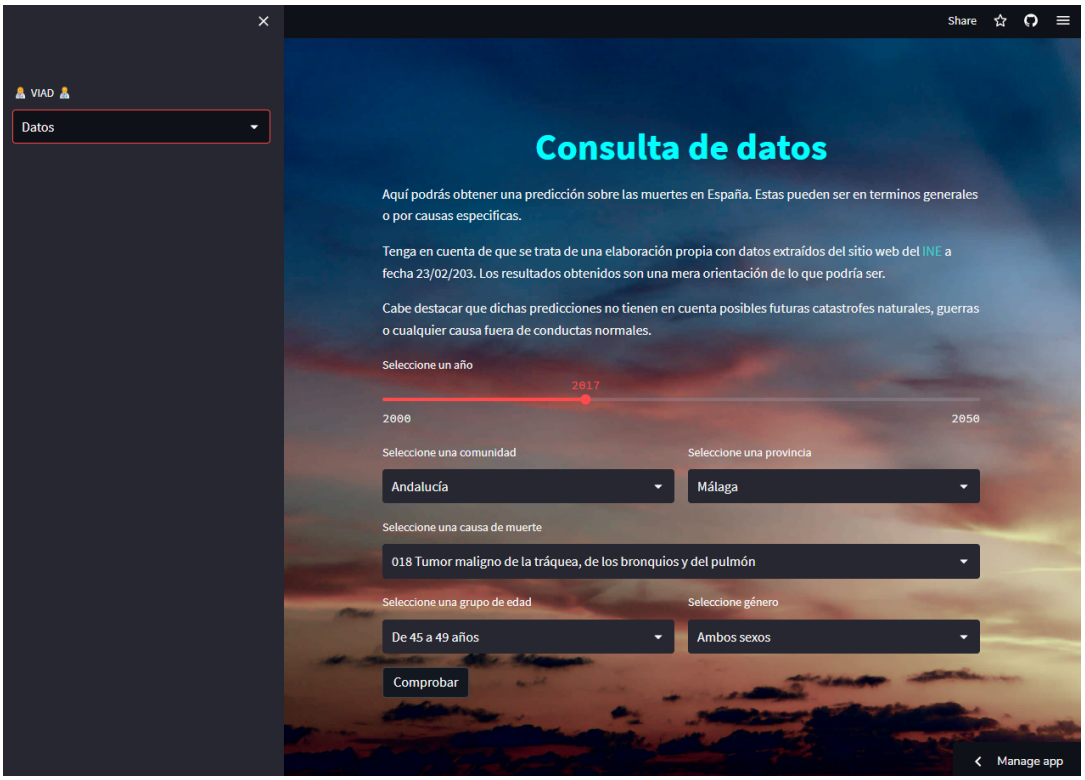
```
#Prediccion rango de fechas
year_to_predict = 2050
last_year = df_both_genders_totalAge_province43_Disease18_ts.index.max()
period = year_to_predict - int(pd.Timestamp(last_year).strftime('%Y'))

index = pd.date_range(start=str(last_year), periods=period, freq='Y')

prediction = model_fit.predict(start=index[0], end=index[-1])
print(prediction.round(0))
```

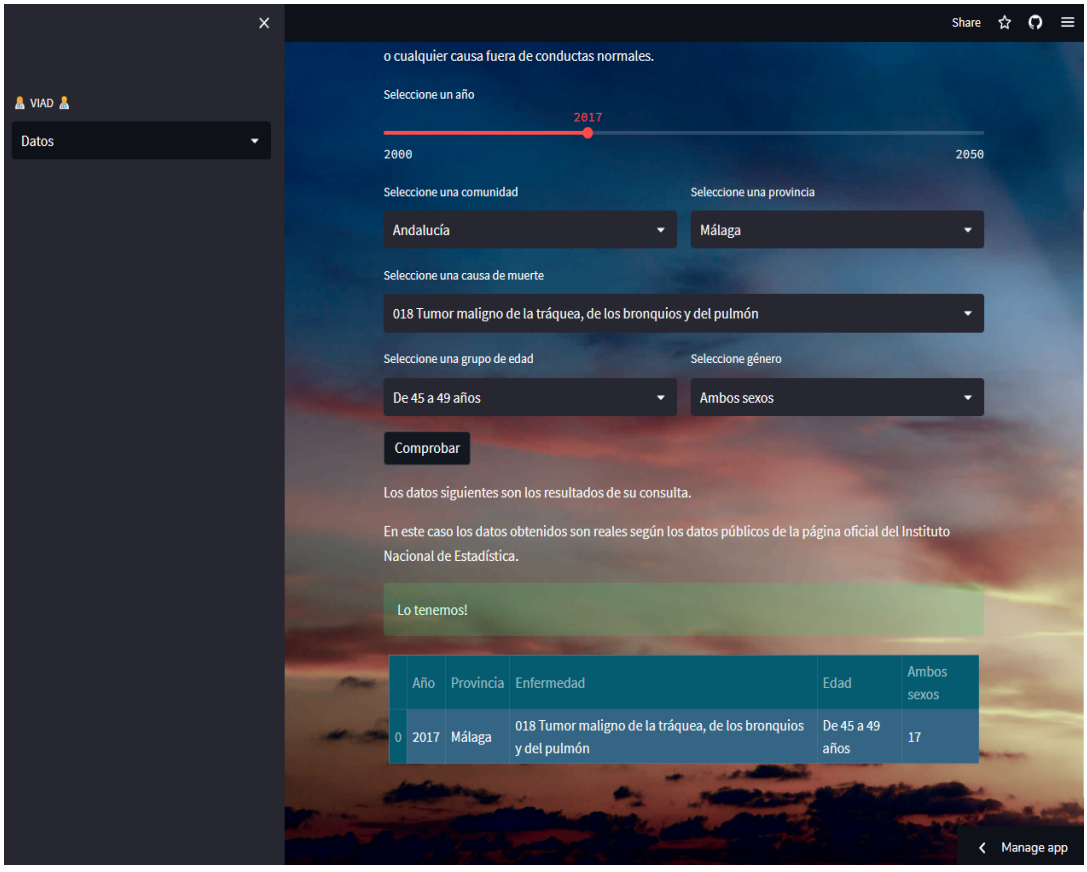
11º Aplicación Web

Página principal

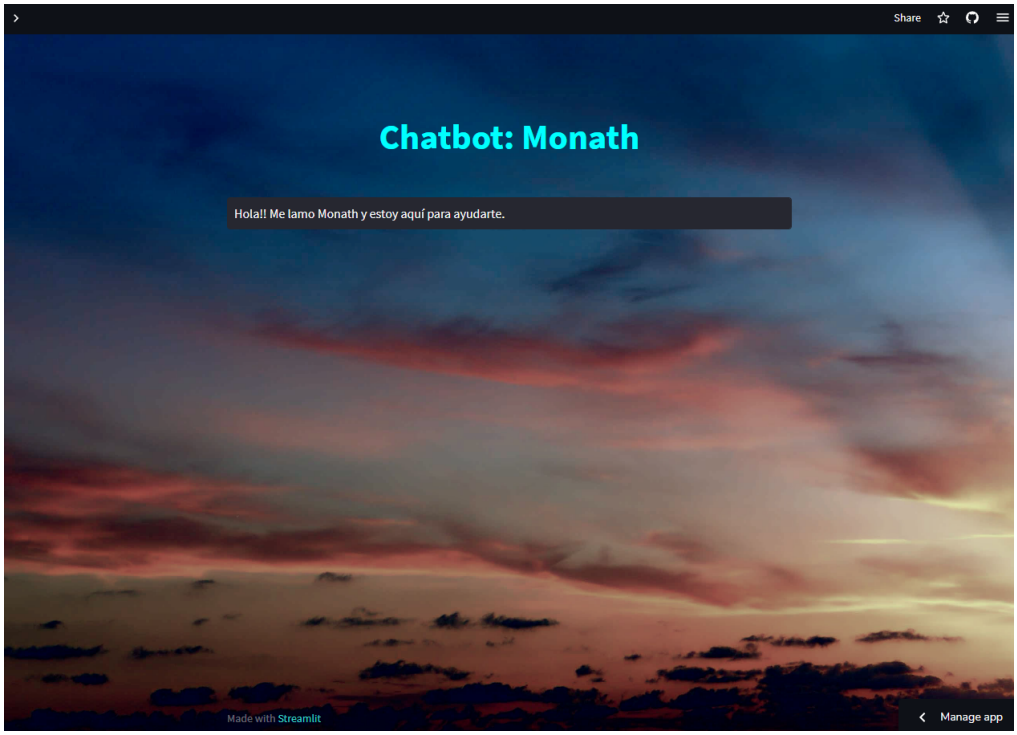
Sección de estadísticas

Consulta de datos

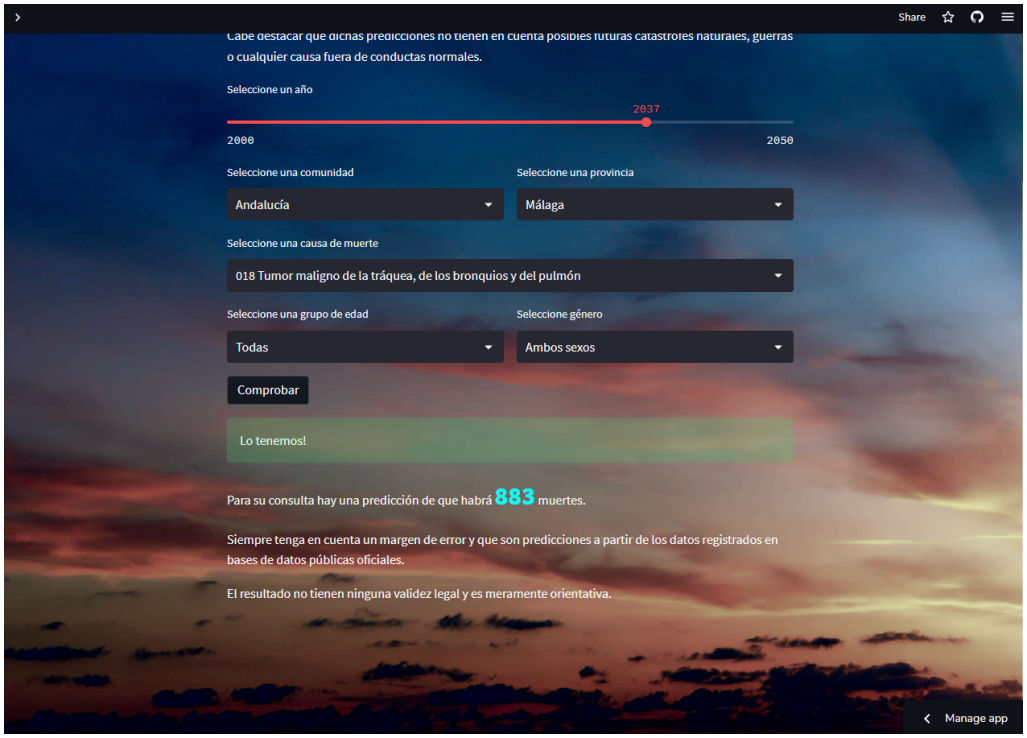




Chatbot Monath



Predicción



12º Conclusiones

A pesar de no haber podido acceder a ciertas estadísticas de tipo mensual para realizar una predicción más concisa en el tiempo hemos llegado a una conclusión.

En el año 2020 con la llegad del Covid19 la mortalidad aumentó drásticamente, pero no únicamente por el propio Covid19, otras causas también aumentaron, puede ser que fueran fallecimientos ocasionados por el propio Covid19 o por debilitamiento de personas que quedaron con las defensas bajas tras el padecimiento de esta enfermedad.

Hemos aprendido también un nuevo algoritmo de entrenamiento llamado ARIMA, es un modelo de series de tiempo muy para analizar y predecir datos de series, como en este caso. Este algoritmo ha sido la base de nuestro modelo, ya que de otra manera no hubiera sido posible.

Si hubiéramos tenido estadísticas mensuales podría haber sido muy útil para hospitales, ya que de esta manera el modelo hubiera podido predecir picos por tipos de fallecimientos y de esa manera preparar a los profesionales de la salud para hacer frente a estas circunstancias.

13º Bibliografía

The Machine Learners
https://www.themachinelearners.com/series-temporales-arima/

Introducción al análisis de series temporales
https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf

Aprendiendo a programar en Python
Paula García, Pedro Salas, Daniel Gutierrez, Ignacio Gonzalez, Mario Javier Durán

Matplotlib
https://matplotlib.org/

ChatGPT
https://chat.openai.com/

INE
https://www.ine.es

Statsmodels
https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html

Streamlit
https://streamlit.io/docs/