Tarea 1
Matemáticas para la Ciencia de Datos

Luis Fernando Izquierdo Berdugo

12 de Agosto del 2024

Durante los últimos 65 años, la población en México ha crecido poco más de cuatro veces. En 1950 había 25.8 millones de personas y en 2015 la población llegó a 119.5 millones. La tabla siguiente muestra la población de nuestro país en el período de 1950 a 2015 de acuerdo al Instituto Nacional de Estadística y Geografía (INEGI).

Año	1950	1960	1970	1980	1990	2000	2010	2015
Población	25.8	34.9	48.2	66.8	81.2	97.5	112.3	119.5

Actividades a realizar:

1. Establecer un modelo de crecimiento poblacional basado en el Modelo Logístico utilizando los datos de 3 distintos años de la tabla.
2. Crear una tabla para comparar la población real con los valores pronosticados por el modelo.
3. Calcular el porcentaje de error para cada par de datos.
4. Dar una conclusión acerca del modelo propuesto y su efectividad.

1. Inciso 1

Establecer un modelo de crecimiento poblacional basado en el Modelo Logístico utilizando los datos de 3 distintos años de la tabla.

$$\frac{dP}{dt} = kP(1 - \frac{P}{K}) \tag{1}$$

- 'P(t)' representa la cantidad de población o la variable de interés en el tiempo t,
- 'K' se le conoce como la barrera poblacional o capacidad de carga, es decir, es la máxima población sostenida por el ambiente,

- ‘k‘ es la tasa de crecimiento y representa la tasa intrínseca de crecimiento de la población cuando la población es pequeña y no hay limitaciones de recursos.

Integrando por fracciones parciales, obtenemos:

$$P(t) = \frac{K}{1 + Ce^{-kt}} \tag{2}$$

Para ejecutar este modelo en Python, primero vamos a importar datos, para crear el modelo de crecimiento, se escogieron los datos de los años 1950, 1970 y 2010, sin embargo, se usarán los datos con relación a nuestra fecha inicial, por lo que se le restará 1950 quedando 0, 20 y 60 respectivamente.

Año	1950	1960	1970	1980	1990	2000	2010	2015
Año (relación 1950)	0	10	20	30	40	50	60	65
Población	25.8	34.9	48.2	66.8	81.2	97.5	112.3	119.5

Se usarán las librerías ”pandas”, ”numpy”, ”matplotlibz ”scipy”.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

data = {'fecha': [0, 20, 60],
        'poblacion': [25.8, 48.2, 112.3]}

df = pd.DataFrame(data)
```

El código anterior da como resultado el dataframe con los datos

	fecha	poblacion
0	0	25.8
1	20	48.2
2	60	112.3

Figura 1: Dataframe con los datos para generación del modelo

A continuación, se hará la definición de la función logística, cambiaremos la variable k (minúscula) por la variable r, esta tendrá la misma función y solamente se cambiará la representación para evitar problemas al ejecutar correctamente el código.

$$P(t) = \frac{K}{1 + Ce^{-rt}} \tag{3}$$

```python
def funcion_logistica(t, K, r, C):
  """
  Parámetros:
    t: tiempo (años)
    K: barrera poblacional
    r: tasa de crecimiento
    C = Constante
  """
  return K / ( 1 + C * (np.exp(-(r*t))) )
```

Lo siguiente será crear una función para encontrar los valores óptimos de los parámetros del modelo (K, r y C). Esta función tomará de entrada el dataframe creado con los datos y los dividirá en fecha y población en $xdata$ e $ydata$ respectivamente.

Lo siguiente sería establecer valores iniciales de K, r y C como una estimación inicial, para posteriormente usar la función çurve fit"de scipy que realiza el ajuste no lineal, esta función compara las predicciones del modelo logístico con los datos reales y ajusta los parámetros para que se minimice el error. Esto devolvera los parametros optimizados.

```python
def optimizar_valores(df):

  xdata = df['fecha']
  ydata = df['poblacion']

  p0 = [148, 0.5, 15] #Estimación inicial

  p_opt, pcov = curve_fit(funcion_logistica, xdata, ydata, p0=p0)

  return p_opt

params = optimizar_valores(df)
K, r, C = params
```

```
170.82067130358317
0.039638260255862166
5.6209562520768666
```

Figura 2: Output optimizado de K, r y C respectivamente

Con esto obtenemos los datos de K, r y C. Sustituyendo en el modelo inicial:

$$P(t) = \frac{170.82}{1 + 5.62e^{-0.0396t}} \tag{4}$$

Si evaluamos para varios casos, obtenemos:

```
test = funcion_logistica(0, *params)
print("Poblacion en 1950: %s" % test)

test = funcion_logistica(40, *params)
print("Poblacion en 1990: %s" % test)

test = funcion_logistica(60, *params)
print("Poblacion en 2010: %s" % test)

test = funcion_logistica(65, *params)
print("Poblacion en 2015: %s" % test)
```

```
Poblacion en 1950: 25.8
Poblacion en 1990: 79.40007890581565
Poblacion en 2010: 112.30000000000003
Poblacion en 2015: 119.6708162078329
```

Figura 3: Output de la evaluación

Graficamos el modelo incluyendo datos futuros para observar la forma de la curva

```
def graficar_modelo(df, params):

  xdata = df['fecha']
  ydata = df['poblacion']

  t = np.linspace(xdata.min(), (200), 100)
  y_pred = funcion_logistica(t, *params)

  plt.plot(xdata, ydata, 'o', label='Datos')
  plt.plot(t, y_pred, '-', label='Modelo logístico')
  plt.xlabel('Año')
  plt.ylabel('Poblacion (millones)')
  plt.legend()
  plt.show()

graficar_modelo(df, params)
```

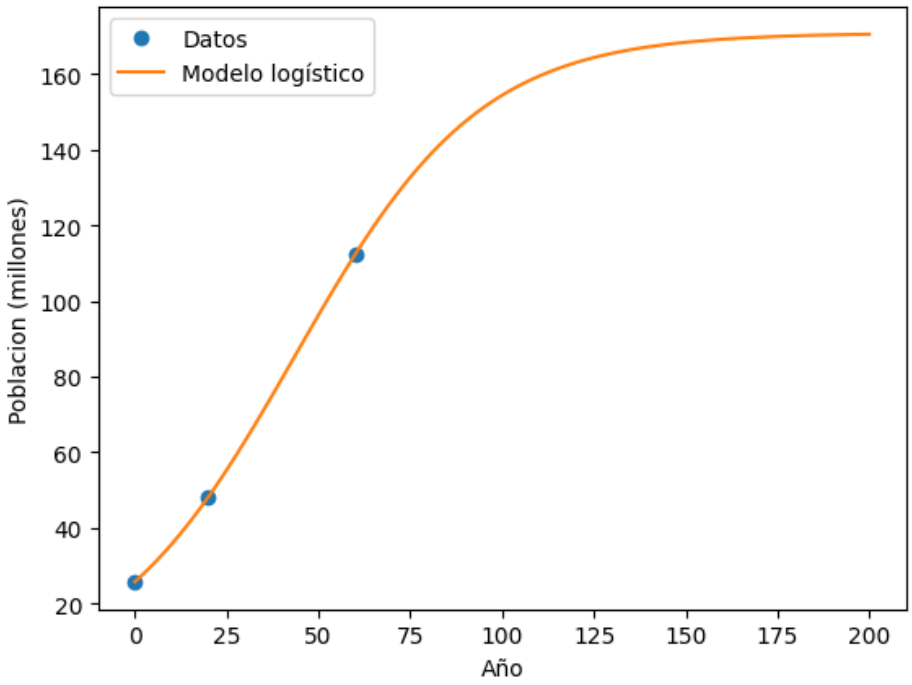

Figura 4: Gráfica del modelo

2. Inciso 2

Crear una tabla para comparar la población real con los valores pronosticados por el modelo.
Primero se crearán los datos y se guardarán en un nuevo Dataframe de pandas.

```
data_modelo = {'fecha': [0,10,20,30,40,50,60,65],
        'poblacion': [funcion_logistica(0, *params),
                      funcion_logistica(10, *params),
                      funcion_logistica(20, *params),
                      funcion_logistica(30, *params),
                      funcion_logistica(40, *params),
                      funcion_logistica(50, *params),
                      funcion_logistica(60, *params),
                      funcion_logistica(65, *params)]}

df_modelo = pd.DataFrame(data_modelo)
```

	fecha	poblacion
0	0	25.800000
1	10	35.725375
2	20	48.200000
3	30	62.999232
4	40	79.400079
5	50	96.258803
6	60	112.300000
7	65	119.670816

Figura 5: Output del dataframe con todos los datos

Con esta información ya podemos hacer la comparación entre los datos obtenidos y los reales.

Año	1950	1960	1970	1980	1990	2000	2010	2015
Población real	25.8	34.9	48.2	66.8	81.2	97.5	112.3	119.5
Población Modelo	25.8	35.7253	48.2	62.9992	79.4	96.2588	112.3	119.6708

3. Inciso 3

Calcular el porcentaje de error para cada par de datos.
Para calcular el porcentaje, se usará la siguiente fórmula:

$$Error = |\frac{V_{real} - V_{predicho}}{V_{real}}| * 100 \tag{5}$$

Entonces, para cada par de datos tenemos:

Año	1950	1960	1970	1980	1990	2000	2010	2015
Población real	25.8	34.9	48.2	66.8	81.2	97.5	112.3	119.5
Población Modelo	25.8	35.7253	48.2	62.9992	79.4	96.2588	112.3	119.6708
Error	0 %	2.36 %	0 %	5.69 %	2.21 %	1.27 %	0 %	0.1429 %

4. Inciso 4

Dar una conclusión acerca del modelo propuesto y su efectividad

$$P(t) = \frac{170.82}{1 + 5.62e^{-0.0396t}} \tag{6}$$

De manera inicial, el modelo propuesto captura bien el crecimiento poblacional, sin embargo, podemos observar que se presentan discrepancias en los datos que no fueron tomados para generar el modelo. Esta diferencia es mínima en algunos casos, obteniendo un error promedio de 1.46 %, pero se nota más en el dato de 1980, que presenta un 5.69 %.

A pesar de las desviaciones se puede observar (Figura 4) que el modelo captura la tendencia general del crecimiento de la población y sigue la curva característica de los modelos logísticos de crecimiento poblacional.

Sería interesante analizar un modelo que pueda tener en cuenta los factores externos que podrían afectar en el crecimiento de la población como lo podrían ser enfermedades, fenómenos naturales (como huracanes, tsunamis), imposiciones gubernamentales, etc.

De manera personal, considero que la tarea me sirvió para tener una idea concisa de los modelos de crecimiento, así como saber desarrollarlos en Python (cosa que nunca había hecho previamente), lo que más me llamó la atención fue la optimización de los valores en la ecuación por medio de código, ya que ahorra muchísimo tiempo y es muy eficaz.

Referencias

[1] Delgado, B. (2024). Clase 1 - Unidad 1: Modelando Cambios

[2] Strang, G., & Herman, E. (2022, March 24). Cálculo volumen 2. OpenStax. https://openstax.org/books/c %C3 %A1lculo-volumen-2/pages/4-4-la-ecuacion-logistica

[3] Google. (2024). Gemini (Aug 10 version) [Large language model]. https://gemini.google.com/app