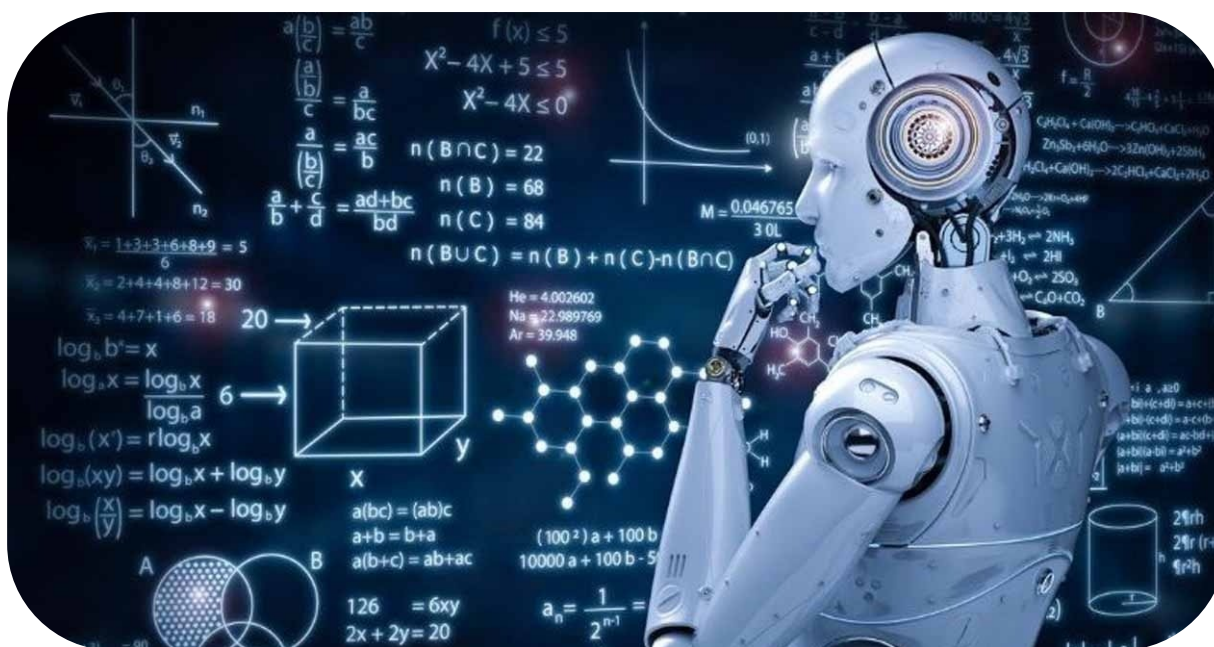




INSTRITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN



Alumno: Jovany Jair Soria Ortega

Email: jovanyjair@gmail.com

Introducción a CHATGPT con Python

Práctica 102

Prof: Alan Badillo Salas

Agosto 2023

INTRODUCCIÓN.

En esta práctica se comenzarán a realizar ejercicios donde se podrá observar la aplicación del uso de Chat GPT en el ámbito de la programación como es Python; Podremos ver una guía paso a paso de las solicitudes realizadas y como la inteligencia artificial interpreta la instrucción para entregar un código de programación listo para ser utilizado, ahorrando así tiempo en la codificación de tareas y realizar la programación con el lenguaje de Python.

Esta utilidad de la inteligencia artificial es muy frutífera en las diferentes aplicaciones industriales y comerciales, porque al acceder a esta herramienta de programación, sencillamente podemos crear soluciones innovadoras, como nuevas aplicaciones y nuevas plataformas de atención a clientes, inventarios en el comercio internacional de mercancías, diseño de flujos de trabajo, encuestas de cultura organizacional, y análisis de bases de datos con textos por tan solo mencionar algunas funcionalidades de la bastedad en su aplicación .

JUSTIFICACIÓN

La programación ha facilitado la creación de nuevos productos y servicios a nivel empresarial, así como la solución de necesidades que antes podrían ser muy complicadas de realizar. Para nosotros como usuarios, un ejemplo claro es el proceso de compra de un vuelo de avión, imagínate que el aeropuerto más cercano se encuentra a una gran distancia de nuestra ubicación; en el pasado, teníamos que recurrir en llamar a un centro de atención telefónica y esperar horas para adquirir nuestro vuelo, con la incertidumbre de que la aerolínea cambiaría el horario de vuelo, tampoco contábamos con una confirmación física o digital al momento de hacer la compra; este era el método tradicional antes de la existencia de las plataformas web o aplicaciones programadas con lenguajes como Python.

Sin embargo, gracias a las nuevas tecnologías y la generación de aplicaciones intuitivas para clientes y software de seguimiento, ahora podemos comprar un vuelo de avión en cuestión de segundos de manera segura, permitiéndonos realizar viajes urgentes o por negocios. Este ejemplo es solo una muestra de las infinitas utilidades que la programación ha creado; el reto actual es simplificarlo tanto para el programador como para el usuario, sin olvidar mantener los presupuestos económicos para su desarrollo.

Es aquí donde Chat GPT como inteligencia artificial, se convierte en un potente motor de programación capaz de lograr avances significativos a una velocidad nunca antes vista, satisfaciendo así las nuevas necesidades que el mercado y la humanidad demandan para su desarrollo.

DESARROLLO DE LA PRÁCTICA

P102.1 - Reporte estadístico

.1 Solicita a CHATGPT que genere el código para abrir un archivo weight-height.csv desde la URL <https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv> usando **Pandas**

```
[ ] import pandas as pd
import requests

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Ahora puedes trabajar con el DataFrame 'df' que contiene los datos del archivo CSV.
# Por ejemplo, puedes imprimir las primeras filas del DataFrame:
print(df.head())
```

```
Archivo CSV cargado correctamente en el DataFrame.
  Gender  Height  Weight
0  Male   73.847017  241.893563
1  Male   68.781904  162.310473
2  Male   74.110105  212.740856
3  Male   71.730978  220.042470
4  Male   69.881796  206.349801
```

.2 Solicita a CHATGPT que genere el código que imprima 10 valores aleatorios del dataset anterior usando la función sample de **Pandas**

```
[ ] import pandas as pd
import requests

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Imprimir 10 valores aleatorios del DataFrame utilizando la función sample
random_sample = df.sample(n=10)
print(random_sample)
```

```
Archivo CSV cargado correctamente en el DataFrame.
  Gender  Height  Weight
4048  Male   64.805303  147.363350
1465  Male   70.732060  197.811882
5224  Female  64.318790  136.113112
1333  Male   67.796214  184.862913
7394  Female  60.143322  118.797659
2995  Male   73.075345  211.114810
7718  Female  69.125275  165.830008
5176  Female  62.028373  103.275029
3105  Male   68.489164  182.331487
8785  Female  61.508429   98.331546
```

.3 Solicita a CHATGPT que genere el código que describa la columna Gender que es categórica, pide que extraiga los valores únicos y sus conteos

```
[2] import pandas as pd
import requests

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Describir la columna "Gender" que es categórica
gender_description = df['Gender'].describe()

# Extraer los valores únicos y sus conteos
gender_counts = df['Gender'].value_counts()

print("Descripción de la columna 'Gender':")
print(gender_description)

print("\nValores únicos de 'Gender' y sus conteos:")
print(gender_counts)
```

Archivo CSV cargado correctamente en el DataFrame.

Descripción de la columna 'Gender':

count 10000

unique 2

top Male

freq 5000

Name: Gender, dtype: object

Valores únicos de 'Gender' y sus conteos:

Male 5000

Female 5000

Name: Gender, dtype: int64

.4 Solicita a CHATGPT que genere el código que describa la columna Height que es numérica y obtenga los estadísticos: total, mínimo, máximo, promedio, desviación estándar, intervalo de confianza al 95%, y los cuartiles

```
[ ] url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Describir la columna "Height" que es numérica
height_description = df['Height'].describe()

# Calcular el intervalo de confianza al 95%
height_mean = df['Height'].mean()
height_std = df['Height'].std()
height_sample_size = df['Height'].count()
confidence_interval = stats.t.interval(0.95, height_sample_size - 1, loc=height_mean, scale=height_std)

# Calcular los cuartiles
height_quartiles = df['Height'].quantile([0.25, 0.5, 0.75])

print("Descripción de la columna 'Height':")
print(height_description)

print("\nIntervalo de confianza al 95% para 'Height':")
print(f"Intervalo: {confidence_interval}")

print("\nCuartiles de 'Height':")
print(height_quartiles)
```

```
Archivo CSV cargado correctamente en el DataFrame.
Descripción de la columna 'Height':
```

```
count    10000.000000
mean       66.367560
std         3.847528
min        54.263133
25%        63.505620
50%        66.318070
75%        69.174262
max        78.998742
Name: Height, dtype: float64
```

```
Intervalo de confianza al 95% para 'Height':
Intervalo: (58.825630270662856, 73.90948923897963)
```

```
Cuartiles de 'Height':
0.25    63.505620
0.50    66.318070
0.75    69.174262
Name: Height, dtype: float64
```

.5 Solicita a CHATGPT que genere el código que describa la columna Weight que es numérica y obtenga los estadísticos: total, mínimo, máximo, promedio, desviación estándar, intervalo de confianza al 95%, y los cuartiles

```
[ ] import pandas as pd
import requests
import scipy.stats as stats

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Describir la columna "Weight" que es numérica
weight_description = df['Weight'].describe()

# Calcular el intervalo de confianza al 95%
weight_mean = df['Weight'].mean()
weight_std = df['Weight'].std()
weight_sample_size = df['Weight'].count()
confidence_interval = stats.t.interval(0.95, weight_sample_size - 1, loc=weight_mean, scale=weight_std)

# Calcular los cuartiles
weight_quartiles = df['Weight'].quantile([0.25, 0.5, 0.75])

print("Descripción de la columna 'Weight':")
print(weight_description)

print("\nIntervalo de confianza al 95% para 'Weight':")
print(f"Intervalo: {confidence_interval}")

print("\nCuartiles de 'Weight':")
print(weight_quartiles)
```

Archivo CSV cargado correctamente en el DataFrame.

Descripción de la columna 'Weight':

| | |
|-------|--------------|
| count | 10000.000000 |
| mean | 161.440357 |
| std | 32.108439 |
| min | 64.700127 |
| 25% | 135.818051 |
| 50% | 161.212928 |
| 75% | 187.169525 |
| max | 269.989699 |

Name: Weight, dtype: float64

Intervalo de confianza al 95% para 'Weight':

Intervalo: (98.50135411934176, 224.37935954631976)

Cuartiles de 'Weight':

| | |
|------|------------|
| 0.25 | 135.818051 |
| 0.50 | 161.212928 |
| 0.75 | 187.169525 |

Name: Weight, dtype: float64

.6 Solicita a CHATGPT que genere el código que grafique el histograma de la columna Height usando **Seaborn**

```
[ ] import pandas as pd
import requests
import seaborn as sns
import matplotlib.pyplot as plt

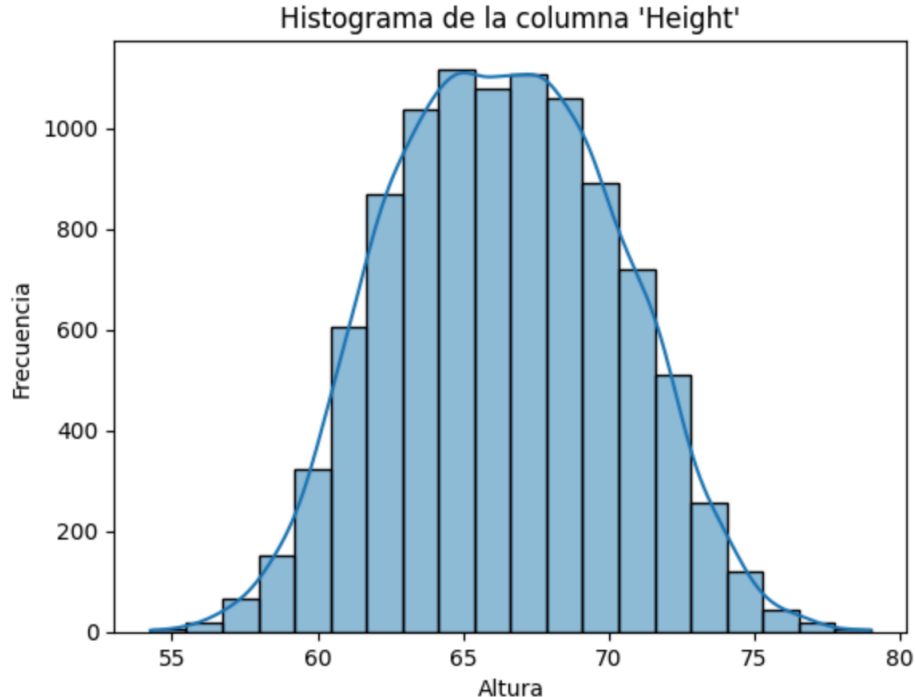
url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Graficar el histograma de la columna "Height" usando Seaborn
sns.histplot(data=df, x='Height', bins=20, kde=True)
plt.title("Histograma de la columna 'Height'")
plt.xlabel("Altura")
plt.ylabel("Frecuencia")
plt.show()
```

Archivo CSV cargado correctamente en el DataFrame.



.7 Solicita a CHATGPT que genere el código que grafique el histograma de la columna Weight usando **Seaborn**

```
[ ] import pandas as pd
import requests
import seaborn as sns
import matplotlib.pyplot as plt

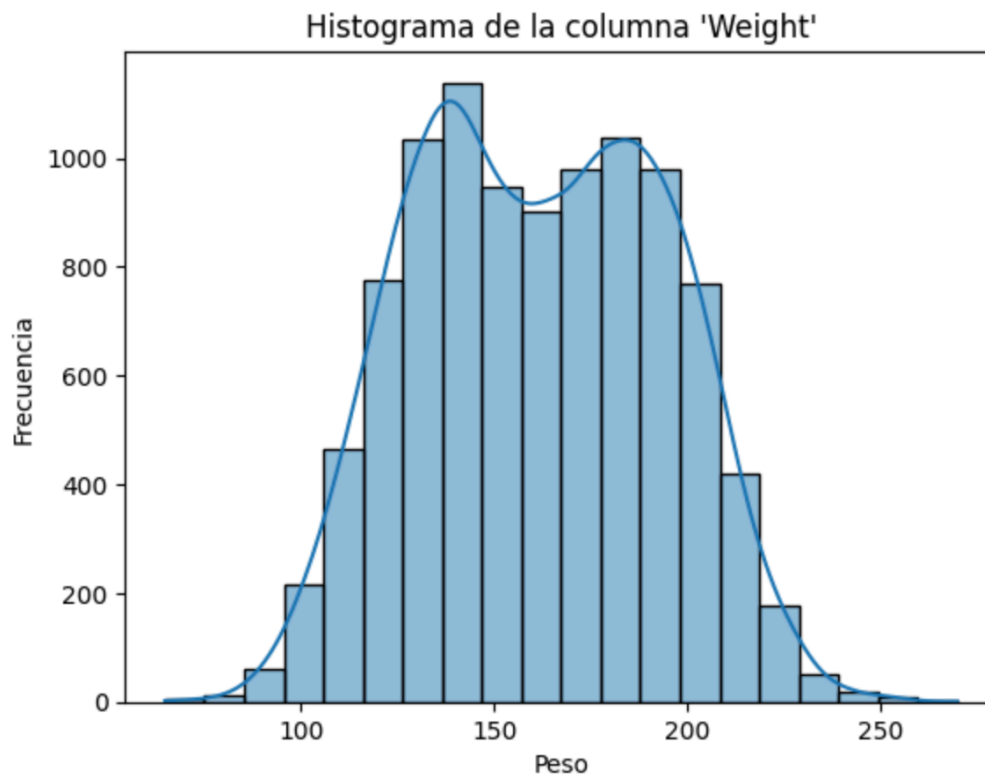
url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Graficar el histograma de la columna "Weight" usando Seaborn
sns.histplot(data=df, x='Weight', bins=20, kde=True)
plt.title("Histograma de la columna 'Weight'")
plt.xlabel("Peso")
plt.ylabel("Frecuencia")
plt.show()
```

➡ Archivo CSV cargado correctamente en el DataFrame.



.8 Solicita a CHATGPT que genere el código que grafique la correlación entre las columnas Height y Weight usando **Seaborn**

```
[ ] import pandas as pd
import requests
import seaborn as sns
import matplotlib.pyplot as plt

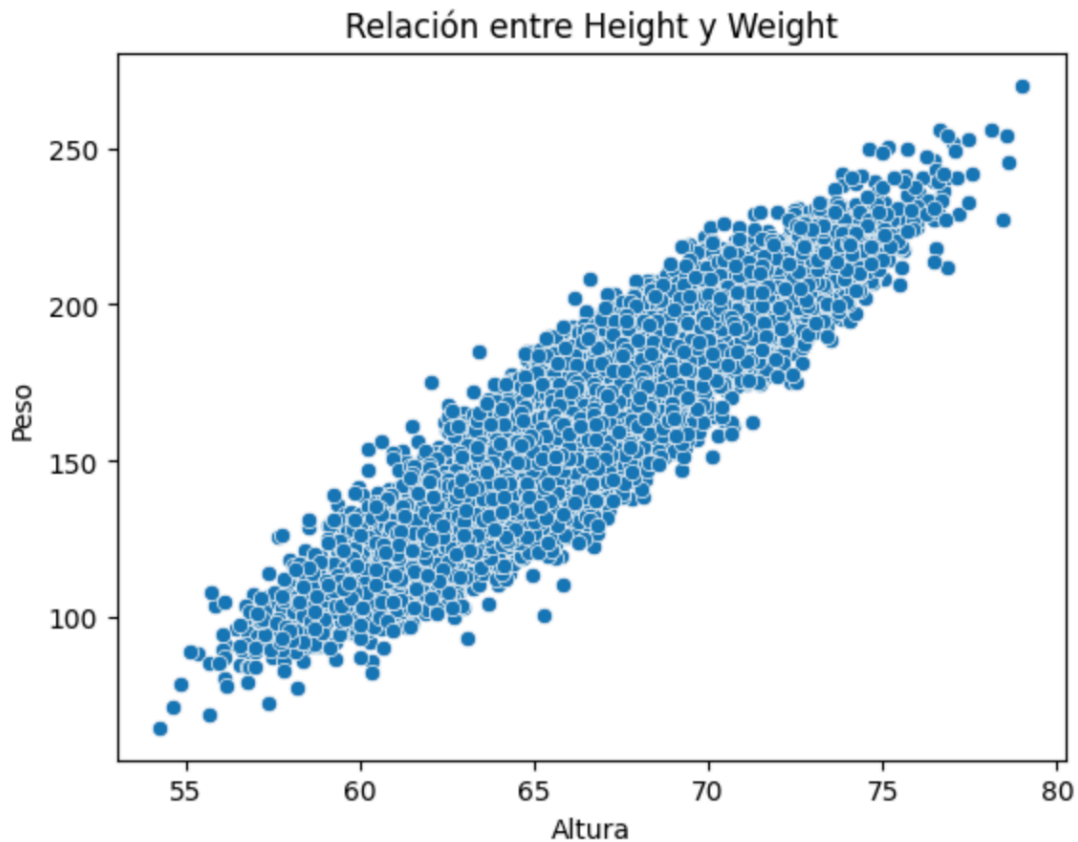
url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Graficar la relación entre las columnas "Height" y "Weight" usando Seaborn
sns.scatterplot(data=df, x='Height', y='Weight')
plt.title("Relación entre Height y Weight")
plt.xlabel("Altura")
plt.ylabel("Peso")
plt.show()
```

➡ Archivo CSV cargado correctamente en el DataFrame.



.9 Solicita a CHATGPT que genere el código que obtenga la correlación entre las columnas Height y Weight usando corr de **Numpy**

```
[ ] import pandas as pd
import requests
import numpy as np

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Calcular la correlación entre las columnas "Height" y "Weight" usando NumPy
height = df['Height']
weight = df['Weight']
correlation = np.corrcoef(height, weight)[0, 1]

print(f"Correlación entre 'Height' y 'Weight': {correlation}")
```

Archivo CSV cargado correctamente en el DataFrame.
Correlación entre 'Height' y 'Weight': 0.9247562987409146

.10 Solicita a CHATGPT que genere el código que reporte el valor promedio de la columna Height donde Gender es Male y donde Gender es Female y repita lo mismo para la columna Weight

```
[ ] import pandas as pd
import requests

url = "https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/weight-height.csv"

# Descargar el archivo desde la URL
response = requests.get(url)

# Verificar si la descarga fue exitosa (código 200 significa éxito)
if response.status_code == 200:
    # Crear un DataFrame con los datos del archivo CSV
    df = pd.read_csv(url)
    print("Archivo CSV cargado correctamente en el DataFrame.")
else:
    print("Error al descargar el archivo CSV.")

# Calcular el valor promedio de la columna "Height" para cada género
mean_height_by_gender = df.groupby('Gender')['Height'].mean()

# Calcular el valor promedio de la columna "Weight" para cada género
mean_weight_by_gender = df.groupby('Gender')['Weight'].mean()

print("Valor promedio de 'Height' para cada género:")
print(mean_height_by_gender)

print("\nValor promedio de 'Weight' para cada género:")
print(mean_weight_by_gender)
```

Archivo CSV cargado correctamente en el DataFrame.
Valor promedio de 'Height' para cada género:
Gender
Female 63.708774
Male 69.026346
Name: Height, dtype: float64

Valor promedio de 'Weight' para cada género:
Gender
Female 135.860093
Male 187.020621
Name: Weight, dtype: float64

P102.2 - Extracción de Información

.1 Solicita a CHATGPT que genere el código que extraiga el texto de una imagen PNG desde la URL https://raw.githubusercontent.com/dragonnomada/chat-gpt-2023/main/datasets/UITD_PAPER.png usando OCR en Python

```
[ ] !pip install pytesseract
    !pip install Pillow
```

```
Collecting pytesseract
  Downloading pytesseract-0.3.10-py3-none-any.whl (14 kB)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from pytesseract)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from pytesseract)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.10
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
```

```
[ ] !sudo apt-get update
    !sudo apt-get install tesseract-ocr
```

```
▶ !sudo apt-get update
   !sudo apt-get install tesseract-ocr
```

Mostra
enca

```
⇒ Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,626 B]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 InRelease [1,581 B]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:7 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ Packages [43.3 kB]
Get:8 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 Packages [456 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,103 kB]
Hit:10 https://ppa.launchpadcontent.net/c2d4u.team/c2d4u4.0+/ubuntu jammy InRelease
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,235 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [25.6 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [49.2 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [979 kB]
Hit:15 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [833 kB]
Hit:17 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Hit:18 https://ppa.launchpadcontent.net/ubuntuugis/ppa/ubuntu jammy InRelease
Fetched 5,068 kB in 2s (2,480 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```

The following additional packages will be installed:
  tesseract-ocr-eng tesseract-ocr-osd
The following NEW packages will be installed:
  tesseract-ocr tesseract-ocr-eng tesseract-ocr-osd
0 upgraded, 3 newly installed, 0 to remove and 30 not upgraded.
Need to get 4,816 kB of archives.
After this operation, 15.6 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr-eng all 1:4.00~git30-7274cfa
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr-osd all 1:4.00~git30-7274cfa
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tesseract-ocr amd64 4.1.1-2.1build1 [236 k
Fetched 4,816 kB in 1s (9,129 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package tesseract-ocr-eng.
(Reading database ... 120500 files and directories currently installed.)
Preparing to unpack .../tesseract-ocr-eng_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr-osd.
Preparing to unpack .../tesseract-ocr-osd_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr.
Preparing to unpack .../tesseract-ocr_4.1.1-2.1build1_amd64.deb ...
Unpacking tesseract-ocr (4.1.1-2.1build1) ...
Setting up tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr (4.1.1-2.1build1) ...
Processing triggers for man-db (2.10.2-1) ...

```

.2 Solicita a CHATGPT que traduzca el texto a español (pega el texto extraído en <https://chat.openai.com/>)

"Ahorrando tiempo con el Editor de Diagramas de Transición de Interfaz de Usuario (UITD)"

"El Editor UITD: Una herramienta para simplificar la modelación de interacción usuario-sistema"

"Optimizando el diseño de la interfaz de usuario con el Editor UITD: Un estudio comparativo"

"Editor UITD: Simplificando la modelación de interacción usuario-sistema para desarrolladores"

¿El editor de Diagramas de Transición de Interfaz de Usuario (UITD) puede ahorrar tiempo a los usuarios?

Cervantes-Ojeda J., Badillo-Salas A., Gomez-Fuentes M.C.

Departamento de Matemáticas Aplicadas y Sistemas

Universidad Autónoma Metropolitana

Ciudad de México, México

jcervantes@cua.uam.mx, dragonnomadal23@gmail.com, mgomez@cua.uam.mx

Resumen— El Diagrama de Transición de Interfaz de Usuario (UITD) es una notación de modelado formal que simplifica la especificación y el diseño de las interacciones usuario-sistema. Es una valiosa herramienta de comunicación para las partes interesadas técnicas y no técnicas durante la fase de obtención de requisitos, ya que proporciona una notación simple y completa que es fácil de entender. En este artículo, investigamos la eficiencia de la creación de UITDs utilizando draw.io, un software de diagramación ampliamente utilizado, en comparación con un editor dedicado de UITD. Realizamos un estudio para comparar el tiempo necesario para completar la tarea, así como la facilidad de uso y satisfacción subjetiva de los participantes. Nuestros resultados muestran que el editor UITD es más eficiente y preferido por los participantes, resaltando la importancia de utilizar herramientas especializadas para crear modelos formales como UITDs. Los hallazgos de este estudio tienen implicaciones para los desarrolladores de software, diseñadores y otras partes interesadas involucradas en la especificación y el diseño de interacciones usuario-sistema.

Eficiencia del Editor; Notación de Modelado; Obtención de Requisitos

I. INTRODUCCIÓN

El Diagrama de Transición de Interfaz de Usuario (UITD) es una notación de modelado formal que simplifica la especificación y el diseño de las interacciones usuario-sistema. Puede utilizarse para modelar el flujo de interfaces de usuario que tendrá el sistema, convirtiéndose en una valiosa herramienta de comunicación para las partes interesadas técnicas y no técnicas durante la fase de obtención de requisitos. Además, es técnicamente preciso, lo que permite utilizarlo de manera confiable para iniciar el desarrollo del sistema [1].

La simplicidad es clave al trabajar con notaciones de modelado, especialmente para los no expertos [3]. El UITD proporciona una notación que es completa y lo suficientemente simple como para que las partes interesadas no técnicas la entiendan [2]. Es una notación de modelado formal que es fácil de aprender, facilitando la comunicación entre partes interesadas con y sin habilidades técnicas en desarrollo de software. En comparación con otras herramientas de modelado formal, el UITD tiene varias ventajas, incluyendo su simplicidad, completitud y capacidad para modelar interacciones usuario-sistema de manera clara y concisa. También se proporciona evidencia empírica sobre la comprensibilidad del UITD por parte de las partes interesadas no técnicas [2]. Al utilizar el UITD, los diseñadores y desarrolladores pueden crear modelos que sean técnicamente precisos y fáciles de entender, lo que ayuda a garantizar que todas las partes interesadas estén alineadas y satisfechas con la especificación de los requisitos del sistema.

En los últimos años, se han desarrollado varias herramientas para apoyar la creación de diagramas generalizados. Una de estas herramientas es draw.io, que es un conocido software de diagramación de código abierto. Draw.io se utiliza ampliamente para crear diagramas, diagramas de flujo y otros tipos de representaciones gráficas. Si bien draw.io es una herramienta versátil, puede que no esté diseñada específicamente para manejar los requisitos de creación de UITDs. Esto plantea la pregunta de qué tan fácil es crear UITDs utilizando draw.io, y si es más eficiente utilizar una herramienta dedicada como el editor UITD [4].

Para investigar esta pregunta, llevamos a cabo un estudio en el que se pidió a los participantes que crearan UITDs utilizando tanto draw.io como el editor UITD. Nuestro estudio tuvo como objetivo comparar la eficiencia de las dos herramientas en

.3 Solicita a CHATGPT que te diga quiénes son los autores del artículo

Pront: ¿Quiénes son los autores del artículo?

Respuesta Chat GPT

Los autores del artículo son:

1. Cervantes-Ojeda J.
2. Badillo-Salas A.
3. Gomez-Fuentes M.C.

.4 Solicita a CHATGPT que te diga cuáles son los correos relacionados al artículo

Direcciones de correo electrónico:

jcervantes@.cua.uam.mx

dragonnomadal23@gmail.com

mgomez@.cua.uam.mx

.5 Solicita a CHATGPT que te diga quién es la universidad, el departamento, la ciudad y el país del artículo

Departamento de Matemáticas Aplicadas y Sistemas

Universidad Autónoma Metropolitana

Ciudad de México, México

.5 Solicita a CHATGPT que te explique el ABSTRACT

Respuesta Chat GPT

El abstract (resumen) del artículo describe brevemente el contenido y los hallazgos del estudio realizado sobre el "User Interface Transition Diagram (UITD) Editor" (Editor de Diagramas de Transición de Interfaz de Usuario). En el resumen, se destacan los siguientes puntos clave:

El User Interface Transition Diagram (UITD) es una notación de modelado formal que simplifica la especificación y el diseño de las interacciones usuario-sistema. Es una herramienta de comunicación valiosa para las partes interesadas técnicas y no técnicas durante la fase de obtención de requisitos. Proporciona una notación simple y completa que es fácil de entender, lo que facilita la comunicación entre las partes interesadas con y sin habilidades técnicas en desarrollo de software.

El artículo investiga la eficiencia de crear UITDs utilizando draw.io, un software de diagramación ampliamente utilizado, en comparación con un editor especializado de UITD. Se llevó a cabo un estudio comparativo para evaluar el tiempo requerido para completar la tarea y la facilidad de uso y satisfacción subjetiva de los participantes.

Los resultados del estudio demuestran que el editor UITD es más eficiente y preferido por los participantes en comparación con el uso de draw.io. Esto destaca la importancia de utilizar herramientas especializadas para crear modelos formales como los UITDs.

Los hallazgos del estudio tienen implicaciones significativas para los desarrolladores de software, diseñadores y otras partes interesadas involucradas en la especificación y el diseño de interacciones usuario-sistema. El uso del editor UITD puede mejorar la eficiencia y la comprensión en la creación de modelos formales, lo que garantiza que todas las partes interesadas estén alineadas y satisfechas con la especificación de los requisitos del sistema.

En resumen, el abstract proporciona una visión general concisa de los objetivos, el enfoque y los resultados clave del estudio sobre el UITD Editor y su eficiencia en comparación con otras herramientas de diagramación.

.6 Solicita a CHATGPT que te explique la introducción

Respuesta Chat GPT

La introducción del artículo presenta el contexto y la importancia del "User Interface Transition Diagram (UITD)" (Diagrama de Transición de Interfaz de Usuario) como una notación de modelado formal para simplificar la especificación y el diseño de interacciones usuario-sistema. A continuación, se proporciona una explicación de los puntos clave de la introducción:

1. Importancia del UITD: Se destaca que el UITD es una notación valiosa y técnica que facilita la comunicación entre las partes interesadas técnicas y no técnicas durante la fase de obtención de requisitos. Permite modelar el flujo de las interfaces de usuario que tendrá el sistema, lo que lo convierte en una herramienta esencial para la especificación y el diseño de interacciones entre usuarios y sistemas.
2. Simplicidad y Comprensibilidad: La introducción enfatiza que el UITD proporciona una notación completa y simple, lo que facilita su comprensión incluso para las partes interesadas que no tienen habilidades técnicas en desarrollo de software. Esta característica es esencial cuando se trabaja con notaciones de modelado, especialmente para aquellos que no son expertos en el tema.
3. Ventajas del UITD: Se menciona que el UITD ofrece varias ventajas en comparación con otras herramientas de modelado formal, incluida su capacidad para ser técnicamente precisa y fácil de aprender. Esta combinación de precisión técnica y facilidad de uso ayuda a garantizar que todos los involucrados en el desarrollo de sistemas estén alineados y satisfechos con la especificación de los requisitos del sistema.
4. Objetivo del Estudio: Se establece el objetivo del artículo, que es investigar la eficiencia de crear UITDs utilizando draw.io, una herramienta de diagramación ampliamente utilizada, en comparación con un editor especializado de UITD. El estudio busca comparar el tiempo requerido para completar la tarea y la facilidad de uso y satisfacción de los participantes con cada herramienta.

La introducción proporciona una visión general de la relevancia y el alcance del estudio, y establece la motivación detrás de la comparación entre draw.io y el editor UITD. También resalta la importancia de la comprensibilidad y eficiencia al trabajar con notaciones de modelado para garantizar el éxito en la especificación y el diseño de interacciones usuario-sistema.

.7 Solicita a CHATGPT que extraiga las palabras principales

Respuesta Chat GPT

Las palabras clave o principales del artículo pueden identificarse a partir del contenido del resumen y la introducción. Algunas de las palabras clave que se pueden extraer del artículo son:

1. User Interface Transition Diagram (UITD)
2. Editor de Diagramas de Transición de Interfaz de Usuario
3. Modelado formal
4. Especificación
5. Diseño
6. Interacciones usuario-sistema
7. Eficiencia
8. Draw.io
9. Editor UITD
10. Herramientas especializadas
11. Complejidad
12. Simplicidad
13. Comunicación
14. Requisitos
15. Partes interesadas
16. Facilidad de uso
17. Satisfacción
18. Desarrolladores
19. Diseñadores
20. Diagramación

Estas palabras clave resumen los temas y conceptos centrales del artículo sobre el UITD Editor y su eficiencia en comparación con otras herramientas de diagramación, y son útiles para identificar rápidamente el enfoque y contenido del artículo.

.8 Solicita a CHATGPT que genere el código para que reescriba el artículo en formato de Markdown

Respuesta Chat GPT

Saving Time with the User Interface Transition Diagrams (UITD) Editor

Abstract

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It is a valuable communication tool for technical and non-technical stakeholders during the requirements elicitation phase, as it provides a simple and complete notation that is easy to understand. In this paper, we investigate the efficiency of creating UITDs using draw.io, a widely used diagramming software, compared to a dedicated UITD editor. We conducted a study to compare the time required to complete the task, as well as the subjective ease of use and satisfaction of participants. Our results show that the UITD editor is more efficient and preferred by participants, highlighting the importance of using specialized tools for creating formal models such as UITDs. The findings of this study have implications for software developers, designers, and other stakeholders involved in the specification and design of user-system interactions.

Keywords: Editor Efficiency; Modelling notation; Requirements elicitation

I. INTRODUCTION

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It can be used to model the flow of user interfaces that the system will have, making it a valuable communication tool for technical and non-technical stakeholders during the requirements elicitation phase. Additionally, it is technically accurate, allowing it to be used reliably to start the development of the system [1].

Simplicity is key when working with modeling notations, particularly for non-experts [3]. The UITD provides a notation that is both complete and simple enough for non-technical stakeholders to understand [2]. It is a formal modeling notation that is easy to learn, facilitating communication between stakeholders with and without software development technical skills. Compared to other formal modeling tools, the UITD has several advantages, including its simplicity, completeness, and ability to model user-system interactions in a clear and concise way. Empirical evidence about the UITD's understandability by non-technical stakeholders is also provided in [2]. By using the UITD, designers and developers can create models that are both technically accurate and easy to understand, helping to ensure that all stakeholders are aligned and satisfied with the specification of the system's requirements.

In recent years, various tools have been developed to support the creation of generalized diagrams. One such tool is draw.io, which is a well-known open-source diagramming software. Draw.io is widely used for creating diagrams, flowcharts, and other types of graphical representations. While draw.io is a versatile tool, it may not be specifically designed to handle the requirements of creating UITDs. This raises the question of how easy it is to create UITDs using draw.io, and whether it is more efficient to use a dedicated tool such as the UITD editor [4].

To investigate this question, we conducted a study in which participants were asked to create UITDs using both draw.io and the UITD editor. Our study aimed to compare the efficiency of the two tools in creating UITDs, as well as the subjective ease of use and satisfaction of the participants. The results showed that the UITD editor was more efficient and preferred by participants, highlighting the importance of using specialized tools for creating formal models like UITDs.

The findings of this study have implications for software developers, designers, and other stakeholders involved in the specification and design of user-system interactions. Using the UITD editor can improve efficiency and understanding in creating formal models, ensuring that all stakeholders are aligned and satisfied with the specification of the system's requirements.

II. CONCLUSION

In conclusion, the UITD Editor is a valuable tool for simplifying the modeling of user-system interactions. Its simplicity, completeness, and technical accuracy make it an effective communication tool during the requirements elicitation phase. The study comparing the efficiency of draw.io and the UITD Editor showed that the dedicated UITD Editor was more efficient and preferred by participants. The findings highlight the importance of using specialized tools for creating formal models like UITDs, and have implications for software developers and designers involved in the specification and design of user-system interactions. By using the UITD Editor, stakeholders can create models that are both technically accurate and easy to understand, leading to better alignment and satisfaction with the system's requirements.

.9 Solicita a CHATGPT que genere el código para que reescriba el artículo en forma Latex

Respuesta Chat GPT

- `\documentclass{article}`
- `\title{Saving Time with the User Interface Transition Diagrams (UITD) Editor}`
- `\author{Cervantes-Ojeda J., Badillo-Salas A., Gomez-Fuentes M.C.\\`
- `\small Department of Applied Mathematics and Systems\\`
- `\small Universidad Autónoma Metropolitana\\`
- `\small México City, México\\`
- `\small \texttt{j cervantes@cua.uam.mx, dragonnomadal23@gmail.com, mgomez@cua.uam.mx}}`
- `\date{}`

- `\begin{document}`
- `\maketitle`
- `\begin{abstract}`

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It is a valuable communication tool for technical and non-technical stakeholders during the requirements elicitation phase, as it provides a simple and complete notation that is easy to understand. In this paper, we investigate the efficiency of creating UITDs using draw.io, a widely used diagramming software, compared to a dedicated UITD editor. We conducted a study to compare the time required to complete the task, as well as the subjective ease of use and satisfaction of participants. Our results show that the UITD editor is more efficient and preferred by participants, highlighting the importance of using specialized tools for creating formal models such as UITDs. The findings of this study have implications for software developers, designers, and other stakeholders involved in the specification and design of user-system interactions.

- `\textbf{Keywords:}` Editor Efficiency; Modelling notation; Requirements elicitation
- `\end{abstract}`
- `\section{Introduction}`

The User Interface Transition Diagram (UITD) is a formal modeling notation that simplifies the specification and design of user-system interactions. It can be used to model the flow of user interfaces that the system will have, making it a valuable communication tool for technical and non-technical stakeholders during the requirements elicitation phase. Additionally, it is technically accurate, allowing it to be used reliably to start the development of the system \cite{reference1}.

Simplicity is key when working with modeling notations, particularly for non-experts \cite{reference3}. The UITD provides a notation that is both complete and simple enough for non-technical stakeholders to understand \cite{reference2}. It is a formal modeling notation that is easy to learn, facilitating communication between stakeholders with and without software development technical skills. Compared to other formal modeling tools, the UITD has several advantages, including its simplicity, completeness, and ability to model user-system interactions in a clear and concise way. Empirical evidence about the UITD's understandability by non-technical stakeholders is also provided in \cite{reference2}. By using the UITD, designers and developers can create models that are both technically accurate and easy to understand, helping to ensure that all stakeholders are aligned and satisfied with the specification of the system's requirements.

In recent years, various tools have been developed to support the creation of generalized diagrams. One such tool is draw.io, which is a well-known open-source diagramming software. Draw.io is widely used for creating diagrams, flowcharts, and other types of graphical representations. While draw.io is a versatile tool, it may not be specifically designed to handle the requirements of creating UITDs. This raises the question of how easy it is to create UITDs using draw.io, and whether it is more efficient to use a dedicated tool such as the UITD editor \cite{reference4}.

To investigate this question, we conducted a study in which participants were asked to create UITDs using both draw.io and the UITD editor. Our study aimed to compare the efficiency of the two tools in creating UITDs, as well as the subjective ease of use and satisfaction of the participants. The results showed that the UITD editor was more efficient and preferred by participants, highlighting the importance of using specialized tools for creating formal models like UITDs.

The findings of this study have implications for software developers, designers, and other stakeholders involved in the specification and design of user-system interactions. Using the UITD editor can improve efficiency and understanding in creating formal models, ensuring that all stakeholders are aligned and satisfied with the specification of the system's requirements.

\section{Conclusion}

In conclusion, the UITD Editor is a valuable tool for simplifying the modeling of user-system interactions. Its simplicity, completeness, and technical accuracy make it an effective communication tool during the requirements elicitation phase. The study comparing the efficiency of draw.io and the UITD Editor showed that the dedicated UITD Editor was more efficient and preferred by participants. The findings highlight the importance of using specialized tools for creating formal models like UITDs, and have implications for software developers and designers involved in the specification and design of user-system interactions. By using the UITD Editor, stakeholders can create models that are both technically accurate and easy to understand, leading to better alignment and satisfaction with the system's requirements.

- \begin{thebibliography}{9}
- \bibitem{reference1}
- Author A, Author B. Title of the article. \textit{Journal Name}. Year; Volume(Issue): page numbers.
- \bibitem{reference2}
- Author C, Author D. Title of the article. \textit{Conference Name}. Year; page numbers.
- \bibitem{reference3}
- Author E, Author F. Title of the book. Publisher; Year.
- \bibitem{reference4}
- Author G. Title of the website. URL. Accessed on Month Day, Year.
- \end{thebibliography}
- \end{document}

CONCLUSIÓN DE LA PRÁCTICA

A través de la realización de esta práctica pudimos observar lo fácil que fue realizar un código de programación, con los conocimientos básicos de Python, únicamente solicitando un lenguaje común para el usuario de Chat GPT, solicitando el código requerido y que esperábamos como resultado.

Durante la práctica observamos que a pesar de encontrar errores mínimos en el código, la misma inteligencia artificial descifraba su solución configurando la nueva estructura del código generado, a un nivel productivo el encontrar las soluciones podría depender de la experiencia del programador, enfocarse a detalle a cada línea del código, generando un consumo excesivo en el tiempo de desarrollo.

En esta práctica hemos utilizado ChatGPT como un auxiliar de programación para llevar a cabo una serie de ejercicios relacionados con el procesamiento de datos, análisis estadístico, visualización de datos y generación de código en Python. Hemos trabajado en la manipulación de conjuntos de datos, la extracción de información clave, la generación de gráficos y la realización de tareas comunes de programación.

Se exploraron las diversas funcionalidades de Python y de bibliotecas como Pandas, Seaborn y Numpy para realizar análisis de datos y visualización. Además, hemos abordado la integración de OCR (Reconocimiento Óptico de Caracteres) en Python para extraer texto de imágenes y cómo resolver problemas relacionados con la instalación de dependencias.

Este enfoque práctico nos ha permitido no solo repasar conceptos clave de programación y análisis de datos, sino también experimentar con la interacción entre ChatGPT y el código, demostrando cómo puede ser una herramienta útil para obtener ejemplos de código, explicaciones y soluciones a problemas específicos.

En resumen, esta práctica ha demostrado cómo utilizar ChatGPT como un auxiliar de programación puede acelerar y mejorar el proceso de aprendizaje y resolución de problemas en programación y análisis de datos, al proporcionar explicaciones detalladas, ejemplos de código y orientación útil en tiempo real.