Proyecto ICS 2

Fatima Isabella Pacheco Vera 202410182
Leonardo Martinez Aquino 202410148
Isaac David Pasache Arroyo 202410214
Tiziano Abraham Lopez Vargas 202310267

1. Introducción

En este proyecto aprenderemos a como entrenar a una inteligencia artificial para que pueda clasificar números e identificarlos, a lo largo del informe usaremos conceptos básicos de Python para implementar un sistema de visión artificial enfocado en la detección y clasificación de dígitos escritos a mano, utilizando el dataset "digits" disponible en el paquete scikit-learn.

2. Importación de Librerías

Importamos las librerias necesarias para el desarrollo del proyecto:

```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
from PIL import Image
import numpy as np
import os
```

3. Importamos datos

Cargamos el dataset de digits de scikit-learn, el cual contiene las imágenes de los números del 0 al 9.

```
# Cargamos el dataset de digit
digits = datasets.load_digits()
```

4. Generamos el DataFrame

Se crea un DataFrame de pandas llamado df_digits con los datos de las imágenes y una columna adicional target que contiene las etiquetas de los dígitos.

```
df_digits = pd.DataFrame(digits.data)
df_digits["target"] = digits.target
```

5. Cálculo de los promedios

Agrupa las filas del DataFrame creado por la columna target y calcula el promedio de cada grupo, obteniendo así la imagen promedio de cada dígito del 0 al 9

```
mean_digits = df_digits.groupby("target").mean()
```

6. Selección de la imagen

Se crea un código que le permita al usuario interactuar con la imagen promedio de cada imagen que desee ver, en este caso se le solicita al usuario que seleccione un número del 0-9 para ver la imagen promedio de ese dígito seleccionado, y si el usuario decide ver todas las imágenes, tiene la opción "-1" para abrir y ver todas las imágenes promedio.

```
# Mostramos la imagen promedio que el usuario escoja
print("Seleccione el número de la imagen que desea ver")
for i in range(10):
    print(f"{i} - imagen promedio")
print("-1 - todas las imágenes promedio")
```

Dependiendo de la opción seleccionada por el usuario, muestra la imagen promedio correspondiente utilizando matplotlib.

Al mismo tiempo se agrega la opción que en caso el número ingresado no sea ninguno en el rango de -1 a 9, se le de la opción al usuario de volver a intentarlo con el mensaje "opción inválida"

```
while True:
    opc = input(": ")
    if opc in ("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "-1"):
        break
    else:
        print("Opción inválida")
```

Para mostrar todas las imágenes:

```
opc = input(": ")

if opc == "-1":
    fig, axs = plt.subplots( nrows: 2,  ncols: 5, figsize=(15, 5))
    for ax, i in zip(axs.ravel(), range(10)):
        ax.imshow(mean_digits.iloc[i].values.reshape(8, 8), cmap='viridis')
    plt.tight_layout()
    plt.show()
```

Para mostrar una sola (del 0-9):

```python
opc = input(": ")

if opc == "-1":
    fig, axs = plt.subplots( nrows: 2,  ncols: 5, figsize=(15, 5))
    for ax, i in zip(axs.ravel(), range(10)):
        ax.imshow(mean_digits.iloc[i].values.reshape(8, 8), cmap='viridis')
    plt.tight_layout()
    plt.show()
```

7. Carga de Imagen

Se crea una carpeta “images” donde añadimos las imágenes que vamos a usar, despues le damos la opción al usuario de decidir cual de todas las imágenes dentro de la carpeta quiere utilizar y el código abre la imagen deseada, al mismo tiempo también se agrega la opcion en caso del usuario no escribe una opción válida

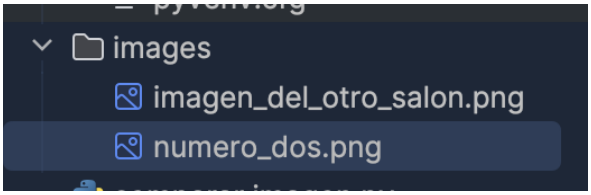
```python
if not os.path.exists("images"):
    os.makedirs("images")

fotos = os.listdir("images")

print("Mencione la imagen que desea utilizar...")
for i, foto in enumerate(fotos, start=1):
    print(f" {i}) {foto[:-4]}")

while True:
    opc = input(": ")
    if opc in [f"{i}" for i in range(1, len(fotos) + 1)]:
        break
    else:
        print("Opción inválida")

imagen = Image.open(f"./images/{fotos[int(opc)-1]}")
```



8. Escalamos la imagen

Se reescala la imagen a un tamaño de 8x8 y escalamos los valores de la imagen a un rango de 0 a 16, siendo 0 el color más claro y 16 el más oscuro

```python
# Escalamos la imagen a 8x8
imagen = imagen.resize( size: (8,8), Image.LANCZOS)

# Escalamos los valores de la imagen a un rango de 0 a 16
imagen = imagen.convert("L")
imagen = imagen.point(lambda x: (255 - x) * (16 / 255))
imagen = np.array(imagen)
```

9. Función de distancia

Se crea una función llamada “distancia” y está calcula la distancia euclidiana entre dos vectores.

```python
def distancia(x, y):
    return np.linalg.norm(x - y)
```

10. Se obtienen los 3 dígitos más cercanos

Añadimos la columna ‘dist’ al DataFrame ‘df_digits’ esta contiene las distancias de cada target, eliminamos la columna ‘target’ para que no afecte en el cálculo y después ordenamos por la distancia más cercana a la imagen.

```python
# Obtener los 3 digitos más cercanos
df_digits['dist'] = df_digits.drop(columns="target").apply(lambda x: distancia(np.array(x), imagen.flatten()), axis=1)

df_digits_sorted = df_digits.sort_values(by="dist")
```

11. Mostramos los 3 dígitos más cercanos

Una vez ordenada la columna ‘dist’ del DataFrame ‘df_digits’, se selecciona los 3 dígitos más cercanos, imprimimos el target con su distancia.

```python
# Mostramos los 3 digitos más cercanos
cercanos_digits = df_digits_sorted.head(3)
print(cercanos_digits[['target', 'dist']])
```

12. Mostramos el número detectado por la IA, Versión 1

Usamos ‘value_counts()’ de pandas para saber cuántas veces aparece cada valor en la columna ‘target’ que está en el DataFrame ‘cercanos_digits’, esta contiene las etiquetas de los dígitos, iteramos en los elementos de ‘target_cercanos’ si el ‘i’

aparece más de 2 veces imprimira que ha encontrado el numero q aparece más de 2 veces o igual.

```
# Mostramos el número que la IA ha detectado
target_cercanos = cercanos_digits['target'].value_counts()

identifico = False
for target, i in target_cercanos.items():
    if i >= 2:
        print(f"Soy la inteligencia artificial, y he detectado que el dígito ingresado corresponde al número {target}")
        identifico = True

if identifico == False:
    print("Soy la inteligencia artificial, y no he podido identificar correctamente el número ingresado")
```

13. Mostramos el número detectado por la IA, Versión 2

```
mean_digits['dist'] = mean_digits.apply(lambda x: distancia(np.array(x), imagen.flatten()), axis=1)

mean_cercanos_digits = mean_digits.sort_values(by="dist")

number_predicted = mean_cercanos_digits.head(1).index[0]
print(f"Soy la inteligencia artificial versión 2, y he detectado que el dígito ingresado corresponde al número {number_predicted}")
```

14. Version 1 o Version 2?

El método de la versión 2 es mejor, ya que se basa en la distancia entre la imagen ingresada y todas las imágenes de los dígitos, lo que permite una mejor clasificación de la imagen ingresada al tener mayores casos de prueba. Por otro lado, la versión 1 se basa en la imagen promedio de cada dígito, lo que puede no ser tan efectivo si la imagen ingresada tiene variaciones en la forma del dígito.

Pese a que la versión 2, es efectiva en su trabajo, las ligeras variaciones en la imagen como algún tipo de ruido puede afectar al resultado final.

Una observación adicional, es que la versión 2 está adaptada a si o si predecir un número que conozca, mientras que la versión 1, puede predecir un número que no conozca, ya que se basa en vecinos cercano y si tiene 3 vecinos distintos, puede retornar que no se identifica correctamente el número.

Otra observación, aparentemente la versión 2 es efectiva en los casos que el número tiene un grosor normal que sea distinguible, pero si el número es muy delgado o muy grueso, puede que no se identifique correctamente el número. Lo que no pasa con la versión 1, al tener varios casos de prueba y distintas formas de clasificar el número.

15. Casos de prueba

```
: prueba.png
      target       dist
404        8  27.386128
1553       8  31.527766
1664       8  32.326460
Soy la inteligencia artificial, y he detectado que el dígito ingresado corresponde al número 8
Soy  la  inteligencia  artificial  versión  2,  y  he detectado  que  el  dígito  ingresado  corresponde  al  número  8
```

```
      target       dist
1535       5  13.341664
281        5  15.748016
330        5  17.832555
Soy la inteligencia artificial, y he detectado que el dígito ingresado corresponde al número 5
Soy  la  inteligencia  artificial  versión  2,  y  he detectado  que  el  dígito  ingresado  corresponde  al  número  5

Process finished with exit code 0
```

```
: numero_seis.png
      target       dist
1497       6  30.495901
583        6  30.724583
598        6  31.016125
Soy la inteligencia artificial, y he detectado que el dígito ingresado corresponde al número 6
Soy  la  inteligencia  artificial  versión  2,  y  he detectado  que  el  dígito  ingresado  corresponde  al  número  5
```

```
      target       dist
1626       1  40.385641
1389       5  44.045431
1195       8  44.754888
Soy la inteligencia artificial, y no he podido identificar correctamente el número ingresado
Soy  la  inteligencia  artificial  versión  2,  y  he detectado  que  el  dígito  ingresado  corresponde  al  número  7
```

16. Conclusiones

- En este proyecto, se ha demostrado la capacidad de entrenar una inteligencia artificial para clasificar números escritos a mano utilizando el dataset "digits" de scikit-learn. Ambos métodos de clasificación, y concordamos que la versión 1 era más eficiente

- El ajuste a un tamaño (8x8) y la normalización de los valores de píxeles fueron esenciales para mejorar la precisión del modelo de visión artificial. Esto hace que las características de las imágenes sean más fácilmente comparables y permite una clasificación más precisa.

- La funcionalidad que permite a la persona seleccionar y visualizar imágenes promedio de los dígitos añade valor al código, mejorando la comprensión y el análisis de los datos por parte de la persona, dando una herramienta visual para verificar cómo se comporta el modelo.

- A pesar de los resultados positivos, el código tiene limitaciones, como su dependencia de las imágenes promedio, lo que puede afectar la precisión del código si las imágenes son un poco diferentes a las del dataset de scikit-learn.