Proyecto ICS 2

Profesor:
Ian Paul Brossard Nuñez

Integrantes:
Estefano Arom Manyari Arias - 202410063 - 50% de participación
Nicolás Alejandro Llerena Silva - 202110190 - 15% de participación
Carlos Zahid Padilla Rosales - 202410314 - 35% de participación

Introducción:

Este proyecto consiste en reconocimiento de números en manuscritos a través de un software que utiliza técnicas de aprendizaje automático y procesamiento de imagen, para de esta manera conseguir clasificar imágenes de dígitos hechos a mano del (0-9).

Antecedentes:

Esto dió a realizar basándonos en la clase en la que se realizó el análisis de imágenes, además del uso de ciertas herramientas como “numpy”, “pandas”, “sk-learn”, “matplot” y entre otras que ayudaron al desarrollo

Desarrollo:

1. Importaciones escenciales para el desarrollo:

```
import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import euclidean_distances
import cv2
```

En conjunto estas librerías como **numpy(operaciones numéricas), pandas(datos), sklearn(algoritmos de aprendizaje autónomo), matplot(visualización de datos y cv2(procesamiento de los datos)** las cuales nos proporcionaron de manera completa para abordar el proyecto del reconocimiento de dígitos manuscritos de manera eficiente y efectiva.

2. Carga del conjunto de datos:

```
# Cargar el conjunto de datos de dígitos
digitos = datasets.load_digits()
```

Se carga el conjunto de datos de los dígitos manuscritos de sckit-learn. Este conjunto incluye las imágenes que representan los dígitos.

3. Lectura y procesamiento:

```
# c) Leer un nuevo dígito desde una imagen
1 usage
def leer_imagen_digito(ruta_imagen):
    img = cv2.imread(ruta_imagen, cv2.IMREAD_GRAYSCALE)  # Leer en escala de grises
    img_redimensionada = cv2.resize(img, dsize: (8, 8), interpolation=cv2.INTER_AREA)
    img_escalada = (16 * (1 - img_redimensionada / 255)).astype(int)
    return img_escalada
```

A través de esta función leemos una imagen de un dígito desde un archivo, donde se convierte a escala de grises y las redimensiona para que coincida con las imágenes de los datos.

4. Identificación de dígitos similares:

```
distancias = euclidean_distances(digitos_aplanados, nuevo_digito_aplanado).flatten()
indices_mas_cercanos = np.argsort(distancias)[:3]  # Índices de los 3 más cercanos
digitos_mas_cercanos = digitos.target[indices_mas_cercanos]
```

Se ordenaron las distancias de menor a mayor y se seleccionaron los índices de 3 dígitos más cercanos.

5. Clasificación del nuevo dígito:

```
# f) Clasificar el nuevo dígito basado en los 3 más cercanos
if np.all(digitos_mas_cercanos == digitos_mas_cercanos[0]):  # Todos son iguales
    digito_clasificado = digitos_mas_cercanos[0]
else:
    digito_clasificado = np.bincount(digitos_mas_cercanos).argmax()  # El más frecuente
```

Se clasifican a los nuevos dígitos con el valor más frecuente entre los 3 más cercanos, si son iguales, se toma simplemente ese valor.

6. Calcular la distancia a las 10 imágenes promedio:

```
# g) Calcular la distancia a las 10 imágenes promedio e identificar la más cercana
distancias_promedio = euclidean_distances(imagenes_promedio.reshape(10, -1), nuevo_digito_aplanado).flatten()
indice_promedio_mas_cercano = np.argmin(distancias_promedio)
print(f"Soy la inteligencia artificial versión 2, y he detectado que el dígito ingresado corresponde al número {indice_promedio_mas_cercano}")
```

Basada en que cada dígito tiene características visuales distintivas y al compararlas con el promedio podría determinar cual es similar de esta manera es más robusta ante variaciones de escritura.

7. Almacenamiento de resultados:

```
63    # Guardar resultados en CSV con 5 grupos para formato condicional
64    resultados = pd.DataFrame(digitos_aplanados)
65    resultados['target'] = digitos.target
```

```
73    resultados['digito_clasificado'] = digito_clasificado
74    resultados.to_csv( path_or_buf: "resultados_con_grupos.csv", index=False)
```

Se crean dataframes con los resultados y se guardan en .csv para su posterior análisis.

Conclusiones:

1.- Generación y Visualización de Imágenes Promedio:

En esta parte nuestro código calcula y visualiza la imagen promedio de cada dígito del 0 al 9 del conjunto de datos de sklearn. Esto se pudo lograr promediando las imágenes correspondientes a cada dígito y luego mostrando estas imágenes en una cuadrícula de 2 filas por 5 columnas utilizando la biblioteca matplotlib para facilitar el trabajo.

2.- Preparar una nueva imágen:

En nuestro proyecto hay una función que lee una imagen, el cual es la foto que tiene por nombre “imagen5.png”, que hace que la imagen tenga un tamaño correcto y que esté lista para compararla con otras imágenes de números.

3.- Clasificación del nuevo dígito:

Hemos utilizado la distancia euclidiana, pues este código nos ayuda a encontrar los tres dígitos más similares al nuevo dígito y determina el dígito clasificado basándose en el más frecuente, por otra parte este también calcula la distancia entre la nueva imagen y las imágenes promedio, para que de esta manera podamos saber cual es el dígito más cercano basado en estas distancias.

4.- Resultados en el csv:

A pesar de varias revisiones por parte del grupo y apoyo de la IA no se pudo llegar a un porque al exportar los resultados a un csv y aplicar formato condicional este no lanzaba números por sí mismo sino que estos estaban dentro ordenados de manera horizontal por lo cual fue necesario la intervención para lograr la visualización de cada número los cuales si estuvieran numerados correctamente del (0-9).