Aula Prática 5

Objetivo:
O objetivo desta tarefa é que você se familiarize com a implementação de um algoritmo de **Programação Genética**.

Descrição:
Você vai usar **Programação Genética** para evoluir regras que classifiquem números como pares ou ímpares. Em vez de passar imagens completas para o GP, você vai fornecer características simples das imagens (como a soma dos pixels), e o GP vai evoluir fórmulas matemáticas para prever se o número é par ou ímpar.

Entrada:
Conjunto de Dados: Dígitos Manuscritos (MNIST reduzido)
O Scikit-learn possui um conjunto de dados chamado load_digits, que contém imagens de números manuscritos de 0 a 9. Cada número é representado por uma imagem de 8x8 pixels. As imagens são convertidas para vetores de 64 números (representando a intensidade dos pixels).

Problema:

Classificação Binária (Pares vs Ímpares)

- **Classe 0:** Números **pares** (0, 2, 4, 6, 8).
- **Classe 1:** Números **ímpares** (1, 3, 5, 7, 9).

Isso cria um **problema de classificação binária**, que o GP precisa resolver evoluindo regras.

<u>Tarefa 1 - Utilizando 2 features</u>

sum_pixels - soma dos pixels (soma de todos os valores dos pixels na imagem)
Números com traços mais "escuros" (mais tinta) terão uma soma maior.

nonzero_pixels - número de pixels não zeros (quantos pixels têm valor maior que zero, ou seja, quantos pixels têm alguma tinta). Isso dá uma ideia da "complexidade" do número (ex: o número 8 tem mais traços que o número 1).

```
def evalDigits(individual):
    func = toolbox.compile(expr=individual)
    correct = 0
    for (sum_pixels, nonzero_pixels, mean_pixels, std_pixels,
active_ratio), label in zip(features, labels):
        try:
            prediction = func(sum_pixels, nonzero_pixels, mean_pixels,
std_pixels, active_ratio)
            predicted_label = 1 if prediction > 0 else 0
        except:
            predicted_label = 0  # Penaliza funções que dão erro

        if predicted_label == label:
            correct += 1
    return correct / len(features),  # Retorna a acurácia
```

Tarefa 2 - Utilizando 5 features

```
features = np.array([
    [np.sum(img),                  # Soma dos pixels
     np.count_nonzero(img),        # Número de pixels não nulos
     np.mean(img),                 # Média da intensidade
     np.std(img),                  # Desvio padrão da intensidade
     np.count_nonzero(img) / 64    # Proporção de pixels ativos
    ]
    for img in X
])
```