**Normalização dos dados e cuidado com data leakage**

Para treinar o modelo LSTM de forma mais eficiente, decidi normalizar os dados usando MinMaxScaler, que transforma os valores para o intervalo entre 0 e 1.

**Atenção importante:** nunca se deve usar informações do conjunto de teste para escalar os dados de treino, pois isso configura **data leakage** (vazamento de informação), fazendo com que o modelo "veja" dados futuros antes de treinar.

**Procedimento adotado**

1. Dividi os dados em train\_data e test\_data.
2. Criei o scalador e ajustei apenas com os dados de treino:
   * scaler = MinMaxScaler(feature\_range=(0,1))
   * data\_train\_scaled = scaler.fit\_transform(train\_data)
3. Transformei os dados de teste usando o mesmo scalador, sem refazer o ajuste:
   * data\_test\_scaled = scaler.transform(test\_data)

Dessa forma, garanti que o modelo não teve acesso a informações do futuro durante o treino.

**Criação das sequências**

Após a normalização, dividi os dados em janelas de sequência para treinar a LSTM:

* Tamanho da sequência (SEQ\_LEN): 14
* Criação das sequências:
  + X\_train\_scaled, y\_train\_scaled = create\_sequences(data\_train\_scaled, SEQ\_LEN)
  + X\_test\_scaled, y\_test\_scaled = create\_sequences(data\_test\_scaled, SEQ\_LEN)