Detección de anomalías

En esta tarea se buscarán las imágenes del conjunto de MNIST que sean más *raras*.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml

from sklearn.cluster import DBSCAN
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from sklearn.ensemble import IsolationForest
```

Leemoes el conjunto de datos. Trabajaremos con los primeros 10000 ejemplos para que los cálculos no tarden mucho. También dividimos entre 255 para tener valores entre 0 y 1.

```
mnist = fetch_openml('mnist_784', as_frame=False)
X = mnist.data[0:10000] / 255
y = mnist.target.astype(np.uint8)[0:10000]

X.shape, y.shape
```

```
((10000, 784), (10000,))
```

Local Outlier Factor

1. Usar Local Outlier Factor con `n_neighbors`=4 para detectar anomalías. Contar cuántas imágenes de cada categoría se clasificaron como anómalas.
2. Mostrar cuáles con los índices de las primeras 16 anomalías encontradas.
3. En una imagen (en una array de 4x4), mostrar las primeras 16 anomalías encontradas.

DBSCAN

1. Usar DBSCAN con `eps`=6.8 y `min_samples`=2 para detectar anomalías. Contar cuántas imágenes de cada categoría se clasificaron como anómalas.
2. Mostrar cuáles con los índices de las primeras 16 anomalías encontradas.
3. En una imagen (en una array de 4x4), mostrar las primeras 16 anomalías encontradas.

SVM de una clase

1. Usar OneClassSVM con `kernel`=rbf y `nu`=0.1. Usar `decision_function` para mostrar las 16 imágenes más raras (más alejadas al hiperplano separador) en una imagen (en una array de 4x4).
2. Mostrar los índices de las 16 imágenes más raras.

Isolation Forest

1. Usar IsolationForest con `contamination`=0.05 y `random_state`=42. Usar `decision_function` para mostrar las 16 imágenes más raras (más alejadas al hiperplano separador) en una imagen (en una array de 4x4).
2. Mostrar los índices de las 16 imágenes más raras.