```
#cp /anvil/projects/x-med240015/data_viz_practice.ipynb $HOME/viz_practice.ipynb
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
from sklearn import metrics

# Load the hyperspectral data and ground truth from .mat files
hyperspectral_data = loadmat('/home/jovyan/PaviaU.mat')['paviaU']
ground_truth_data = loadmat('/home/jovyan/PaviaU_gt.mat')['paviaU_gt']
def plot_spectralSignature(hyperspectral_image, pixels):
   fig, ax = plt.subplots(figsize=(12, 4))
   for (x, y) in pixels:
      reflectance = hyperspectral_image[x, y, :]
      ax.plot(reflectance, label=f'Pixel ({x}, {y})')

   ax.set_title('Spectral Signatures of Selected Pixels')
   ax.set_xlabel('Band')
   ax.set_ylabel('Reflectance')
   ax.legend()
   plt.show()

pixels_to_plot = [(10, 10), (20, 20), (30, 30)]
plot_spectralSignature(hyperspectral_data, pixels_to_plot)

#Reshaping Ground Truth
def reshape_ground_truth(hyperspectral_data):
   # Get the ground truth image into a 1D array
   ground_truth_1d = hyperspectral_data.flatten()
   return ground_truth_1d
ground_truth_image = np.array([[1, 2, 3, 4]])
print(reshape_ground_truth(hyperspectral_data))
plt.imshow(ground_truth_image)

#Reshape Hyperspectral Image
def reshape_hyperspectral_data(hyperspectral_data):
   shape_image = hyperspectral_data.shape
```

```
    reshaped_data = hyperspectral_data.reshape((shape_image[0] * shape_image[1],
shape_image[2]))
    return reshaped_data
print (reshape_hyperspectral_data(hyperspectral_data))

#Normalization Function
def normalization(bands_pixels_HS):
    r, c = bands_pixels_HS.shape
    normalized_pixels = np.zeros((r, c))
    i = 0
    #For loop
    for x in bands_pixels_HS:
        #Minimum value
        min_val = min(x)
        #Maximum Value
        max_val = max(x)
        #NORMALIZATION Formula
        x_norm = (x - min_val) / (max_val - min_val)
        normalized_pixels[i,:] = x_norm
        i = i+1
    return normalized_pixels

#Making a Standardization Function
def standardization(bands_pixels_HS):
    r, c = bands_pixels_HS.shape
    standardized_pixels = np.zeros((r, c))
    i = 0
    #For loop
    for x in bands_pixels_HS:
        #Mean value
        mean_val = np.mean(bands_pixels_HS)
        #Standard Deviation
        std_val = np.std(bands_pixels_HS)
        #STANDARDIZATION Formula
        x_stand = (x - mean_val) / std_val
        standardized_pixels[i,:] = x_stand
        i = i+1
    return standardized_pixels
```