Sample code for Landscape Dynamics, Hotspot and Prediction

import cv2

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Read the landscape image

image\_path = 'G:/GCU/projects/LULC\_2005\_15\_vik/0506\_vik.tif'

image = cv2.imread(image\_path)

# Convert the image to a 2D array of RGB values

height, width, channels = image.shape

image\_2d = image.reshape((height \* width, channels))

# Use K-Means clustering to identify hotspots

num\_clusters = 5  # You can adjust this based on the characteristics of your data

kmeans = KMeans(n\_clusters=num\_clusters, random\_state=42)

kmeans.fit(image\_2d)

# Assign each pixel to a cluster

labels = kmeans.labels\_

hotspot\_label = np.argmax(np.bincount(labels))

# Highlight hotspots in the original image

hotspot\_mask = (labels == hotspot\_label).reshape((height, width, 1))

hotspot\_highlight = image.copy()

#hotspot\_highlight[hotspot\_mask == 0] = [0, 0, 0]  # Set non-hotspot areas to black

# Display the original and highlighted images

cv2.imshow('Original Image', image)

cv2.imshow('Hotspot Highlight', hotspot\_highlight)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Detect anomalies (areas that deviate from the cluster)

hotspot\_centroid = kmeans.cluster\_centers\_[hotspot\_label]

# Calculate distances from each pixel to the hotspot cluster centroid

distances = np.linalg.norm(image\_2d - hotspot\_centroid, axis=1)

# Normalize distances using StandardScaler

scaler = StandardScaler()

distances\_scaled = scaler.fit\_transform(distances.reshape(-1, 1))

# Define a threshold for anomaly detection (adjust as needed)

anomaly\_threshold = 2.0

# Identify anomalies

anomalies = distances\_scaled > anomaly\_threshold

# Highlight anomalies in the original image

anomaly\_mask = anomalies.reshape((height, width, 1))

anomaly\_highlight = image.copy()

#anomaly\_highlight[anomaly\_mask == 1] = [0, 0, 255]  # Set anomaly areas to red

# Display the anomaly-highlighted image

cv2.imshow('Anomaly Highlight', anomaly\_highlight)

cv2.waitKey(0)

cv2.destroyAllWindows()