

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

LAB ASSIGNMENT # 12

**Adaptive Smoothing and HAC**

**PREPARED BY:**

**Group No. 4**

Sandhya Yadav-201652023

Payal Mahisaniya-201652015

Pooja Gurjar-201652016

**PROBLEM STATEMENT 1:**

Given the noisy image test\_noisy.jpg (available in the laboratory\_work folder). Use the adaptive smoothing (Ref. Diffusions and Confusions in Signal and Image Processing) process to remove the noise. Explain the selected values of the parameters in the algorithm.

**Python Code:**

import numpy as np

import cv2

from matplotlib import pyplot as plt

def dist(i, j, k, l, kernel):

   return np.square(i - k) +  np.square(j - l) + np.square(kernel[i, j] -  kernel[k, l])

def get\_sum\_beta(i, j, dist\_mat, gamma):

    m = dist\_mat.shape[0]

    n = dist\_mat.shape[1]

    s =0

    for k in range(m):

       for l in range(n):

           if i == k and l == j:

                s +=1 / (1 + np.power(dist\_mat[i,j,k,l], gamma))

    return s

def get\_beta(i, j , alpha, dist\_mat, gamma):

    dist\_sum = alpha / get\_sum\_beta(i, j, dist\_mat,  gamma)

    return dist\_sum

def get\_wt(i, j, k, l, alpha, dist\_mat, kernel):

    return get\_beta(i, j, alpha, dist\_mat, kernel) / (1 + np.power(dist\_mat[i, j, k, l], gamma))

def create\_distance\_matrix(kernel):

    m,n = kernel.shape

    dist\_mat = np.zeros((m,n,m,n))

    for i in range(m):

        for j in range(n):

            for k in range(m):

                for l in range( n):

                    dist\_mat[i, j, k, l] = dist(i, j, k, l, kernel)

    return dist\_mat

def create\_weight\_matrix(kernel, alpha, dist\_mat):

    m,n = kernel.shape

    weight\_mat = np.zeros((m,n,m,n))

    for i in range(m):

        for j in range(n):

            for k in range(m):

                for l in range( n):

                    weight\_mat[i, j, k, l] = get\_wt(i, j, k, l, alpha, dist\_mat, kernel)

    return weight\_mat

def adaptive\_smoothing\_images(kernel, alpha=0.2):

    dist\_matrix = create\_distance\_matrix(kernel)

    weight\_mat = create\_weight\_matrix(kernel, alpha, dist\_mat)

    pass

kernel = np.array([[1,2,3],[4,5,6], [7,8,9]])

d = create\_distance\_matrix(kernel)

print(1 / d[0,0,:, :])

# Using cv2

img = cv2.imread('test\_noisy.jpg')

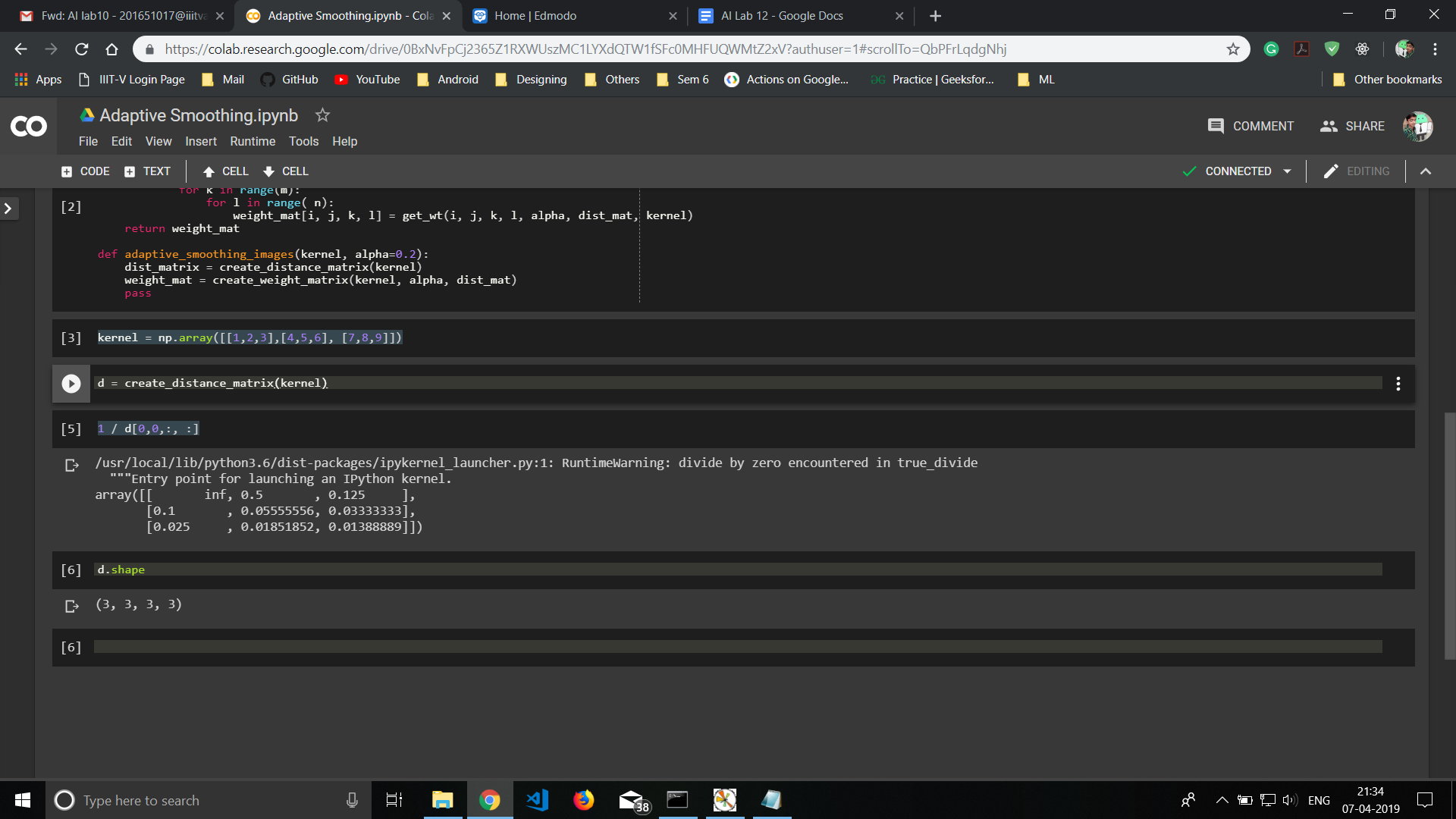
dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21

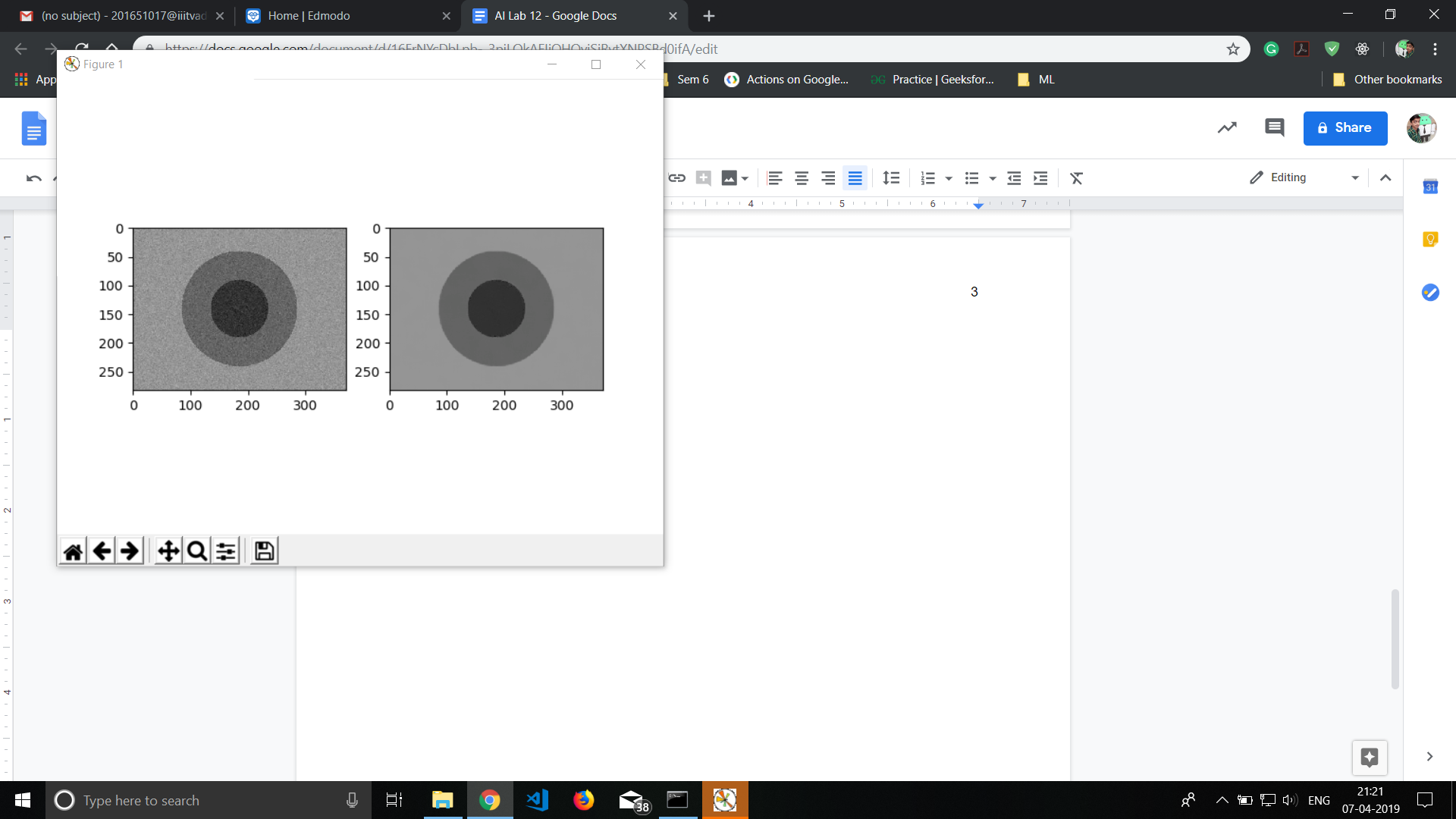
plt.subplot(121),plt.imshow(img)

plt.subplot(122),plt.imshow(dst)

plt.show()

**Output:**





**PROBLEM STATEMENT 2:**

Use HAC with Euclidean/ Manhattan distance as a measure (Single link, complete link, Ward’s distance, Group average, Centroid, Clusteroid) cluster the states of India based on the feature vector comprising of the following parameters (for one of the financial year values available in the data-set)

Percentage of schools with electricity

Percentage of schools with girls toilet

Percentage of schools with drinking water

Percentage of schools with boys toilet

For second problem you should get the data from the following:

<https://data.gov.in/>

As an example, a simple visualization based on percentage of schools with electricity is available at<https://data.gov.in/major-indicator/percentage-schools-electricity>

**Code:**

import pandas as pd

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

data = pd.read\_csv('data.csv')

print(data.columns)

measures = ['single', 'complete','ward','average','centroid','median']

measureTitles = ['SINGLE', 'COMPLETE','WARD','GROUP AVERAGE','CENTROID','CLUSTEROID']

attribute=data.columns[1:]

for i in range(len(measures)):

ytdist = data.set\_index('State')[attribute]

Z = hierarchy.linkage(ytdist, measures[i])

labelList = data['State']

fig, axes = plt.subplots(figsize=(15,10))

dn = hierarchy.dendrogram(Z,orientation='top',ax=axes,distance\_sort='descending',show\_leaf\_counts=False)

labels = [data['State'][i] for i in dn['leaves']]

axes.set\_xticklabels(labels, rotation='vertical')

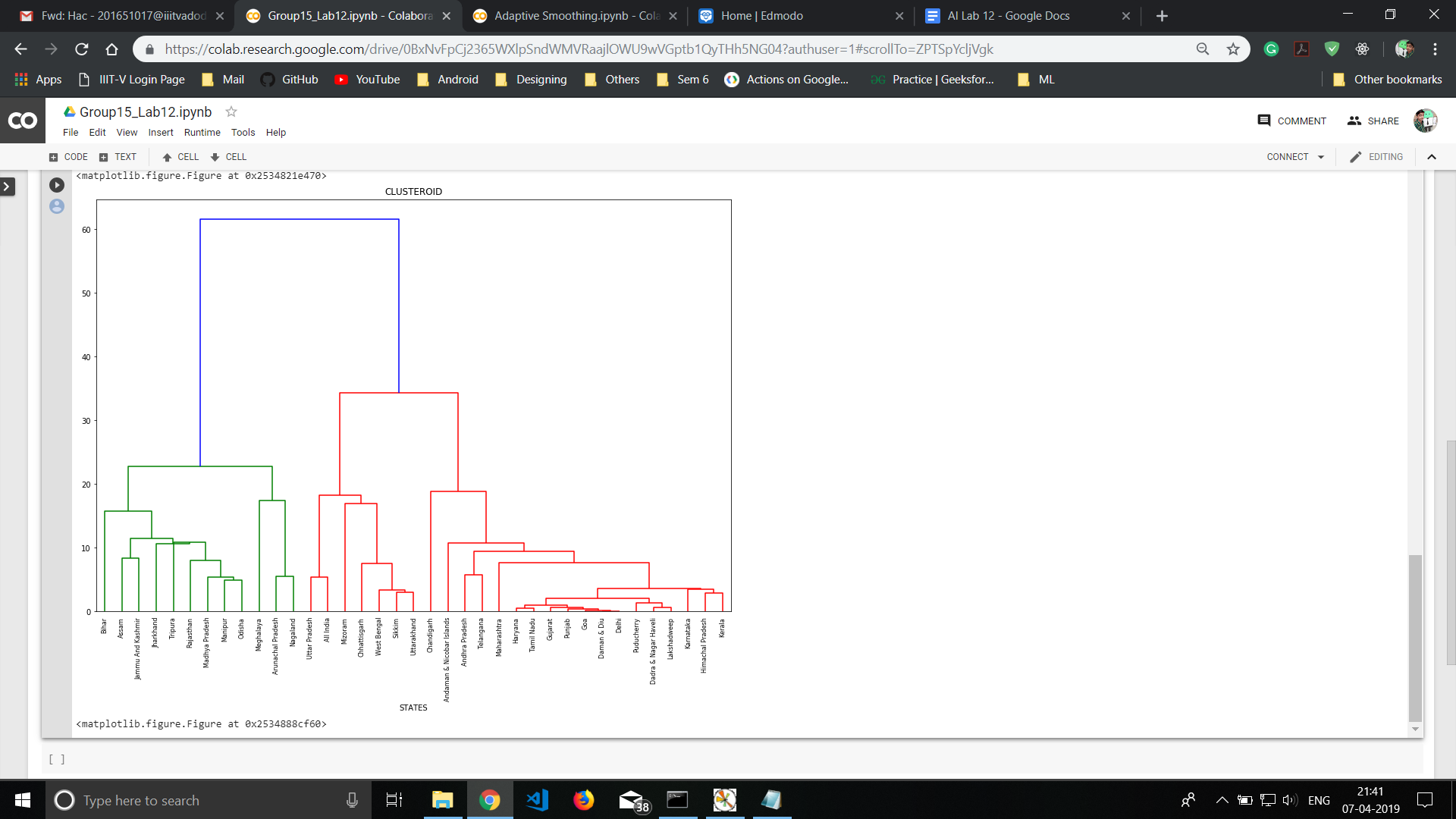
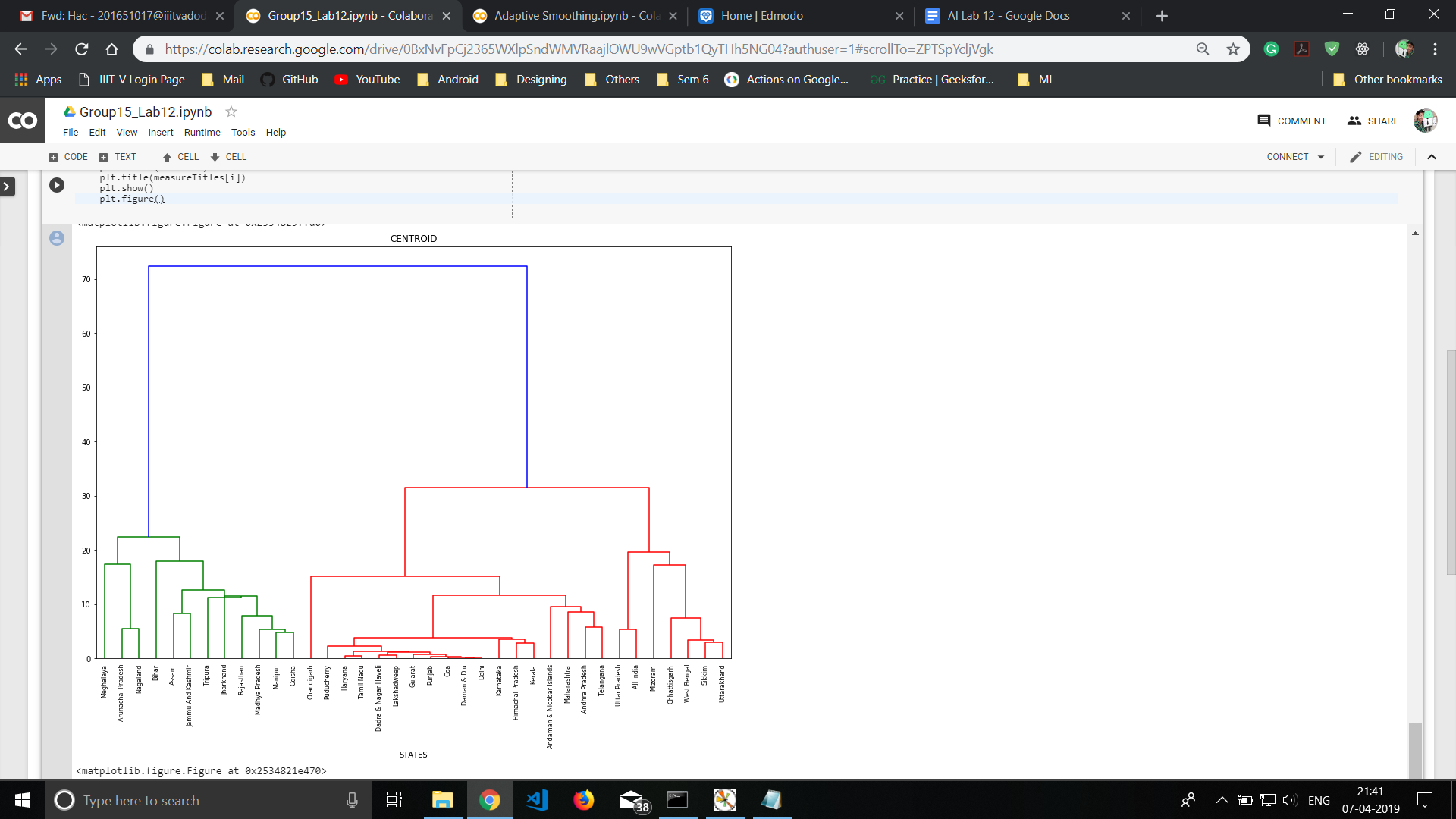
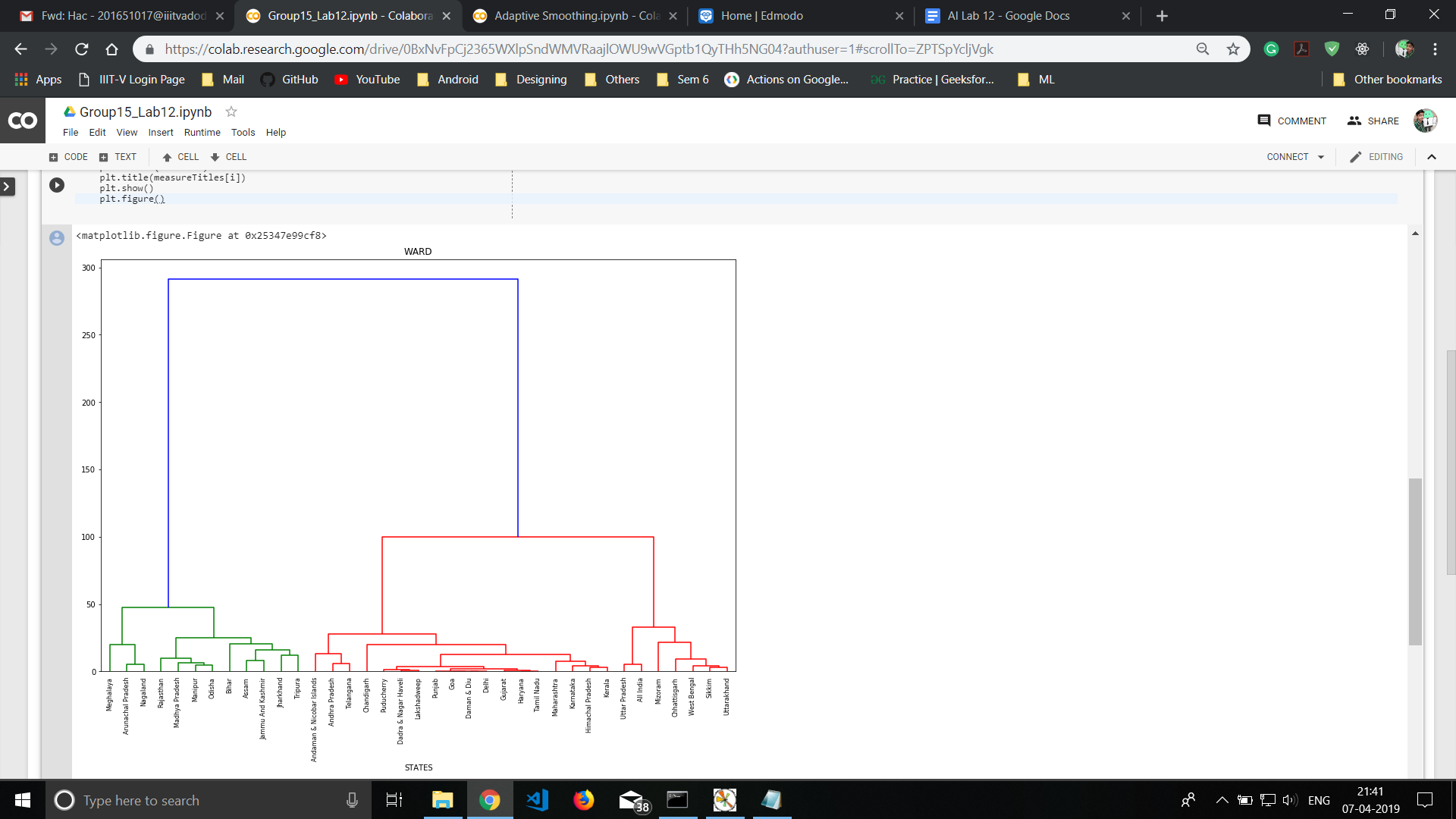
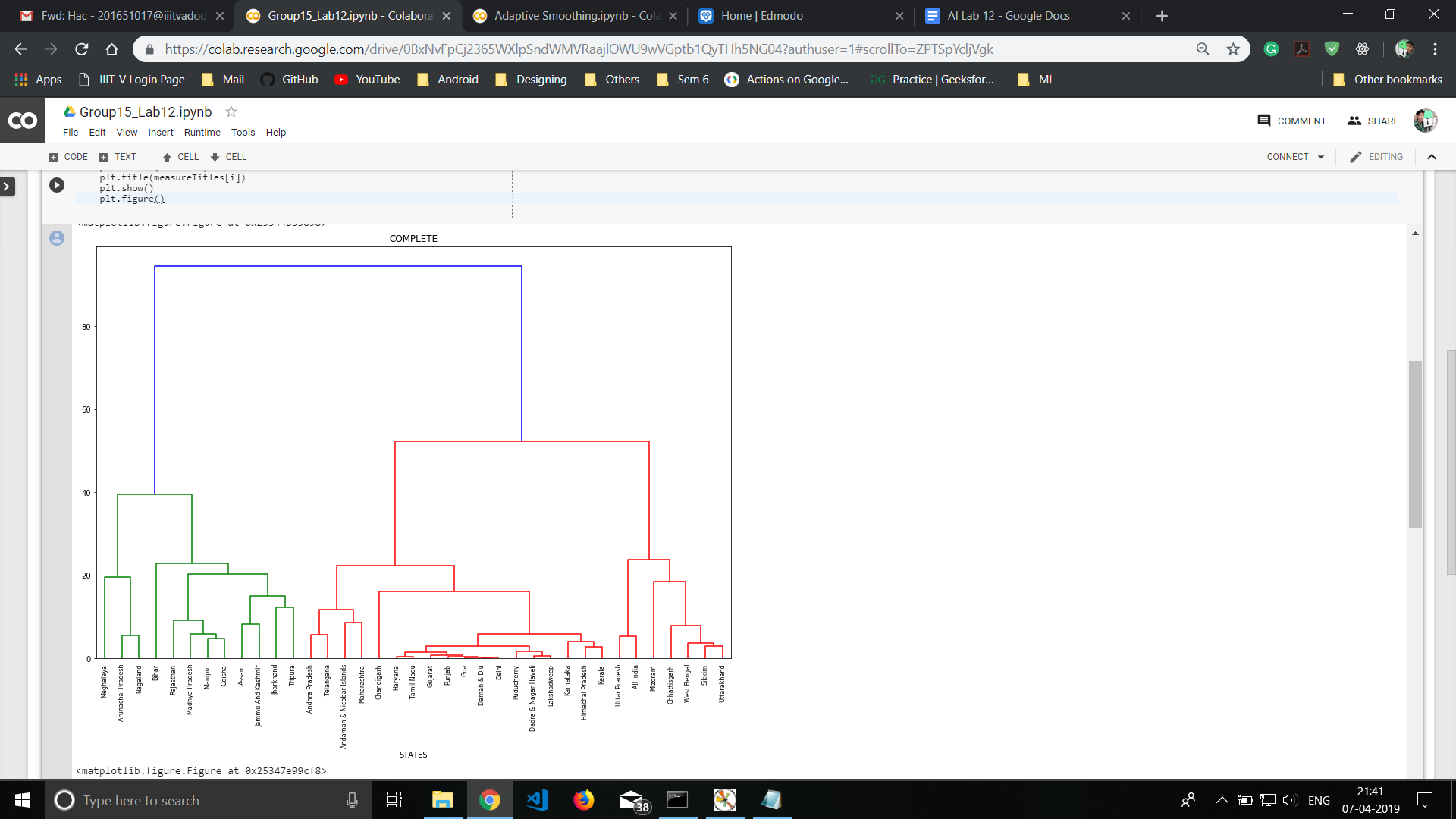
plt.xlabel('STATES')

plt.title(measureTitles[i])

plt.show()

plt.figure()

**Outputs:**

****