```
from scipy import io
import numpy as np
import networkx as nx


def mat2np(file):
    '''
    Transform matlab matrix to python numpy array
    :param file: path of matlab file
    :return: numpy array
    '''

    mat = io.loadmat(file)
    info_key = list(mat.keys())[-1]
    new_array = np.array(mat[info_key])
    return new_array

def onsetTime2class(onsetTime_array, split_list):
    '''
    Transform the real value of onset time to multiple classes based on log
interval
    :param onsetTime_array: onset time array in real value type
    :param split_list: critical time points
    :return: onset time array in separated class
    '''

    y = onsetTime_array.flatten()
    new_y = np.zeros(y.shape)
    # ori = deepcopy(y)

    y[y==-1] = 10000

    new_y[y >= split_list[1]] = 0
    y[y >= split_list[1]] = -1
    new_y[y >= split_list[0]] = 2
    y[y >= split_list[0]] = -1
    new_y[y >= 0] = 1
    y[y >= 0] = -1
    new_y = new_y.astype(np.int64)

    return new_y

def load_data(split_list):
    '''
    Load four power matrices based on different mode
    :param split_list: critical time points
    :return: before_array, after_array, y
    '''

    before_array = np.array([])
    load_list = ['One', 'Two', 'Three', 'Four', 'Five']
    # load_list = ['One']
    for i in range(len(load_list)):
        print('Loading set', load_list[i])
        before_mat_nor = 'data/data_inout_uiuc150_Nm2/data_inout_set' +
load_list[i] + '/data_mat_beforeNmk_3d.mat'
        after_mat_nor = 'data/data_inout_uiuc150_Nm2/data_inout_set' + load_list[i]
+ '/data_mat_afterNmk_3d.mat'
        onsetTime_mat = 'data/data_inout_uiuc150_Nm2/data_inout_set' + load_list[i]
```

```
+ '/data_onsetTime_output.mat'

        # get all data
        if i == 0:
            before_array = mat2np(before_mat_nor)
            after_array = mat2np(after_mat_nor)
            onsetTime_array = mat2np(onsetTime_mat)
        else:
            before_array = np.vstack((before_array, mat2np(before_mat_nor)))
            after_array = np.vstack((after_array, mat2np(after_mat_nor)))
            onsetTime_array = np.vstack((onsetTime_array, mat2np(onsetTime_mat)))

    # transform real data to class
    y = onsetTime2class(onsetTime_array, split_list)

    return before_array, after_array, y

def load_topology():
    '''
    Load the adjacent matrix of UIUC 150 power system
    :return: adjacent matrix
    '''

    adj_file = 'data/adj.csv'
    g = nx.read_edgelist(adj_file)
    adj = nx.to_numpy_array(g, nodelist=np.array(sorted(np.array(list(g.nodes),
dtype=int)), dtype=str))
    return adj


def load_nm2_changed(split_list):
    '''
    Load the modified n-2 data
    :param split_list: critical time points
    :return: before_array, after_array, y
    '''

    print('load testing nm2_changed')
    before_mat_nor =
'data/dataSet_Nm2_changed/data_Nm2_changed/data_mat_beforeNmk_3d.mat'

    after_mat_nor =
'data/dataSet_Nm2_changed/data_Nm2_changed/data_mat_afterNmk_3d.mat'

    onsetTime_mat =
'data/dataSet_Nm2_changed/data_Nm2_changed/data_onsetTime_output.mat'

    # get all data
    before_array = mat2np(before_mat_nor)
    after_array = mat2np(after_mat_nor)
    onsetTime_array = mat2np(onsetTime_mat)

    # transform real data to class
    y = onsetTime2class(onsetTime_array, split_list)
    return before_array, after_array, y

def load_nm3to6(k, split_list):
    '''
    Load N-k data
    :param k: links be removed
    :param split_list: critical time points
    :return: before_array, after_array, y
    '''
    print('load testing nm'+str(k))
    before_mat_nor = 'data/data_inout_uiuc150_Nm'+ str(k) +
'/data_mat_beforeNmk_3d.mat'
    after_mat_nor = 'data/data_inout_uiuc150_Nm'+ str(k) +
'/data_mat_afterNmk_3d.mat'
    onsetTime_mat = 'data/data_inout_uiuc150_Nm'+ str(k) +
'/data_onsetTime_output.mat'

    # get all data
    before_array = mat2np(before_mat_nor)
    after_array = mat2np(after_mat_nor)


    onsetTime_array = mat2np(onsetTime_mat)


    # transform real data to class
    y = onsetTime2class(onsetTime_array, split_list)


    return before_array, after_array, y
```