**MACHINE INTELLIGENCE**

**LAB ASSIGNMENT WEEK-1**

**NAME :** SANMAT SANJAYAKUMAR PAYAGOUDAR

**SECTION :** G

**SRN :** PES1UG20CS385

**CODE :**

PES1UG20CS385.py

#This week code focuses on understading basic functions of pandas and numpy

import numpy as np

import pandas as pd

#input: tuple (x,y)   x,y: int

def create\_numpy\_ones\_array(shape):

    #return a numpy array with one at all index

    array=None

    array = np.ones (shape=shape)

    return array

#input: tuple (x,y)   X,y: int

def create\_numpy\_zeros\_array(shape) :

    #return a numpy array with zeros at all index

    array=None

    array = np.zeros (shape=shape)

    return array

#input: int

def create\_identity\_numpy\_array(order) :

    #return a identity numpy array of the defined order

    array=None

    array = np.identity(order)

    return array

#Input: numpy array

def matrix\_cofactor (array):

    #return coractor matrix or the given array

    det = np.linalg.det (array)

    if (det!=0):

        cofact = None

        cofact = np.linalg.inv(array).T \* det

        #return cofactor matrix of the given matrix

        return array

#input : (numpy array, int, numpy array, int, int, int, int, tuple, tuple)

#tuple (x,y)  x,y: int

def f1 (X1, coef1, X2, coef2, seed1, seed2, seed3, shape1, shape2):

    np.random.seed(seed1)

    m1=np.random.rand(shape1[0], shape1[1])

    np.random.seed(seed2)

    m2=np.random.rand(shape2[0], shape2[1])

    if(shape1 == shape2 and shape1[1] == X1.shape[0] == X1.shape[1] and shape2[1] == X2.shape[0] == X2.shape[1]):

        shape3 = (shape1[0], shape2[1])

        np.random.seed(seed3)

        S = np.random.rand(shape3[0], shape3[1])

        x = np.matmul(m1, np.linalg.matrix\_power(X1, coef1))

        y = np.matmul(m2, np.linalg.matrix\_power(X2, coef2))

        ans = x+y+S

        return ans

    return -1

def fill\_with\_mode(filename, column):

    df = pd.read\_csv(filename)

    df[column].fillna(df[column].mode()[0], inplace = True)

    return df

def fill\_with\_group\_average(df, group, column):

    df[column].fillna(df.groupby(group)[column].transform('mean'), inplace=True)

    return df

def get\_rows\_greater\_than\_avg(df, column):

    df=df[df[column] > df[column].mean()]

    return df

SampleTest.py :

import sys

import importlib

import argparse

import pandas as pd

import numpy as np

parser = argparse.ArgumentParser()

parser.add\_argument('--SRN', required=True)

args = parser.parse\_args()

subname = args.SRN

try:

    mymodule = importlib.import\_module(subname)

except Exception as e:

    print(e)

    print("Rename your written program as YOUR\_SRN.py and run python3.7 SampleTest.py --SRN YOUR\_SRN ")

    sys.exit()

fill\_with\_mode = mymodule.fill\_with\_mode

fill\_with\_group\_average = mymodule.fill\_with\_group\_average

get\_rows\_greater\_than\_avg = mymodule.get\_rows\_greater\_than\_avg

def test\_case():

    df = pd.read\_csv('example.csv')

    df['Attribute C'].fillna(

        df['Attribute C'].mode()[0], inplace=True)

    try:

        if mymodule.create\_numpy\_ones\_array((2,2)).all()==np.array([[1,1],[1,1]]).all():

            print("Test Case 1 for create\_numpy\_ones\_array PASSED")

        else:

            print("Test Case 1 for create\_numpy\_ones\_array FAILED")

    except Exception as e:

        print("Test Case 1 for create\_numpy\_ones\_array FAILED due to ",e)

    try:

        if mymodule.create\_numpy\_zeros\_array((2,2)).all()==np.array([[0,0],[0,0]]).all():

            print("Test Case 2 for create\_numpy\_zeros\_array PASSED")

        else:

            print("Test Case 2 for create\_numpy\_zeros\_array FAILED")

    except Exception as e:

        print("Test Case 2 for create\_numpy\_zeros\_array FAILED due to ",e)

    try:

        if mymodule.create\_identity\_numpy\_array(2).all()==np.array([[1,0],[0,1]]).all():

            print("Test Case 3 for create\_identity\_numpy\_array PASSED")

        else:

            print("Test Case 3 for create\_identity\_numpy\_array FAILED")

    except Exception as e:

        print("Test Case 3 for create\_identity\_numpy\_array FAILED due to ",e)

    try:

        if mymodule.matrix\_cofactor(np.array([[4,6],[8,5]])).all()==np.array([[ 5., -8.],[-6. , 4.]]).all():

            print("Test Case 4 for matrix\_cofactor PASSED")

        else:

            print("Test Case 4 for matrix\_cofactor FAILED")

    except Exception as e:

        print("Test Case 4 for matrix\_cofactor FAILED due to ",e)

    try:

        if mymodule.f1(np.array([[1,2],[3,4]]),3,np.array([[1,2],[3,4]]),2,1,2,3,(3,2),(3,2)).all()==np.array([[415.11116764, 604.9332781 ],[187.42695991 ,273.27266349],[112.57538713, 163.6775407 ]]).all():

            print("Test Case 5 for f1 PASSED")

        else:

            print("Test Case 5 for f1 FAILED")

    except Exception as e:

        print("Test Case 5 for f1 FAILED due to ",e)

    try:

        if mymodule.f1(np.array([[1,2],[3,4]]),3,np.array([[1,2],[3,4]]),2,1,2,3,(3,2),(4,2))==-1:

            print("Test Case 6 for f1 PASSED")

        else:

            print("Test Case 6 for f1 FAILED")

    except Exception as e:

        print("Test Case 6 for f1 FAILED due to ",e)

    try:

        if fill\_with\_mode('example.csv', 'Attribute C').equals(df):

            print("Test Case 7 for the function fill\_with\_mode PASSED")

        else:

            print("Test Case 7 for the function fill\_with\_mode FAILED")

    except:

        print("Test Case 7 for the function fill\_with\_mode FAILED")

    df\_copy = df.copy()

    df['Attribute A'].fillna(df.groupby(

        'Attribute C')['Attribute A'].transform('mean'), inplace=True)

    try:

        if fill\_with\_group\_average(df\_copy, 'Attribute C', 'Attribute A').equals(df):

            print("Test Case 8 for the function fill\_with\_group\_average PASSED")

        else:

            print("Test Case 8 for the function fill\_with\_group\_average FAILED")

    except:

        print("Test Case 8 for the function fill\_with\_group\_average FAILED")

    df\_copy = df[df['Attribute B'] > df['Attribute B'].mean()]

    try:

        if get\_rows\_greater\_than\_avg(df, 'Attribute B').equals(df\_copy):

            print("Test Case 9 for the function get\_rows\_greater\_than\_avg PASSED")

        else:

            print("Test Case 9 for the function get\_rows\_greater\_than\_avg FAILED")

    except:

        print("Test Case 9 for the function get\_rows\_greater\_than\_avg FAILED")

if \_\_name\_\_ == "\_\_main\_\_":

    test\_case()

**OUT PUT :**

