PES1UG20CS435 Sriram Radhakrishna Section : H Roll # : 16

MI Laboratory Week 1

PES1UG20CS435.py :

```
# Do not change the function definitions or the parameters
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)
        #TODO
        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)
        #TODO
        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)
        #TODO
        return array

#input: numpy array
def matrix_cofactor(array):

        #return cofactor matrix of the given array
        determinant = np.linalg.det(array)
        if(determinant != 0):
                cofactor = None
                cofactor = np.linalg.inv(array).T * determinant

        array = cofactor

        #TODO
        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):
        #note: shape is of the forst (x1,x2)
        #return W1 x (X1 ** coef1) + W2 x (X2 ** coef2) +b
        # where W1 is random matrix of shape shape1 with seed1
        # where W2 is random matrix of shape shape2 with seed2
        # where B is a random matrix of compatible shape with seed3
        # if dimension mismatch occur return -1
        ans=None

        try :
```

```
                np.random.seed(seed1)
                W1 = np.random.rand(*shape1)
                np.random.seed(seed2)
                W2 = np.random.rand(*shape2)

                np.random.seed(seed3)
                A = np.dot(W1, (X1 ** coef1)) + np.dot(W2, (X2 ** coef2))
                BShape = A.shape
                B = np.random.rand(*BShape)

                ans = A + B

        except :

                ans = -1

        #TODO
        return ans



def fill_with_mode(filename, column):
        """
        Fill the missing values(NaN) in a column with the mode of that column
        Args:
                filename: Name of the CSV file.
                column: Name of the column to fill
        Returns:
                df: Pandas DataFrame object.
                (Representing entire data and where 'column' does not contain
NaN values)
                (Filled with above mentioned rules)
        """

        # assuming filename is supplied as 'filename.csv' & column name is
supplied as a string
        df = pd.read_csv(filename)
        df[column] = df[column].fillna(df[column].mode()[0])

        return df

def fill_with_group_average(df, group, column):
        """
        Fill the missing values(NaN) in column with the mean value of the
        group the row belongs to.
        The rows are grouped based on the values of another column

        Args:
                df: A pandas DataFrame object representing the data.
                group: The column to group the rows with
                column: Name of the column to fill
        Returns:
                df: Pandas DataFrame object.
                (Representing entire data and where 'column' does not contain
NaN values)
                (Filled with above mentioned rules)
        """

        df[column] = df[column].fillna(df.groupby(group)
[column].transform('mean'))
        return df


def get_rows_greater_than_avg(df, column):
```

```
        """
        Return all the rows(with all columns) where the value in a certain
'column'
        is greater than the average value of that column.

        row where row.column > mean(data.column)

        Args:
                df: A pandas DataFrame object representing the data.
                column: Name of the column to fill
        Returns:
                df: Pandas DataFrame object.
        """
        df = df[df[column] > df[column].mean()]

        return df
```