Week 1 - Numpy and Pandas

MACHINE INTELLIGENCE LABORATORY

Teaching Assistants
vighneshkamath43@gmail.com
sarasharish2000@gmail.com

Numpy and Pandas are one of the most important libraries in python that helps in implementation of various machine learning algorithm.

In this week the task is to understand these libraries in depth and implement the few function using these libraries.

Your task is to complete the code for these function.

You are provided with the following files:

1. **week1.py**
2. **SampleTest.py**
3. **example.csv (you need not alter this , this a dependent file for Sample test case)**

Before you head to implement week1.py , you're required to implement and experiment with the codes mentioned at the end of this document and understand the library and its functions. Also note that a detail understanding can be achieved using the below links:

Numpy : https://www.youtube.com/watch?v=QUT1VHiLmmI

Pandas: https://www.youtube.com/watch?v=vmEHCJofslg

Important Points:

1. **Please do not make changes to the function definitions that are provided to you. Use the skeleton as it has been given. Also do not make changes to the sample test file provided to you. Run it as it is.**
2. **Any changes to code skeleton may result in failure of hidden test case**
3. **Remember first you need to experiment the below code snippets in python interactive shell (type python3 or python3.7 in terminal and press enter) , you need to try different inputs and see how the output changes**
4. You are free to write any helper functions that can be called in any of these predefined functions given to you.
5. Your code will be auto evaluated by our testing script and our dataset and test cases will not be revealed. Please ensure you take care of all edge cases!

Practice Codes Numpy:

import numpy #before you start using below snippets ,import these library

#for the all the below snippets try different types of input parameter and check the output, sample input parameter is given , change it and observe the output

# example if input parameter is (2,1), try out (1,1) (3,2) etc…

1. **numpy.ones(shape, dtype=None)**

   Parameter:

   shape: *int or sequence of ints*

   Shape of the new array, e.g., (2, 3) or 2.

   dtype: *data-type, optional*

   The desired data-type for the array, e.g., numpy.int8. Default is numpy.float64.

   Return:

   out: *ndarray*

   Array of ones with the given shape, dtype, and order.

2. **numpy.zeros(shape, dtype=None)**

   Parameter:

   shape: *int or sequence of ints*

   Shape of the new array, e.g., (2, 3) or 2.

   dtype:*data-type, optional*

   The desired data-type for the array, e.g., numpy.int8. Default is numpy.float64.

Return:
out: *ndarray*
Array of ones with the given shape, dtype, and order.

3. numpy.random.rand(d0, d1, ..., dn)

Parameter:
d0, d1, …, dn *int, optional*
The dimensions of the returned array must be non-negative. If no argument is given a single Python float is returned.
Return:
ndarray, shape (d0, d1, ..., dn)
Random values.

4. numpy.random.seed(x)

Parameter:
X: int
Usage:
```
numpy.random.seed(0)
numpy.random.rand(1,2) => suppose it returns array([[0.5488135 , 0.71518937]])
numpy.random.seed(0)
numpy.random.rand(1,2) => returns same array([[0.5488135 , 0.71518937]])
numpy.random.rand(1,2) => returns different array([[0.60276338, 0.54488318]])
```

i.e every time a seed is set it returns same random array

5. Converting a list to numpy array

```
x=[1,2,3]
a=np.array(x)
```

6. Accessing element of numpy array

```
x=[[1,2,3],[4,5,6]]
a=np.array(x)
print(a[0][0]) => 1
a[0][0]=10
print(a)      =>array([[10,2,3],[4,5,6]])
```

7. Element-wise multiplication of two array

```
a=np.array([[1,2],[3,4]])
b=np.array([[5,6],[7,8]])
m=np.multiply(a,b)     =>array([[ 5, 12],[21, 32]])
```

8. Matrix multiplication

```
a=np.array([[1,2],[3,4]])
b=np.array([[5,6],[7,8]])
m=np.matmul(a,b)     =>array([[19, 22],[43, 50]])
```

9. Dot Product

```
a=np.array([1,2])
b=np.array([3,4])
m=np.dot(a,b)                    =>11
```

10. Cross Product

```
 x = [1, 2]
y = [4, 5, 6]
np.cross(x, y) =>array([12, -6, -3])
```

11. Inverse of a matrix

```
x=np.array([[1,2],[3,4]])
Inverse=np.linalg.inv(x) =>array([[-2,1],[1.5,-0.5]])
```

12. Transpose of a matrix

```
x=np.array([[1,2],[3,4]])
t=np.transpose(x)     =>array([[1, 3], [2, 4]])
```

13. Determinant of a matrix

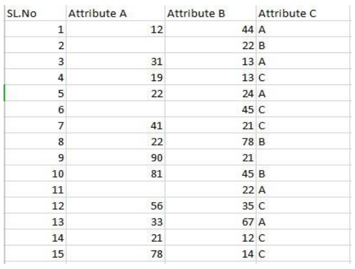
```
x=np.array([[1,2],[3,4]])
det=np.linalg.det(x) =>-2
```

14. Shape of numpy array

```
x=np.array([[1,2],[3,4]])
x.shape
```

Practice Codes Pandas:

We will use the below data as example



SL.No	Attribute A	Attribute B	Attribute C
1	12	44	A
2		22	B
3	31	13	A
4	19	13	C
5	22	24	A
6		45	C
7	41	21	C
8	22	78	B
9	90	21	
10	81	45	B
11		22	A
12	56	35	C
13	33	67	A
14	21	12	C
15	78	14	C

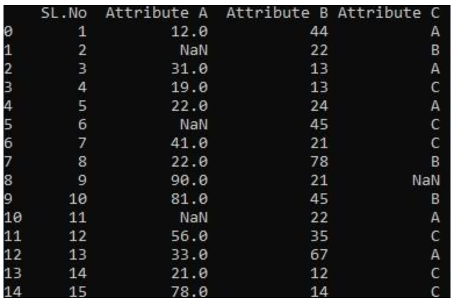
import pandas as pd

1. **Creating pandas dataframe from a csv file**

   df=pd.read_csv("example.csv")

   print(df)

   Output:



```
    SL.No  Attribute A  Attribute B Attribute C
0       1         12.0           44           A
1       2          NaN           22           B
2       3         31.0           13           A
3       4         19.0           13           C
4       5         22.0           24           A
5       6          NaN           45           C
6       7         41.0           21           C
7       8         22.0           78           B
8       9         90.0           21         NaN
9      10         81.0           45           B
10     11          NaN           22           A
11     12         56.0           35           C
12     13         33.0           67           A
13     14         21.0           12           C
14     15         78.0           14           C
```

2. **Finding column names**

   print(list(df.columns)) =>['SL.No', 'Attribute A', 'Attribute B', 'Attribute C']

3. **Retrieving column data given column name**

   print(df['SL.No']) =>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

4. **Retrieving row data**

   print(list(df.iloc[0])) =>[1, 12.0, 44, 'A'] #0 indicates first row

   print(list(df.iloc[1])) =>[2, nan, 22, 'B']

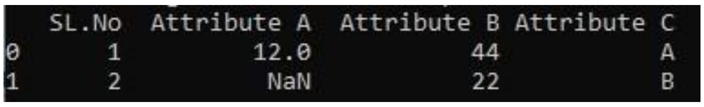
5. **Retrieving column data using column number**

   print(list(df.iloc[:,0])) =>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] #0 indicates first column

   print(list(df.iloc[:,1])) =>[12.0, nan, 31.0, 19.0, 22.0, nan, 41.0, 22.0, 90.0, 81.0, nan, 56.0, 33.0, 21.0, 78.0]

6. **Slicing the data frame wrt rows**

   print(df[0:2])



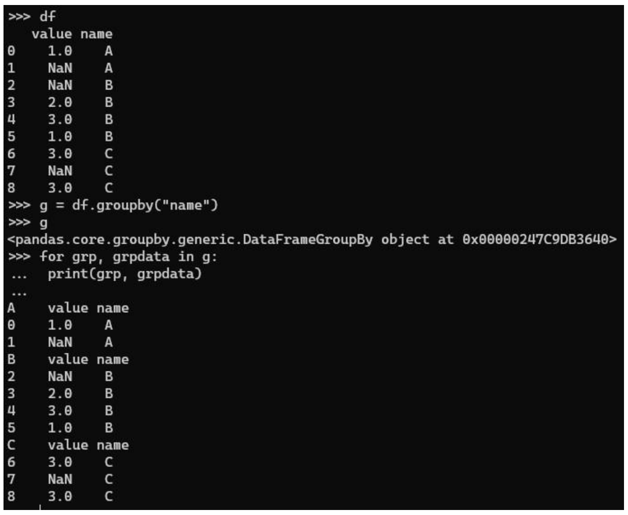
```
   SL.No  Attribute A  Attribute B Attribute C
0      1         12.0           44           A
1      2          NaN           22           B
```

7. **Mean**

   DataFrame.mean(axis = <0 row, 1 col>, skipna (Skip Na values when computing) -> returns DataFrame | Series

8. **Group DataFrame based on columns or series.**

   DataFrame.groupby(by = <label, list of labels, function, mapping>) -> groupby object



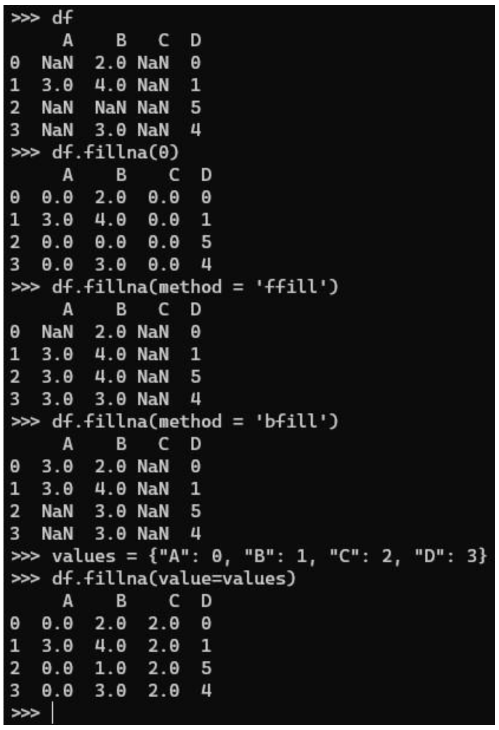
```
>>> df
   value name
0    1.0    A
1    NaN    A
2    NaN    B
3    2.0    B
4    3.0    B
5    1.0    B
6    3.0    C
7    NaN    C
8    3.0    C
>>> g = df.groupby("name")
>>> g
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000247C9DB3640>
>>> for grp, grpdata in g:
...   print(grp, grpdata)
...
A    value name
0    1.0    A
1    NaN    A
B    value name
2    NaN    B
3    2.0    B
4    3.0    B
5    1.0    B
C    value name
6    3.0    C
7    NaN    C
8    3.0    C
```

9. **Filling NaN values**

DataFrame.fillna(value) - > returns DataFrame | None

value -> scalar, dict, Series, DataFrame

inplace -> boolean to modify DataFrame inplace (returns None when set)



```
>>> df
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
3  NaN  3.0 NaN  4
>>> df.fillna(0)
     A    B    C  D
0  0.0  2.0  0.0  0
1  3.0  4.0  0.0  1
2  0.0  0.0  0.0  5
3  0.0  3.0  0.0  4
>>> df.fillna(method = 'ffill')
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  3.0  4.0 NaN  5
3  3.0  3.0 NaN  4
>>> df.fillna(method = 'bfill')
     A    B   C  D
0  3.0  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  3.0 NaN  5
3  NaN  3.0 NaN  4
>>> values = {"A": 0, "B": 1, "C": 2, "D": 3}
>>> df.fillna(value=values)
     A    B    C  D
0  0.0  2.0  2.0  0
1  3.0  4.0  2.0  1
2  0.0  1.0  2.0  5
3  0.0  3.0  2.0  4
>>> 
```

<u>week1.py</u>

- create_numpy_ones_array(shape)
  - Input : tuple (x,y)
  - Output: numpy array of the shape (x,y) with 1 at all position

- create_numpy_zeros_array(shape)
  - Input : tuple (x,y)
  - Output: numpy array of the shape (x,y) with 0 at all position

- create_identity_numpy_array(order)
  - Input : int
  - Output: Identity matrix in the form of numpy array of dimension order x order

- matrix_cofactor(array)

    - Input: numpy array
    - Output: cofactor matrix of the input in the form of numpy array

- f1(X1,coef1,X2,coef2,seed1,seed2,seed3,shape1,shape2)
    - Input: (numpy array, int ,numpy array, int , int , int , tuple,tuple)

    - Perform 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
    - if dimension mismatch occur return -1

    - Output: computed function(numpy array) or -1

- fill_with_mode(filename, column)
    - Input: (str, str)
    - Fill the missing values(NaN) in a column with the mode of that column
    - output: df: Pandas DataFrame object.(Representing entire data and where 'column' does not contain NaN values)

- fill_with_group_average(df, group, column)
    - Input: (DataFrame,str, str)
    - Fill the missing values(NaN) in ‘column’ with the mean value of the group the row belongs to.
    - output: df: Pandas DataFrame object.(Representing entire data and where 'column' does not contain NaN values)

- get_rows_greater_than_avg(df, column)
    - Input: (DataFrame, str)
    - Return all the rows(with all columns) where the value in a certain 'column' is greater than the average value of that column.
    - output: df: Pandas DataFrame object.

SampleTest.py

1. This will help you check your code.
2. Passing the cases in this does not ensure full marks ,you will need to take care of edge cases
3. Name your code file as YOUR_SRN.py
4. Run the command **python3 SampleTest.py --SRN YOUR_SRN (incase of any import error use the below command)**

   python3.7 SampleTest.py --SRN YOUR_SRN