Data Mining - Lab - 2

Numpy & Perform Data Exploration with Pandas

Numpy

1) NumPy (Numerical Python) is a powerful open-source library in Python used for numerical and scientific computing.
2) It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them efficiently.
3) NumPy is highly optimized and written in C, making it much faster than using regular Python lists for numerical operations.
4) It serves as the foundation for many other Python libraries in data science and machine learning, like pandas, TensorFlow, and scikit-learn.
5) With features like broadcasting, vectorization, and integration with C/C++ code, NumPy allows for cleaner and faster code in numerical computations.

Step 1. Import the Numpy library

In [2]:

```
import numpy as np
```

Step 2. Create a 1D array of numbers

In [6]:

```
a=np.arange(11)
print(a)
print(type(a))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10]
<class 'numpy.ndarray'>
```

In [4]:

```
a = np.arange(2,9)
a
```

Out[4]:

```
array([2, 3, 4, 5, 6, 7, 8])
```

Step 3. Reshape 1D to 2D Array

In [7]:

```
a = np.arange(12).reshape(3,4)
a
```

Out[7]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Step 4. Create a Linspace array

In [10]:

```
np.linspace(0,5,20)
```

Out[10]:

```
array([0.        , 0.26315789, 0.52631579, 0.78947368, 1.05263158,
       1.31578947, 1.57894737, 1.84210526, 2.10526316, 2.36842105,
       2.63157895, 2.89473684, 3.15789474, 3.42105263, 3.68421053,
       3.94736842, 4.21052632, 4.47368421, 4.73684211, 5.        
])
```

Step 5. Create a Random Numbered Array

In [74]:

```
np.random.rand(2)
```

Out[74]:

```
array([0.83463274, 0.17866857])
```

In [75]:

```
np.random.rand(2,4)
```

Out[75]:

```
array([[0.08750821, 0.68581566, 0.14007458, 0.39346818],
       [0.31916189, 0.67722915, 0.81519024, 0.51836977]])
```

Step 6. Create a Random Integer Array

In [79]:

```
np.random.randint(1,100,size=10)
```

Out[79]:

```
array([58, 32, 52, 81, 10, 42,  3, 94, 45, 76])
```

In [78]:

```
np.random.randint(1,100,size=(2,4))
```

Out[78]:

```
array([[65, 95, 93, 71],
       [92, 83, 98, 59]])
```

Step 7. Create a 1D Array and get Max,Min,ArgMax,ArgMin

In [80]:

```
arr = np.random.randint(1,100,size=10)
arr
```

Out[80]:

```
array([89, 26, 63, 69,  1, 28,  7, 87, 86, 90])
```

```
In [81]: arr.max()

Out[81]: 90
```

```
In [82]: arr.min()

Out[82]: 1
```

```
In [83]: arr.argmax()

Out[83]: 9
```

```
In [84]: arr.argmin()

Out[84]: 4
```

Step 8. Indexing in 1D Array

```
In [85]: arr[8]

Out[85]: 86
```

```
In [86]: arr[1:5]

Out[86]: array([26, 63, 69,  1])
```

Step 9. Indexing in 2D Array

```
In [87]: arr2d=arr.reshape(2,5)
         arr2d

Out[87]: array([[89, 26, 63, 69,  1],
                [28,  7, 87, 86, 90]])
```

```
In [88]: arr2d[0]

Out[88]: array([89, 26, 63, 69,  1])
```

```
In [89]: arr2d[0][1]
         arr2d[0,1]

Out[89]: 26
```

```
In [90]: arr2d[:1,2:]

Out[90]: array([[63, 69,  1]])
```

Step 10. Conditional Selection

```
In [91]: arr[arr>4]
```

```
Out[91]: array([89, 26, 63, 69, 28,  7, 87, 86, 90])
```

```
In [92]: arr2d[arr2d>2]
```

```
Out[92]: array([89, 26, 63, 69, 28,  7, 87, 86, 90])
```

🔥You did it! 10 exercises down — you're on fire! 🔥

Pandas

Step 1. Import the necessary libraries

```
In [1]: import pandas as pd
```

Step 2. Import the dataset from this address (https://raw.githubusercontent.com/justmarkham/DAT8/master/da

Step 3. Assign it to a variable called users and use the 'user_id' as index

```
In [2]: users = pd.read_csv('https://raw.githubusercontent.com/justmarkham/
```

Step 4. See the first 25 entries

In [3]:
```
users.head(25)
```

13	47	M	educator	29206
14	45	M	scientist	55106
15	49	F	educator	97301
16	21	M	entertainment	10309
17	30	M	programmer	06355
18	35	F	other	37212
19	40	M	librarian	02138
20	42	F	homemaker	95660
21	26	M	writer	30068
22	25	M	writer	40206
23	30	F	artist	48197
24	21	F	artist	94533
25	39	M	engineer	55107

Step 5. See the last 10 entries

In [4]:
```
users.tail(10)
```

Out[4]:

	age	gender	occupation	zip_code
user_id				
934	61	M	engineer	22902
935	42	M	doctor	66221
936	24	M	other	32789
937	48	M	educator	98072
938	38	F	technician	55038
939	26	F	student	33319
940	32	M	administrator	02215
941	20	M	student	97229
942	48	F	librarian	78209
943	22	M	student	77841

Step 6. What is the number of observations in the dataset?

In [5]:
```
users.shape[0]
```

Out[5]:
```
943
```

Step 7. What is the number of columns in the dataset?

In [6]:
```
users.shape[1]
```

Out[6]:
```
4
```

Step 8. Print the name of all the columns.

In [7]:
```
users.columns
```

Out[7]:
```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

In [8]:
```
# "the index" (aka "the labels")
users.index
```

Out[8]:
```
Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
       ...
       934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
      dtype='int64', name='user_id', length=943)
```

Step 10. What is the data type of each column?

In [9]:
```
users.dtypes
```

Out[9]:
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

```
In [10]: users['occupation']
```

```
Out[10]: user_id
         1         technician
         2              other
         3             writer
         4         technician
         5              other
                    ...
         939          student
         940    administrator
         941          student
         942        librarian
         943          student
         Name: occupation, Length: 943, dtype: object
```

Step 12. How many different occupations are in this dataset?

```
In [11]: users.occupation.nunique()
```

```
Out[11]: 21
```

Step 13. What is the most frequent occupation?

```
In [12]: users.occupation.value_counts().head(1)
```

```
Out[12]: occupation
         student    196
         Name: count, dtype: int64
```

Step 14. Summarize the DataFrame.

```
In [13]: users.describe()
```

```
Out[13]:
                  age
       count  943.000000
        mean   34.051962
         std   12.192740
         min    7.000000
         25%   25.000000
         50%   31.000000
         75%   43.000000
         max   73.000000
```

Step 15. Summarize all the columns

```
In [14]: users.describe(include='all')
```

```
Out[14]:
                  age  gender  occupation  zip_code
       count  943.000000    943         943       943
      unique         NaN      2          21       795
         top         NaN      M     student     55414
        freq         NaN    670         196         9
        mean   34.051962    NaN         NaN       NaN
         std   12.192740    NaN         NaN       NaN
         min    7.000000    NaN         NaN       NaN
         25%   25.000000    NaN         NaN       NaN
         50%   31.000000    NaN         NaN       NaN
         75%   43.000000    NaN         NaN       NaN
         max   73.000000    NaN         NaN       NaN
```

Step 16. Summarize only the occupation column

```
In [15]: users['occupation'].describe()
```

```
Out[15]: count          943
         unique          21
         top        student
         freq           196
         Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

```
In [16]: users.age.mean()
```

```
Out[16]: 34.05196182396607
```

Step 18. What is the age with least occurrence?

```
In [17]: users.age.value_counts().tail()
```

```
Out[17]: age
         7     1
         66    1
         11    1
         10    1
         73    1
         Name: count, dtype: int64
```

You're not just learning, you're mastering it. Keep aiming higher! 🚀

```
In [ ]:
```