Experiment No. 3

Title: Exploratory data analysis using NUMPY

Batch: B-1 **Roll No: 16010422234** **Name: Chandana Ramesh galgali**

Experiment No.:3

Aim: To perform exploratory data analysis using python NUMPY

Resources needed: Python IDE

Theory:

• Data Analysis is basically where you use statistics and probability to figure out trends in the data set. It helps you to sort out the "real" trends from the statistical noise

• Exploratory Data Analysis (EDA) in Python is the first step in your data analysis process developed by "John Tukey" in the 1970s.

• In statistics, exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods.

• The main aim of exploratory data analysis is to obtain confidence in your data to an extent where you're ready to engage a machine learning algorithm.

Basically we do the following things in EDA.

1) Quickly describe a dataset; number of rows/columns, missing data, data types, preview.

2) Clean corrupted data; handle missing data, invalid data types, incorrect values.

3) Visualize data distributions; bar charts, histograms, box plots.

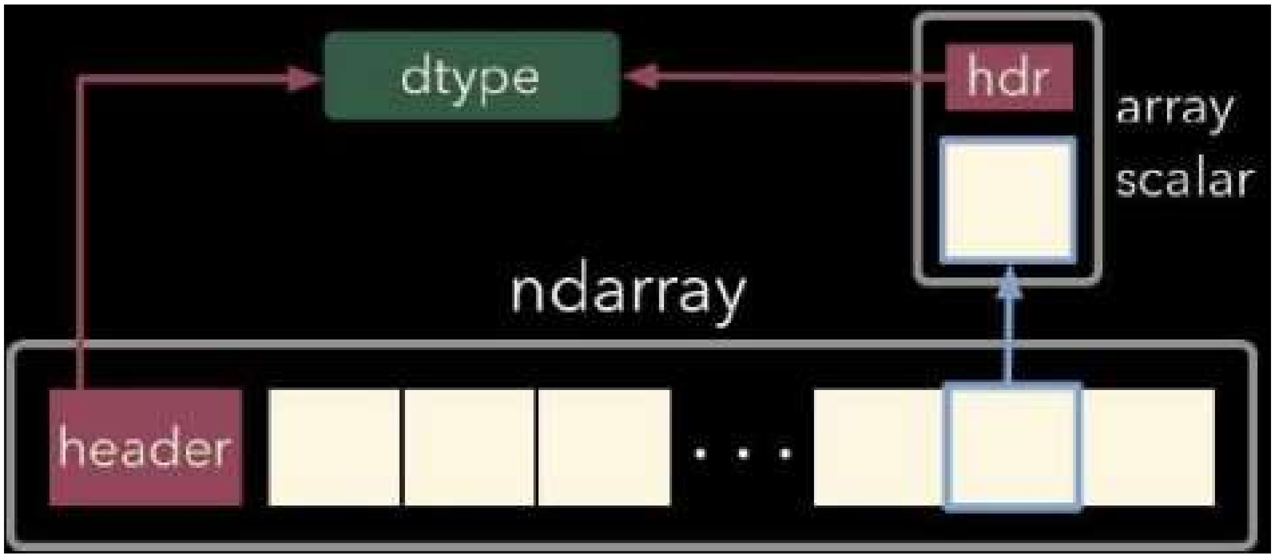
4) Calculate and visualize correlations (relationships) between variables;

NUMPY(Numeric or Numerical Python):

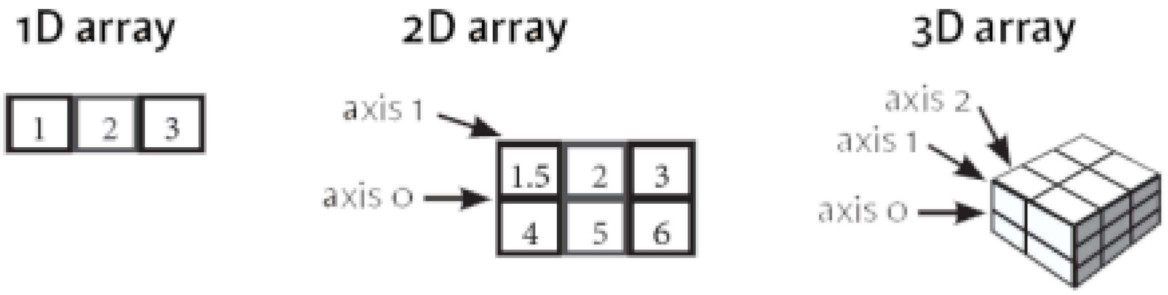
- NumPy is a Python library that is the core library for scientific computing in Python.

- It contains a collection of tools and techniques that can be used to solve on a computer, mathematical models of problems in Science and Engineering.

- One of these tools is a high-performance multidimensional array object, ndarray, that is a powerful data structure for efficient computation of arrays and matrices. Memory layout of ndarray is shown in figure below.



Memory layout of ndarrary of python



1D, 2D and 3D arrays in numpy

To work with these arrays, there's a vast amount of high-level mathematical functions that operate on these matrices and arrays.

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*.

For example, the coordinates of a point in 3D space [1, 2, 1] has one axis. That axis has 3 elements in it, so we say it has a length of 3. In the example pictured below, the array has 2 axes. The first axis has a length of 2, the second axis has a length of 3.

[[1., 0., 0.], [0., 1., 2.]]

NumPy's array class is called ndarray. It is also known by the alias array.

numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality.

ndarray.ndim the number of axes (dimensions) of the array.

The more important attributes of an ndarray object are:

ndarray.ndim the number of axes (dimensions) of the array.

ndarray.shape the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with *n* rows and *m* columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

ndarray.size the total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype an object describing the type of the elements in the array. One can create or specify dtype using standard Python types.

ndarray.itemsize the size in bytes of each element of the array.

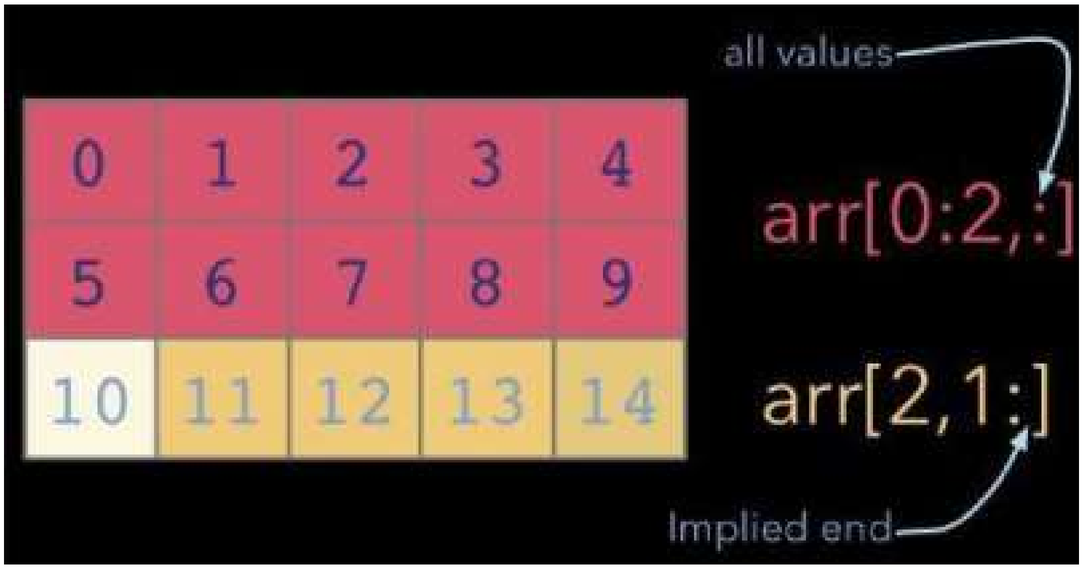
ndarray.data the buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

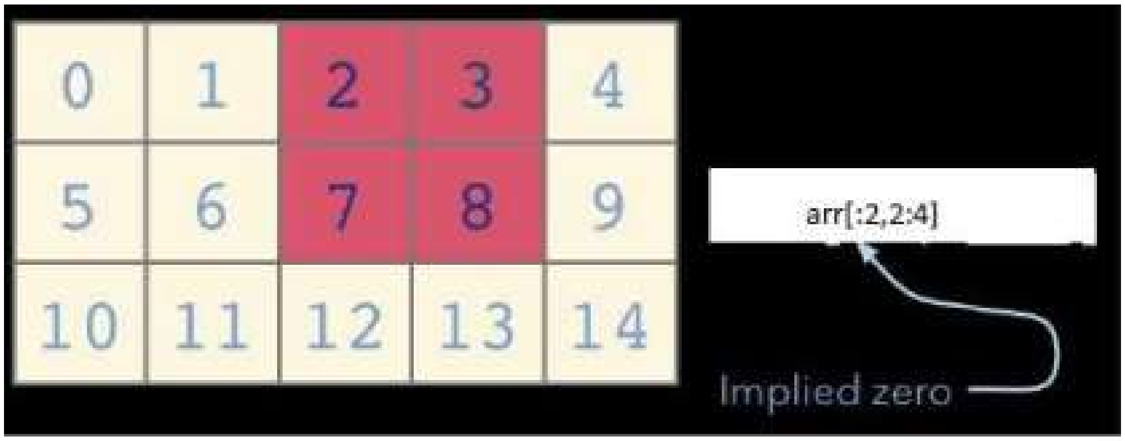
Using numpy.ones(), numpy.zeros(), numpy.empty() we can create standard arrays of ones, zeros and uninitialized numbers respectively.

We can create an array from a list of homogeneous numbers as well.

Slicing and indexing in numpy arrays:

The figure below gives an idea about slicing and indexing in a numpy array.





In order to use ndarray and its related attributes and functions, we first have to make sure that numpy is installed. Since numpy is a basic library of python it comes along with most of the python IDE. In case it is not installed we can download the latest wheel of numpy and install it using pip install.

Once it is installed using the following statement it can be imported and its functionalities can be used:

import numpy as nd

#creating array of zeros

np.zeros(5, float)

Similarly we can use the following functions to find statistical measures using ndarray. x.sum(),x.mean(),x.min(0,x.max() etc

one can pass axis=0 or axis=1 to do columnwise and rowwise operations.

reshape() function will resize the array as per new dimensions passed as an argument to it. vstack() and vstack() for concatenation of two compatible arrays

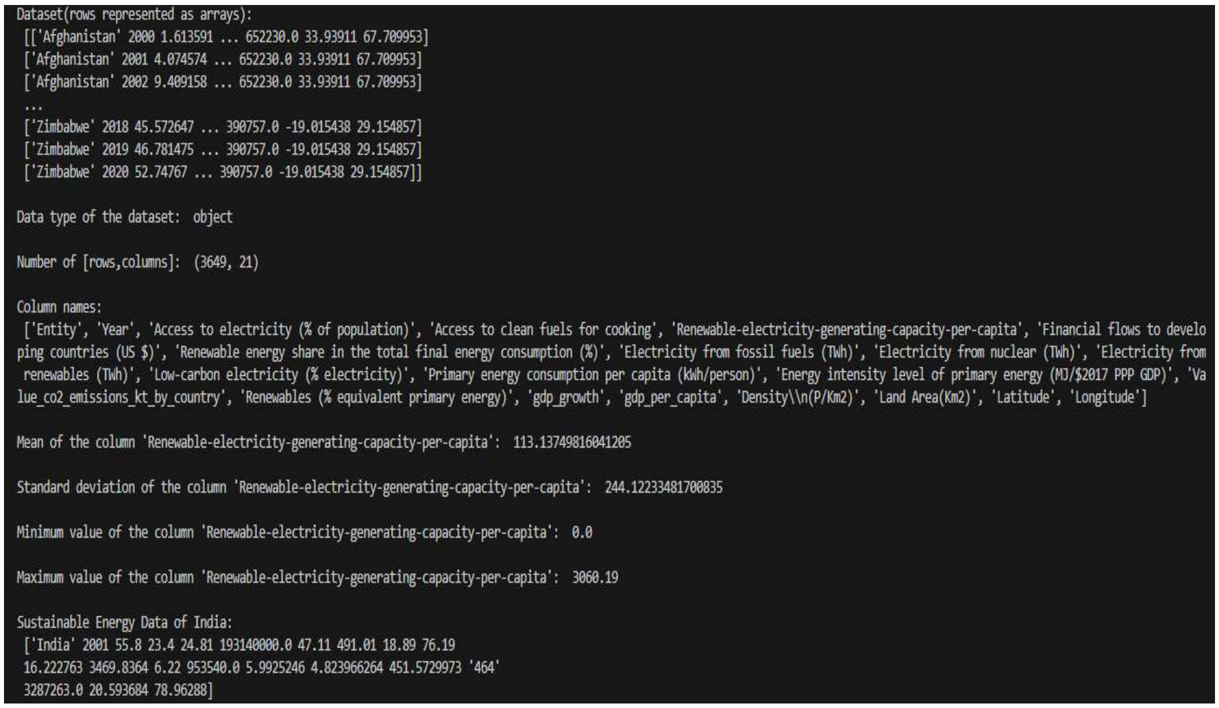
various matrix operations like add(), subtract(),multiply(), divide(), dot() can be performed on 2D arrays in numpy. Numpy allows broadcasting of arrays for incompatible dimensions which will help while performing these operations.

Activities:

1. Download data set with at least 1500 rows and 10-20 columns(numeric and non numeric) from valid data sources
2. Perform in detail Exploratory data analysis of this dataset
3. Write down a description of your dataset based on analysis done in activity
4. Write at least 5 different types of conclusions on your dataset

Result: (script and output)

```
import pandas as pd

import numpy as np

df =
pd.read_csv(r'C:\Users\EXAM.DESKTOP-6KL69TL\Desktop\16010422234-chandana\Gl
obalDataOnSustainableEnergy.csv')

data_array = df.to_numpy()

print("Dataset(rows represented as arrays):\n",data_array)

print("\nData type of the dataset: ",data_array.dtype)

print("\nNumber of [rows,columns]: ",data_array.shape)

column_names = df.columns.tolist()

print("\nColumn names:\n",column_names)

column_name = "Renewable-electricity-generating-capacity-per-capita"

numeric_column = pd.to_numeric(df[column_name], errors='coerce')

mean_value = np.mean(numeric_column)

print("\nMean of the column
'Renewable-electricity-generating-capacity-per-capita': ", mean_value)

std_value = np.std(numeric_column)

print("\nStandard deviation of the column
'Renewable-electricity-generating-capacity-per-capita': ", std_value)

min_value = np.min(numeric_column)

print("\nMinimum value of the column
'Renewable-electricity-generating-capacity-per-capita': ", min_value)

max_value = np.max(numeric_column)

print("\nMaximum value of the column
'Renewable-electricity-generating-capacity-per-capita': ", max_value)

print("\nSustainable Energy Data of India:\n",data_array[1535,])
```



```
Dataset(rows represented as arrays):
 [['Afghanistan' 2000 1.613591 ... 652230.0 33.93911 67.709953]
 ['Afghanistan' 2001 4.074574 ... 652230.0 33.93911 67.709953]
 ['Afghanistan' 2002 9.409158 ... 652230.0 33.93911 67.709953]
 ...
 ['Zimbabwe' 2018 45.572647 ... 390757.0 -19.015438 29.154857]
 ['Zimbabwe' 2019 46.781475 ... 390757.0 -19.015438 29.154857]
 ['Zimbabwe' 2020 52.74767 ... 390757.0 -19.015438 29.154857]]

Data type of the dataset:  object

Number of [rows,columns]:  (3649, 21)

Column names:
 ['Entity', 'Year', 'Access to electricity (% of population)', 'Access to clean fuels for cooking', 'Renewable-electricity-generating-capacity-per-capita', 'Financial flows to develo
ping countries (US $)', 'Renewable energy share in the total final energy consumption (%)', 'Electricity from fossil fuels (TWh)', 'Electricity from nuclear (TWh)', 'Electricity from
 renewables (TWh)', 'Low-carbon electricity (% electricity)', 'Primary energy consumption per capita (kWh/person)', 'Energy intensity level of primary energy (MJ/$2017 PPP GDP)', 'Va
lue_co2_emissions_kt_by_country', 'Renewables (% equivalent primary energy)', 'gdp_growth', 'gdp_per_capita', 'Density\\n(P/Km2)', 'Land Area(Km2)', 'Latitude', 'Longitude']

Mean of the column 'Renewable-electricity-generating-capacity-per-capita':  113.13749816041205

Standard deviation of the column 'Renewable-electricity-generating-capacity-per-capita':  244.12233481700835

Minimum value of the column 'Renewable-electricity-generating-capacity-per-capita':  0.0

Maximum value of the column 'Renewable-electricity-generating-capacity-per-capita':  3060.19

Sustainable Energy Data of India:
 ['India' 2001 55.8 23.4 24.81 193140000.0 47.11 491.01 18.89 76.19
 16.222763 3469.8364 6.22 953540.0 5.9925246 4.823966264 451.5729973 '464'
 3287263.0 20.593684 78.96288]
```

- Entity: The name of the country or region for which the data is reported.
- Year: The year for which the data is reported, ranging from 2000 to 2020.
- Access to electricity (% of population): The percentage of population with access to electricity.
- Access to clean fuels for cooking (% of population): The percentage of the population with primary reliance on clean fuels.
- Renewable-electricity-generating-capacity-per-capita: Installed Renewable energy capacity per person
- Financial flows to developing countries (US $): Aid and assistance from developed countries for clean energy projects.
- Renewable energy share in total final energy consumption (%): Percentage of renewable energy in final energy consumption.
- Electricity from fossil fuels (TWh): Electricity generated from fossil fuels (coal, oil, gas) in terawatt-hours.
- Electricity from nuclear (TWh): Electricity generated from nuclear power in terawatt-hours.
- Electricity from renewables (TWh): Electricity generated from renewable sources (hydro, solar, wind, etc.) in terawatt-hours.
- Low-carbon electricity (% electricity): Percentage of electricity from low-carbon sources (nuclear and renewables).
- Primary energy consumption per capita (kWh/person): Energy consumption per person in kilowatt-hours.
- Energy intensity level of primary energy (MJ/$2011 PPP GDP): Energy use per unit of GDP at purchasing power parity.

- Value_co2_emissions (metric tons per capita): Carbon dioxide emissions per person in metric tons.
- Renewables (% equivalent primary energy): Equivalent primary energy that is derived from renewable sources.
- GDP growth (annual %): Annual GDP growth rate based on constant local currency.
- GDP per capita: Gross domestic product per person.
- Density (P/Km2): Population density in persons per square kilometer.
- Land Area (Km2): Total land area in square kilometers.
- Latitude: Latitude of the country's centroid in decimal degrees.
- Longitude: Longitude of the country's centroid in decimal degrees.

This comprehensive dataset showcases sustainable energy indicators and other useful factors across all countries from 2000 to 2020. Dive into vital aspects such as electricity access, renewable energy, carbon emissions, energy intensity, Financial flows, and economic growth. It compares nations, tracks progress towards Sustainable Development Goal 7, and helps gain profound insights into global energy consumption patterns over time.

Potential Use cases

- Energy Consumption Prediction: Predict future energy usage, aid planning, and track SDG 7 progress.
- Carbon Emission Forecasting: Forecast CO2 emissions, support climate strategies.
- Energy Access Classification: Categorize regions for infrastructure development, understand sustainable energy's role.
- Sustainable Development Goal Tracking: Monitor progress towards Goal 7, evaluate policy impact.
- Energy Equity Analysis: Analyze access, density, and growth for equitable distribution.
- Energy Efficiency Optimization: Identify intensive areas for environmental impact reduction.
- Renewable Energy Potential Assessment: Identify regions for green investments based on capacity.
- Renewable Energy Investment Strategies: Guide investors towards sustainable opportunities.

Outcomes:

Inculcate the knowledge of python libraries like numpy, pandas, matplotlib for scientific- computing and data visualization.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

The experiment successfully showcased the importance and utility of NumPy in exploratory data analysis. The outcomes achieved through this experiment contribute to a deeper understanding of the dataset and provide a solid foundation for further analysis and decision-making processes.

References:

1. https://www.geeksforgeeks.org/python-numpy/