Lab-05: Introduction to NumPy, Pandas, Scikit-learn and Matplotlib

5.1 Objectives:

1. To learn about Python most widely used libraries in machine learning

5.2 NumPy

NumPy is the cornerstone toolbox for scientific computing with Python. NumPy provides, among other things, support for multidimensional arrays with basic operations on them and useful linear algebra functions. Many toolboxes use the NumPy array representations as an efficient basic data structure.

Examples

#importing numpy package

```
mport numpy as np
b=np.array([[[1,2,3,5],[2,3,4,4]],[[1,2,3,5],[2,3,4,4]]])

#printing the data type
print(b.dtype)
out: int32

#printing the dimension of the NumPy array
print(b.ndim)
Out: 3

#printing the shape the NumPy array
print(b.shape)
Out: (2, 2, 4)

# printing the size the NumPy array i.e. total number of elements
print(b.size)
out: 16

#to generate an array of numerical numbers from 10 to 100 with 2 steps
c=np.arange(10,100,2)
print(c)

#to generate an array of zeros
b=np.zeros(10)
print(b)

#to generate a array of ones
b=np.ones(10)
print(b)
```

```
#to shuffle data
a=np.random.permutation(c)
print(a)

#to generate random uniform noise
d=np.random.rand(1000)

#to generate random gaussian noise
d=np.random.randn(1000)

#to select random integer
f=np.random.randint(10,40)

#to reshape an NumPy array
d=np.arange(20).reshape(4,5)
print(d)

#slicing
print(d[1:3,2:4])
```

5.3 SCIKIT-Learn

Scikit-learn is a machine learning library built from NumPy, SciPy, and Matplotlib. Scikit-learn offers simple and efficient tools for common tasks in data analysis such as classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.

Example

#Loading an example dataset

from sklearn import datasets

iris = datasets.load_iris()

digits = datasets.load_digits()

5.4 PANDAS

Pandas5 provides high-performance data structures and data analysis tools. The key feature of Pandas is a fast and efficient DataFrame object for data manipulation with integrated indexing. The DataFrame structure can be seen as a spreadsheet which offers very flexible ways of working with it. You can easily transform any dataset in the way you want, by reshaping it and adding or removing columns or rows. It also provides high-performance functions for aggregating, merging, and joining datasets. Pandas also has tools for importing and exporting data from different formats: comma-separated value (CSV), text files, Microsoft Excel, SQL databases, and the fast HDF5 format. In many situations, the data you have in such formats will not be complete or totally structured. For such cases, Pandas offers handling of missing data and intelligent data alignment. Furthermore, Pandas provides a convenient Matplotlib interface.

Examples

```
#importing pandas library
import pandas as pd
```

```python
#creating Pandas series
a=pd.Series([1,2,3,4],index=['a','b','c','d'])
print(a)

marks={"A":10,"B":20,"C":30}
print(marks)
grades={"A":2,"B":3,"C":5}

#converting dictionaries to the Pandas series
pd1=pd.Series(marks)
print(pd1)
pd2=pd.Series(grades)
#print(marks)
#print(pd1)
#Pandas DataFrame
pd3=pd.DataFrame({"marks":pd1,"grades":pd2})
print(pd3)

#adding dictionary to the Pandas DataFrame
pd3["percentage"]=pd3["marks"]/100
print(pd3)

#deleting from Pandas Dataframe
del pd3["percentage"]

#Thresholding
print(pd3[pd3['marks']>10])
print(pd3)
```

5.5 Matplotlib

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits. matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Majority of plotting commands in pyplot have MATLAB analogs with similar arguments.

Example

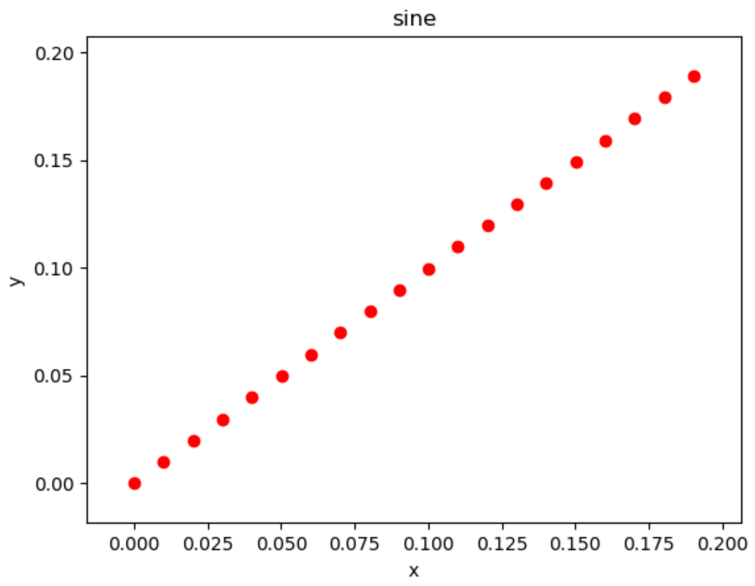
#importing Matplotlib.Pyplot

```python
import matplotlib.pyplot as plt
x=np.linspace(0,10,1000)

#conitonus plotting

plt.plot(x,np.sin(x), color="red")

# Discrete Plotting
plt.scatter(x[0:20],np.sin(x[0:20]), color="red")
plt.xlabel("x")
plt.ylabel("y")
plt.title("sine")
plt.show()
```



TASK 1:

Write a NumPy program to create a random 10x4 array and extract the first five rows of the array and store them into a variable.

TASK 2:

Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
Expected Output:
Number of attempts in the examination is greater than 2:
name score attempts qualify
b Dima 9.0 3 no
d James NaN 3 no
f Michael 20.0 3 yes

TASK 3:

From the sample data given in TASK 2; write a program to calculate the average of the scores. The program should be able to ignore NaN values.

Expected Output:

The average score is : 13.56