AIML

- **What is AI?**

  The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making and translation between languages.

- **What is ML?**

  A computer program is said to learn from experience *E* wrt some class of tasks *T* and performance measure *P*, if its performance at tasks *T*, as measured by *P*, improves with experience *E*. *

- **ML PROCESS**

  Used to build PREDICTIVE MODEL that can be used to find a solution for a problem statement.

1. Define the Objective (the problem)
2. Data Gathering
3. Preparing Data
4. Explorartory Data Analysis
5. Building a ML model
6. Model Evaluation and Optimization
7. Predictions

NumPy

NumPy is a fundamental library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

```
#1D Array
import numpy as np
n1=np.array([10,20,30,40])
print("1D Array:",n1)
#Multi Array
n2 = np.array([[1, 2, 3], [4, 5, 6],[7,8,9],[10,20,30]])
print("Multi Array:\n",n2)
```

```
#Initializing Numpy Array with zeroes
n1=np.zeros((1,2))
print("Zeroes:",n1)
#Initializing Numpy Array with same numbers
n2=np.full((2,2),10)
print("Full:",n2)
#Initializing Numpy Array within a range
n3=np.arange(10,20)
print("Arange:",n3)
#Initializing Numpy Array with Random Number
n4=np.random.randint(1,100,5)
print("Random:",n4)
#Shape
n5=np.array([[1,2,3],[4,5,6]])
print("Shape:",n5.shape)
#Vertical Stack
n6=np.array([10,20,30])
n7=np.array([40,50,60])
print("Vstack:",np.vstack((n6,n7)))
#Horizontal Stack
print("Hstack:",np.hstack((n6,n7)))
#Column Stack
print("ColumnStack:",np.column_stack((n6,n7)))
#Intersection 1D
n8=np.array([10,20,30,40,50,60])
n9=np.array([50,60,70,80,90])
print("Intersection:",np.intersect1d(n8,n9))
#Difference
n10=np.array([10,20,30,40,50,60])
n11=np.array([50,60,70,80,90])
print("Difference:",np.setdiff1d(n10,n11))
#Sum
n12=np.array([10,20])
n13=np.array([30,40])
```

```
print("Sum:",np.sum([n12,n13]))
print("Sum Axis=0:",np.sum([n12,n13],axis=0))
print("Sum Axis=1:",np.sum([n12,n13],axis=1))
#n1+1
n14=np.array([10,20,30])
n14=n14+1
print("n14+1:",n14)
#n1-1
n15=np.array([10,20,30])
n15=n15-1
print("n15-1:",n15)
#n1*2
n16=np.array([10,20,30])
n16=n16*2
print("n16*2:",n16)
#n1/2
n17=np.array([10,20,30])
n17=n17/2
print("n17/2:",n17)



#Initializing array through index
n2 = np.array([[1, 2, 3], [4, 5, 6],[7,8,9],[10,20,30]])
print("Multi Array:\n",n2)
print("Index:",n2[0,1])
#Initializing array through index slicing
print("Slicing:",n2[0:,2])
#Transpose
print("Transpose: \n",n2.transpose())
#Reshape
print("Reshape: \n",n2.reshape(3,4))
#Flatten
print("Flatten:",n2.flatten())
#Dot Product
n1 = np.array([[1, 2, 3], [4, 5, 6],[7,8,9],[10,20,30]])
print("Multi Array 1:\n",n1)
n2 = np.array([[10, 20, 30], [40, 50, 60],[70,80,90],[100,200,300]])
print("Multi Array 2:\n",n2)
print("Dot Product:\n",np.dot(n1,n2.transpose()))
```

PANDA

Pandas is a powerful open-source data analysis and manipulation library for Python, providing data structures like DataFrames and Series for efficient data handling.

```
# 1D-Array
import pandas as pd
s1=pd.Series([1,2,3,4,5])
print("Series:\n",s1)
# 2D-Array
df1=pd.DataFrame({
    "C1":[1,2,3],
    "C2":[4,5,6],
    "C3":[7,8,9]
})
print("DataFrame:\n",df1)
#Changing Index
#1D-Array
s2=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
print("Series:\n",s2)
#2d-Array
df2=pd.DataFrame({
    "C1":[1,2,3],
    "C2":[4,5,6],
    "C3":[7,8,9]
},index=['a','b','c'])
print("DataFrame:\n",df2)
#Series Object from Dictionary
s3=pd.Series({'a':10,'b':20,'c':30})
print("Series:\n",s3)
#Changing the Index position
s4=pd.Series({'a':10,'b':20,'c':30},index=['b','c','d','a'])
print("Series:\n",s4)
#Extracting Single Element
s1=pd.Series([1,2,3,4,5,6,7,8,9])
print("Single Element:\n",s1[3])
#Extracting Sequence of Elements
```

```
print("Sequence of Elements:\n",s1[:4])
print("Sequence of Elements:\n",s1[-5:-1])
#Basic Math Operations
s5=pd.Series([1,2,3])
s6=pd.Series([10,20,30])
print("Addition:\n",s5+s6)
print("Subtraction:\n",s5-s6)
print("Multiplication:\n",s5*s6)
print("Division:\n",s5/s6)
print("Addition of each element:\n",s5+5)
print("Subtraction of each element:\n",s5-9)
print("Multiplication of each element:\n",s5*2)
print("Division of each element:\n",s5/3)



#Dataframe in a 2d Labelled data structure
s3 = pd.DataFrame({"Name":['Ram','Sham','Ananya'],"Marks":[76,25,92]})
print("DataFrame:\n",s3)
```

Iris Dataset Analysis - Descriptive Statistics

This notebook performs basic descriptive statistics on the Iris dataset.

First, the dataset is loaded into a pandas DataFrame.

The following descriptive statistics were calculated for the numeric columns:

- **Mean:** The average value for each column.
- **Median:** The middle value for each column when sorted.
- **Mode:** The most frequent value(s) for each column.
- **Standard Deviation:** A measure of the spread of the data from the mean.
- **Variance:** The average of the squared differences from the mean.
- **Minimum:** The smallest value in each column.
- **Maximum:** The largest value in each column.
- **Count:** The number of non-null values in each column.
- **Describe:** A summary table containing various descriptive statistics.

```
#Read a csv file
import pandas as pd
df=pd.read_csv('/content/Iris.csv')
print("\nTop 5 Columns:\n",df.head(5))
print("\nDescribe:\n", df.describe())


print("\nMean:\n",df.mean(numeric_only=True))


print("Median:\n", df.median(numeric_only=True))
print("\nMode:\n", df.mode(numeric_only=True))


print("Standard Deviation:\n", df.std(numeric_only=True))
print("\nVariance:\n", df.var(numeric_only=True))
print("\nMinimum:\n", df.min(numeric_only=True))
print("\nMaximum:\n", df.max(numeric_only=True))
print("\nCount:\n", df.count())


df.drop([1,2,3],axis=0)#drop row
```

MATPLOTLIB

We used the `matplotlib.pyplot` library to create a basic line plot. The key steps were defining the x and y data points, plotting them using `plt.plot()`, adding a title and axis labels, and finally displaying the plot with `plt.show()`.

```
import numpy as np
from matplotlib import pyplot as plt
x=np.arange(1,11)
x
```

```
y=2*x
y
```

`plt.plot(x, y)` : This is the core plotting function. It takes the x and y arrays as input and draws a line connecting the corresponding points.

`plt.title('Line Plot')` : Sets the title of the plot to 'Line Plot'.

`plt.xlabel('x-axis')` : Labels the x-axis as 'x-axis'.

`plt.ylabel('y-axis')` : Labels the y-axis as 'y-axis'.

`plt.show()` : This command displays the generated plot.

```
plt.plot(x,y)
plt.title('Line Plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

SubPlot

This code generates two subplots side by side using `plt.subplot(1, 2, ...)`.

- The first subplot (`plt.subplot(1, 2, 1)`) plots `y1` against `x` with a violet dashed line.
- The second subplot (`plt.subplot(1, 2, 2)`) plots `y2` against `x` with a pink dotted line.

Both subplots have titles and labeled x and y axes for clarity. `plt.show()` displays the combined figure.

```
x=np.arange(1,11)
y1=2*x
y2=3*x
plt.subplot(1,2,1)
plt.plot(x,y,1,color='violet',linestyle='--',linewidth=8),
plt.title('Subplot 1')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.subplot(1,2,2)
plt.plot(x,y2,color='pink',linestyle=':',linewidth=8)
plt.title('Subplot 2')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

Purojita Singh
