MACHINE LEARNING -ICP#3

Student Name: Manjunath Reddy Velagacherla

Student Id: 700759509

GitHub link: https://github.com/Mxv95090/ICP-3

QUESTION-1:

In [ ]:

```
a. Using NumPy create random vector of size 15 having only Integers in the range 1-20.

1. Reshape the array to 3 by 5
2. Print array shape.
3. Replace the max in each row by 0
Create a 2-dimensional array of size 4 x 3 (composed of 4-byte integer elements), also print the shape, type and
of the array.
```

In [2]:

```
import numpy as np

# Create random vector of size 15 with integers in the range 1-20
random_vector = np.random.randint(1, 21, size=15)

# Reshape the array to 3 by 5
reshaped_array = random_vector.reshape(3, 5)

# Print array shape
print("Array shape:", reshaped_array.shape)

# Replace the max in each row by 0
reshaped_array[np.arange(len(reshaped_array)), np.argmax(reshaped_array, axis=1)] = 0

print("Array after replacing max in each row by 0:")
print(reshaped_array)

# Create a 2-dimensional array of size 4 x 3 with 4-byte integer elements
array_4x3 = np.random.randint(0, 1000, size=(4, 3), dtype=np.int32)

# Print shape, type, and data type of the array
print("\nShape of the array:", array_4x3.shape)
print("Type of the array:", type(array_4x3))
print("Data type of the array:", array_4x3.dtype)
```

```
Array shape: (3, 5)
Array after replacing max in each row by 0:
[[ 3  8  9 10  0]
 [15 10 11  0  1]
 [11  7  6  4  0]]

Shape of the array: (4, 3)
Type of the array: <class 'numpy.ndarray'>
Data type of the array: int32
```

QUESTION - 2:

b. Write a program to compute the eigenvalues and right eigenvectors of a given square array given below: [[3 -2] [1 0]]

In [3]:
```
import numpy as np

# Define the square array
array = np.array([[3, -2],
                  [1, 0]])

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(array)

# Print the eigenvalues and right eigenvectors
print("Eigenvalues:")
print(eigenvalues)
print("\nRight Eigenvectors:")
print(eigenvectors)
```

```
Eigenvalues:
[2. 1.]

Right Eigenvectors:
[[0.89442719 0.70710678]
 [0.4472136  0.70710678]]
```

QUESTION-3:

c. Compute the sum of the diagonal element of a given array. [[0 1 2] [3.45]

In [4]:
```
import numpy as np

# Define the array
array = np.array([[0, 1, 2],
                  [3, 4, 5]])

# Compute the sum of diagonal elements
diagonal_sum = np.trace(array)

# Print the sum
print("Sum of diagonal elements:", diagonal_sum)
```

```
Sum of diagonal elements: 4
```

QUESTION-4:

d. Write a NumPy program to create a new shape to an array without changing its data. Reshape 3x2: [[1 2] [3.4] [5 6] Reshape 2x3: [[1 2 3] [45 6]]

In [5]:
```
import numpy as np

# Define the arrays
array1 = np.array([[1, 2],
                   [3, 4],
                   [5, 6]])

array2 = np.array([[1, 2, 3],
                   [4, 5, 6]])

# Reshape array1 to 3x2
reshaped_array1 = array1.reshape(3, 2)

# Reshape array2 to 2x3
reshaped_array2 = array2.reshape(2, 3)

# Print the reshaped arrays
print("Reshaped array1 (3x2):")
print(reshaped_array1)

print("\nReshaped array2 (2x3):")
print(reshaped_array2)
```

```
Reshaped array1 (3x2):
[[1 2]
 [3 4]
 [5 6]]
```

```
Reshaped array1 (3x2):
[[1 2]
 [3 4]
 [5 6]]

Reshaped array2 (2x3):
[[1 2 3]
 [4 5 6]]
```