Git:- https://github.com/MadhuKumar-k/ICP-3---ML

ML_ICP3.ipynb - Colab

colab.research.google.com/drive/14VpNl1S6wejh_VIaUWJOV_U0N7yucW_s#scrollTo=1VlzSqWy7vbq

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
#Q1

import numpy as np

# Create a random vector of size 15 with integers in the range 1-20
random_vector = n                    =15)
print(random_vect
                    ndarray: random_vector
                    ndarray with shape (15,)
# Reshape the array to 3 by 5
reshaped_array = random_vector.reshape(3, 5)

# Print array shape
print("Array shape before replacing max values:")
print(reshaped_array.shape)

# Replace the max in each row by 0
max_indices = np.argmax(reshaped_array, axis=1)
for i in range(len(reshaped_array)):
    reshaped_array[i, max_indices[i]] = 0

# Print array shape after replacing max values
print("\nArray shape after replacing max values:")
print(reshaped_array.shape)

# Create a 2-dimensional array of size 4 x 3 (composed of 4-byte integer elements)
array_2d = np.random.randint(low=-100, high=100, size=(4, 3),dtype=np.int32)

# Print shape, type, and data type of the array
print("\nArray shape:", array_2d.shape)
print("Array type:", type(array_2d))
print("Array data type:", array_2d.dtype)
```

```
[10 18  7  3  4 13  3 16 18  9  1  5 10 17 19]
Array shape before replacing max values:
(3, 5)

Array shape after replacing max values:
(3, 5)

Array shape: (4, 3)
Array type: <class 'numpy.ndarray'>
Array data type: int32
```

ML_ICP3.ipynb - Colab

colab.research.google.com/drive/14VpNl1S6wejh_VIaUWJOV_U0N7yucW_s#scrollTo=1VlzSqWy7vbq

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
#Q2

import numpy as np

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

# Compute eigenvalues and right 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]]
```

+ Code

```python
[49] #Q3
import numpy as np

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


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

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

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

ML_ICP3.ipynb - Colab

colab.research.google.com/drive/14VpNI1S6wejh_VlaUWJOV_U0N7yucW_s#scrollTo=1VlzSqWy7vbq

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
print("Sum of diagonal elements:", diagonal_sum)
```

[49]

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

```
#Q4

import numpy as np

# Define the arrays
array1 = np.array([[1, 2], [3, 4], [5, 6]])  # 3x2 array
array2 = np.array([[1, 2, 3], [4, 5, 6]])    # 2x3 array

# Reshape array1 to 2x3 without changing its data
reshaped_array1 = np.reshape(array1, (2, 3))

# Reshape array2 to 3x2 without changing its data
reshaped_array2 = np.reshape(array2, (3, 2))

# Print the reshaped arrays
print("Reshaped array1 (2x3):")
print(reshaped_array1)
print("\nReshaped array2 (3x2):")
print(reshaped_array2)
```

```
Reshaped array1 (2x3):
[[1 2 3]
 [4 5 6]]

Reshaped array2 (3x2):
[[1 2]
 [3 4]
 [5 6]]
```