Machine Learning Assignment

Karakoram International University

Department of Computer Sciences

Instructor: Dr. Aftab Ahmed Khan

Submitted by: Kashif Aziz Date: 18th November, 2023 Registraion No.: 2020-KIU-BS2432

==========

Task 1: Lists, Dictonaries, Tuples (10 Marks)

Lists

Given a list:

nums = [3, 4, 7, 8, 15]

Make another list named cubes and append the cubes of the given list in this list and print it.

```
# Given list
nums = [3, 4, 7, 8, 15]

# Create a new list named cubes and append the cubes of nums
cubes = [num**3 for num in nums]

# Print the cubes list
print(cubes)
```

```
[27, 64, 343, 512, 3375]
```

Dictionaries

You are given an empty dictionary:

dic = {}

=> Add the following data to the dictionary: 'person': 2, 'cat': 4, 'spider': 8, 'horse': 4 as key value pairs.

=> Use the 'items' method to loop over the dictionary and print the animals and their corresponding legs.

=> Sum the legs of each animal, and print the total at the end. Tuples

```
# Given empty dictionary
dic = {}

# Add data to the dictionary
```

```python
dic['person'] = 2
dic['cat'] = 4
dic['spider'] = 8
dic['horse'] = 4

# Use 'items' method to loop over the dictionary and print animals and
their legs
for animal, legs in dic.items():
    print(f"{animal}: {legs} legs")

# Sum the legs of each animal
total_legs = sum(dic.values())

# Print the total legs
print(f"Total legs: {total_legs}")
```

```
person: 2 legs
cat: 4 legs
spider: 8 legs
horse: 4 legs
Total legs: 18
```

Given the following tuple:

D = (1,15,4,[5,10])

=> Change the value in the list from ‘5’ to ‘3’.

=> Delete the tuple D. Given another tuple:

E = (‘a’,‘p’,‘p’,‘l’,‘e’)

=> Print the number of occurences of ‘p’ in tuple E.

=> Print the index of ‘l’ in tuple E.

```python
# Given tuple
D = (1, 15, 4, [5, 10])

# Change the value in the list from '5' to '3'
D[3][0] = 3

# Delete the tuple D
del D

# Given another tuple
E = ('a', 'p', 'p', 'l', 'e')

# Print the number of occurrences of 'p' in tuple E
occurrences_p = E.count('p')
print(f"Number of occurrences of 'p': {occurrences_p}")
```

```
# Print the index of 'l' in tuple E
index_l = E.index('l')
print(f"Index of 'l': {index_l}")
```

```
Number of occurrences of 'p': 2
Index of 'l': 3
```

Task 2: Numpy (15 marks)

You should use all built in functions of numpy in this task, a list is available

M = [1 2 3 4,

5 6 7 8,

9 10 11 12]

z = np.array([1, 0, 1])

=> Convert matrix M into numpy array

=> Use slicing to pull out the subarray consisting of the first 2 rows and columns 1 and 2. Store it in b which is a numpy array of shape (2, 2).

=> Create an empty matrix 'y' with the same shape as 'M'.

=> Add the vector z to each column of the matrix M with an explicit loop and store it in y.

Given:

A = np.array([[1,2],[3,4]])

B = np.array([[5,6],[7,8]])

v = np.array([9,10])

=> Add the two matrices A and B.

=> Multiply the two matrices A and B.

=> Take the element wise square root of matrix A.

=> Take the dot product of the matrix A and vector v.

=> Compute sum of each column of A.

=> Print the transpose of B.

```
import numpy as np

# Given matrix M
M = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 10, 11, 12]])
```

```
# Given vector z
z = np.array([1, 0, 1])

# Convert matrix M into a numpy array
M = np.array(M)

# Use slicing to pull out the subarray consisting of the first 2 rows
and columns 1 and 2.
b = M[:2, 1:3]

# Create an empty matrix 'y' with the same shape as 'M'
y = np.empty_like(M)

# Add the vector z to each column of the matrix M with an explicit
loop
for i in range(M.shape[1]):
    y[:, i] = M[:, i] + z

# Given matrices A and B, and vector v
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
v = np.array([9, 10])

# Add the two matrices A and B
addition_result = np.add(A, B)

# Multiply the two matrices A and B
multiplication_result = np.dot(A, B)

# Take the element-wise square root of matrix A
sqrt_A = np.sqrt(A)

# Take the dot product of the matrix A and vector v
dot_product_result = np.dot(A, v)

# Compute the sum of each column of A
column_sum_A = np.sum(A, axis=0)

# Print the transpose of B
transpose_B = np.transpose(B)

# Print the results
print("b:", b)
print("y:", y)
print("Addition Result:", addition_result)
print("Multiplication Result:", multiplication_result)
print("Square Root of A:", sqrt_A)
print("Dot Product of A and v:", dot_product_result)
```

```
print("Column Sum of A:", column_sum_A)
print("Transpose of B:", transpose_B)
```

```
b: [[2 3]
 [6 7]]
y: [[ 2  3  4  5]
 [ 5  6  7  8]
 [10 11 12 13]]
Addition Result: [[ 6  8]
 [10 12]]
Multiplication Result: [[19 22]
 [43 50]]
Square Root of A: [[1.         1.41421356]
 [1.73205081 2.        ]]
Dot Product of A and v: [29 67]
Column Sum of A: [4 6]
Transpose of B: [[5 7]
 [6 8]]
```

Task 3: Functions and For Loops (10 marks)

Function

Declare a function Compute that takes two arguments: distance and time, and use it to calculate velocity.

Forloop

Declare a list even that contains all even numbers up till 16. Declare a function sum that takes the list as an argument and calculates the sum of all entries using a for loop.

```
# Declare a function Compute to calculate velocity
def Compute(distance, time):
    velocity = distance / time
    return velocity

# Example usage:
distance = 50
time = 2
result_velocity = Compute(distance, time)
print(f"Velocity: {result_velocity}")
```

```
Velocity: 25.0
```

```
# Declare a list even that contains all even numbers up till 16
even = [num for num in range(2, 17, 2)]

# Declare a function sum to calculate the sum of all entries using a
for loop
def calculate_sum(lst):
```

```
    total = 0
    for num in lst:
        total += num
    return total

# Example usage:
result_sum = calculate_sum(even)
print(f"Sum of even numbers: {result_sum}")
```

```
Sum of even numbers: 72
```

Task 4: Matplotlib (15 marks)

Import the plotting function by the command:

import matplotlib.pyplot as plt

Plotting a single line

Compute the x and y coordinates for points on a sine curve and plot the points using matplotlib. Use the function plt.show()

Plotting multiple lines

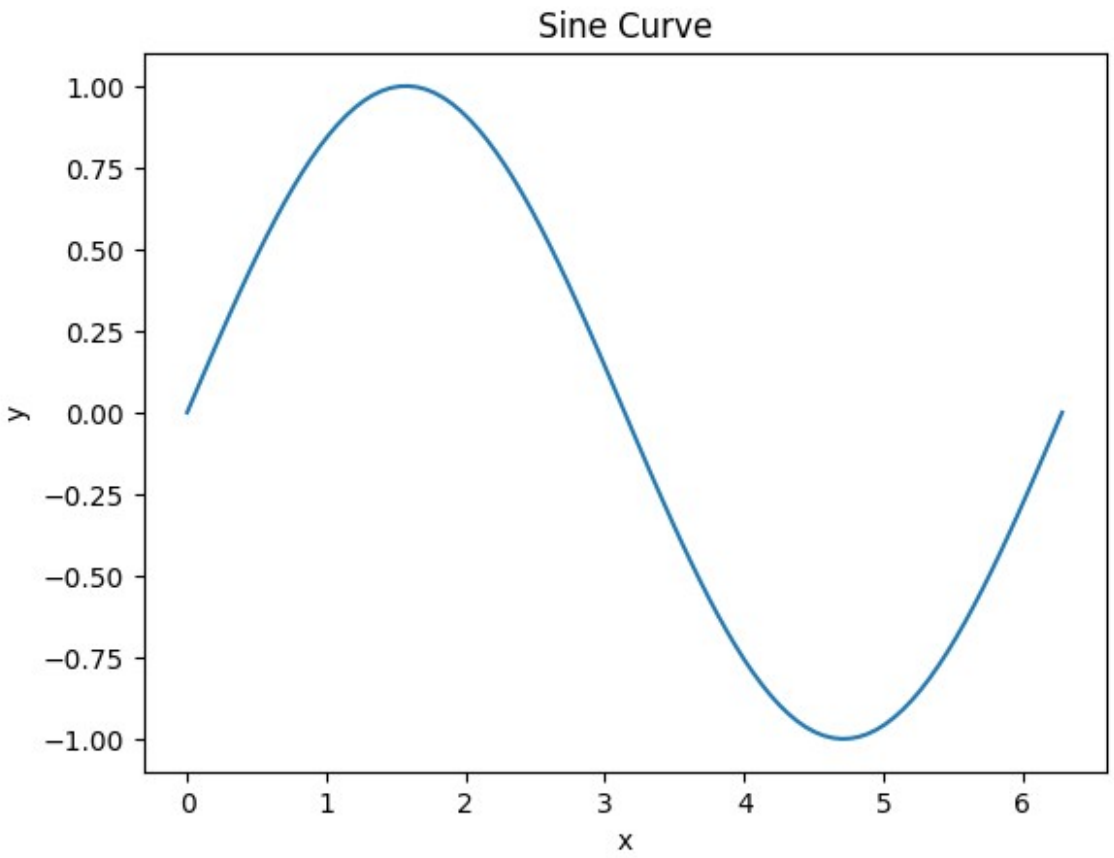
Compute the x and y coordinates for points on sine and cosine curves and plot them on the same graph using matplotlib. Add x and y labels to the graph as well.

Subplots

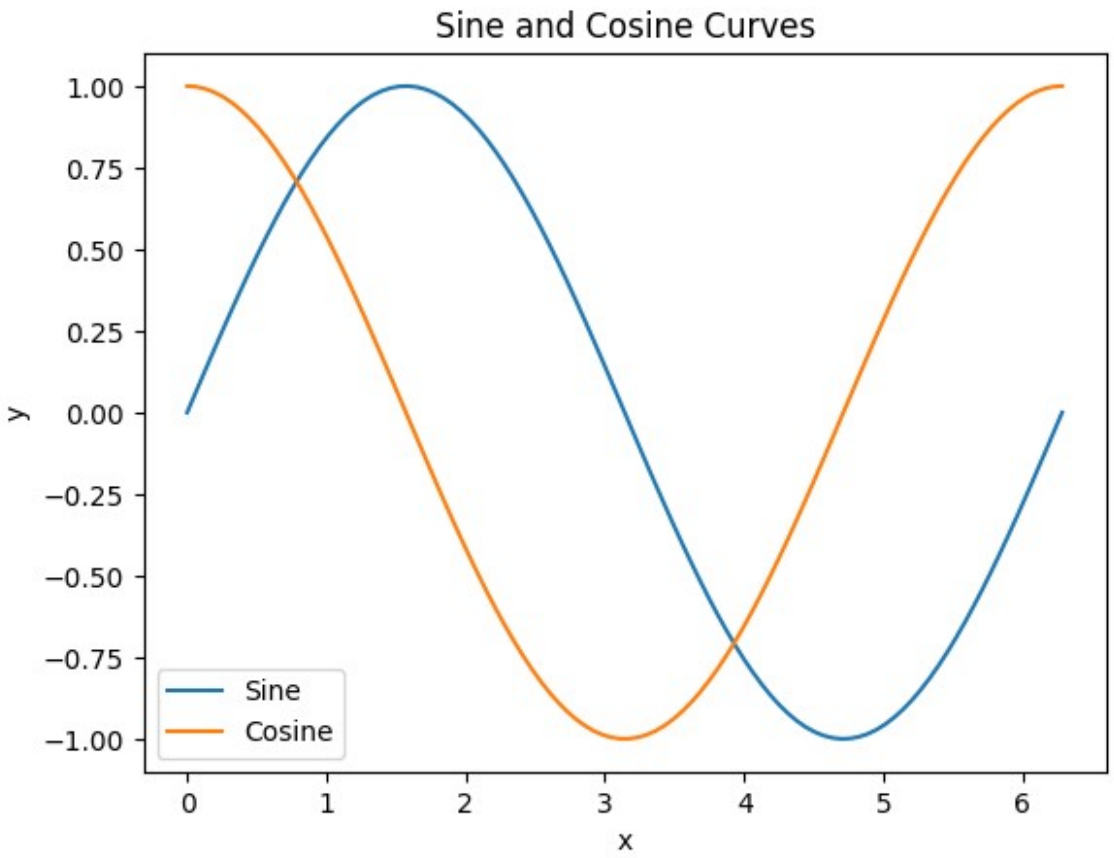
Compute the x and y coordinates for points on sine and cosine curves. Set up a subplot grid that has height 2 and width 1, and set the first such subplot as active. Plot the sine and cosine graphs.

Hint: Use the plt.subplot() function

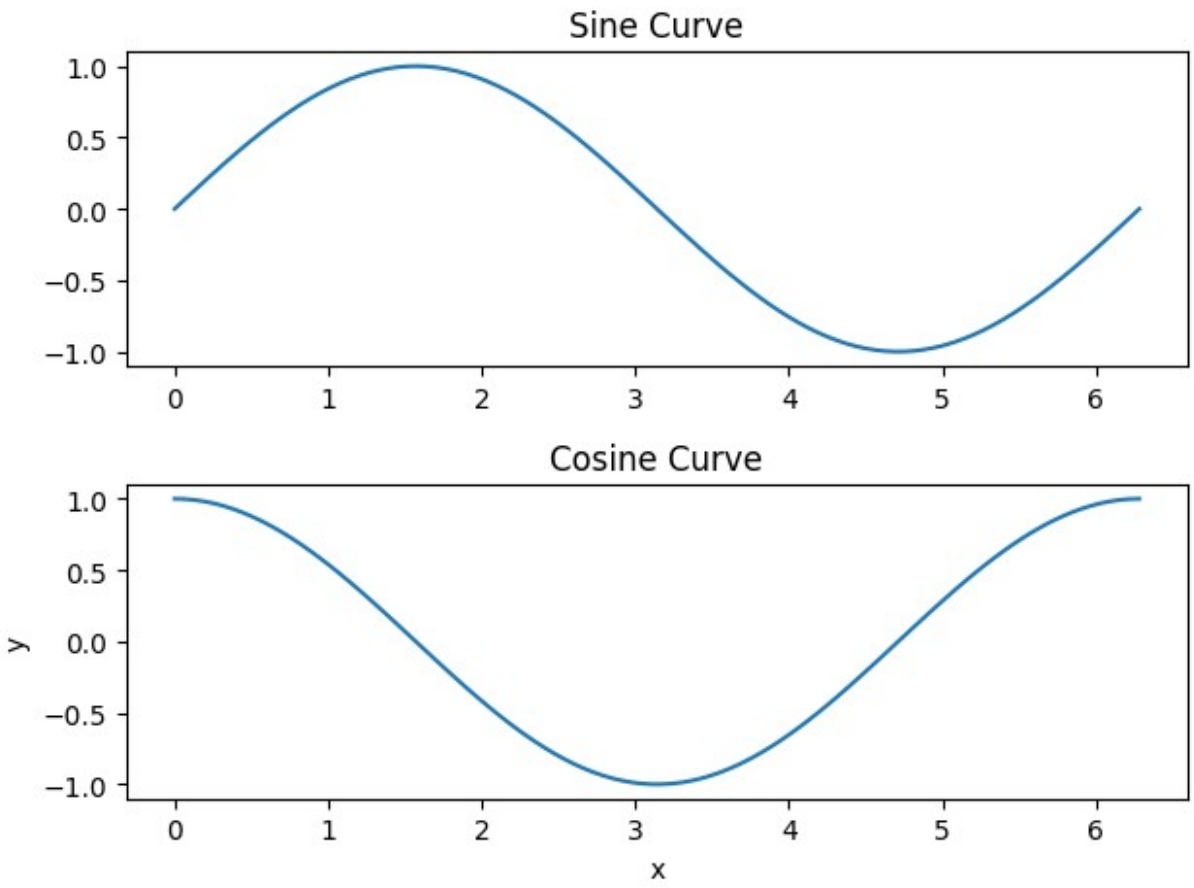
```
import numpy as np
import matplotlib.pyplot as plt

# Compute x and y coordinates for points on a sine curve
x_single = np.linspace(0, 2 * np.pi, 100)
y_single = np.sin(x_single)

# Plot the points
plt.plot(x_single, y_single)
plt.title('Sine Curve')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```




```
# Compute x and y coordinates for points on sine and cosine curves
x_multiple = np.linspace(0, 2 * np.pi, 100)
y_sine = np.sin(x_multiple)
y_cosine = np.cos(x_multiple)

# Plot sine and cosine curves on the same graph
plt.plot(x_multiple, y_sine, label='Sine')
plt.plot(x_multiple, y_cosine, label='Cosine')
plt.title('Sine and Cosine Curves')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```




```
# Compute x and y coordinates for points on sine and cosine curves
x_subplots = np.linspace(0, 2 * np.pi, 100)
y_sine_subplots = np.sin(x_subplots)
y_cosine_subplots = np.cos(x_subplots)

# Set up a subplot grid with height 2 and width 1
plt.subplot(2, 1, 1)
plt.plot(x_subplots, y_sine_subplots)
plt.title('Sine Curve')

plt.subplot(2, 1, 2)
plt.plot(x_subplots, y_cosine_subplots)
plt.title('Cosine Curve')

plt.xlabel('x')
plt.ylabel('y')
plt.tight_layout()
plt.show()
```



Task 5: Pandas DataFrame (15 marks)

Making a DataFrame

Create a dataframe pd that contains 5 rows and 4 columns, similar to the one given below:

Col1 Col2 Col3 Col4

1 6 7 7

2 7 78 5

3 5 78 707

4 5 18 60

5 8 88 4

=> Print only the first two rows of the dataframe.

=> Print the second column.

=> Change the name of the third column from “Col3” to “XYZ”.

=> Add a new column to the dataframe and name it “Sum”.

=> Sum the entries of each row and add the result in the column “Sum”.

```
import pandas as pd

# Create the DataFrame
data = {'Col1': [1, 2, 3, 4, 5],
        'Col2': [6, 7, 5, 18, 8],
        'Col3': [7, 78, 78, 60, 88],
        'Col4': [7, 5, 707, 60, 4]}

df = pd.DataFrame(data)

# Print only the first two rows of the dataframe
print("First two rows:")
print(df.head(2))

# Print the second column
print("\nSecond column:")
print(df['Col2'])

# Change the name of the third column from "Col3" to "XYZ"
df.rename(columns={'Col3': 'XYZ'}, inplace=True)
print("\nDataFrame with column name changed:")
print(df)

# Add a new column to the dataframe and name it "Sum"
df['Sum'] = df.sum(axis=1)

# Print the updated dataframe with the new "Sum" column
print("\nDataFrame with Sum column:")
print(df)
```

```
First two rows:
   Col1  Col2  Col3  Col4
0     1     6     7     7
1     2     7    78     5

Second column:
0     6
1     7
2     5
3    18
4     8
Name: Col2, dtype: int64

DataFrame with column name changed:
   Col1  Col2  XYZ  Col4
0     1     6    7     7
1     2     7   78     5
2     3     5   78   707
3     4    18   60    60
```

```
4     5     8   88     4

DataFrame with Sum column:
   Col1  Col2  XYZ  Col4  Sum
0     1     6    7     7   21
1     2     7   78     5   92
2     3     5   78   707  793
3     4    18   60    60  142
4     5     8   88     4  105
```

Task 6: Image Manipulation (20 marks)

Importing Libraries

Run the following commands to import libraries

import matplotlib

from mpl_toolkits import mplot3d

from matplotlib import pyplot as plt

from matplotlib import cm

from matplotlib import image as mpimg

from matplotlib.pyplot import figure

%matplotlib inline

import seaborn as sns

import numpy as np

import matplotlib.pylab as pl

from matplotlib.colors import ListedColormap

Loading an Image

Read image from file to memory using the given command:

img = np.array(mpimg.imread('path'))

Replace 'path' in the above command with the path of your image e.g. intro/cat.jpg

Now run the command:

img.setflags(write=1)

This allows us to manipulate the image

Tasks

=>Display your image using the plt.imshow function.

=> Crop the image.

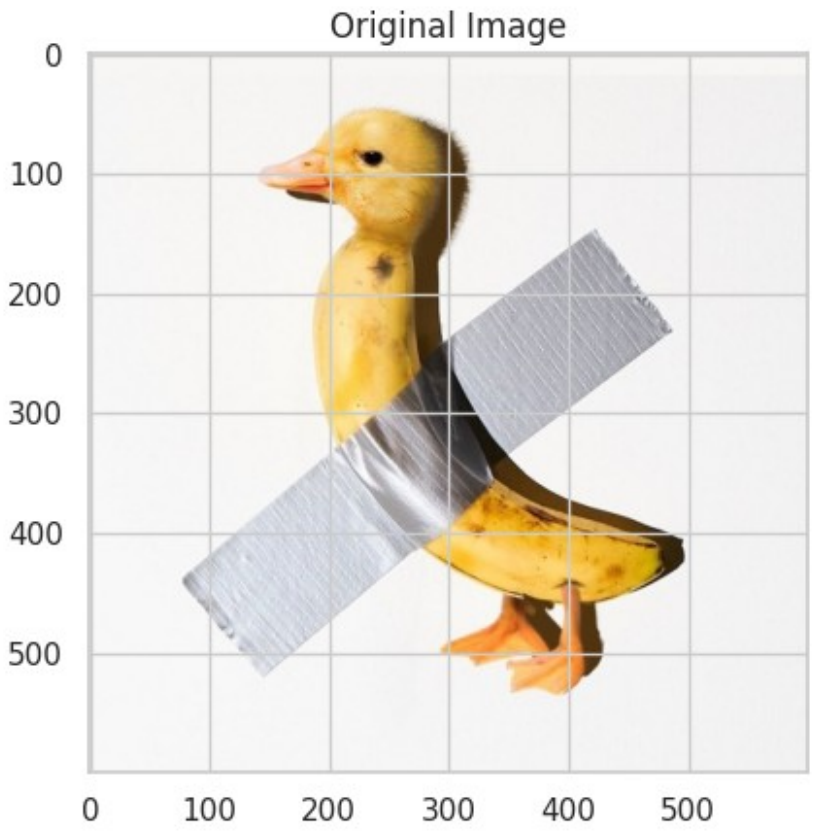
=> Create 50 randomly placed markers on your image. You can check the following link for learning about markers.

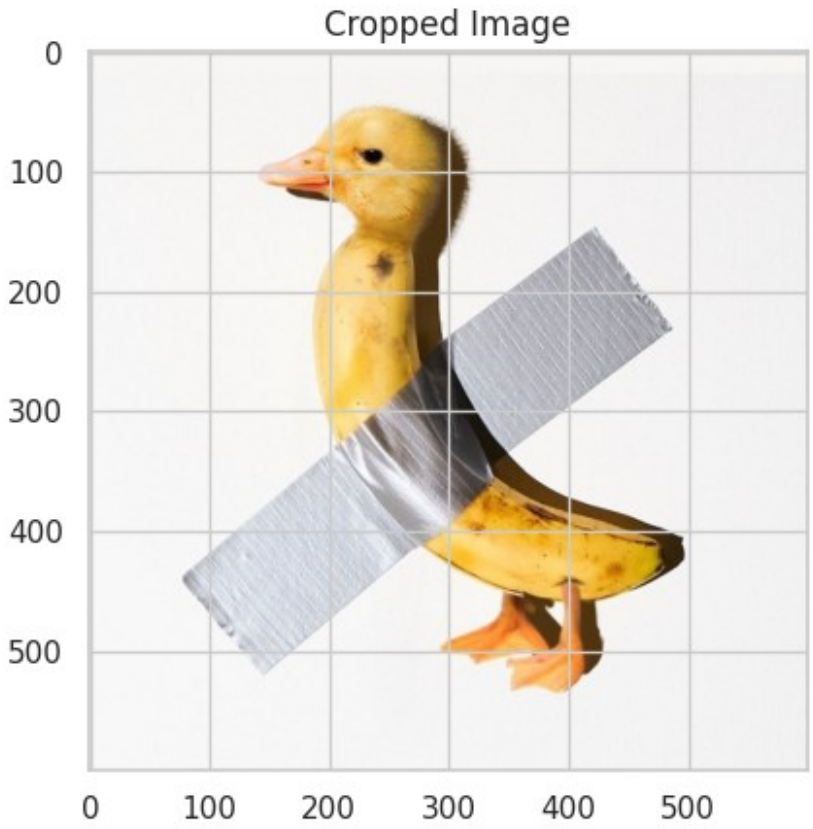
=> Carry out color analysis by accessing the RGB values and plotting them. Use the seaborn library.

=> View only the red values. Hint: Set the blue and green values to 0

```
# Importing Libraries
import matplotlib
from mpl_toolkits import mplot3d
from matplotlib import pyplot as plt
from matplotlib import cm
from matplotlib import image as mpimg
from matplotlib.pyplot import figure
%matplotlib inline
import seaborn as sns
import numpy as np
import matplotlib.pylab as pl
from matplotlib.colors import ListedColormap

# Load Image
img_path = 'myImage.jpg'  # Replace 'path_to_image/myImage.jpg' with
the actual path to your image
img = np.array(mpimg.imread(img_path))
img.setflags(write=1)

# Display Image
plt.imshow(img)
plt.title('Original Image')
plt.show()

# Crop the Image (Assuming a square crop for simplicity)
crop_size = min(img.shape[0], img.shape[1])
cropped_img = img[:crop_size, :crop_size, :]

# Display Cropped Image
plt.imshow(cropped_img)
plt.title('Cropped Image')
plt.show()

# Create 50 randomly placed markers on your image
markers = np.zeros_like(cropped_img)
for _ in range(50):
    x, y = np.random.randint(0, cropped_img.shape[0]),
np.random.randint(0, cropped_img.shape[1])
    markers[x, y, :] = 255

# Display Image with Random Markers
```

```
plt.imshow(markers)
plt.title('Image with Random Markers')
plt.show()

# Color Analysis using Seaborn
# Reshape the image to a 2D array of pixels
pixels = cropped_img.reshape((-1, 3))

# Create a seaborn plot for RGB values
sns.set(style="whitegrid")
fig, ax = plt.subplots(3, 1, figsize=(10, 6))
fig.suptitle('Color Analysis - RGB Values')

# Plotting RGB values separately
for i in range(3):
    sns.histplot(pixels[:, i], kde=False, bins=256, ax=ax[i],
color=f'C{i}', stat="density")
    ax[i].set_xlim(0, 255)
    ax[i].set_xlabel(f'Intensity for Channel {i}')
    ax[i].set_ylabel('Density')

plt.show()

# View only the red values (set blue and green values to 0)
red_values_only = cropped_img.copy()
red_values_only[:, :, 1] = 0  # Set green channel to 0
red_values_only[:, :, 2] = 0  # Set blue channel to 0

# Display Image with Only Red Values
plt.imshow(red_values_only)
plt.title('Image with Only Red Values')
plt.show()
```


Original Image
0
100
200
300
400
500
0
100
200
300
400
500


Cropped Image
0
100
200
300
400
500
0
100
200
300
400
500

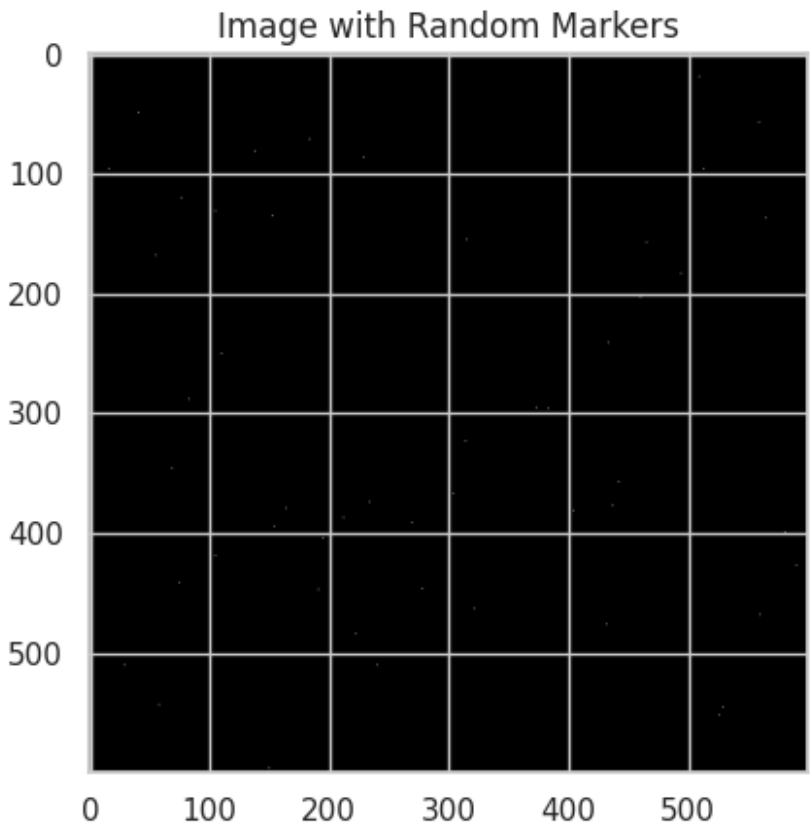
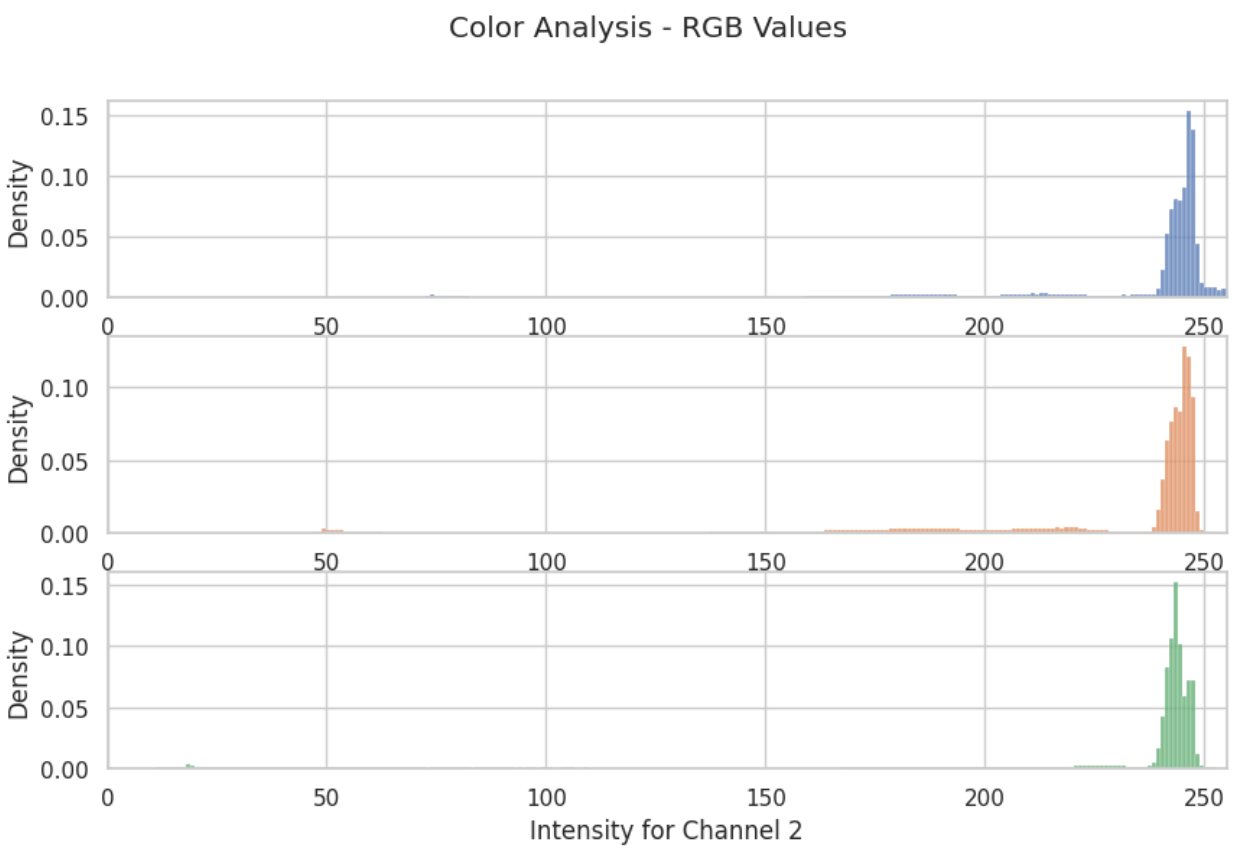
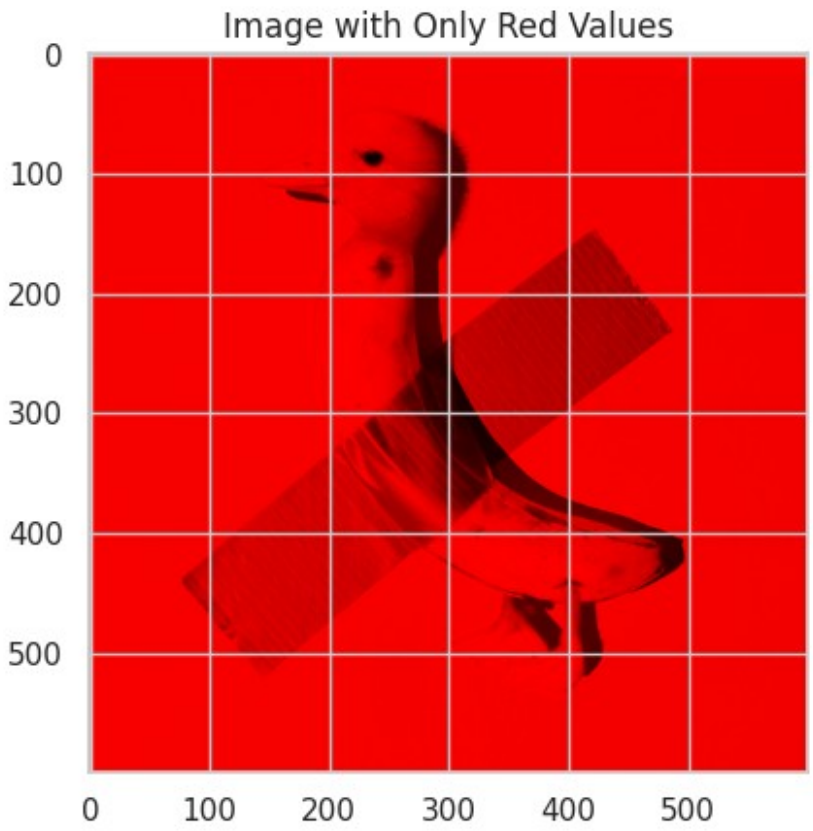


Image with Random Markers
0
100
200
300
400
500
0
100
200
300
400
500


Color Analysis - RGB Values
Density
0.15
0.10
0.05
0.00
0
50
100
150
200
250
Density
0.10
0.05
0.00
0
50
100
150
200
250
Density
0.15
0.10
0.05
0.00
0
50
100
150
200
250
Intensity for Channel 2



Task 7: Video Manipulation (15 marks)

Importing Libraries

Run the following commands to import libraries

import cv2

import numpy as np

Loading a Video

Create a VideoCapture object and read the

input from camera:

video = cv2.VideoCapture(0)

Tasks

=> Check if the camera is open.

=> Read the video from camera feed.

=> Write the video file into memory using the videoWriter function.

```
import cv2
import numpy as np

# Create a VideoCapture object and read the input from the camera
video = cv2.VideoCapture(0)

# Check if the camera is open
if not video.isOpened():
    print("Error: Could not open camera.")
else:
    print("Camera is open.")

# Read the video from the camera feed
while True:
    ret, frame = video.read()

    # Break the loop if reading the frame fails
    if not ret:
        print("Error: Failed to read frame.")
        break

    # Display the frame
    cv2.imshow('Camera Feed', frame)

    # Break the loop if 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the VideoCapture and close all windows
video.release()
cv2.destroyAllWindows()
```

```
Error: Could not open camera.
Error: Failed to read frame.
```

My Laptop does not have a camera connected to it that's why these two cells are showing error although I have written the correct code to read frames from the camera and save it in the memory.

```
# Create a VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))  #
Adjust parameters as needed

# Read the video from the camera feed and write to the output file
while True:
    ret, frame = video.read()
```

```
    if not ret:
        print("Error: Failed to read frame.")
        break

    # Write the frame to the output file
    out.write(frame)

    cv2.imshow('Camera Feed', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the VideoCapture, VideoWriter, and close all windows
video.release()
out.release()
cv2.destroyAllWindows()
```

```
Error: Failed to read frame.
```

---THE END---