Name: Drumil Kotecha, Course: BTI, Roll: C052;

Aim: Use K-means to segment image using Gen AI

Task 1: Download an image

Task 2: Apply K-means clustring algorithm using k=6

Task 3: Vary the value of k and observe the effects

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

image = cv2.imread('/content/sample_data/salah.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7eca7304da50>
```


```
def apply_kmeans(image, k):
    pixels = image.reshape(-1, 3)
```

```
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(pixels)
    segmented_img =
kmeans.cluster_centers_[kmeans.labels_].reshape(image.shape)
    return segmented_img

# Plot images for varying k values from 4 to 15
k_values = list(range(4, 16))  # k = 4 to k = 15
fig, axes = plt.subplots(len(k_values), 2, figsize=(12, 3 *
len(k_values)))  # Create 2 columns (original and K-means)

for i, k in enumerate(k_values):
    segmented_img = apply_kmeans(image, k)

    # Display the original image on the left (first column)
    axes[i, 0].imshow(image)
    axes[i, 0].set_title('Original Image')
    axes[i, 0].axis('off')

    # Display the K-means segmented image on the right (second column)
    axes[i, 1].imshow(segmented_img.astype(int))
    axes[i, 1].set_title(f'k = {k}')
    axes[i, 1].axis('off')

plt.tight_layout()
plt.show()
```

Original Image

k = 4

Original Image

k = 5

Original Image

k = 6

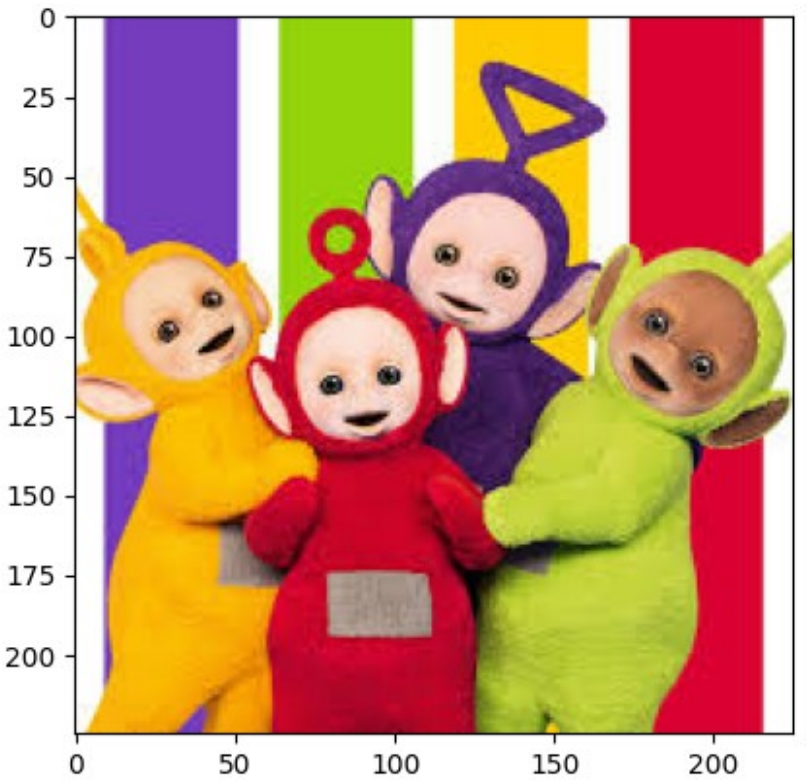
Original Image

k = 7

```
image1 = cv2.imread('/content/sample_data/vro.jpg')
image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)

plt.imshow(image1)
```

```
<matplotlib.image.AxesImage at 0x7eca66bc8e10>
```




```
k_values = list(range(4, 26))  # k = 4 to k = 25
fig, axes = plt.subplots(len(k_values), 2, figsize=(12, 3 *
len(k_values)))  # Create 2 columns (original and K-means)

for i, k in enumerate(k_values):
    segmented_img1 = apply_kmeans(image1, k)

    # Display the original image on the left (first column)
    axes[i, 0].imshow(image1)
    axes[i, 0].set_title('Original Image')
    axes[i, 0].axis('off')

    # Display the K-means segmented image on the right (second column)
    axes[i, 1].imshow(segmented_img1.astype(int))
    axes[i, 1].set_title(f'k = {k}')
    axes[i, 1].axis('off')
```

```
plt.tight_layout()
plt.show()
```

Original Image

k = 4

Original Image

k = 5

Original Image

k = 6

Original Image

k = 7

Conclusion

- K means algorithm is used to segment the image: `salah.jpg` for `K=1`, it shows image with one intensity. The cluster mean is the average of the color channels.
- If value of `K` is increased, the number of segments increases proportionally.
- If `K` reaches 15 the segmented image is almost similar to the original image. This is because the maximum number of possible clusters is number of possible colors for that image. This is usefull for compressing the given image.
- Similarly, K means algorithm is used to segment the image: `teletubby.png` for `K=1`, it shows image with one intensity. The cluster mean is the average of the color channels.
- If value of `K` is increased, the number of segments increases proportionally.
- If `K` reaches 25 the segmented image is almost similar to the original image. This is because the maximum number of possible clusters is number of possible colors for that image. This is usefull for compressing the given image.