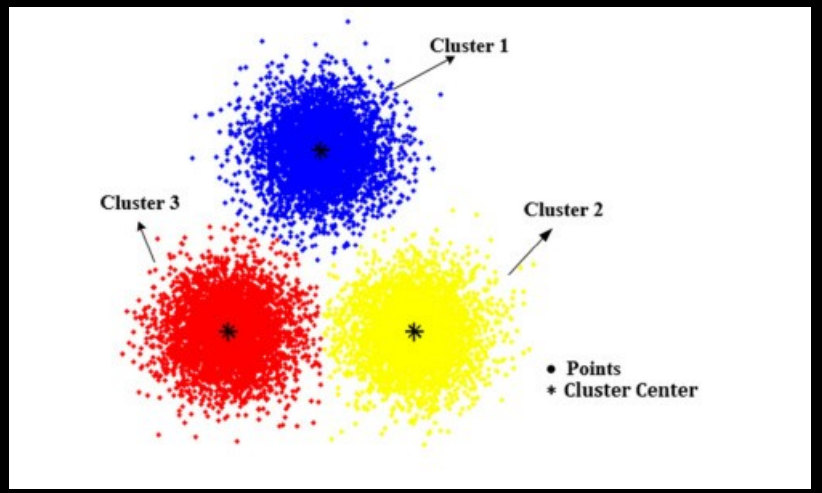
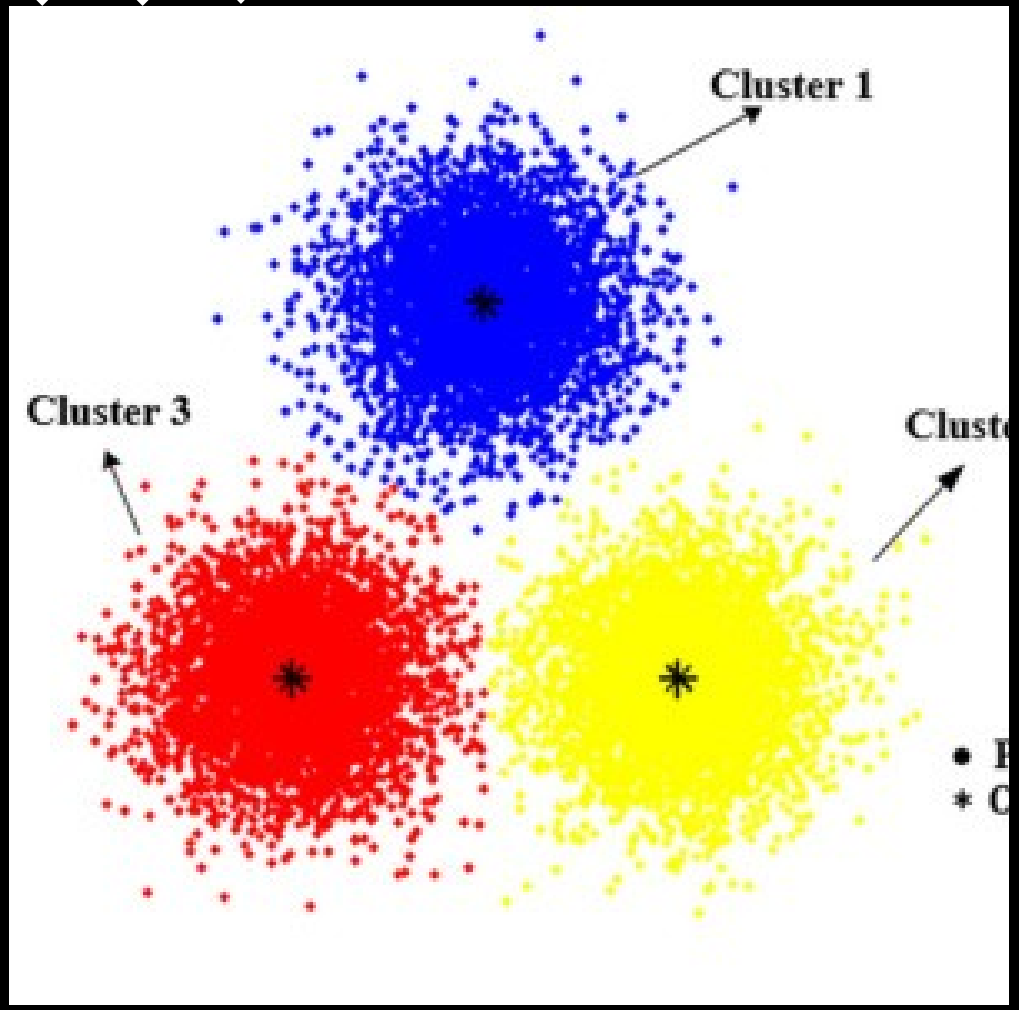
K_MEANS(IMAGE DETECTION)



- **1.Definition and Explanation of Image Segmentation: Image segmentation**
- is the task of partitioning an image into multiple segments. In semantic segmentation, all pixels that are part of the same object type get assigned to the same segment. For example, in a selfdriving car's vision system, all pixels that are part of a pedestrian's image might be assigned to the "pedestrian" segment (there would be one segment containing all the pedestrians).In some applications, this may be sufficient. For example, if you want to analyze satellite images to measure how much total forest area there is in a region, color segmentation may be just fine. Image segmentation is the process of partitioning a digital image into multiple distinct regions containing each pixel(sets of pixels, also known as superpixels) with similar attributes. The goal of Image segmentation is to change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries(lines, curves, etc.) in images. More precisely, Image Segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

K_MEANS(IMAGE DETECTION)

- **2.Where Can We Use Image Segmentation:**
- -In Autonomous Vehicles: they need sensory input devices like cameras, radar, and lasers to allow the car to perceive the world around it, creating a digital map. Autonomous driving is not even possible without object detection which itself involves image classification/segmentation. -Healthcare Industry:if we talk about Cancer, even in todayвЂ™s age of technological advancements, cancer can be fatal if we donвЂ™t identify it at an early stage. Detecting cancerous cell(s) as quickly as possible can potentially save millions of lives. The shape of the cancerous cells plays a vital role in determining the severity of cancer which can be identified using image classification algorithms.



K_MEANS(IMAGE DETECTION)

- **3. K Means Algorithm for Image Segmentation:**
- K Means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background. It clusters, or partitions the given data into K-clusters or parts based on the Kcentroids.

Image Segmentation.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 1:01 PM

+ Code + Text

```
[ ] import array
    import numpy as np
    import cv2
    import matplotlib.pyplot as plt
    #%matplotlib inline
    image = cv2.imread('Image.jpg')

    # Change color to RGB (from BGR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB, 1)

    # preview image
    plt.imshow(image)
    plt.axis('off');
    image.shape
```

```
(600, 1024, 3)
```

```
[ ] pixel_values = np.reshape(image, (-1, 3))   # 400 * 794

    # Convert to float type
    pixel_values = np.float32(pixel_values)

    pixel_values.shape
```

```
(614400, 3)
```

EXAMPLE OF IMAGE DETECTION BY K-MEANS CLUSTERING

- HERE AS YOU SEE WE IMPORTED THE LIBRARIES AND CHOOSE ONE IMAGE FOR DETECTION

Image Segmentation.ipynb

File Edit View Insert Runtime Tools Help Last saved at 1:01 PM

+ Code + Text

```
[ ] k = 2
    # define stopping criteria
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

    # perform k-means clustering
    # kmeans(data, K, bestLabels, criteria, attempts, flags[, centers])
    retval, labels, centers = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

    # check center points
    centers
```

```
array([[ 58.56179 , 101.68155 ,  10.84005 ],
       [ 61.57463 , 105.895485, 185.57219 ]], dtype=float32)
```

```
[ ] # convert data into 8-bit values
    centers = np.uint8(centers)
    centers
```

```
array([[ 58, 101,  10],
       [ 61, 105, 185]], dtype=uint8)
```

```
[ ] labels.flatten()
```

```
array([1, 1, 1, ..., 0, 0, 0], dtype=int32)
```

```
[ ] # assign centers to labels
    segmented_data = centers[labels.flatten()]
    segmented_data
```

```
array([[ 61, 105, 185],
       [ 61, 105, 185],
       [ 61, 105, 185],
       ...,
       [ 58, 101,  10],
       [ 58, 101,  10],
       [ 58, 101,  10]], dtype=uint8)
```

- HERE WE DID DATA PREPROCESSING, DEFINED STOPPING CRITERIA, PERFORMED K-MEANS CLUSTERING AND CHECKED CENTER POINTS

```
[ ]  plt.imshow(masked_image)
     plt.axis('off');
```

```
[ ]  # mask an image segment by cluster
     masked_image = np.copy(image)

     # turn the mask green!
     cluster = 1                    # the second cluster
     masked_image[labels_reshape == cluster] = [0, 225, 0]

     plt.imshow(masked_image)
     plt.axis('off');
```

- HERE WE MASKED AN IMAGE SEGMENTS BY CLUSTER

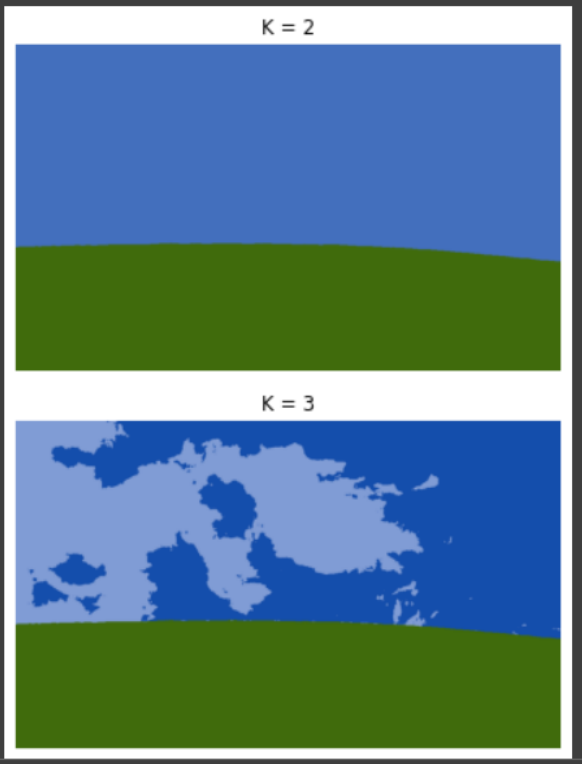
Image Segmentation.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 1:01 PM

+ Code + Text

```
for k in range(2, 11):
    # Implementing kMeans
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
    retval, labels, centers = cv2.kmeans(pixel_values, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
    centers = np.uint8(centers)
    segmented_data = centers[labels.flatten()]
    segmented_image = segmented_data.reshape((image.shape))
    labels_reshape = labels.reshape(image.shape[0], image.shape[1])

    # plot Kmeans results
    plt.imshow(segmented_image)
    plt.title(f"K = {k}")
    plt.axis('off')
    plt.show()
```



RESULT

HERE WE IMPLEMENTED K-MEANS WITH DIFFERENT VALUES AND PLOTTED K-MEANS RESULTS