[25]
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin
from sklearn.datasets import load_sample_image
from sklearn.utils import shuffle
from time import time
```

[32]
```
# Load the Summer Palace photo
image = load_sample_image(r"C:\Users\maniv\Downloads\Anisha
Agrawal Cover\CT scan.jpg")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call
last)
<ipython-input-32-9f2a09181ca6> in <module>
      1 # Load the Summer Palace photo
----> 2 image = load_sample_image(r"C:\Users\maniv\Downloads\Anisha
Agrawal Cover\CT scan.jpg")

~\Anaconda3\lib\site-packages\sklearn\datasets\_base.py in
load_sample_image(image_name)
    842             break
    843     if index is None:
--> 844         raise AttributeError("Cannot find sample image: %s" %
image_name)
    845     return images.images[index]
    846

AttributeError: Cannot find sample image:
C:\Users\maniv\Downloads\Anisha Agrawal Cover\CT scan.jpg
```

[34]
```
image = cv2.imread(r"C:\Users\maniv\Downloads\Anisha Agrawal
Cover\CT scan.JPG")
```

[35]
```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

[36]
```
# reshape the image to a 2D array of pixels and 3 color values
(RGB)
pixel_values = image.reshape((-1, 3))
```

[37]
```
# convert to float
pixel_values = np.float32(pixel_values)
```

[38]
```
print(pixel_values.shape)
```

```
(1347944, 3)
```

[39]
```
# define stopping criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
100, 0.2)
```

[40]
```
# number of clusters (K)
k = 3
_, labels, (centers) = cv2.kmeans(pixel_values, k, None,
criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
```

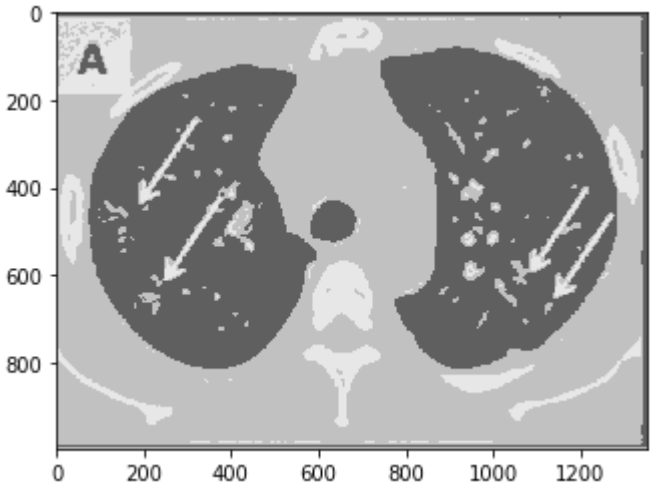
[41]
```
# convert back to 8 bit values
centers = np.uint8(centers)
```

[42]
```
# flatten the labels array
labels = labels.flatten()
```

[43]
```
# convert all pixels to the color of the centroids
segmented_image = centers[labels.flatten()]
```

[44]
```
# reshape back to the original image dimension
segmented_image = segmented_image.reshape(image.shape)
```

[45]
```
# show the image
plt.imshow(segmented_image)
plt.show()
```



[46]
```
# disable only the cluster number 2 (turn the pixel into black)
masked_image = np.copy(image)
```

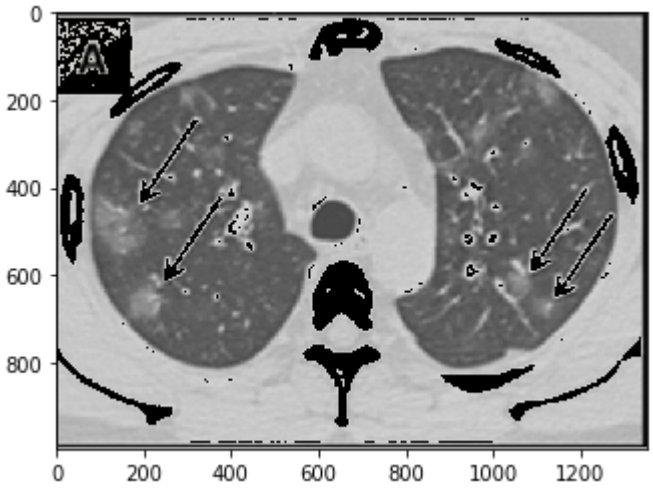
[47]
```
# convert to the shape of a vector of pixel values
masked_image = masked_image.reshape((-1, 3))
```

[48]
```
# color (i.e cluster) to disable
cluster = 2
```

[49]
```
masked_image[labels == cluster] = [0, 0, 0]
```

[50]
```
# convert back to original shape
masked_image = masked_image.reshape(image.shape)
```

[66]
```
# show the image
plt.imshow(masked_image)
plt.show()
```



[69]
```
n_colors = 64
```

[70]
```
print("Fitting model on a small sub-sample of the data")
t0 = time()
image_array_sample = shuffle(pixel_values, random_state=0)[:1000]
kmeans = KMeans(n_clusters=n_colors,
random_state=0).fit(image_array_sample)
print("done in %0.3fs." % (time() - t0))
```

```
Fitting model on a small sub-sample of the data
done in 0.672s.
```

[72]
```
# Get labels for all points
print("Predicting color indices on the full image (k-means)")
t0 = time()
labels = kmeans.predict(pixel_values)
print("done in %0.3fs." % (time() - t0))
```

```
Predicting color indices on the full image (k-means)
done in 1.952s.
```

[ ]