Programming for Image Compression

1. Within the K-medoids framework, I started by implementing PAM method which seems to work fine for small n (data points) but was taking long running times for images (since it has many points). So I made a simplification and found J points closest to mean point and chose the point with lowest loss function (total distance) as representative for cluster.

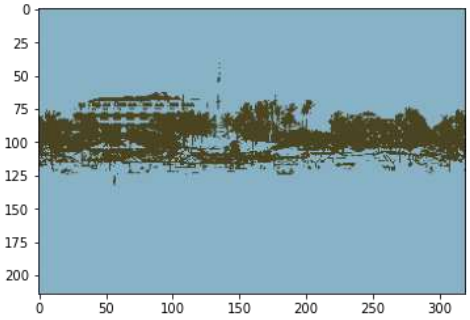
Algorithm :

1. Initialize K medoids randomly from the n points.
2. Assign the rest of the points to these K medoids using distance measure and form K clusters. I tried with 2 different distance measures (Euclidean and Manhattan) and on average Euclidean gave lesser loss function value on convergence and better outputs. So I used Euclidean for the rest of my code. (I have given an option in my code to change the distance measure by simply changing the l value)
3. Choose J points nearest to the K^{th} cluster center and calculate the loss value for the J points. (I have chosen 1000 unique points)
4. Assign the point with least loss value as the new medoid of the cluster
5. Repeat Step 2-5 until convergence
6. Convergence defined as : δloss between two iteration < 10^-6 or Iteration reaches MaxIter

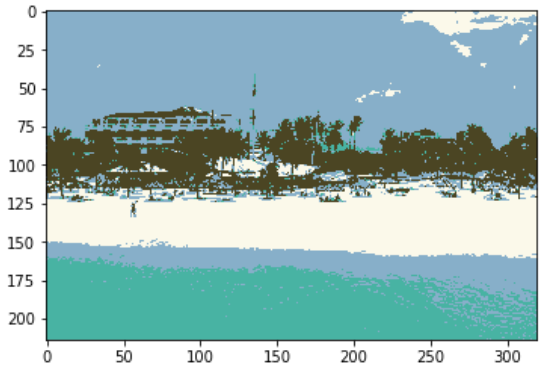
K_Medoids

Original Image

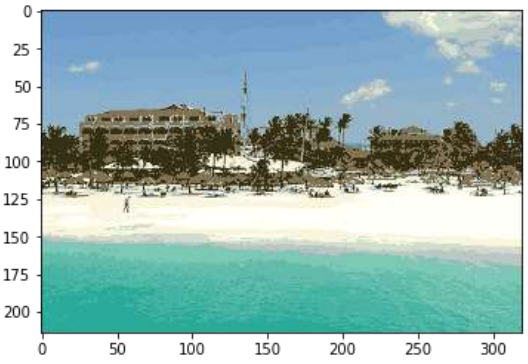
K-medoids with K=2



K-medoids with K=4



K-medoids with K=16



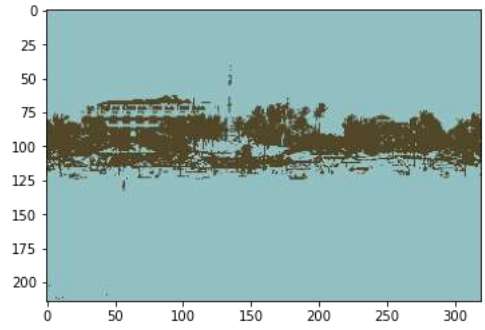
- As K value increases, the output image quality is increasing (K, the number of clusters is also the number of colors in our output image). With high values (K=32 or 64) the output is almost resembling the original image.
- It takes more iterations and longer time for convergence with increasing K values.
- I have randomized the initialization of initial cluster centers in my code. Different initialization did not seem to be affecting the final image very much but it was taking more time to converge

K values (K-Medoids)	No. of Iterations	Run time (sec)
K=2	4	10.156
K=4	9	23.24
K=16	16	34.24
K=50	23	41.01

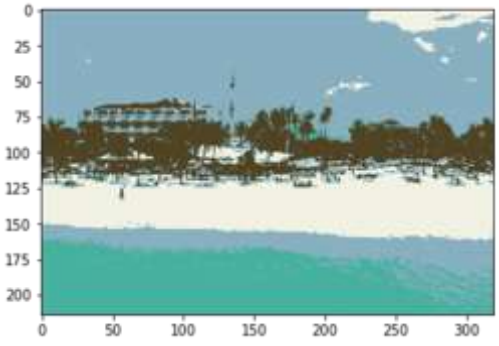
K-means

Original Image

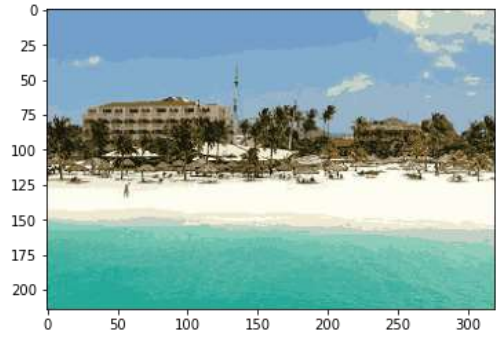
K-means with K=2 (run time=16.93 sec)



K-means with K=4 (run time=41.35 sec)



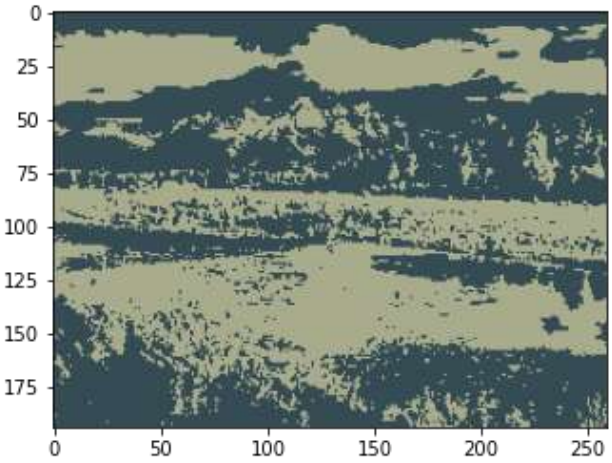
K-means with K=16 (run time=89 sec)



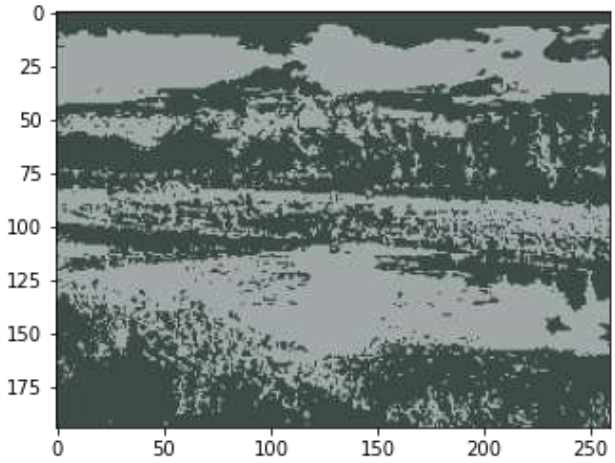
- Run times and No.of iterations for convergence increase significantly with increasing K
- As K values increase, the quality of image increase with very high K values producing images very similar to the original image
- Intialisation are not affecting the final output of the code significantly.

Original Image:

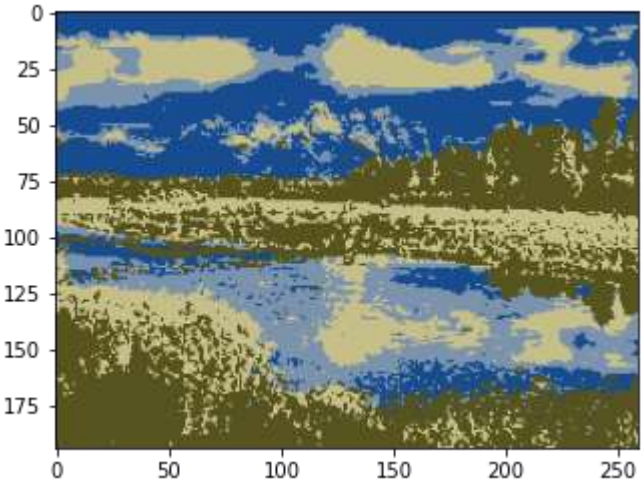
K-means K=2



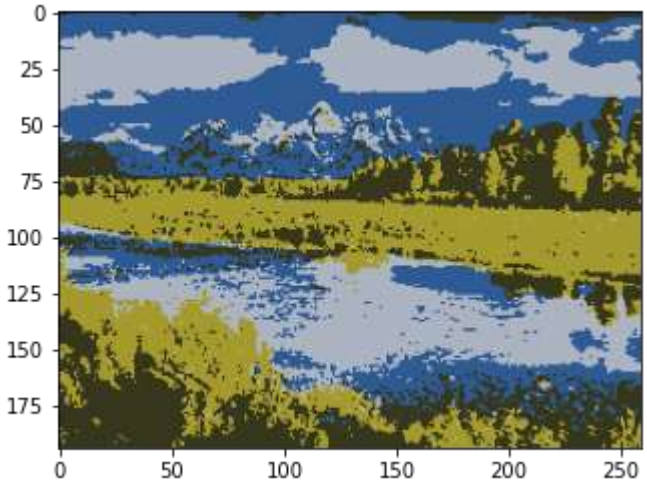
K-medoids K=2



K-means K=4



K-medoids K=4



K-medoids is producing higher quality images in shorter (similar) run times. One reason is that , the K-means algorithm is heavily affected by outliers where K-medoids is more robust to treating outliers.