Assignment 2
Pattern Recognition
Report
Image Segmentation

Esraa Wael El-Hawash	ID: 3959
Guehad Mohamed Ahmed	ID: 3861

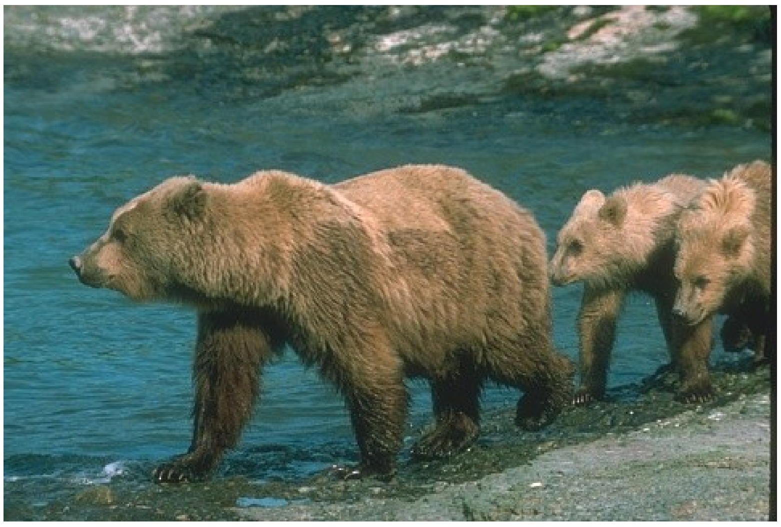
Assignment 2 Image Segmentation

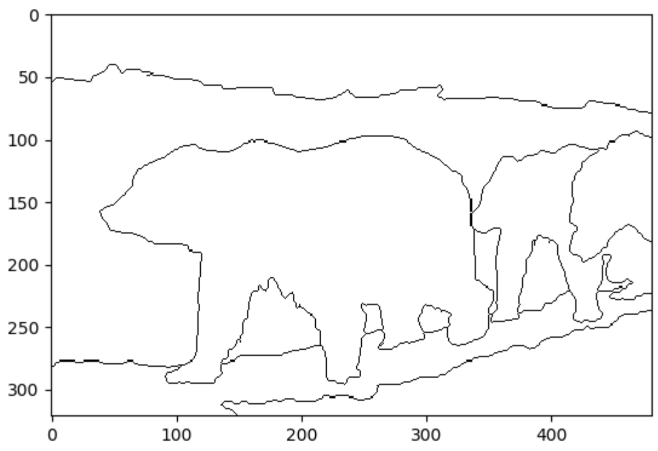
We were asked to perform image segmentation. Image segmentation means that we can group similar pixels together and give these grouped pixels the same label. The grouping problem is a clustering problem.
We will be using k-means and normalized cut as clustering algorithms and we'll be mentioning the steps and results in details in this report.

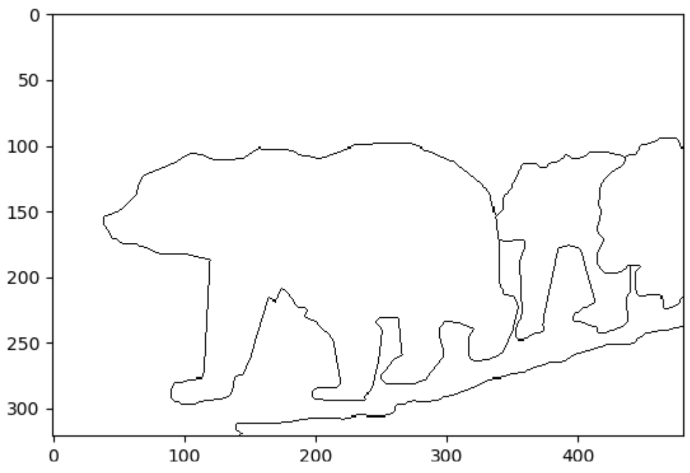
Step 1:
Download & Visualization:
Firstly we downloaded the Berkeley Segmentation Benchmark which contains a dataset that has 500 images. The test set is 200 images only. We will report our results on the first 50 images of the test set only.
We will train our model on 200 images.
We load the images using the image.open() function which finds the image name in both the train images folder and it's exact equivalent in the ground truth file in the .mat format.
This function simply maps every image to it's ground truth.
Now in order to display or visualize the two together
we show the normal .jpg image using image.show() function.
However for the ground truth image it took a little work
basically what we had to do is open the mat file using scipy library's .loadmat function and then we loop over the shape of the ground truth which essentially is the number of columns in the mat file or to make more sense , the number of segments for this specific picture's ground truth.
Inside the loop we access each segment from the array of ground truths and we extract the bounds of it which is also included in the same mat file , we then use the function toimage() that checks the bounds and colors them in black while leaving everything else into the white color to give a segmentation of the given image.

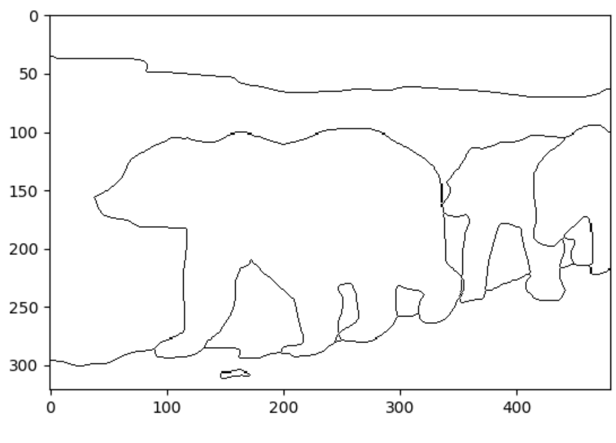
Here are some plots to show the image and it’s ground truth segments.

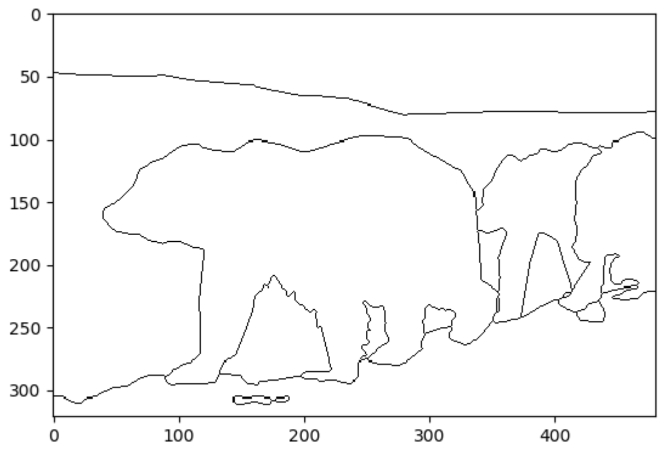
Original image:

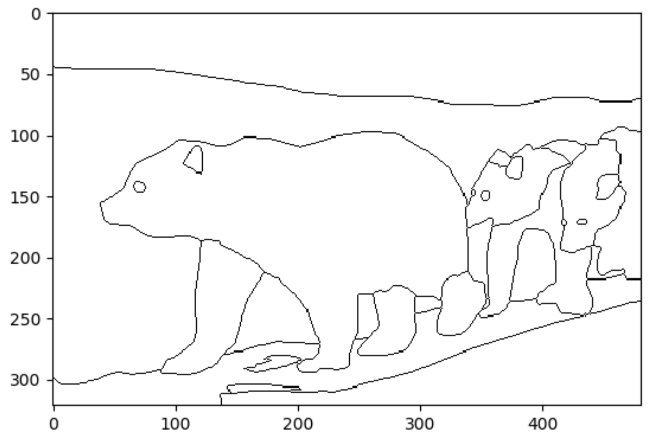


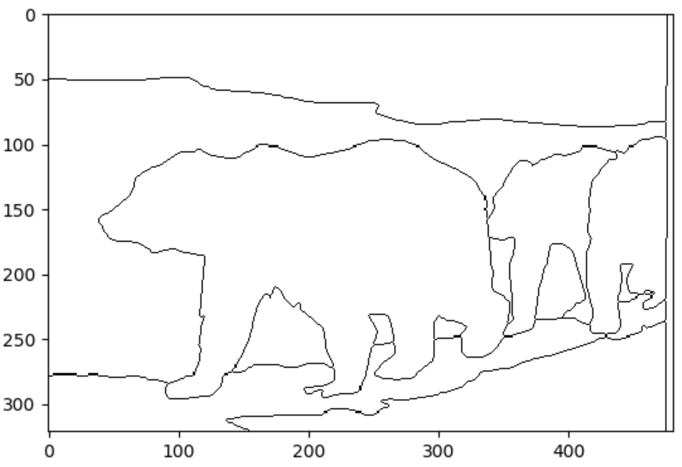












This was an example to an image and it’s associated ground truth segmentations.
 That concludes the first step of the assignment.

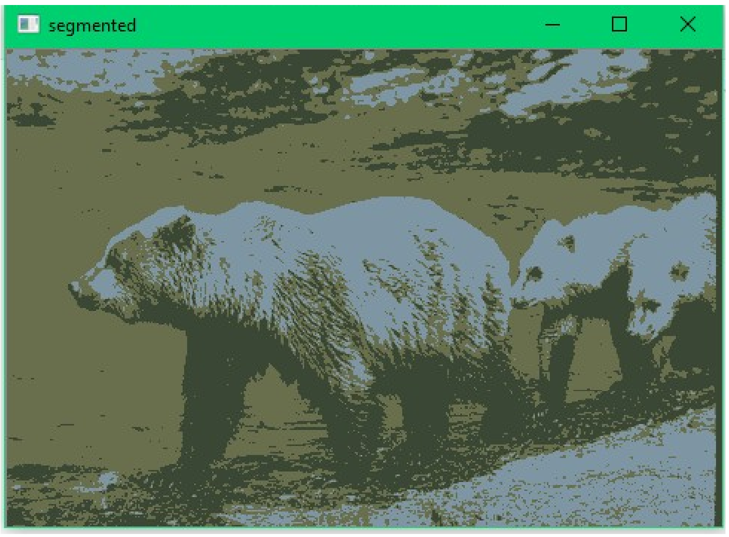
__Step 2:__
__K-means Segmentation:__

we used the built-in k-means function alongside some adjustments for this segmentation.
Basically the k-means function we made takes as parameters the image and the list of ks=[3,5,7,9,11]
the image is turned to an array using the numpy library,
then it’s reshaped to a vector.
The criteria is chosen according to the cv2 built-in functions and so is the max iterations.
The labels and the centroids are returned after calling the builtin k-means function with the image vector , the l and the criteria set previously as parameters.
The labels returned from the function are reshaped to the original image size and both labels and the segmented / reshaped image are returned.

This is the results from the segmentation using k-means from every k.

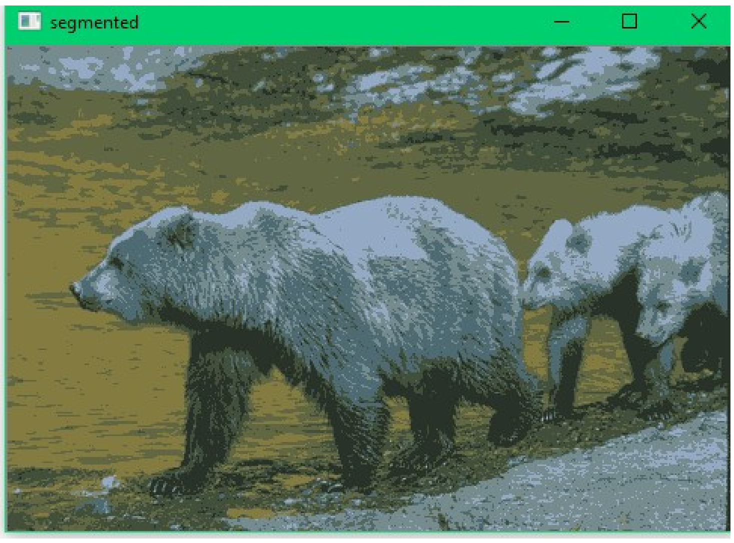
K=3

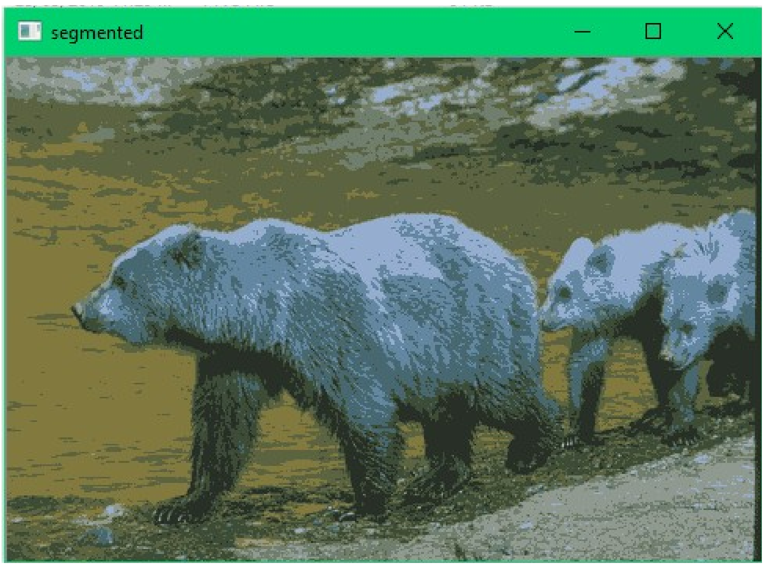
K=5



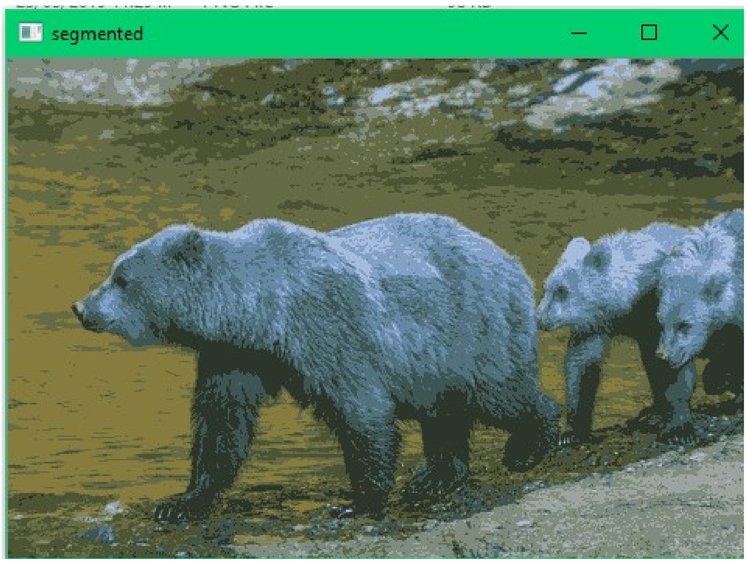
K=7

K=9





K=11

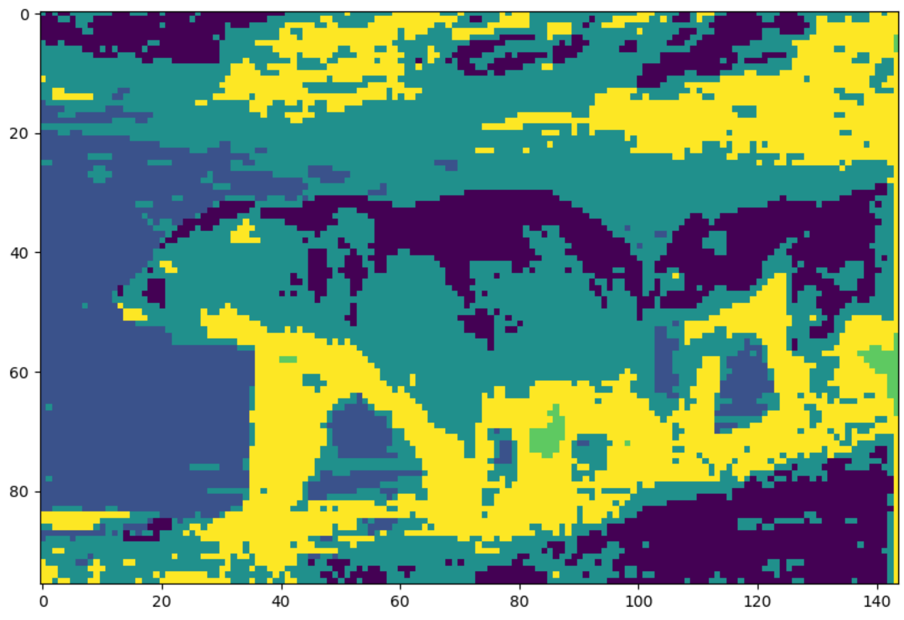


The **Evaluation** using F-measure and Conditional Entropy for the previous output (segmentations)
in order to calculate the **F-measure** we made a loop that iterates over all segments of the ground truth (to compare the output of the k-means with them) and for each true segment we extract the bound of it and that is considered as our y_true (true labels)
and the labels that are returned from the k-mean function are turned into a 1-d array using .ravel() and are considered our y_pred (predicted labels) , now those two parameters are inserted in the f_score function which is the built-in function that calculates the weighted f-measure for each segment.
And all the scores for all segments are summed and divided by the number of segments to get the average score for a particular picture for a particular K of all it's segments.
For example the picture displayed above in the k-means had the following average F-measures

K=3	F-Measure= 0.65
K=5	F-Measure= 0.57
K=7	F-Measure=0.50
K=9	F-Measure=0.44
k=11	F-Measure=0.23

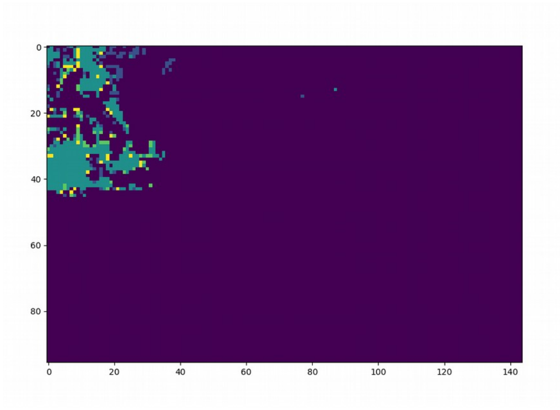
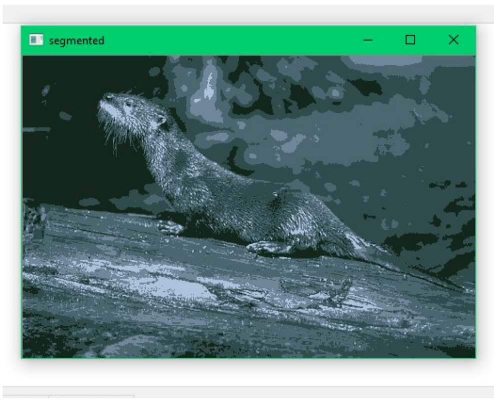
Conditional Entropy is calculated from scratch by getting the segment and turning it into an array and for every cluster in the segment we check for the segment that equals the cluster and we get that partition and then we calculate the entropy using the log2 formula.
And then it's called the same way the F-measure is called , using the same loops to compare with all segments of a picture and getting an average of all the entropies
The pre-shown photo showed results somewhere between 0.07 to 0.14 to 0.176 to 0.2
and gave an average of 0.1248

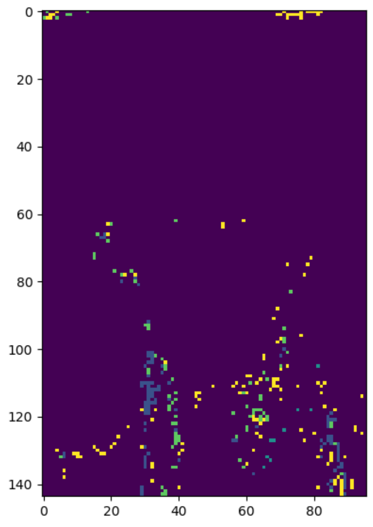
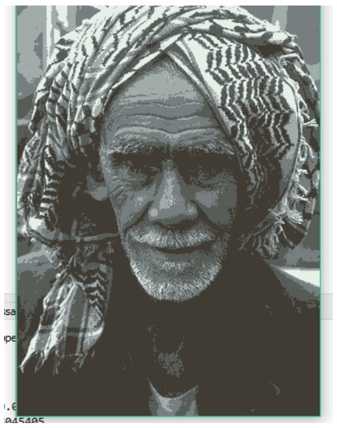
<u>Step 3:</u>
<u>Big Picture(N-cut):</u>
in this step we need to implement an N-cut segmentation and compare it with the k-means segmentation both for k=5 for 5 random pictures of our training data.
For the N-cut implementation we start by resizing the input image because otherwise it wouldn’t work and we do so using the function imresize() and we save the new dimensions.
We vectorize the image after resizing it and then we used the bui;t-in Spectral Clustering function choosing nearest neighbour for affinity and k=5 as requested from us.
We fit the image vector and an array of labels is returned to us.
here’s the result of segmentation using 5-NN spectral clustering.

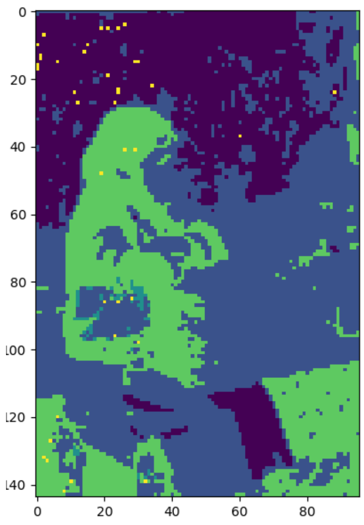


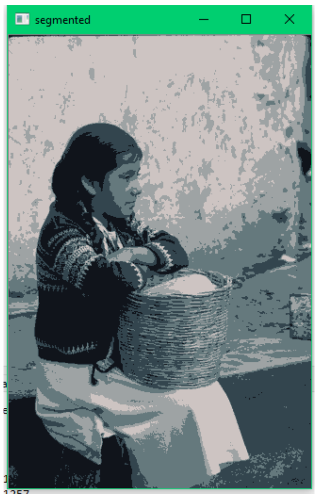
In order to compare this to it’s ground truth we have to resize the segments of the ground truths all to the same dimensions we resized the original photo at.
Other then that the same functions and steps mentioned for evaluating the k-means are used again to evaluate in this step.

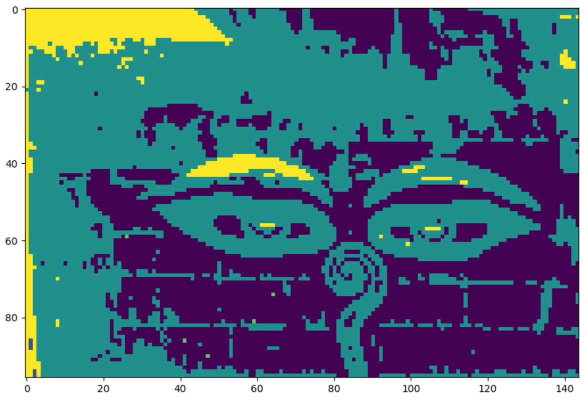
We implemented a function called big_picture()
this function goes to the training data and chooses 5 pictures at random and then it gets their ground truths and in turn it calls the k-mean function and chooses the k =5 and calls the n-cut function. And the plots for both algorithms are shown and compared for those 5 pictures by calculating the average f-measure and the conditional entropies.

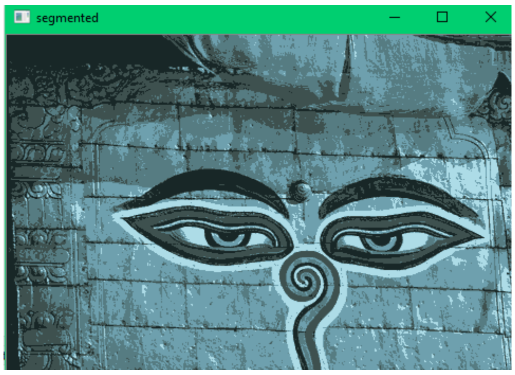



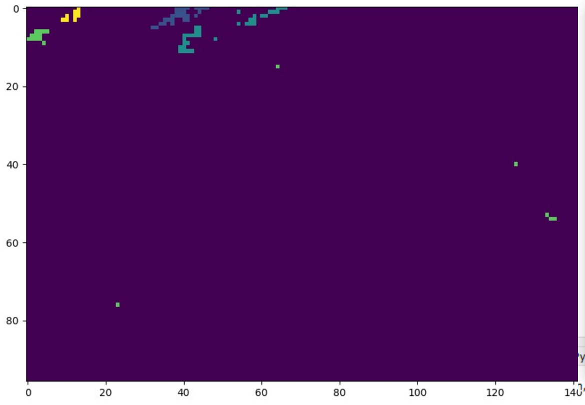



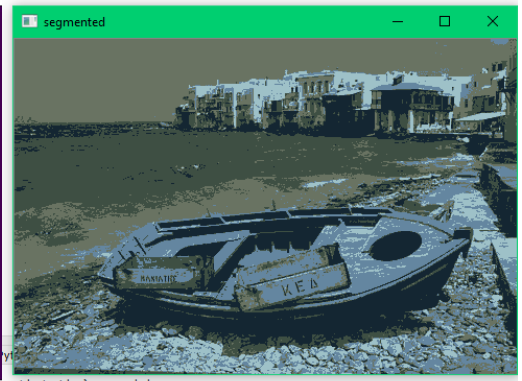






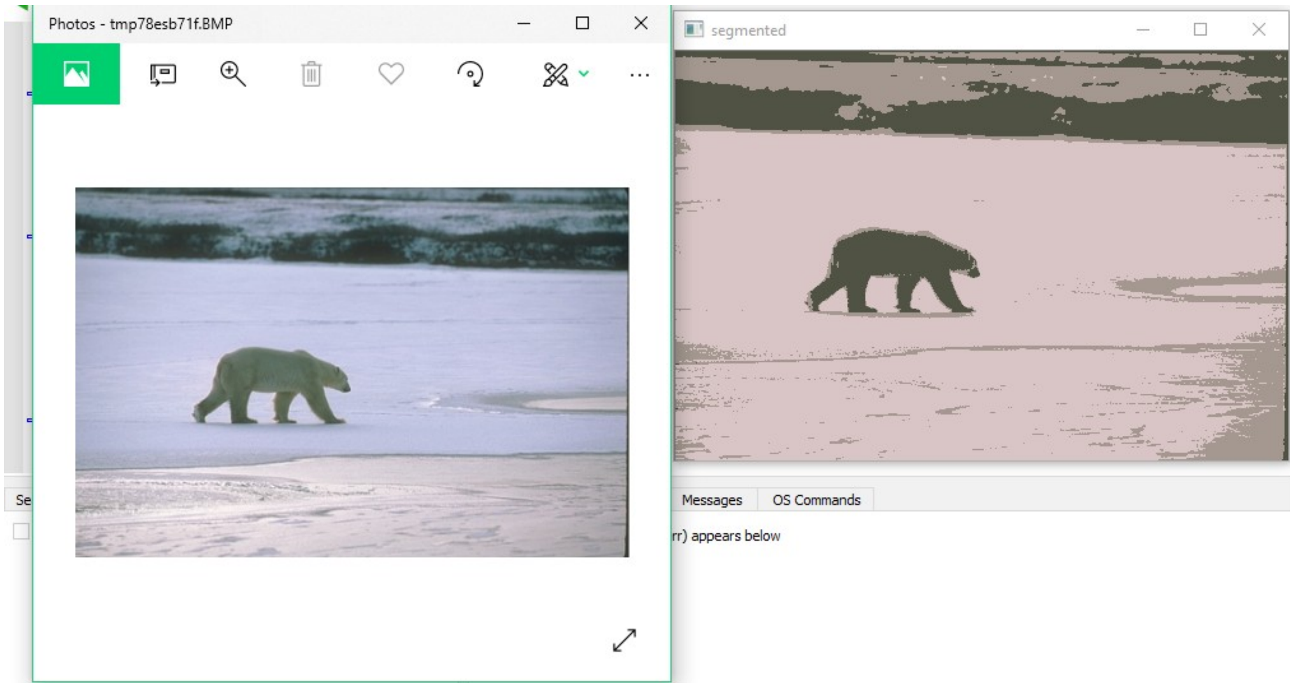




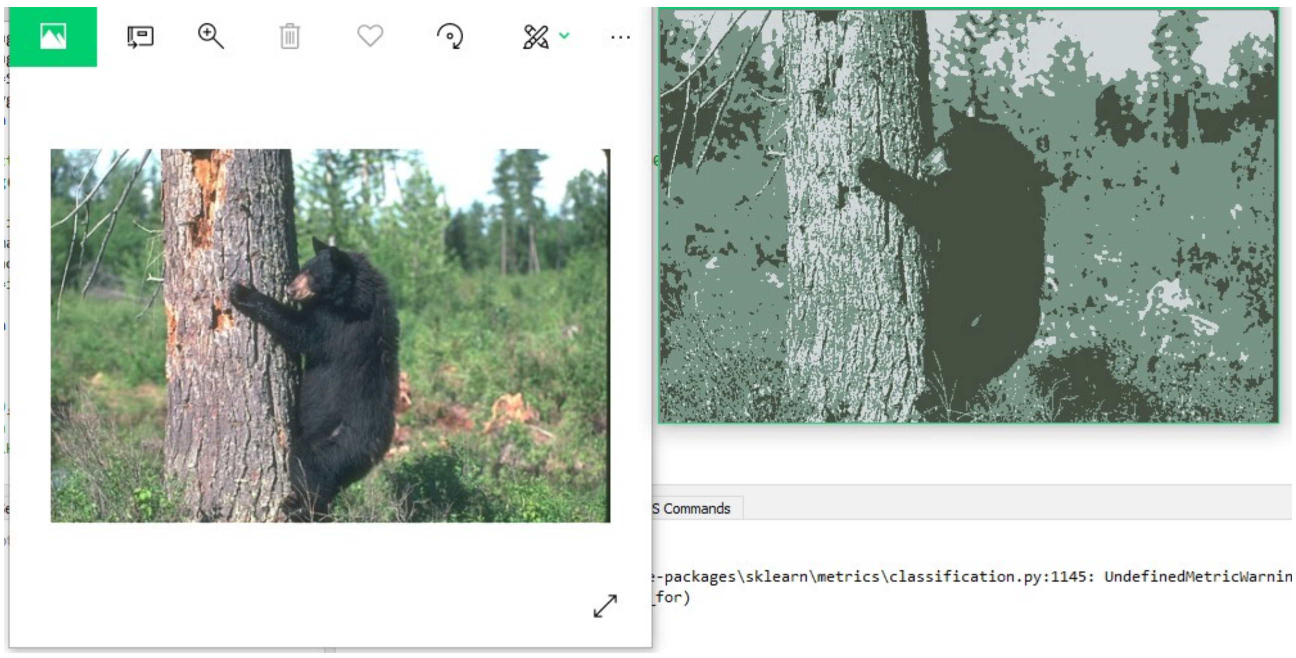


, stdout, stderr) appears below

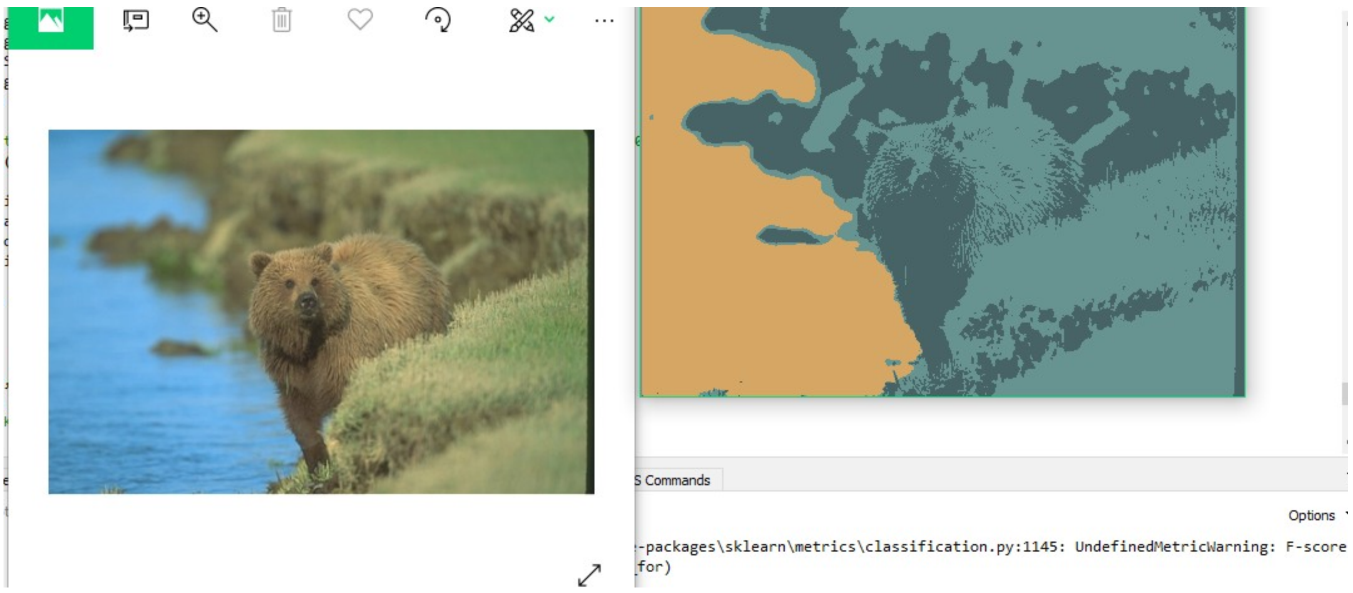
Step 4:
Testing:
The final step is testing our model on the test set
based on our training set we chose the best results and mapped it to the K that cause it.
We did so by trying all Ks on all 200 images and getting the average scores of all segments , sorting them in Descending order and picking the top 3 scores for each K , the highest of averages of the top scores gives us the highest results and the equivalent K.
we use that K on the first 50 pictures in the test set and we get the following segments:
here is a sample on the first 4..



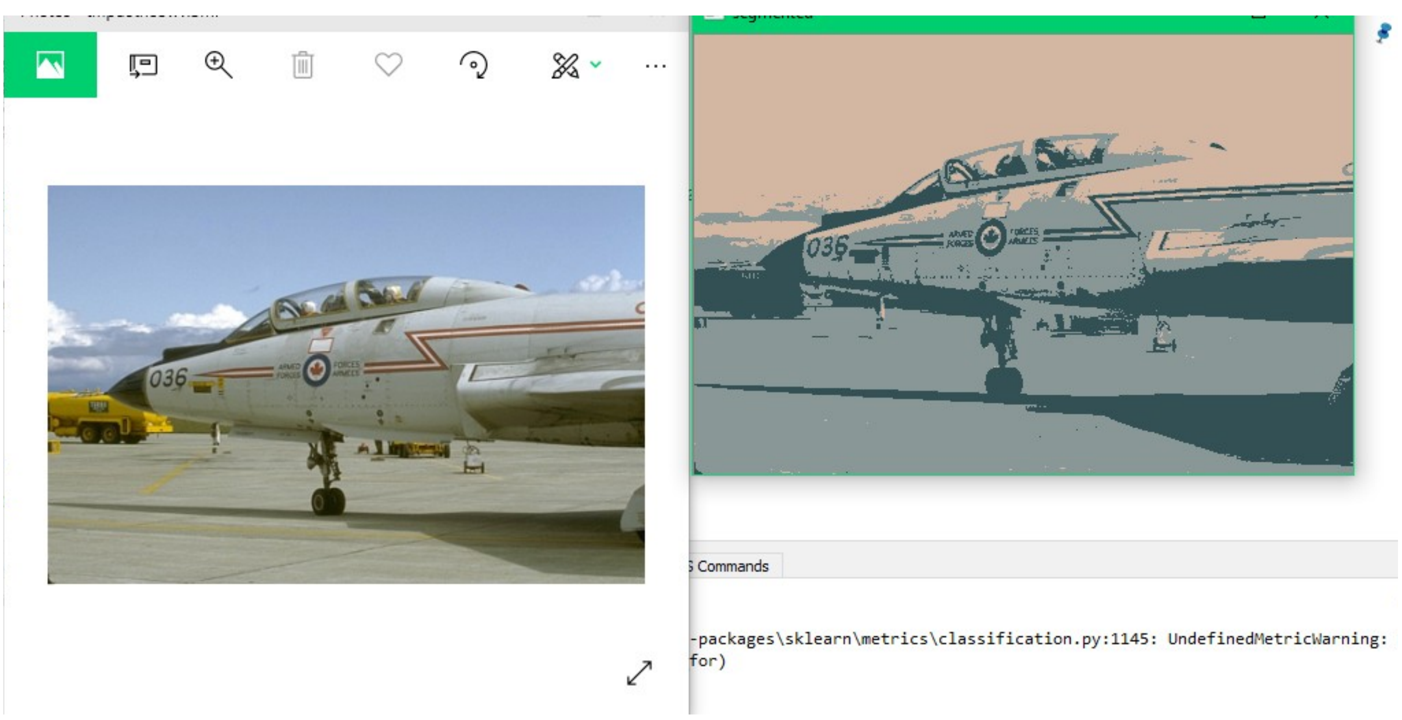
```
rr) appears below
```



```
-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarnin
for)
```



```
-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: F-score
for)
```



```
-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning:
for)
```