RESULTS=

with 8 bins and 1-nearest neighbors:

Test image 1 of class "coast" has been assigned to class "coast".

Test image 2 of class "coast" has been assigned to class "coast".

Test image 3 of class "coast" has been assigned to class "coast".

Test image 4 of class "coast" has been assigned to class "coast".

Test image 1 of class "forest" has been assigned to class "forest".

Test image 2 of class "forest" has been assigned to class "forest".

Test image 3 of class "forest" has been assigned to class "forest".

Test image 4 of class "forest" has been assigned to class "forest".

Test image 1 of class "insidecity" has been assigned to class "forest".

Test image 2 of class "insidecity" has been assigned to class "forest".

Test image 3 of class "insidecity" has been assigned to class "insidecity".

Test image 4 of class "insidecity" has been assigned to class "insidecity".

accuracy = 0.833333

with 4 bins and 1-nearest neighbors:

Test image 1 of class "coast" has been assigned to class "coast".

Test image 2 of class "coast" has been assigned to class "coast".

Test image 3 of class "coast" has been assigned to class "coast".

Test image 4 of class "coast" has been assigned to class "coast".

Test image 1 of class "forest" has been assigned to class "forest".

Test image 2 of class "forest" has been assigned to class "forest".

Test image 3 of class "forest" has been assigned to class "forest".

Test image 4 of class "forest" has been assigned to class "forest".

Test image 1 of class "insidecity" has been assigned to class "forest".

Test image 2 of class "insidecity" has been assigned to class "insidecity".

Test image 3 of class "insidecity" has been assigned to class "insidecity".

Test image 4 of class "insidecity" has been assigned to class "coast".

accuracy = 0.833333

with 16 bins and 1-nearest neighbors:

Test image 1 of class "coast" has been assigned to class "coast".

Test image 2 of class "coast" has been assigned to class "coast".

Test image 3 of class "coast" has been assigned to class "coast".

Test image 4 of class "coast" has been assigned to class "coast".

Test image 1 of class "forest" has been assigned to class "forest".

Test image 2 of class "forest" has been assigned to class "forest".

Test image 3 of class "forest" has been assigned to class "forest".

Test image 4 of class "forest" has been assigned to class "forest".

Test image 1 of class "insidecity" has been assigned to class "forest".

Test image 2 of class "insidecity" has been assigned to class "forest".

Test image 3 of class "insidecity" has been assigned to class "forest".

Test image 4 of class "insidecity" has been assigned to class "insidecity".

accuracy = 0.750000

with 32 bins and 1-nearest neighbors:

Test image 1 of class "coast" has been assigned to class "coast".

Test image 2 of class "coast" has been assigned to class "coast".

Test image 3 of class "coast" has been assigned to class "coast".

Test image 4 of class "coast" has been assigned to class "coast".

Test image 1 of class "forest" has been assigned to class "forest".

Test image 2 of class "forest" has been assigned to class "forest".

Test image 3 of class "forest" has been assigned to class "forest".

Test image 4 of class "forest" has been assigned to class "forest".

Test image 1 of class "insidecity" has been assigned to class "forest".

Test image 2 of class "insidecity" has been assigned to class "forest".

Test image 3 of class "insidecity" has been assigned to class "forest".

Test image 4 of class "insidecity" has been assigned to class "insidecity".

accuracy = 0.750000

with 8 bins and 3-nearest neighbors:

Test image 1 of class "coast" has been assigned to class "coast".

Test image 2 of class "coast" has been assigned to class "coast".

Test image 3 of class "coast" has been assigned to class "forest".

Test image 4 of class "coast" has been assigned to class "coast".

Test image 1 of class "forest" has been assigned to class "coast".

Test image 2 of class "forest" has been assigned to class "forest".

Test image 3 of class "forest" has been assigned to class "forest".

Test image 4 of class "forest" has been assigned to class "forest".

Test image 1 of class "insidecity" has been assigned to class "insidecity".

Test image 2 of class "insidecity" has been assigned to class "forest".

Test image 3 of class "insidecity" has been assigned to class "insidecity".

Test image 4 of class "insidecity" has been assigned to class "insidecity".

accuracy = 0.750000

In conclusion the greater the number of bins and the greater the number of nearest neighbors considered generally lowers the accuracy of the model. This makes sense because with the knn algo the magnitude of the distance itself is not considered and this leads to higher error with datapoints closer to the margin separating class clusters. As for the number of bins as the concentration across the image becomes smaller the difference in R,G,B values will become less pronounced regarding the whole image so it will be more difficult to correctly classify the landscape.

I completed this assignment by getting the rgb values of the pixels of the training images for each pixel I incremented 1 of 8 ints in the 3 histograms for the red, green, and blue color values based on the intensity of that color in that pixel. I concatenated the three lists to get the final histogram for the image. Then I repeated the process for the testing images. Using labels for the training collected earlier my training model and my testing model I got the classification of each of my testing images using the Euclidean distance and the k-nearest neighbors' formula with k=1. After I finished this changing my script to accommodate different values for bins and k was trivial. For more information see the code below.

```
#import libraries
from PIL import Image
import numpy as np
import math


#in this function I create the histograms for the images in the test and training set
def makeImageClassifier(FSs,nbin):
   #the counter is used to keep track of the index of my model
   counter = 0
   #imageCLassifier is my model. An array of size n by 3*i.
   imageClassifier = np.zeros((len(FSs),3*nbin),dtype=np.int64)
   #I iterate through each string/filename in argv[0]
   for fs in FSs:
      #these values store the R,G,B values for the histogram to use later
      rhist,bhist,ghist = [0 for i in range(nbin)],[0 for i in range(nbin)],[0 for i in range(nbin)]
      #here I get the data from the image, put it in array format, get the dimensions and flatten it with
reshape
      im = Image.open(fs)
      rgb_im = np.asarray(im)
      sz = rgb_im.shape
      rgb_im = rgb_im.reshape(1,sz[0]*sz[1],3)
      #I go through each pixel and add it's rgb values to the i-th bins of the 3 seperate histogram lists
      for pix in rgb_im[0]:
         rhist[int(pix[0]/(256/nbin))]+=1
         ghist[int(pix[1]/(256/nbin))]+=1
         bhist[int(pix[2]/(256/nbin))]+=1
      #when I am done I merge the lists and verify that each pixel has been added
      hist = rhist+bhist+ghist
```

```
        if(sum(hist)/3 == sz[0]*sz[1]):
            #print('correct histogram for ' + fs + '.')
            counter+=0
        else:
            print('incorrect histogram for ' + fs + '!')
        #finally I add the hist to the model close the image increment the counter and iterate again
        imageClassifier[counter,:] = np.array(hist)
        im.close()
        counter += 1
    #I return my image classifier
    return imageClassifier

#this implements the K-nearest neighbors algorithm
def knn(testX,trainX,trainY,k):
    #euclidean distances are stored here
    edists = []
    #ed is a temporary variable storing the euclidean distance
    ed = 0
    #counts acts as the keys for the datapoints
    counts = 0
    #in this loop I get the distance from each point in the training data to my testing point I store
    # all this information and then sort the training datapoints by their distance in ascending order
    for tx in trainX:
        ed = 0
        for i in range(len(tx)):
            ed += (tx[i] - testX[i])**2
        edists.append((counts,math.sqrt(ed)))
        counts += 1
    edists = sorted(edists, key=lambda x:x[1])
```

```
    #once I take the nearest k neighbors I count the votes of them to get the class label for the training point
    nn = edists[:k]
    votes = {}
    for n in nn:
        votes.update({trainY[n[0]]:votes.get(trainY[n[0]],0)+1})
    return sorted([(k, v) for k, v in votes.items()], key=lambda x:x[1])[0][0]


#my main method takes the value for k for kmeans and the number of bins to be used in building the model
def main(numBins, k):
    print('with %i bins and %i-nearest neighbors:'%(numBins, k))
    #use ~brute-force to collect the file strings for the training images and their class labels
    #afterwards I call my method to create the model from the training data and repeat with the testing images
    fileStrings = []
    for b in ['coast','forest','insidecity']:
        for i in range(1,5):
            fileStrings.append('./ImClass/%s_train%i.jpg'%(b,i))
    labels = ['coast']*4+['forest']*4+['insidecity']*4
    imgC = makeImageClassifier(fileStrings,numBins)
    fileStrings2 = []
    for b in ['coast','forest','insidecity']:
        for i in range(1,5):
            fileStrings2.append('./ImClass/%s_test%i.jpg'%(b,i))
    imgC2 = makeImageClassifier(fileStrings2,numBins)
    labels2 = []
    #finally I assign labels to my testing images' histograms using my training information and my knn function
    #acc stores the accuracy of my model and count stores the index of the correct labels in: labels
```

```
    acc, count = 0,0
    for img2 in imgC2:
        labels2.append(knn(img2,imgC,labels,k))
        print('Test image %i of class \"%s\" has been assigned to class
\"%s\".'%(math.fmod(count,4)+1,labels[count],labels2[-1]))
        count += 1
        if(labels[count-1] == labels2[-1]):
            acc+=1/len(imgC2)
    print('accuracy = %f\n'%acc)


main(8, 1)
main(4, 1)
main(16, 1)
main(32, 1)
main(8, 3)
```