```
import cv2
import math
import numpy as np
import os

# Mathew Seedhom
# CS 558
# Homework 1
# I pledge my honor that I have abided by the Stevens Honor System.

def histBins(pic, k):
    # Creates a k bin histogram for an image by looking at RGB
channels of each pixel
    bins = []
    for i in range(k):
        bins += [0, 0, 0]
    for i in range(len(pic)):
        for j in range(len(pic)):
            red = (pic[i][j][0])
            green = (pic[i][j][1])
            blue = (pic[i][j][2])
            bins[int(red // (256/k))] += 1
            bins[k + int(green // (256/k))] += 1
            bins[2*k + int(blue // (256/k))] += 1
    return bins

def distance(first, second):
    # Euclidean distance function
    d = 0
    for i in range(len(first)):
         d += (first[i]-second[i]) ** 2
    return d

def categorize(labels):
    # Determines what label an image will be predicted as
    label = {"coast": 0, "forest": 0, "insidecity": 0, "sky": 0}
    for i in range(len(labels)):
        if labels[i] == "coast":
            label["coast"] += len(labels)-i
        elif labels[i] == "forest":
            label["forest"] += len(labels)-i
        elif labels[i] == "insidecity":
            label["insidecity"] += len(labels)-i
        elif labels[i] == "sky":
            label["sky"] += len(labels)-i
    return max(label, key=label.get)

def kNearest(test, train, n):
    # Finds closest predictions to an image
    accuracy = 0
    for i in range(len(test)):
        minK = []
        minLabels = []
        for l in range(n):
            minK += [-1]
            minLabels += [0]
        for j in range(len(train)):
            d = distance(test[i][0], train[j][0])
            for k in range(n):
                if d < minK[k] or minK[k] == -1:
                    minK = minK[:k] + [d] + minK[k:-1]
                    minLabels = minLabels[:k] + [train[j][1]] + 
minLabels[k:-1]
                    break
        category = categorize(minLabels)
        print(test[i][2] + " of class " + test[i][1] + " was assigned 
to " + category + " class.")
        if test[i][1] == category:
            accuracy += 1
    print("\nRatio of images categorized correctly: " + str(accuracy / 
len(test) * 100) + "% or " + str(accuracy) + "/" + str(len(test)))

if __name__ == "__main__":
    print("Working on it!\n")
    trainedBins = []
    testBins = []
    verification = []
    bins = int(input("How many bins would you like? "))
    knear = int(input("How many neighbors would you like to compare 
to? "))

    while True:
        sky = input("Would you like to include sky categorization? [Y/
N] ")
        if sky.lower() == "y" or sky.lower() == "yes":
            sky = 1
            break
        elif sky.lower() == "n" or sky.lower() == "no":
            sky = 0
            break
        else:
            print("Inavlid response. Please type y or n!")

    #Pre-processing
    for filename in os.listdir("ImClass/"):
        pic = cv2.imread("ImClass/" + filename, 1)
        which = 0
        if filename.startswith("coast_train"):
            trainedBins.append([histBins(pic, bins), "coast"])
        elif filename.startswith("forest_train"):
            trainedBins.append([histBins(pic, bins), "forest"])
        elif filename.startswith("insidecity_train"):
            trainedBins.append([histBins(pic, bins), "insidecity"])
        elif filename.startswith("coast_test"):
            which = 1
            testBins.append([histBins(pic, bins), "coast", filename])
        elif filename.startswith("forest_test"):
            which = 1
            testBins.append([histBins(pic, bins), "forest", filename])
        elif filename.startswith("insidecity_test"):
            which = 1
            testBins.append([histBins(pic, bins), "insidecity",
filename])
        if which == 0:
            verification += [not (len(pic)*len(pic[0])*3 ==
sum(trainedBins[-1][0]))]
        else:
            verification += [not (len(pic)*len(pic[0])*3 ==
sum(testBins[-1][0]))]
    if sky == 1:
        for filename in os.listdir("sky/"):
            pic = cv2.imread("sky/" + filename, 1)
            which = 0
            if filename.startswith("sky_train"):
                trainedBins.append([histBins(pic, bins), "sky"])
            elif filename.startswith("sky_test"):
                which = 1
                testBins.append([histBins(pic, bins), "sky",
filename])
            if which == 0:
                verification += [not (len(pic)*len(pic[0])*3 ==
sum(trainedBins[-1][0]))]
            else:
                verification += [not (len(pic)*len(pic[0])*3 ==
sum(testBins[-1][0]))]

    # Verifies each pixel was properly accounted in each image
    if not sum(verification):
        print("\nAll pixels have been properly accounted for in each
picture.")
    else:
        print("\n" + str(sum(verification)) + " pictures did not
account for all pixels in training.")
    print("\nResults for " + str(bins) + " bins, " + str(knear) + "
nearest neighbors, " + "sky images " + (1-sky) * "not " +
"included\n")
    kNearest(testBins, trainedBins, knear)
    print("\nAll done!")
```