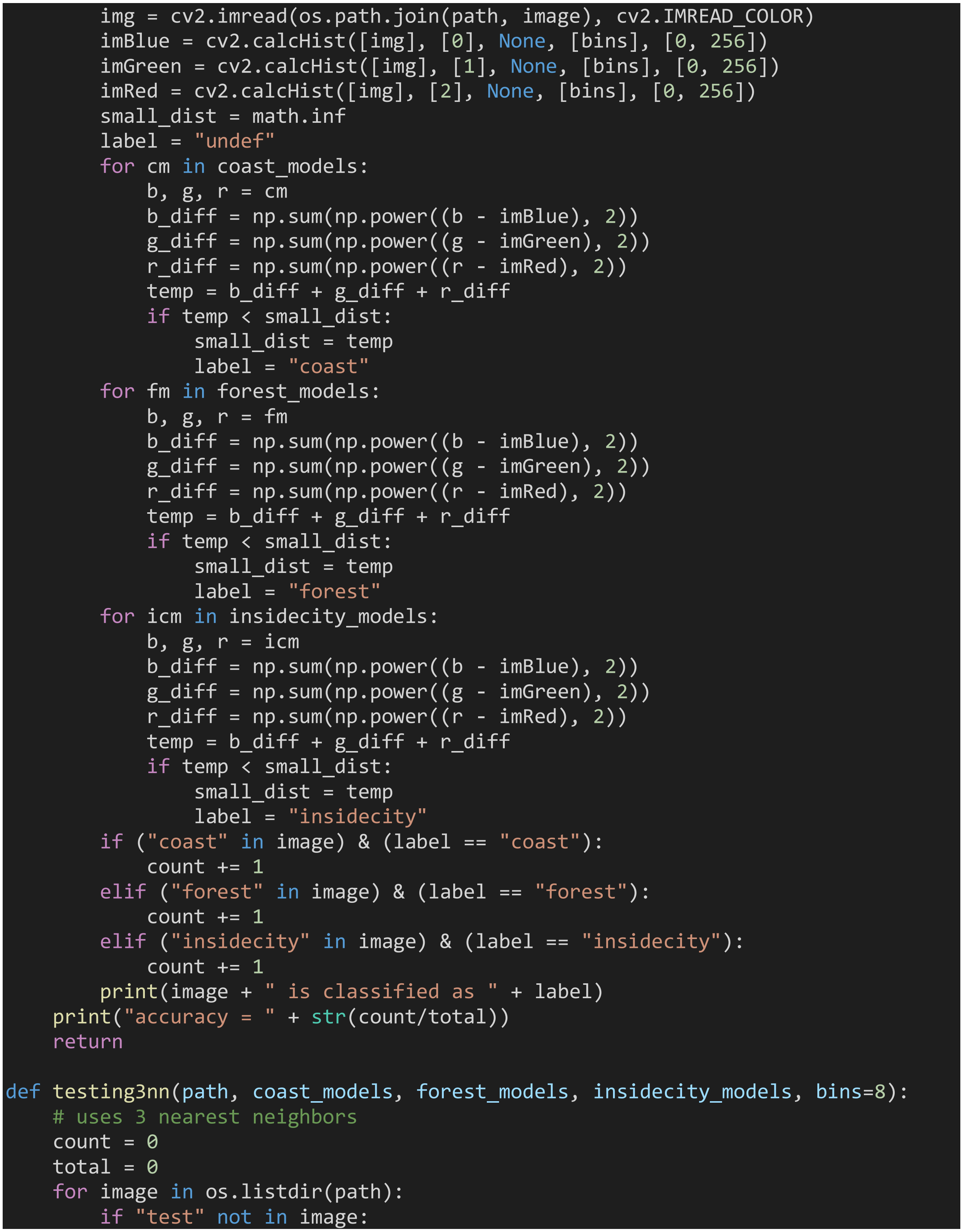
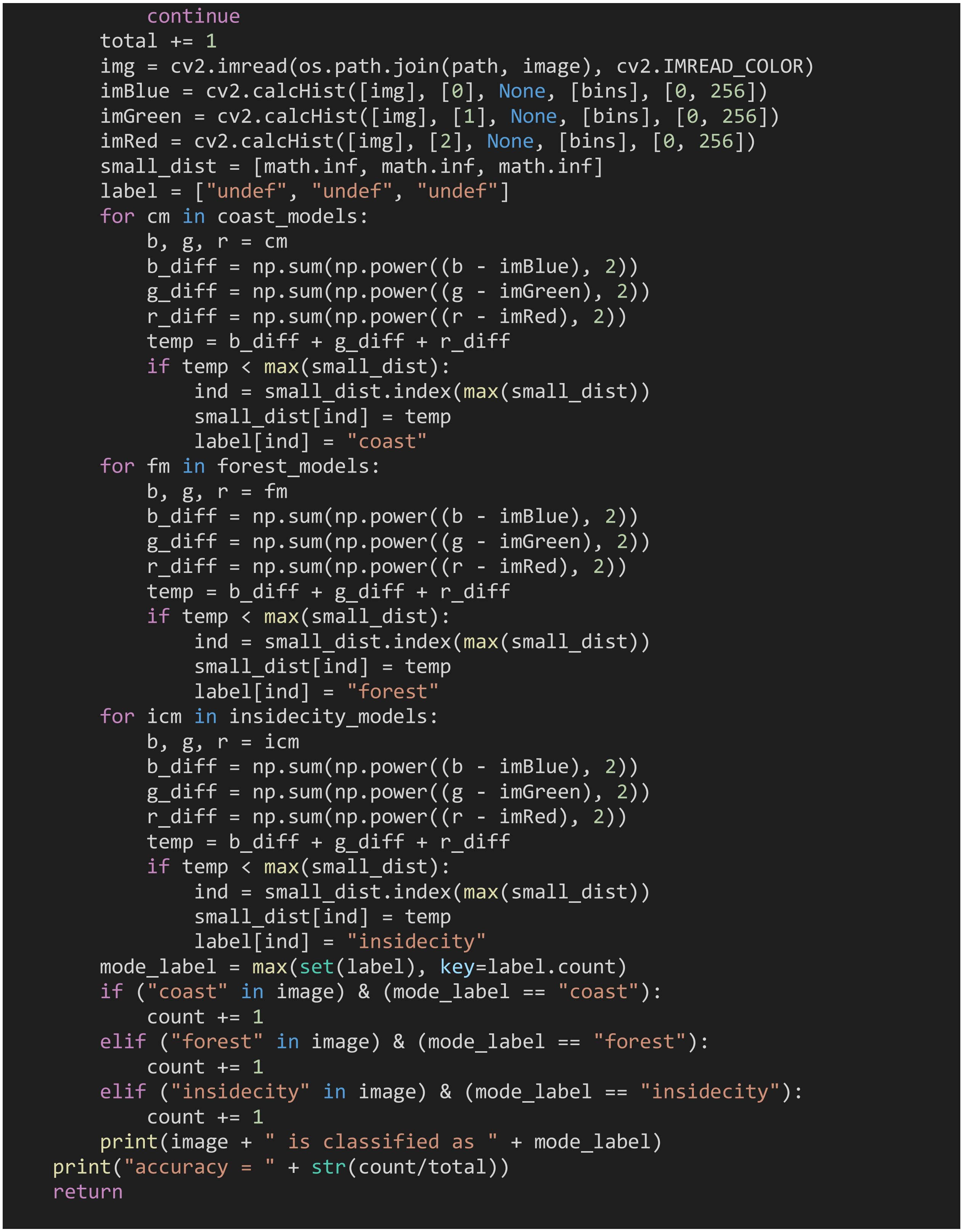
Yoshika Takezawa
5/11/19
CS 558, hw3
Code :

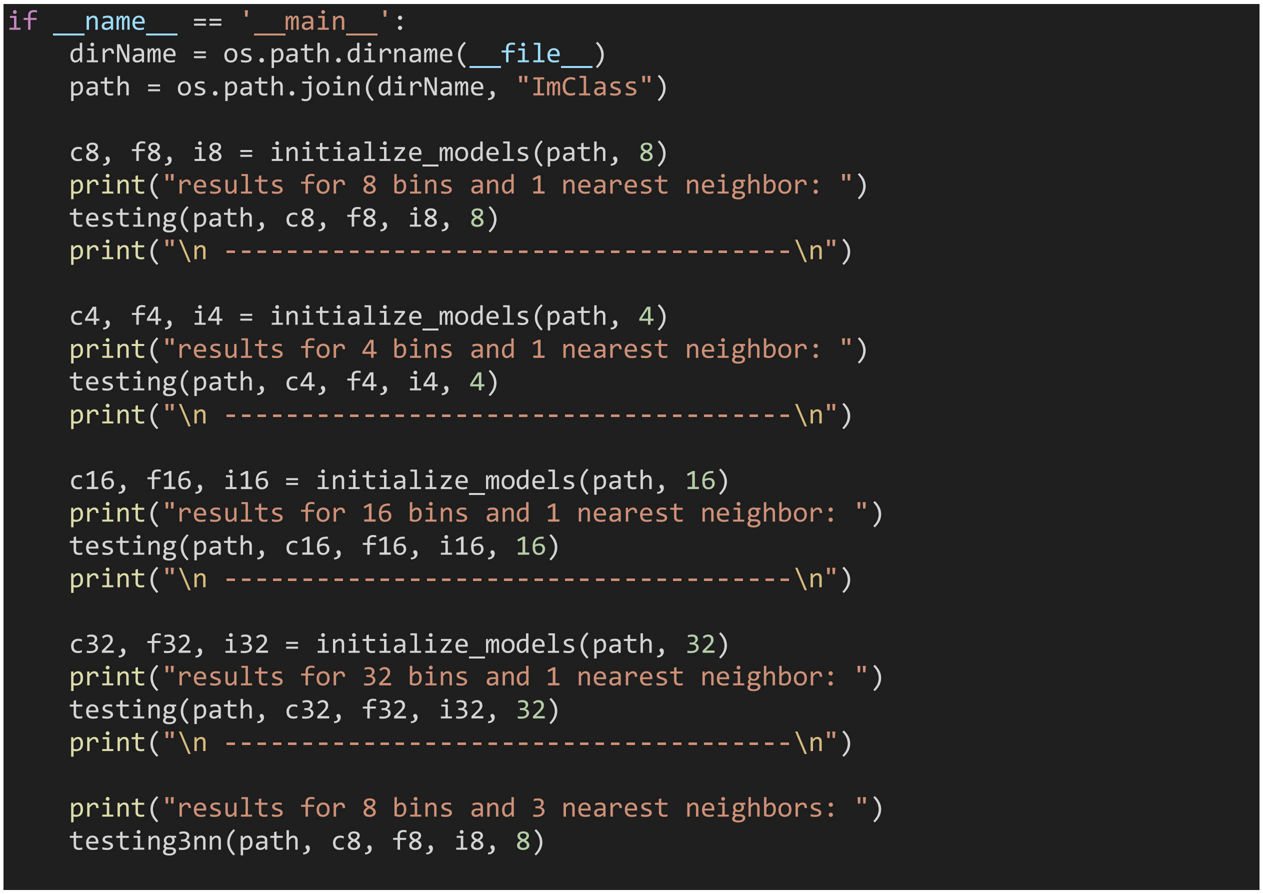
```
# Yoshika Takezawa
# 5/11/19
# HW 3 : image classifier based on color histograms
# I pledge my honor that I have abided by the Stevens Honor System
import numpy as np
import sys
import os
import cv2
import math

def initialize_models(path, bins=8):
    # return an array of coast, forest, inside city
    # each category is set up as (img1 blue, green, red), (img2 blue, green,
red), etc
    coast = list()
    forest = list()
    insidecity = list()
    for image in os.listdir(path):
        if "train" not in image:
            continue
        img = cv2.imread(os.path.join(path, image), cv2.IMREAD_COLOR)
        imBlue = cv2.calcHist([img], [0], None, [bins], [0, 256])
        imGreen = cv2.calcHist([img], [1], None, [bins], [0, 256])
        imRed = cv2.calcHist([img], [2], None, [bins], [0, 256])
        temp = (imBlue, imGreen, imRed)
        if "coast" in image:
            coast.append(temp)
        elif "forest" in image:
            forest.append(temp)
        elif "insidecity" in image:
            insidecity.append(temp)
        else:
            continue
    # print(coast, forest, insidecity)
    return coast, forest, insidecity

def testing(path, coast_models, forest_models, insidecity_models, bins=8):
    # uses 1 nearest neighbor
    count = 0
    total = 0
    for image in os.listdir(path):
        if "test" not in image:
            continue
        total += 1
```



```
        img = cv2.imread(os.path.join(path, image), cv2.IMREAD_COLOR)
        imBlue = cv2.calcHist([img], [0], None, [bins], [0, 256])
        imGreen = cv2.calcHist([img], [1], None, [bins], [0, 256])
        imRed = cv2.calcHist([img], [2], None, [bins], [0, 256])
        small_dist = math.inf
        label = "undef"
        for cm in coast_models:
            b, g, r = cm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < small_dist:
                small_dist = temp
                label = "coast"
        for fm in forest_models:
            b, g, r = fm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < small_dist:
                small_dist = temp
                label = "forest"
        for icm in insidecity_models:
            b, g, r = icm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < small_dist:
                small_dist = temp
                label = "insidecity"
        if ("coast" in image) & (label == "coast"):
            count += 1
        elif ("forest" in image) & (label == "forest"):
            count += 1
        elif ("insidecity" in image) & (label == "insidecity"):
            count += 1
        print(image + " is classified as " + label)
    print("accuracy = " + str(count/total))
    return

def testing3nn(path, coast_models, forest_models, insidecity_models, bins=8):
    # uses 3 nearest neighbors
    count = 0
    total = 0
    for image in os.listdir(path):
        if "test" not in image:
```



```
            continue
        total += 1
        img = cv2.imread(os.path.join(path, image), cv2.IMREAD_COLOR)
        imBlue = cv2.calcHist([img], [0], None, [bins], [0, 256])
        imGreen = cv2.calcHist([img], [1], None, [bins], [0, 256])
        imRed = cv2.calcHist([img], [2], None, [bins], [0, 256])
        small_dist = [math.inf, math.inf, math.inf]
        label = ["undef", "undef", "undef"]
        for cm in coast_models:
            b, g, r = cm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < max(small_dist):
                ind = small_dist.index(max(small_dist))
                small_dist[ind] = temp
                label[ind] = "coast"
        for fm in forest_models:
            b, g, r = fm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < max(small_dist):
                ind = small_dist.index(max(small_dist))
                small_dist[ind] = temp
                label[ind] = "forest"
        for icm in insidecity_models:
            b, g, r = icm
            b_diff = np.sum(np.power((b - imBlue), 2))
            g_diff = np.sum(np.power((g - imGreen), 2))
            r_diff = np.sum(np.power((r - imRed), 2))
            temp = b_diff + g_diff + r_diff
            if temp < max(small_dist):
                ind = small_dist.index(max(small_dist))
                small_dist[ind] = temp
                label[ind] = "insidecity"
        mode_label = max(set(label), key=label.count)
        if ("coast" in image) & (mode_label == "coast"):
            count += 1
        elif ("forest" in image) & (mode_label == "forest"):
            count += 1
        elif ("insidecity" in image) & (mode_label == "insidecity"):
            count += 1
        print(image + " is classified as " + mode_label)
    print("accuracy = " + str(count/total))
    return
```



```
if __name__ == '__main__':
    dirName = os.path.dirname(__file__)
    path = os.path.join(dirName, "ImClass")

    c8, f8, i8 = initialize_models(path, 8)
    print("results for 8 bins and 1 nearest neighbor: ")
    testing(path, c8, f8, i8, 8)
    print("\n --------------------------------------\n")

    c4, f4, i4 = initialize_models(path, 4)
    print("results for 4 bins and 1 nearest neighbor: ")
    testing(path, c4, f4, i4, 4)
    print("\n --------------------------------------\n")

    c16, f16, i16 = initialize_models(path, 16)
    print("results for 16 bins and 1 nearest neighbor: ")
    testing(path, c16, f16, i16, 16)
    print("\n --------------------------------------\n")

    c32, f32, i32 = initialize_models(path, 32)
    print("results for 32 bins and 1 nearest neighbor: ")
    testing(path, c32, f32, i32, 32)
    print("\n --------------------------------------\n")

    print("results for 8 bins and 3 nearest neighbors: ")
    testing3nn(path, c8, f8, i8, 8)
```

Results:

```
D:\cs 558\hw3>python ./hw3.py
results for 8 bins and 1 nearest neighbor:
coast_test1.jpg is classified as coast
coast_test2.jpg is classified as coast
coast_test3.jpg is classified as coast
coast_test4.jpg is classified as coast
forest_test1.jpg is classified as forest
forest_test2.jpg is classified as forest
forest_test3.jpg is classified as forest
forest_test4.jpg is classified as forest
insidecity_test1.jpg is classified as forest
insidecity_test2.jpg is classified as forest
insidecity_test3.jpg is classified as insidecity
insidecity_test4.jpg is classified as insidecity
accuracy = 0.8333333333333334

 -------------------------------------

results for 4 bins and 1 nearest neighbor:
coast_test1.jpg is classified as coast
coast_test2.jpg is classified as coast
coast_test3.jpg is classified as coast
coast_test4.jpg is classified as coast
forest_test1.jpg is classified as forest
forest_test2.jpg is classified as forest
forest_test3.jpg is classified as forest
forest_test4.jpg is classified as forest
insidecity_test1.jpg is classified as forest
insidecity_test2.jpg is classified as insidecity
insidecity_test3.jpg is classified as insidecity
insidecity_test4.jpg is classified as coast
accuracy = 0.8333333333333334

 -------------------------------------

results for 16 bins and 1 nearest neighbor:
coast_test1.jpg is classified as coast
coast_test2.jpg is classified as coast
coast_test3.jpg is classified as coast
coast_test4.jpg is classified as coast
forest_test1.jpg is classified as forest
forest_test2.jpg is classified as forest
forest_test3.jpg is classified as forest
```

```
forest_test4.jpg is classified as forest
insidecity_test1.jpg is classified as forest
insidecity_test2.jpg is classified as forest
insidecity_test3.jpg is classified as forest
insidecity_test4.jpg is classified as insidecity
accuracy = 0.75

 ------------------------------------

results for 32 bins and 1 nearest neighbor:
coast_test1.jpg is classified as coast
coast_test2.jpg is classified as coast
coast_test3.jpg is classified as coast
coast_test4.jpg is classified as coast
forest_test1.jpg is classified as forest
forest_test2.jpg is classified as forest
forest_test3.jpg is classified as forest
forest_test4.jpg is classified as forest
insidecity_test1.jpg is classified as forest
insidecity_test2.jpg is classified as forest
insidecity_test3.jpg is classified as forest
insidecity_test4.jpg is classified as insidecity
accuracy = 0.75

 ------------------------------------

results for 8 bins and 3 nearest neighbors:
coast_test1.jpg is classified as coast
coast_test2.jpg is classified as coast
coast_test3.jpg is classified as coast
coast_test4.jpg is classified as coast
forest_test1.jpg is classified as forest
forest_test2.jpg is classified as coast
forest_test3.jpg is classified as forest
forest_test4.jpg is classified as insidecity
insidecity_test1.jpg is classified as forest
insidecity_test2.jpg is classified as insidecity
insidecity_test3.jpg is classified as forest
insidecity_test4.jpg is classified as coast
accuracy = 0.5833333333333334
```

The image classifier based on image histograms and Euclidean distance results in an accuracy ranging from 0.533 and 0.833. As the training images did not have a large enough variety, the accuracy was not as stable as it could be. Despite this, it produces decent results. The algorithm bases heavily on brute force, calculating solely on the color distance, and a low level of machine learning.