Alejandro Garcia
5/4/2020
"I pledge my honor that I have abided by the stevens honor system"

Problem 1

The first step of the image classification is to go through all the training files and extract their features/assign them labels. The features that were extracted were the values of the histogram in a 1d array. While the labels were assigned based on their file names(i.e 'coast', 'forest', 'insidecity'). The training files and test files were separated into their own folders.

```
    training_files = [f for f in listdir('./ImClass/train/') if
isfile(join('./ImClass/train/', f))]

    features = []
    labels = []

    for i in range(len(training_files)):
        img = mpimg.imread('./ImClass/train/' + training_files[i])

        hist = histogram(img)

        string = re.search(r'([\w.-]+)_([\w.-]+)', training_files[i])
        label = string.group(1)



        features.append(hist)
        labels.append(label)

    features = np.array(features)
    labels = np.array(labels)
```

The histogram function works as followed below. The RGB values of each pixel were stored in their assigned bins. In this case, there are 8 bins. This returns the histogram in a 1d array for the features array to create a 2d array as the K-Neighbors Classifier takes in a 2d array or less.

```
def histogram(img):
    colors = ['Red', 'Green', 'Blue']

    hist_r = np.histogram(img[:, :, 0].ravel(), bins = range(9))[0]
    hist_g = np.histogram(img[:, :, 1].ravel(), bins = range(9))[0]
    hist_b = np.histogram(img[:, :, 2].ravel(), bins = range(9))[0]

    hist = np.array([hist_r, hist_g, hist_b])

    hist = hist.ravel()

    print('Verifying iteration of img pixels is three times: ' +
str(len(hist)))

    return hist
```

The K Neighbors data structure used was sklearn.

```
from sklearn.neighbors import KNeighborsClassifier
```

With this data structure, the K Neighbors Classifier was set to that number of neighbors inputted. In the case below, the number of neighbors is 1. Then, we fit the features with their perspective labels.

```
    knn = KNeighborsClassifier(n_neighbors=1)

    knn.fit(features, labels)
```

Next, for each of the testing files. The testing features and labels would be extracted. Using the testing features, the predicted label would then be outputted.

```
testing_features = []
    testing_labels = []

    for i in range(len(testing_files)):
        img = mpimg.imread('./ImClass/test/' + testing_files[i])

        hist = histogram(img)
```

```
        string = re.search(r'([\w.-]+)_([\w.-]+)', testing_files[i])
        label = string.group(1)


        testing_features.append(hist)
        testing_labels.append(label)

    testing_features = np.array(testing_features)

    predicted = knn.predict(testing_features)
```

Below is the format described as:
Test image 1 of class 2 has been assigned to class 1

As well as the accuracy.

```
    print()

    for i in range(len(predicted)):
        print('Test Image ' + testing_files[i] + ' of class ' +
testing_labels[i] + ' has been assigned to ' + predicted[i])

    acc = knn.score(testing_features, testing_labels)

    print()

    print('Classifier Accuracy: ' + str(acc))
```

Outputs:

For a K-Neighbors = 1 and 8 bins for the histogram:

Test Image coast_test4.jpg of class coast has been assigned to coast
Test Image coast_test3.jpg of class coast has been assigned to insidecity
Test Image coast_test1.jpg of class coast has been assigned to insidecity
Test Image insidecity_test3.jpg of class insidecity has been assigned to insidecity
Test Image insidecity_test2.jpg of class insidecity has been assigned to insidecity

Test Image insidecity_test4.jpg of class insidecity has been assigned to forest
Test Image forest_test2.jpg of class forest has been assigned to coast
Test Image forest_test1.jpg of class forest has been assigned to forest
Test Image insidecity_test1.jpg of class insidecity has been assigned to forest
Test Image forest_test3.jpg of class forest has been assigned to insidecity
Test Image forest_test4.jpg of class forest has been assigned to coast
Test Image coast_test2.jpg of class coast has been assigned to forest

Classifier Accuracy: 0.3333333333333333

For a K-Neighbors = 1 and 4 bins for the histogram:

Test Image coast_test4.jpg of class coast has been assigned to insidecity
Test Image coast_test3.jpg of class coast has been assigned to coast
Test Image coast_test1.jpg of class coast has been assigned to insidecity
Test Image insidecity_test3.jpg of class insidecity has been assigned to forest
Test Image insidecity_test2.jpg of class insidecity has been assigned to forest
Test Image insidecity_test4.jpg of class insidecity has been assigned to coast
Test Image forest_test2.jpg of class forest has been assigned to coast
Test Image forest_test1.jpg of class forest has been assigned to forest
Test Image insidecity_test1.jpg of class insidecity has been assigned to coast
Test Image forest_test3.jpg of class forest has been assigned to insidecity
Test Image forest_test4.jpg of class forest has been assigned to coast
Test Image coast_test2.jpg of class coast has been assigned to insidecity

Classifier Accuracy: 0.16666666666666666

For a K-Neighbors = 1 and 16 bins for the histogram:

Test Image coast_test4.jpg of class coast has been assigned to insidecity
Test Image coast_test3.jpg of class coast has been assigned to insidecity
Test Image coast_test1.jpg of class coast has been assigned to insidecity
Test Image insidecity_test3.jpg of class insidecity has been assigned to insidecity
Test Image insidecity_test2.jpg of class insidecity has been assigned to insidecity
Test Image insidecity_test4.jpg of class insidecity has been assigned to insidecity
Test Image forest_test2.jpg of class forest has been assigned to coast
Test Image forest_test1.jpg of class forest has been assigned to forest
Test Image insidecity_test1.jpg of class insidecity has been assigned to insidecity
Test Image forest_test3.jpg of class forest has been assigned to insidecity
Test Image forest_test4.jpg of class forest has been assigned to coast
Test Image coast_test2.jpg of class coast has been assigned to forest

Classifier Accuracy: 0.4166666666666667

For a K-Neighbors = 1 and 32 bins for the histogram:

Test Image coast_test4.jpg of class coast has been assigned to coast
Test Image coast_test3.jpg of class coast has been assigned to insidecity
Test Image coast_test1.jpg of class coast has been assigned to insidecity
Test Image insidecity_test3.jpg of class insidecity has been assigned to insidecity
Test Image insidecity_test2.jpg of class insidecity has been assigned to forest
Test Image insidecity_test4.jpg of class insidecity has been assigned to insidecity
Test Image forest_test2.jpg of class forest has been assigned to forest
Test Image forest_test1.jpg of class forest has been assigned to forest
Test Image insidecity_test1.jpg of class insidecity has been assigned to insidecity
Test Image forest_test3.jpg of class forest has been assigned to coast
Test Image forest_test4.jpg of class forest has been assigned to coast
Test Image coast_test2.jpg of class coast has been assigned to forest

Classifier Accuracy: 0.5

For a K-Neighbors = 3 and 8 bins for the histogram:

Test Image coast_test4.jpg of class coast has been assigned to insidecity
Test Image coast_test3.jpg of class coast has been assigned to insidecity
Test Image coast_test1.jpg of class coast has been assigned to insidecity
Test Image insidecity_test3.jpg of class insidecity has been assigned to forest
Test Image insidecity_test2.jpg of class insidecity has been assigned to forest
Test Image insidecity_test4.jpg of class insidecity has been assigned to forest
Test Image forest_test2.jpg of class forest has been assigned to coast
Test Image forest_test1.jpg of class forest has been assigned to forest
Test Image insidecity_test1.jpg of class insidecity has been assigned to forest
Test Image forest_test3.jpg of class forest has been assigned to insidecity
Test Image forest_test4.jpg of class forest has been assigned to coast
Test Image coast_test2.jpg of class coast has been assigned to forest

Classifier Accuracy: 0.08333333333333333

Problem 2

The first step was to use gimp to label the sky pixels within the training image. This allowed us to produce the gimp training image as shown below.

Using the original training image and the gimp image, a mask was created by subtracting the two images and retrieving the difference of the images. If the RGB pixels add up to 0 then that is a non sky pixel. If not then the pixels belong to the sky set.

```
sky_train = Image.open('./sky/train/sky_train.jpg')
    sky_gimp = Image.open('./sky/train/sky_train_gimp.jpg')

    sky_points = []
    non_sky_points = []

    diff = ImageChops.difference(sky_train, sky_gimp)

    diff = np.array(diff, dtype = np.uint8)
    sky_train = np.array(sky_train, dtype = np.uint8)

    img_y = sky_train.shape[0]
    img_x = sky_train.shape[1]

    for i in range(img_y):
        for j in range(img_x):
            total_diff = int(diff[i][j][0]) + int(diff[i][j][1]) +
int(diff[i][j][2])
            if total_diff != 0:
                r = sky_train[i][j][0]
                g = sky_train[i][j][1]
```

```
                b = sky_train[i][j][2]

                sky_points.append([r, g, b])

            else:
                r = sky_train[i][j][0]
                g = sky_train[i][j][1]
                b = sky_train[i][j][2]

                non_sky_points.append([r, g, b])

    sky_points = np.array(sky_points)
    non_sky_points = np.array(non_sky_points)
```

Next K-means was applied by using the sklearn library. This would allow us to obtain the 10 visual bags for both the sky set and non sky set.

```
    kmeans_sky = KMeans(n_clusters=10)
    kmeans_sky.fit(sky_points)
    sky_centroids = kmeans_sky.cluster_centers_
    sky_centroids = sky_centroids.astype(int)

    kmeans_non_sky = KMeans(n_clusters=10)
    kmeans_non_sky.fit(non_sky_points)
    non_sky_centroids = kmeans_non_sky.cluster_centers_
    non_sky_centroids = non_sky_centroids.astype(int)
```

Then we would create a features array to extract and label both of the k means sets. With that information we create our model using a k neighbors value of 5 which can be arbitrary.

```
    features = []
    labels = []

    for i in range(20):
        if i < 10:
            label = 'sky'
            features.append(sky_centroids[i])
        else:
```

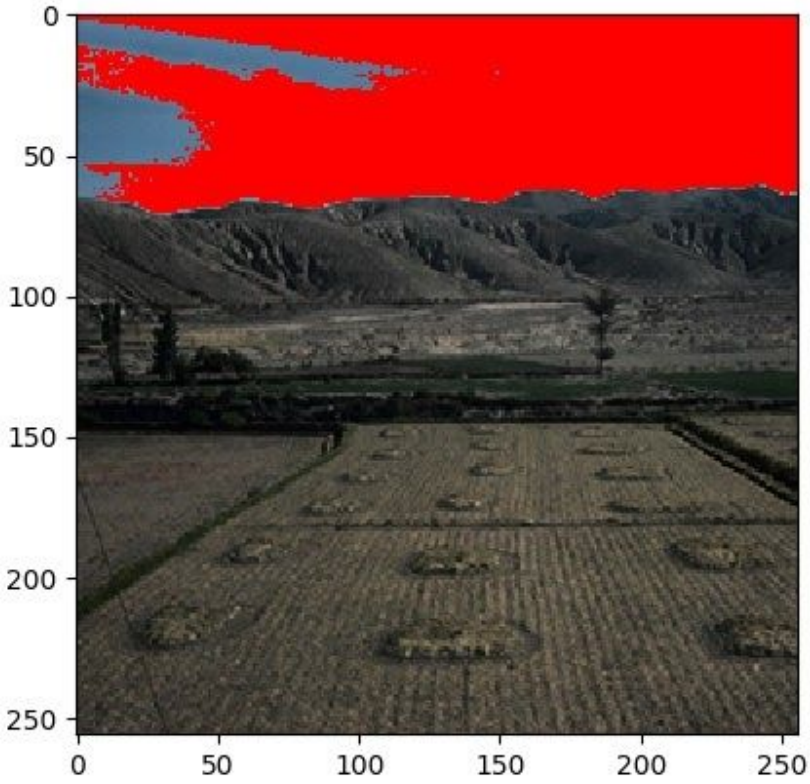
```
            label = 'non-sky'
            features.append(non_sky_centroids[i - 10])

        labels.append(label)

    features = np.array(features)
    labels = np.array(labels)

    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(features, labels)
```

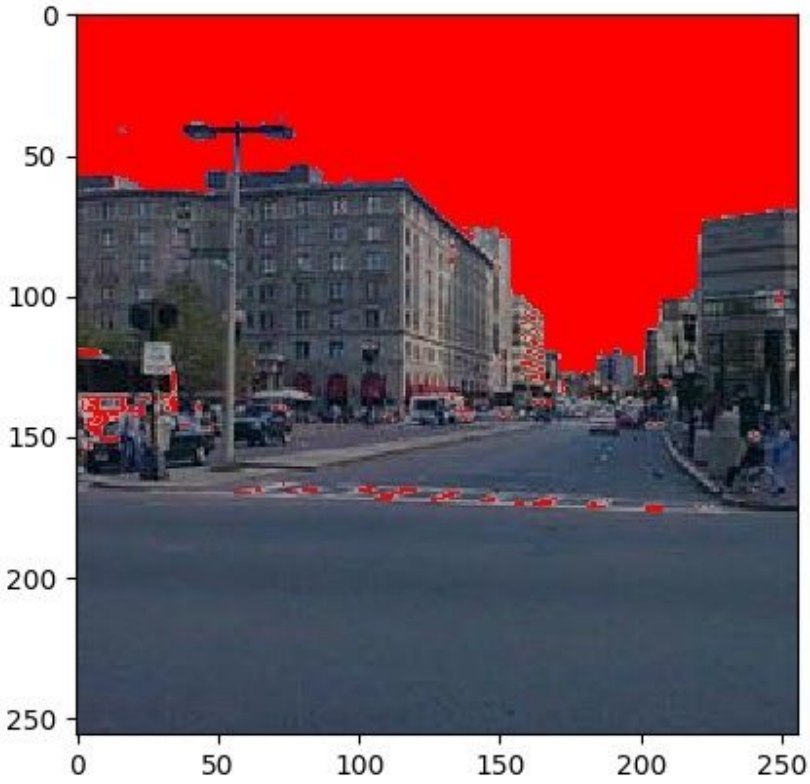
For each of the testing files, we would then extract their features and predict whether or not that pixel value is part of the sky or not. The color red was then used to demonstrate what pixels are part of the sky.

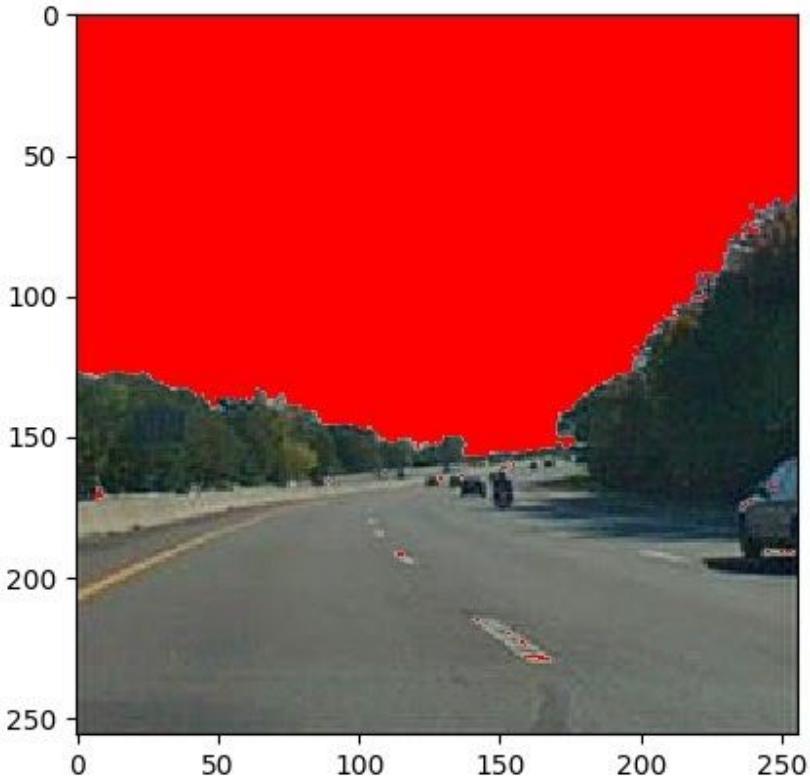
```
    testing_files = [f for f in listdir('./sky/test/') if
isfile(join('./sky/test/', f))]

    for i in range(len(testing_files)):
        test_img = mpimg.imread('./sky/test/' + testing_files[i])

        testing_features = []

        test_img_y = test_img.shape[0]
        test_img_x = test_img.shape[1]

        for y in range(test_img_y):
            for x in range(test_img_x):
                r = test_img[y][x][0]
                g = test_img[y][x][1]
                b = test_img[y][x][2]

                testing_features.append([r, g, b])

        testing_features = np.array(testing_features)

        predicted = knn.predict(testing_features)
```

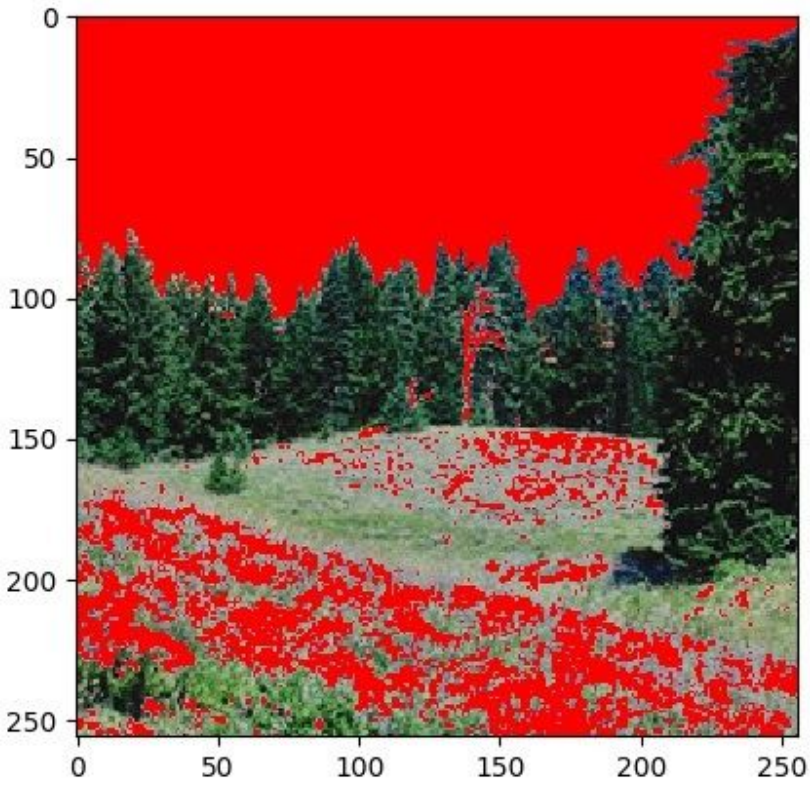
```
        copy_img = test_img.copy()

        for y in range(test_img_y):
            for x in range(test_img_x):
                if predicted[test_img_x*y + x] == 'sky':
                    copy_img[y][x][0] = 255
                    copy_img[y][x][1] = 0
                    copy_img[y][x][2] = 0

        plt.imshow(copy_img)
        img_name = 'pixel_classification_' + str(i + 1) + '.jpg'
        plt.savefig(img_name)
        plt.show()
```

Output:




0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250



The reason why it captures the sky so well is due to the fact the sky is majority blue and the k means clusters had around a 180 to 240 interval of blue values. However, this would not go good with other high intensity blue values such as the image above. Due to that fact, the ground is considered as the sky. To fix this, we could use feature extraction rather than pixel extraction because we have a collection of what the sky looks like rather than interpreting the RGB values.