IMAGE CLASSIFICATION USING KNN

Data Set:

Collection of 92 images with 30 images of face, 31 images of night and 31 images of landscape taken from google images.

Steps Involved:

#Importing necessary packages:

- opencv(for image operations)
- numpy(for array operations)
- pandas(for dataframes)
- glob(for multiple file selection at a time)
- matpltolib(for graphical display)
- sklearn(knn)

#Defining our Image Descriptor:

- My image descriptor is 3D color histogram in the HSV color space (Hue, Saturation, Value).
- RGB values are simple to understand, the RGB color space fails to mimic how humans perceive color. Instead, I have used the **HSV color space** which maps pixel intensities into a cylinder.
- Now we need to define the number of **bins** for our histogram. Histograms are used to give a (rough) sense of the density of pixel intensities in an image. Essentially, histogram will estimate the probability density of the underlying function, or in this case, the probability of a pixel color occurring in image.
- If we select ***too few bins*****, then our histogram will have less components** and **unable to disambiguate between images** with substantially different color distributions. Likewise, if we use ***too many bins*** **our histogram will have many components and images with very similar contents may be regarded and "not similar" when in reality they are.**
- For our dataset, I am utilizing a 3D color histogram in the HSV color space with 8 bins for the Hue channel, 12 bins for the saturation channel, and 3 bins for the value channel, yielding a total feature vector of dimension ***8 x 12 x 3 = 288*****.**
- Instead of computing a 3D HSV color histogram for the *entire* image, computed a 3D HSV color histogram for **different *regions* of the image.**
- For our image descriptor, I have divided image into five different regions: (1) the top-left corner, (2) the top-right corner, (3) the bottom-right corner, (4) the bottom-left corner, and finally (5) the center of the image.

#Feature Extraction(Indexing):

- Looping over all the images in our dataset.
- For each of the images we'll extract an imageID, which is simply the filename of the image.
- The describe method of our ColorDescriptor returns a list of floating point values used to represent and quantify our image.
- **Given 5 entries, our overall feature vector is *5 x 288 = 1440* dimensionality.** Thus each image is quantified and represented using 1,440 numbers.

#Training Dataframe:

- training features(x_train) are the features we get from ColorDescriptor and the trained output(y_train) is put by me (labelled) as last column of our df dataframe.
- Face is given 0, Night images as 1 and Landscape images as 2
- Finally used KNN to train the data and later on predicted by some random images which gives us result as whether the image is face or night or landscape

#Result:

```
        print('landscape')

    #displaying accuracy
    print(knn.score(x_test, y_test))

    #asking for next image
    print("do you want another classification:  yes/no")
    y = input()
    if y == "yes":
        continue  #again new image
    else:
        break #break the loop
```

```
enter the file name of image:

/home/subarna/Documents/ml_notebooks/nigh.jpeg
night
1.0
do you want another classification:  yes/no
yes
enter the file name of image:

/home/subarna/Documents/ml_notebooks/images.jpeg
face
1.0
do you want another classification:  yes/no
```

In [ ]: