Project-3 Report

This project consisted of 2 parts, Part A dealt with Face Detection, where we built an algorithm to detect faces in an image. Part B dealt with performing face Clustering, where we detect similar faces in a set of images and group them together.

PART A (Face Detection)

- Part A of the project was pretty straightforward.
  For Facial Detection we use Cascade Classifier, many variations of the cascade classifier were used and 'haarcascade_frontalface_default.xml', gave the best result with F1 score of **0.81**.

- The Cascade Classifier takes in various arguments (Scale factor and minimum neighbors) which could be tweaked for a better result. After many trial values, the values 1.2 and 4 were chosen as they gave the best results., a higher Scale factor detected faces where there weren't any and a lower Scale factor did not detect existing faces in the image folder.

- After the Face Detection was done bounding boxes were created around the detected faces and saved in a json file, "results.json" for computing the F1 score. This was done using the Validation folder in the project3_data folder. The reults.json in the submission folder is the output of Face detection on Test data in project3 data folder.

PART B (Face Clustering)

- Part B was the challenging part of the project, we had to build a clustering model to cluster similar faces from a set of images in a folder.

- We used Part A to detect faces from a folder and encoded the images using face_encoding from face_recognition library. This encodes the image into a 128-D list, with each vector which resembles a particular feature in a face.

- This is necessary for clustering as all clustering algorithms work on spatial distance between the vectors.

- K-Means Clustering was chosen as the preferred Classifier for this task. Which had the following basic steps.

1. Pick K points as the initial centroids from the data set, either randomly or the first K.

   This was done at random with cluster centres being any random number between 1 and the number of Clusters to be formed, caution was taken that the numbers were unique and didn't repeat. This value of K was selected from the 'faceCluster_K' folder.

2. Find the Euclidean distance of each point in the data set with the identified K points — cluster centroids.

3. Assign each data point to the closest centroid using the distance found in the previous step.

4. Find the new centroid by taking the average of the points in each cluster group.

The average of all distances in a cluster were takes to set a new centroid. This would be the new Centroid for clustering.

5. This was repeated for N_Iterations (30 in my algorithm) or till the centroids don't change.

- The output of the model was saved in a json file "clusters.json".

The following clusters were the output of the model.

'cluster0'

'cluster1'

'cluster2'

'cluster3'

'cluster4'