MLBD Assignment-1

By:- Kwanit Gupta (B19EE046)

Description:-

Download the MNIST dataset from here. Pick its training test set which contains 60K samples, each represented by a 784-dimensional vector. There are ten classes (the ten digits, `0’ to `9’), and each sample is associated with one class. Preprocess the features using the appropriate technique(s), and store them in a file.

(1) Implement the BFR and CURE clustering algorithms on this data assuming that you can store `K1’ samples in the main memory at a time. One may use any existing libraries for performing the initial k-means clustering in case of BFR, and agglomerative clustering in case of CURE. After this, every step needs to be implemented from scratch.

(2) For the BFR algorithm, ensure that the number of clusters obtained at the end of the training the process is 10.

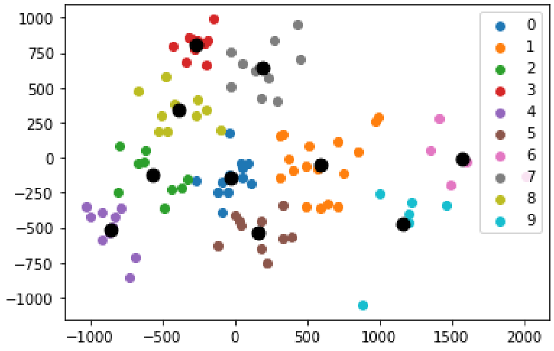
(3) For the CURE algorithm, keep the number of clusters as 10.

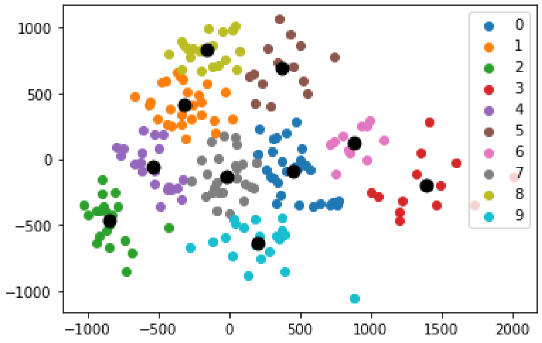
(4) After clustering, calculate the percentage of samples from each class and convert it into probability values. Using these, calculate the entropy of each cluster. Also calculate the total entropy of all the clusters by summing the entropy of individual clusters.

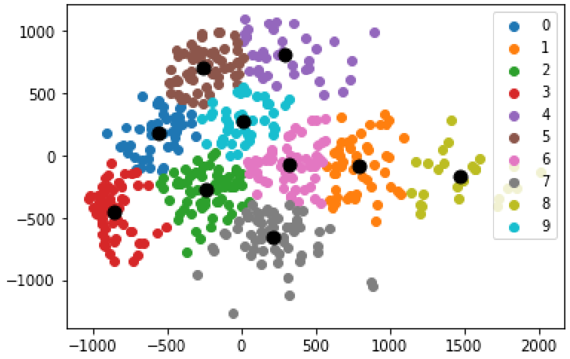
(5) Re-run the two algorithms five times assuming K1 = {100,200,500}, and report the above result.

Solution:-

For the pre-processing part, I utilized PCA (Principal Component Analysis), since variance has to be retained and escape from multiple 0s and 1s, since the visualization in the code wasn’t evident, eventually. After bringing down the n_components to 2, for useful visualizations,



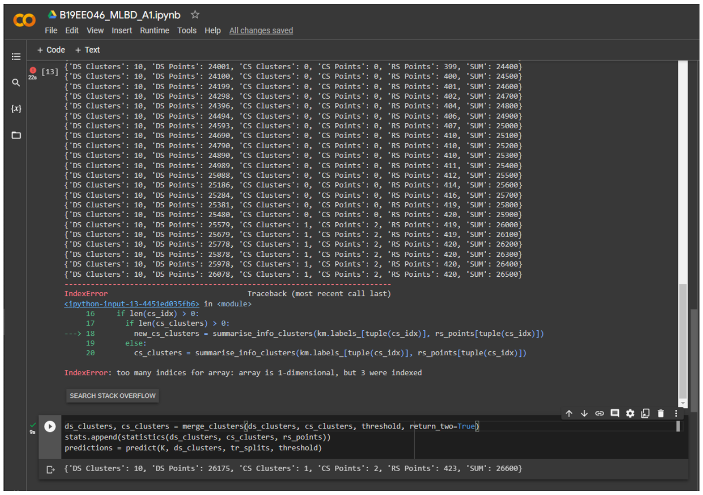




The Above 2 plots were the clustering results for K1=100 and K1=200. Whereas for the middle one, K1=500. Hereby, I observed the extensive intermixing of the PCA Components when K1 was increased.

Talking about the algorithms BFR with K-Means and CURE with Agglomerative Clustering, the dedicated code bases will be attached within the zip. Following are some of the screenshots for the working (Final conclusions didn't came since bugs weren't resolved, but some of the verbose can extensively show the promise of the implementations):-

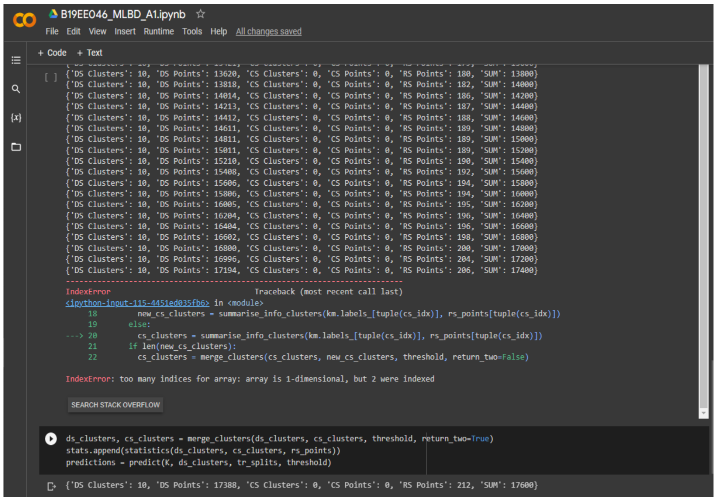
1. BFR with K-Means Clustering:-



```
{'DS Clusters': 10, 'DS Points': 24001, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 399, 'SUM': 24400}
{'DS Clusters': 10, 'DS Points': 24100, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 400, 'SUM': 24500}
{'DS Clusters': 10, 'DS Points': 24199, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 401, 'SUM': 24600}
{'DS Clusters': 10, 'DS Points': 24298, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 402, 'SUM': 24700}
{'DS Clusters': 10, 'DS Points': 24396, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 404, 'SUM': 24800}
{'DS Clusters': 10, 'DS Points': 24494, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 406, 'SUM': 24900}
{'DS Clusters': 10, 'DS Points': 24593, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 407, 'SUM': 25000}
{'DS Clusters': 10, 'DS Points': 24690, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 410, 'SUM': 25100}
{'DS Clusters': 10, 'DS Points': 24790, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 410, 'SUM': 25200}
{'DS Clusters': 10, 'DS Points': 24890, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 410, 'SUM': 25300}
{'DS Clusters': 10, 'DS Points': 24989, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 411, 'SUM': 25400}
{'DS Clusters': 10, 'DS Points': 25088, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 412, 'SUM': 25500}
{'DS Clusters': 10, 'DS Points': 25186, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 414, 'SUM': 25600}
{'DS Clusters': 10, 'DS Points': 25284, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 416, 'SUM': 25700}
{'DS Clusters': 10, 'DS Points': 25381, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 419, 'SUM': 25800}
{'DS Clusters': 10, 'DS Points': 25480, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 420, 'SUM': 25900}
{'DS Clusters': 10, 'DS Points': 25579, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 419, 'SUM': 26000}
{'DS Clusters': 10, 'DS Points': 25679, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 419, 'SUM': 26100}
{'DS Clusters': 10, 'DS Points': 25778, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 420, 'SUM': 26200}
{'DS Clusters': 10, 'DS Points': 25878, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 420, 'SUM': 26300}
{'DS Clusters': 10, 'DS Points': 25978, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 420, 'SUM': 26400}
{'DS Clusters': 10, 'DS Points': 26078, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 420, 'SUM': 26500}
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-13-4451ed035fb6> in <module>
     16     if len(cs_idx) > 0:
     17       if len(cs_clusters) > 0:
---> 18         new_cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
     19       else:
     20         cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])

IndexError: too many indices for array: array is 1-dimensional, but 3 were indexed
```

```
ds_clusters, cs_clusters = merge_clusters(ds_clusters, cs_clusters, threshold, return_two=True)
stats.append(statistics(ds_clusters, cs_clusters, rs_points))
predictions = predict(K, ds_clusters, tr_splits, threshold)
```

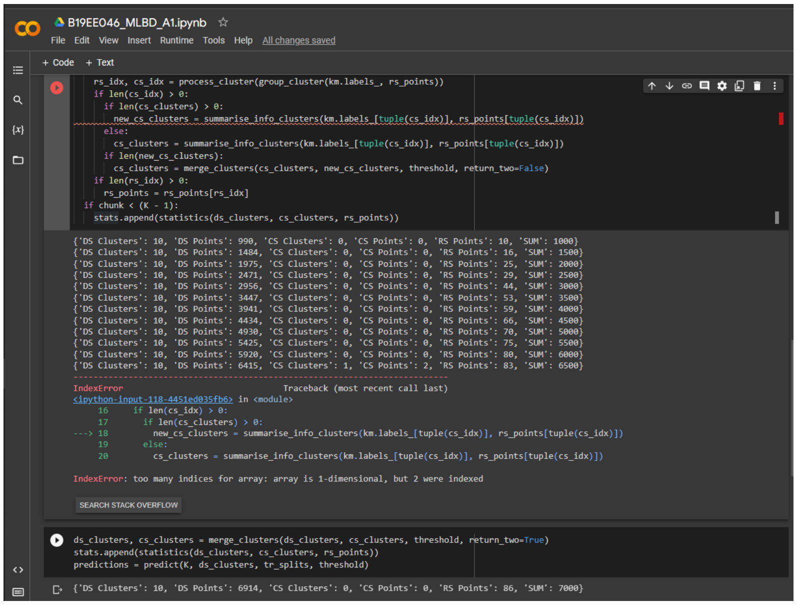
```
{'DS Clusters': 10, 'DS Points': 26175, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 423, 'SUM': 26600}
```



```
{'DS Clusters': 10, 'DS Points': 13620, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 180, 'SUM': 13800}
{'DS Clusters': 10, 'DS Points': 13818, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 182, 'SUM': 14000}
{'DS Clusters': 10, 'DS Points': 14014, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 186, 'SUM': 14200}
{'DS Clusters': 10, 'DS Points': 14213, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 187, 'SUM': 14400}
{'DS Clusters': 10, 'DS Points': 14412, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 188, 'SUM': 14600}
{'DS Clusters': 10, 'DS Points': 14611, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 189, 'SUM': 14800}
{'DS Clusters': 10, 'DS Points': 14811, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 189, 'SUM': 15000}
{'DS Clusters': 10, 'DS Points': 15011, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 189, 'SUM': 15200}
{'DS Clusters': 10, 'DS Points': 15210, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 190, 'SUM': 15400}
{'DS Clusters': 10, 'DS Points': 15408, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 192, 'SUM': 15600}
{'DS Clusters': 10, 'DS Points': 15606, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 194, 'SUM': 15800}
{'DS Clusters': 10, 'DS Points': 15806, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 194, 'SUM': 16000}
{'DS Clusters': 10, 'DS Points': 16005, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 195, 'SUM': 16200}
{'DS Clusters': 10, 'DS Points': 16204, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 196, 'SUM': 16400}
{'DS Clusters': 10, 'DS Points': 16404, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 196, 'SUM': 16600}
{'DS Clusters': 10, 'DS Points': 16602, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 198, 'SUM': 16800}
{'DS Clusters': 10, 'DS Points': 16800, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 200, 'SUM': 17000}
{'DS Clusters': 10, 'DS Points': 16996, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 204, 'SUM': 17200}
{'DS Clusters': 10, 'DS Points': 17194, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 206, 'SUM': 17400}
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-115-4451ed035fb6> in <module>
     18         new_cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
     19       else:
---> 20         cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
     21       if len(new_cs_clusters):
     22         cs_clusters = merge_clusters(cs_clusters, new_cs_clusters, threshold, return_two=False)

IndexError: too many indices for array: array is 1-dimensional, but 2 were indexed
```

```
ds_clusters, cs_clusters = merge_clusters(ds_clusters, cs_clusters, threshold, return_two=True)
stats.append(statistics(ds_clusters, cs_clusters, rs_points))
predictions = predict(K, ds_clusters, tr_splits, threshold)
```

```
{'DS Clusters': 10, 'DS Points': 17388, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 212, 'SUM': 17600}
```



```
    rs_idx, cs_idx = process_cluster(group_cluster(km.labels_, rs_points))
    if len(cs_idx) > 0:
      if len(cs_clusters) > 0:
        new_cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
      else:
        cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
      if len(new_cs_clusters):
        cs_clusters = merge_clusters(cs_clusters, new_cs_clusters, threshold, return_two=False)
    if len(rs_idx) > 0:
      rs_points = rs_points[rs_idx]
  if chunk < (K - 1):
    stats.append(statistics(ds_clusters, cs_clusters, rs_points))
```

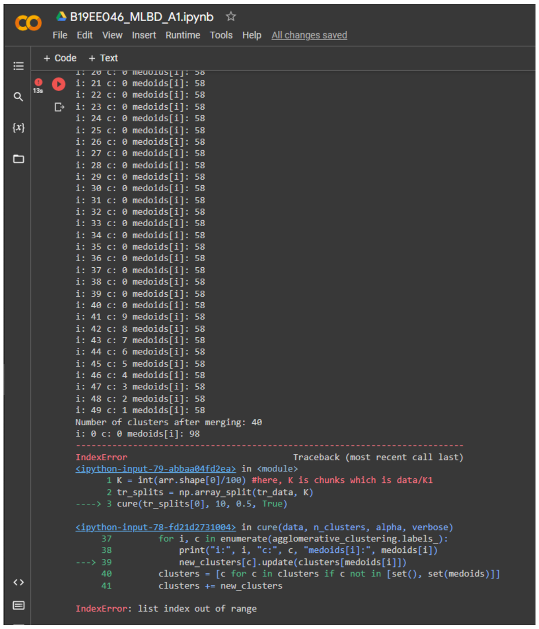
```
{'DS Clusters': 10, 'DS Points': 990, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 10, 'SUM': 1000}
{'DS Clusters': 10, 'DS Points': 1484, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 16, 'SUM': 1500}
{'DS Clusters': 10, 'DS Points': 1975, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 25, 'SUM': 2000}
{'DS Clusters': 10, 'DS Points': 2471, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 29, 'SUM': 2500}
{'DS Clusters': 10, 'DS Points': 2956, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 44, 'SUM': 3000}
{'DS Clusters': 10, 'DS Points': 3447, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 53, 'SUM': 3500}
{'DS Clusters': 10, 'DS Points': 3941, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 59, 'SUM': 4000}
{'DS Clusters': 10, 'DS Points': 4434, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 66, 'SUM': 4500}
{'DS Clusters': 10, 'DS Points': 4930, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 70, 'SUM': 5000}
{'DS Clusters': 10, 'DS Points': 5425, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 75, 'SUM': 5500}
{'DS Clusters': 10, 'DS Points': 5920, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 80, 'SUM': 6000}
{'DS Clusters': 10, 'DS Points': 6415, 'CS Clusters': 1, 'CS Points': 2, 'RS Points': 83, 'SUM': 6500}
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-118-4451ed035fb6> in <module>
     16     if len(cs_idx) > 0:
     17       if len(cs_clusters) > 0:
---> 18         new_cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])
     19       else:
     20         cs_clusters = summarise_info_clusters(km.labels_[tuple(cs_idx)], rs_points[tuple(cs_idx)])

IndexError: too many indices for array: array is 1-dimensional, but 2 were indexed
```

```
ds_clusters, cs_clusters = merge_clusters(ds_clusters, cs_clusters, threshold, return_two=True)
stats.append(statistics(ds_clusters, cs_clusters, rs_points))
predictions = predict(K, ds_clusters, tr_splits, threshold)
```

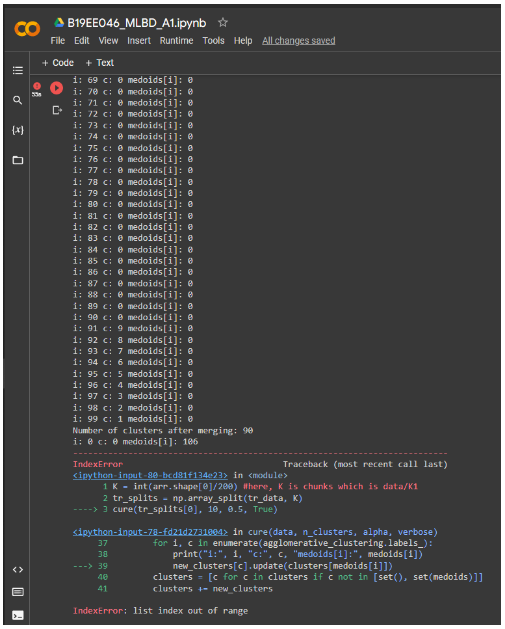
```
{'DS Clusters': 10, 'DS Points': 6914, 'CS Clusters': 0, 'CS Points': 0, 'RS Points': 86, 'SUM': 7000}
```

The above 2 was for K1=100, K1=200, while the middle one is for K1=500.

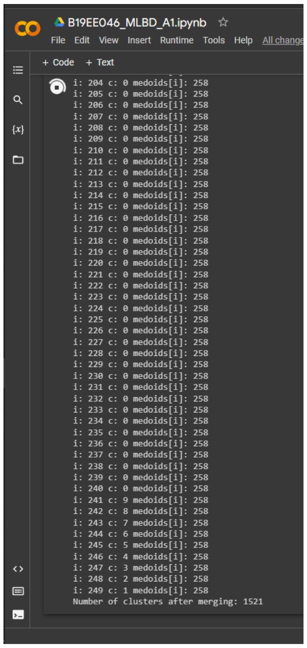
2. CURE with Agglomerative Clustering:-



```
i: 20 c: 0 medoids[i]: 58
i: 21 c: 0 medoids[i]: 58
i: 22 c: 0 medoids[i]: 58
i: 23 c: 0 medoids[i]: 58
i: 24 c: 0 medoids[i]: 58
i: 25 c: 0 medoids[i]: 58
i: 26 c: 0 medoids[i]: 58
i: 27 c: 0 medoids[i]: 58
i: 28 c: 0 medoids[i]: 58
i: 29 c: 0 medoids[i]: 58
i: 30 c: 0 medoids[i]: 58
i: 31 c: 0 medoids[i]: 58
i: 32 c: 0 medoids[i]: 58
i: 33 c: 0 medoids[i]: 58
i: 34 c: 0 medoids[i]: 58
i: 35 c: 0 medoids[i]: 58
i: 36 c: 0 medoids[i]: 58
i: 37 c: 0 medoids[i]: 58
i: 38 c: 0 medoids[i]: 58
i: 39 c: 0 medoids[i]: 58
i: 40 c: 0 medoids[i]: 58
i: 41 c: 9 medoids[i]: 58
i: 42 c: 8 medoids[i]: 58
i: 43 c: 7 medoids[i]: 58
i: 44 c: 6 medoids[i]: 58
i: 45 c: 5 medoids[i]: 58
i: 46 c: 4 medoids[i]: 58
i: 47 c: 3 medoids[i]: 58
i: 48 c: 2 medoids[i]: 58
i: 49 c: 1 medoids[i]: 58
Number of clusters after merging: 40
i: 0 c: 0 medoids[i]: 98
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-79-abbaa04fd2ea> in <module>
      1 K = int(arr.shape[0]/100) #here, K is chunks which is data/K1
      2 tr_splits = np.array_split(tr_data, K)
----> 3 cure(tr_splits[0], 10, 0.5, True)

<ipython-input-78-fd21d2731004> in cure(data, n_clusters, alpha, verbose)
     37         for i, c in enumerate(agglomerative_clustering.labels_):
     38             print("i:", i, "c:", c, "medoids[i]:", medoids[i])
---> 39             new_clusters[c].update(clusters[medoids[i]])
     40         clusters = [c for c in clusters if c not in [set(), set(medoids)]]
     41         clusters += new_clusters

IndexError: list index out of range
```



```
i: 69 c: 0 medoids[i]: 0
i: 70 c: 0 medoids[i]: 0
i: 71 c: 0 medoids[i]: 0
i: 72 c: 0 medoids[i]: 0
i: 73 c: 0 medoids[i]: 0
i: 74 c: 0 medoids[i]: 0
i: 75 c: 0 medoids[i]: 0
i: 76 c: 0 medoids[i]: 0
i: 77 c: 0 medoids[i]: 0
i: 78 c: 0 medoids[i]: 0
i: 79 c: 0 medoids[i]: 0
i: 80 c: 0 medoids[i]: 0
i: 81 c: 0 medoids[i]: 0
i: 82 c: 0 medoids[i]: 0
i: 83 c: 0 medoids[i]: 0
i: 84 c: 0 medoids[i]: 0
i: 85 c: 0 medoids[i]: 0
i: 86 c: 0 medoids[i]: 0
i: 87 c: 0 medoids[i]: 0
i: 88 c: 0 medoids[i]: 0
i: 89 c: 0 medoids[i]: 0
i: 90 c: 0 medoids[i]: 0
i: 91 c: 9 medoids[i]: 0
i: 92 c: 8 medoids[i]: 0
i: 93 c: 7 medoids[i]: 0
i: 94 c: 6 medoids[i]: 0
i: 95 c: 5 medoids[i]: 0
i: 96 c: 4 medoids[i]: 0
i: 97 c: 3 medoids[i]: 0
i: 98 c: 2 medoids[i]: 0
i: 99 c: 1 medoids[i]: 0
Number of clusters after merging: 90
i: 0 c: 0 medoids[i]: 106
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-80-bcd81f134e23> in <module>
      1 K = int(arr.shape[0]/200) #here, K is chunks which is data/K1
      2 tr_splits = np.array_split(tr_data, K)
----> 3 cure(tr_splits[0], 10, 0.5, True)

<ipython-input-78-fd21d2731004> in cure(data, n_clusters, alpha, verbose)
     37         for i, c in enumerate(agglomerative_clustering.labels_):
     38             print("i:", i, "c:", c, "medoids[i]:", medoids[i])
---> 39             new_clusters[c].update(clusters[medoids[i]])
     40         clusters = [c for c in clusters if c not in [set(), set(medoids)]]
     41         clusters += new_clusters

IndexError: list index out of range
```



```
i: 204 c: 0 medoids[i]: 258
i: 205 c: 0 medoids[i]: 258
i: 206 c: 0 medoids[i]: 258
i: 207 c: 0 medoids[i]: 258
i: 208 c: 0 medoids[i]: 258
i: 209 c: 0 medoids[i]: 258
i: 210 c: 0 medoids[i]: 258
i: 211 c: 0 medoids[i]: 258
i: 212 c: 0 medoids[i]: 258
i: 213 c: 0 medoids[i]: 258
i: 214 c: 0 medoids[i]: 258
i: 215 c: 0 medoids[i]: 258
i: 216 c: 0 medoids[i]: 258
i: 217 c: 0 medoids[i]: 258
i: 218 c: 0 medoids[i]: 258
i: 219 c: 0 medoids[i]: 258
i: 220 c: 0 medoids[i]: 258
i: 221 c: 0 medoids[i]: 258
i: 222 c: 0 medoids[i]: 258
i: 223 c: 0 medoids[i]: 258
i: 224 c: 0 medoids[i]: 258
i: 225 c: 0 medoids[i]: 258
i: 226 c: 0 medoids[i]: 258
i: 227 c: 0 medoids[i]: 258
i: 228 c: 0 medoids[i]: 258
i: 229 c: 0 medoids[i]: 258
i: 230 c: 0 medoids[i]: 258
i: 231 c: 0 medoids[i]: 258
i: 232 c: 0 medoids[i]: 258
i: 233 c: 0 medoids[i]: 258
i: 234 c: 0 medoids[i]: 258
i: 235 c: 0 medoids[i]: 258
i: 236 c: 0 medoids[i]: 258
i: 237 c: 0 medoids[i]: 258
i: 238 c: 0 medoids[i]: 258
i: 239 c: 0 medoids[i]: 258
i: 240 c: 0 medoids[i]: 258
i: 241 c: 9 medoids[i]: 258
i: 242 c: 8 medoids[i]: 258
i: 243 c: 7 medoids[i]: 258
i: 244 c: 6 medoids[i]: 258
i: 245 c: 5 medoids[i]: 258
i: 246 c: 4 medoids[i]: 258
i: 247 c: 3 medoids[i]: 258
i: 248 c: 2 medoids[i]: 258
i: 249 c: 1 medoids[i]: 258
Number of clusters after merging: 1521
```

The above 2 was for K1=100, K1=200, while the middle one is for K1=500. The middle one was on run during the submission.

Due to recurring bugs, I wasn't able to solve them timely, but following the logic of the course slides, I was able to extensively understand the core components of the program chunks. That's why I won't be able to present the observations and maybe the last part too, since I also tried to run these algorithms 5 times, the initiations of the clusters were different each time, but bugs being the same.