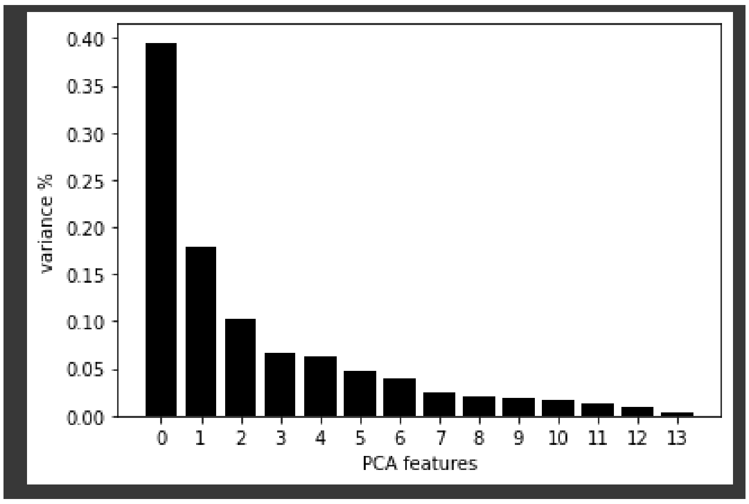
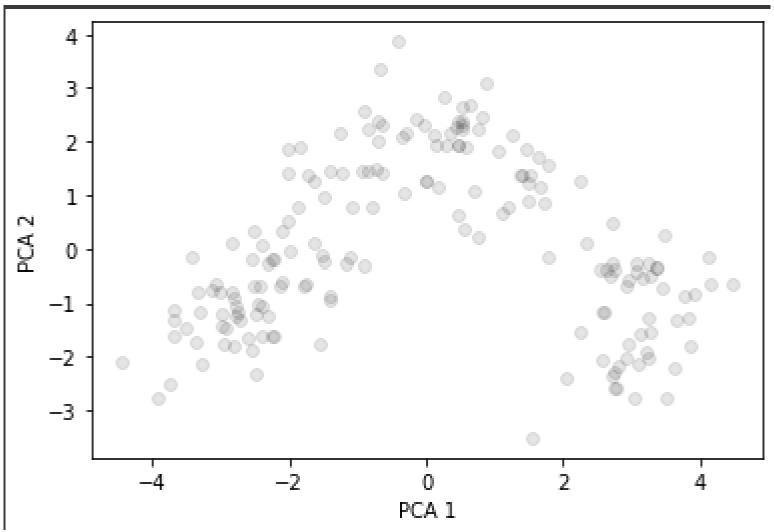
Lab 9
Abu Shahid
B20CS003

<u>Question 1</u>

Subpart a: Dimension Reduction and Visualization

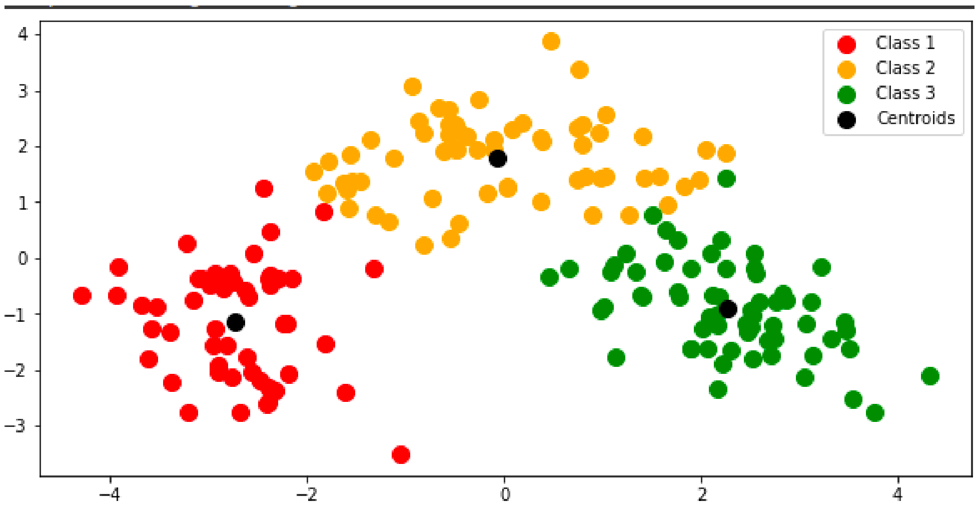
- Data is loaded and processed. No features values were missing and data was complete.
- Data was normalized using StandardScaler(). Since K-Means is a distance based algorithm, in order to insure that distance is not biased towards a particular feature, it is advised to scale the data before feeding in training.
- Inorder to select an appropriate value of **k**, PCA was implemented on our dataset and drop in variance-retention across features was recorded.





- By looking at the plotted result, it is advisable to keep number of clusters=3 or if we want more clusters, then it should not exceed 5.
- These three features preserve maximum variance in our data and after that, other features hardly contribute. Even looking at the plot, we can roughly see 3 clustering of close vicinity points.

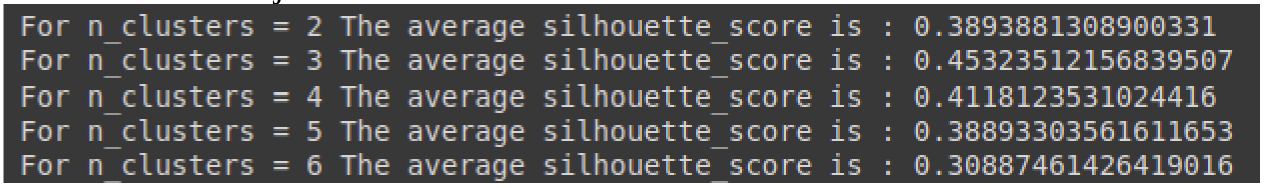
Subpart b: Implementing K-Means



- Next we implemented K_Means using inbuilt library, with n_clusters=3 and max_iter=100. The clusters for the same is visualized along with the centroids in the previous page.

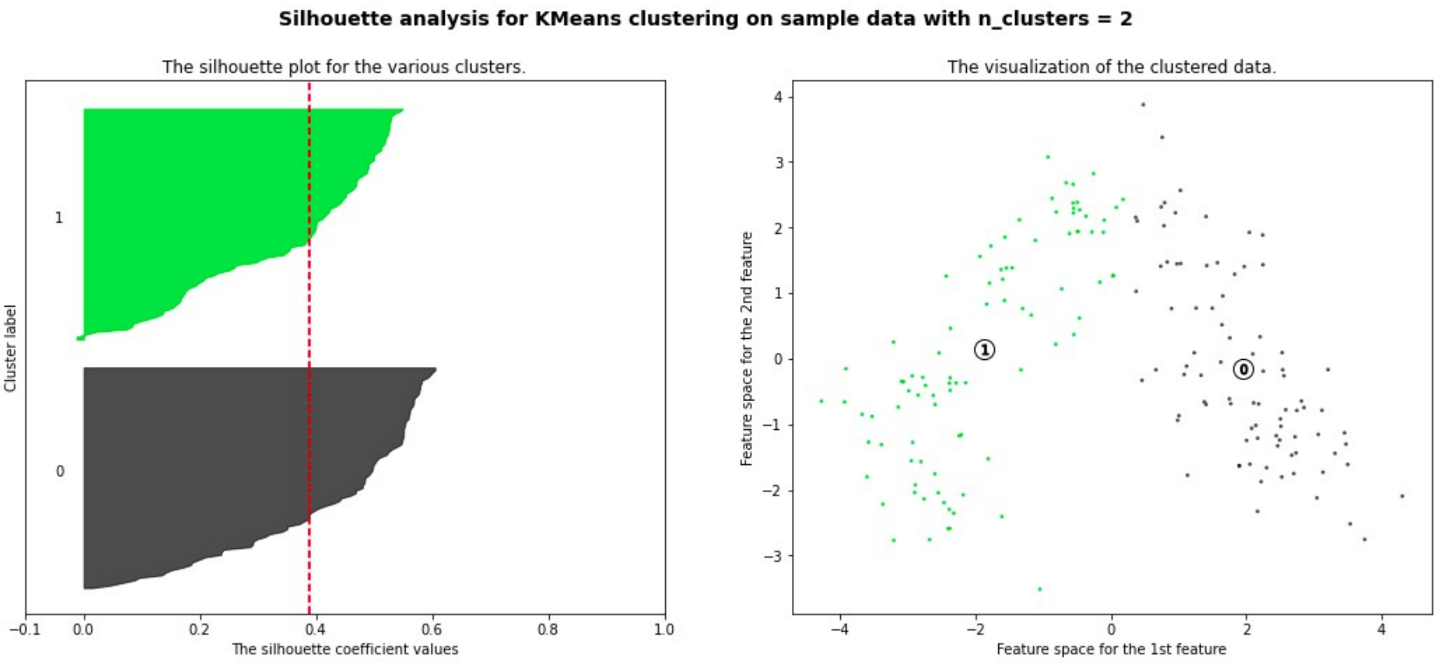
Subpart C: Silhouette Score Analysis

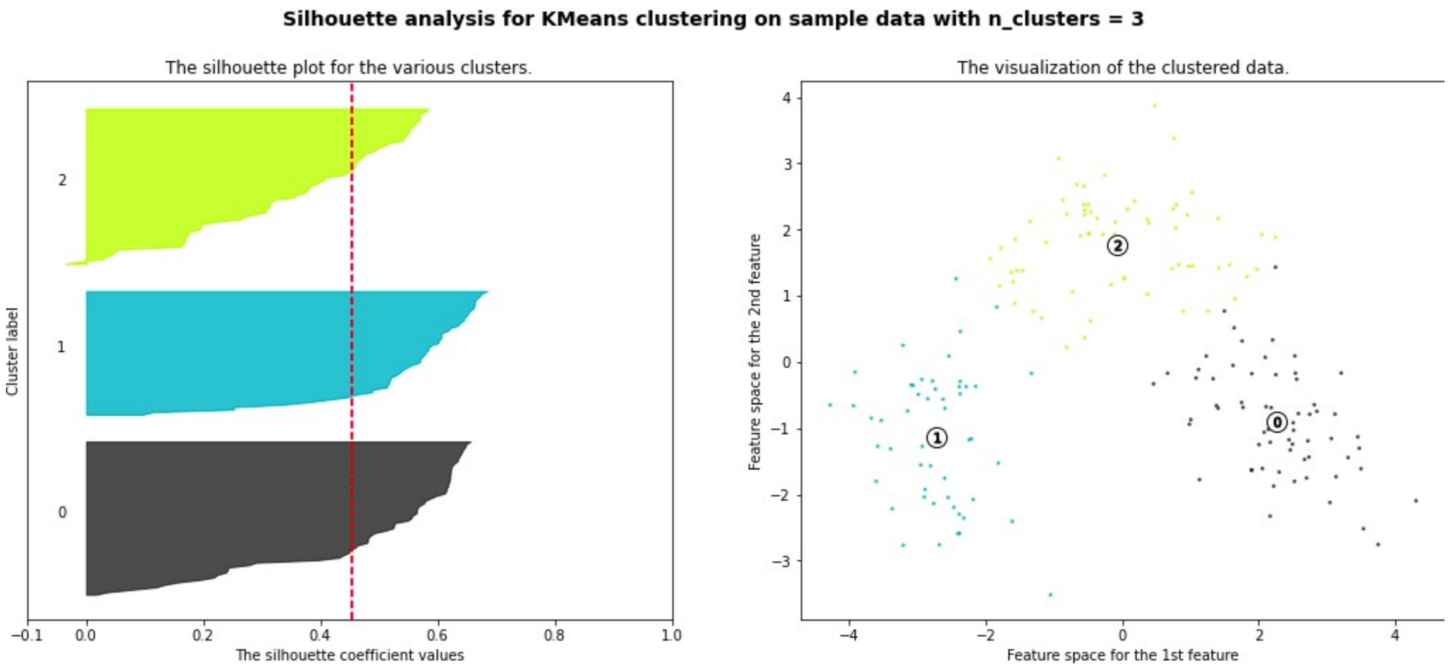
- Silhouette scores refers to interpretation and validation of consistency within the clusters. The technique provides a succinct graphical representation of how well each object has been classified.

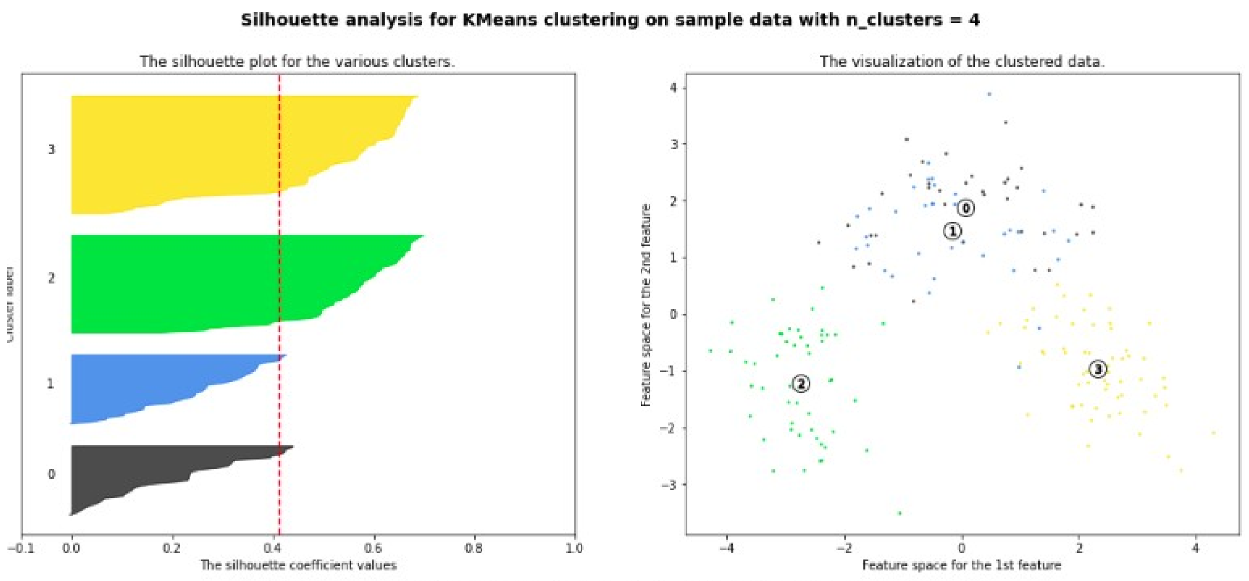


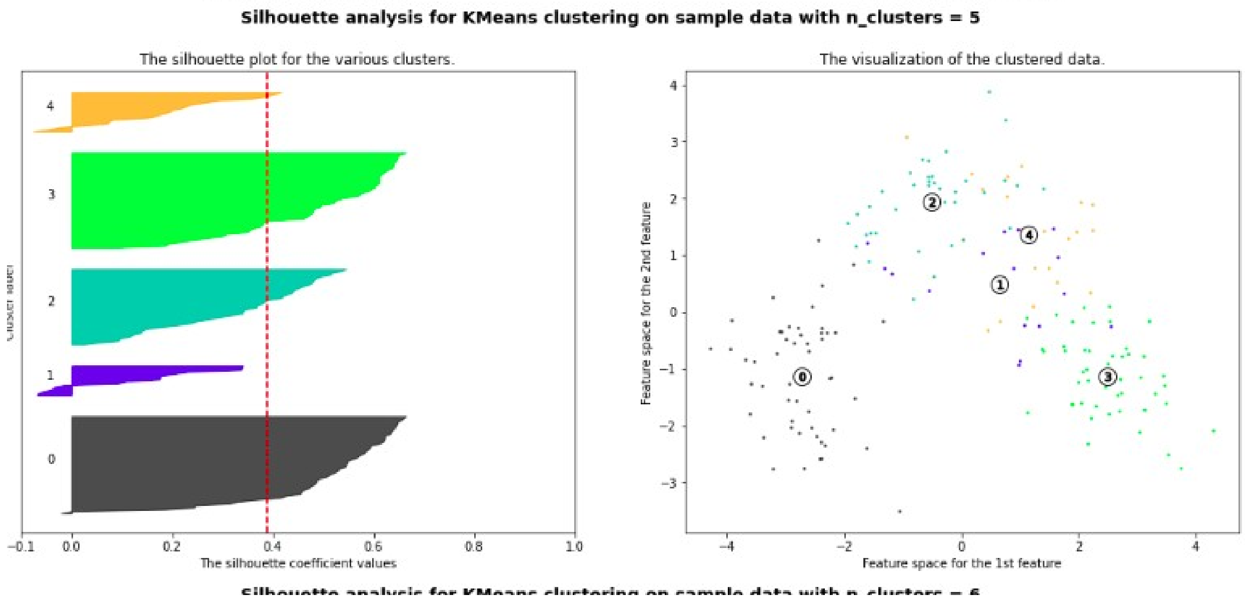
```
For n_clusters = 2 The average silhouette_score is : 0.3893881308900331
For n_clusters = 3 The average silhouette_score is : 0.45323512156839507
For n_clusters = 4 The average silhouette_score is : 0.4118123531024416
For n_clusters = 5 The average silhouette_score is : 0.38893303561611653
For n_clusters = 6 The average silhouette_score is : 0.30887461426419016
```

Further,



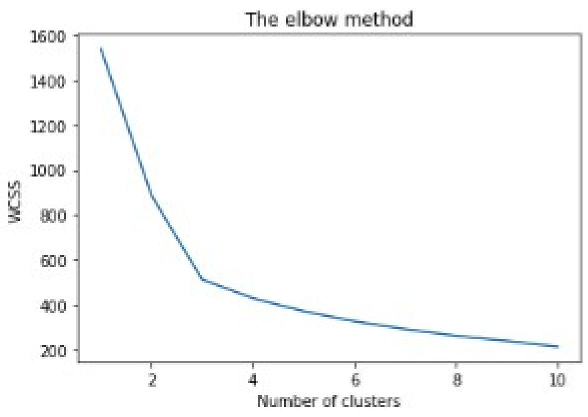






- k=3 seems to be the best choice for the number of clusters as
  - gives highest average silhouette score
  - individual clusters cross 0.4 silhouette coefficient threshold
  - the clsuters are of approximately same size.
    - Same cannot be said for k=2, 4, 5 and 6; which were other promising candidates for number of clusters

Subpart d: Elbow method to find k



- To determine optimal value of clusters, we select value of k at the elbow, that is the point after which inertia starts decreasing in a linear fashion. Here it is k=3 which further validates our exercise done previously.

<u>Question 2</u>

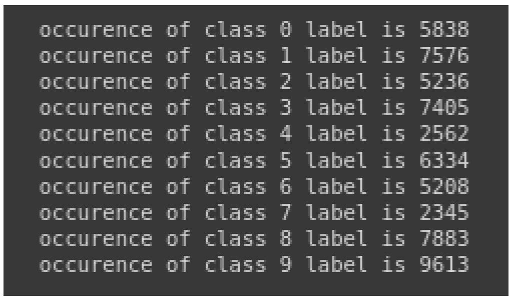
Subpart a and b: Implementing k-means from scratch

- Class **Kmeans** was created
- Kmeans(n_clusters, max_iter, c_provided, user_centroids)
  - n_clusters: number of clusters
  - max_iter: maximum number of iterations; default=100
  - c_provided: marks if centroid is provided by the user of not; default **False**
  - user_centroid: centroid coordiantes provided by the user

- Member methods:
  - __init__: constructor
  - initializ_centroids: to initialze centroids randomly
  - compute_centroids: given data along with its labels, this functions calculates centroids of all the clusters
  - compute_distance: to compute distance between a labelled datapoint and its cluster centroid
  - find_closest_cluster: given centroids of cluster, assigns a data point to a cluster and updates its label
  - compute_sse: computes sum of squared error for all the instances in our input data
  - fit()
  - and, predict()

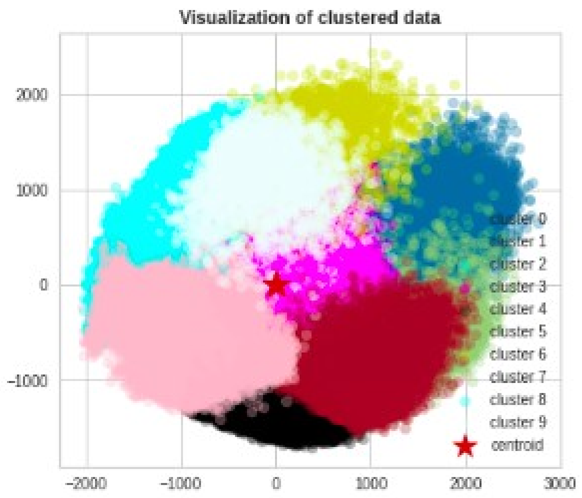
Subpart c: Train K_means with k=10 and random initailization for centroids

- A model with said params was trained.
- Occurance of each label can be summarized as:

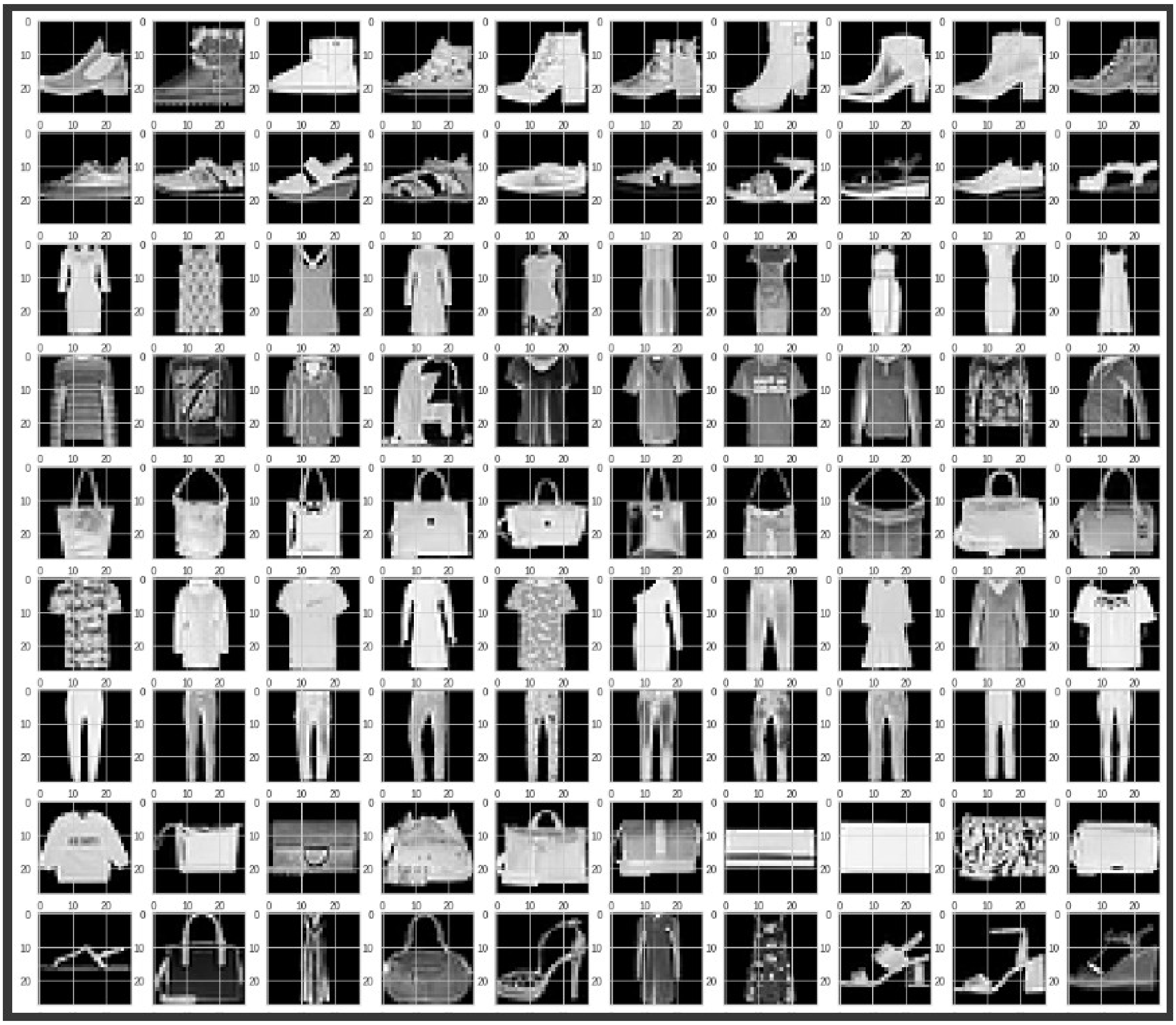


```
occurence of class 0 label is 5838
occurence of class 1 label is 7576
occurence of class 2 label is 5236
occurence of class 3 label is 7405
occurence of class 4 label is 2562
occurence of class 5 label is 6334
occurence of class 6 label is 5208
occurence of class 7 label is 2345
occurence of class 8 label is 7883
occurence of class 9 label is 9613
```

Subpart d: Visualization of cluster centre of each cluster for all the clusters



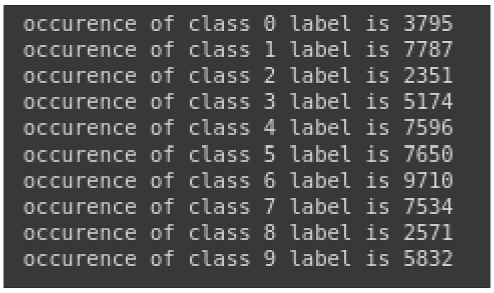
Subpart e: Visualization of 10 images from each cluster.



Next we repeat subpart c and e, but this time instead of randomly initializing centroids; we use 10 images from each class for initialization

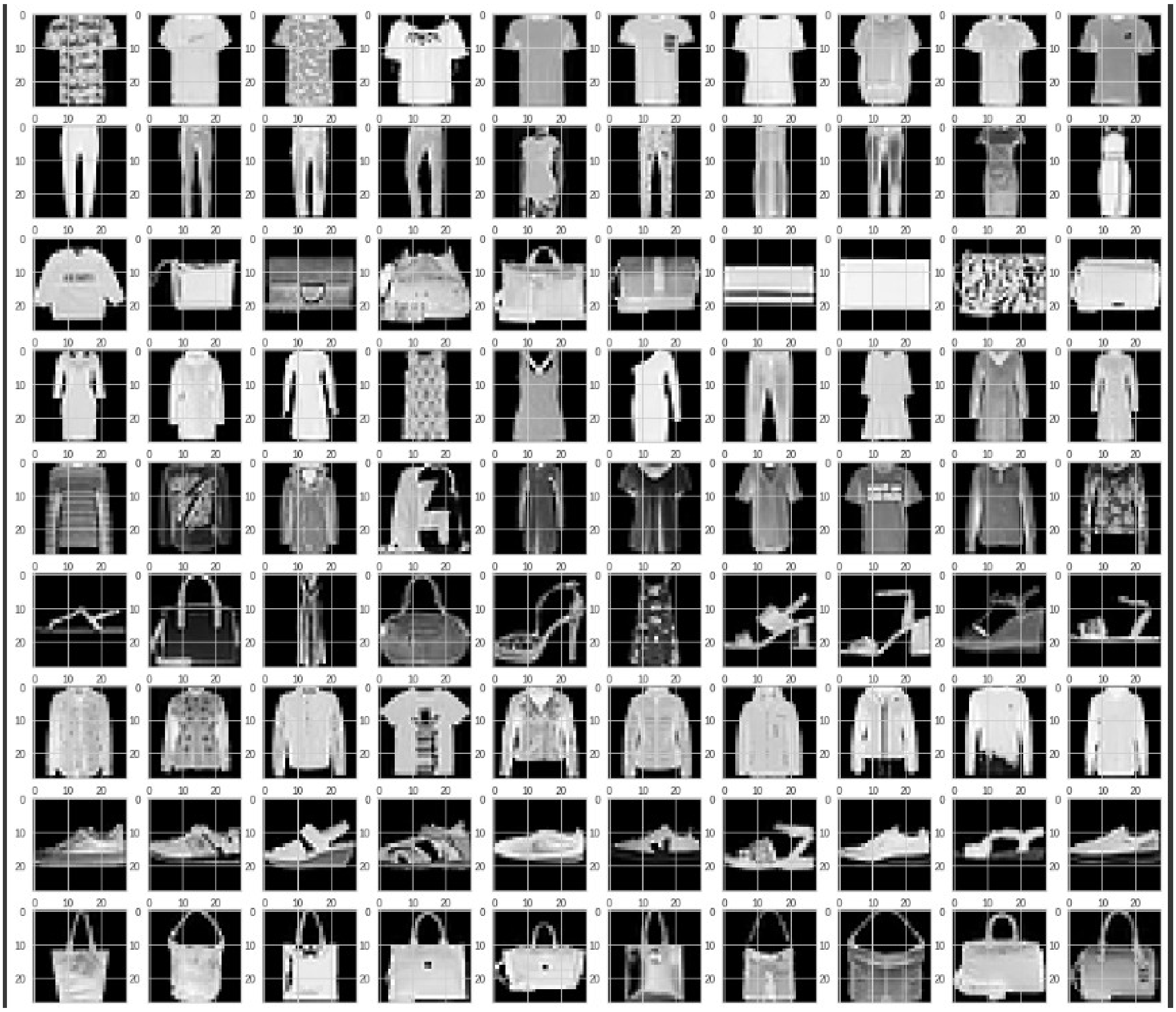
Subpart f: Custom initialization

- A model with said params was trained.
- Occurance of each label can be summaried as:



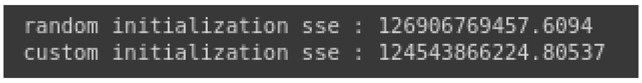
```
occurence of class 0 label is 3795
occurence of class 1 label is 7787
occurence of class 2 label is 2351
occurence of class 3 label is 5174
occurence of class 4 label is 7596
occurence of class 5 label is 7650
occurence of class 6 label is 9710
occurence of class 7 label is 7534
occurence of class 8 label is 2571
occurence of class 9 label is 5832
```

Subpart g: Visualization of 10 images from each cluster



Subpart h: Evaluation and reporting

- Performances of the 2 models can be summarized using the SSE score:



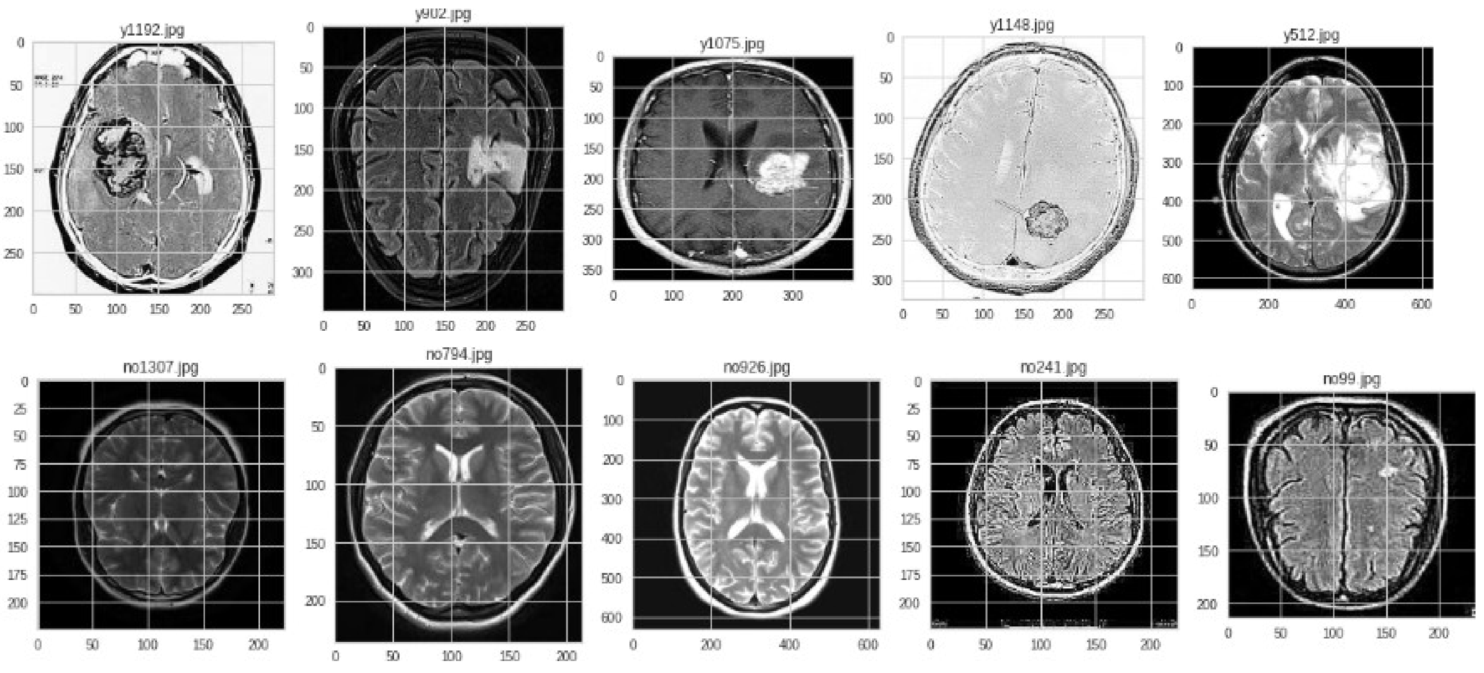
```
random initialization sse : 126906769457.6094
custom initialization sse : 124543866224.80537
```

- As expected, initialization will images from all the labels gave a better result.
- However there is no way of knowing how good this score is or if it is at all good to begin with.
- It could also mean that both the models converged really fast (in less that 100 iterations) and therefore are giving somewhat similar score.
- Keeping max_iter value low, we can expect the second initialization techinque to perform better in most of the cases, except when the initializations happen via outlier datapoint.

Question 3

Subpart a and b: Normalizing our dataset and visualizing it

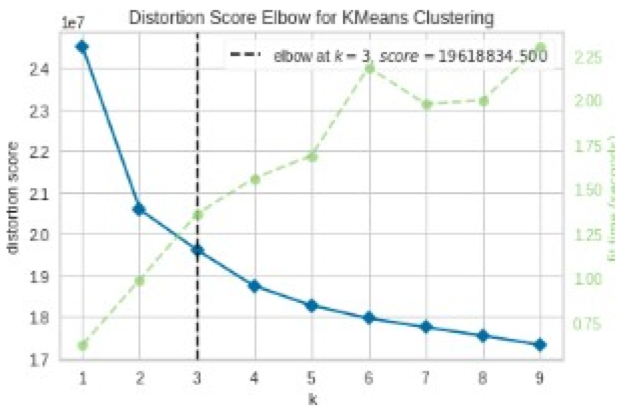
- Data was loaded and necessary pre-processing was performed.



- Image in our dataset had varying length and breadth and therefore all were standardized and resized to 400x400
- Further image provided was in RGB (3 channels) and was converted to Grayscale.
- As grayscale feature values vary from 0 to 255, this was linearly scaled to 0 to 1.

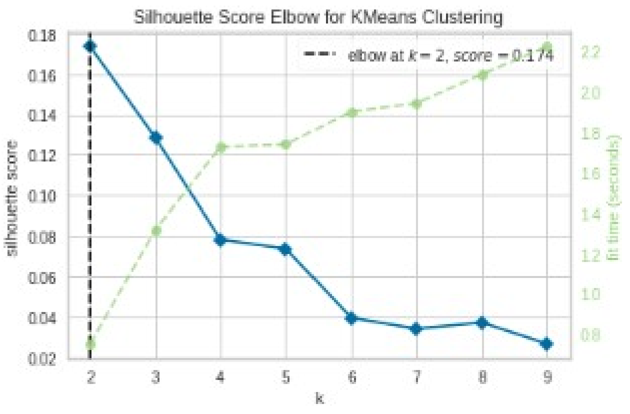
Subpart b and d: Dimension Reduction and finding number of Communities

- As we now have 400x400= 160,000; our dataset has become enormous.
- To better deal with it, PCA was implemented preserving 95 percent of the variance. This yielded 537 features which is good enough.
- Next to find number of communities,
  - **Elbow Method**



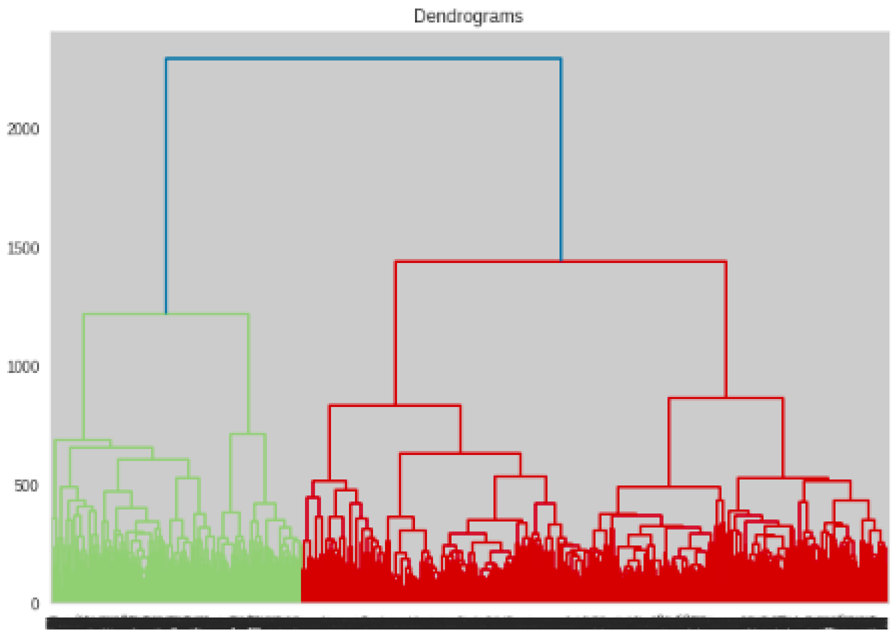
Although we got an elbow at k=3, stagnation in inertia was not satisfactory and Silhouette score analysis was done.

- **Silhouette Score**



With this we got an optimal value of k=2 which unlike elbow method is not only satisfactory but also better founded.

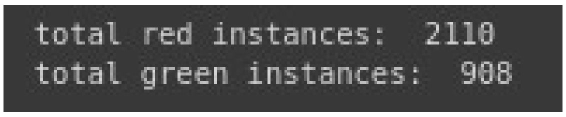
- **Dendogram**
- Next we applied Agglomerative Clustering using sklearn and plotted a dendogram.



Even dendogram gave k=2 communities.

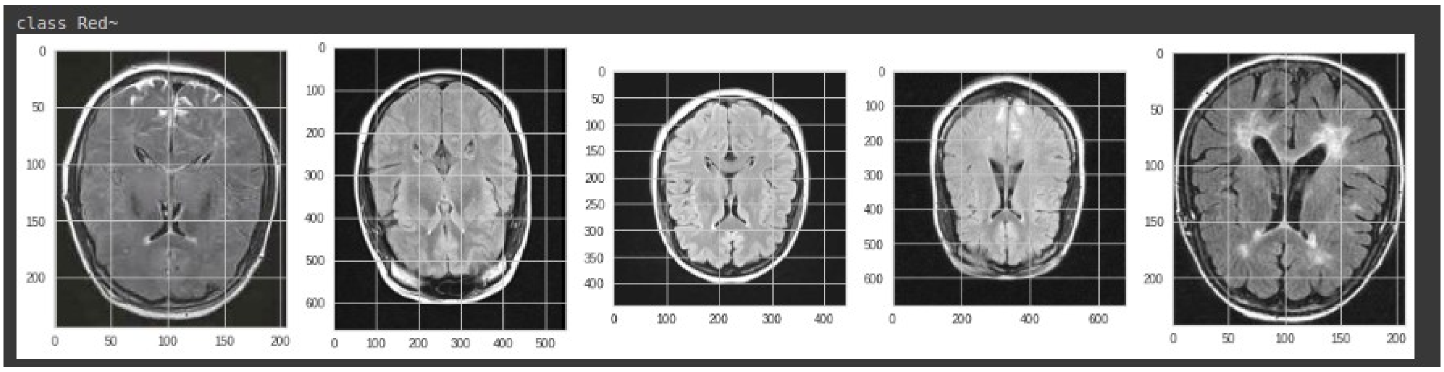
Subpart C: Visualizing communities

- Having gotten our clusters, next we visualize them.

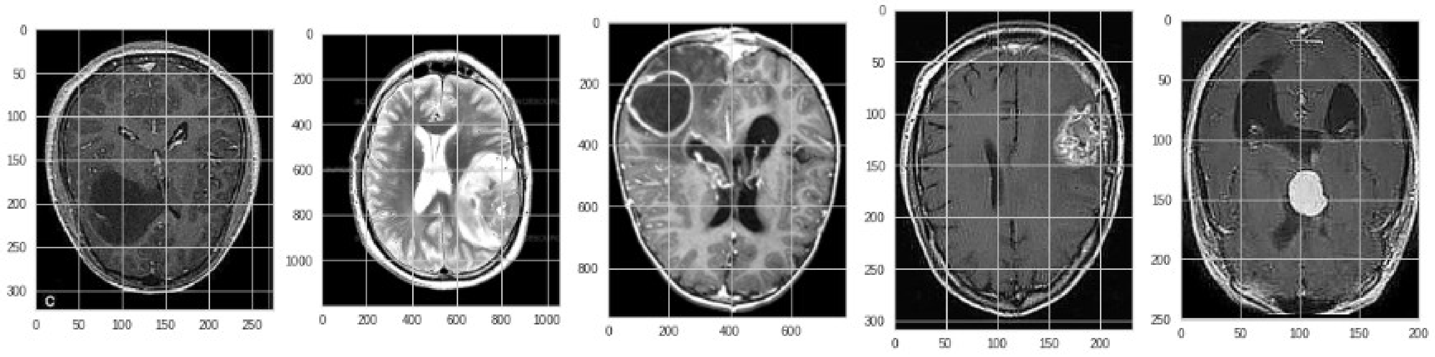


```
total red instances:  2110
total green instances:  908
```

- This shows a rather skewed clustering, considering both the classes should be having an equal representation, citing higher misclassification (or perhaps classification based on some hidden ‘irrelavent’ feature)



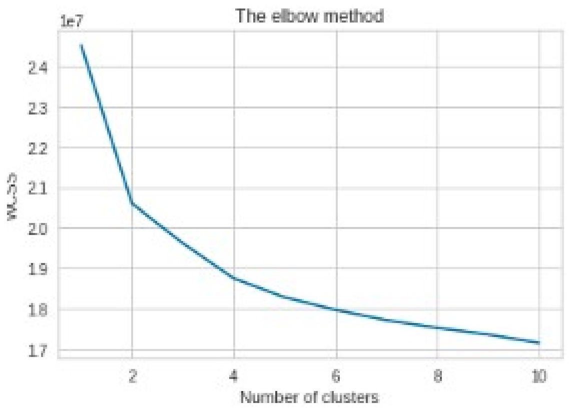
Class red



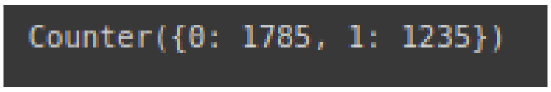
Class green

Subpart e: Applying k-means and drawing comparisons

- To find optimal value of clusters, we used elbow method



- Clearly optimal value of k=2
- Next we apply kmeans using sklearn and the number of instances in each clusters are:



```
Counter({0: 1785, 1: 1235})
```

,which although is not balanced as it should be; is still better than what Agglomerative Clustering gave us.

- Agglomerative clustering mechanism finds points of data that are closest to each other, and successively groups them together. Agglomerative clustering is hierarchical because it performs operations sequentially. This algorithm is

useful in cases where we want to make decisions about how coarsely or finely we want to group our data, or what resolution we want your data in.
    - Then can perform better in tasks such as clustering people based on skin tones here there will be a fringe from dark to lighter tones.
    - Futher here, we can use it to cluster different severity of cancer based on tumor sizes.
- K-means clustering mechanism is useful when we want to divide your data in the k sets simultaneously. It’s useful when data needs to be compared to each other and is “comparable”.
    - For example, clustering a flower set based on three ‘distinct’ colors it is available in.
    - Here, it can perform better as the answer to our question is **yes** or **no** to where the person has cancer or not.
- Reference