IT350 - Data Analytics

Annam Indhu Lekha

191IT207

1) Find the clusters in the given dataset based on the content similarity and image similarity using k-means clustering and hierarchical clustering methods.

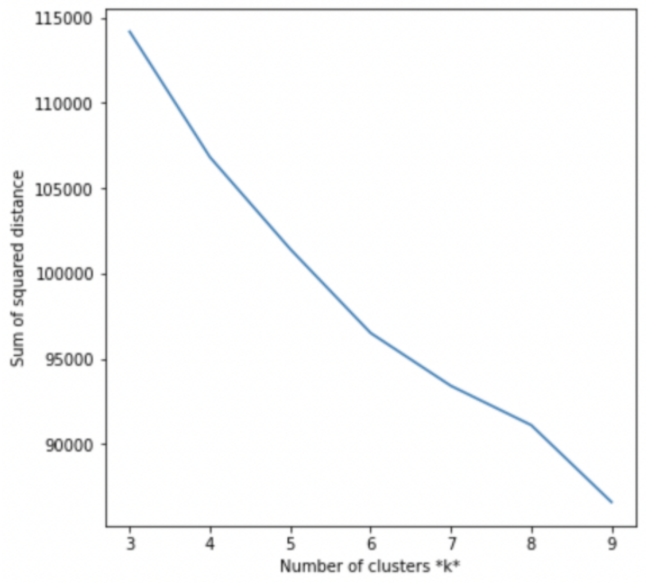
Data preprocessing - feature extraction

We have used the VGG model and removed the output layer manually. The new final layer is a fully-connected layer with 4,096 output nodes. This vector of 4,096 numbers is the feature vector that can be used to cluster the images.

K means clustering

We chose a target number k, which refers to the number of centroids we wanted in the dataset. K means algorithm will allow us to group our feature vectors into k clusters. Each cluster contains images that are visually similar.

```
Text(0, 0.5, 'Sum of squared distance')
```

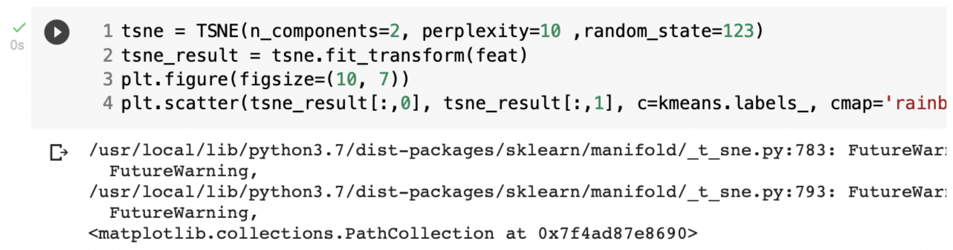


Clusters are chosen by observing the trade of between inertia (sum of squared distance of samples of their closed cluster) and number of clusters.

By increasing the number of clusters, inertia decreases. The optimal number of clusters is achieved by finding the break-even point in plot between number of clusters and inertia.

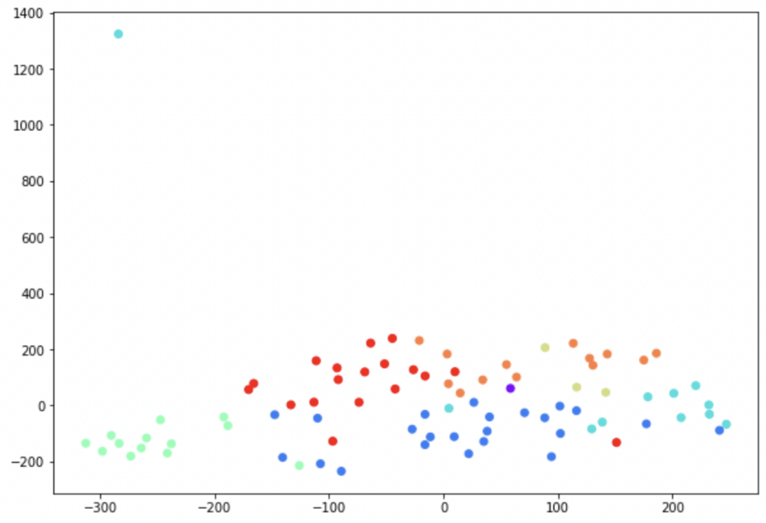
The break-even point is at 6. So, I took K as 6 for clustering.

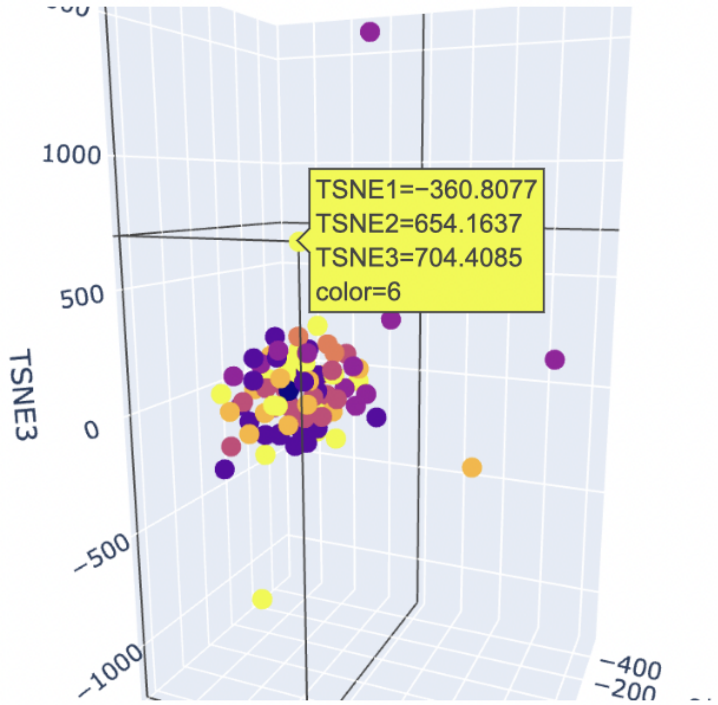
2D visualisation of Kmeans clusters



```
tsne = TSNE(n_components=2, perplexity=10 ,random_state=123)
tsne_result = tsne.fit_transform(feat)
plt.figure(figsize=(10, 7))
plt.scatter(tsne_result[:,0], tsne_result[:,1], c=kmeans.labels_, cmap='rainb
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: FutureWar
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWar
  FutureWarning,
<matplotlib.collections.PathCollection at 0x7f4ad87e8690>
```

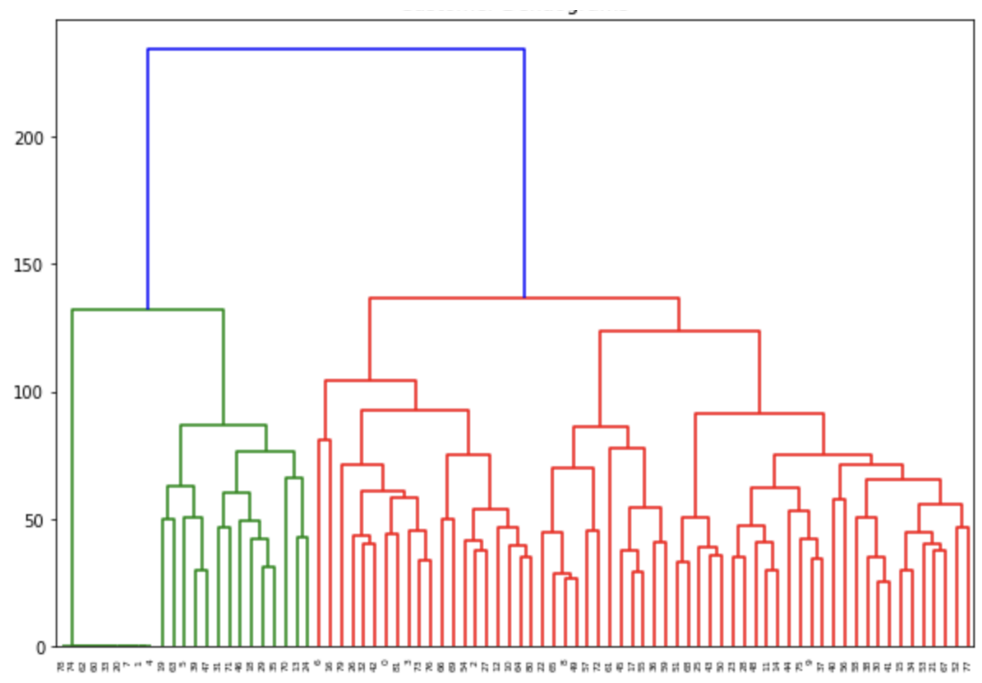




Hierarchical Clustering

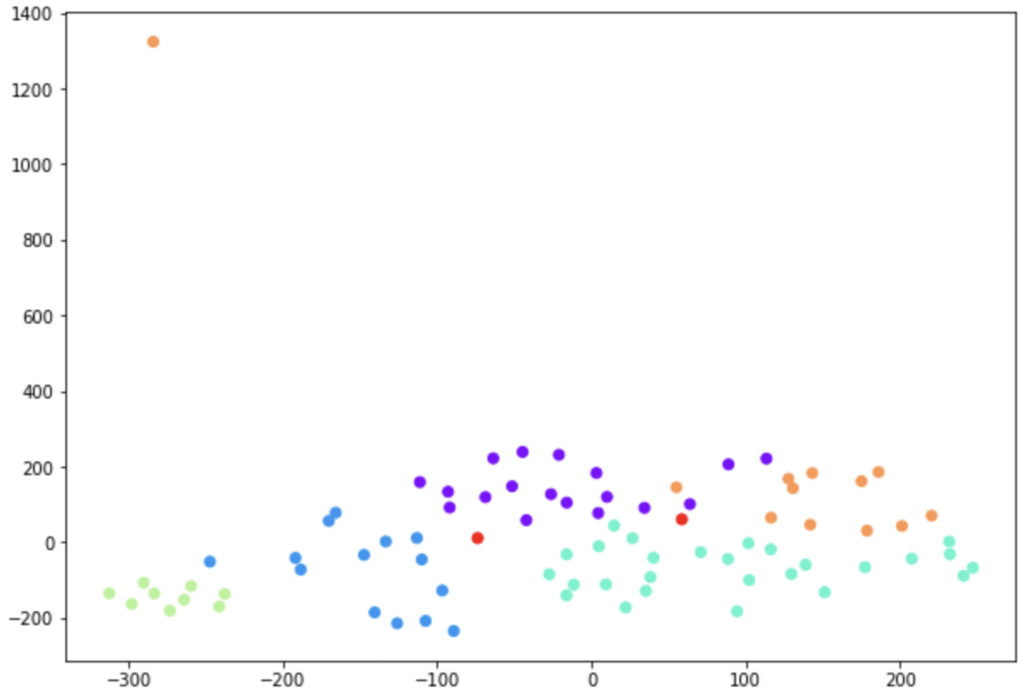
I have used Agglomerative clustering. It is the most common type of hierarchical clustering used to group objects in clusters based on the similarity. It starts by considering each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster. It is a top – down approach.

we slice this structure horizontally. All the resulting child branches formed below the horizontal cut represent an individual cluster at the highest level in your system and it defines the associated cluster membership for each data sample.

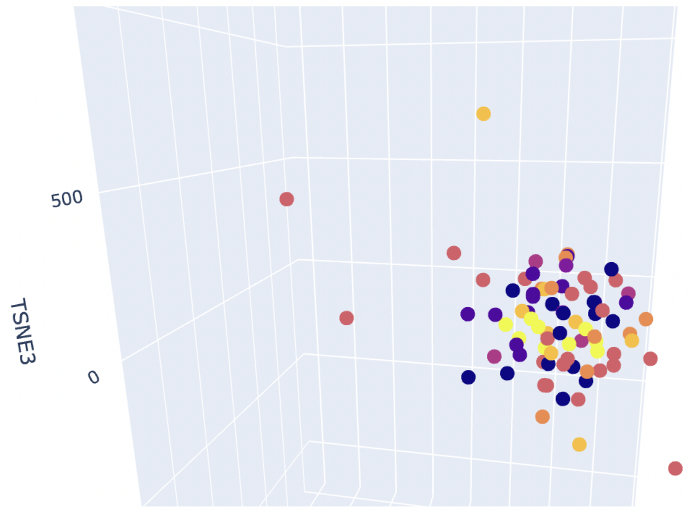


2 D visualization

```
<matplotlib.collections.PathCollection at 0x7f4ad1e1e610>
```



3 - D visualization



2) Evaluation

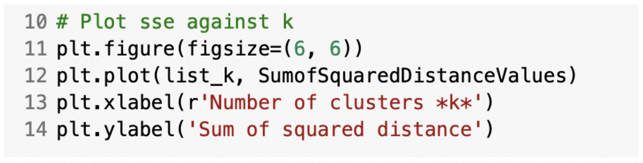
3) Evaluate the clusters that are obtained using appropriate methods.

Clusters obtained by K-Means are justified by using the sum of squared distance from the nearest cluster. When K value is 6, this error is less, and it is a breakpoint from the plot.

In Clusters obtained by a hierarchical method, I had chosen the line in such a way that this horizontal line passes through the longest distance without a horizontal line. From this point 8 clusters were appropriate to use.

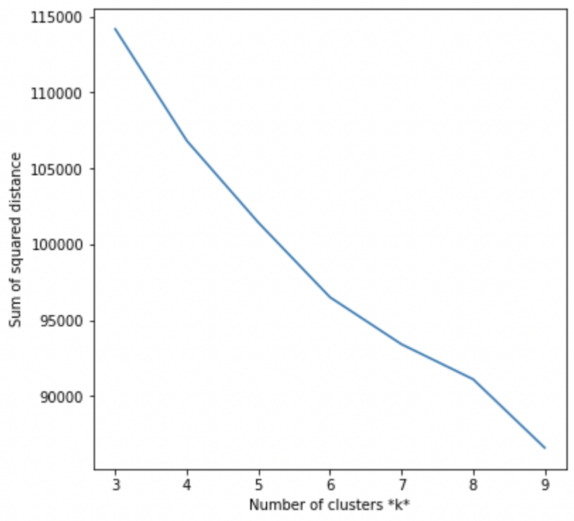
Here are some of the methods for evaluating clusters:

- **Cross Validation-** divides the data into n parts. Trains the model on n-1 parts and validates the model on remaining part
- Getting the optimal number of clusters is incredibly significant in the analysis. If *k* is too high, each point will broadly start representing a cluster and if *k* is too low, then data points are incorrectly clustered. Finding the optimal number of clusters leads to granularity in clustering.
  Elbow Method - This method calculates the best k value by considering the percentage of variance explained by each cluster

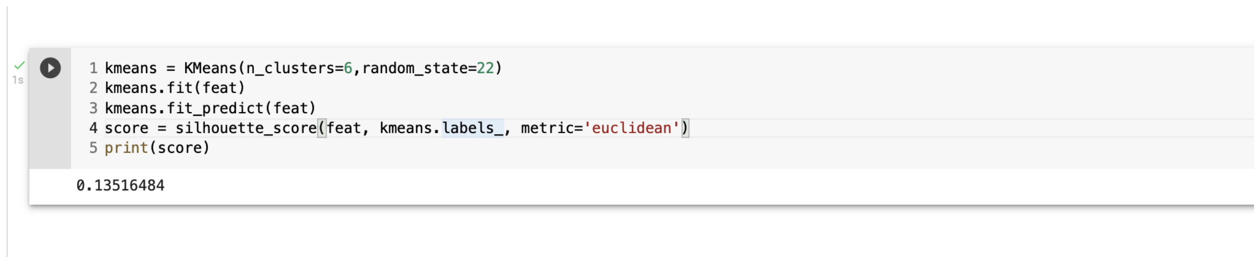


```
# Plot sse against k
plt.figure(figsize=(6, 6))
plt.plot(list_k, SumofSquaredDistanceValues)
plt.xlabel(r'Number of clusters *k*')
plt.ylabel('Sum of squared distance')
```

```
Text(0, 0.5, 'Sum of squared distance')
```



- **Silhouette Method -** it returns a value between -1 and 1 based on the similarity of an observation with its own cluster. high value indicates high match, and vice versa. We can use any distance metric to calculate the silhouette score.
  Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.



```
kmeans = KMeans(n_clusters=6,random_state=22)
kmeans.fit(feat)
kmeans.fit_predict(feat)
score = silhouette_score(feat, kmeans.labels_, metric='euclidean')
print(score)
```

```
0.13516484
```

- **Dunn's index:** Dunn's Index is equal to the minimum inter-cluster distance divided by the maximum cluster size. A higher DI implies better clustering. It assumes that better clustering means that clusters are compact and well-separated from other clusters.

Link to colab -
https://colab.research.google.com/drive/1Iydx-fOmtTuME97BFKZNe510a753Fmeg?usp=sharing