HW3: K-Means Clustering

Name: Venkata Pratyush Kodavanti

Miner username: vkodavan

V-Measure Score: 86% (Iris dataset) and 65% (Image dataset).

I've submitted 3 files.

a) 584_assignment_3_iris (Final submission of iris dataset)
b) 584_assignment_3_image_TSNE (Final submission of image dataset)
c) 584_assignment_3_image_PCA

For reading the data I've used standard pandas, pandas.read_csv() function which takes the file name and the separator (space for Iris dataset and ',' comma for Image dataset). The header for both the datasets is None as there is column name defined for them.

Checking for Null Values: After reading the data, I've checked if the dataset contains any null values using the pandas.isnull().sum(). Both the datasets do not contain any null values.

Normalizing the datasets:

Iris Dataset: For the Iris dataset as I do not know the range (lower and upper bound) of the data so I've written a function normalized. It takes one parameter (Dataset).

Once the dataset is received for each value I would subtract the minimum value in that column and divide it with the difference of the maximum and the minimum value in the column. To do this I've use numpy.min and numpy.max to get the minimum and the maximum values respectively. By doing so I make sure that the all my values are in the range of [0, 1].

Image Dataset: For the Image dataset as it is given that the range of values are from [0,255] I did not use the normalize function but divided every value by 255. If I would have used the normalize function in some columns all the values are zero so it would return me a NaN (Not a number or Infinity).

Reducing the Features for the Image Dataset: As the number of features (784) are high, I've used scikit learns PCA from the library sklearn.decomposition **PCA** stands for Principal Component Analysis, which is one of the dimensionality reduction method.

First, we need to standardize the data. As the standardization is already done, then it checks for the covariance by calculating the covariance matrix (here the matrix would be of the shape 784x784). Then it computes the Eigen Vectors and Eigen Values of the Covariance Matrix to identify the Principal Matrix. once the Eigen vectors and values are computed then we know what features important and what features does not change much around our data. There by removing features which does not affect much.

The parameters used in this are: (a) Number of Components (b) Variance

But doing so did not help much with the accuracy.

TSNE: (t-distributed stochastic neighborhood embedding): t-SNE is also a unsupervised non-linear dimensionality reduction and data visualization technique.

The algorithm first calculates the probability of similarity of points in high dimensionality. Then it calculates the probability of similarity of points in the corresponding low-dimension.

It tries to reduce the difference between these similarities in high-dimension and low dimension space for a perfect representation.

I've used tsne from sklearn.manifold which takes in the following parameters

TSNE(n_components=2, verbose=1, perplexity=30, n_iter=300)
N_components: Dimension of the embedded space.

Perplexity: Related number of neighbors used in the manifold algorithm. For this dataset I've tried with 30, 35 and 40. (Currently 30 works best).

N_iter: Maximum number of iterations for the optimization. Should be at least 250. For the sake of convenience, I've used 300.

Implementing the K-Means Clustering Algorithm: The K-Means clustering is broken into 3 parts

1) Initialize random centroid points and then assign the clusters
2) Reassign Compute the centroids and
3) Reassign the clusters if the centroids change

Initialize Random Centroids and Assigning the clusters to the data points: To initial random centroids, I've used np.random_choice to select 3 random indices from the dataset and used them as centroids. After that I've clustered the data based on the distance of each point to the centroid. To compute the distance I've used numpy's np.linalg.norm which is basically Euclidian distance. This functionality is implemented in **Cluster_Assignmet** function. It takes 3 parameters 1) dataset 2) Current centroids and K

After computing the initial clusters, I compute new centroids based on the clusters.
Take the average of the all datapoints in the cluster. This average will be new centroids.
Returns the previous centroids and the current centroids. This function is implemented in **Move_Centroid** function. It takes 6 parameters

data: n dimensional Numpy array
Centroids: K-dimensional Numpy array
K: Default value is 3(for iris dataset and 2 for image dataset), splits the data into k clusters

Attributes: Number of Features in the dataset
Instances: # rows in the dataset
cluster_assignments: The current cluster assignment of each data point in the dataset.
If the centroids are changed this whole process of assigning clusters and recomputing the centroids based on the mean value of the clusters continues until all the centroids do not change or the silhouette score is less than 0.25

To compute the silhouette score, I've used the silhouette_score from sklearn.metrics

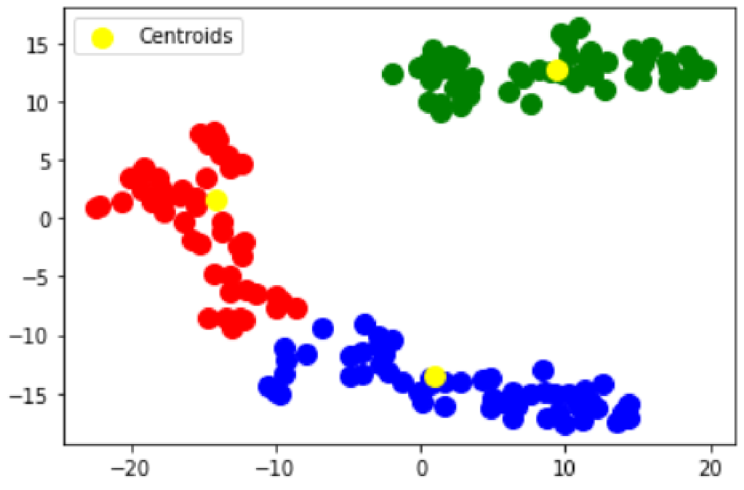
Silhouette score is calculated using the mean intra_distance of the cluster and the mean nearest cluster distance. This function returns the mean score of all samples. A score of 1 is the best and -1 is the worst.

The silhouette_score takes 2 parameters 1) dataset (N-dimensional numpy array) 2) Labels of the clusters. To plot the data I've used matplotlib.pyplot and seaborn.

Results:

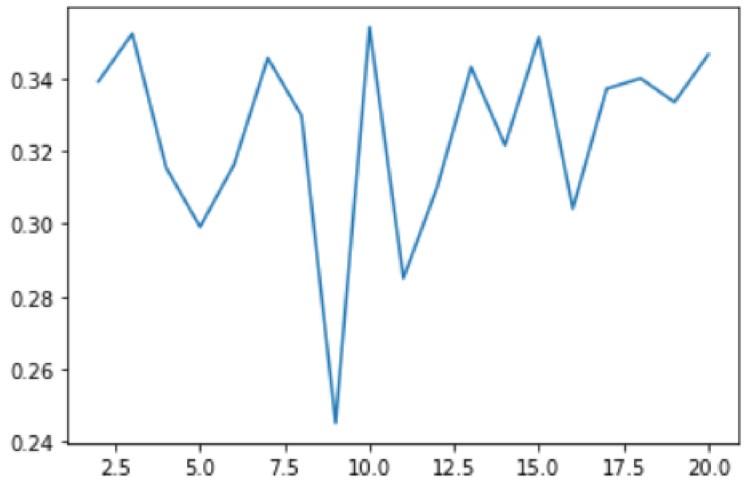
Plot of the Iris dataset

```
<matplotlib.legend.Legend at 0x7f2bd03dfe48>
```



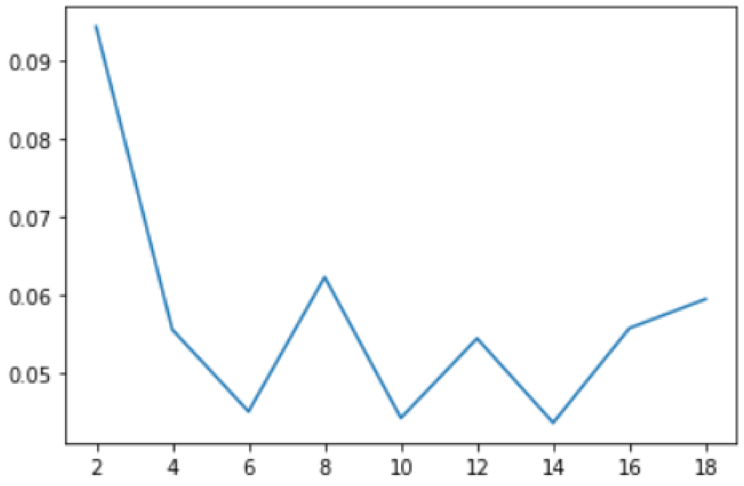
Elbow plot of Image dataset using TSNE.

```
[<matplotlib.lines.Line2D at 0x7f0120c2cd30>]
```

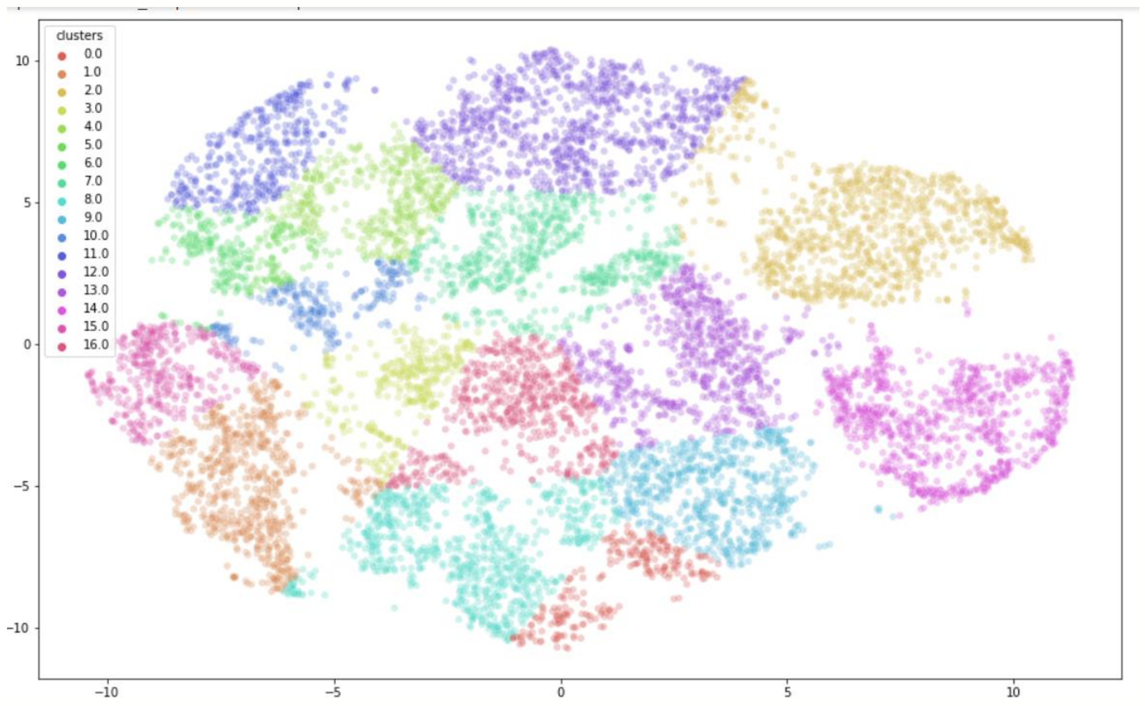


Elbow plot of Image dataset after feature reduction using PCA.

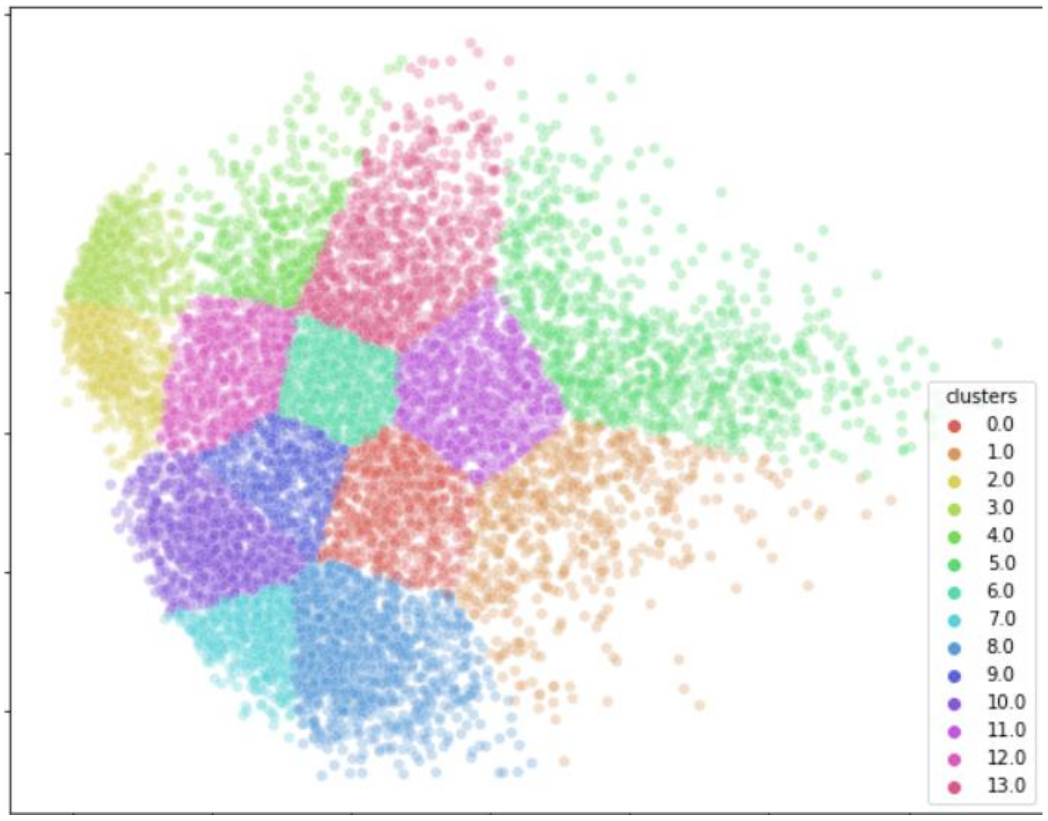
```
[<matplotlib.lines.Line2D at 0x7f4cf2c0bc88>]
```



Clustering of the Image dataset after feature reduction using TSNE



Clustering of the Image dataset after feature reduction using PCA:



References:

https://docs.scipy.org/doc/numpy1.14.0/reference/generated/numpy.linalg.norm.html#numpy-linalg-norm

https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

https://stats.stackexchange.com/questions/238538/are-there-cases-where-pca-is-more-suitable-than-t-sne

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

https://www.geeksforgeeks.org/difference-between-pca-vs-t-sne/