K-MEANS

OVERVIEW: The purpose of this assignment was to implement your own version of KMeans and test that on the Iris dataset provided. The second part of this assignment deals with using the implemented KMeans algorithm on an image dataset for clustering.

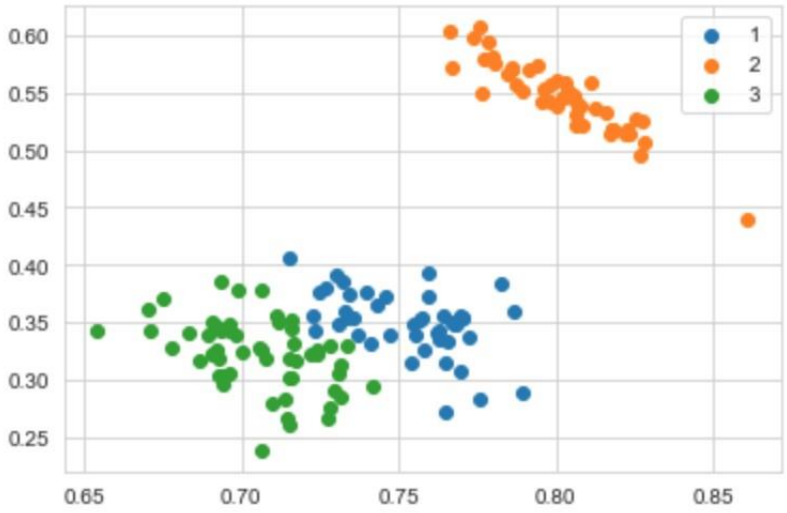
K-MEANS Implementation Steps:

1. Initialize random centroids depending on the value of K.
2. Find distances of each data point to each centroid and assign the data point to the closest centroid.
3. Updating the centroids by taking the mean of all the cluster points within that cluster 4. Repeat that process until there is no further change in the cluster formation.

Data Preprocessing for KMeans:

1. At first, I tried only standardizing the data but the accuracy I was achieving was pretty low. This could have been because of the negative values.
2. I then tried using sklearn's preprocessing normalization along with standardization but soon realized that doing both of these together yields no actual benefit.
3. In the end, I just decided to use normalization and achieved good accuracy with that.

The image below shows the centroid and clustering results that my KMeans achieved. As you can see, the clusters are assigned to their rightful centroids. They are well separated and distinct from other clusters.



PART 2

OVERVIEW: For this part of the assignment, we were given a file of a list of 28x28 pixels of digits and the goal was to assign them to clusters 1 to K.

Implementation Steps:

1. Use techniques for unsupervised learning algorithm to clean the data.
2. Experiment with multiple techniques to see which one results in the best accuracy.
3. Once the data is cleaned, use the KMeans implemented in the previous step to assign clusters to their closest centroids.

Data preprocessing steps:

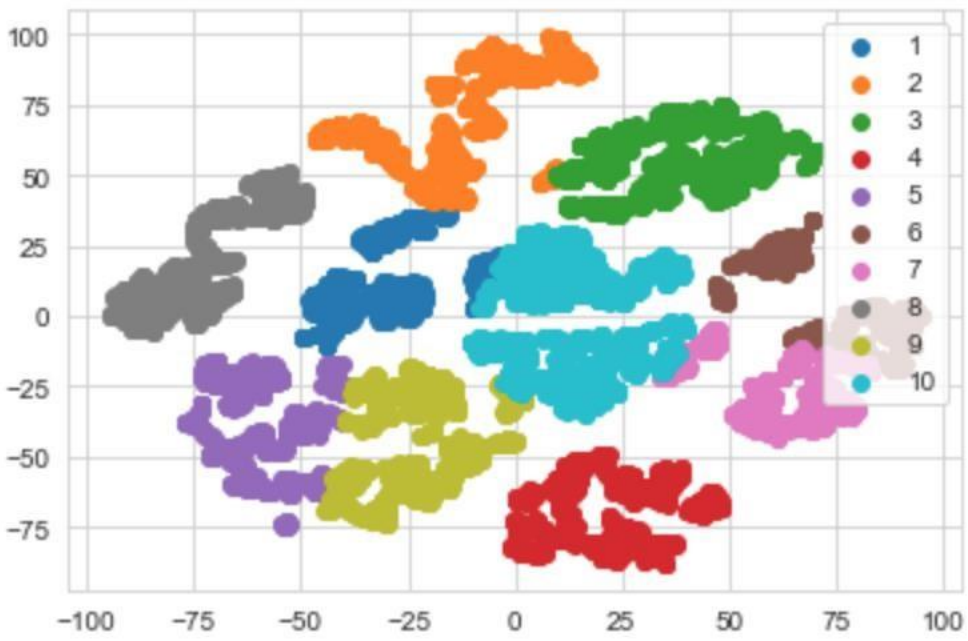
1. The dataset had 784 features and 10000. To reduce the time that it takes to run the program and to make it more efficient, I used some dimensionality reduction techniques.
2. When I started the second part of the assignment, I had to research about the best techniques for high number of data points. I found out that feature selection is mostly used for supervised learning so I decided to go with other techniques.
3. The first linear technique I attempted was Principal Component Analysis (PCA). I used 400 as the total number of components but didn't get a decent accuracy.
4. Then I studied more about PCA and learned that it's advised not to use any actual numbers for components but instead use a percentage of variance you want to be explained. I used 95% variance and got a better result.
5. The result was still not very good, so I decided to couple the PCA-ed dataset with the non- linear technique t-Distributed Stochastic Neighbor Embedding (t-SNE).
6. I decided to go with 2 as my number of dimensions for the t-SNE and my accuracy increased by over 10%.

Some of the experiment results are stated in the table below:

Exp.	PCA	t-SNE	Accuracy	Time
1	No of Components = 400	_____	0.42	7 seconds
2	95% Variance	_____	0.66	9 seconds
3	95% Variance	N = 2	0.77	9 seconds | 44 seconds
4	_____	N = 2	0.78	40 seconds

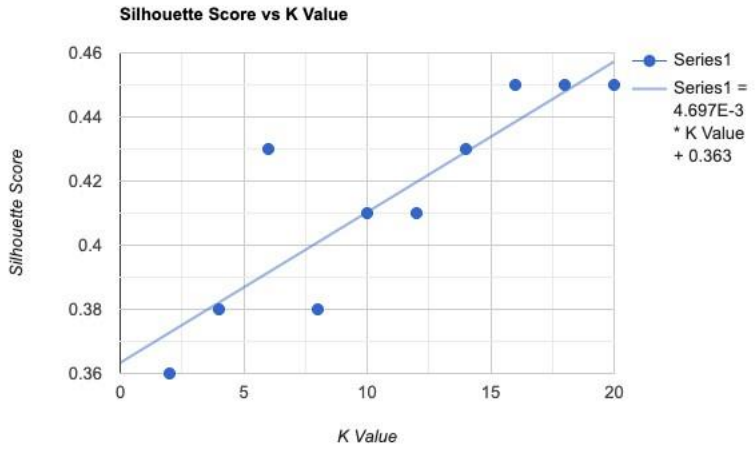
As can be seen from some of the results above, I tried PCA alone, then PCA along with t-SNE, and last t-SNE alone. I got the best accuracy with just the t-SNE.

A cluster plot of the latest results can be seen below:



As can be seen, even though the cluster accuracy is good, but it is far from perfect and definitely have plenty of room for improvement.

To see how varying the value of K would affect the clustering, the Silhouette Score was used from sklearn's library. The score (on the y – axis) was plotted against the value of K (x-axis). The graph of the results are shown below:



Overall, a linearly proportional relationship can be seen between the K value and the silhouette score. A score close to 1 would suggest perfect clustering while a score of 0 would suggest some data points might be in the wrong cluster.

There is one big outlier at K = 6 and the score tends to level off at around K = 16 and above. Through this analysis, we can confirm the claim that this algorithm has plenty of room for improvement.

Sources:

1.https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_analysis_of_silhouette_score.html (Silhouette Score)

2. https://www.rapidtables.com/tools/scatter-plot.html (Plot maker)
3. https://towardsdatascience.com/pca-vs-tsne-el-clásico-9948181a5f87 (PCA vs t-SNE)
4. https://medium.com/machine-learning-bites/machine-learning-unsupervised-learning-featureselection-a9bdccd70f95 (Filtering techniques for unsupervised learning)