Clustering Report

Venkat Sai Charan Bandi
G01219773

Miner UserName: Cherry_SLYR

Part 1: Iris Clustering

Rank	**User**	**Submission Time**	**Public Score**
42	Cherry_SLYR	Oct. 14, 2020, 11:37 a.m.	0.95

Part 2: Image Clustering

Rank	**User**	**Submission Time**	**Public Score**
211	Cherry_SLYR	Oct. 15, 2020, 12:33 p.m.	0.69

Instructions to the TA

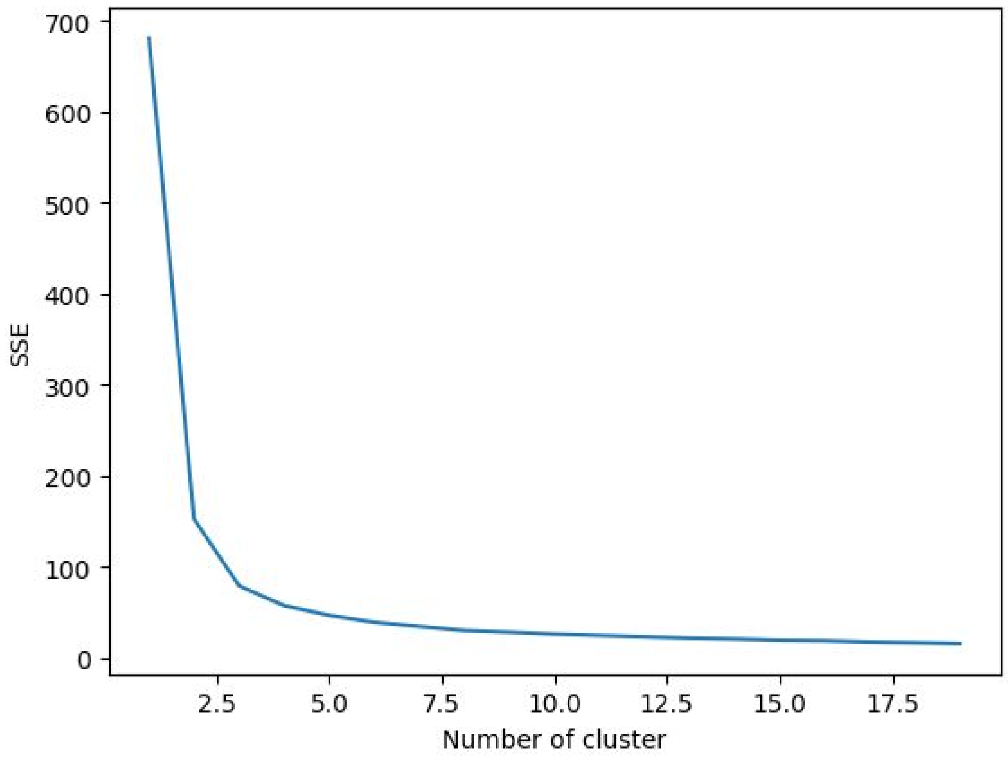
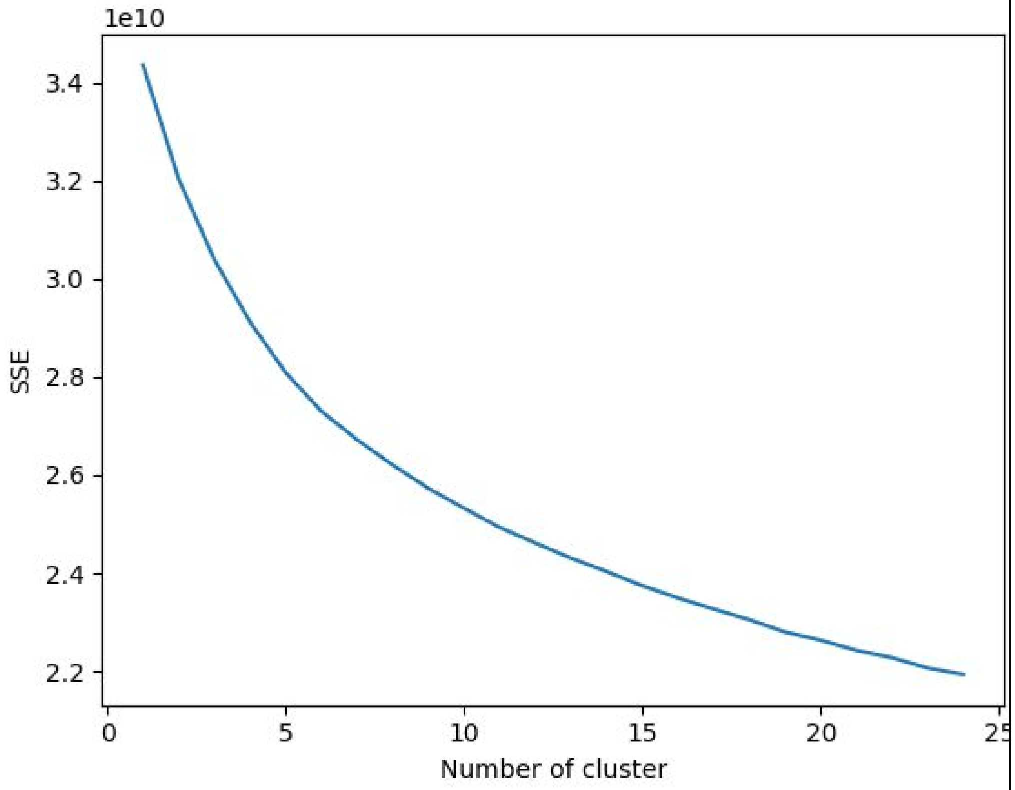
I attached my PyCharm Project folder as **code** in the src folder, this contains three main python files

- The iris part 1 file called 'code_iris.py'
- The image part 2 file called 'code_image.py'
- The third one is the visualizer for the sse plot called the 'sse_graphs.py'
- Running this code you will also need to install multiple libraries like matplotlib, scipy, sklearn, seaborn, pandas, etc. I used pip to install them.

Internal evaluation metric

To choose a value for K, I considered the elbow method and the average silhouette model. I implemented a **SSE model** using a range of values from 0 to 25 using pre made kmeans models. And getting the sum of distances of samples to their closest cluster center. The two graphs i got gave me information to continue the experiments with the observation that as K increases SSE decreases as disortation will be small.

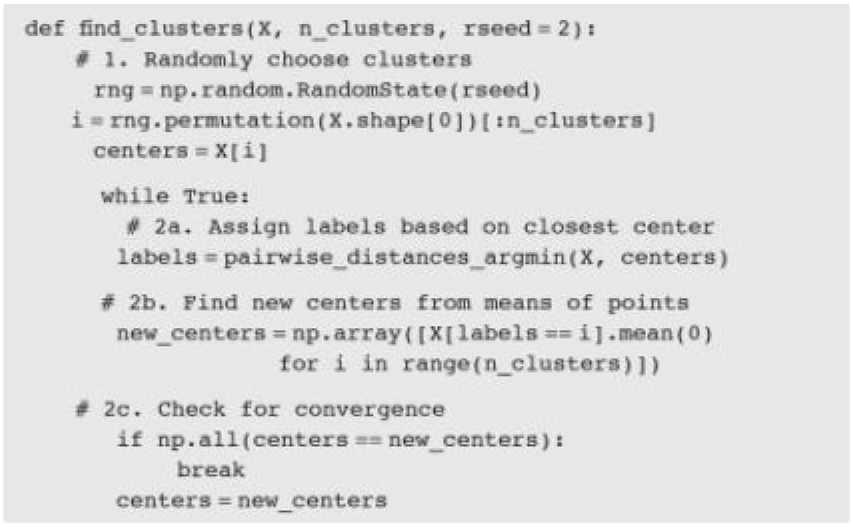
- Here we observe the iris dataset has a significant drop from 3 and so on, so i picked 3.
- Here we observe that the image has a consistent drop from 8 and so on, so i experimented my cluster by changing values from 8 to 20 and stuck with 10 that gave me the greatest results.values

Iris Clustering	
Image Clustering	

The Approach

Part 1: Iris Clustering
During the initial parts of understanding the problem, i read about the famous iris classification and looked up some most appropriate algorithms to use,, one such algorithm that proved consistent results was the k means clustering algorithm. I used the algorithm mentioned in the book :

Practical Machine Learning for Data Analysis Using Python By Abdulhamit Subasi
2.4 clustering for feature extractions and dimension reduction

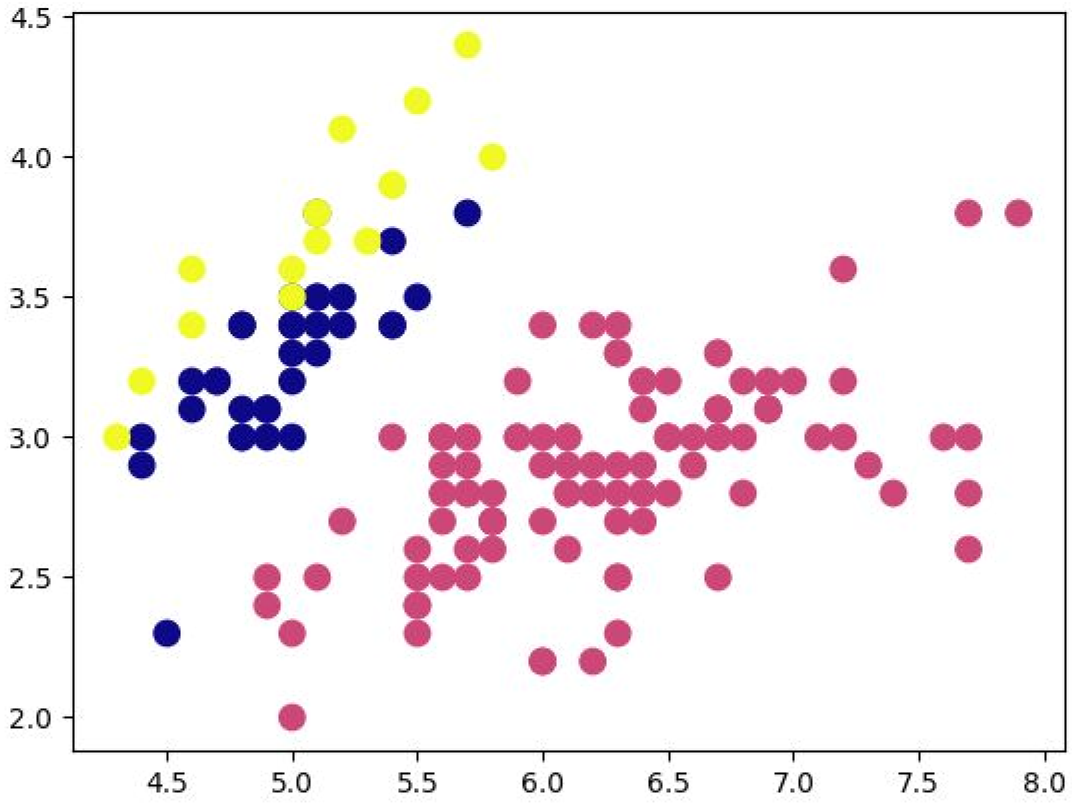
Pseudo Code: 

```
def find_clusters(X, n_clusters, rseed = 2):
    # 1. Randomly choose clusters
     rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
     centers = X[i]

      while True:
        # 2a. Assign labels based on closest center
        labels = pairwise_distances_argmin(X, centers)

      # 2b. Find new centers from means of points
        new_centers = np.array([X[labels == i].mean(0)
                     for i in range(n_clusters)])

    # 2c. Check for convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers
```

The algorithm is self explanatory and uses self learning mechanism by following:

- Randomly picks k number of cluster centers from, this k value depends on the SSE metric from above.
- We use the distance measure to calculate the distance from each of these data points to the newly assigned centers, for distance, I tried using the Euclidean distance as a parameter for the distance calculation but it wasn’t a good metric for this problem and reported my score of 0.72 for the iris dataset, hence stuck kept using the cosine distance.
- Then find the mean of these new clusters data points assigned and we check if the new centroids are equivalent to the old ones, if they aren't the same it means we found a better centroid.
- We keep doing this recursion until we get the best value and then we exit out, finding the k means clusters.

With this initial algorithm done i recorded a score 0.95 on the iris and moved on to the image dataset.

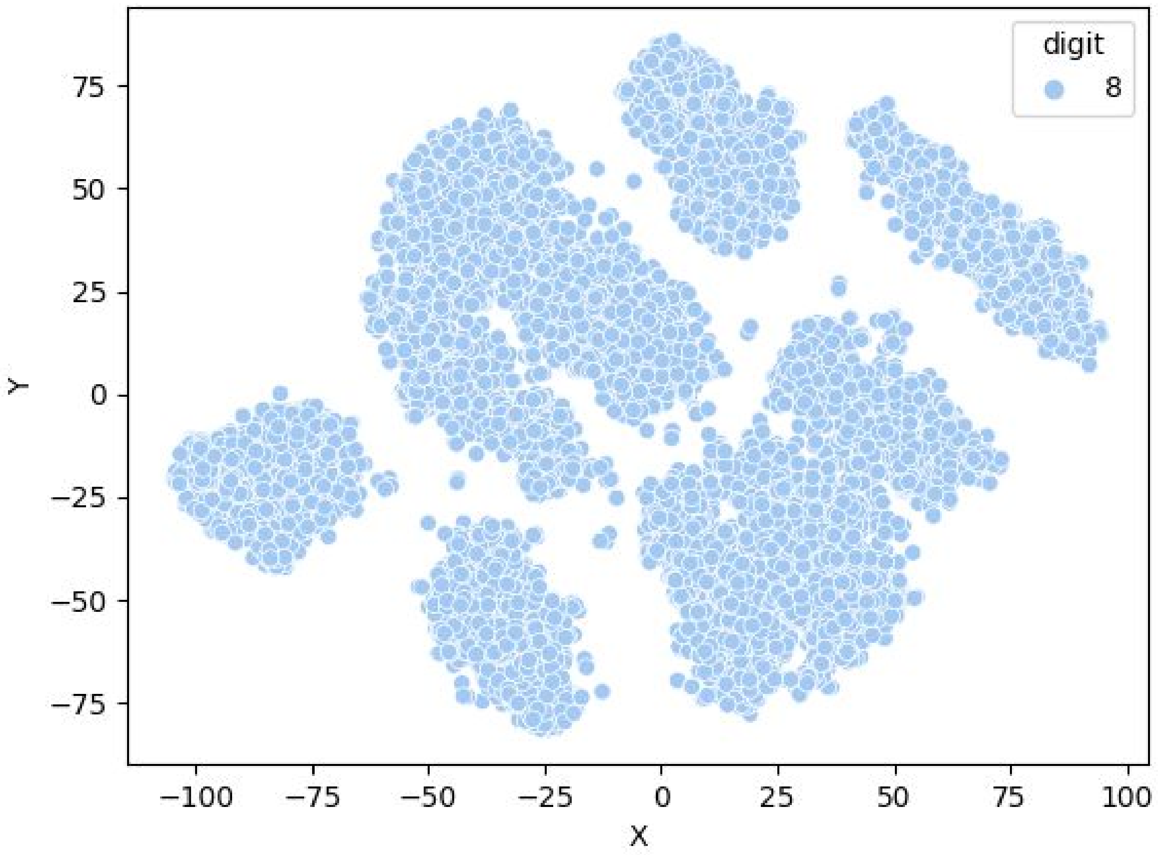


Part 2: Image Clustering

Here my first score with the same algorithm proved weak and recorded a score of 52. I then looked up the high dimensionality problem with the image dataset having a lot of features that aren't always proving worthy and hence i followed some optimizations, namely:

1. **T-Distributed Stochastic Neighbouring Entities (t-SNE)**: for dimensionality reduction and for the visualization of high-dimensional datasets, helped with graphing the data also. This essentially minimizes the divergence between two distributions helps to create a much clearer differentiation running through the k means algorithm
2. **Principal component analysis (PCA):** Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.

After initially running the algorithm with both TSNE and PCA, I was able to differentiate the clusters much clearly with visualization (graph attached below) and the kmeans then could identify the clusters easily.



The most time taking part of this project was to identify the proper parameters to separate the clusters and to properly normalize them from a higher dimension of 1x784 vector to a two dimensional distance matrix coordinates as such. And hence i spent my submission trying to optimize the parameters by running trails on each TSNE and PCA as shown in the table below and then getting a score of 0.69 on the miner.

Method	Score on Miner	K Clusters Value	Parameters
K means	0.53	10	Dist = cosine
K means	0.54	11	Dist = cosine
K means	0.54	12	Dist = cosine
K means	0.56	13	Dist = cosine
K means	0.57	14	Dist = cosine
K means	0.57	15	Dist = cosine
K means	0.58	16	Dist = cosine
K means	0.58	17	Dist = cosine
K means	0.57	18	Dist = cosine
K means with TSNE	0.59	10	learning_rate=200.0, n_iter=2000
K means with TSNE	0.65	10	perplexity=50.0, learning_rate=200.0, n_iter=2000
K means with TSNE and PCA	0.66	10	PCA (n_components=30)
K means with TSNE and PCA	0.65	10	PCA (n_components=30)
K means with TSNE and PCA	0.65	10	PCA (n_components=30)
K means with TSNE and PCA	0.62	20	PCA (n_components=30)
K means with TSNE and PCA	0.66	12	PCA (n_components=35)
K means with TSNE and PCA	0.69	10	TSNE(perplexity=50.0, learning_rate=200.0, n_iter=2000, n_iter_without_progress=1000, early_perceptron=4) && PCA (n_components=30)