HW 3: Image Clustering

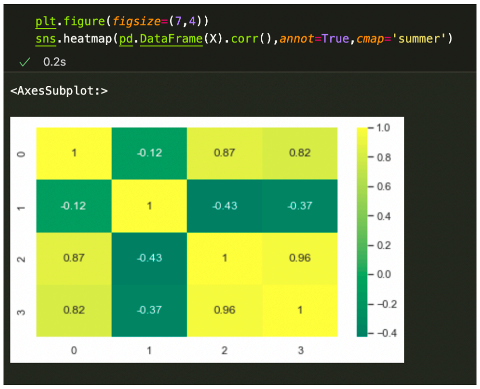
Details:

Name : Siddhanth Kalyanpur
Miner username : mb13
Miner Score : part 1) 0.73 part 2) 0.61

Approach:

1. Applying K means and K means ++ for clustering on Iris data:

- No Preprocessing required as the dataset was small and the features were highly correlated as seen in the below heat map. Received a lower score V score with PCA. Tried applying PCA to better visualise on 2-d graph but the result was un affected and hence didn't perform feature reduction.



```
plt.figure(figsize=(7,4))
sns.heatmap(pd.DataFrame(X).corr(),annot=True,cmap='summer')
```

✓ 0.2s

```
<AxesSubplot:>
```

- Silhouette Coefficient :

  To compute the optimal value of K used Silhouette Coefficient metric .The value ranges between -1 and 1. Its calculated using the below method.

  Silhouette Score = (b-a) / max(a,b) where,

  a= average intra-cluster distance i.e the average distance between each point within a cluster.

  b= average inter-cluster distance i.e the average distance between all clusters.

K	2	3	4	5	6	7	8	9	10	11
Score	0.8524	0.7261	0.225	0.245	0.304	0.222	-0.119	0.003	0.160	0.017

K Means Algorithm (V-Score : 0.73)

- Based on the Exploratory data analysis will be working with 3 clusters. with the initial set of centroids selected randomly.
- Calculating the Euclidean distance all the data points from the centroids deleted and assign data points to the cluster with the least distance from respective centroid.
- After all the data points are assigned to a cluster, we calculate the mean value of the cluster, and call this the new centroid for that cluster. This will further refine our clusters. And we repeat this process of selecting centroids and clustering until we don’t see any change in our new centroids. This means we have found the optimal centroid for that cluster and that marks the end of K means algorithm.

K Means ++ Algorithm: (V-Score : 0.73)

- Initialise first centroid randomly and get the remaining centroids based on Maximum probability.
- Calculate distance of all data points from the centroid and pick the point with the maximum probability (which will be the point with max distance from any of our selected centroids) to be out next centroid.
- The above step will be repeated until we get the required number of unchanged centroids

2. Applying K means and K means ++ for clustering on MNIST data:

- Due to a huge number of features we need to preprocess the data set. Applied the below Preprocessing techniques.

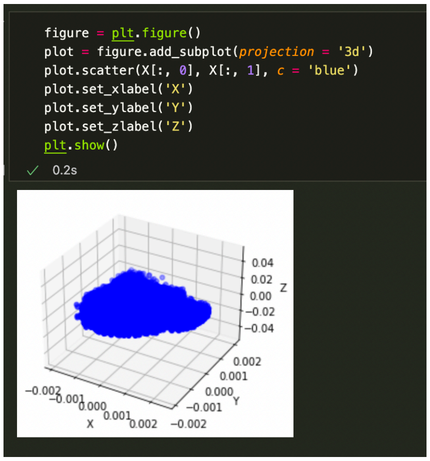
Normalization:

Scaling the data set to a common scale for similarity.

PCA:

Dimensionality reduction on the normalised data with number of components = 50

The shape of the data is now : (10740, 50)

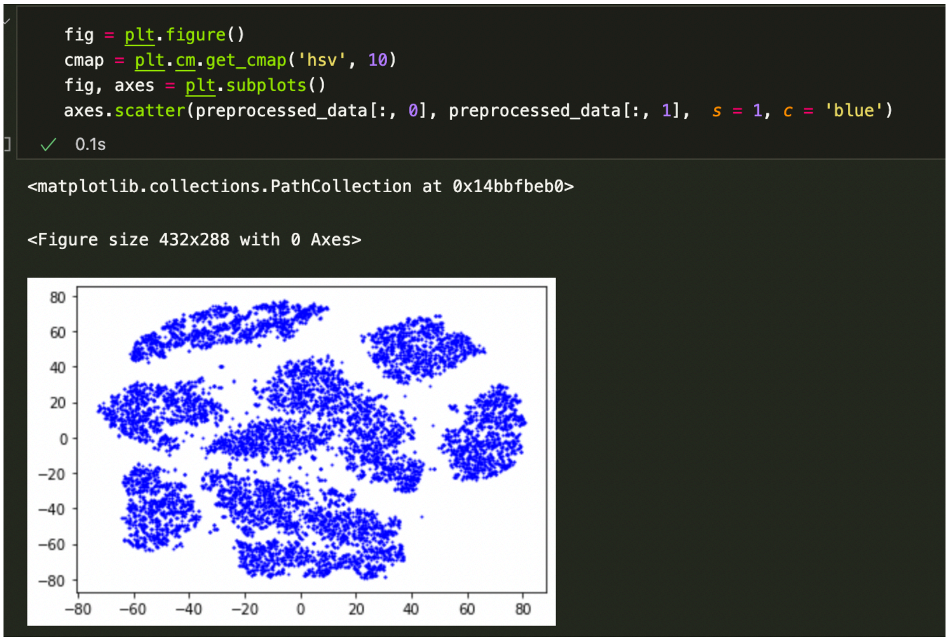


```
figure = plt.figure()
plot = figure.add_subplot(projection = '3d')
plot.scatter(X[:, 0], X[:, 1], c = 'blue')
plot.set_xlabel('X')
plot.set_ylabel('Y')
plot.set_zlabel('Z')
plt.show()
```

✓ 0.2s

t-SNE:

To project the feature set on a different plane and re cluster the data points based on its distance . After applying t- SNE we are reduced to 2 features.



```
fig = plt.figure()
cmap = plt.cm.get_cmap('hsv', 10)
fig, axes = plt.subplots()
axes.scatter(preprocessed_data[:, 0], preprocessed_data[:, 1],  s = 1, c = 'blue')
```

✓ 0.1s

```
<matplotlib.collections.PathCollection at 0x14bbfbeb0>

<Figure size 432x288 with 0 Axes>
```

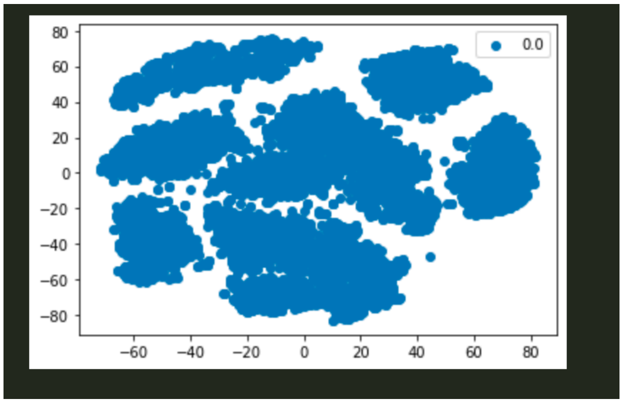
After applying the above preprocessing techniques and simplifying the dataset, we can directly apply the model we created for iris dataset, directly to the image data with the value of k = 10 (given)

K Means performed better than KMeans plus (0.58) hence choose KMeans.

We try values of k ranging from 2 to 20 with a step of 2. This gives us the following observation for the sum of squared errors.

K	2	4	6	8	10	12	14	16	18	20
Score	0.44	0.52	0.55	0.62	0.61	0.59	0.55	0.55	0.52	0.51

The cluster we get on k=10



● 0.0
80
60
40
20
0
−20
−40
−60
−80
−60
−40
−20
0
20
40
60
80