ECE — Large-Scale Data Mining: Models and Algorithms

Name: Anwesha Chattoraj, Zhaoliang Zheng, Shivam Kumar Panda
Student UID: 405350265, 605432345, 105730045
Due Date: Feb. 7 2022, 11:59 PM **Project:** 2

1 Part 1

Question 1

The size of the TF-IDF matrix obtained is: (4154, 16360).

Question 2

The contigency matrix is shown in the Fig 1.

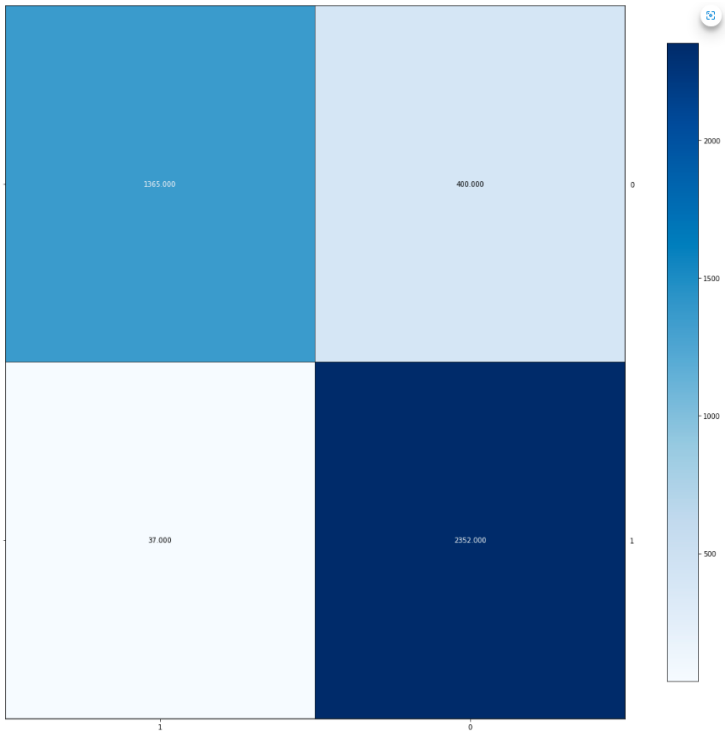


Figure 1: The Contigency Matrix for Q2

No the contingency matrix may not square-shaped. The case where the number of clusters predicted is different from the ground truth number of clusters, then the contingency matrix would be a non-square rectangular matrix.

Question 3

The 5 required measures for K-means are given below.
Homogeneity: 0.537
Completeness: 0.572
V-measure: 0.554
Adjusted Rand-Index: 0.622
Adjusted Mutual Information Score: 0.554

Question 4

The required plot is shown in the Fig 2.

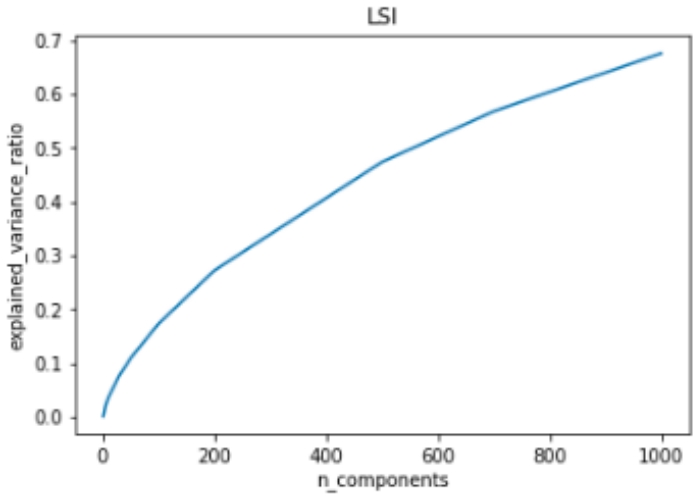


Figure 2: The plot of the percentage of variance that the top r principle components retain v.s. r

Question 5

The required plot for SVD are in the Fig 3 and Fig 4 After trying all cases of given r values, r=3 is found to the best case for both SVD and NMF. r=2 also has comparable results, however we go with r=3.

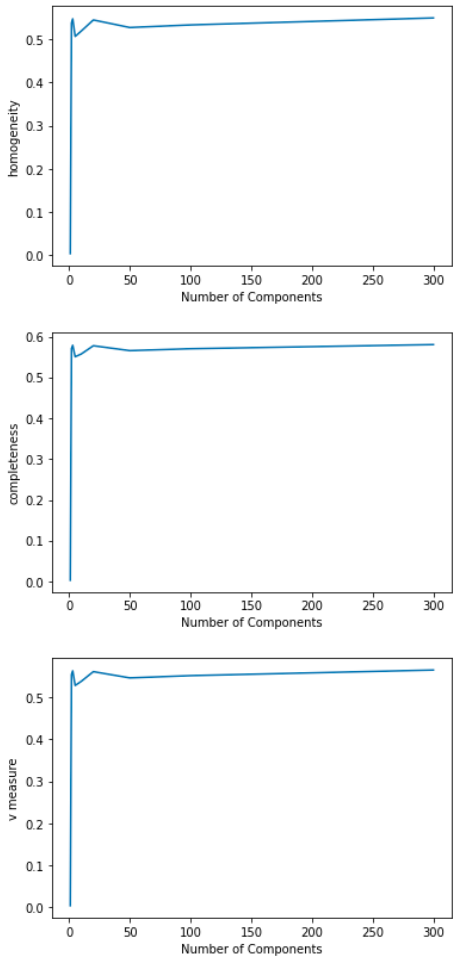


Figure 3: Homogeneity, Completeness and V-measure with SVD

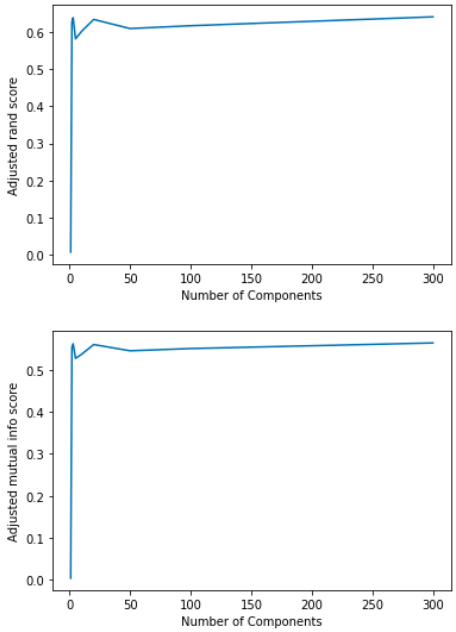


Figure 4: Adjusted rand score and adjusted info score with SVD

Question 6

With low values of r, it can be possible that a lot of information is lost during the dimensionality reduction process. Hence it's difficult for the K-means to cluster them. When r increases, although the information can be retained but the performance of K-means may deteriorate as the dimension increases. Hence the behaviour of r is usually increasing at lower dimension and then decreasing or remains constant over higher dimension.

Question 7

The measures for SVD with the best r values is better than that in Q3 (without dimensionality reduction). However the measures for NMF with the best r values is not better than that in Q3.

2 Question 8

1. **SVD** with $\mathbf{r = 3}$ with K-Means clustering.
   Homogeneity: 0.547
   Completeness: 0.579
   V-measure: 0.563
   Adjusted Rand-Index: 0.638
   Adjusted Mutual Information Score: 0.563

The Contingency Matrix can be found in the Fig 5.

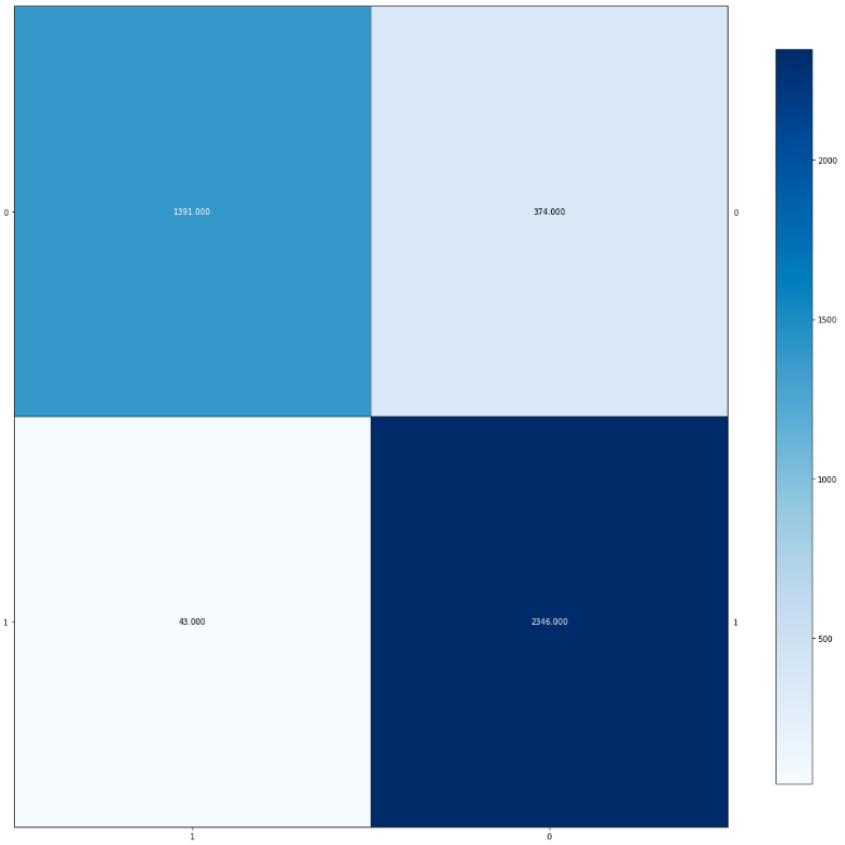


Figure 5: Contingency Matrix for SVD with r = 3 with K-Means clustering

A Visualization of the data in 2-Dimension is shown in the Fig 6.

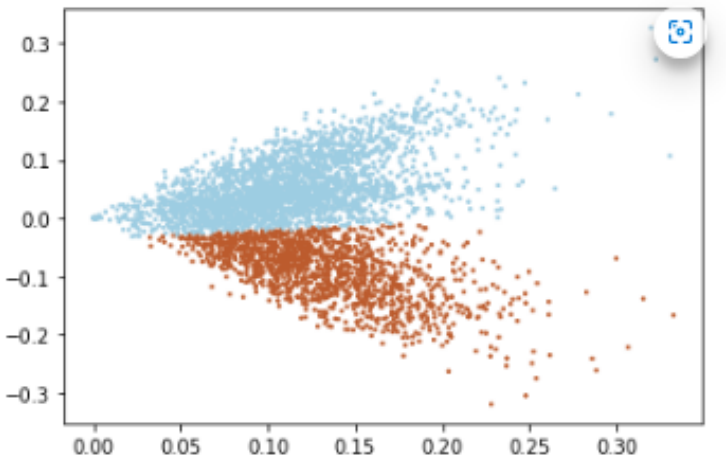


Figure 6: Visualization on 2D for SVD with r = 3 with K-Means clustering

2. **NMF** with $\mathbf{r = 3}$ with K-Means clustering.
   Homogeneity: 0.500
   Completeness: 0.545
   V-measure: 0.521
   Adjusted Rand-Index: 0.574
   Adjusted Mutual Information Score: 0.521
   The Contingency Matrix can be found in the Fig 7.

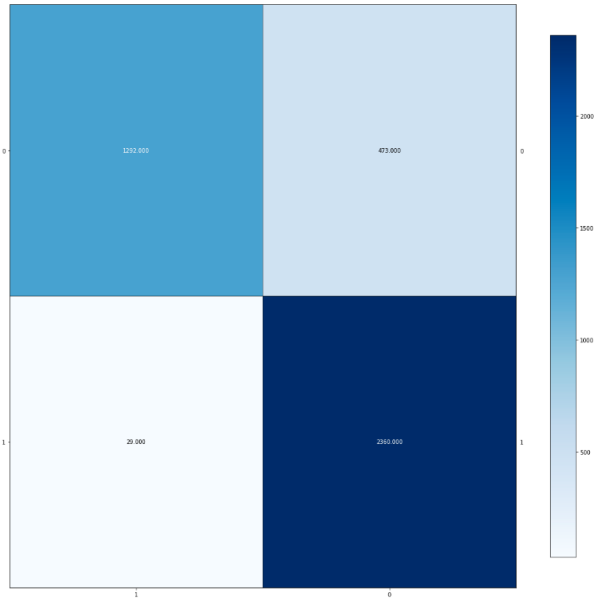


Figure 7: Contingency Matrix for NMF with r = 3 with K-Means clustering

A Visualization of the data in 2-Dimension is shown in the Fig 8.

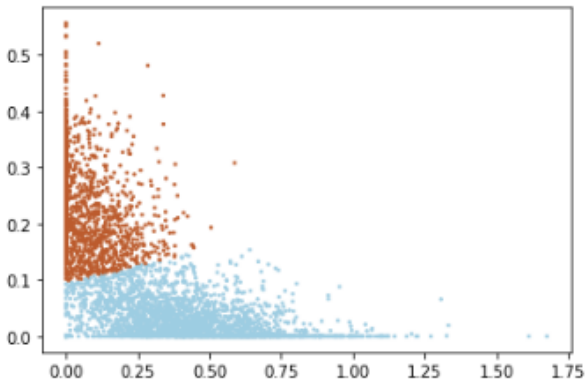


Figure 8: Visualization on 2D for NMF with r = 3 with K-Means clustering

Question 9

1. **KMeans with SVD**
   In this case with a visualization of the ground truth on 2D plane we observe that the cluster is closely packed. Hence it's difficult (not ideal) for the KMeans algorithm to obtain an optimized centroid point which would cluster the data points accurately. We also observe few points overlapping on each other's cluster. However the KMeans does a good enough job, however it misses out the overlapping points. Also in case of KMeans it captures good enough when the structure of the cluster is spherical-like (on 2D plane circle like). Since we are visualizing it on a 2D plane it's difficult to judge this as the clustering has been carried out on 3-dimensions. Observing the results we can say that on 3-dimensions, the structure of the data points in the cluster is more spherical-like.

2. **KMeans with NMF**
   In this case also with a visualization of the ground truth on 2D plane we observe that the cluster is closely packed. Hence again it's not ideal for the KMeans algorithm to obtain an optimized centroid point which would cluster the data points accurately. In this case also observe few points overlapping on each other's cluster. However the KMeans does a good enough job, but not as good as compared to KMeans with SVD. Also the structure of the data points in the cluster is spherical-like but less than that of KMeans with SVD.

Hence we can conclude that SVD is a better process for dimensional reduction of this data while we are doing KMeans.

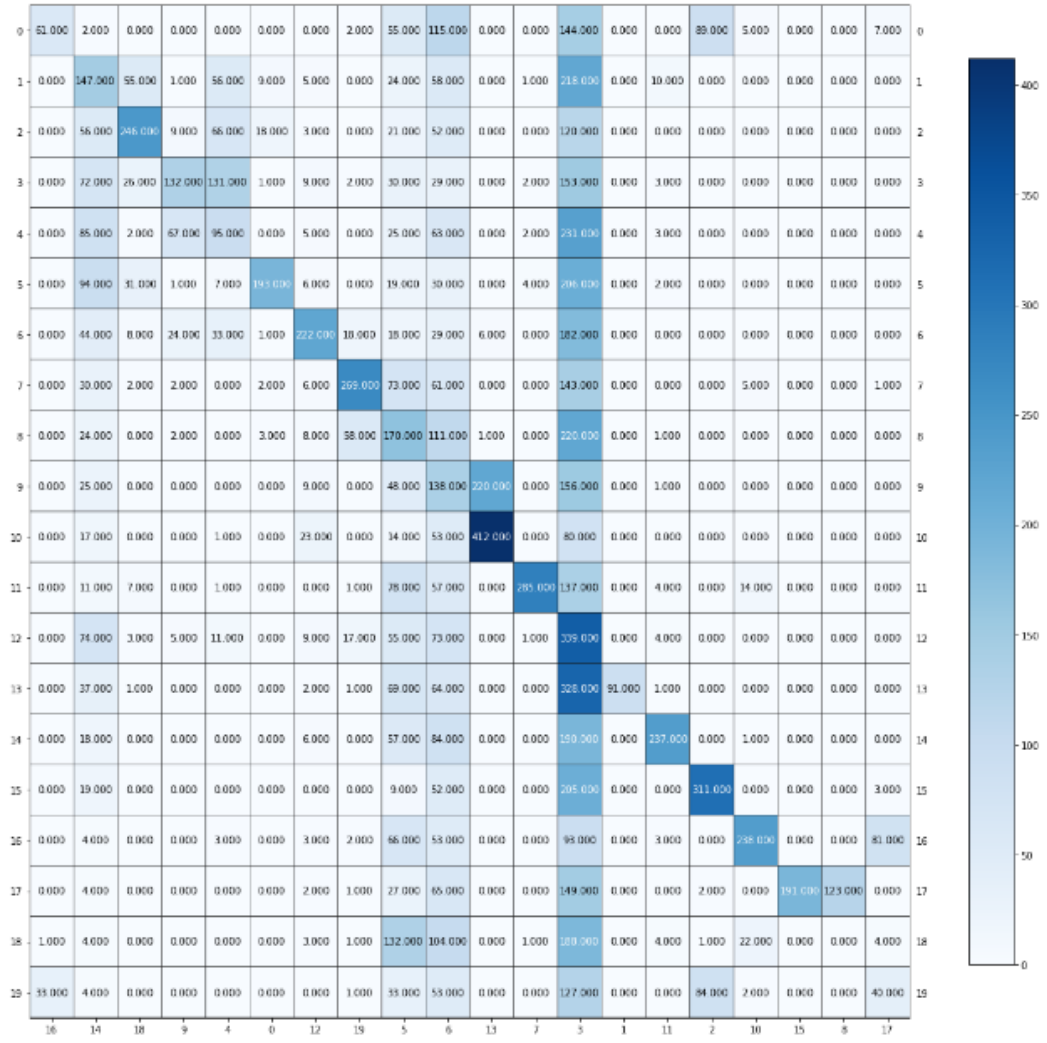
Question 10

We ran through a the components 5, 20, 200 to check performance of svd and nmf with these hyperparameters on full dataset The detailed plots of the runs are in the jupyter notebook

Best NMF Performance

```
print_metrics(kms_nmf[best_index_hom_nmf],labels=y_train_full,train_embeddings=train_full_nmf)#nmf_embeddings[best_i
print("Best Number of components for NMF full dataset  {} ".format(*hypers_nmf))
```

```
Homogeneity: 0.313
Completeness: 0.382
V-measure: 0.344
Adjusted Rand-Index: 0.083
Adjusted Mutual Information Score: 0.340
Contingency matrix:
```



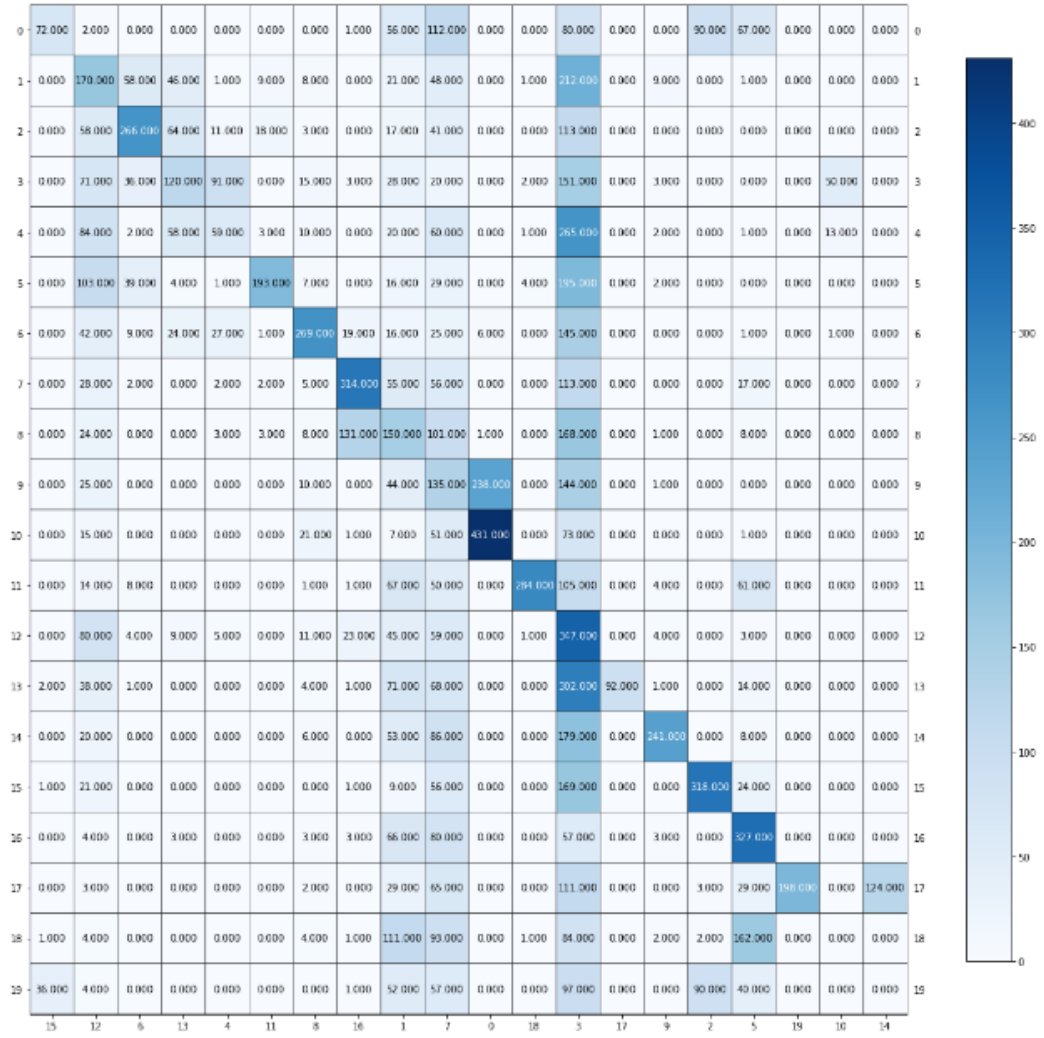
```
Best Number of components for NMF full dataset  20
```

Figure 9: Contingency Matrix for NMF with r = 20 with K-Means clustering for full-dataset

Best SVD Performance

```
print_metrics(kms_svd[best_index_hom_svd],labels=y_train_full,train_embeddings=train_full_svd) #svd_embeddings[best_
print("Best Number of components for SVD full dataset {} ".format(*hypers_svd))
```

```
Homogeneity: 0.332
Completeness: 0.392
V-measure: 0.360
Adjusted Rand-Index: 0.111
Adjusted Mutual Information Score: 0.356
Contingency matrix:
```



```
Best Number of components for SVD full dataset 20
```

Figure 10: Contingency Matrix for SVD with r = 20 with K-Means clustering for full-dataset

Question 11

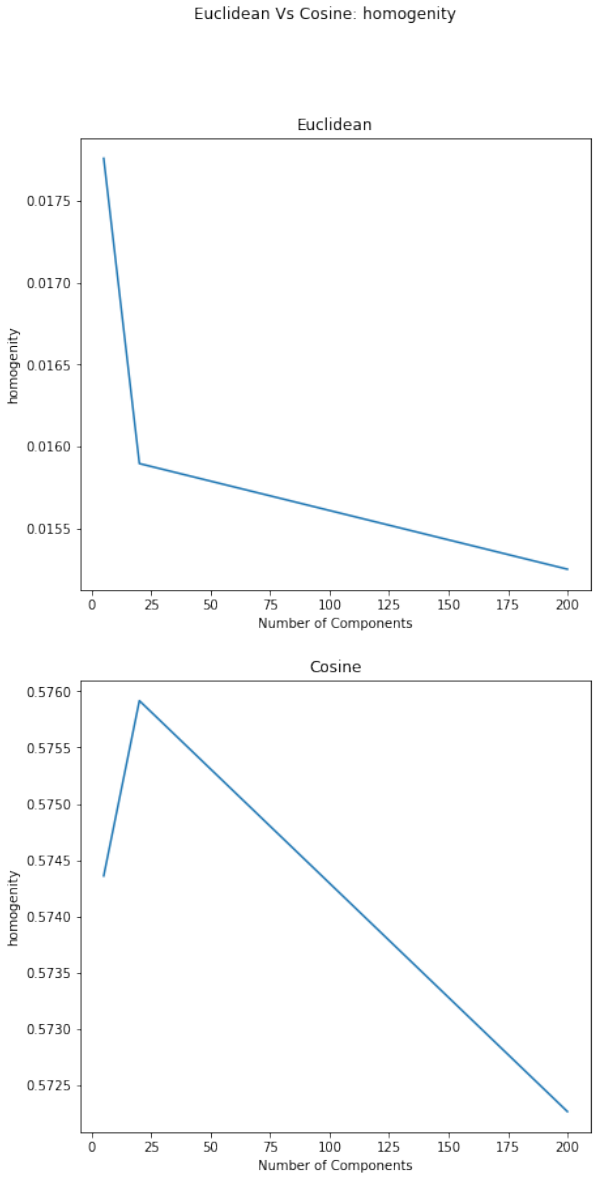


Figure 11: Comparing UMAP Hyper parameters: Number of components and Distance Metric

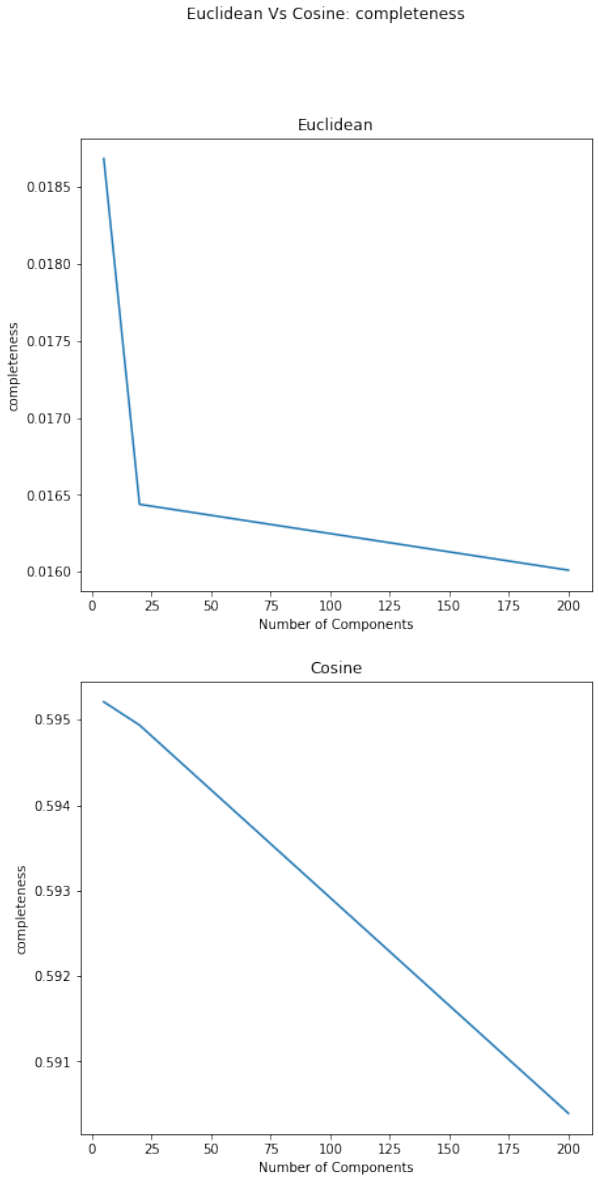


Figure 12: Comparing UMAP Hyper parameters: Number of components and Distance Metric

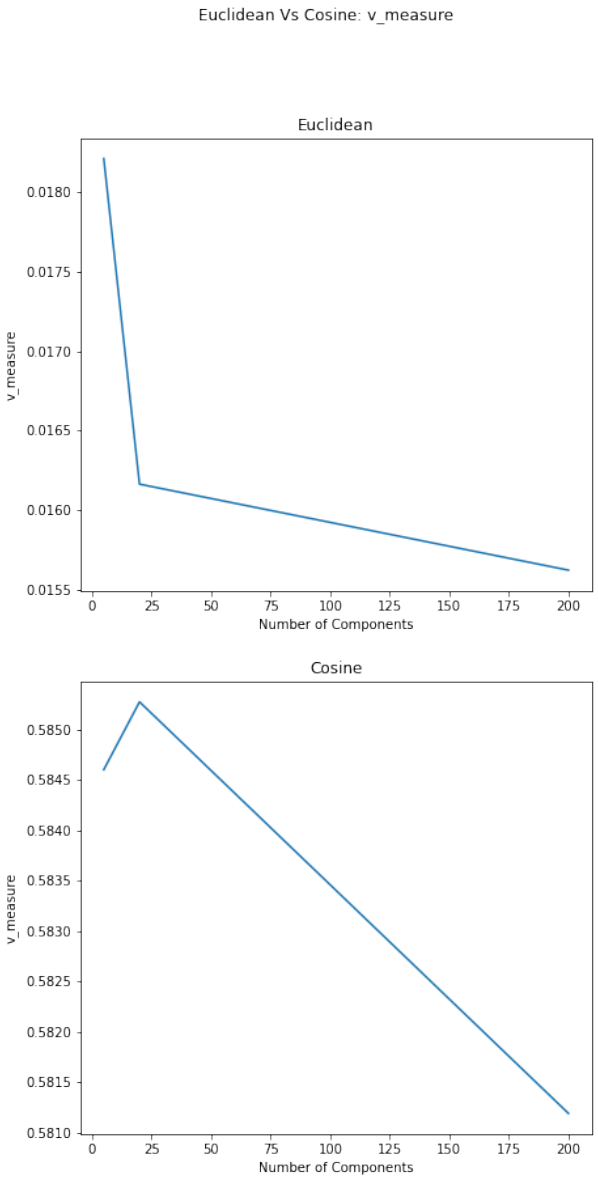


Figure 13: Comparing UMAP Hyper parameters: Number of components and Distance Metric

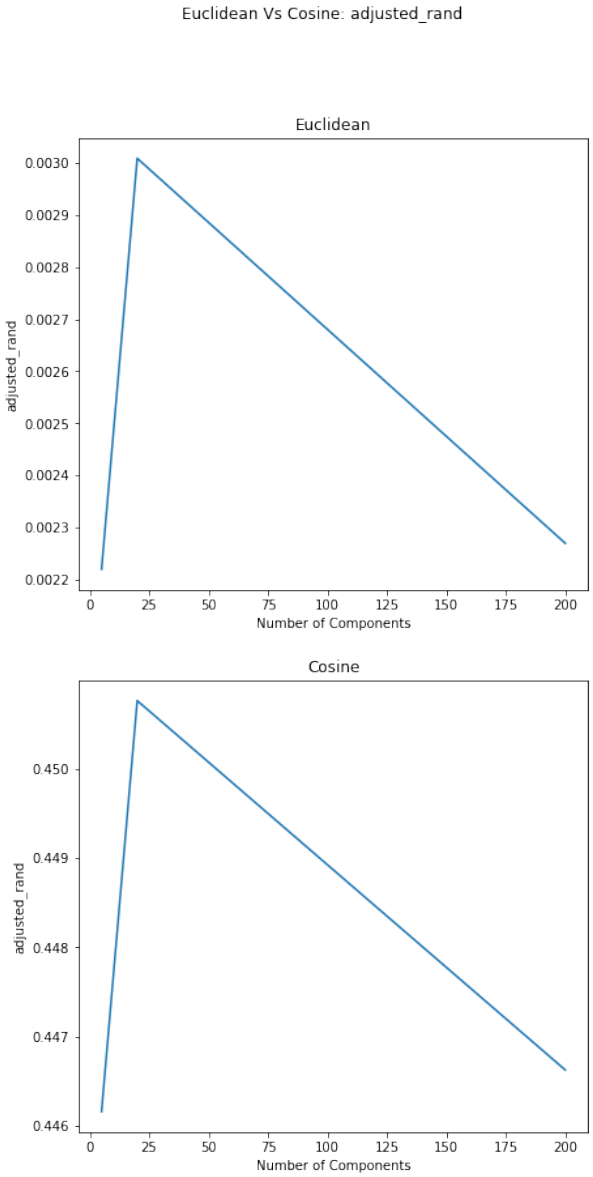


Figure 14: Comparing UMAP Hyper parameters: Number of components and Distance Metric

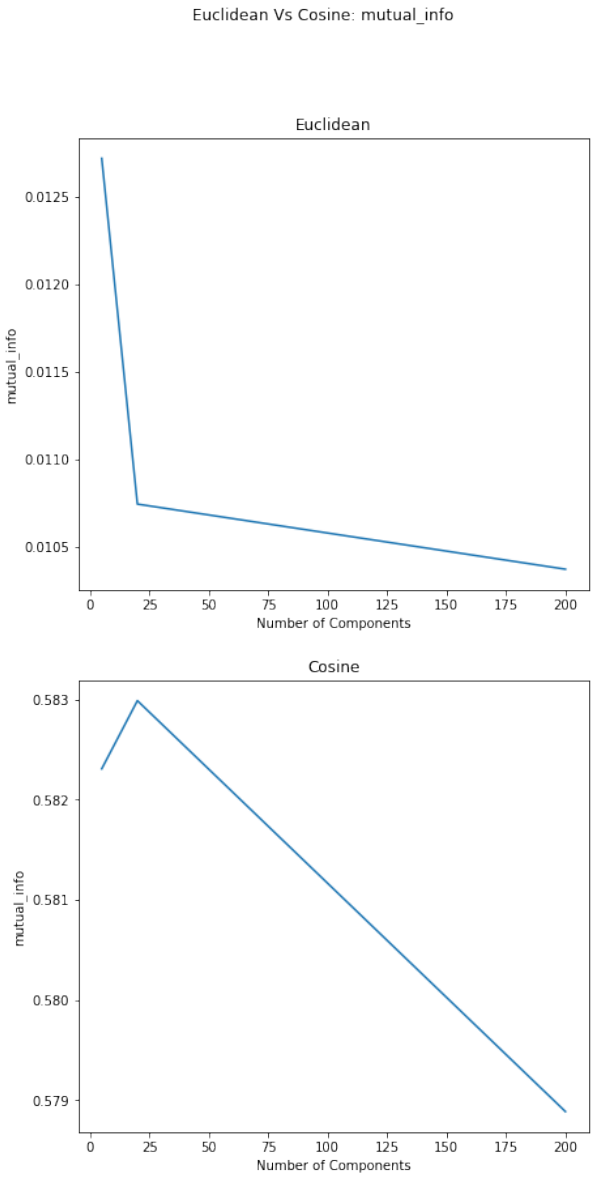


Figure 15: Comparing UMAP Hyper parameters: Number of components and Distance Metric

Best Hyperparameters: Cosine distance metric and number of components = 20

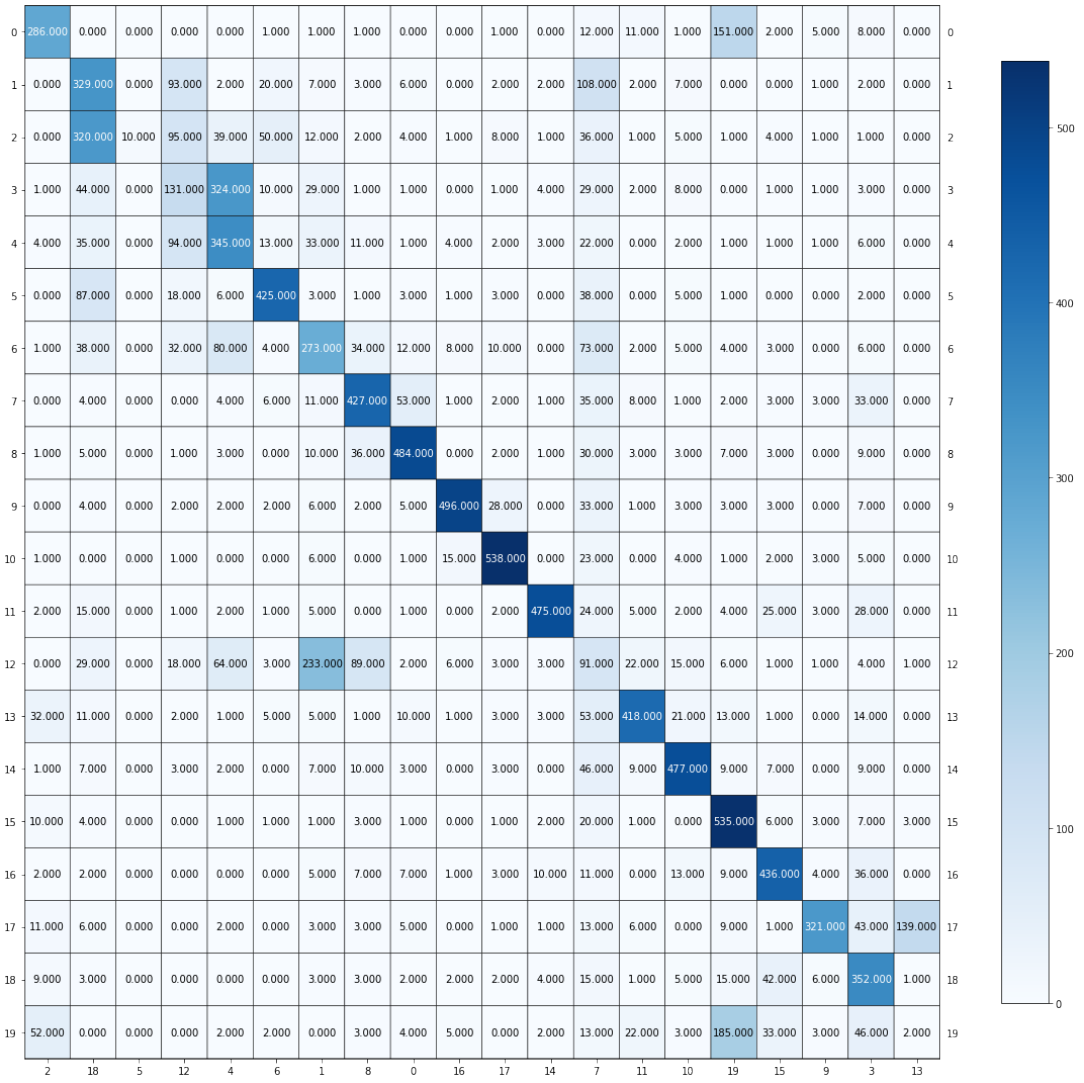


Figure 16: Best UMAP

```
Best UMAP Metrics
 homogenity  0.5759152462117662
 completeness  0.5949369784845447
 v_measure  0.5852715982724142
 adjusted_rand  0.4507659475681236
 mutual_info  0.5829900657364246
```

Figure 17: Best UMAP metrics

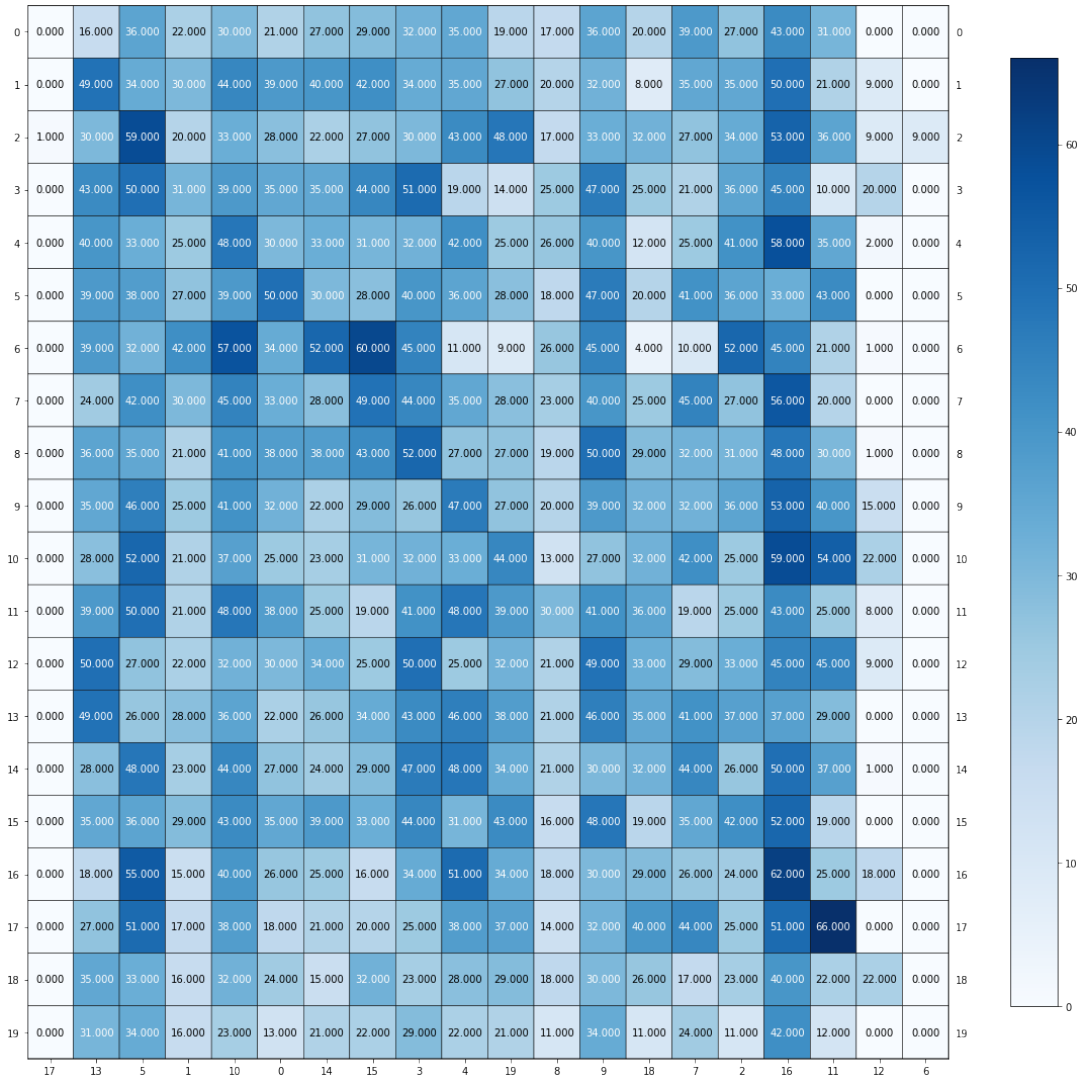


Figure 18: Euclidean Performance

```
Euclidean Metrics
 homogenity  0.015251501592623144
 completeness  0.016009659852659813
 v_measure  0.0156213871431216
 adjusted_rand  0.0022696554873147335
 mutual_info  0.010371666928549161
```

Figure 19: Euclidean Performance metrics

Question 12

- Cosine distance metric performs much better than Euclidean distance metric

- Given this is a high dimensional space, the cosine distance works better since it does not take into account the magnitude of the tfidf vectors, only the similarity between them. Comparing magnitudes in this case is not that relevant, since document sizes are uneven.
- UMAP number of components =20 has the best clustering metrics overall
- The clusters names don't match the actual clusters, but we can see a clear match between the actual classes and the clustering performance of the kmeans
- The homogeniety is relatively good for cosine and n_com=20. For Euclidean distance metric, the contingency matrix does not show a good association between clusters and actual classes, since it takes into account the magnitude of the vectors

Question 13

- UMAP is much faster than SVD or NMF, for fit transform for the same number of components.
- Searching through 3 components for SVD took 17m 27.4 s
- Searching through 3 components for NMF took 22m 24 s
- Searching through 3 components for UMAP took 6m 13.8 s
- 200 component data on UMAP fits faster than 20 component on SVD or NMF
- The clustering metrics for UMAP dimensionality reduction is better than SVD and NMF
- The contingency matrix for UMAP shows a better association between actual classes and clusters

Question 14

Linkage criteria Ward has the better performance

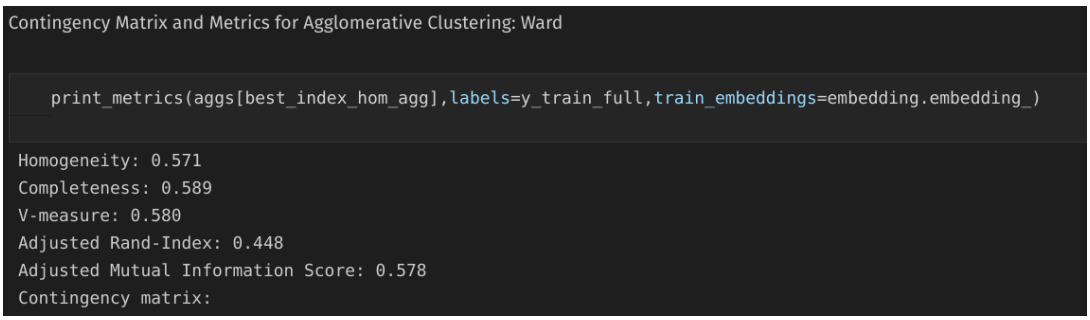


Figure 20: Metrics for Agglomerative Clustering: Ward

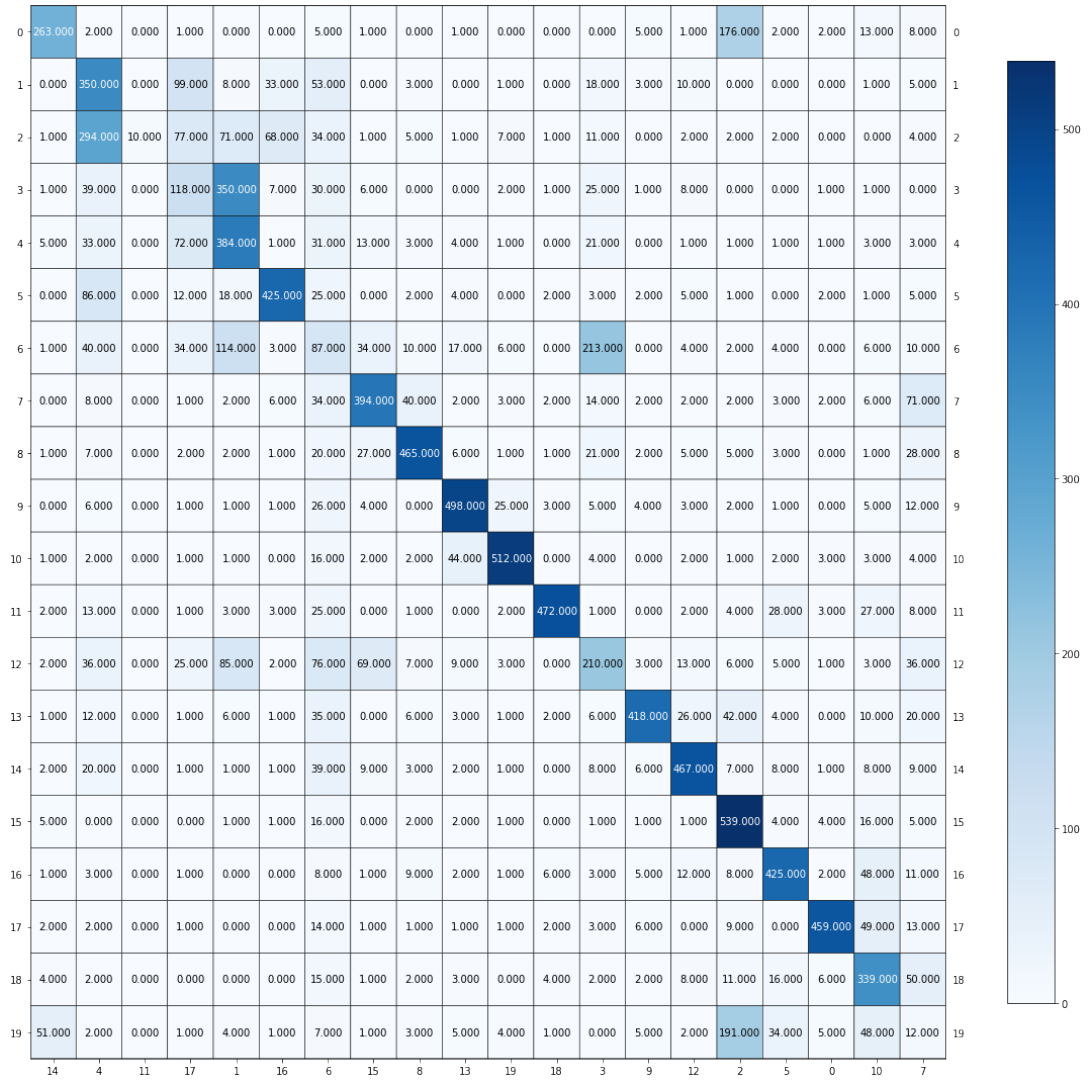


Figure 21: Contingency Matrix for Agglomerative Clustering: Ward

```
Homogeneity: 0.019
Completeness: 0.390
V-measure: 0.037
Adjusted Rand-Index: 0.001
Adjusted Mutual Information Score: 0.030
Contingency matrix:
```

Figure 22: Metrics for Agglomerative Clustering: Single

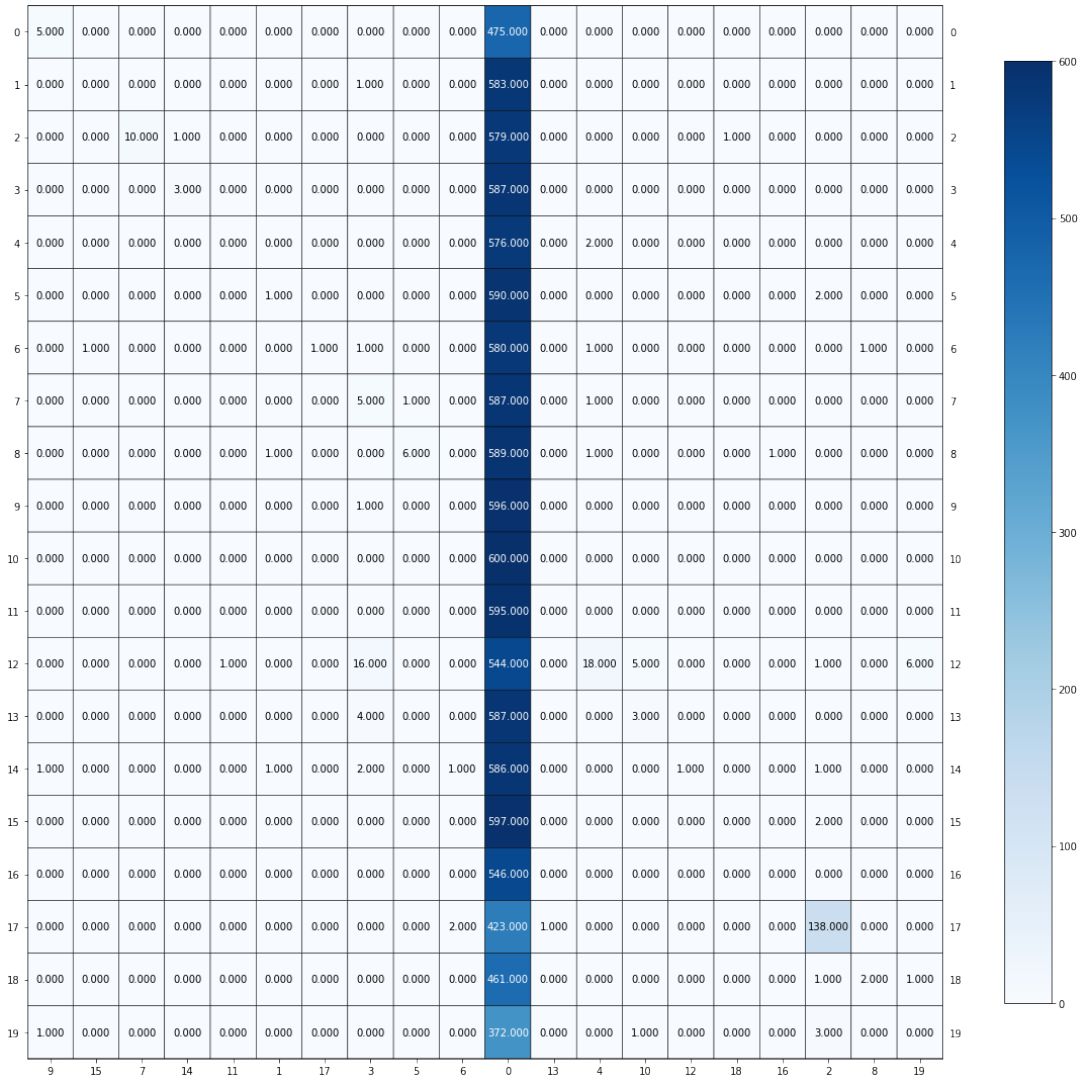


Figure 23: Contingency Matrix for Agglomerative Clustering: Single

3 Question 15

With DBSCAN we find best results with eps = 0.5 and min-samples = 100. With HDBSCAN we find best results with eps = 0.3 and min-samples =

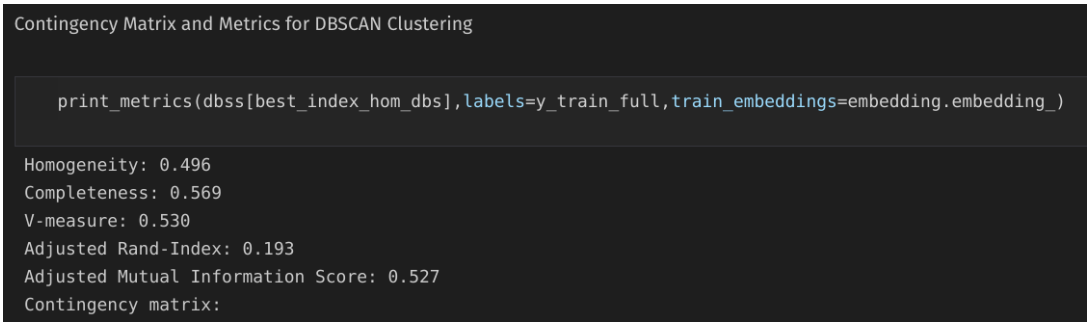


Figure 24: DBSCAN Metrics

Question 16

With the best *eps* and *min_samples*, we get 22 clusters with DBSCAN and 21 clusters with HDBSCAN. We observed that epsilon values lower than 0.5 gives many more clusters for DBSCAN, which results in a contingency martix with a large number of rows and clusters. Similarly for HDBSCAN epsilon values lower than 0.3 gives many more clusters.

For DBSCAN contingency matrix we observed a lot of data points belonging to a different cluster were grouped under a particular cluster, we have observed this even before with other clustering algorithms. Apart from this we observed some other discrepancies in clustering, those can be seen marked in the Fig.
25

For HDBSCAN contingency matrix also we observed a lot of data points belonging to a different cluster were grouped under a particular cluster. Apart from this other discrepancies in clustering observed were similar to that of DBSCAN contingency matrix, those can be seen marked in the Fig. 26

Question 17

We performed a grid search with the hyperparameters listed in the table (hyperparameter table). The top results from the grid search can be seen in the Fig 27
Hence we observe that we receive the best clustering results from Agglomerative Clustering and HDBSCAN both with UMAP reduction technique. The details of the hyperparameters choosen can be seen in the Fig 27.

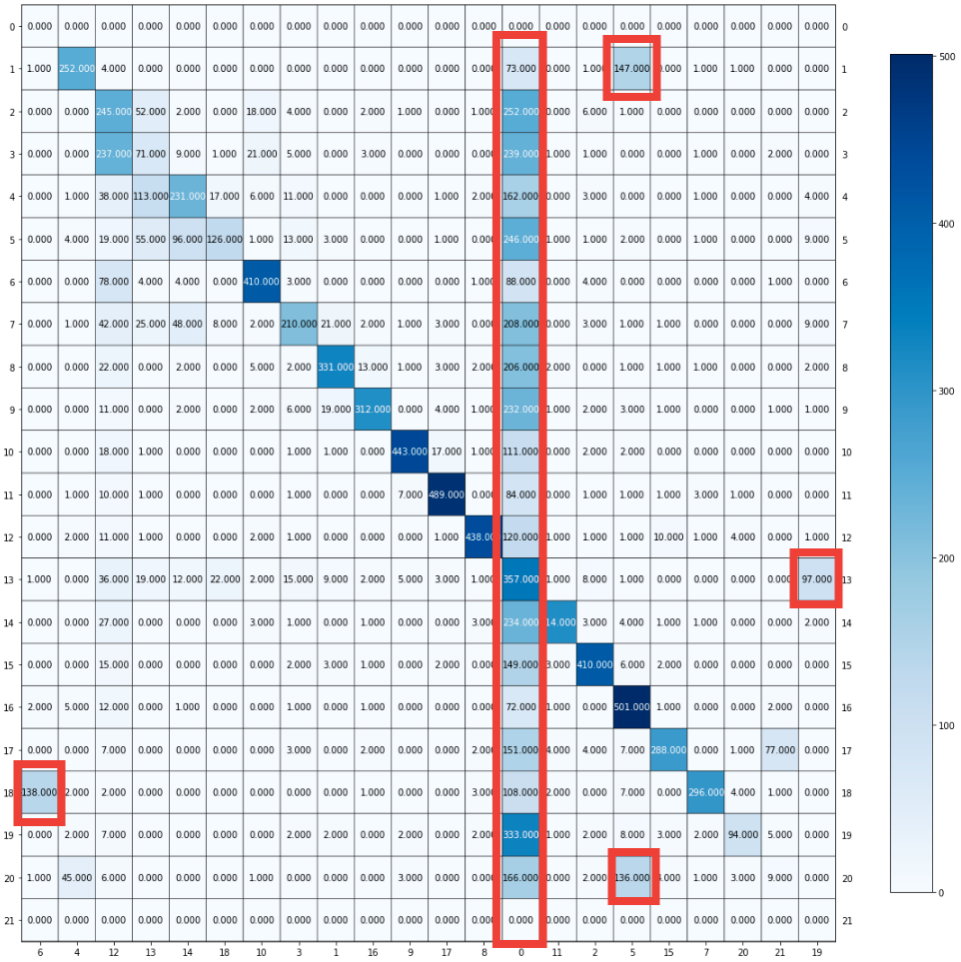


Figure 25: DBSCAN Contingency Matrix

4 Part 2

Question 19

VGG network can be used for transfer learning because often times transfer learning has two different types: (1) This model can do feature extraction; (2) This model can be fine-tuning. For the first one, we can use our pre-trained model as an arbitrary feature extractor. Send pictures through input layer, propagate forward and stop at the specified layer, use the output of this layer as the feature of the trained pictures.

Out VGG network is this find of model, therefore, even VGG network is trained on the dataset with perhaps totally different classes as targets, it can still be used as the arbitrary feature extractor to discriminate features of custom dataset.

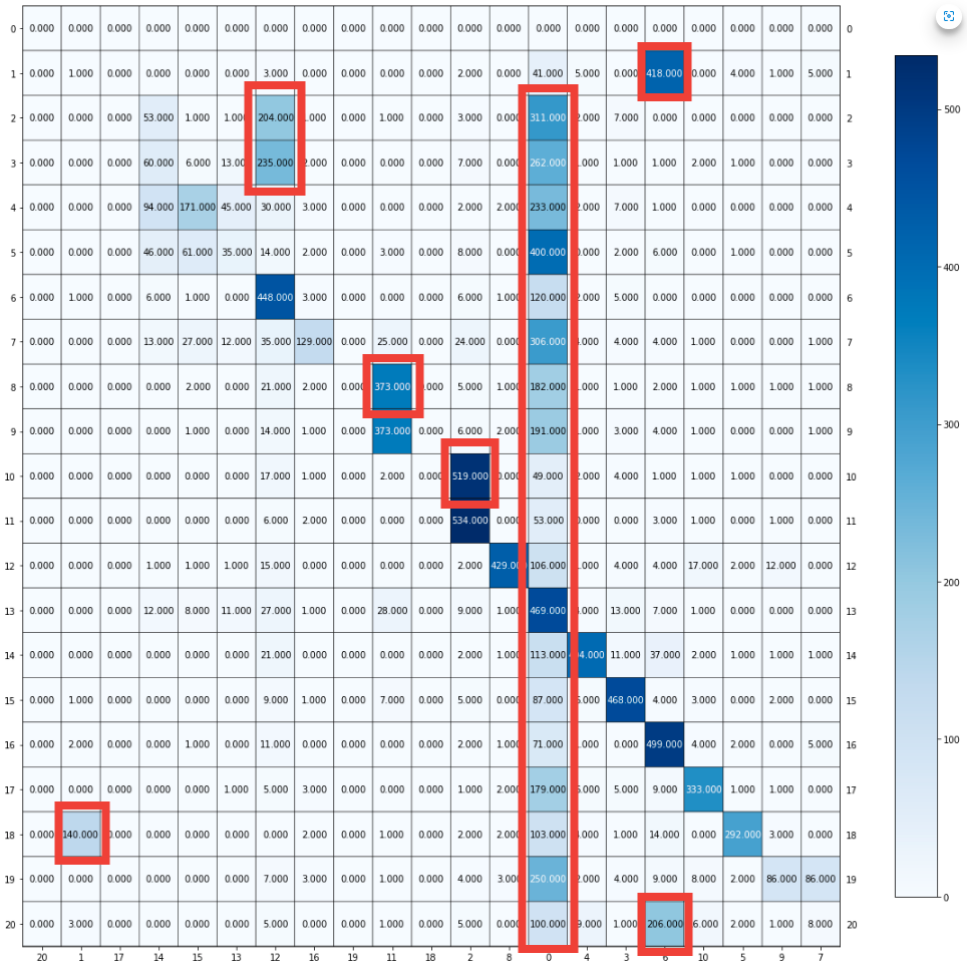


Figure 26: HDBSCAN Contingency Matrix

Question 20

- The first step of helper code is to do feature extraction using the kernal function from VGG 16 network. And it will keep doing so until it return out all the feature layers.
- After feature extraction, it conducts a max pooling on all the feature layers.
- Then it flatten the result from max pooling layer
- After flatten the results, it connects all the flatten layer together and output the final results.

Question 21

The pixels of the images in the original flower folder are not uniform. But after we input into the VGG network, we compress and resize them into 244 * 244 pixels input images.

```
rank_test_score:
1

mean_test_score:
0.9031402558567272

model parameters:
{'clf': AgglomerativeClusteringWrapper(n_clusters=20), 'reduce_dim': UMAP(metric='cosine', n_components=20), 'vect': CountVectorizer(max_df=0.7, min_df=3, stop_words='english')}

rank_test_score:
2

mean_test_score:
0.9024112251595886

model parameters:
{'clf': HDBSCANWrapper(cluster_selection_epsilon=0.3, min_cluster_size=200,
                min_samples=30), 'reduce_dim': UMAP(metric='cosine', n_components=20), 'vect': CountVectorizer(max_df=0.7, min_df=3, stop_words='english')}

rank_test_score:
3

mean_test_score:
0.898488400655385

model parameters:
{'clf': AgglomerativeClusteringWrapper(n_clusters=20), 'reduce_dim': UMAP(metric='cosine', n_components=5), 'vect': CountVectorizer(max_df=0.7, min_df=3, stop_words='english')}

rank_test_score:
4

mean_test_score:
0.8974808979933757

model parameters:
{'clf': HDBSCANWrapper(cluster_selection_epsilon=0.3, min_cluster_size=200,
                min_samples=30), 'reduce_dim': UMAP(metric='cosine', n_components=200), 'vect': CountVectorizer(max_df=0.7, min_df=3, stop_words='english')}
```

Figure 27: Results from the Grid Search

There are 4096 features extracted by VGG per image.

Question 22

Compared to the sparse TF-IDF, the extracted features are dense.

Question 23

One of the most major differences between PCA and t-SNE is it preserves only local similarities whereas PA preserves large pairwise distance maximize variance. It takes a set of points in high dimensional data and converts it into low dimensional data.

For t-SNE, it has several features:

- It takes a set of points in high dimensional data and converts it into low dimensional data.
- It's a non-linear method and adapts to the underlying data performing different transformations in different regions.

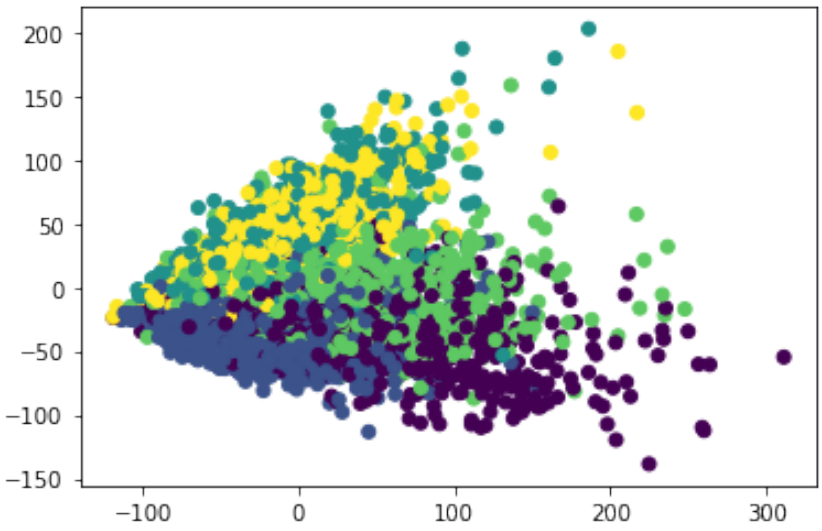


Figure 28: PCA plot

- It's incredibly flexible and often finds a structure where other dimensionality reduction algorithms can't.

Therefore, as you can see from the Fig. 29, it distribute the high-dimensional data more equal in the 2d space while PCA only reduce data along PCA 1st dimension.

Question 24

The results are shown in the following table:

Methods	None	SVD	UMAP	Autoencoder
k-Means	0.7043	0.7043	**0.7945**	0.6537
Agglomerative Clustering	0.6862	0.6529	0.7922	0.6227
HDBSCAN	0.3785	0.4135	0.6782	0.6194

Table 1: Accuracy of different methods combination

So the best result is: **UMAP + K-Means** algorithm, the best result is about 0.7922

Question 25

The test accuracy of the MLP classifier on the original VGG features is **0.9186**.

The accuracy of reduced-dimension features are:

- SVD: 0.9073
- UMAP: 0.8787

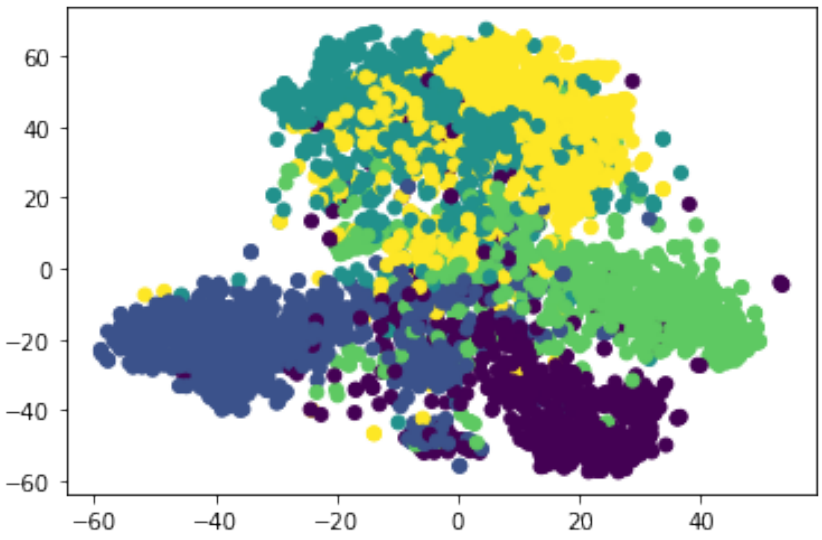


Figure 29: t-SNE plot

- Autoencoder: 0.8181

The performance of the model will suffer with the reduced-dimension representations. It's significant, especially for the Autoencoder, the difference is about 10%.

Comparing with previous results in the Table 1, you can see that original VGG features perform is not always the worst, and it performs pretty well in K-Means and Agglomerative Clustering, which indicates that reduced-dimension features don't always mean you can do better in Clustering, especially for Image Clustering.