**ECE 219: Large-Scale Data Mining: Models and Algorithms**

Prof. Vwani Roychowdhury, Spring 23

Project 2: Data Representations and Clustering

Authors: Erik Ebrahimi, Kelvin Vasquez, Diane Leal-Delgado

# Introduction

In this project we convert raw data into feature representations to be used in classification models or other downstream algorithms. Here we first applied these techniques on textual data and then on images. For the textual data, we used unsupervised learning to see if a combination of feature engineering and clustering techniques can automatically separate a document set into groups that match known labels. We then looked at how to obtain image features using deep neural networks for unsupervised learning with clustering.

To measure the performance of the different combinations we looked at the common clustering evaluation metrics: homogeneity, completeness, V-measure, adjusted rand index, and adjusted mutual information score.

# Part 1 - Clustering on Text Data

## Clustering with Sparse Text Representations

### *QUESTION 1:* Report the dimensions of the TF-IDF matrix you get

TF-IDF Matrix Shape: (7882, 23522)

*QUESTION 2:* Report the contingency table of your clustering result

Contingency Table: [ [3170, 733] , [47 3932] ]

Does the contingency matrix have to be square-shaped?

No, if the dataset is balanced i.e. if the number of unique labels which if they only appear at least once in both the true and predicted labels then the matrix will be square-shaped

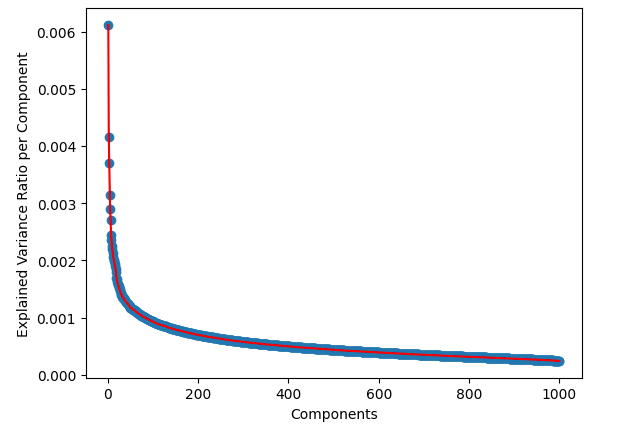
*QUESTION 3:* Report the 5 clustering measures explained in the introduction for Kmeans clustering.

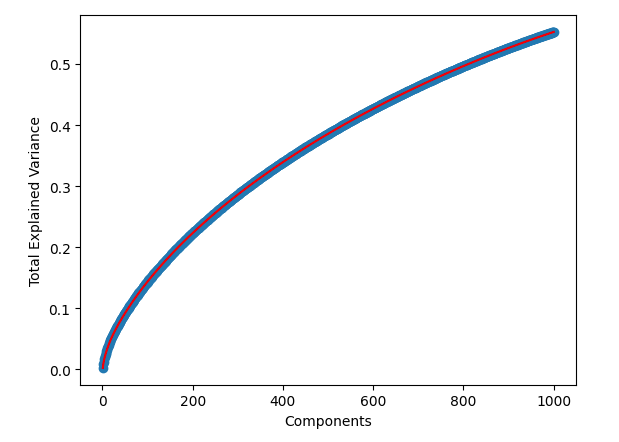
Table 1: Clustering Measure

| **Clustering Measure** | **Value** |
| --- | --- |
| Homogeneity Score | 0.58375 |
| Completeness Score | 0.59836 |
| V-Measure Score | 0.59096 |
| Adjusted Random Score | 0.94328 |
| Adjusted Mutual Info Score | 0.59092 |

## Clustering with Dense Text Representations

### *QUESTION 4:* Report the plot of the percentage of variance that the top r principal components retain v.s. r, for r = 1 to 1000.

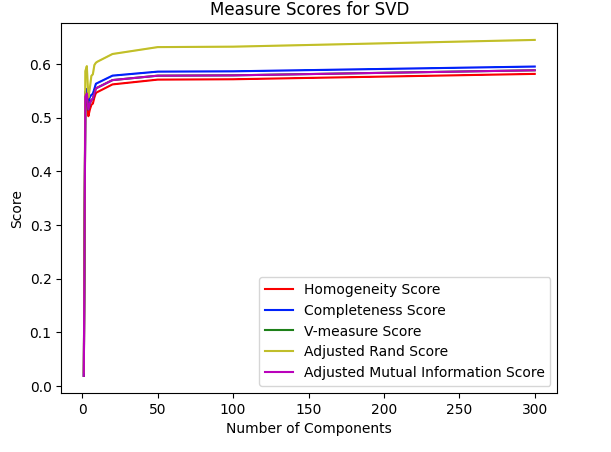
Figure 1: Explained Variance Ratio Per Component and Total Explained Variance



### *QUESTION 5:* Let r be the dimension that we want to reduce the data to (i.e. n components). Try r = 1 − 10, 20, 50, 100, 300, and plot the 5 measure scores v.s. r for both SVD and NMF. Report a good choice of r for SVD and NMF respectively.

Best SVD: 3

Best NMF: 2

Figure 2: Measure Scores for SVD

SVD:

[0.018956544515354657, 0.5299872319287099, 0.5367850707975124, 0.5024767689891753, 0.5148164148429758, 0.5236156697734383, 0.5260710703947462, 0.537327954793755, 0.5458349201631516, 0.5474215248195626, 0.561687101914137, 0.5706032817609463, 0.5712651570193955, 0.5812549696357695]

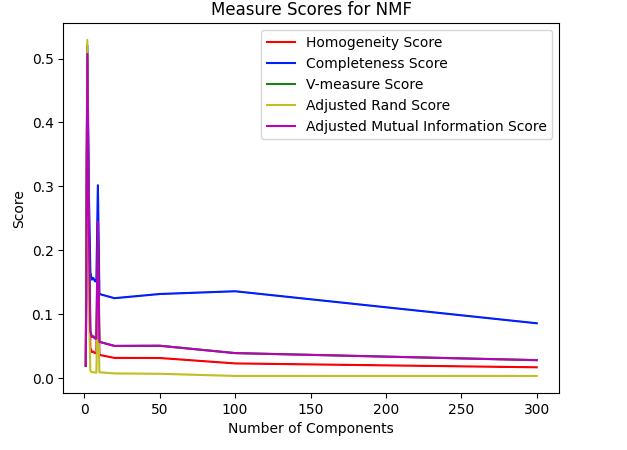
[0.019260434831635448, 0.5474718149871534, 0.5532677491688455, 0.5278865768783753, 0.5380451727509755, 0.5420889406344178, 0.5442693539631456, 0.5533584393831902, 0.5628584861030622, 0.5642746120047634, 0.5780055102632735, 0.5854624086370032, 0.5860582025920705, 0.5949452135583514]

[0.01910728144974337, 0.5385876569779003, 0.544901792771372, 0.51486835708284, 0.5261745515708813, 0.532692194356621, 0.5350155055466917, 0.5452253921733549, 0.5542160079401561, 0.555720323726244, 0.5697294805305662, 0.577937351641233, 0.5785671365670185, 0.5880204186037479]

[0.025920756456215145, 0.5863908629238136, 0.5957562038017276, 0.5369512180516358, 0.5545743626399822, 0.577485357244855, 0.5805751990281313, 0.5977166467666782, 0.6020410119571624, 0.6040117662250577, 0.6182962629417924, 0.6311336913379769, 0.6319404101570526, 0.6445104262991748]

[0.01901676579673226, 0.5385447284899507, 0.5448594978783622, 0.5148228469326553, 0.526130214873496, 0.5326486699833842, 0.5349722117081862, 0.5451831449965414, 0.5541745691278109, 0.5556790326817868, 0.5696895246650355, 0.5778982146345757, 0.5785280607032259, 0.587982262398501]

Figure 3: Measure Scores for NMF

NMF:

[0.018956544515354657, 0.49465130975968324, 0.19951876419783976, 0.04692504995474571, 0.040791321221174695, 0.04127230889715677, 0.039056839016831595, 0.038657486694680235, 0.20604076139141206, 0.03613763583869197, 0.03147448959307454, 0.03129702733698987, 0.022804025728248495, 0.01673497495346444]

[0.019260434831635448, 0.52035408497699, 0.2947765044994327, 0.16542651155887647, 0.1539550035833579, 0.15643112840263684, 0.1515933118619265, 0.15104055673150027, 0.30168934811342524, 0.13144159172247633, 0.12492257196180623, 0.131468501050522, 0.13576445094529882, 0.08565178315318572]

[0.01910728144974337, 0.5071772642931544, 0.23796887252143872, 0.07311128077804721, 0.06449444436067417, 0.06531271221930507, 0.06211120788658438, 0.061559394148176426, 0.24485568937241323, 0.056689464975497386, 0.05028066578824123, 0.05055828850363574, 0.03904907327465802, 0.027999332575704056]

[0.025920756456215145, 0.5291689826328091, 0.11951375550462902, 0.011097673265433138, 0.00915047637987103, 0.009150507416154589, 0.008527044719464024, 0.00838626681667614, 0.13620772121950603, 0.008955992342204715, 0.007171111547812483, 0.006578312155606838, 0.0032510606622685972, 0.0032503799797708854]

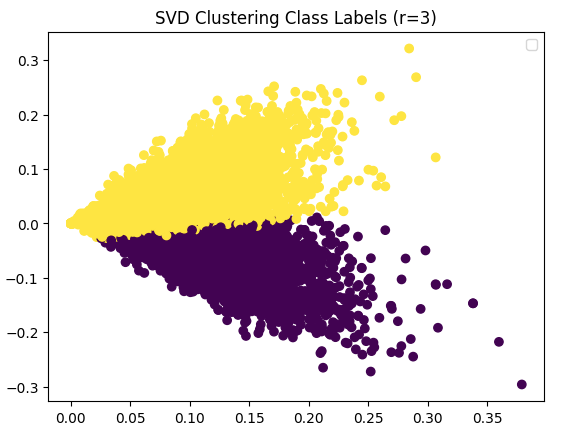
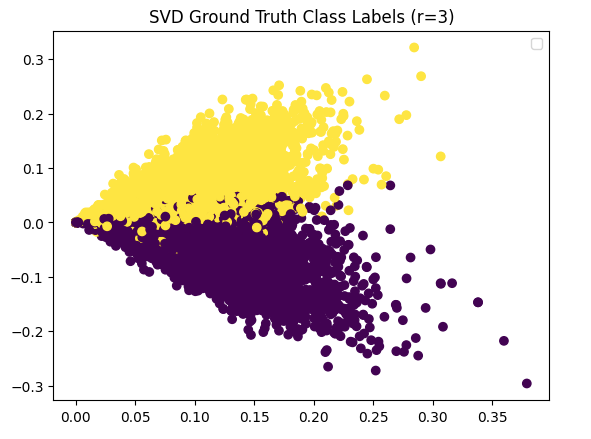
[0.01901676579673226, 0.5071310027576791, 0.2378856474417694, 0.07297891744709399, 0.06435885802704513, 0.0651771234127268, 0.061974478155877234, 0.061422396859528675, 0.24477351627233143, 0.05655382671969904, 0.05014157581213706, 0.05041766181352057, 0.038898058076244, 0.027850152623569827]

### *QUESTION 6:* The measures exhibit a non-monotonic trend as r grows, showing an initial peak followed by a decrease and then stabilization. This pattern is observed across all measures, as the number of components increases and the dimensions for k-means clustering expand. Due to the Euclidean distance metric's unsuitability in high dimensions, k-means is known to suffer from the curse of dimensionality, as the nearest and farthest points converge in distance. Consequently, clustering becomes difficult, and increasing the number of components beyond the elbow point does not provide additional information to the k-means algorithm. Consequently, the measures do not change beyond the elbow point, as no new information is provided to the algorithm.

### *QUESTION 7:* On average, they are miniscule better than those compared in Question #3

### *QUESTION 8:*

Figure 4: Best SVD and NMF Clustering Results



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### *QUESTION 9:* The data visualized here does not reflect the usual K-Mean clustering because the data points overlap each other. An ideal K-Mean clustering algorithm should have the data points in well-separated clusters

## Clustering of the Entire 20 Classes

### *QUESTION 10:*

Contingency Table for all 20 Categories: [[ 1 143 1 0 97 0 93 5 0 0 0 279 2 0 0 0 0 172

6 0]

[ 0 0 0 0 278 37 32 134 0 0 0 10 0 333 29 55 6 44

0 15]

[ 0 0 0 0 153 62 26 74 20 0 0 9 0 43 415 96 2 52

0 33]

[ 3 0 3 0 157 187 42 102 217 0 0 3 0 5 46 168 12 36

0 1]

[ 0 0 1 0 137 522 18 63 97 0 0 4 0 5 6 46 8 55

0 1]

[ 0 0 5 0 200 11 22 101 1 0 0 2 0 33 58 6 7 27

0 515]

[ 22 0 0 8 228 61 18 72 52 0 0 8 0 1 12 28 435 29

0 1]

[445 0 0 0 233 0 118 42 2 0 0 28 0 0 2 0 9 108

1 2]

[ 39 0 0 2 402 0 305 38 5 0 0 31 0 0 0 0 9 164

0 1]

[ 0 0 0 382 262 0 89 45 0 0 0 20 0 1 0 0 4 191

0 0]

[ 0 0 0 699 166 0 19 20 0 0 0 17 0 0 0 0 12 66

0 0]

[ 1 0 466 0 149 21 127 27 0 0 0 109 0 4 6 1 1 74

4 1]

[ 29 0 2 0 562 87 68 98 7 0 0 9 0 6 4 10 7 94

0 1]

[ 1 4 0 0 495 1 100 64 0 78 0 155 0 1 1 0 0 90

0 0]

[ 0 0 0 0 609 4 75 27 1 0 0 75 0 12 0 0 6 178

0 0]

[ 0 494 0 0 236 0 18 26 0 2 0 136 1 0 1 0 0 80

3 0]

[ 4 0 1 0 114 0 115 8 0 0 0 327 0 1 0 3 2 88

246 1]

[ 0 4 0 0 149 0 42 3 0 0 183 124 349 0 0 0 3 83

0 0]

[ 1 2 2 0 118 0 108 8 0 116 0 257 0 0 0 0 0 87

76 0]

[ 1 146 0 0 120 0 71 5 0 2 0 140 1 1 0 0 0 80

61 0]]

Table 2: Clustering Measure for All 20 Categories

| **Clustering Measure** | **Value** |
| --- | --- |
| Homogeneity Score | 0.34724 |
| Completeness Score | 0.39735 |
| V-Measure Score | 0.37061 |
| Adjusted Random Score | 0.12499 |
| Adjusted Mutual Info Score | 0.36843 |

## SVD

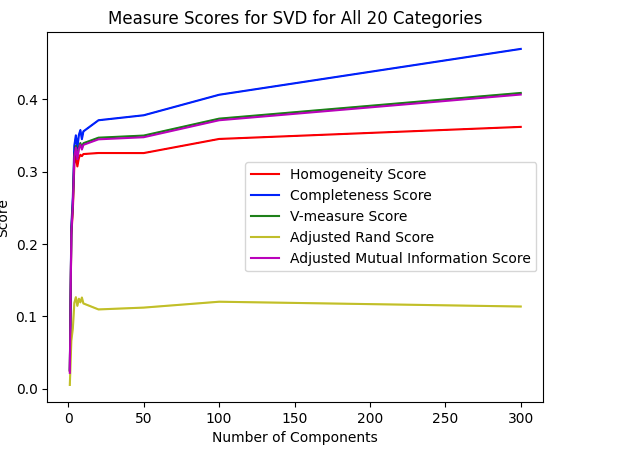
## [0.024149681169492516, 0.2120580832490991, 0.24656045181570196, 0.3104742493602024, 0.3222693294024222, 0.30725550795690626, 0.31893871829390347, 0.3233506373471898, 0.3215423220096135, 0.3243552195460371, 0.3257929450149544, 0.3257090407243409, 0.3452402495223717, 0.36191649318848534]

## [0.026119008338653597, 0.22397963803620416, 0.2646113145329956, 0.33599568656574297, 0.35021598223661476, 0.3327261642546189, 0.35067662717308135, 0.3575276012625997, 0.3449097472898532, 0.3556957906588993, 0.37109346136931143, 0.37799030478686835, 0.40634707729767505, 0.4697474743912917]

## [0.025095769554111184, 0.21785588911335022, 0.2552671706922745, 0.3227311983978007, 0.3356619625381059, 0.3194839822688438, 0.33405552833683094, 0.3395813559073891, 0.3328163753588548, 0.339303330307533, 0.3469708421565527, 0.34990755742651475, 0.3733095579258614, 0.4088414678120248]

## [0.005060627159850475, 0.06605150490694527, 0.08324001178481244, 0.11854248377857042, 0.1264150636753105, 0.11440722958073675, 0.12428521262990547, 0.11926538624749801, 0.12592967782997513, 0.1177473349957544, 0.10947162775230299, 0.11199029241237067, 0.12012056522883684, 0.1134968550097584]

## [0.021783816938750818, 0.21525590036633344, 0.25276387913221643, 0.32044627903782286, 0.33342030586211674, 0.31719341258061073, 0.3317955790196139, 0.3373334830195454, 0.3305826368689192, 0.3370657741321096, 0.34471665478215097, 0.347639896383843, 0.37110717115155656, 0.4066638580904045]

Figure 5: Measure Scores for SVD for All 20 Categories

NMF

[0.02414477211123332, 0.1912056689667014, 0.2186383223258225, 0.24690551862591745, 0.297368780469481, 0.2595136470872969, 0.2976486521884839, 0.2893767865520299, 0.29219319923278886, 0.3236107770523622, 0.31954449081682096, 0.21490892360487884, 0.15543395044809705, 0.07590953930290195]

[0.026109012106166563, 0.20432644325820276, 0.2366967277284554, 0.2940512708055242, 0.3325269314588892, 0.2884623775541164, 0.3314271642984859, 0.31883761125442195, 0.3346634883411802, 0.3567531327995482, 0.3856259773255632, 0.286956578439138, 0.20522411781692143, 0.10709875443742181]

[0.02508850456418185, 0.1975484319136985, 0.22730943047056248, 0.26842395895300764, 0.3139666653023136, 0.27322335387037666, 0.3136310319573026, 0.3033936839031903, 0.3119896373866978, 0.33937472828697013, 0.3494889878041665, 0.2457611816015412, 0.17689217664233675, 0.08884643360256633]

### [0.005069466603507795, 0.05764558714976883, 0.06473139112283063, 0.07629393223527081, 0.10347936842992879, 0.0919140141917494, 0.11063451550057324, 0.10106333313788916, 0.09379266322356809, 0.1219977446311461, 0.09765643321750891, 0.05066400252971902, 0.038173543331083595, 0.010112246619928838]

### [0.021778778227128114, 0.19485515419845628, 0.22470451667154936, 0.265804730751165, 0.31161679501616585, 0.2707383828215484, 0.31128331339277154, 0.3010271459124649, 0.3096072862529095, 0.3371292471247677, 0.3471709421642408, 0.2429321863990534, 0.17377954462259926, 0.08527366363421271]

Figure 6: Measure Scores for NMF for All 20 Categories

### 

We set the n\_componets=300 for SVD and set n\_components=20 for NMF based on the scores from above. From here we plotted the contingency matrix for each and their respective scores can be seen in *Table 3*.

Figure 7: NMF contingency matrix for All 20 Categories

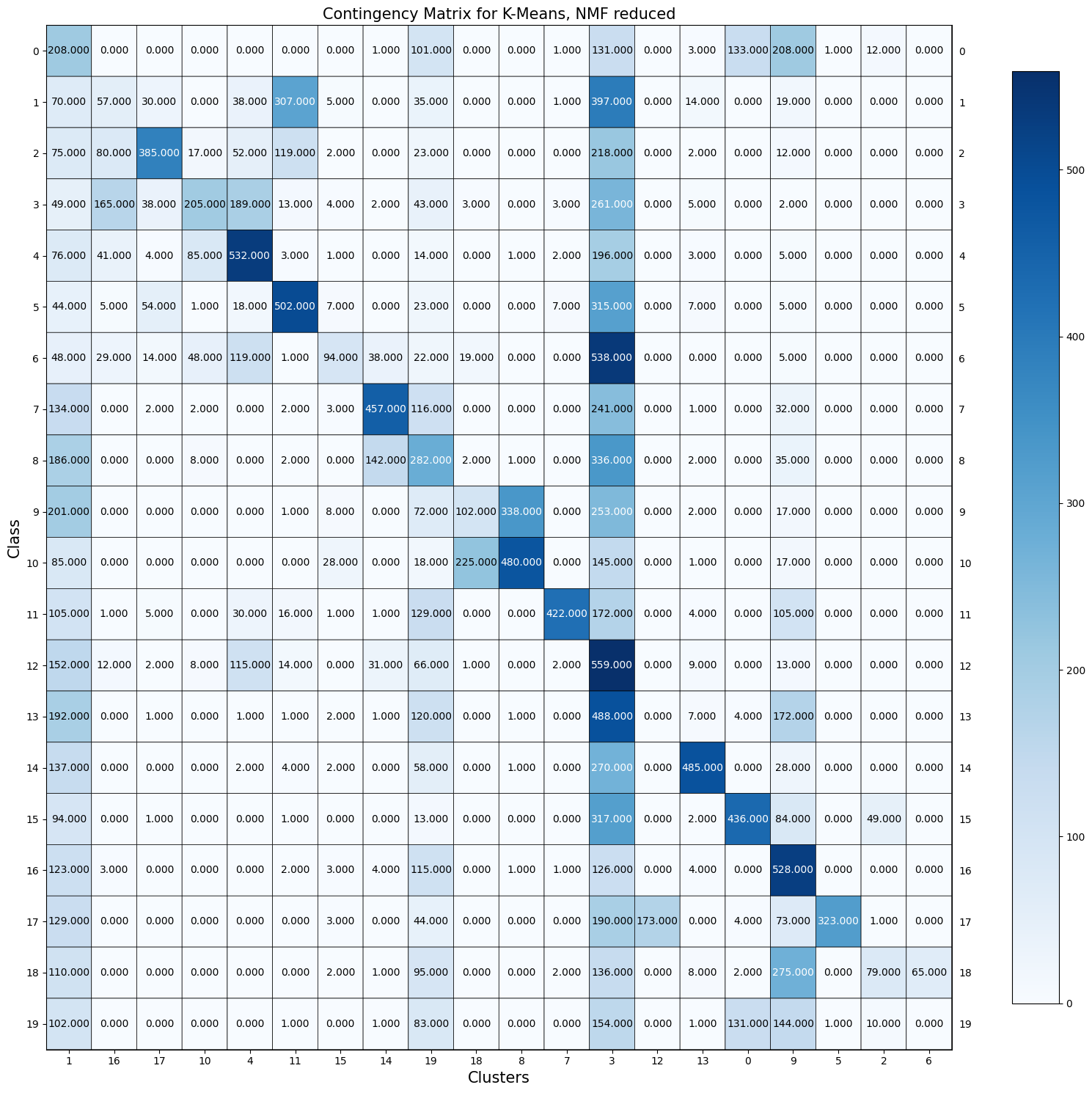


Figure 8: SVD contingency matrix for All 20 Categories

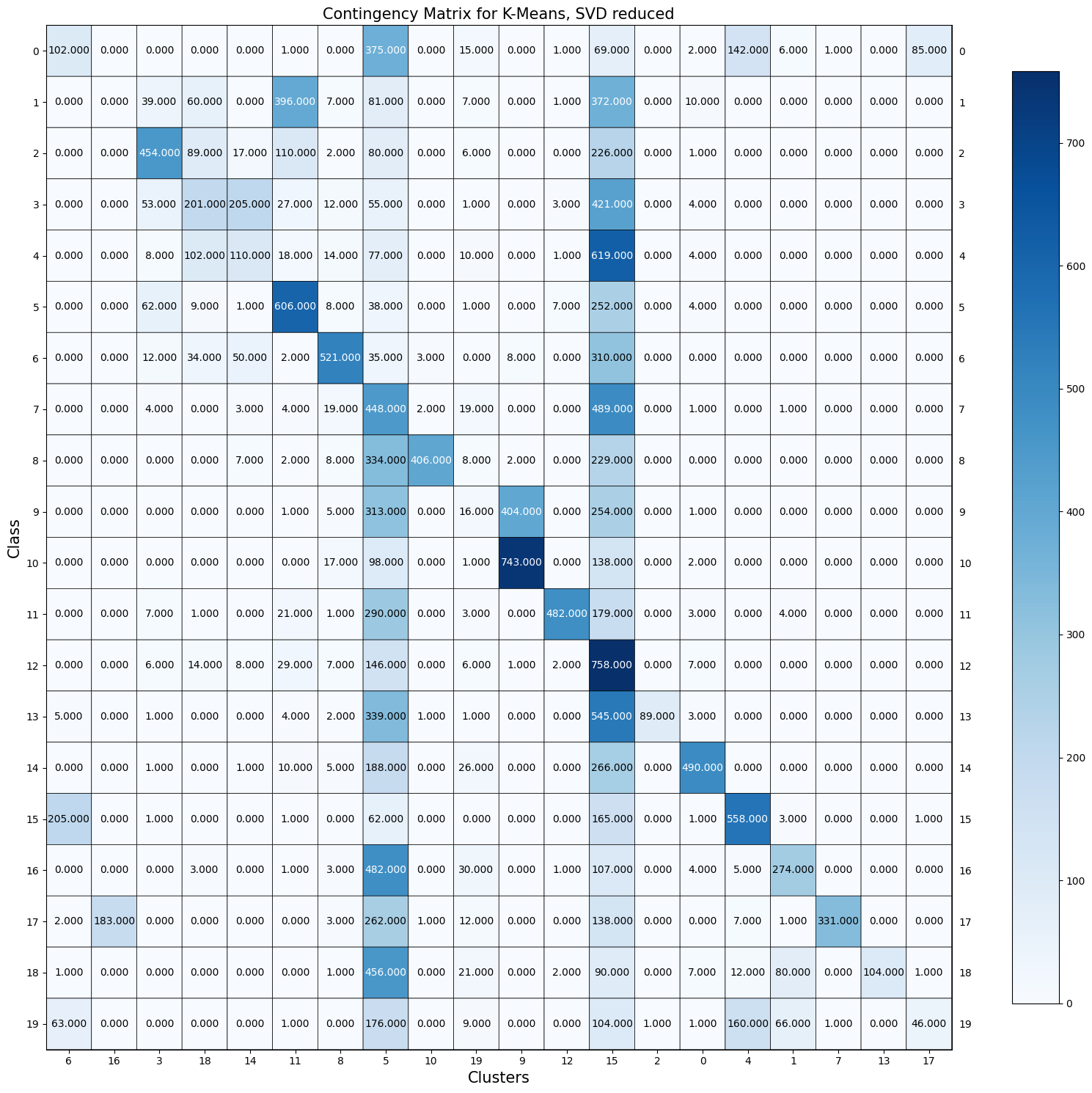


Table 3: NMF and SVD Clustering Measure for All 20 Categories

| **Clustering Measure** | **NMF Value** | **SVD Value** |
| --- | --- | --- |
| Homogeneity Score | 0.3195 | 0.3619 |
| Completeness Score | 0.3856 | 0.4697 |
| V-Measure Score | 0.3494 | 0.4088 |
| Adjusted Random Score | 0.0976 | 0.1134 |
| Adjusted Mutual Info Score | 0.3471 | 0.4066 |

### *QUESTION 11*

To improve the clustering performance for the 20 categories data, we used UMAP for dimensionality reduction. We considered different combinations for UMAP and tried the following settings: n\_components = [5, 20, 200] and metric = ‘cosine’ vs ‘euclidean’. *Figures 9* to *14* show the different contingency matrices for the different settings tried and *Table 4* summarizes the five clustering evaluation metrics for the different combinations.

Table 4: UMAP different combination’s clustering evaluation metrics

| **UMAP**  **Combinations** | **Homogeneity score** | **Completeness score** | **V-measure score** | **Adjusted Rand Index score** | **Adjusted mutual information score** |
| --- | --- | --- | --- | --- | --- |
| n\_components= 5  metric = ‘cosine’ | 0.5666 | 0.5869 | 0.5766 | 0.4444 | 0.5752 |
| n\_components= 20  metric = ‘cosine’ | 0.5786 | 0.605 | 0.5915 | 0.4711 | 0.5901 |
| n\_components= 200  metric = ‘cosine’ | 0.5722 | 0.5998 | 0.5857 | 0.4506 | 0.5843 |
| n\_components= 5  metric = ‘euclidean’ | 0.0074 | 0.0076 | 0.0075 | 0.001 | 0.0043 |
| n\_components= 20  metric = ‘euclidean’ | 0.0076 | 0.0079 | 0.0078 | 0.0008 | 0.0045 |
| n\_components= 200  metric = ‘euclidean’ | 0.006 | 0.0062 | 0.0061 | 0.0008 | 0.0028 |

Figure 9: UMAP contingency matrix for n\_components = 5, metric = ‘cosine’

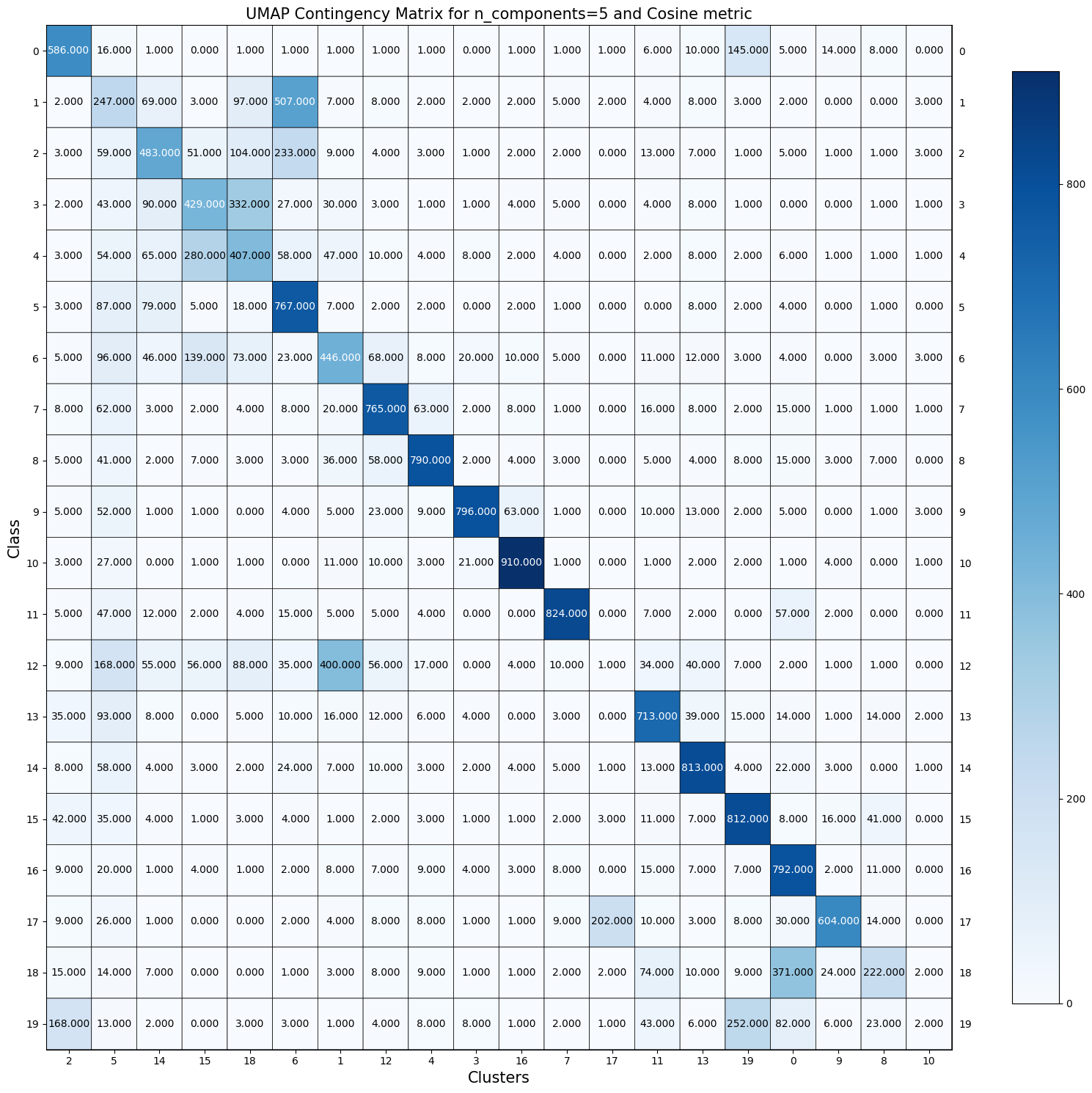


Figure 10: UMAP contingency matrix for n\_components = 20, metric = ‘cosine’

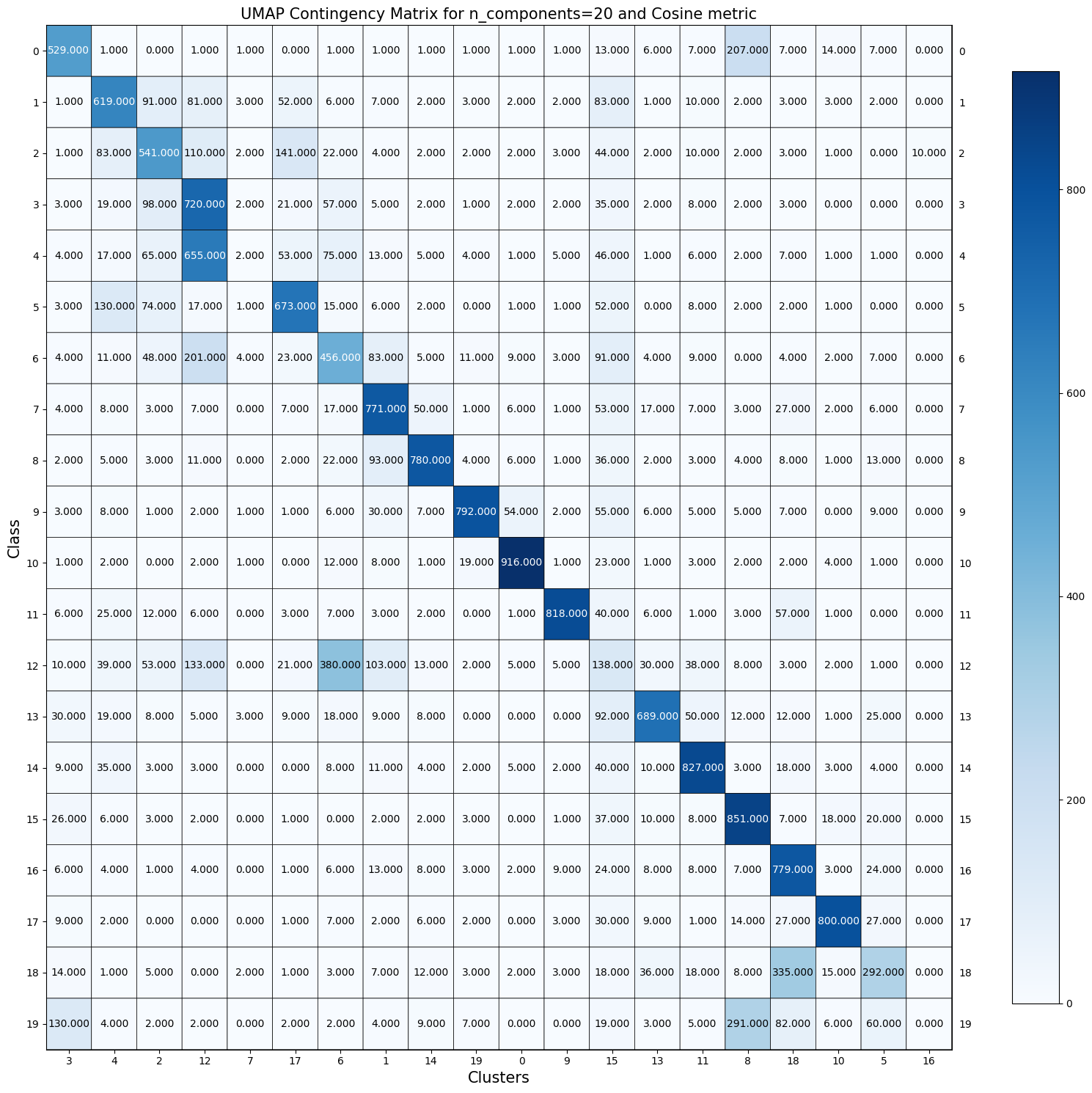


Figure 11: UMAP contingency matrix for n\_components = 200, metric = ‘cosine’

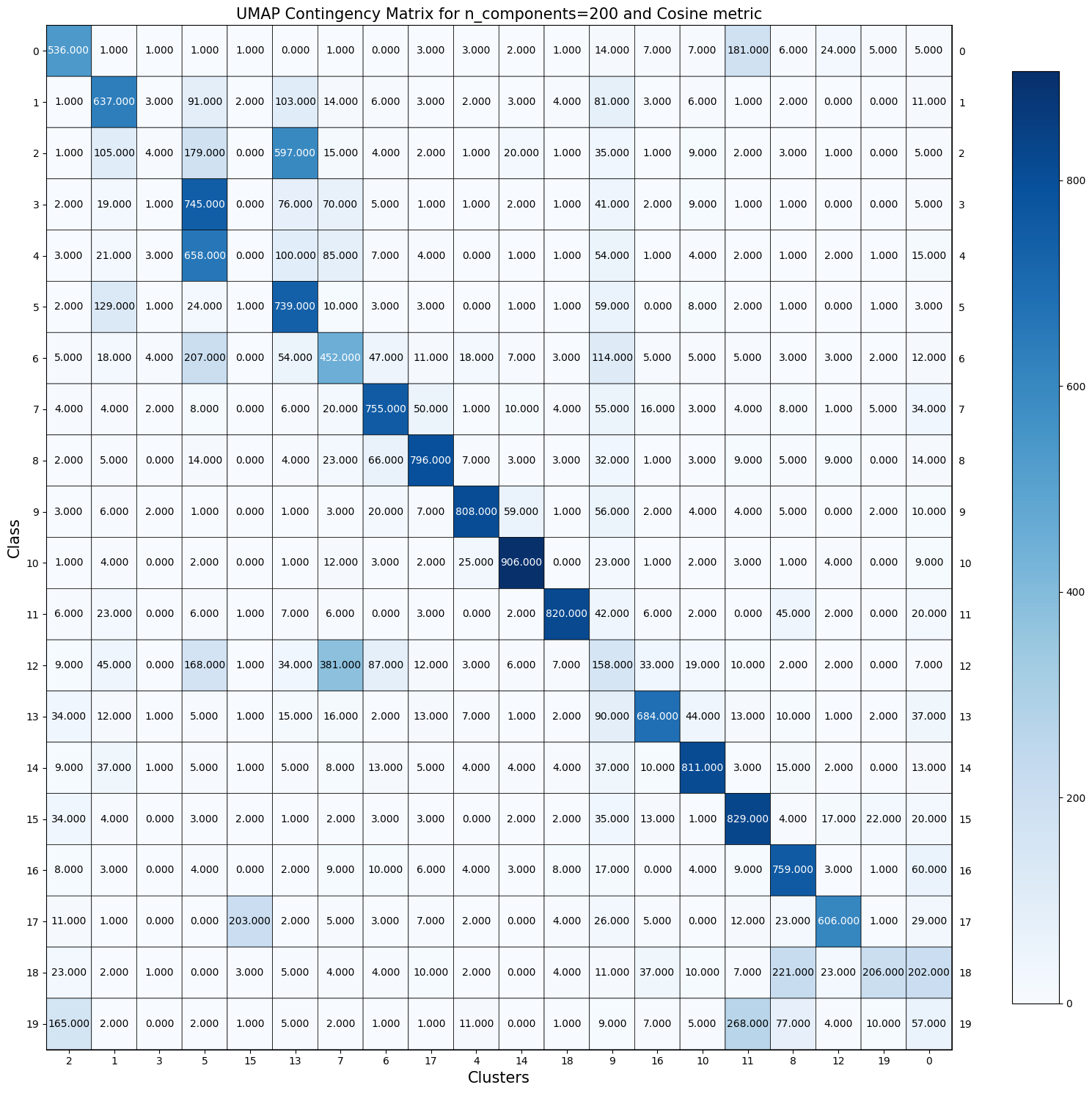


Figure 12: UMAP contingency matrix for n\_components = 5, metric = ‘euclidean’

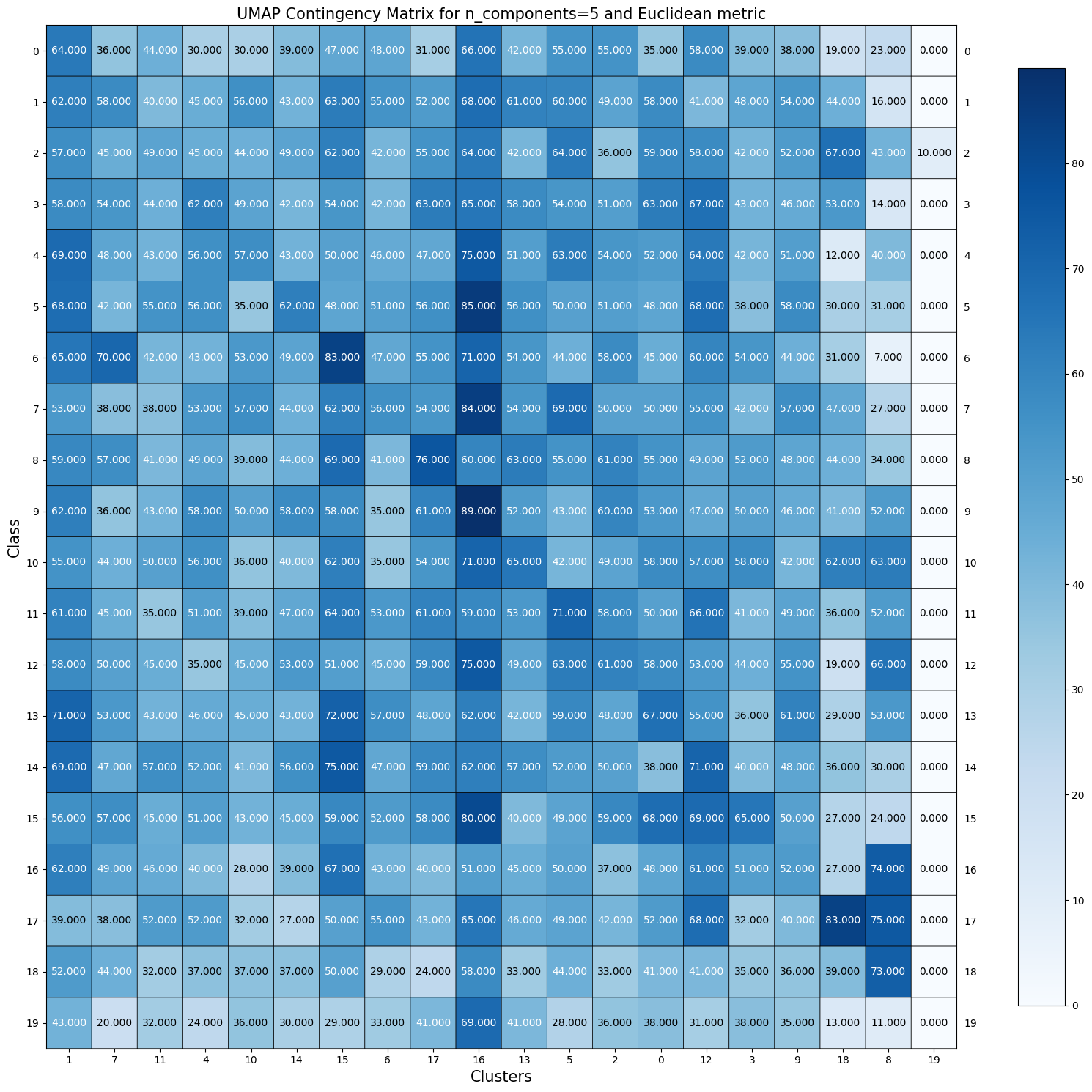


Figure 13: UMAP contingency matrix for n\_components = 20, metric = ‘euclidean’

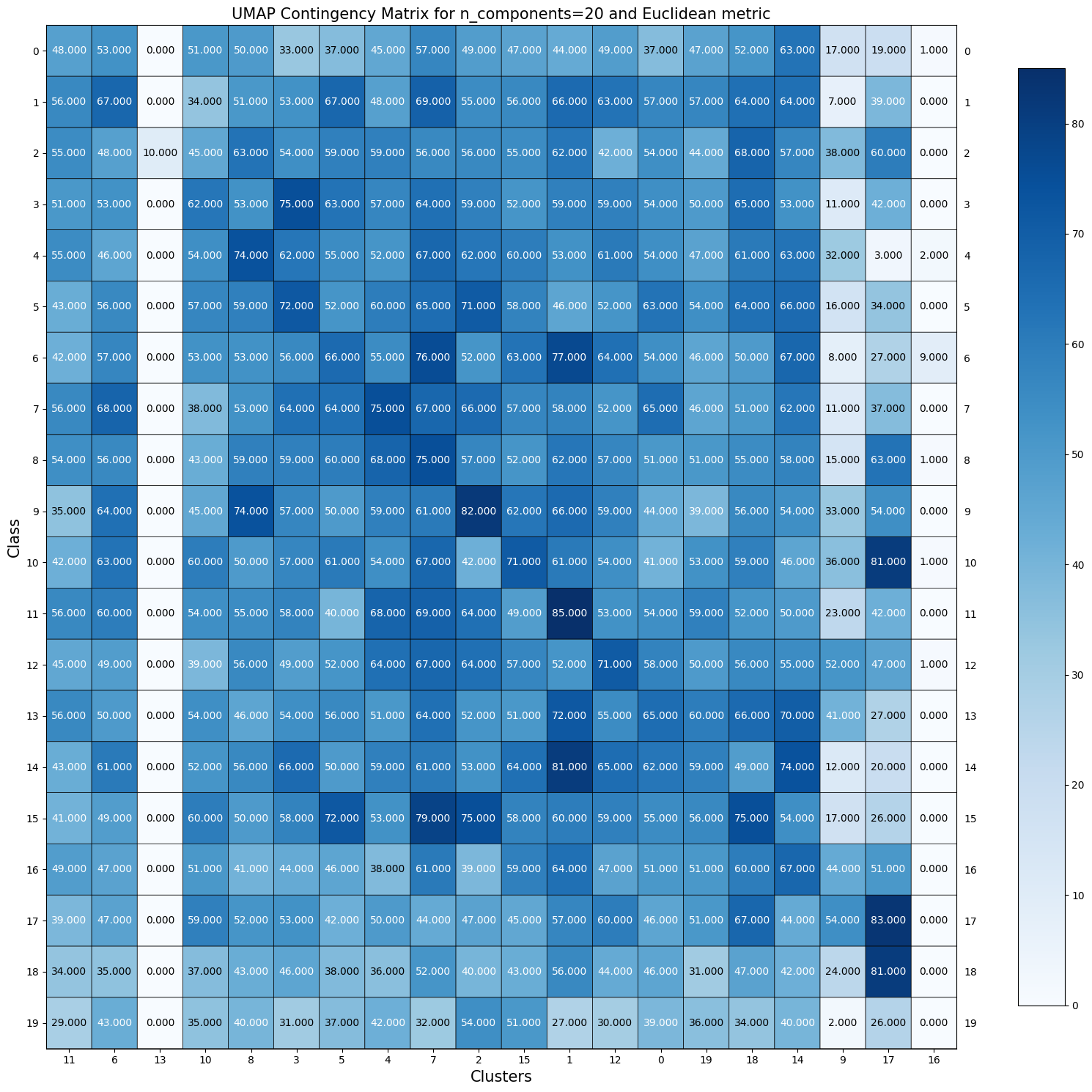
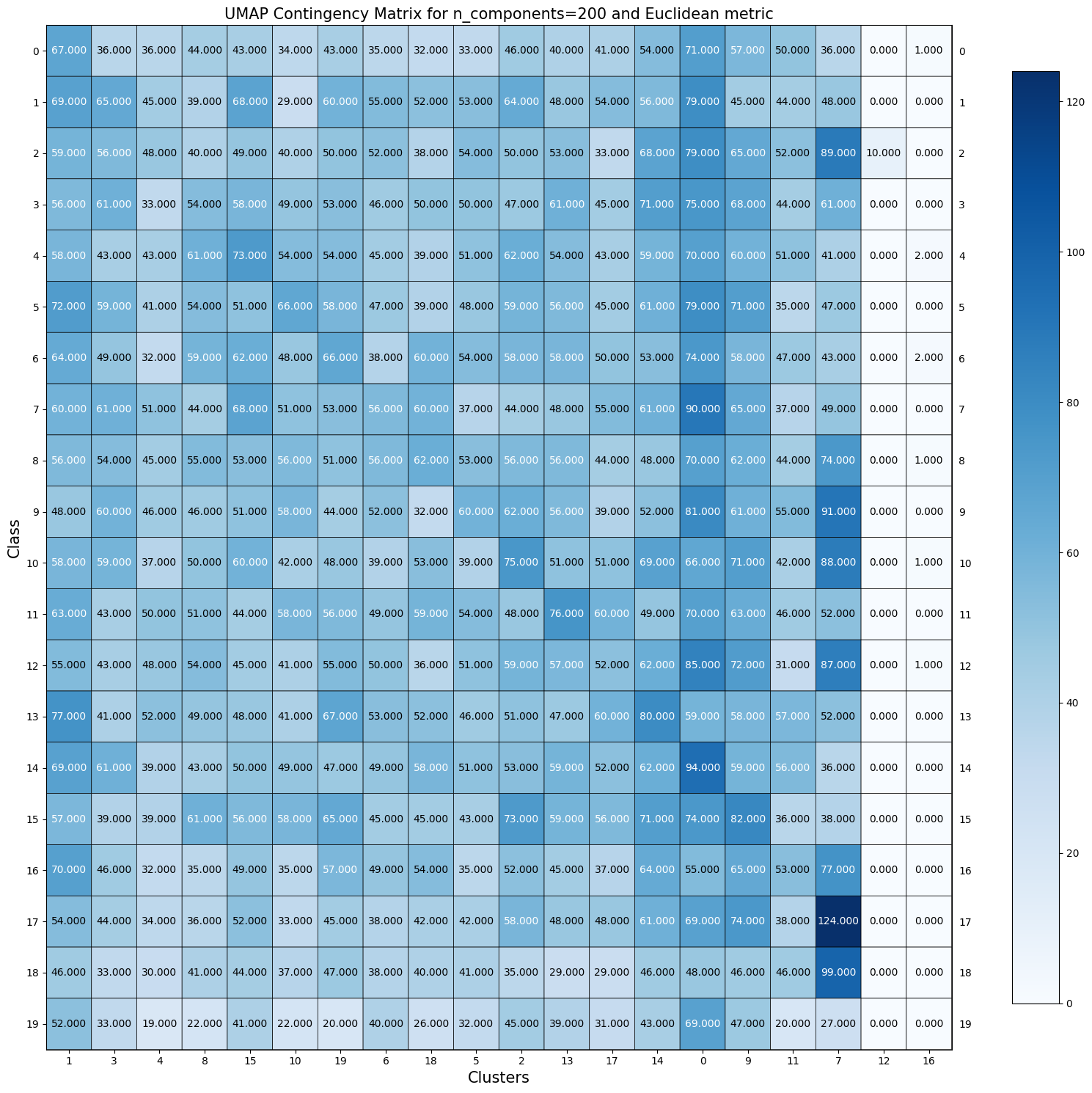


Figure 14: UMAP contingency matrix for n\_components = 200, metric = ‘euclidean’



### *QUESTION 12*

Analyzing the contingency matrices shown in *Question 11* we found that for the Euclidean metric all of the settings result in poor performance, but 20 n\_components seems to perform best. Setting n\_components to 20 for the cosine metric also works best. This is because it is beneficial to choose an embedding dimension (n\_components) which is closer to the dimension of the underlying manifold on which the data lies, which we know is 20 in our dataset. We also see that the cosine metric outperforms the Euclidean metric. This is because the cosine distance in UMAP mostly correctly identifies the similarity between the TF-IDF vector of documents while the Euclidean distance usually fails to correctly identify the similarities since it compares the magnitudes of the document in terms of length.

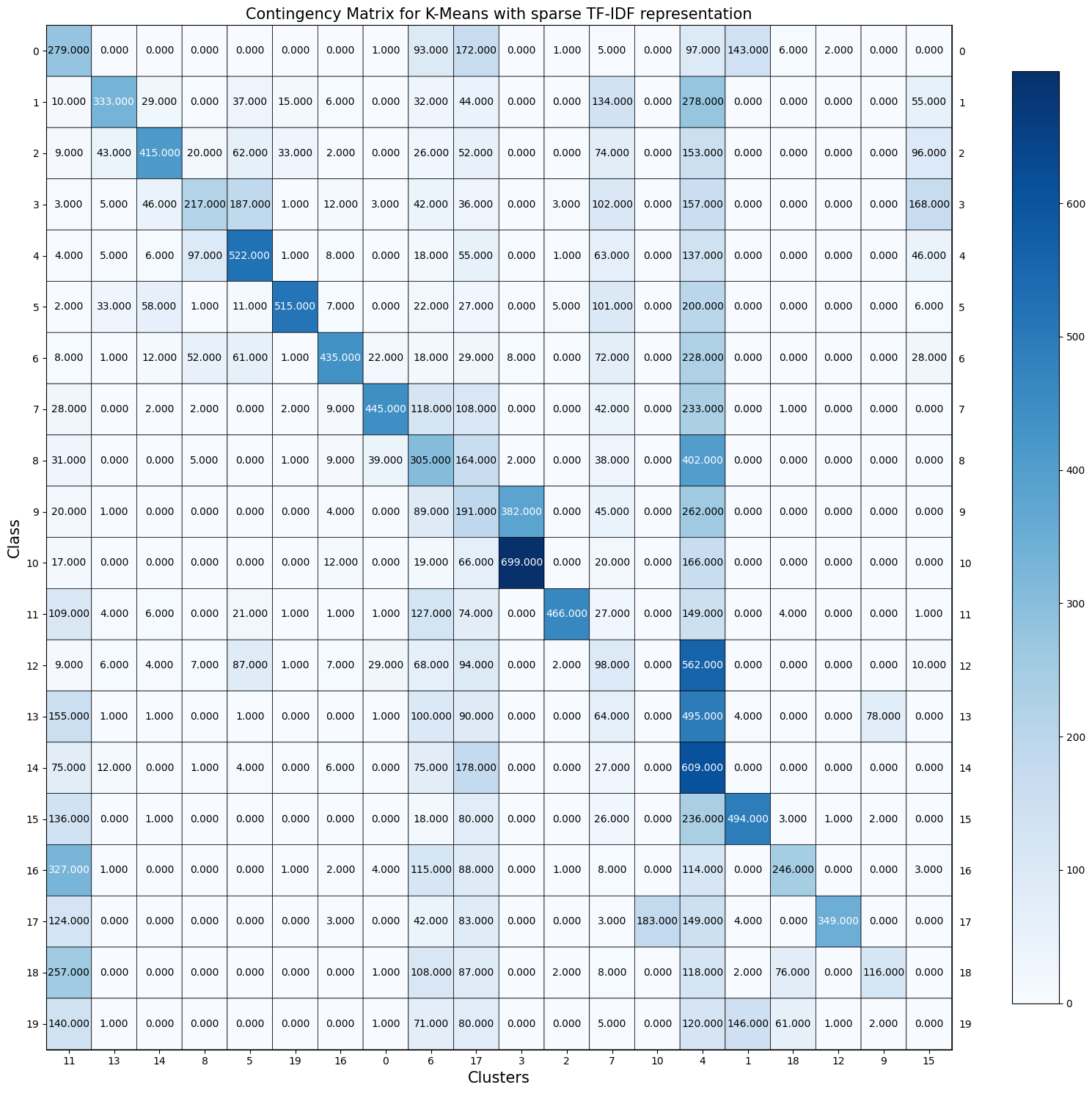
### *QUESTION 13*

So far we have attempted K-Means clustering on the 20-class text data with four different representation learning techniques: sparse TF-IDF representation, PCA-reduced, NMF-reduced, and UMAP-reduced. The five clustering evaluation metrics for K-means clustering with sparse TF-IDF representation can be found in *Table 5* and the contingency matrix in *Figure 15*.

Table 5: Clustering evaluation metrics for K-Means clustering task with sparse TF-IDF representation

| **Homogeneity score** | **Completeness score** | **V-measure score** | **Adjusted Rand Index score** | **Adjusted mutual information score** |
| --- | --- | --- | --- | --- |
| 0.3472 | 0.3973 | 0.3706 | 0.1249 | 0.3684 |

Figure 15: Contingency matrix for K-Means clustering task with sparse TF-IDF representation



To compare the clustering results across the four choices, we used the Adjusted rand index since it is similar to accuracy and we also used the V-measure score since it measures the goodness of the clustering partition. Where these two measures can tell us a lot due to their similarities to the other clustering evaluation metrics.

From *Table 6*  we see that reducing the dimension of our dataset for K-Means clustering with sparse TF-IDF representation, PCA, and NMF results in roughly the same performance. This can also be attributed to the fact that we set the n\_componets in NMF and SVD to values we knew would result in higher scores. The Adjusted Rand Index, which computes the similarity between the clustering labels and ground truth labels, is quite low as the range is from 0-1 where 1 is when clusters are identical. On average these 3 reduction techniques result in an Adjusted Rand Index score of ~0.1. However, their V-measure does perform better where we see an average of ~0.37 for cluster labeling given a ground truth. The star of the show, however, is UMAP-reduced as it overall outperformed the other representation techniques. For UMAP, we set n\_components and the metric to the settings that worked best, based on *Question 11/12*.

Therefore, we would suggest reducing the dimension of this 20-class text data with UMAP (with its settings that work best), before performing K-Means clustering.

Table 6: Clustering metrics for K-Means across the four choices

| **Representation Learning**  **Techniques** | **Adjusted Rand Index** | **V-measure** |
| --- | --- | --- |
| **Sparse TF-IDF representation** | 0.1249 | 0.3706 |
| **PCA-reduced** (n\_components =300 ) | 0.1134 | 0.4088 |
| **NMF-reduced** (n\_components =20 ) | 0.0976 | 0.3494 |
| **UMAP-reduced** (n\_components = 20, metric=’cosine’) | 0.4711 | 0.5824 |

## Other Clustering Algorithms

In this section we will try clustering methods that don’t explicitly rely on the Gaussian distribution per cluster. First we try Agglomerative clustering which seeks to build a hierarchy of clusters. Then we try HDBSCAN which also involves hierarchical clustering, but then extracts a flat clustering based on the stability of clusters. For both of these clustering methods, we apply it on UMAP-transformed 20-category data.

### *QUESTION 14*

After using UMAP to reduce the dimensionality properly, we performed Agglomerative clustering with n\_clustering=20 and compared the performance of ‘ward’ and ‘single’ linkage criteria. We see from *Table 7* that the “ward” linkage criteria performs better than the “single” linkage criteria. This is because the “ward” linkage criteria minimizes variance around cluster centroids, whilst the “single” linkage criteria uses distance minimum for two sets of clusters; because of this, the latter does not perform well on our 20-category text data.

Table 7: Clustering evaluation metrics for Agglomerative clustering

| **Clustering Measure** | **Linkage = ‘ward’** | **Linkage = ‘single’** |
| --- | --- | --- |
| **Homogeneity** | 0.5492 | 0.0192 |
| **Completeness** | 0.5895 | 0.3518 |
| **V-measure** | 0.5686 | 0.0365 |
| **Adjusted Rand Index** | 0.4147 | 0.0005 |
| **Adjusted Mutual Information score** | 0.5672 | 0.0315 |

### *QUESTION 15*

Next, we applied HDBSCAN on UMAP-transformed 20-category data. We varied the min\_cluster size by using 20, 100, 200 and then found each minimum cluster size’s five clustering evaluation metrics (which can be found in *Table 8* below). We see that a min\_cluster\_size of 100 and 200 are similar but a min\_cluster\_size=200 provides slightly better performance.

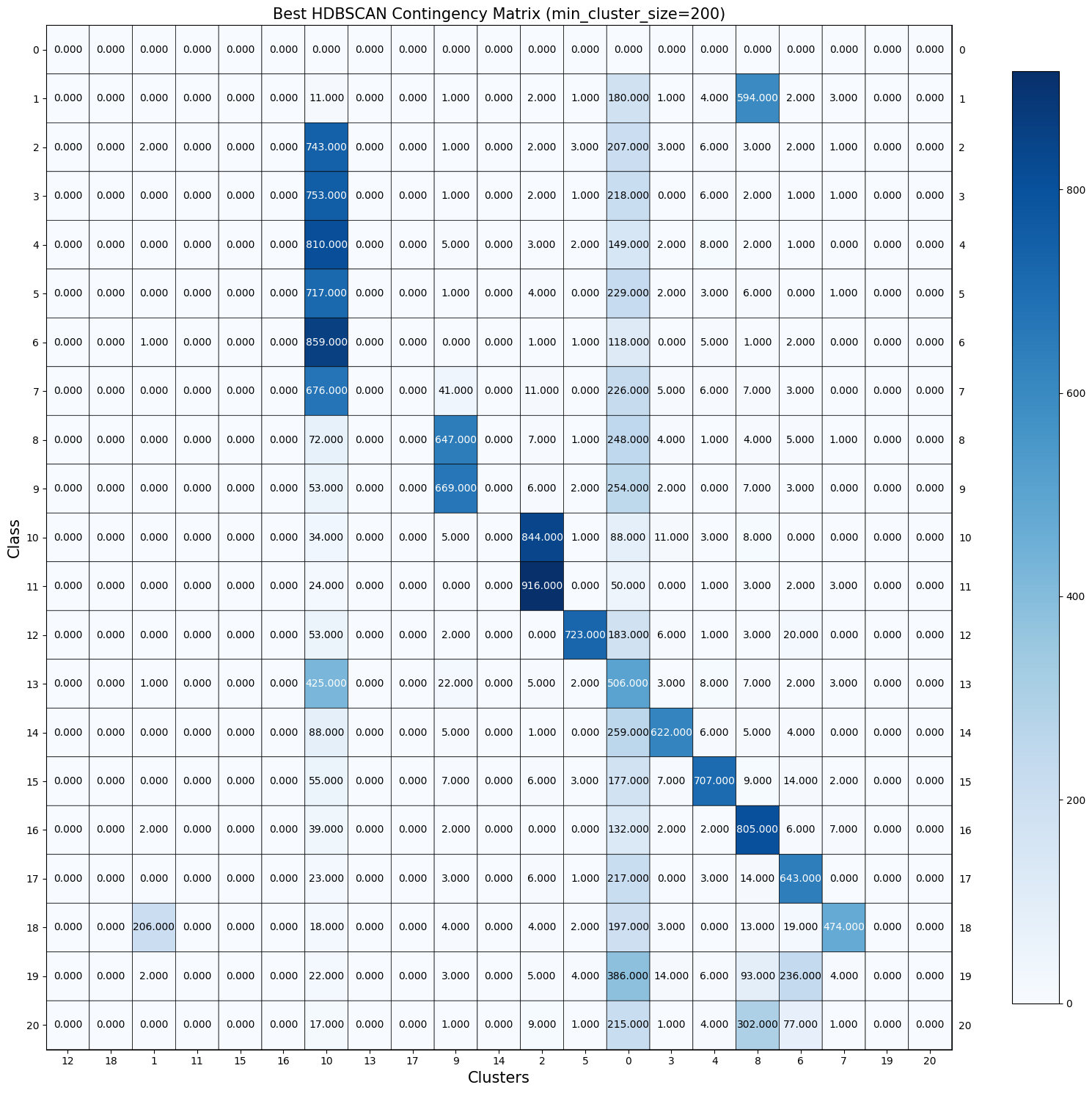
Table 8: Clustering evaluation metrics for HDBSCAN clustering

| **Clustering Measure** | **Min\_cluster\_size = 20** | **Min\_cluster\_size = 100** | **Min\_cluster\_size = 200** |
| --- | --- | --- | --- |
| **Homogeneity** | 0.4407 | 0.4165 | 0.4186 |
| **Completeness** | 0.4512 | 0.6138 | 0.6153 |
| **V-measure** | 0.4459 | 0.4963 | 0.4982 |
| **Adjusted Rand Index** | 0.086 | 0.2106 | 0.2161 |
| **Adjusted Mutual Information score** | 0.4337 | 0.4951 | 0.4972 |

### *QUESTION 16*

From *Question 15*, we use the hdbscan clustering model with min\_cluster\_size=200 as our best clustering model, which can be seen in *Figure 16*. Here there are 21 clusters, -1 to 19, but in the plot it is 0-20 because the plotmat.py file provided by the TA gives the axis labels based on length. HSBSCAN is noise aware, where it gives the label -1 to the data samples that are not assigned to any cluster (like mentioned before, the “-1” cluster label is “0” in the matrix below. So we see that there was a significant amount of data that was not assigned to any cluster and because of the high min\_cluster\_size, this is why we see some clusters don’t have many data points in them.

Figure 16: Contingency matrix for best HDBSCAN model



### *QUESTION 17*

To find the dimensionality reduction technique and clustering methods that worked best together for 20-class text data, we created a gridsearch for the TF-IDF reduced data. The evaluation of each combination was performed with 5-fold cross validation and the average accuracy across the folds was reported. *Table 9* shows the set of hyperparameters that were considered for pipeline comparison. The pipeline we made can be found in *Figure 17*. From the grid search we saw that the best dimensionality reduction technique and clustering method that work best together is: steps=[('Dim\_reduction', UMAP(angular\_rp\_forest=True, metric='cosine', n\_components=5), ('Clustering', KMeans(max\_iter=1000, n\_clusters=10, n\_init=40, random\_state=0))]).

Note: the wrong score metric was used for these clusters, but will still use results found with score=’accuracy’ because the grid search takes too long to run.

Table 9: Hyperparameters to consider for pipeline comparison

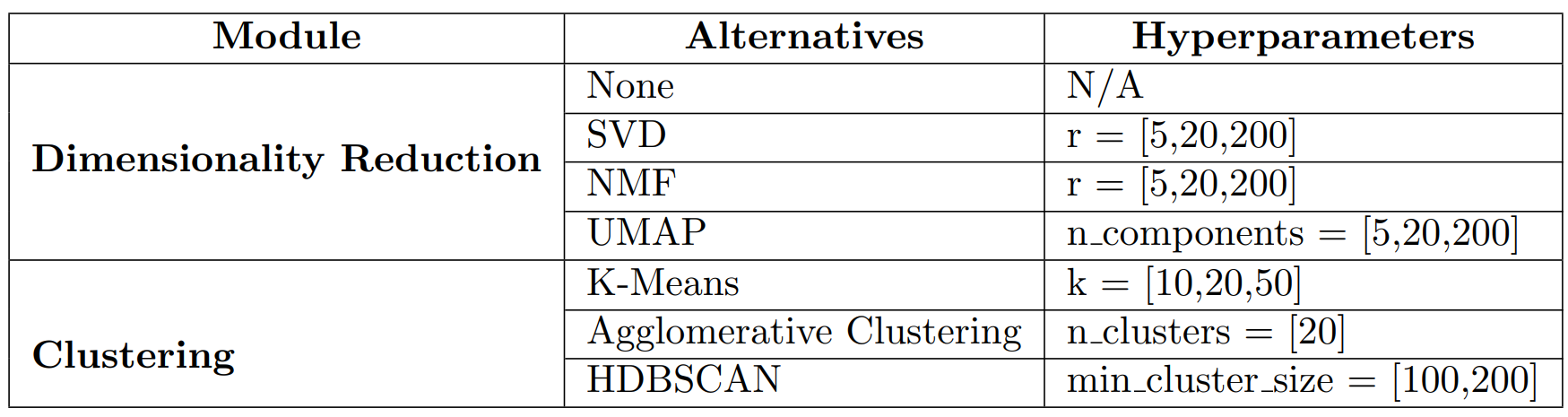
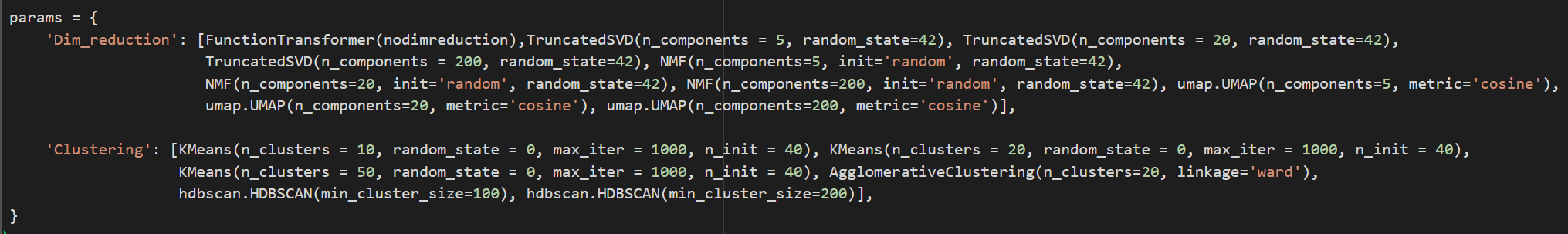


Figure 17: Pipeline and its hyperparameters



### *QUESTION 18*

To further enhance the clustering performance, we clean the data before performing TF-IDF representation by lemmatizing. From *Table 10* we see that actually not cleaning the data before anything resulted in better performance according to the clustering evaluation metrics. For this example we used the best pipeline found in *Question 17*. ({'Clustering': KMeans(max\_iter=1000, n\_clusters=10, n\_init=40, random\_state=0), 'Dim\_reduction': UMAP(metric='cosine', n\_components=5)})

Table 10: Clustering evaluation metrics for clean vs unclean data on best pipeline found

| **Clustering Measure** | **Cleaned dataset** | **Uncleaned dataset** |
| --- | --- | --- |
| **Homogeneity** | 0.3951 | 0.4578 |
| **Completeness** | 0.5504 | 0.6488 |
| **V-measure** | 0.46 | 0.5368 |
| **Adjusted Rand Index** | 0.2777 | 0.3331 |
| **Adjusted Mutual Information score** | 0.4591 | 0.536 |

### 

# Part 2 - Deep Learning and Clustering. of Image Data

### *QUESTION 19*

The VGG network is trained on Flower Image net dataset; the network learned to classify different target labels. The Weights learned by the model is another method use to train the dataset. The Weight technique for the pre-trained model in the custom dataset is called Transfer Learning. Instead of intizaling the weights randomly, the weights from the pre-trained model will be used as the initial weights. In fact, the model will in turn learn from scratch and directly inherit simple features by using the pre-train weights, which in turn the model will concentrate to learn the complex features in the present custom data-set. The transfer learning method offers an advantage: it reduces the train time for the model . The method is only useful when the dataset is not large enough to be trained from the beginning

*QUESTION 20*

The helper code loads the flower dataset and applies resizing, center cropping, normalizing transformation to homogenize the dataset. The dataset is split into 59 batches with 64 samples each. In addition, the feature extractor calls the pre-trained VGG model. The VGG architecture consists of 2D convolutions, ReLU activations and max pooling. The 2D convolutions down samples the image and extracts the complex images, the ReLU activation function and average max pooling is also applied to the dataset. The Fully connected and softmax layer provide the scores and probability estimations for the target labels

*QUESTION 21*

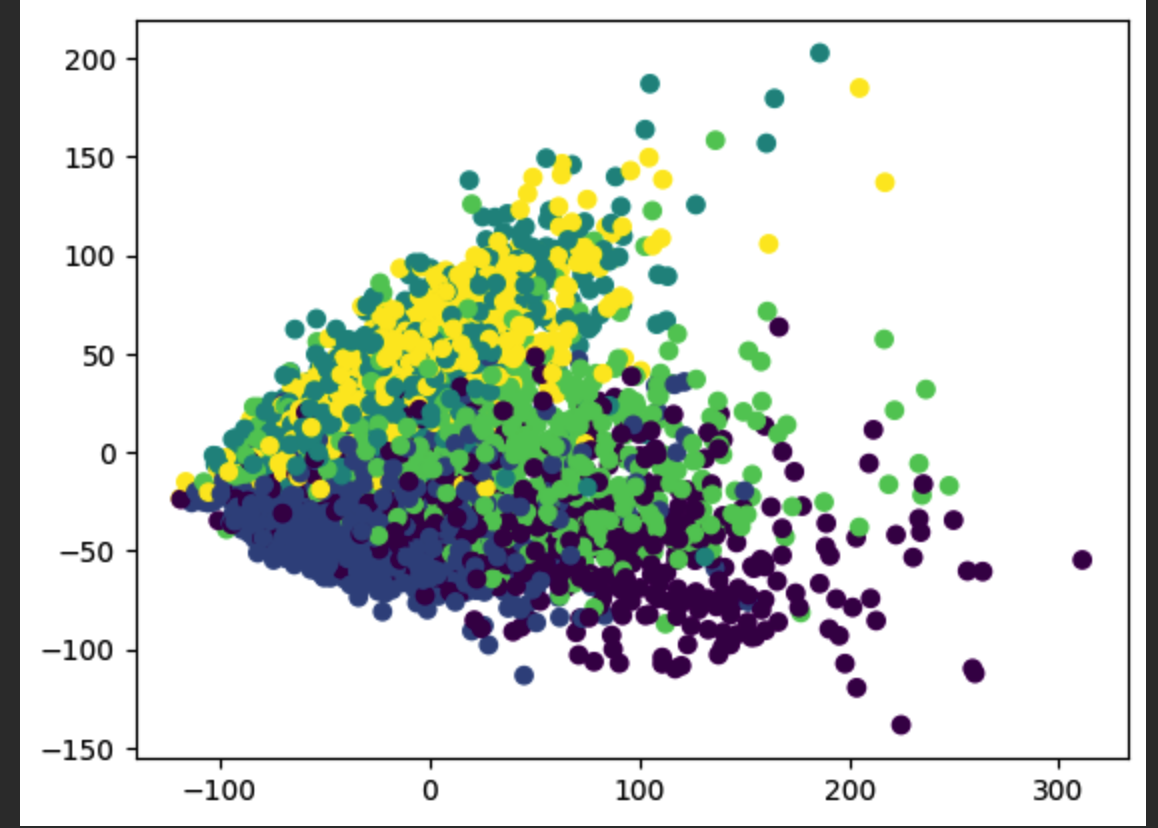
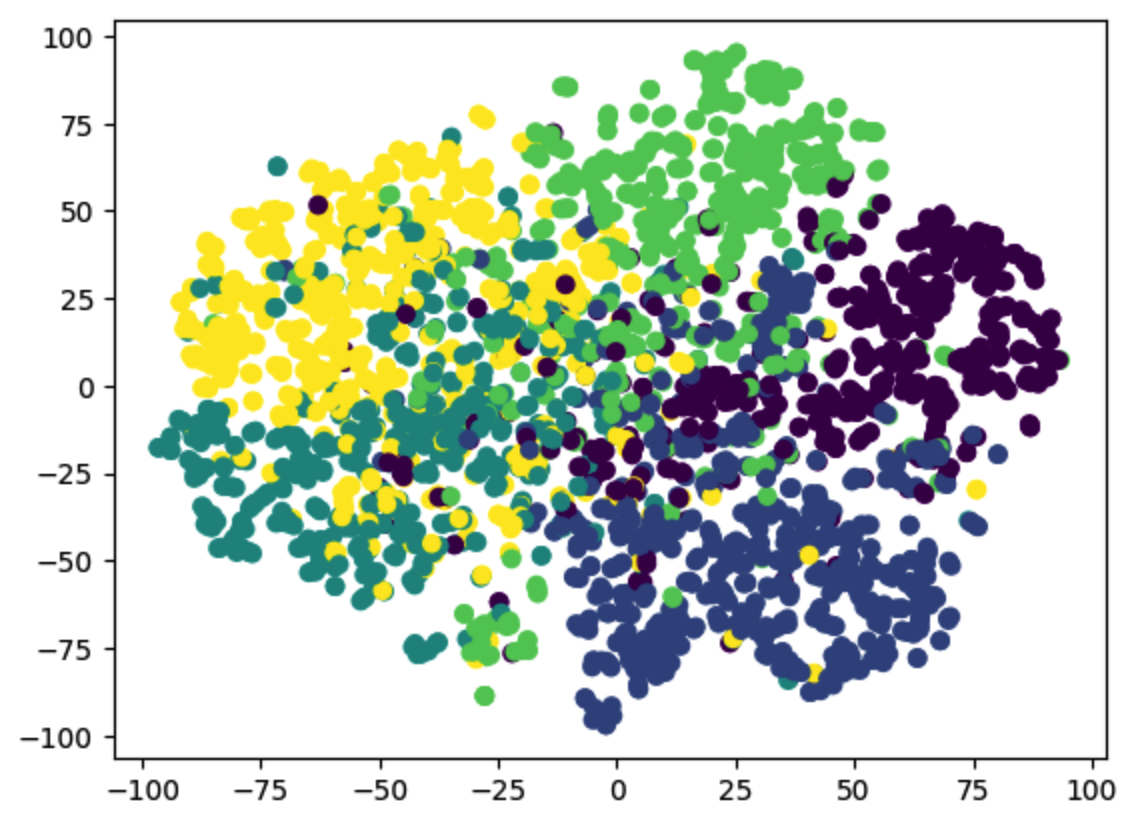
The images in the dataset. The transform is applied to resize the images to the same size. The VGG network extracts 4096 features per image sample.

### *QUESTION 22*

The TF\_IDF matrix has 363490 nonzero elements with a shape of 7882 x 1849. The features extracted from the VGG network, the feature space are non zero using np.count\_nonzero; therefore proving a denser than the TF\_IDF matrix.

### *QUESTION 23*

Based on the results, T-SNE preserves class information in comparison to PCA. T-SNE uses a probabilistic approach ,which tries to group points that are nearest to each other based on the probability of two closest points from the same probability distribution. PCA separates points based on the variance. PCA is a linear dimensionality reduction technique, while T-SNE is a nonlinear dimensionality reduction technique. T-SNE preserves the local data structure whereas PCA does not, which in turn T-SNE is a more popular technique.

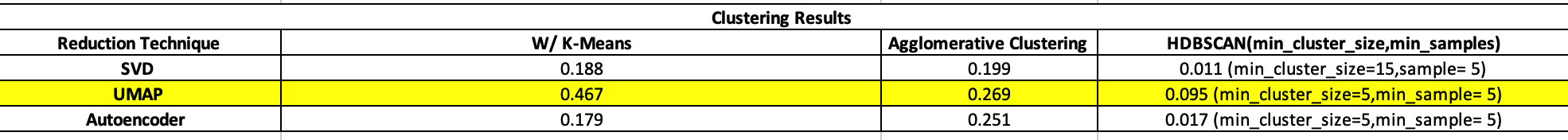
**PCA Scatter Plot T-SNE Scatter Plot**

*QUESTION 24*

*Four dimensionality reduction techniques were tested for Adjusted Random Index, the results can be as shown below. The UMAP reduced features with K-Means clustering has the best performance. The results for the UMAP are highlighted on the table below. HBDSCAN for the test values for both min\_cluster\_size and min\_sample size are the following*

*min \_cluster\_size = [5,15,30,60,100,200,500,1000,3000]*

*Min\_sample\_size =[5,15,30,60,100,200,500]*

**

### *QUESTION 25*

The test accuracy for MLP classifier based on dimensionality reduction, UMAP, and Autoencoder are shown below. MLP classifier without Dim\_Red had the best test accuracy in comparison with UMAP and Autoencoder. MLP classifier learns the complex features and predicts based on the classes. On the contrary, the cluster models are formed based on distances, there is a trade-off between feature complexity and accuracy. Therefore, MLP classifier generally out performs better than classical clustering methods only for complex data. 