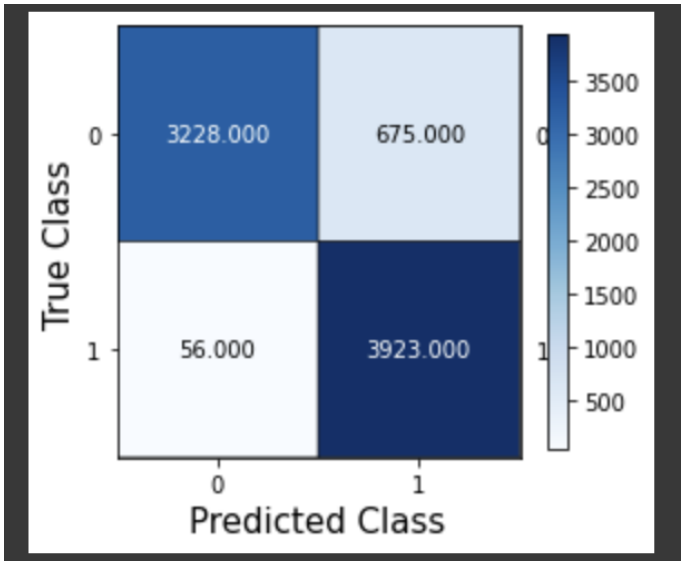
Part 1

Question 1:

(7882, 23522)

Question 2:

The contingency table of the clustering results is as following:



11

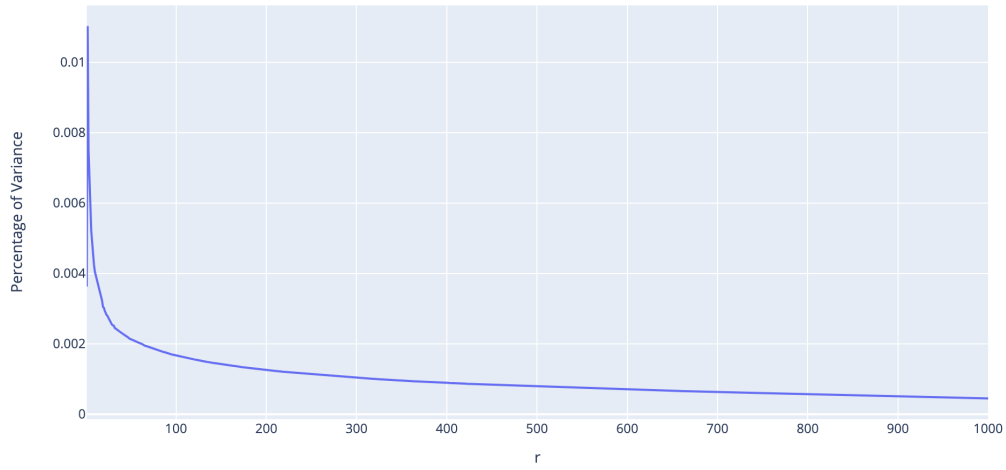
Contingency matrix doesn't have to be square-shaped. The number of predicted classes we got in the contingency table depends on the number of clusters we set in the k-means algorithm. Therefore, if we set the number of clusters to be smaller than the number of true classes, the number of True Class (rows) will be more than that of the Predicted Class (columns).

Question 3:

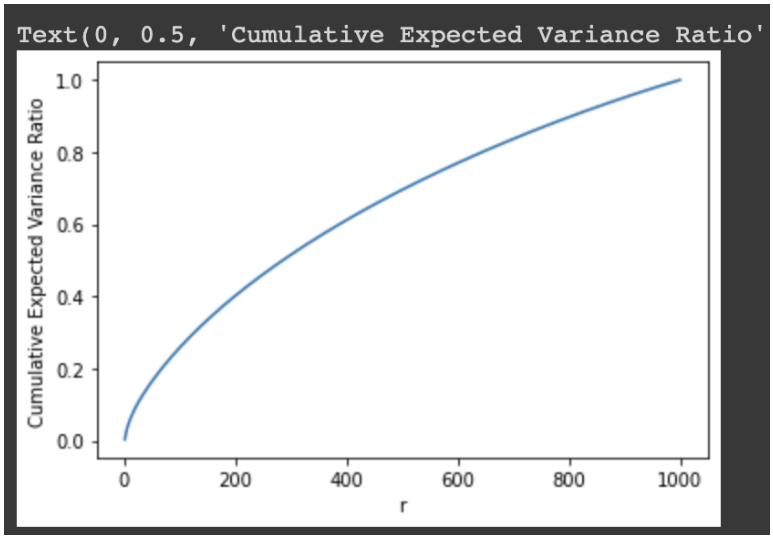
```
Homogeneity: 0.582 ± 0.012
Completeness: 0.596 ± 0.011
V-measure: 0.589 ± 0.012
Adjusted Rand-Index: 0.644 ± 0.015
Adjusted mutual information score: 0.589 ± 0.012
```

Question 4:

Explained Variance Ratio Per Components

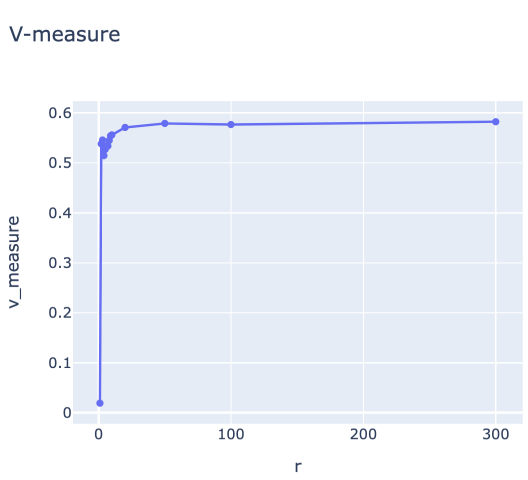


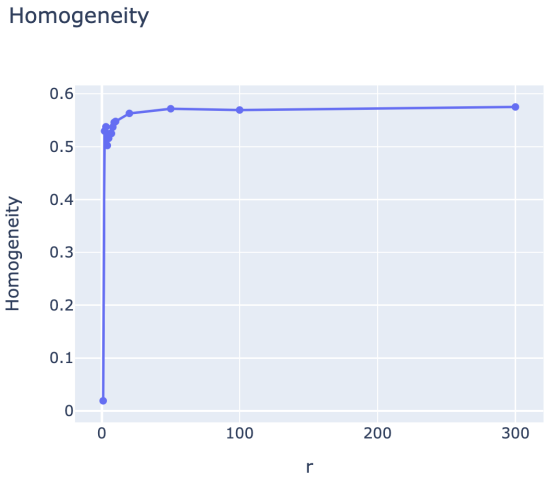
Cumulative Expected Variance Ratio

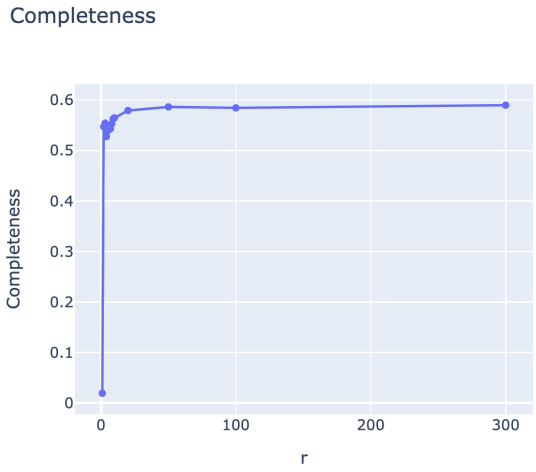


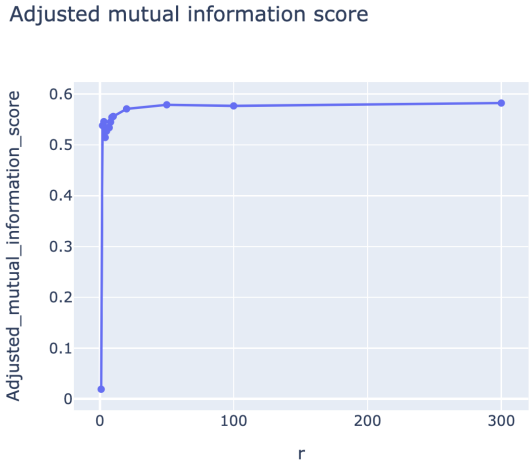
Question 5:

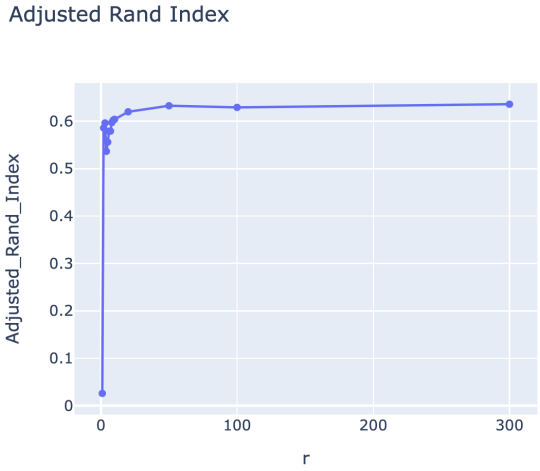
SVD (the r with the best performance is 50)





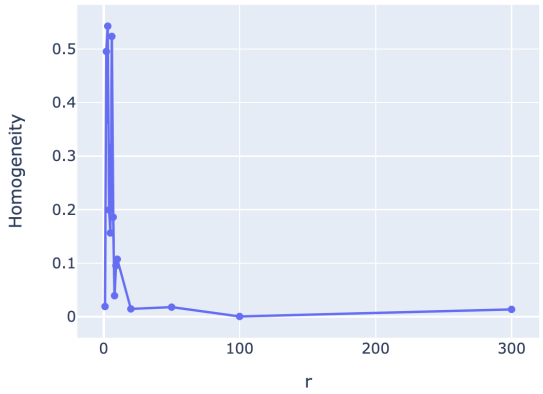




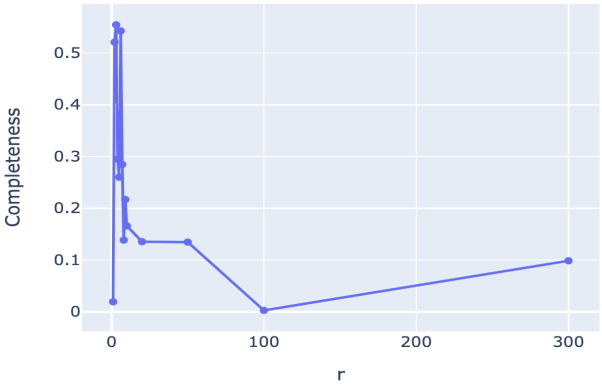


NMF (the r with the best performance is 3):

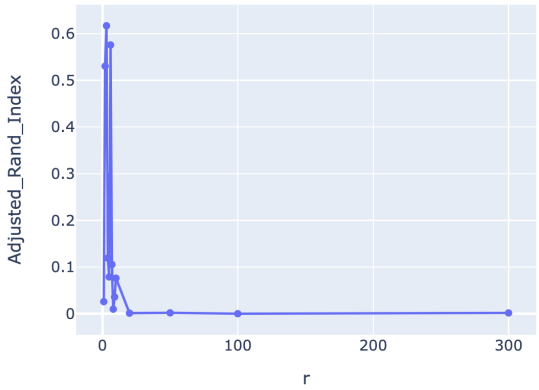
Homogeneity



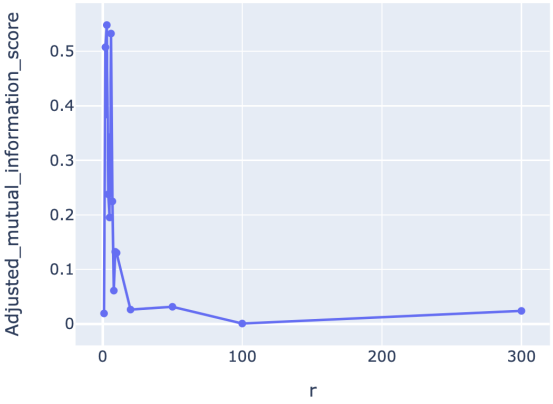
Completeness



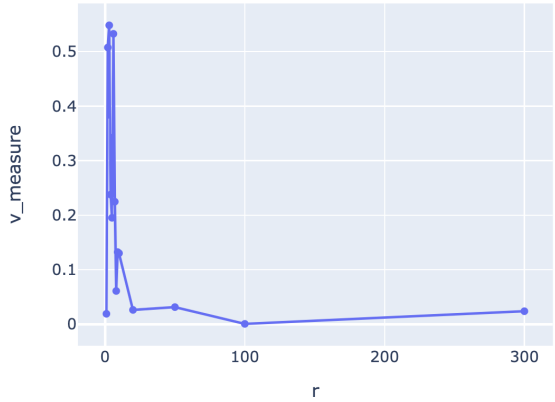
Adjusted Rand Index



Adjusted mutual information score



V-measure



Question 6:

In our example, the performance for SVD drastically improves as the number of components goes from 0 to 50. After this point, the model seems not improving as much as before. This might be due to the fact that more relatively irrelevant data come in as we increase the # of components, resulting in no further improvement in making predictions. For NMF, the average performance score reaches the peak at r = 3 and then decreases to nearly 0. This might be due to NMF being more noise sensitive compared to SVD.

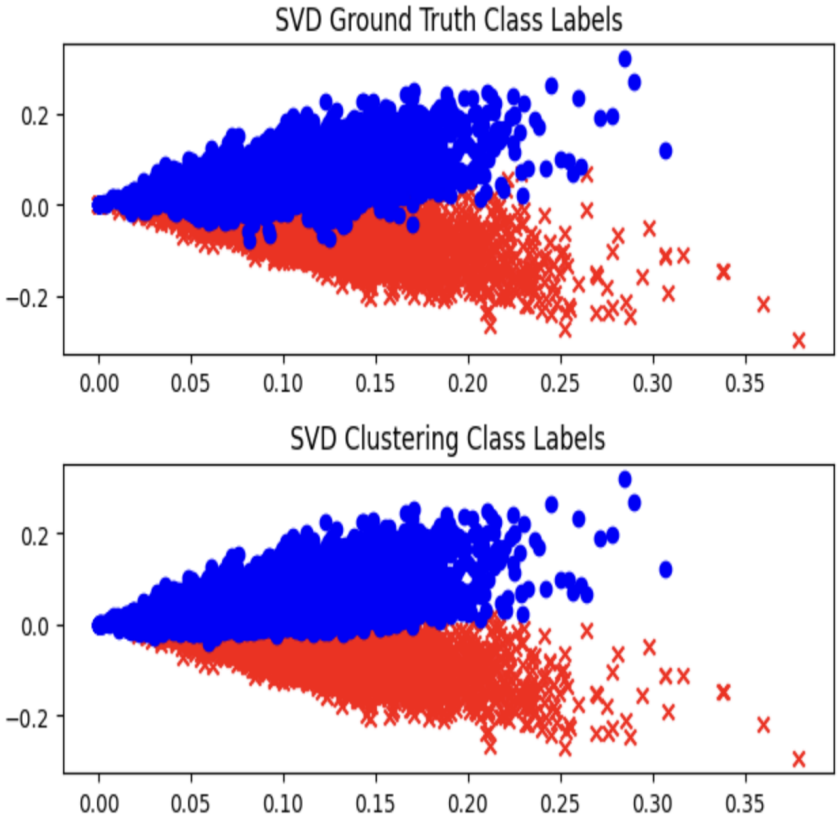
Question 7:

No, the average in Question 3 is about 0.6, whereas here we only have a maximum average of 0.58 in Question 6.

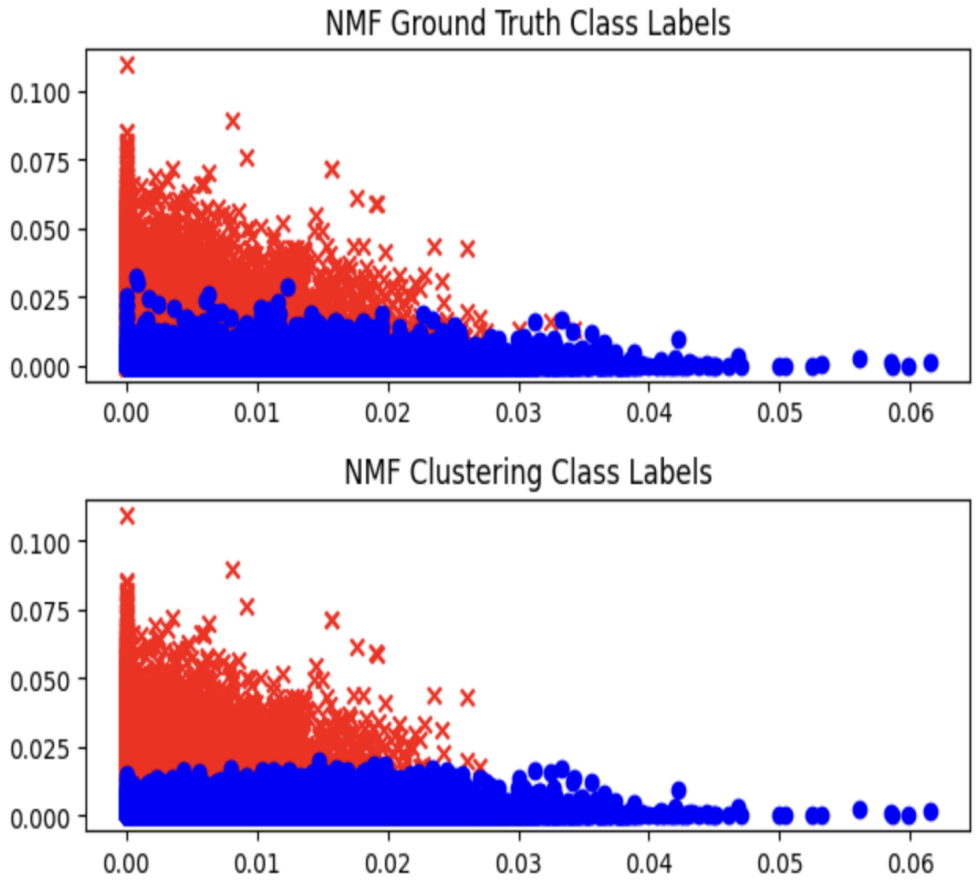
Question 8:

Visualize the clustering results for:

SVD with r = 50:



NMF with r = 3:

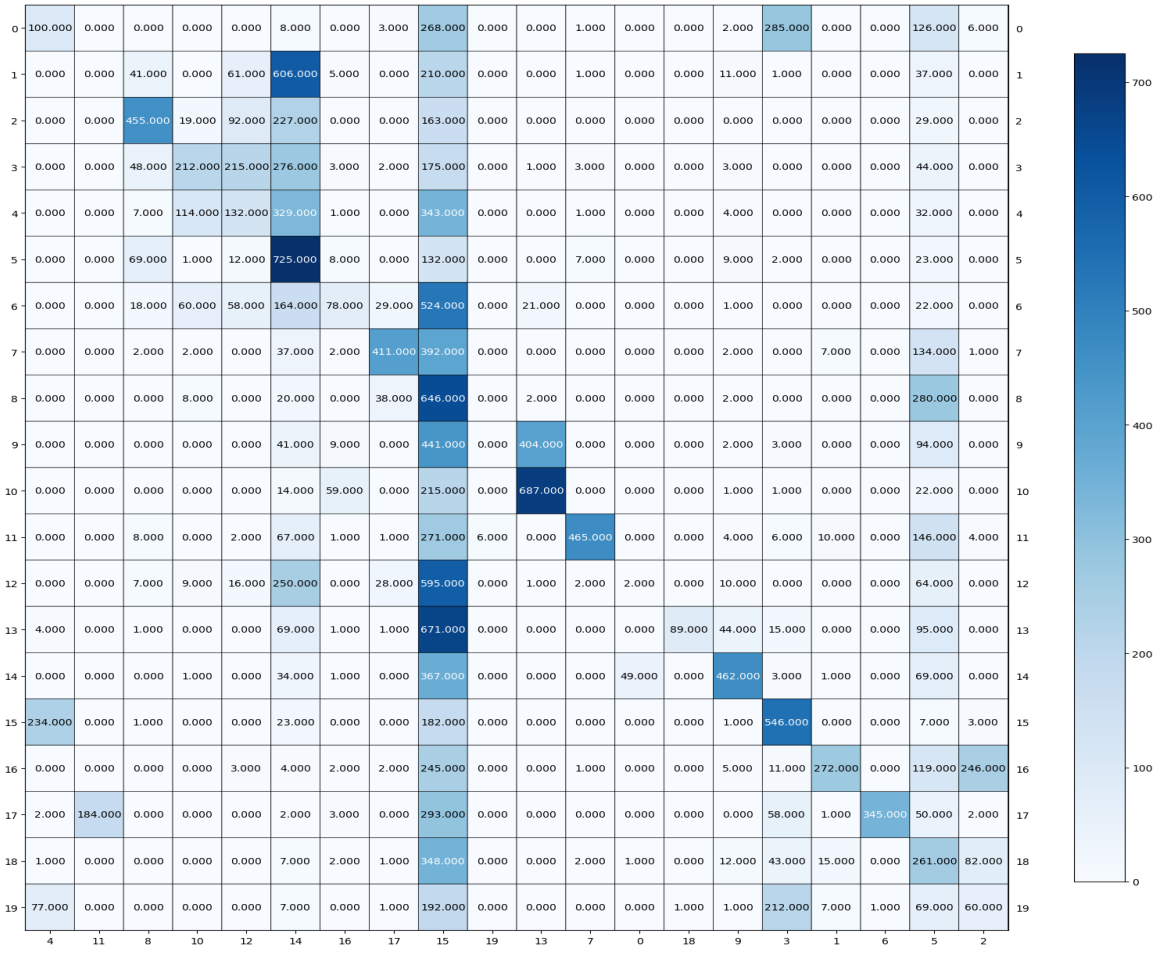


Question 9:

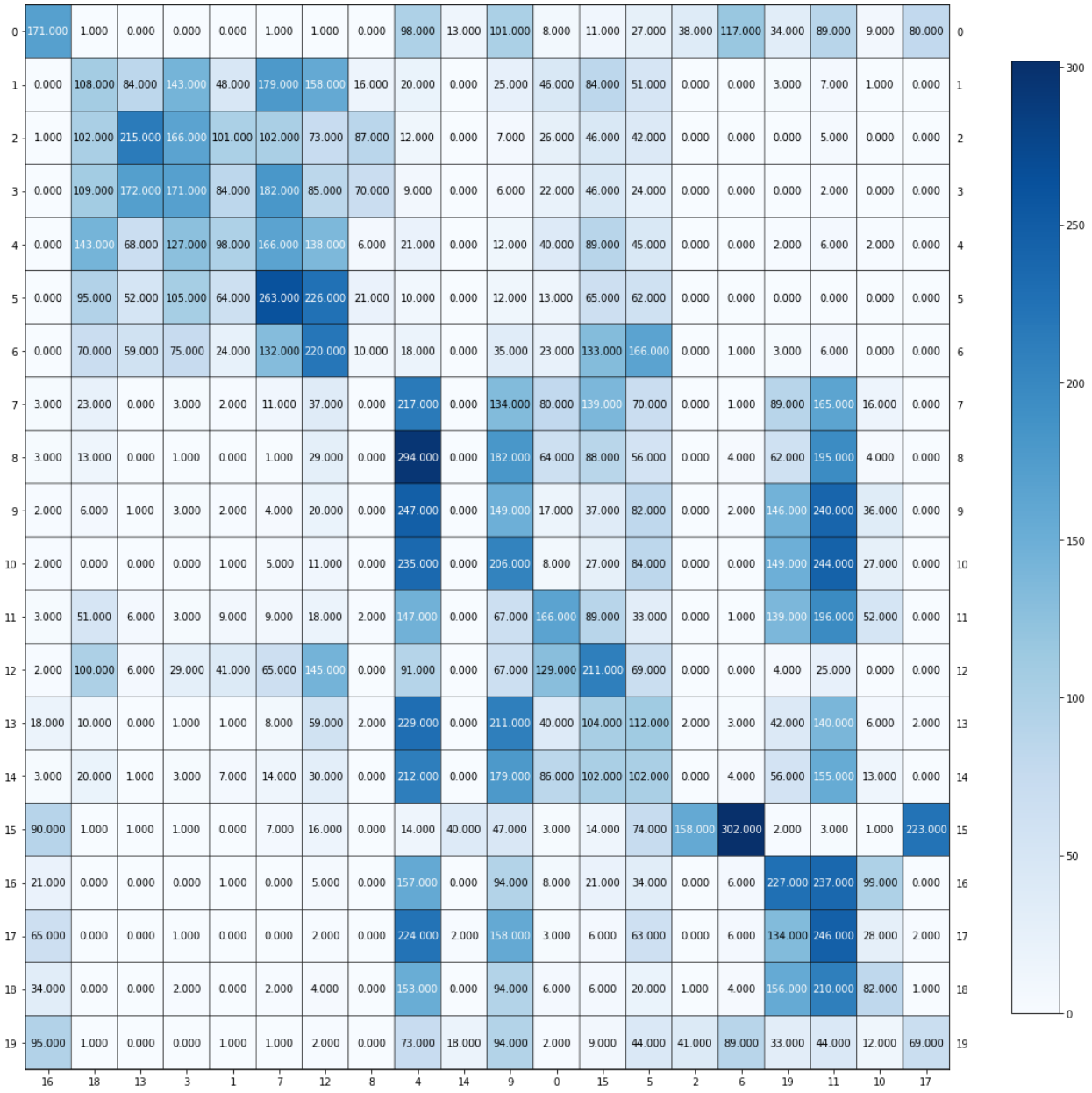
From these visualizations, we can see that the boundaries in the SVD/NMF Clustering Class Labels figures are more smooth and linear (some blue points are being classified as red points after the SVD/NMF clustering). The clustering data of the two classes are evenly distributed by the boundaries. Overall, we believe that the distribution is ideal for K-Means clustering, because 1. the boundaries are clear and well-defined, and 2. the clusters in SVD and NMF make sense, and 3. the data within the clusters are almost from the same class.

Question 10:

For SVD, we set r = 1000, because we got better measure scores in Q3 (r = 1000) compared to the scores in Q5.

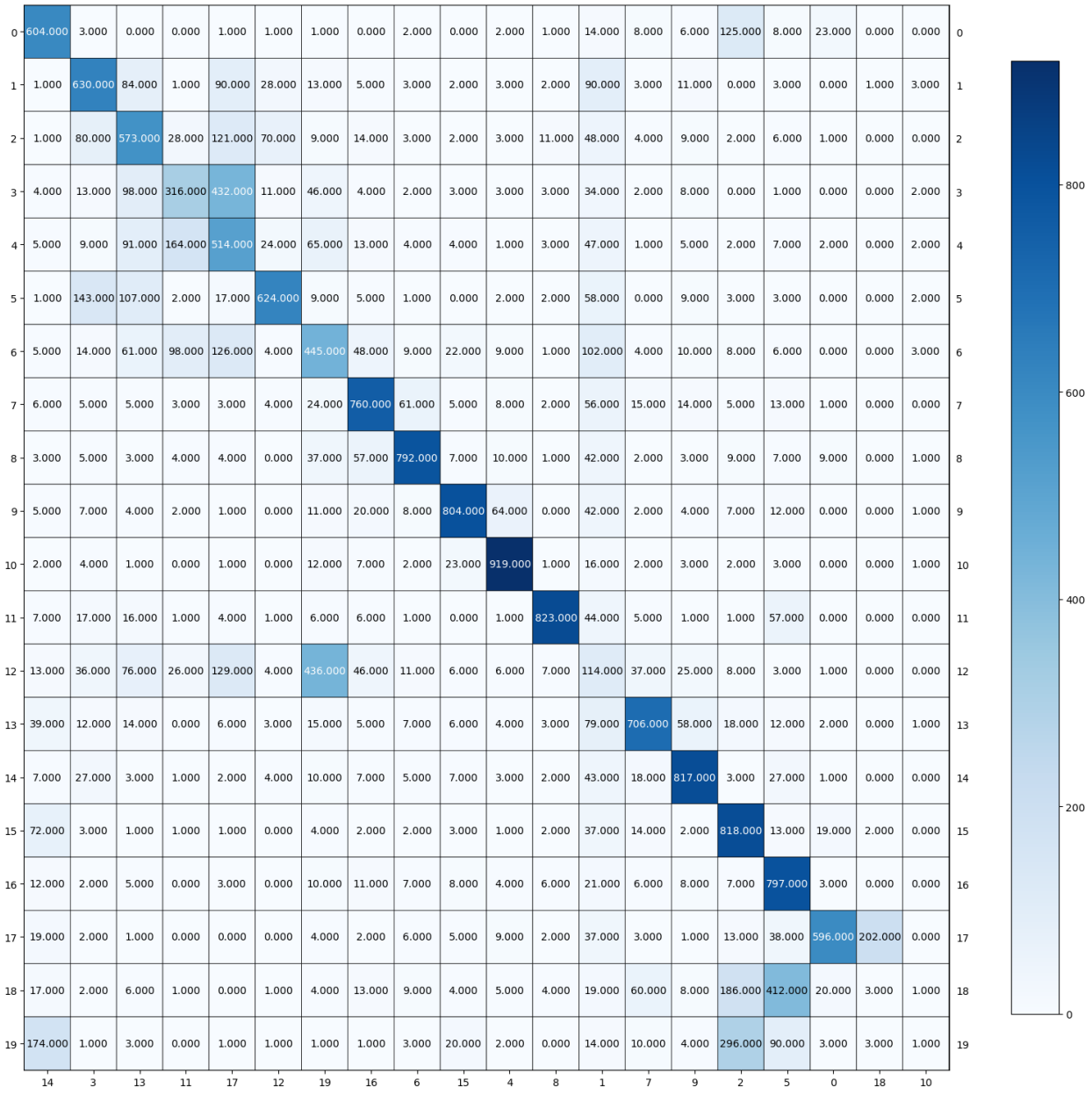


For NMF, we decided r to be 3, because the result of Question 8 indicates that the result is the best when r = 3.

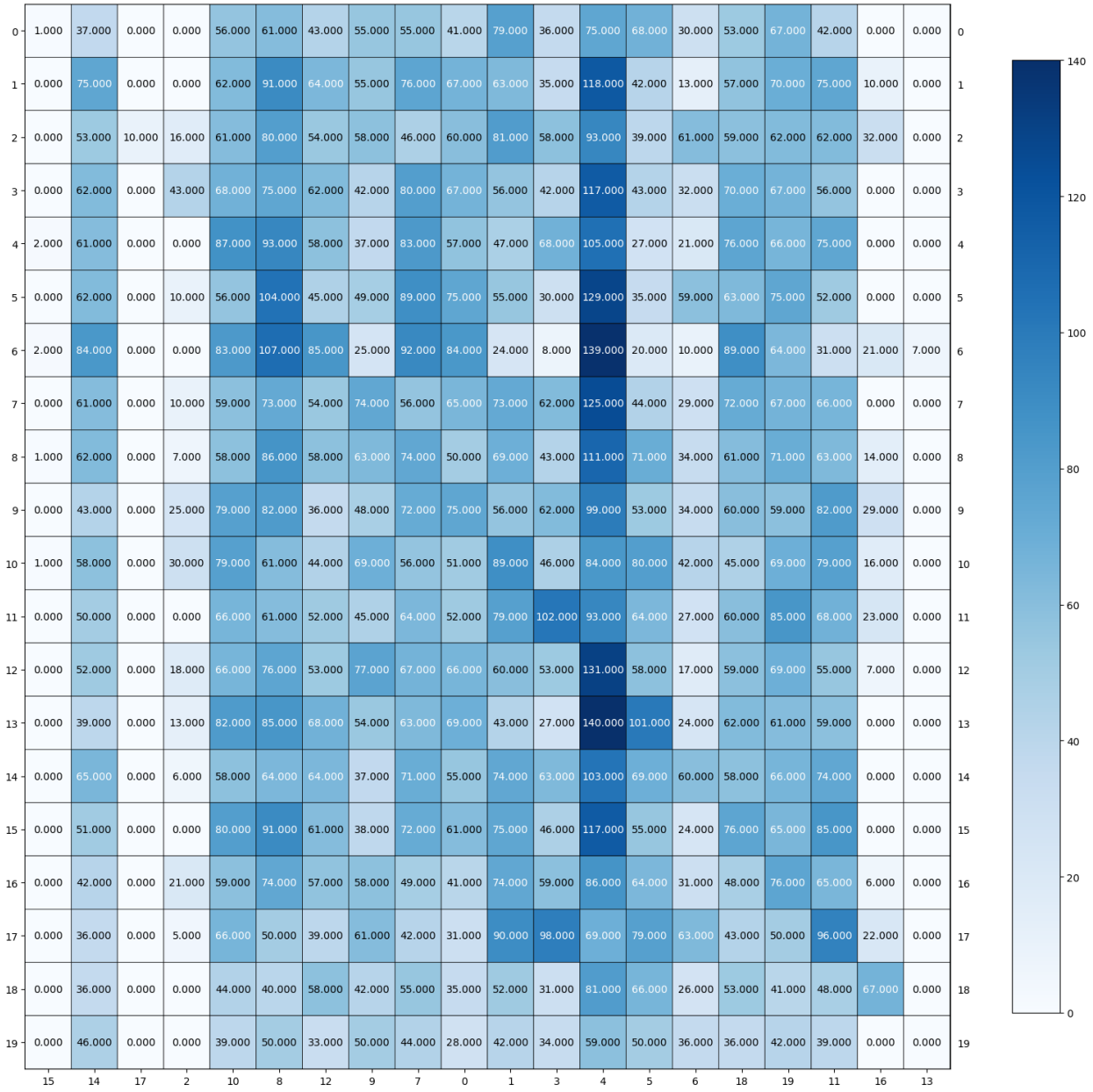


Question 11:

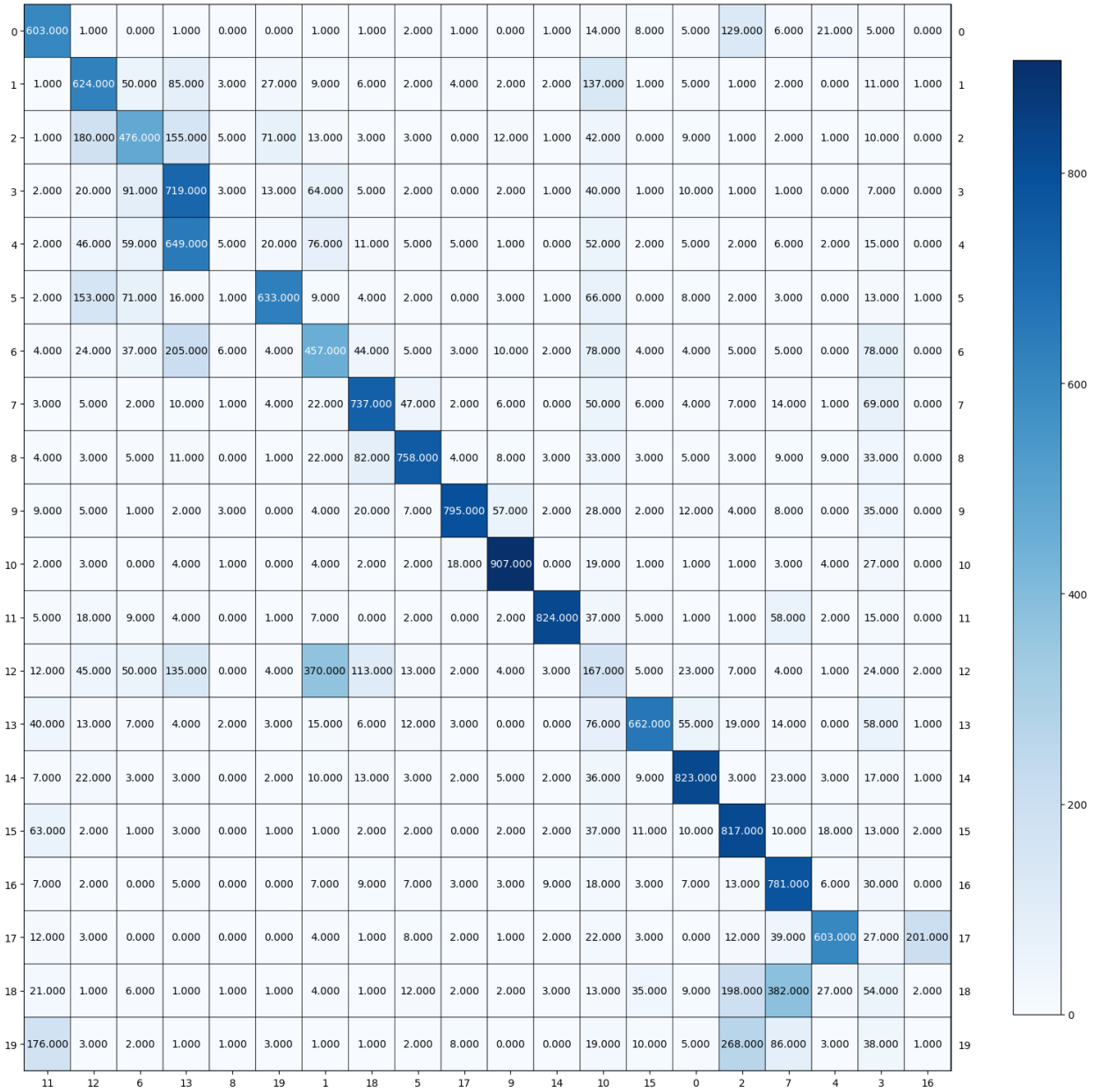
```
************** This is for components = 5 and metric = cosine ***********************
Homogeneity score:  0.5787205426049561
Completeness score:  0.5887460516747586
V-measure score:  0.5836902505839211
Adjusted Rand score:  0.4595680059173617
Adjusted mutual information score:  0.5823341943107677
```



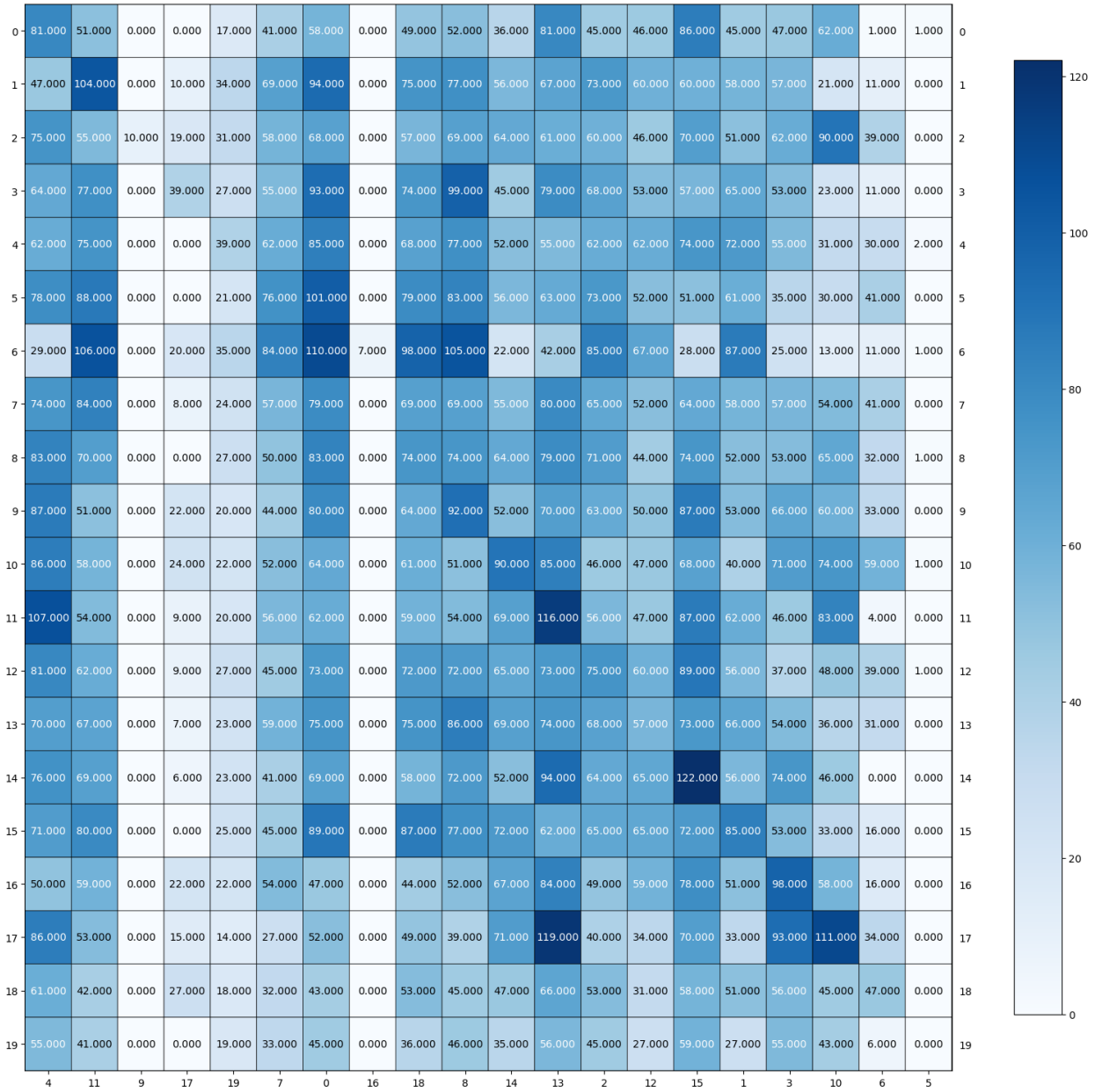
********** This is for components = 5 and metric = euclidean**********
Homogeneity score: 0.015028207643473433
Completeness score: 0.016265879653337943
V-measure score: 0.015622568865206654
Adjusted Rand score: 0.0025790051084612175
Adjusted mutual information score: 0.012336506360402432



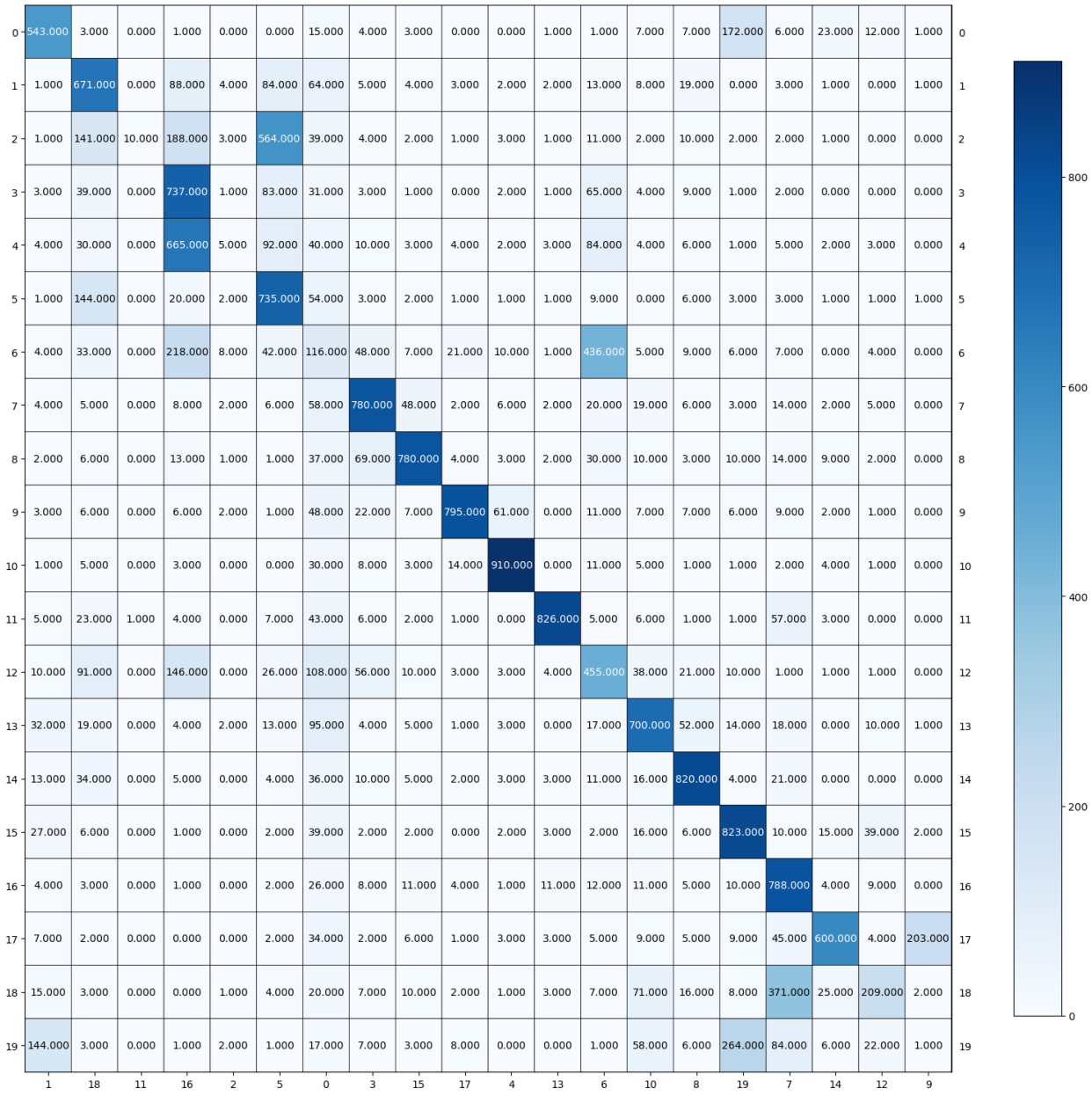
********** This is for components = 20 and metric = cosine **********
Homogeneity score: 0.5653669230886433
Completeness score: 0.5873123138431823
V-measure score: 0.5761307137854349
Adjusted Rand score: 0.4405249114723275
Adjusted mutual information score: 0.5747234858808318



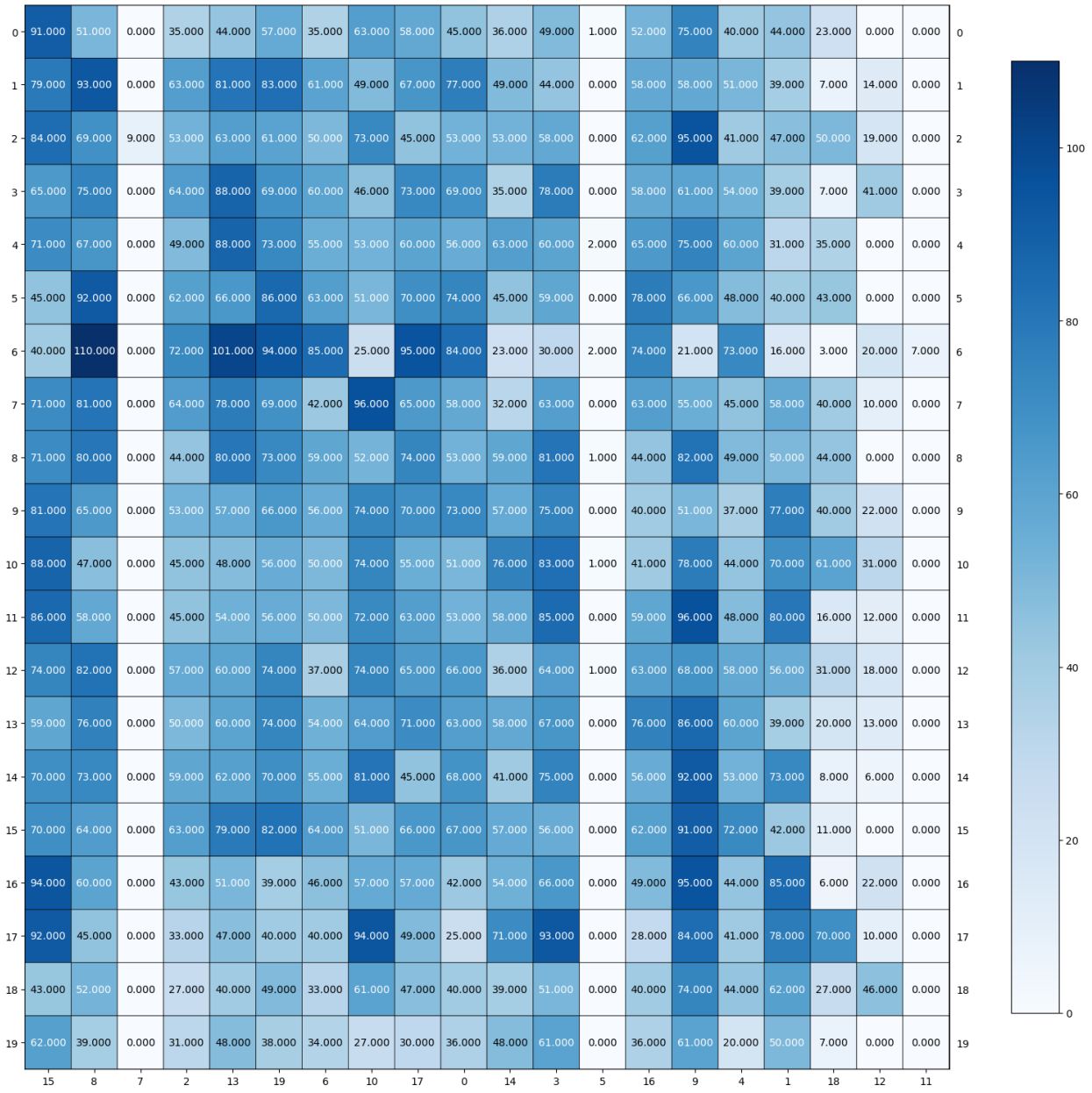
********** This is for components = 20 and metric = euclidean**********
Homogeneity score: 0.01604421948605504
Completeness score: 0.01686563660129975
V-measure score: 0.01644467691897745
Adjusted Rand score: 0.00329081513922261
Adjusted mutual information score: 0.01320192276043358



********** This is for components = 200 and metric = cosine **********
Homogeneity score: 0.5483641463877107
Completeness score: 0.5751842516757032
V-measure score: 0.5614540890796574
Adjusted Rand score: 0.4210116316148985
Adjusted mutual information score: 0.5599906979952898



```
************* This is for components = 200 and metric = euclidean*********************
Homogeneity score:  0.015418617033985728
Completeness score:  0.016614834994086638
V-measure score:  0.015994390965556018
Adjusted Rand score:  0.003162182536014381
Adjusted mutual information score:  0.012725610671530494
```



Question 12:

Based on the matrix and the evaluation metrics, we can see that when n_components = 5, it has a higher average performance score than the other. In terms of the metric choice, 'cosine' generally works much better than 'euclidean'. cosine similarity works better in the high-dimensional and

sparse data, and the goal is to preserve this similarity structure in the lower-dimensional representation. Cosine metric doesn't measure the magnitude. Instead, it measures the direction and orientation of the normalized vector. So, it's more suitable for high dimensional data.

Question 13:

```
* * * * * * * * * * * * * * * * * * * * * model = non * * * * * * * * * * * * * * * * * * * * *
Homogeneity score:  0.3488058772382193
Completeness score:  0.3977815186570097
V-measure score:  0.37168731303837976
Adjusted Rand score:  0.12268896281424234
Adjusted mutual information score:  0.369515271261787
* * * * * * * * * * * * * * * * * * * * * model = svd * * * * * * * * * * * * * * * * * * * * *
Homogeneity score:  0.32313809864466086
Completeness score:  0.35118683066780604
V-measure score:  0.3365791172713842
Adjusted Rand score:  0.12666159829052642
Adjusted mutual information score:  0.33434061518575153
* * * * * * * * * * * * * * * * * * * * * model = nmf * * * * * * * * * * * * * * * * * * * * *
Homogeneity score:  0.2984695950441029
Completeness score:  0.347013570400772
V-measure score:  0.3209161923253039
Adjusted Rand score:  0.10409342440782972
Adjusted mutual information score:  0.31852083220776384
* * * * * * * * * * * * * * * * * * * * * model = umap * * * * * * * * * * * * * * * * * * * * *
Homogeneity score:  0.5626567911160126
Completeness score:  0.5986812998984997
V-measure score:  0.5801103084594229
Adjusted Rand score:  0.4407262326533106
Adjusted mutual information score:  0.5787001873527008
```

Based on the above graph, we can clearly see that UMAP works much better on average than other models.

Question 14:

```
* * * * * * * * * * * * * Agglomerative clustering with ward  * * * * * * * * * * * * *
Homogeneity score:  0.5646435777462193
Completeness score:  0.5889903838189804
V-measure score:  0.576560067851075
Adjusted Rand score:  0.4360571940971298
Adjusted mutual information score:  0.5751513512195747

* * * * * * * * * * * * Agglomerative clustering with single  * * * * * * * * * * * * *
Homogeneity score:  0.016507812112058495
Completeness score:  0.3484280424185708
V-measure score:  0.031522168005201384
Adjusted Rand score:  0.0004794725050577410 6
```

```
Adjusted mutual information score:  0.026419525777930903
```

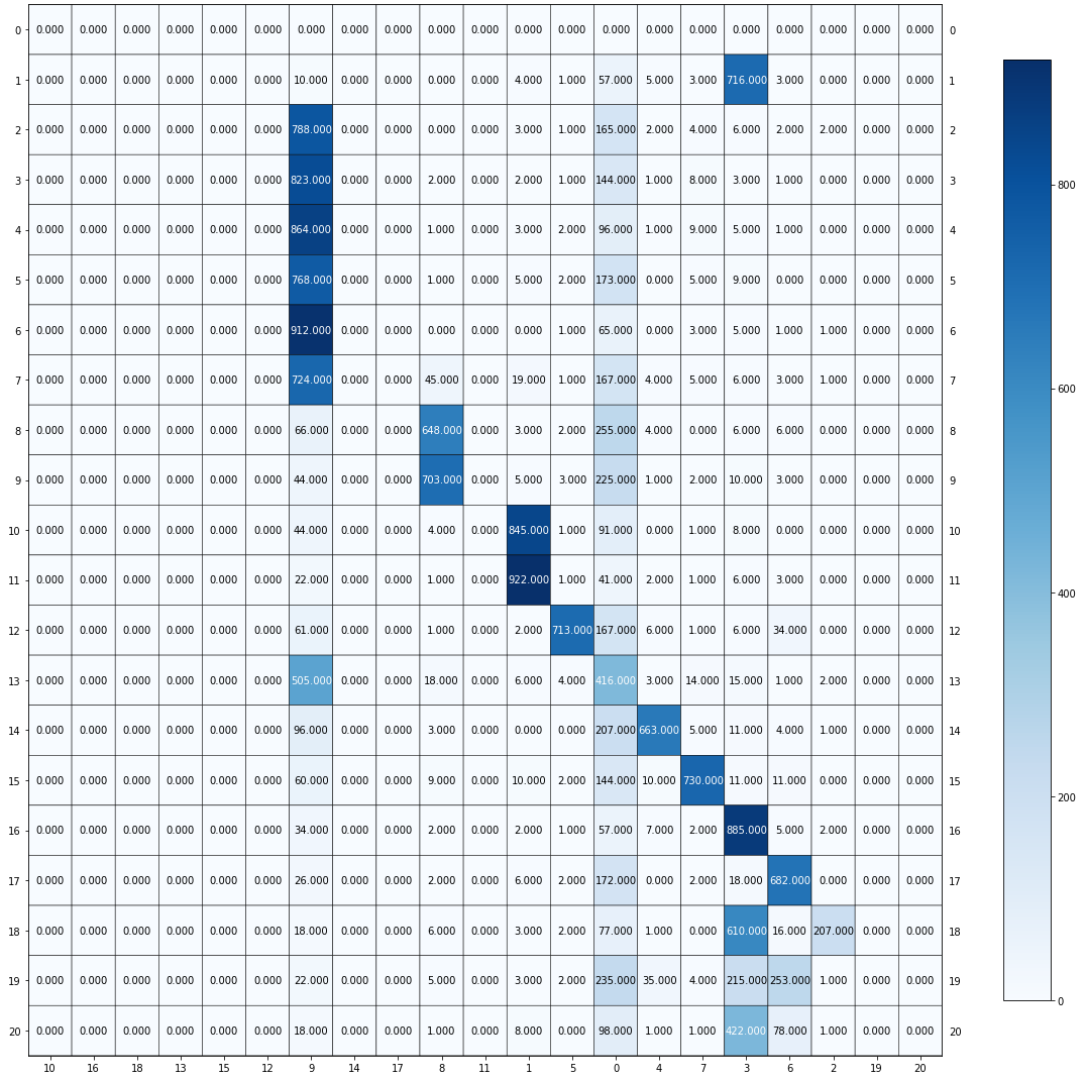
Question 15:

HDBSCAN clustering using n = 200,

```
* * * * * * * * * * * * * HDBSCAN clustering using n = 20 * * * * * * * * * * * * * *
Homogeneity score:  0.00037348001795512565
Completeness score:  0.08251480356407795
V-measure score:  0.0007435943654491489
Adjusted Rand score:  -2.495451403736458e-06
Adjusted mutual information score:  0.00035476952203728103
* * * * * * * * * * * * * * HDBSCAN clustering using n = 100 * * * * * * * * * * * * * *
Homogeneity score:  0.4015384506528541
Completeness score:  0.6125333385609805
V-measure score:  0.4850853565893493
Adjusted Rand score:  0.20812476293885468
Adjusted mutual information score:  0.48402734491274263
* * * * * * * * * * * * * * HDBSCAN clustering using n = 200 * * * * * * * * * * * * * *
Homogeneity score:  0.41338483157974676
Completeness score:  0.6136479744870404
V-measure score:  0.4939915513586462
Adjusted Rand score:  0.20838513655272836
Adjusted mutual information score:  0.4929646448093636
```

Question 16:

The model (with min_cluster_size = 100) gives 9 clusters. And we have found around 3300 items that have a label of ‘-1’, and it refers to that data that does not belong to any cluster (they may belong to some clusters that contain less than 100 data points). They are classified as cluster 0 in the contingency matrix. (From the column with label ‘0’, we can find how many data points in each class are classified as ‘-1’, being to none clusters).



Question 17:

Based on our experiments, UMAP with n_components = 5 and metric = 'cosine' and K-Means(k = 20) have the best performance score (because almost all the measure scores are the highest), reaching v-measures about 0.582. This combination works the best because SVD and NMF are both linear methods, while UMAP is a non-linear technique that could preserve the local structure of the data. So, UMAP is able to capture the non-linear relationships between points in the data, and the results would be better by using ‘cosine’ distance metric. While the choice of hyperparameters in HDBSCAN highly affect the performance of the model, and given the data we have is well-separated, KMeans is a good choice here.

Question 18:

We first decided to normalize the data before we pass it into the K-Means (n_clusters = 20) algorithm (for dimensionality reduction, we used UMAP with n_components =) And the result has been improved by a little. The scalar we used is the StandardScaler(with_mean=False). The result we got is:
Homogeneity score: 0.5993390576636822
Completeness score: 0.6205075032379176
V-measure score: 0.6097396085438602
Adjusted Rand score: 0.4944043000970835
Adjusted mutual information score: 0.6084367375884226

Then, we tried to use Glove, but failed. Because we are not able to generate a keywords list for each data sample, while the original data is too large (contains too many words).

Part 2:

Question 19:

The pretrained VGG neural network have discriminative power for a custom dataset because the earlier layers of VGG neural networks(VGGNN) are trained to find low level and general features such as shapes, edges, and textures that can help to distinguish between different objects and classes in an image.Therefore, when VGGNN is used on custom dataset, it will act as a important feature extraction tool to extract meaningful features in custom dataset. Those extracted features will allow users to classify their custom dataset accurately and perform their own task.

Question 20:

The helper code created its own feature extraction neural network called FeatureExtractor. The first few layers are extracted from VGG16 feature layers. In other words, the helper code uses pretrained VGG16 feature extraction layers as its own feature extraction layer. In VGG -16 feature extraction layers, Conv2D layers are used to extract features and each Conv2D layer is followed by a ReLU layer to ensure no-linearlity. Conv2D layers apply multiple filters to the input image, and those filters with pretrained weights allow features such as edges, shapes to be extracted.

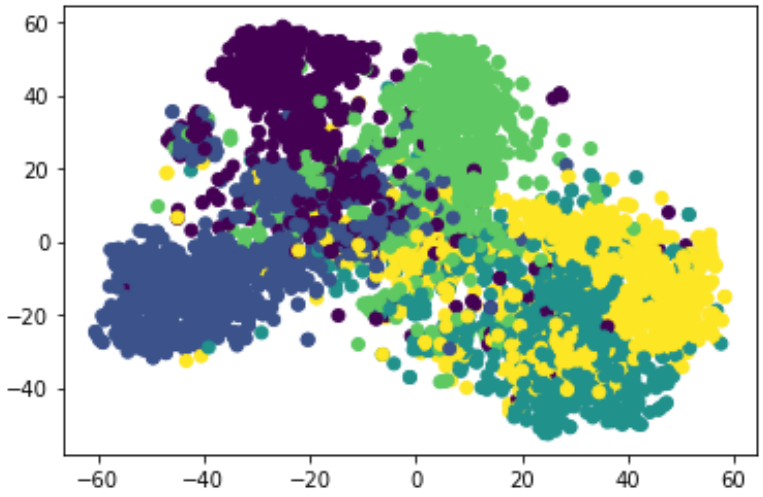
Question 21:

The size of the input image is (320*263 = 84160 pixels). The VGG network extracts 4096 features per image because each feature vector has 4096 dimensions.

Question 22:

The extracted features are pretty dense compared to the TF-IDF features. In the TF-IDF feature matrix, most of the cells/elements are 0. However, the cells in the extracted feature matrix we get from the neural network are all non-zero. Therefore, the extracted features from the VGG16 network are dense compared to that of TF-IDF.

Question 23:



After we map the features we extracted onto 2 dimensions with t-SNE, we observed clear signs of clustering from the plot.

Question 24:

In this question, we first tried to find an appropriate value for the hyperparameter min_cluster_size and min_samples in HDBSCAN (we directly used the 'f_all' as our fitting data, and did not apply any dimensionality reduction methods here). For min_cluster_size, we considered the following values (100, 200, 300), and for min_samples, we used (50, 100, 200). The combination of min_cluster_size & min_samples with the best rand_score is (300, 200). So, we used this combination in HDBSCAN for the following report.

This figure shows the rand_score report, the first number represents the dimensionality reduction being used (0: None, 1: SVD, 2: UMAP, 3. Autoencoder), the second number represents the Clustering model we used (0: K-Means, 1: Agglomerative Clustering, 2: HDBSCAN).

```
comb_scores_dic
```

```
{0.7079360694173067: ['0; 0'],
 0.6862000871875192: ['0; 1'],
 0.20358404572368982: ['0; 2', '1; 2', '3; 2'],
 0.7078806674672471: ['1; 0'],
 0.68676286999925: ['1; 1'],
 0.7989080023141083: ['2; 0'],
 0.7797151626819594: ['2; 1'],
 0.6171422248264604: ['2; 2'],
 0.7116075997216534: ['3; 0'],
 0.6986368595263505: ['3; 1']}
```

The best result (for rand score) is about 0.7989. The Clustering model used here is the K-Means with k = 5, and the dimensionality reduction method used here is the UMAP with n_components = 50.

```
l=list(comb_scores_dic.items())
l.sort(reverse=True)
l[0]
```

```
(0.7989080023141083, ['2; 0'])
```

Question 25:

The test accuracy of the MLP classifier on the original VGG features is: 0.9168937329700273.
For the accuracy, we just simply used # of correct predictions / total # of data.
Then, we used PCA (num_features = 50) for dimensionality reduction, and got a test accuracy: 0.885558583106267.
Yes, the performance of the model suffers with the reduced-dimension representations, might because the number of features has been reduced. Not significant.
Yes, HDBSCAN and other clustering models are sensitive to hyperparameter choices, i.e. the num_cluster_size and num_samples we chose might not be good enough for the data. While MLP is insensitive to the number of features (num_features = 50 in the case with PCA, and 4096 on the original VGG features), and it can be used for supervised learning tasks, and with the help of cuda, it is capable of learning complex non-linear relationships between features and labels. So, it makes sense that MLP's performance is better than that of methods in Q24.