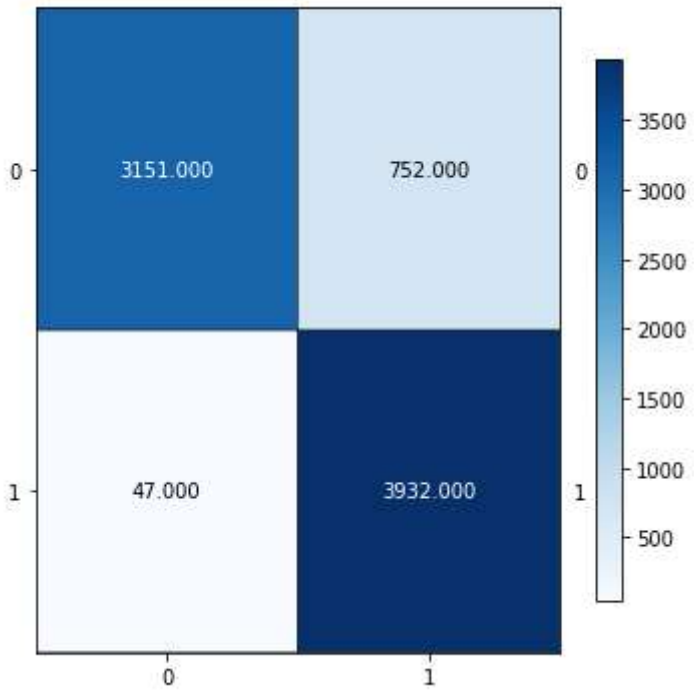
Project 2_ Report
Haoting Ni (905545789), Yikai Wang (905522085), Yuanxuan Fang (005949389)

Question 1:
The shape of TF-IDF matrix is (7882, 21798)

Question 2:

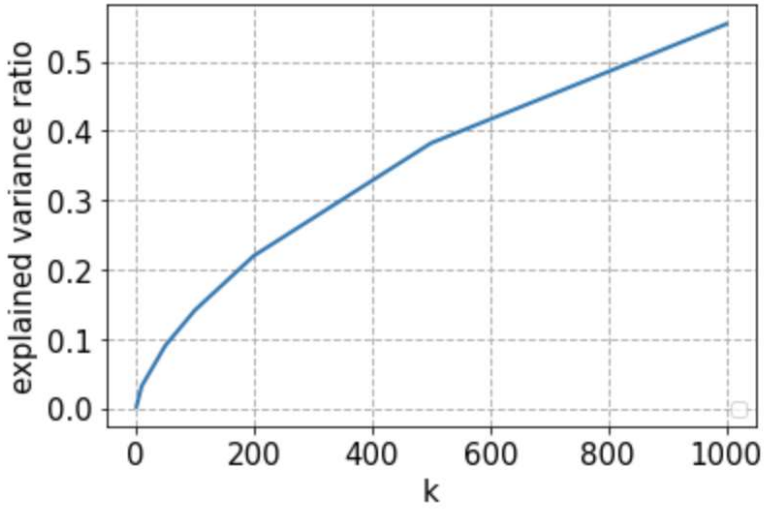


The contingency matrix is shown above. The contingency matrix is squared most of the time, but it also exists that is not squared.

Question 3:
Homogeneity: 0.5774035409987348
Completeness: 0.5926514149929692
V-measure: 0.5849281246876363
Adjusted Rand Index: 0.6355770387145221
Adjusted mutual information score: 0.5848896291295917

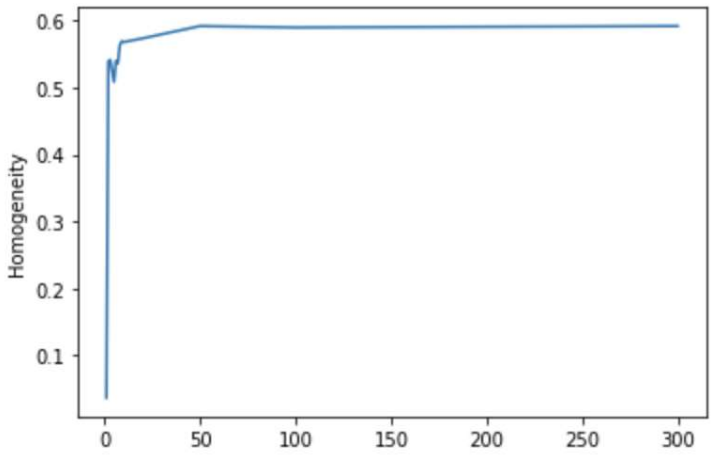
Question 4:

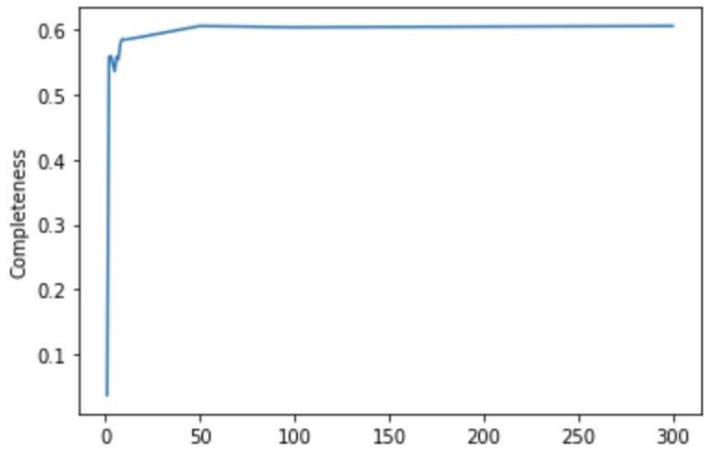
```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```

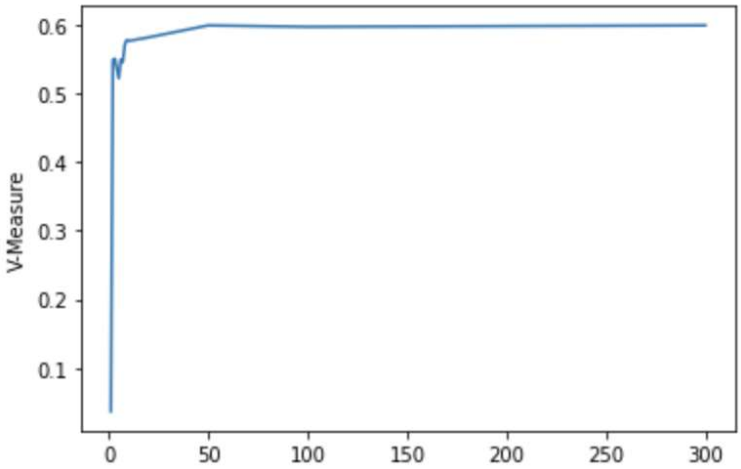


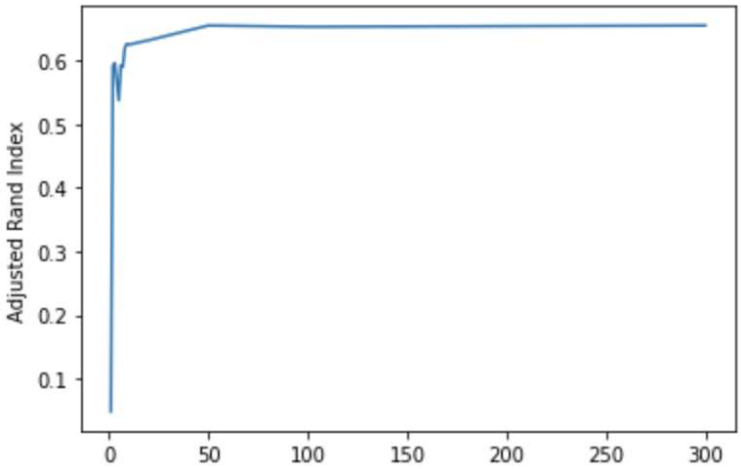
The plot is shown above as x-axis is k from 1 to 1000, y-axis is the explained variance ratio with respect to k.

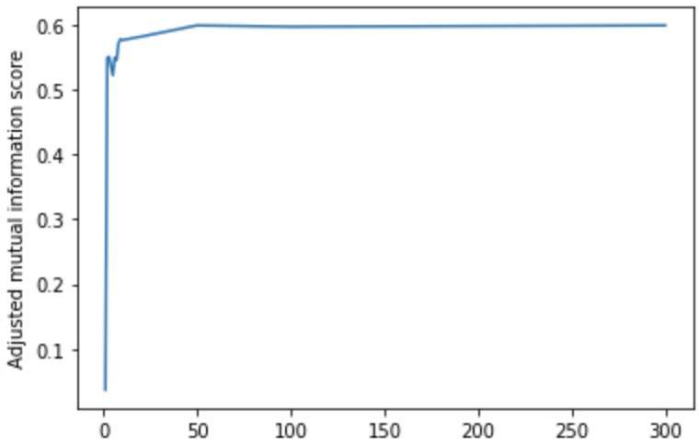
Question 5:
5 measure scores BY SVD:



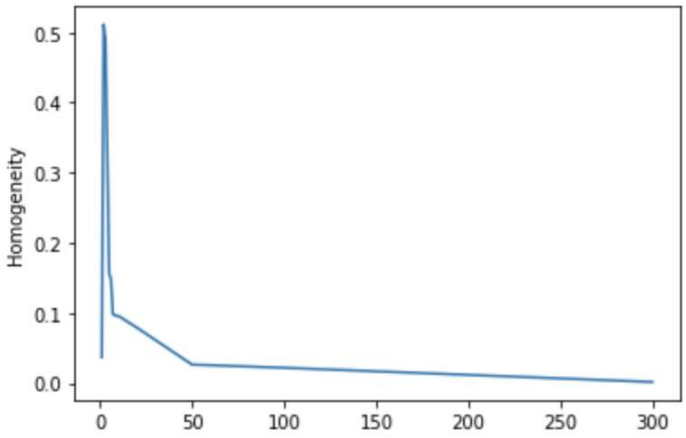


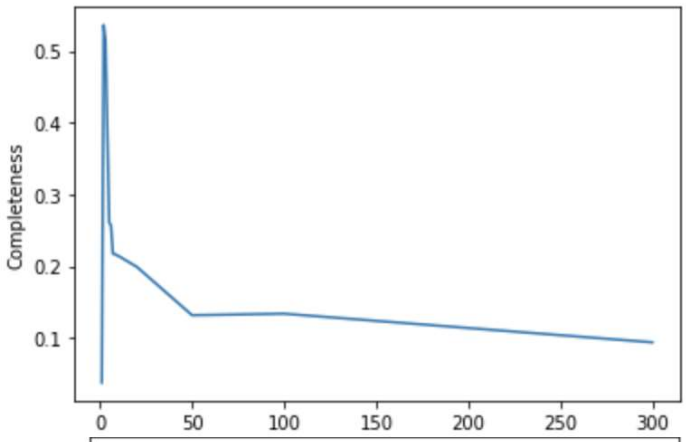

V-Measure
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

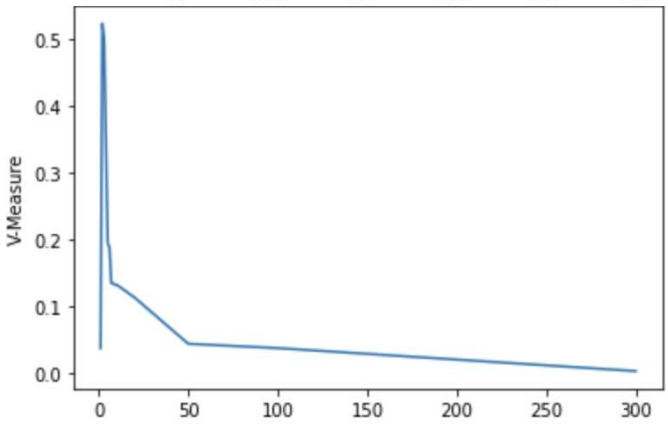

Adjusted Rand Index
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

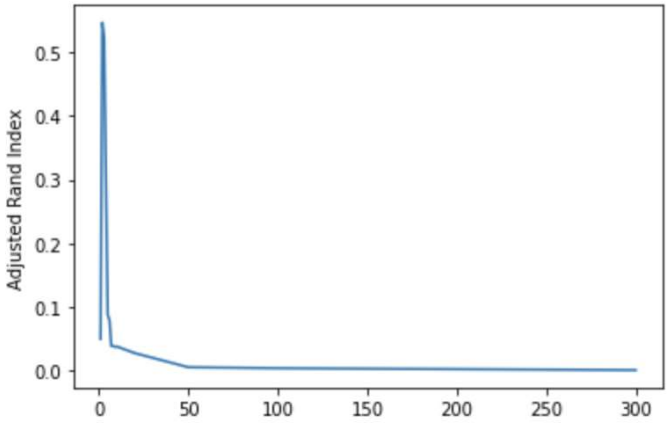

Adjusted mutual information score
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

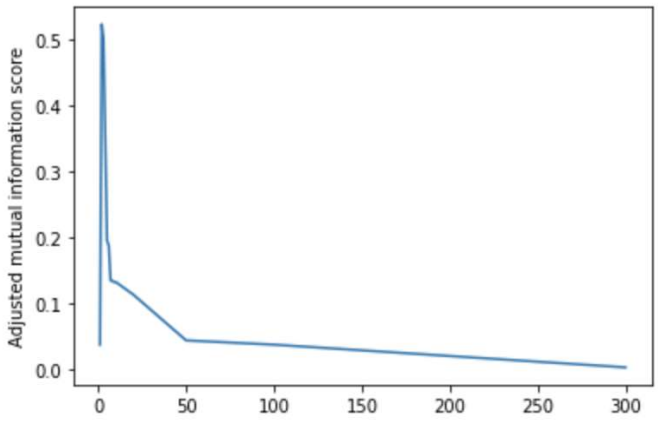
5 measure scores BY NMF











These plots are different r as x-axis with respect to the y-axis of Homogeneity, Completeness, V-measure, Adjusted Rand Index, and Adjusted mutual information score.

For SVD, the best r is 100, and for NMF, the best r is 2.

Question 6:

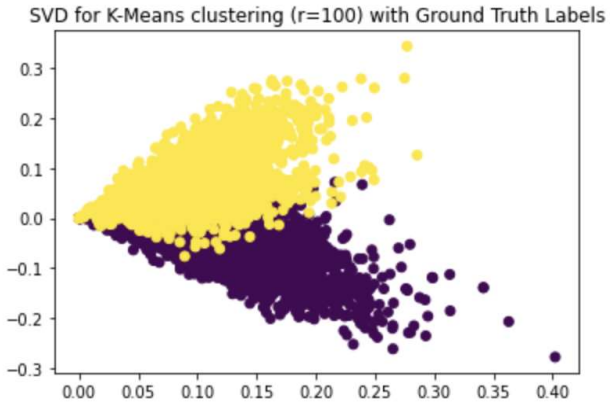
Non-monotonic behavior occurs because with a higher r, it carries more features and information. More information preserved should lead to a better clustering performance, but a large r also means more noise.

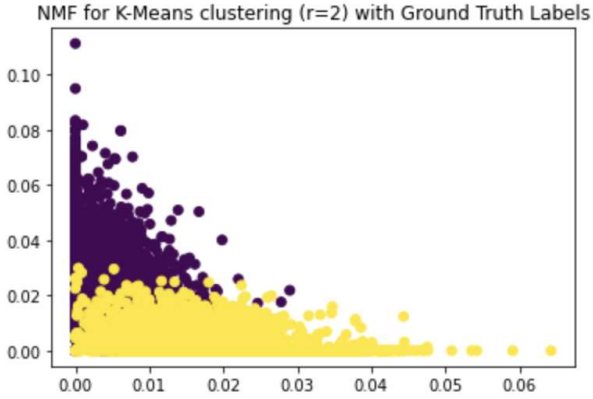
NMF shows a hard drop after r=2 because NMF only allows positive entries in reduced dimensional matrices.

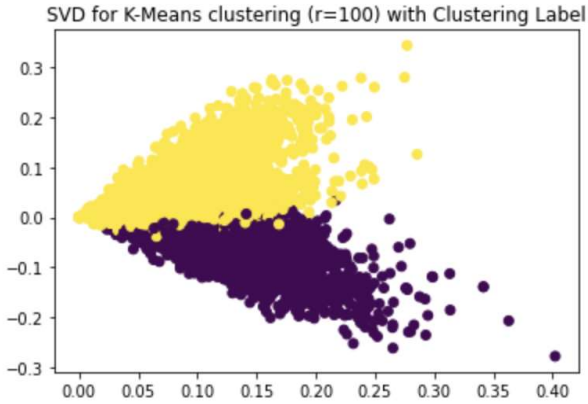
Question 7:

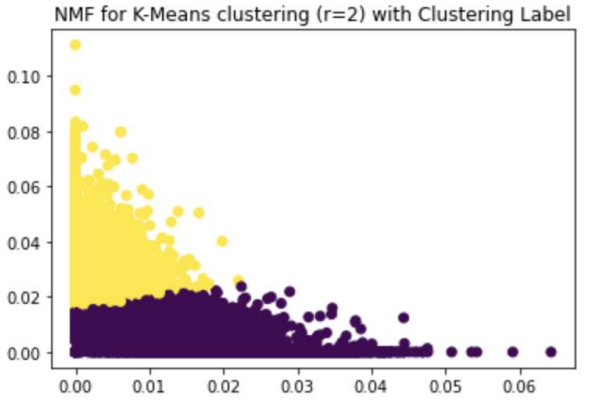
These average measures are a little less than what we did in Question 3.

Question 8:









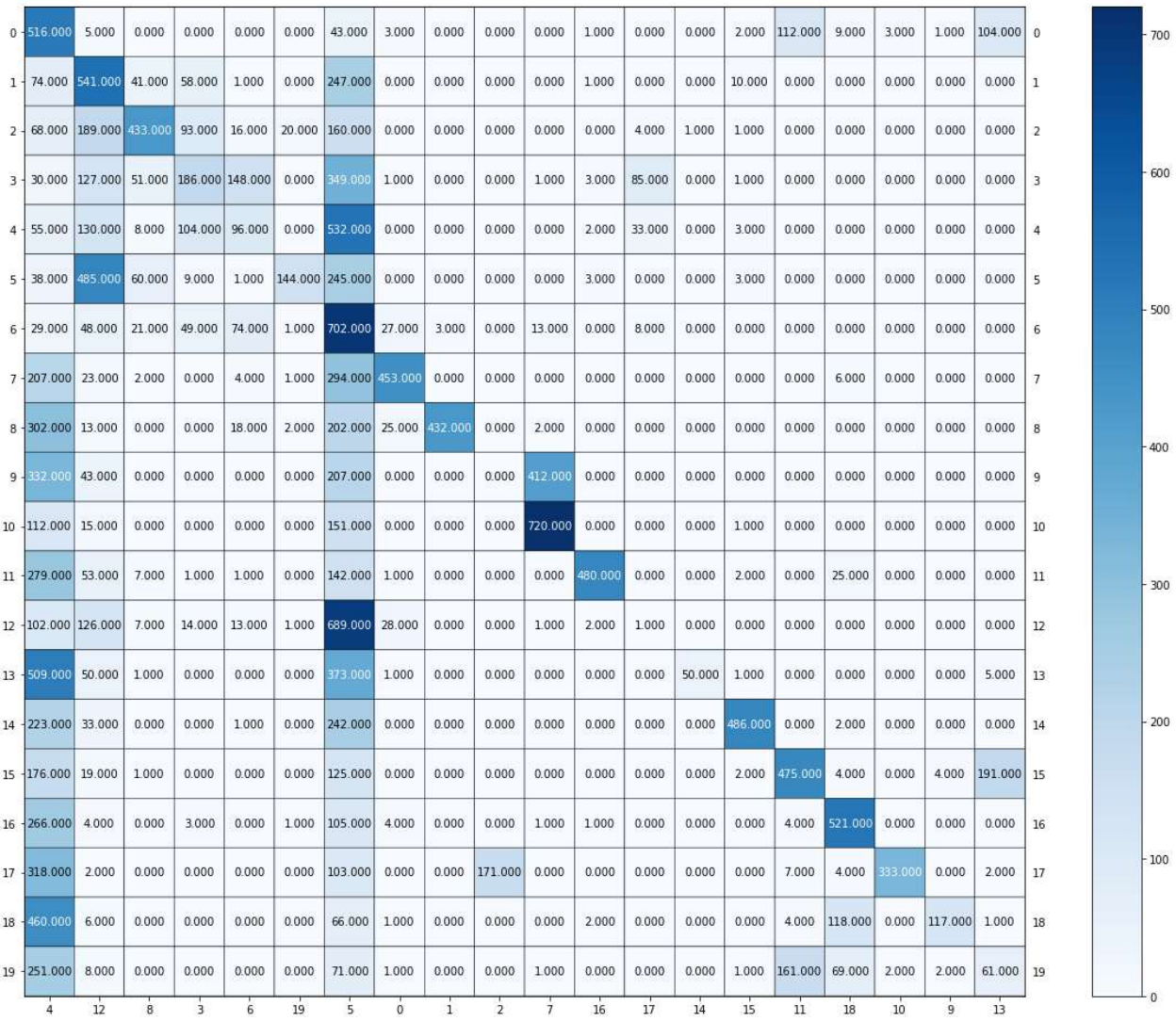
The visualization of the SVD and NMF for K-means clustering are shown above.

Question 9:

We observe that there are two clear clusters appearing on the graphs, and they separate by a minimal gap of euclidean distance. It barely draws a decision boundary between two clusters. However, it is not like K-means assumes that is centralized and clusters are far away from each other. So, the data is not ideal for K-means.

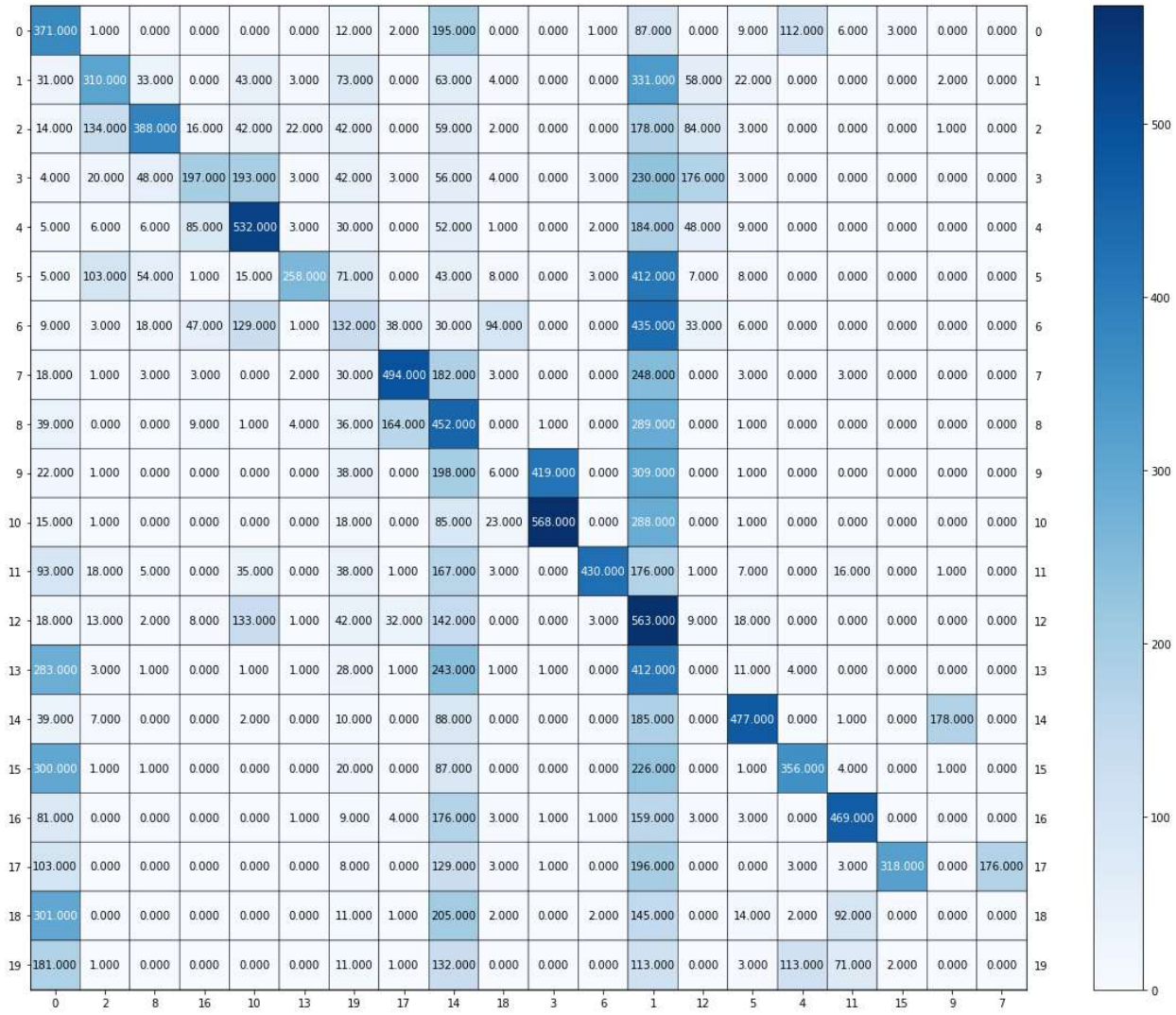
Question 10:

As the whole 20 classes loaded, the TF-IDF matrix has the shape of (18846, 42170). For SVD, we choose the components of 100, For NMF, we choose the components of 20, and we both get it from Question 8.



The contingency matrix of SVD for k-means is shown above and 5 metrics:

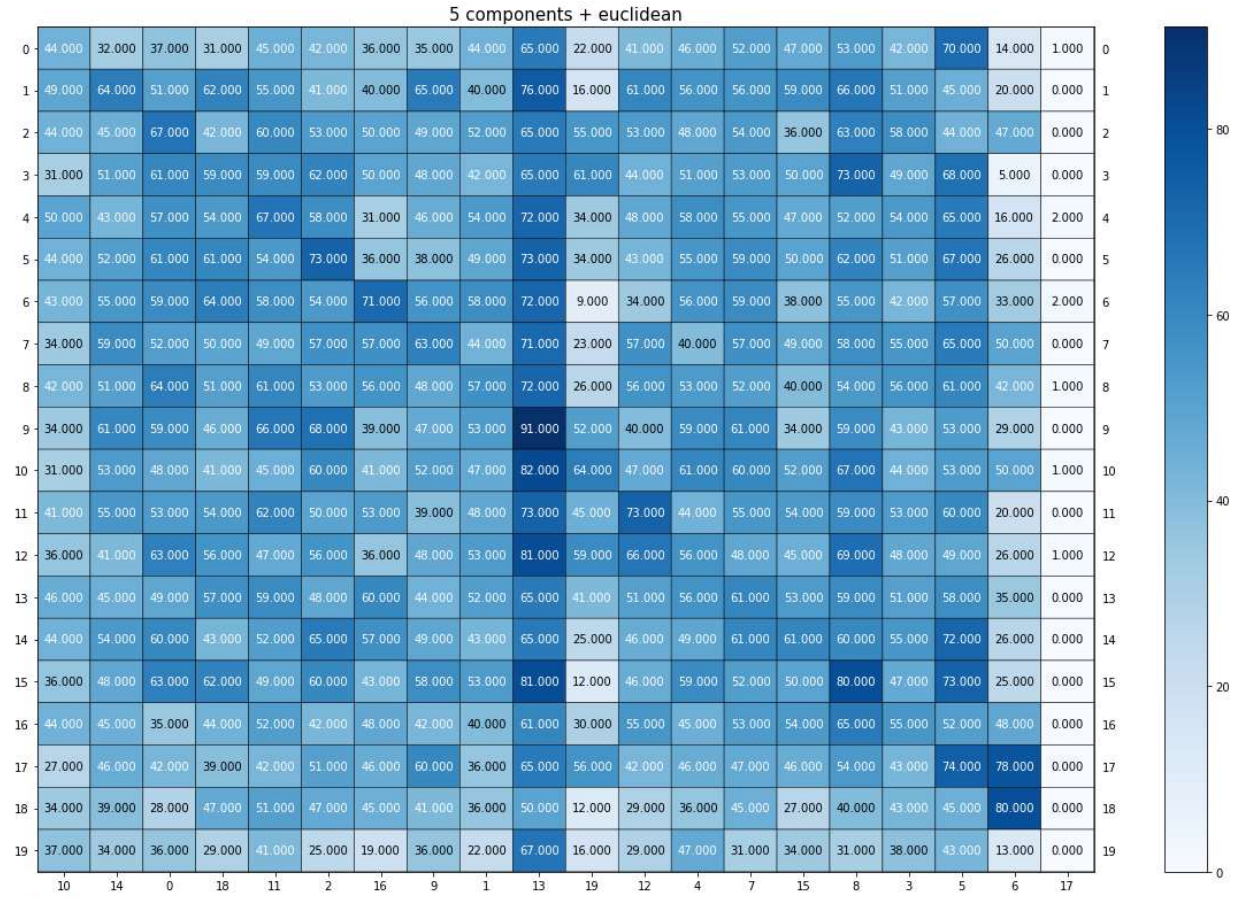
```
Homogeneity:  0.36344527780811314
Completeness:  0.4636394506312786
V-measure:  0.40747353479839793
Adjusted Rand Index:  0.12180153342657082
Adjusted mutual information score:  0.4053011885269787
```



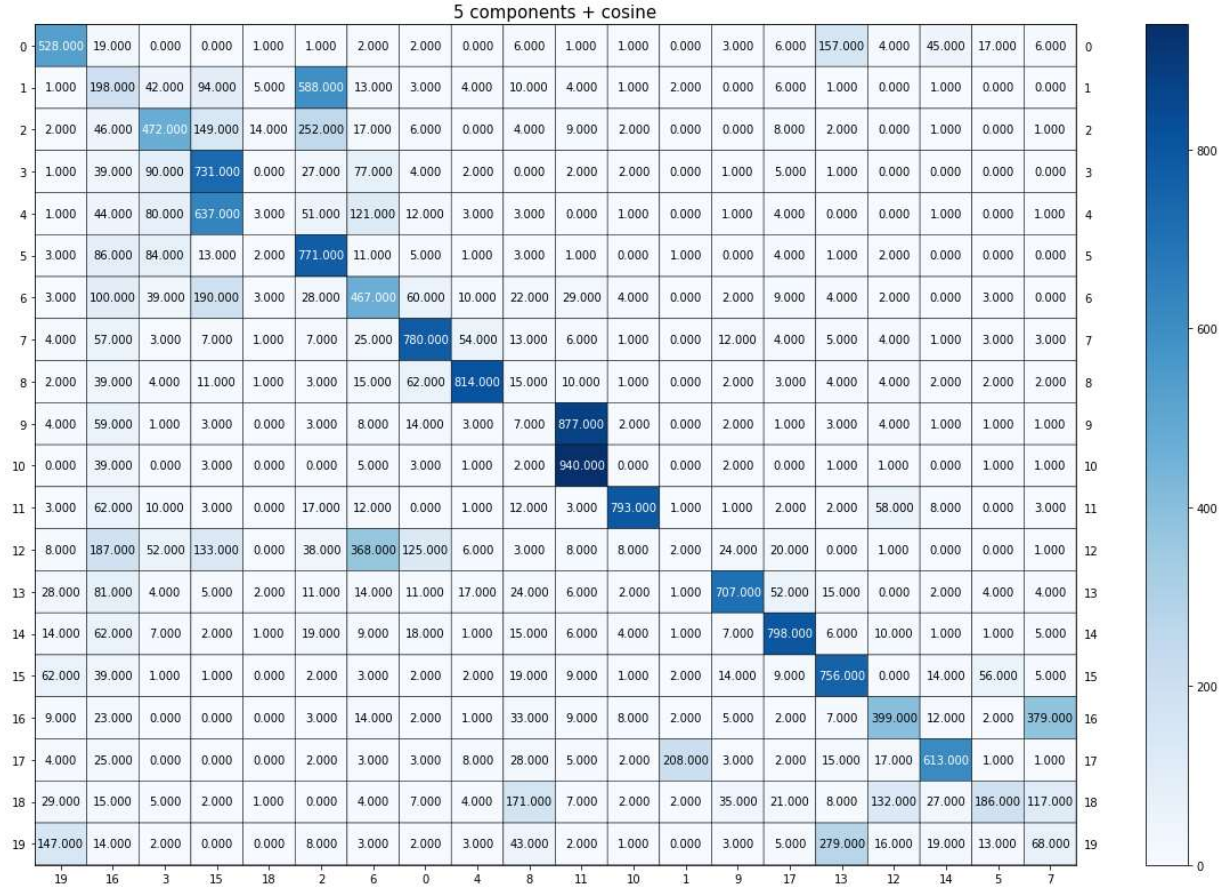
The contingency matrix of NMF for k-means is shown above
and 5 metrics:

```
Homogeneity:  0.33395741408685753
Completeness:  0.39777564656828923
V-measure:  0.36308357092882243
Adjusted Rand Index:  0.10406823198250942
Adjusted mutual information score:  0.3608372984487715
```

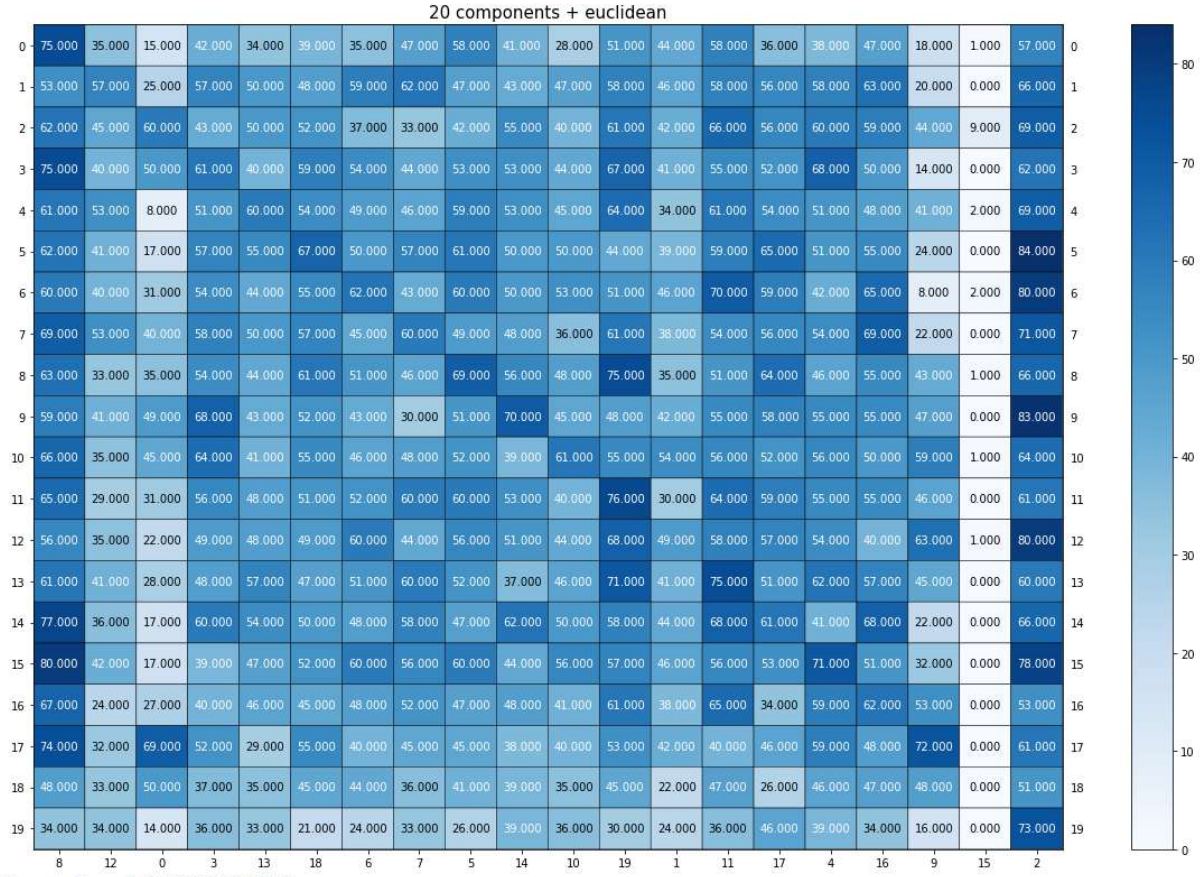
Question 11:



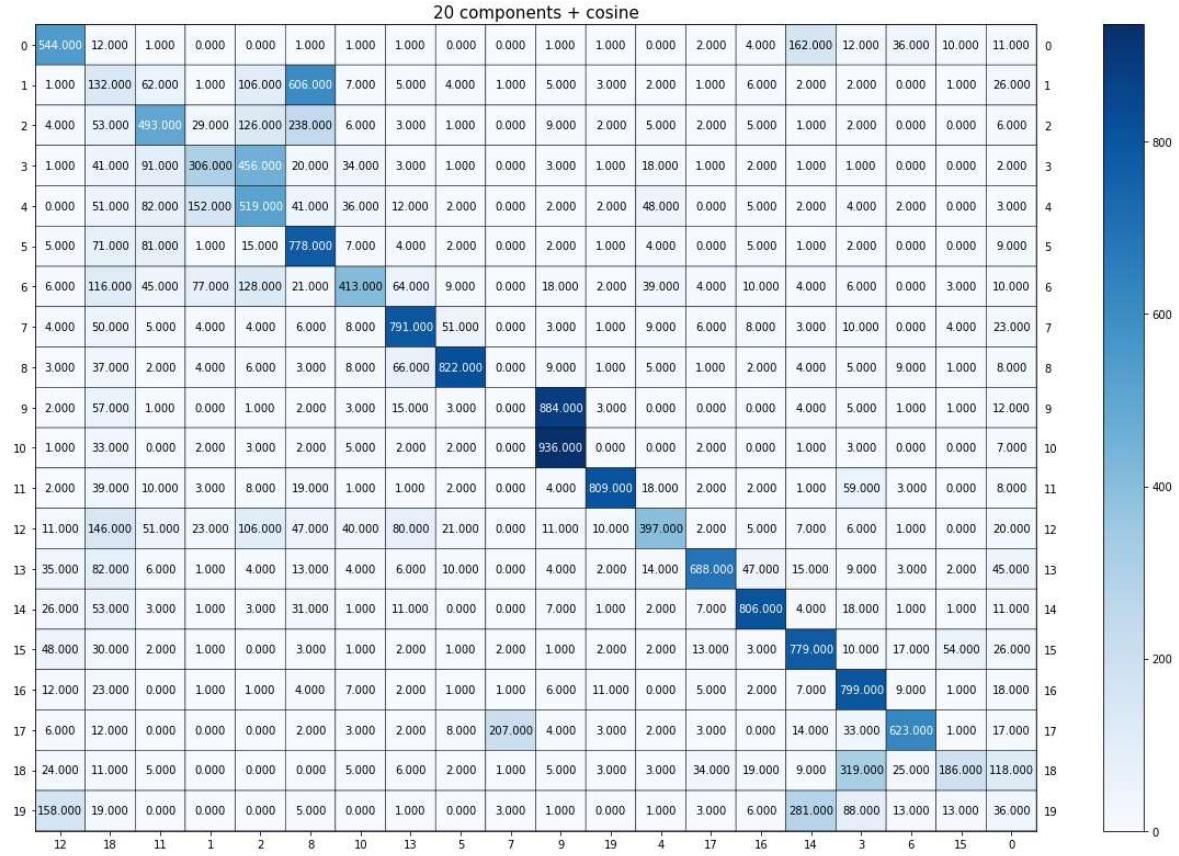

```
Homogeneity:  0.006963310345264541
Completeness:  0.007098643880587983
V-measure:  0.007030325881757074
Adjusted Rand Index:  0.000860714947023135
Adjusted mutual information score:  0.0038033666578713276
```



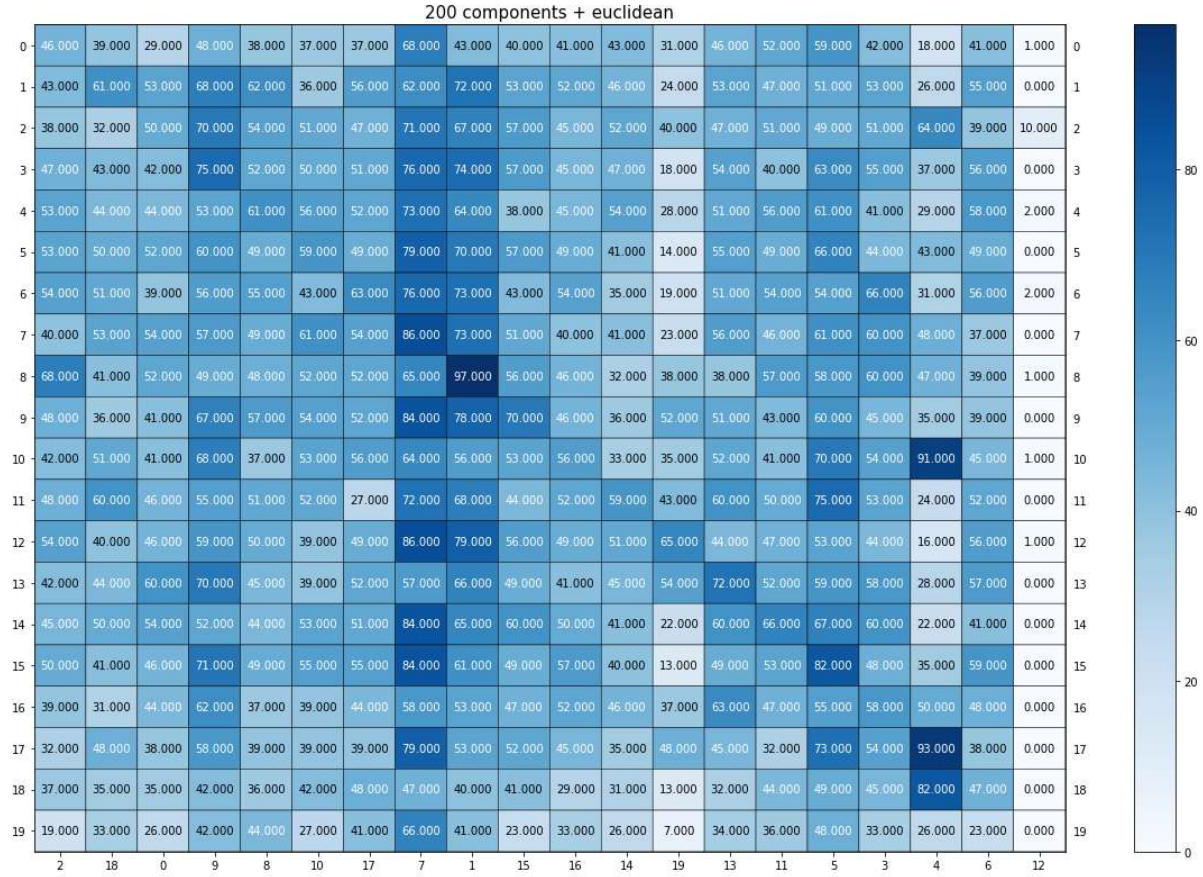
Homogeneity: 0.5579646383397878
Completeness: 0.5897530610088147
V-measure: 0.5734186265182245
Adjusted Rand Index: 0.42012638848880346
Adjusted mutual information score: 0.5719905310550398



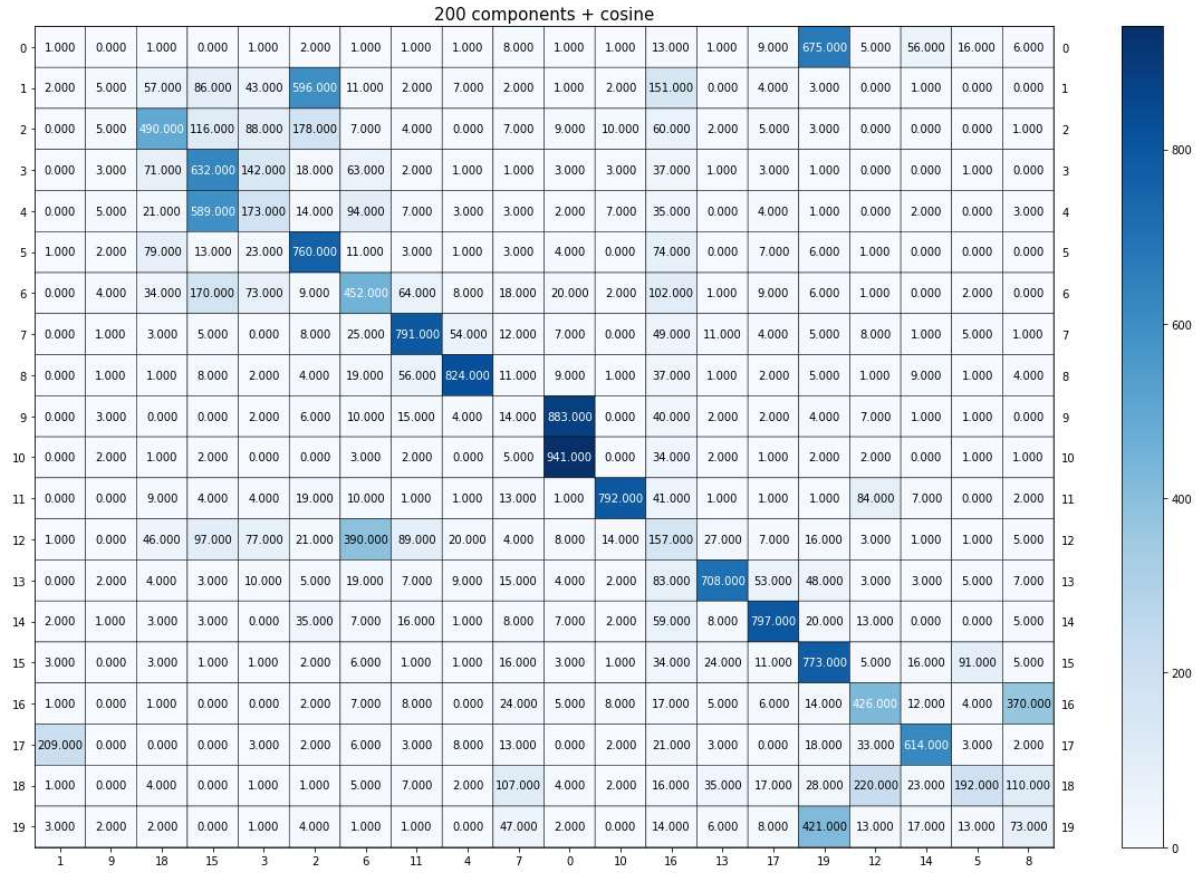

```
Homogeneity:  0.0067272342388865325
Completeness:  0.006856830499122388
V-measure:  0.006791414174403483
Adjusted Rand Index:  0.0007462319543346768
Adjusted mutual information score:  0.0035326420178268107
```



Homogeneity: 0.5727266672477204
Completeness: 0.5950927361971373
V-measure: 0.5836955243253182
Adjusted Rand Index: 0.44247696596196257
Adjusted mutual information score: 0.5823243993906418



Homogeneity: 0.007043579932128244
Completeness: 0.007179839386213435
V-measure: 0.007111056980711039
Adjusted Rand Index: 0.0009614342903204385
Adjusted mutual information score: 0.0038519780440671642



All these 6 combinations of the contingency matrix and five clustering evaluations are shown as above.

Question 12:

As the comparison from the five clustering evaluations of each component respected to euclidean and cosine, cosine distance performs better. Since cosine distance is not affected by the dimensions. On the other hand, the euclidean distance will perform less well in high dimension features. For each metric choice, r = 20, cosine distance UMAP with the highest adjusted rand index (accuracy).

Question 13:

So far, K-means clustering with sparse TF-IDF has the worst performance, since it does not reduce the dimension. For SVD reduced and NMF reduced, we found out that components r = 100 and 20 respectively are the best choice for SVD reduced and NMF reduced.

Question 14:

Agglomerative, for “ward” linkage criteria, five clustering evaluations are shown below.

```
Homogeneity:  0.5664705707502917
Completeness:  0.5827393082216585
V-measure:  0.5744897856643124
Adjusted Rand Index:  0.425330910828589
Adjusted mutual information score:  0.5730958277607423
```

Agglomerative, for “single” linkage criteria, five clustering evaluations are shown below.

```
Homogeneity:  0.016304604013505815
Completeness:  0.36263221868186957
V-measure:  0.03120612394483502
Adjusted Rand Index:  0.0005136327378515986
Adjusted mutual information score:  0.026289810915222936
```

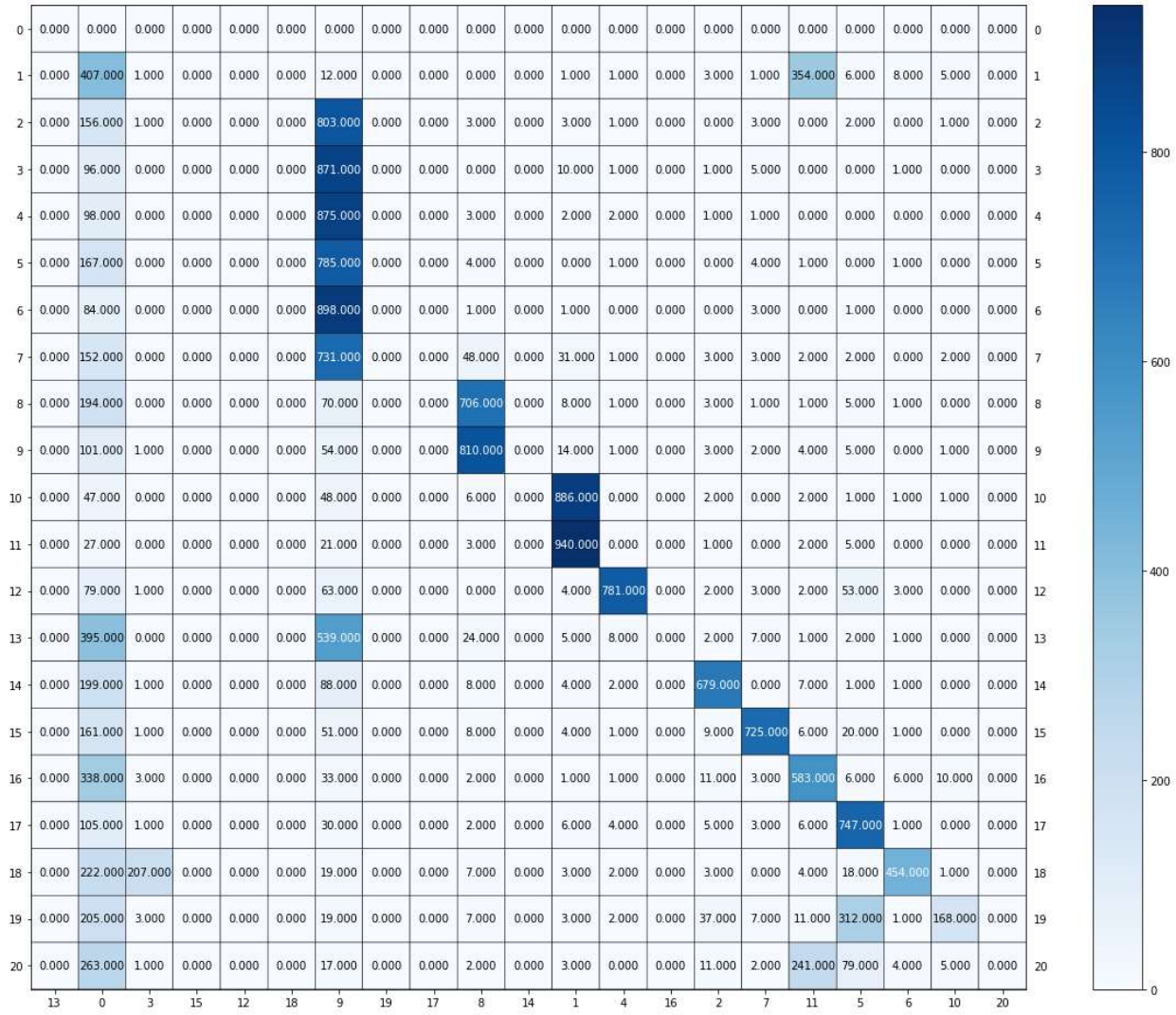
Question 15:

HDBSCAN five clustering evaluations with respect to min_clsuter_size of 20, 100, 200 are shown below:

```
20
Homogeneity:  0.4213436898781575
Completeness:  0.43394333368258675
V-measure:  0.42755070607911316
Adjusted Rand Index:  0.06949144572608194
Adjusted mutual information score:  0.4149583559942783
100
Homogeneity:  0.4336789202796464
Completeness:  0.6375099981242074
V-measure:  0.5162014709150505
Adjusted Rand Index:  0.21859244455476776
Adjusted mutual information score:  0.5151237897905137
200
Homogeneity:  0.41815386406777416
Completeness:  0.6154862906128278
V-measure:  0.49798369294200345
Adjusted Rand Index:  0.2115996277850882
Adjusted mutual information score:  0.4969685477987435
```

Question 16:

The contingency matrix for HDBSCAN is shown down below:



"-1" means the noise that has not been classified into any class labels.

By observing the HBDSCAN contingency matrix, we have 11 clusters. For example, cluster 9 has 2-7 classes combined.

Question 17:

{'SVD5 + K_MEAN + 10': 0.09679702072269666,
'SVD5 + K_MEAN + 20': 0.13344166546423533,
'SVD5 + K_MEAN + 50': 0.1089927051735795,
'SVD20 + K_MEAN + 10': 0.08588229350837168,
'SVD20 + K_MEAN + 20': 0.12297443619840195,
'SVD20 + K_MEAN + 50': 0.15404690479353797,

'SVD200 + K_MEAN + 10': 0.09918873421997113,
'SVD200 + K_MEAN + 20': 0.0985557850547592,
'SVD200 + K_MEAN + 50': 0.13006995979150218,
'SVD5 + AGG + 20': 0.11567593953990528,
'SVD20 + AGG + 20': 0.16830745559066143,
'SVD200 + AGG + 20': 0.12443854396797484,
'SVD5 + HDB + 100': -9.419666101039764e-06,
'SVD20 + HDB + 100': 0.0,
'SVD200 + HDB + 100': 0.0,
'SVD5 + HDB + 200': 0.0,
'SVD20 + HDB + 200': 0.0,
'SVD200 + HDB + 200': 0.0,
'NMF200 + K_MEAN + 10': 0.004249134517982495,
'NMF200 + K_MEAN + 20': 0.035679745688306376,
'NMF200 + K_MEAN + 50': 0.04527096828501377,
'NMF5 + K_MEAN + 10': 0.08986787288671771,
'NMF5 + K_MEAN + 20': 0.1035619753598765,
'NMF5 + K_MEAN + 50': 0.09109340339245127,
'NMF20 + K_MEAN + 10': 0.03177639947680652,
'NMF20 + K_MEAN + 20': 0.10406823198250942,
'NMF20 + K_MEAN + 50': 0.13559778428965924,
'NMF5 + AGG + 50': 0.10533647441851773,
'NMF20 + AGG + 50': 0.10990489971472493,
'NMF200 + AGG + 50': 0.04625650654815621,
'NMF5 + HDB + 100': 0.034730587914443604,
'NMF20 + HDB + 100': 0.0,
'NMF200 + HDB + 100': 0.0,
'NMF5 + HDB + 200': 0.0,
'NMF20 + HDB + 200': 0.0,
'NMF200 + HDB + 200': 0.0,
'UMAP5 + K_MEAN + 10': 0.36760645530133773,
'UMAP5 + K_MEAN + 20': 0.4690091818375202,
'UMAP5 + K_MEAN + 50': 0.402091823310506,
'UMAP20 + K_MEAN + 10': 0.3307463748469388,
'UMAP20 + K_MEAN + 20': 0.46765868857262133,
'UMAP20 + K_MEAN + 50': 0.3696683148548335,
'UMAP200 + K_MEAN + 10': 0.32794978430798466,
'UMAP200 + K_MEAN + 20': 0.4199456856826519,
'UMAP200 + K_MEAN + 50': 0.3986889321980805,
'UMAP200 + AGG + 20': 0.4440650914683766,
'UMAP5 + AGG + 20': 0.4207952176167666,
'UMAP20 + AGG + 20': 0.4366819550548756,
'UMAP5 + HDB + 100': 0.20593084183962104,
'UMAP20 + HDB + 100': 0.19523198997207222,

'UMAP200 + HDB + 100': 0.20379728827942695,
'UMAP5 + HDB + 200': 0.22302916559650812,
'UMAP20 + HDB + 200': 0.2220502699941576,
'UMAP200 + HDB + 200': 0.22108064519743173}
'None + K_MEAN + 10': 0.10635425868004754,
'None + K_MEAN + 20': 0.11499566821171768,
'None + K_MEAN + 50': 0.12343873501123943,
'None + HDB + 100': 0.0,
'None + HDB + 200': 0.0
'NoneAGG + 20': 0.1594899626397469}

Thus, from the result we get from above, the best adjusted rand index we get from the combination of UMAP20 + K_MEAN + 20': 0.46765868857262133, which is UMAP with 20 components with k_mean and parameter with 20.

Question 19:

We know that VGG is a pre-trained model for image detection on different datasets. We also assume that it can extract its features from the other datasets. Since the basic structure of the VGG is a neural network, the gradient in the hidden layers can make sure we are able to capture some features of the image in the dataset. Furthermore, the image dataset is large, and we can make sure VGG has the ability to learn and have discriminative power to the custom dataset.

Question 20:

It first downloads the flower dataset and extracts its images. Then, it defines a class of FeatureExtractor, with _init_, and forward function in it. It extracts the feature layers and average pooling layer from the VGG, reduces the obtained feature map size into a one-dimensional vector, then passes it through the final fully-connected layer to get the final features.

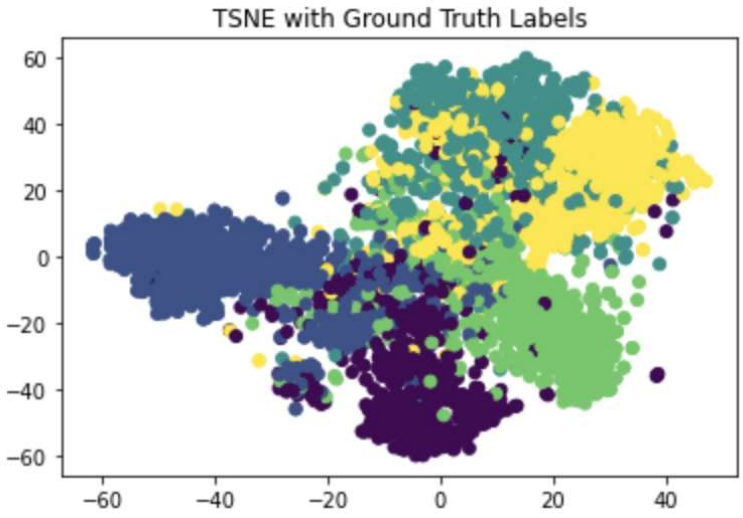
Question 21:

There are many different pixels in the original images. It has (250,320), (320,263). After VGG transformation, each picture has (224,224). In the end, it extracts 4096 features for each picitures.

Question 22:

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

Question 23:



The above plot is the TSNE visualization in 2-D. We can see that green, blue, and dark red has clear boundary line between each other, even though there are some spots landing into the other color regions. The yellow and dark green overlap in the region ([0-20], [20,40]). So they might have similarities in that clusters.

Question 24:

{'SVD50 + K_MEAN + 5': 0.19146937613851173,
'SVD50 + AGG + 5': 0.2334745470751877,
'SVD50 + HDB + 5sample 5': 0.005498837928082272,
'SVD50 + HDB + 5sample 20': 0.0,
'SVD50 + HDB + 50sample 5': 0.0,
'SVD50 + HDB + 50sample 20': 0.0,
'SVD50 + HDB + 100sample 5': 0.0,
'SVD50 + HDB + 100sample 20': 0.0,
'UMAP50 + K_MEAN + 5': 0.4676822988641824,
'UMAP50 + AGG + 5': 0.48631519971009973,
'UMAP50 + HDB + 5sample 5': 0.09411014415380671,
'UMAP50 + HDB + 5sample 10': 0.09494009559863244,
'UMAP50 + HDB + 10sample 5': 0.09494009559863244,
'UMAP50 + HDB + 10sample 10': 0.09483164455701254,
'AUTO50 + K_MEAN + 5': 0.1988387986337919,
'AUTO50 + AGG + 5': 0.14208845180661883,
'AUTO50 + HDB + 5sample 5': 0.020317450403088272,
'AUTO50 + HDB + 5sample 10': 0.023403788945569347,
'AUTO50 + HDB + 10sample 5': 0.001313600583448493,
'AUTO50 + HDB + 10sample 10': 0.0,
'NONE50 + K_MEAN + 5': 0.19447072328729068,
'NONE + K_MEAN + 5': 0.19447072328729068,
'NONE + AGG + 5': 0.18855278251971858,

'NONE + HDB + 5sample 5': 0.006705947729476718,
'NONE + HDB + 5sample 10': 0.0,
'NONE + HDB + 10sample 5': 0.006705947729476718,
'NONE + HDB + 10sample 10': 0.0}

Thus, from the result we get above, the best-adjusted rand index we get from the combination of 'UMAP50 + AGG + 5': 0.48631519971009973, which is UMAP with 50 components and AGG with parameter 5.

Question 25:

```
class MLP(torch.nn.Module):
    def __init__(self, num_features):
        super().__init__()
        self.model = nn.Sequential(
            nn.Linear(num_features, 1280),
            nn.ReLU(True),
            nn.Linear(1280, 640),
            nn.ReLU(True),
            nn.Linear(640, 5),
            nn.LogSoftmax(dim=1)
        )
        self.cuda()


    def forward(self, X):
        return self.model(X)

    def train(self, X, y):
        X = torch.tensor(X, dtype=torch.float32, device='cuda')
        y = torch.tensor(y, dtype=torch.int64, device='cuda')

        self.model.train()

        criterion = nn.NLLLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=1e-3, weight_decay=1e-5)

        dataset = TensorDataset(X, y)
        dataloader = DataLoader(dataset, batch_size=128, shuffle=True)

        for epoch in tqdm(range(100)):
            for (X_, y_) in dataloader:
                ##################################
                optimizer.zero_grad()

                criterion(self(X_), y_).backward()
                optimizer.step()
                ##################################

        return self

    def eval(self, X_test, y_test):
        dataset = TensorDataset(torch.tensor(X_test, dtype=torch.float32, device='cuda'), torch.tensor(y_test, dtype=torch.int64, device='cuda'))
        dataloader = DataLoader(dataset, batch_size=64, shuffle=True)

        nums = 0
        count = 0

        for (X,y) in dataloader :
          output = self(X)
          nums += y.shape[0]
          count  += (torch.max(output,1)[1]==y).sum().item()


        return count/nums
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(f_all, y_all, test_size=0.2)
mlp = MLP(f_all.shape[1])
mlp.train(X_train, y_train)
mlp.eval(X_test,y_test)
```

```
100%|██████████| 100/100 [00:08<00:00, 11.13it/s]
0.8923705722070845
```

```
autoencode = Autoencoder(50).fit_transform(f_all)

X_train, X_test, y_train, y_test = train_test_split(autoencode, y_all, test_size=0.2)
mlp = MLP(autoencode.shape[1])
mlp.train(X_train, y_train)
mlp.eval(X_test,y_test)
```

```
100%|██████████| 100/100 [00:24<00:00,  4.12it/s]
100%|██████████| 100/100 [00:07<00:00, 13.65it/s]
0.8896457765667575
```

As the above result showed, the test accuracy of the MLP classifier on the original VGG features was 0.8923; the test accuracy of the MLP classifier on the reduced VGG features was 0.8896. Since the dimension is reduced, the performance does lower a little, but it is not significant. We can see, the accuracy of MLP is way better than what we got from Q24. This means that MLP improves the performance of classification tasks.