EE219 Project2

Clustering

Qidi Sang 705028670
Hui Wang 205036597

1. Build the TF-IDF Matrix

In order to capture the importance of a word to a document in a corpus, we use the Term Frequency-Inverse Document Frequency (TFxIDF) metric. Different from Project1 using supervised learning, clustering is unsupervised learning. Therefore all the operations in Project2 were targeted at all subsets of 20newsgroup instead of dividing them into training/testing subsets. Before transforming the documents into TF-IDF vectors, we processed the document in the following steps:

(1) Exclude numbers and punctuations, including !"#$%&()*+,-./:;<=>?@[\\]^_`{|}, since we only care about words in each documents;
(2) Token each documents into words and vectorize them, using CountVectorizer function;
(3) Exclude common stop words, by setting the parameter “stop_words” in CountVectorizer function;

Here we used min_df = 3, which means only extract a term when it appears more than three times.
And then we got the TFxIDF vector representation of each document, using TfidfTransformer function.
The result of building the TF-IDF matrix is:

```
------------------------------
The dimensions of the TF-IDF matrix:
(7882, 23793)
------------------------------
```

2. Apply K-means Clustering

After getting the TFxIDF vector representation of the data, we applied the K-means clustering with k = 2 (2 clusters) using the TF-IDF data we got in step1.
(a) Inspect the contigency matrix of the clustering result.
The result we got are as below:

```
------------------------------
Contingency_matrix:
[[   4 3899]
[1733 2246]]
------------------------------
```

Similar to confusion matrix in supervised learning, contingency matrix in unsupervised learning evaluates the performance of clustering algorithm. The larger two values of the contingency matrix are not in the same diagonal, which means that the clustering result is not very good. Thus we need to further process the data.

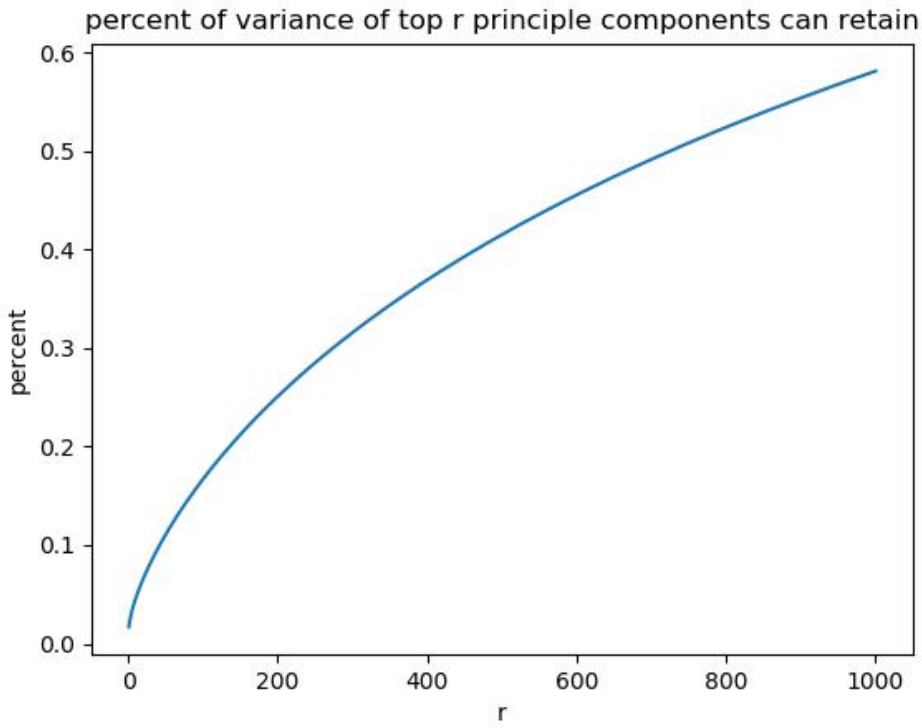
(b) Evaluate the clustering result using five measurements: homogeneity score, completeness score, V-measure, adjusted rand score, adjusted mutual info score.
The result we got are as below:

```
------------------------------
Homogeneity_score = 0.256347
Completeness_score = 0.336898
Adjusted_rand_score = 0.184014
V_measure_score = 0.291154
Adjusted_mutual_info_score = 0.256279
------------------------------
```

Here we used five different methods to measure the purity for a given partition of the data points with respect to the ground truth. Homogeneity score measures how purely the clusters contain only contain data points that belong to one single class. Completeness score is a measure of whether all data points of a class are assigned to the same cluster. Both of these scores span between 0 and 1, where 1 stands for perfect clustering. V-measure means the harmonic average of homogeneity score and completeness score. Adjusted rand score is similar to accuracy score, which counts all pairs of points that both fall either in the same cluster and the same class or in different clusters and different classes. And adjusted mutual info score measures the mutual information between the cluster label distribution and the ground truth label distributions. Higher the scores are, better performance the clustering algorithm get.
The scores we got are quite low, which still shows that the performance of clustering results is not ideal. We need to further process the data to get a better performance of clustering.

3. Preprocess the Data

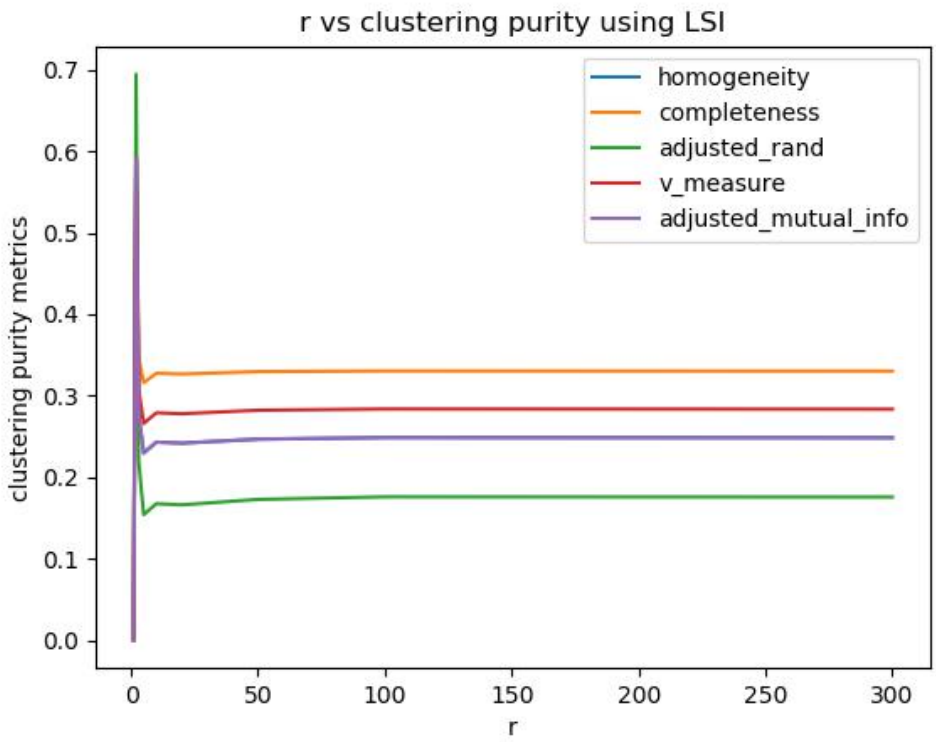
(a) Dimensionality reduction
(i) Find the effective dimension of the data.



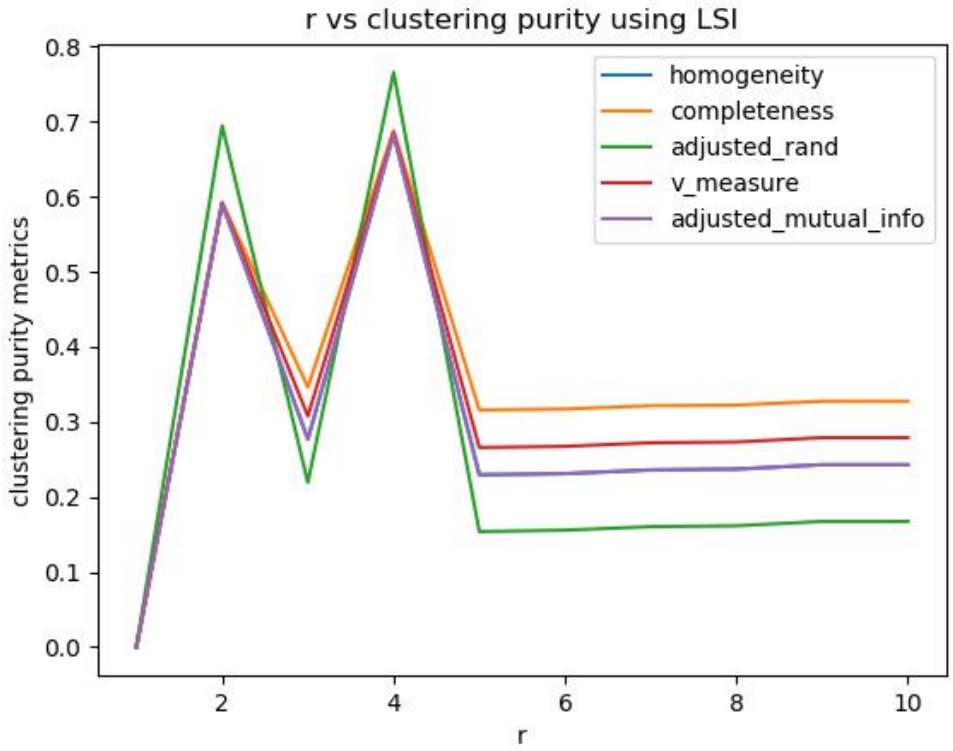
We can see that even the top 1000 principle components can only retain about 60% of variance.

(ii) Use LSI/NMF to reduce the dimension of the data.

(1) using LSI



We can see that the spike arise at 0~10, thus we choose r = 0 to 10 to test again.



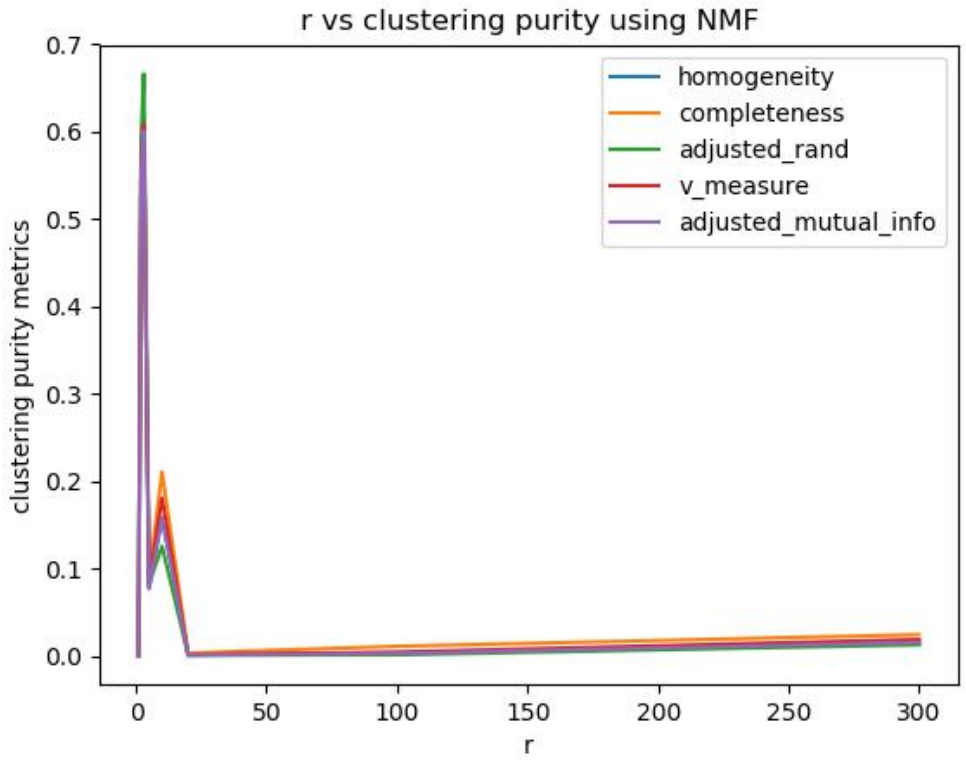
We can see the best r is 4 for LSI even it is not listed in given set. Considering its higher purity scores and it retains more variances, we decided the best r for LSI should be 4.

Here are the contingency matrices for different r's:

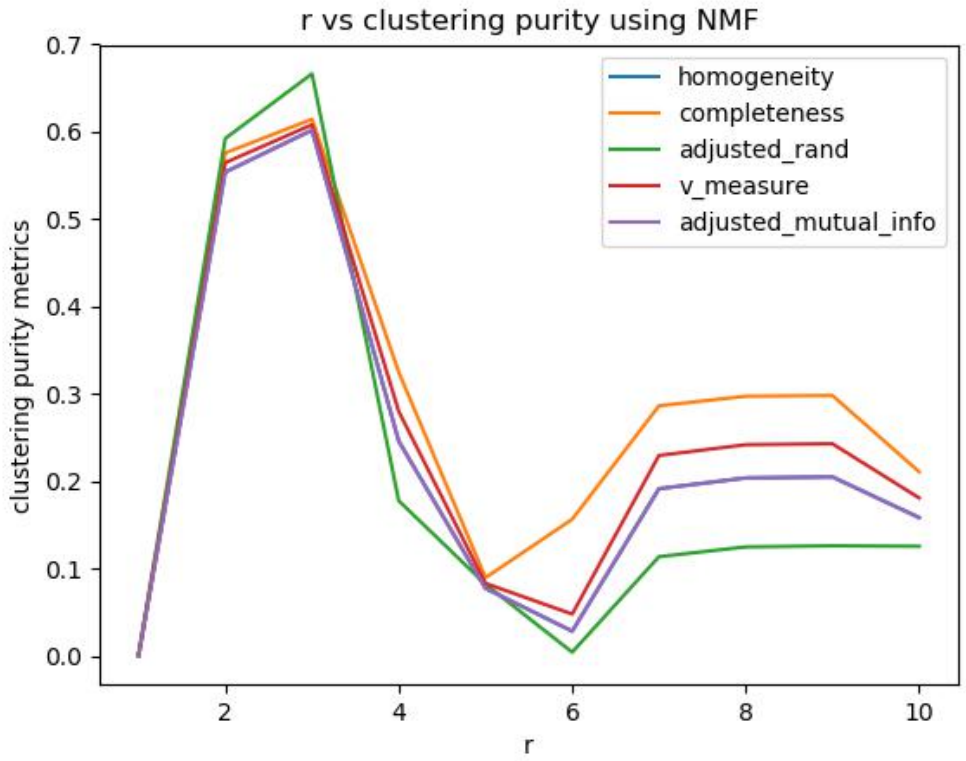
```
Contingency matrix when r = 1 using LSI
[[1727 2176]
 [1679 2300]]

Contingency matrix when r = 2 using LSI
[[3471  432]
 [ 224 3755]]

Contingency matrix when r = 3 using LSI
[[3887   16]
 [2079 1900]]

Contingency matrix when r = 5 using LSI
[[   5 3898]
 [1590 2389]]

Contingency matrix when r = 10 using LSI
[[3900    3]
```

```
 [2325 1654]]

Contingency matrix when r = 20 using LSI
[[   3 3900]
 [1647 2332]]

Contingency matrix when r = 50 using LSI
[[   4 3899]
 [1680 2299]]

Contingency matrix when r = 100 using LSI
[[   5 3898]
 [1696 2283]]

Contingency matrix when r = 300 using LSI
[[   5 3898].
 [1695 2284]]
```

(2) using NMF



We can see that the spike arise at 0~10, thus we choose r = 0 to 10 to test again.



We can see the best r is 3 for NMF.

Here are the contingency matrices for different r's:

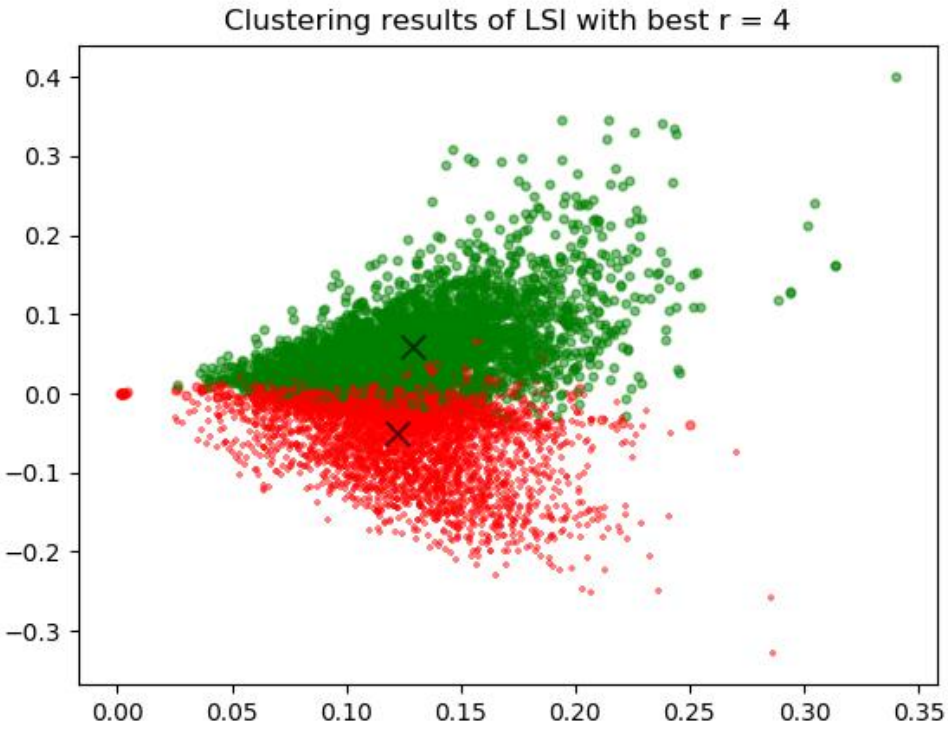
```
Contingency matrix when r = 1 using NMF
[[1727 2176]
 [1679 2300]]

Contingency matrix when r = 2 using NMF
[[ 886 3017]
 [3956   23]]

Contingency matrix when r = 3 using NMF
[[3228  675]
 [  50 3929]]

Contingency matrix when r = 5 using NMF
[[3355  548]
 [2265 1714]]

Contingency matrix when r = 10 using NMF
[[ 131 3772]
 [1564 2415]]

Contingency matrix when r = 20 using NMF
[[3705  198]
```

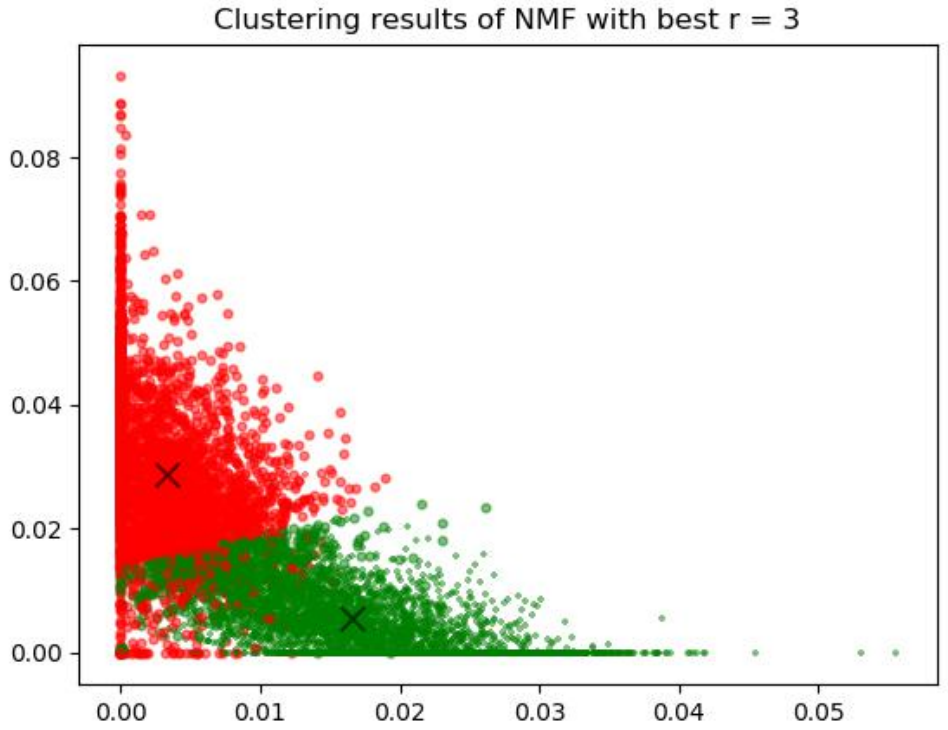
```
 [3830  149]]

Contingency matrix when r = 50 using NMF
[[3696  207]
 [3843  136]]

Contingency matrix when r = 100 using NMF
[[ 204 3699]
 [ 114 3865]]

Contingency matrix when r = 300 using NMF
[[3076  827]
 [3555  424]]
```

For both LSI and NMF, the purity measures experience a great drop when r is larger than 10. This is because in higher dimension the Euclidean distance is not a good metric any more since the distances between data points tend to be the same, which makes clustering not very effective.

4. Visualize and Apply Normalize/Logarithm Transformation

(a) Visualize the performance of the case with best clustering results.
After dimension reduction in Part3, we found that the best r for LSI is 4 and best r for NMF is 3. Thus we set the function parameter n_component = 4, 3 , respectively, in TruncatedSVD and NMF and got different clustering results by applying these two dimension reduction method. To make the results more intuitive, we plot the data points in a scatter plot by projecting final data vectors onto 2 dimensional plan with different colors and markers, where colors stand for labels predicted by K-means and markers stand for ground truth labels. And we marked the centers of clusters with a black X.
And below is the results we got:



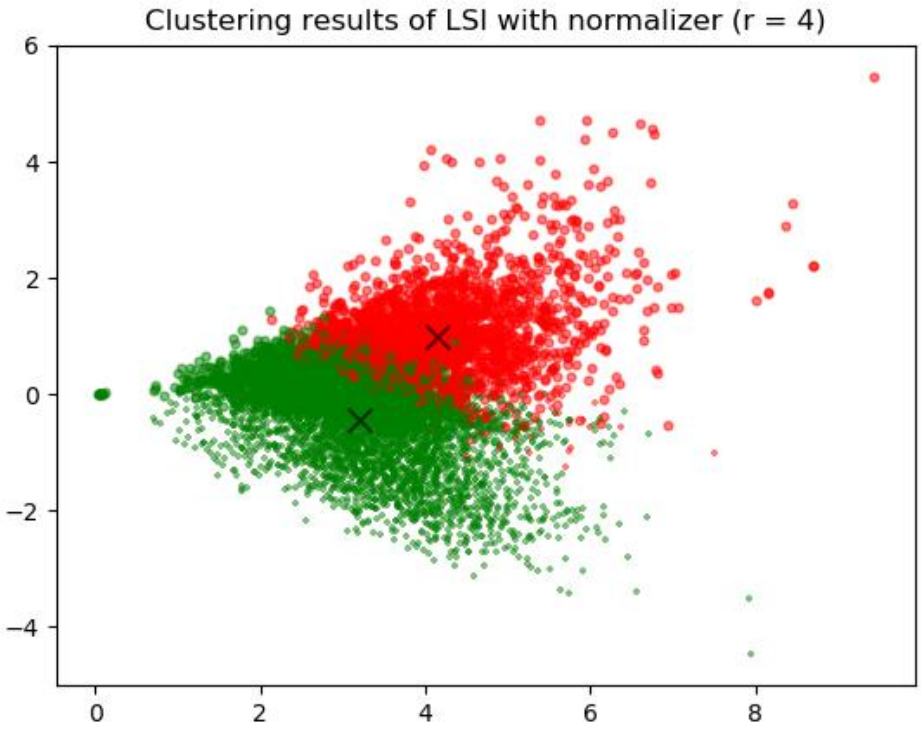


Form the scatter plots we can see that for both methods, red and green color shows reasonable clusters with minor overlapping area.

(b) Use three transformation methods to the data, measure performances and visualize.

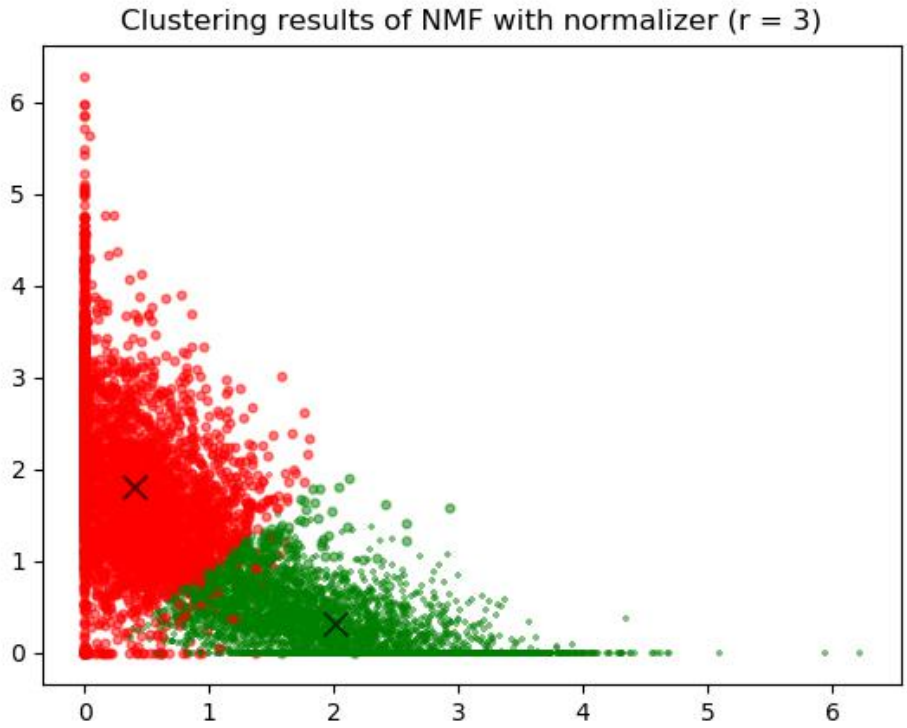
(1) Normalize Transformation

For LSI:




```
------------------------------
LSI with normalizer:
Contingency_matrix:
[[2248 1655]
 [  56 3923]]
Homogeneity_score = 0.330931
Completeness_score = 0.379619
Adjusted_rand_score = 0.320100
V_measure_score = 0.353607
Adjusted_mutual_info_score = 0.330870
------------------------------
```

For NMF:

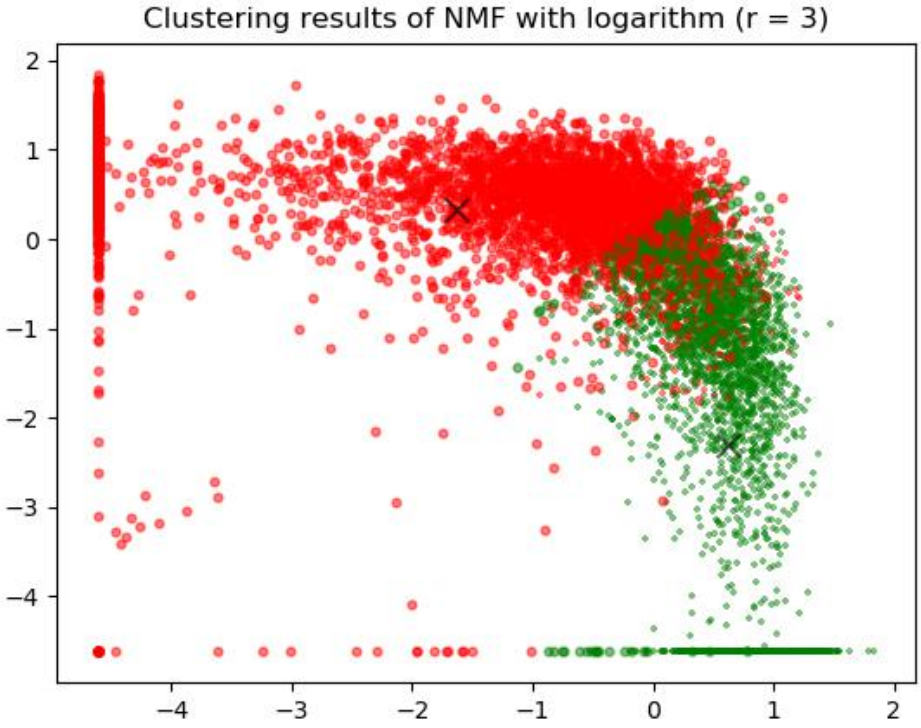



```
------------------------------
NMF with normalizer:
Contingency_matrix:
[[3608  295]
 [ 157 3822]]
Homogeneity_score = 0.686205
Completeness_score = 0.687147
Adjusted_rand_score = 0.783743
V_measure_score = 0.686676
Adjusted_mutual_info_score = 0.686176
------------------------------
```

We can see that after normalizing features, clustering result for LSI got worse while clustering result for NMF showed better performance. This means normalizing is not a good transformation for data after LSI. However, data after NMF is more stable and normalizing is good method for them.

(2) Non-linear Transformation (Logarithm)

Only data after NMF can be used to do the logarithm transformation since data after LSI contain negative values. However after NMF dimension reduction, the data still contained many zeros, which led to an input error in logarithm transformation. So we added a small constant(0.01) to every element of data and finally got a reasonable output. The results are as below.

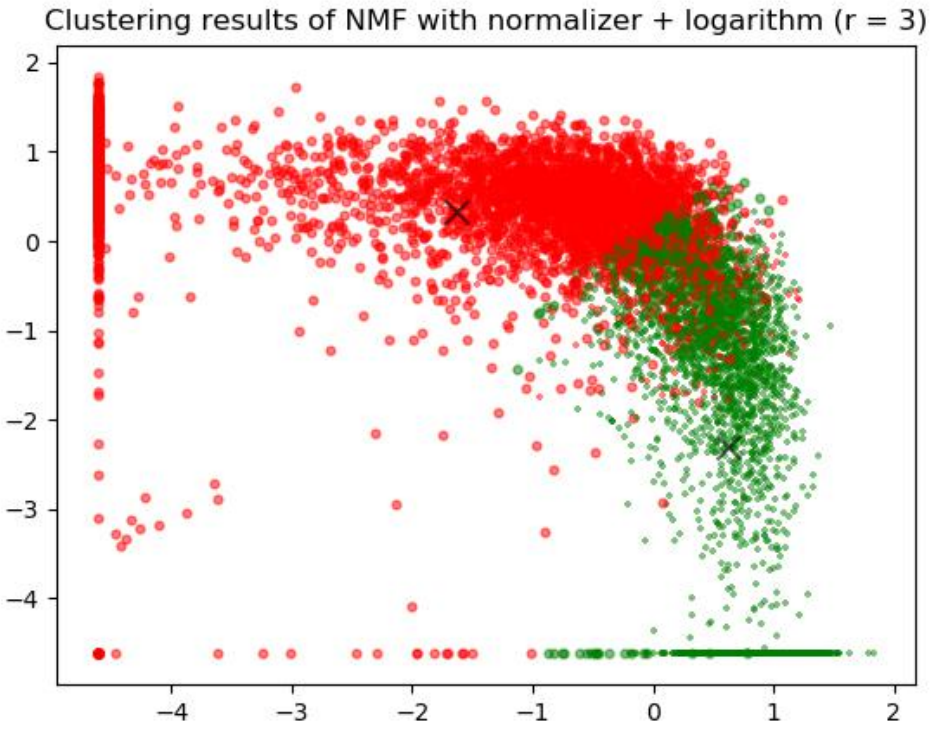



```
------------------------------
NMF with logarithm:
Contingency_matrix:
[[3609  294]
 [ 515 3464]]
Homogeneity_score = 0.526975
Completeness_score = 0.527761
Adjusted_rand_score = 0.631537
V_measure_score = 0.527368
Adjusted_mutual_info_score = 0.526932
------------------------------
```

From the scatter plot, contingency matrix and five measuring scores, we can see that though not as good as normalize transformation, logarithm transformation can still be seen as a good transformation method. The reason is that after NMF, data points were around the origin and made a concentration here. While logarithm transformation made the points more spread out than being stacked together with each other, which achieved a better separation to begin with.

(3) First Normalize Then Logarithm

In this step we combined both transformations. Since we need to do logarithm transformation here, we only used NMF and also added a small constant (0.01) to the data vectors. And the results are as below.

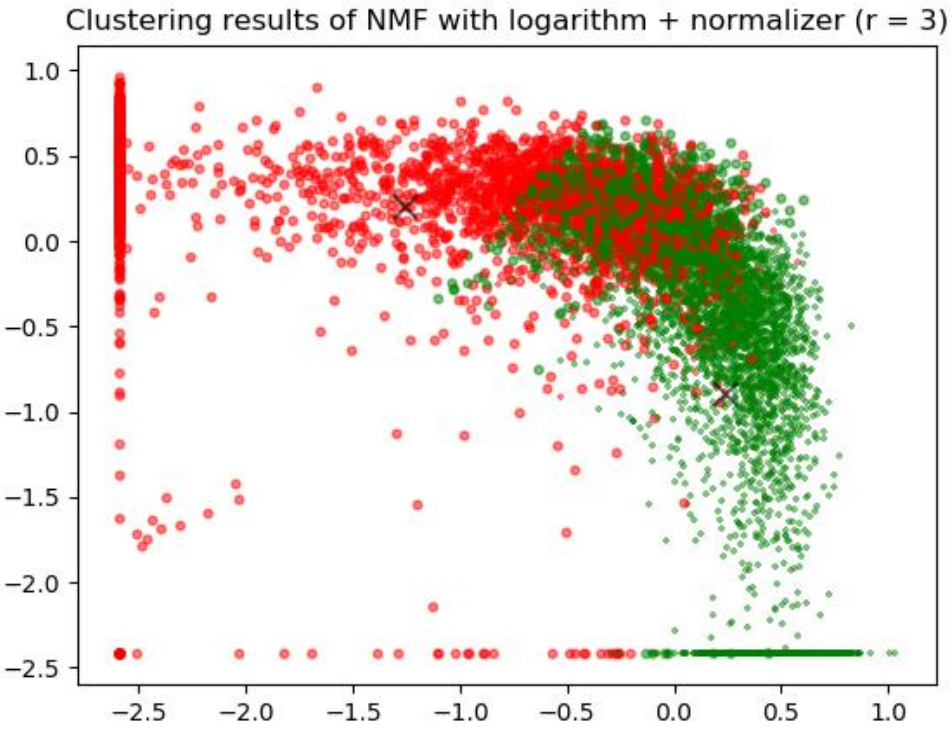



```
------------------------------
NMF with normalizer + logarithm:
Contingency_matrix:
[[3609  294]
 [ 515 3464]]
Homogeneity_score = 0.526975
Completeness_score = 0.527761
Adjusted_rand_score = 0.631537
V_measure_score = 0.527368
Adjusted_mutual_info_score = 0.526932
------------------------------
```

From the data points distribution and performance measures, we can see that the performance is almost the same as only logarithm.

(4) First Logarithm Then Normalize

In this step we also combined two methods but changed the order. The results are as below.



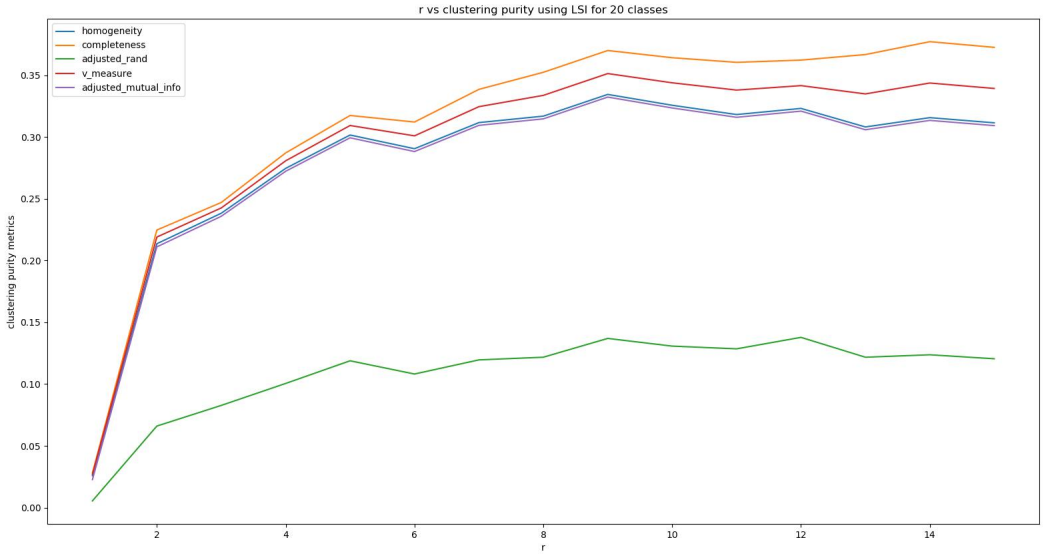

```
------------------------------
NMF with logarithm + normalizer:
Contingency_matrix:
[[2755 1148]
 [ 132 3847]]
Homogeneity_score = 0.408965
Completeness_score = 0.431473
Adjusted_rand_score = 0.455840
V_measure_score = 0.419918
Adjusted_mutual_info_score = 0.408911
------------------------------
```

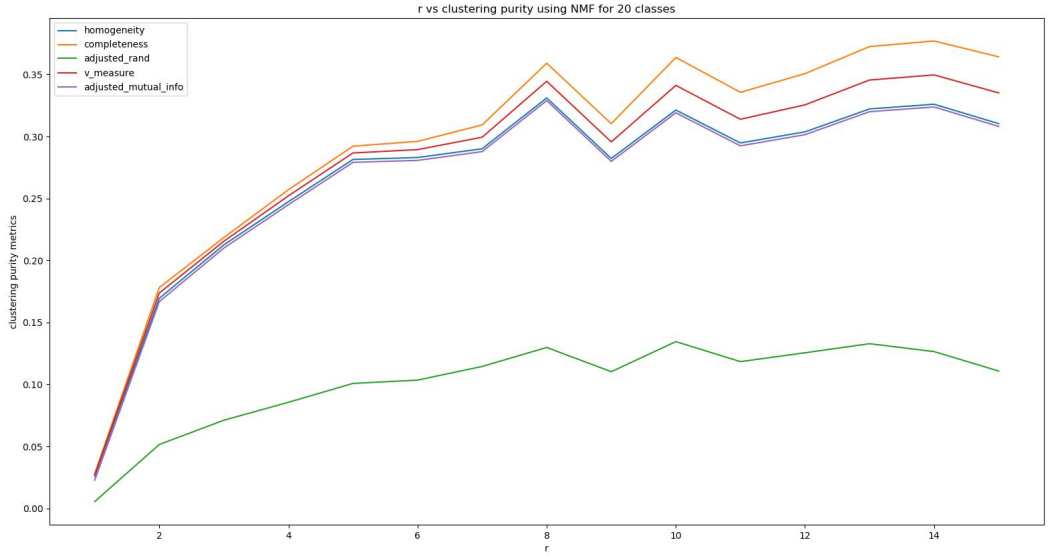
From the results we can see that when doing normalize transformation first, the clustering performance got worse. The possible reason is that after normalize data, the data points are more stacked on top of each other around the origin, thus making it harder for the logarithm transformation to spread out the data points.

5. Expand Dataset into 20 Categories

In this part we examined how purely we can retrieve all 20 original sub-class labels with clustering. Also we preprocessed the data by excluding the punctuation, stop-words and got the TF-IDF representation of data. But differently, here we need 20 clusters by setting n_clusters = 20 in K-means function.

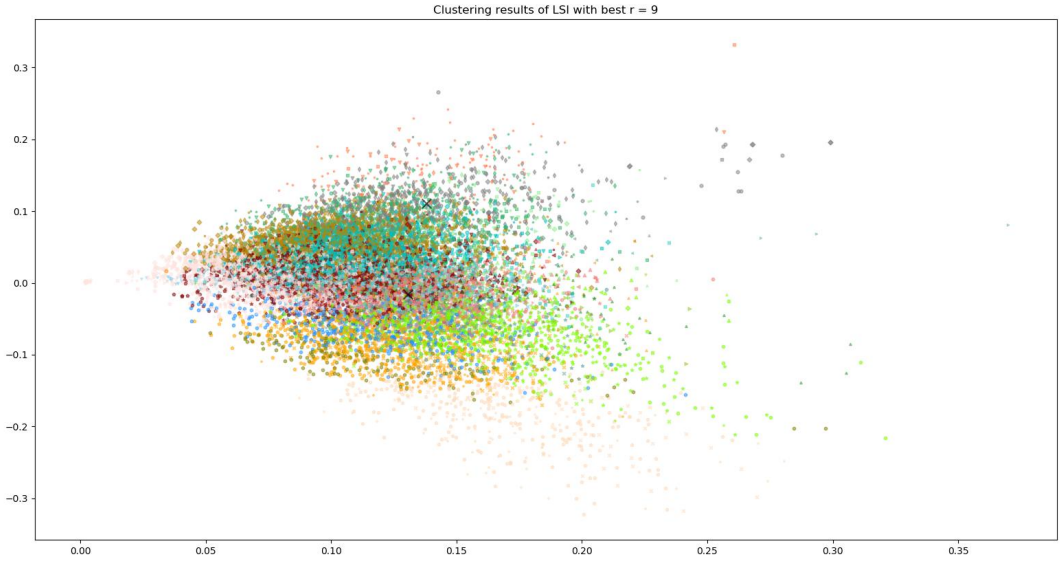
To find the best dimension, we also used different r and plotted the 5 measure scores v.s. r for both LSI and NMF. The results are as below.

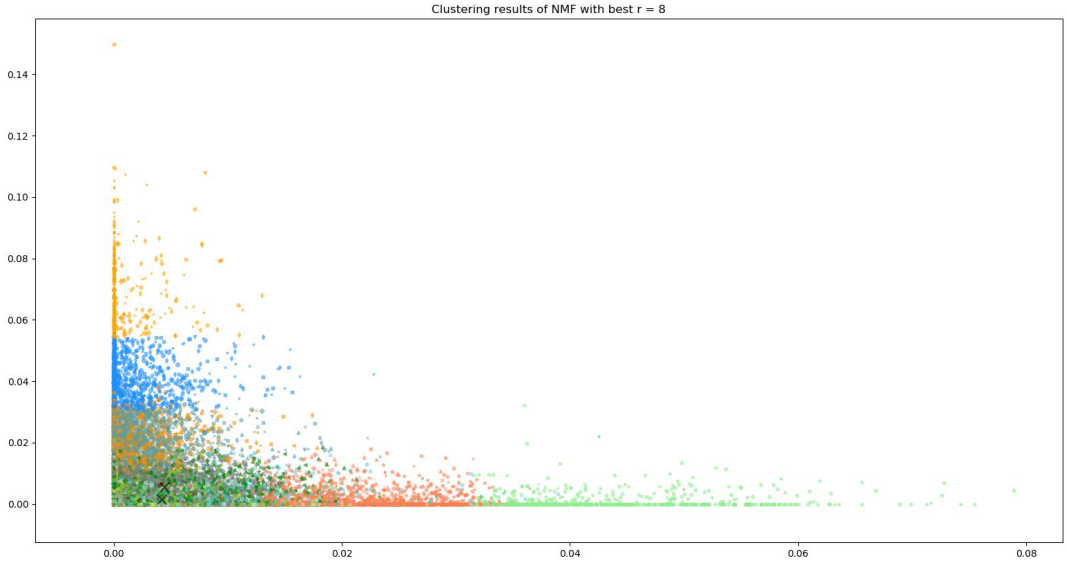




From the above graphs we can see that for LSI, the best r is 9 and for NMF, the best r is 8.

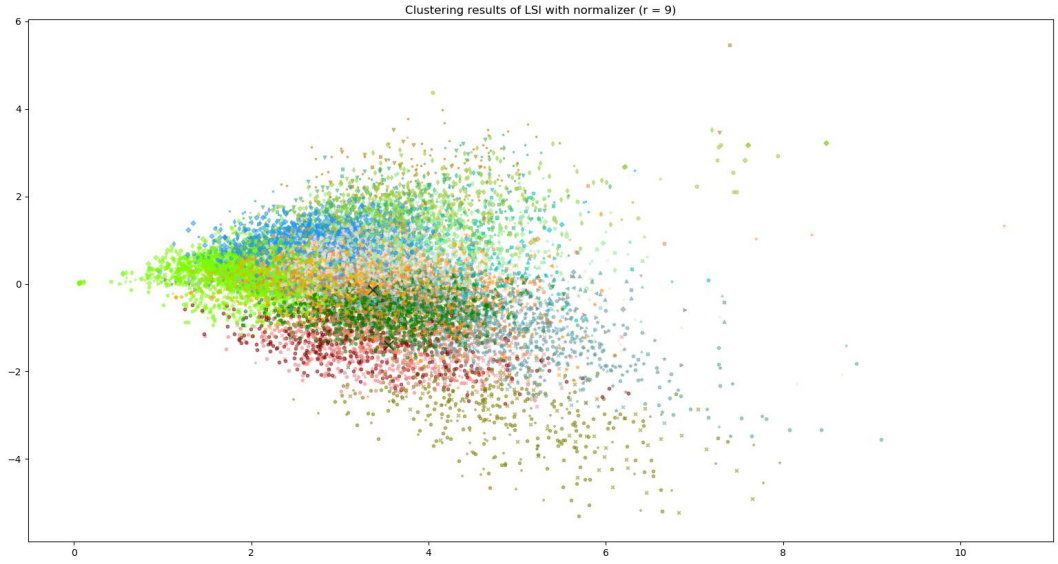
Next, following the same steps, we also visualize the clustering results of LSI and NMF, with their best r. Here we used 20 different colors and markers to represent their predicted label and ground truth label. The scatter plots are as below.





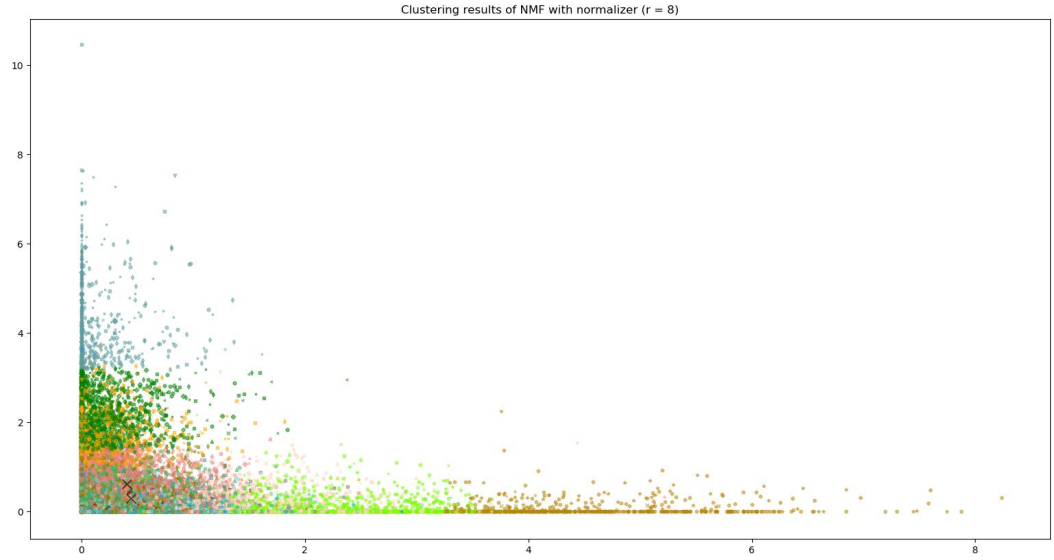
Then we tried normalize, logarithm transformation and the combination of them to transform the data.

(1) Normalize (for both LSI and NMF)



LSI with normalizer:
Contingency_matrix:
[[0 272 0 97 1 1 121 35 0 75 2 47 1 79 0 0 27
25
1 15]
[0 1 0 135 8 0 3 80 0 0 65 197 112 6 5 1 9
4
13 334]
[0 0 0 64 1 0 3 55 2 0 312 83 99 3 21 0 6
1
8 327]
[3 0 0 68 3 0 12 123 84 0 36 126 199 1 223 4 25
0
11 64]
[1 0 0 171 7 0 2 78 15 0 4 183 269 4 148 1 41
2
6 31]
[0 0 0 86 15 0 1 95 0 0 99 178 31 0 1 9 4
0
17 452]
[12 1 0 296 1 0 10 108 15 0 10 263 86 11 93 1 37
0
10 21]
[0 0 0 216 4 0 34 294 0 0 0 163 26 146 0 0 38
39
21 9]

```
 [  3   8   0 169   6   0  63 422   0   0   1 157   9  76   2   0  13
17
   22  28]
 [398   0   0 266   1   0   8 129   0   0   0 134   1  35   0   0   6
9
    4   3]
 [702   1   0 113   0   0   2  38   0   0   0 124   1   2   0   0   9
3
    4   0]
 [  0   0   0  38  10 247  27  53   0   0   6  37  33  35   0 433   4
29
   13  26]
 [  1   1   0 190  17   0   2 176   1   0   4 279 177  20  14   7   6
1
   39  49]
 [  1  12   0 250   3   0   7 154   0   2   2 329   8 156   0   0   5
16
   32  13]
 [  0   1   0  73 287   0   4  57   0   0   0 120   1  22   1   0   1
9
  402   9]
 [  0 485   0  51   3   0   3  45   0 220   1 133   4  31   0   0   1
10
    2   8]
 [  0   1   0  65   3   2 128 110   0   0   1  43   1 321   0   5  25
199
    4   2]
 [  0   4 375  62   0   0   8  19   0   2   0  88   1 303   0   0  30
44
    0   4]
 [  1   5   0  70   5   1  71  78   0   1   0  58   0 267   0   1  45
143
   25   4]
 [  0 163   0  73   0   0  60  51   0  88   0  61   3  66   0   1  20
35
    5   2]]
Homogeneity_score = 0.316889
Completeness_score = 0.350215
Adjusted_rand_score = 0.127680
V_measure_score = 0.332720
Adjusted_mutual_info_score = 0.314676
------------------------------
```

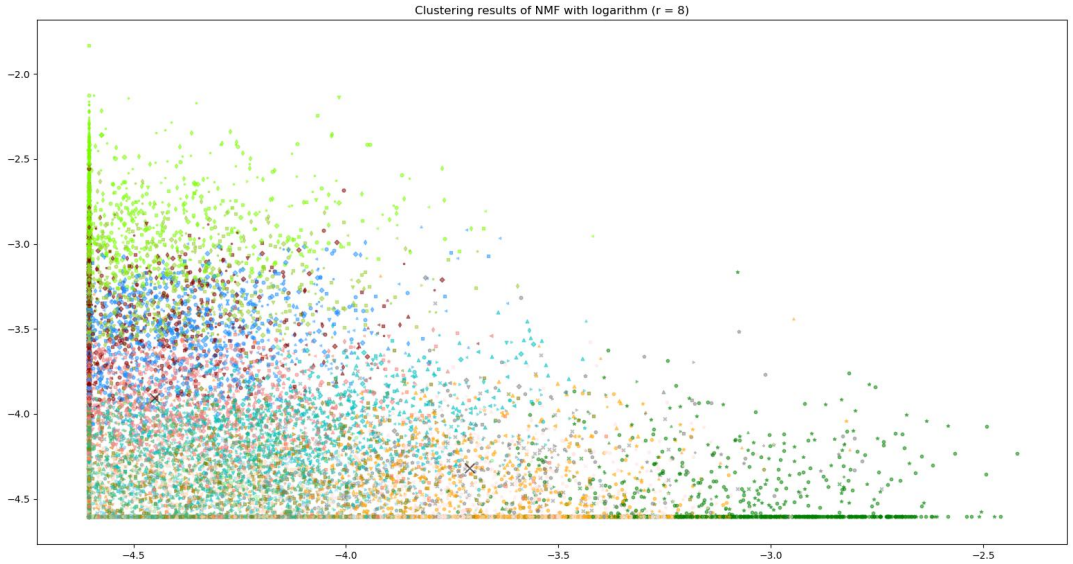


NMF with normalizer:
Contingency_matrix:

```
[[  0  38   0 255   1  46   1   2   6  56  58  50   0   1 102   1  29
0
   95  58]
 [ 12 204   0   1   8   3   0 159   0  97  33   2   1 267 125   1   0
42
    0  18]
 [  9  86   0   0   2   2   0 211   0  78  18   0   2 286  45   2   0
231
    0  13]
 [ 13  70   1   0   3   1   0 169   0 114  34   0   2 289  46   5   0
204
    0  31]
 [  9 124   1   0   7   1   0 250   0  81  82   1   1 234 108   2   0
52
    0  10]
 [ 15 215   0   0  17   0   0 108   0 156  30   0  13 304  71   3   0
40
    0  16]
 [ 10 158   3   2   1   1   0 149   0 124 123   2   1  65 251  15   0
49
    0  21]
 [ 14 158   0   0   4  94   0   4   0 258  78  34   0   6 261   6   0
0
    0  73]
```

```
 [ 17 143   0   9   5  65   0   0   0 325  58   9   0   3 175  20   0
0
   0 167]
 [  4  55 131   0   1   6   0   0   0  96  57   1   0   0 198 411   0
0
   0  34]
 [  3  27 395   1   0   0   0   1   0  29  26   1   0   0  67 442   0
0
   0   7]
 [ 18  43   0   0  11  19 230  11   5  61  18  27 432  16  51   2   0
3
   0  44]
 [ 42 281   0   1  13   7   0  75   0 171  52   0   6 112 180  10   0
7
   0  27]
 [ 30 286   0  11   3  62   0   2   1 160  94  65   0   2 243   3   0
2
   2  24]
 [389 105   0   1 277   6   0   3   0  61  19  18   0   2  88   2   0
0
   0  16]
 [  2  60   0 374   3   6   0   0   2  48   9  13   0   4  71   0 113
0
  284   8]
 [  3  29   0   1   3 206   2   0 143  44  45 270   3   0  65   0   0
0
   0  96]
 [  0  40   0   5   0  79   0   0 332  14  55 332   0   0  77   0   2
0
   0   4]
 [ 26  37   0   7   5 204   1   1  62  48  53 188   1   0  87   2   0
0
   3  50]
 [  5  47   0 154   0  50   0   1  13  55  47  36   1   0  61   1  51
0
  82  24]]
Homogeneity_score = 0.313329
Completeness_score = 0.335300
Adjusted_rand_score = 0.118955
V_measure_score = 0.323943
Adjusted_mutual_info_score = 0.311109
----------------------------
```

(2) Logarithm (only for NMF)



NMF with logarithm:
Contingency_matrix:
[[56 18 1 3 174 1 2 234 3 2 0 0 44 1 1 78 30
43
2 106]
[0 42 93 28 2 93 0 2 122 16 2 6 15 0 159 1 109
0
148 135]
[0 43 164 36 1 167 0 0 112 2 1 2 12 0 245 0 60
0
71 69]
[0 42 109 25 2 248 0 0 133 7 4 6 8 0 226 1 44
0
68 59]
[0 33 120 63 5 67 4 0 110 9 1 6 7 4 211 0 93
0
105 125]
[0 93 59 30 2 143 0 2 164 26 8 5 11 2 122 0 56
0
198 67]
[5 9 167 88 5 53 8 3 137 12 4 27 1 18 247 1 50
0
82 58]
[102 64 21 285 5 16 11 2 80 24 6 20 0 7 11 78 13
22
29 194]

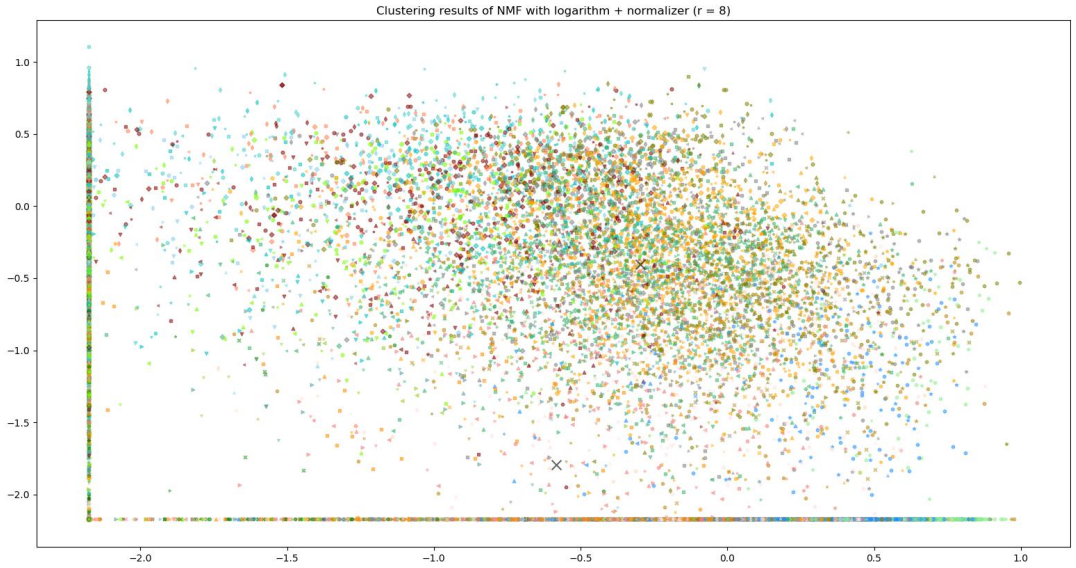
```
 [135  82  12 181   3  26   3  11  94  22   4  93   0   7   6  46   0
52
  12 207]
 [176  32  64  40  20   2   2   0   4   6   1 191  12 323   6   0  34
4
   3  74]
 [ 41  22  46  17  10   1   1   0   1  10   0 225  44 546   0   1  12
2
   1  19]
 [ 29  80   7   8  15  17  24   0   4   9 500   2  28   1  10  14  81
20
  102  40]
 [  9 113  83 119   1  38   3   4  42  32   5  15   4   1  51   4 152
4
  133 171]
 [103  78  17  54  59   4  24  14  40  25   4  14   5  13  28  68 130
16
   44 250]
 [ 60  78  10  31   8   5   5   1   9 477   5   4  19   1  15   9 114
5
   37  94]
 [ 24  29   6   0 222   3   0 367   4   8   2   6 128   5   9  13  48
32
   5  86]
 [ 85  35   0  50  49   7 148   2   7   4   4   6  13   1   1 244  44
170
   7  33]
 [ 36   4   2  24 110   0 456   1   8   0   3   2  32   5   3 133  17
89
   3  12]
 [ 91  32   0  34  58   1  76   8   3  18   4   5  23   3   4 179  57
110
   5  64]
 [ 50  22   3   4  75   3  12 178   8   5   1   2  45   9   3  59   9
47
   2  91]]
Homogeneity_score = 0.277897
Completeness_score = 0.282306
Adjusted_rand_score = 0.122093
V_measure_score = 0.280084
Adjusted_mutual_info_score = 0.275566
----------------------------
```

(3) First Normalize Then Logarithm (only for NMF)

NMF with normalizer + logarithm:
Contingency_matrix:
[[56 18   1   3 174   1   2 234   3   2   0   0 44   1   1 78 30
43
   2 106]
 [0 42 93 28   2 93   0   2 122 16   2   6 15   0 159   1 109
0
  148 135]
 [0 43 164 36   1 167   0   0 112   2   1   2 12   0 245   0 60
0
  71 69]
 [0 42 109 25   2 248   0   0 133   7   4   6   8   0 226   1 44
0
  68 59]
 [0 33 120 63   5 67   4   0 110   9   1   6   7   4 211   0 93
0
  105 125]
 [0 93 59 30   2 143   0   2 164 26   8   5 11   2 122   0 56
0
  198 67]
 [5   9 167 88   5 53   8   3 137 12   4 27   1 18 247   1 50
0
  82 58]
 [102 64 21 285   5 16 11   2 80 24   6 20   0   7 11 78 13
22
  29 194]

```
 [135  82  12 181   3  26   3  11  94  22   4  93   0   7   6  46   0
52
   12 207]
 [176  32  64  40  20   2   2   0   4   6   1 191  12 323   6   0  34
4
    3  74]
 [ 41  22  46  17  10   1   1   0   1  10   0 225  44 546   0   1  12
2
    1  19]
 [ 29  80   7   8  15  17  24   0   4   9 500   2  28   1  10  14  81
20
  102  40]
 [  9 113  83 119   1  38   3   4  42  32   5  15   4   1  51   4 152
4
  133 171]
 [103  78  17  54  59   4  24  14  40  25   4  14   5  13  28  68 130
16
   44 250]
 [ 60  78  10  31   8   5   5   1   9 477   5   4  19   1  15   9 114
5
   37  94]
 [ 24  29   6   0 222   3   0 367   4   8   2   6 128   5   9  13  48
32
    5  86]
 [ 85  35   0  50  49   7 148   2   7   4   4   6  13   1   1 244  44
170
    7  33]
 [ 36   4   2  24 110   0 456   1   8   0   3   2  32   5   3 133  17
89
    3  12]
 [ 91  32   0  34  58   1  76   8   3  18   4   5  23   3   4 179  57
110
    5  64]
 [ 50  22   3   4  75   3  12 178   8   5   1   2  45   9   3  59   9
47
    2  91]]
Homogeneity_score = 0.277897
Completeness_score = 0.282306
Adjusted_rand_score = 0.122093
V_measure_score = 0.280084
Adjusted_mutual_info_score = 0.275566
------------------------------
```

(4) First Logarithm Then Normalize (only for NMF)



NMF with logarithm + normalizer:
Contingency_matrix:

```
[[ 27   5   3 220   1   2   2  63 178  55   0   2   1 210  25   1   0
2
   0   2]
 [100   2 142   1 121   2  20 113   2  49   4 113   0   2  56  81  83
0
  82   0]
 [ 42   1  62   0 211   1   7  40   5  48   3 112   0   0  50 175 159
0
  69   0]
 [ 33   2  76   1 170   5  16  44   4  41   3 127   0   0  32 245  92
0
  91   0]
 [ 55   6  87   2 178   0  12  85   7  33   1 114   0   0  91  64 121
5
  98   4]
 [ 45   1 188   0  87  10  30  68   5  73   3 161   0   2  35 135  45
2
  98   0]
 [ 46  13 109   0 208   4   9  50   3   5  13 127   0   2  64  47 163
19
  82  11]
 [ 13 142  52  36   6   8  26 111   6  61  10  99   5   2 289  26  30
6
  10  52]
```

```
 [  0 201  31  30   5  17  26 113   3  71  40 111  15  11 222  33  31
8
    4  24]
 [ 29 257   2   1   6   0   7  64  23  29 116   5   0   0  51   5  73
318
    6   2]
 [  4  80   2   0   0   0   6  19  16  23 329   2   0   0  19   0  64
434
    0   1]
 [ 50   3  60   1   0 565   6  88  11  53   1   3   1   0   2   3   3
1
  134   6]
 [129   8 150   1  23  14  33 202   1  85   7  46   1   4  76  38  66
1
   97   2]
 [135  67  46  66  18   7  27 167  62  74   4  39   1  14 158   5  20
15
   18  47]
 [141   6  59   3   5   2 495  46   5  76   3   8   2   1 116   2   7
1
    7   2]
 [ 43   0   4 120   7   2   9  39 280 103   5   3   0 349  20   1   4
3
    5   0]
 [ 57  73  10  91   0  55  15  46  58  44   6  10 118   2  33   8   1
1
   11 271]
 [ 18  44   3  32   2   1   0   9 116  14   3  12 262   1  20   0   1
8
    4 390]
 [ 67  60   9 116   2  33  26  76  67  64   6   5  52   8  26   2   1
7
    3 145]
 [ 10   5   4 150   2   3   6  62  78  62   0   8   5 173  22   2   2
9
    1  24]]
Homogeneity_score = 0.285768
Completeness_score = 0.288340
Adjusted_rand_score = 0.125482
V_measure_score = 0.287048
Adjusted_mutual_info_score = 0.283463
------------------------------
```

From the results above, we can see that compared to 2 class clustering, the scatter plots are much more vague and the five measure scores are also much lower. This is reasonable since it will be more complicated to cluster more classes. And similar to part 4(b), normalize transformation showed better performance than logarithm transformation.