EE219 Project 2 Report

Clustering

Winter 2018

Zhengxu Xia (104250792), Tairan Zhu (605031908)

Introduction

In this project, we worked with 20 newsgroup dataset and performed clustering algorithm on documents which are evenly distributed across 20 groups. We treated each document as unlabeled and unrecognized at first, and then identified each of them as an element of one clusters by repeatedly performing the clustering methods. Our goals include seeking proper representation of data, evaluating the performance of clustering, and try different preprocess methods which may increase the performance of the clustering.

Problem 1

In Problem 1, we fetched eight categories of data from ‘20 Newsgroup’ dataset and manually separate them into two major classes as our ground truth. The ground truth table is shown here in Table 1.

Table 1. Two well-separated classes

Class Name	Computer Technology	Recreational Activity
Sub-class	comp.graphics	rec.autos
	comp.os.ms-windows.misc	rec.motorcycles
	comp.sys.ibm.pc.hardware	rec.sport.baseball
	comp.sys.mac.hardware	rec.sport.hockey

We preprocessed the data by performing stop-word removal and word frequency vectorization with min df = 3. By computing term frequency–inverse document frequency, we found the **dimension of TF-IDF Matrix** is **(7882, 27768)**

Problem 2

Our group applied K-means clustering with k = 2 using the TF-IDF data we derived previously. We observed contingency matrix, homogeneity score, the completeness score, the V-measure, the adjusted Rand score and the adjusted mutual info score to compare the purity for a given partition of the data points with respect to the ground truth among different clustering methods.

contingency matrix (before dimension reduction)

	True Computer Technology	True Recreational Activity
Labeled Computer Technology	4	3899
Labeled Recreational Activity	1717	2262

Homogeneity: 0.253413
Completeness: 0.334677
V-measure: 0.288430
Adjusted Rand Score: 0.180546
Adjusted Mutual Info Score: 0.253345

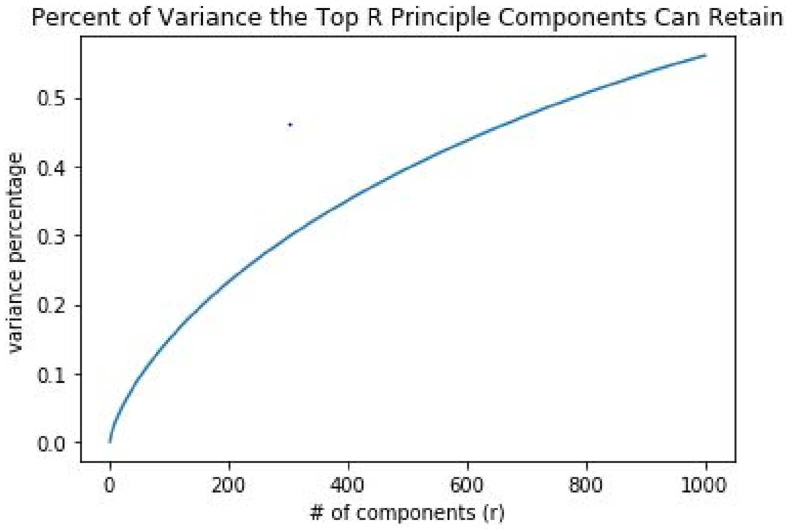
Problem 3

Since high dimensional sparse TF-IDF vectors do not result in a fairly good representation of clustering, in this part we performed dimension reduction in order to find a better representation tailored to the way that K-means clustering algorithm works, by preprocessing our data before clustering.

(a) Dimensionality Reduction

(i) We used ***Latent Semantic Indexing (LSI)*** and ***Non-negative Matrix Factorization (NMF)*** for dimensionality reduction.

In order to find an effective dimension of data we observed the top singular values of the TF-IDF matrix and explore what ratio of the variance of the original data is retained after the dimensionality reduction. We plotted the ***percent of variance the top r principal components can retain v.s. r, for r = 1 to 1000.***



We find see that as the number of components increases, the percentage of variance the top r principle components can retain also increases. It is monotonically increasing.

(ii) We performed dimensionality reduction using LSI and NMF. Specifically, we tried r = 1, 2, 3, 5, 10, 20, 50, 100, 300, and plot the 5 measure scores v.s. r for both SVD and NMF; Additionally, we reported the contingency matrices for each r.

Find best r using SVD

contingency matrix when ***r = 1***

```
 [2195 1708]
 [2318 1661]
```

contingency matrix when ***r = 2***

```
 [3691  212]
 [ 436 3543]
```

contingency matrix when ***r = 3***

```
 [3874   29]
 [1492 2487]
```

contingency matrix when ***r = 5***

```
 [   5 3898]
 [1543 2436]
```

contingency matrix when ***r = 10***

```
 [   3 3900]
 [1607 2372]
```

contingency matrix when ***r = 20***

```
 [3900    3]
 [2364 1615]
```

contingency matrix when ***r = 50***

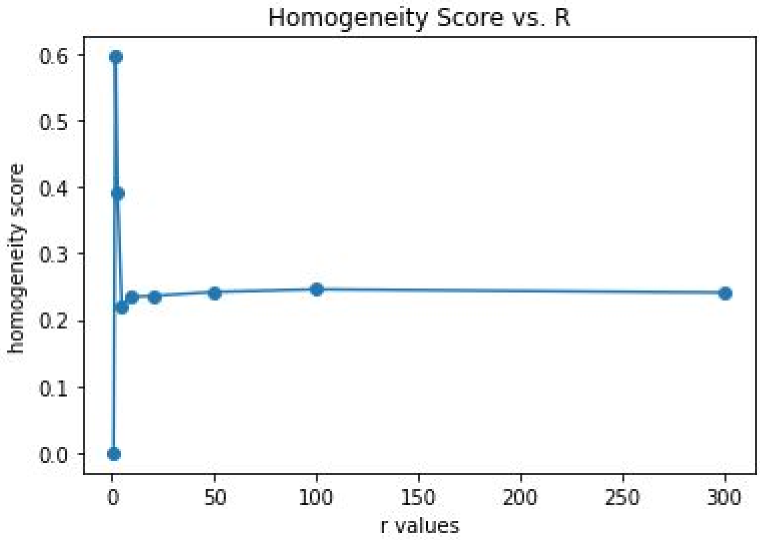
```
 [3899    4]
 [2325 1654]
```

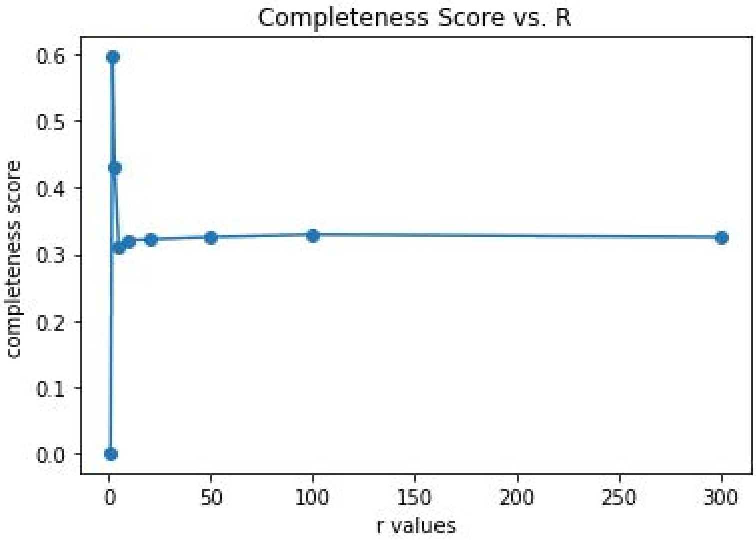
contingency matrix when ***r = 100***

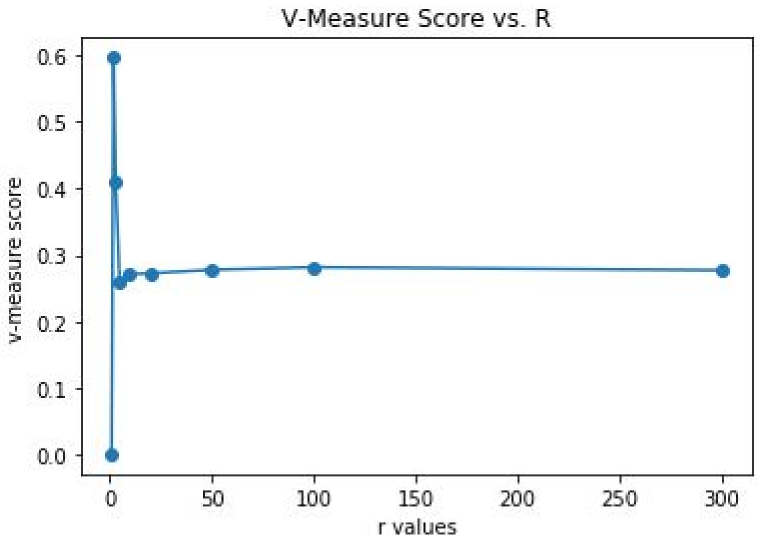
```
[   3 3900]
[1670 2309]
```

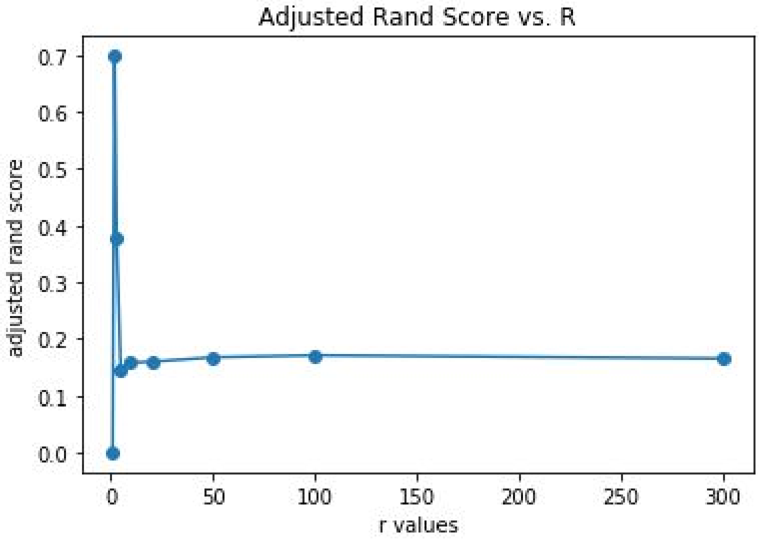
contingency matrix when ***r = 300***

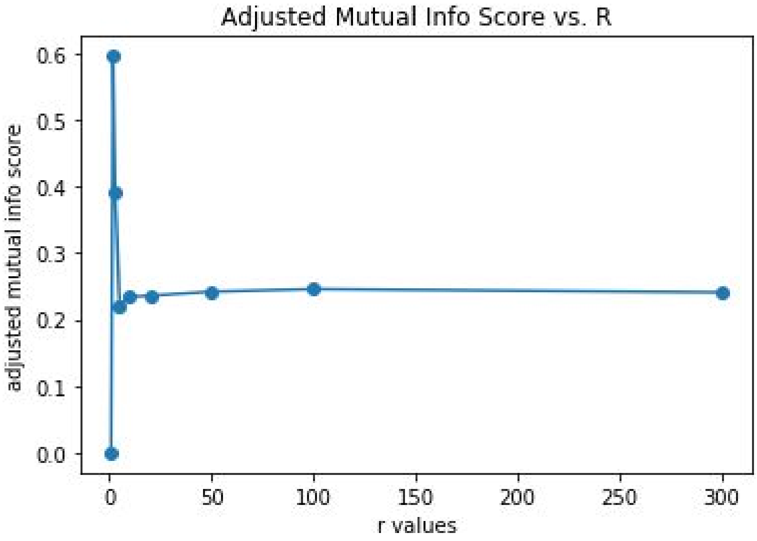
```
[   3 3900]
[1643 2336]
```






V-Measure Score vs. R
v-measure score
r values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
50
100
150
200
250
300


Adjusted Rand Score vs. R
adjusted rand score
r values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
50
100
150
200
250
300


Adjusted Mutual Info Score vs. R
adjusted mutual info score
r values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
50
100
150
200
250
300

Find best r using NMF

contingency matrix when ***r = 1***

```
[[2195 1708]
 [2318 1661]]
```

contingency matrix when ***r = 2***

```
[[3594  309]
 [ 158 3821]]
```

contingency matrix when ***r = 3***

```
[[3899    4]
 [2396 1583]]
```

contingency matrix when ***r = 5***

```
[[3898    5]
 [2677 1302]]
```

contingency matrix when ***r = 10***

```
[[3899    4]
 [2627 1352]]
```

contingency matrix when ***r = 20***

```
[[3899    4]
 [2511 1468]]
```

contingency matrix when ***r = 50***

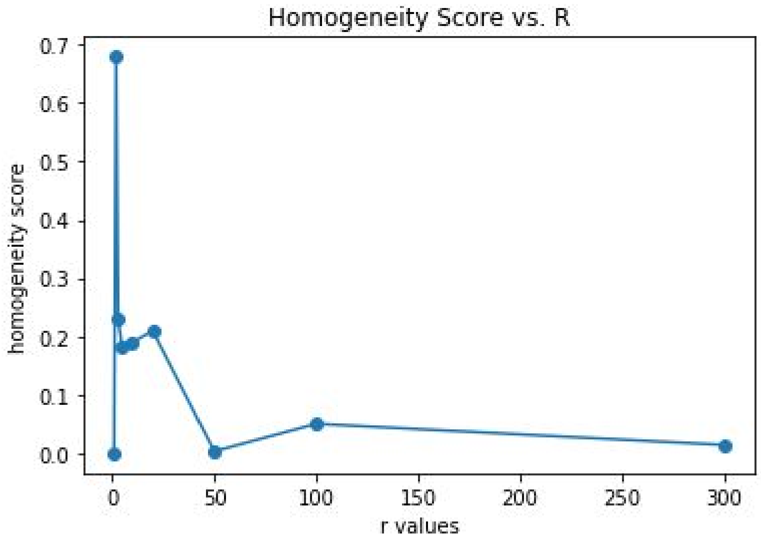
```
[[3895    8]
 [3922   57]]
```

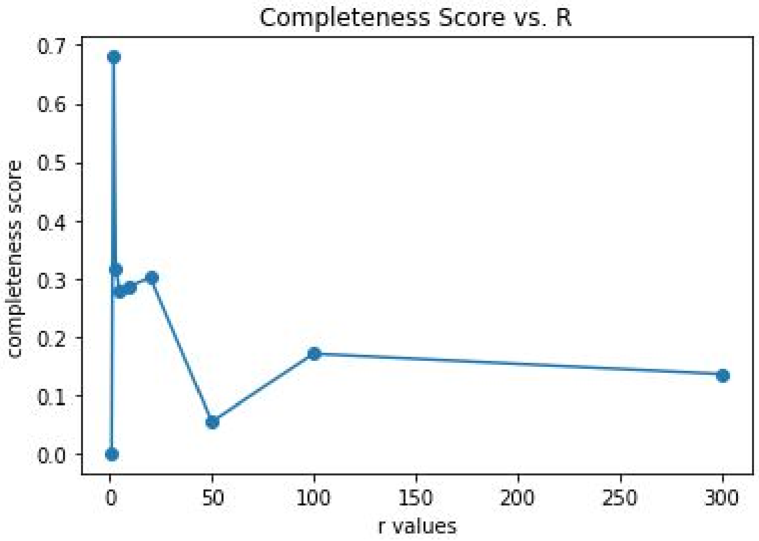
contingency matrix when ***r = 100***

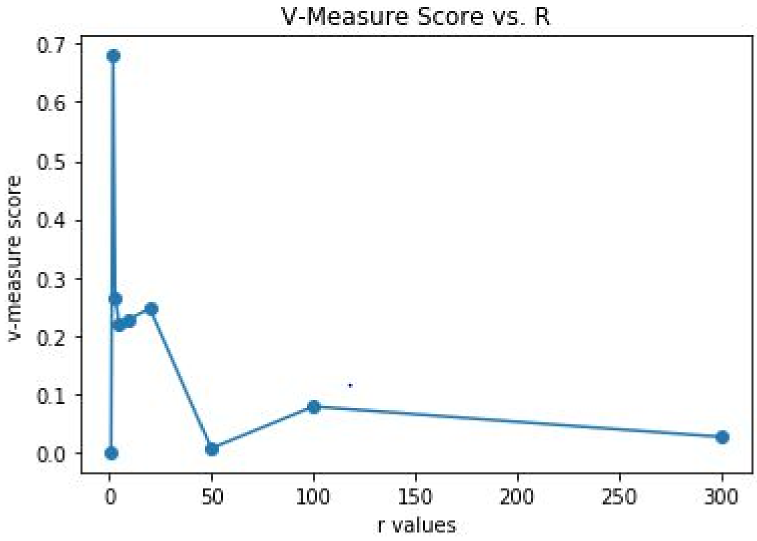
```
[[   3 3900]
 [ 413 3566]]
```

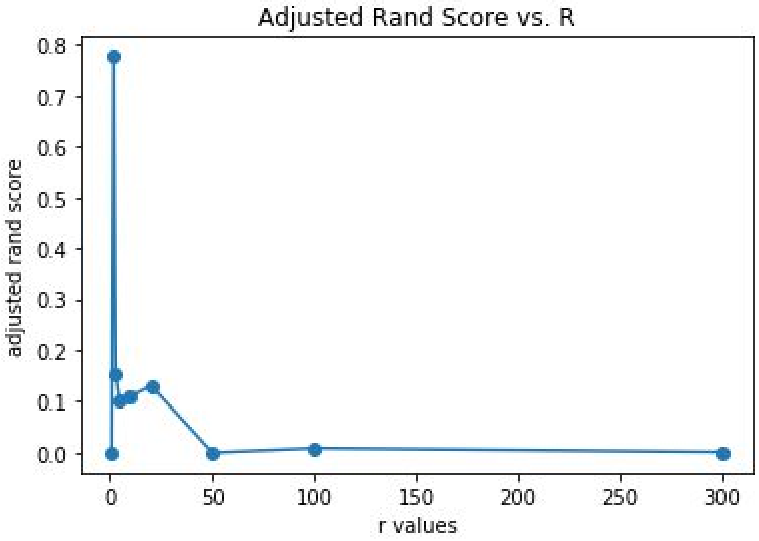
contingency matrix when ***r = 300***

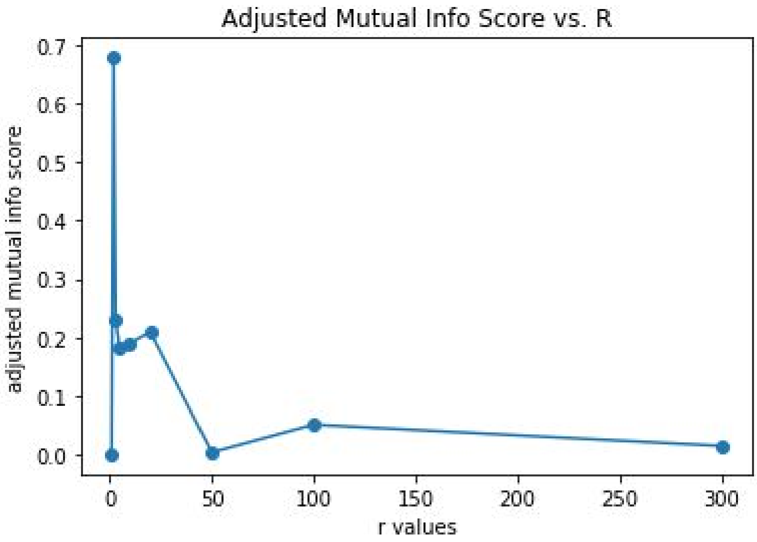
```
[[ 113 3790]
 [   0 3979]]
```




Completeness Score vs. R
completeness score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
50
100
150
200
250
300
r values


V-Measure Score vs. R
v-measure score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
50
100
150
200
250
300
r values


Adjusted Rand Score vs. R
adjusted rand score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250
300
r values



For both SVD and NMF, we observed that homogeneity score, the completeness score, the V-measure, the adjusted Rand score and the adjusted mutual info score achieve peak value at r = 2; in other words, five of the score curves achieve the maximum value at r = 2 instead of monotonically increasing. The reason for that is ***K-means clustering would likely to have the best performance at a low dimensionality***, which according to the plot, at 2-dimension. If the dimensionality keeps increasing, K-means algorithm would become inefficient and unorganized which resulted in a bad performance.

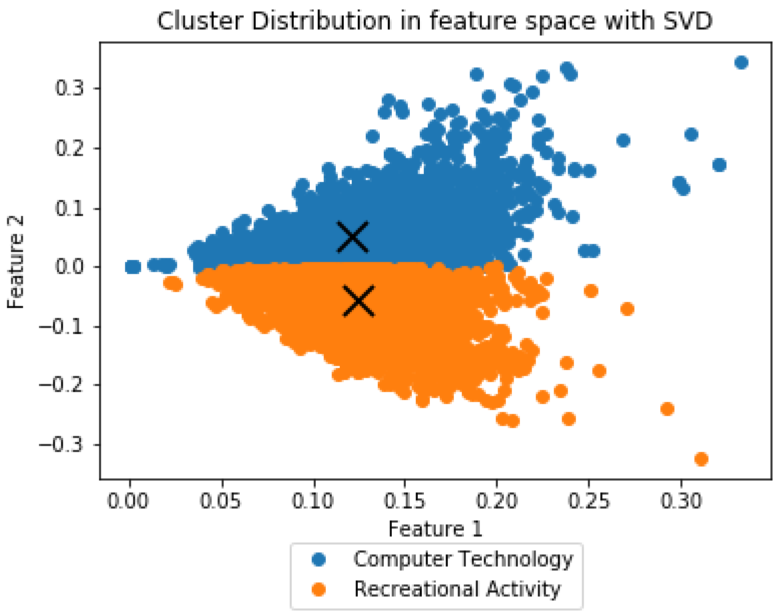
Another thing we observed is that NMF reduction has a worse performance in clustering in a high dimensionality (i.e r > 2) compared to SVD method. Furthermore, the performance scores using LSI almost keep unchanged beyond 2-dimension, whereas performance score using NMF achieve minimum value at r = 50.

Problem 4

(a) Visualization of best clustering results

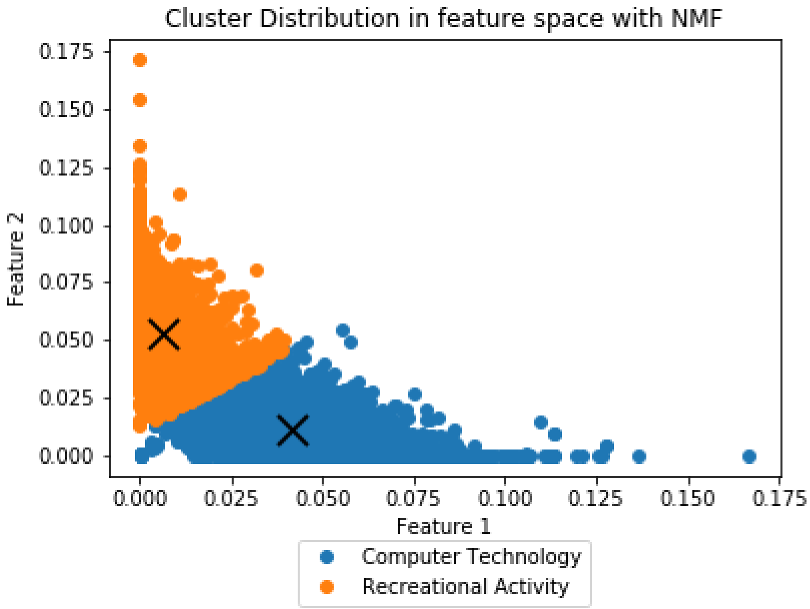
SVD

From problem 3, we found the best r-value of SVD is 2. By performing Kmean algorithm on data after SVD with only 2 components, we obtained the visualization of clusters and contingency matrix.



	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3691	212
Labeled Recreational Activity	436	3543

NMF



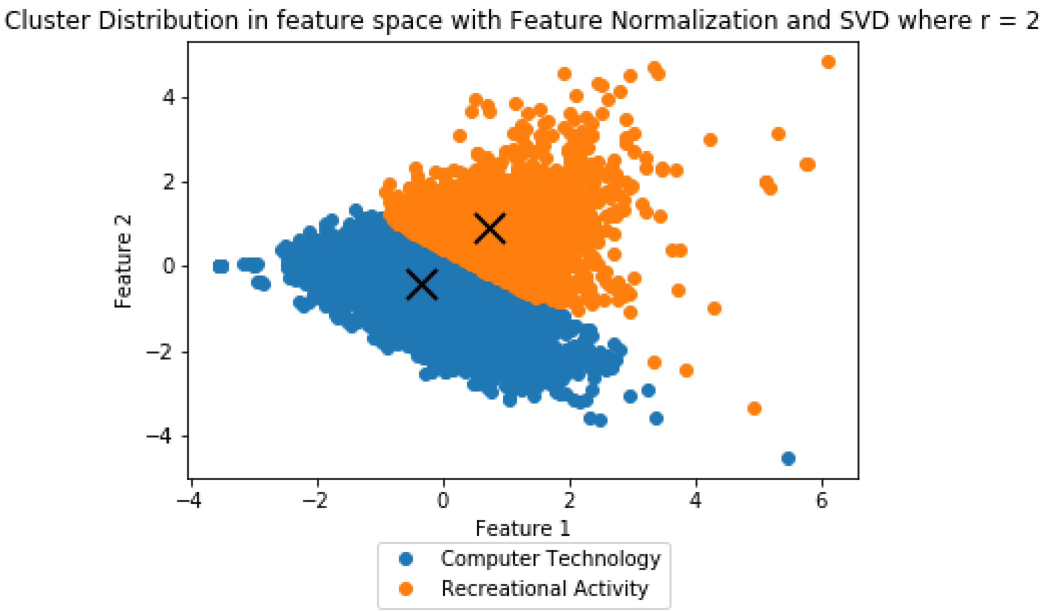
	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3594	309
Labeled Recreational Activity	158	3821

Analysis:
Both figures from SVD and NMF make sense. SVD can generate negative values thus two features contains some negative values while NMF features are all non-negative. From the perspective of contingency matrix, NMF does a little better job since there are fewer data points are classified wrongly. This might be caused by the fact that term tfidf value is a physical value which cannot be negative. NMF's results may be more related to its physical meaning.

(b) Visualization of clustering results after other statistical techniques

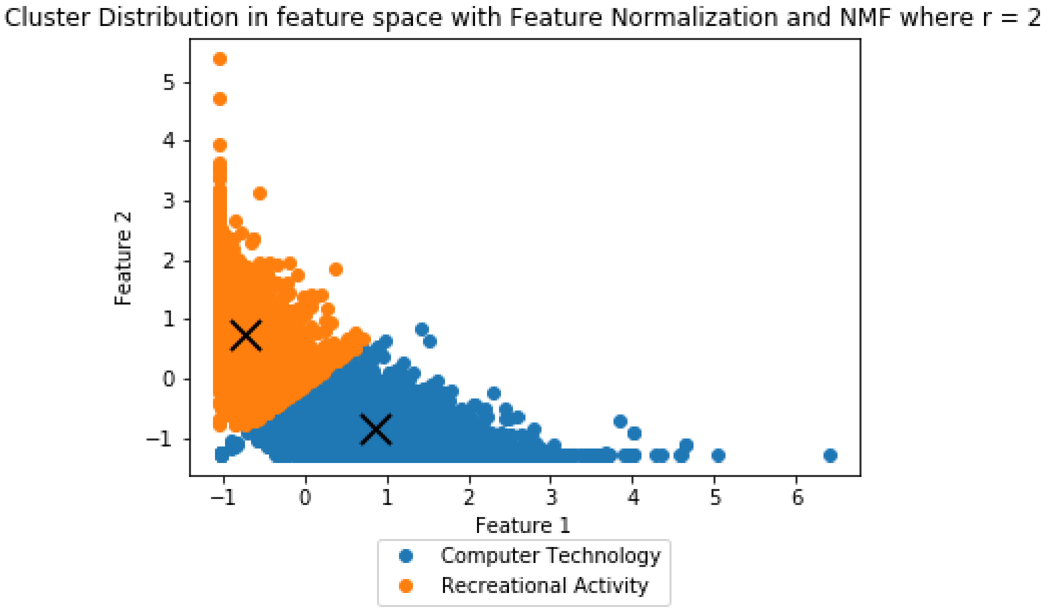
1. Feature Normalization

The feature normalization step is performed on dimensionally reduced data by calling StandardScaler() in sklearn package. This step justs removed the mean of each feature column and scale the variance of feature column to unit vector. Then the normalized data were fed into k-mean classifier and the following results are obtained.



	True Computer Technology	True Recreational Activity

Labeled Computer Technology	1705	2198
Labeled Recreational Activity	3733	246

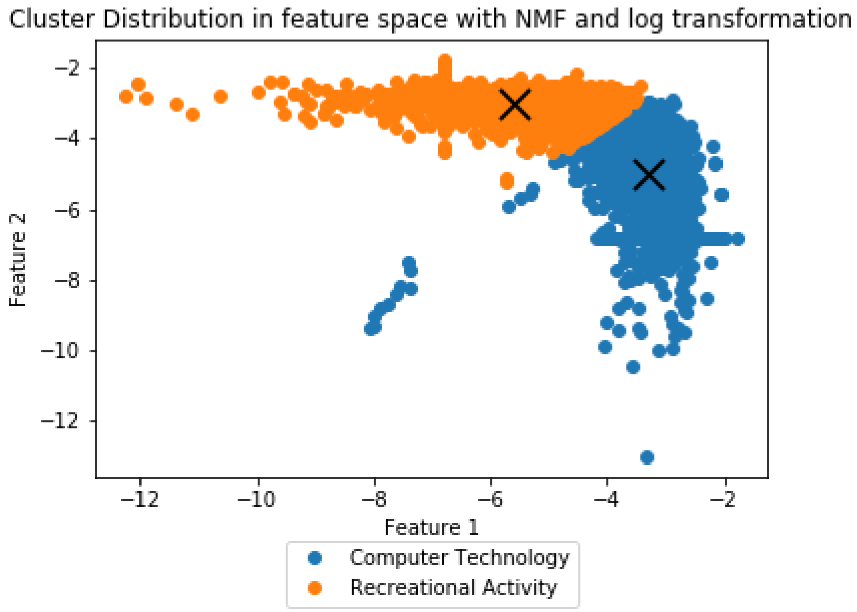


	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3534	369
Labeled Recreational Activity	106	3873

Analysis:
From the perspective of contingency matrix, the results of SVD and normalization became worse. For NMF, the results did not vary much.

2. Log Transformation on NMF

Before feeding NMF data into kmean classifier, log transformation of each matrix entry is performed. However, since NMF can generate zero entries. In order to avoid zero entry in logarithm. We simply added 0.00111 into zero entries so that they would be positive. The choice of bias was empirical and manually selected based on the resulted contingency matrix.



	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3701	202
Labeled Recreational Activity	189	3790

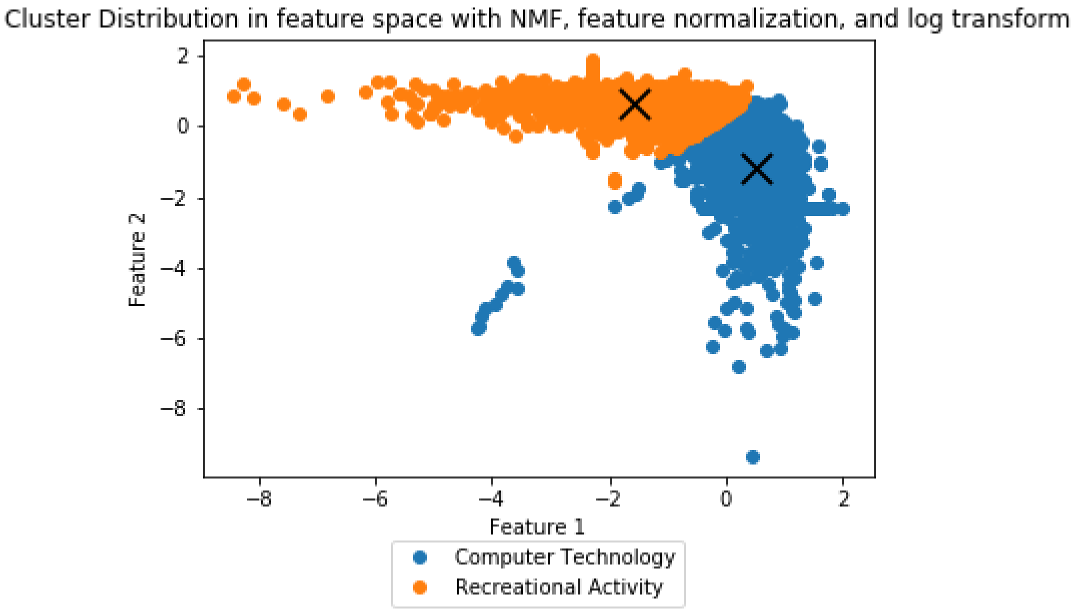
Analysis: Comparing the obtained contingency matrix to the one in Problem 4(a), we found that results became slightly better in terms of misclassified data pointes. This might be due to that the variance in the magnitude in features in greater than the the pure value of features.

3. Both Transformations

(1) Feature Nomalization and then Log Transformation

Because StandardScaler() function generates negative values by default, we set **with_mean** flag to **false** so that the mean value would not be zeroed. This procedure was recommanded in Piazza by Teaching Assistant.

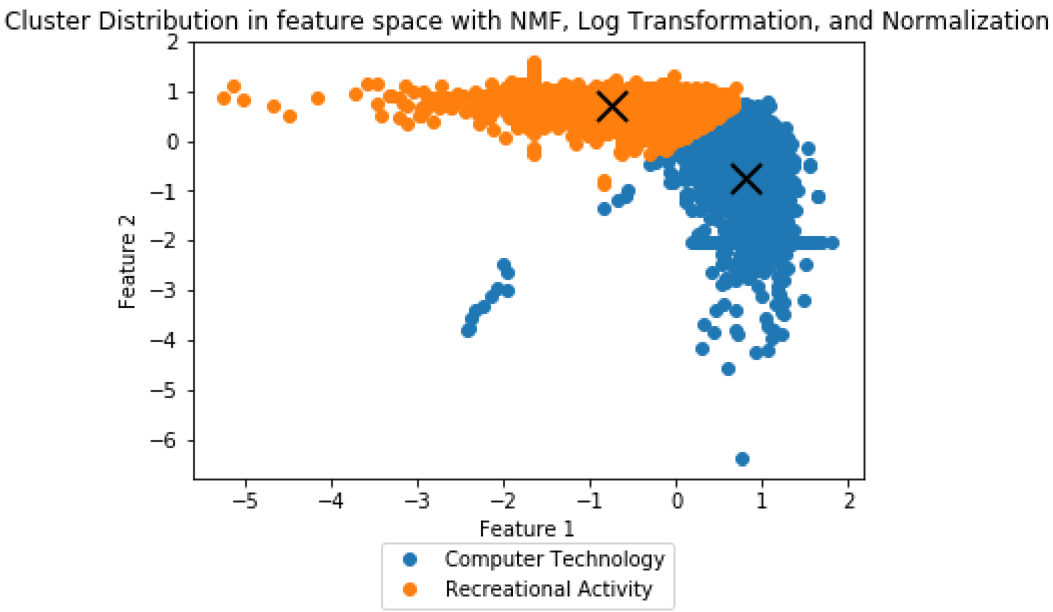
Once we obtained the non-negative values we iterated each entry of the matrix and added a small bias to zero entries. The **bias** was empirically set to **0.1**.



	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3671	232
Labeled Recreational Activity	170	3809

(2) Log Transformation and then Feature Nomalization

Once we obtained the non-negative values from NMF, we iterated each entry of the matrix and added a small bias to zero entries. The **bias** was empirically set to **0.001**. Then we we feed all positive features into logarithm transformation. Then before K-Means, we performed feature normalization by using StandardScaler() function.



	True Computer Technology	True Recreational Activity
Labeled Computer Technology	3656	247
Labeled Recreational Activity	153	3826

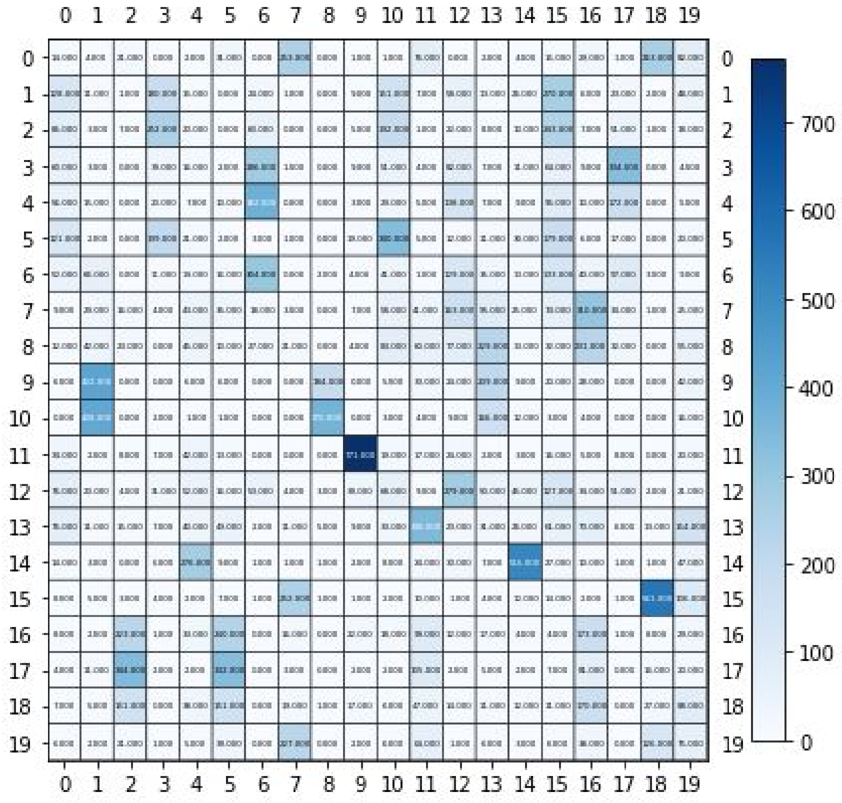
Analysis:
Either order of transformation did not large improve the homogeneity of K-means classifiers. The cluster distribution did not change much from each other and that of the original logarithm transformation.

Problem 5

We expanded dataset into 20 classes performed methods as described previously.

First thing we did is to derive TFIDF matrix, contingency matrix, 5 performance scores before dimensionality reduction.
Dimensions of TF-IDF matrix: (18846, 52295)

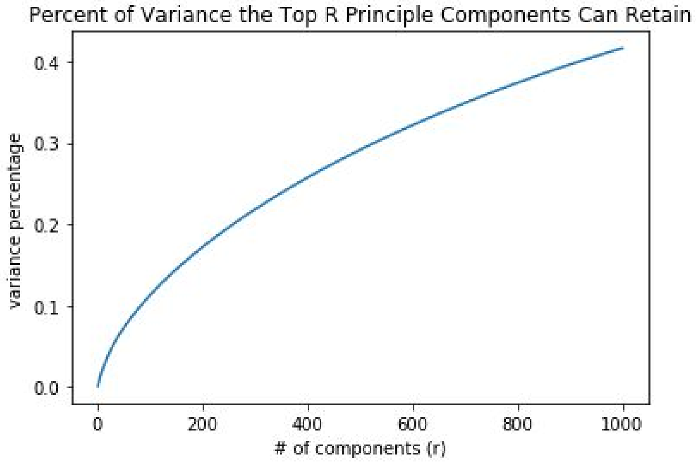
Contingency Matrix:
The darker the color is, the greater number a checker represents.



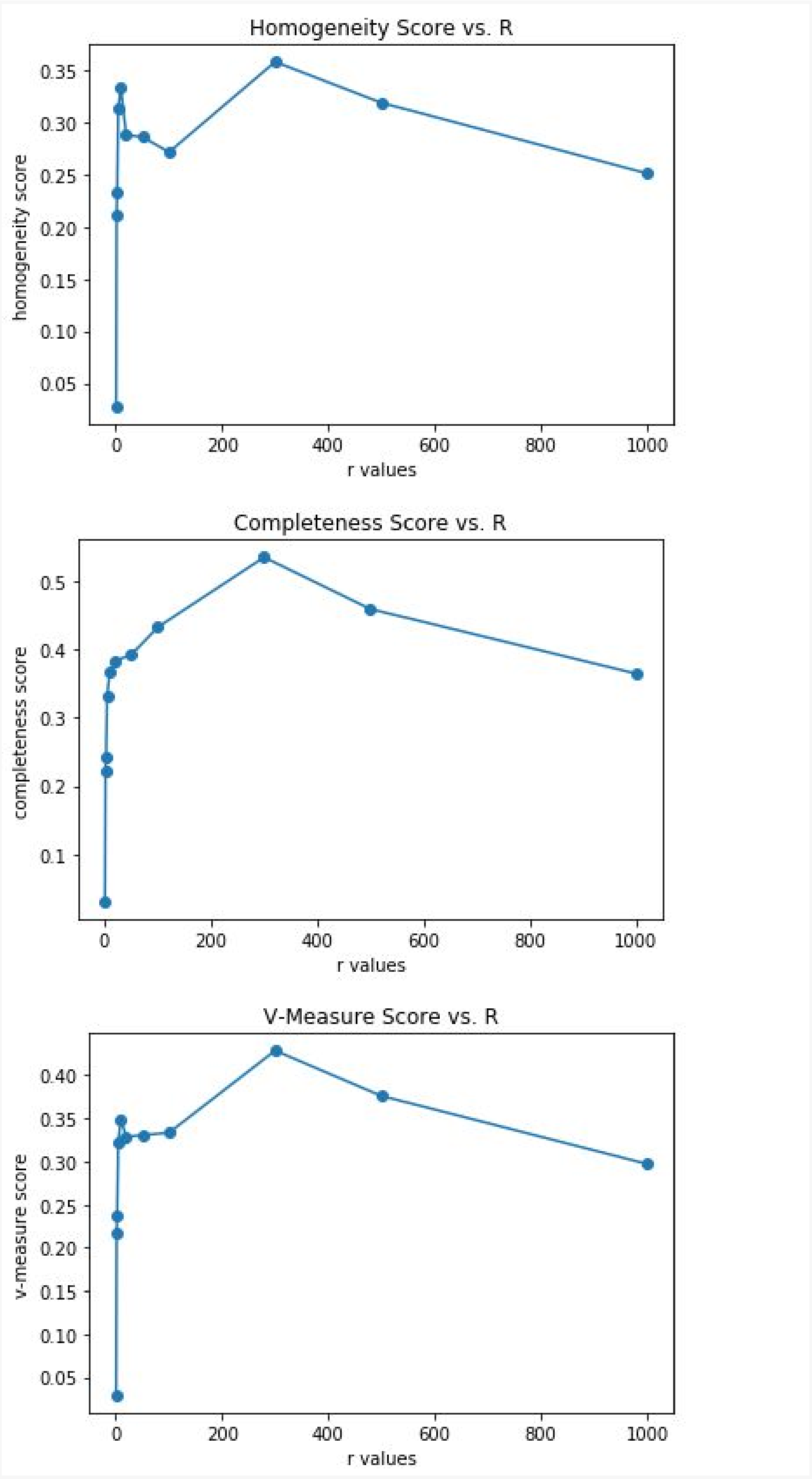
Homogeneity: 0.320873
Completeness: 0.377859
V-measure: 0.347042
Adjusted Rand Score: 0.118117
Adjusted Mutual Info Score: 0.318669

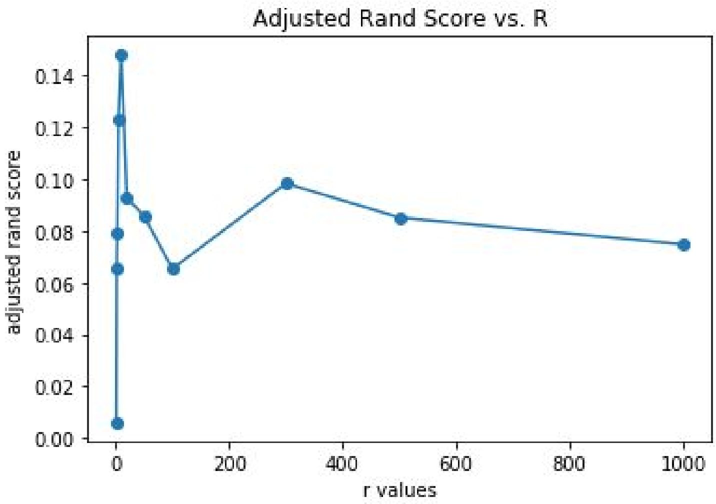
Find the best R value for LSA

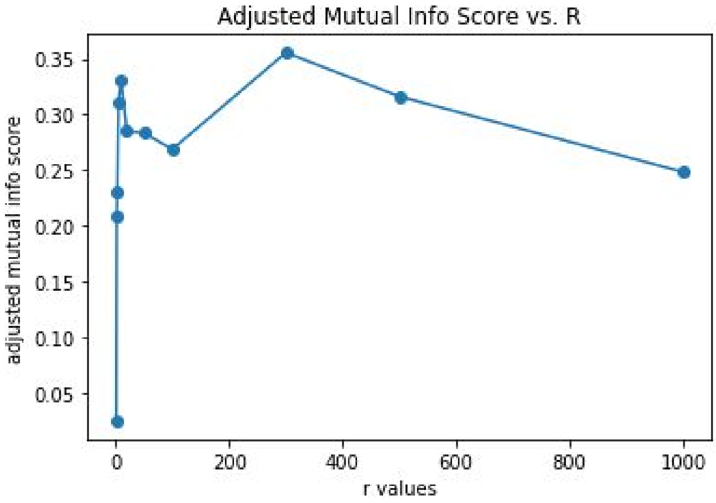
Next we performed ***percent of variance the top r principal components can retain v.s. r, for r = 1 to 1000.***



We performed SVD reduction and swept r through [1, 2, 3, 5, 10, 20, 50, 100, 300, 500, 1000], and get the following performance scores curves.

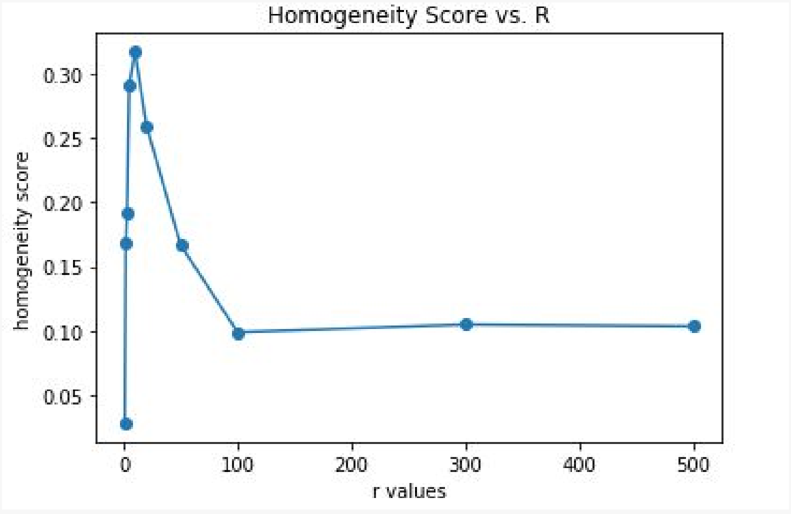


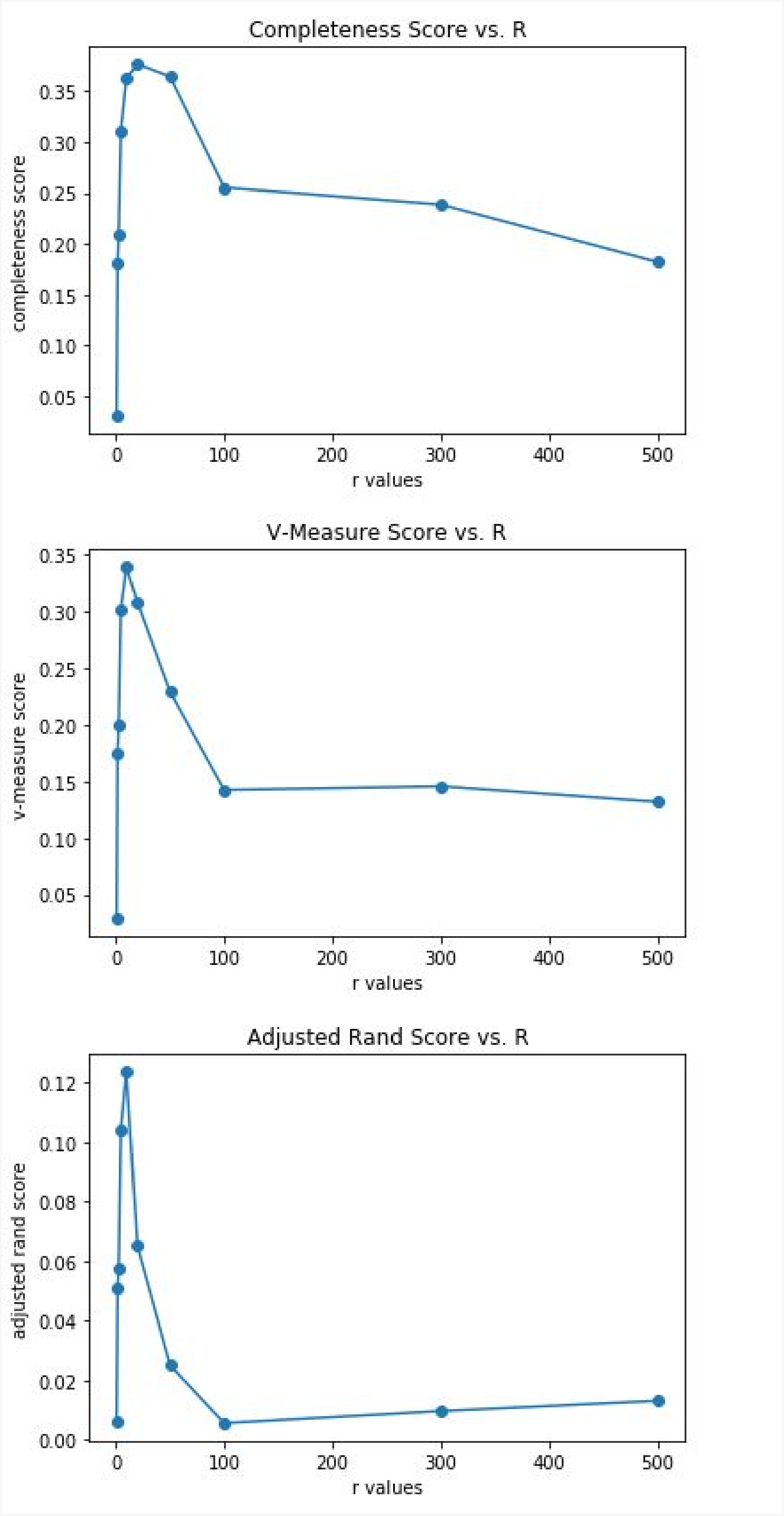


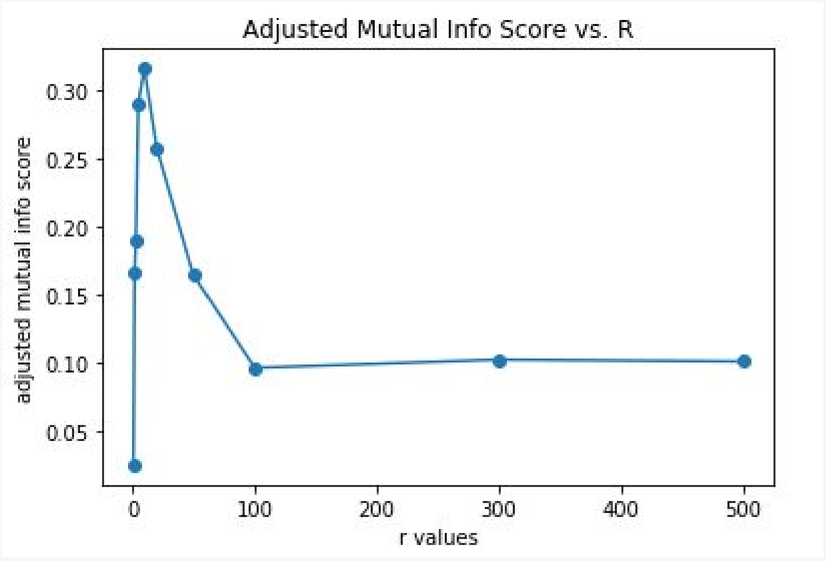


Find the best R value for NMF

Then we performed SVD reduction and swept r through [1, 2, 3, 5, 10, 20, 50, 100, 300, 500], and get the following performance scores curves.



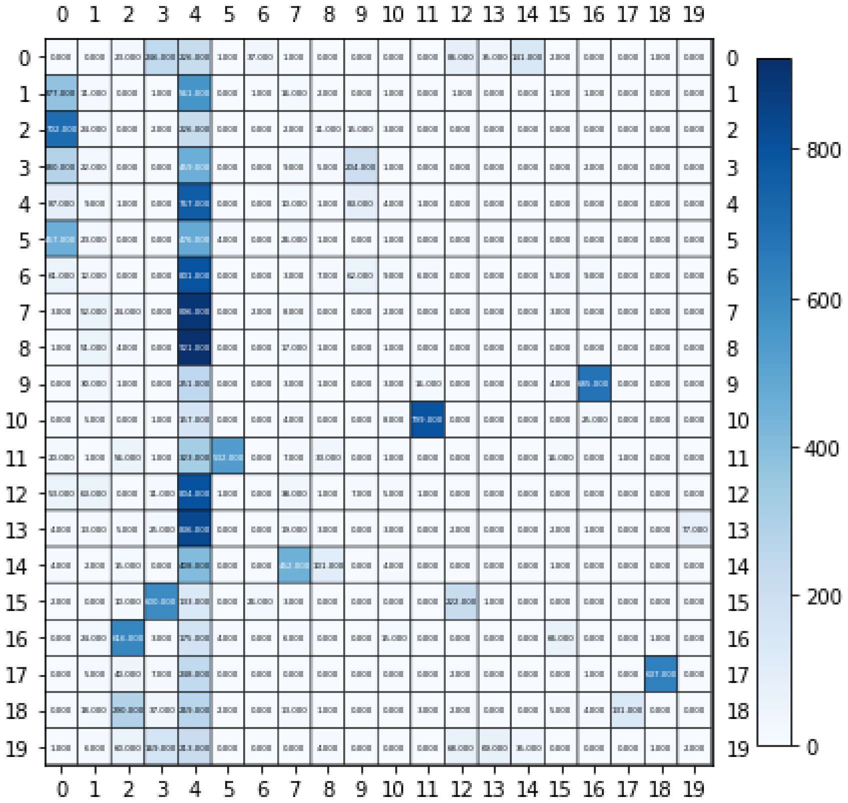

Completeness Score vs. R
completeness score
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
100
200
300
400
500
r values
V-Measure Score vs. R
v-measure score
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
100
200
300
400
500
r values
Adjusted Rand Score vs. R
adjusted rand score
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
100
200
300
400
500
r values



We find that LSI method achieves the best performance score at r = 300, whereas NMF method achieves best performance score at r = 10.

Best Results of SVD when r = 300

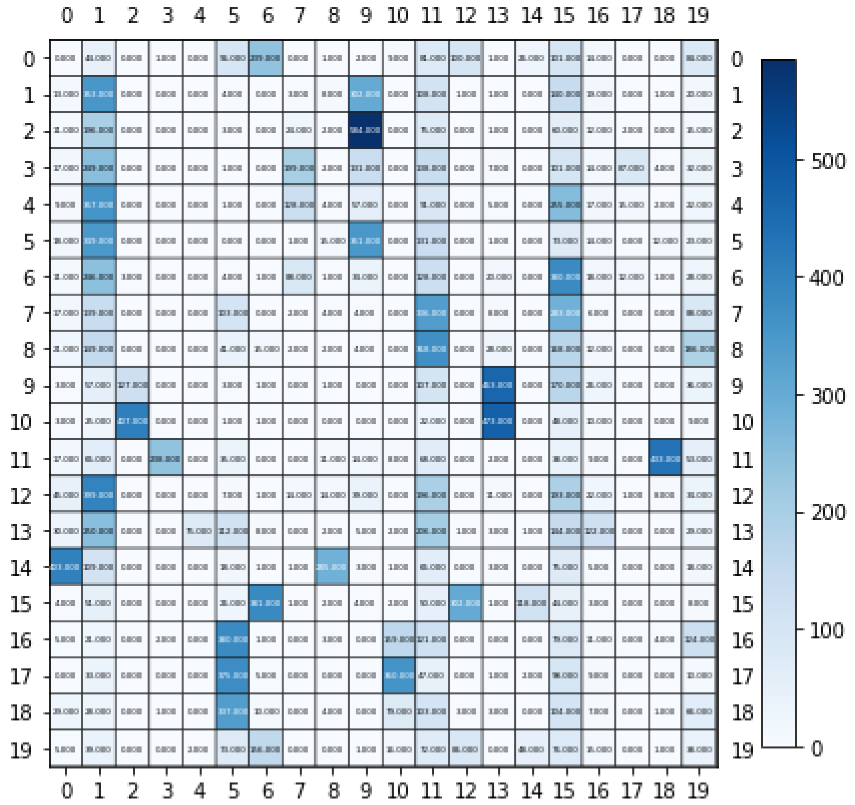
Contingency Matrix:



Homogeneity: 0.357572
Completeness: 0.535362
V-measure: 0.428768
Adjusted Rand Score: 0.098286
Adjusted Mutual Info Score: 0.355455

Best Results of NMF when r = 10

Contingency Matrix:



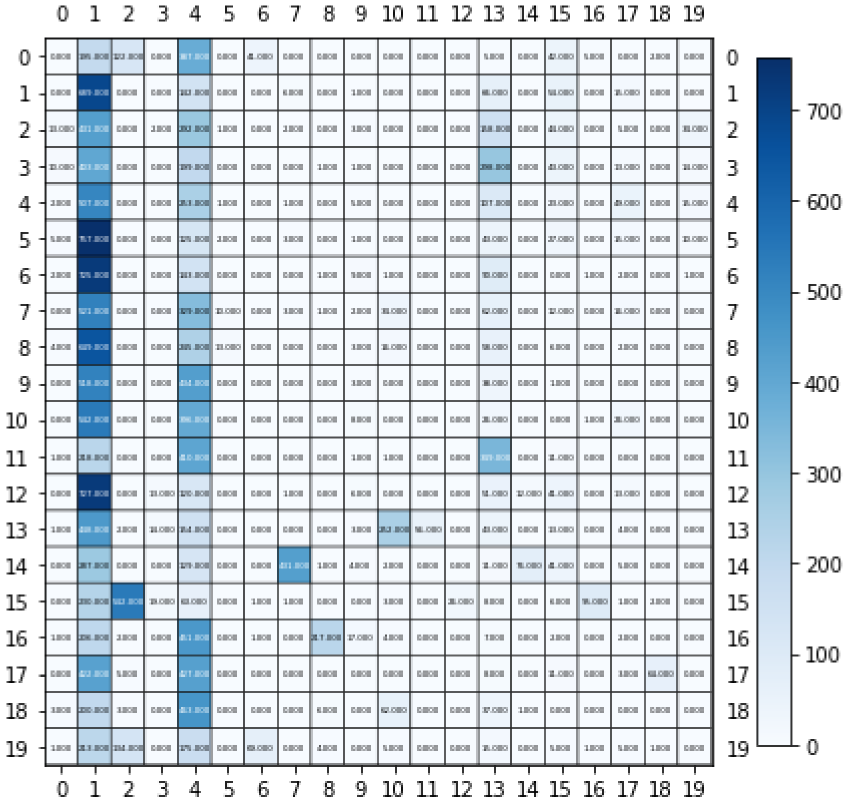
Homogeneity: 0.317665
Completeness: 0.362294
V-measure: 0.338515
Adjusted Rand Score: 0.123640
Adjusted Mutual Info Score: 0.315446

Analysis:
By comparing the scores above, we found out SVD did better job than NMF did in classifying the entire dataset into 20 classes.

Feature Normalization

SVD with r = 300

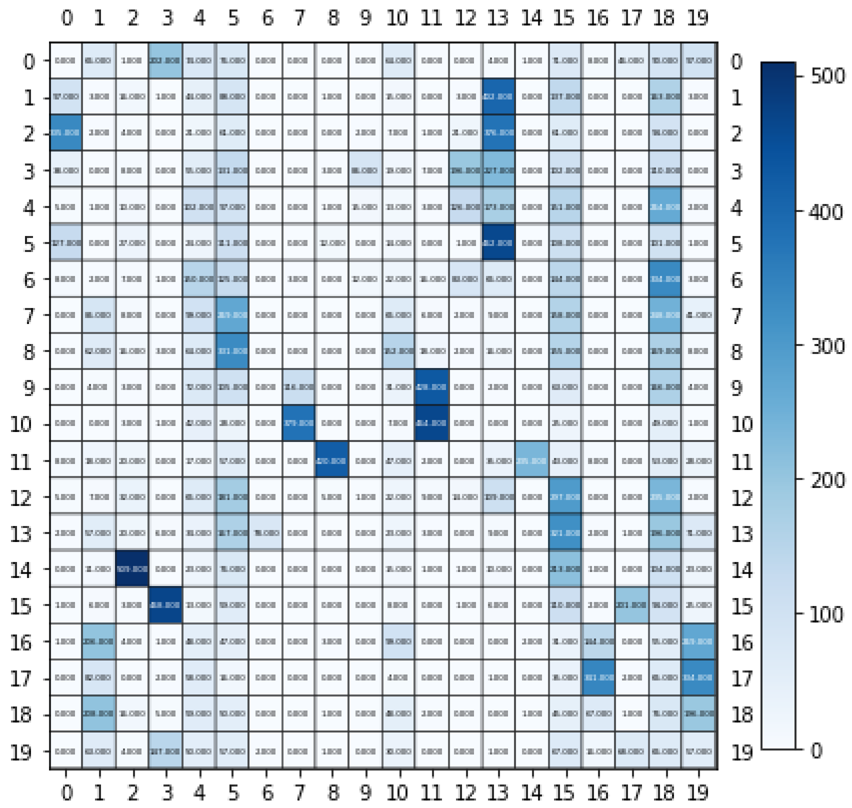
Contingency Matrix:



Homogeneity: 0.148145
Completeness: 0.276448
V-measure: 0.192912
Adjusted Rand Score: 0.030520
Adjusted Mutual Info Score: 0.145237

NMF with r = 10

Contingency Matrix:

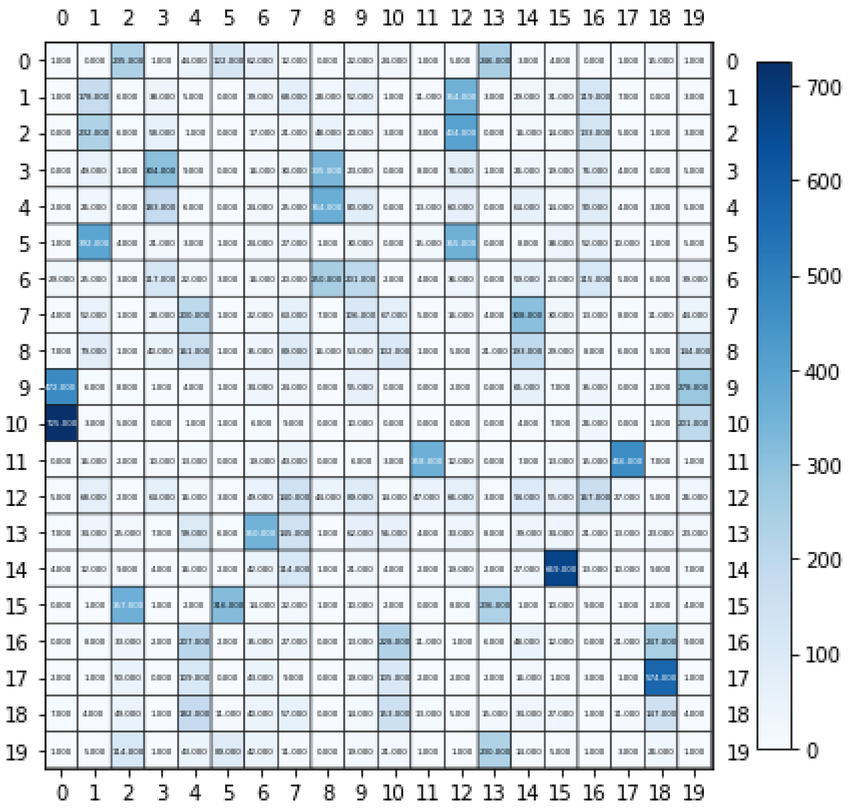


Homogeneity: 0.307228
Completeness: 0.338926
V-measure: 0.322299
Adjusted Rand Score: 0.120608
Adjusted Mutual Info Score: 0.304978

Analysis:
By performing the feature normalization, the classification results of SVD data became worse with homogeneity score as low as 0.15. However, the feature normalization did not really influence the classification results of NMF data.

Logarithm Transformation with NMF and r = 10

Contingency Matrix:



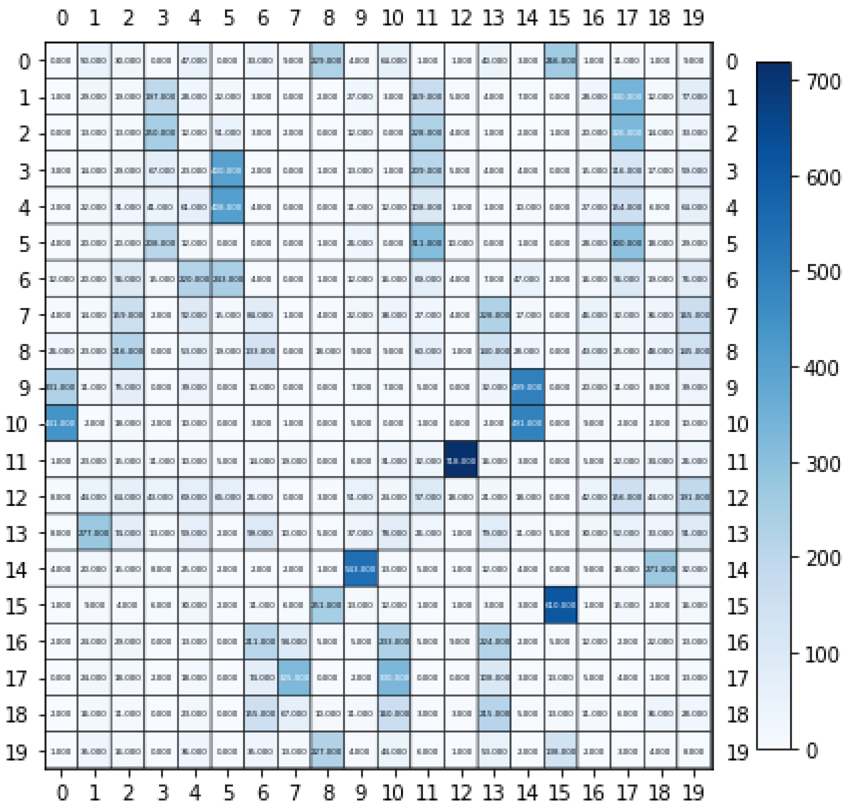
Homogeneity: 0.374389
Completeness: 0.377642
V-measure: 0.376008
Adjusted Rand Score: 0.208431
Adjusted Mutual Info Score: 0.372370

Both Transformations with NMF and r = 10

(1) Feature Normalization and then Logarithm Transformation

Before doing log transformation of feature normalized data, we just added a empirical bias of 0.1 to every zero entry to avoid runtime error of log computation.

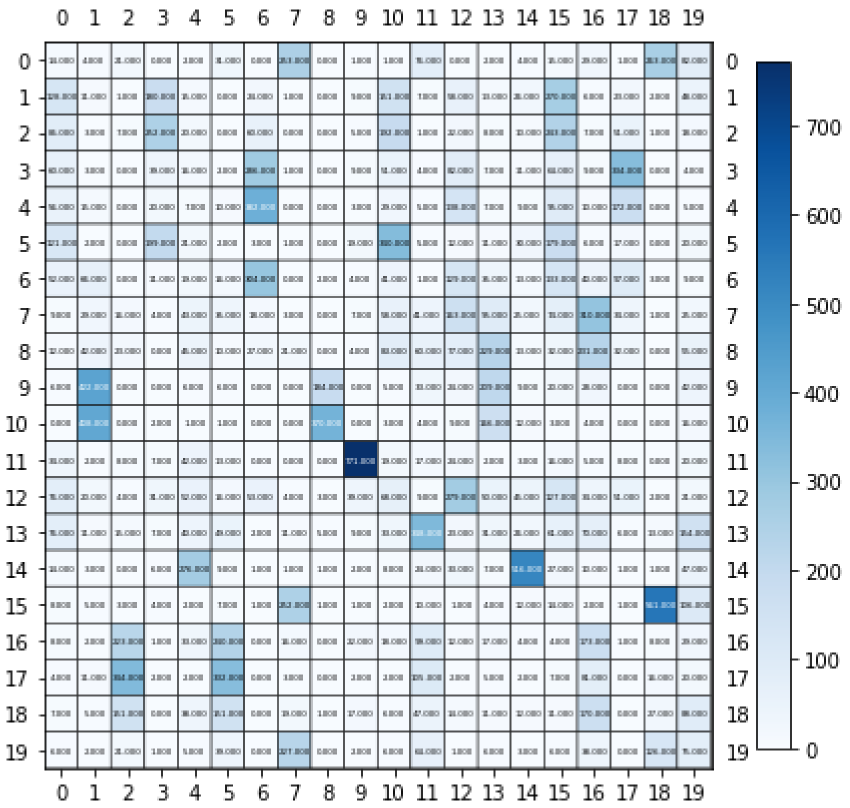
Contingency Matrix:



Homogeneity: 0.362442
Completeness: 0.367884
V-measure: 0.365142
Adjusted Rand Score: 0.196914
Adjusted Mutual Info Score: 0.360384

(2) Logarithm Transformation and then Feature Normalization
Before doing log transformation of NMF data, we just added a empirical bias of 0.001 to every zero entry to avoid runtime error of log computation.

Contingency Matrix:



Homogeneity: 0.365961
Completeness: 0.367951
V-measure: 0.366953
Adjusted Rand Score: 0.194651
Adjusted Mutual Info Score: 0.363915

Analysis:
Changing the order of the transformation did not influence the classification results of NMF. In general, k-means clustering technique does poor job in classifying data with high dimensional features.