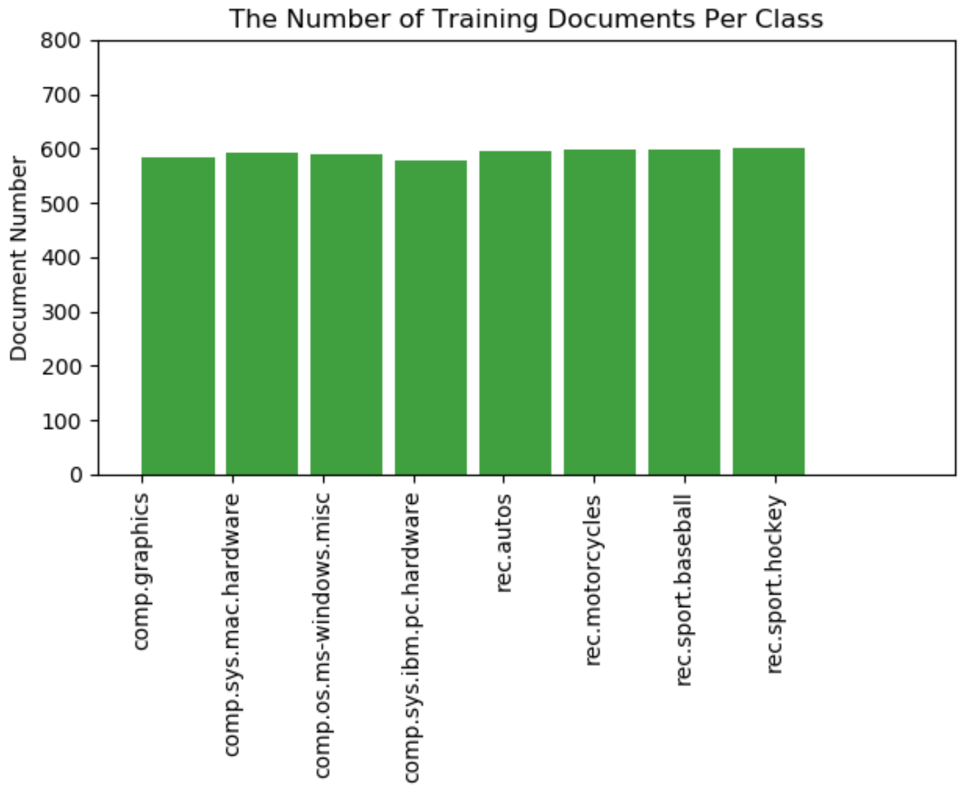
EE219 Project1
Classification Analysis on Textual Data

Qidi Sang 705028670
Hui Wang 205036597

I. Dataset and Problem Statement

(a) plot histogram



In the above histogram, X-axis represents 8 different classes in Table 1 and Y-axis represents the number of training documents in each class. As we can see, training documents in each of the 8 classes are approximately evenly distributed.

II. Modeling Text Data and Feature Extraction

(b) TFxIDF extract terms

In order to get the TFxIDF vector representation of each document, we processed the documents in the following steps:

(1) Exclude numbers and punctuations, including !"#$%&()*+,-./:;<=>?@[\\]^_`{|}, since we only care about words in each documents;

(2) Stem words with SnowballStemmer in the nltk toolkit, since in English, a word has many forms or tenses which have almost the same meaning. In order to improve the accuracy of term extraction, we used the stemmed version of words;

(3) Token each documents into words and vectorize them, using CountVectorizer function;

(4) Exclude common stop words, by setting the parameter “stop_words” in CountVectorizer function;
(5) Get the TFxIDF vector representation of each document, using TfidfTransformer function.

By setting different min_df value, we got different number of extracted terms:

(1) min_df = 2 : extracted terms number = 20209
(2) min_df = 5 : extracted terms number = 9110

The number is much smaller when min_df = 5, which makes sense since we only extract a term when it appears more than twice at min_df = 2. However we only extract a term when it appears more than 5 times at min_df = 5.

(c) TFxICF find the 10 most significant terms
By setting different min_df value, we got different results:

(1) Setting min_df = 2, here are the results:
    10 most significant terms in comp.sys.ibm.pc.hardware are:
    ['scsi', 'drive', 'edu', 'ide', 'com', 'use', 'line', 'subject', 'mb', 'organ']
    10 most significant terms in comp.sys.mac.hardware are:
    ['edu', 'mac', 'line', 'subject', 'organ', 'use', 'appl', 'simm', 'scsi', 'post']
    10 most significant terms in misc.forsale are:
    ['edu', 'line', 'sale', 'subject', 'organ', 'com', 'post', 'new', 'univers', 'use']
    10 most significant terms in soc.religion.christian are:
    ['god', 'christian', 'edu', 'jesus', 'church', 'subject', 'peopl', 'line', 'say', 'believ']
(2) Setting min_df = 5, here are the results:
    10 most significant terms in comp.sys.ibm.pc.hardware are:
    ['scsi', 'drive', 'edu', 'ide', 'com', 'use', 'line', 'subject', 'mb', 'organ']
    10 most significant terms in comp.sys.mac.hardware are:
    ['edu', 'mac', 'line', 'subject', 'organ', 'use', 'appl', 'simm', 'scsi', 'post']
    10 most significant terms in misc.forsale are:
    ['edu', 'line', 'sale', 'subject', 'organ', 'com', 'post', 'new', 'univers', 'use']
    10 most significant terms in soc.religion.christian are:
    ['god', 'christian', 'edu', 'jesus', 'church', 'subject', 'peopl', 'line', 'say', 'believ']

The term significance is in a descending order in each list.
There isn’t any difference whether min_df is 2 or 5, which is not surprising, since this parameter will only affect those words that don’t appear often, but will not have any effect for the most significant words.
From the list we can see that the 10 most significant terms are highly related to corresponding topic. Though stemming might make it look weird, we can still predict what the original word is. For example, 'believ' should be ‘believe’, and it is indeed a common word in Christian topics.

III. Feature Selection
(d) reduce dimensionality by applying LSI and NMF

After the operations of part (b), the TFxIDF vectors are high-dimensional, which will lead to poor performances of the learning algorithms. Therefore, we need to reduce the dimension of TFxIDF matrix, specifically to 50-dimensional as required.
We applied two different ways to reduce dimensionality:

(1) Latent Semantic Indexing (LSI)
By applying this method, singular value decomposition is performed on the TFxIDF matrix. We used the TruncatedSVD function, and set parameters n_components = 50, algorithm = 'arpack'. Both train data and test data were mapped to a 50-dimensional matrix.

(2) Non-Negative Matrix Factorization (NMF)
Likewise, we also reduced the dimension of both train data and test data using NMF function. We also set n_components = 50, corresponding to 50-dimension, used 'random' method to initialize the procedure, and random_state = 0 to get a fixed output.

After applying LSI and NMF methods, we were able to use the learning algorithms in parts (e)-(i). It is noteworthy that after LSI, the data contains negative values while after NMF, the data is non-negative.

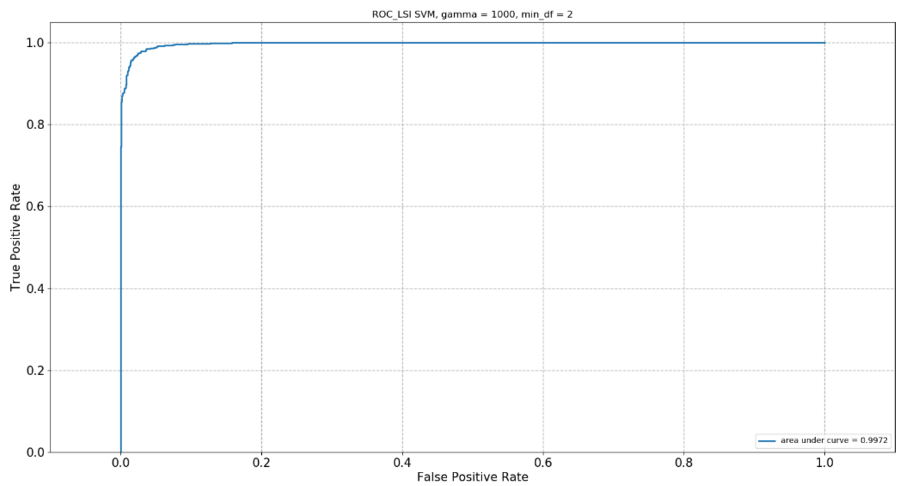
IV. Learning Algorithms

(e) Hard and soft SVM Classifier (SVC)

(1) Hard margin SVM

1) LSI, min_df = 2

For hard margin SVM with LSI, we set γ = 1000, here are the results:

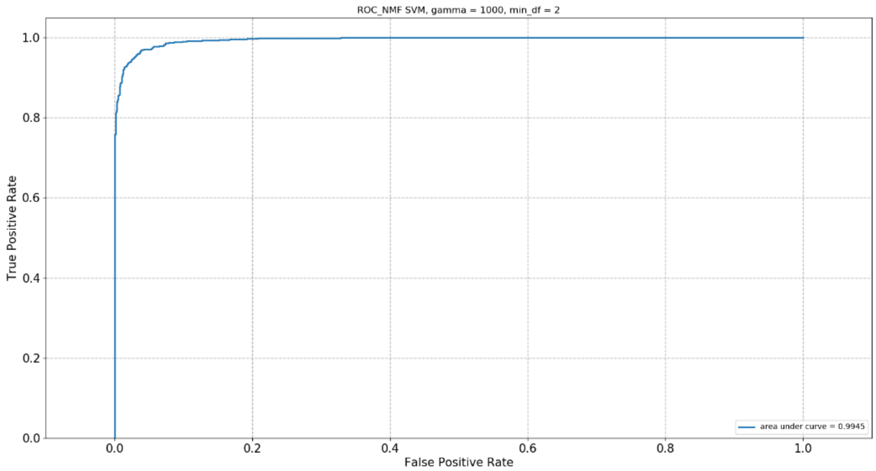



```
Confusion matrix:
[[1522   38]
 [44   1546]]

Accuracy: 0.973968253968
```

```
Classification Report:
             precision    recall  f1-score   support

       comp       0.97      0.98      0.97      1560
        rec       0.98      0.97      0.97      1590

avg / total       0.97      0.97      0.97      3150
```

2) NMF, min_df = 2

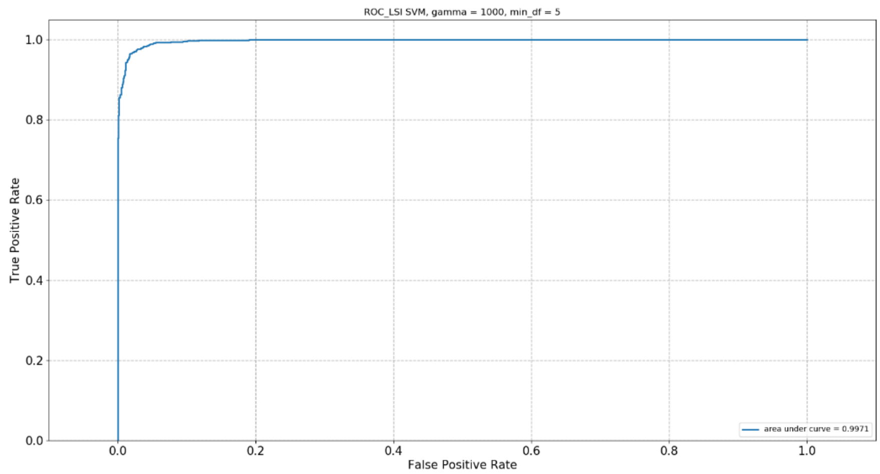
For hard margin SVM with LSI, we set γ = 1000, here are the results:




```
Confusion matrix:
[[1375  185]
 [14   1576]]

Accuracy: 0.936825396825

Classification Report:
             precision    recall  f1-score   support

       comp       0.99      0.88      0.93      1560
        rec       0.89      0.99      0.94      1590

avg / total       0.94      0.94      0.94      3150
```

3) LSI, min_df = 5

For hard margin SVM with LSI, we set γ = 1000, here are the results:

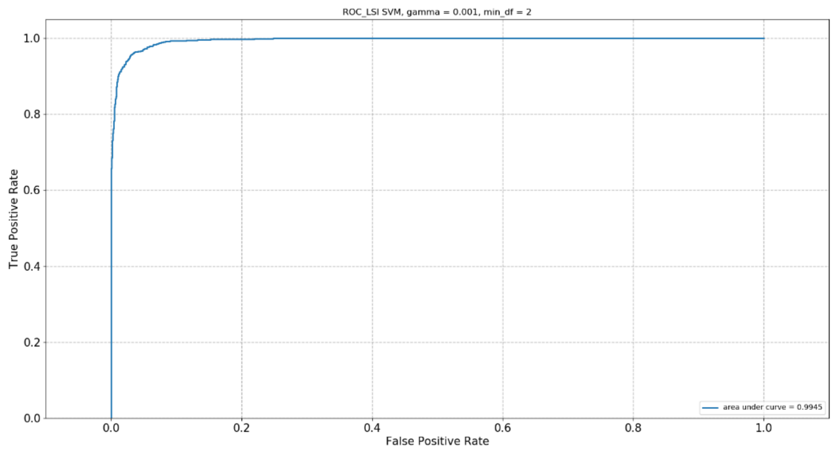



```
Confusion matrix:
[[1494   66]
 [24   1566]]

Accuracy: 0.971428571429

Classification Report:
             precision    recall  f1-score   support

       comp       0.98      0.96      0.97      1560
        rec       0.96      0.98      0.97      1590

avg / total       0.97      0.97      0.97      3150
```

(2) Soft margin SVM

1) LSI, min_df = 2

For soft margin SVM with LSI, we set γ = 0.001, here are the results:



Confusion matrix:
[[1367 193]
 [9   1581]]

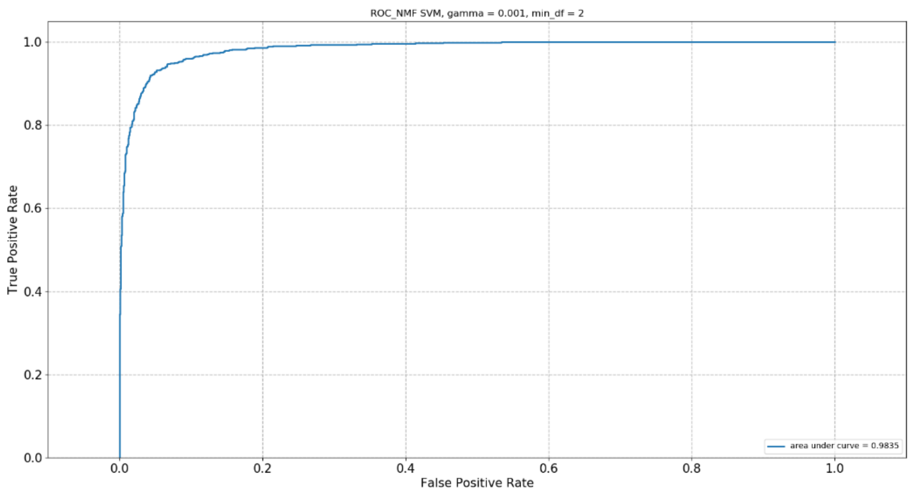
Accuracy: 0.935873015873

Classification Report:

	precision	recall	f1-score	support
comp	0.99	0.88	0.93	1560
rec	0.89	0.99	0.94	1590
avg / total	0.94	0.94	0.94	3150

2) NMF, min_df = 2

For soft margin SVM with NMF, we set γ = 0.001, here are the results:



Confusion matrix:
[[66 1494]
 [0   1590]]

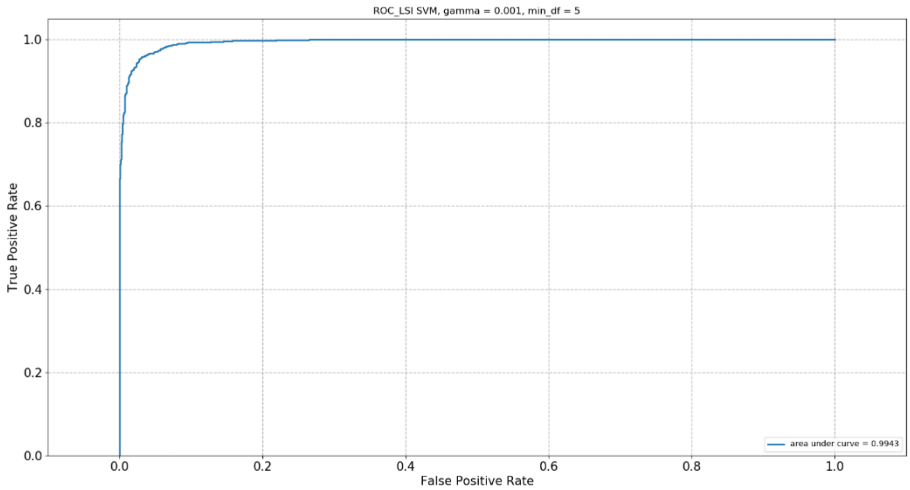
Accuracy: 0.525714285714

Classification Report:

	precision	recall	f1-score	support
comp	1.00	0.04	0.08	1560
rec	0.52	1.00	0.68	1590
avg / total	0.76	0.53	0.38	3150

3) LSI, min_df = 5

For soft margin SVM with LSI, we set γ = 0.001, here are the results:




```
Confusion matrix:
[[1378  182]]
 [10   1580]

Accuracy: 0.939047619048

Classification Report:
             precision    recall  f1-score   support

       comp       0.99      0.88      0.93      1560
        rec       0.90      0.99      0.94      1590

avg / total       0.94      0.94      0.94      3150
```

From the above 6 sets of results, we can see that:
Overall, LSI is better than NMF, hard SVM is better than soft SVM; and when we use soft margin SVM, in terms of accuracy, LSI just dropped a little bit, from 0.97 to 0.94, while NMF dropped drastically, from 0.93 to 0.52.
This could be that NMF is trying to model our text based on term probabilities, which makes the difference between topics smaller (since probabilities must be non-zero while LSI can take negative values), and a soft margin SVM can't make good judgement when difference is small. Besides, we also did NMF SVM when min_df = 5, and the soft margin NMF accuracy increases from 0.52 changes to 0.62, which also implies that by getting rid of some low-probability terms NMF can do better, especially when applying soft margin SVM.

For LSI, change of γ and change of min_df seems to have little influence. It is comparatively stable.

(f) 5-fold cross-validation

(1) min_df = 2

Using LSI SVM, here are the results of accuracy corresponding to γ:

γ	accuracy
0.001	0.914602025758
0.01	0.974851037325
0.1	0.983727850941
1	0.986686565565
10	0.986686565565
100	0.986263732584
1000	0.986263732584

Using NMF SVM, here are the results of accuracy corresponding to γ:

γ	accuracy
0.001	0.914602025758
0.01	0.974851037325
0.1	0.983727850941
1	0.986686565565
10	0.986686565565
100	0.986263732584
1000	0.986263732584

(2) min_df = 5

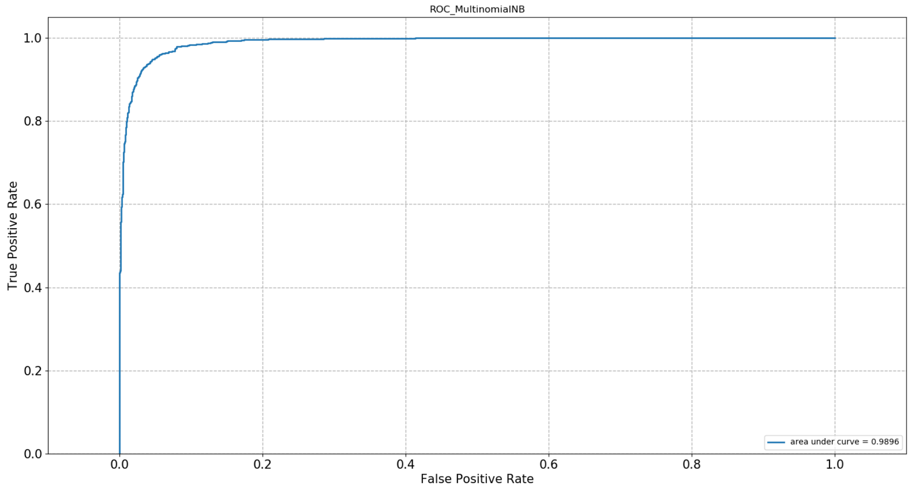
Using LSI SVM, here are the results of accuracy corresponding to γ:

γ	accuracy
0.001	0.924327184321
0.01	0.973372461384
0.1	0.983094047967
1	0.984572847157
10	0.983516880948
100	0.984150907171
1000	0.984150907171

It appears that for both min_df = 2 and 5, γ = 1 is the best choice in terms of accuracy for LSI.

And γ = 1 is also the best choice for NMF.

(g) Naïve Bayes Classifier
After reducing dimensionality, data is passed to Naïve Bayes Classifier for training. Since Multinomial Naïve Bayes Classifier only takes non-negative values, we only used data after NMF method. We used MultinomialNB function, got the prediction of test data and the corresponding prediction probability. The final result is listed below.
(The following results are from min_df = 2)



As the ROC curve shows, the true positive rate is very high even when false positive rate is very low. The threshold corresponding to the top left region of the ROC is desired to get high true positive rate and low false positive rate.

```
Confusion matrix:
[[1393  167]
 [  26 1564]]
```

In confusion matrix, all correct predictions are located in the diagonal and prediction errors will be represented by values outside the diagonal. As we can see from the above confusion matrix, the values on the diagonal are much higher than other values, which means this classifier predicted very well.

```
Accuracy: 0.93873015873
```

The accuracy of the Multinomial Naive Bayes Classifier is 93.9%, which also indicates this model fits well with the data.

Classification Report:

	precision	recall	f1-score	support
comp	0.98	0.89	0.94	1560
rec	0.90	0.98	0.94	1590

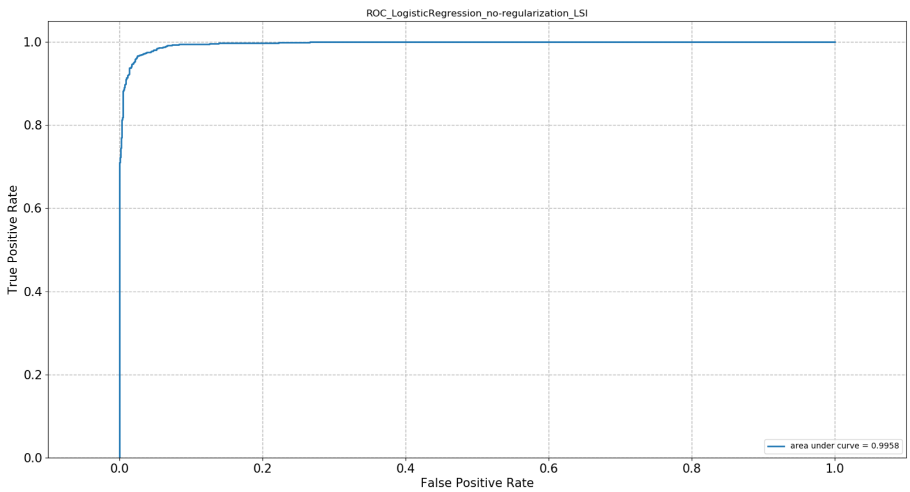
```
avg / total       0.94          0.94          0.94          3150
```

The classification report contains the precision and recall for the classier. As we can see from above, precision and recall values for both classes are pretty high (> 90%), which means the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

(h) Logistic Regression Classifier without regularization
Since logistic regression classifier takes both non-negative and negative data as input, we used two kinds of data (i.e. after LSI, NMF) here. And in this part, we used LogisticRegression function with default settings.

(1) min_df = 2

1) LSI



As the ROC curve shows, the true positive rate is very high even when false positive rate is very low. The threshold corresponding to the top left region of the ROC is desired to get high true positive rate and low false positive rate.

```
Confusion matrix:
[[1487   73]
 [  35 1555]]
```

As we can see from the above confusion matrix, the values on the diagonal are much higher than other values, which means this classifier predicted very well.

```
Accuracy: 0.965714285714
```

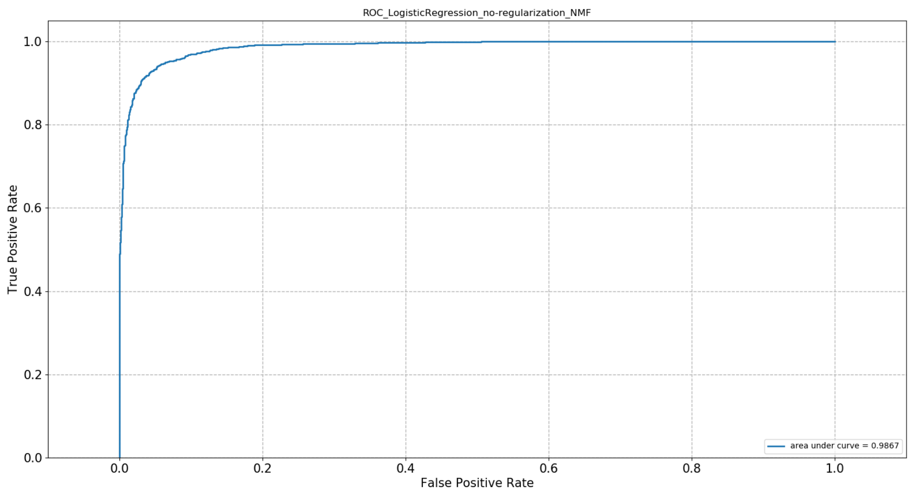
The accuracy of the Logistic Regression Classifier without regularization is

96.6%, which also indicates this model fits well with the data.

```
Classification Report:
             precision    recall  f1-score   support

       comp       0.98      0.95      0.96      1560
        rec       0.96      0.98      0.97      1590

avg / total       0.97      0.97      0.97      3150
```

As we can see from above, precision and recall values for both classes are pretty high (> 90%), which means the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

2) NMF



As the ROC curve shows, the true positive rate is very high even when false positive rate is very low. The threshold corresponding to the top left region of the ROC is desired to get high true positive rate and low false positive rate.

```
Confusion matrix:
[[1432  128]
 [  68 1522]]
```

As we can see from the above confusion matrix, the values on the diagonal are much higher than other values, which means this classifier predicted very well.

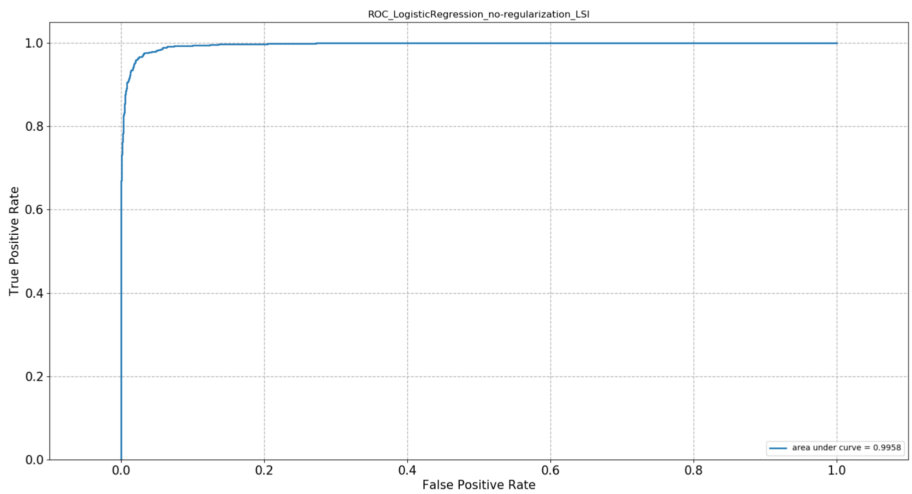
```
Accuracy: 0.937777777778
```

The accuracy of the Logistic Regression Classifier without regularization is 93.8%, which also indicates this model fits well with the data.

```
Classification Report:
             precision    recall  f1-score   support

       comp       0.95      0.92      0.94      1560
        rec       0.92      0.96      0.94      1590

avg / total       0.94      0.94      0.94      3150
```

As we can see from above, precision and recall values for both classes are pretty high (> 90%), which means the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

(2) min_df = 5 (only LSI method is required here)



As the ROC curve shows, the true positive rate is very high even when false positive rate is very low. The threshold corresponding to the top left region of the ROC is desired to get high true positive rate and low false positive rate.

```
Confusion matrix:
[[1483   77]
 [  29 1561]]
```

As we can see from the above confusion matrix, the values on the diagonal are much higher than other values, which means this classifier predicted very well.

```
Accuracy: 0.966349206349
```

The accuracy of the Logistic Regression Classifier without regularization is 96.6%, which also indicates this model fits well with the data.

```
Classification Report:
             precision    recall  f1-score   support

       comp       0.98      0.95      0.97      1560
        rec       0.95      0.98      0.97      1590

avg / total       0.97      0.97      0.97      3150
```

As we can see from above, precision and recall values for both classes are pretty high (> 90%), which means the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

(i) Logistic Regression Classifier with regularization

In this part, we added a regularization term to the optimization objective. We performed both l1 and l2 norm regularizations, with a range of different regularization coefficients, and used accuracy as a way to evaluate the performance of the classifier.

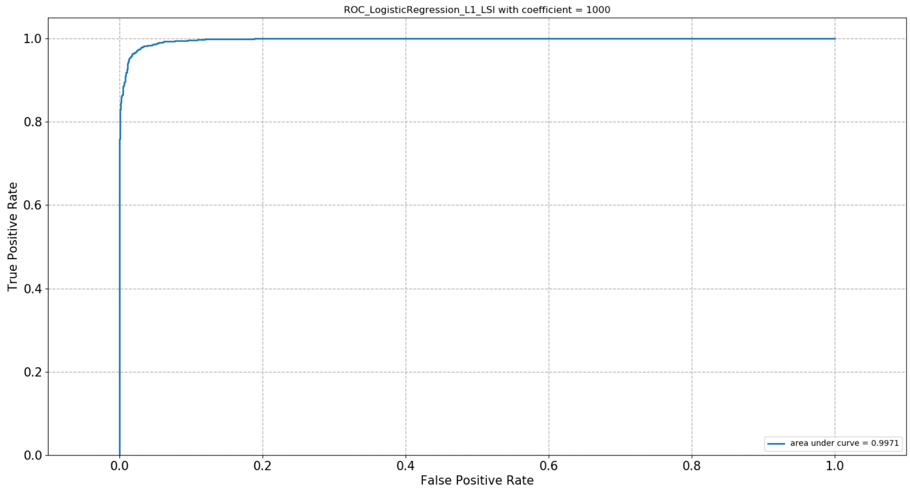
Specifically, we tried 5 different regularization coefficients = 0.001, 0.1, 10, 1000, 100000, ranging from very small ones to large ones. The results are as below.

(1) min_df = 2

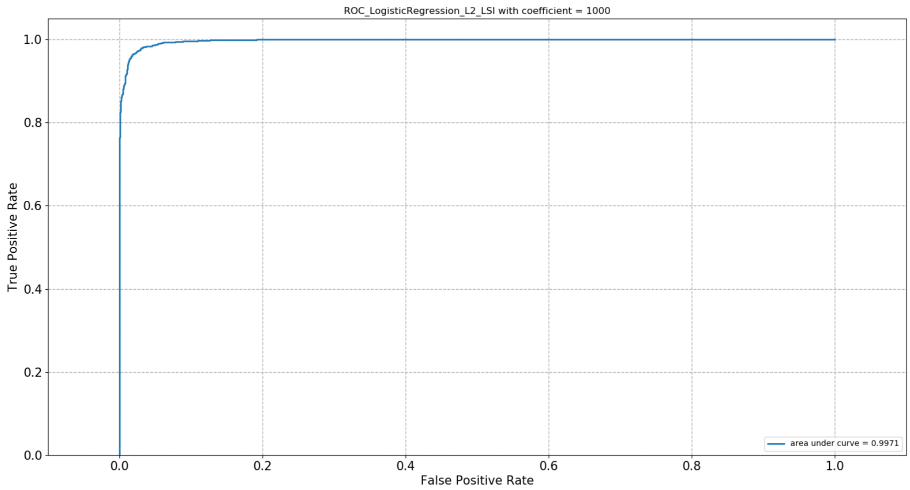
1) LSI

C / norm	0.001	0.1	10	1000	100000
l1	0.495238095238	0.953968253968	0.975238095238	0.973650793651	0.973650793651
l2	0.713333333333	0.961587301587	0.973015873016	0.973650793651	0.973650793651

From the values in above table, we can see that for both l1 and l2 penalty, the accuracy increases as C getting larger. The optimal accuracy is achieved when C = 1000, which is a large value. The ROC curve and metrics analysis of l1 and l2 for optimal parameter C are listed below.




```
Confusion matrix of l1:
[[1507   53]
 [  30 1560]]

Accuracy of l1: 0.973650793651

Classification Report of l1:
             precision    recall  f1-score   support

       comp       0.98      0.97      0.97      1560
        rec       0.97      0.98      0.97      1590

avg / total       0.97      0.97      0.97      3150
```



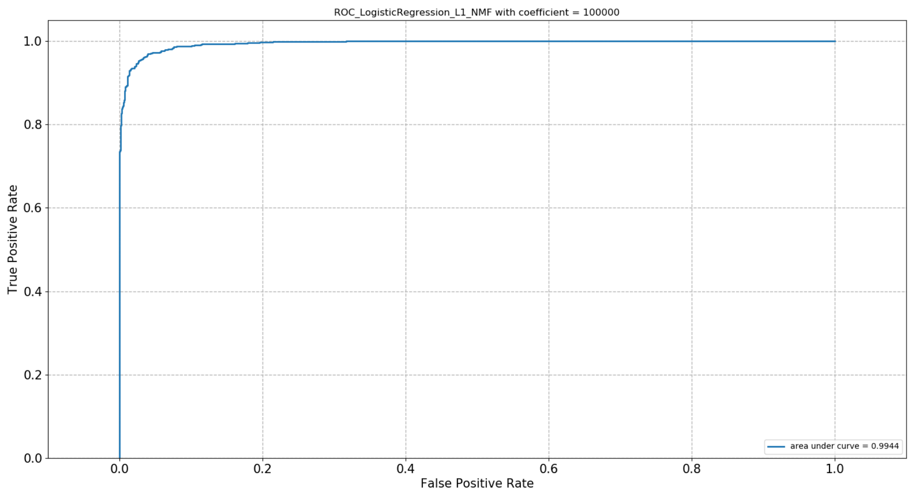

```
Confusion matrix of l2:
[[1506   54]
```

```
 [  29 1561]]

Accuracy of l2: 0.973650793651

Classification Report of l2:
                precision   recall   f1-score   support

        comp      0.98       0.97      0.97      1560
         rec      0.97       0.98      0.97      1590

avg / total       0.97       0.97      0.97      3150
```

2) NMF

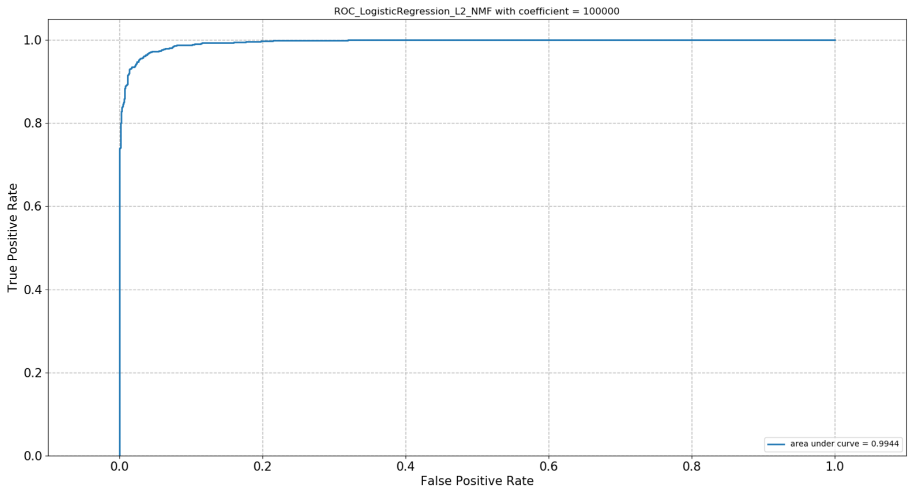
C / norm	0.001	0.1	10	1000	100000
l1	0.495238095238	0.719682539683	0.963492063492	0.96380952381	0.96380952381
l2	0.504761904762	0.892380952381	0.948253968254	0.961587301587	0.96380952381

Likewise, for both l1 and l2 penalty, the accuracy increases as C getting larger. The optimal accuracy is achieved when C = 100000, which is a large value. The ROC curve and metrics analysis of l1 and l2 for optimal parameter C are listed below.




```
Confusion matrix of l1:
[[1495   65]
 [  49 1541]]

Accuracy of l1: 0.96380952381

Classification Report of l1:
                precision   recall   f1-score   support
```

```
            comp     0.97     0.96     0.96     1560
             rec     0.96     0.97     0.96     1590

     avg / total     0.96     0.96     0.96     3150
```

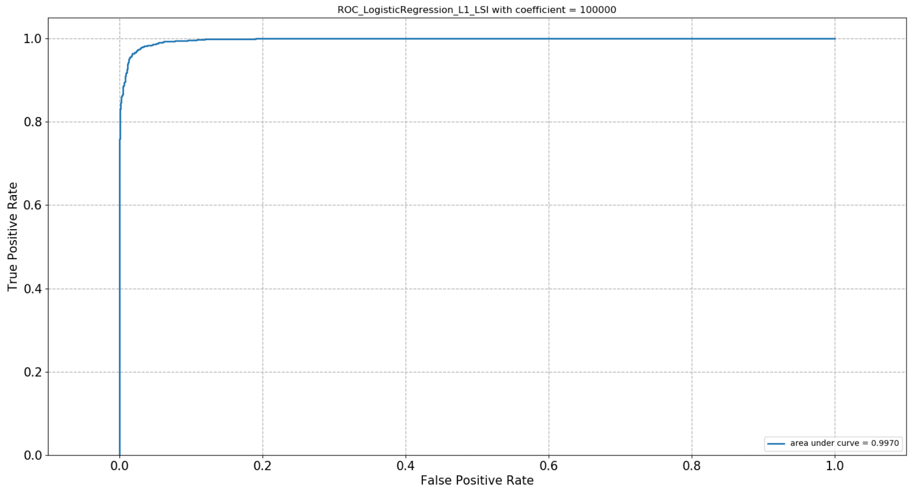



```
Confusion matrix of l2:
[[1495   65]
 [  49 1541]]

Accuracy of l2: 0.96380952381

Classification Report of l2:
              precision    recall  f1-score   support

            comp     0.97     0.96     0.96     1560
             rec     0.96     0.97     0.96     1590

     avg / total     0.96     0.96     0.96     3150
```

(2) min_df = 5 (only LSI required)

norm \ C	0.001	0.1	10	1000	100000
l1	0.495238095238	0.949206349206	0.973650793651	0.973650793651	0.973650793651
l2	0.755238095238	0.962222222222	0.973015873016	0.974285714286	0.973650793651

Likewise, for both l1 and l2 penalty, the accuracy increases as C getting larger. The optimal accuracy is achieved when C = 100000, which is a large value. The ROC curve and metrics analysis of l1 and l2 for optimal parameter C are listed below.

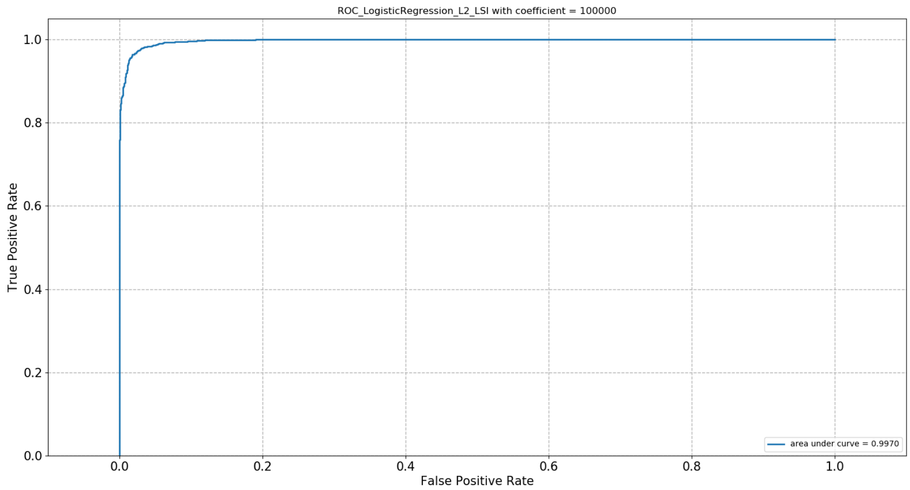


Confusion matrix of l1:
[[1506   54]
 [29 1561]]

Accuracy of l1: 0.973650793651

Classification Report of l1:

	precision	recall	f1-score	support
comp	0.98	0.97	0.97	1560
rec	0.97	0.98	0.97	1590
avg / total	0.97	0.97	0.97	3150



Confusion matrix of l2:
[[1506   54]

```
 [  29 1561]]

Accuracy of l2: 0.973650793651

Classification Report of l2:
             precision    recall  f1-score   support

       comp       0.98      0.97      0.97      1560
        rec       0.97      0.98      0.97      1590

avg / total       0.97      0.97      0.97      3150
```

From the above results, we can find that in both l1 and l2 regularization, the accuracy increases with the regularization coefficient C. When C is very small (near 0), the accuracy is very low, and small C has larger influence on l1 penalty.

With the increased parameter C, the absolute value of coefficients of the fitted hyperplane is also increasing.

Both l1 and l2 norm regularization are used to avoid overfitting. Thus adding either a l1 or l2 penalty will raise the accuracy of prediction. However, if the penalty term is too large (i.e. when C is very small), the coefficients turn to be close to 0, it means that the classifier will delete certain dimensions when fitting, and could potentially lead to poor accuracy, which is also known as underfitting.

V. Multiclass Classification

(1) Multiclass Naïve Bayes Classification

1) LSI

Since Multinomial Naïve Bayes Classifier can only take non-negative values, we used Gaussian Naïve Bayes Classifier instead, by calling GaussianNB function. The results are as below:

```
Confusion matrix:
[[231  33 128   0]
 [ 83 156 144   2]
 [ 44  31 314   1]
 [  0   0  31 367]]
```

The values on the principal diagonal of the confusion matrix is much higher than other values, which means the classifier predicted quite well.

```
Accuracy: 0.682428115016
```

The accuracy of the Gaussian Naïve Bayes Classifier is not very high, lower than that of SVM, which means the SVM fits the data better.

```
Classification Report:
```

	precision	recall	f1-score	support
comp.sys.ibm.pc.hardware	0.65	0.59	0.62	392
comp.sys.mac.hardware	0.71	0.41	0.52	385
misc.forsale	0.51	0.81	0.62	390
soc.religion.christian	0.99	0.92	0.96	398
avg / total	0.72	0.68	0.68	1565

As the classification report shows, the precision of comp.sys.ibm.pc.hardware , comp.sys.mac.hardware and misc.forsale are relatively low. Also the recall of these three categories are not very high, which indicates not all positive results are returned. By comparison, soc.religion.christian has high precision and recal, which means this classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall). On average, precision and recall of the classifier is not very high.

2) NMF

Since data after NMF are non-negative, we used Multinomial Naïve Bayes Classifier here. The results are as below:

```
Confusion matrix:
[[311  30  47   4]
 [ 89 236  51   9]
 [ 59  14 303  14]
 [  0   2   4 392]]
```

The values on the principal diagonal of the confusion matrix is much higher than other values, which means the classifier predicted quite well.

Accuracy: 0.793610223642

The accuracy of the Multinomial Naïve Bayes Classifier is not very high, lower than that of SVM, which means the SVM fits the data better.

Classification Report:

	precision	recall	f1-score	support
comp.sys.ibm.pc.hardware	0.68	0.79	0.73	392
comp.sys.mac.hardware	0.84	0.61	0.71	385
misc.forsale	0.75	0.78	0.76	390
soc.religion.christian	0.94	0.98	0.96	398
avg / total	0.80	0.79	0.79	1565

Likewise, the precision of the first 3 categories are relatively low. Also the recall of these three categories are not very high, which indicates not all positive results are returned. By comparison, soc.religion.christian has high precision and recal, which means this classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall). On average, precision and recall of the classifier is not very high.

(2) One vs One SVM

1) LSI

```
Confusion matrix:
[[324  45  23   0]
 [40  322  22   1]
 [28   22  339  1]
 [5     2   3  388]]

Accuracy: 0.87731629393

Classification Report:
                           precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.82      0.83      0.82       392
   comp.sys.mac.hardware       0.82      0.84      0.83       385
            misc.forsale       0.88      0.87      0.87       390
  soc.religion.christian       0.99      0.97      0.98       398

             avg / total       0.88      0.88      0.88      1565
```

2) NMF

```
Confusion matrix:
[[299  61  32   0]
 [65  289  30   1]
 [55   17  317  1]
 [10   3   21  364]]

Accuracy: 0.810862619808

Classification Report:
                           precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.70      0.76      0.73       392
   comp.sys.mac.hardware       0.78      0.75      0.77       385
            misc.forsale       0.79      0.81      0.80       390
  soc.religion.christian       0.99      0.91      0.95       398

             avg / total       0.82      0.81      0.81      1565
```

(3) One vs Rest SVM

1) LSI

```
Confusion matrix:
[[318  50  22   2]
 [34  323  27   1]
 [26  21   342  1]
 [ 4   1   3   390]]

Accuracy: 0.87731629393

Classification Report:
                           precision  recall   f1-score    support

comp.sys.ibm.pc.hardware       0.83      0.81      0.82       392
   comp.sys.mac.hardware       0.82      0.84      0.83       385
            misc.forsale       0.87      0.88      0.87       390
  soc.religion.christian       0.99      0.98      0.98       398

             avg / total       0.88      0.88      0.88      1565
```

2) NMF

```
Confusion matrix:
[[298  66  28   0]
 [61  294  29   1]
 [50  19  320   1]
 [3   3    13  379]]

Accuracy: 0.824920127796

Classification Report:
                           precision   recall   f1-score   support

comp.sys.ibm.pc.hardware       0.72      0.76      0.74       392
   comp.sys.mac.hardware       0.77      0.76      0.77       385
            misc.forsale       0.82      0.82      0.82       390
  soc.religion.christian       0.99      0.95      0.97       398

              avg / total      0.83      0.82      0.83      1565
```

Again, LSI beats NMF. And there isn't much difference between one vs one and one vs rest. However, SVM is better than Naïve Bayes in multiclassification. This could be that the four categories are not completely independent of each other because Naïve Bayes is best performed under independent cases. As we can see from the f1 score, topic comp.sys.ibm.pc.hardware and topic comp.sys.mac.hardware are kind of related to each other.