EE219 Project 1 Report

Classification Analysis of Texture Data

Winter 2018

Will Li (004356402)

Ruiyi Wu (304615036)

Introduction

In this project, we learned and practiced different methods of handling large text data with python and python libraries. The main goal is to extract important information from the text files and find a good method to classify the data into different classes. First, we fetched data from the "20 newsgroups" dataset and performed desired feature extraction and selection, which would be used for training and testing different classifiers. Then, we applied three different methods for binary classification: linear SVM, naive Bayes algorithm, and logistic regression classification. Finally, the multiclass classification was also performed through different algorithms: naive Bayes classification and both one-vs-one and one-vs-rest multiclass SVM classifications. Along the way, the ROC curve, confusion matrix, accuracy, precision and recall for each classifier were reported as well.

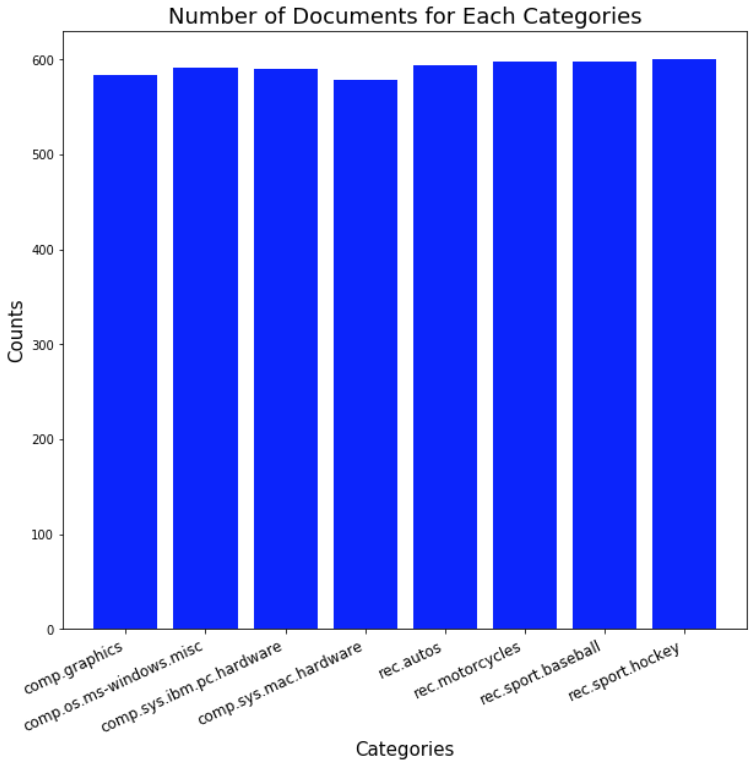
Dataset and Problem Statement:

a) Before feeding data into the training classifiers, we have to make sure that the two major classes, "Computer Technology" and "Rectiontional Activity", have balanced number of samples. Each major class has four subclasses/categories as follows:

Computer technology	Recreational activity
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

Table 1. Subclasses of 'Computer technology' and 'Recreational activity'

To check if the data set is evenly distributed, we wrote a python script to load the target categories and count the total number of articles in each categories. And then we used the library 'matplotlib' to plot a histogram of the number of training documents for each category and obtained the following graph.



The result in the graph agrees with the statement in the problem that the data set is already balanced, with each category has a number of data from 580 to 600 and averaged to nearly 590:

```
[584, 591, 590, 578, 594, 598, 597, 600]
```

Modeling Text Data and Feature Extraction

b) TFxIDF vector representations

In this part, we need to tokenize each document into words first. Then, by excluding the stop words and punctuations and using a stemmed version of words, we created a TFxIDF vector representation. The detailed procedure is as the following. The function ‘CountVectorizer’ from ‘sklearn’ library has an option to exclude stop words by setting it to ‘english’. As for punctuations, the best way we can think of is to replace all the punctuations in the documentation with space. Thus, we need to use the regular expression library ‘re’. As for stemmed version, we use a library from ‘nltk’ to create a stemmed version of words. Last, we use ‘TfidfTransformer’ to create the vector representations. There are many random numbers we think that we should

not take them into account, so we remove all the number instead. Although TAs provided an example code for doing these steps, we decide not to change our code since we are more confident with our original ideas where the numbers are mostly random and unnecessary.

Report the feature:

```
For min_df = 2: (4732, 18713)
For min_df = 5: (4732, 8030)
```

As shown, for min_df = 2, the number of terms extracted is 18713; and for min_df = 5, the number of terms extracted is 8030

c) The goal of this section is to find the ten most significant terms in each of the following classes concerning TFxICF measure, and this part is similar to the previous part. The only difference is that we need to classify each article to the corresponding class and create a matrix with 20 rows where each row corresponds to a class. And then we applied the 'tficf' function to calculate the value.

Now report the words we found (with df_min = 2):

comp.sys.ibm.pc.hardware:

Rank	words	value
1	scsi	0.432286048349
2	drive	0.293867386907
3	ide	0.229096307748
4	use	0.197409386311
5	line	0.190819088134
6	mb	0.185714829191
7	subject	0.183330112933
8	organ	0.175541578723

9	card	0.143189205853
10	control	0.120722280248

com.sys.mac.hardware:

Rank	**words**	**value**
1	line	0.250988400994
2	mac	0.232328584229
3	subject	0.228276260414
4	organ	0.214033053609
5	quadra	0.1996298907
6	simm	0.184107729093
7	use	0.183621882323
8	appl	0.161679644812
9	scsi	0.156238270422
10	centri	0.147213103757

misc.forsale:

Rank	**words**	**value**
1	line	0.311300516762
2	subject	0.297916577347
3	sale	0.292352595224
4	organ	0.286515443772
5	univers	0.165068586118

6	new	0.163085780278
7	use	0.140779214587
8	offer	0.138300707288
9	nntppostinghost	0.130369483931
10	includ	0.125412469333

soc.religion.christian:

Rank	**words**	**value**
1	god	0.343657781593
2	christian	0.263724413989
3	jesu	0.200741305877
4	church	0.164787639152
5	subject	0.160078912482
6	peopl	0.152633381669
7	line	0.147979924911
8	say	0.146583887883
9	christ	0.145761263015
10	believ	0.136113610177

From four tables above we can see that most of the top rated words are related to the topic. For example, we see “sale” and “off” in the subclass of forsale. This means that our method can indeed filter the relevant word for us. We also see that there are some words being rated top in more than one subclass, although this is normal behavior, whether this will confuse classification worth discussing later on.

Feature Selection:

d) In this part, we compare two methods of dimensionality reduction, naming latent semantic index (LSI) and Non-Negative Matrix Factorization (NMF), and the comparison between these two methods will be discussed later. The code to perform the reduction are included in the report.

```
# using LSI
from sklearn.decomposition import TruncatedSVD

SVD = TruncatedSVD(n_components=50, random_state=42, algorithm="arpack")
reduced_train_tfidf = SVD.fit_transform(train_tfidf)
print "TFxIDF matrix after LSI: " + str (reduced_train_tfidf.shape)
```

```
TFxIDF matrix after LSI: (4732, 50)
```

```
# using NMF
from sklearn.decomposition import NMF

trainNMF = NMF(n_components=50, init='random', random_state=42)
NMFed_train = trainNMF.fit_transform(train_tfidf)
print "TFxIDF matrix after NMF: " + str (NMFed_train.shape)
```

```
TFxIDF matrix after NMF: (4732, 50)
```

Learning Algorithms

e) In this part, we performed SVM classification for hard and soft margins using both LSI and NMF. Hard margin SVM was achieved with a large enough gamma, the penalty parameter of the error term. In this case, it would be 1000. On the other hand, a soft margin SVM has a small gamma and we set it to 0.001 in our simulation. And then we plotted the receiver operating characteristic curve (ROC) in order to see the significance of true positive rate (TPR) and false positive rate (FPR). Also, a confusion matrix was presented and accuracy was calculated.
The result are as follows:

SVM Classification

- Hard margin:

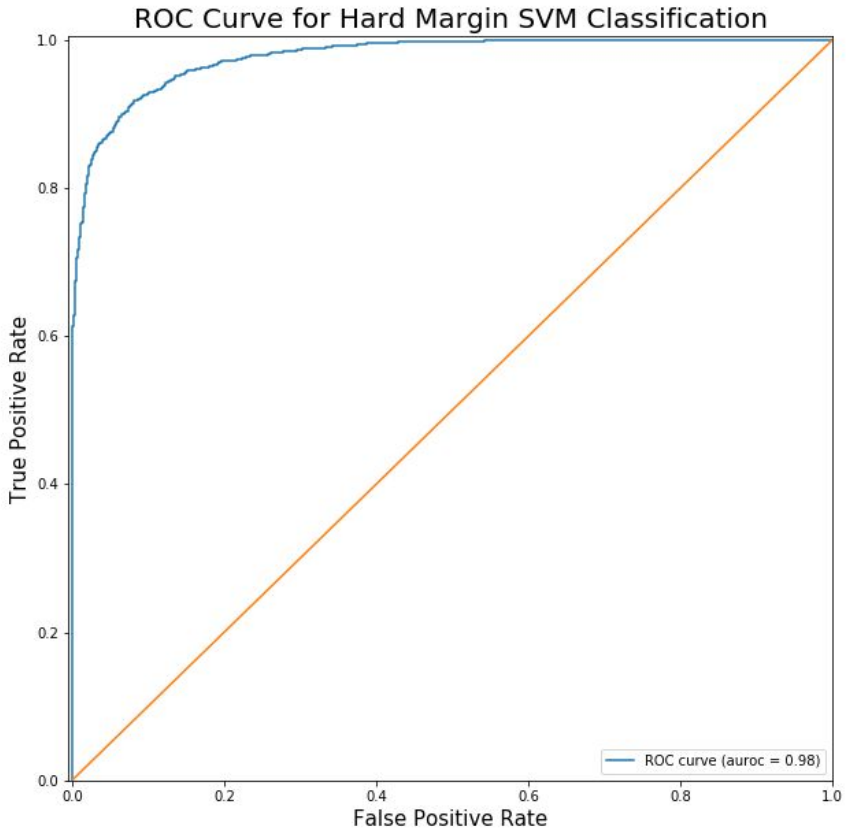
For LSI min_df = 2:

```
Hard Margin SVM Accuracy: 0.916507936508
--------------------------------------------------
Classification report:
                      precision    recall  f1-score   support

  Computer technology       0.92      0.91      0.92      1590
Recreational activity       0.91      0.92      0.92      1560

          avg / total       0.92      0.92      0.92      3150

--------------------------------------------------
Confusion Matrix:
[[1453  137]
 [ 126 1434]]
--------------------------------------------------
```



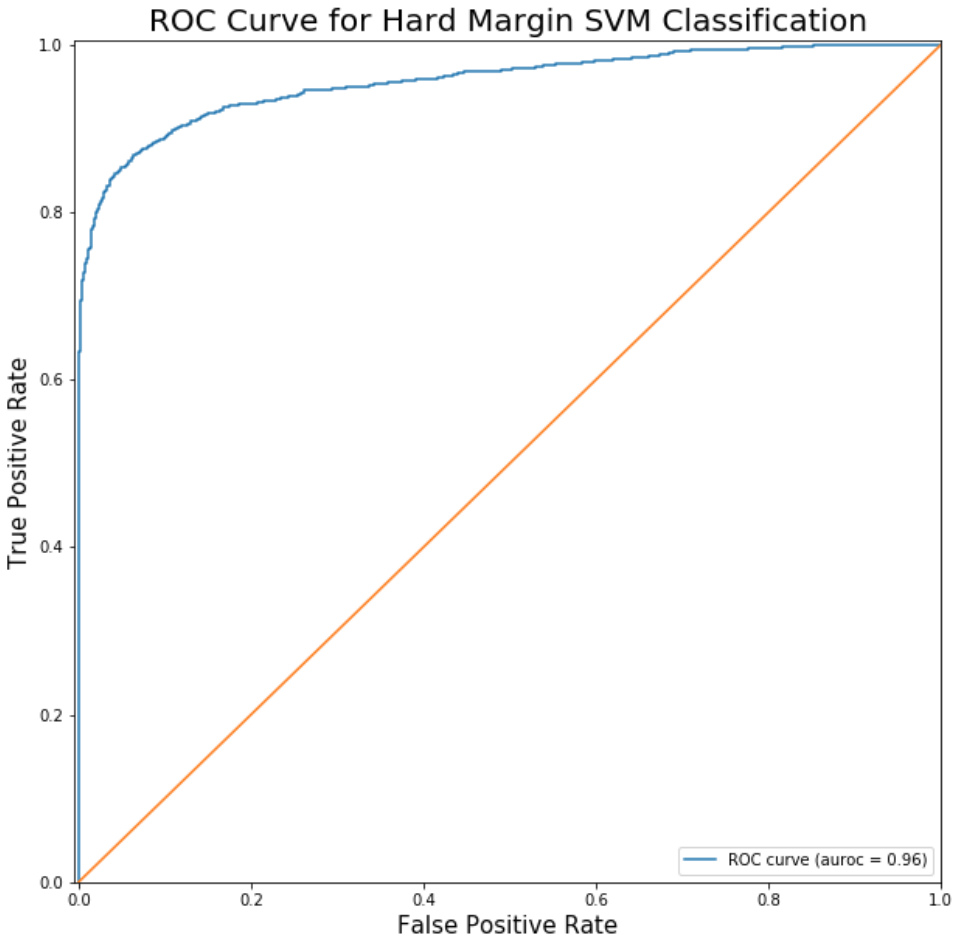
For LSI min_df = 5:

```
Hard Margin SVM Accuracy: 0.89619047619
--------------------------------------------------
Classification report:
                       precision    recall  f1-score   support

  Computer technology       0.89      0.91      0.90      1590
Recreational activity       0.90      0.89      0.89      1560

          avg / total       0.90      0.90      0.90      3150

--------------------------------------------------
Confusion Matrix:
[[1441  149]
 [ 178 1382]]
--------------------------------------------------
```



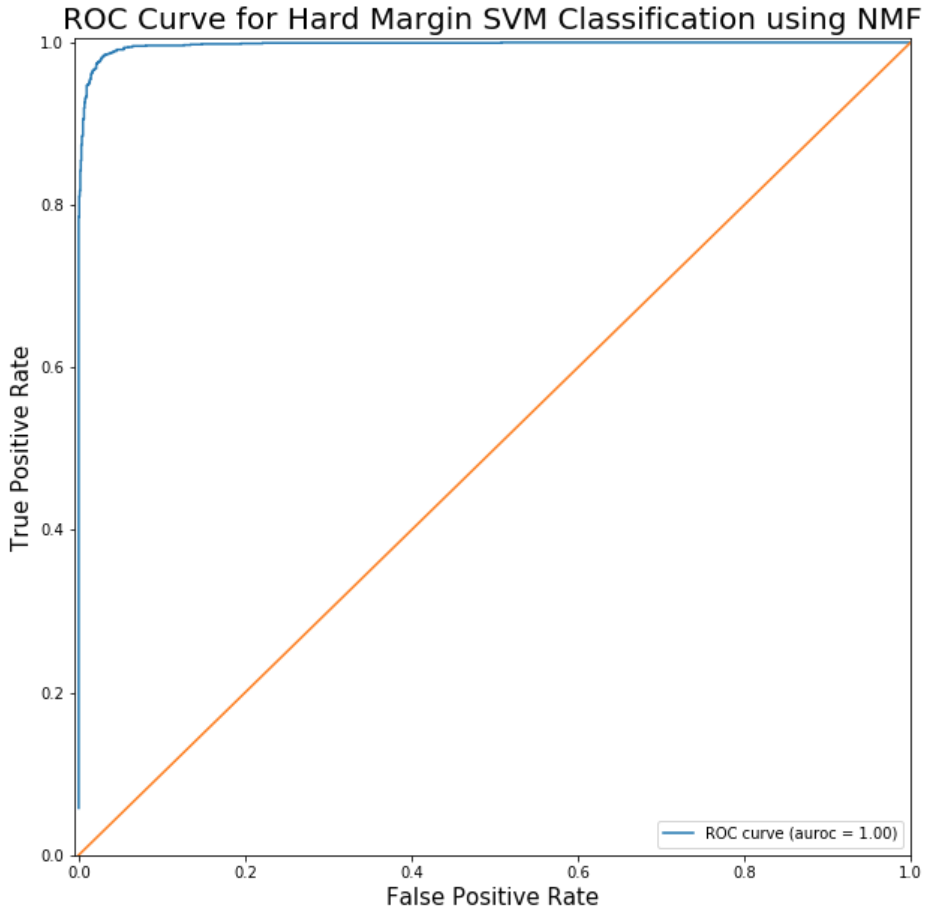
For NMF min_df = 2:

```
Hard Margin SVM Accuracy (NMF): 0.973968253968
-------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.99      0.96      0.97      1560
Recreational activity       0.96      0.99      0.97      1590

          avg / total       0.97      0.97      0.97      3150

-------------------------------------------------
Confusion Matrix:
[[1498   62]
 [  20 1570]]
-------------------------------------------------
```



- Soft margin:

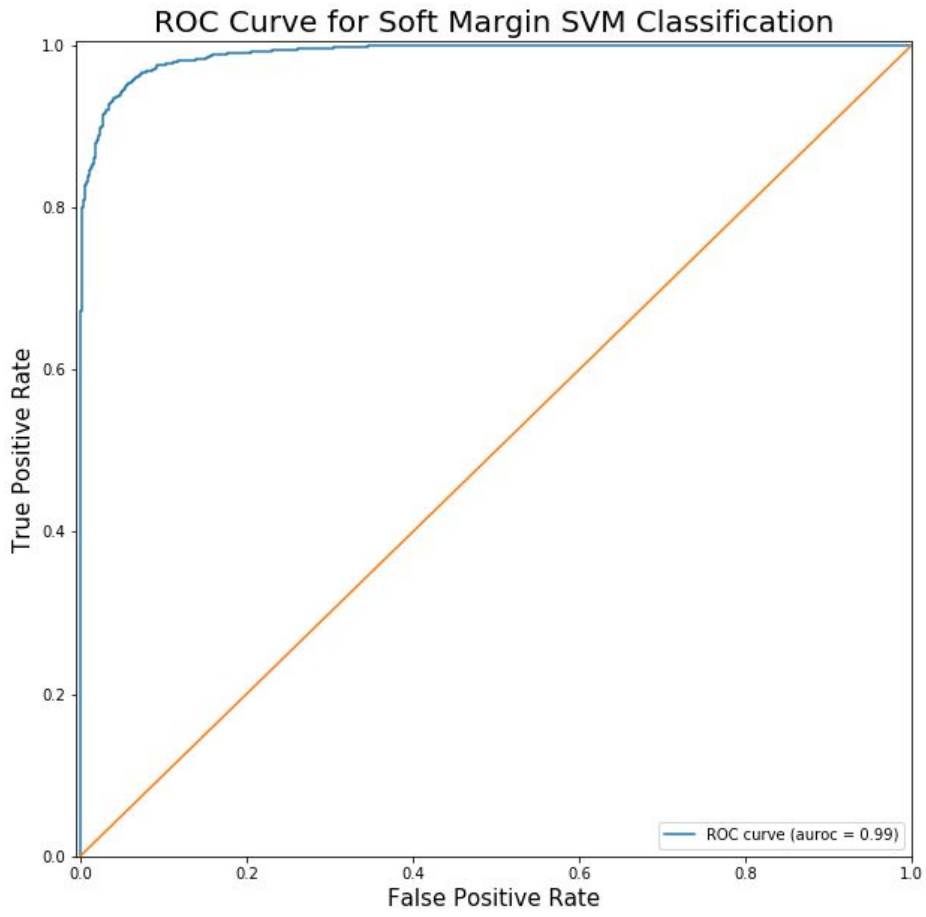
For LSI min_df = 2:

```
Soft Margin SVM Accuracy: 0.927936507937
--------------------------------------------------
Classification report:
                       precision    recall  f1-score   support

  Computer technology       0.89      0.98      0.93      1590
Recreational activity       0.98      0.87      0.92      1560

          avg / total       0.93      0.93      0.93      3150

--------------------------------------------------
Confusion Matrix:
[[1562   28]
 [ 199 1361]]
--------------------------------------------------
```



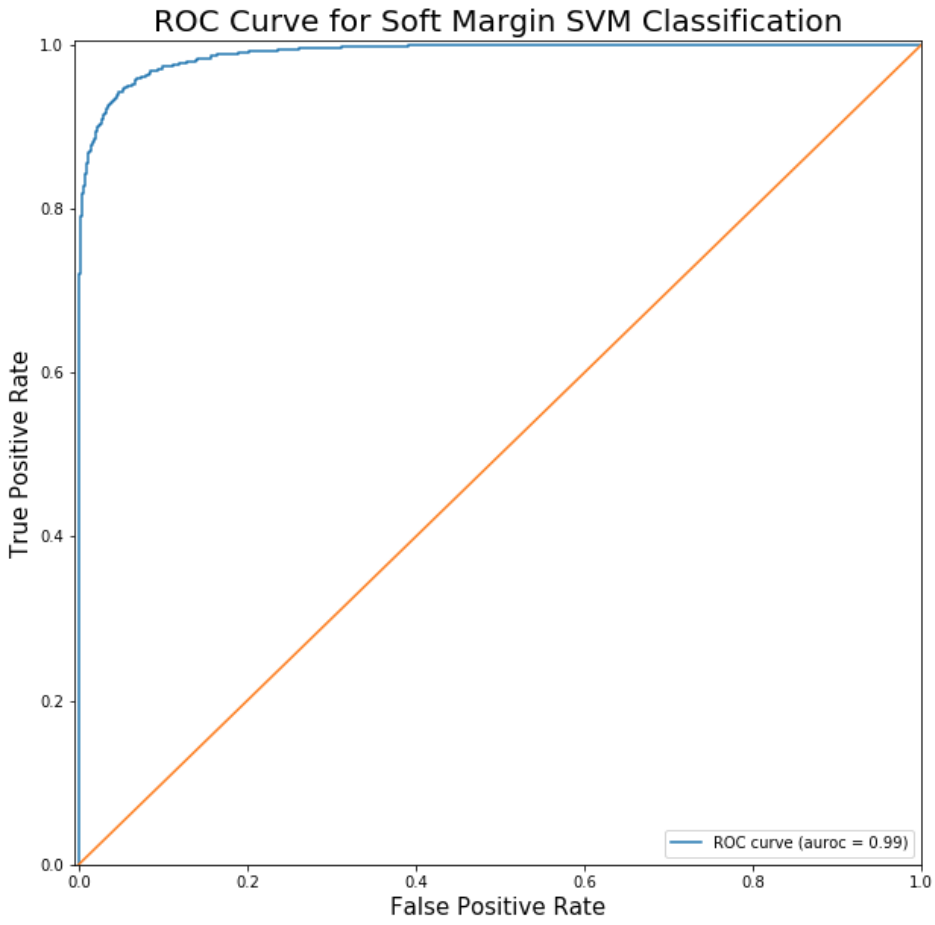
For LSI min_df = 5:

```
Soft Margin SVM Accuracy: 0.933333333333
--------------------------------------------------
Classification report:
                       precision    recall  f1-score   support

  Computer technology       0.90      0.98      0.94      1590
Recreational activity       0.98      0.89      0.93      1560

          avg / total       0.94      0.93      0.93      3150

--------------------------------------------------
Confusion Matrix:
[[1558   32]
 [ 178 1382]]
--------------------------------------------------
```



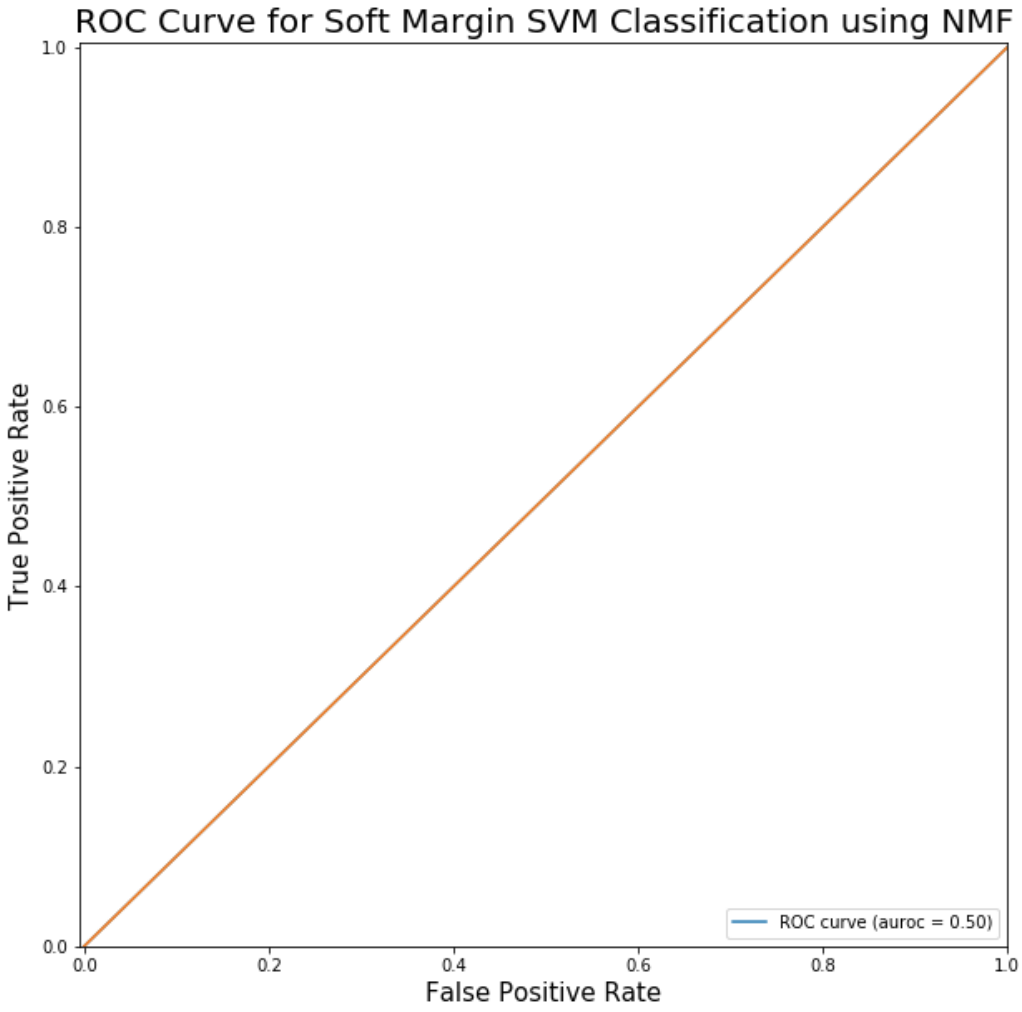
For NMF min_df = 2:

```
Soft Margin SVM Accuracy (NMF): 0.504761904762
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.00      0.00      0.00      1560
Recreational activity       0.50      1.00      0.67      1590

          avg / total       0.25      0.50      0.34      3150

--------------------------------------------------
Confusion Matrix:
[[    0 1560]
 [    0 1590]]
--------------------------------------------------
```



Analysis of the result:

In this part, we perform three different settings under hard and soft SVM, and the accuracy are shown in the following table:

Setting	Hard	Soft
LSI & min_df = 2	0.9165	0.9279
LSI & min_df = 5	0.8962	0.9333
NMF & min_df = 2	0.9740	0.5048

As shown in the table, soft SVM has a better performance than hard SVM under LSI in general. This is because a hard SVM is easy to overfit during the training process for LSI reduction. A reasonable setting of gamma would make the training more suitable for a practical case. In next part, we will use 5-fold cross-validation to find the best setting for gamma. The setting of min_df has a slightly effect on the accuracy. With min_df = 5, hard SVM perform even worse while the soft SVM perform better. On the other hand, if we use NMF method of reduction, a hard SVM outperform all the other method while a soft SVM perform like a random machine. This is because if we use NMF method of reduction, a soft SVM will allow too many training errors during the training process, and it lost the ability to classify.

f) 5-fold cross-validation

In this part, we used 5-fold cross-validation to find the best value of gamma. The python script runs automatically and uses the function 'cross_val_score' to calculate the performance of the classifier. Find the value of gamma which gives the best accuracy, and the results are as follows:

For LSI min_df = 2:

```
Accuracy for 0.001 : 0.94158730
Accuracy for 0.01 : 0.97047619
Accuracy for 0.1 : 0.97746032
Accuracy for 1 : 0.98444444
Accuracy for 10 : 0.98412698
Accuracy for 100 : 0.98349206
Accuracy for 1000 : 0.98095238
Best value for gamma is 1 with accuracy of 0.98444444
```

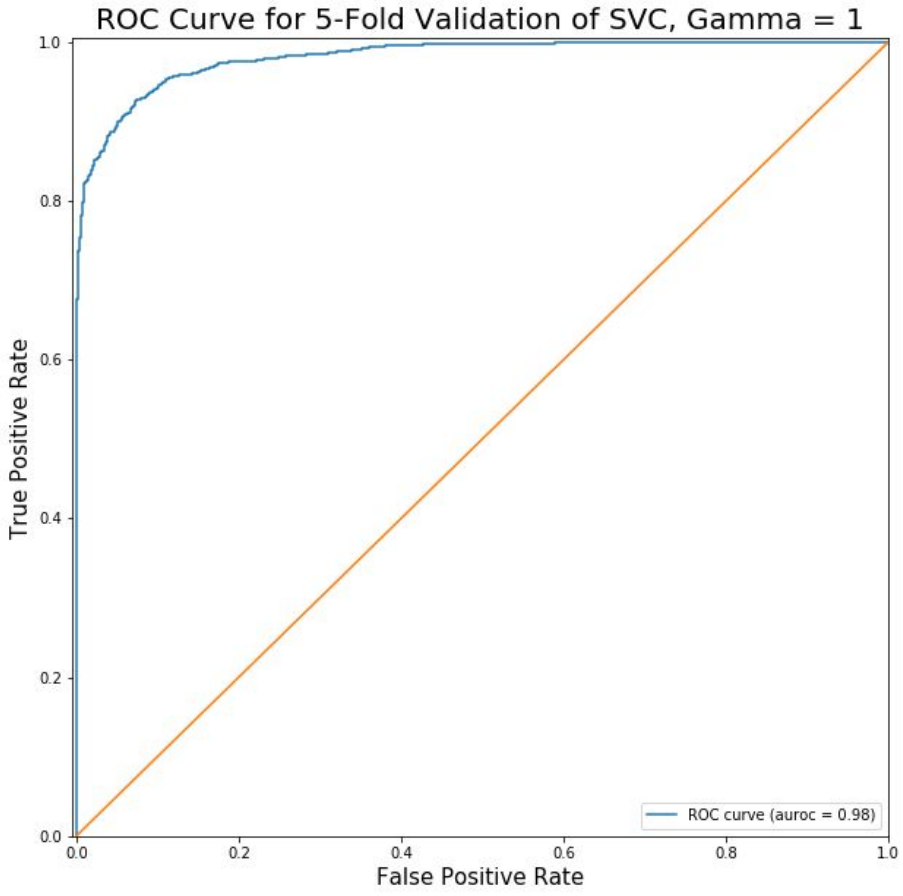
Apply the best value of gamma as 1:

```
5-Fold Validation of SVC with Gamma = 1 has Accuracy : 0.924126984127
---------------------------------------------------
Classification report:
                      precision    recall  f1-score   support

  Computer technology       0.92      0.93      0.93      1590
Recreational activity       0.93      0.92      0.92      1560

          avg / total       0.92      0.92      0.92      3150

---------------------------------------------------
Confusion Matrix:
[[1483  107]
 [ 132 1428]]
---------------------------------------------------
```



For LSI min_df = 5:

```
Accuracy for 0.001 : 0.95111111
Accuracy for 0.01 : 0.97142857
Accuracy for 0.1 : 0.97968254
Accuracy for 1 : 0.98539683
Accuracy for 10 : 0.98380952
Accuracy for 100 : 0.98317460
Accuracy for 1000 : 0.98158730
Best value for gamma is 1 with accuracy of 0.98539683
```

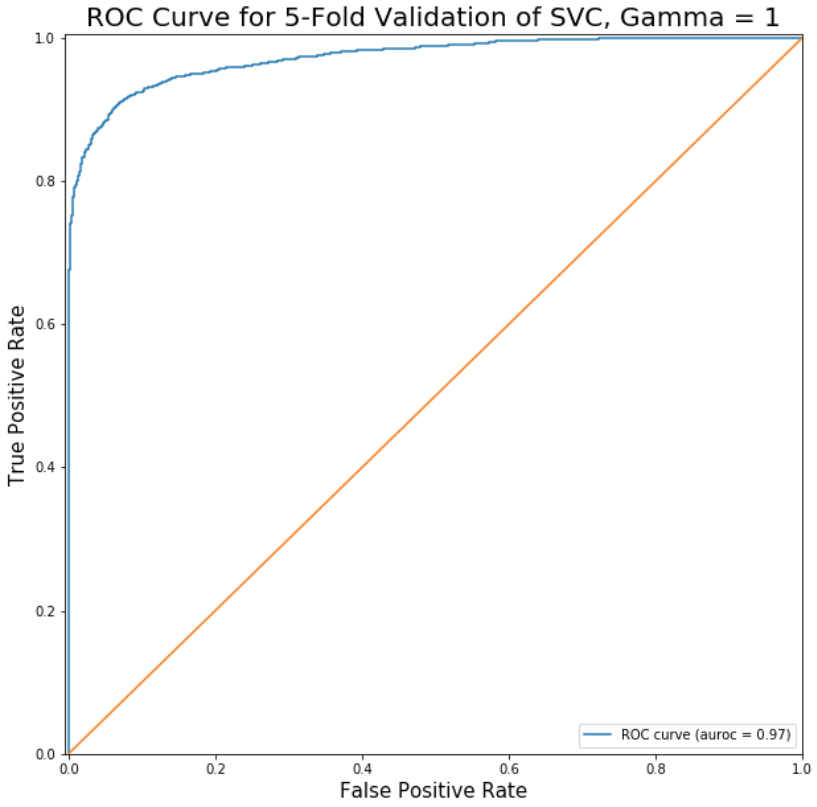
Again apply the best value of gamma as 1:

```
5-Fold Validation of SVC with Gamma = 1 has Accuracy : 0.92
---------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.90      0.94      0.92      1590
Recreational activity       0.94      0.90      0.92      1560

          avg / total       0.92      0.92      0.92      3150

---------------------------------------------------
Confusion Matrix:
[[1495   95]
 [ 157 1403]]
---------------------------------------------------
```



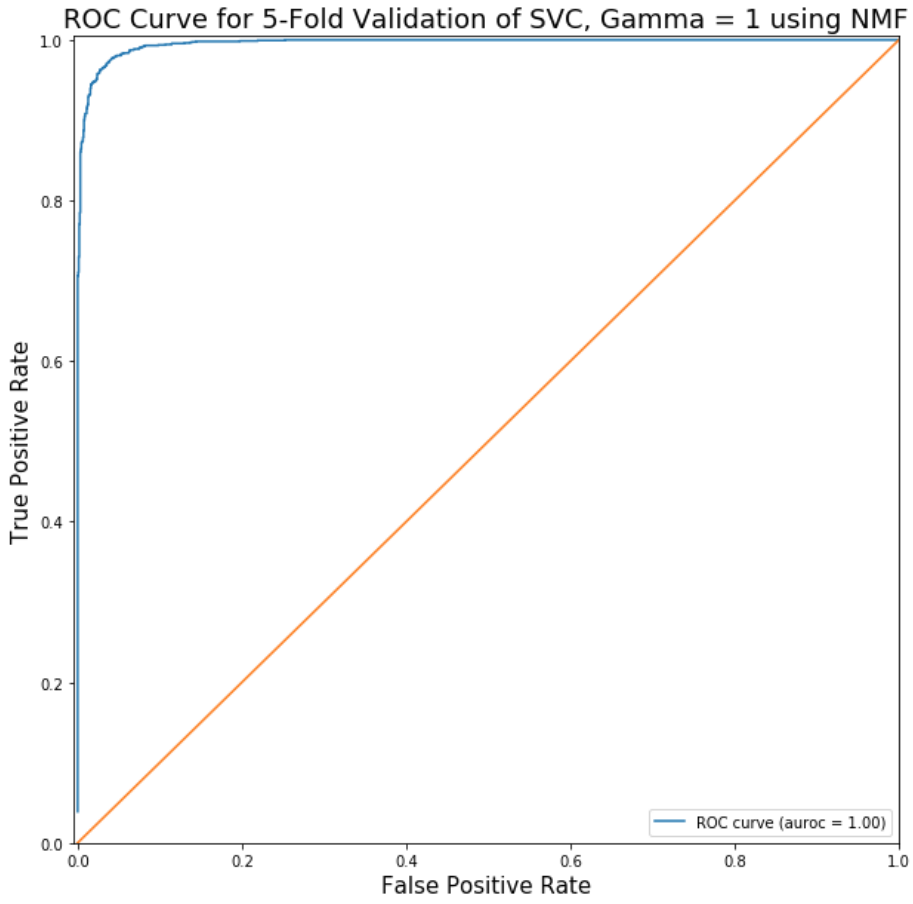
For NMF min_df = 2:

```
5-Fold Validation of SVC Accuracy with Gamma = 1 using NMF: 0.959365079365
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.99      0.93      0.96      1560
Recreational activity       0.93      0.99      0.96      1590

          avg / total       0.96      0.96      0.96      3150

--------------------------------------------------
Confusion Matrix:
[[1450  110]
 [  18 1572]]
--------------------------------------------------
```



Analysis of the result:

As shown above, for all the setting, the best gamma is 1. By setting gamma to 1, we get a plot with a maximal area under ROC curve. So setting gamma to 1 is the best under SVM classifier. This is what we expected since we do not want the classifier to overfit or too loose during the training process.

g) Naive Bayes Algorithm

We applied Naive Bayes algorithm for the same task in this section. The Naive Bayes algorithm would estimate the maximum likelihood, assuming the features are statistically independent with given class. Since the input for MultinomialNB classifier had to be non-negative, we first normalized the data and mapped them to a range from 0 to 1, or we could have use GaussianNB classifier. The classification was used for both LSI and NMF reduced matrices, and ROC curves, confusion matrix, accuracy, precision, and recall are as follows:

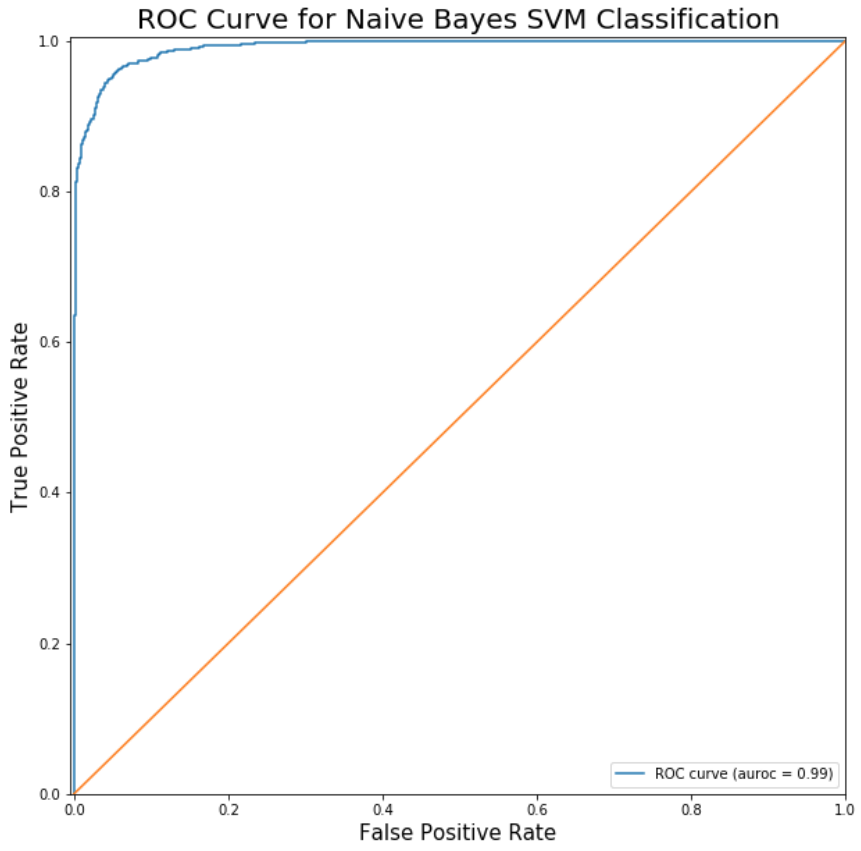
For LSI min_df = 2:

```
Naive Bayes SVM Accuracy: 0.83746031746
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.76      1.00      0.86      1590
Recreational activity       1.00      0.67      0.80      1560

          avg / total       0.88      0.84      0.83      3150

--------------------------------------------------
Confusion Matrix:
[[1588    2]
 [ 510 1050]]
--------------------------------------------------
```



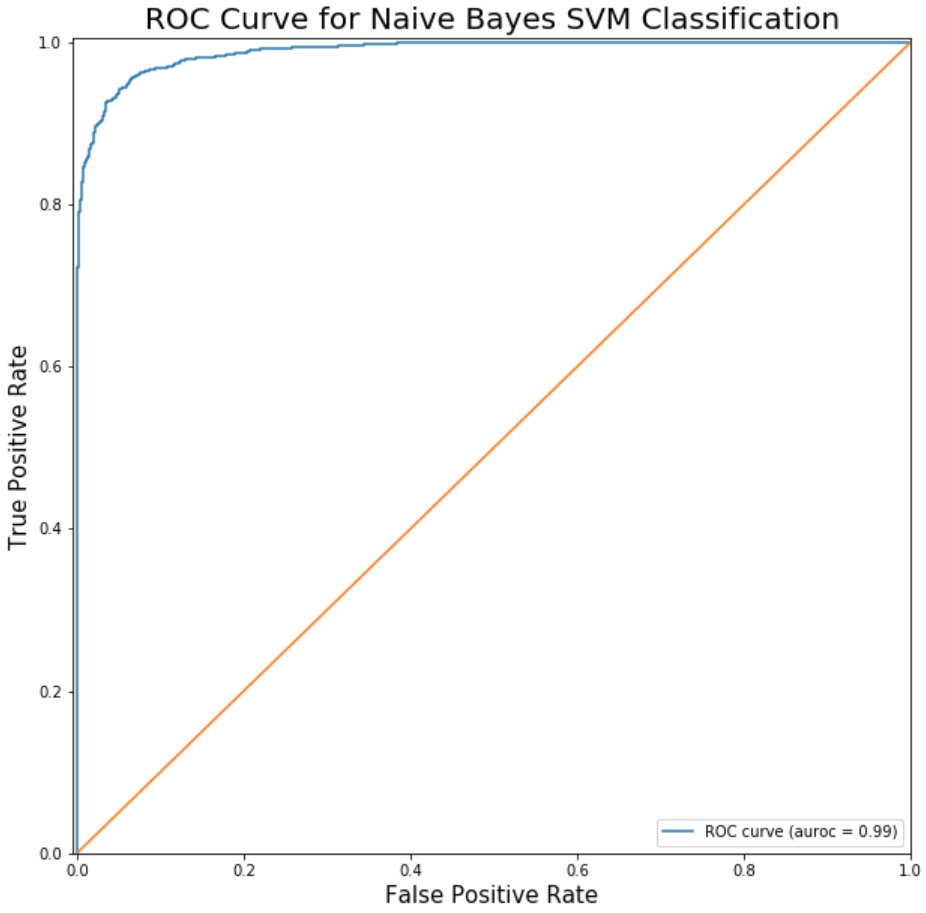
For LSI min_df = 5:

```
Naive Bayes SVM Accuracy: 0.878095238095
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.81      1.00      0.89      1590
Recreational activity       1.00      0.76      0.86      1560

          avg / total       0.90      0.88      0.88      3150

--------------------------------------------------
Confusion Matrix:
[[1587    3]
 [ 381 1179]]
--------------------------------------------------
```



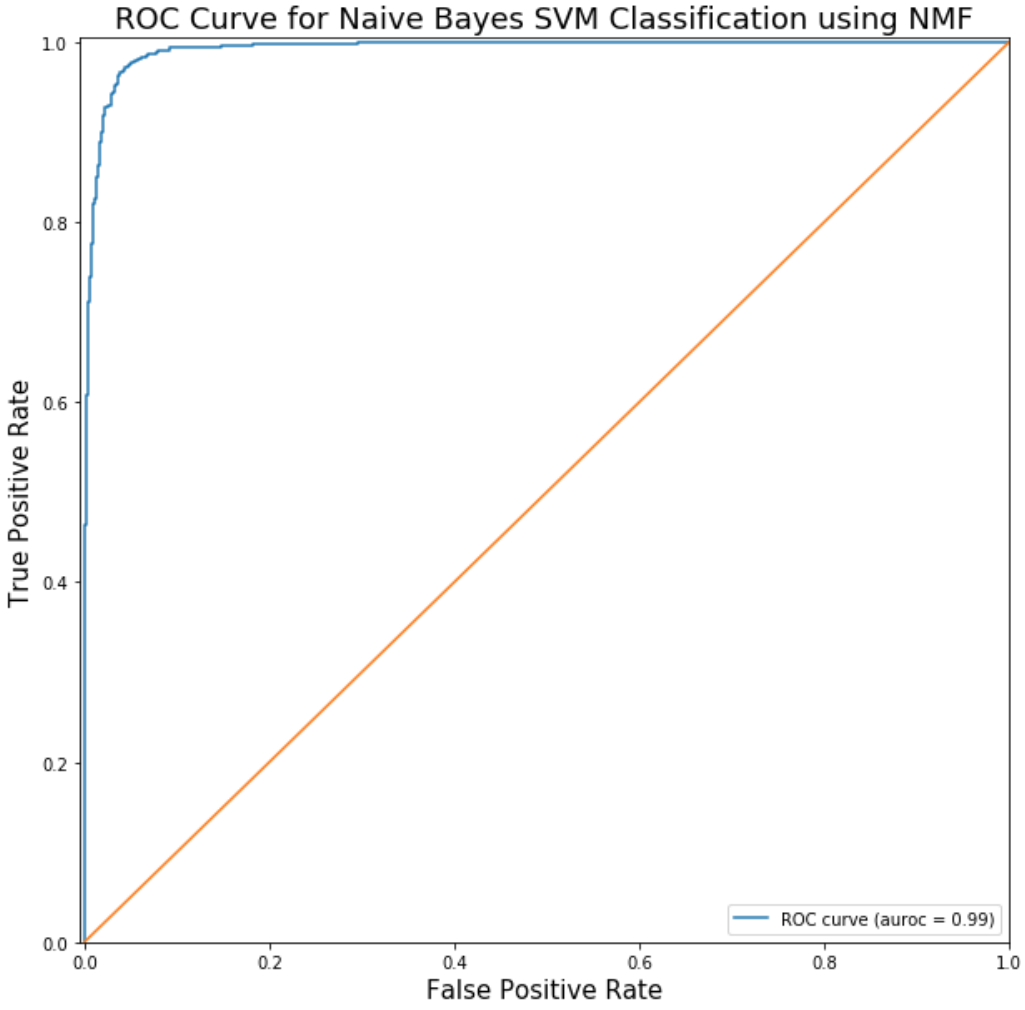
For NMF min_df = 2:

```
Naive Bayes SVM Accuracy using NMF: 0.953968253968
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.99      0.92      0.95      1560
Recreational activity       0.92      0.99      0.96      1590

          avg / total       0.96      0.95      0.95      3150

--------------------------------------------------
Confusion Matrix:
[[1430  130]
 [  15 1575]]
--------------------------------------------------
```



Analysis of the result:

The results of accuracy are shown in the following table:

Setting	Accuracy
LSI & min_df = 2	0.8375
LSI & min_df = 5	0.8781
NMF & min_df = 2	0.9540

From the results above, we could see that for naive Bayes algorithm with min_df = 2, NMF provided a better accuracy of 0.9540 than that of LSI as 0.8375. Again, mind_df = 5 has a better performance than min_df = 2 under LSI method of reduction.

h) Logistic Regression

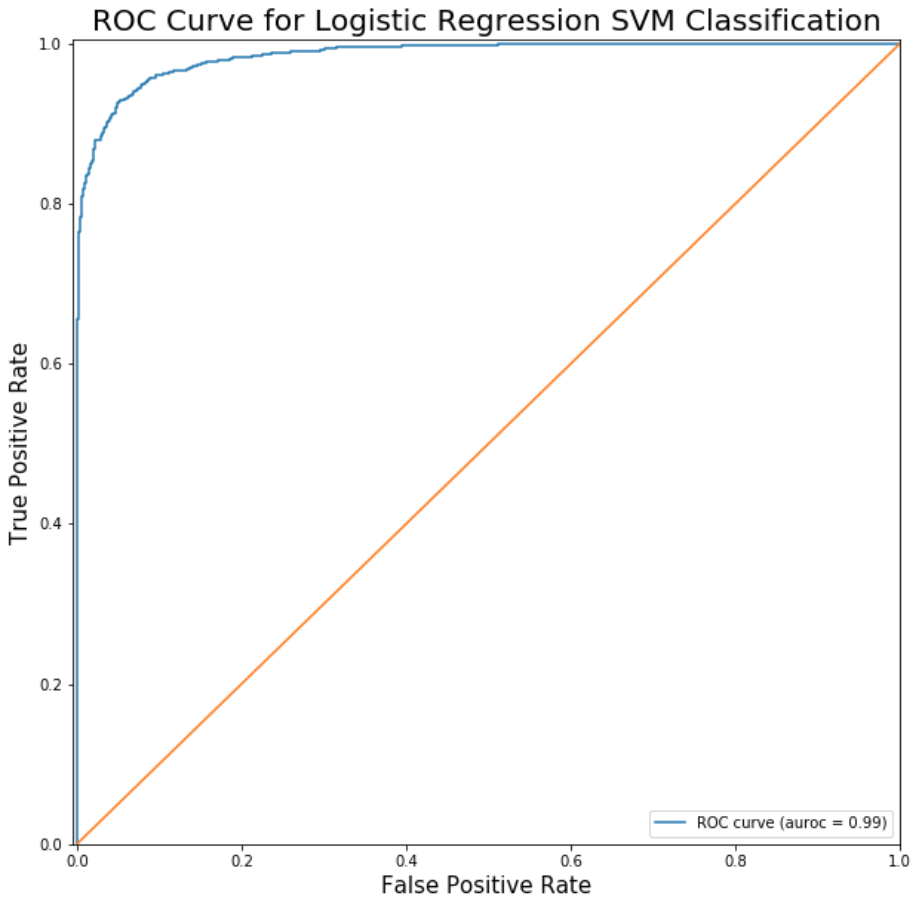
In this section and section (i), we applied the logistic regression classifier, first without regularization and then with L1-norm and L2-norm regularizations, respectively. Again, the classifier was applied for both LSI and NMF with min_df = 2 and min_df = 5, and the ROC curve, confusion matrix, accuracy, precision and recall for each are shown below:

```
For LSI min_df = 2:
        Logistic Regression SVC Accuracy: 0.935238095238
        -------------------------------------------------
        Classification report:
                              precision    recall  f1-score   support

          Computer technology       0.92      0.95      0.94      1590
        Recreational activity       0.95      0.92      0.93      1560

                   avg / total       0.94      0.94      0.94      3150

        -------------------------------------------------
        Confusion Matrix:
        [[1515   75]
         [ 129 1431]]
        -------------------------------------------------
```



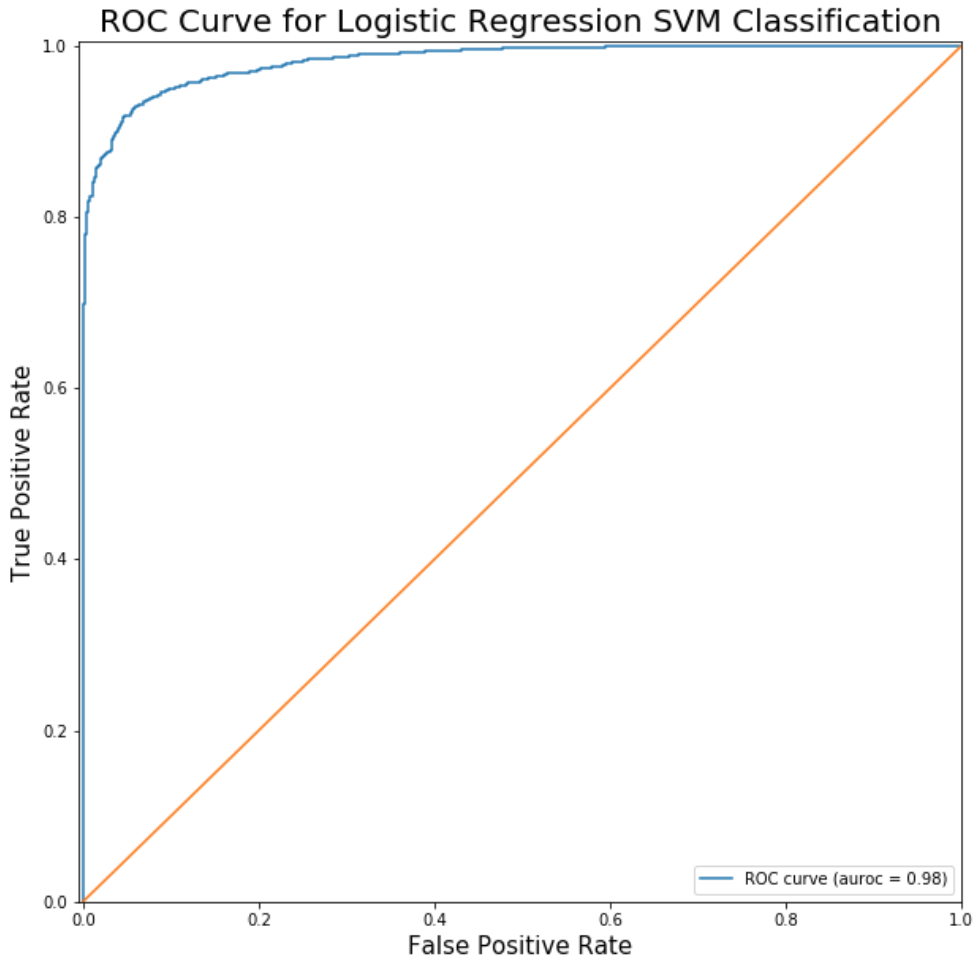
For LSI min_df = 5:

```
Logistic Regression SVC Accuracy: 0.931428571429
-------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.91      0.96      0.93      1590
Recreational activity       0.96      0.90      0.93      1560

          avg / total       0.93      0.93      0.93      3150

-------------------------------------------------
Confusion Matrix:
[[1525   65]
 [ 151 1409]]
-------------------------------------------------
```



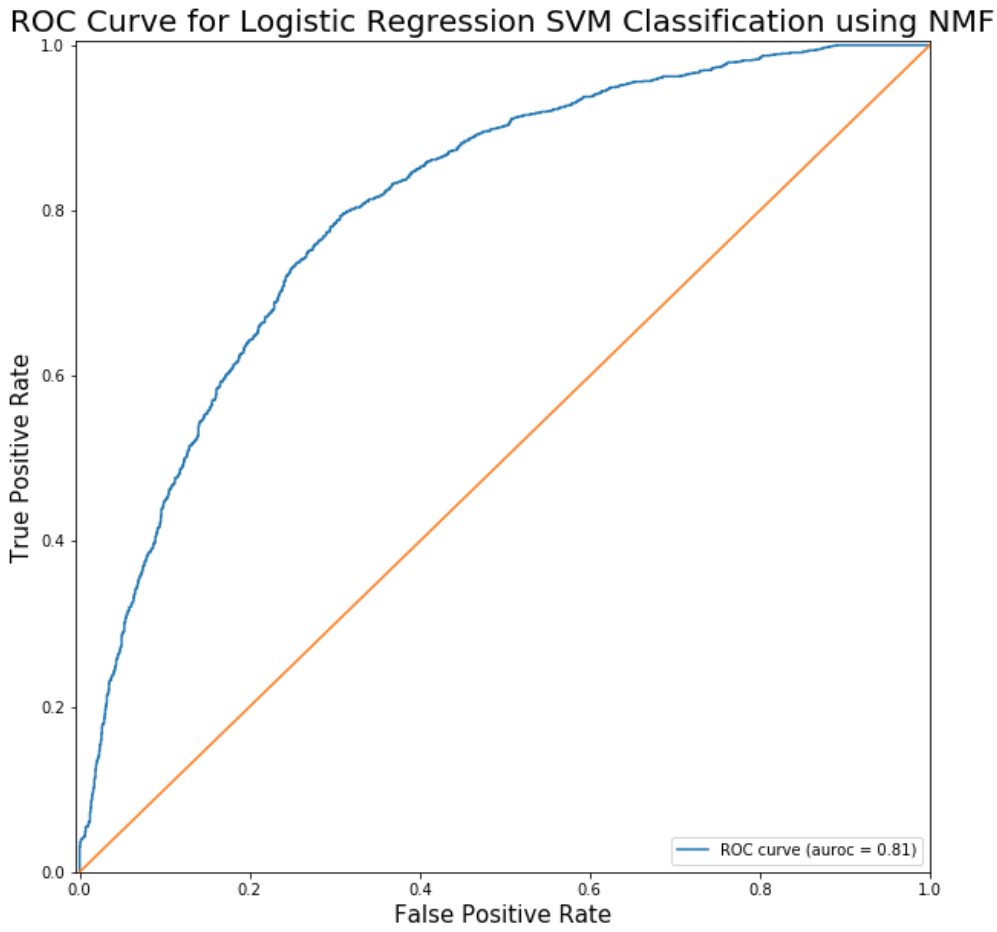
For NMF min_df = 2:

```
Logistic Regression SVC Accuracy using NMF: 0.733015873016
-------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.78      0.65      0.71      1560
Recreational activity       0.70      0.82      0.76      1590

          avg / total       0.74      0.73      0.73      3150

-------------------------------------------------
Confusion Matrix:
[[1009  551]
 [ 290 1300]]
-------------------------------------------------
```



Analysis of the result:

The results of accuracy are show in the following table:

Setting	**Accuracy**
LSI & min_df = 2	0.9352
LSI & min_df = 5	0.9314
NMF & min_df = 2	0.7330

For unregularized logistic regression classification, LSI had better performance than NMF, and LSI with min_df = 2 has a better accuracy than that of min_df =5.

i) Logistic Regression with Regularization

- L-1 Form

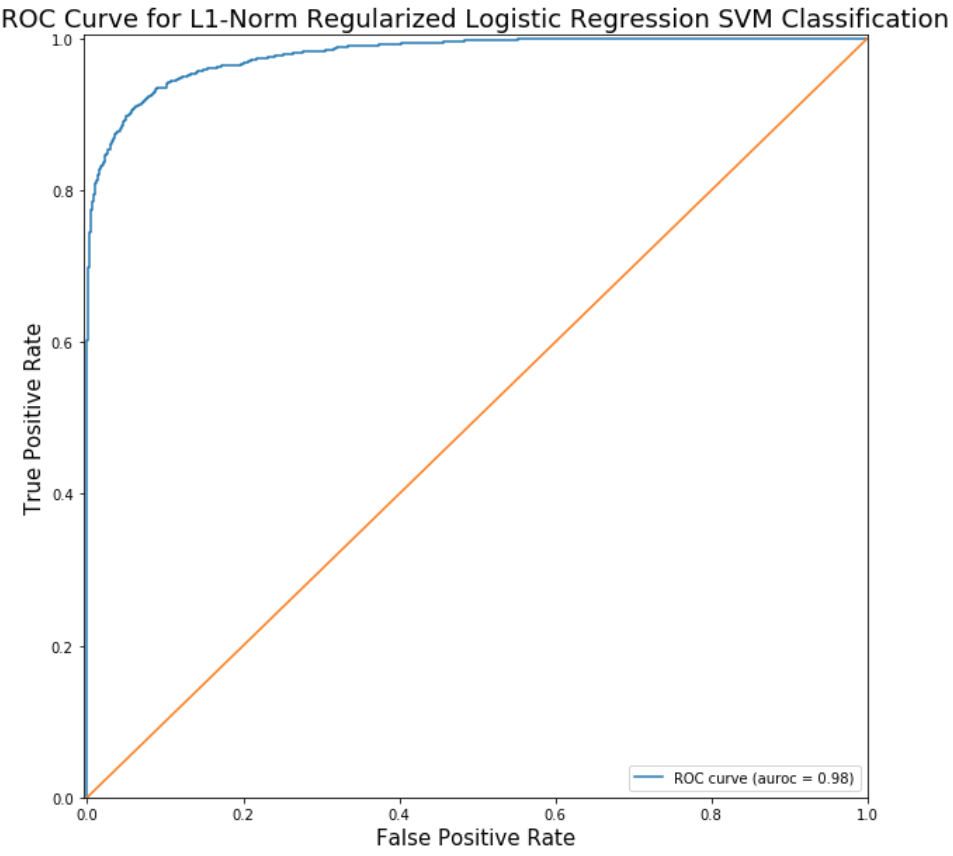
For LSI min_df = 2:

```
L1-Norm Regularized Logistic Regression SVC Accuracy: 0.923174603175
-------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.91      0.94      0.92      1590
Recreational activity       0.93      0.91      0.92      1560

          avg / total       0.92      0.92      0.92      3150

-------------------------------------------------
Confusion Matrix:
[[1487  103]
 [ 139 1421]]
-------------------------------------------------
```



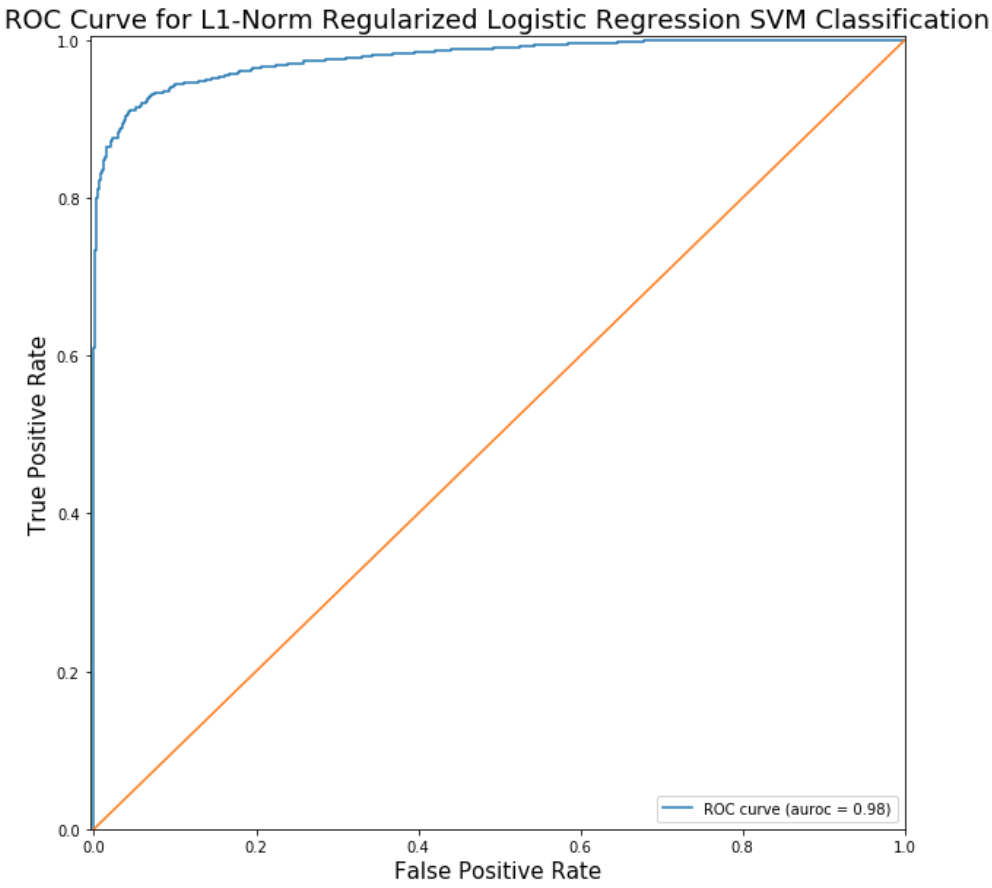
For LSI min_df = 5:

```
L1-Norm Regularized Logistic Regression SVC Accuracy: 0.933015873016
-----------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.91      0.96      0.94      1590
Recreational activity       0.96      0.91      0.93      1560

          avg / total       0.93      0.93      0.93      3150

-----------------------------------------------------
Confusion Matrix:
[[1526   64]
 [ 147 1413]]
-----------------------------------------------------
```



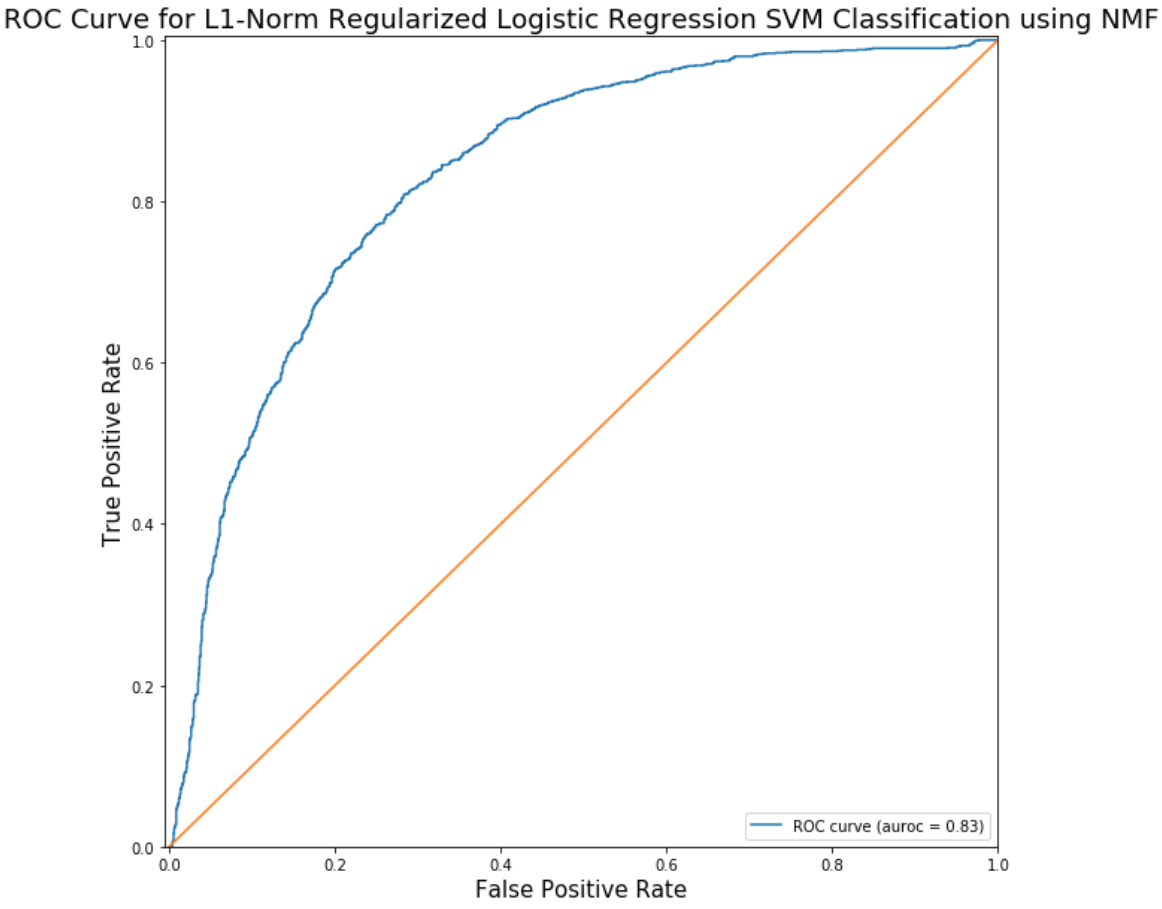
For NMF min_df = 2:

```
L1-Norm Regularized Logistic Regression SVC Accuracy using NMF: 0.761904761905
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.79      0.71      0.75      1560
Recreational activity       0.74      0.81      0.77      1590

          avg / total       0.76      0.76      0.76      3150

--------------------------------------------------
Confusion Matrix:
[[1114  446]
 [ 304 1286]]
--------------------------------------------------
```



- L-2 Form

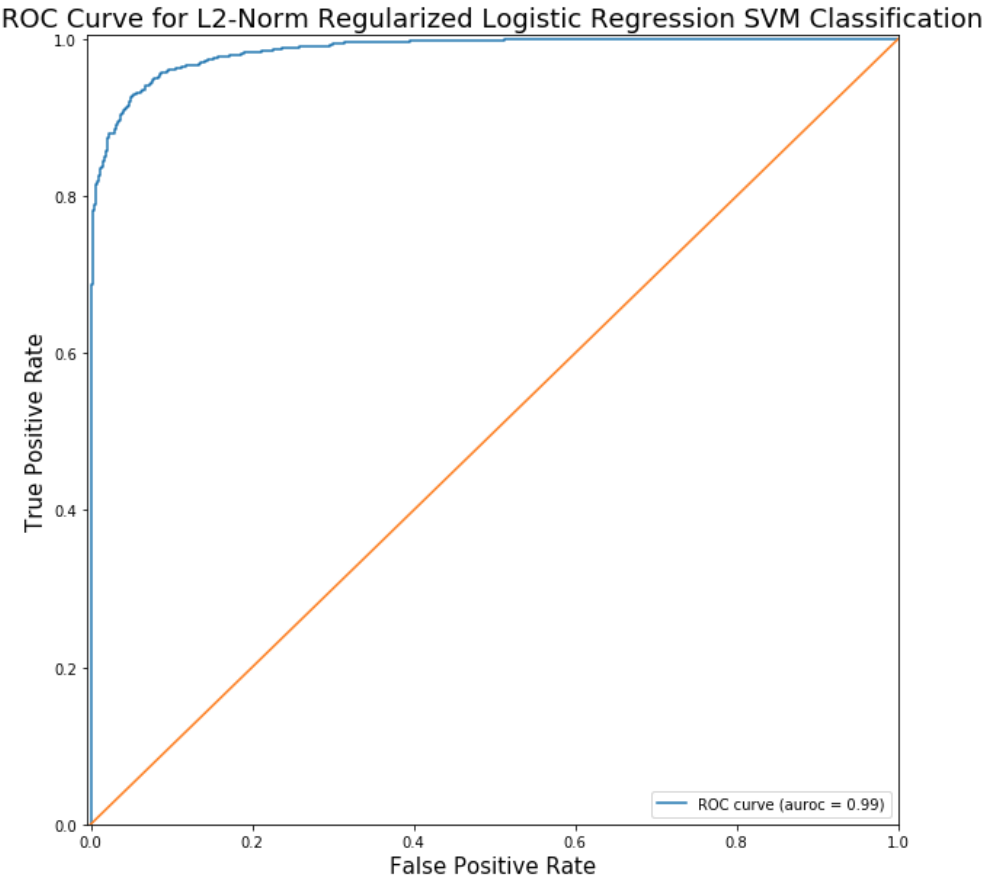
For LSI min_df = 2:

```
L2-Norm Regularized Logistic Regression SVC Accuracy: 0.935238095238
-------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.91      0.94      0.92      1590
Recreational activity       0.93      0.91      0.92      1560

          avg / total       0.92      0.92      0.92      3150

-------------------------------------------------
Confusion Matrix:
[[1515   75]
 [ 129 1431]]
-------------------------------------------------
```



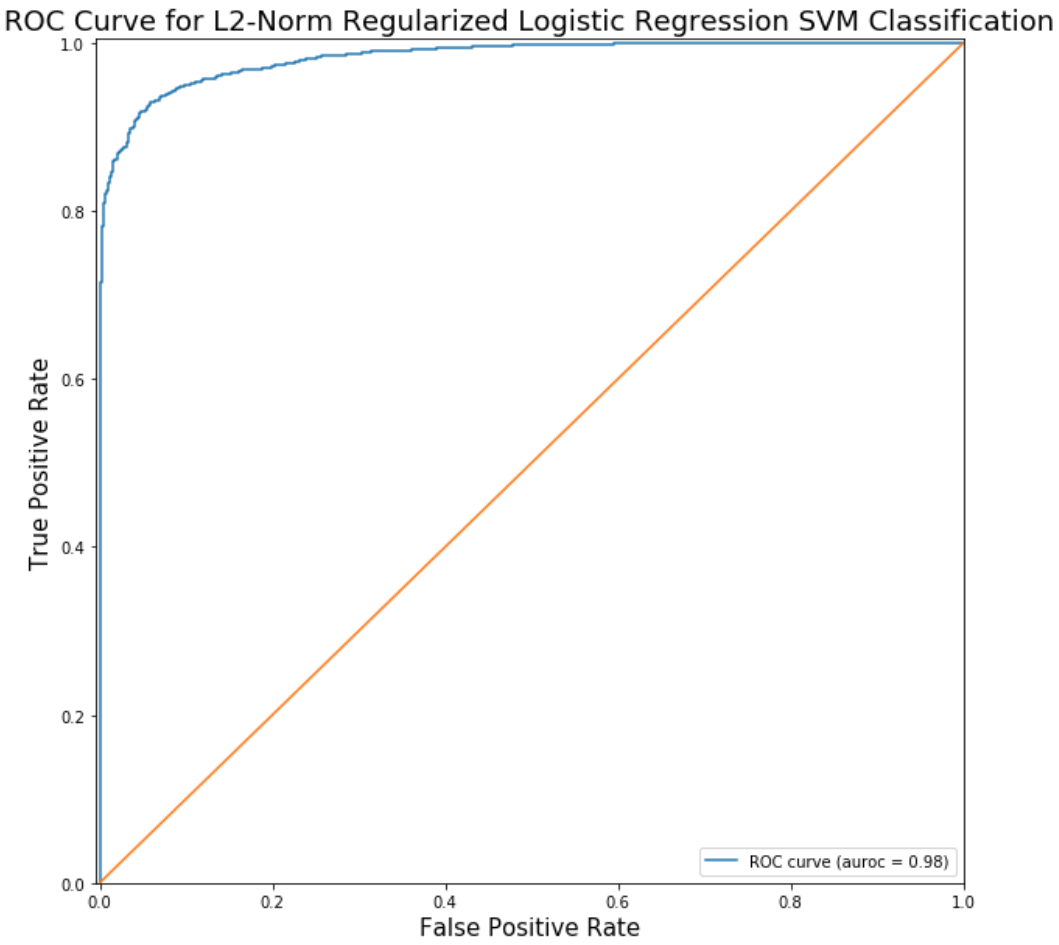
For LSI min_df = 5:

```
L2-Norm Regularized Logistic Regression SVC Accuracy: 0.931428571429
-----------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.91      0.96      0.94      1590
Recreational activity       0.96      0.91      0.93      1560

          avg / total       0.93      0.93      0.93      3150

-----------------------------------------------------
Confusion Matrix:
[[1525   65]
 [ 151 1409]]
-----------------------------------------------------
```



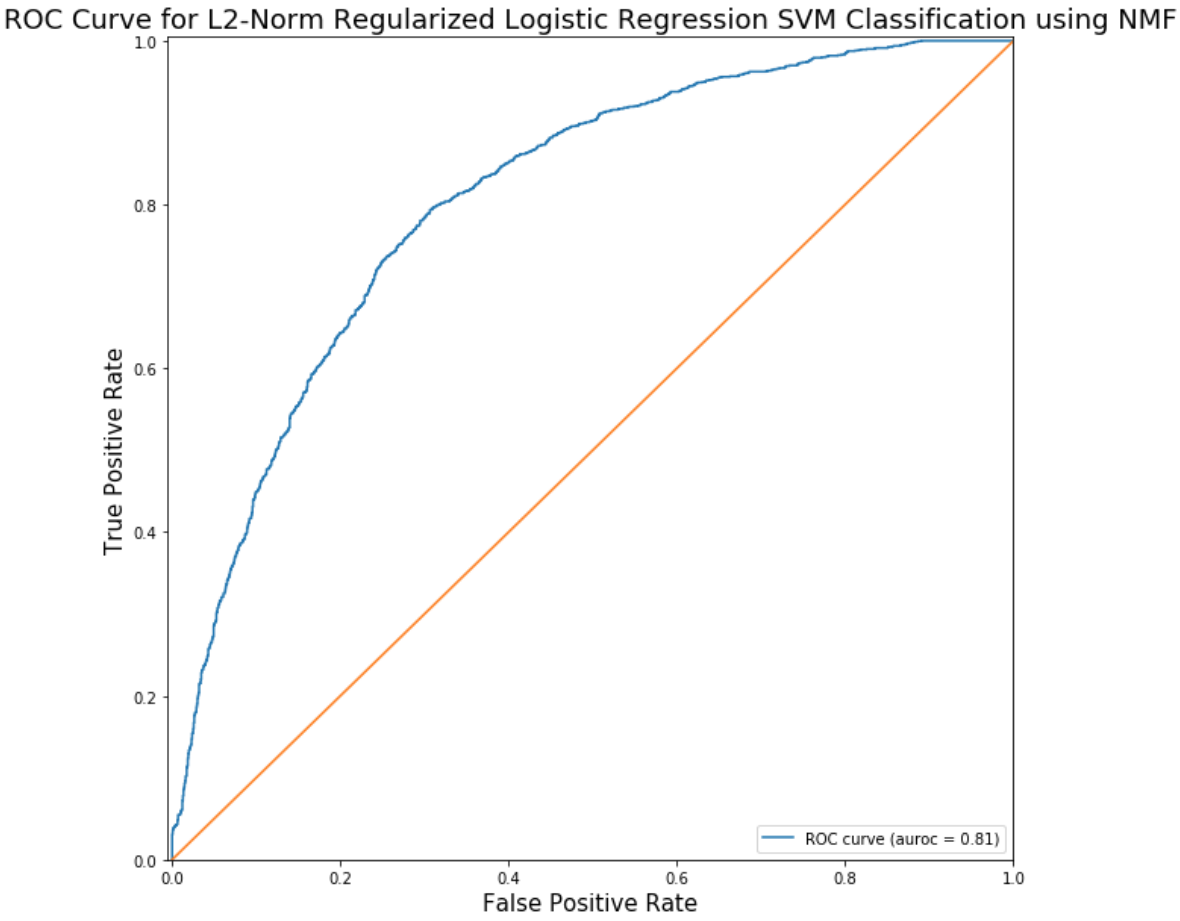
For NMF min_df = 2:

```
L2-Norm Regularized Logistic Regression SVC Accuracy using NMF: 0.733015873016
--------------------------------------------------
Classification report:
                        precision    recall  f1-score   support

  Computer technology       0.78      0.65      0.71      1560
Recreational activity       0.70      0.82      0.76      1590

          avg / total       0.74      0.73      0.73      3150

--------------------------------------------------
Confusion Matrix:
[[1009  551]
 [ 290 1300]]
--------------------------------------------------
```



Analysis of the result:

The results of accuracy of the simulation are shown as follows:

Setting	L1 Regularization	L2 Regularization
LSI & min_df = 2	0.9232	0.9352
LSI & min_df = 5	0.9330	0.9314
NMF & min_df = 2	0.7619	0.7330

Comparing the results above, L1 regularization has better performance under the setting of LSI & mind_df = 5 and NMF & min_df = 2. The purpose of regularization is trying to prevent overfit during the training process.

j) Multiclass Classification (with min_df = 2)

We had studied various binary classification methods in the previous sections. In this section, we explored multiclass classification techniques with different algorithms, namely Naive Bayes, One-vs-one SVM and One-vs-rest SVM classifications. We aimed to train and test these classifiers to be able to classify documents into the four categories/classes mentioned in part b. Again, LSI and NMF were applied with min_df =2, and confusion matrix, accuracy, precision and recall were represented for each method as shown below:

For LSI min_df = 2:

- Naive Bayes:

```
Naive Bayes Multiclass Classification Accuracy: 0.807667731629
--------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.64      0.95      0.76       392
   comp.sys.mac.hardware       0.98      0.45      0.62       385
            misc.forsale       0.92      0.82      0.87       390
  soc.religion.christian       0.87      0.99      0.93       398

             avg / total       0.85      0.81      0.80      1565

--------------------------------------------------
Confusion Matrix:
[[373   0   8  11]
 [159 175  18  33]
 [ 51   3 320  16]
 [  2   0   0 396]]
--------------------------------------------------
```

- One-to-one:

```
One-vs-One Multiclass Classification Accuracy: 0.879233226837
--------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.64      0.95      0.76       392
   comp.sys.mac.hardware       0.98      0.45      0.62       385
            misc.forsale       0.92      0.82      0.87       390
  soc.religion.christian       0.87      0.99      0.93       398

             avg / total       0.85      0.81      0.80      1565

--------------------------------------------------
Confusion Matrix:
[[292  72  28   0]
 [ 14 348  23   0]
 [ 12  20 358   0]
 [  1   6  13 378]]
--------------------------------------------------
```

- One-to-rest:

```
One-vs-Rest Multiclass Classification Accuracy: 0.861980830671
--------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.64      0.95      0.76       392
   comp.sys.mac.hardware       0.98      0.45      0.62       385
            misc.forsale       0.92      0.82      0.87       390
  soc.religion.christian       0.87      0.99      0.93       398

             avg / total       0.85      0.81      0.80      1565

--------------------------------------------------
Confusion Matrix:
[[260  99  33   0]
 [  9 355  21   0]
 [  8  24 358   0]
 [  0  10  12 376]]
--------------------------------------------------
```

For NMF min_df = 2:

- Naive Bayes:

```
Naive Bayes Multiclass Classification Accuracy using NMF: 0.83642172524
-------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.71      0.84      0.77       392
   comp.sys.mac.hardware       0.90      0.67      0.77       385
            misc.forsale       0.81      0.84      0.83       390
  soc.religion.christian       0.96      0.99      0.98       398

             avg / total       0.85      0.84      0.84      1565

-------------------------------------------------
Confusion Matrix:
[[328  20  40   4]
 [ 90 257  33   5]
 [ 44   9 329   8]
 [  0   0   3 395]]
-------------------------------------------------
```

- One-to-one:

```
One-vs-One Multiclass Classification Accuracy using NMF: 0.851118210863
-------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.69      0.86      0.77       392
   comp.sys.mac.hardware       0.83      0.75      0.79       385
            misc.forsale       0.93      0.83      0.88       390
  soc.religion.christian       1.00      0.96      0.98       398

             avg / total       0.86      0.85      0.85      1565

-------------------------------------------------
Confusion Matrix:
[[337  45  10   0]
 [ 83 289  13   0]
 [ 53  12 325   0]
 [ 13   1   3 381]]
-------------------------------------------------
```

- One-to-rest:

```
One-vs-Rest Multiclass Classification Accuracy using NMF: 0.856230031949
-------------------------------------------------
Classification report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.74      0.83      0.78       392
   comp.sys.mac.hardware       0.83      0.76      0.79       385
            misc.forsale       0.89      0.86      0.87       390
  soc.religion.christian       0.97      0.98      0.98       398

             avg / total       0.86      0.86      0.86      1565

-------------------------------------------------
Confusion Matrix:
[[324  46  19   3]
 [ 71 291  19   4]
 [ 36  15 334   5]
 [  4   0   3 391]]
-------------------------------------------------
```

Analysis of the result:

The results of accuracy are shown in the following table:

Setting	Naive	One-to-One	One-to-Rest
LSI & min_df = 2	0.8077	0.8792	0.8620
NMF & min_df = 2	0.8364	0.8511	0.8562

From the results above, we could see that LSI had a slightly better overall accuracy for each classification than that of NMF, and One-vs-Rest SVM classification provided the best accuracy among these three methods.

Conclusion

In this project, we have studied and performed classification analysis with texture data using different classification methods, both binary and multiclass. We have first applied feature extraction to reduce the size of documents into an array of terms, which would then be fitted into TFxIDF vectors. Next, for feature selection, we have used both LSI and NMF to further reduce the dimensionality of TFxIDF vectors and picked 50 features. We have then studied different algorithms for binary classification, such as hard/soft margin SVM, Bayes algorithm and logistic regression with/without regularization; and we have examined the ROC curve, confusion matrix, accuracy, precision and recall for each of these classifiers. Finally, we have implemented multiclass classification, namely Naive Bayes algorithm, One-vs-one SVM and One-vs-rest SVM classifications. We have also studied their performance, following the similar steps for binary case.