EE 219

Project 1

Classification Analysis on Textual Data

WenShan Li
UID: 105026914
helenali19@g.ucla.edu

Wei Du
UID:005024944
ericdw@g.ucla.edu

I. Introduction

In this project, our general task is to use statistical classification methods to identify a category, from a predefined set, to which a data point belongs, given a training data set with known labels. More specifically, we are given a dataset containing approximately 20,000 newsgroup documents, each corresponding to a specific topic. Our goal is to first, extract training data for a total number of eight subclasses, each of which belongs to two major classes 'Computer Technology' and 'Recreational activity'. Then we build different classification models based on those training data and test those classification models with testing data to compare the performance of each classification model along with the different inputting parameters. Table 1 shows the 2 major classes together with the subclasses each major class contain.

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）

II. Dataset and Problem Statement

2.1. Question a

To begin with, we first check the dataset status that if we need to handle any imbalance in the relative sizes of the data sets corresponding to different classes. Our data set contains two major classes, computer technology and recreational activity, which can be further divided into 8 subclasses. In order to understand the data distribution, here we plot a histogram of the number of training documents for each class in the data set.

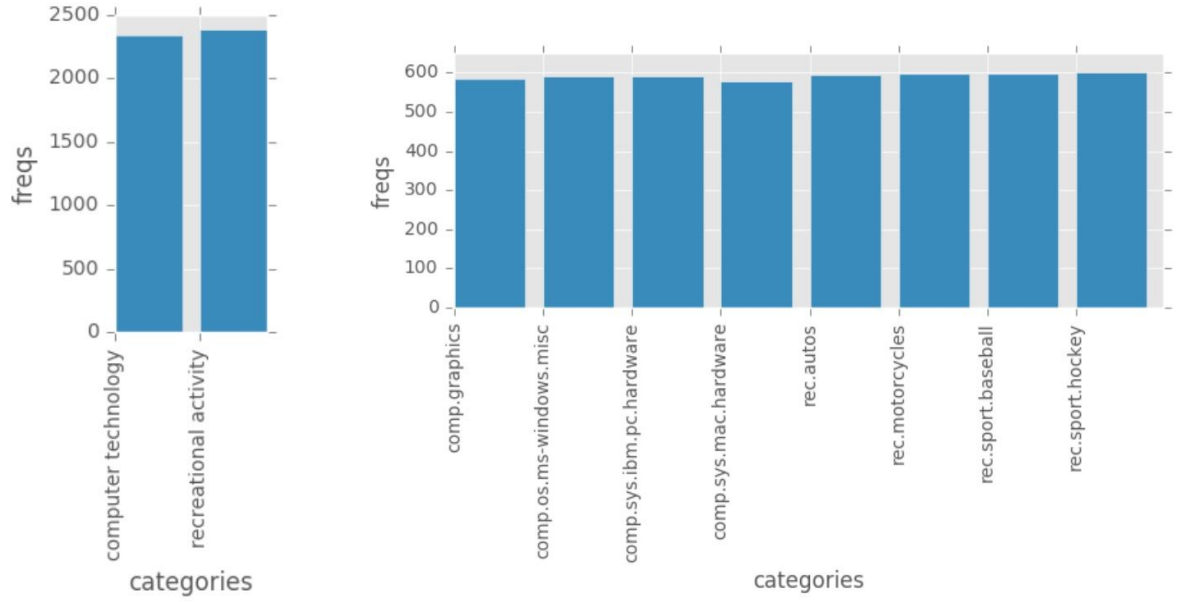


Figure 1. Distribution of 2 major classes(left); distribution of 8 subclasses(right)

From the figure above, we notice that the data for each class is evenly distributed, and our data set is balanced.

III. Modeling Text Data and Feature Extraction

Since each document may contain a large number of stopping words, punctuations and irrelevant words that might interfere with the accuracy while we build our classification model, to increase efficiency and accuracy, we need to first do some preprocessing of the dataset. To do so, we write a tokenizer which exclude all the punctuations. We also import nltk.stem package to convert all the words to their stemmed version. Then, by setting the value of min_df and max_df, we exclude all the words that appears in less than min_df documents or more than max_df of the documents.

3.1. Question b

The second step is to convert the documents into numerical feature vectors, which can be used for learning algorithms. We first need to tokenize each document into words, excluding the stop words, punctuations, and then using stemmed version of words, we create a TFxIDF vector representation. We use the CountVectorizer and TFxIDF package to implement the transformation. We compare the performance by setting min_df to different values, where min_df is the minimum threshold of the word frequency in documents. The result shows that by increasing min_df value, we can filter out a large number of terms, which can help us remove irrelevant features. However, if min_df is too large, there is a risk that we can also remove relevant features.

	Min_df = 2	Min_df = 5
Number of terms	22050	9144

3.2. Question c

In order to quantify how significant a word is to a class, we also measure TFxICF for each of the eight subclasses. For this part of the problem, we are asked to find the 10 most significant terms in four of the eight subclasses. The subclasses titles as well as their 10 most significant terms are all listed below.

comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
card	bit	sale	thing
use	line	new	christ
control	organ	mail	church
drive	appl	condit	word
disk	subject	price	peopl
scsi	post	ship	god
problem	use	card	way
bus	drive	use	jesus
time	mac	game	sin
pc	work	offer	time

We notice that the ending character ‘e’ are missing for some words. This is because we remove the ending during the process of stemming.

IV. Feature Selection

4.1. Question d

In this section, we need to perform dimension reduction to reduce the dimensionality of our TFxIDF matrix. Here we apply two different methods to reduce the dimension of input data, Latent Semantic Indexing (LSI) and Non-negative Matrix Factorization (NMF). We set

n_components = 50, which helps us to reduce the input dimension to 50 with removing some irrelevant features. We compare the performance of two dimension reduction methods in the following parts.

V. Learning algorithm

Our task is to classify the documents into "Computer Technology" and "Recreational Activity" in this part --- V. Learning algorithm . In total,we build some classification models to classify the documents: hard and soft margin SVM, multinomial naive Bayes classifier, and logistic regression classifier. Then we evaluate the performance for each classifier and compare the results based on ROC curve, confusion matrix, accuracy, recall and precision. Furthermore, we implement ROC curve to characterize the trade-off between the two quantities. Then we calculate its confusion matrix and draw the ROC curve.

5.1 Question e

5.1.1 Hard Margin SVM

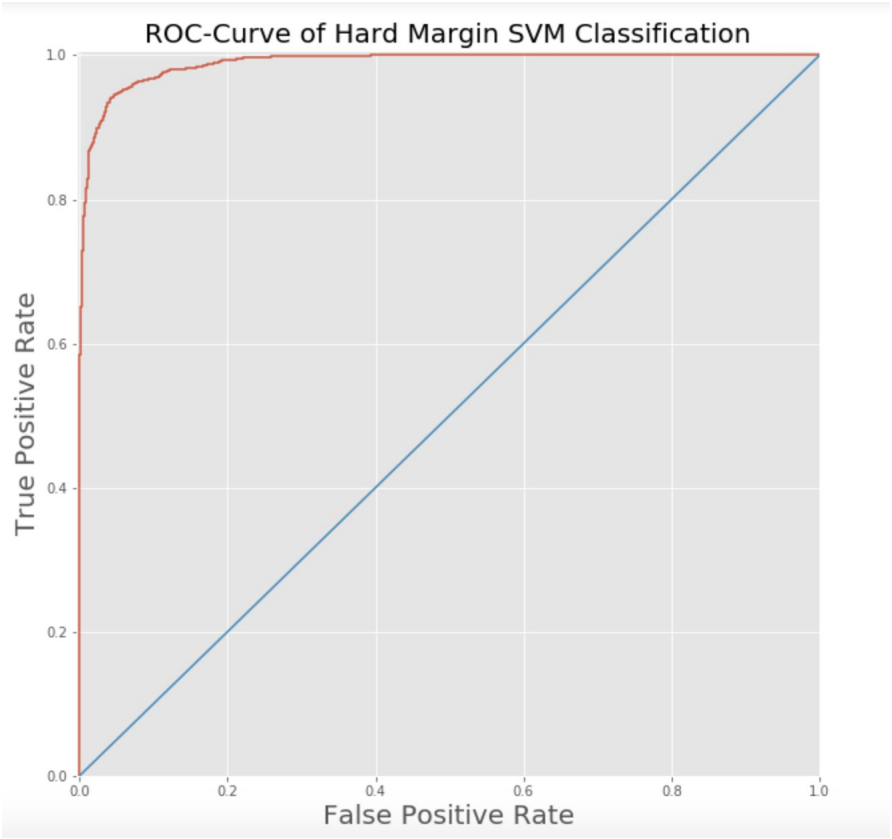
First, we used the hard margin SVM classifier by setting the $\gamma = 1000$. In addition, we first used min_df = 2 and min_df=5 respectively, to build the classifier for LSI. Then we set min_df = 2 for NMF. The 20 Newsgroups train data is used to train the model and the test data is used to draw ROC curve. And then we use confusion matrix to evaluate the algorithm and draw the confusion matrix in picture.

5.1.1.1 LSI with min_df = 2

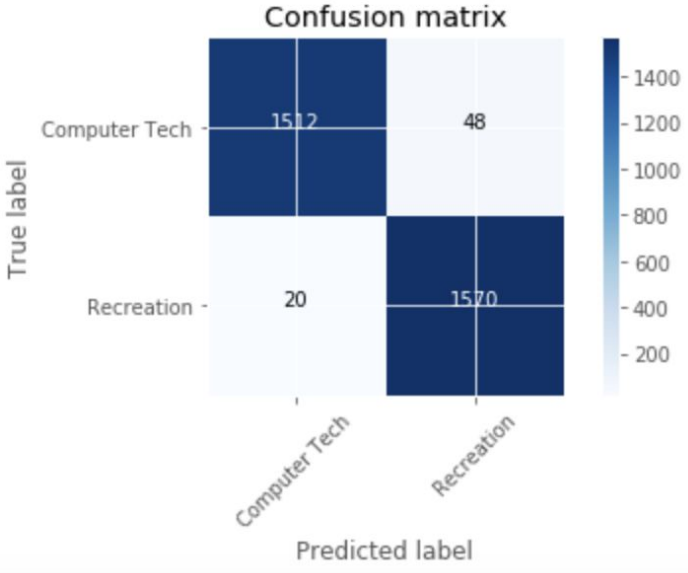
Accuracy of Hard Margin SVM: 0.978412698413

Classification report:

	precision	recall	f1-score	support
Computer technology	0.99	0.97	0.98	1560
Recreational activity	0.97	0.99	0.98	1590
avg / total	0.98	0.98	0.98	3150




```
Confusion matrix
[[1512   48]
 [  20 1570]]
```



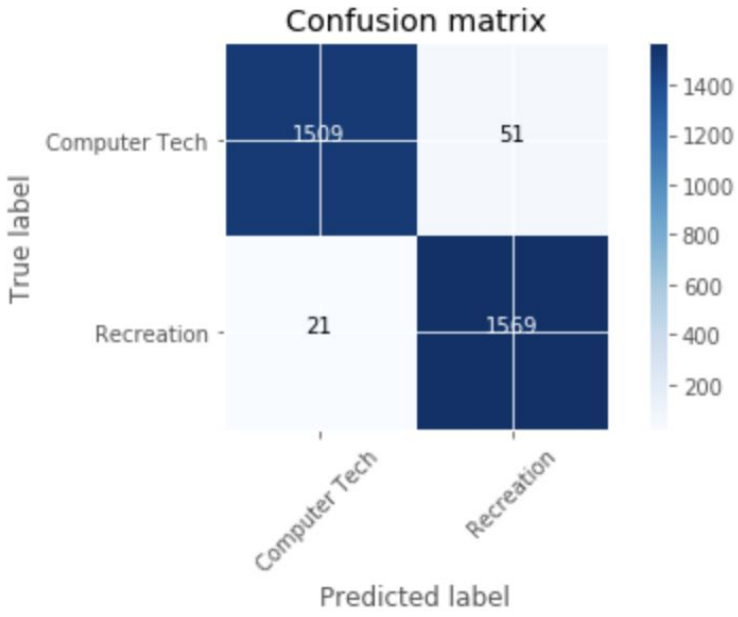
5.1.1.2 LSI with min_df = 5

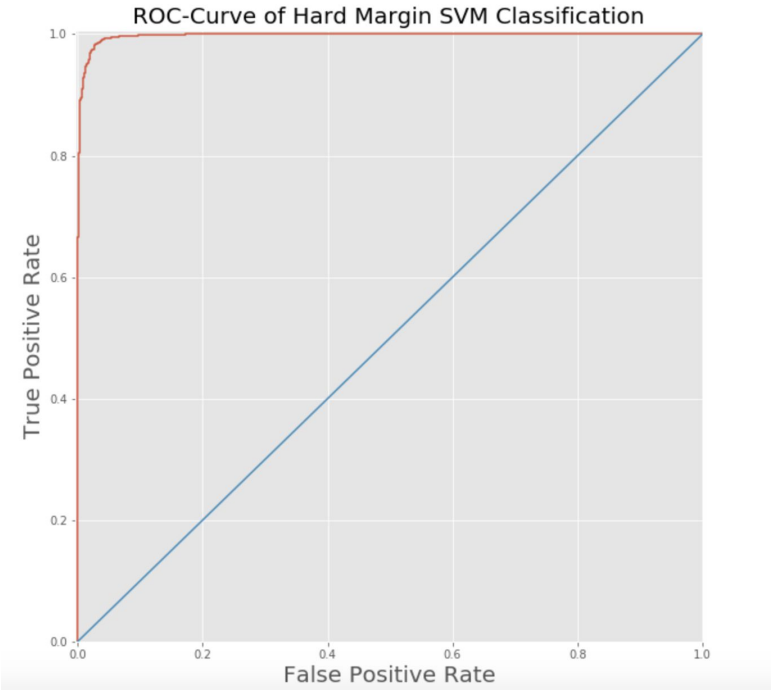
Accuracy of Hard Margin SVM: 0.977142857143

Classification report:

	precision	recall	f1-score	support
Computer technology	0.99	0.97	0.98	1560
Recreational activity	0.97	0.99	0.98	1590
avg / total	0.98	0.98	0.98	3150

```
Confusion matrix
[[1509   51]
 [  21 1569]]
```





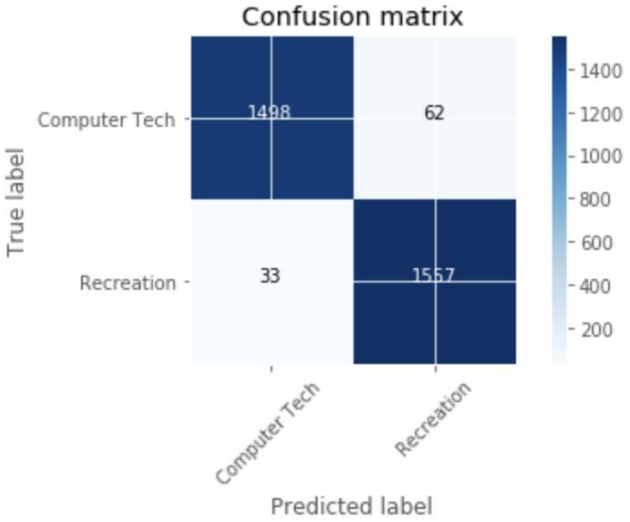
5.1.1.3 NMF with min_df = 2

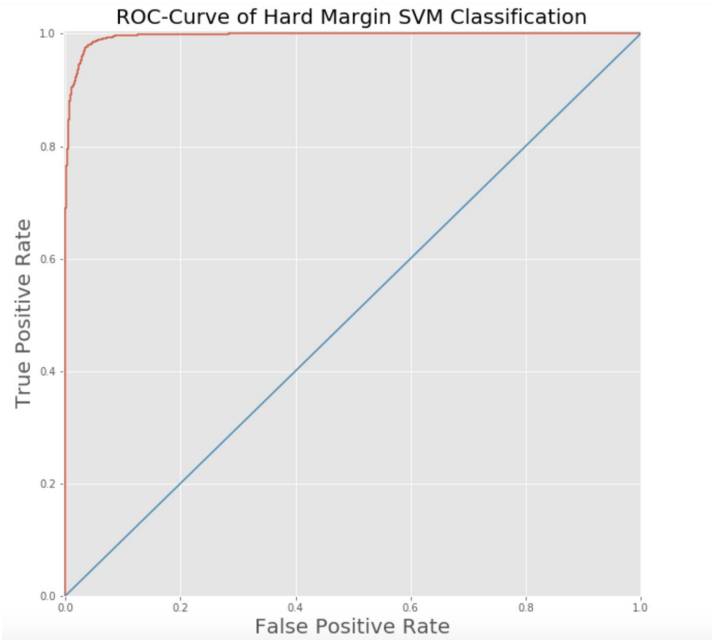
Accuracy of Hard Margin SVM: 0.969841269841

Classification report:

	precision	recall	f1-score	support
Computer technology	0.98	0.96	0.97	1560
Recreational activity	0.96	0.98	0.97	1590
avg / total	0.97	0.97	0.97	3150

```
Confusion matrix
[[1498   62]
 [  33 1557]]
```





5.1.2 Soft Margin SVM

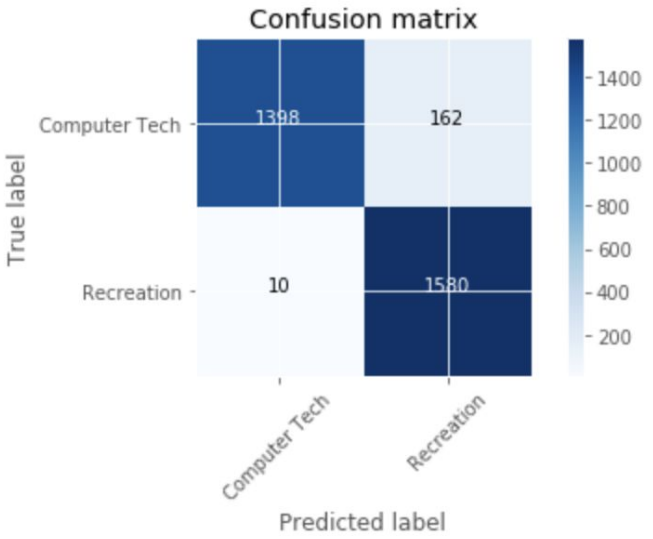
Then, we used the soft margin SVM classifier by setting the $\gamma = 0.001$. Again, we used min_df = 2 and max_df = 0.99 for both LSI and NMF and min_df = 5 for only LSI to build the classifier.

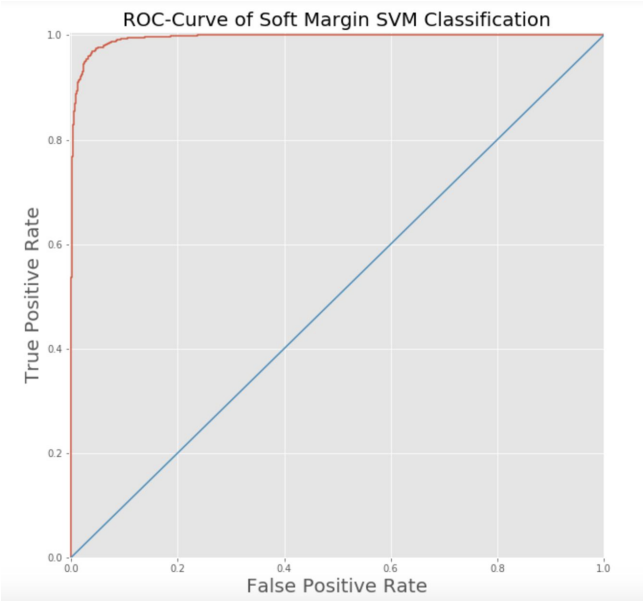
5.1.2.1 LSI with min_df = 2

Accuracy of Soft Margin SVM: 0.945396825397

Classification report:

	precision	recall	f1-score	support
Computer technology	0.99	0.90	0.94	1560
Recreational activity	0.91	0.99	0.95	1590
avg / total	0.95	0.95	0.95	3150





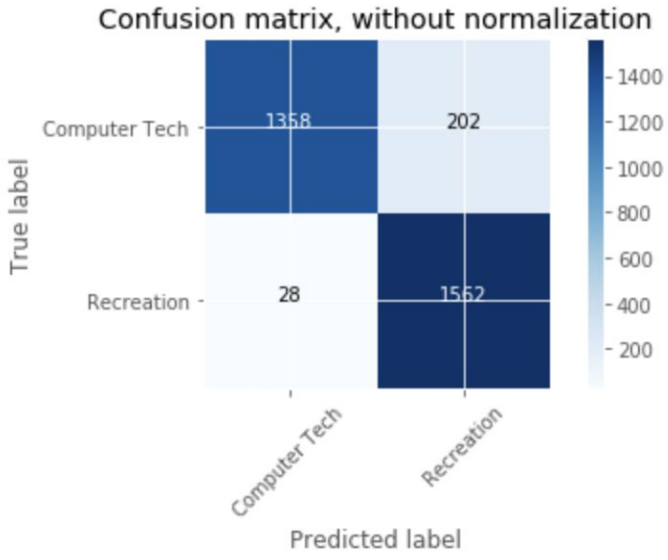
5.1.2.2 LSI with min_df = 5

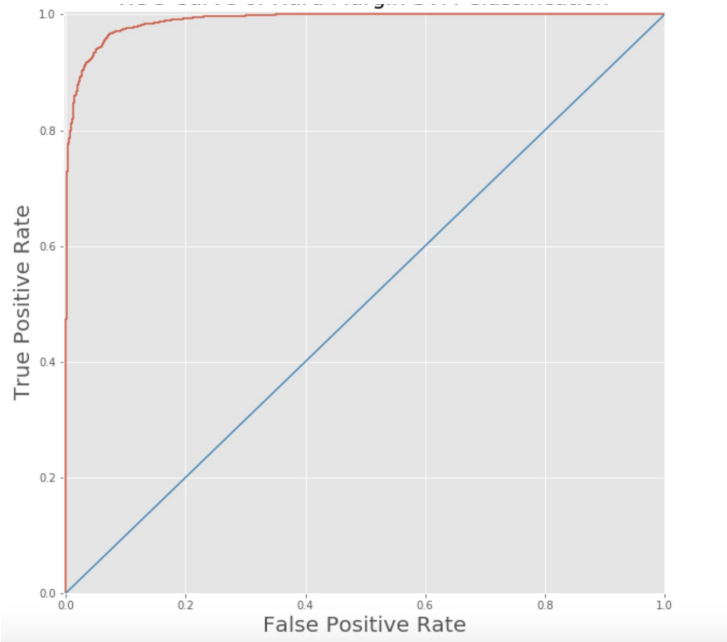
Accuracy of Soft Margin SVM: 0.948888888889

Classification report:

	precision	recall	f1-score	support
Computer technology	0.99	0.90	0.95	1560
Recreational activity	0.91	0.99	0.95	1590
avg / total	0.95	0.95	0.95	3150

```
Confusion matrix, without normalization
[[1358  202]
 [  28 1562]]
```





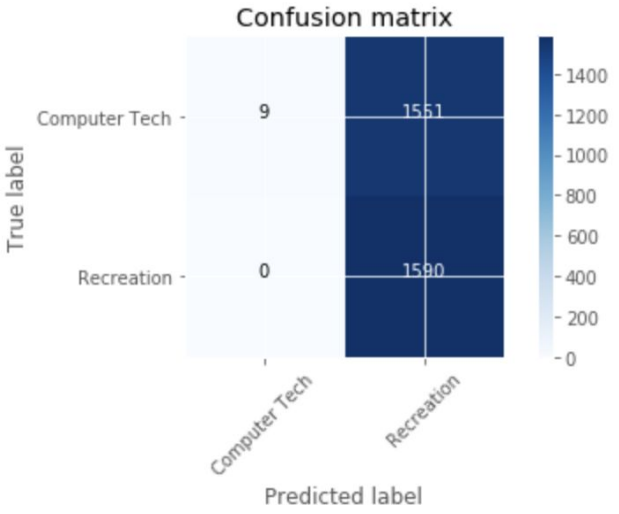
5.1.2.3 NMF with min_df = 2

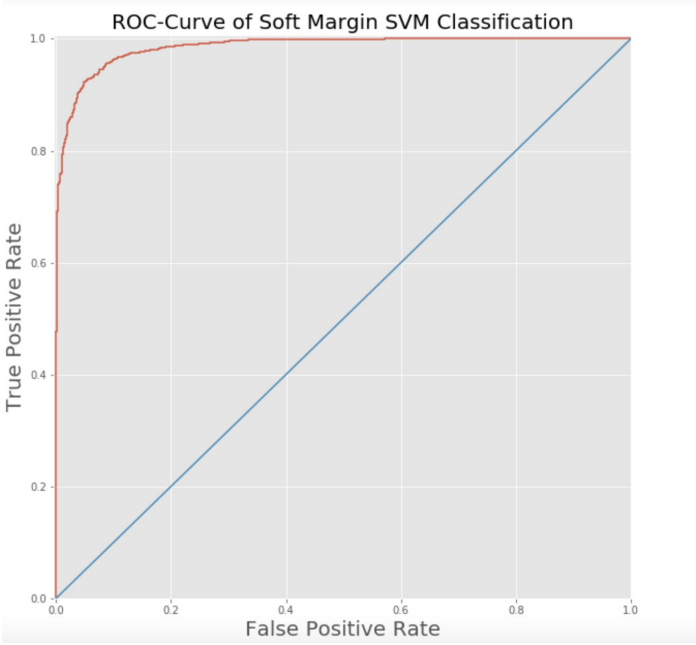
Accuracy of Soft Margin SVM: 0.507619047619

Classification report:

	precision	recall	f1-score	support
Computer technology	1.00	0.01	0.01	1560
Recreational activity	0.51	1.00	0.67	1590
avg / total	0.75	0.51	0.34	3150

```
Confusion matrix
[[   9 1551]
 [   0 1590]]
```





5.1.3 Analysis and Compare

As we can see by comparing the 6 different results(and these results are shown in the former chart), the accuracy, recall, precision and score for hard SVM classifier are high. On the other hand, soft SVM has high score and recall, but the accuracy and precision for soft SVM tends to be low. This makes sense because soft SVM uses a really small penalty value C = 0.001 which means that the ability the SVM optimization avoid misclassifying in each training example. Therefore, by setting C to 0.001, we allow more misclassification, which will result in a

relatively low accuracy and precision. Furthermore, by measuring these elements in both SLI and NMF, we find that the LSI dimensionality reduction method performs nearly the same as the NMF. And the min_df in this case doesn't matter much.

5.2 Question f

For this question, we use a 5-fold cross-validation method to find the best value of the parameter C given a range that C = 10^k while -3 <=k<=3. In order to record the best C value, we used 5-fold cross-validation to find an average score using each parameter. Finally, we will compare the results using both LSI and NMF.

5.2.1 LSI with min_df = 2

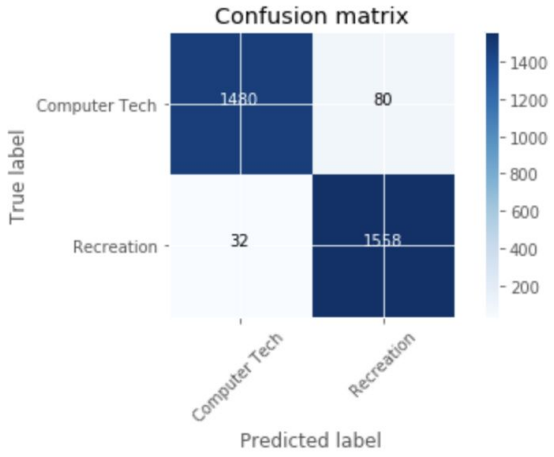
By applying LSI techniques while setting min_df = 2, our best penalty C parameter is 100 with a score of 0.9782.

Accuracy = 0.96444444444

Recall = 0.95115995116

Precision = 0.979874213836

```
Confusion matrix
[[1480   80]
 [  32 1558]]
```



5.2.2 LSI with min_df = 5

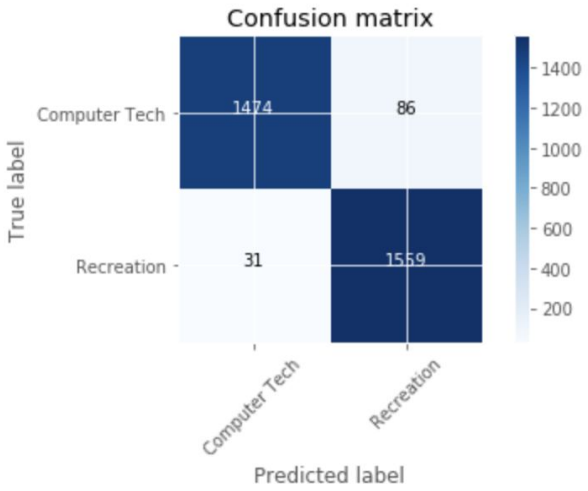
By applying LSI techniques, our best penalty C parameter is 10 with a score of 0.962857142857.

Accuracy = 0.962857142857

Recall = 0.947720364742

Precision = 0.980503144654

```
Confusion matrix
[[1474   86]
 [  31 1559]]
```



5.2.2 NMF with min_df = 2

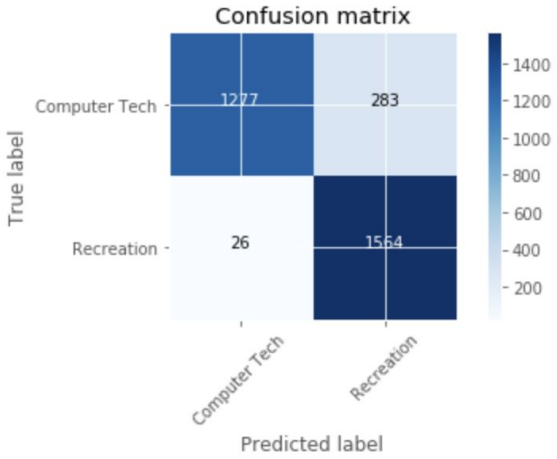
By applying LSI techniques while setting min_df = 2, our best penalty C parameter is 100 with a score of 0.901904761905.

Accuracy = 0.901904761905

Recall = 0.846778559827

Precision = 0.983647798742

```
Confusion matrix
[[1277  283]
 [  26 1564]]
```



5.2.3 Analysis and Compare

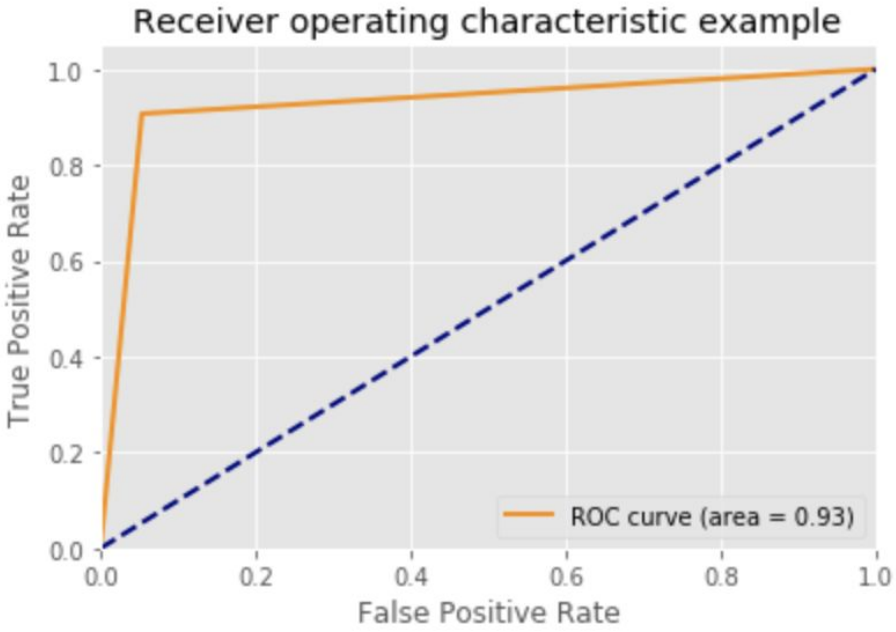
In conclusion, by using LSI and NMF, we can both find different best penalty parameter values. We further find that when we change the C to the best value, the accuracies and scores are equal to the number of the situation that C=1000. The reason why this happen is that the best C values found by the program are all pretty large, and when we use new classifier to train them, it is interpreted as hard SVM. And no matter what the min_df is, the result will not affected by this parameter.

5.3. Question g

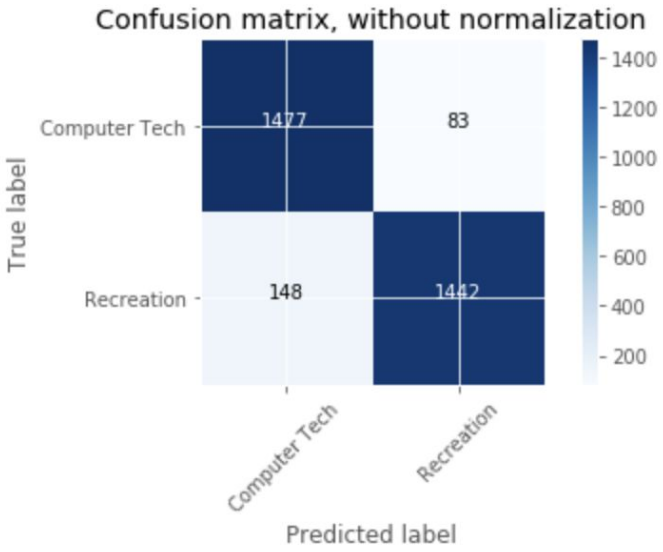
The second data mining algorithm we use is naive Bayes. Naive Bayes is a algorithm designed specifically for textual data. It estimates the maximum likelihood probability of a class given a document with a set of features using Bayes rule. Since Multinomial Naïve Bayes requires non-negative features, we only train it with features from NMF.

5.3.1. NMF with min_df = 2

Accuracy = 0.926666666667
Recall = 0.906918238994
Precision = 0.945573770492



```
Confusion matrix, without normalization
[[1477   83]
 [ 148 1442]]
```



5.4. Question h & i

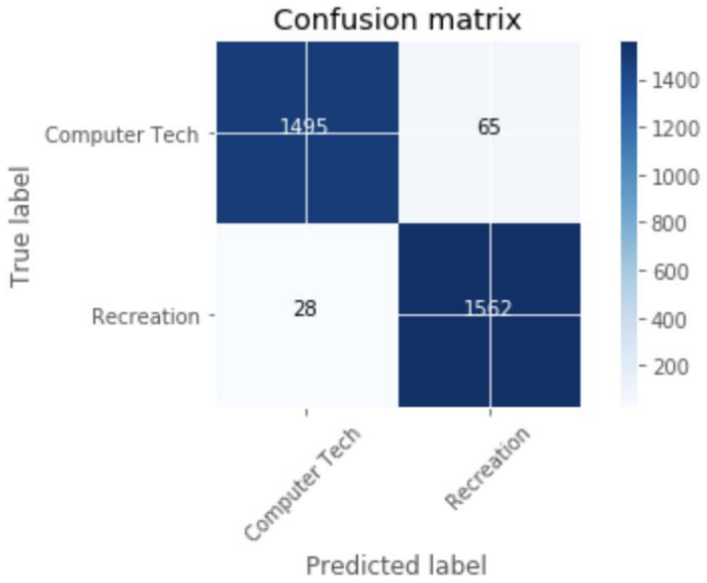
Lastly, we build logistic regression classifier to train the training data. We perform the same evaluation as the the SVM algorithm. Repeat it by adding a regularization term to the optimization objective. We will try both l_1 and l_2 norm regularizations and sweep through different regularization coefficients, ranging from very small ones to large ones. We used both NMF and LSI dimensionality reduction methods for both regularization.

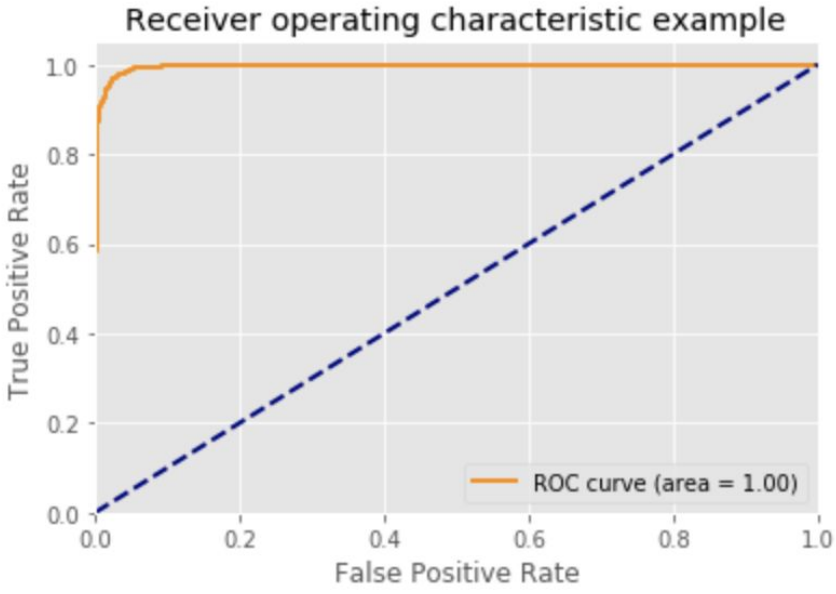
5.4.1. L2 norm regularization

5.4.1.1. LSI with min_df = 2

For L1 norm regulation using LSI and set min_df = 2, we find the best coefficient = 100.
Accuracy = 0.977777777778
Recall = 0.987421383648
Precision = 0.969135802469

```
Confusion matrix
[[1495   65]
 [  28 1562]]
```





The predicted matrix with different coef:

coef: 0.01
[[1419 141]
 [12 1578]]

coef: 0.1
[[1485   75]
 [32 1558]]

coef: 1
[[1495   65]
 [28 1562]]

coef: 10
[[1504 56]
[21 1569]]

coef: 100
[[1515 45]
[20 1570]]

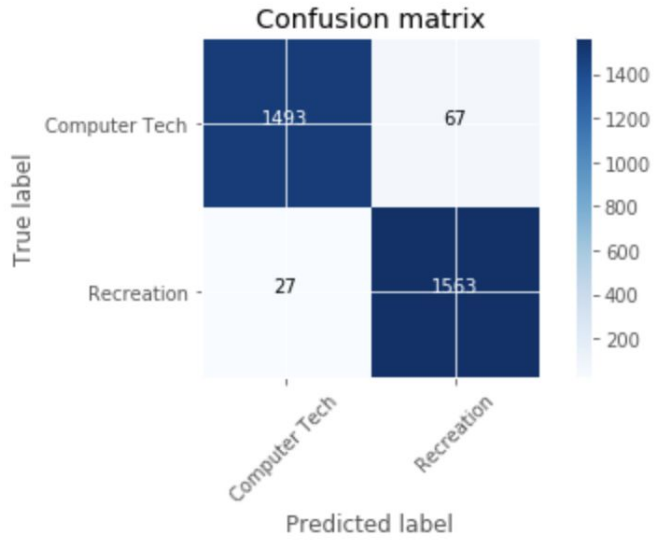
coef: 1000
[[1510 50]
[20 1570]]

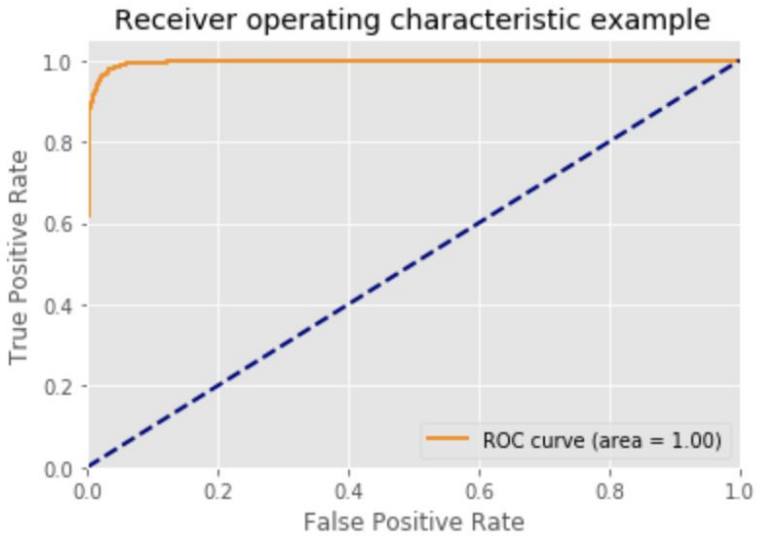
coef: 10000
[[1510 50]
[20 1570]]

5.4.1.2. LSI with min_df = 5

For L2 norm regulation using LSI and set min_df = 5, we find the best coefficient = 1000
Accuracy = 0.970158730159
Recall =0.983018867925
Precision = 0.958895705521

```
Confusion matrix
[[1493   67]
 [  27 1563]]
```





The predicted matrix with different coef:

coef: 0.01
[[1419 141]
[17 1573]]

coef: 1
[[1493 67]
[27 1563]]

coef: 100
[[1511 49]
[23 1567]]

coef: 0.1
[[1477 83]
[31 1559]]

coef: 10
[[1505 55]
[24 1566]]

coef: 1000
[[1511 49]
[24 1566]]

coef: 10000
[[1511 49]
[24 1566]]

5.4.1.3. NMF with min_df = 2

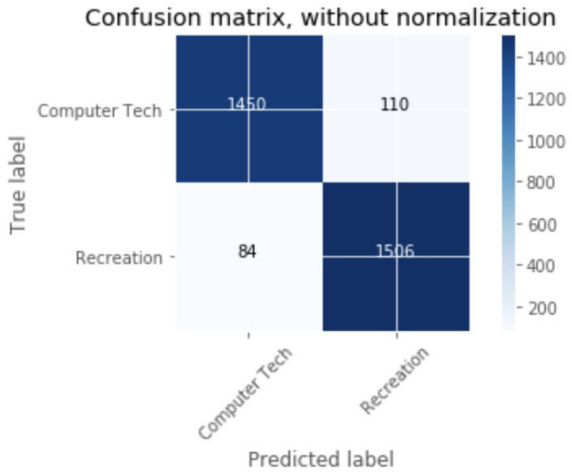
For L2 norm regulation using LSI and set min_df = 2, we find the best coefficient = 1000.

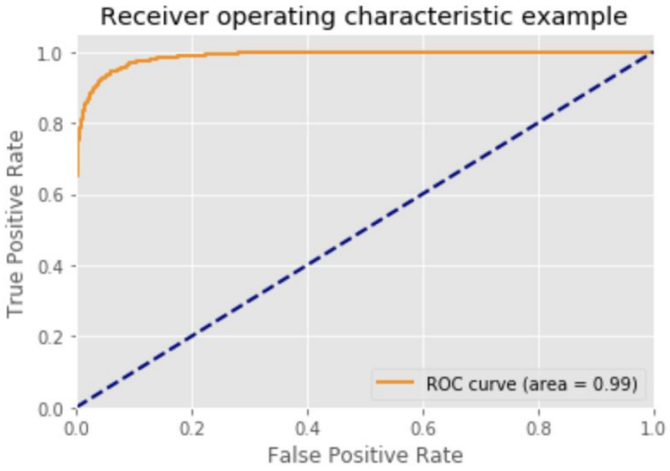
Accuracy = 0.938412698413

Recall = 0.947169811321

Precision = 0.931930693069

```
Confusion matrix
[[1450  110]
 [  84 1506]]
```





The predicted matrix with different coef:

coef: 0.01
[[24 1536]
 [0 1590]]

coef: 1
[[1450 110]
 [84 1506]]

coef: 100
[[1474   86]
 [57 1533]]

coef: 0.1
[[1322 238]
 [34 1556]]

coef: 10
[[1465   95]
 [78 1512]]

coef: 1000
[[1488   72]
 [39 1551]]

coef: 10000
[[1496   64]
 [35 1555]]

5.4.2. L1 norm regularization

5.4.2.1. LSI with min_df = 2

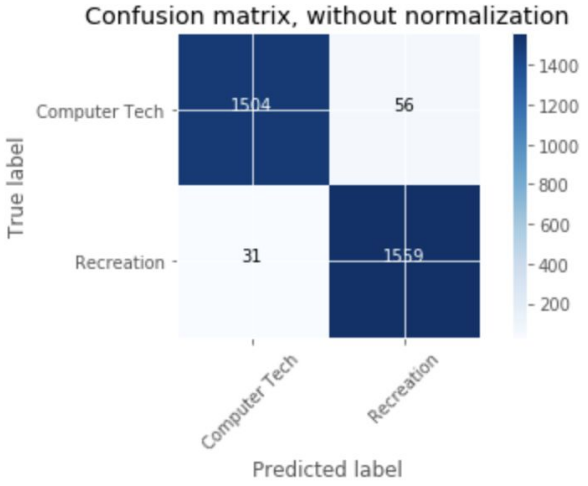
For L1 norm regulation using LSI and set min_df = 2, we find the best coefficient = 100.

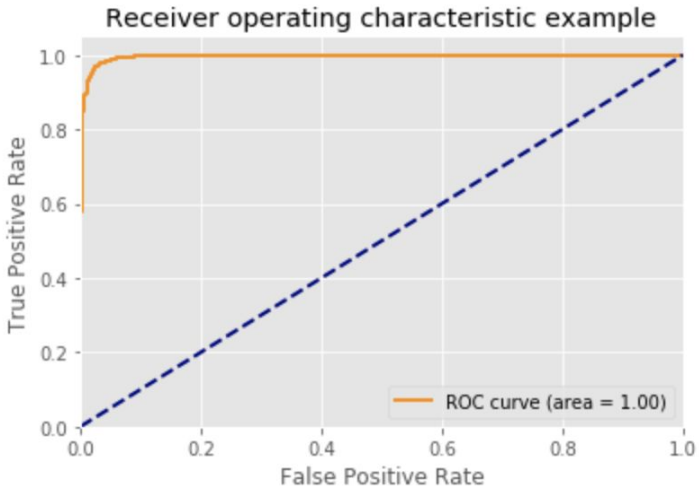
Accuracy = 0.977777777778

Recall =0.987421383648

Precision = 0.969135802469

```
Confusion matrix
[[1504   56]
 [  31 1559]]
```





The predicted matrix with different coef:

coef: 0.01
[[1419 141]
[12 1578]]

coef: 0.1
[[1485 75]
[32 1558]]

coef: 1
[[1495 65]
[28 1562]]

coef: 10
[[1504 56]
 [21 1569]]

coef: 100
[[1515 45]
 [20 1570]]

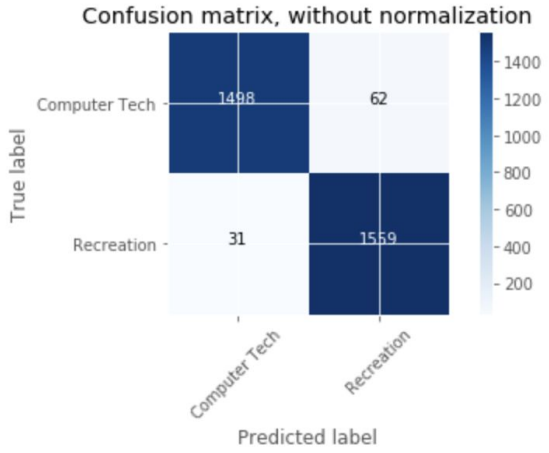
coef: 1000
[[1510 50]
 [20 1570]]

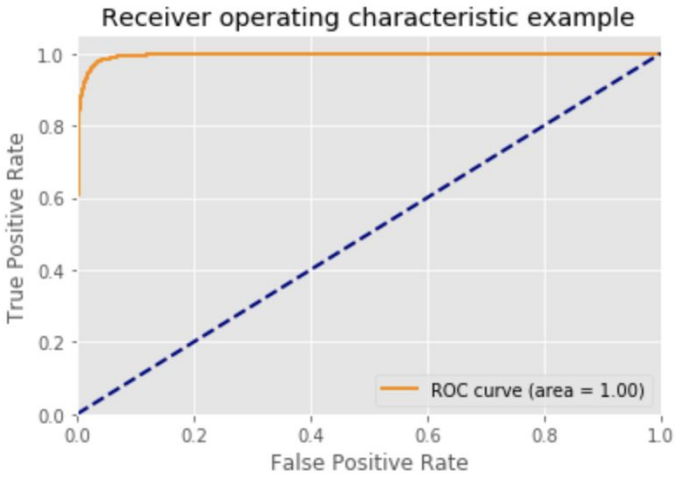
coef: 10000
[[1510 50]
 [20 1570]]

5.4.2.1. LSI with min_df = 5

For L1 norm regulation using NMF and set min_df = 5, we find the best coefficient = 100
Accuracy = 0.970476190476
Recall = 0.980503144654
Precision = 0.961752004935

```
Confusion matrix
[[1498   62]
 [  31 1559]]
```





The predicted matrix with different coef:

coef: 0.01
[[1419 141]
[17 1573]]

coef: 1
[[1493 67]
[27 1563]]

coef: 100
[[1511 49]
[23 1567]]

coef: 0.1
[[1477 83]
[31 1559]]

coef: 10
[[1505 55]
[24 1566]]

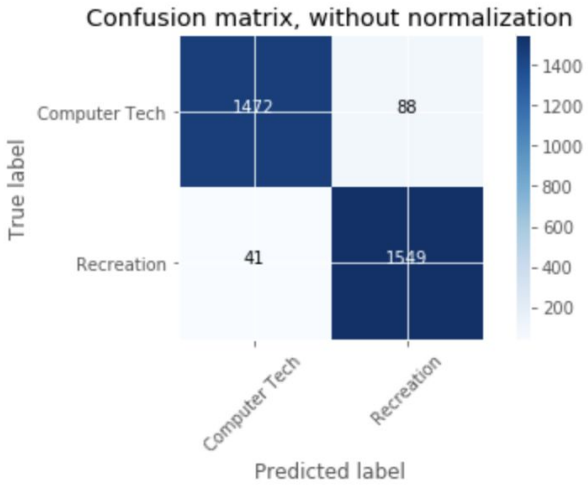
coef: 1000
[[1511 49]
[24 1566]]

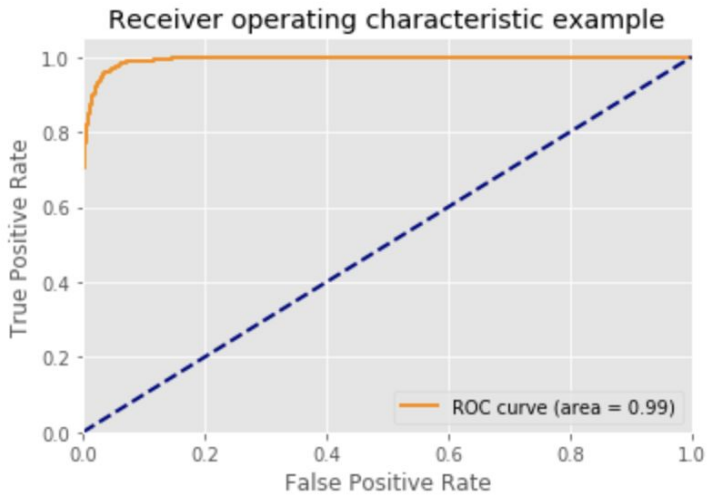
coef: 10000
[[1511 49]
[24 1566]]

5.4.2.3. NMF with min_df = 2

For L1 norm regulation using NMF and set min_df = 2, we find the best coefficient = 1000.
Accuracy = 0.959047619048
Recall = 0.974213836478
Precision = 0.946243127673

```
Confusion matrix
[[1472   88]
 [  41 1549]]
```





The predicted matrix with different coef:

coef: 0.01
[[24 1536]
[0 1590]]

coef: 1
[[1450 110]
[84 1506]]

coef: 100
[[1474 86]
[57 1533]]

coef: 0.1
[[1322 238]
[34 1556]]

coef: 10
[[1465 95]
[78 1512]]

coef: 1000
[[1488 72]
[39 1551]]

coef: 10000
[[1496 64]
[35 1555]]

5.4.3. Analysis and Compare

As we can see from the above results, the regularization parameter doesn't affect the test error directly if we only change this parameter. The test error changes when the coefficient together changes, then we can conclude that the regularization parameter affects the test error along with the coefficient. In this case, we can see that when we apply L1 norm regularization, the accuracy reaches the max when C equals 10/100. The accuracy reaches max when C equals 100/1000 in the case of L2 norm regularization. However, the accuracies for L1 norm and L2 norm doesn't change much when we use the best coefficient for L1 norm and L2 norm and train the classifier using the best coefficient.

The purpose of using a regularization term is to prevent overfit. L2 norm has the property to be calculated computationally efficiently, on the contrary, L1 norm has the property of producing many coefficients with zero values or very small values with few large coefficients. Thus, we

should consider the trade-off between accuracy and efficiency when we choose L1 norm or L2 norm.

VI. Multiclass Classification

In this section, we expand binary classification of two major classes to multiclass classification of 4 subclasses. We pick 4 categories from the 8 subclasses in total: comp.sys.ibm.pc.hard, comp.sys.mac.hard, misc.forsale, soc.religion.christian. Every sample may belong to one of these four classes. We use the same method as the previous section to vectorize the input data into TFxIDF matrix, where we set min_df = 2. After that, we use LSI and NMF to reduce the input dimension to 50.

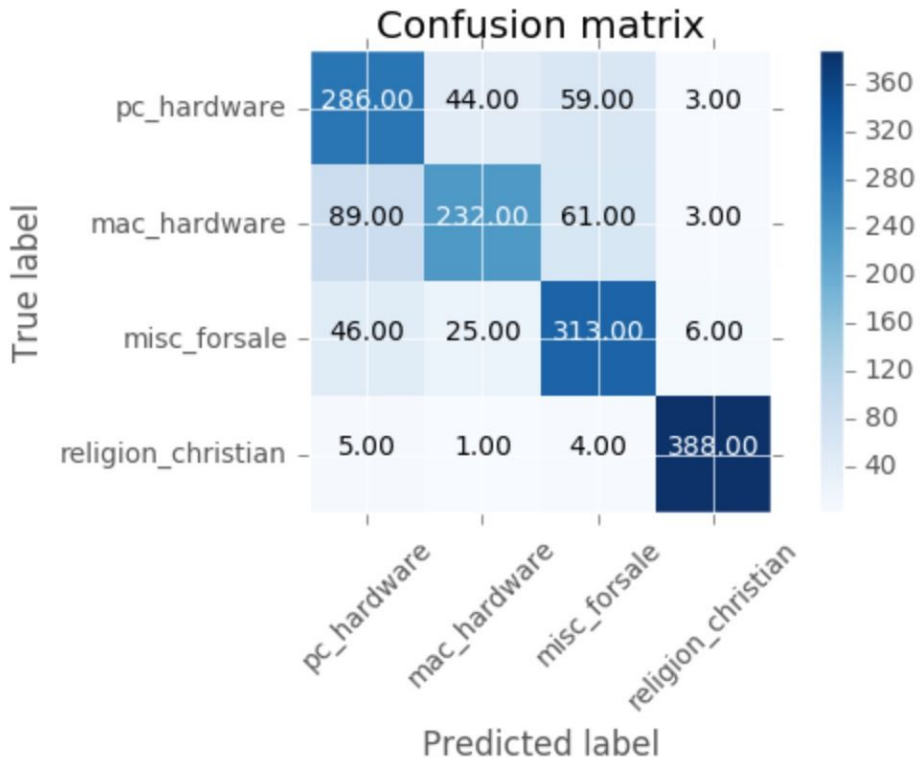
6.1. Question j

We use the Naive Bayes classifier and SVM classifier to perform the classification task. For both classifiers, we try One Vs One and One Vs Rest methods to train them. We test our classifiers on the test data and the comparison of results is listed as follows.

6.1.1. Naive Bayes using NMF and min_df = 2

We apply NMF to reduce the dimension of input data, and train a Multinomial Naive Bayes classifier on the processed data. We do not perform LSI to process the data, since Multinomial Naive Bayes cannot deal with negative features. The statistics of Naive Bayes after training are listed as follows.

Naive Bayes Accuracy	0.77124600639
Naive Bayes Recall	0.77124600639
Naive Bayes Precision	0.776252187772



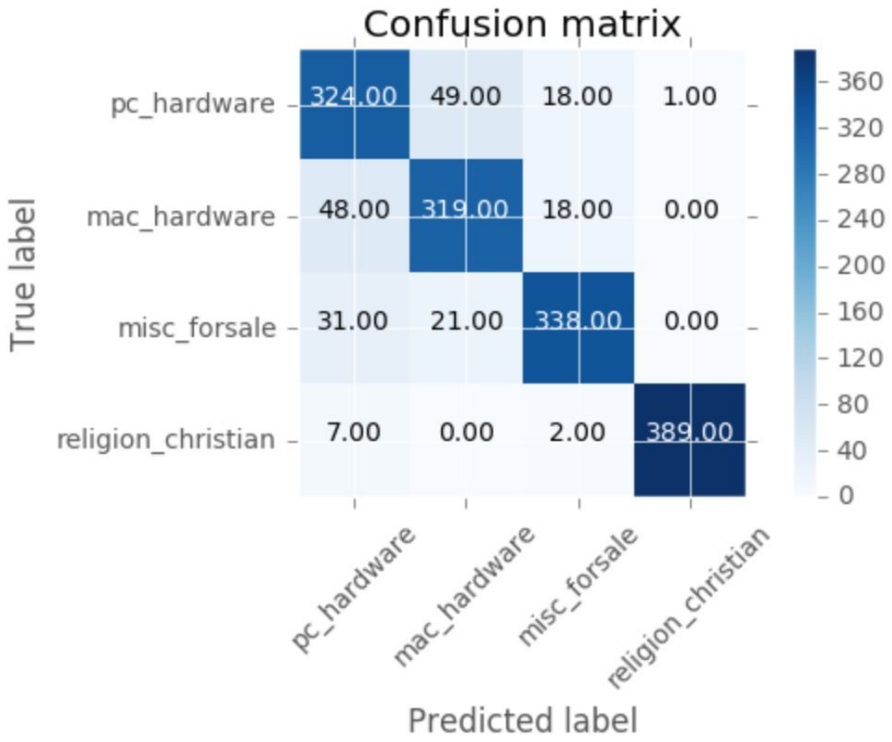
As we can see from the results above, Naive Bayes achieves a good accuracy, but there is still space for improvement.

6.1.2. Multiclass SVM 6.1.2.1. One VS One

6.1.2.1.1. NMF with min_df = 2

Now we move on with SVM classifier to perform multiclass classification. First we apply the same methods as above the preprocess the data. We use NMF to reduce the input dimension. We build the classifier based on One Vs One method. The statistics are listed as follows.

SVM Accuracy	0.875399361022
SVM Recall	0.875399361022
SVM Precision	0.877354631095

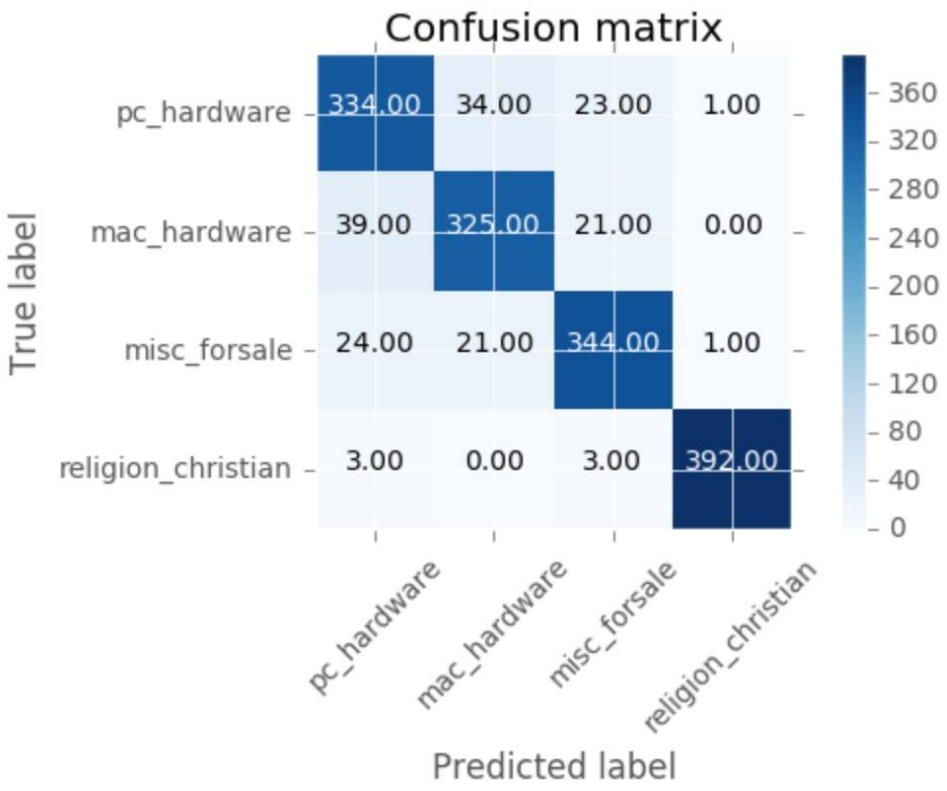


As we can see from the figure above, our One Vs One SVM classifier achieves an accuracy of 0.87. The performance of SVM is better than Naive Bayes.

6.1.2.1.2. LSI with min_df = 2

We repeat the same steps as the previous part, but this time we use LSI to preprocess the training data. The statistics of our trained SVM model are listed as follows.

SVM Accuracy	0.891373801917
SVM Recall	0.891373801917
SVM Precision	0.891818668258



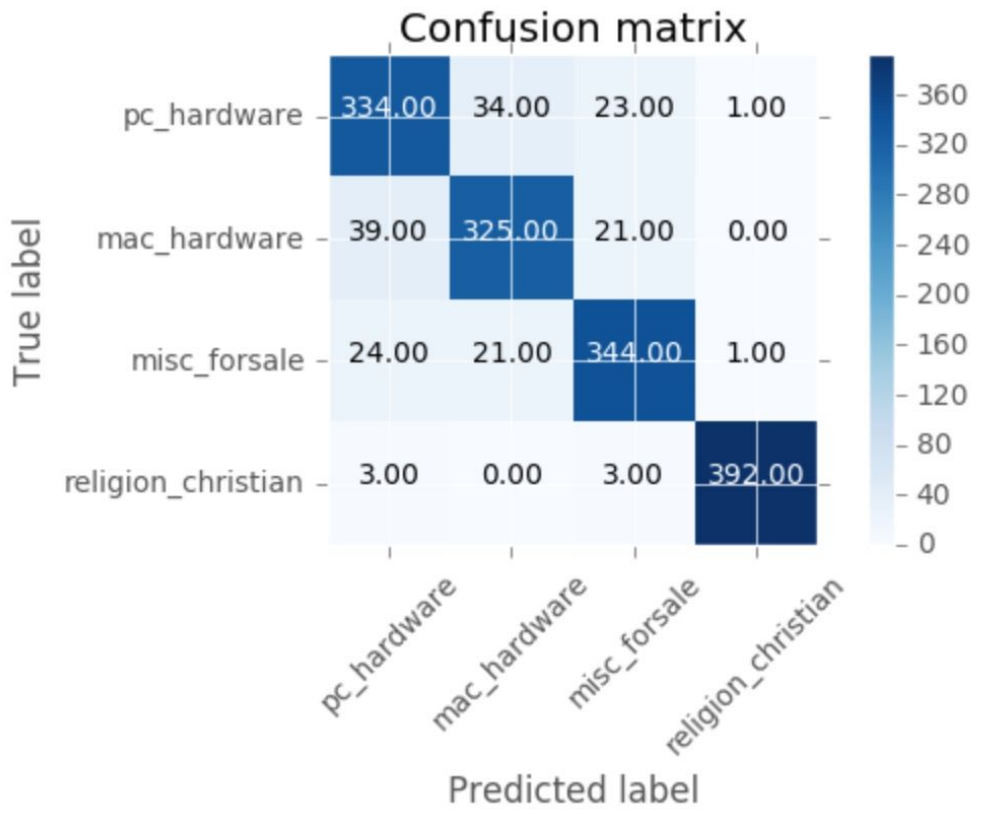
As we can see, the multiclass SVM classifier using LSI is slightly better than that of using NMF.

6.1.2.2. One Vs Rest

6.1.2.2.1. LSI with min_df = 2

In this part, instead of using One Vs One, we build our SVM with One Vs Rest method. We preprocess the data as the same before, and use LSI to reduce input dimension. The statistics of the trained SVM is listed as follows.

SVM Accuracy	0.898402555911
SVM Recall	0.898402555911
SVM Precision	0.898693624539

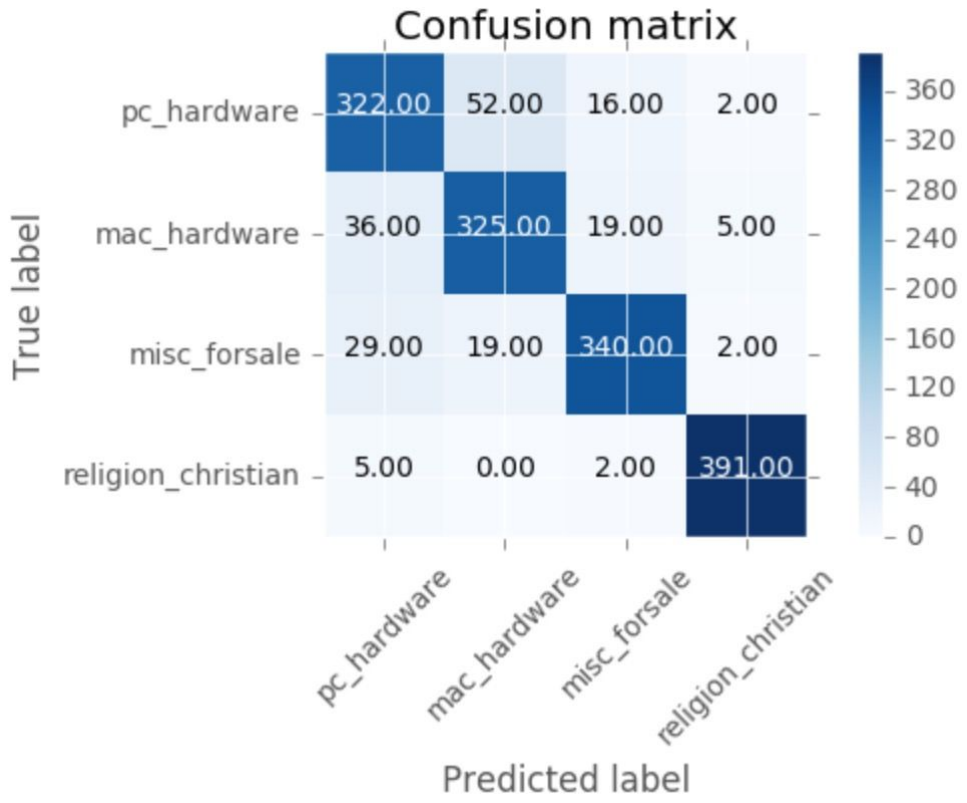


As we can see from the figure above, the performance of One Vs Rest SVM is similar to the performance of One Vs One.

6.1.2.2.2. NMF with min_df = 2

Then we use SVM classifier with One Vs Rest method to perform the same task. This time we use NMF to preprocess the training data. The statistics of the trained classifier are listed as follows.

SVM Accuracy	0.880511182109
SVM Recall	0.880511182109
SVM Precision	0.880984894667



As we can see from the figure above, the performance of SVM using LSI is slightly better than the performance of NMF.

6.2. Analysis

We can infer from the above experiments: the SVM classifier has better performance than Naive Bayes classifier. For SVM classifier, using One Vs One and One Vs Rest model have similar performance on our dataset. We also find that for dimension reduction of the input data, LSI results in slightly better performance than NMF.