EE 219 Project 1
Classification Analysis on Textual Data
Winter 2018

Xiaohan Wang

Yang Tang

Introduction

In this project, we did classification analysis on textual dataset "20 Newsgroup". We obtained data from "train" and "test" subsets separately and used training datasets to build classifier and then used testing datasets to verify the correctness. To train the classifier, we implemented and compared three different methods – Support Vector Machines, Naïve Bayes Algorithms and Logistic Regression, and plotted ROC curves, confusion matrices and accuracy results.

Dataset and Problem Statement

Our dataset comes from "20 Newsgroups" dataset, which is a collection of approximately 20000 newsgroup documents. It has been partitioned evenly across 20 different newsgroups and each corresponds to a different topic.

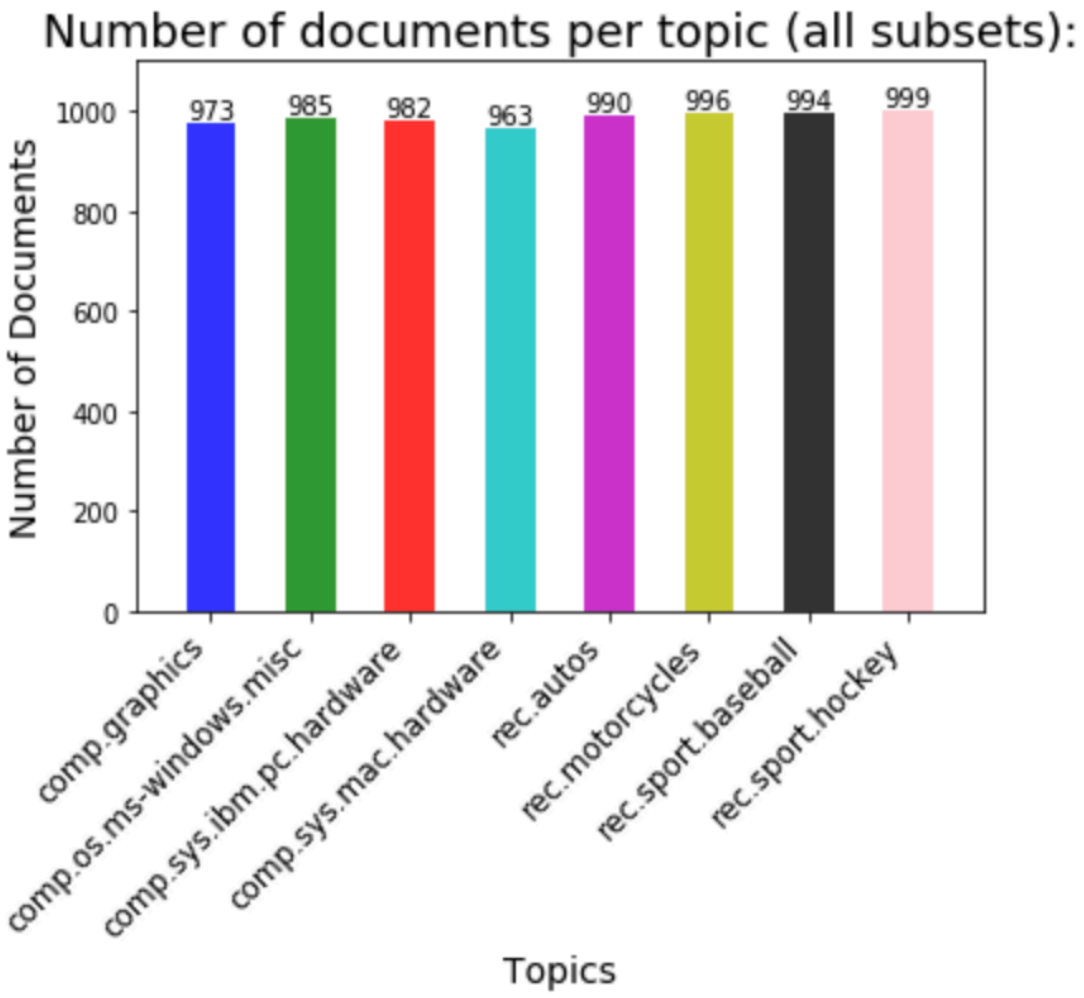
In this project, we aim to train a classifier to group those documents into two classes: Computer Technology and Recreational Activity. The subclasses of these two classes are listed in Table 1.

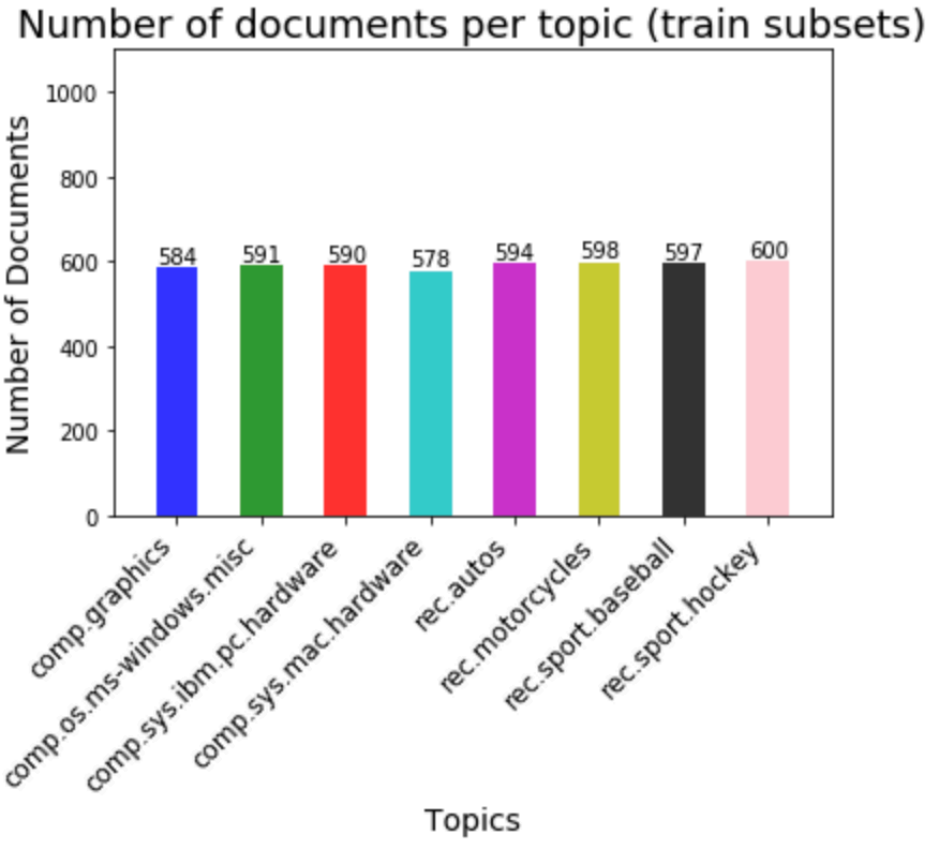
Table 1. Subclasses of 'Computer technology' and 'Recreational activity'

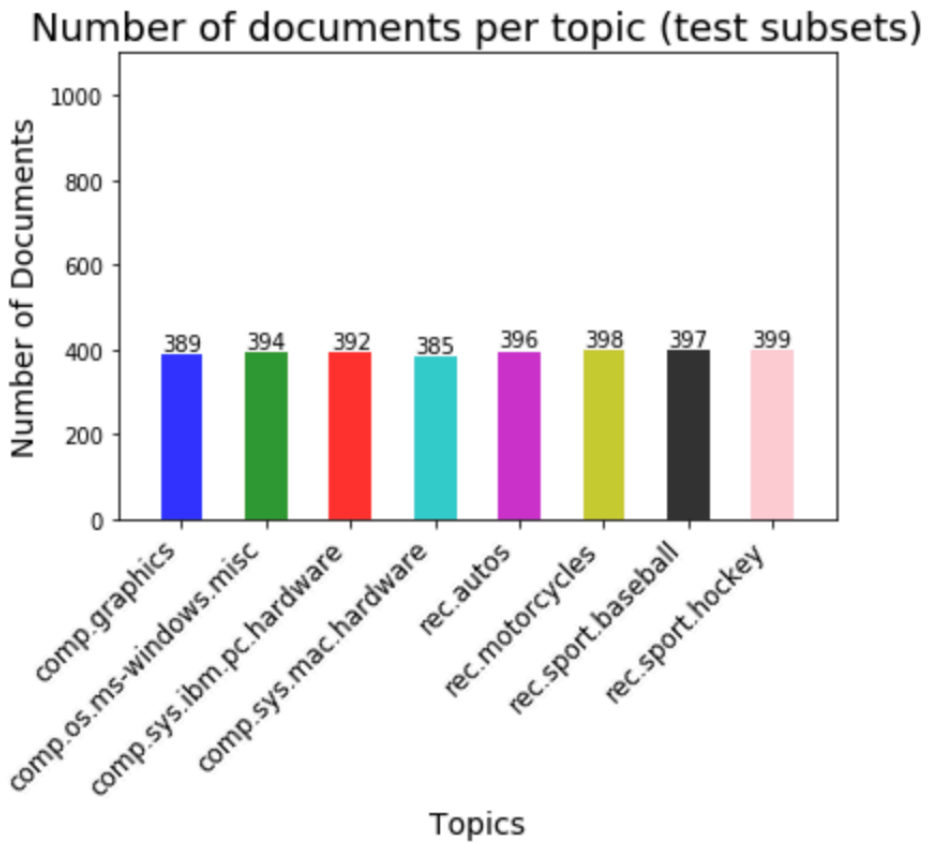
Computer Technology	Recreational Activity
comp.graphics	rec.autos
comp.os.ms-windows.misc	rec.motorcycles
comp.sys.ibm.pc.hardware	rec.sport.baseball
comp.sys.mac.hardware	rec.sport.hockey

Before we start classification, we need to make sure if the sizes of datasets corresponding to different classes are balance or not. In “20 Newsgroup” dataset, there are three subsets: all, train and test. Usually we need to train a classifier with training data and then verify the correctness with testing data. So we load data from these three subsets independently and analyze their size-balance properties. The results are in Question a).

Question a)




Number of documents per topic (train subsets)
Number of Documents
1000
800
600
400
200
0
584
591
590
578
594
598
597
600
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
Topics


Number of documents per topic (test subsets)
Number of Documents
1000
800
600
400
200
0
389
394
392
385
396
398
397
399
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
Topics

The total number in each class are as below:

Table 2. Number of documents in each class

	Computer Technology	Recreational Activity
all subsets	3903	3979
training subsets	2343	2389
testing subsets	1560	1590

From the figures and table, we can find that the dataset is evenly distributed. In figure 1, the number of documents in each topic distributed closely ranged from 960 to 1000. In figure 2, the number of documents per topic in training subsets ranged from 578 to 600. And for testing subsets, the data also distributed closely from 385 to 400. In count table, the number of documents in each class are near to each other for all three subsets. From those statistic data, we can conclude that the dataset is evenly distributed and we can start our classification analysis.

Modeling Text Data and Feature Extraction

In this part, the objective is to find a proper document representation which can represent the content of the document concisely and succinctly. In a document, there are many noise information which may overwhelm our feature extraction and decision-making process, so we need to purify the data in documents and extract the key representation using Term Frequency-Inverse Document Frequency (TFxIDF) metric (show how significant a word is to a document) and TFxICF metric (show how significant a word is to a class).

Question b)

We need to extract key terms of documents and quantify how significant a word is to a document by using TFxIDF. The data process is as follow:

1. Remove the punctuations in original data. Punctuations are meaningless to the content of documents, while they can add much noise to feature extraction, so we need to remove those punctuations. We listed all possible punctuations referring to

ASCII codes, and then split string using regex tokenizer.

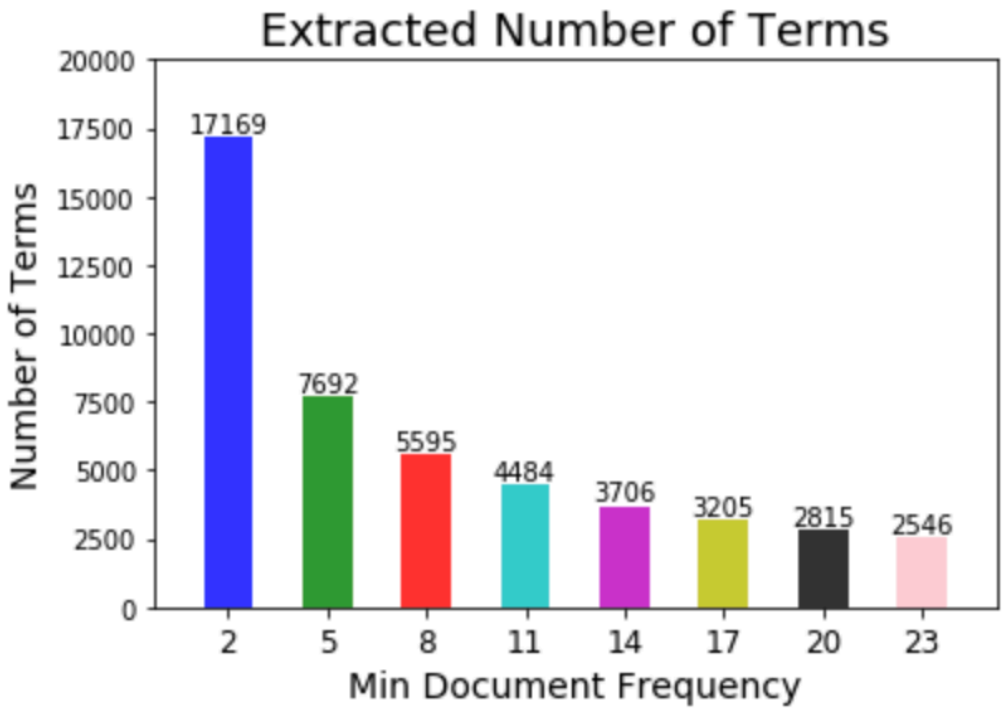
2. Remove newsgroup headers, footer signatures and quotes. Those are irrelevant information to key meaning of the documents.
3. Use stemmed version words. For some words like “play” and “playing”, “Basketball” and “basketball”, they mean same to the key meaning of the document, so we need to convert them to same stemmed words to make accurate count.
4. Exclude stop words. Stop words like “I”, “she”, etc. can appear in all texts and cannot contribute the feature to classify the text, so we need to remove them.
5. Create TFxIDF vector to get feature terms. By setting the min_df parameter, we can set the baseline of appearance that a candidate word need to have. If a word appears less time than min_df, we can ignore it as irrelevant word. The larger the min_df is, the less number of featured terms we will get.

Below are our results:

```
Min Document Frequency: 2
Extracted Number of Terms: 17169

Min Document Frequency: 5
Extracted Number of Terms: 7692
```

we increased min_df and got the number of terms as below:



Question c)

In question b, we quantify how significant a word is to a document. In this part, we quantify how significant a word is to a specific class. Our objective is to find the 10 most significant terms in the following 4 classes:

`comp.sys.ibm.pc.hardware`

`comp.sys.mac.hardware`

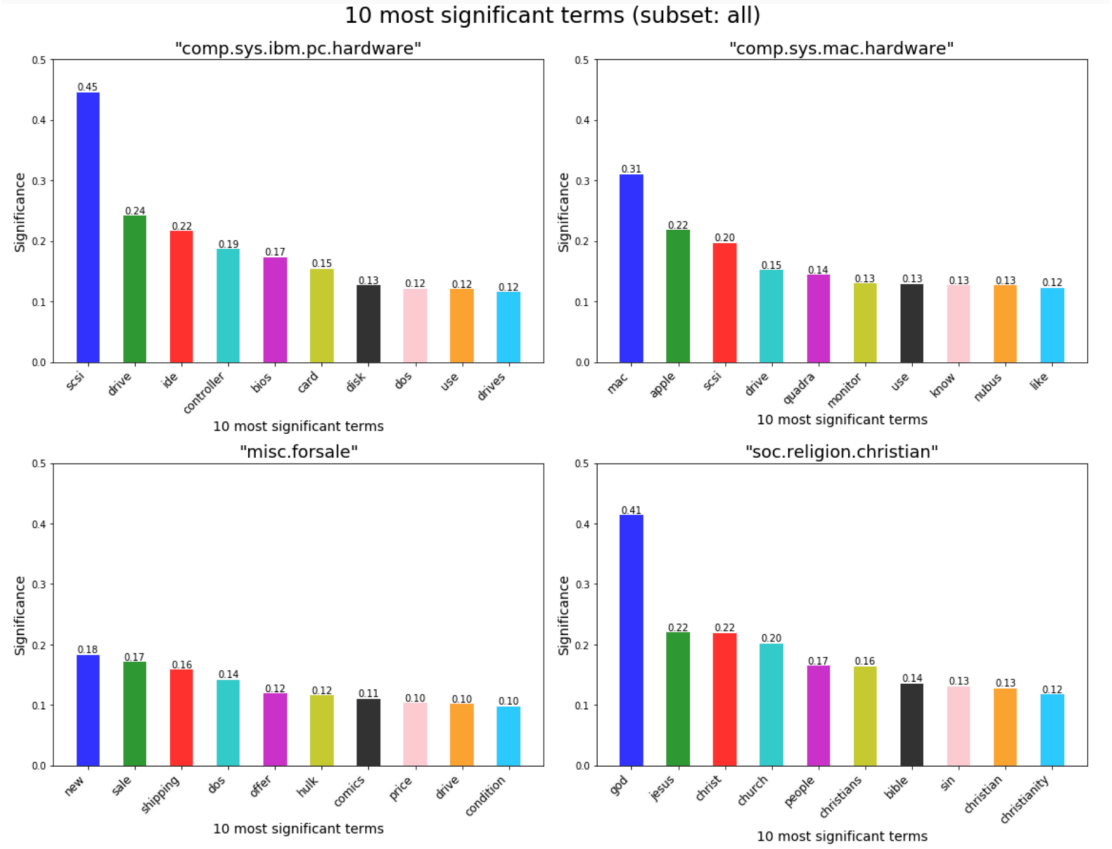
`misc.forsale`

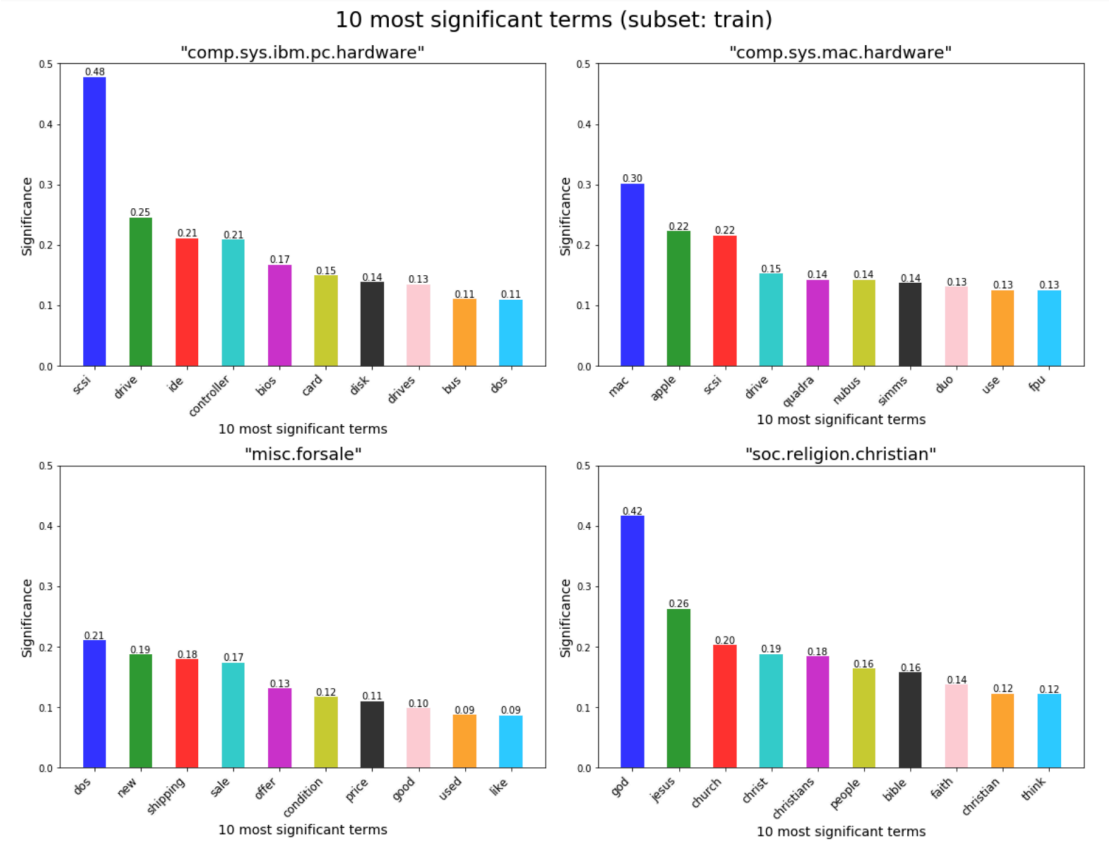
`soc.religion.christian`

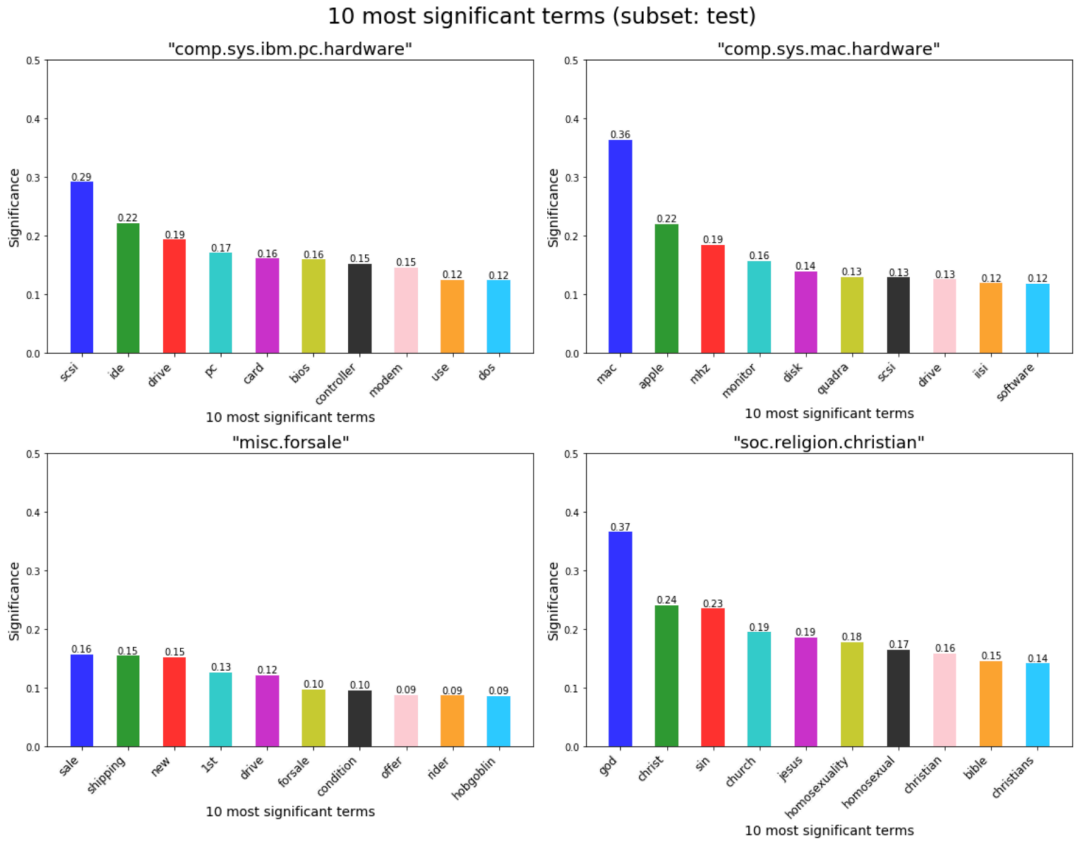
Data processing is the same as that in question b, except we use TFxICF instead of TFxIDF. The process is as below:

1. Pre-processed all documents in 20 newsgroups as question b.
2. Created count vector with documents(rows) – featured terms(columns).
3. Classified and combined documents into 20 subclasses and got count vector with classes(rows) – featured terms(columns).
4. Used TFxICF to quantify the significance of a word to a specific class.
5. For the 4 classes above, we sorted the featured terms and extracted top 10 significant terms respectively.

Below are our results:







From the figure above, we can find some words that are strongly related to the topic. For example, in "`comp.sys.ibm.pc.hardware`", we can find terms like "pc", "dos", "drive", "controller", etc. which are all relevant words to hardware. So we believe that those are significant words. In "`comp.sys.mac.hardware`", we can distinguish "apple", "mac", "drive", etc. which are all related to mac hardware. In "`misc.forsale`", we can find "offer", "sale", "shipping" and so on, which are all relevant words to sale. And in "`soc.religion.christian`", the significant terms include "church", "christian", "jesus" and the like, which all related to Christian topic. Thus, we can conclude that these words are the most significant terms corresponding to each topic.

Feature Selection

Now, we got TFxIDF matrix with over thousands of dimensions. The matrix is sparse and may diminish the performance of many learning algorithms. Thus, in the next step, we need to find a way to reduce its dimension.

Question d)

In this project, we use two methods to reduce dimension and compare their effects. One is Latent Semantic Indexing (LSI), the other is Non-Negative Matrix Factorization (NMF).

1. Latent Semantic Indexing (LSI)

   In general, words appeared in the same context tend to have similar or close meanings. LSI method can construct the relationship among words in the same context and use this relationship to identify the meaning of the context. LSI method is obtained by Singular Value Decomposition (SVD) of matrices.

   Let D represents the TFxIDF matrix, then the SVD of D is

   $$D = U\Sigma V^T$$

   The larger the singular value is, the more information it contains. So we can truncate the singular value matrix and get truncated SVD

   $$D_k = U_k \Sigma_k {V_k}^T$$

   Here, we pick $k = 50$ to project document column vectors into a 50-dimentional representation.

2. Non-Negative Matrix Factorization (NMF)

   $$\min_{W,H} \|X - WH\|_F^2$$
   $$\text{s.t. W} \geq 0$$
   $$\text{H} \geq 0$$

Below are our results:

```
LSI method:
(4732, 50)

NMF method:
(4732, 50)
```

The results using two methods are same, and the TFxIDF matrix has been reduced to 50 dimensions.

Learning Algorithms

It's efficient to use linear Support Vector Machines to learn the feature and classify the data. We can directly use the built-in class SVC in scikit-learn package to train the data.

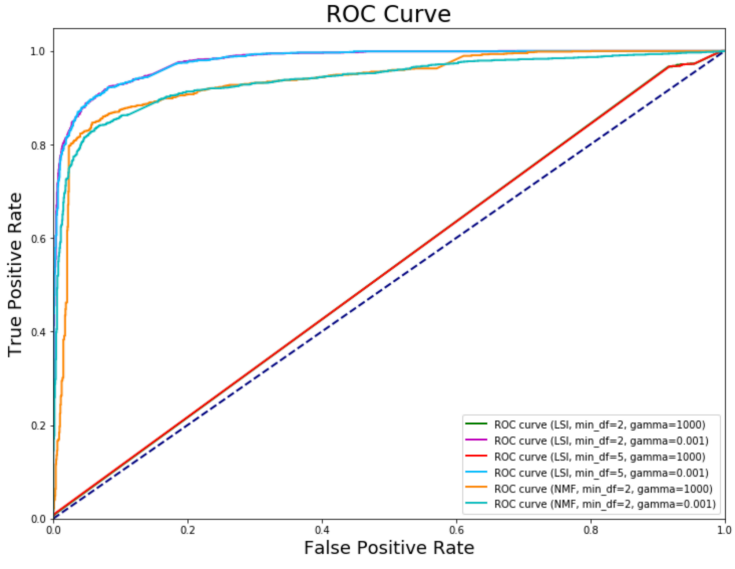
Question e)

Data processing are as below:

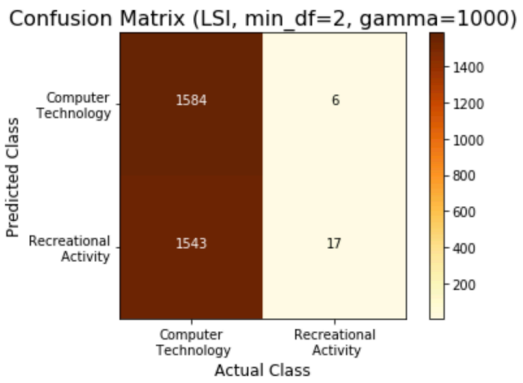
1. Load train and test data of 8 subclasses.
2. Pre-process data as in question b, including removing punctuations, stop words, using stemmed version words, etc.
3. Create TFxIDF matrix and use LSI and NMF methods to do feature extraction.
4. Separate training and testing datasets by labeling the target data as "True" to represent Computer Technology class and "False" to represent Recreational Activity class.
5. Use training dataset to train a SVM classifier with `svm.SVM()`.
6. Use testing dataset to test our classification model.
7. Plot the ROC curve and report the confusion matrix, accuracy, recall and precision of our classifier. Compare the results of LSI and NMF with different min_df parameters and compare the results using hard margin SVC and soft margin SVC.

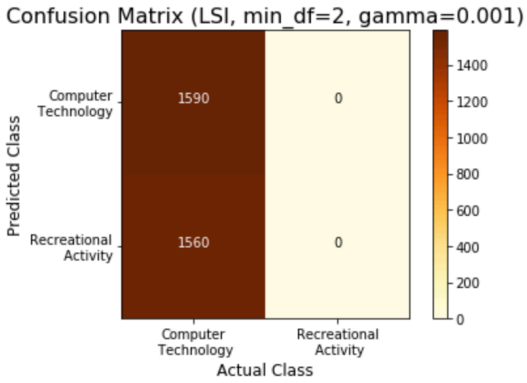
Below are our results:

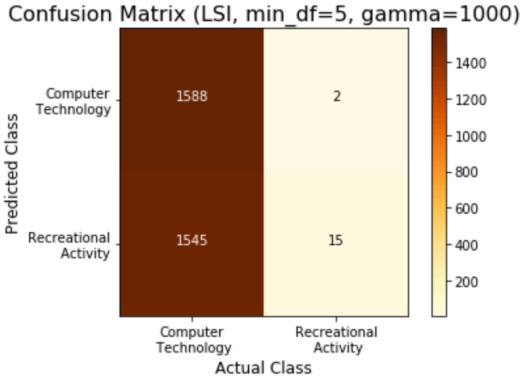
1. ROC curve

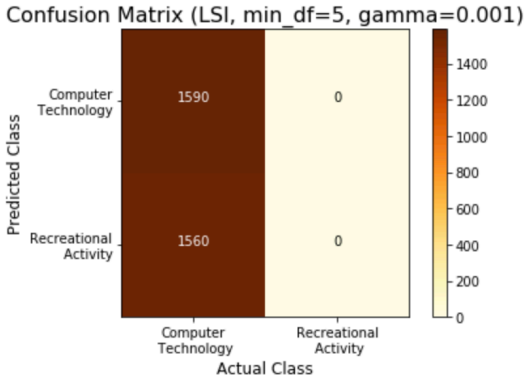


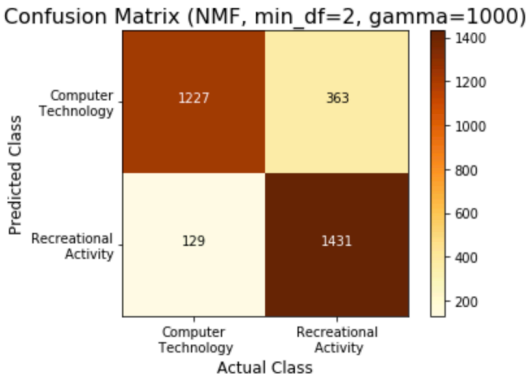
2. Confusion matrix

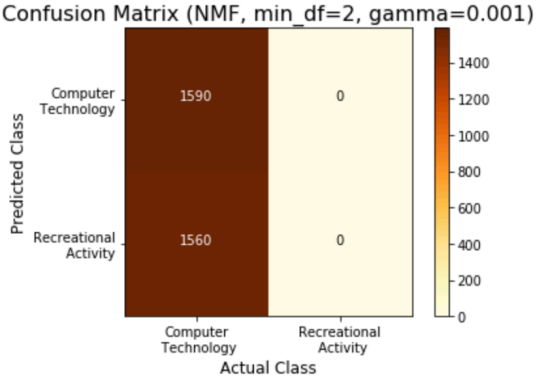












3. Accuracy, recall and precision

Table 3. accuracy, recall and precision of classifier (hard margin SVC, $\gamma = 1000$)

method	min _df	accuracy	recall	precision
LSI	2	0.508	0.011	0.739
	5	0.509	0.010	0.882
NMF	2	0.844	0.917	0.798

Table 4. accuracy, recall and precision of classifier (soft margin SVC, $\gamma = 0.001$)

method	min _df	accuracy	recall	precision
LSI	2	0.505	0	0
	5	0.505	0	0
NMF	2	0.505	0	0

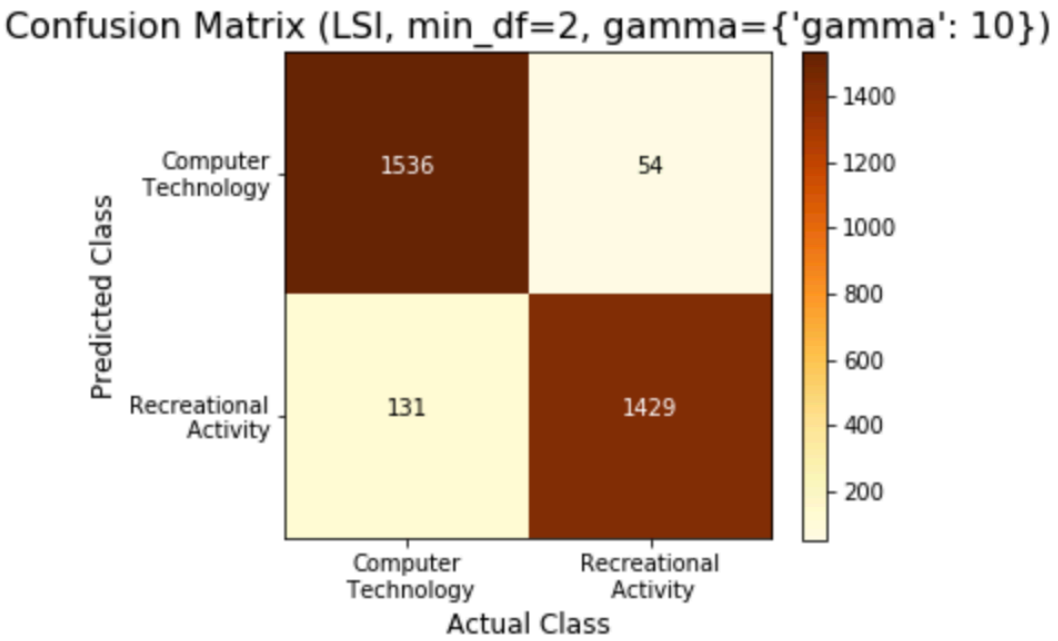
From the figure and table above, we can see that using NMF method, and setting min_df to 2, letting $\gamma = 1000$, we can get better results among those six experiments. Parameter min_df has a few impacts on performance of classification. Hard margin SVC performed better than soft margin SVC.

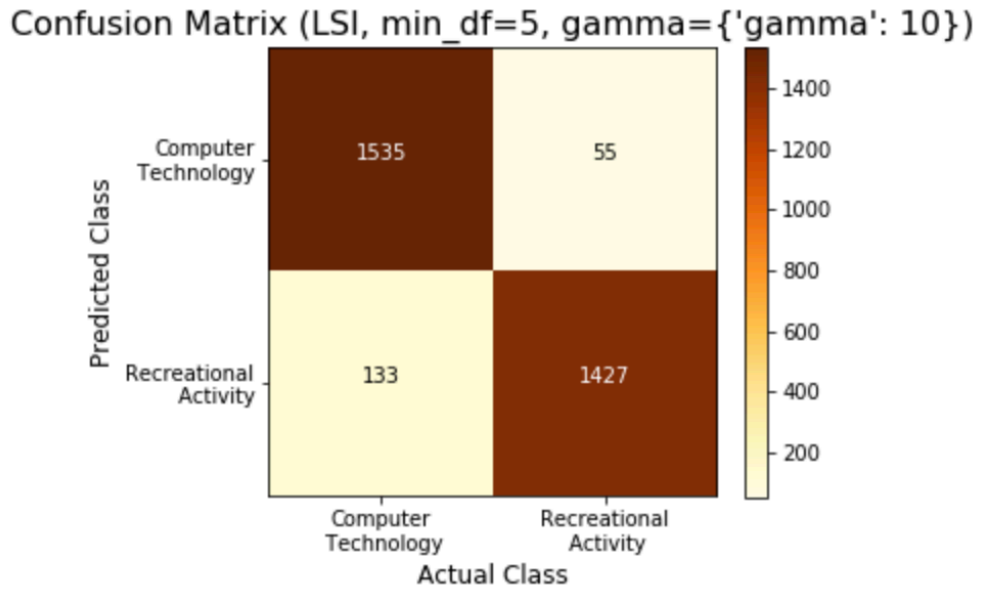
Question f)

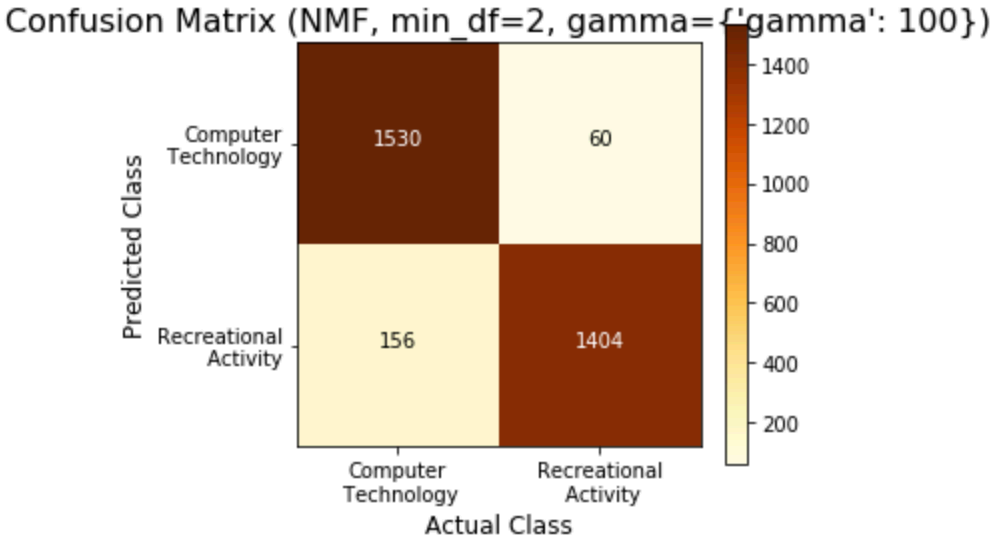
In this part, we separate all documents with 5-fold cross-validation, and then find the best value of parameter γ in the range $\{10^{-k} | -3 \leq k \leq 3, k \in Z\}$.

With the same method and min_df parameter settings, we test gamma for $[0.001, 0.01, 0.1, 1, 10, 100, 1000]$ and find the best value of parameter γ for three cases. Below are our results:

1. Confusion matrix







2. Accuracy, recall, precision

Table 5. accuracy, recall and precision of best classifier

method	min _df	γ_{opt}	accuracy	recall	precision
LSI	2	10	0.941	0.916	0.964
	5	10	0.940	0.915	0.963
NMF	2	100	0.931	0.900	0.960

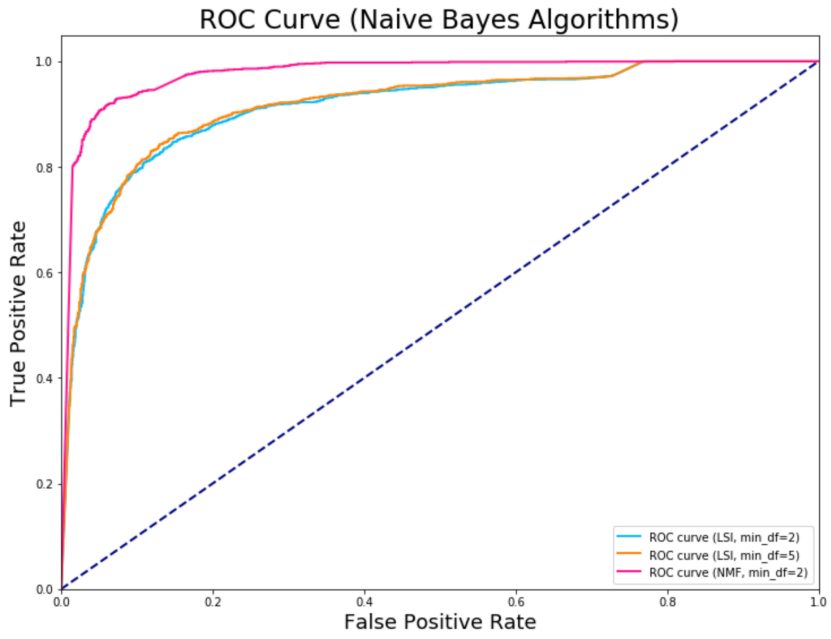
From the figure and table above, we can see that the best value of parameter γ depends on cases. For LSI method, the optimal γ is 10, and for NMF method, the optimal γ is 100.

Question g)

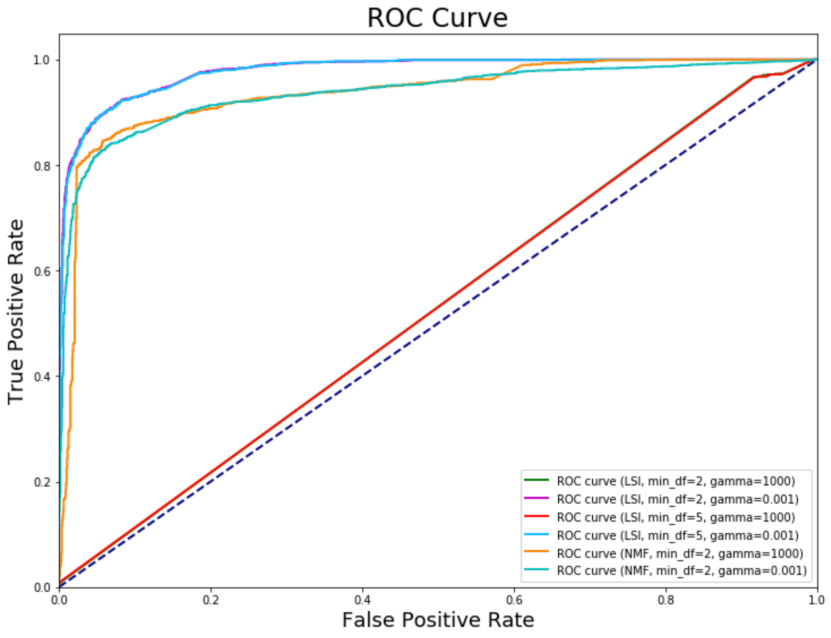
In this part, we use Naïve Bayes Algorithms instead of SVM to do the classification as in question e). We can use the built-in class `sklearn.GaussianNB` to predict classification. Below are our results:

1. ROC Curve

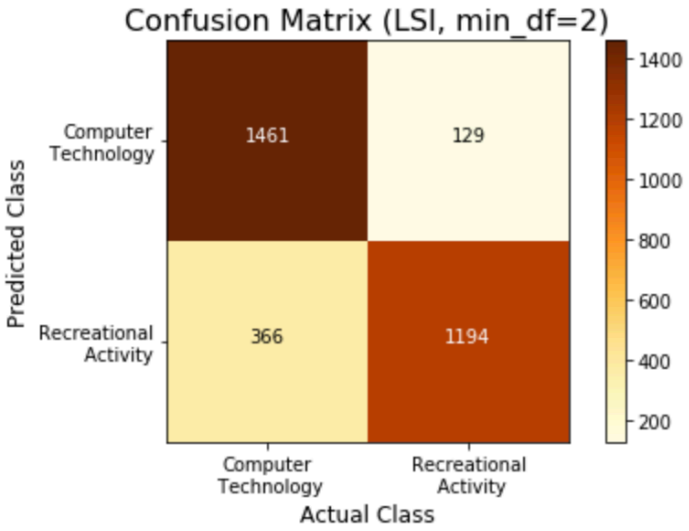
   Naïve Bayes Algorithms classifier:

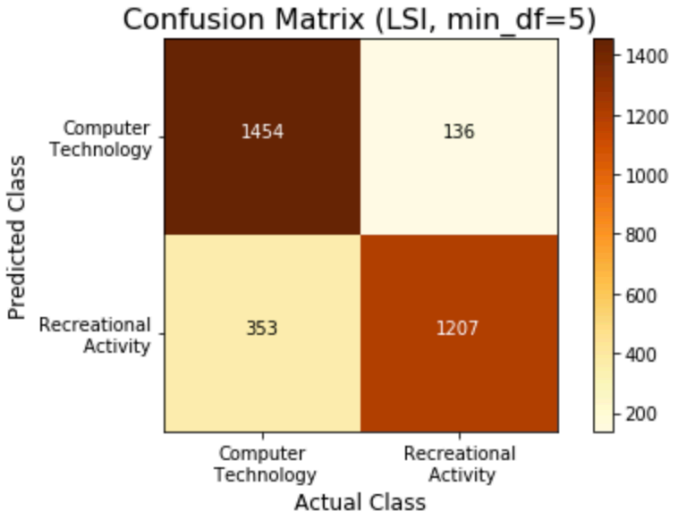


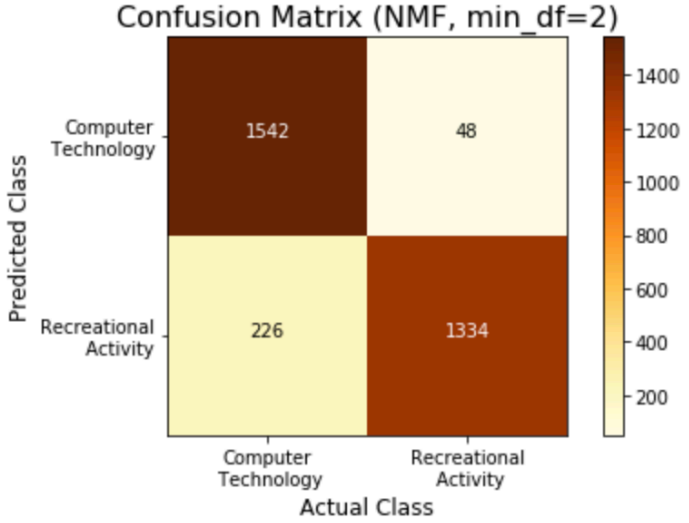
   SVM classifier:



2. Confusion matrix







3. Accuracy, recall, precision

Table 6. accuracy, recall and precision of classifier

method	min _df	accuracy	recall	precision
LSI	2	0.843	0.765	0.902
	5	0.845	0.774	0.899
NMF	2	0.913	0.855	0.965

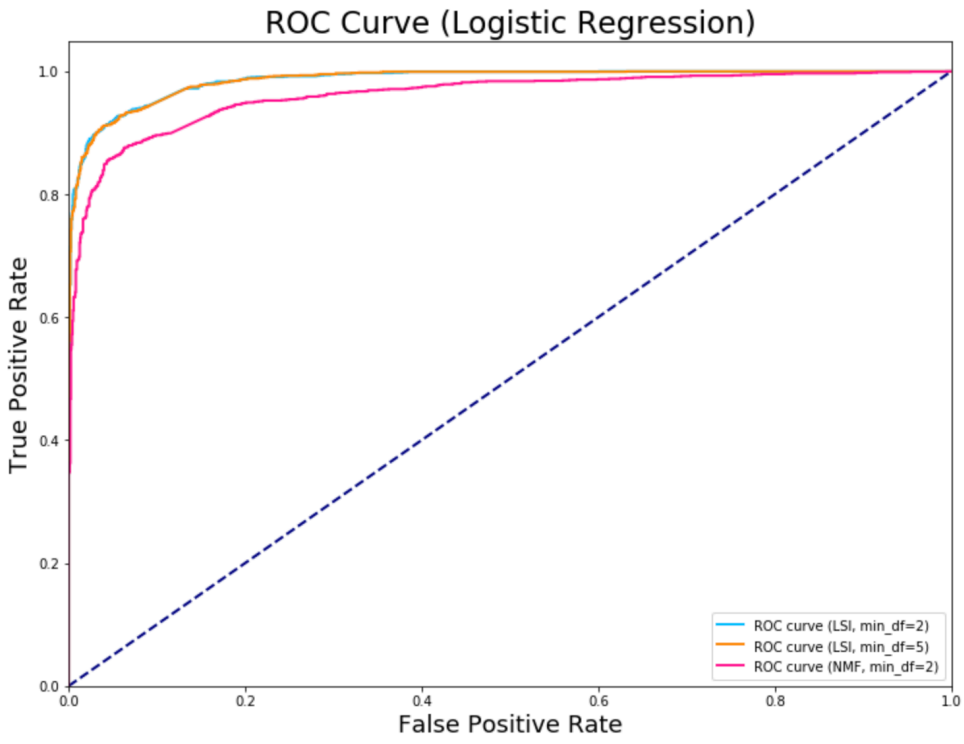
From the figures and table above, we can see that Naïve Bayes Algorithms performs better than SVM with higher accuracy, recall and precision generally since it's not restricted to trade-off parameter γ.

Question h)

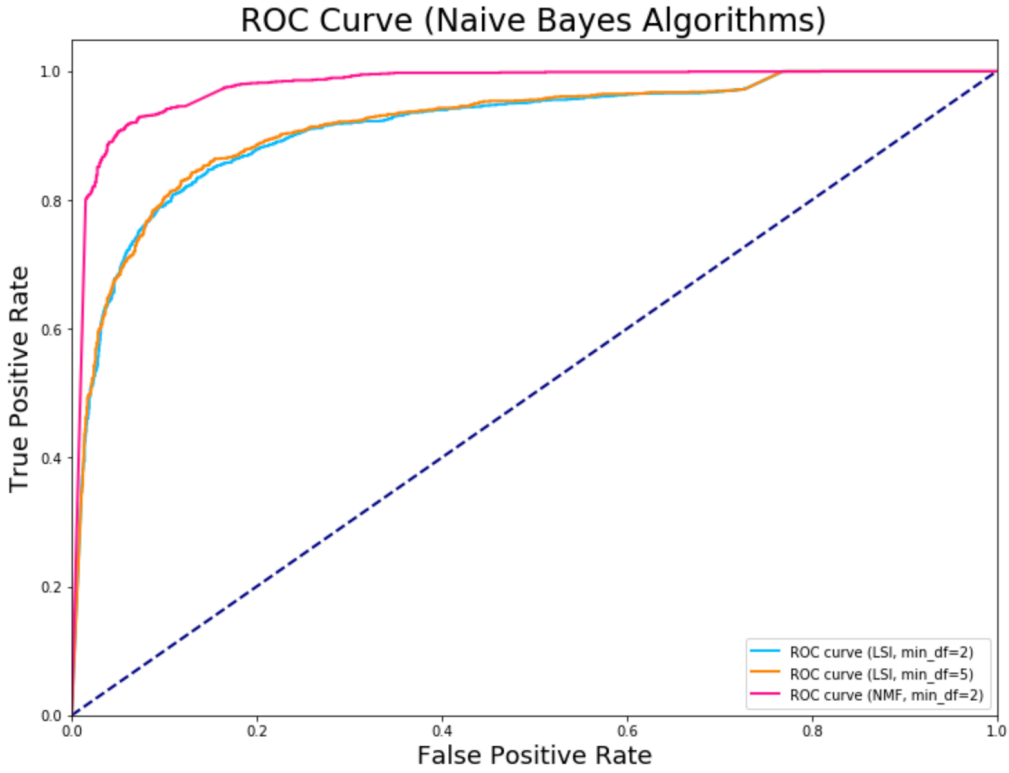
In this part, we use Logistic Regression instead of SVM to do the classification as in question e). In question h, we explore the general l2 regularized logistic regression with liblinear as the solver. Below are our results:

1. ROC Curve

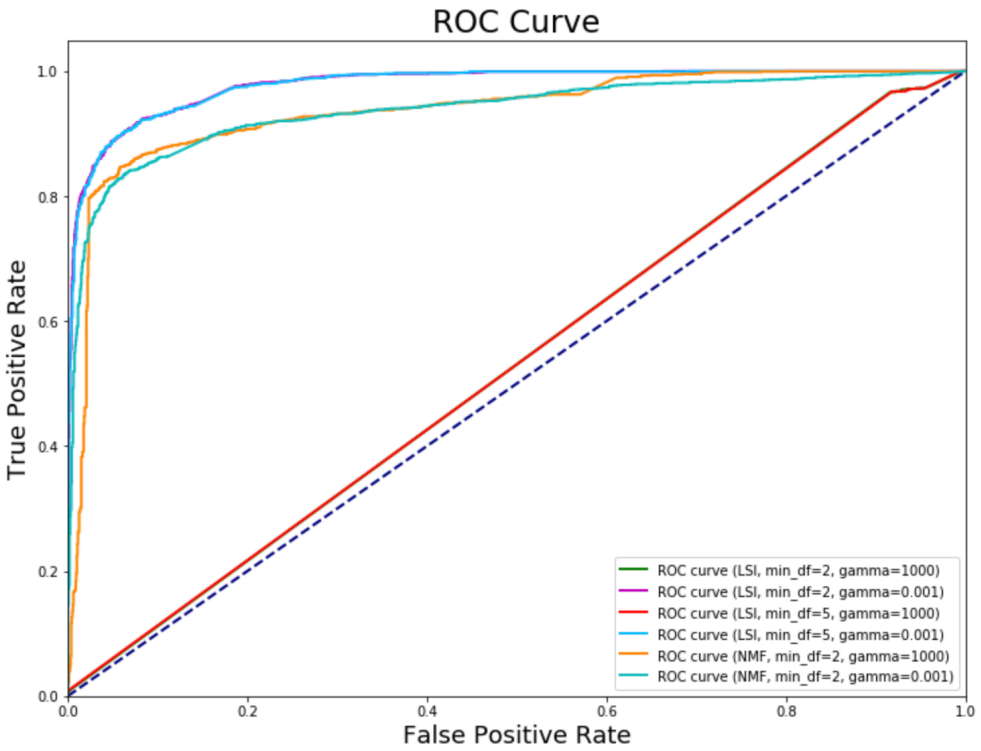
   Logistic Regression:



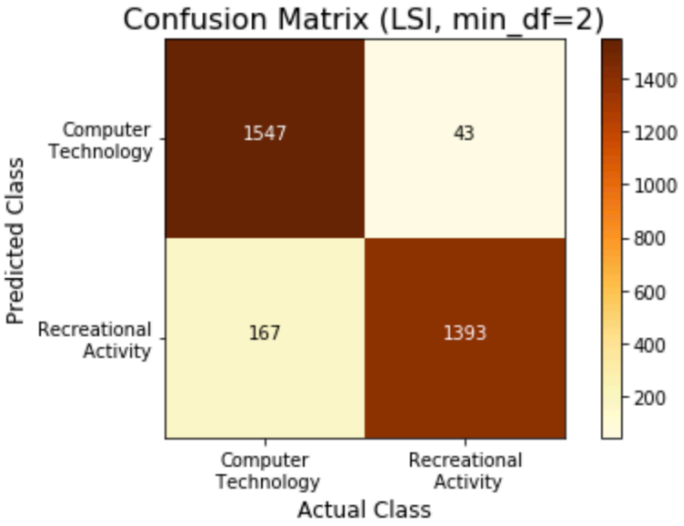
Naïve Bayes Algorithms classifier:

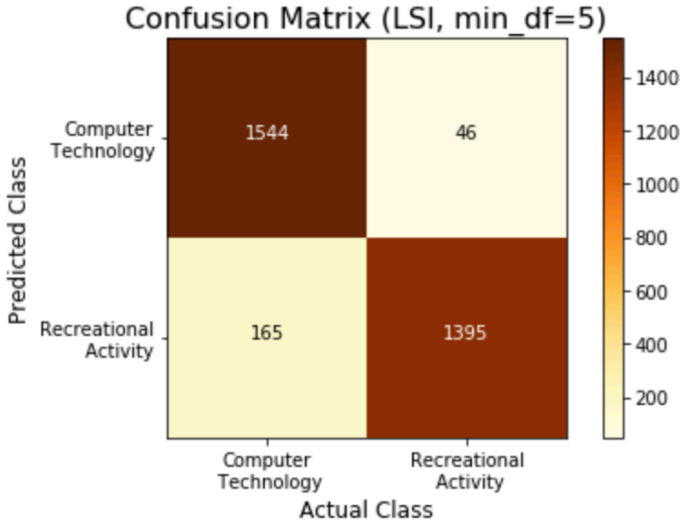


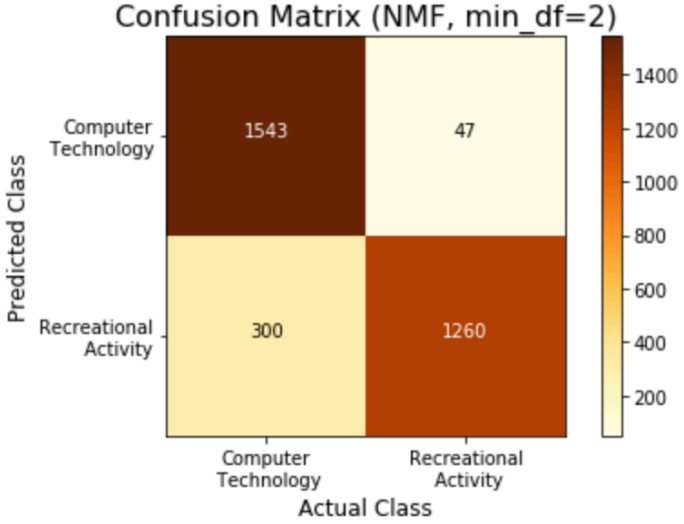
SVM classifier:



2. Confusion matrix







3. Accuracy, recall, precision

Table 7. accuracy, recall and precision of classifier

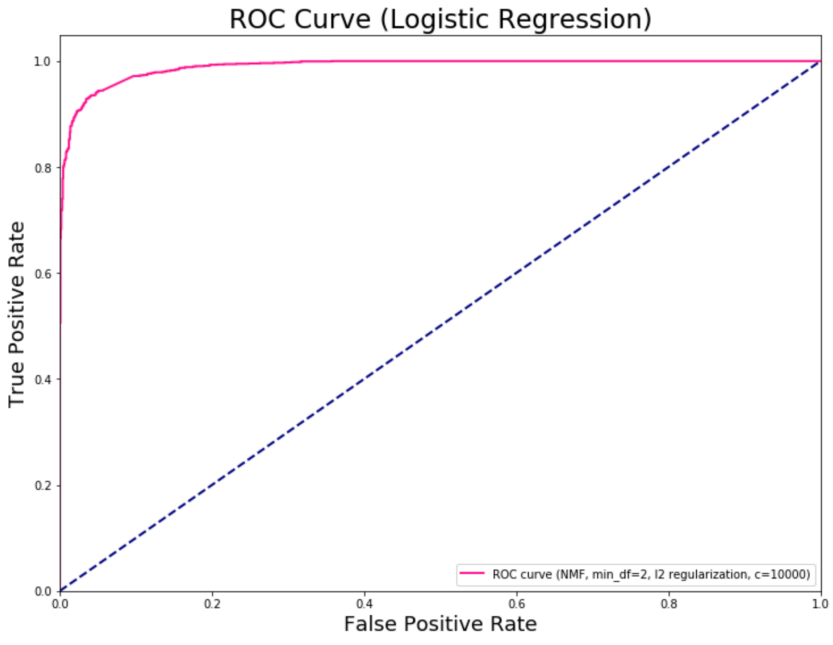
method	min _df	accuracy	recall	precision
LSI	2	0.933	0.893	0.970
	5	0.933	0.894	0.968
NMF	2	0.890	0.808	0.964

From the figures and table above, we can see that among SVM classifier, Naïve Bayes classifier and Logistic Regression classifier, Logistic Regression performs a little better than the other two with higher accuracy, recall and precision values. Since SVM is restricted to the value of trade-off parameter γ, the other two classifiers perform more static than SVM. Besides, from all three classifiers, we find that the computer technology class has higher probability to be recognized correctly in general.

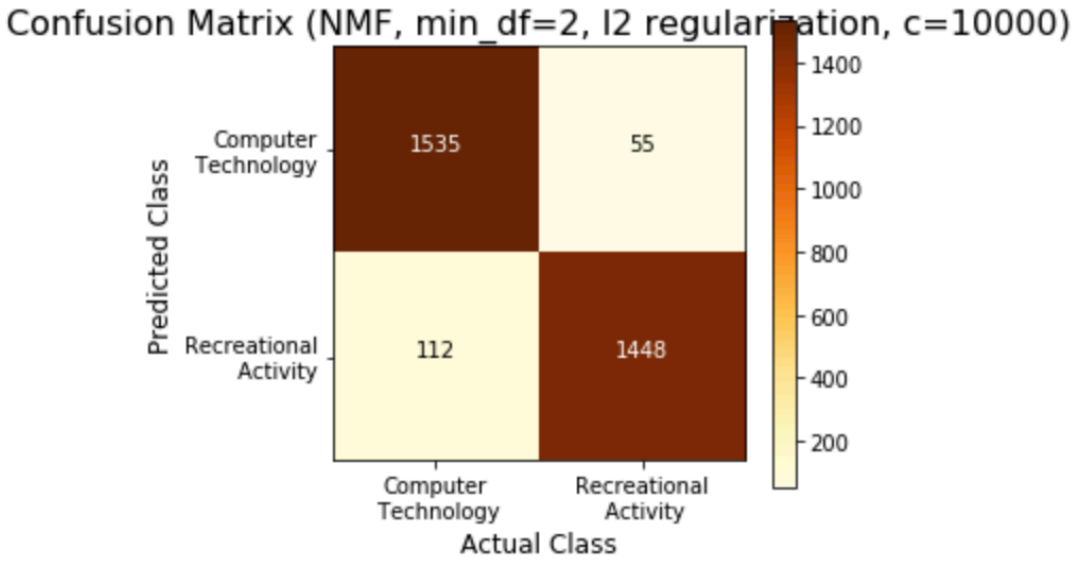
Question i)

In this part, we explore the performance of l1 and l2 norm regularized logistic regression setting with different coefficient values in a set of $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$. Below are our results:

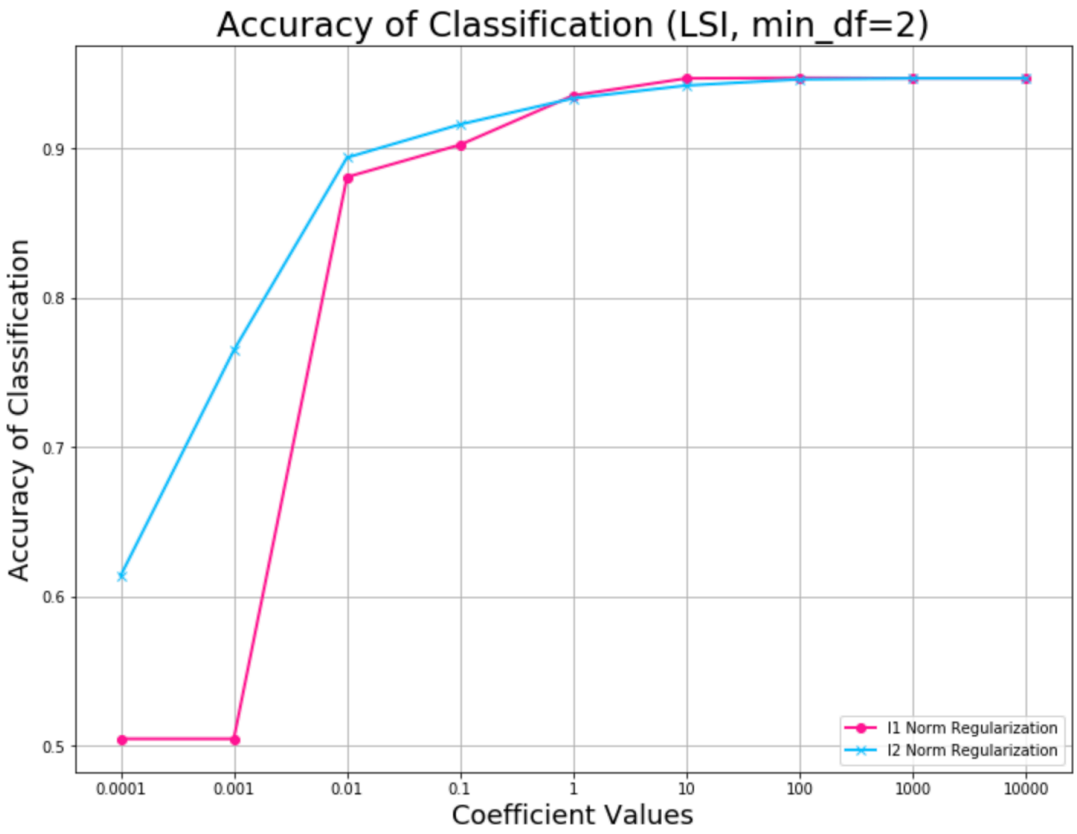
1. One of ROC Curves

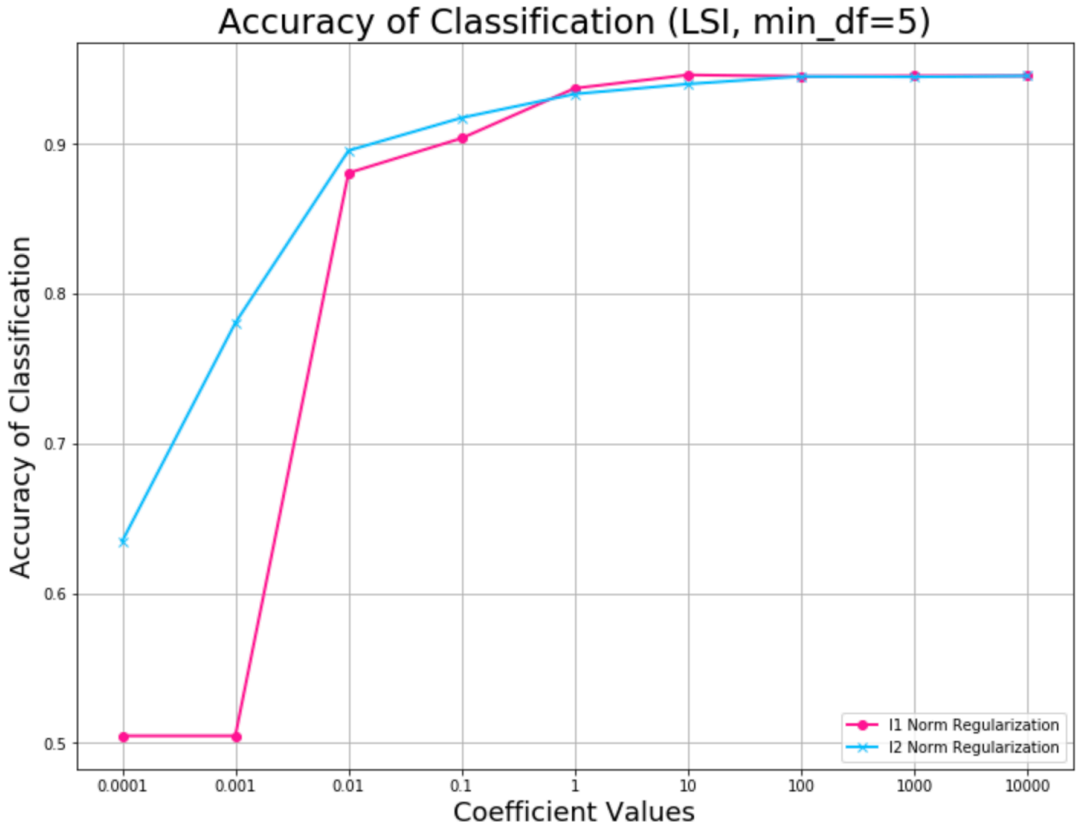


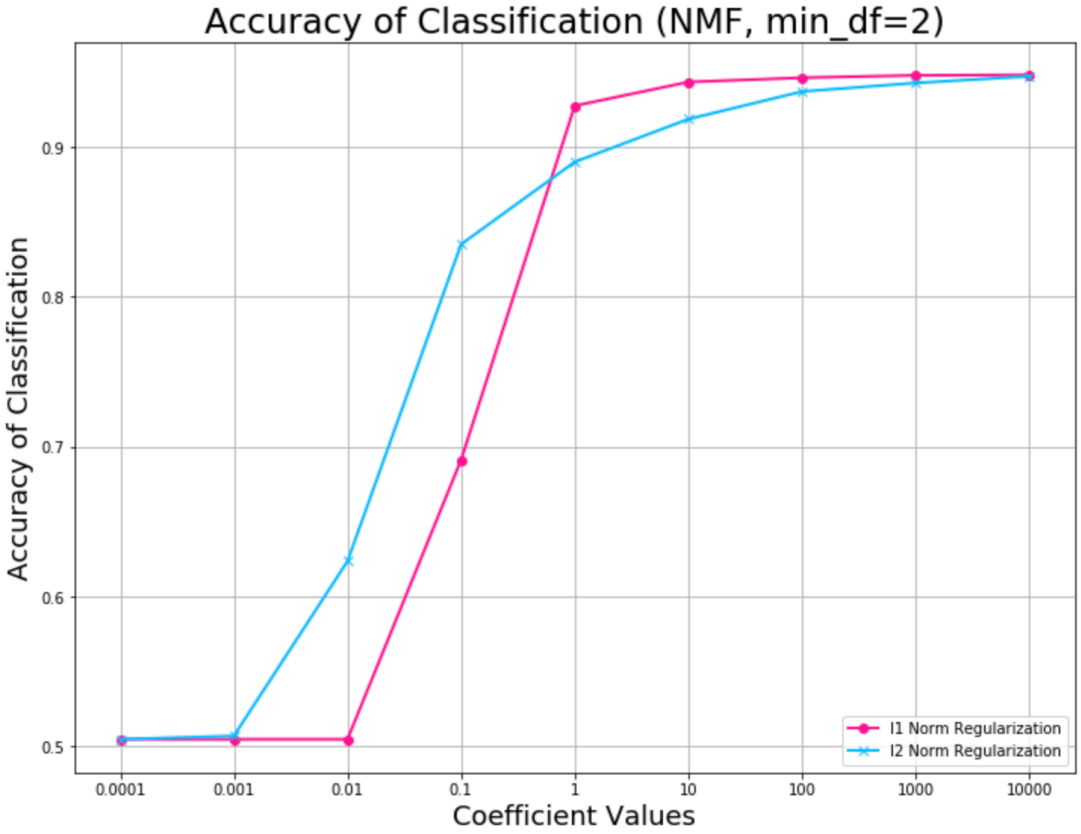
2. One of confusion matrices



3. Accuracy compares




Accuracy of Classification (LSI, min_df=5)
Accuracy of Classification
0.9
0.8
0.7
0.6
0.5
0.0001
0.001
0.01
0.1
1
10
100
1000
10000
Coefficient Values
I1 Norm Regularization
I2 Norm Regularization


Accuracy of Classification (NMF, min_df=2)
Accuracy of Classification
0.9
0.8
0.7
0.6
0.5
0.0001
0.001
0.01
0.1
1
10
100
1000
10000
Coefficient Values
I1 Norm Regularization
I2 Norm Regularization

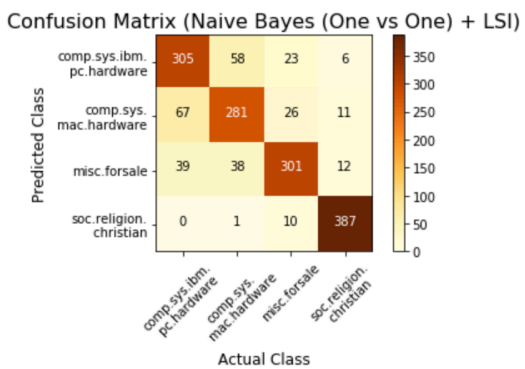
From the figures above, we can see that when coefficient is at very small values, the accuracy of classification is very low and all documents seem to be classified as computer technology class. As the coefficient increasing, the accuracy of classification raised up and achieved good accuracy. Specifically, for LSI dimension-reducing method, when the coefficient grows larger than 0.01, the classification accuracy increases to an acceptable range. For NMF dimension-reducing method, when the coefficient grows larger than 1, the accuracy becomes acceptable. And at last, the accuracy grows to a stable level at around 0.95 for both l1 and l2 norm regularization.

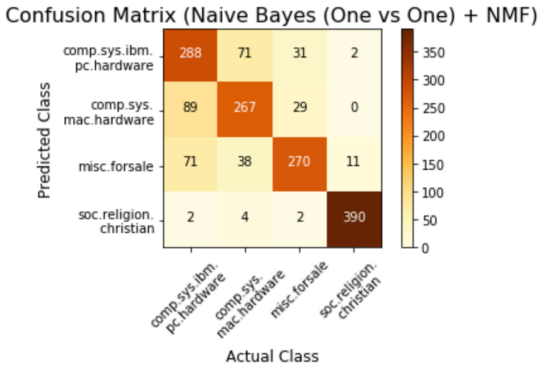
However, it's still hard to tell which one is better because it depends on cases. L1 regularization can drive one or more weight values to zero so that some insignificant features would not be selected. Besides, L1 regularization seems to be more robust. While L2 regularization can suppress the weight value but not to zero completely, it performs better for unstable datasets.

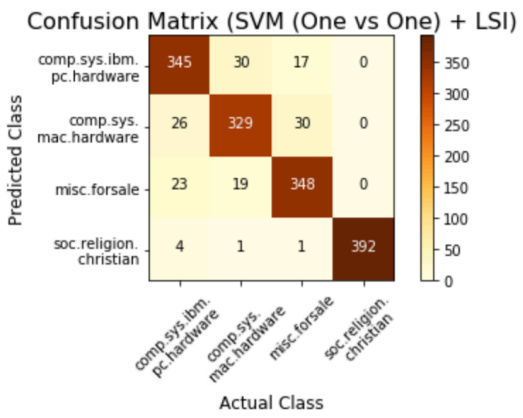
Multiclass Classification

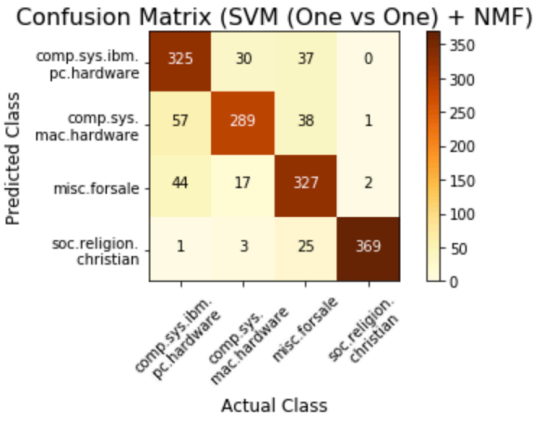
If the dataset to be classified belongs to multiclass, there usually are two methods to deal with. One way is to perform one versus one classification on all pairs of classes, and given a document the class is assigned with the majority vote. An alternative way is to fit one classifier per class. Then, for each classifier, the class is fitted against all the other classes. Below are our results:

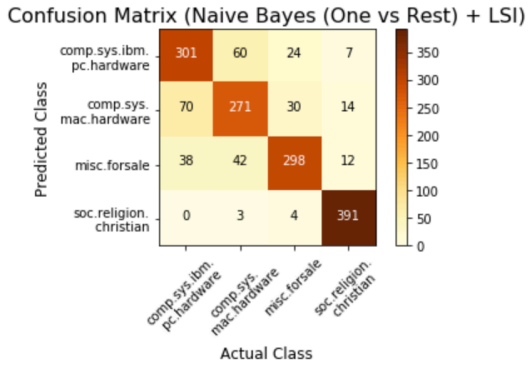
1. Confusion matrix

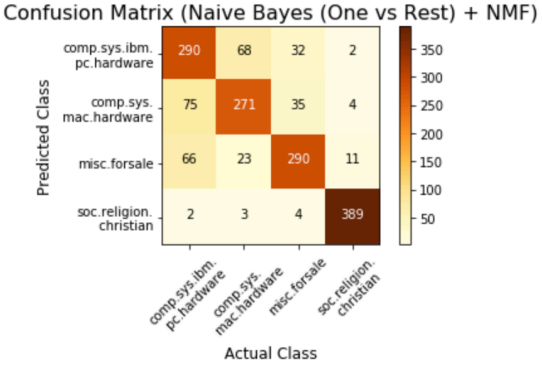


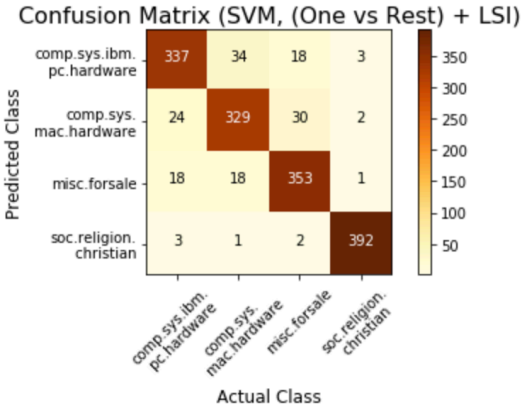


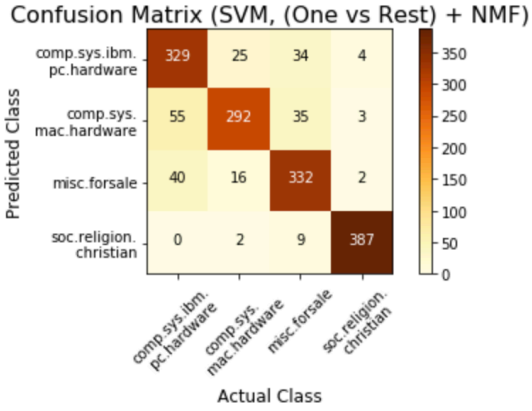

Confusion Matrix (SVM (One vs One) + LSI)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
345 30 17 0
26 329 30 0
23 19 348 0
4 1 1 392
Predicted Class
Actual Class
350 300 250 200 150 100 50 0


Confusion Matrix (SVM (One vs One) + NMF)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
325 30 37 0
57 289 38 1
44 17 327 2
1 3 25 369
Predicted Class
Actual Class
350 300 250 200 150 100 50 0


Confusion Matrix (Naive Bayes (One vs Rest) + LSI)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
301 60 24 7
70 271 30 14
38 42 298 12
0 3 4 391
Predicted Class
Actual Class
350 300 250 200 150 100 50 0


Confusion Matrix (Naive Bayes (One vs Rest) + NMF)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
290 68 32 2
75 271 35 4
66 23 290 11
2 3 4 389
Predicted Class
Actual Class
350 300 250 200 150 100 50


Confusion Matrix (SVM, (One vs Rest) + LSI)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
337 34 18 3
24 329 30 2
18 18 353 1
3 1 2 392
Predicted Class
Actual Class
350 300 250 200 150 100 50


Confusion Matrix (SVM, (One vs Rest) + NMF)
comp.sys.ibm. pc.hardware
comp.sys. mac.hardware
misc.forsale
soc.religion. christian
329 25 34 4
55 292 35 3
40 16 332 2
0 2 9 387
Predicted Class
Actual Class
350 300 250 200 150 100 50 0

2. Accuracy, recall, precision

Table 8. accuracy, recall and precision of classifier (min_df = 2)

classifier	method	dimension-reduce	accuracy	recall	precision
Naïve Bayes	One vs One	LSI	0.814	0.81	0.81
Naïve Bayes	One vs One	NMF	0.776	0.78	0.78
SVM	One vs One	LSI	0.904	0.90	0.90
SVM	One vs One	NMF	0.837	0.84	0.84
Naïve Bayes	One vs Rest	LSI	0.806	0.81	0.80
Naïve Bayes	One vs Rest	NMF	0.792	0.79	0.79
SVM	One vs Rest	LSI	0.902	0.90	0.90
SVM	One vs Rest	NMF	0.856	0.86	0.86

In this part, we performed Naïve Bayes classification and multiclass SVM classification with both One VS One and One VS the rest methods on two datasets preprocessed by LSI and NMF separately. From the results, we can find that Naïve Bayes classification is a little bit more efficient than SVM.