EE219 Project 1

Classification Analysis on Textual Data

Winter 2018

Xinyi Gu 205034975

Yutan Gu 005034976

Introduction

In this project, we worked on the textual classification on the datasheet "20 Newsgroups". The datasheet contains nearly 20,000 documents which partitions into 20 different newsgroups. Our goal is to extract the feature from these documents, and to use learning algorithms to classify them into their topic correctly. The learning algorithms we used in this project includes Linear Support Vector Machines(SVM), Naive Bayes, and Logistic Regression.

Problem (A)

In this project, we mainly dealed with 8 of 20 classes in the datasheet. These 8 newsgroups can be grouped into 2 major classes "Computer Technology" and "Recreational Activity". Table 1 is the names of 8 classes.

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"

Before the process of classification, it is important to check the sizes of the datasets corresponding to different topics. In order to make sure the sizes of each data groups is relatively even, we plotted a histogram of the numbers of documents per topic. Shown in Fig.1, we can see that the data of each groups is already balanced.

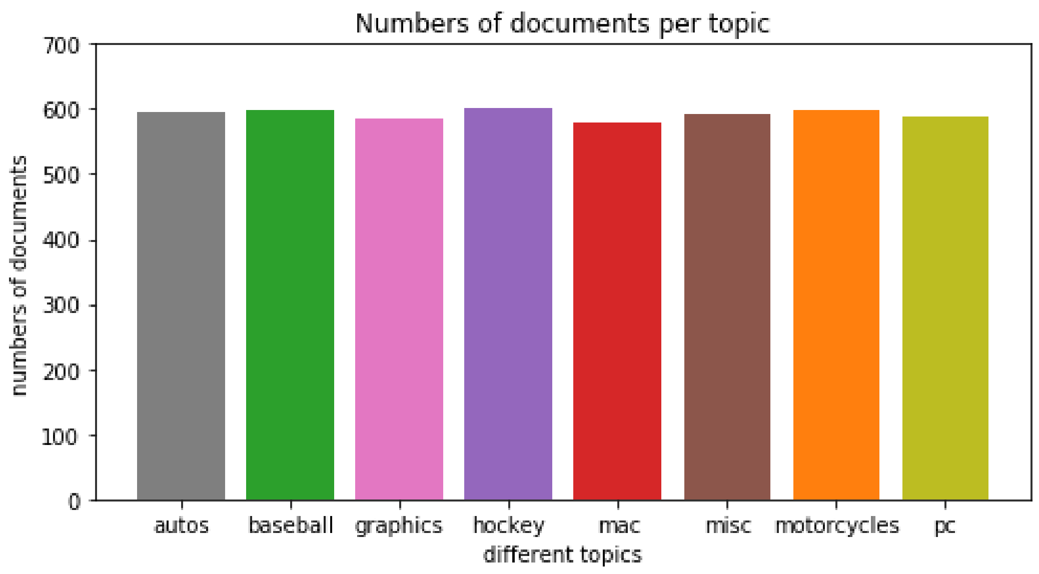


Figure 1. Numbers of Documents per Topic

Problem (B)

The first step of the classification is to remove unnecessary information from the documents to avoid large feature vectors. First, we dropped all the punctuations from the documents since they have no contributions to the feature extraction. Then we removed all the stop words and different stem of the words in the documents to reduce the dimension of the feature vectors. After removing all relevant information, we then converted all documents into a term-document matrix representation. We chose TFxIDF metric because it measures the importance of a term to the document. We used two different minimum document frequency of vocabulary terms, min_df = 2 and min_df = 5. For min_df = 2, the shape of the TFxIDF is (4732, 16449), which means the final number of terms we extracted was 16449. For min_df =5, the final number of terms we got was 6910.

Problem (C)

Similar to problem (B), to find the 10 most significant terms in each of the classes: “comp.sys.ibm.pc.hardware”, “comp.sys.mac.hardware”, “misc.forsale”, “soc.religion.christian”, we first removed all punctuations, stop words and stemmed all the terms in the documents. Then we built the TFxICF matrix and found the 10 most significant terms of each group. The result is sorted and shown in the following tables. Comparing two tables, we can find that the small difference between minimum document frequency of vocabulary terms does not affect the result. The reason is that the 10 most significant terms are 10 terms which has the most counts, and the exclusion of low frequency terms will not remove them.

10 most significant terms(min_df = 2)			
comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
ani	ani	00	ani
card	appl	10	believ
control	card	50	christian
disk	drive	dos	church
drive	know	includ	god
ide	like	new	jesus
problem	mac	offer	know
scsi	problem	price	peopl
use	use	sale	say
work	work	use	think

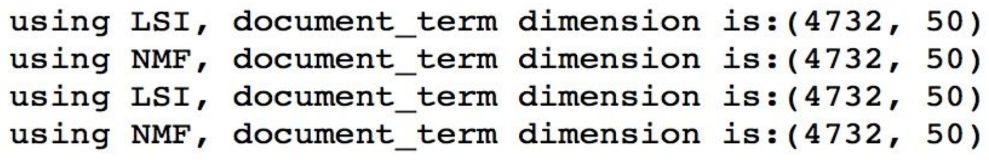
Table 2. 10 most significant terms in each topic(min_df = 2)

10 most significant terms(min_df = 5)			
comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
ani	ani	00	ani
card	appl	10	believ
control	card	50	christian
disk	drive	dos	church
drive	know	includ	god
ide	like	new	jesus
problem	mac	offer	know
scsi	problem	price	peopl
use	use	sale	say
work	work	use	think

Table 3. 10 most significant terms in each topic(min_df = 5)

Problem (D)

Since the TFxIDF matrix is sparse and low rank, we used Latent Semantic Indexing (LSI) to reduce the dimension of the matrix. To apply the LSI, we calculated the Truncated SVD of the matrix, and transfer the matrix to a 50 dimension vector. Alternatively, we also used Non-negative Matrix Factorization(NMF) to map the documents into a 50 dimension vector. Under two min_df, both of the two methods successfully reduced the dimension of TFxIDF. We printed out the shape of the resulted matrix to verify the answer:



```
using LSI, document_term dimension is:(4732, 50)
using NMF, document_term dimension is:(4732, 50)
using LSI, document_term dimension is:(4732, 50)
using NMF, document_term dimension is:(4732, 50)
```

Figure 2. Reduced dimension matrix of TFxIDF

Problem (E)

Support Vector Machine is proved to be an efficient classifier for textual data. Basically, it generalizes a hyperplane that tries to maximize the margin of data points, which provides great classification ability. In Linear SVM, a tradeoff parameter gamma is used to control how much penalty will be applied for a mis-classified data. When gamma is relatively large, it means that the mis-classified data will be highly penalized. Otherwise, the mis-classified data is more likely to be tolerable when gamma is small. These different situations are called Hard SVM and Soft SVM relatively.

We use svm.SVC function from scikit-learn package to implement a typical SVM classifier. Setting the kernel to ‘linear’ gives us a linear SVM model. The parameter ‘C’ refers to the gamma in the previous discussion. For each type of SVM, we apply them towards three types of feature - LSI&mindf=2, LSI&mindf=5 and NMF&mindf=2. Similar procedure is used as what we did in the previous parts. The results are shown below:

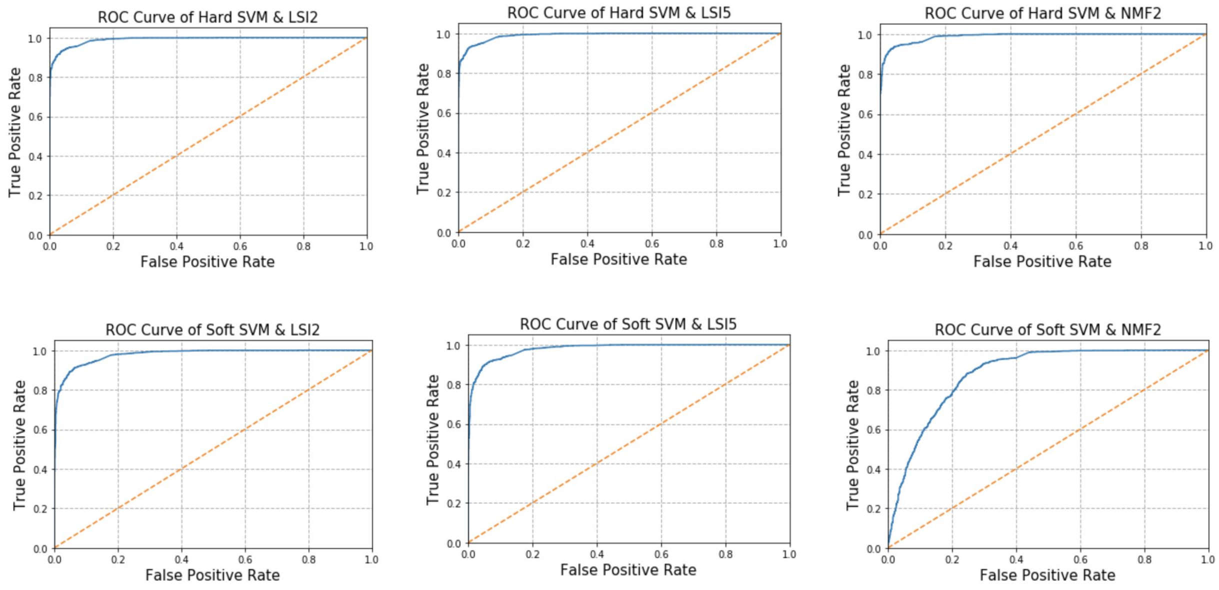


Figure 3. ROC curve of SVM classification

	Accuracy	Precision	Recall	Confusion Matrix
HardSVM&LSI2	0.944444444444	0.963831212324	0.922435897436	[1536 54] [121 1439]
HardSVM&LSI5	0.946031746032	0.965194109772	0.924358974359	[1538 52] [118 1442]
HardSVM&NMF2	0.944444444444	0.965702757229	0.920512820513	[1539 51] [124 1436]
SoftSVM&LSI2	0.504761904762	0.0	0.0	[1590 0] [1560 0]
SoftSVM&LSI5	0.504761904762	0.0	0.0	[1590 0] [1560 0]
SoftSVM&NMF2	0.504761904762	0.0	0.0	[1590 0] [1560 0]

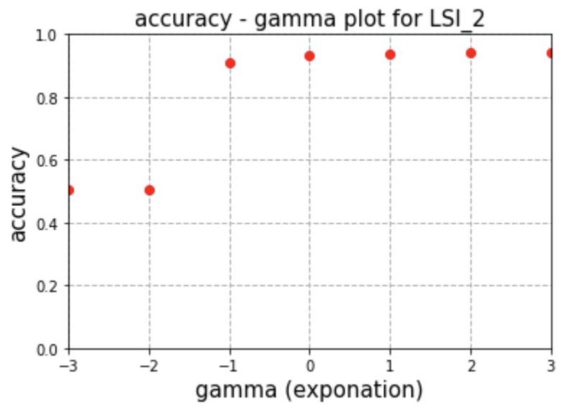
Table 4. Statistical Result of SVM classification

As shown in the result, Hard Margin SVM classifiers give the best performance with all three types of features. Furthermore, with Hard Margin SVM classifier, features with min_df = 5 performs slightly better than features with min_df = 2. This result indicates that we do not need to keep too much infrequent terms during feature extraction.

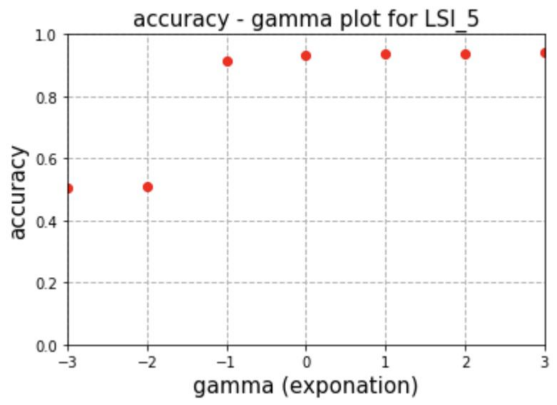
For the Soft Margin SVM, it seems a bit odd that some entries get zero results. This is because the classifier classifies all the data into one class. For example, the definition of precision is the ratio of correctly predicted positive observations to the total predicted positive observations. However, the classifier predicts all the data to the negative class, making the denominator to be 0, which is not allowed. So Python sets the precision result to 0. This observation shows that setting gamma to 0.001 is not a good idea.

Problem (F)

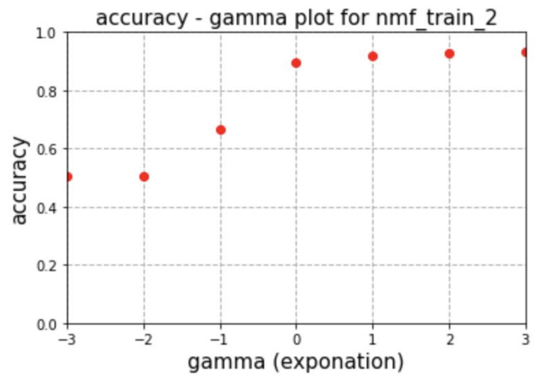
As we discovered in problem(e), gamma is an important hyperparameter for a SVM classifier. Thus, we use five-fold cross validation to find the best value of gamma for our dataset. We are trying to find the best gammas which give the best accuracy for three types of features. The accuracy - gamma plot shows how the accuracy changes along with the gamma.



```
Best gamma for LSI_2 is: 1000
Name: LSI_2 & Gamma = 1000
Confusion Matrix:
[[ 2316.    73.]
 [  193.  2150.]]
Accuracy:
0.943788887128
Precision:
0.967256827288
Recall:
0.917756877176
```



```
Best gamma for LSI_5 is: 1000
Name: LSI_5 & Gamma = 1000
Confusion Matrix:
[[ 2308.    81.]
 [  199.  2144.]]
Accuracy:
0.94082927951
Precision:
0.963835515142
Recall:
0.915133577222
```



```
Best gamma for nmf_train_2 is: 1000
Name: nmf_train_2 & Gamma = 1000
Confusion Matrix:
[[ 2298.    91.]
 [  228.  2115.]]
Accuracy:
0.932587608359
Precision:
0.958936668588
Recall:
0.902768904322
```

Figure 4. Statistical Result of part f

As shown in the result, in all cases, gamma=1000 gives the best accuracy.

Problem (G)

In this section, we used naive bayes classifier to classify the data into two classes: “Computer Technology” and “Recreational Activity”. We calculated the confusion matrix, accuracy, precision and recall of the classification result. We also plotted the ROC curve of the classification. We used both LSI and NMF method to reduce the dimension of TFxIDF, and we chose two different min_df =2 and min_df =5 when using LSI method. The result is shown in the following tables and figures.

Comparing (a) and (b), we can conclude that the accuracy will be a little bit higher when we choose min_df=5 instead of min_df=2. Moreover, comparing (a) and (c), we can conclude that, when we applying Naive Bayes to the classification, using NMF to reduce the matrix dimension will lead to higher accuracy than using LSI.

(a). Using LSI, min_df = 2

accuracy:	82.06%
precision:	89.02%
recall:	72.76%
confusion matrix:	[[1450 140] [425 1135]]

Table 5. Statistical Result of NB classification (a)

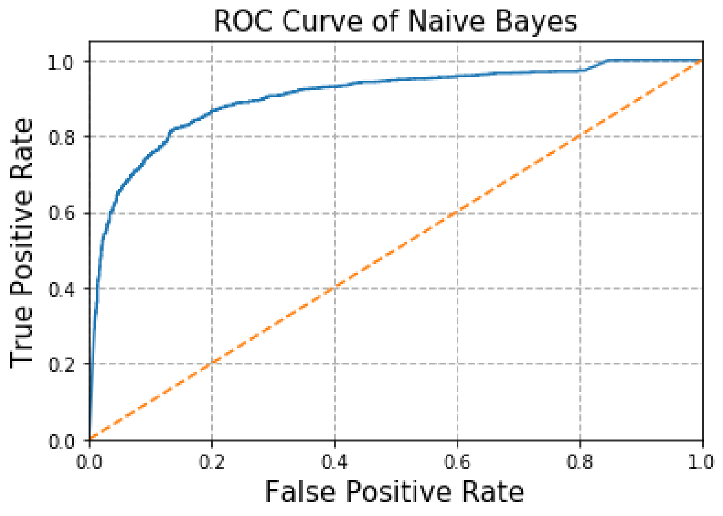


Figure 5. ROC Curve of NB classification (a)

(b) Using LSI, min_df = 5

accuracy:	82.83%
precision:	89.47%
recall:	74.04%
confusion matrix:	[[1454 136] [405 1155]]

Table 6. Statistical Result of NB classification (b)

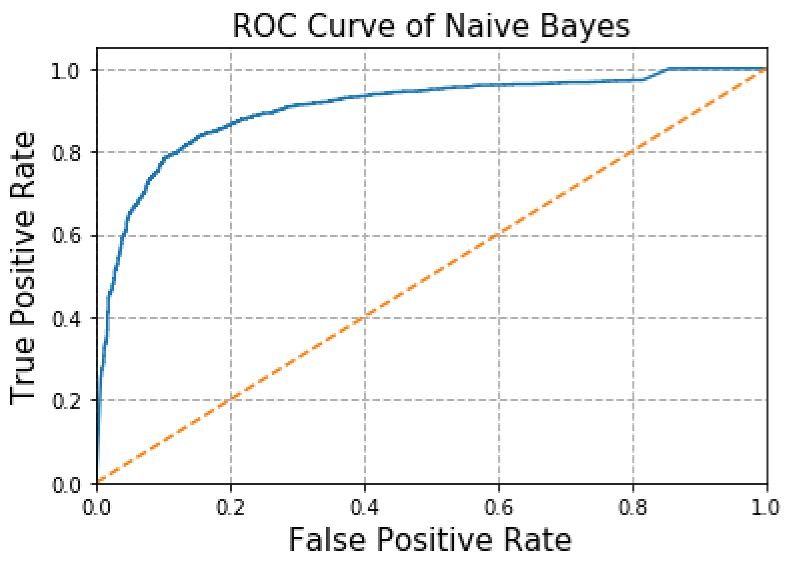


Figure 6. ROC Curve of NB classification (b)

(c) Using NMF, min_df = 2

accuracy:	91.75%
precision:	96.63%
recall:	86.35%
confusion matrix:	[[1543 47] [213 1347]]

Table 7. Statistical Result of NB classification (c)

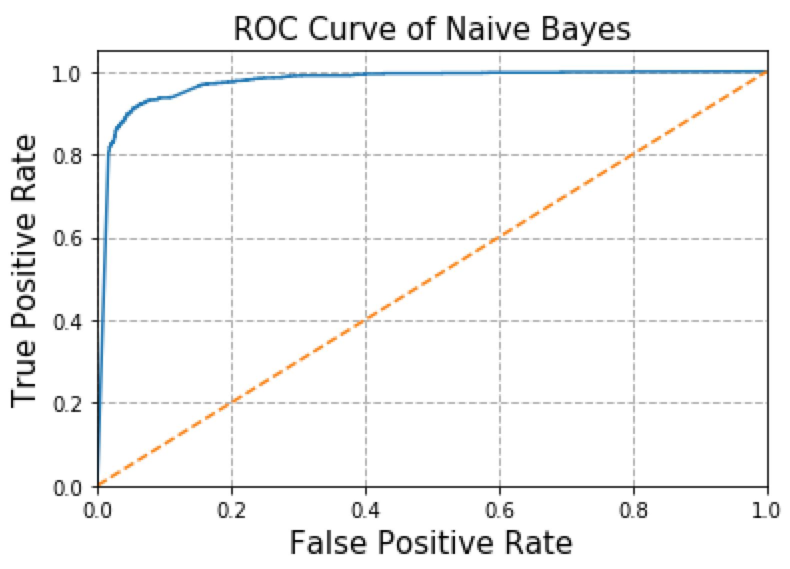


Figure 7. ROC Curve of NB classification (c)

Problem (H)

Instead of Naive Bayes, we used Logistic Regression algorithm for this section. The result is shown in the following tables and figures.

Comparing (a) and (b), we can conclude that the accuracy will be a little bit higher when we choose min_df=5 instead of min_df=2. Moreover, comparing (a) and (c), we can conclude that, when we applying Logistic Regression to the classification, using LSI to reduce the matrix dimension is a better choice than NMF.

(a). Using LSI, min_df = 2

accuracy:	93.33%
precision:	97.14%
recall:	89.17%
confusion matrix:	[[1549 41] [169 1391]]

Table 8. Statistical Result of LR classification (a)

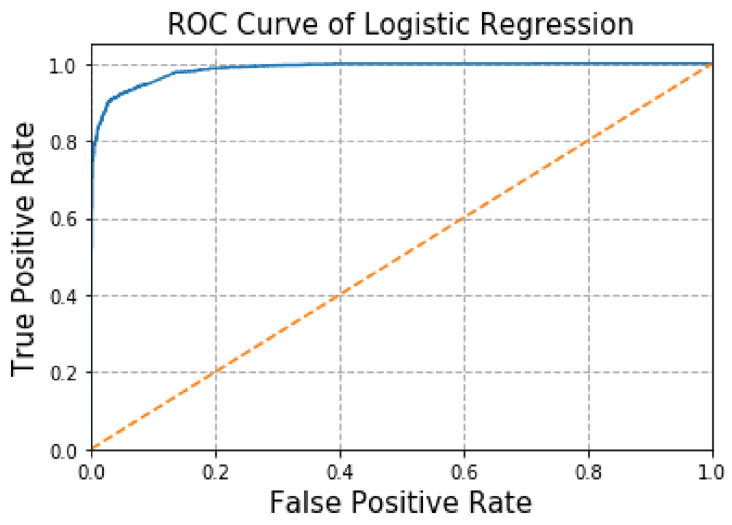


Figure 8. ROC Curve of LR classification (a)

(b). Using LSI, min_df = 5

accuracy:	93.65%
precision:	97.09%
recall:	89.87%
confusion matrix:	[[1548 42] [158 1402]]

Table 9. Statistical Result of LR classification (b)

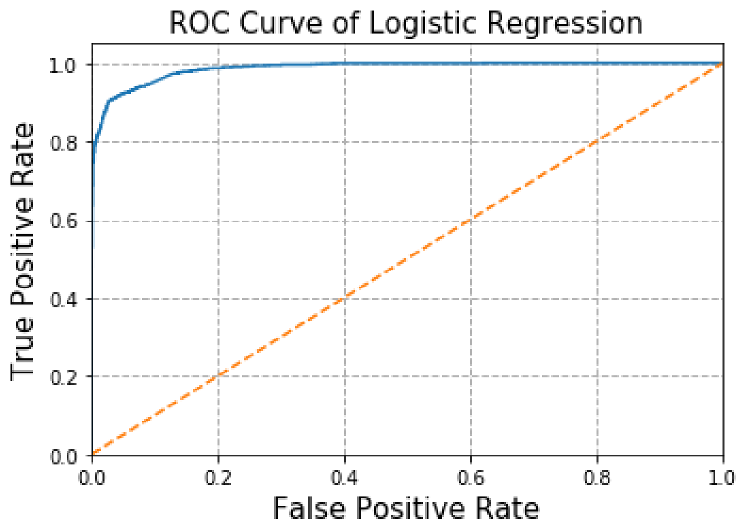


Figure 9. ROC Curve of LR classification (b)

(c). Using NMF, min_df = 2

accuracy:	85.59%
precision:	91.39%
recall:	78.27%
confusion matrix:	[[1475 115] [339 1221]]

Table 10. Statistical Result of LR classification (c)

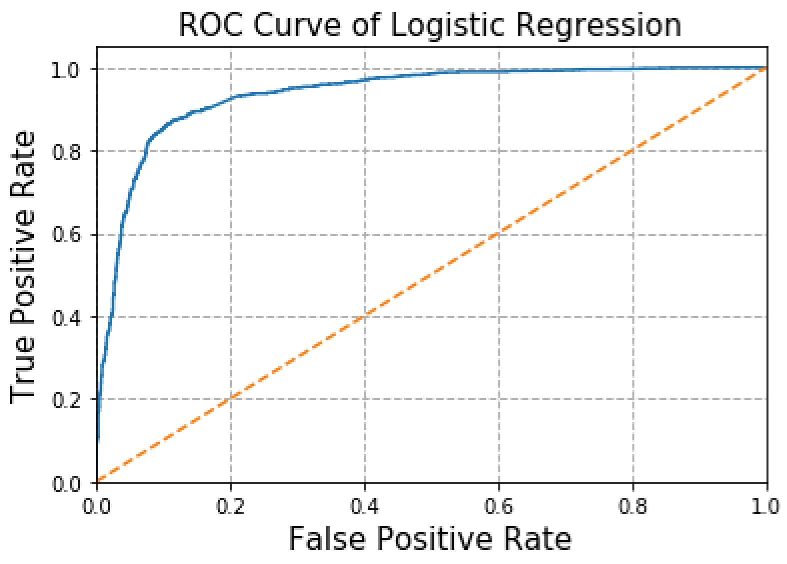


Figure 10. ROC Curve of LR classification (c)

Problem (l)

In this section, we continued to use Logistic Regression as the classifier. However, we added a regularization and penalty to the classifier. We tried both norm 1 and norm 2 regularization, and sweep through the regularization from 0.01 to 10,000 in each norm. The result is shown in the following tables.

(a) **penalty = l1**

(i) Using LSI, min_df = 2

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	88.38%	93.26%	82.5%	[[1497 93] [273 1287]]
0.1	90.51%	93.39%	86.99%	[[1494 96] [203 1357]]
1	93.62%	96.07%	90.83%	[[1532 58] [143 1417]]
10	94.54%	96.27%	92.56%	[[1534 56] [116 1444]]
100	94.57%	96.45%	92.44%	[[1537 53] [118 1442]]
1000	94.57%	96.45%	92.44%	[[1537 53] [118 1442]]
10,000	94.57%	96.45%	92.44%	[[1537 53] [118 1442]]

Table 11. Statistic Result of LR classification, penalty = l1(i)

(ii) Using LSI, min_df = 5

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	88.25%	92.996%	82.5%	[[1497 93] [273 1287]]
0.1	90.76%	93.79%	87.12%	[[1500 90] [201 1359]]
1	93.84%	96.59%	90.77%	[[1540 50] [144 1416]]
10	94.51%	96.51%	92.24%	[[1538 52] [121 1439]]
100	94.57%	96.27%	92.63%	[[1534 56] [115 1445]]
1000	94.60%	96.27%	92.69%	[[1534 56] [114 1446]]

10,000	94.60%	96.27%	92.69%	[[1534 56] [114 1446]]

Table 12. Statistic Result of LR classification, penalty = l1(ii)

(ii) Using NMF, min_df = 2

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	50.48%	0.0%	0.0%	[[1590 0] [1560 0]]
0.1	64.70%	76.92%	41.03%	[[1398 192] [920 640]]
1	91.81%	94.53%	88.59%	[[1510 80] [178 1382]]
10	94.41%	96.19%	92.37%	[[1533 57] [119 1441]]
100	94.67%	96.28%	92.82%	[[1534 56] [112 1448]]
1000	94.70%	96.16%	93.01%	[[1532 58] [109 1451]]
10,000	94.63%	96.09%	92.95%	[[1531 59] [110 1450]]

Table 13. Statistic Result of LR classification, penalty = l1(iii)

(b) **penalty = l2**

(i) Using LSI, min_df = 2

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	88.98%	98.25%	79.17%	[[1568 22] [325 1235]]
0.1	91.24%	96.86%	85.06%	[[1547 43] [233 1327]]
1	93.33%	97.14%	89.17%	[[1549 41] [169 1391]]

10	94.41%	97.07%	91.47%	[[1547 43] [133 1427]]
100	94.67%	96.59%	92.50%	[[1539 51] [117 1443]]
1000	94.51%	96.39%	92.37%	[[1536 54] [119 1441]]
10,000	94.57%	96.45%	92.44%	[[1537 53] [118 1442]]

Table 14. Statistic Result of LR classification, penalty = l2(i)

(ii) Using LSI, min_df = 5

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	89.30%	97.74%	80.26%	[[1561 29] [308 1252]]
0.1	91.71%	96.76%	85.15%	[[1545 45] [216 1344]]
1	93.65%	97.09%	89.87%	[[1548 42] [158 1402]]
10	94.06%	96.73%	91.09%	[[1542 48] [139 1421]]
100	94.54%	96.58%	92.24%	[[1539 51] [121 1439]]
1000	94.60%	96.33%	92.63%	[[1535 55] [115 1445]]
10,000	94.60%	96.27%	92.69%	[[1534 56] [114 1446]]

Table 15. Statistic Result of LR classification, penalty = l2(ii)

(iii) Using NMF, min_df = 2

Regularization	Accuracy	Precision	Recall	Confusion Matrix
0.01	61.33%	89.58%	24.81%	[[1545 45] [1173 387]]

0.1	76.29%	89.28%	59.23%	[[1479 111] [636 924]]
1	85.59%	91.39%	78.27%	[[1475 115] [339 1221]]
10	91.21%	94.77%	87.05%	[[1515 75] [202 1358]]
100	93.37%	96.05%	90.32%	[[1532 58] [151 1409]]
1000	94.44%	96.38%	92.24%	[[1536 54] [121 1439]]
10,000	94.63%	96.27%	92.75%	[[1534 56] [113 1447]]

Table 16. Statistic Result of LR classification, penalty = l2(iii)

From the result, we can conclude that the test accuracy is very low when the regularization parameter is small. However, as the parameter increases, the accuracy increases steadily. That is, small regularization parameter will lead to very high test error, while the test error will drop significantly when we increasing the regularization parameter. As the regularization parameter increases, other coefficient of the fitted hyperplane will decreases.

For the preference of two loss function, if we want a robust solution to the classification, we will choose norm 1 loss function. However, if we want stable and always one solution, norm 2 loss function will be a better choice.

<u>Problem (J)</u>

In this final part, we are dealing with a multi class problem. We have four classes, which are comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian. For the naive bayesian classifier, we only apply it to the feature with LSI. For the 1 VS 1 SVM and the 1 VS REST SVM, we apply them to both feature with LSI and feature with NMF. The results are shown below:

	Accuracy	Precision	Recall	Confusion Matrix
NB	0.722044728435	0.717472155899	0.722044728435	[217 90 65 20] [45 261 44 35] [39 37 263 51] [4 2 3 389]
SVM1V1 & LSI	0.839616613419	0.843397293018	0.839616613419	[308 56 26 2] [50 311 20 4] [20 37 329 4] [3 15 14 366]
SVM1V1 & NMF	0.771884984026	0.776382868805	0.771884984026	[282 72 35 3] [75 263 39 8] [44 24 315 7] [10 9 31 348]
SVM1VR & LSI	0.833865814696	0.835575670621	0.833865814696	[304 48 35 5] [48 291 40 6] [20 27 338 5] [3 1 22 372]
SVM1VR & NMF	0.78785942492	0.787650111786	0.78785942492	[281 61 43 7] [69 256 47 13] [33 18 325 14] [3 1 23 371]

Table. 17 Statistical Result of Multiclass Classification

As we can see, SVM performs better than NB. Using SVM classifier, features with LSI performs better than features with NMF.