ECE 219

Project 1

Shengrong Wu & Shuangyu Li

Introduction

In this project, we Implemented the classification of eight classes from *sklearn.fetch_20newsgroups* with different supervised learning model. In this project, we utilized *sklearn* and *nltk* to implement algorithms like support vector machine (SVM), Naive Bayes and logistic regression with different hyperparameters. In the last part of the project, we performed multiclass classification on four subclasses from *sklearn.fetch_20newsgroups*, by extending our SVM and Naive Bayes classifier techniques. Their ROC graphs and statistical results are plotted and calculated to compare performance throughout this report.

Problem Statement and Datasets

In this project, we work with “20 Newsgroups” dataset. It is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, each corresponding to a different topic. However, we just need to work on 8 different classes shown in table 1. We need to train the and test the data to classify them into 2 parent classes, ‘ Computer technology ’ and ‘Recreational activity ’

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

Problem a) Display document histogram

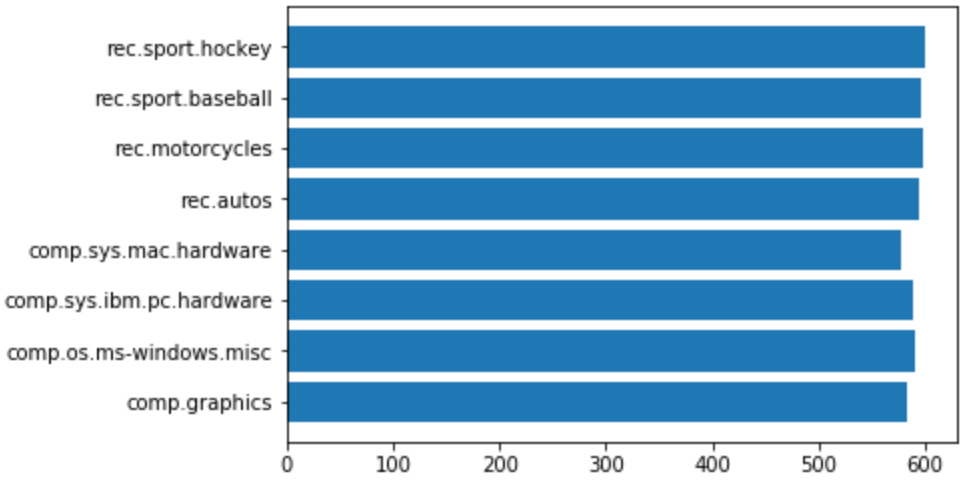

Fig 1: Distribution of the Data size

To avoid the bias brought by the size of training data set, we need to balance the data set before the training . In this project, the data is most evenly distributed for different classes since we use the given data directly. The exact amount of each class corresponding to the Figure 1.1 from top to bottom is [584, 591, 590, 578, 594, 598, 597, 600].

Modeling Text Data and Feature Extraction

Problem b) Tokenize the Input Data and Processing

The purpose of this part is to tokenize the documents, to stem the words and to remove the unneeded stopping words and punctuation, which is common in all documents but useless for classification. We use the dictionary of *text.ENGLISH_STOP_WORDS* from *sklearn*, stopwords from *nltk.corpus* and punctuation set from *python.string*. They are combined to build a set of stopwords which is called later in the *CountVectorizer*. With different min document frequency threshold, we could fetch different term frequency(TF) matrix.

For eight classes required,
When Min_df = 5, we extract 10396 terms from 4732 documents
When Min_df = 2, we extract 24849 terms from 4732 documents

It follows intuition that with higher threshold, we filtered out more terms like inappropriate abbreviations or typos.

Problem c) Extract 10 Most Significant Terms

We also want to measure the significance of a term to each class. To achieve this, we extend TFxIDF matrix to a TFxICF matrix. It is the same as a TFxIDF matrix, except that a class sits in place of a document, i.e. TFxICF(c,t) = tf(t,c) * icf(t,c) , where icf(t,c) = $\log[n_{classes}/cf(t)] + 1$. Hence, TFICF is a matrix with $n_{classes}$ rows, with each row corresponding to one class . To calculate TFICF for our problem, we need to use the entire dataset with $n_{classes} = 20$. To find the 10 most significant terms to the each of the 4 classes of our concern, we simply search for the 10 terms with the highest TFICF value in each corresponding row. We ran the code with two cases, by setting min_df to 2 and 5. The two results are almost identical with one exception in *comp.sys.mac.hardware*, as shown in Table 2, where the term 'problem' in the case of <min_df=5>replaced 'quadra' in the case of <min_df=2>.

Without looking at all documents in the dataset, it is reasonable to assume 'quadra' occured fewer than 5 times in the entire dataset while mostly, if not all, in *comp.sys.mac.hardware.* Besides the terms which make intuitive sense to their classes, there are terms that are common in all classes, such as "edu" and "organization". These happened because the term frequencies are very high and dominates the TFxICF value. One possible improvement to this problem could be stemming the terms or redesign TFxICF matrix to predict term importance.

Table 2. 10 Most Significant Terms (not in order)

comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
'com'	'apple'	'com'	'christian'
'controller'	'edu'	'edu'	'church'
'drive'	'line'	'line'	'edu'
'edu'	'mac'	'new'	'god'
'ide'	'organization'	'offer'	'jesus'
'line'	'post'	'organization’	'know'
'organization'	‘quadra’(2), ‘problem’(5)	'post'	'line'
'scsi'	'scsi'	'sale'	'people'
'subject'	'subject'	'subject'	'say'
'use'	'use'	'university'	'subject'

Feature Selection

Problem d) Dimension Reduction Applying LSI and NMF

To speed up calculation and find more important feature for training, Latent Semantic Indexing is applied to reduce the dimension of the TFxIDF matrix. In this problem, we set the hyperparameter k to 50, which represents the number of most significant feature in the training sets and use *sklearn* package to implement it. The dimension of the reduced version of training data is (4732, 50) . The dimension of the reduced version of test data is (3150, 50). The result of Non-Negative Matrix Factorization is slightly different but in the same size. These matrix representations are passed for training and testing in the following classification tasks and the results will be compared.

Learning Algorithms

In this part, we will apply different learning algorithm with different parameter to reach the best result for two classes classification problem.

Problem e) SVM

The Support vector machine(SVM) algorithm is called to solve the classification problem in this question and we tried with different minimum document term frequency threshold and the two distinct misclassification weight - γ .

With larger γ, the hard SVM model works quite well for the classification problem and barely changes with different term frequency thresholds. However, for the soft SVM model, it classifies everything into 'Rec' class no matter how the other parameters changes . Their results are all shown in the tables and ROC curves below.

Table 3: Result for Hard SVM, γ = 1000, **LSI & min_df = 5**

Hard SVM γ = 1000	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 5	0.9717460317460317	0.9716920657958394	0.971844058782038	[[1507 53] [36 1554]]

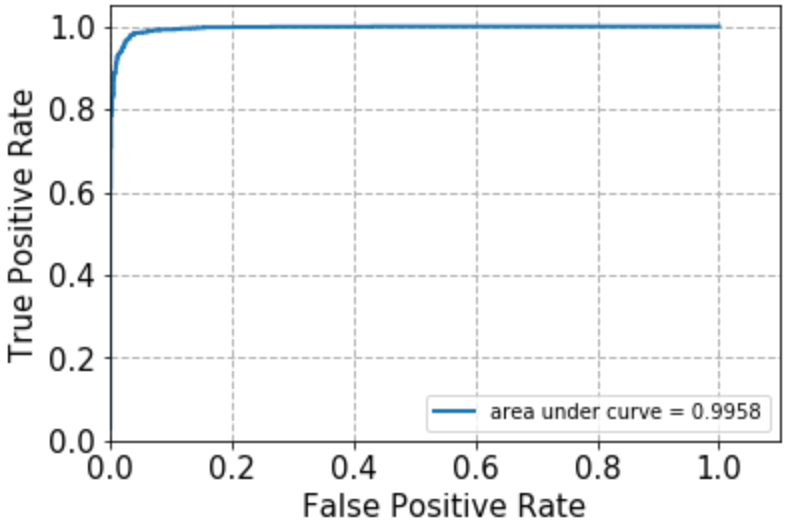

Fig 2: ROC curve for Hard SVM with **LSI & min_df = 5**

Table 4: Result for Hard SVM with **LSI & min_df = 2**

Hard SVM γ = 1000	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	0.9698412698412698	0.9697750362844703	0.9699779658125243	**[[1502 58] [37 1553]]**

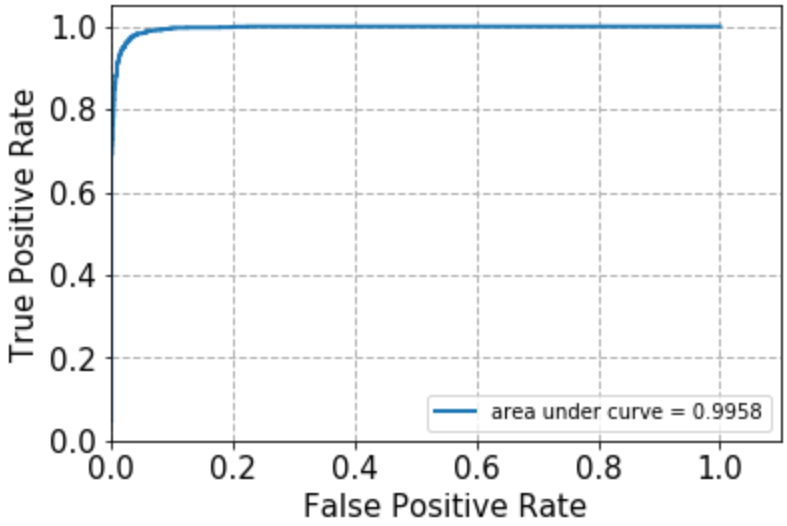

Fig 3: ROC curve for Hard SVM with **LSI & min_df = 2**

Table 5: for Hard SVM with **NMF & min_df = 2**

Hard SVM γ = 1000	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	**0.9593650793650793**	**0.959288824383164**	**0.9595289265799256**	**[[1484 76] [52 1538]]**

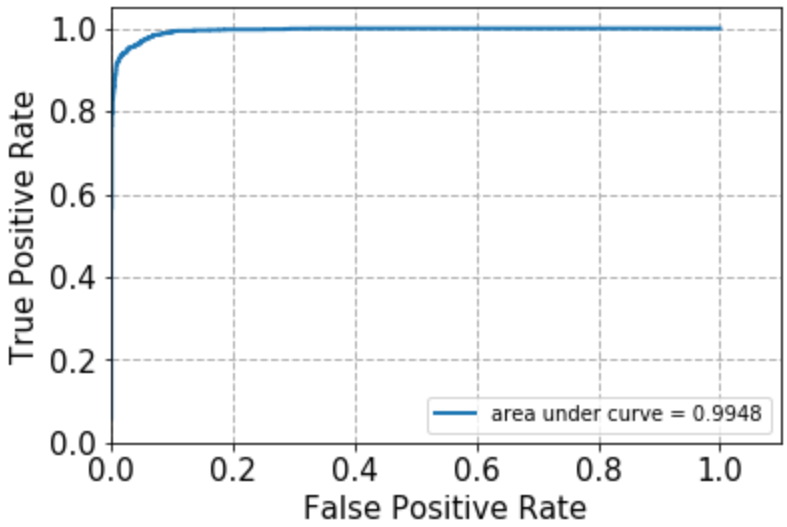

Fig 4: ROC curve for Hard SVM with **NMF & min_df = 2**

Table 6:Result for Soft SVM with **LSI & min_df = 5**

soft SVM γ = 0.001	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 5	**0.5047619047619047**	**0.5**	**0.2523809523809524**	**[[0 1560] [0 1590]]**

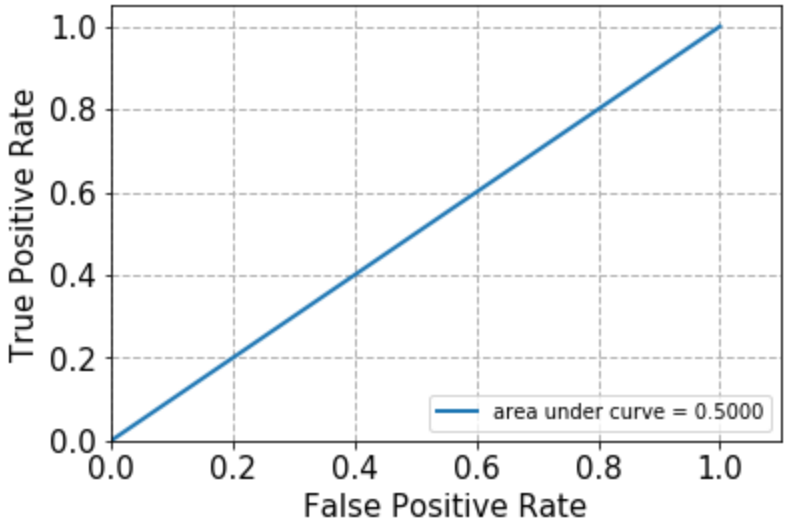

Fig 5: ROC curve for Soft SVM with **LSI & min_df = 5**

Table 7:Result for Soft SVM with **LSI & min_df = 2**

soft SVM γ = 0.001	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	**0.5047619047619047**	**0.5**	**0.2523809523809524**	**[[0 1560] [0 1590]]**

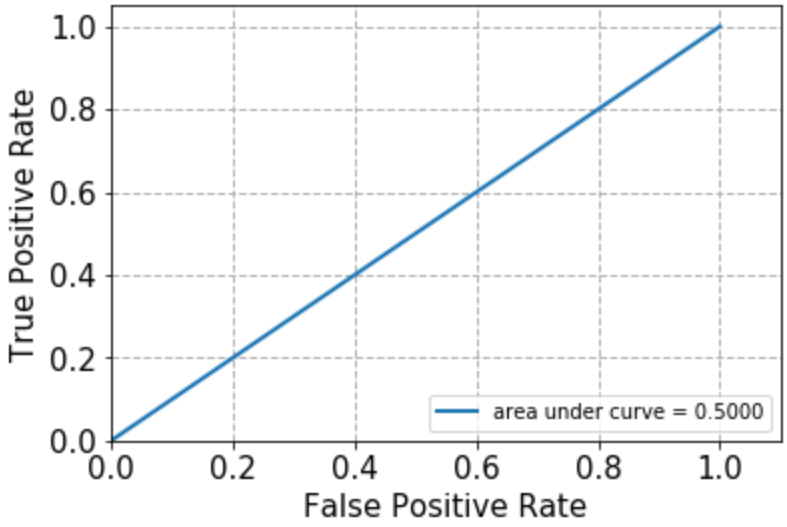

Fig 6: ROC curve for Soft SVM with **LSI & min_df = 2**

Table 8:Result for Soft SVM with **NMF & min_df = 2**

soft SVM γ = 0.001	Accuracy	Recall	Precision	Confusion matrix
NMF & min_df = 2	**0.5047619047619047**	**0.5**	**0.2523809523809524**	**[[0 1560] [0 1590]]**

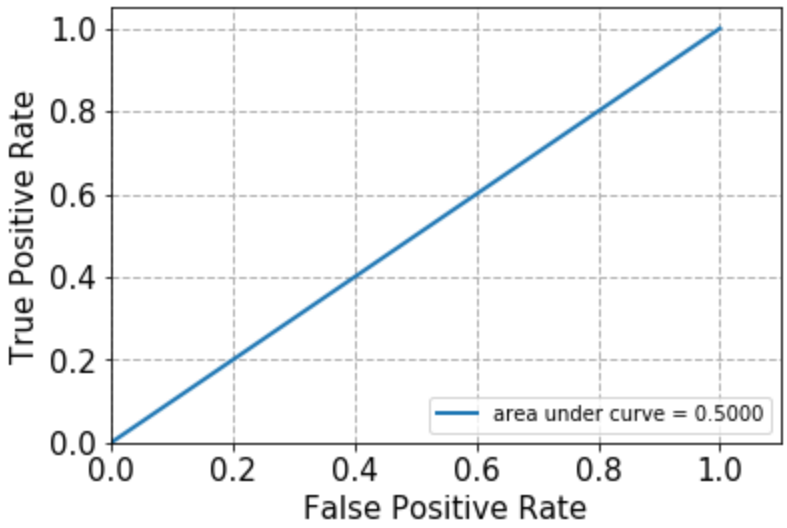

Fig 7: ROC curve for Soft SVM with **NMF & min_df = 2**

Problem f) Cross Validation

It is obvious to see the important role of γ from the result of problem e. A cross validation is performed for different min_df, different dimension reduction methods and for γ range from 0.001 to 1000. The results are shown below. Element (i, j) represents the accuracy for γ = 10^(i-3) in j fold. The best result of most of them wiggle between 100 and 1000 depends on the random factor in folds.

Table 8: Accuracy result for LSI & min_df = 2

```
[[0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.97043295 0.97465681 0.96617336 0.97145877 0.96934461]
 [0.97360084 0.97571278 0.96723044 0.9756871  0.97251586]
 [0.97571278 0.97993664 0.97040169 0.97463002 0.97251586]
 [0.97360084 0.97993664 0.9756871  0.97251586 0.9756871 ]
 [0.97254488 0.97993664 0.97357294 0.97357294 0.97674419]]

Best -> 100
```

Table 9:Result for SVM gamma = 100 with **LSI & min_df = 2**

Hard SVM γ = 100	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	0.97301587 3015873	0.972943880 0193517	0.9731764239 884059	[[1506 54] [31 1559]]

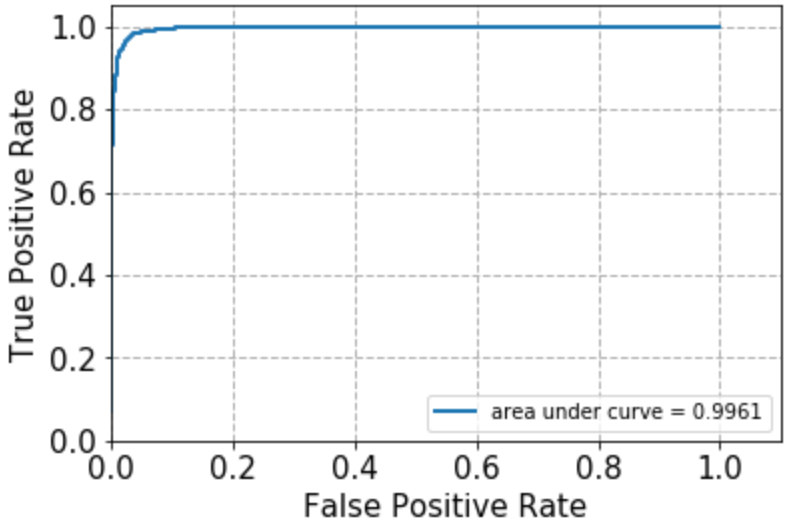

Fig 8: ROC curve for SVM gamma = 100 with **LSI & min_df = 2**

Table 10: Accuracy result for LSI & min_df = 5

```
[[0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.45828933 0.4656811  0.51268499 0.51057082 0.51585624]
 [0.96832101 0.97571278 0.96194503 0.96194503 0.96088795]
 [0.97360084 0.97676874 0.96934461 0.96828753 0.9640592 ]
 [0.97782471 0.97676874 0.96617336 0.97040169 0.97674419]
 [0.97465681 0.97993664 0.97040169 0.96723044 0.97463002]
 [0.97465681 0.97888068 0.97357294 0.97357294 0.97674419]]
Best -> 1000
```

Table 11: Result for Hard SVM, γ = 1000, **LSI & min_df = 5**

Hard SVM γ = 1000	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 5	0.9717460317460317	0.9716920657958394	0.971844058782038	[[1507 53] [36 1554]]

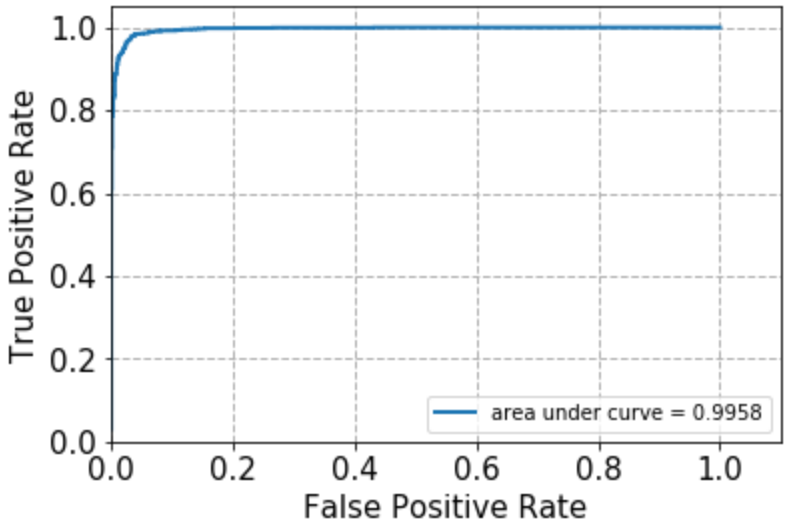

Fig 9: ROC curve for SVM gamma = 1000 with LSI & min_df = 5

Table 12: Accuracy result for NMF & min_df = 2

```
[[0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.45828933 0.4656811  0.51057082 0.50845666 0.51268499]
 [0.95459345 0.9313622  0.94608879 0.95348837 0.94714588]
 [0.97676874 0.96620908 0.9577167  0.96617336 0.96300211]
 [0.97571278 0.97254488 0.96511628 0.9640592  0.96723044]
 [0.97571278 0.97148891 0.96828753 0.96194503 0.9756871 ]]

Best -> 1000
```

Table 13: Result for SVM gamma = 1000 with **NMF & min_df = 2**

SVM γ = 1000	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	0.9593650793650793	0.959288824383164	0.9595289265799256	[[1484 76] [52 1538]]

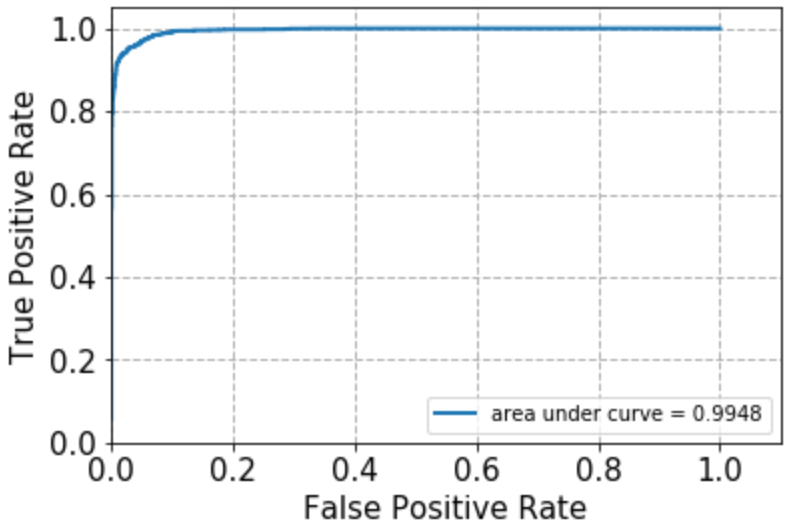

Fig 10: ROC curve for SVM gamma = 1000 with **NMF & min_df = 2**

Problem g) Naive Bayes

Because Naive Bayes model does not accept the negative input, we only perform the NMF reduction in this part.

Table 14: Result for Naive Bayes with NMF & min_df = 2

Naive Bayes	Accuracy	Recall	Precision	Confusion matrix
NMF & min_df = 2	0.9358730158730159	0.9354015481373972	0.9404330392680143	[[1382 178] [24 1566]]

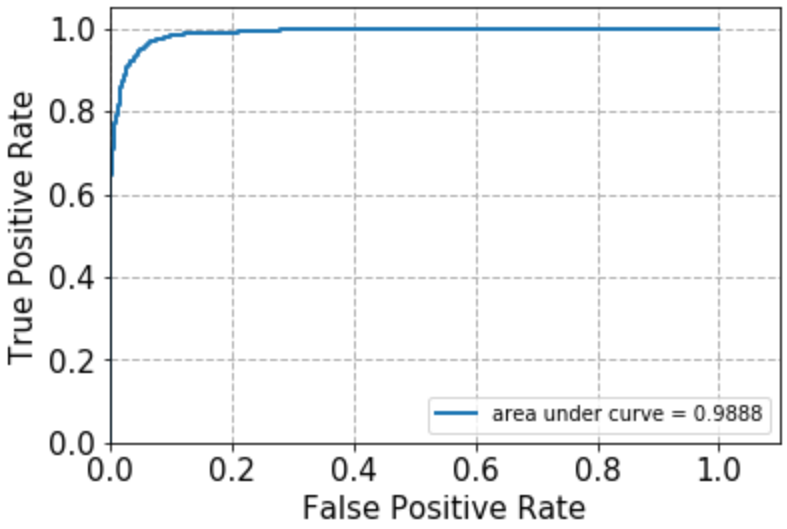

Fig 11: ROC curve for Naive Bayes with NMF & min_df = 2

Problem h) Logistic Regression

Same procedure is repeated for logistic regression.

Table 15: Result for Logistic Regression with NMF & min_df = 2

Logistic Regression	Accuracy	Recall	Precision	Confusion matrix
NMF & min_df = 2	0.9352380952380952	0.9350507982583454	0.9359999999999999	[[1428 132] [72 1518]]

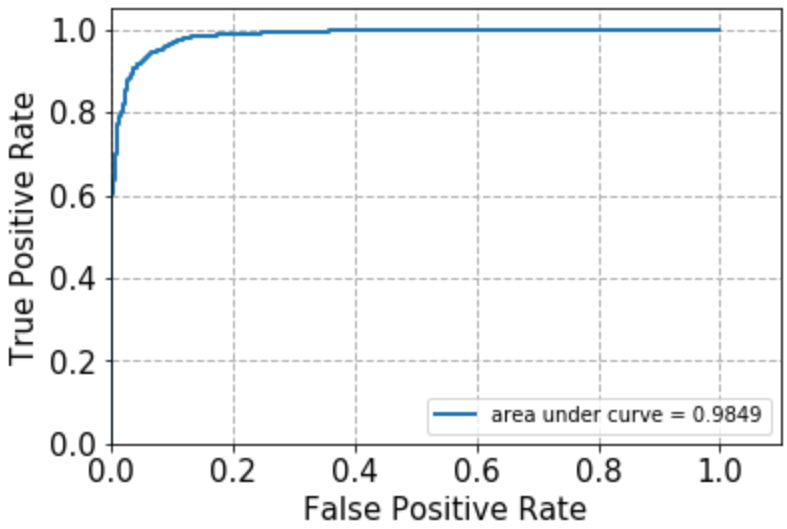

Fig 12: ROC curve for Logistic Regression with NMF & min_df = 2

Table 16: Result for Logistic Regression with LSI & min_df = 2

Logistic Regression	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 2	0.9663492063492064	0.9662373004354137	0.9666843420402709	[[1489 71] [35 1555]]

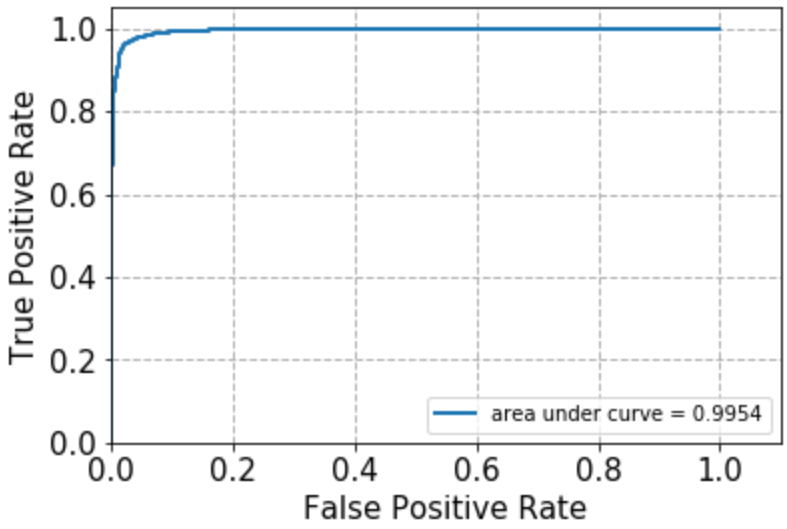

Fig 13: ROC curve for Logistic Regression with LSI & min_df = 2

Table 17 : Result for Logistic Regression with LSI & min_df = 5 Result

Logistic Regression	Accuracy	Recall	Precision	Confusion matrix
LSI & min_df = 5	0.9657142857142857	0.9655962747943888	0.9660818236705228	[[1489 71] [35 1555]]

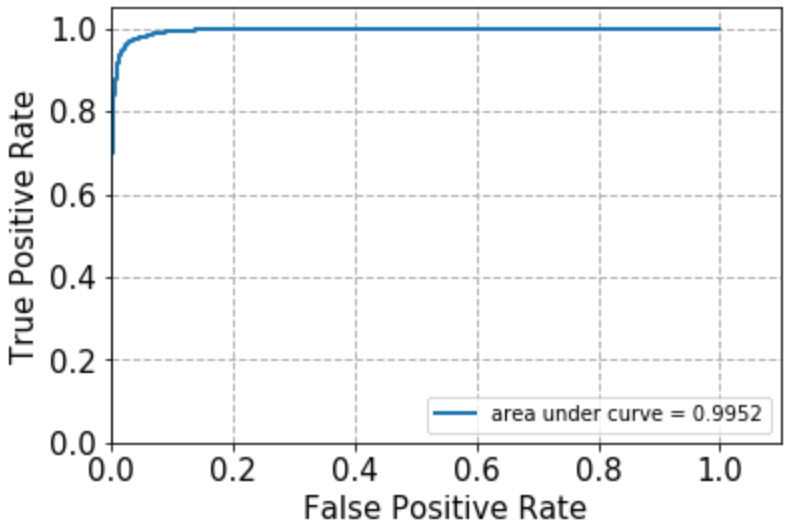

Fig 14: ROC curve for Logistic Regression with LSI & min_df = 5

Problem I) Logistic Regularization

The accuracy of logistic model with different regularization parameter C is shown below. Generally, with higher regularization parameter, the model perform slightly better but for some the of the model, the second largest parameter lead to the best performance. The difference norm, L1 and L2, are calculated to do comparison. Since at a higher degree vector space the 'distance' between sample is hard to describe like the 2D plane, therefore the various norm are tried here. In this part, the accuracy is selected as the key to compare their performance. L1 norm works poorly at a low parameter value, are as good as L2 norm at a high regularization parameter. The detailed accuracy score tables are shown below.

Table 18: accuracy score table of l1 norm

L1 Norm	C = 10^-3	C = 10^-2	C = 10^-1	C = 10^0	C = 10^1	C = 10^2	C = 10^3
NMF & min_df = 2	0.49523809523809526	0.49523809523809526	0.6993650793650794	0.9584126984126984	0.9603174603174603	0.9615873015873015	0.9625396825396826
LSI & min_df = 2	0.49523809523809526	0.8711111111111111	0.9580952380952381	0.9657142857142857	0.9723809523809523	0.9707936507936508	0.9704761904761905
LSI & min_df = 5	0.49523809523809526	0.8888888888888888	0.9552380952380952	0.9634920634920635	0.9711111111111111	0.9704761904761905	0.9704761904761905

Table 19: accuracy score table of l2 norm

L2 Norm	C = 10^-3	C = 10^-2	C = 10^-1	C = 10^0	C = 10^1	C = 10^2	C = 10^3
NMF & min_df = 2	0.5047619047619047	0.5057142857142857	0.8765079365079365	0.9352380952380952	0.9488888888888889	0.9523809523809523	0.9565079365079365
LSI & min_df = 2	0.68	0.9419047619047619	0.9615873015873015	0.9663492063492064	0.9711111111111111	0.9730158730158730	0.9707936507936508
LSI & min_df = 5	0.7333333333333333	0.9453968253968253	0.9587301587301588	0.9657142857142857	0.9695238095238096	0.9714285714285714	0.9707936507936508

Multiclass Classification

So far, we have been dealing with classifying the data points into two classes. In this part, we explore multiclass classification by extending the algorithms - Naive Bayes and SVM. Naive Bayes perform the multiclass classification inherently, regardless of the number of classes. Here we are using NMF as multinomial Naive Bayes classification only takes non-negative values.

For SVM, we have two implementation for linear multiclass classification, namely one-to-one method and one-vs-rest method. From the sklearn package, "SVC" function assumes a one-to-one implementation and "linearSVC" function assumes a one-vs-rest implementation. We used these functions to generate results with both NMF and LSI, and with a hyperparameter $\gamma = 100$. The results are shown below. Among all, SVM are better than Naive Bayes, while the two implementation of SVM are almost identical. SVM with LSI have slightly better results than SVM with NMF.

Table 24: Result for Naive Bayes with NMF & min_df = 2

Naive Bayes	Accuracy	Recall	Precision	Confusion matrix
NMF & min_df = 2	0.8044728434504792	0.8030669796821986	0.8100486186980468	[[304 42 42 4] [102 246 34 3] [52 12 318 8] [2 1 4 391]]

Table 25: Result for One-to-One SVM with NMF & $\gamma = 100$

SVM $\gamma = 100$	Accuracy	Recall	Precision	Confusion matrix
One-to-One NMF & min_df = 2	0.8466453674121406	0.8457436512398824	0.8489677120848334	[[315 48 28 1] [70 290 23 2] [40 17 332 1] [7 2 1 388]]

Table 26: Result for One-to-One SVM with LSI & γ = 100

SVM γ = 100	Accuracy	Recall	Precision	Confusion matrix
One-to-One LSI & min_df = 2	**0.8830670926517572**	**0.8824786394510012**	**0.88269648140134 62**	**[[322 46 24 0] [40 321 24 0] [23 17 348 2] [3 2 2 391]]**

Table 27: Result for One-vs-Rest SVM with NMF & γ = 100

SVM γ = 100	Accuracy	Recall	Precision	Confusion matrix
One-vs-Rest NMF & min_df = 2	**0.8472843450479233**	**0.8463837819275722**	**0.84632883805944 6**	**[[306 52 32 2] [64 291 26 4] [30 17 338 5] [4 2 1 391]]**

Table 28: Result for One-vs-Rest SVM with LSI & γ = 100

SVM γ = 100	Accuracy	Recall	Precision	Confusion matrix
One-vs-Rest LSI & min_df = 2	**0.8843450479233227**	**0.8838239201573732**	**0.88394902012475 04**	**[[314 52 26 0] [34 327 23 1] [18 19 351 2] [4 0 2 392]]**