EE219

Large-Scale Data Mining

Project 1

Classification Analysis on Textual Data

Winter 2018

By Xudong Li, Zeyu Jin

January 29, 2018

Introduction

Statistical classification is a task of identifying a category from a predefined set given a training dataset with known category memberships. Classification differs from the task of clustering, which concerns grouping data points with no predefined category memberships, where the objective is to seek inherent structures in data with respect to suitable measures. Classification turns out as an essential element of data analysis, especially when dealing with a large amount of data. In this project, we investigate different methods for classifying textual data, and work with the "20 Newsgroups" dataset, which is a collection of approximately 20,000 newsgroup documents, partitioned evenly across 20 different newsgroups, each corresponding to a different topic. The programming language we use is Python 3.5, with a combination of iPython Notebook and Spyder.

Part A

In this part of the project, we are asked to plot a histogram of the number of training documents per class to check if they are evenly distributed, and a following plot is generated:

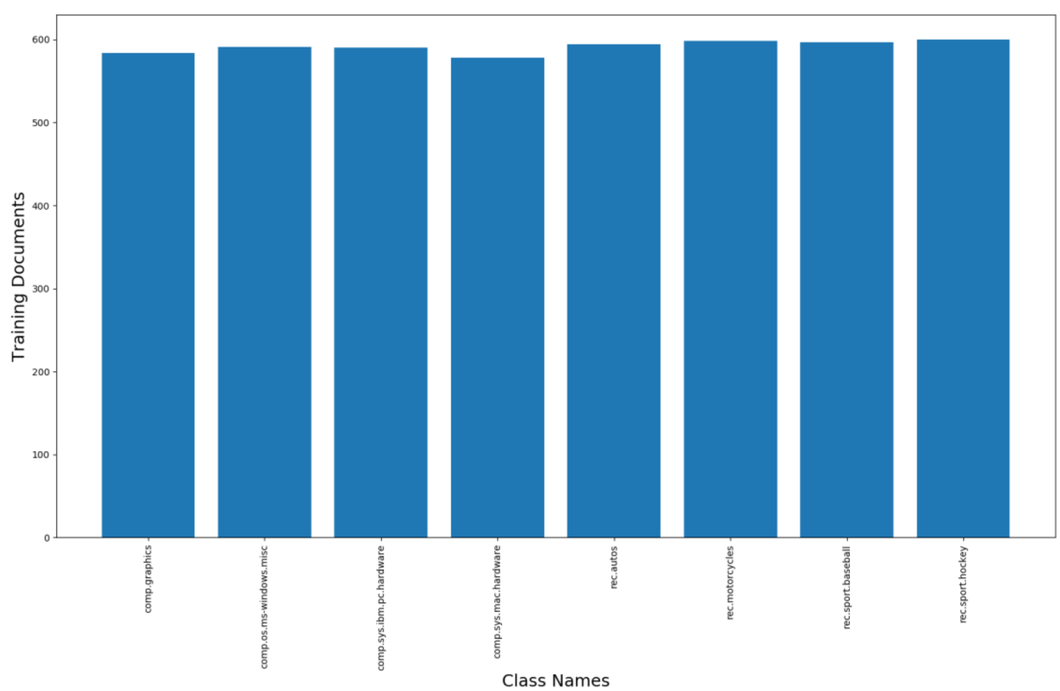


Figure 1. Training documents per class

We can see that the above training documents are evenly distributed very well.

Part B

In this part, we first tokenize each document into words, and then excluding the stops words, punctuations, and using stemmed version of words. Finally, create a TFxIDF vector representations of the document. In our code, we follow the implementation given in the discussion notes by combining stopwords from different libraries, remove punctuations, and lemmatize the words. The size of TFxIDF matrices are shown below:

min_df=2:

	documents	terms
Training	4732	24848
Testing	3150	24848

Table 1. TFxIDF size with min_df =2

min_df=5:

	documents	terms
Training	4732	10396
Testing	3150	10396

Table 2. TFxIDF size with min_df =5

Part C

In this part of the assignment, we want to quantify how significant a word is to a class, and therefore we define a measurement called TFxICF. It is very similar to TFxIDF, except that a class sits in place of a document. In our code, we change the implementation of TFxIDF by replacing the document to class and recount the appearance of each term inside a class.

The following result is generated:

```
Top 10 significant terms in class comp.sys.ibm.pc.hardware:
scsi            (significance = 0.408235)
drive           (significance = 0.282953)
edu             (significance = 0.261100)
ide             (significance = 0.204117)
line            (significance = 0.185040)
use             (significance = 0.180216)
com             (significance = 0.177945)
subject         (significance = 0.173404)
organization    (significance = 0.166025)
controller      (significance = 0.155366)
```

Figure 2. Top 10 terms in class comp.sys.ibm.pc.hardware when df_min=2

```
Top 10 significant terms in class comp.sys.mac.hardware:
edu             (significance = 0.382650)
line            (significance = 0.232702)
mac             (significance = 0.231093)
subject         (significance = 0.209007)
organization    (significance = 0.195569)
apple           (significance = 0.165862)
use             (significance = 0.165508)
quadra          (significance = 0.157934)
scsi            (significance = 0.141075)
problem         (significance = 0.133680)
```

Figure 3. Top 10 terms in class comp.sys.mac.hardware when df_min=2

```
Top 10 significant terms in class misc.forsale:
edu             (significance = 0.425642)
line            (significance = 0.270209)
sale            (significance = 0.255408)
subject         (significance = 0.254538)
organization    (significance = 0.242680)
new             (significance = 0.143575)
post            (significance = 0.142304)
com             (significance = 0.140610)
university      (significance = 0.138069)
offer           (significance = 0.119010)
```

Figure 4. Top 10 terms in class misc.forsale when df_min=2

```
Top 10 significant terms in class soc.religion.christian:
god             (significance = 0.343931)
edu             (significance = 0.237563)
christian       (significance = 0.204520)
jesus           (significance = 0.188124)
say             (significance = 0.186998)
church          (significance = 0.159100)
subject         (significance = 0.153516)
people          (significance = 0.149416)
line            (significance = 0.145772)
know            (significance = 0.140306)
```

Figure 5. Top 10 terms in class soc.religion.christian when df_min=2

Now, we change the "df_min=2" to "df_min=5":

```
Top 10 significant terms in class comp.sys.ibm.pc.hardware:
scsi            (significance = 0.415795)
drive           (significance = 0.288192)
edu             (significance = 0.265935)
ide             (significance = 0.207897)
line            (significance = 0.188467)
use             (significance = 0.183553)
com             (significance = 0.181240)
subject         (significance = 0.176615)
organization    (significance = 0.169100)
controller      (significance = 0.158243)
```

Figure 6. Top 10 terms in class comp.sys.ibm.pc.hardware when df_min=5

```
Top 10 significant terms in class comp.sys.mac.hardware:
edu             (significance = 0.399257)
line            (significance = 0.242802)
mac             (significance = 0.241123)
subject         (significance = 0.218079)
organization    (significance = 0.204057)
apple           (significance = 0.173061)
use             (significance = 0.172692)
scsi            (significance = 0.147198)
problem         (significance = 0.139482)
post            (significance = 0.138006)
```

Figure 7. Top 10 terms in class comp.sys.mac.hardware when df_min=5

```
Top 10 significant terms in class misc.forsale:
edu             (significance = 0.434367)
line            (significance = 0.275747)
sale            (significance = 0.260644)
subject         (significance = 0.259756)
organization    (significance = 0.247654)
new             (significance = 0.146518)
post            (significance = 0.145221)
com             (significance = 0.143492)
university      (significance = 0.140899)
offer           (significance = 0.121450)
```

Figure 8. Top 10 terms in class misc.forsale when df_min=5

```
Top 10 significant terms in class soc.religion.christian:
god             (significance = 0.351841)
edu             (significance = 0.243027)
christian       (significance = 0.209224)
jesus           (significance = 0.192450)
say             (significance = 0.191299)
church          (significance = 0.162759)
subject         (significance = 0.157047)
people          (significance = 0.152853)
line            (significance = 0.149125)
know            (significance = 0.143533)
```

Figure 9. Top 10 terms in class soc.religion.christian when df_min=5

According to the results above, we can see that the significant words are very reasonable to each class. For example, the most significant word in Christian class is god. When we change the minimum frequency of a word from 2 to 5 in our CountVectorzier() function, we see the that significance increases. A possible explanation is that by eliminating the uncommon words, the weight of common words increases, and therefore their significance also improved.

Part D

In this section, we apply LSI and NMF to the TFxIDF matrices we just built, so each document is mapped to a 50-dimensional vector. This is a dimension reduction technique that will only extract the important terms from the original data, which will give us a better results and faster computation when we make predictions. After doing the transformations, the shape of our training data is:

```
In [38]: Dk_train_transpose.shape
Out[38]: (4732, 50)

In [39]: W_train.shape
Out[39]: (4732, 50)
```

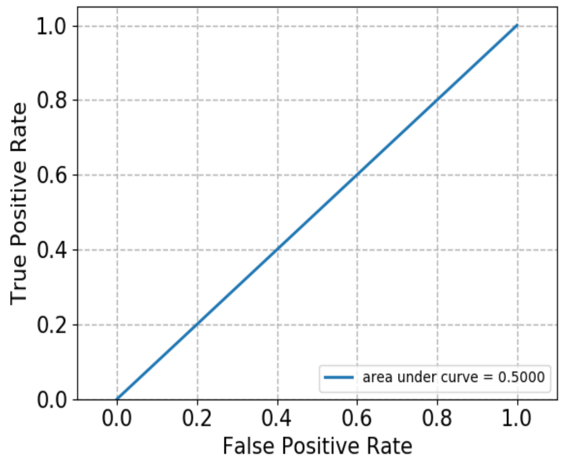
Figure 10. LSI and NMF matrices size

Part E

In this part of the project, we need to separate the documents into “Computer Technology” vs. “Recreational Activity” groups by using SVM classifier. Since we only need to the binary classifiers for LSI only, the parameter we can adjust is LSI vs NMF, “Hard Margin (C = 0.001)” vs. “Soft Margin (C = 1000)” and “min_df =2” vs “min_df=5”. There total of 6 results below:

Case 1: Soft SVM, LSI, min_df = 2

ROC Curve



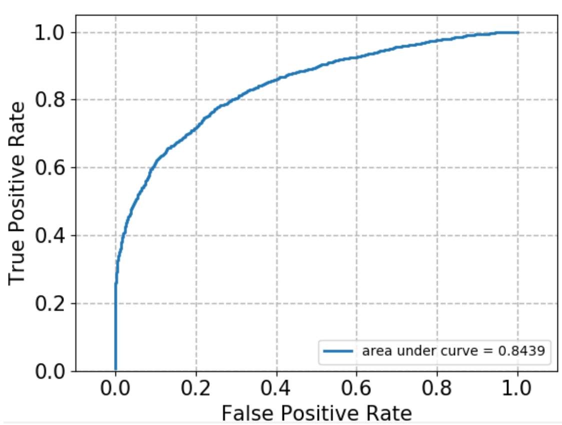
Confusion Matrix

	0	1
0	0	1560
1	0	1590

Accuracy	Precision	Recall
0.505	0.505	1.0

Case 2: Soft SVM, LSI, min_df=5

ROC Curve



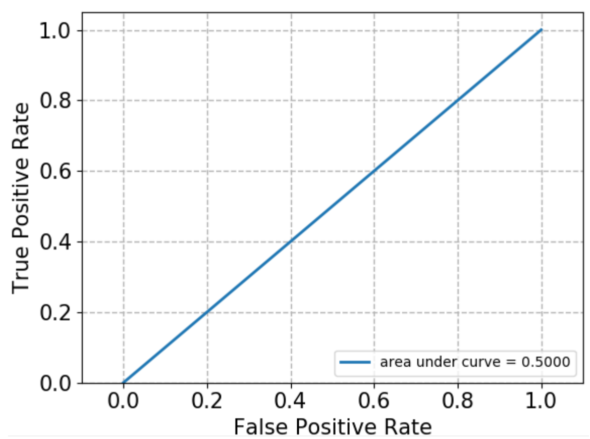
Confusion Matrix

	0	1
0	0	1560
1	0	1590

Accuracy	Precision	Recall
0.505	0.505	1.0

Case 3: Soft SVM, NMF, min_df=2

ROC Curve



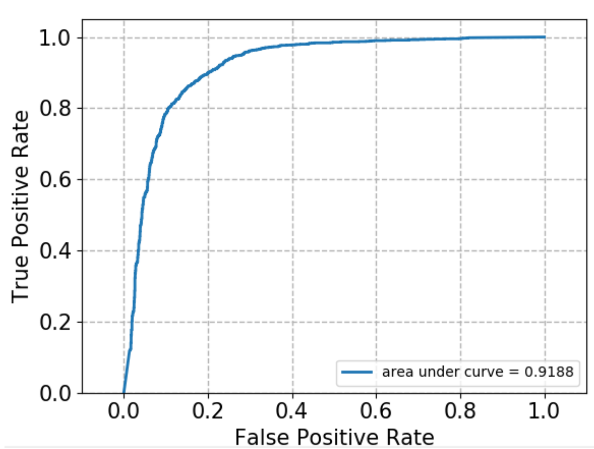
Confusion Matrix

	0	1
0	0	1560
1	0	1590

Accuracy	Precision	Recall
0.505	**0.505**	**1.0**

Case 4: Hard SVM, LSI, min_df=2

ROC Curve



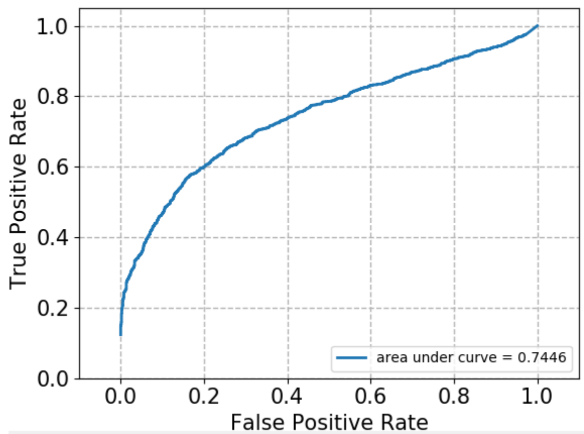
Confusion Matrix

	0	1
0	1326	234
1	226	1364

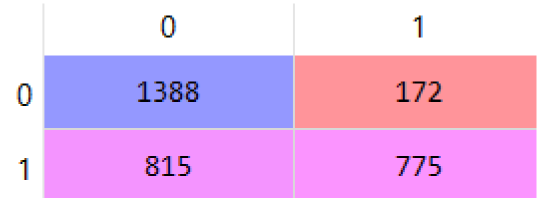
Accuracy	Precision	Recall
0.854	**0.854**	**0.858**

Case 5: Hard SVM, LSI, min_df=5

ROC Curve



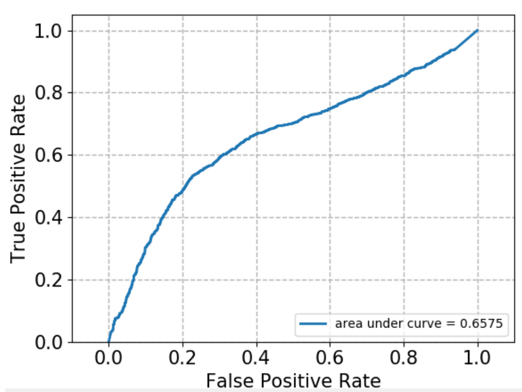
Confusion Matrix



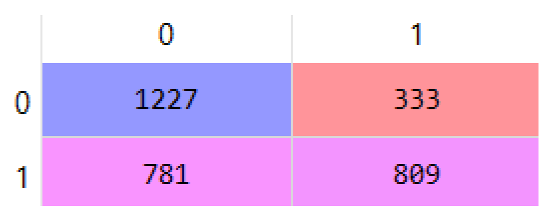
Accuracy	Precision	Recall
0.687	**0.818**	**0.487**

Case 6: Hard SVM, NMF, min_df=2

ROC Curve



Confusion Matrix



Accuracy	Precision	Recall
0.646	**0.708**	**0.508**

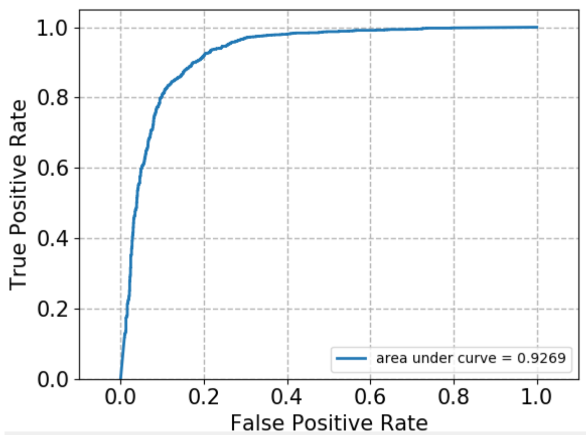
Part F

In this section, we use 5-fold cross-validation to find the best value of the regularization parameter for SVM classifier under LSI or NMF, and min_df=2 or min_df=5. The best parameter is found based on the validation score.

Case 1: LSI, min_df=2

In this case, we find C = 90, which means gamma is 90 is the best parameter for regularization. The calculation is based on the mean of 5 cross validation scores. The score we get is 0.975695.

ROC Curve



Confusion Matrix

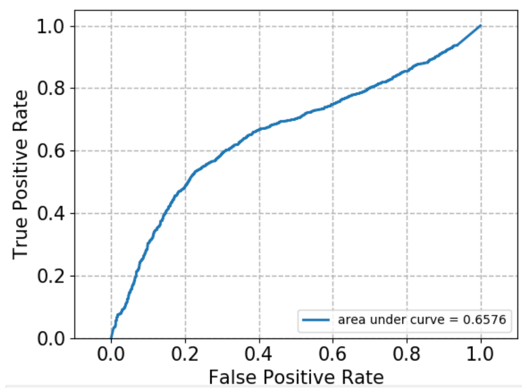
	0	1
0	1357	203
1	239	1351

Accuracy	Precision	Recall
0.860	**0.869**	**0.850**

Case 2: NMF, min_df=2

In this case, we find C = 1000 (the larger the better), which means gamma of 1000 is the best parameter for regularization. The calculation is based on the mean of 5 cross validation scores. The score we get is 0.967245

ROC Curve



Confusion Matrix

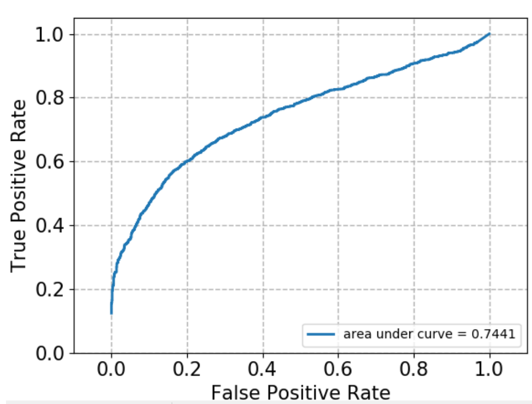
	0	1
0	1227	333
1	781	809

Accuracy	Precision	Recall
0.646	**0.708**	**0.509**

Case 3: LSI, min_df=5

In this case, we find C = 300, which means gamma is 300 is the best parameter for regularization. The calculation is based on the mean of 5 cross validation scores. The score we get is 0.97591.

ROC Curve



Confusion Matrix

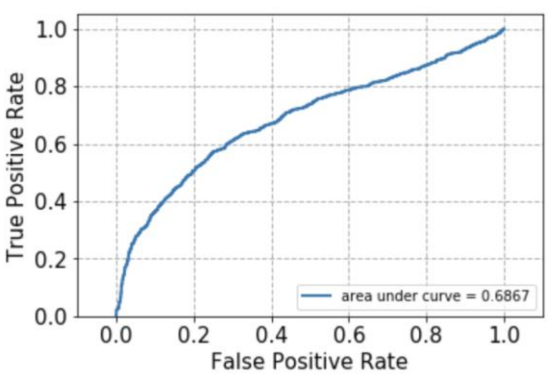
	0	1
0	1388	172
1	818	772

Accuracy	Precision	Recall
0.686	**0.818**	**0.486**

Part G

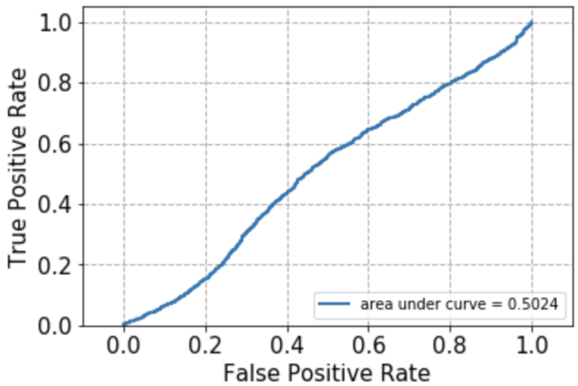
In this section, we use naïve Bayes algorithm to find the best value of the regularization parameter for SVM classifier under NMF, in min_df=2 or min_df=5. The best parameter is found based on the validation score.

Class1: NMF, min_df = 2

ROC Curve	**Confusion Matrix**
	array([[932, 628], [523, 1067]])

Accuracy	Precision	Recall
0.635	**0.630**	**0.671**

Class2: NMF, min_df = 5

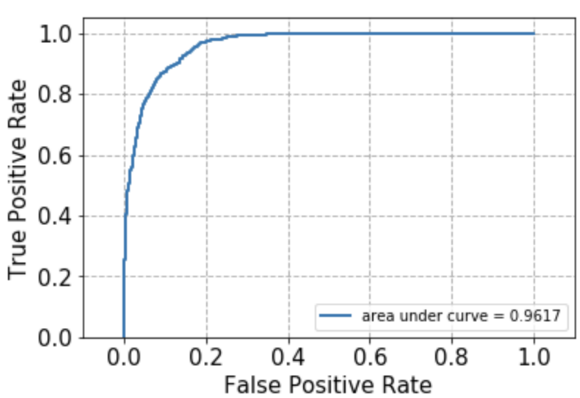
ROC Curve	**Confusion Matrix**
	array([[631, 929], [573, 1017]])

Accuracy	Precision	Recall
0.523	**0.523**	**0.640**

Part H

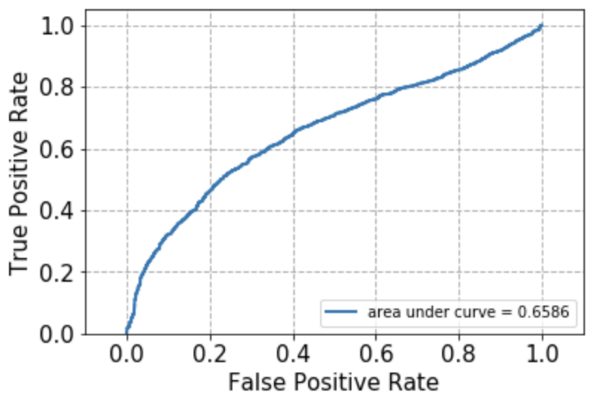
In this section, we use logistic regression classifier to find the best value of the regularization parameter for SVM classifier under LSI or NMF, and min_df=2 or min_df=5. The best parameter is found based on the validation score.

Class1: LSI, min_df = 2

ROC Curve	**Confusion Matrix**
	array([[1361, 199], [163, 1427]])

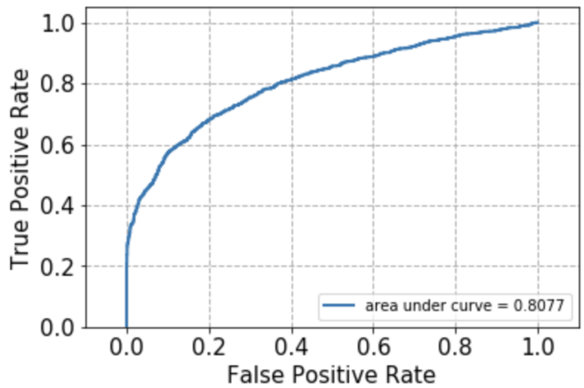
Accuracy	Precision	Recall
0.885	**0.878**	**0.897**

Class2: NMF, min_df = 2

ROC Curve	**Confusion Matrix**
	array([[1055, 505], [659, 931]])

Accuracy	Precision	Recall
0.630	**0.648**	**0.586**

Class3: LSI, min_df = 5

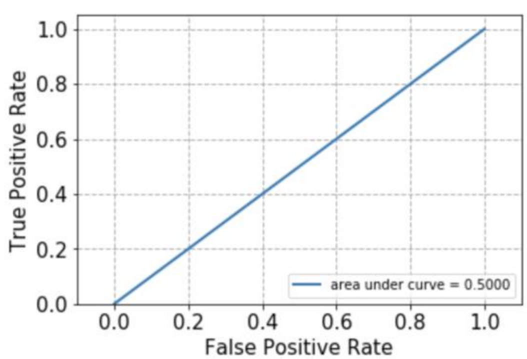
<table>
<tr><th>ROC Curve</th><th>Confusion Matrix</th></tr>
<tr><td>
</td><td>array([[1401, 159],
[679, 911]])</td></tr>
</table>

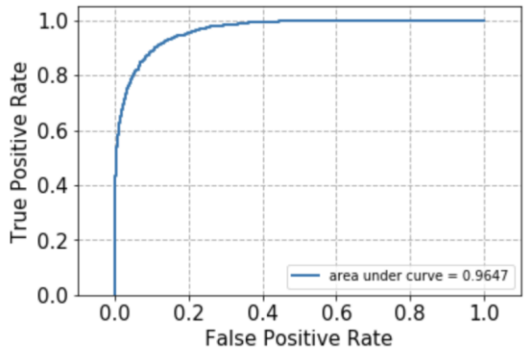
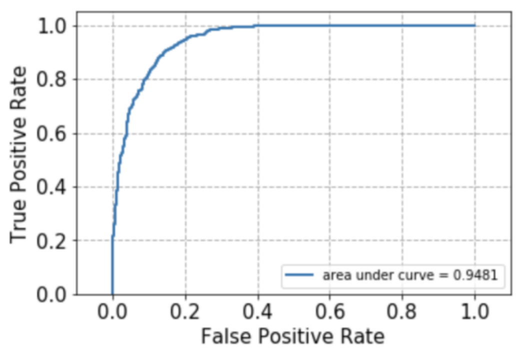
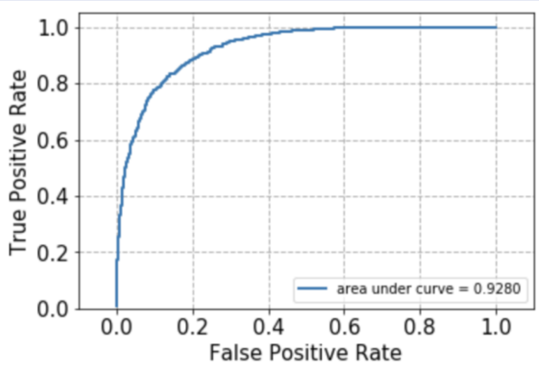
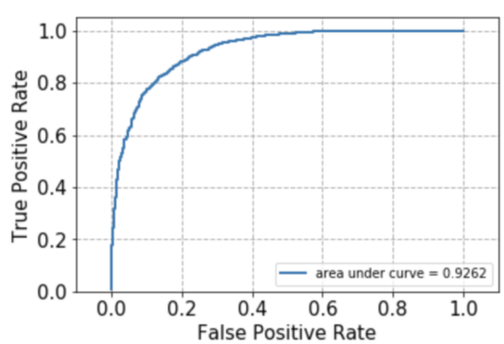
Accuracy	Precision	Recall
0.734	**0.648**	**0.573**

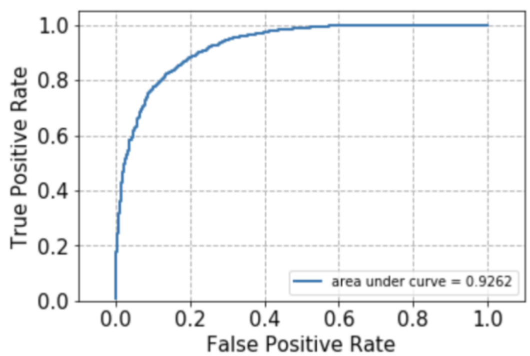
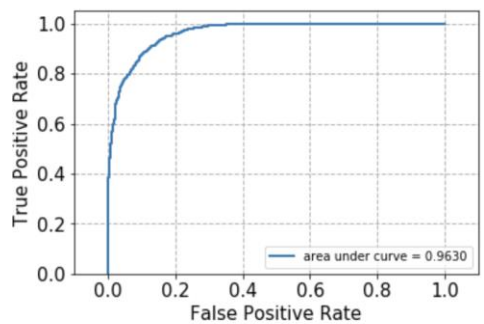
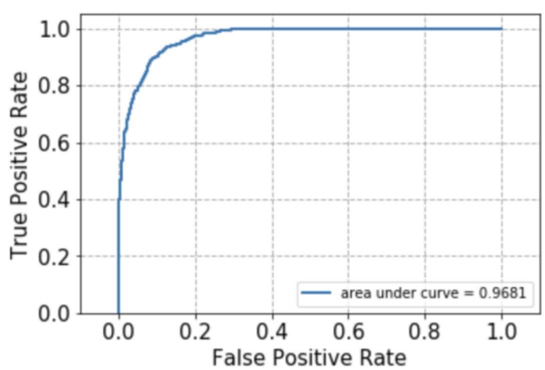
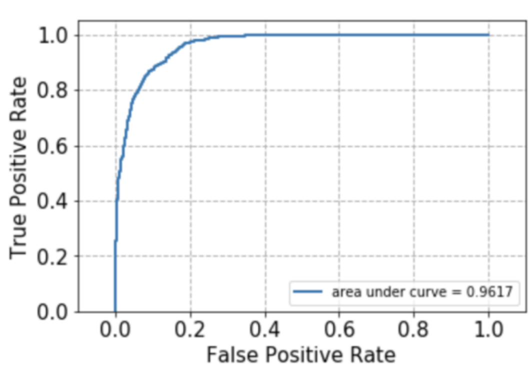
Part I

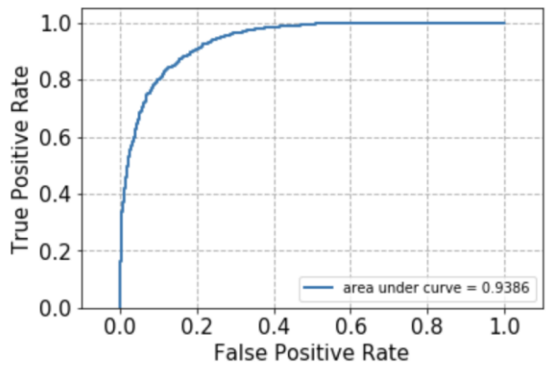
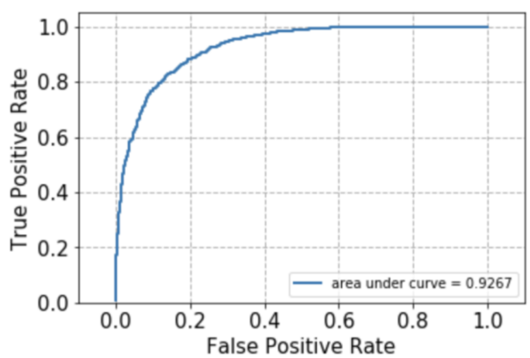
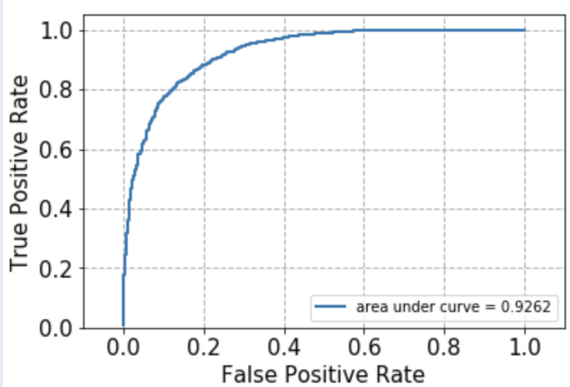
In this section, we add a regularization term to the optimization objective in the logistic regression classifier to find the best value of the regularization parameter for SVM classifier under LSI or NMF, and min_df=2 or min_df=5. The best parameter is found based on the validation score.

Class1: LSI， min_df = 2

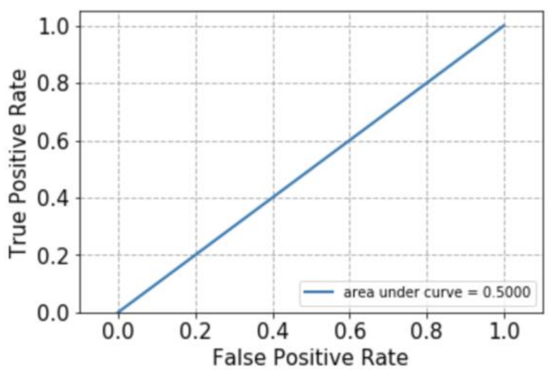
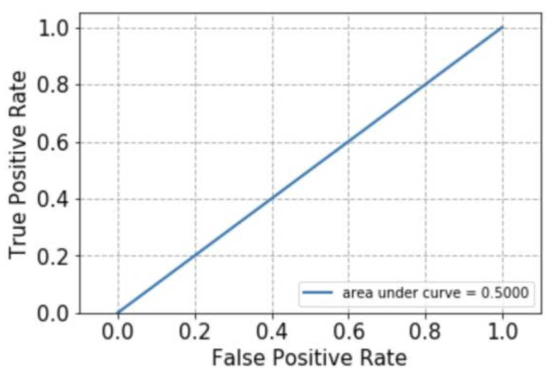
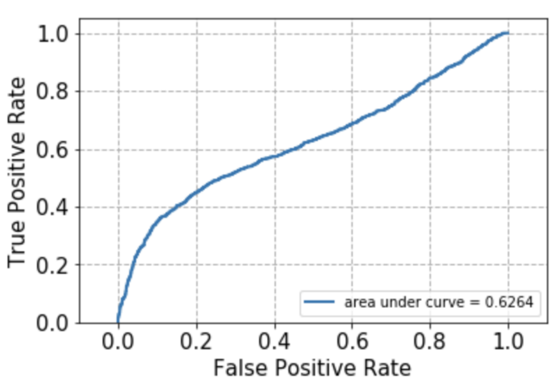
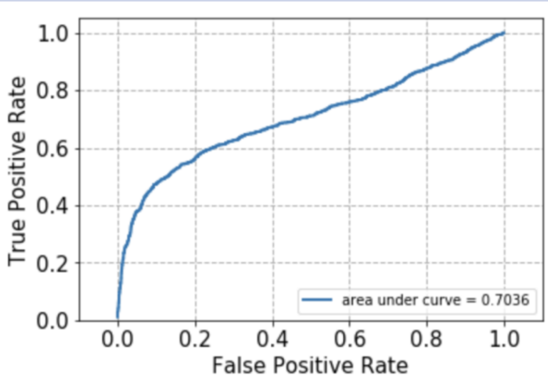
<table>
<tr><th>Params</th><th>ROC Curve</th><th>Confusion Matrix</th><th>A&P&R</th></tr>
<tr><td>penalty is l1
C = 10^-4</td><td>
</td><td>[[1560 0]
[1590 0]]</td><td>acc = 0.495238
pre = 0.000000
rec = 0.000000</td></tr>
</table>

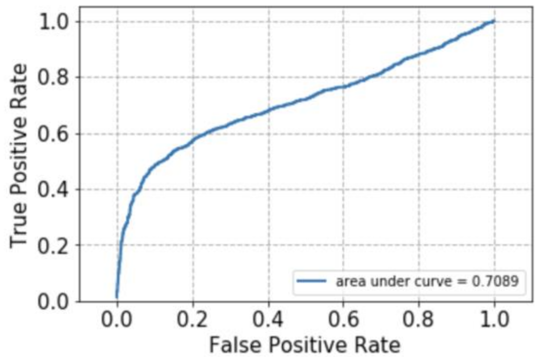
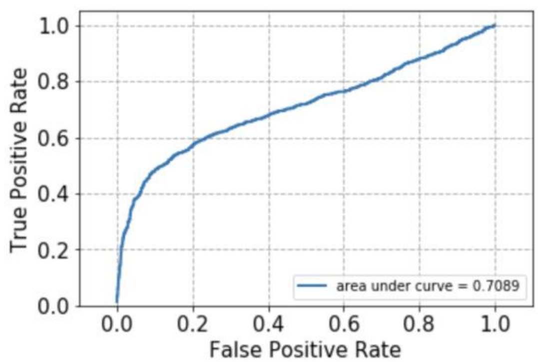
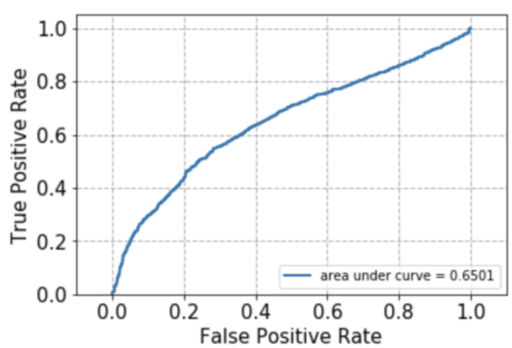
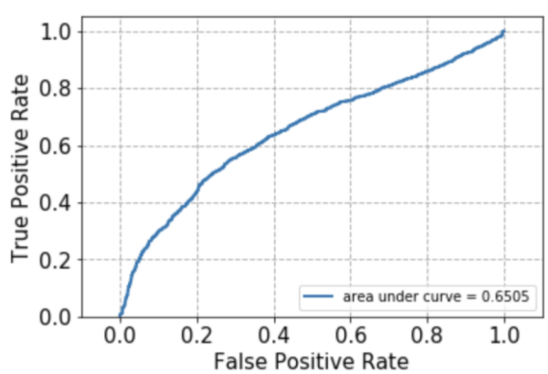
Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^-2		[[1517 43] [457 1133]]	acc = 0.841270 pre = 0.963435 rec = 0.712579
penalty is l1 C = 10^0		[[1370 190] [217 1373]]	acc = 0.870794 pre = 0.878439 rec = 0.863522
penalty is l1 C = 10^2		[[1342 218] [267 1323]]	acc = 0.846032 pre = 0.858533 rec = 0.832075
penalty is l1 C = 10^4		[[1330 230] [270 1320]]	acc = 0.841270 pre = 0.851613 rec = 0.830189

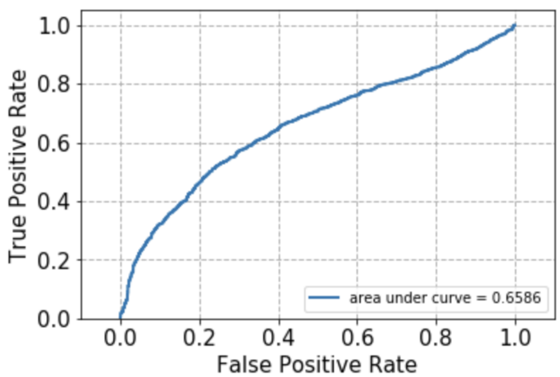
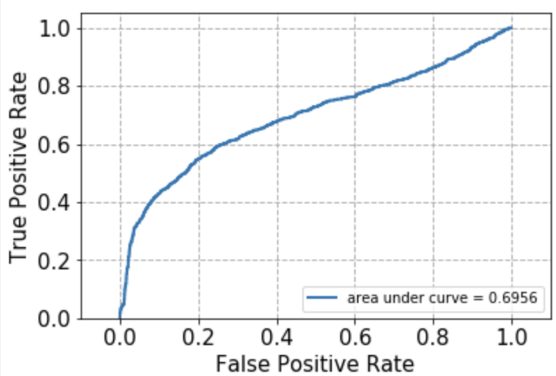
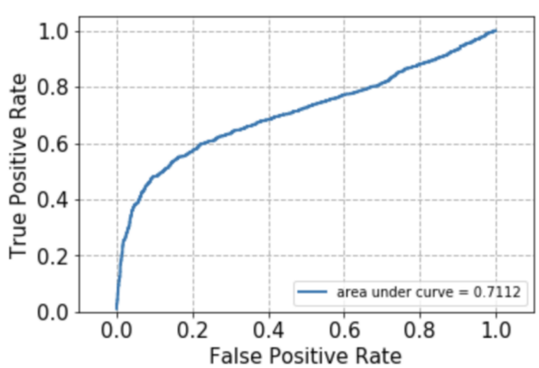
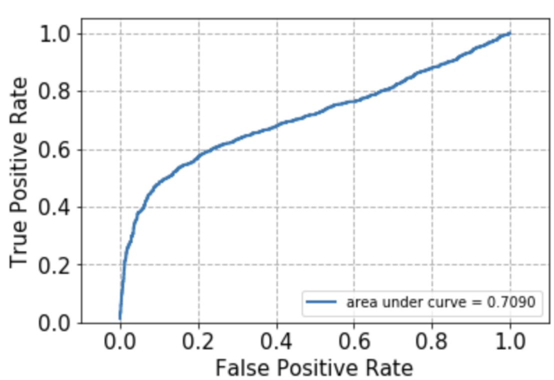
Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^6		[[1330 230] [270 1320]]	acc = 0.841270 pre = 0.851613 rec = 0.830189
penalty is l2 C = 10^-4		[[13 1547] [0 1590]]	acc = 0.508889 pre = 0.506854 rec = 1.000000
penalty is l2 C = 10^-2		[[1304 256] [75 1515]]	acc = 0.894921 pre = 0.855449 rec = 0.952830
penalty is l2 C = 10^0		[[1361 199] [163 1427]]	acc = 0.885079 pre = 0.877614 rec = 0.897484

Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l2 C = 10^2		[[1343 217] [242 1348]]	acc = 0.854286 pre = 0.861342 rec = 0.847799
penalty is l2 C = 10^4		[[1332 228] [270 1320]]	acc = 0.841905 pre = 0.852713 rec = 0.830189
penalty is l2 C = 10^6		[[1330 230] [270 1320]]	acc = 0.841270 pre = 0.851613 rec = 0.830189

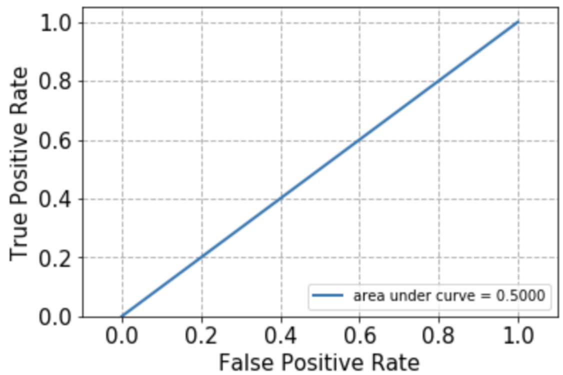
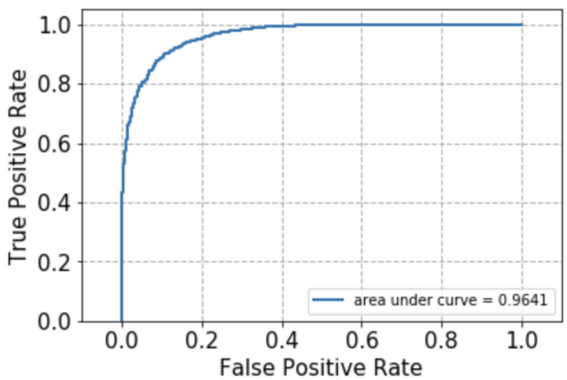
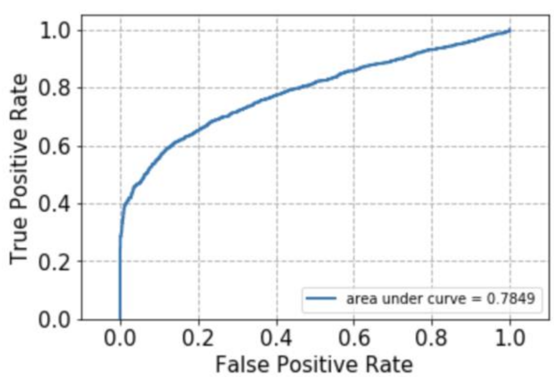
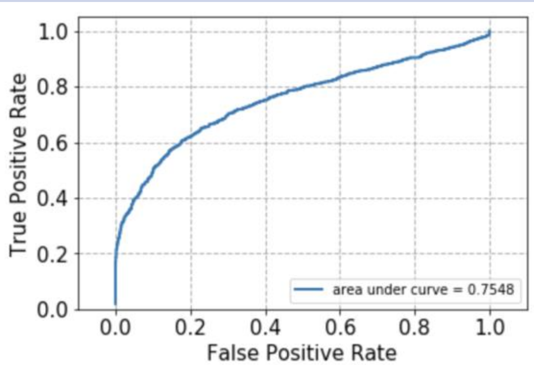
Class2: LSI， min_df = 2

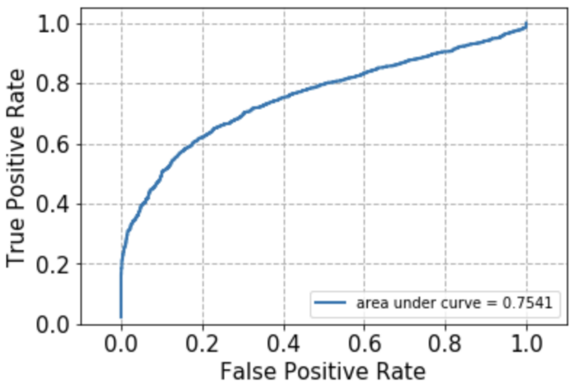
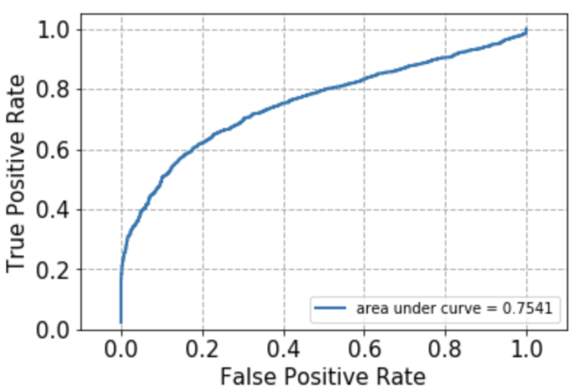
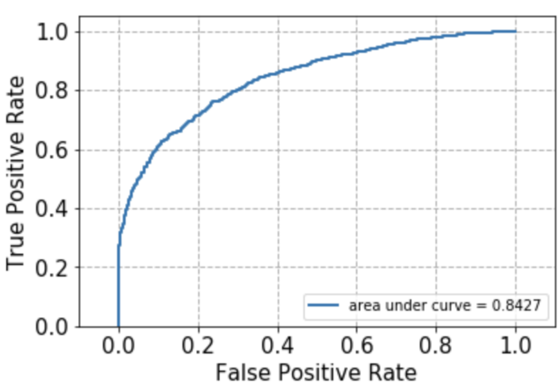
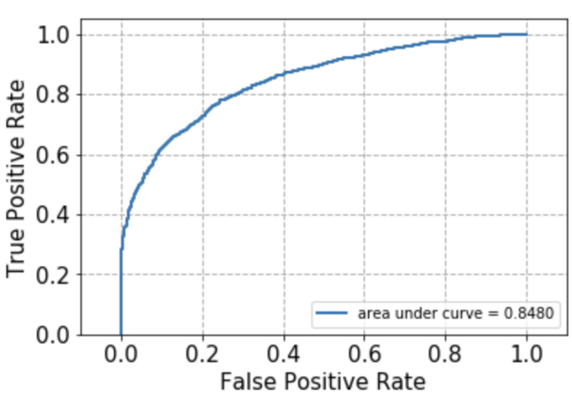
Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^-4		[[1560 0] [1590 0]]	acc = 0.495238 pre = 0.000000 rec = 0.000000
penalty is l1 C = 10^-2		[[1560 0] [1590 0]]	acc = 0.495238 pre = 0.000000 rec = 0.000000
penalty is l1 C = 10^0		[[993 567] [703 887]]	acc = 0.596825 pre = 0.610041 rec = 0.557862
penalty is l1 C = 10^2		[[1224 336] [668 922]]	acc = 0.681270 pre = 0.732909 rec = 0.579874

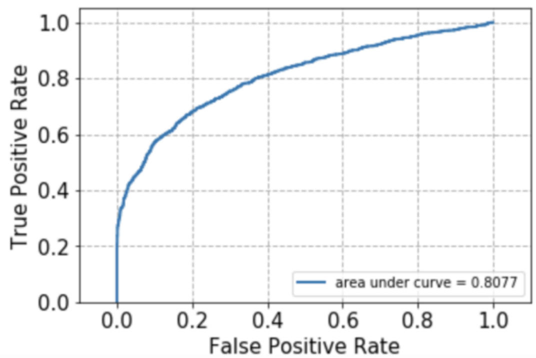
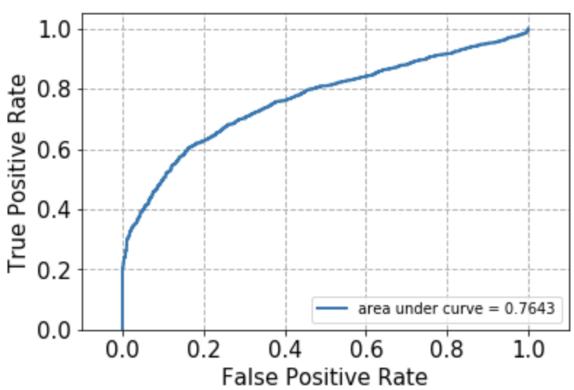
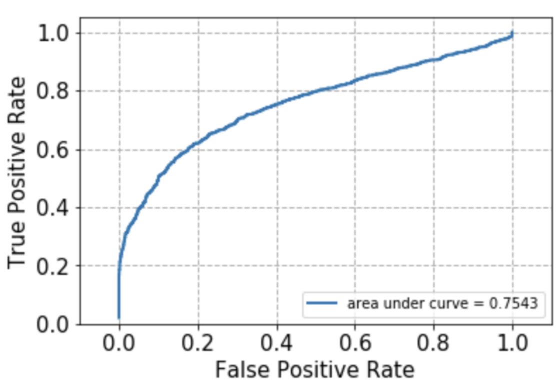
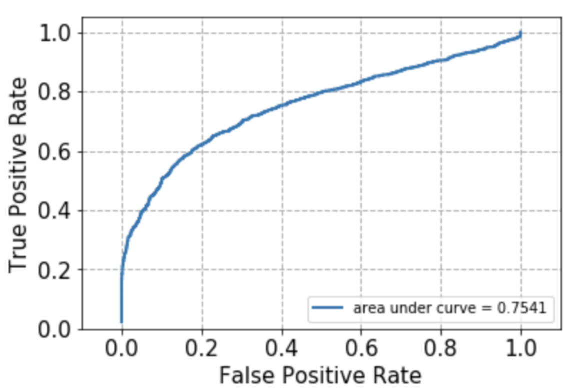
Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^4		[[1238 322] [670 920]]	acc = 0.685079 pre = 0.740741 rec = 0.578616
penalty is l1 C = 10^6		[[1238 322] [670 920]]	acc = 0.685079 pre = 0.740741 rec = 0.578616
penalty is l2 C = 10^-4		[[0 1560] [0 1590]]	acc = 0.504762 pre = 0.504762 rec = 1.000000
penalty is l2 C = 10^-2		[[22 1538] [35 1555]]	acc = 0.500635 pre = 0.502748 rec = 0.977987

Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l2 C = 10^0		[[1055 505] [659 931]]	acc = 0.630476 pre = 0.648329 rec = 0.585535
penalty is l2 C = 10^2		[[1124 436] [620 970]]	acc = 0.664762 pre = 0.689900 rec = 0.610063
penalty is l2 C = 10^4		[[1229 331] [667 923]]	acc = 0.683175 pre = 0.736045 rec = 0.580503
penalty is l2 C = 10^6		[[1238 322] [670 920]]	acc = 0.685079 pre = 0.740741 rec = 0.578616

Class3: LSI, min_df = 5

Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^-4		[[1560 0] [1590 0]]	acc = 0.495238 pre = 0.000000 rec = 0.000000
penalty is l1 C = 10^-2		[[1515 45] [437 1153]]	acc = 0.846984 pre = 0.962437 rec = 0.725157
penalty is l1 C = 10^0		[[1456 104] [793 797]]	acc = 0.715238 pre = 0.884573 rec = 0.501258
penalty is l1 C = 10^2		[[1404 156] [803 787]]	acc = 0.695556 pre = 0.834571 rec = 0.494969

Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l1 C = 10^4		[[1403 157] [800 790]]	acc = 0.696190 pre = 0.834213 rec = 0.496855
penalty is l1 C = 10^6		[[1403 157] [800 790]]	acc = 0.696190 pre = 0.834213 rec = 0.496855
penalty is l2 C = 10^-4		[[2 1558] [0 1590]]	acc = 0.505397 pre = 0.505083 rec = 1.000000
penalty is l2 C = 10^-2		[[1321 239] [517 1073]]	acc = 0.760000 pre = 0.817835 rec = 0.674843

Params	ROC Curve	Confusion Matrix	A&P&R
penalty is l2 C = 10^0		[[1401 159] [679 911]]	acc = 0.733968 pre = 0.851402 rec = 0.572956
penalty is l2 C = 10^2		[[1397 163] [786 804]]	acc = 0.698730 pre = 0.831437 rec = 0.505660
penalty is l2 C = 10^4		[[1403 157] [799 791]]	acc = 0.696508 pre = 0.834388 rec = 0.497484
penalty is l2 C = 10^6		[[1403 157] [800 790]]	acc = 0.696190 pre = 0.834213 rec = 0.496855

We find that when the penalty coefficient is very small, the classification cannot behave well because our model overfits to the training data, so it makes bad predictions to the testing data that never seen before. But with the increase of the coefficient, the ROC curve becomes better and better, and accuracy also improves, because the regularization is stronger, and we move from overfitting to “just right”. When this parameter is big enough, increase of the coefficient decreases the quality of classification in a tiny way, because our model goes from “just right” to underfitting. The penalty parameter tells the model how much we want to avoid misclassifying each training example. If penalty is large, the optimization will choose a hyperplane that allows little misclassification. On the other hand, if the penalty is small, we will get more misclassification, and the coefficient of hyperplane can be small.

As for the penalty, it is hard to tell which one is better just based on our experiment. Generally, it is L1 regularization preferable due the reason that it is possible to drive one or more weight values to zero with L1 regularization, while L2 can suppress the weight but not entirely to zero. The actual choice is data dependent. L1 is more suitable for non-sparse matrix while L2 has an analytical solution. In terms of error, L2 squares the error, which the model penalizes more on a huge misclassification, while L1 simply takes the absolute value of the error.

Part I+

In this section, we Perform Naïve Bayes classification and multiclass SVM classification with both One VS One and One VS the Rest methods described above and report the confusion matrix and calculate the accuracy, recall and precision of our classifiers.

Class1: LSI, min_df = 2

<table>
<tr><th>Type</th><th>Result</th><th>Confusion Matrix</th></tr>
<tr><td>Naive Bayes
-
One vs One</td><td><pre>
                        precision   recall f1-score   support

comp.sys.ibm.pc.hardware   0.64   0.64   0.64   392
   comp.sys.mac.hardware   0.60   0.54   0.57   385
            misc.forsale   0.61   0.70   0.65   390
  soc.religion.christian   0.98   0.94   0.96   398

             avg / total   0.71   0.71   0.71   1565

acuracy is 0.707348
</pre></td><td>[[252 73 63   4]
[85 206 93   1]
[51 61 274   4]
[4   1 18 375]]</td></tr>
</table>

Type	Result	Confusion Matrix
SVM-One vs One	precision recall f1-score support comp.sys.ibm.pc.hardware 0.80 0.86 0.83 392 comp.sys.mac.hardware 0.87 0.79 0.83 385 misc.forsale 0.86 0.89 0.87 390 soc.religion.christian 0.99 0.96 0.98 398 avg / total 0.88 0.88 0.88 1565 acuracy is 0.876677	[[339 28 25 0] [53 304 27 1] [26 16 347 1] [8 2 6 382]]
Naive Bayes-One vs Rest	precision recall f1-score support comp.sys.ibm.pc.hardware 0.64 0.63 0.63 392 comp.sys.mac.hardware 0.61 0.55 0.58 385 misc.forsale 0.61 0.69 0.65 390 soc.religion.christian 0.96 0.94 0.95 398 avg / total 0.71 0.70 0.70 1565 acuracy is 0.704792	[[246 66 69 11] [84 211 89 1] [50 66 270 4] [3 1 18 376]]
SVM-One vs Rest	comp.sys.ibm.pc.hardware 0.81 0.85 0.83 392 comp.sys.mac.hardware 0.86 0.79 0.82 385 misc.forsale 0.86 0.90 0.88 390 soc.religion.christian 0.99 0.98 0.98 398 avg / total 0.88 0.88 0.88 1565 acuracy is 0.879233	[[333 32 25 2] [51 303 30 1] [22 16 351 1] [4 1 4 389]]

Class2: NMF, min_df = 2

Type	Result	Confusion Matrix
Naive Bayes - One vs One	precision recall f1-score support comp.sys.ibm.pc.hardware 0.62 0.75 0.68 392 comp.sys.mac.hardware 0.72 0.64 0.68 385 misc.forsale 0.75 0.61 0.67 390 soc.religion.christian 0.90 0.98 0.94 398 avg / total 0.75 0.75 0.74 1565 acuracy is 0.746326	[[295 51 34 12] [90 247 39 9] [85 47 237 21] [4 0 5 389]]

Type	Result	Confusion Matrix
SVM-One vs One	precision recall f1-score support comp.sys.ibm.pc.hardware 0.63 0.87 0.73 392 comp.sys.mac.hardware 0.83 0.65 0.73 385 misc.forsale 0.82 0.77 0.80 390 soc.religion.christian 1.00 0.90 0.95 398 avg / total 0.82 0.80 0.80 1565 acuracy is 0.798083	[[340 26 26 0] [103 249 33 0] [65 22 302 1] [30 3 7 358]]
Naive Bayes-One vs Rest	precision recall f1-score support comp.sys.ibm.pc.hardware 0.65 0.70 0.67 392 comp.sys.mac.hardware 0.74 0.67 0.70 385 misc.forsale 0.72 0.68 0.70 390 soc.religion.christian 0.92 0.98 0.95 398 avg / total 0.76 0.76 0.76 1565 acuracy is 0.761022	[[276 58 49 9] [73 259 47 6] [75 32 266 17] [2 1 5 390]]
SVM-One vs Rest	precision recall f1-score support comp.sys.ibm.pc.hardware 0.70 0.82 0.75 392 comp.sys.mac.hardware 0.82 0.69 0.75 385 misc.forsale 0.80 0.81 0.81 390 soc.religion.christian 0.98 0.96 0.97 398 avg / total 0.83 0.82 0.82 1565 acuracy is 0.820447	[[323 33 34 2] [79 265 39 2] [51 22 315 2] [11 2 4 381]]

Conclusion

The key takeaway in this project is first to learn how to feature scale the dataset. We initially have a matrix of documents and terms, then it is converted to term frequencies. We later use two different dimension reduction methods, SLI and NMF to see how it can affect our result. In general, SLI preforms better than the NMF, because SLI use SVD to extract the most important parameters, i.e. the highest singular value, while NMF is simply split the origin matrix into two submatrices with dimension reduced. During the term counting process, we also have min_df=2 and min_df=5, which we ignore terms that that have a document frequency strictly lower than the given threshold when building the frequency matrix. We see that min_df=2 performs better than min_df=5 in general, because min_df=2 has a significantly more terms in the dataset, and a low frequency term in a document could also sometimes provide useful information. Second, we learned that there how to use machine learning models from scikit-learn, and how important the penalty parameter affects our prediction. We applied many models such as logistic regression with L1 and L2 regularization, linear SVM, and Naïve Bayes with various combinations of parameters and number of classes. By implementing different models, we see that it is important to choose the right model to our dataset. Logistic regression and linear SVM are both linear classifiers which are the most suitable for the linear dataset. However, Naïve Bayes is a non-linear classifier and the result is calculated based on the probability produced by the Bayes Theorem. For parameter tuning, we range from a very small penalty parameter to a very large one, which our corresponding machine learning model goes from overfitting to underfitting. Somewhere in the middle, there is a penalty value that provides the best accuracy, because at that penalty, our model is just right. In conclusion, data mining and machine learning are the arts of data analysis, it is crucial for one to choose the best features, models and parameters to make great predictions.