EE219 Project 1

Classification Analysis on Textual Data

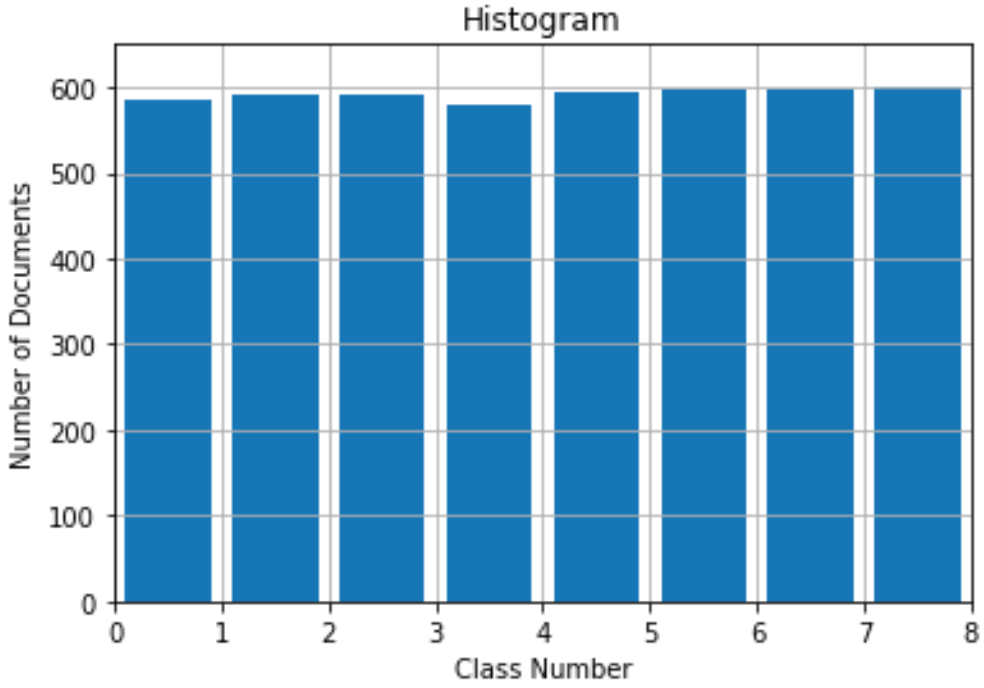
Report

Yi Jia 805033204

Shuojian Ye 904946811

Part A

Plot a histogram of the number of training documents per class to check if they are evenly distributed.



From the histogram, we can see that all the eight classes have balanced document number.

Part B

In this part, we did the following extraction.

1. Choose the min_df=3 and min_df=5
2. Applied combined stop word from nltk and text.
3. Applied lemmatizer for stemming.
4. Excluded punctuation.
5. Tokenized documents into words.

(Number of doc, Number of terms)

min_df=3	min_df=5
(4732, 16292)	(4732, 10396)

Part C

Find the 10 most significant terms in each of the following classes with respect to TFxICF measure. Note that in this part of your assignment, n_class is 20.

The results are as follows.

```
comp.sys.ibm.pc.hardware has the following 10 most significant terms:
[[u'scsi' u'0']
 [u'drive' u'1']
 [u'edu' u'2']
 [u'ide' u'3']
 [u'line' u'4']
 [u'use' u'5']
 [u'com' u'6']
 [u'subject' u'7']
 [u'organization' u'8']
 [u'controller' u'9']]
comp.sys.mac.hardware has the following 10 most significant terms:
[[u'edu' u'0']
 [u'line' u'1']
 [u'mac' u'2']
 [u'subject' u'3']
 [u'organization' u'4']
 [u'apple' u'5']
 [u'use' u'6']
 [u'quadra' u'7']
 [u'scsi' u'8']
 [u'problem' u'9']]
misc.forsale has the following 10 most significant terms:
[[u'edu' u'0']
 [u'line' u'1']
 [u'sale' u'2']
 [u'subject' u'3']
 [u'organization' u'4']
 [u'new' u'5']
 [u'post' u'6']
 [u'com' u'7']
 [u'university' u'8']
 [u'offer' u'9']]
soc.religion.christian has the following 10 most significant terms:
[[u'god' u'0']
 [u'edu' u'1']
 [u'christian' u'2']
 [u'jesus' u'3']
 [u'say' u'4']
 [u'church' u'5']
 [u'subject' u'6']
 [u'people' u'7']
 [u'line' u'8']
 [u'know' u'9']]
```

Part D

Apply LSI to do dimension reduction.

In [311]:
```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=50, n_iter=5, random_state=42)
X_train_lsi=svd.fit_transform(X_train_tfidf)
X_train_lsi
```

Out[311]:
```
array([[ 0.12727739,  0.10087277,  0.02831868, ..., -0.00153826,
        -0.04095367,  0.00891065],
       [ 0.12633974,  0.12898094,  0.03774992, ...,  0.1491302 ,
         0.28068439,  0.0533624 ],
       [ 0.1770882 , -0.02824277,  0.0002872 , ...,  0.01672192,
        -0.02992802,  0.01647884],
       ...,
       [ 0.13645309,  0.11315775,  0.0277115 , ...,  0.0260539 ,
         0.03039151, -0.02383439],
       [ 0.20046062,  0.15859621,  0.03016406, ...,  0.03259629,
         0.01178131,  0.01282113],
       [ 0.12063002, -0.02342554, -0.04576324, ...,  0.03987772,
        -0.00619211,  0.01265135]])
```

NMF

In [312]:
```
X_train_lsi.shape
```

Out[312]:
```
(4732, 50)
```

Apply NMF to do dimension reduction.

In [313]:
```
from sklearn.decomposition import NMF

model = NMF(n_components=50, init='random', random_state=0)
X_train_nmf = model.fit_transform(X_train_tfidf)
X_train_nmf
```

Out[313]:
```
array([[ 0.08328405,  0.02441659,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.01062915,  0.        ,  0.03132605, ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [ 0.01868059,  0.05391546,  0.        , ...,  0.        ,
         0.        ,  0.00025616],
       [ 0.14854722,  0.0623291 ,  0.00076439, ...,  0.00108973,
         0.        ,  0.00125722],
       [ 0.        ,  0.        ,  0.00807991, ...,  0.        ,
         0.        ,  0.00222103]])
```

In [314]:
```
X_train_nmf.shape
```

Out[314]:
```
(4732, 50)
```

Part E

1) **C=0.001, min_df=3, dim_reduce=LSI**

   Confusion matrix:

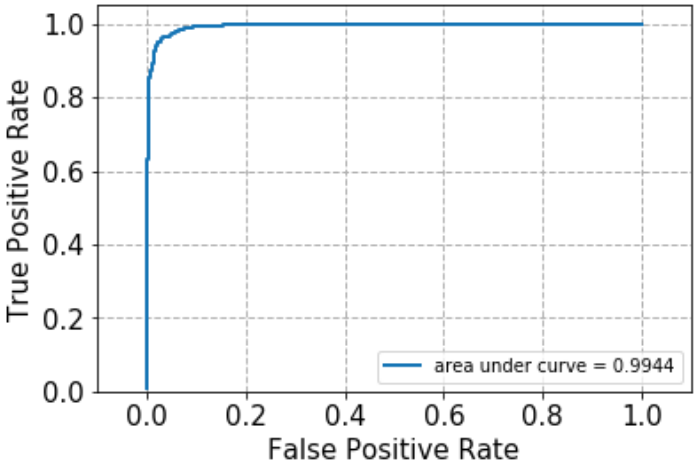
   1218 6

   372 1554

   Recall: 0.766037735849

   Accuracy: 0.880

   Precision: 0.995098039216



2) **C=0.001, min_df=5, dim_reduce=LSI**

   Confusion Matrix:

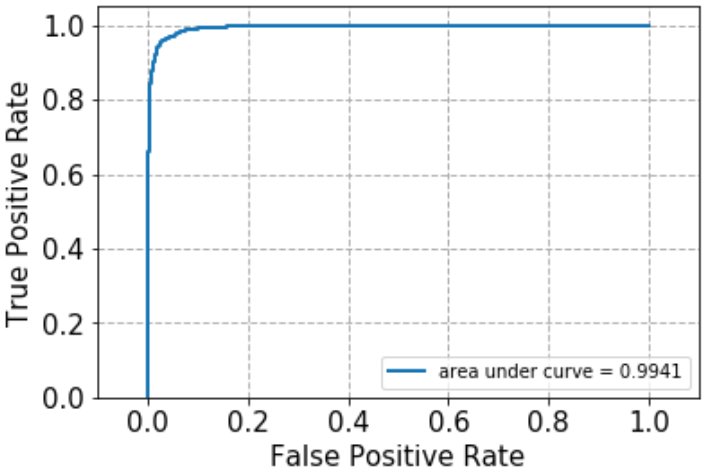
   1319 8

   271 1552

   Recall: 0.829559748428

   Accuracy: 0.911428571429

   Precision: 0.993971363979



3) **C=0.001, min_df=3, dim_reduce=NMF**

Confusion Matrix:

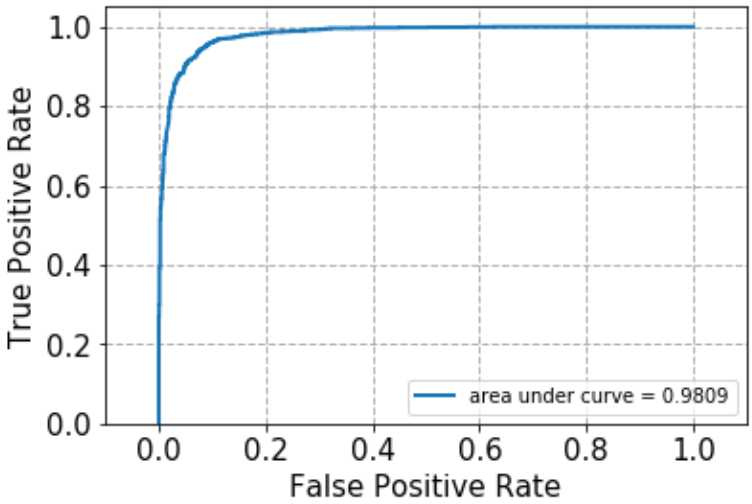
1161 22

429 1538

Recall: 0.730

Accuracy: 0.8568

Precision: 0.9814



4) **C=1000, min_df=3, dim_reduce=LSI**

Confusion Matrix:

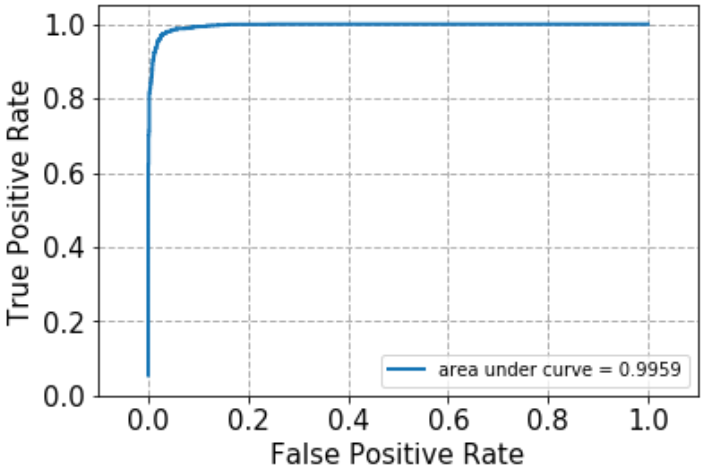
1557 57

33 1503

recall: 0.979245283019

accuracy: 0.971428571429

precision: 0.96468401487



5) C=1000, min_df=5, dim_reduce=LSI

Confusion Matrix:

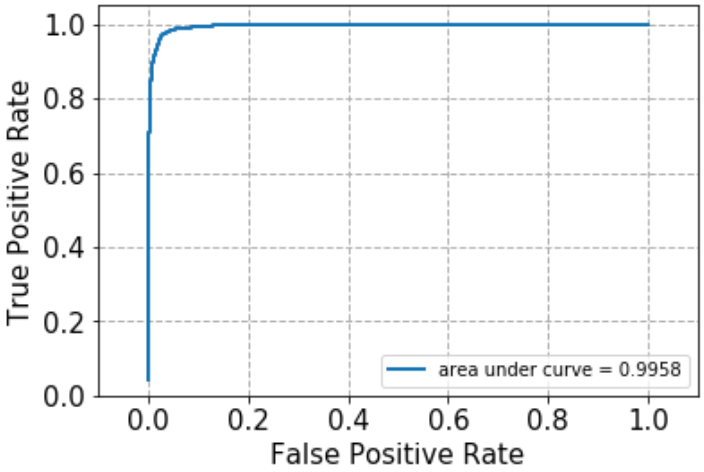
1560 63

30 1497

recall: 0.981132075472

accuracy: 0.970476190476

precision: 0.961182994455



6) C=1000, min_df=3, dim_reduce=NMF

Confusion matrix:

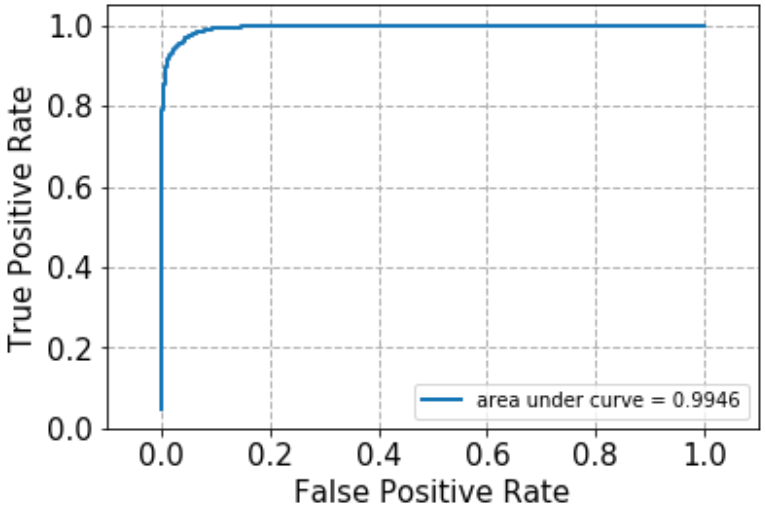
1538 70

52 1490

Recall: 0.967

Accuracy: 0.961

Precision: 0.956



Comparison of SVM classifier based on C, MIN_DF and dim_reduction type.

SVM Type	**Scores**	MIN_DF=3, LSI	MIN_DF=5, LSI	MIN_DF=3,NMF
Soft Margin C=0.001	**recall**	0.766	0.830	0.730
	accuracy	0.880	0.911	0.856
	precision	0.995	0.994	0971
Hard Margin C=1000	**recall**	0.979	0.981	0.967
	accuracy	0.971	0.970	0.961
	precision	0.965	0.961	0.956

From the table, we can see the following conclusions:

1) Hard Margin SVM classifier is better than Soft Margin SVM classifier.
2) Dimension reduction using LSI will get better results than NMF.
3) The result is similar when choose min_df=3 and min_df=5. It is not worth to set low min_df.

Part F

Using a 5-fold cross-validation, find the best value of the parameter C in the range {10^k }, where $-3 \leq k \leq 3$.

k	Accuracy
-3	0.505
-2	0.505
-1	0.969
0	0.972
1	**0.975**
2	0.974
3	0.974

Therefore, we use k=1, C=10.

C=10	min_df=3, LSI	min_df=5, LSI	min_df=3, NMF
Confusion Matrix	1562 52 28 1508	1560 55 30 1505	1519 87 71 1473
Recall	0.982	0.981	0.955
Accuracy	0.975	0.973	0.949
Precision	0.968	0.966	0.946

From the table, we can see:

1) LSI performs better than NMF.
2) Change of min_df does not change much in performance.

Part G

Multinomial Naive Bayes Classifier

Min_df=3, NMF

Confusion Matrix:

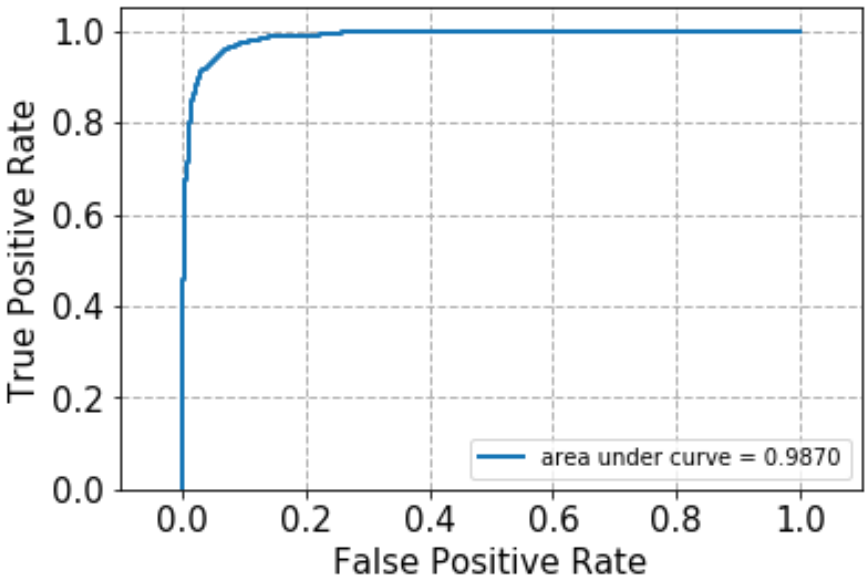
1414 146

43 1547

Recall: 0.972

Accuracy: 0.940

Precision: 0.914



Part H

logistic regression classifier LRC

1) **min_df=3, lsi**

Confusion Matrix:

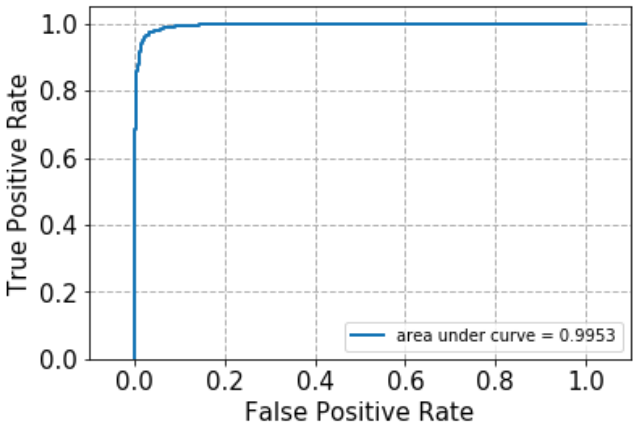
[[1486 74]

[34 1556]]

recall: 0.978

accuracy: 0.965

precision: 0.954



2) **min_df=5, lsi**

Confusion Matrix:

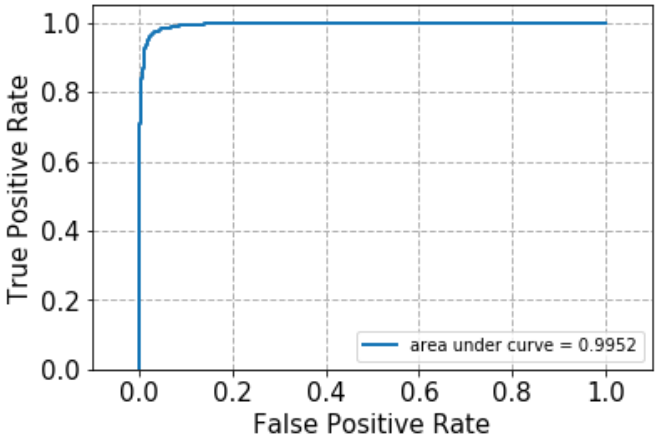
[[1486 74]

[32 1558]]

Recall: 0.979

Accuracy: 0.966

Precision: 0.954



3) min_df=3, nmf

Confusion Matrix:

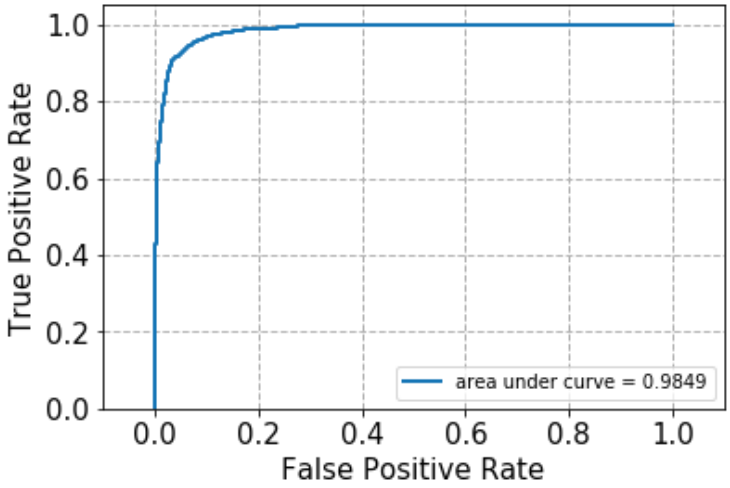
[[1447 113]

[80 1510]]

Recall: 0.949

Accuracy: 0.938

Precision: 0.930



LRC	min_df=3, LSI	min_df=5, LSI	min_df=3, NMF
Confusion Matrix	[[1486 74] [34 1556]]	[[1486 74] [32 1558]]	[[1447 113] [80 1510]]
Recall	0.978	0.979	0.949
Accuracy	0.965	0.966	0.938
Precision	0.954	0.954	0.930

From this table, we can see:

1) LSI performs better than NMF.

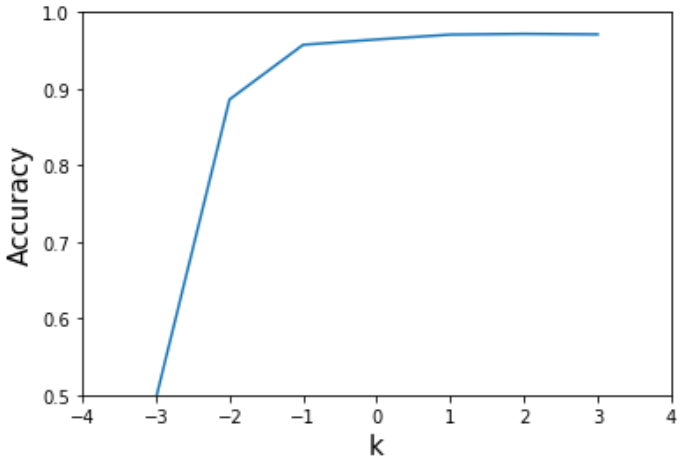
2) Change of min_df does not change much in performance.

Part I

L1 Regularizations

Parameter C: inverse of regulation strength in {10^k}, k=[-3,3] k belong to Z.

1) min_df=3, lsi



We can see when C increases, Accuracy increases, test error decreases.

The best value at k=2, C=100

Confusion Matrix:

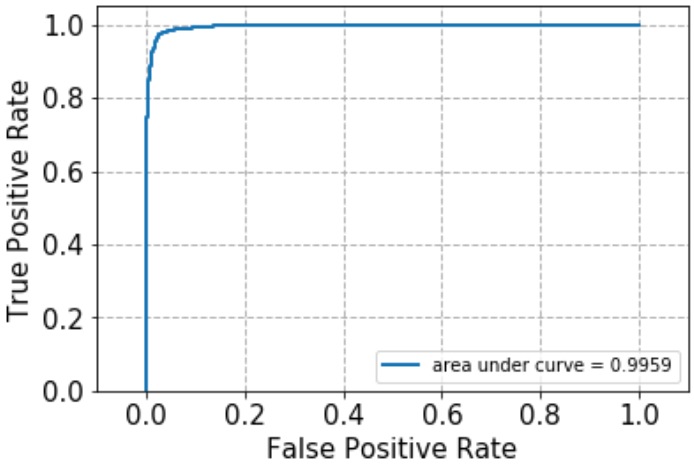
[[1502 58]

[31 1559]]

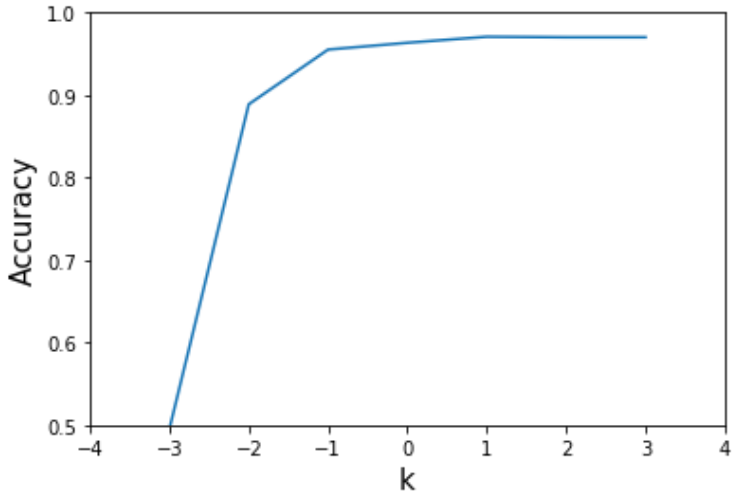
recall: 0.981

accuracy: 0.972

precision: 0.964



2) min_df=5, lsi



We can see when C increases, accuracy increases, test error decreases.

When C reaches 1, accuracy tends to be stable around 0.97.

Best value at k=1, C=10

Confusion Matrix:

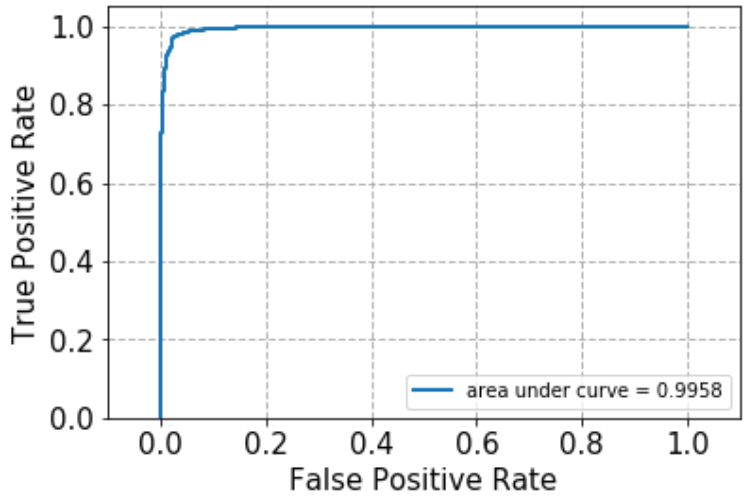
[[1500 60]

[32 1558]]

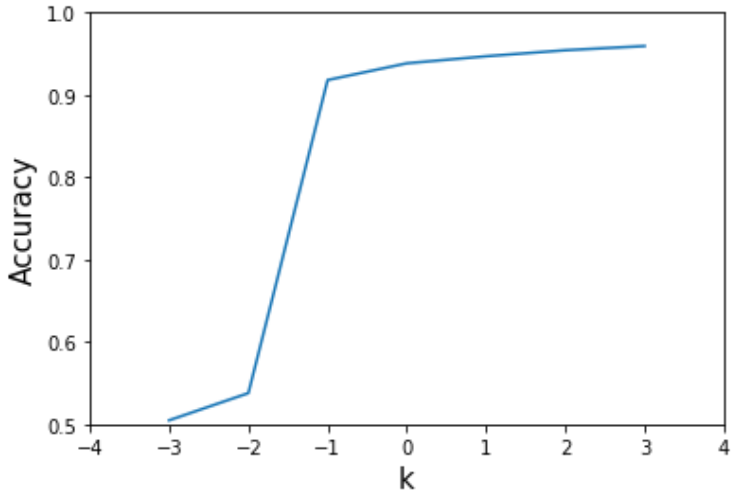
recall: 0.979

accuracy: 0.971

precision: 0.963



3) min_df=3, nmf



We can see when C increases, Accuracy increases, test error decreases.

Best value: k=3, C=1000

Confusion Matrix:

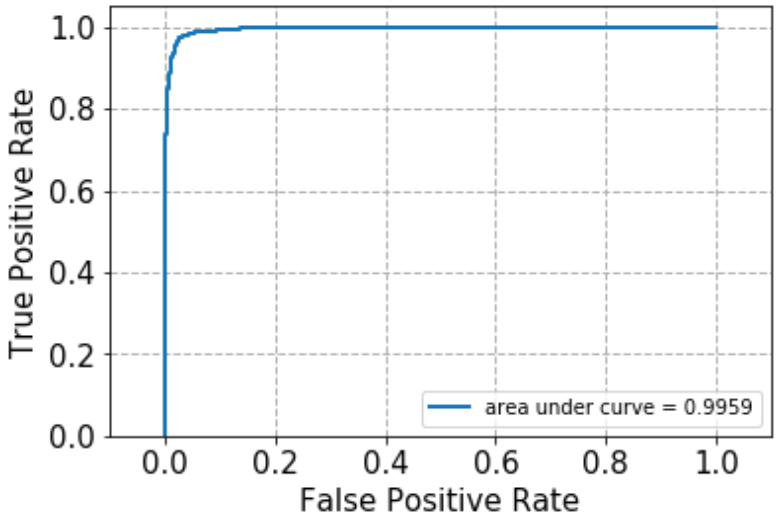
[[1480 80]

[48 1542]]

recall: 0.969

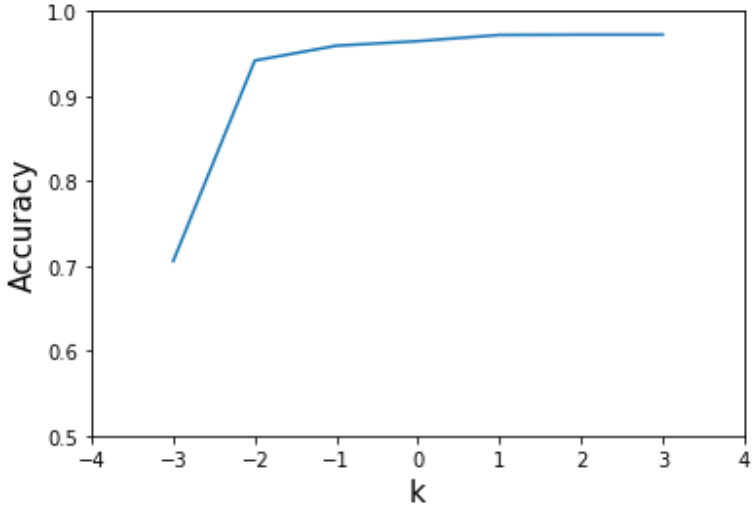
accuracy: 0.959

precision: 0.951



L2 Regularizations

1) min_df=3, lsi



We can see when C increases, Accuracy increases, test error decreases.

When C reaches 10, accuracy tends to be stable.

Best value at k=2, C=100.

Confusion Matrix:

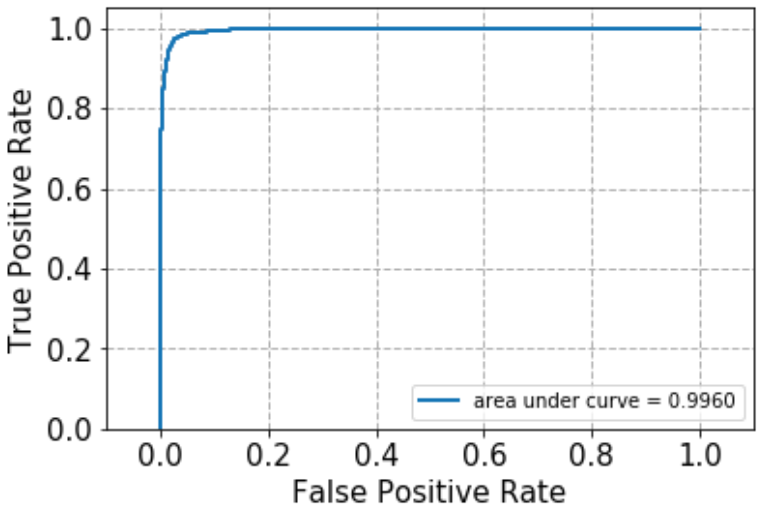
[[1505 55]

[32 1558]]

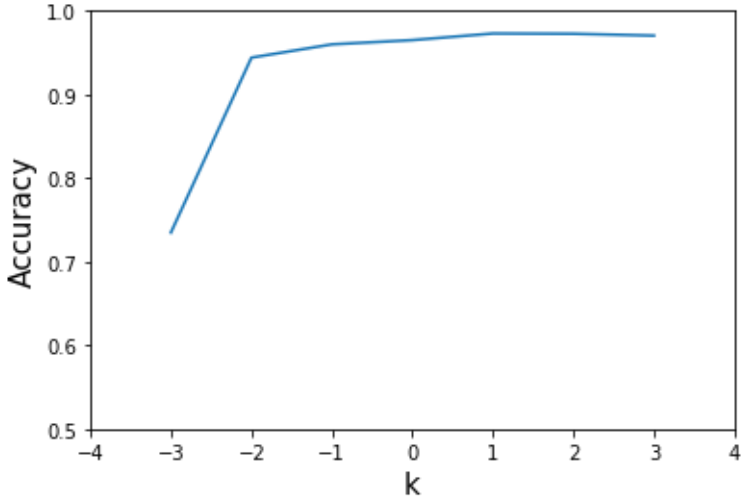
recall: 0.979

accuracy: 0.972

precision: 0.966



2) min_df=5, lsi



We can see when C increases, Accuracy increases, test error decreases.

When C reaches 1, accuracy tends to be stable.

Best value: k=1, C=10

Confusion Matrix:

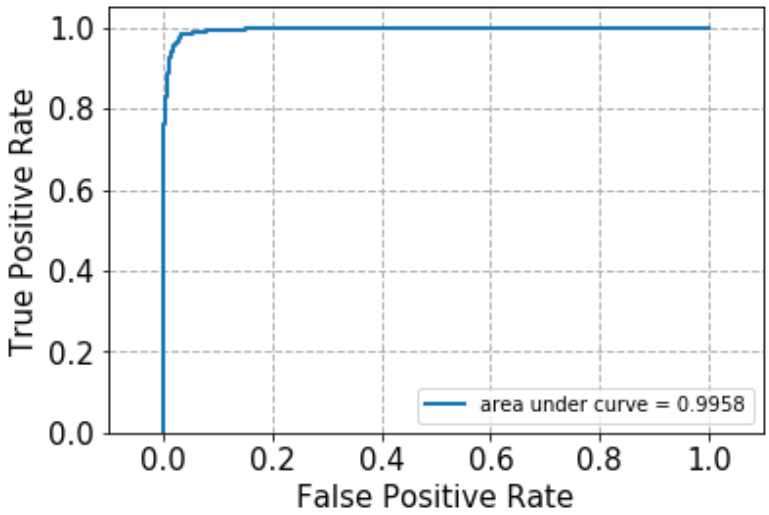
[[1502 58]

[27 1563]]

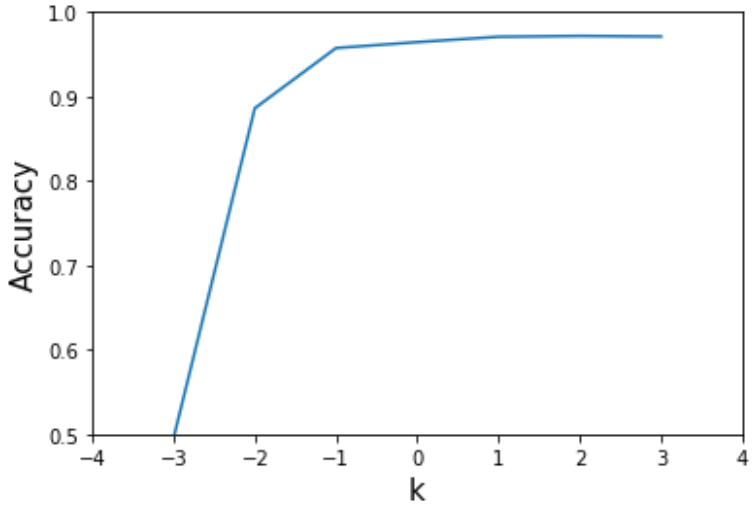
recall: 0.983

accuracy: 0.973

precision: 0.964



3) min_df=3, nmf



We can see when C increases, Accuracy increases, test error decreases.

When C reaches 10, accuracy tends to be stable.

Best value: k=2, C=100

Confusion Matrix:

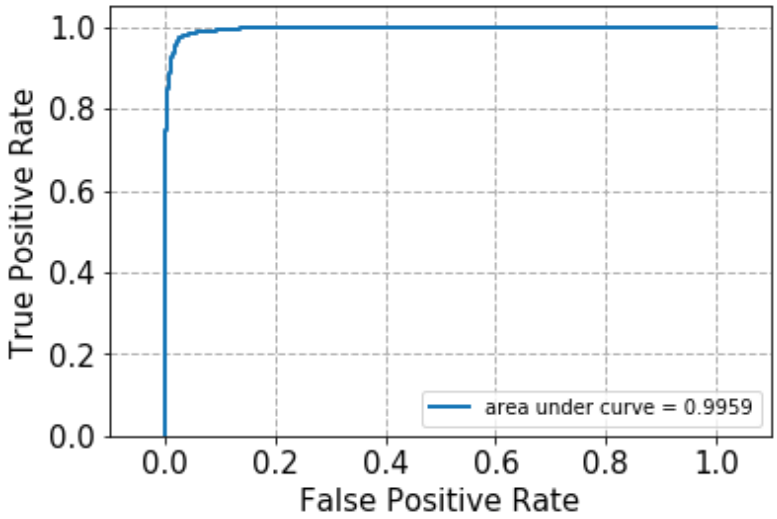
[[1502 58]

[31 1559]]

recall: 0.981

accuracy: 0.972

precision: 0.964



The reason for we want to analyze different regularization is that we do not know how the fitting for training set influence the test set. Literally, larger C value will gives us larger minimum distance from data to hyperplane, but this does not means

that the hyperplane really works in actual condition. A smaller C value could possibly gives us better fitting for test set. Thus, to get a better fitting for the future data, we want to find appropriate coefficient for each C and penalty value choosing from l1 and l2.

Multiclass

1)Multinomial Naive Bayes

I) min_df = 3 NMF

Confusion Matrix:

[[317 41 29 5]

[111 234 38 2]

[35 9 342 4]

[4 0 1 393]]

recall: 0.822

accuracy: 0.822

precision: 0.830

II) min_df = 3 SVD

Confusion Matrix:

[[339 34 19 0]

[28 339 18 0]

[35 20 334 1]

[5 1 3 389]]

recall: 0.895

accuracy: 0.895

precision: 0.897

2) One vs One SVM

I) NMF

Confusion Matrix:

[[312 60 20 0]

[93 274 18 0]

[41 25 324   0]

[11   7   3 377]]

recall: 0.822

accuracy: 0.822

precision: 0.831

II) SVD

Confusion Matrix:

[[323 45 24   0]

[38 323 24   0]

[24 19 345   2]

[7   2   2 387]]

recall: 0.881

accuracy: 0.881

precision: 0.881

3) One vs Rest SVM

I) NMF

Confusion Matrix:

[[321 26 45   0]

[93 258 32   2]

[51 12 325   2]

[5   1 23 369]]

recall: 0.813

accuracy: 0.813

precision: 0.827

II) SVD

Confusion Matrix:

[[323 45 24   0]

[38 323 24   0]

[24 19 345   2]

[7   2   2 387]]

recall: 0.881

accuracy: 0.881

precision: 0.881