EE 219 Project 1

Classification Analysis on Textual Data

Introduction

In this project, we work with a popular data set for experiments in text applications of machine learning techniques, “20 Newsgroups” dataset. It is a collection of approximately 20,000 newsgroup documents with 20 evenly distributed newsgroups, each corresponding to a different topic.

Throughout this project, we will:

1. Construct tf-idf representations of textual data, and apply two dimensionality reduction methods: PCA & NMF.
2. Implement various classification methods (e.g. SVM, LR, NB) with complete pipelines to classify the newsgroup.
3. Evaluate and diagnose classification results.

Question 1: Plot the distribution of the size of the dataset.

In a classification problem, imbalanced data size of different classes should be avoided. To see whether we need to modify the size our dataset, we can first visualize the number of training documents from different categories with a histogram. As we can see from fig. 1, the dataset is balanced and so in this case we do not need to modify the data.

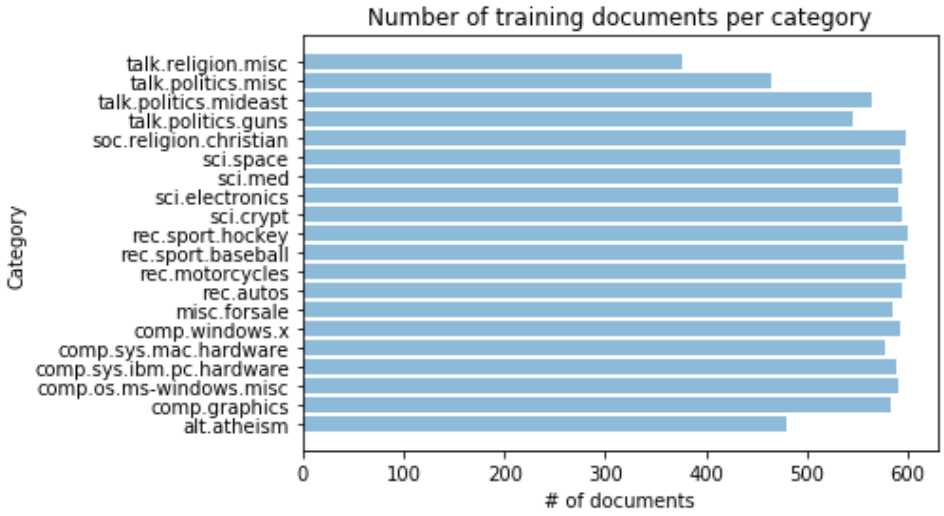


Fig. 1. Histogram of the number of training documents

Question 2: Extract features from the textual data.

Some words are too common to be useful in classification (stopwords), and the numbers are usually not important for determining the category. Meanwhile, we want to lemmatize the words to their stem word.Thus, the documents are pre-possessed by lemmatization and deleting numbers and stopwords, and then translated into TD-IDF matrices.

The size of training and test matrix are as follow:

Training Matrix	(4732, 16593)
Test Matrix	(3150, 16593)

Question 3: Dimensionality Reduction with NMF and LSI.

The original TF-IDF matrices are high dimensional, leading to the poor performance of learning algorithms. Thus, before the classification, we need to reduce the dimension while keeping the features of TF-IDF matrices. We applied Non-negative Matrix Factorization (NMF) and Latent Semantic Indexing (LSI), both of which minimize mean squared residual between the original data and a reconstruction from its low-dimensional approximation. We then calculated the Frobenius norm of the difference between original matrix and “reconstructed matrix”.

The results are as follow:

F-Norm	Train Set	Test Set
NMF	3938.93750384	2687.4999548
LSI	3895.38427120	2676.4614317

As we can see from the result, for both training and test set, LSI performs better than NMF. Usually LSI performs better than NMF, and reasoning is as follows.
Given two facts:

(1) SVD is the best approximation for k-rank approximation.
(2) Decrease rank can only produce worse approximation ($min\ \{\ ||\ X- X'||\ \}$ is larger, where X' is the approximation matrix)

And for NMF, the optimization problem is:

$minimize\ \ ||\ X - WH\ ||$

$s.t.\ W >= 0,\ H >= 0$

we may find an invertible matrix A,

s.t. $W = U_k\Sigma_k A >= 0,\ and\ H = A^{-1}V_k >= 0$

where $U_k,\ \Sigma_k$ and V_k are the SVD components for matrix X.

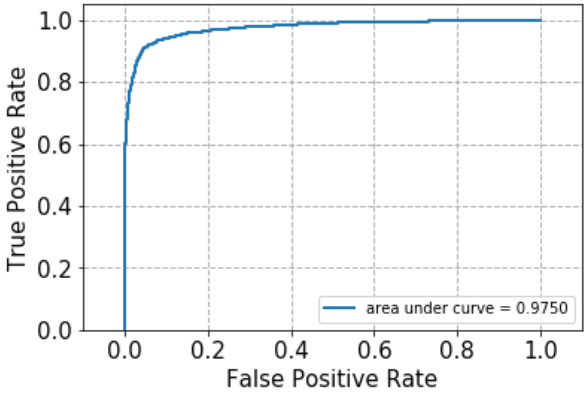
Then we have $||X - WH|| = ||X - U_k\Sigma_k AA^{-1}V_k||$, and this becomes a m-rank approximation problem, where m <= k. Given facts (1) and (2), we know $||X - WH||$ is greater than or equal to $||X - U_k\Sigma_k AA^{-1}V_k||$.

Question 4: Hard SVM and Soft SVM

(1) Train two linear SVM with hard margin and soft margin

Soft Margin (C = 0.0001)

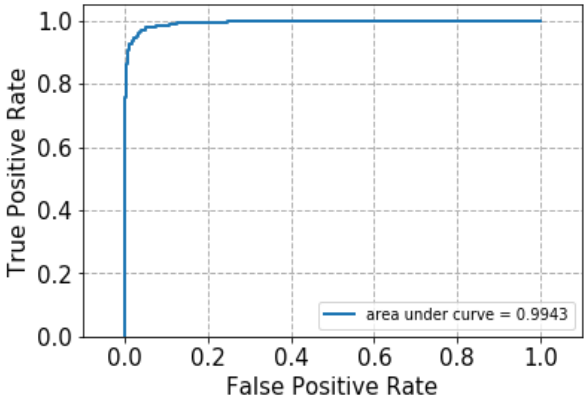
NMF



confusion_matrix =
[0 1590]
[0 1560]

accuracy score	0.504761904762
recall score	1.0
precision score	0.504761904762
F1 score	0.670886075949

LSI

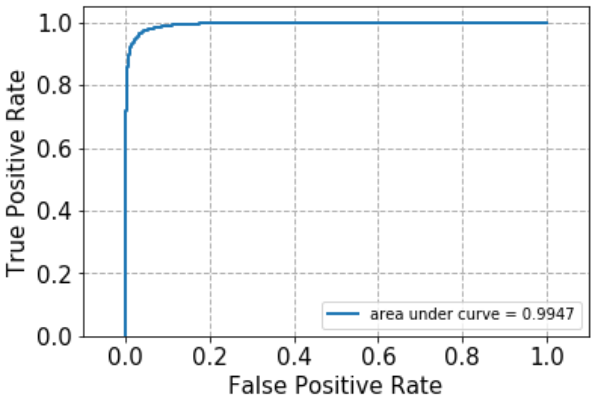


confusion_matrix =
[0 1590]
[0 1560]

accuracy score	0.504761904762
recall score	1.0
precision score	0.504761904762
F1 score	0.670886075949

Hard Margin (C = 1000)

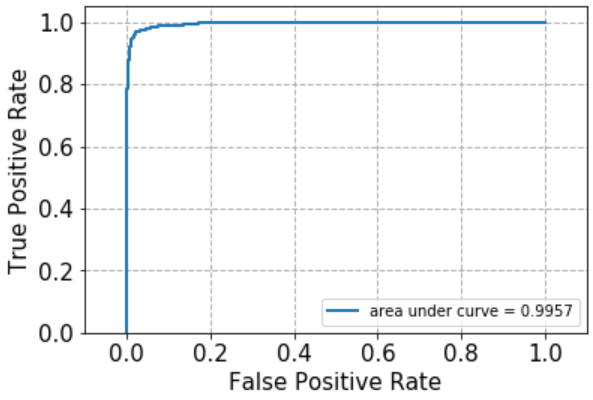
NMF



confusion_matrix =
[1539 51]
[61 1499]

accuracy score	0.964444444444
recall score	0.960897435897
precision score	0.967096774194
F1 score	0.963987138264

LSI



confusion_matrix =
[1556 34]
[51 1509]

accuracy score	0.973015873016
recall score	0.967307692308
precision score	0.97796500324
F1 score	0.972607154367

Analysis:

Based on the results, it turns out that hard margin SVM performs much better than the soft-margin one. This is because the C parameter for soft margin SVM is too small, and thus all the labels are predicted to be 'true', i.e. belonging to 'recreational'.

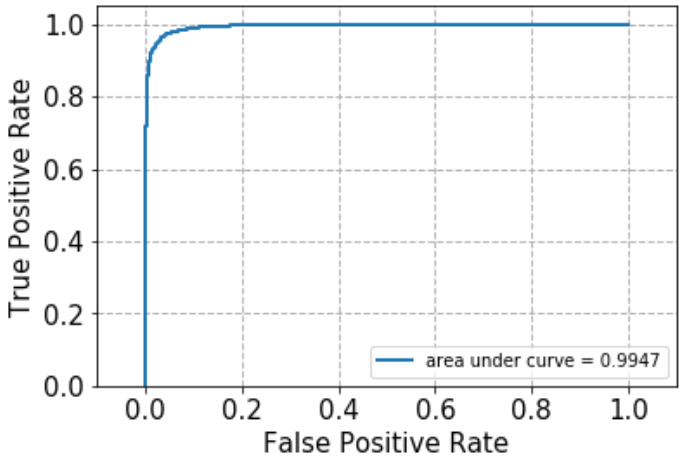
And it's worth noticing that if we don't perform dimensional reduction, the accuracy score for soft-margin SVM is roughly 0.85, which is much better than that after dimensional reduction. This might be because dimensional reduction in this case (with very small c) removes too much information. As a result, the margin between two classes are not clear.

For soft-margin SVM, the accuracy is roughly 0.5, i.e. no classification at all. As mentioned earlier, this is because the parameter c is set to be too small, and the penalty of misclassification is neglectable. But the ROC looks good (because the area under the curve is still roughly 0.99). This doesn't conflict with other metrics because this classifier classifies all the data to have label 'true'. Therefore, all true positive will be identified as positive, and true positive rate is always 1.

(2) use cross validation to choose best c (w.r.t. average accuracy)

By iterating through all possible c in [0.001, 0.01, 0.1, 1, 10, 100, 1000], we found that the best C parameter is 1000, which happens to be the hard margin SVM, with score (average accuracy) = 0.97296726504751851

Results are as follows:



confusion_matrix =
[1539 51]
[61 1499]

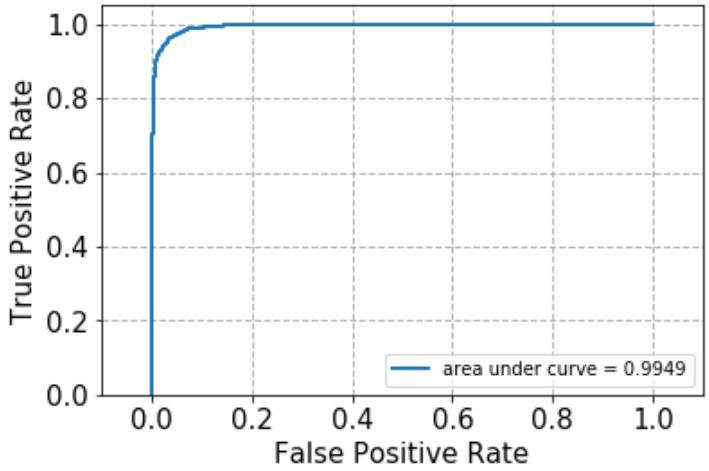
Accuracy Score	Recall Score	Precision Score	F1 Score
0.964444444444	0.960897435897	0.967096774194	0.963987138264

Notice here we use LSI as the way to reduce dimension because the matrix provided by LSI is a better approximation for the original matrix (as mentioned in Question 3)

Question 5: Logistic Regression

(1) No regularization

To use sklearn.linear_model.LogisticRegression, we need to set C to be large enough, and use l2 penalty to approximate no regularization. Results are as follows.



Accuracy Score	Recall Score	Precision Score	F1 Score
0.962222222222	0.957692307692	0.965740142211	0.961699388478

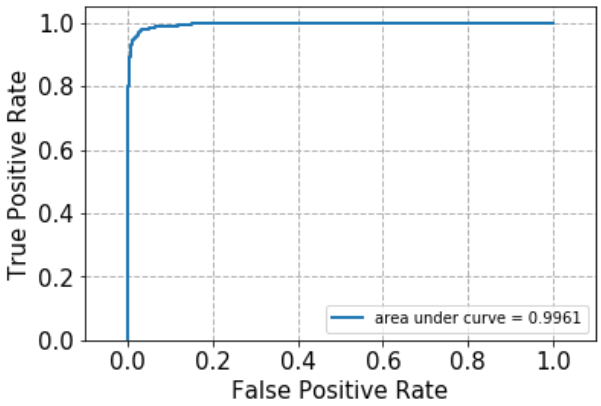
(2) With regularization

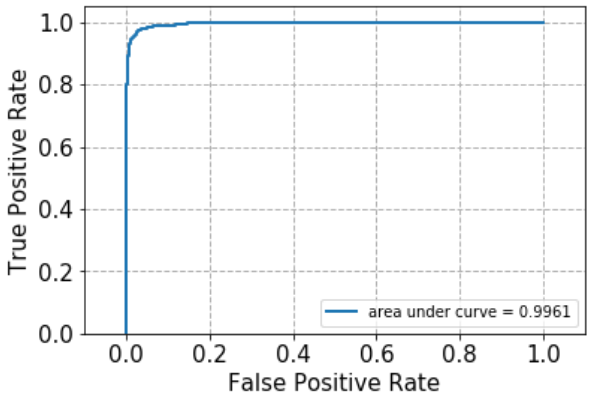
a) Find the best regularization strength for both l1 and l2

By iterating all possible C in [0.001, 0.01, 0.1, 1, 10, 100, 1000] for both l1 and l2, we got the results:

Best C for l1 is 100 with average accuracy = 0.973812038015
Best C for l2 is 100 with average accuracy = 0.973600844773

l1 **l2**





confusion_matrix =
[1553 37]
[62 1498]

accuracy score	0.968571428571
recall score	0.960256410256
precision score	0.975895765472
F1 score	0.968012924071

confusion_matrix =
[1554 36]
[61 1499]

accuracy score	0.969206349206
recall score	0.960897435897
precision score	0.97654723127
F1 score	0.968659127625

b) Compare the performance for no-reg(no-regularization), l1, l2

In terms of accuracy, no-reg < l1 ≈ l2
In terms of recall score, no-reg < l1 ≈ l2
In terms of precision score, no-reg < l1 ≈ l2
Over all, (comparing F1 score), no-reg < l1 ≈ l2

To sum up, l1 and l2 have similar performance, and share the same best C parameter (1000). And both l1 and l2 regularization have better performance than the non-regularized one.

c) How does the regularization parameter affect the test error?

According to the previous results, we can conclude that as the regularization parameter C increases, the performance of the Logistic Regression Classifier becomes better (for both NMF and LSI). But when C continues to increase after a certain point, the performance starts to decrease. When C is infinitely large, this actually turns into non-regularized case, and the performance is not as good as regularized ones with best C (C = 100).

One may consider the following properties to decide when to use l1 regularization and l2 regularization.

L1 Regularization	L2 Regularization
Computational more efficient due to analytical solutions	Less Efficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No-feature selection	Built-in feature selection

L2 regularization technique adds “squared magnitude” of coefficient as penalty term to the loss function.

Here, if regularization parameter is zero then you can imagine we get back OLS.

However, if parameter is very large then it will add too much weight and it will lead to under-fitting.

L1 regularization technique adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

Again, if regularization parameter is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

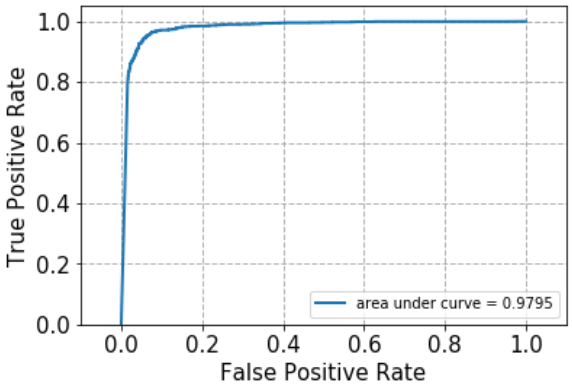
The key difference between these techniques is that l1 shrinks the less important features’ coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

d) What's the difference between logistic regression and linear SVM in finding boundary? why their performance differ?

Both methods can be seen as hyper plane separators, and the main difference between them is the loss function they use: SVM minimizes hinge loss while logistic regression minimizes logistic loss. Since logistic loss diverges faster, it will be more sensitive to outliers. linear SVMs are confidently classifies training since the loss can reach 0, while each sample in logistic regression has a non-zero influence on the decision boundary. That is why SVM perform marginally better than logistic regression.

Question 6: Naive Bayes

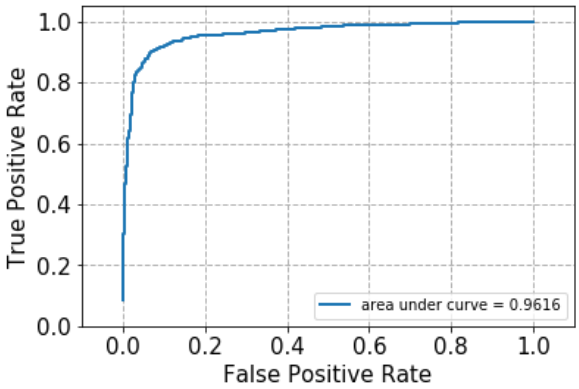
NMF



confusion_matrix =
[1521 69]
[127 1433]

accuracy score	0.906349206349
recall score	0.857692307692
precision score	0.948263642807
F1 score	0.900706832716

LSI



confusion_matrix =
[1517 73]
[222 1338]

accuracy score	0.906349206349
recall score	0.857692307692
precision score	0.948263642807
F1 score	0.900706832716

Question 7: Grid search of parameters

We did two grid searches, for data with headers and footers and without headers and footers respectively. The best parameters are with respect to average accuracy using 5-fold cross-validation.

In each grid search, the possible combinations we use are as follows:

Procedures	Options

min_df	3 or 5
lemmatization	use or not use
dimension reduction	LSI or NMF
classifier	SVM (C=1000) l1 Logistic Regression (C=100) l2 Logistic Regression(C=10)

(1) With headers and footers

The best choices are:

min_df	lemmatization	dimension reduction	classifier
3	not use	LSI	L1 Logistic Regression(C=100)

with accuracy score = 0.97667.

scores for all possible combinations are as shown in the table below.

clf	reduce_dim	n_components	analyzer	min_df	penalty	mean_test_score
SVC(C=1000)	TruncatedSVD()	50	<function stem_rmv_punc >	3	NaN	0.974429
SVC(C=1000)	TruncatedSVD()	50	<function stem_rmv_punc >	5	NaN	0.973795
SVC(C=1000)	TruncatedSVD()	50	word	3	NaN	0.975486
SVC(C=1000)	TruncatedSVD()	50	word	5	NaN	0.973161
SVC(C=1000)	NMF()	50	<function stem_rmv_punc >	3	NaN	0.967878

SVC(C=1000)	NMF()	50	<function stem_rmv_punc >	5	NaN	0.96978
SVC(C=1000)	NMF()	50	word	3	NaN	0.966188
SVC(C=1000)	NMF()	50	word	5	NaN	0.968512
GaussianNB()	TruncatedSVD()	50	<function stem_rmv_punc >	3	NaN	0.914201
GaussianNB()	TruncatedSVD()	50	<function stem_rmv_punc >	5	NaN	0.903001
GaussianNB()	TruncatedSVD()	50	word	3	NaN	0.911665
GaussianNB()	TruncatedSVD()	50	word	5	NaN	0.922866
GaussianNB()	NMF()	50	<function stem_rmv_punc >	3	NaN	0.946534
GaussianNB()	NMF()	50	<function stem_rmv_punc >	5	NaN	0.946112
GaussianNB()	NMF()	50	word	3	NaN	0.942096
GaussianNB()	NMF()	50	word	5	NaN	0.943364
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	3	l1	0.97612

LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	5	l1	0.974218
LogisticRegression(C=100)	TruncatedSVD()	50	word	3	l1	0.976754
LogisticRegression(C=100)	TruncatedSVD()	50	word	5	l1	0.975063
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	3	l1	0.970203
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	5	l1	0.970837
LogisticRegression(C=100)	NMF()	50	word	3	l1	0.967667
LogisticRegression(C=100)	NMF()	50	word	5	l1	0.967667
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	3	l2	0.975486
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	5	l2	0.975486
LogisticRegression(C=100)	TruncatedSVD()	50	word	3	l2	0.975697
LogisticRegression(C=100)	TruncatedSVD()	50	word	5	l2	0.975275
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	3	l2	0.965554
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	5	l2	0.964074

notes:

(1) <function stem_rmv_punc > represents using remmatization, and word represents not using.

(2) NAN represents "penalty" was not applicable for SVM

(2) Without headers and footers

The best choices are:

min_df	lemmatization	dimension reduction	classifier
3	not use	LSI	L2 Logistic Regression(C=100)

with accuracy score = 0.974218.

scores for all possible combinations are as shown in the table below.

clf	reduce_dim	n_components	analyzer	min_df	penalty	mean_test_score
SVC(C=1000)	TruncatedSVD()	50	<function stem_rmv_punc >	3	NaN	0.968301
SVC(C=1000)	TruncatedSVD()	50	<function stem_rmv_punc >	5	NaN	0.967667
SVC(C=1000)	TruncatedSVD()	50	word	3	NaN	0.971682
SVC(C=1000)	TruncatedSVD()	50	word	5	NaN	0.974007
SVC(C=1000)	NMF()	50	<function stem_rmv_punc >	3	NaN	0.962384
SVC(C=1000)	NMF()	50	<function stem_rmv_punc >	5	NaN	0.963229

SVC(C=1000)	NMF()	50	word	3	NaN	0.967244
SVC(C=1000)	NMF()	50	word	5	NaN	0.968301
GaussianNB()	TruncatedSVD()	50	<function stem_rmv_punc >	3	NaN	0.841082
GaussianNB()	TruncatedSVD()	50	<function stem_rmv_punc >	5	NaN	0.845309
GaussianNB()	TruncatedSVD()	50	word	3	NaN	0.850169
GaussianNB()	TruncatedSVD()	50	word	5	NaN	0.849746
GaussianNB()	NMF()	50	<function stem_rmv_punc >	3	NaN	0.939983
GaussianNB()	NMF()	50	<function stem_rmv_punc >	5	NaN	0.945266
GaussianNB()	NMF()	50	word	3	NaN	0.946323
GaussianNB()	NMF()	50	word	5	NaN	0.945689
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	3	l1	0.969146
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	5	l1	0.970837
LogisticRegression(C=100)	TruncatedSVD()	50	word	3	l1	0.974007

LogisticRegression(C=100)	TruncatedSVD()	50	word	5	l1	0.973584
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	3	l1	0.963229
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	5	l1	0.964074
LogisticRegression(C=100)	NMF()	50	word	3	l1	0.968512
LogisticRegression(C=100)	NMF()	50	word	5	l1	0.969992
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	3	l2	0.968935
LogisticRegression(C=100)	TruncatedSVD()	50	<function stem_rmv_punc >	5	l2	0.970414
LogisticRegression(C=100)	TruncatedSVD()	50	word	3	l2	0.974641
LogisticRegression(C=100)	TruncatedSVD()	50	word	5	l2	0.974218
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	3	l2	0.958580
LogisticRegression(C=100)	NMF()	50	<function stem_rmv_punc >	5	l2	0.959003
LogisticRegression(C=100)	NMF()	50	word	3	l2	0.958791
LogisticRegression(C=100)	NMF()	50	word	5	l2	0.960270

notes:

(1) <function stem_rmv_punc > represents using remmatization, and word represents not using.
(2) NAN represents "penalty" was not applicable for SVM

(3) Analysis

It turns out the best combination for documents with headers and footers is different from those without headers and footers. And the best accuracy after removing headers and footers (0.974218) is not as good as before(0.97667). This may be because the headers and footers in emails contain some useful information for identifying classes.

And min_df = 3 is better than min_df = 5, this might be because that some words in each class, even though with relatively smaller occurrences(under 5), can still help the classification.

For lemmatization, the result seems to be unexpected because intuitively lemmatization should help. This might be because lemmatization didn't play a very important role in this case. It can be seen that with lemmatization, the accuracy(0.97612) is almost the same as the best accuracy (0.97667).

LSI is usually better than NMF, because the matrix provided by SVD is usually a better approximation for the original matrix, as shown in Question 3. As a result, LSI can maintain more information of the original matrix and helps classification.

After removing headers and footers, L2 Logistic Regression performs better than L1 Logistic Regression and SVM. This might be because that there are many "margin" data in the train data set, i.e. the data points in train data. Since L2 logistic regression is more sensitive to the margins, it may perform better than the rest of classifiers.

Question 8: Multi-class Classifier

For this part, we classify the input documents into 4 classes:

(1) comp.sys.ibm.pc.hardware
(2) comp.sys.mac.hardware
(3) misc.forsale
(4) soc.religion.christian

using the following methods:

(1) Naive Bayes
(2) SVM: one vs one
(3) SVM: one vs rest

Results are as follows:

(1) Naive Bayes (using NMF to do dimensional reduction)

confusion_matrix =
[287 50 46 9]
[121 208 51 5]
[52 45 284 9]
[2 1 3 392]

Accuracy Score	Recall Score	Precision Score	F1 Score
0.748242811502	0.746383087181	0.747396073529	0.743521126236

(a) SVM: one vs one

NMF

confusion_matrix =
[303 64 25 0]
[76 281 27 1]
[47 22 320 1]
[9 2 8 379]

accuracy score	0.819808306709
recall score	0.818900860147
precision score	0.823731314661
F1 score	0.820584582024

LSI

confusion_matrix =
[326 39 26 1]
[44 314 27 0]
[22 18 348 2]
[6 1 1 390]

accuracy score	0.880511182109
recall score	0.879856064611
precision score	0.880304888224
F1 score	0.879947093932

It can be seen that using LSI to do dimensional reduction can have better performance than NMF. This is because the approximate matrix got from LSI is closer to the original matrix according to Question 3.

(b) SVM: one vs rest

NMF

confusion_matrix =
[282 65 43 2]
[68 278 35 4]
[41 19 327 3]
[0 1 9 388]

accuracy score	0.814696485623
recall score	0.813700396875
precision score	0.813563222286
F1 score	0.813288871546

LSI

confusion_matrix =
[314 53 25 0]
[42 316 24 3]
[20 20 348 2]
[4 1 1 392]

accuracy score	0.875399361022
recall score	0.874757986091
precision score	0.874587400147
F1 score	0.874591906634

Similarly, the performance for classifier using LSI is better than that using NMF.

To summarize, in this multi-class classifier case, SVM (one vs one) performs the best, and SVM (one vs rest) has almost the same performance. This might be because one vs rest can have data skew problem, which affects the performance of classifier. And both two SVM has better F1 score than naive bayes method.