EE219 Project 1 Report

Classification Analysis on Textual Data

Winter 2018

Introduction

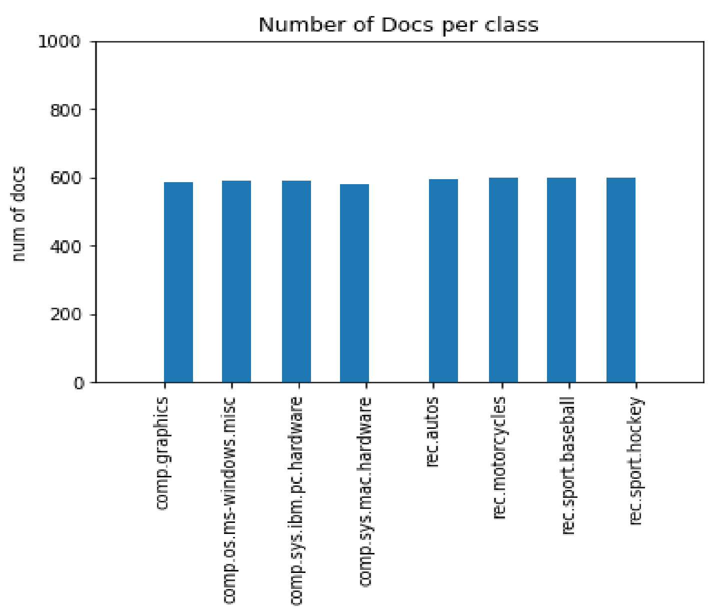
In this project, our group fetched 20 Newsgroup dataset, partitioned them evenly across different categories, extracted informative terms while getting rid of unnecessary ones, and implemented a variety of methods to classify textual data.

Dataset and Problem Statement

Problem a

The relative size of all data sets are evenly balanced across eight classes as shown in Figure 1 below.

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



Modeling Text Data and Feature Extraction

Problem b

Our group removed punctuations, non-ascii characters, and utilized Porter Stemmer to stem and tokenize the text.
When **min_df = 2**, the number of terms extracted is **21842** after punctuation removal, digit removal, and stemming with PorterStemmer.
When **min_df = 5**, the number of terms extracted is **8928** after punctuation removal, digit removal, and stemming with PorterStemmer.

Comments:
The reason we chose stemming instead of lemmatization in this case is that stemming is empirically considered as a fast method and effective way. “Lemmatizer is a tool from Natural Language Processing which does full morphological analysis to accurately identify the lemma for each word. Doing full morphological analysis produces at most very modest benefits for retrieval. It is hard to say more, because either form of normalization tends not to improve English information retrieval performance in aggregate - at least not by very much. While it helps a lot for some queries, it equally hurts performance a lot for others. Stemming increases recall while harming precision.” [1]

Problem c

After stemming and tokenizing the text, we performed TFICF, sorted and extracted most significant 10 terms in the following 4 classes.

10 most significant terms for **comp.sys.ibm.pc.hardware**:
['control', 'card', 'organ', 'subject', 'line', 'use', 'ide', 'thi', 'scsi', 'drive']

10 most significant terms for **comp.sys.mac.hardware**:
['problem', 'appl', 'simm', 'quadra', 'use', 'organ', 'subject', 'mac', 'line', 'thi']

10 most significant terms for **misc.forsale**:
['nntppostinghost', 'offer', 'use', 'new', 'thi', 'univers', 'organ', 'sale', 'subject', 'line']

10 most significant terms for **soc.religion.christian**:
['line', 'peopl', 'subject', 'church', 'hi', 'jesu', 'christian', 'wa', 'god', 'thi']

Feature Selection:

Problem d

We performed dimension reduction and derived dense matrices using *Latent Semantic Analysis(LSA)* and *Non-Negative Matrix Factorization (NMF).*
Please see Dimension Reduction Section in code for detail.

Learning Algorithms

Problem e

In this part, we applied **Support Vector Machine** Classifier to classify the documents between two categories ''*Computer Technology*' vs '*Recreational Activity*'. We analyzed hard margin and soft margin of it and derived quite discrepant results. After comparing results, we figured that LSI has a better performance than NMF. Furthermore, accuracy and precision we derived using hard margin method is almost doubled compared to the accuracy and precision calculated using soft margin. However, we observed that setting **min_df** to 2 or 5 doesn't make that much difference.

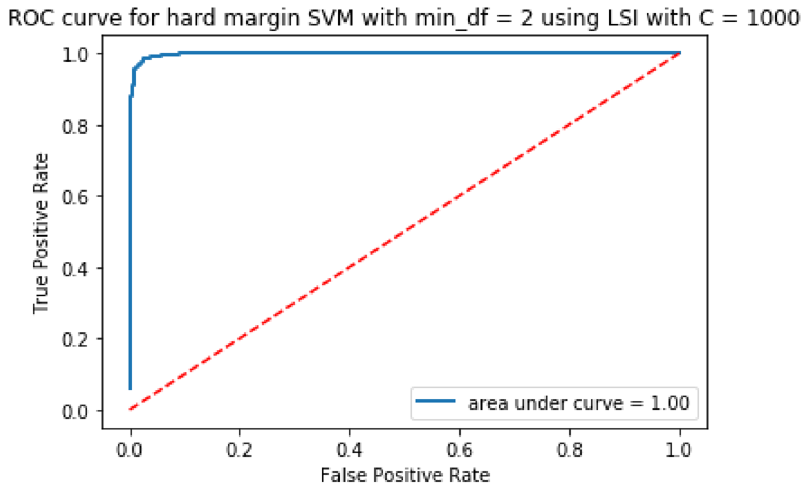
min_df = 2 and LSI:

Hard Margin SVM

Confusion Matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1517	43
Predicted Recreational Activity	26	1564

accuracy = 0.978095238095
precision = 0.973242065961
recall = 0.983647798742

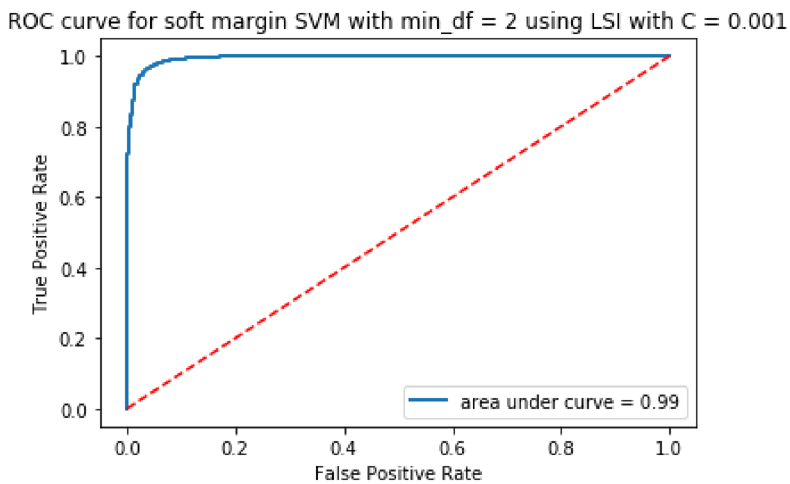


Soft Margin SVM

confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	0	1560
Predicted Recreational Activity	0	1590

accuracy = 0.504761904762
precision = 0.504761904762
recall = 1.0



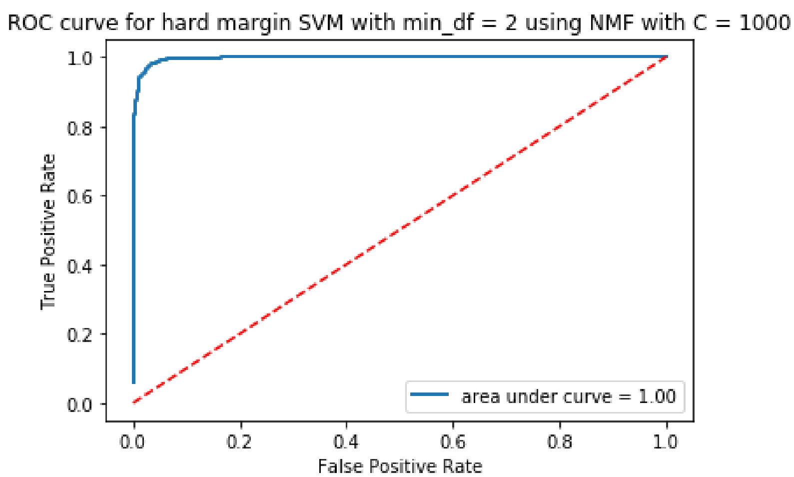
min_df = 2 and NMF

Hard Margin SVM

confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1496	64
Predicted Recreational Activity	28	1562

accuracy = 0.970793650794
precision = 0.960639606396
recall = 0.982389937107

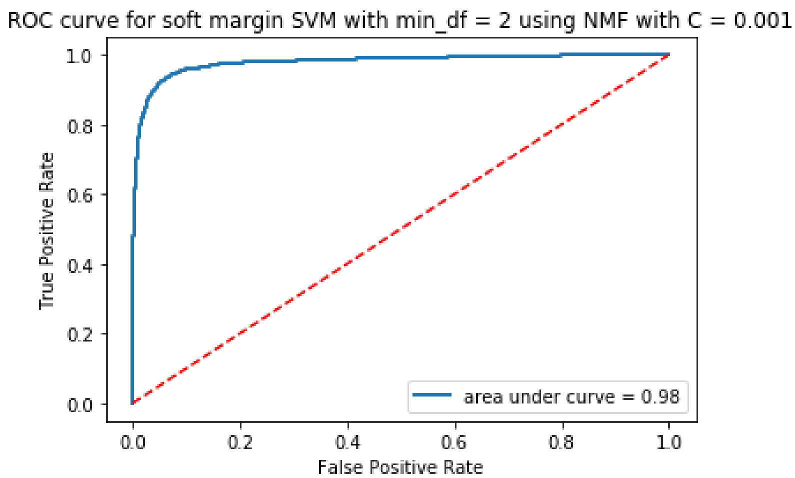


Soft Margin SVM

confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	0	1560
Predicted Recreational Activity	0	1590

accuracy = 0.504761904762
precision = 0.504761904762
recall = 1.0



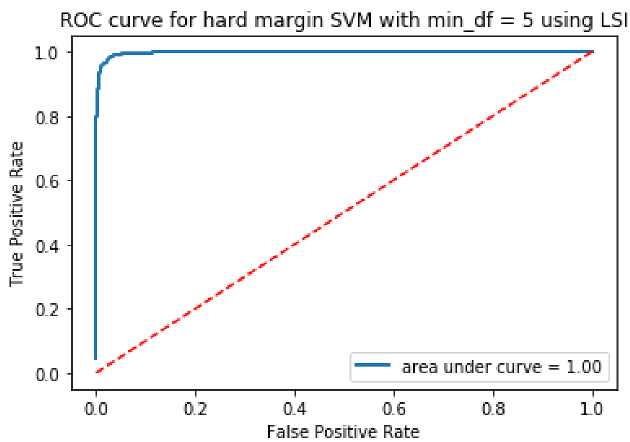
min_df = 5 and LSI

Hard Margin SVM

confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1517	43
Predicted Recreational Activity	23	1567

accuracy = 0.979047619048
precision = 0.973291925466
recall = 0.985534591195

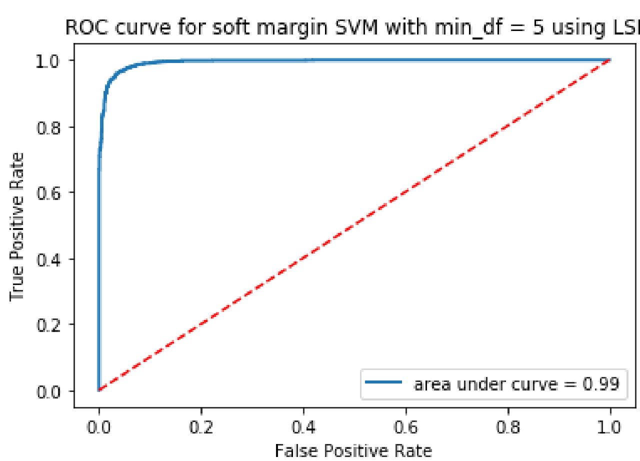


Soft Margin SVM

confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	0	1560
Predicted Recreational Activity	0	1590

accuracy = 0.504761904762
precision = 0.504761904762
recall = 1.0



Problem f

Next, we utilized **5-fold Validation** method for SVM by tuning the parameters with respect to the best accuracy result. See the attached code for details.

min_df = 2 and LSI

Parameter Tuning Results:
Tuning hyper-parameters for accuracy

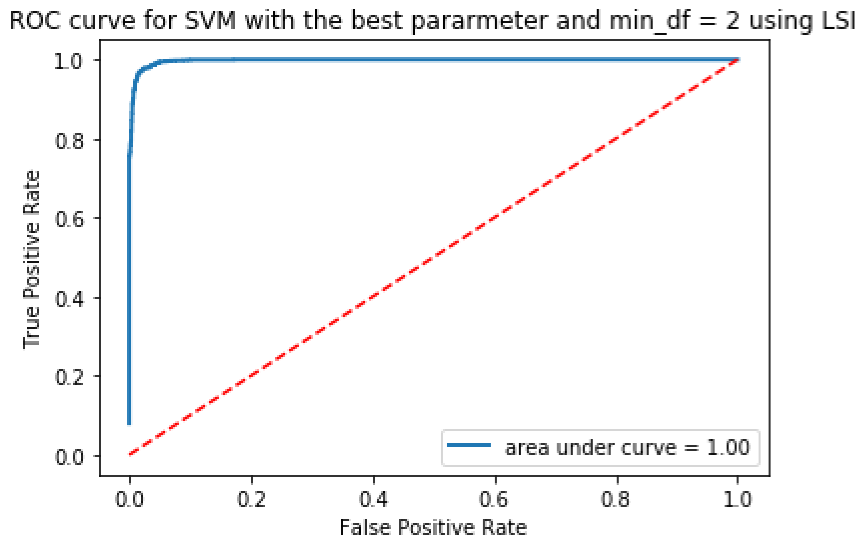
Grid scores on development set:

0.505 (+/-0.000) for {'C': 0.001, 'kernel': 'linear', 'probability': True}
0.507 (+/-0.001) for {'C': 0.01, 'kernel': 'linear', 'probability': True}
0.967 (+/-0.010) for {'C': 0.1, 'kernel': 'linear', 'probability': True}
0.974 (+/-0.005) for {'C': 1, 'kernel': 'linear', 'probability': True}
0.977 (+/-0.007) for {'C': 10, 'kernel': 'linear', 'probability': True}
0.976 (+/-0.011) for {'C': 100, 'kernel': 'linear', 'probability': True}
0.977 (+/-0.009) for {'C': 1000, 'kernel': 'linear', 'probability': True}
Best parameters: {'C': 10, 'kernel': 'linear', 'probability': True}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1503	57
Predicted Recreational Activity	24	1566

accuracy = 0.974285714286
precision = 0.964879852126
recall = 0.984905660377



min_df = 2 and NMF

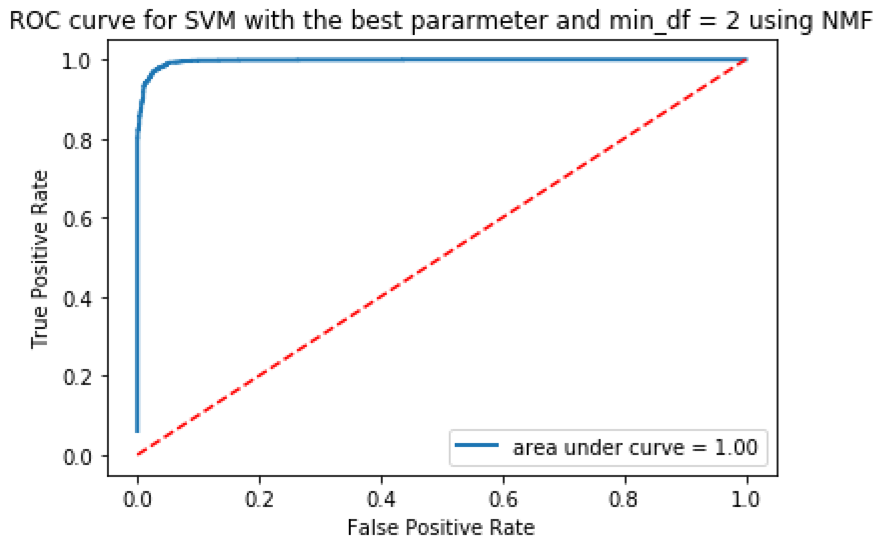
Tuning hyper-parameters for accuracy

Grid scores on development set:

0.505 (+/-0.000) for {'C': 0.001, 'kernel': 'linear', 'probability': True}
0.505 (+/-0.000) for {'C': 0.01, 'kernel': 'linear', 'probability': True}
0.505 (+/-0.001) for {'C': 0.1, 'kernel': 'linear', 'probability': True}
0.946 (+/-0.016) for {'C': 1, 'kernel': 'linear', 'probability': True}
0.963 (+/-0.013) for {'C': 10, 'kernel': 'linear', 'probability': True}
0.971 (+/-0.009) for {'C': 100, 'kernel': 'linear', 'probability': True}
0.974 (+/-0.007) for {'C': 1000, 'kernel': 'linear', 'probability': True}
Best parameters: {'C': 1000, 'kernel': 'linear', 'probability': True}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1496	64
Predicted Recreational Activity	28	1562



accuracy = 0.970793650794
precision = 0.960639606396
recall = 0.982389937107

min_df = 5 and LSI

Tuning hyper-parameters for accuracy

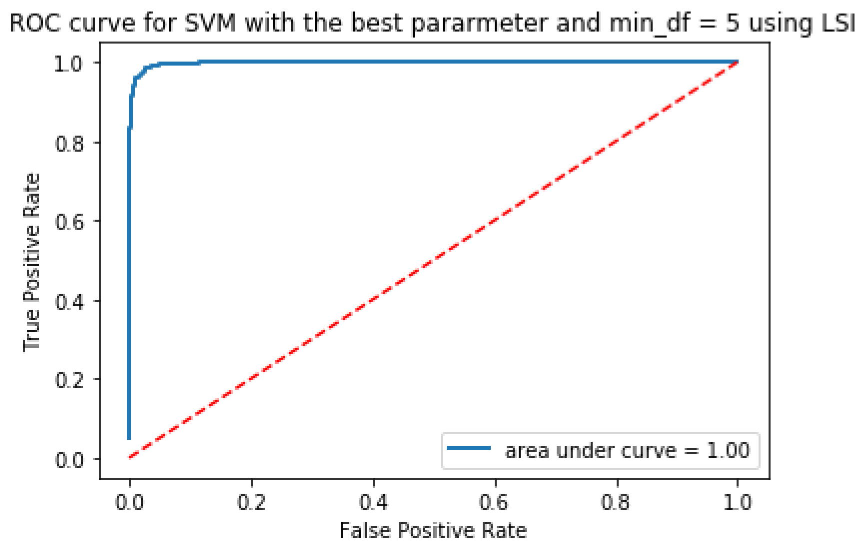
Grid scores on development set:

0.505 (+/-0.000) for {'C': 0.001, 'kernel': 'linear', 'probability': True}
0.516 (+/-0.002) for {'C': 0.01, 'kernel': 'linear', 'probability': True}
0.967 (+/-0.010) for {'C': 0.1, 'kernel': 'linear', 'probability': True}
0.973 (+/-0.004) for {'C': 1, 'kernel': 'linear', 'probability': True}
0.975 (+/-0.008) for {'C': 10, 'kernel': 'linear', 'probability': True}
0.975 (+/-0.007) for {'C': 100, 'kernel': 'linear', 'probability': True}
0.976 (+/-0.008) for {'C': 1000, 'kernel': 'linear', 'probability': True}
Best parameters: {'C': 1000, 'kernel': 'linear', 'probability': True}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1517	43
Predicted Recreational Activity	23	1567

accuracy = 0.979047619048
precision = 0.973291925466
recall = 0.985534591195



Problem g (Multinomial Naive Bayes)

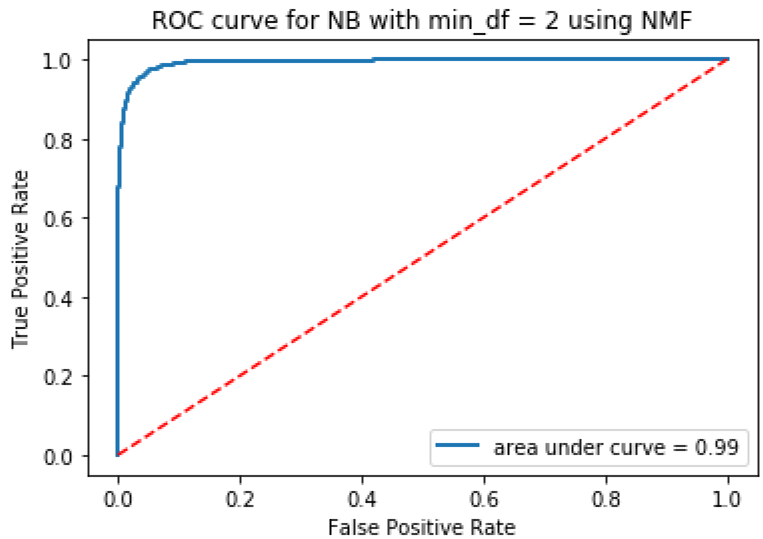
We implemented **Multinomial Naive Bayes** for the same classification task.

min_df = 2 and NMF

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1434	126
Predicted Recreational Activity	20	1570

accuracy = 0.953650793651
precision = 0.92570754717
recall = 0.987421383648



Problem h

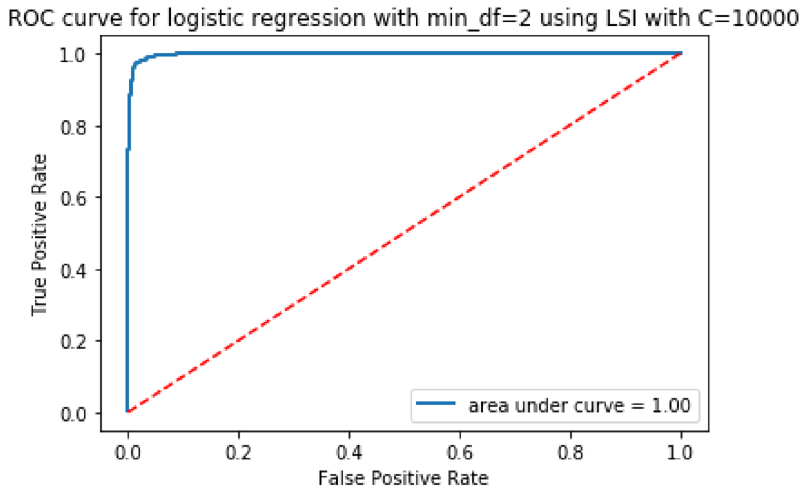
We used ***Logistic Regression without Regularization.*** In this section, penalty parameter C is set to 1000 because C is the inverse of regularization strength. A large C will lead to small regularization strength such that the effect of regularization can be ignored.

min_df = 2 and LSI

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1509	51
Predicted Recreational Activity	24	1566

accuracy = 0.97619047619
precision = 0.968460111317
recall = 0.984905660377

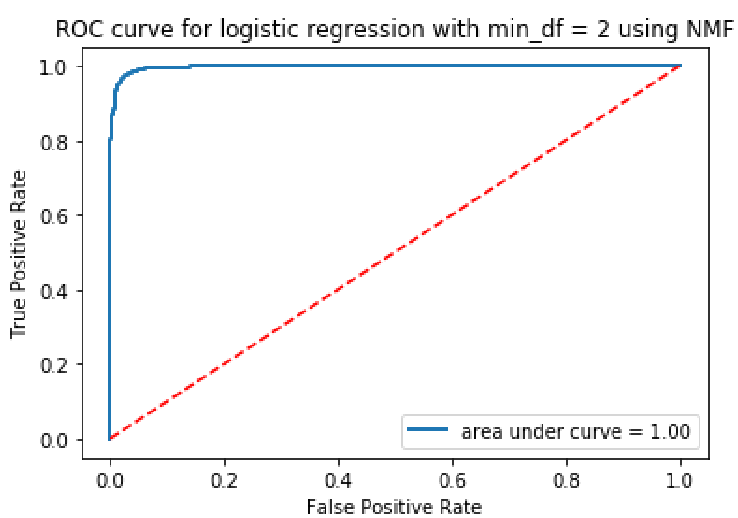


min_df = 2 and NMF

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1502	58
Predicted Recreational Activity	27	1563

accuracy = 0.973015873016
precision = 0.96421961752
recall = 0.983018867925

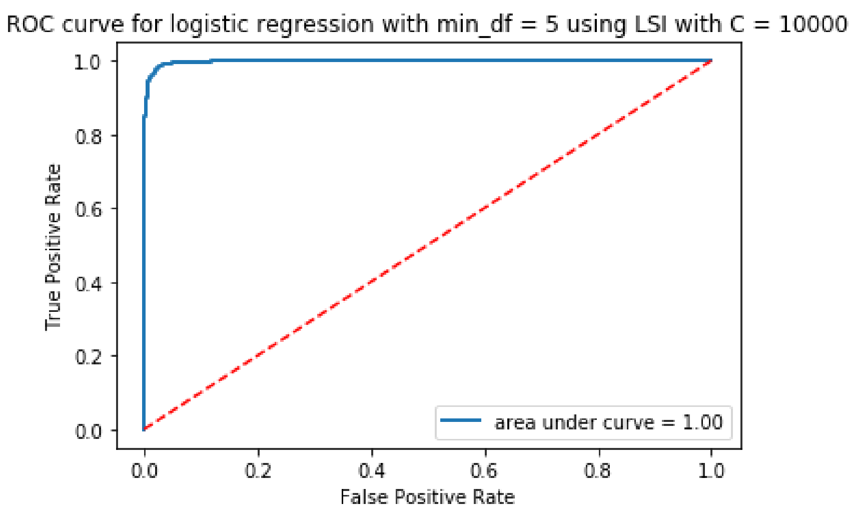


min_df = 5 and LSI

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1514	46
Predicted Recreational Activity	23	1567

accuracy = 0.978095238095
precision = 0.971481711097
recall = 0.985534591195



Problem i

In this problem, we used ***Logistic Regression with Regularization***. Firstly we tuned parameter with l1 and l2 norm regularization separately and figured the best penalty parameter C which is the inverse of regularization strength, and from that, we had the corresponding confusion matrices, accuracy, precision, recall, etc.

Regularization = ‘l1’

min_df = 2 and LSI

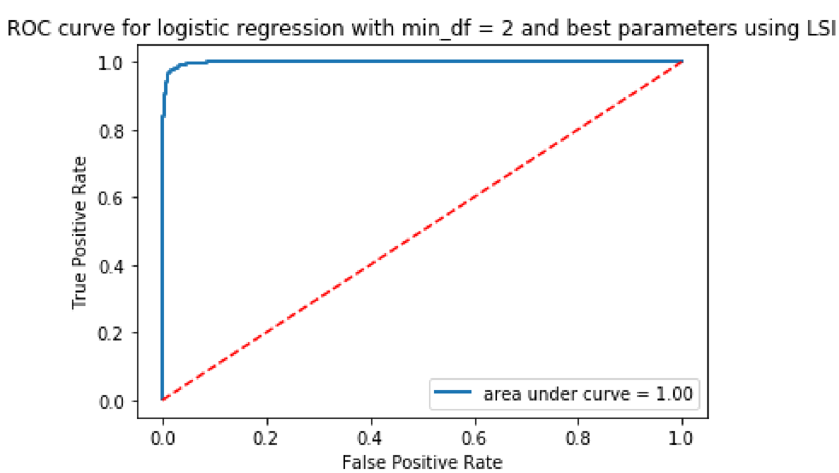
Tuning hyper-parameters for accuracy

Grid scores on development set:
0.495 (+/-0.000) for {'C': 0.001, 'penalty': 'l1'}
0.919 (+/-0.026) for {'C': 0.01, 'penalty': 'l1'}
0.947 (+/-0.007) for {'C': 0.1, 'penalty': 'l1'}
0.973 (+/-0.005) for {'C': 1, 'penalty': 'l1'}
0.977 (+/-0.008) for {'C': 10, 'penalty': 'l1'}
0.977 (+/-0.009) for {'C': 100, 'penalty': 'l1'}
0.977 (+/-0.008) for {'C': 1000, 'penalty': 'l1'}
Best parameters: {'C': 100, 'penalty': 'l1'}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1509	51
Predicted Recreational Activity	25	1565

accuracy = 0.975873015873
precision = 0.968440594059
recall = 0.98427672956



min_df = 2 and NMF

Tuning hyper-parameters for accuracy

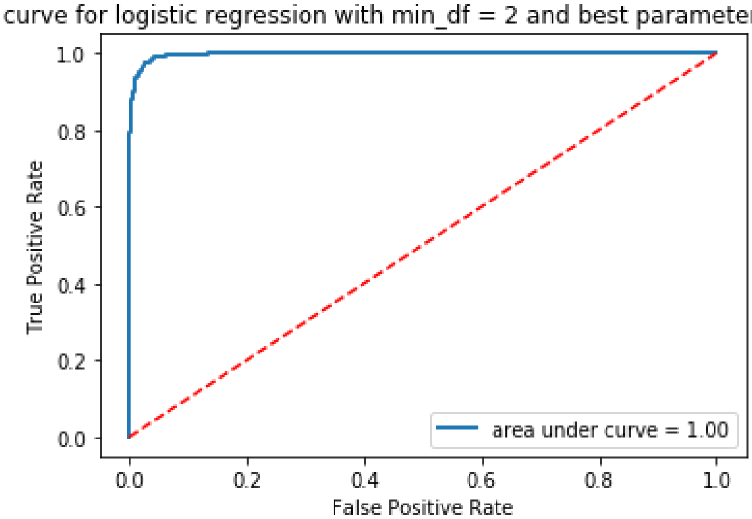
Grid scores on development set:
0.495 (+/-0.000) for {'C': 0.001, 'penalty': 'l1'}
0.495 (+/-0.000) for {'C': 0.01, 'penalty': 'l1'}
0.680 (+/-0.016) for {'C': 0.1, 'penalty': 'l1'}
0.959 (+/-0.014) for {'C': 1, 'penalty': 'l1'}
0.974 (+/-0.007) for {'C': 10, 'penalty': 'l1'}
0.973 (+/-0.009) for {'C': 100, 'penalty': 'l1'}
0.972 (+/-0.008) for {'C': 1000, 'penalty': 'l1'}
Best parameters: {'C': 10, 'penalty': 'l1'}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1497	63
Predicted Recreational Activity	25	1565

accuracy = 0.972063492063
precision = 0.961302211302
recall = 0.98427672956

ROC curve for logistic regression with min_df = 2 and best parameters using NMF



min_df = 5 and LSI

Tuning hyper-parameters for accuracy

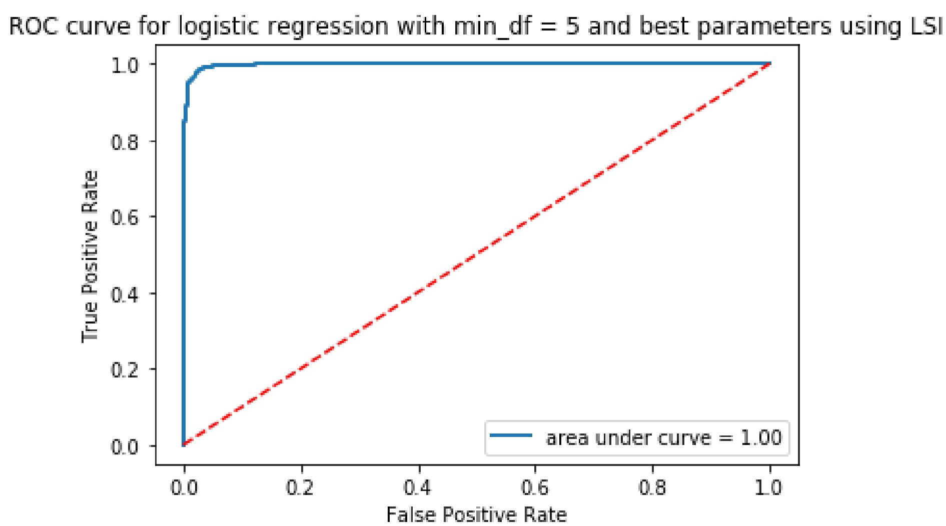
Grid scores on development set:

0.495 (+/-0.000) for {'C': 0.001, 'penalty': 'l1'}
0.929 (+/-0.021) for {'C': 0.01, 'penalty': 'l1'}
0.947 (+/-0.006) for {'C': 0.1, 'penalty': 'l1'}
0.972 (+/-0.004) for {'C': 1, 'penalty': 'l1'}
0.976 (+/-0.008) for {'C': 10, 'penalty': 'l1'}
0.976 (+/-0.009) for {'C': 100, 'penalty': 'l1'}
0.976 (+/-0.010) for {'C': 1000, 'penalty': 'l1'}
Best parameters: {'C': 10, 'penalty': 'l1'}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1516	44
Predicted Recreational Activity	24	1566

accuracy = 0.978412698413
precision = 0.972670807453
recall = 0.984905660377



Regularization = ‘l2’

min_df = 2 and LSI

Tuning hyper-parameters for accuracy

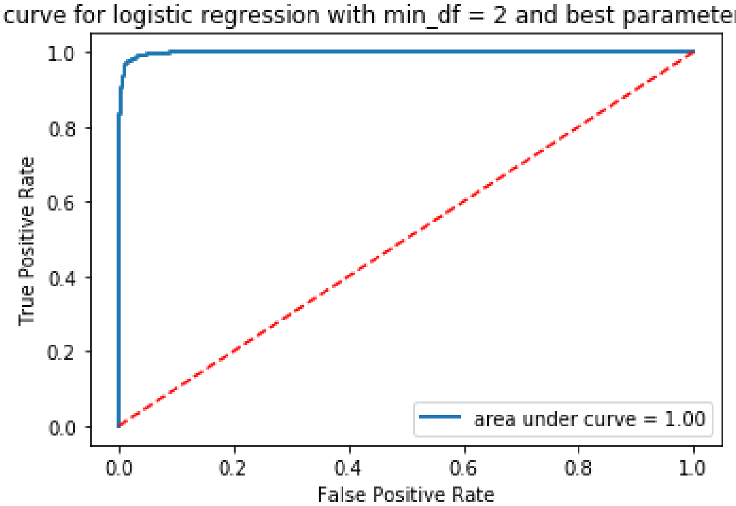
Grid scores on development set:
0.726 (+/-0.023) for {'C': 0.001, 'penalty': 'l2'}
0.951 (+/-0.007) for {'C': 0.01, 'penalty': 'l2'}
0.965 (+/-0.008) for {'C': 0.1, 'penalty': 'l2'}
0.970 (+/-0.008) for {'C': 1, 'penalty': 'l2'}
0.976 (+/-0.004) for {'C': 10, 'penalty': 'l2'}
0.976 (+/-0.008) for {'C': 100, 'penalty': 'l2'}
0.977 (+/-0.010) for {'C': 1000, 'penalty': 'l2'}
Best parameters: {'C': 1000, 'penalty': 'l2'}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1509	51
Predicted Recreational Activity	23	1567

accuracy = 0.976507936508
precision = 0.96847960445
recall = 0.985534591195

ROC curve for logistic regression with min_df = 2 and best parameters using LSI



min_df = 2 and NMF

Tuning hyper-parameters for accuracy

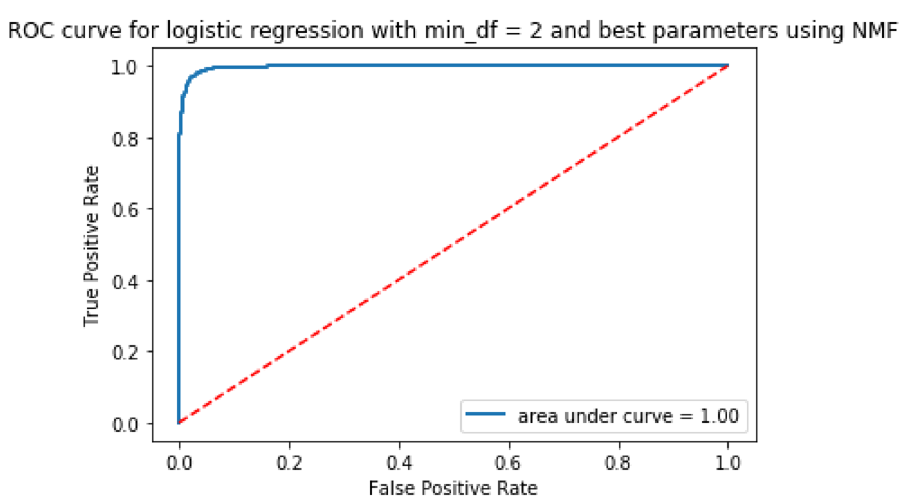
Grid scores on development set:

0.505 (+/-0.000) for {'C': 0.001, 'penalty': 'l2'}
0.520 (+/-0.005) for {'C': 0.01, 'penalty': 'l2'}
0.922 (+/-0.010) for {'C': 0.1, 'penalty': 'l2'}
0.944 (+/-0.019) for {'C': 1, 'penalty': 'l2'}
0.958 (+/-0.011) for {'C': 10, 'penalty': 'l2'}
0.970 (+/-0.010) for {'C': 100, 'penalty': 'l2'}
0.974 (+/-0.007) for {'C': 1000, 'penalty': 'l2'}
Best parameters: {'C': 1000, 'penalty': 'l2'}

Confusion matrix

	Computer Technology	Recreational Activity
Predicted Computer Technology	1509	51
Predicted Recreational Activity	23	1567

accuracy = 0.972380952381
precision = 0.961325966851
recall = 0.984905660377



min_df = 5 and LSI

Confusion matrix

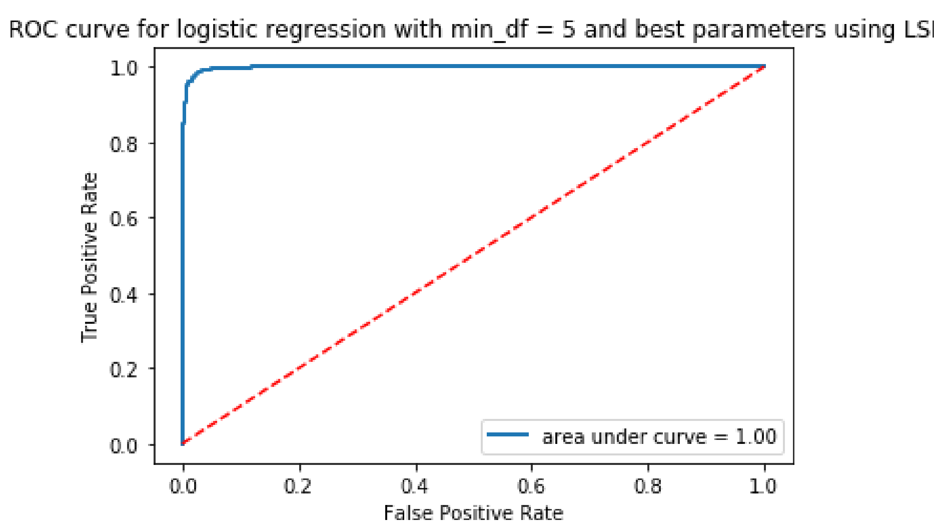
	Computer Technology	Recreational Activity
Predicted Computer Technology	1515	45
Predicted Recreational Activity	22	1568

accuracy = 0.97873015873
precision = 0.9721016739
recall = 0.986163522013

Tuning hyper-parameters for accuracy

Grid scores on development set:

0.782 (+/-0.016) for {'C': 0.001, 'penalty': 'l2'}
0.956 (+/-0.009) for {'C': 0.01, 'penalty': 'l2'}
0.964 (+/-0.007) for {'C': 0.1, 'penalty': 'l2'}
0.970 (+/-0.008) for {'C': 1, 'penalty': 'l2'}
0.974 (+/-0.004) for {'C': 10, 'penalty': 'l2'}
0.974 (+/-0.005) for {'C': 100, 'penalty': 'l2'}
0.975 (+/-0.009) for {'C': 1000, 'penalty': 'l2'}
Best parameters: {'C': 1000, 'penalty': 'l2'}



Comments:

As C increases(regularization strength decreases), the accuracy increases because the solution of regularized logistic regression cost function is moving towards unconstrained minimum. With high regularization strength, the coefficients of fitted hyperplane will not have peculiarly large value so that overfitting can be prevented.[2] However, with regularization strength too high, it will compromise important features that can be used to classify, making the accuracy really bad. L1 regularization usually produces very sparse coefficients while L2 regularization does not. L2 regularization is usually computationally efficient. [3]

Multiclass Classification

Problem j (Multiclass Classification)

In this section, we separated data into 4 classes: *comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian.* Next, we performed Multiclass Naive Bayes and Multiclass SVM classification.

Multiclass Naive Bayes

min_df = 2 and NMF

Confusion matrix

	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
Predicted comp.sys.ibm.pc.hardware	333	20	37	2
Predicted comp.sys.mac.hardware	82	254	47	2
Predicted misc.forsale	56	12	313	9
Predicted	3	0	5	390

soc.religion.christian				

accuracy = 0.82428115016
precision = 0.82428115016
recall = 0.82428115016

Multiclass SVM

One Vs One and LSI with C = 1000

Confusion matrix

	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
Predicted comp.sys.ibm.pc.hardware	330	36	25	1
Predicted comp.sys.mac.hardware	43	315	27	0
Predicted misc.forsale	27	12	349	2
Predicted soc.religion.christian	5	0	6	387

accuracy = 0.882428115016
precision = 0.882428115016
recall = 0.882428115016

One Vs One and NMF with C = 1000

	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian

Predicted comp.sys.ibm.pc.hardware	327	44	19	2
Predicted comp.sys.mac.hardware	62	303	18	2
Predicted misc.forsale	28	17	344	1
Predicted soc.religion.christian	5	2	3	388

accuracy = 0.870287539936
precision = 0.870287539936
recall = 0.870287539936

One Vs Rest and LSI with C = 1000

	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
Predicted comp.sys.ibm.pc.hardware	330	36	25	1
Predicted comp.sys.mac.hardware	43	315	27	0
Predicted misc.forsale	27	12	349	2
Predicted soc.religion.christian	5	0	6	387

accuracy = 0.882428115016
precision = 0.882428115016
recall = 0.882428115016

One Vs Rest and NMF with C = 1000

	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
Predicted comp.sys.ibm.pc.hardware	327	44	19	2
Predicted comp.sys.mac.hardware	62	303	18	2
Predicted misc.forsale	28	17	344	1
Predicted soc.religion.christian	5	2	3	388

accuracy = 0.870287539936
precision = 0.870287539936
recall = 0.870287539936

References

1. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
2. https://www.kdnuggets.com/2016/06/regularization-logistic-regression.html
3. http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/