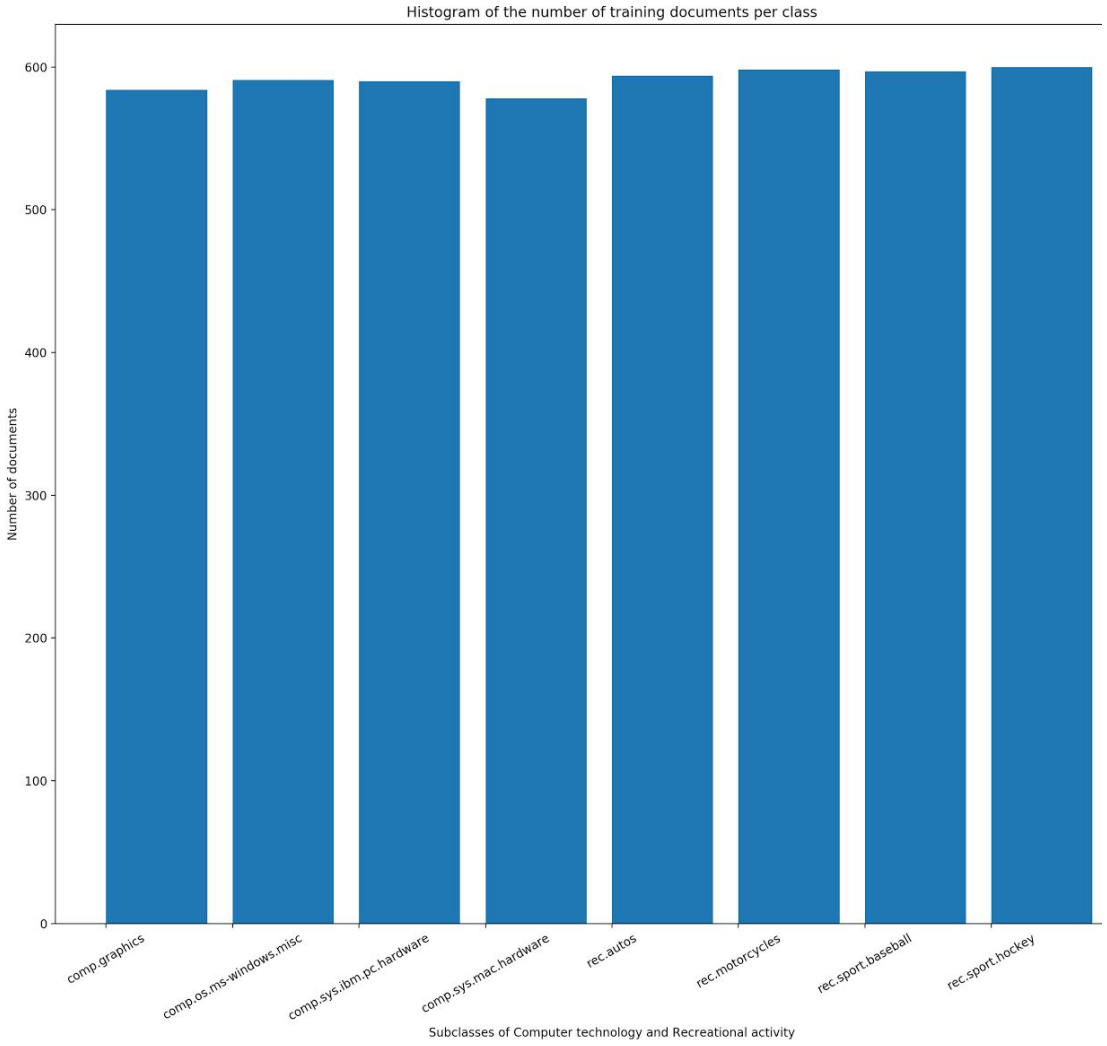
Classification Analysis on Textual Data Report

Abdullah-Al-Zubaer Imran
Curtis Crawford

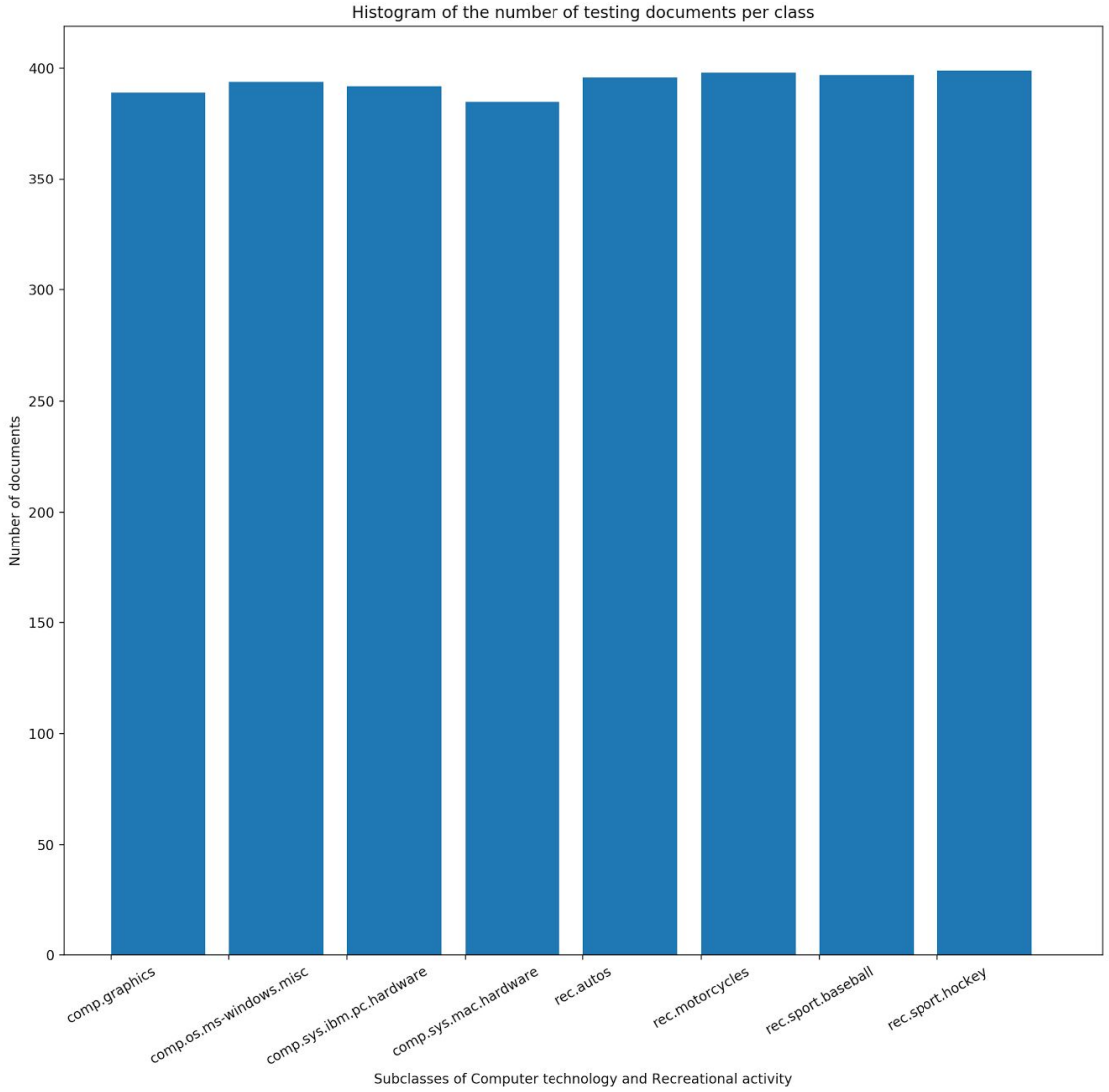
Parts A,B,D

These portions of the project focused on preparing the documents in the 20 Newsgroups dataset for training and testing classifiers. Using the built-in functions of the sklearn python module, these tasks were very easy to implement.

The total number of documents in the each of the 8 sub-classes from the Computer technology and Recreational activity classes contained about 1000 documents in total, split between 600 in a training set and 400 in a testing set. Sklearn provides functions for loading the dataset quickly. Below is a figure showing the even distribution of documents in the training set:



And also the even distribution of documents in the test set.



To vectorize the documents, the standard sklearn CountVectorizer class was extended to also stem words so that words such as “run” and “running” are counted as the same word. The Porter stemming algorithm was used. After vectorization and transformation to a TFxIDF matrix with another sklearn built in function, the dimensions of the feature vector as as follows for different values of the min_df parameter.

Reduction Methodology	TFxIDF dimensions (documents x terms)
LSI, min_df=2	4732 x 25335
LSI, min_df=5	4732 x 10691
NMF, min_df=2	4732 x 25335

Clearly, the higher min_df factor removed about ⅔ of the terms that using a min_df factor used. Using Latent Semantic Indexing (LSI) or NMF, the TFxIDF can be significantly reduced in dimensionality in order to improve classification tasks. LSI was done using the sklearn provided TruncatedSVD processing function, with a target of 50, such that the dimensions after LSI were:

4732 documents x 50 terms.

NMF was done using the same target of 50, using the sklearn NMF function. The dimensions after reduction were:

4732 documents x 50 terms.

So both LSI and NMF resulted in the same size of reduced TFxIDF matrices. The impact of the two methods will be further explored in the next section.

Part C

To determine the most important terms for the *comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale,* and *soc.religion.christian* sub classes, first all of the documents of a specific class were merged together, so there were 20 documents total where each one contained all the documents from that class. These 20 documents were then vectorized and converted to a TFxICF using the process as creating a TFxIDF. The 10 most important terms for each of these sub-classes is as follows, for a min_df of 2:

comp.sys.ibm.pc.hardware: 0th most common item is: scsi 1th most common item is: drive 2th most common item is: edu 3th most common item is: ide 4th most common item is: use 5th most common item is: line 6th most common item is: com 7th most common item is: subject 8th most common item is: organ 9th most common item is: card	comp.sys.mac.hardware: 0th most common item is: edu 1th most common item is: mac 2th most common item is: line 3th most common item is: subject 4th most common item is: organ **5th most common item is: quadra** 6th most common item is: use 7th most common item is: appl 8th most common item is: simm 9th most common item is: scsi
misc.forsale: 0th most common item is: edu 1th most common item is: 00 2th most common item is: line 3th most common item is: sale 4th most common item is: subject 5th most common item is: organ 6th most common item is: post 7th most common item is: new 8th most common item is: com 9th most common item is: univers	soc.religion.christian: 0th most common item is: god 1th most common item is: christian 2th most common item is: edu 3th most common item is: jesu 4th most common item is: church 5th most common item is: subject 6th most common item is: peopl 7th most common item is: line 8th most common item is: say 9th most common item is: christ

Notably the stemmer made some made some interesting choices, such as shortening “jesus” to “jesu.” However, this likely would not affect classification as any document run through the classifier would also be stemmed.

And for a min_df of 5:

comp.sys.ibm.pc.hardware: 0th most common item is: scsi 1th most common item is: drive 2th most common item is: edu 3th most common item is: ide 4th most common item is: use 5th most common item is: line 6th most common item is: com 7th most common item is: subject 8th most common item is: organ 9th most common item is: card	comp.sys.mac.hardware: 0th most common item is: edu 1th most common item is: mac 2th most common item is: line 3th most common item is: subject 4th most common item is: organ 5th most common item is: use 6th most common item is: appl 7th most common item is: simm 8th most common item is: scsi **9th most common item is: post**
misc.forsale: 0th most common item is: edu 1th most common item is: 00 2th most common item is: line 3th most common item is: sale 4th most common item is: subject 5th most common item is: organ 6th most common item is: post 7th most common item is: new 8th most common item is: com 9th most common item is: univers	soc.religion.christian: 0th most common item is: god 1th most common item is: christian 2th most common item is: edu 3th most common item is: jesu 4th most common item is: church 5th most common item is: subject 6th most common item is: peopl 7th most common item is: line 8th most common item is: say 9th most common item is: christ

Terms that have been bolded show the effect of different value of min_df. In this case, only one word from one class, comp.sys.mac.hardware, was removed due to raisin the min_df value from 2 to 5

Parts E, F, G, H, I

First, the documents were re-classified as either a Computer or a Recreational document, based on the high-level class of the sub-class they belonged to. This was done as these sections work with binary classifiers.

For parts E and F, the sklearn SVC classifier class was used to train the classifier and make predictions. Using a linear kernel and various values of γ to investigate its impact. The value of min_df between 2 and 5 had little impact on classifier performance.

In all cases, the “comp” class was treated as the “negative” class (target value of 0), and the “rec” class was treated as the “positive” class (target value of 1).

Hard-Margin vs Soft-Margin SVC

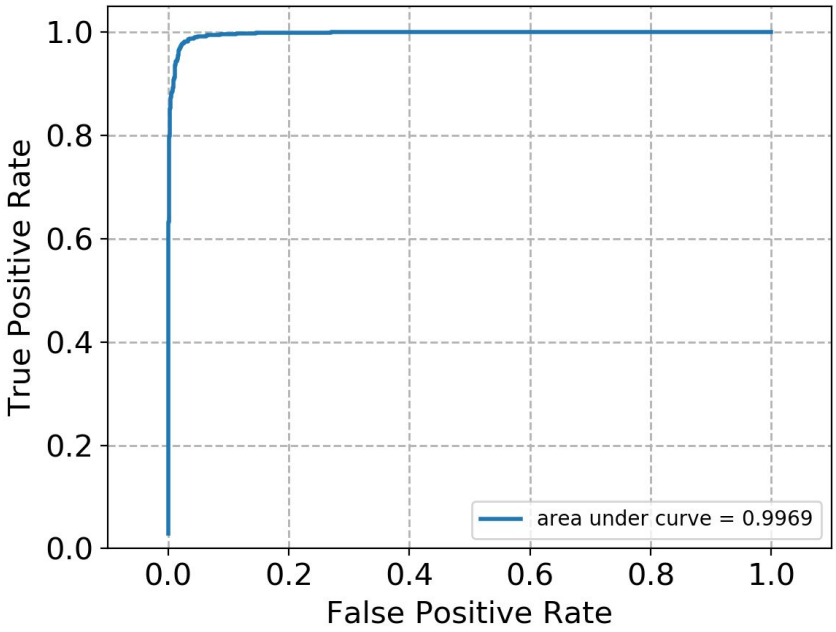
While for both min_df=2 and min_df=5 Hard-margin SVC with a linear kernel lead to approximately 97% accuracy, soft-margin only achieved a 50% accuracy score. Thus, with γ=0.001, the SVC was no better than a flip of a coin at determining if an article was computer focused or recreation focused. Below are the ROC curves and a table relating the accuracy, precision, recall, and confusion matrix for these classifiers.

Value of γ	min_df	Accuracy	Precision	Recall	Confusion Matrix
1000	2	0.975	0.968	0.982	1508 | 52 28 | 1562
1000	5	0.973	0.970	0.977	1512 | 48 37 | 1553
1000	2, NMF	0.963	0.961	0.967	1497 | 63 52 | 1538

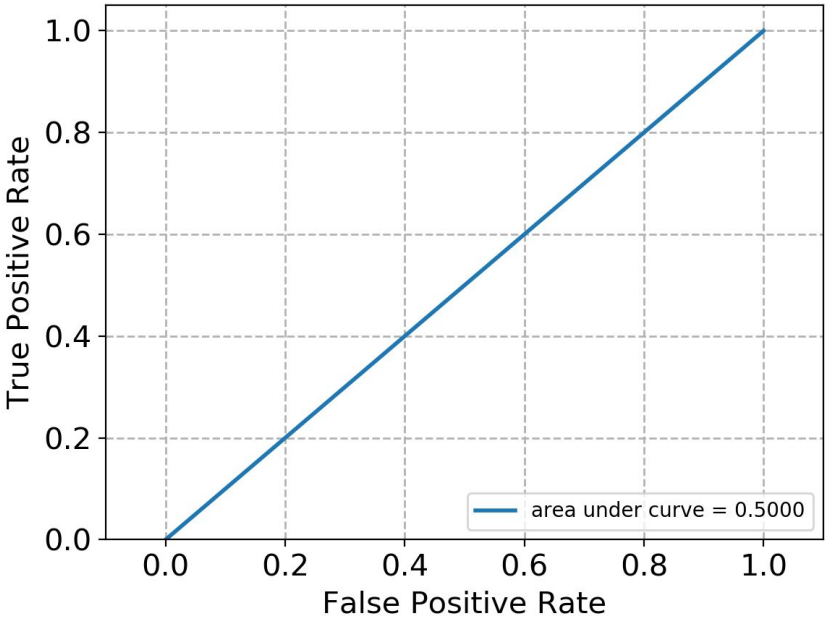
0.001	2	0.505	0.505	1.0	0 1560 0 1590
0.001	5	0.505	0.505	1.0	0 1560 0 1590
0.001	2, NMF	0.505	0.505	1.0	0 1560 0 1590

So, while the use of NMF slightly reduced performance in the hard-margin SVC, it only reduced accuracy by 1 percentage point. Similar changes were observed in recall and precision. However, the soft-margin SVM wasn't impacted at all. Further, using min_df of 2 or 5 had a negligible effect on accuracy, precision, and recall as well. Only one set of ROC curves are shown for each, as the differences between them for the three types of analysis are undetectable, even in the critical FP-rate from 0.0 to 0.2 section.

Heavy Margin roc curve:



Soft Margin roc curve:



SVC Cross Validation

Cross validation was done using the sklearn provided function that performs cross validation and predicts labels for the dataset, cross_val_predict(). The reported “best” value of γ is the first value of γ at which accuracy was the highest.

Min_df and NMF or LSI	Best γ value found
min_df=2, NMF	1000
min_df=2, LSI	100
min_df=5, LSI	100

Lowest value of γ for which accuracy was within 1% (0.01) of the “best” γ value:

Min_df and NMF or LSI	Lowest Near-best γ value found
min_df=2, NMF	10
min_df=2, LSI	0.1
min_df=5, LSI	1

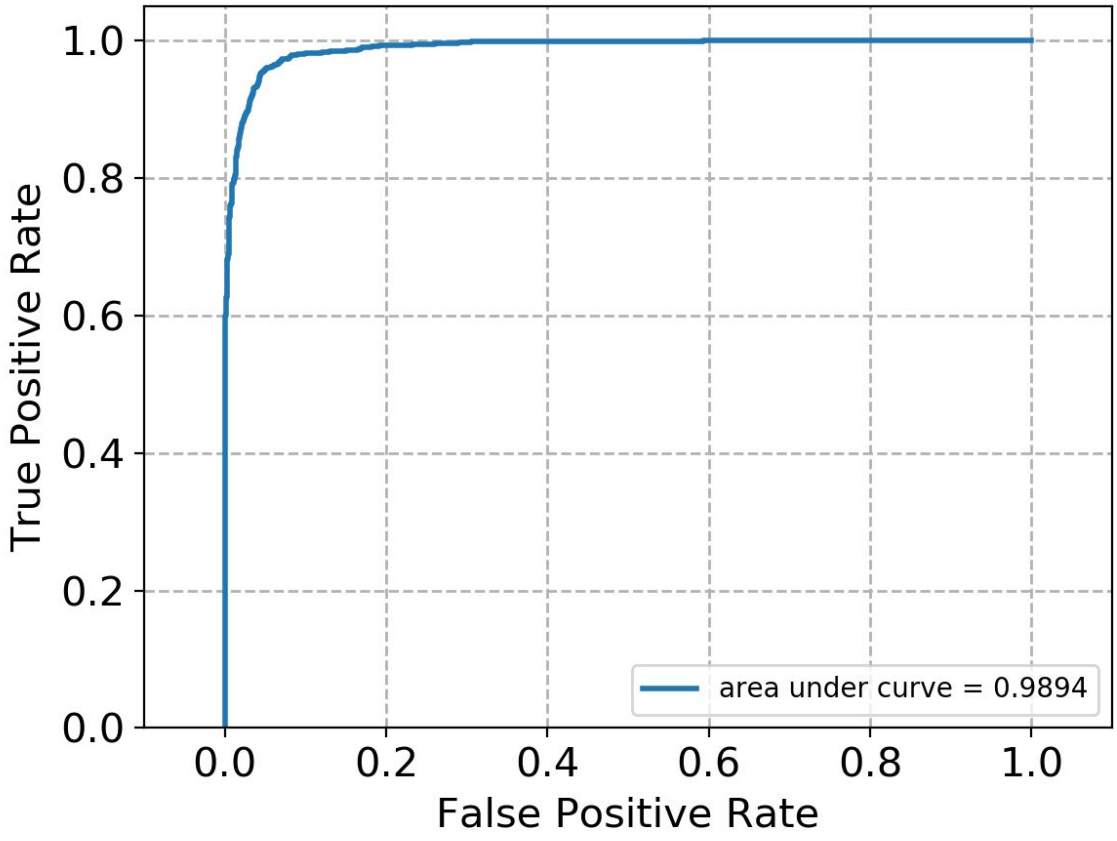
Naive Bayes Classifier

Using the Naive Bayes Classifier required that instead of performing dimension reduction on the TFxIDF matrix using LSI we use Non-Negative Matrix Factorization (NMF). This is because LSI can result in negative terms in the reduced matrix, which are not valid inputs for the Naive Bayes algorithm. Sklearn provides an implementation of NMF so we were able to slightly modify our pipeline to perform this classification, as well as a Multinomial Naive Bayes classifier implementation. Below are the accuracy, recall, and precision for the Naive Bayes classifier:

Accuracy: 0.930
Precision: 0.891
Recall: 0.983
Confusion Matrix:

```
[[1368  192]
 [  27 1563]]
```

ROC curve:



Logistic Regression without normalization

Below is a table detailing the effects of min_df and LSI vs NMF dimension reduction when using Logistic Regression to create a classifier where regularization is minimized. For sklearn, this means that the “C” parameter to the logistic regression implementation is a high value. C=1000 was used. As is detailed in the table below, the accuracy of the NMF reduction method is slightly lower than the LSI method, and the min_df parameter between 2 and 5 had negligible impact on accuracy.

Parameters	Accuracy	Confusion Matrix
Min_df = 2, NMF	0.963	`[[1491  69]` `[  48 1542]]`
Min_df = 2, LSI	0.974	`[[1508  52]` `[  31 1559]]`
Min_df = 5, LSI	0.973	`[[1509  51]` `[  34 1556]]`

Logistic Regression with “l2” normalization

Below is a table detailing the effects of min_df and LSI vs NMF dimension reduction when using Logistic Regression with “l2” regularization. A regularization value of 1.0 was used. As is detailed in the table below, the accuracy of the NMF reduction method is slightly lower than the LSI method, and the min_df parameter between 2 and 5 had negligible impact on accuracy.

Parameters	Accuracy	Confusion Matrix
Min_df = 2, NMF	0.945	`[[1443  117]` `[  56 1534]]`
Min_df = 2, LSI	0.968	`[[1495  65]` `[  36 1554]]`
Min_df = 5, LSI	0.968	`[[1491  69]` `[  32 1558]]`

Logistic Regression with “l1” normalization

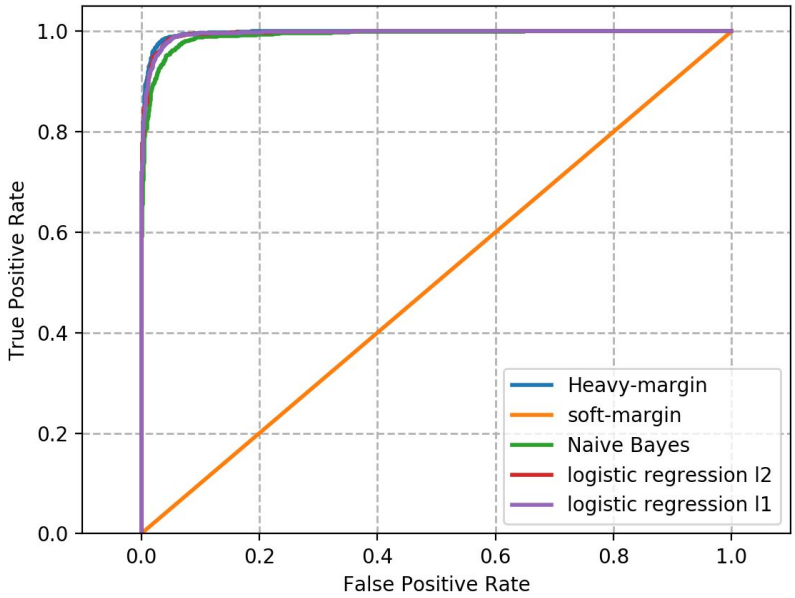
Below is a table detailing the effects of min_df and LSI vs NMF dimension reduction when using Logistic Regression with “l1” regularization. A regularization value of 1.0 was used. As is detailed in the table below, the accuracy of the NMF reduction method is slightly lower than the LSI method, and the min_df parameter between 2 and 5 had negligible impact on accuracy.

Parameters	Accuracy	Confusion Matrix
Min_df = 2, NMF	0.959	`[[1482  78]` `[  51 1539]]`
Min_df = 2, LSI	0.967	`[[1494  66]` `[  37 1553]]`
Min_df = 5, LSI	0.966	`[[1490  70]` `[  36 1554]]`

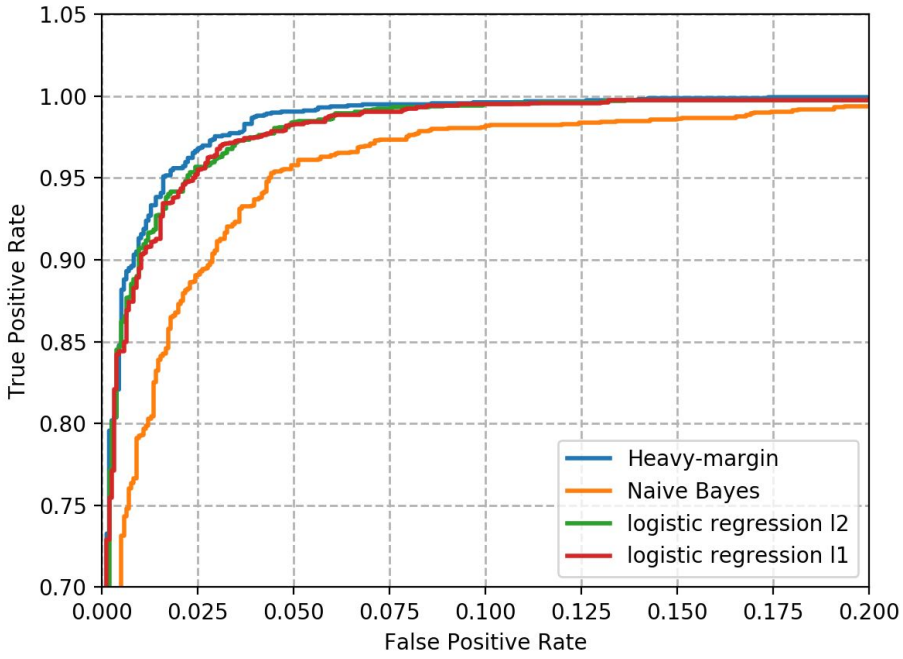
Between the three Logistic Regression approaches used, the impact of regularization had far less impact on accuracy than the method of dimension reduction did.

Results Comparison for LSI, min_df=2:

ROC Curves including the Soft-margin results. All are nearly as good, with Naive Bayes slightly under performing the others.



Close-up of the curve to show their slightly different plots, using NMF for Bayes only and min_df=2:



Multi-class classification

We performed multi-class classification on a dataset containing just documents from the four classes called out in the problem statement. Thus, the number of targets in our dataset was 4. Sklearn implements two different types of multiclass SVM, one that uses one-versus-one classification and one that uses one-versus-rest classification. Both can be done using the linear kernel. This means that all our classification tasks for these three different approaches, with 11 total classifiers if done by hand, can be implemented in only three sklearn module functions. The results are below, with the precision and recall given for each class individually

Naive Bayes

Accuracy: 0.803
Precision & Recall:

Class	Recall	Precision
comp.sys.ibm.pc.hardware	0.793	0.663
comp.sys.mac.hardware	0.566	0.845
misc.forsale	0.854	0.789
soc.religion.christian	0.990	0.947

Confusion Matrix:

```
[[311  35  39   7]
 [112 218  46   9]
 [ 46   5 333   6]
 [  0   0   4 394]]
```

One-vs-One Linear SVM

Using LSI

Accuracy: 0.875
Precision & Recall:

Class	Recall	Precision
comp.sys.ibm.pc.hardware	0.850	0.782
comp.sys.mac.hardware	0.795	0.845
misc.forsale	0.895	0.886

soc.religion.christian	0.957	0.995

Confusion Matrix:

```
[[333  41  18   0]
 [ 55 306  23   1]
 [ 26  14 349   1]
 [ 12   1   4 381]]
```

Using NMF

Accuracy: 0.734
Precision & Recall:

Class	Recall	Precision
comp.sys.ibm.pc.hardware	0.938	0.499
comp.sys.mac.hardware	0.426	0.906
misc.forsale	0.741	0.906
soc.religion.christian	0.834	1.0

Confusion Matrix:

```
[[368  11  13   0]
 [205 164  16   0]
 [ 95   6 289   0]
 [ 69   0   1 328]]
```

One-vs-Rest Linear SVM

Using LSI

Accuracy: 0.888
Precision & Recall:

Class	Recall	Precision
comp.sys.ibm.pc.hardware	0.821	0.850
comp.sys.mac.hardware	0.847	0.842
misc.forsale	0.903	0.867
soc.religion.christian	0.978	0.992

Confusion Matrix:

```
[[322  45  25   0]
 [ 32 326  26   1]
 [ 21  15 352   2]
 [  4   1   3 390]]
```

Using NMF

Accuracy: 0.820

Precision & Recall:

Class	Recall	Precision
comp.sys.ibm.pc.hardware	0.740	0.711
comp.sys.mac.hardware	0.709	0.756
misc.forsale	0.862	0.832
soc.religion.christian	0.967	0.982

Confusion Matrix:

```
[[290  70  31   1]
 [ 75 273  32   5]
 [ 39  14 336   1]
 [  4   4   5 385]]
```

Multi-class conclusion

In all cases, accuracy was reduced when using the NMF reduction methodology. However, it did allow the use of a wider variety of classifiers as Naive Bayes requires non-negative terms for training and classification. Also interesting is that regardless of the reduction method used, articles from the soc.religion.christian sub-class had better precision and recall scores than the other three sub-classes. One-vs-One Linear SVM accuracy was severely reduced when using NMF, while one-vs-rest SVM was not affected so dramatically.