
UCLA ECE219 - PROJECT 1

January 24th, 2019

AUTHORS

NING AN - 205034447

ZAURBEK TSOROJEV - 805029443

ENDI XU - 005030030

MINYA YAO - 704161217

Introduction

In this first project, we will classify data from the "20 Newsgroups" dataset. We will learn about TF-IDF representations of textual data, dimensionality reduction methods (PCA and NMF), different classification methods (SVM, Logistic Regression, Naïve Bayes, Multiclass), how to analyze results and get familiar with the complete pipeline of a textual data classification.

Question 1

Before starting classification, we need to make sure that our data is balanced. To verify that, we plot a histogram of the distribution of training documents for each category.

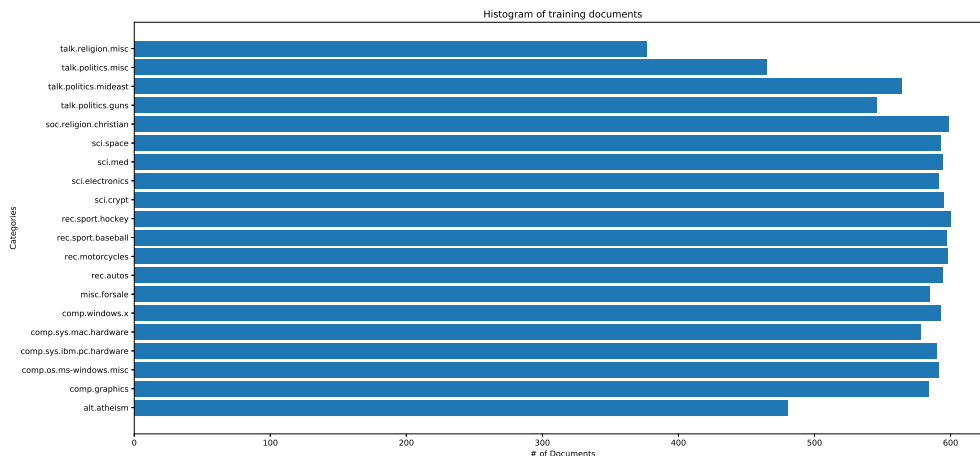


FIGURE 1 – Histogram of the training data

We see that the data is indeed balanced, as we have roughly 550 documents for most categories.

Question 2

We will now focus on documents from two separated classes : Computer Technology and Recreational Activities.

The document representation will be the "Bag of Words" representation. We split the text document into a list of words and calculate the frequency of each term in the document. This can be achieved with the CountVectorizer function in Python, which transforms a collection of text documents into a document-term matrix. We need to include some parameters in CountVectorizer to make sure to exclude stop words and we filter terms that are numbers using a regular expression. We also exclude rare words that appear less than 3 times through the min_df parameter. Finally, we perform lemmatization to group together inflected forms of a word back to its lemma.

Performing these operations on the given dataset results in TF-IDF matrices shape sizes of :

- Train subset : (4732, 16292)
- Test subset : (3150, 16292)

This means that we have 4,732 documents in the train subset, where we extracted 16,292 terms. And 3,150 documents in the test subset, where we extracted 16,292 terms.

Question 3

The dimensionality of the TF-IDF matrices we have is very large, which will lead to poor performances if we use it as it is. Our goal in this question is to find a lower dimension approximation of these matrices. We will try two dimensionality reduction methods : Latent Semantic Indexing (LSI) and Non-negative Matrix Factorization (NMF).

We get the following shapes of reduced TF-IDF matrices for both LSI and NMF :

- Train : (4732, 50)
- Test : (3150, 50)

Both of these reduction methods try to minimize the mean squared residual between the original data and the reconstruction from its low-dimensional approximation. The residuals are showed in the table bellow.

	<i>LSI</i>	<i>NMF</i>
<i>Train</i>	4107.97	4142.64
<i>Test</i>	2827.99	2838.32

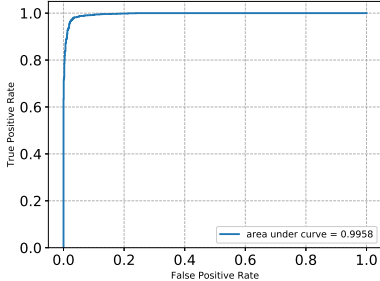
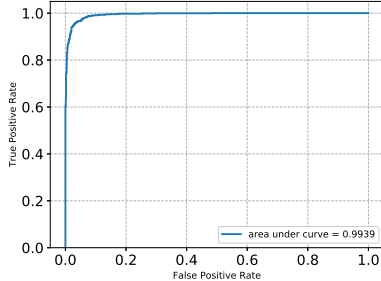
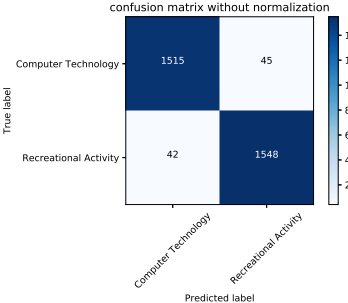
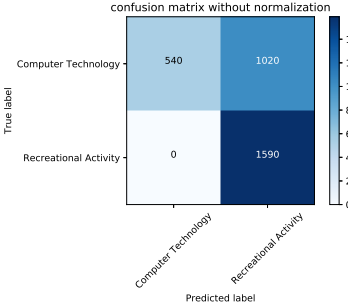
The NMF residual is the biggest, which means that LSI provides the better reduced estimate. The reason why LSI is slightly better is that we are selecting the top k principal components in the feature space for the representation, whereas NMF tries to find approximations to fit it the best.

Question 4

In this question, we will use the LSI dimension-reduced training data from Question 3 to train our SVM classifiers and evaluate their quality using several metrics such as precision, recall, F1-score and ROC curves.

Train two linear SVMs

Let's train two linear SVMs, one with hard margin and the other with soft margin, and analyze who is performing the best.

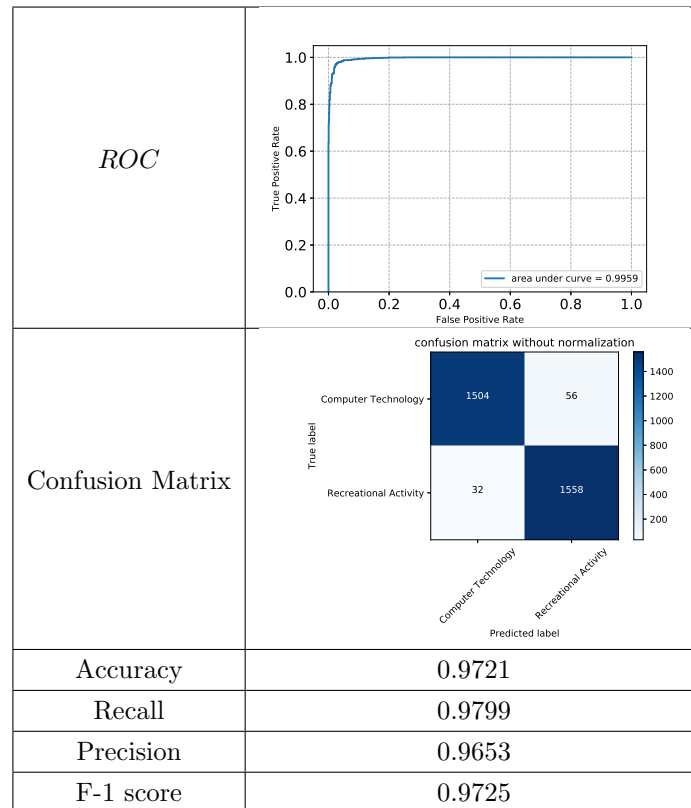
	$\gamma = 1000$ (Hard Margin)	$\gamma = 0.0001$ (Soft Margin)
<i>ROC</i>		
Confusion Matrix		
Accuracy	0.9723	0.6762
Recall	0.9735	1.000
Precision	0.9718	0.6092
F-1 score	0.9727	0.7572

The hard margin SVM clearly performs the best. We have better metrics overall and a better confusion matrix. The reason for this is that the soft margin SVM doesn't minimize the loss function, which means that misclassification is less penalized. Analyzing the confusion matrix, it seems it is mostly classifying everything into one class (Recreational Activity). Whereas the hard margin SVM tries to both minimize the loss function and maximize the margin between two classes. These are stricter constraints and it leads to a better classification.

Regarding the ROC curves, both look good as they are close to an ideal curve, where the area under the curve equals 1. We have a slightly better result for hard margin compared to soft margin : area under the curve of 0.9958 vs 0.9939, however this is not significant. This seems surprising, because the soft margin SVM has worse results for all the other metrics listed in the table. But when we analyze its confusion matrix, we see that it perfectly classifies one of the two classes. This means that it will have a good True Positive Rate, no matter the false positive rate.

Use cross-validation to choose γ

Using a 5-fold cross-validation, the best value of γ is 100 and it is obtained with a cross-validation score of 0.8673. As expected from the previous section, the best value $\gamma \gg 1$ (hard margin). However, the value is not too big, because an overconstrained system (over-fitting) will not deliver the best results.

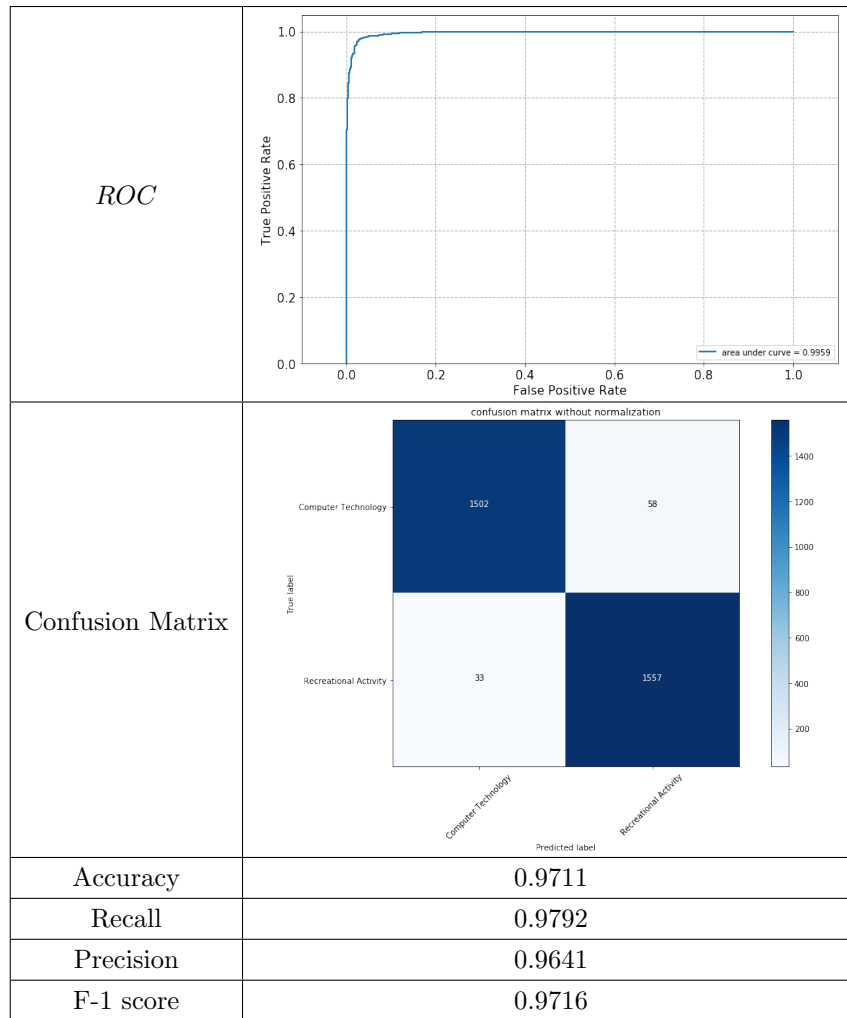


Question 5

In question 5, we use Logistic Regression to train our model and predict the class for each data.

Without Regularization

For a logistic regression classifier without regularization, we get the following results :



Note : in `LogisticRegression()`, to approximate the condition "without regularization", we use a very large regularization parameter (10^8) as this parameter is inversely proportional to the regularization strength.

With Regularization

Using 5-fold cross-validation, we find that the best regularization strength in the range $\{10^k \mid -3 \leq k \leq 3; k \in \mathbb{Z}\}$ is :

- L1 regularization : 10 ($k = 1$)
- L2 regularization : 10 ($k = 1$)

Let's now compare the performance of 3 logistic classifiers : without regularization, with L1 regularization

and with L2 regularization.

	w/o regularization	L1 regularization	L2 regularization
<i>Accuracy</i>	0.9711	0.9708	0.9721
<i>Precision</i>	0.9792	0.9629	0.9630
<i>Recall</i>	0.9641	0.9799	0.9824
<i>F1 – score</i>	0.9716	0.9713	0.973

Overall, it seems that L2 regularization provides the best performance on most metrics. It has the highest F1-score out of the three cases. On the other hand, the case without regularization provides the worst metrics, except for the precision where it is the highest, which makes sense because there we are only focused on maximizing the likelihood of the classification.

If the regularization strength is too small, it will lead to over-fitting and ill-conditioned results. As we increase its value, the model will learn better weight coefficients and the test error will decrease. But after some point, there will be too much weight on the regularization term and this will lead to under-fitting (test error will increase). That's why we found that the best regularization strength was 10 for both L1 and L2 and not the maximum value in our test range (1000).

The main difference between L1 and L2 regularization is the penalty term added to the loss function. In L1 regularization, we penalize the absolute value of the weights, whereas in L2, we penalize with the squared magnitude of the weights.

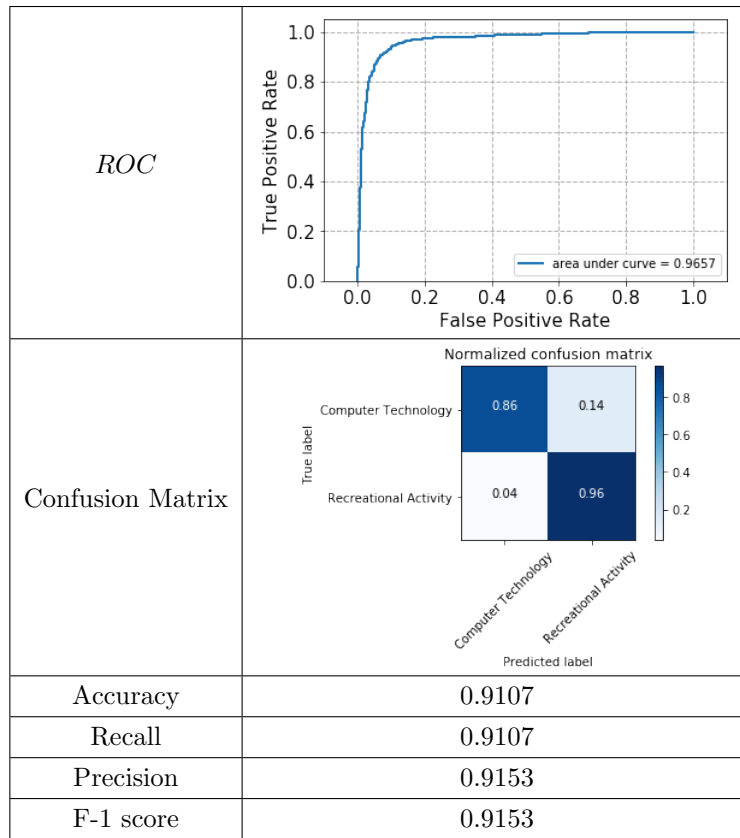
This results in the fact that L1 can yield sparse models while L2 doesn't. L1 will remove the less significant features as it will shrink their coefficients to 0. This works well for feature selection if we have a large number of features and it is good for compression.

Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary, so one may wonder what's the difference between their ways to find this boundary. SVM's goal is essentially to maximize the margin between the classes, while Logistic Regression maximizes the posterior class probability. SVM is deterministic while logistic regression is probabilistic. Logistic regression is good to use when we have a low number of dimensions and when we want is an estimation or we do not have high confidence into data. SVMs do better when we have a higher number of dimensions, and especially on problems where the predictors determine the responses in a confident way.

One can also note that logistic regression is more sensitive to outliers, because its cost function diverges faster.

Question 6

In question 6, we use Gaussian Naïve Bay to train our model and predict the class for each data. We get the following results :



The result shows that the Gaussian Naïve Bay has the worst performance in binary classification case among the three classifiers we tested in Question 4, Question 5 and Question 6. The performance of Gaussian Naïve Bay in multiple cases will be discussed in question 8 in details.

Question 7

In the previous sections, we have used different methods to deal with the text documents, reduce the dimensions and classify the test cases. However, it is hard to say which combination has the best performance. So, it is essential to do a grid search to find the best combination of these methods.

The following table shows the top 3 accuracy score with data with headers and footers by using grid search

	t score 1	t score 2	t score 3	t score 4	t score 5	mean	std
mindf=3,LSI,RMV,LR('L1')	0.81	0.76	0.75	0.79	0.79	0.78	0.02
mindf=3,LSI,RMV,SVM	0.81	0.714	0.75	0.79	0.74	0.76	0.035
mindf=3,LSI,RMV,LR('L2')	0.76	0.81	0.70	0.79	0.74	0.76	0.038765

The following table shows the top 3 accuracy score with data without headers and footers by using grid search

	t score 1	t score 2	t score 3	t score 4	t score 5	mean	std
mindf=3,LSI,RMV,LR('L1')	0.76	0.71	0.70	0.84	0.84	0.77	0.06
mindf=3,LSI,RMV,SVM	0.76	0.71	0.70	0.84	0.84	0.7	0.06
mindf=3,LSI,RMV,LR('L2')	0.76	0.71	0.70	0.79	0.84	0.76	0.05

from the two tables, we can see that if we keep the headers and footers, the result will be more accurate. This is expected because headers and footers sometimes contain important words which can be helpful in classification. However, remove headers and footers in the text will reduce the size of the data, which can improve the efficiency.

The top 3 combination is the same in both the two tables, which are (mindf=3,LSI,RMV,LR('L1')), (mindf=3,LSI,RMV,SVM) and (mindf=3,LSI,RMV,LR('L2')). This is also expected. Set the value of mindf smaller will filter less words and these words might be helpful in classification. Do the lemmatization will merge the word which contains the same meaning, which will reduce the risk that the classifier put the same meaning words into different categories. Also, as shown in the previous sections, SVM and logistic regression have a better performance than Gaussian Naive Bay with LSI.

Question 8

In this question, we used SVM and GaussianNB from sklearn to perform the multiclass SVM classification and the Naïve Bayes classification based on both LSI and NMF dimension-reduced TF-IDF matrices we found in Question 3.

For the multiclass SVM, we trained a One VS One and a One VS the rest classifiers using OneVsOneClassifier and OneVsRestClassifier from sklearn.multiclass. The confusion matrices obtained using LSI and NMF dimensionality reduction methods (unnormalized and normalized) are shown in Figure 2 to Figure 5, respectively.

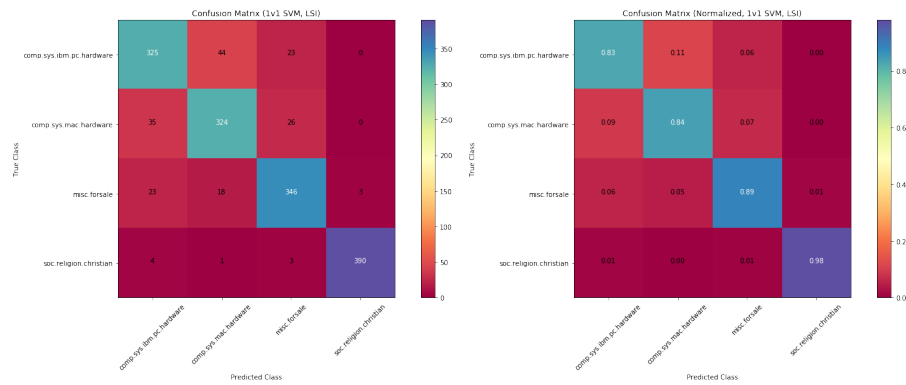


FIGURE 2 – Confusion matrix for One Vs One SVM classifier using LSI : unnormalized (left) and normalized (right)

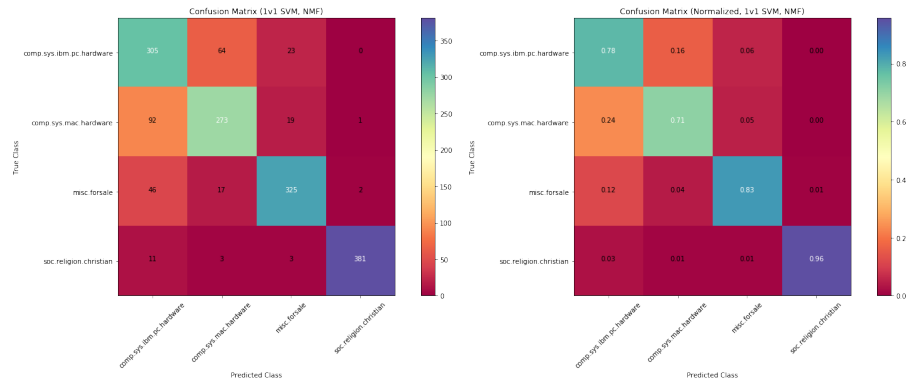


FIGURE 3 – Confusion matrix for One Vs One SVM classifier using NMF : unnormalized (left) and normalized (right)

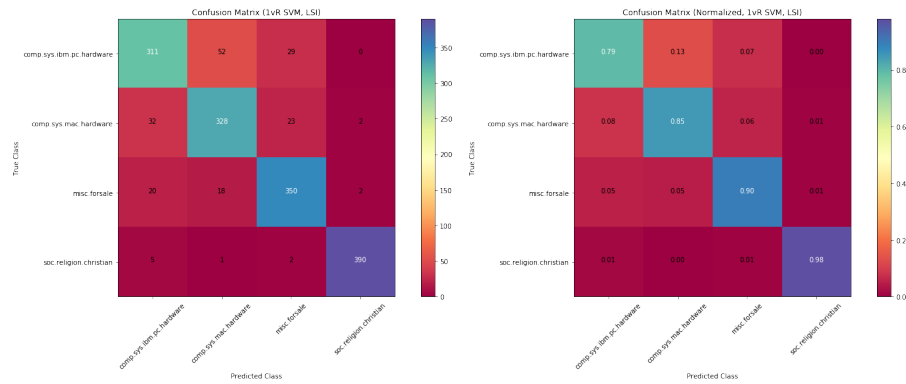


FIGURE 4 – Confusion matrix for One Vs Rest SVM classifier using LSI : unnormalized (left) and normalized (right)

The accuracy, recall, precision, and F-1 score for One Vs One and One Vs Rest multiclass SVM classifiers are summarized in table below.

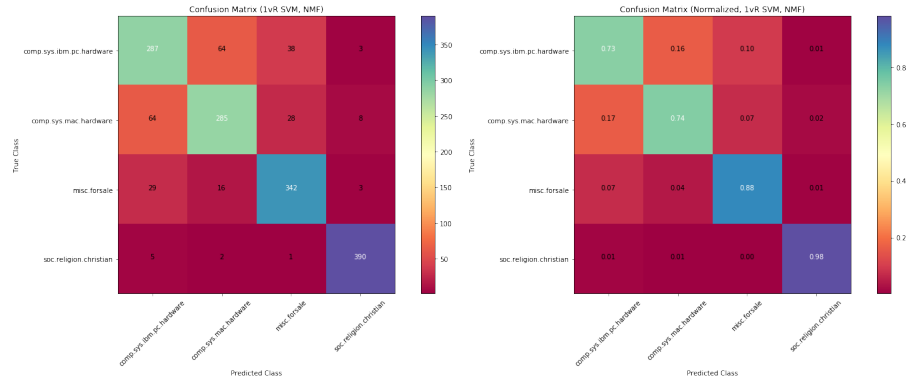


FIGURE 5 – Confusion matrix for One Vs Rest SVM classifier using NMF : unnormalized (left) and normalized (right)

	One Vs One Multiclass SVM		One Vs Rest Multiclass SVM	
	LSI	NMF	LSI	NMF
Accuracy	0.88498	0.82045	0.88115	0.83322
Recall	0.88443	0.81944	0.88066	0.83231
Precision	0.88468	0.82677	0.88084	0.83089
F-1 Score	0.88451	0.82164	0.88040	0.83134

For the Naïve Bayes classifier, we trained a Gaussian Naïve Bayes classifiers using `GaussianNB` from `sk-learn.naive_bayes`. The confusion matrices obtained using LSI and NMF dimensionality reduction methods (unnormalized and normalized) are shown in Figure 6 to Figure 7, respectively.

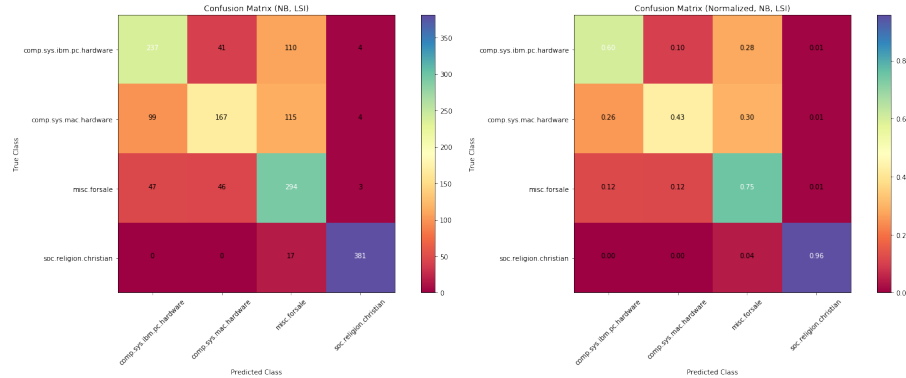


FIGURE 6 – Confusion matrix for Gaussian NB classifier using LSI : unnormalized (left) and normalized (right)

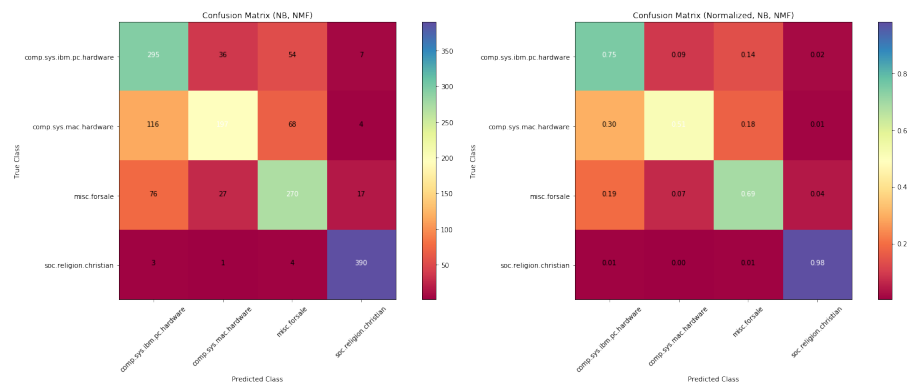


FIGURE 7 – Confusion matrix for Gaussian NB classifier using NMF : unnormalized (left) and normalized (right)

The accuracy, recall, precision, and F-1 score for the Naïve Bayes classifier are summarized in the table below.

	Naïve Bayes	
	LSI	NMF
Accuracy	0.68945	0.73610
Recall	0.68737	0.73411
Precision	0.69918	0.74292
F-1 Score	0.68346	0.73044