ECE-219 Project 1: Classification Analysis on Textual Data

Chenyang Wang, UID:105425757
Haochen Yin, UID:505332049

1. Introduction

In this first project of this course, we are going to do some classification analysis on some textual data. In particular, we will use different methods to classify the textual data. The goal for this project is:

1. to learn how to construct tf-idf representations of textual data
2. to get familiar with various common classification methods
3. to learn ways to evaluate and diagnose classification results.
4. to learn two dimensionality reduction methods: PCA & NMF
5. to get familiar with the complete pipeline of a textual data classification task. Detailed discussions on each term will be presented in the following section throughout the report.

2. The Dataset

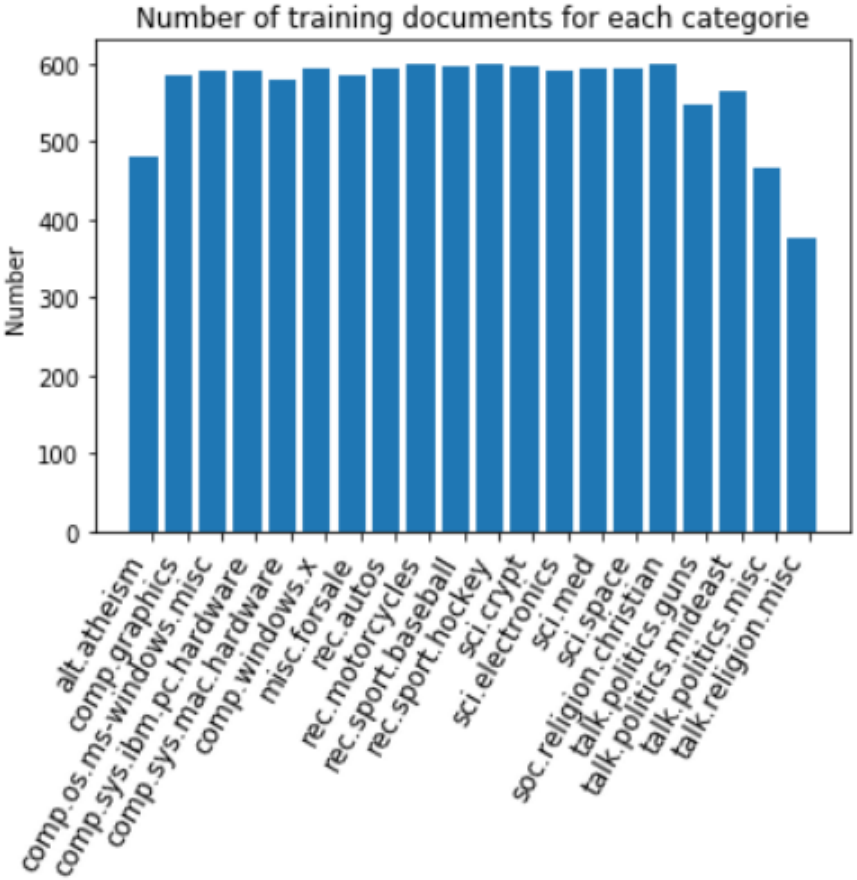
In this project, we will use the dataset called '20 Newsgroups' dataset, which is provided by ***scikit-learn***. In particular, we know that it is a collection of approximately 20,000 documents, divided evenly into 20 different newsgroups, and each corresponding to a different topic. To download this dataset is easy, simple import this dataset from ***sklearn*** in python, as shown below:

```
from sklearn.datasets import fetch_20newsgroups
```

One thing should be aware of is that we need to check whether the training dataset if balanced, meaning that we need to check whether the training dataset is evenly distributed. If it is not evenly distributed, we can then introduce some methods to handle this situation. For example, we can assign more weight to errors from the minority classes, or we can down-sample the majority classes. Thus, to begin with this project, we need to check whether our dataset is evenly distributed.

QUESTION 1: To get started, plot a histogram of the number of training documents for each of the 20 categories to check if they are evenly distributed.

Our result is shown below:



We can see that overall the dataset is already balanced, except for some classes. The categories we will be working on are perfectly balanced, thus there is no further action needed at this stage.

3. Feature Extraction

To classify something, we need to choose the features to do the learning process. In particular, in our case, since we are classifying textual data, the only features we can extract from the dataset are different words. We need to choose some good representations of a text. According to Prof. Roychowdhury, a good representation should retain enough information that enable us to perform the classification, yet in the meantime, be concise to avoid computational intractability and over-fitting. One common way to choose these good representations is to count how many times a certain word appears in a given text, i.e. the frequency of a given word. However, according to common sense, there are words that are meaningless, for example 'and', 'an', and 'the', etc. can be dropped when counting. And also, we can say that words sharing the same stem in the vocabulary, for example 'go' and 'going' can be considered as same word. To actually compute the frequencies of words in each document, a popular method that can be used is called 'Term Frequency-Inverse Document Frequency' (TF-IDF). Using Prof. Roychowdhury to describe the idea of this method, this measurement takes into account count of the words in the document, as normalized by a certain function of the frequency of the individual words in the whole corpus. The TF-IDF score is defined as below:

$$tf-idf(d,t)=tf(t,d)\times idf(t)$$

Where *tf(t,d)* represents the frequency of term *t* in document *d*, and the inverse document frequency is defined as below:

$$idf(t)=\log\left(\frac{n}{df(t)}\right)+1$$

Where *n* is the total number of documents and *df(t)* is the document frequency. With this metric, we can then construct a term-number matrix whose entries are just this metric.

QUESTION 2: Use the following specs to extract features from the textual data:

- **Use the "english" stopwords of the CountVectorizer**
- **Exclude terms that are numbers (e.g. "123", "-45", "6.7" etc.)**
- **Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos tag**
- **Use min df=3**

Report the shape of the TF-IDF matrices of the train and test subsets respectively. Our result is shown below:

```
train_tf_idf: (4732, 14186)
test_tf_idf: (3150, 14186)
array([6, 7, 4, ..., 6, 6, 2])
```

The first entry is the number of rows of the matrix, and the second entry is the number of columns of the matrix. To interpret this matrix, we say that for example, there are 4732 documents in the training dataset, and we extracted 14186 words (features) from the training dataset.

4. Dimensionality Reduction

In the previous section, we see that there are too many features, and the problem is that learning algorithms may perform poorly in high-dimensional data. Thus, we need to introduce the concept of dimensionality reduction to transform the features into a lower dimensional space. The idea behind this concept is very simple: we want to find the most significant 'features' of the TF-IDF matrix within a certain dimension and discard all the other 'features' that are not so significant. To do this, we use two methods in this project: 1. : Latent Semantic Indexing (LSI) 2. Non-negative Matrix Factorization (NMF). The idea behind LSI is just to use the concept of Singular Value Decomposition (SVD). I believe at this stage, everyone in this class should be familiar with SVD, so we will not discuss the details in this report. In short, suppose our TF-IDF matrix is X, then by performing SVD, X can be decomposed into three matrices: $X = U\Sigma V^T$. And we take the first k columns of U and V. Then we just use XV_k to represent the dimension-reduced data matrix, since V_k consists of the k principal components in the feature space. The idea of NMF is similar to LSI. Instead of decomposing X into three matrices, NMF method decomposes X into two matrices W and H, such that the Frobenius Norm

squared of X-WH is minimized. And then, we use W as the dimension-reduced data matrix.

QUESTION 3: Reduce the dimensionality of the data using the methods above

- **Apply LSI to the TF-IDF matrix corresponding to the 8 categories with k = 50; so each document is mapped to a 50-dimensional vector.**
- **Also reduce dimensionality through NMF (k = 50) and compare with LSI:**

Which one is larger, the $| X – WH|_F^2$ in NMF or the $|X- U\Sigma V^T|_F^2$ in LSI? Why is the case?

Our result is shown below:

```
    LSI method:
train:  (4732, 50)

test:  (3150, 50)

    NMF method:
train:  (4732, 50)

test:  (3150, 50)

Frobenius Norm squared for LSI: 63.844
Frobenius Norm squared for NMF: 64.204
```

We can see that we successfully transformed our TF-IDF matrix into a much smaller dimension matrix. For the Frobenius Norm squared values, we can see that LSI is a little bit smaller than NMF, we think that it is because SVD is an optimization problem without constraints, while NMF is solving an optimization problem with constraints that the resulting matrices must be non-negative, thus the error of NMF should be larger than SVD.

5. Classification Algorithms

In this section, we are going to use the dimension-reduced training dataset by using LSI to train different types of classifiers and evaluate the trained classifiers with test data. First of all, the documents are classified into two classes: ‘Computer Technology’ and ‘Recreational Activity’. Since in this case, we are only classifying two classes, thus we can use the idea of binary classification, i.e. we can give ‘Computer Technology’ class a label of 0, and give ‘Recreational Activity’ class a label of 1, or reverse, either way will work just fine. After we properly give the two classes their labels, we can then apply the training data to the classifiers.

The first classifier we used is called **Support Vector Machine (SVM)**. Detailed description of the theory behind this model is mentioned in the project handout, so we will not discuss it again in this report. However, we will discuss some performance metrics that are used to evaluate a given model. These are **accuracy**, **recall**, **precision**, **F-1 score**, **confusion matrix**, and **ROC curve.** Before we go into details of these metrics, there are four terms need to be clarified, and they are: **True Positive (TP)**, **True Negative (TN)**, **False Positive (FP)**, and **False Negative (FN)**. The definitions of these four terms are given as follows:

1. True Positive(TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.
2. True Negative(TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.
3. False Positives (FP) – When actual class is no and predicted class is yes.
4. False Negatives (FN) – When actual class is yes but predicted class in no.

With the understanding of these four terms, now we can define the performance metrics one by one:

1. Accuracy = TP+TN/TP+FP+FN+TN
2. Precision = TP/TP+FP
3. Recall = TP/TP+FN
4. F1 Score = 2*(Recall * Precision) / (Recall + Precision)
5. Confusion Matrix:

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

6. ROC Curve: ROC curve is created by plotting the true positive (TP) against the false positive(FP) at various threshold settings.

QUESTION 4: Hard margin and soft margin linear SVMs:

- **Train two linear SVMs and compare:**
  - **Train one SVM with $\gamma = 1000$ (hard margin), another with $\gamma = 0.0001$ (soft margin).**
  - **Plot the ROC curve, report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of both SVM classifier. Which one performs better?**
  - **What happens for the soft margin SVM? Why is the case? ∗ Does the ROC curve of the soft margin SVM look good? Does this conflict with other metrics?**

- **Use cross-validation to choose γ (use average validation accuracy to compare): Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10k | -3 \le k \le 3, k \in Z\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this best SVM.**

The result is shown below:

Table 1 Accuracy, recall, precision and F-1 score of SVM with hard/soft margin

	Accuracy	Recall	Precision	F-1 score
Hard margin	0.9714285714285714	0.9735849056603774	0.9699248120300752	0.9717514124293785
Soft margin	0.5047619047619047	1.0	0.5047619047619047	0.6708860759493671

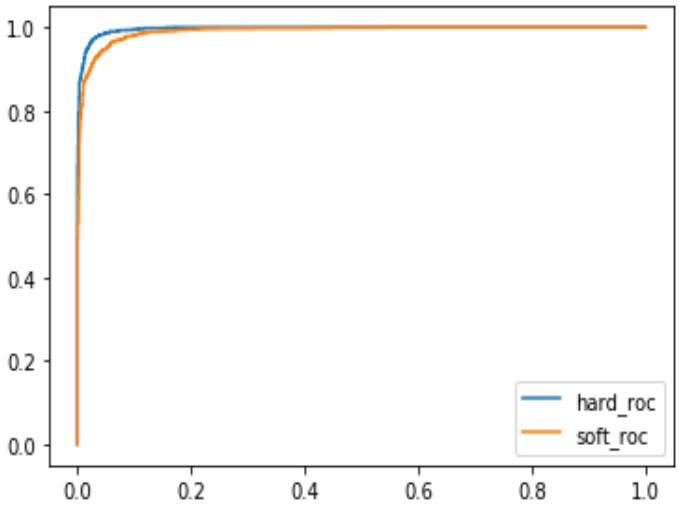
The confusion matrix for hard margin is:

```
[[1512   48]
[  42 1548]]
```

The confusion matrix for soft margin is:

```
[[   0 1560]
[   0 1590]]
```

And the ROC curves for hard/soft margins are shown below:



We can clearly see that the area under the hard roc curve is larger than the area under the soft roc curve, which means that the hard margin has better performance than the soft margin. By using the 5-fold cross validation, we find out the best value of the parameter gamma and the best score are:

```
0.9731612681417451
{'C': 10.0}
```

Which means gamma is 10 in this case. Then by repeating the same process in the previous steps, we get the results as follows:

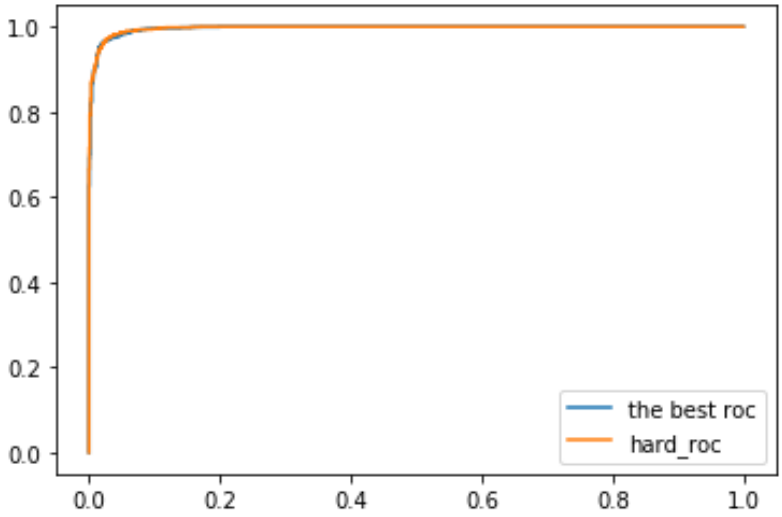
Table 2. Accuracy, recall, precision and F-1 score of SVM with gamma = 10

Accuracy	Recall	Precision	F-1 score
0.966984126984127	0.9767295597484277	0.9586419753086419	0.9676012461059189

And the confusion matrix is:

```
[[1493   67]
 [  37 1553]]
```

The ROC curve is:



The second classifier we used is **Logistic Regression**. Logistic Regression is a well-known model that is used for binary classification. One thing to be noticed is that logistic regression is a probability model and the idea behind this model is that a logistic function $\sigma(\varphi) = 1/(1+\exp(-\varphi))$ is acting on a linear function of the features: $\varphi(x) = w^T x + b$, to calculate the probability that a given data point belongs to class one. During the training process, w and b that maximizes the likelihood are found. Regularization term can also be added to deal with the problems of ill-conditioned results or over-fitting. Detailed discussion of regularization in Logistics Regression will be presented in the following section.

QUESTION 5: Logistic classifier:

- **Train a logistic classifier without regularization (you may need to come up with some way to approximate this if you use sklearn.linear_model.LogisticRegression); plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this classifier.**
- **Regularization:**
  - **Using 5-fold cross-validation on the dimension-reduced-by-svd training data, find the best regularization strength in the range $\{10k \mid -3 \le k \le 3, k \in Z\}$ for logistic regression with L1 regularization and logistic regression L2 regularization, respectively.**
  - **Compare the performance (accuracy, precision, recall and F-1 score) of 3 logistic classifiers: w/o regularization, w/ L1 regularization and w/ L2 regularization (with the best parameters you found from the part above), using test data.**
  - **How does the regularization parameter affect the test error? How are the learnt coefficients affected? Why might one be interested in each type of regularization?**
  - **Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary, then what is the difference between their ways to find this boundary? Why their performance differs?**

The result of this part is shown below:

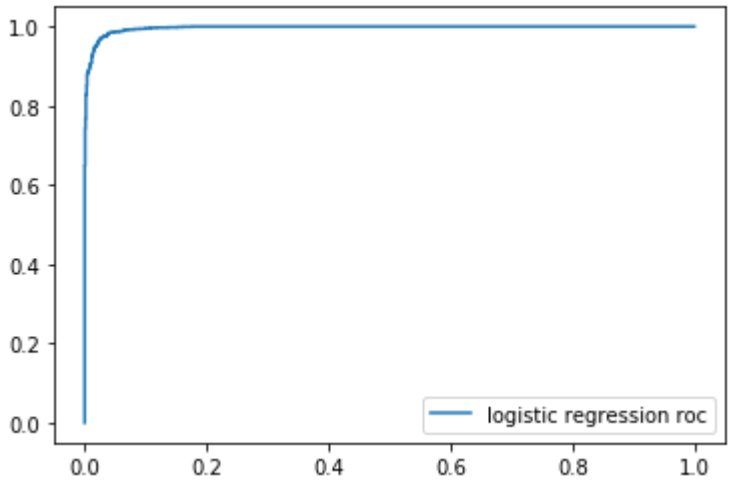
Table 3. Accuracy, recall, precision and F-1 score of logistic regression without regularization

Accuracy	Recall	Precision	F-1 score
0.9707936507936508	0.9761006289308176	0.9663760896637609	0.9712140175219023

The confusion matrix is:

```
logistic_regression Confusion Matrix:
            [[1506   54]
            [  38 1552]]
```

And the ROC curve is shown below:



Then, we used the 5-fold cross-validation to find the best regularization coefficients for L1 regularization and L2 regularization. It turned out that 10 is best for L1 and 100 is best for L2:

```
L1:  0.9712596359707188
   L1:  {'C': 10.0}
L2:  0.9699911370278012
   L2:  {'C': 100.0}
```

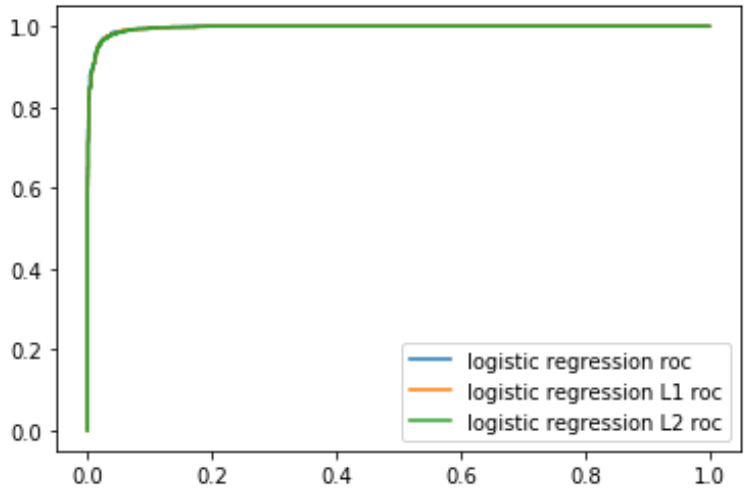
Then we compare the performance of these 3 classifiers:

Table 4. Accuracy, recall, precision and F-1 score of the 3 classifiers

	Accuracy	Recall	Precision	F-1 score
No regularization	0.9707936507936508	0.9761006289308176	0.9663760896637609	0.9712140175219023
L1	0.9711111111111111	0.9761006289308176	0.9669781931464174	0.9715179968701095
L2	0.9698412698412698	0.9767295597484277	0.9639975170701428	0.9703217744454857

For completeness, the confusion matrices for L1 and L2 regularizations and their ROC curves are shown below:

```
logistic regression L1 Confusion Matrix:
              [[1507   53]
              [  38 1552]]
logistic regression L2 Confusion Matrix:
              [[1502   58]
              [  37 1553]]
```



Furthermore, the Area Under the Curve (AUC) values of these 3 classifiers are presented below for further comparison:

```
No Regularization AUC:  0.9965344299306562
L1 AUC:  0.99632760845025
L2 AUC:  0.996306644089663
```

In general, regularization terms only improve the performance on new and unseen data, and there is no need to discuss its performance on the training data. Thus, in general, with regularization terms, the trained model will perform better on new dataset. As for the coefficients, we all know that regularization penalizes high coefficients, thus with regularization terms introduced, the coefficient values will be smaller. Now, one interesting thing to be noticed is that, L1 regularization will create sparse coefficient values with just a few higher value coefficients, while L2 regularization will in general create small coefficient values. Given these properties, we can say that when dealing with feature selection with a large number of features, L1 regularization will help, since it will shrink the less important feature's coefficient to zero. On the other hand, L2 regularization will deal with the problem of multicollinearity.

Although logistic regression and linear SVM are trying to classify data points using a linear decision boundary, there are some major difference between these two methods. SVM tends to maximize the margin between the closest support vectors, thus SVM can find a solution which is as fair as possible for the two categories, while Logistic Regression cannot do this. Another important thing to be noticed is that SVM produces binary classification values, i.e. 1 or 0, and Logistic Regression produces probabilistic values, thus we can say that Logistic Regression does not make absolute prediction. Overall, the approaches of these two methods are fundamentally different, SVM tries to find the farthest possible separating margin, while Logistic Regression tries to optimize the log-likelihood function with the sigmoid function providing the probability model.

Next, we played with the **Naïve Bayes** classifiers. There are three different classifiers provided: **Multinomial NB**, **BernoulliNB**, and **GaussianNB**. Naïve Bayes' assumption is really simple, this classifier assume that features are statistically independent of each other, and calculate the Maximum A Posteriori estimation of the labels:

$$\Pr\left(x_i \middle| y, x_1, \ldots x_i, x_{i+1}, \ldots = x_m\right) = P\left(x_i \middle| y\right), i \in \{1, \ldots, m\}$$

where x_i are features, i.e. components of a data point, and y is the label of the data point. Each NB classifier is backed by different probabilistic model. As indicated by their names, **Multinomial NB** uses Multinomial Distribution, **BernoulliNB** uses Bernoulli Distribution, and **GaussianNB** uses Gaussian Distribution. The probability density function of each distribution is shown below:

1. Multinomial:

$$f\left(x_1, \ldots x_k; n, p_1, \ldots p_k\right) = \Pr\left(X_1 = x_1 \text{and} \ldots \text{and} X_k = x_k\right)$$
$$= \begin{cases} \dfrac{n!}{x_1! \ldots x_k!} p_1^{x_1} \times \ldots \times p_k^{x_k}, when \sum_{i=1}^{k} x_i = n \\ 0, otherwise \end{cases}$$

2. Bernoulli:

$$f\left(k; p\right) = \begin{cases} p, \text{if } k=1 \\ q = 1 - p, \text{if } k=0 \end{cases}$$

3. Gaussian:

$$P\left(x\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/\left(2\sigma^2\right)}$$

QUESTION 6: Naive Bayes classifier: train a GaussianNB classifier; plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of this classifier.

The results are shown below:

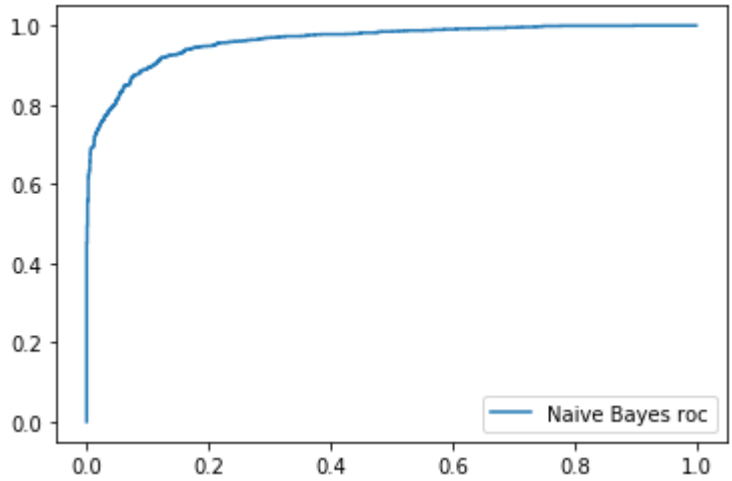
Table 5. Accuracy, recall, precision and F-1 score of the GaussianNB classifier

Accuracy	Recall	Precision	F-1 score
0.7984126984126985	0.9773584905660377	0.7217835578262889	0.8303499866417312

The confusion matrix is:

```
Naive Bayes Confusion Matrix:
[[ 961  599]
[  36 1554]]
```

The ROC curve is:



QUESTION 7: Grid search of parameters:

At this stage, we finished playing with all three classifiers. Next, we want to find out a best combination that can provide the best test accuracy when doing the 5-fold cross-validation. There are many parameters (options) that we can tune. Here are some options we can tune:

1. Loading Data: remove ‘headers’ and ‘footers’ or not
2. Feature Extraction: min_df = 3 or 5; use lemmatization or not
3. Dimensionality Reduction: LSI or NMF
4. Classifier: SVM or Logistic Regression or GaussianNB

One thing can be immediately determined is that when comparing the performance of different classifiers, we can see that SVM and Logistic Regression have almost the same test accuracy, while GaussianNB’s accuracy is less than these two, thus we think that we do not have to consider GaussianNB in this testing process. Another thing is that including ‘headers’ and ‘footers’ will definitely not improve the performance. Moreover, not doing lemmatization will not improve the performance, thus in order to speed up the running time of the program, we decided just to focus on ‘min_df’, Dimensionality Reduction, and different classifiers.

After conducting the grid search with various parameters, we can see that the best combination is using Logistic Regression with L1 regularization strength 10, SVD method, and ‘min_df’ = 5, with the accuracy = 0.968299. For detailed results, please see our notebook file.

In the last section, we are focusing on the multiclass classification instead of two. For Naïve Bayes classification, it only care about the likelihood function that maximum the possibility of each classes. So it could be used in multiclass classification without any changes. However, for SVM, it needs some extended. Both One vs One classification and One vs Rest classification can be used to extend the binary classification.

QUESTION 8: In this part, we aim to learn classifiers on the documents belonging to the classes:

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

Perform Naïve Bayes classification and multiclass SVM classification (with both One VS One and One VS the rest methods described above) and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of your classifiers.

```
Naive Bayes (One vs One)

Accuracy:0.670926517571885

Recall:[0.59693878 0.36883117 0.72564103 0.98241206]

Precision:[0.61096606 0.7244898  0.56039604 0.81288981]

F1-score:[0.60387097 0.48881239 0.63240223 0.88964733]

Confusion Matrix:
[[234  34 107  17]
 [ 94 142 108  41]
 [ 55  20 283  32]
 [  0   0   7 391]]


SVM (One vs One)

Accuracy:0.8485623003194889

Recall:[0.81122449 0.76883117 0.85897436 0.95226131]

Precision:[0.7535545  0.80216802 0.85897436 0.98697917]

F1-score:[0.78132678 0.78514589 0.85897436 0.96930946]

Confusion Matrix:
[[318  50  23   1]
 [ 67 296  20   2]
 [ 31  22 335   2]
 [  6   1  12 379]]


Naive Bayes (One vs Rest)

Accuracy:0.6702875399361022

Recall:[0.57142857 0.37402597 0.74358974 0.98241206]

Precision:[0.61707989 0.70588235 0.55343511 0.82489451]

F1-score:[0.59337748 0.48896435 0.6345733  0.89678899]

Confusion Matrix:
[[224  38 115  15]
 [ 89 144 112  40]
 [ 50  22 290  28]
 [  0   0   7 391]]
```

```
SVM (One vs Rest)

Accuracy:0.8523961661341853

Recall:[0.77806122 0.79220779 0.86410256 0.97236181]

Precision:[0.78608247 0.78205128 0.85969388 0.97974684]

F1-score:[0.78205128 0.78709677 0.86189258 0.97604035]

Confusion Matrix:
[[305  60  26   1]
 [ 52 305  23   5]
 [ 28  23 337   2]
 [  3   2   6 387]]
```

According to the data, it shows that the SVM with One vs Rest technique has the best performance of accuracy.