Project No. 02:
Classification Analysis

By: Gu Dazhong, Zhu Yannan, DUCK-HA HWANG

Instructor: Roychowdhury Vwani

EE 239AS Winter 2016

February 22, 2016

I. INTRODUCTION

A. Purpose

In this project, we will focus on the classification problem. Classification means the task of identifying a category, from a predefined set. Classification turns out as an essential element of data analysis, especially when dealing with a large amount of data. So how to make an efficient method of classification is of great importance in analyzing data. In this project, we will use different methods for classifying texual data. Both two-class classification and multiclass classification.

B. Equipment

There is a minimal amount of equipment to be used in this project. The few requirements are listed below:

- Spyder-app (v2.3.8)
- Computer capable of running the software mentioned

C. Procedure

1. Read the data from the dataset and sample the documents in each class to make sure that each class has the same documents number.
2. Extract term-document matrix of the dataset and make the TFxIDF matrix
3. Make a TFxICF matrix of the data.
4. Select 50 most important terms of the features.
5. Use hard margin SVM classification to group 8 subclasses into two classes
6. Use soft margin SVM classification to group 8 subclasses into two classes
7. Use multinomial naive bayes classification to group 8 subclasses into two classes
8. Use logistic regression classification to group 8 subclasses into two classes
9. Use multinomial naive bayes classification to group 4 subclasses
10. Use One VS One SVM classification to group 4 subclasses
11. Use One VS the rest SVM classification to group 4 subclasses

D. Structure

This report will be seprated into 5 parts:

1. Introduction
2. Modeling Data and Feature
3. Two Class Classification
4. Multiclass Classification
5. Discussion & Conclusion

II. MODELING DATA AND FEATURE

This section focus on the pre-processing of our data before classification. Our origin data is in the type of string, which seem like a chaos to the classifer and make the classifer very difficult to find the feature of each class. So we will transfer the data into the type of TFxIDF, which express the content of each documents in a very orderly way. We will also reduce the size of the TFxIDF matrix. There are thousands of terms in the whole dataset. However, only tens of the key terms really represent the feature of each class. So we will only keep these important key terms, which will remarkably decrease the workload of our program.

A. Dataset and Problem Statement

In this project we work with 20 Newsgroups dataset. It is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, each corresponding to a different topic.

To train a classifer, we must keep the train data "balanced", which means the number of documents in each class should be the same. To achieve this goal we must first know how many documetns is in each class.

The histogram shows the number of documents in each class:

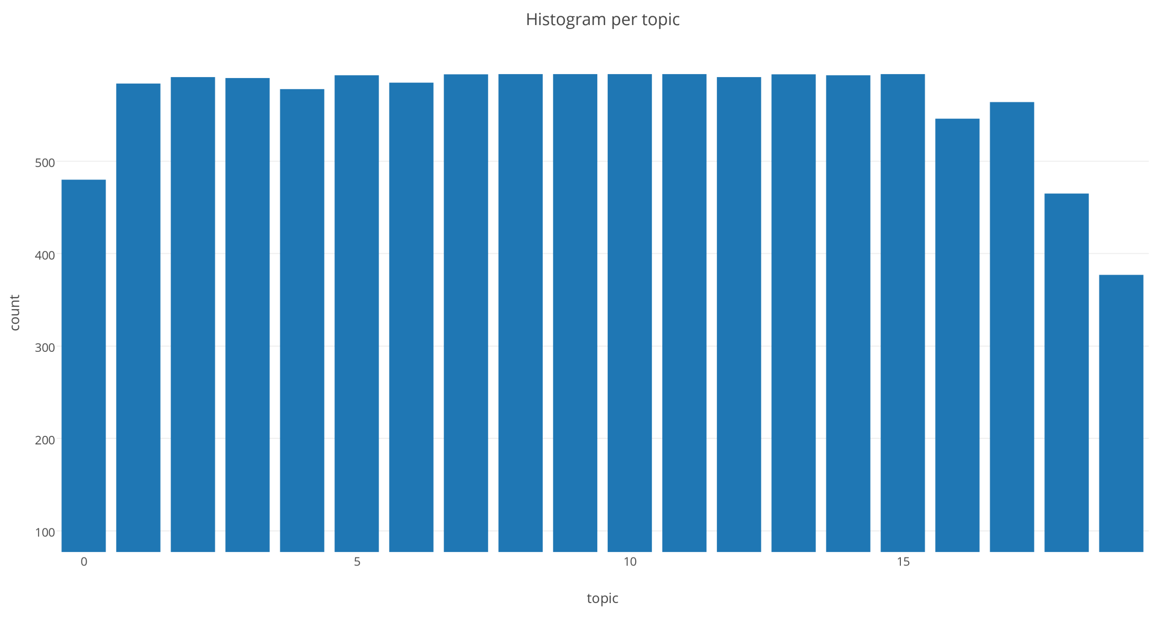
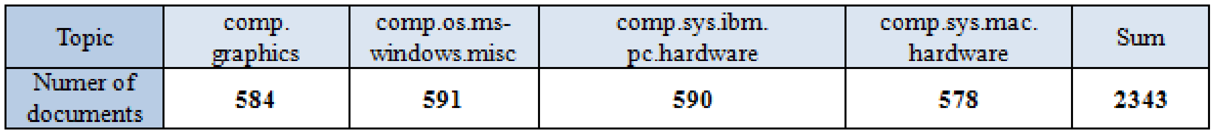


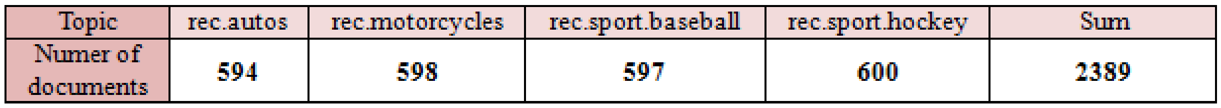
FIG. 1: Document number per topic

In the two-class classification part of out project. We take two classes into consideration, which are "computer technology" and "recreational activity". Both of them contain 4 subclasses. The two tables below show the documents number in each class:



Topic	comp. graphics	comp.os.ms-windows.misc	comp.sys.ibm. pc.hardware	comp.sys.mac. hardware	Sum
Numer of documents	**584**	**591**	**590**	**578**	**2343**

FIG. 2: Document number of computer technology



Topic	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey	Sum
Numer of documents	**594**	**598**	**597**	**600**	**2389**

FIG. 3: Document number of recreational activity

B. Modeling Text Data and Feature Extraction

Now we already know the document number in each topics. The topic “talk.religion.misc” has the fewest document number which is 377. To keep the dataset balanced, we randomly sample each topic to make all the topics have the same number of documents, which equals to the document num of topic “talk.religion.misc”.

Then we will extract the features of each documents. Here feature means the terms in the documents. We can make a term-document matrix to represent the features. We should be careful that in the documents there are some useless terms like “am, is, it, etc.” (we call these things stop words), which will interfere our classification. So in our term-document matrix, we should exclude the stop words, punctuations, and different stems of a word.

In python, we can use the fuction “CountVectorizer()” to do this easily.

Finally, in all the 20 topics, we get a term-document matrix the total number of terms of which is 109619 terms.

Then we can transfer the term-document matrix into TFxIDF matrix, which is a better way to show the features of each documents.

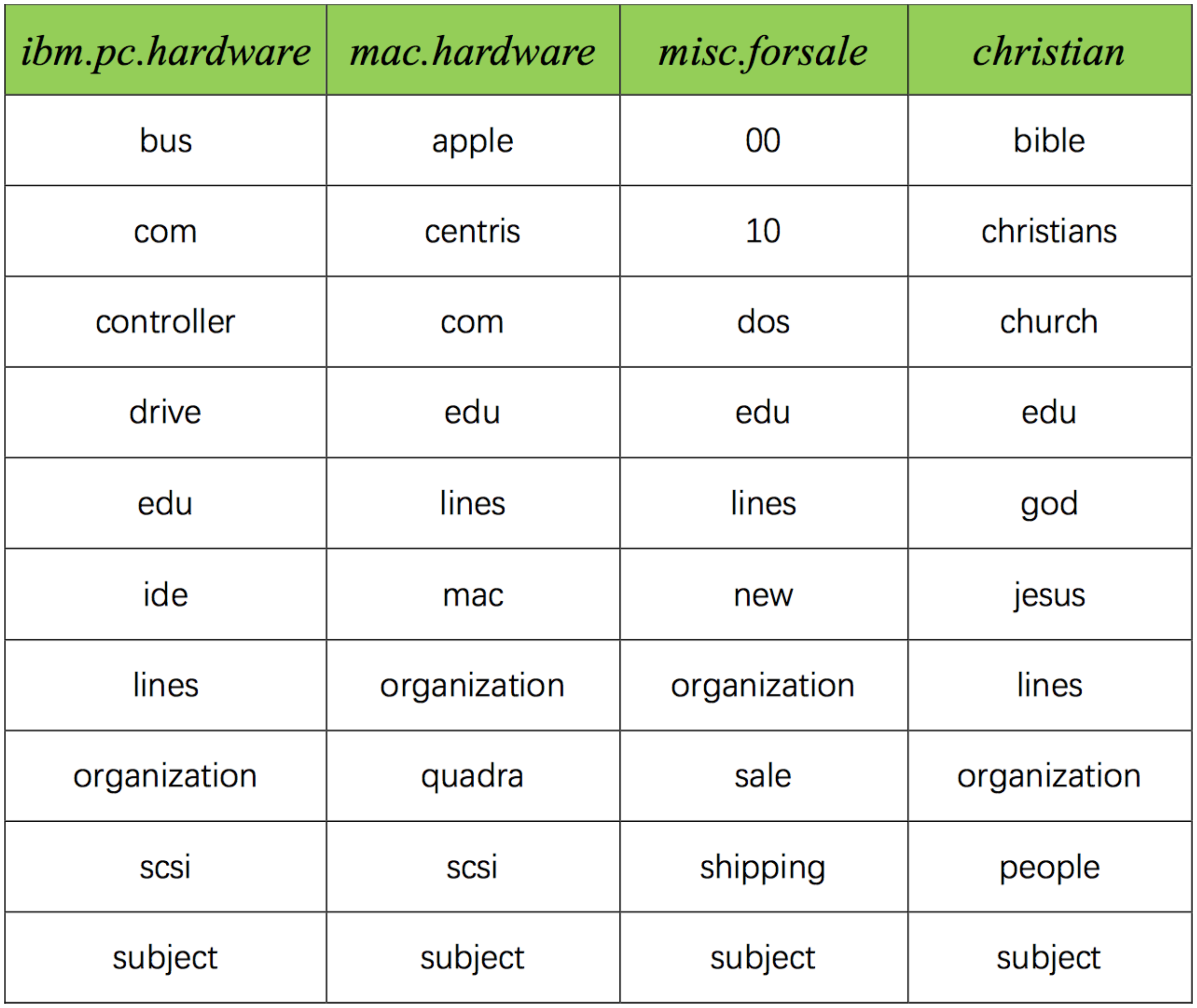
Similarly, in python we have fuction “TfidfTransformer()” to help us transfer term-document matrix into TFxIDF matrix.

C. TFxICF

The TFxIDF only give us the feature of each document. However we want to have a look of the whole featrue picture of each topics. So here we make a TFxICF matrix of the 20 topics. The definition of TFxICF is as below:

$$\left(0.5 + 0.5\frac{f_{t,c}}{max f_{t,c}}\right) * (log\frac{|C|}{c \in C : t \in c})$$

We can see it is very similiar to TFxIDF. The only difference is to change D to C. Using the TFx-ICF we can see that the 10 most important terms of the four class: “comp.sys.ibm.pc.hardware” , “comp.sys.mac.hardware”, “misc.forsale”, and “soc.religion.christian” are as below:



ibm.pc.hardware	*mac.hardware*	*misc.forsale*	*christian*
bus	apple	00	bible
com	centris	10	christians
controller	com	dos	church
drive	edu	edu	edu
edu	lines	lines	god
ide	mac	new	jesus
lines	organization	organization	lines
organization	quadra	sale	organization
scsi	scsi	shipping	people
subject	subject	subject	subject

FIG. 4: 10 most important terms of 4 topics

D. Feature Selection

From the above part, we get that the total number of terms in our matrix is 109619, which is a lot of terms. To handle these amount of data will be a big challange to our program. However, we find that most of the terms only show up once or twice in the whole dataset. So these terms doesn’t matter to our classifer at all. We should find a way to exclude them.

So we decide to keep only the most important 50 terms in our matrix. The way we are using is called Latent Semantic Indexing (LSI). The LSI does a SVD to ouor matrix and only keep the 50 biggest singular value of our matrix. So only the 50 most important terms remain in our matrix.

In python, the fuction "TruncatedSVD()", can help us select the 50 most important terms

III. TWO CLASS CLASSIFICATION

In this part we will use three method to do a two-class classification task. For SVM method we have a hard margin SVM and a soft margin SVM.

A. Hard Margin SVM

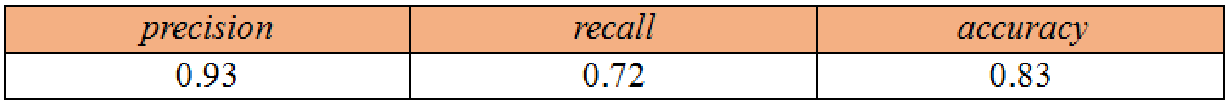
Linear Support Vector Machines have been proved efficient when dealing with sparse high dimensional datasets, including textual data. Its content is to solve the following optimization problem:

$$min\frac{1}{2}\left\|w\right\|_2^2+C\sum_{i=1}^{n}\xi_i$$

The value of C determine whether the margin is soft or hard. If C equals to 0, it is a hard margin SVM. If the C is not 0, it is a soft margin SVM. In this subsection, we use a hard margin SVM, which means we set C to 0.

In python, the fuction SVC() use SVM method to do two-class classification. Let us see how well this method works.

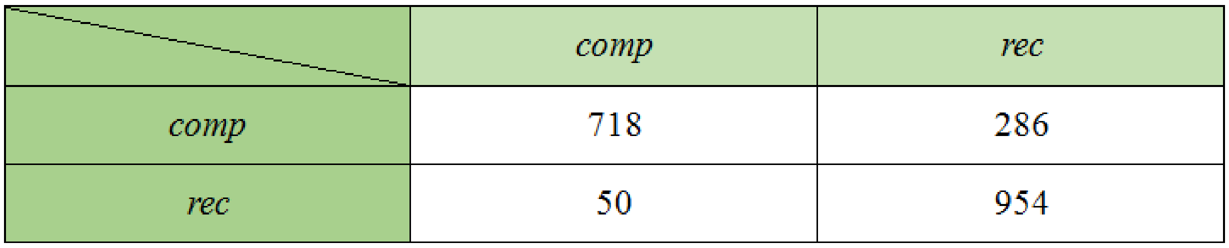
The precision, recall and accuracy is as below:



precision	*recall*	*accuracy*
0.93	0.72	0.83

FIG. 5: Precision, recall and accuracy for hard margin SVM classification

The confusion matrix is as below:



	comp	*rec*
comp	718	286
rec	50	954

FIG. 6: Confusion matrix for hard margin SVM classification

The ROC curve is as below:

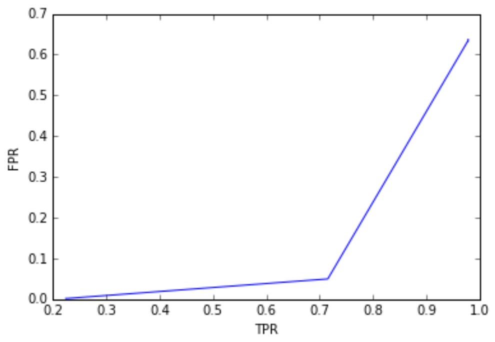
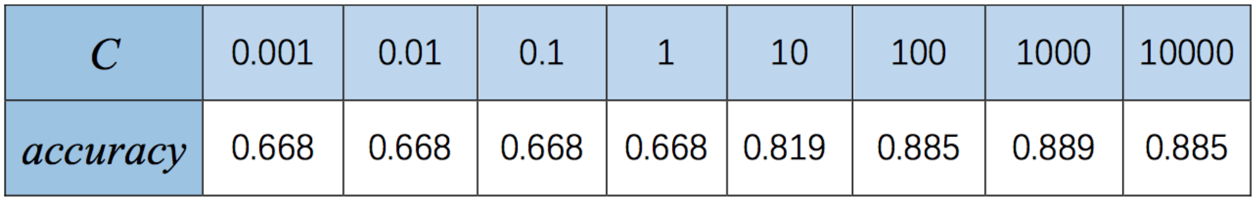


FIG. 7: ROC curves for hard margin SVM

B. Soft Margin SVM

As we said above the value of C determines whether it is a hard margin SVM or a soft margin SVM. In this part we use a soft margin SVM, which means we set C not equal to 0. In fact in this part we will try different C value to find the best one. C is the tolerance to error. If C is big, our model is less tolerable to error. So the bigger the C is, our model will more close to the train data. However, if the C is too big, our model will be too close to the train data, which make our model not suitable for other data. The over-fit problem comes. So a reasonable C value is important to the soft margin SVM.

In the soft margin SVM, we use a five-cross validation to train and test our model. This is our result of accuracy of models with different C values:



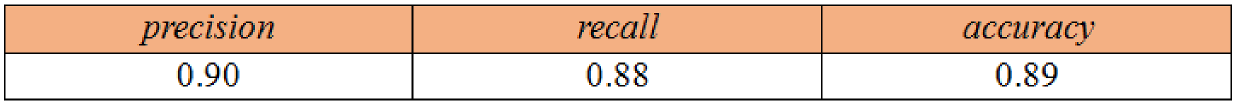
C	0.001	0.01	0.1	1	10	100	1000	10000
accuracy	0.668	0.668	0.668	0.668	0.819	0.885	0.889	0.885

FIG. 8: Accuracy vs different C values

We can see that when C equals to 1000 the result is the best. and when C is larger than 1000, like 10000 the result becomes bad because of the over-fit problem. However, when C is small, it doesn't do much influnce on the result, which is very different with the result of other team. In fact according to the theory wether C is too large or too small will influnce the result significantly. And the the over-fit problem should show up when C is much smaller. So we will find out the reason why our result is different from the theory in the future.

In python, the fuction SVC() use SVM method to do two-class classification. Let us see how well this method works. We just show the result when C is 1000, which is the best C for the model.

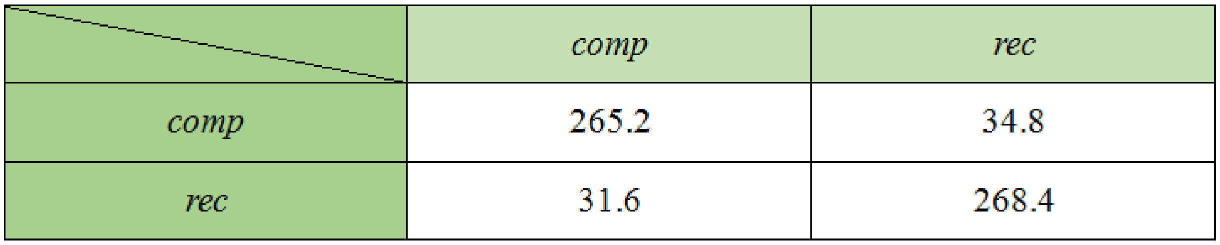
The precision, recall and accuracy is as below:



precision	*recall*	*accuracy*
0.90	0.88	0.89

FIG. 9: Precision, recall and accuracy for soft margin SVM classification

The confusion matrix is as below:



	comp	*rec*
comp	265.2	34.8
rec	31.6	268.4

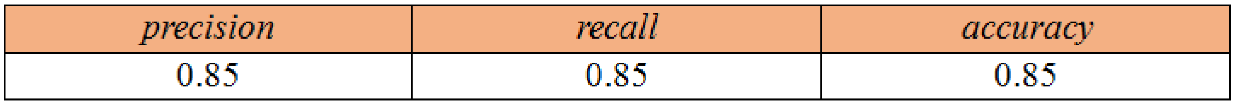
FIG. 10: Confusion matrix for soft margin SVM classification

C. Multinomial Naive Bayes Classifier

Next, we use naive Bayes algorithm for the same classification task. The algorithm estimates the maximum likelihood probability of a class given a document with feature set x, using Bayes rule, based upon the assumption that given the class, the features are statistically independent.

In python, the fuction MultinomialNB() use navie bayes method to do two-class classification. Let us see how well this method works.

The precision, recall and accuracy is as below:



precision	*recall*	*accuracy*
0.85	0.85	0.85

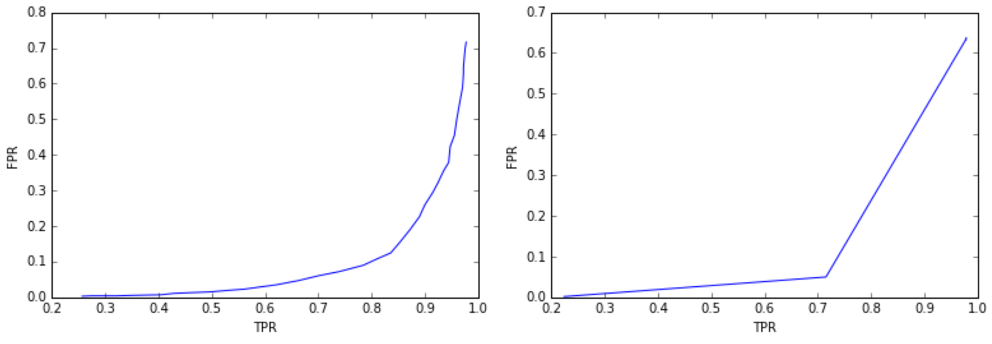
FIG. 11: Precision, recall and accuracy for navie bayes classification

The confusion matrix is as below:

	comp	*rec*
comp	855	149
rec	153	851

FIG. 12: Confusion matrix for navie bayes classification

The we compare its ROC curve with others as below:



(a) Naive Bayes (b) Hard margin SVM

FIG. 13: ROC curves for different method

D. Logistic Regression Classifier

In python, the fuction LogisticRegression() use logistic regression method to do two-class classification. Let us see how well this method works.

The precision, recall and accuracy is as below:

precision	*recall*	*accuracy*
0.90	0.86	0.88

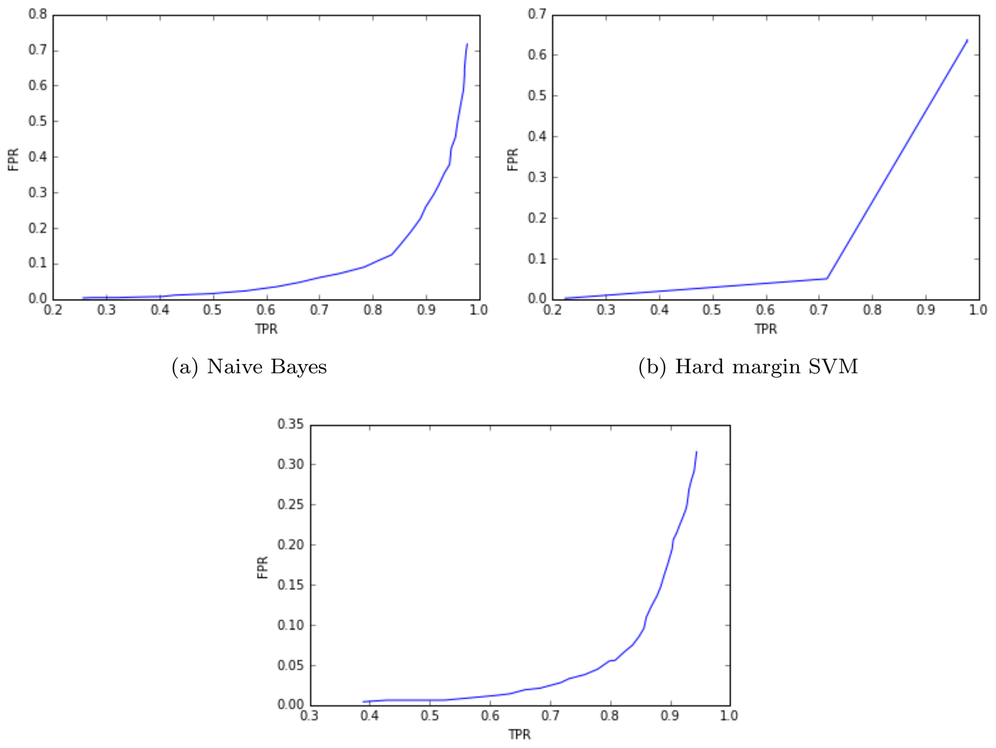
FIG. 14: Precision, recall and accuracy for logistic regression classification

The confusion matrix is as below:

	comp	*rec*
comp	860	144
rec	96	908

FIG. 15: Confusion matrix for logistic regression classification

The we compare its ROC curve with others as below:



(a) Naive Bayes

(b) Hard margin SVM

(c) Logistic Regression

FIG. 16: ROC curves for different method

IV. MULTICLASS CLASSIFICATION

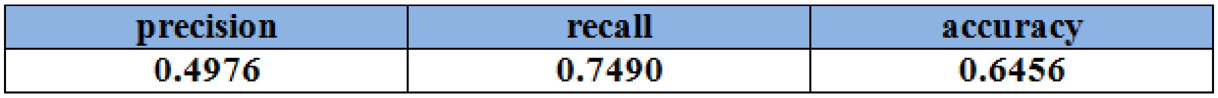
In the above section, we discuss how to train and use a two-class classifer. However, in daily application, there are usually more than two class which need to be classified. So in this section, we will talk about multiclass classification. In this section we mainly focus on two methods to do the multiclass classification: "naive bayes classification" and "SVM classification".

Naive bayes classification can do multiclass classification inherently. However, for SVM classifer, it can only do two-class classification. So we need to transfer the multiclass classification problem into two-class classification problem. There are to two ways to do this transformation: "One vs one SVM" and "One vs rest SVM"

A. Naive Bayes Classification

First, let us look at the naive bayes classification. The naive bayes classification can do multiclass classification inherently. So there is no big difference with two-class classification. The only change is now our train data contains four topics. Let us see how well this method works.

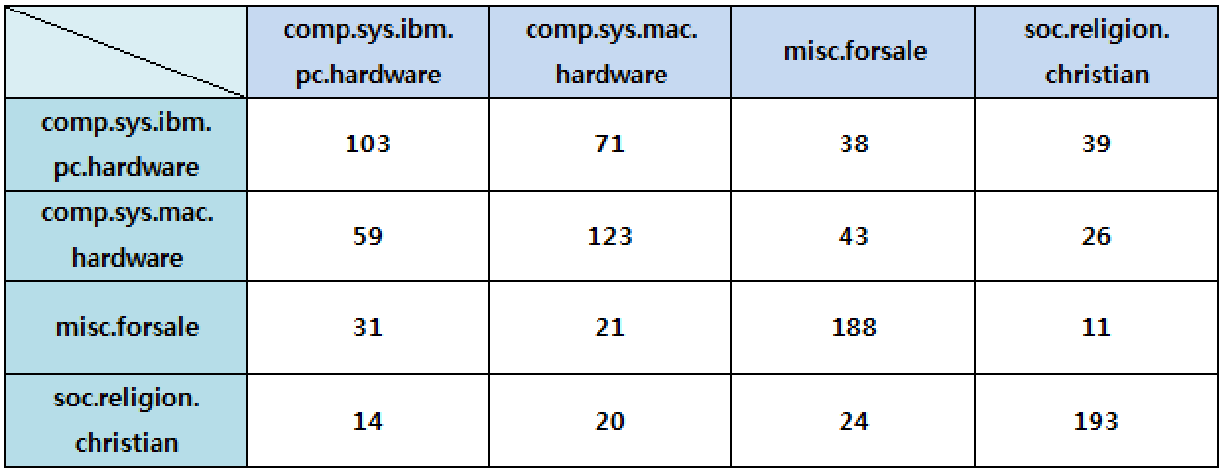
The precision, recall and accuracy is as below:



precision	recall	accuracy
0.4976	0.7490	0.6456

FIG. 17: Precision, recall and accuracy for naive bayes multiclass classification

The confusion matrix is as below:



	comp.sys.ibm. pc.hardware	comp.sys.mac. hardware	misc.forsale	soc.religion. christian
comp.sys.ibm. pc.hardware	103	71	38	39
comp.sys.mac. hardware	59	123	43	26
misc.forsale	31	21	188	11
soc.religion. christian	14	20	24	193

FIG. 18: Confusion matrix for naive bayes multiclass classification

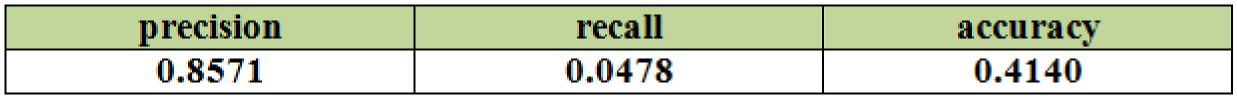
B. One VS One SVM Classification

Then we will use the SVM to do multiclass classification. As we said before, the SVM must transfer the multiclass classification problem into two-class classification. In this subsection, we talk about the first method, “the one vs one method”.

In this method, we build a classifer between each two class. So for 4 classes, we will build 6 classifer. For a particular document, the 6 classifer will give out 6 results. Maybe 3 classifer say that the document belongs to class 1. the other three say the document should be class 2,3,4. So class 1 has the highest score. We will put the document in class one. This is the basic content of the one vs one method.

In python, the fuction SVC() use one vs one method to do multiclass classification. Let us see how well this method works.

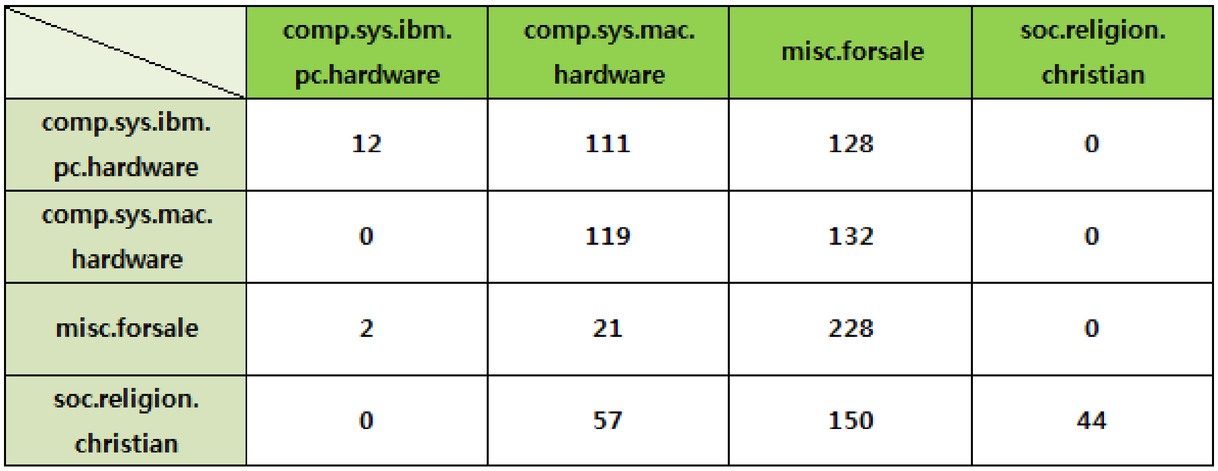
The precision, recall and accuracy is as below:



precision	recall	accuracy
0.8571	0.0478	0.4140

FIG. 19: Precision, recall and accuracy for one vs one SVM classification

The confusion matrix is as below:



	comp.sys.ibm. pc.hardware	comp.sys.mac. hardware	misc.forsale	soc.religion. christian
comp.sys.ibm. pc.hardware	12	111	128	0
comp.sys.mac. hardware	0	119	132	0
misc.forsale	2	21	228	0
soc.religion. christian	0	57	150	44

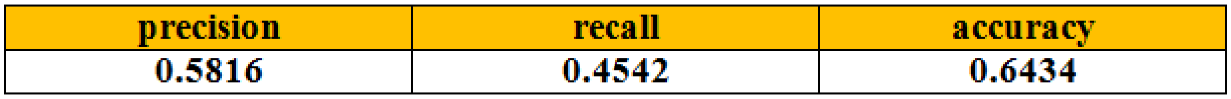
FIG. 20: Confusion matrix for one vs one SVM classification

C. One VS the Rest SVM Classification

The second method using SVM to do multiclass classification is the “one vs rest method”. In this method, for each class we build a classifer. For a particular document, each classfier will give out that whether the document belongs to this class. For example, classifer 1 says that the document belongs to class 1 and the classifer 2,3,4 say that the document dosen't belong to class 2,3,4. Then we can put the document in class 1. This is the basic content of the one vs one method.

In python, the fuction LinearSVC() use one vs one method to do multiclass classification. Let us see how well this method works.

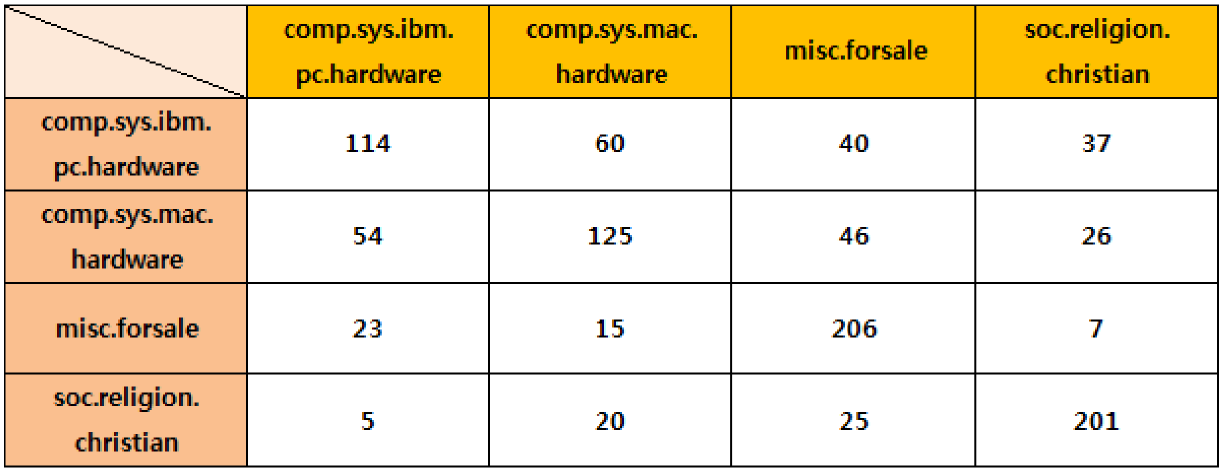
The precision, recall and accuracy is as below:



precision	recall	accuracy
0.5816	0.4542	0.6434

FIG. 21: Precision, recall and accuracy for one vs rest SVM classification

The confusion matrix is as below:



	comp.sys.ibm. pc.hardware	comp.sys.mac. hardware	misc.forsale	soc.religion. christian
comp.sys.ibm. pc.hardware	114	60	40	37
comp.sys.mac. hardware	54	125	46	26
misc.forsale	23	15	206	7
soc.religion. christian	5	20	25	201

FIG. 22: Confusion matrix for one vs rest SVM classification

V. DISCUSSION & CONCLUSION

In this project, we learn some basic ways to do classification. We really learn some useful ideas in this this project. Here I want to talk about two points.

First, in the data modeling part. The term-document matrix tell us an interesting way about how to deal with documents. Originally, the document is just a chaos of string, which is very difficult to handle in data. However the term-document matrix give us a way to order the strings. And the TFxIDF tell us a way to measure the importance of each term. This is also the basic idea of how to sort the search result on search engines like Google. The feature selection give us the idea to focus on the most important data and ignore the others, which can help us to reduce a lot of workload.

Second, in the SVM multiclass classification problem, the one vs one method and one vs rest method give us an idea about how to transfer one diffcult problem into several easy problem. The SVM originally can only do two-class classification. However, by combine several SVM classifer, we can use it to do multiclass classification problem. This is a very useful idea for us to use in our future works.