

EE 219  
Large-Scale Data Mining: Models and Algorithms

Project 2  
Classification Analysis  
Winter 2017

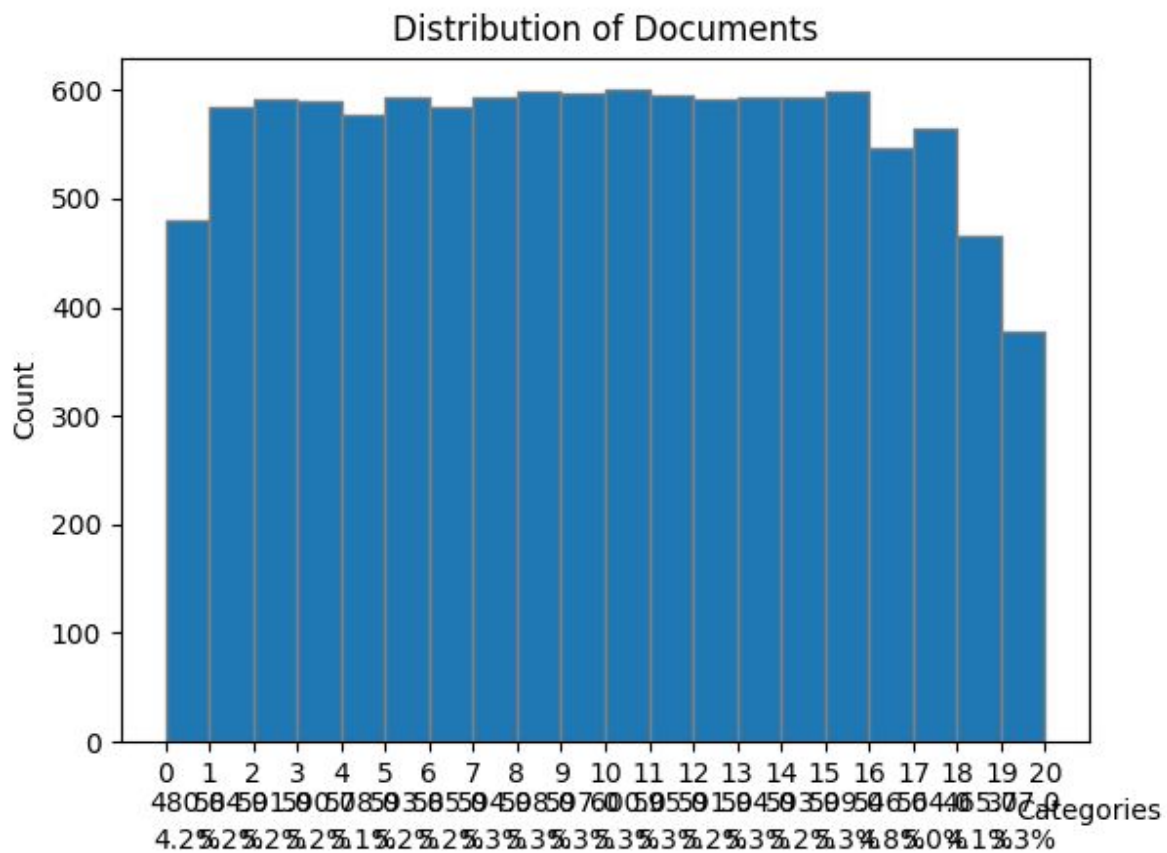
Guanqun Mao, Jianing Liu  
204777289 804759999

February 15th 2017

## a. Dataset and Problem Statement

The 20newsgroups dataset has about 18000 newsgroups posts on 20 topics split into two subsets: one for training and the other for testing. The split is based on the message posted before and after a specific date. There are 2354 documents of computer technology and 2389 documents of recreational activity. The numbers are fairly close, therefore there is no need to adjust the number of documents in preprocessing.

To make the datasets more balanced, we need to sample the data randomly and reduce the the number of classes to the lowest number of original classes. Here we plot a histogram (Figure 1) of the number of documents per topic to make sure they are evenly distributed.



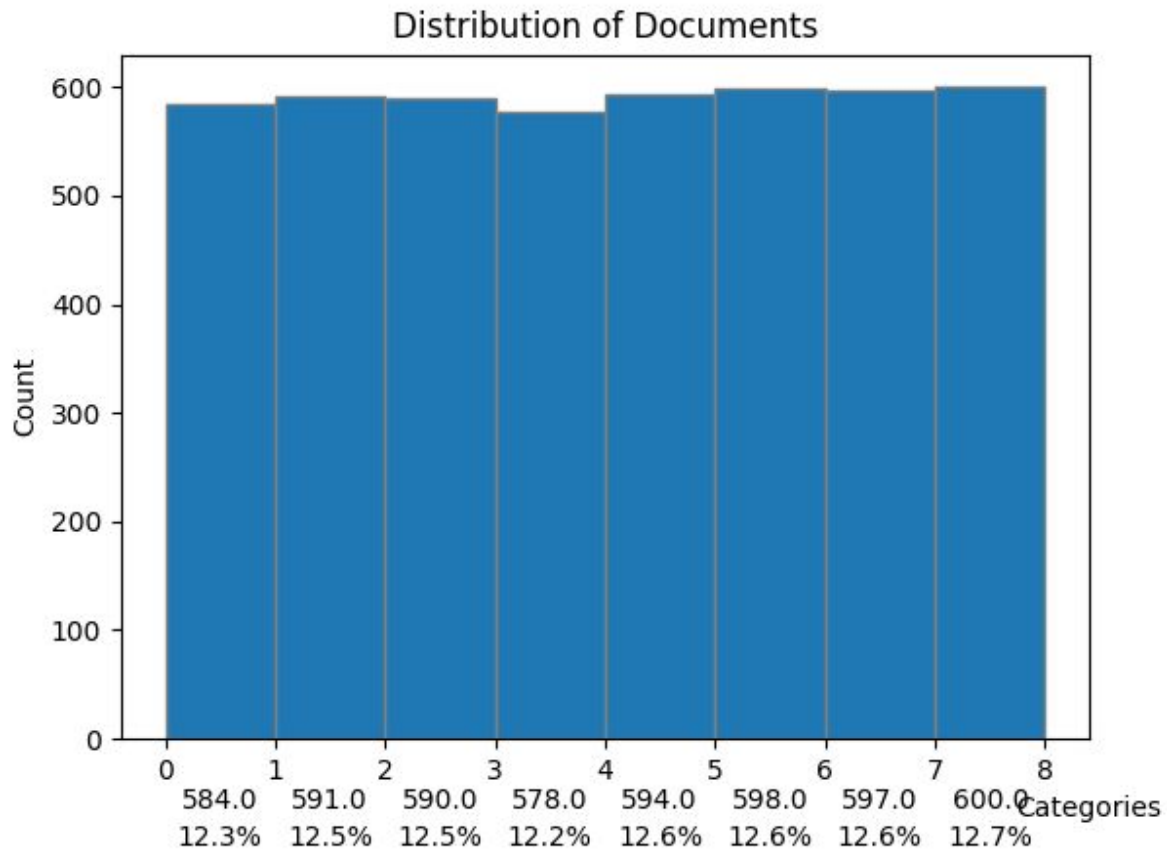


Figure 1 Histogram of the Number of Documents

## b. TFxIDF Vector Representations

For this part, we use “Bag of Words” assumption. This model is a simplifying representation used in natural language processing and information retrieval. It is commonly used in document classification, where the frequency of occurrence of every word is treated as a feature for training classifiers. The documents in our datasets will be represented as the bag of words, regardless of the grammar and word orders. The different stems of verbs, stop words and punctuations should be removed. In addition, special characters and words containing numbers will also be removed as they rarely occur.

TFxIDF is the short for Frequency Inverse Document Frequency. It is a numerical statistic intended to reflect how significant a word is to a document in a corpus of text. Given this, it is often used as a weight factor in information retrieval and text mining. The TFxIDF value is proportional to the number of times a word appears in a document, but is offset by the frequency

of the word in the corpus, which helps to adjust for the fact that some words appear more frequently than others.

We use TfidfVectorizer of the sklearn package to calculate all 11314 documents in 20 classes. In the end there are 54357 features.

### c. TFxICF and 10 Significant Terms

In part c, we find out whether a term is significant to a class in order to treat the whole set of documents in the same class as one document and calculate its TFxICF value. To calculate the 10 most significant terms in the following four classes: comp.sys.ibm.pc.hardware , comp.sys.mac.hardware, misc.forsale, and soc.religion.christian, we use CountVectorizer instead of tfidfVectorizer in sklearn, as TFxICF is not an viable option in this package. The 10 most significant terms are listed in Table 1:

Rank	comp.sys.ibm.pc.h ardware	comp.sys.mac.har dware	misc.forsale	soc.religion.christi an
1	balog	powerbook	sabretoo	clh
2	penev	lcii	liefeld	liturgy
3	scsiha	iis	hobgoblin	kulikauska
4	husak	adb	uccxkvb	mmalt
5	korenek	bmug	radley	copt
6	buslog	iivx	kou	caralv
7	laton	iifx	keown	monophysit
8	fasst	jartsu	koutd	mussack
9	mbyet	firstclass	spiderm	sspx
10	schaufenbuel	macus	mcfarlane	atterlep

Table 1: 10 Most Significant Terms in 4 Classes

### d. Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a dimension reducing transform that finds the optimal representation of the data in a lower dimensional space in the mean squared error sense. It can

be obtained by computing eigenvectors corresponding to the largest eigen values of the term-document matrix. The new low dimensional representations are the magnitudes of the projections of the documents into these latent topics. In this part, LSI is applied to the TFxIDF matrix with  $k = 50$ . Each document is mapped to a 50-dimensional vector. The selected features will then be used in our learning algorithm.

#### e. Hard Margin SVM

In the binary classification problem, we choose eight classes and separate the into two classes: Computer Technology and Recreational Activity, with 0 and 1 assigned to them correspondingly. The number of each subclass is close enough, therefore we do not need to balance the datasets.

In the hard margin SVM, the objective function is

$$\min \frac{1}{2} \|W\|_2^2$$

And the constraint is

$$y_i(W^T \vec{x}_i + b) \geq 1, i \in \{1, \dots, n\}$$

We set C to 100000 to make sure that the effect of hard margin is shown. The average precision is 96.83%, the average recall is 98.11% and the accuracy is 97.43%. The confusion table and the ROC curve are shown in Table 2 and Figure 2 respectively.

	Predicted comp	Predicted rect
Actual comp	1509	51
Actual rect	30	1560

Table 2

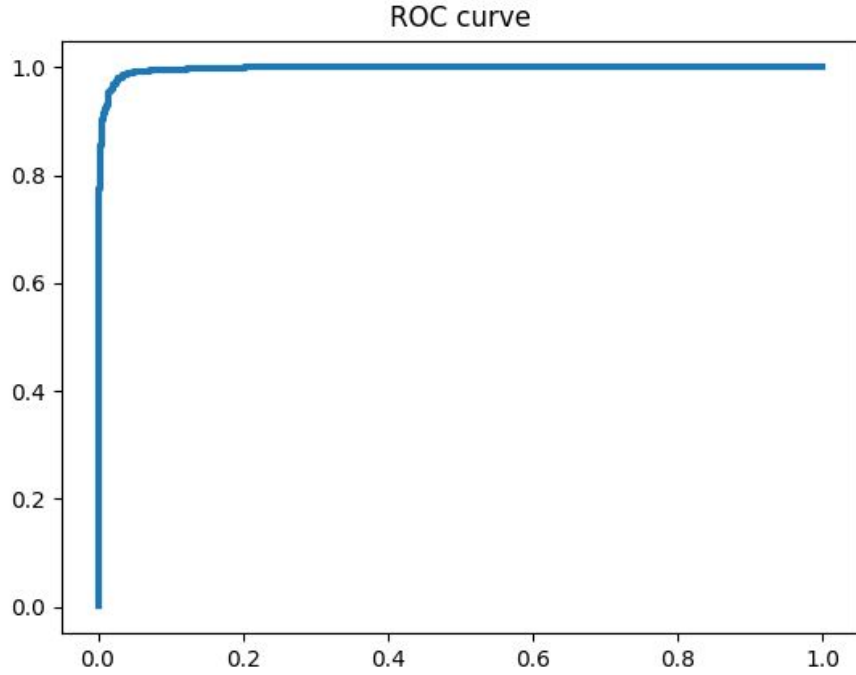


Figure 2

#### f. Soft Margin SVM

Although the hard margin SVM gives fairly good precision, average recall and accuracy for our classifier, it may create overfitting for our data. One alternative is to use the soft margin SVM. In this model, we add error parameter in the objective function and the constraints:

$$\min \frac{1}{2} \|W\|_2^2 + \gamma \sum_{i=1}^n \xi_i$$

Accordingly, the constraints are:

$$y_i(W^T \vec{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i \in \{1, \dots, n\}$$

The  $\gamma$  is the hyperparameter. Different values of  $\gamma$  will affect the classification results. We implement 5-fold cross validation to fit the model and choose the optimal  $\gamma$  value. The best  $\gamma$  value we could obtain is 100. The average precision is 98.11%, the average recall is 98.36% and the accuracy is 98.22%. The confusion matrix and the ROC curve are shown in Table 3.

	Predicted Comp	Predicted Rect
Actual Comp	769	15
Actual Rect	13	779

Table 3

### g. Naive Bayes Classifier

In this part, we perform the same classification task but with naive Bayes algorithm. This algorithm estimates the maximum likelihood probability of a class given a document with feature set based on the assumption that given the class, the features are statistically independent. The assumption may seem too simple, but this algorithm actually performs well for many practical problems and operates fast.

score= 0.900317460317

precision= 0.915364583333

recall= 0.88427672956

auc= 0.952333293017

We first use our matrix to train a Gaussian naive Bayes classifier. The precision is 91.54%, the recall is 88.43% and the accuracy is 90.03%. The confusion matrix and the ROC curve are shown in Table 4 and Figure 4 respectively. The area under the curve is 0.9523.

	Predicted Comp	Predicted Rect
Actual Comp	1430	130
Actual Rect	184	1406

Table 4

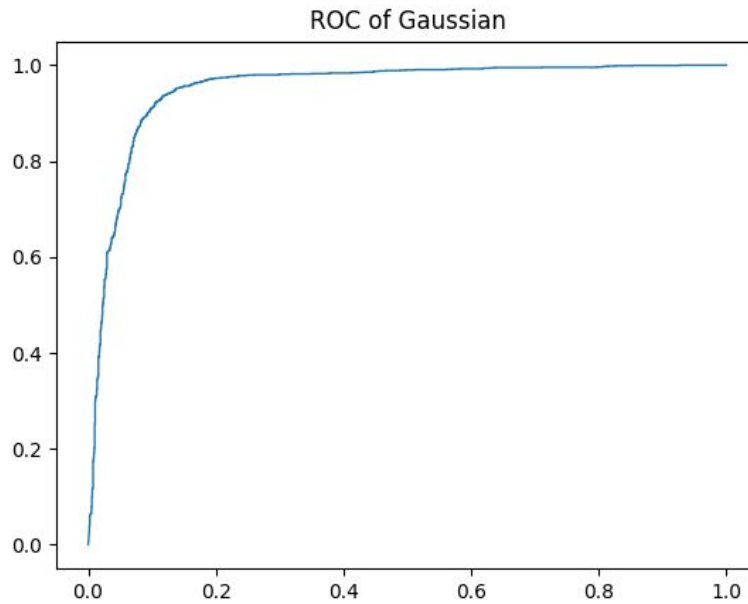


Figure 4

We also implement a Bernoulli naive Bayes classifier, which is suitable for text classification as well and may be better than the multinomial one. The main difference is that Bernoulli penalizes the non-occurrence of a feature explicitly. However, it should be noted that Bernoulli is only suitable for binary-value features. Therefore we first need to binarize the features before training the classifier.

The precision is 91.46%, the recall is 91.57% and the accuracy is 91.43%. The confusion matrix and the ROC curve are shown in Table 5 and Figure 5. The area under the curve is 0.9639.

	Predicted Comp	Predicted Rect
Actual Comp	1424	136
Actual Rect	134	1456

Table 5



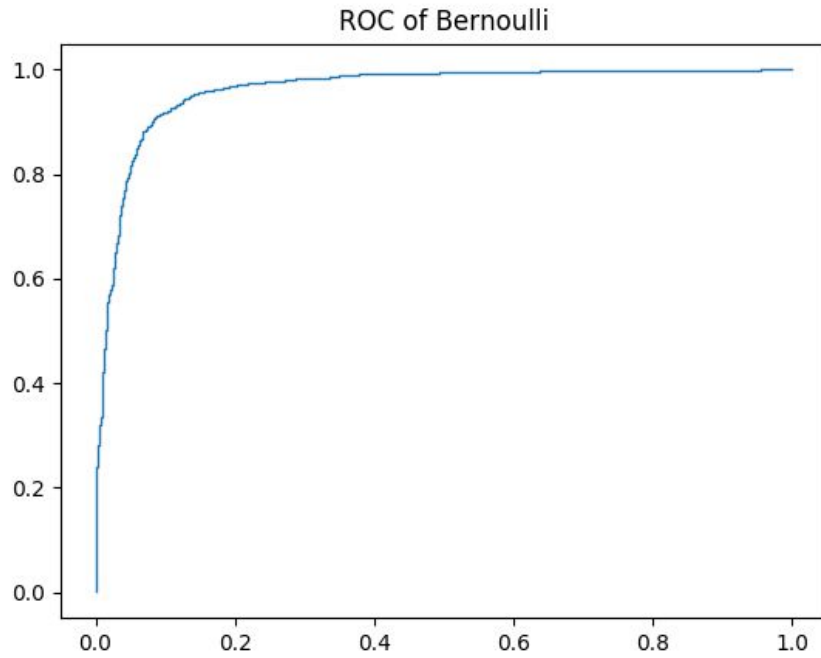


Figure 5

The result tells us that the performance of both kinds of classifiers are similar to each other and both can achieve accuracy of about 90%.

#### **h. Logistic Regression Classifier**

In this part we apply the logistic regression algorithm, a linear model for classification. There are two alternatives, L1 and L2 regularization, of penalization function to optimize this algorithm. In this part, we implement both regularization and compare the results.

Under L1 regularization, the precision is 95.98%, the accuracy is 96.73% and the recall is 97.61%. The confusion matrix and the ROC curve are shown in Table 6 and Figure 6. The area under curve is 0.9956.

	Predicted Comp	Predicted Rect
Actual Comp	1495	65
Actual Rect	38	1552

Table 6

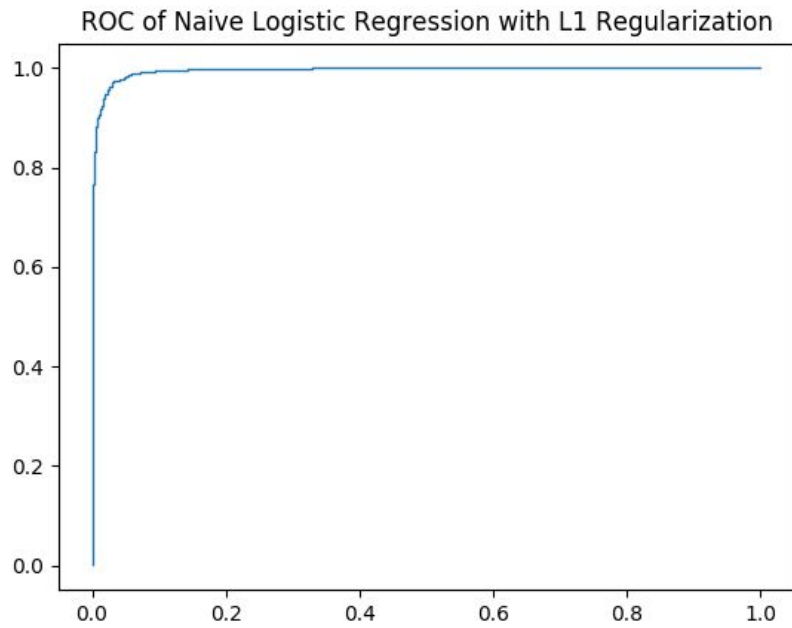


Figure 6

Under L2 regularization, the precision is 95.34%, the accuracy is 96.54% and the recall is 97.92%. The confusion matrix and the ROC curve are shown in Table 7 and Figure 7. The area under curve is 0.9950.

	Predicted Comp	Predicted Rect
Actual Comp	1484	76
Actual Rect	33	1557

Table 7

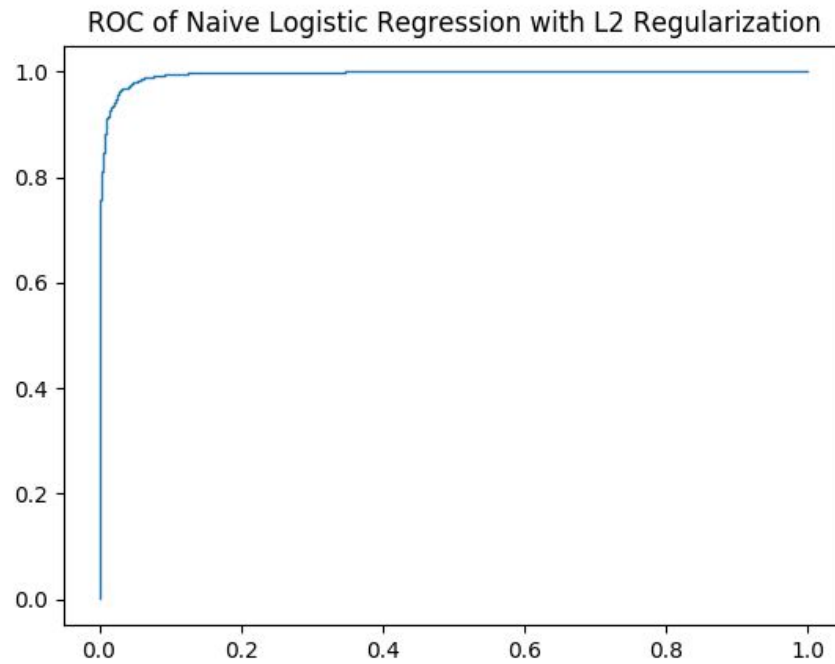


Figure 7

The results show that when trained with data after LSI processing, the form of regularization does not affect the performance of the logistic regression classifier much.

The ROC curves of all classifiers are shown in Figure 8.

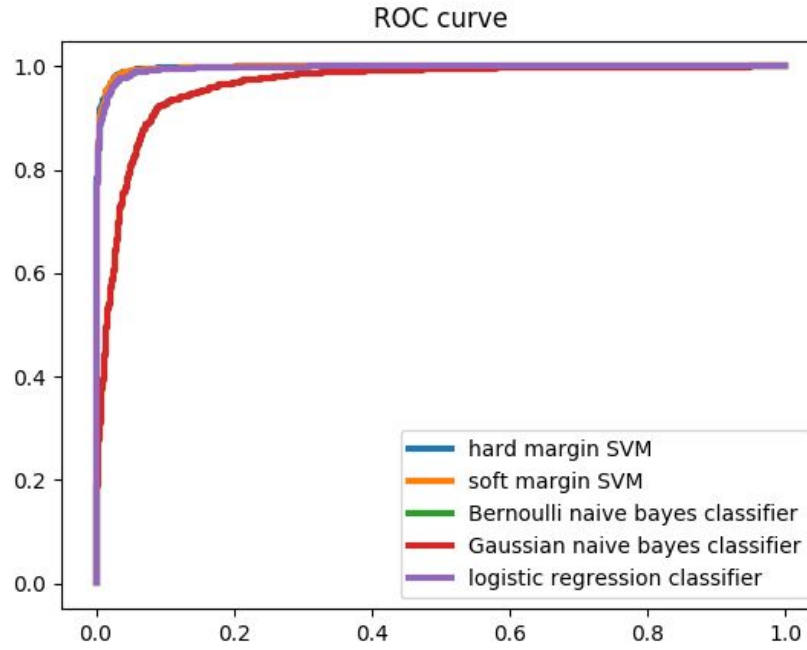


Figure 8

By applying both  $l_1$  and  $l_2$  norm regularizations and sweeping through different regularization coefficients, it appears that small values for parameters correspond to a simpler hypothesis and a simpler hypothesis is less prone to overfitting. In other words, smaller values tend to have fewer test errors and larger ones more. The coefficients of the fitted hyperplane are supposed to be those that separate the two target classes. When we have less error, it means the separation of two classes are good and the coefficients of the fitted hyperplane will decrease. In the reverse case where the separation of two classes is not handled well the coefficients will increase.

We observe that  $L_1$  regularization works well for recovering truly sparse data points, as they are computationally tractable but still capable of recovering the exact sparse solution.  $L_2$  regularization is preferable for data that is not at all sparse, i.e. where we do not expect regression coefficients to show a decaying property. Therefore, depending on the characteristics of the data, we may be interested in different types of regularization.

### i. Multiclass Classification

In this part we perform naive Bayes classification and multiclass SVM classification. Here we first discuss the unbalancing strategy between the 4 target classes. The number of files in each class is shown in Table 8. All 4 classes have about 390 files. Thus it is well-balanced.

Class Name	Number of Files
comp.sys.ibm.pc.hardware	392
comps.sys.mac.hardware	385
misc.forsale	390
soc.religion.christian	398

Table 8 Number of Files

The naive Bayes classifier performs the multiclass classification inherently. It finds the class with the maximum likelihood given the datasets. We implement both Gaussian and Bernoulli naive Bayes classifiers.

### **Gaussian**

For Gaussian classifier, the precision is 70.22%, the accuracy is 67.09% and the recall is 67.09%. The confusion matrix is shown in Table 9. The performance of Gaussian classifier is much worse than the binary case. Most of the errors occur between the first two classes - this may suggest that the content of these classes are quite similar. Thus, this classifier can not distinguish them.

Predicted	pc.hardware	mac.hardware	forsale	christian
Actual pc.hardware	244	36	112	0
Actual mac.hardware	98	142	144	1
Actual forsale	41	36	313	0
Actual christian	4	0	43	351

Table 9

### **Bernoulli**

For Bernoulli classifier, the precision is 83.33%, the accuracy is 83.19% and the recall is 83.19%. The confusion matrix is shown in Table 10. The matrix show that the Bernoulli gives much better performance than the Gaussian classifier. Although there are still errors in the first

two classes, the total number has decreased. This may be due to the feature binarization, with which we can distinguish between the ibm pc and mac.

Predicted	pc.hardware	mac.hardware	forsale	christian
Actual pc.hardware	296	69	25	2
Actual mac.hardware	62	299	21	3
Actual forsale	43	28	313	6
Actual christian	2	1	1	394

Table 10

### One VS One SVM

To implement SVM for multiclass classification, one way is to create a set of one-to-one SVMs between each pair of target classes, each of which vote for one of the two classes for a given piece of data. The class with highest votes will be picked.

We implement soft margin SVMs and set  $\gamma$  to 100. The precision is 88.29%, the accuracy is 88.24% and the recall is 88.24%. The confusion matrix is shown in Table 11.

Predicted	pc.hardware	mac.hardware	forsale	christian
Actual pc.hardware	318	50	24	0
Actual mac.hardware	35	324	26	0
Actual forsale	26	16	348	0
Actual christian	4	1	2	391

Table 11

### One VS Rest SVM

Another way to implement SVM for multiclass classification is to create a set of one vs. rest SVMs for every class. We only need  $n$  classifiers for  $n$  classes for efficiency. Since each class is represented by a classifier, it is convenient for us to gain insight into the classes.

We implement soft margin SVMs and set  $\gamma$  to 100. The precision is 88.65%, the accuracy is 88.63% and the recall is 88.63%. The confusion matrix is shown in Table 12.

Predicted	pc.hardware	mac.hardware	forsale	christian
Actual pc.hardware	318	45	27	2
Actual mac.hardware	32	326	26	1
Actual forsale	19	15	354	2
Actual christian	5	1	3	389

Table 12

We learn from the results that SVM classifiers outperform the naive Bayes classifiers and achieve an accuracy of near 90%. The first two classes are very similar to each other while the last class is most different from all other classes.

## Appendices

C:

number of terms: 54357

comp.sys.ibm.pc.hardware

[u'balog', u'penev', u'scsiha', u'husak', u'korenek', u'buslog', u'laton', u'fasst', u'mbyet', u'schaufenbuel']

comp.sys.mac.hardware

[u'powerbook', u'lci', u'iis', u'adb', u'bmug', u'iivx', u'iifx', u'jartsu', u'firstclass', u'macus']

misc.forsale

[u'sabretoo', u'liefeld', u'hobgoblin', u'uccxkv', u'radley', u'kou', u'keown', u'koutd', u'spiderm', u'mcfarlane']

soc.religion.christian

[u'clh', u'liturgy', u'kulikauska', u'mmalt', u'copt', u'caralv', u'monophysit', u'mussack', u'sspx',  
u'atterlep']

E

[[1509 51]

[ 30 1560]]

score = 0.974285714286

precision = 0.96834264432

recall = 0.981132075472

G

GaussianNB

confusion matrix:

[[1430 130]

[ 184 1406]]

score= 0.900317460317

precision= 0.915364583333

recall= 0.88427672956

auc= 0.952333293017

BernoulliNB, -0.007

confusion matrix:

[[1430 130]

[ 152 1438]]

score= 0.910476190476

precision= 0.917091836735

recall= 0.904402515723

auc= 0.96217989034

BernoulliNB, -0.0075

confusion matrix:

[[1429 131]

[ 142 1448]]



score= 0.913333333333  
precision= 0.917036098797  
recall= 0.910691823899  
auc= 0.962514916949

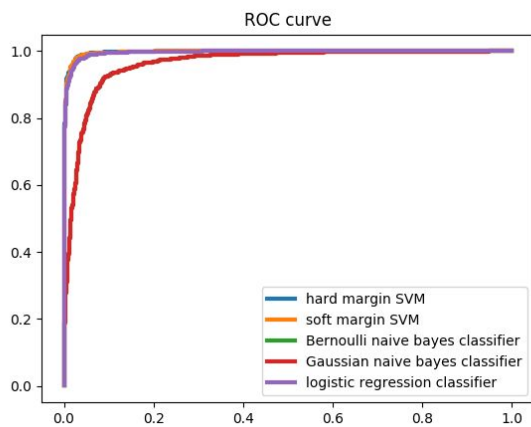
BernoulliNB, -0.008  
confusion matrix:  
[[1424 136]  
 [ 134 1456]]  
score= 0.914285714286  
precision= 0.914572864322  
recall= 0.91572327044  
auc= 0.963884857281

H1

LR1  
confusion matrix:  
[[1495 65]  
 [ 38 1552]]  
score= 0.967301587302  
precision= 0.959802102659  
recall= 0.976100628931  
auc= 0.99559425899

LR2  
confusion matrix:  
[[1484 76]  
 [ 33 1557]]  
score= 0.965396825397  
precision= 0.953459889773  
recall= 0.979245283019  
auc= 0.995007660055

H2



I

1 VS 1 SVC

confusion matrix:

[[318 50 24 0]

[ 35 324 26 0]

[ 26 16 348 0]

[ 4 1 2 391]]

score= 0.882428115016

precision= 0.882939755814

recall= 0.882428115016

1 VS Rest SVC

confusion matrix:

[[318 45 27 2]

[ 32 326 26 1]

[ 19 15 354 2]

[ 5 1 3 389]]

score= 0.886261980831

precision= 0.886454315788

recall= 0.886261980831

GaussianNB

confusion matrix:

[[244 36 112 0]

[ 98 142 144 1]

```
[ 41 36 313  0]
[  4  0 43 351]]
score= 0.670926517572
precision= 0.702204375077
recall= 0.670926517572
```

BernoulliNB

confusion matrix:

```
[[296 69 25  2]
 [ 62 299 21  3]
 [ 43 28 313  6]
 [  2  1  1 394]]
score= 0.831948881789
precision= 0.833326689424
recall= 0.831948881789
```