EE219 Project 1 - Report

Classification Analysis on Textual Data

Winter 2018

Nrithya Theetharappan
nrithya@ucla.edu
004946349

Shraddha Manchekar
smanchekar@ucla.edu
004945217

Introduction

Statistical classification refers to the task of identifying a category, from a predefined set, to which a data point belongs, given a training data set with known category memberships. In this project, we implement different methods for classifying textual data. We work with the '20 newsgroup' dataset, which is a collection of approximately 20,000 newsgroup documents. We've used several classifiers such as SVM, Naive Bayes and Logistic Regression to classify the documents based on the textual data.

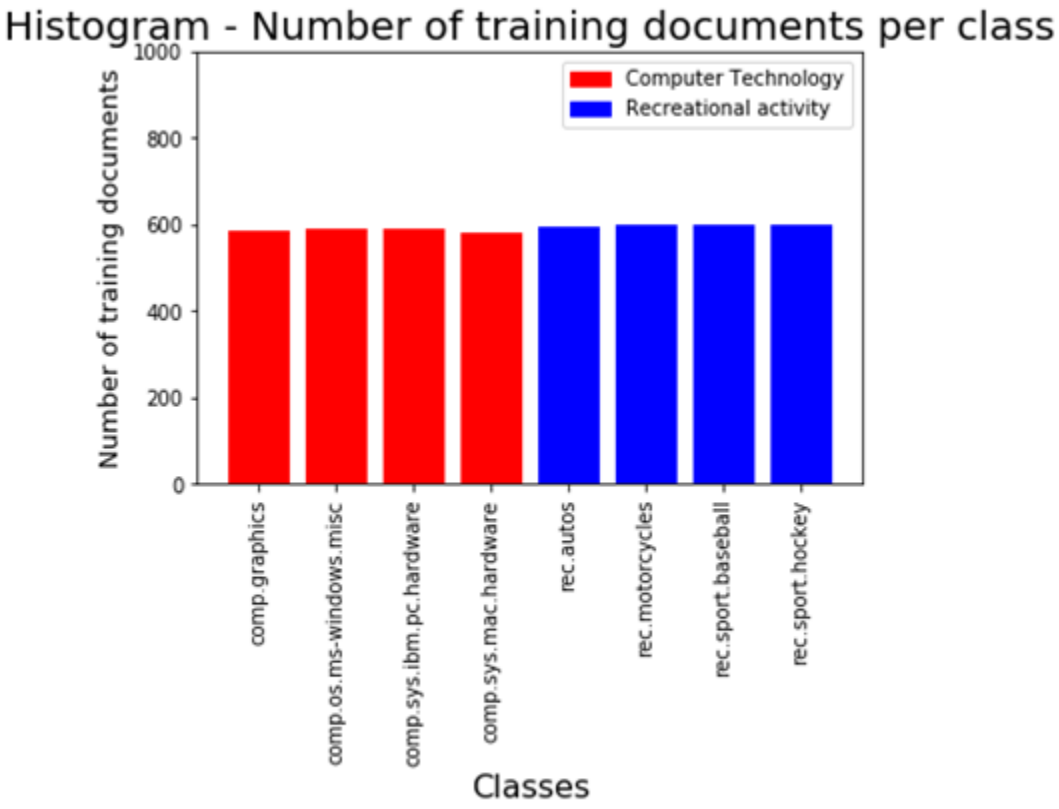
Dataset and problem statement

Dataset: We use the "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. We mainly work with 8 subclasses of two major classes: 'Computer Technology' and 'Recreational activity'.

Computer technology	Recreational activity
comp.graphics	rec.autos
comp.os.ms-windows.misc	rec.motorcycles
comp.sys.ibm.pc.hardware	rec.sport.baseball
comp.sys.mac.hardware	rec.sport.hockey

Part A: Plotting a histogram for number of training documents per class

In this part, we plotted a histogram of number of training documents per class and checked if the number of documents are evenly distributed across all the classes. In case of unbalanced datasets, special handling would have been required.




```
The number of documents in Computer Technology category:  2343
The number of documents in Recreational activity category:  2389
```

The number of documents in each class are as shown below:

	Computer Technology	**Recreational activity**
Training set	2343	2389
Testing set	1560	1590

It is found that both the classes have nearly the same number of documents

Part B: TFxIDF Vector Representation

In this part, we created the TFxIDF vector representation of the documents by tokenizing each document into words. After tokenizing, we remove the commonly occurring stop words, punctuations and use the stemmed version of the words to create the TFxIDF vector representation of the documents. This is done in order to remove words that are too common and too rare to be useful for classification while preserving the essential features.

We implemented a function 'stem_tokenizer()' using the *'nltk'* package to tokenize the documents and remove stopwords and punctuations. We used SnowballStemmer to stem the tokens and regex to clean the tokens. A TFxIDF transformer from sklearn package was used to build the TFxIDF matrix. We used min_df = 2 and min_df = 5 to remove infrequent terms.

The shapes of our TFxIDF matrices for training and testing sets are as follows:

	min_df=2	**min_df=5**
Training set	(4732, 12832)	(4732, 5766)
Testing set	(3150, 12832)	(3150, 5766)

Part C: TFxICF - Class wise TFxIDF

In order to find how significant a word is to a class TFXICF is computed. In this task, we found the 10 most significant terms in each of the following class with respect to TFxICF measure, as shown in the table below:

comp.sys.ibm.pc.hardware	**comp.sys.mac.hardware**	**misc.forsale**	**soc.religion.christian**
drive	appl	use	god
work	problem	sale	jesus
control	know	ship	say
card	work	new	ani

ani	card	offer	believ
problem	mac	includ	church
use	use	pleas	christian
mb	ani	price	think
scsi	drive	sell	peopl
disk	mb	dos	know

Part D: LSI and NMF

In this task, we use Latent Semantic Indexing (LSI), a dimensionality reduction method, to reduce the number of features in our training and testing set. First, we use Truncated Singular Value Decomposition (Truncated SVD) to obtain the TFxIDF matrices for the training and testing data. We use k=50, i.e. each document is mapped to a 50-dimensional vector. Similarly, we also use Non-Negative Matrix Factorization (NMF) for dimensionality reduction and compare the results of these two methods in the following sections.

Part E: Hard-margin and soft-margin SVM Classifier

In this part, we used the linear SVM classifier to classify the documents into the two given classes. We compare hard margin SVM(gamma=1000) and soft margin SVM(gamma=0.001) by reporting the following parameters:

- **ROC curve**: Plots the true positive rate against the false positive rate for different thresholds. IT gives a sensitivity/specificity pair at each threshold
- **Confusion matrix:** Contains information about the actual and predicted values classification done by the classifier
- **Accuracy**: It gives ratio of correctly predicted observations against the total number of observations
- **Precision:** It gives the ratio of correctly predicted positive values against the total number of positive predictions
- **Recall:** It gives the ratio of correctly predicted positive values against the total number observations

The tradeoff parameter is set to 100 for hard margin classifier and 0.001 for soft margin classifier

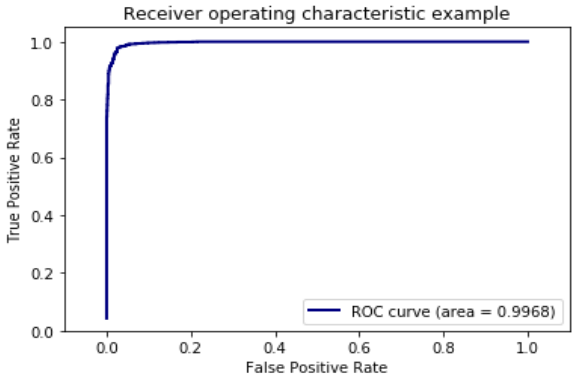
The following table shows the results that we obtained for different values of the min_df parameter:

	LSI				**NMF**	
	HARD MARGIN SVM		**SOFT MARGIN SVM**		**HARD MARGIN SVM**	**SOFT MARGIN SVM**
	min_df=2	*min_df=5*	*min_df=2*	*min_df=5*	*min_df = 2*	*min_df = 2*
Accuracy	97.5238095238	97.4285714286	50.4761904762	50.4761904762	93.8095238095	50.4761904762
Precision	97.5311559394	97.4424733466	25.2380952381	25.2380952381	93.9766679126	25.2380952381
Recall	97.5193517175	97.421988389	50.0	50.0	93.7814465409	50.0
Confusion Matrix	[[1514 46] [32 1558]]	[[1509 51] [30 1560]]	[[0 1560] [0 1590]]	[[0 1560] [0 1590]]	[[1417 143] [52 1538]]	[[0 1560] [0 1590]]

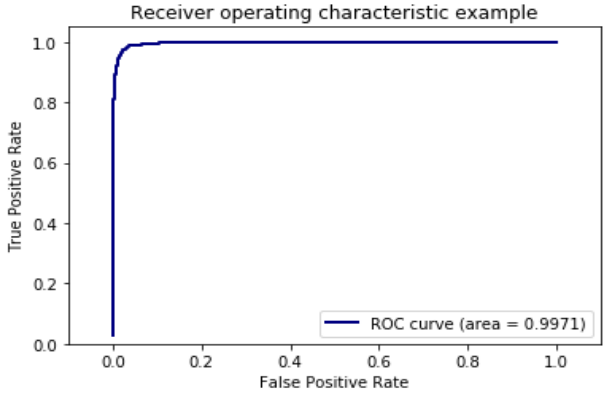
ROC curves:

The ROC curves that we obtained are as follows:

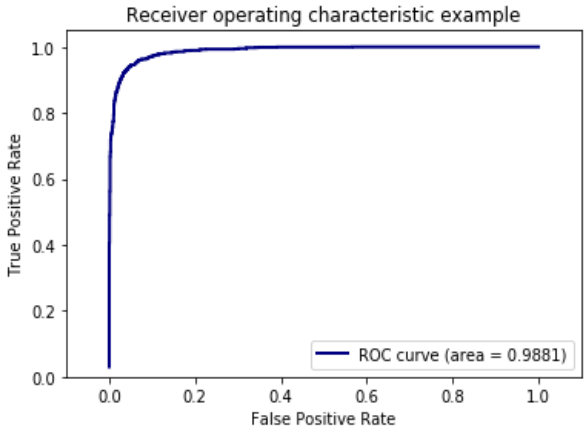
- Hard margin SVM with min_df = 2 using LSI



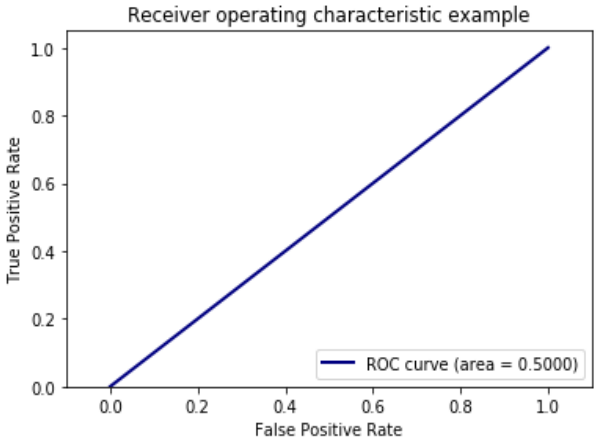
- Hard margin SVM with min_df = 5 using LSI



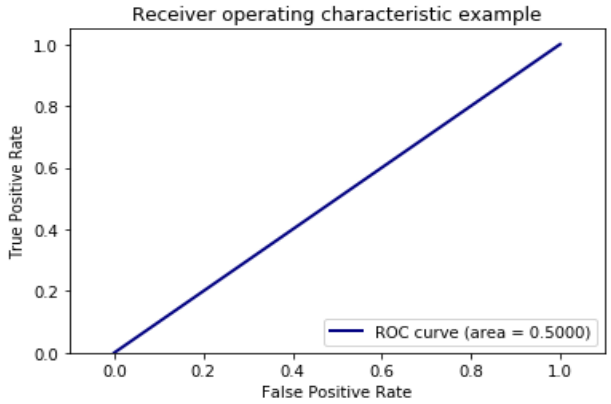
- Hard margin SVM with min_df = 2 using NMF



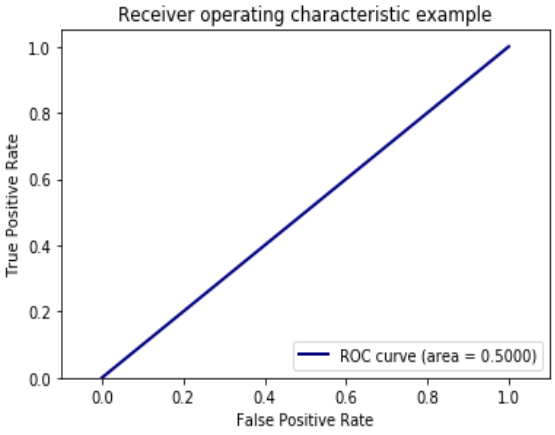
- Soft margin SVM with min_df = 2 using LSI



- Soft margin SVM with min_df = 5 using LSI



- Soft margin SVM with min_df = 2 using NMF



An area of 1 under the ROC curve, represents perfect classification. For LSI, The Hard margin SVM with min_df=5 ROC indicates slightly better classification performance than the same classifier with min_df=2. The performance of NMF is not quite as good as LSI for min_d=2. On the contrary, Soft margin SVM's for all the three classes report the same classification performance with respect to the ROC curve.

Part F: SVM with Cross validation

In this part, we use the SVM classifier with 5-fold cross validation to find the best value of parameter 'gamma' to classify the documents into two classes. Cross validation is usually used to predict the accuracy of the predictive model. We found that the value of k=6 was the best parameter in a range of (10^-3, 10^3) on performing 5-fold cross validation.

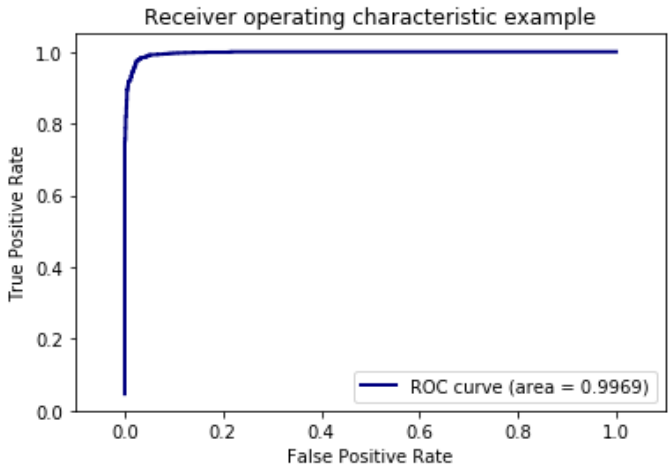
The result that we obtained in the part are as shown in the table below:

	LSI		**NMF**
	min_df=2	*min_df=5*	*min_df=2*
Accuracy	97.4603174603	97.4920634921	96.3174603175
Precision	97.467642343	97.5059985988	96.3297571993
Recall	97.4558538945	97.4854862119	96.3110788582
Confusion matrix	[[1513 47] [33 1557]]	[[1510 50] [29 1561]]	[[1492 68] [48 1542]]

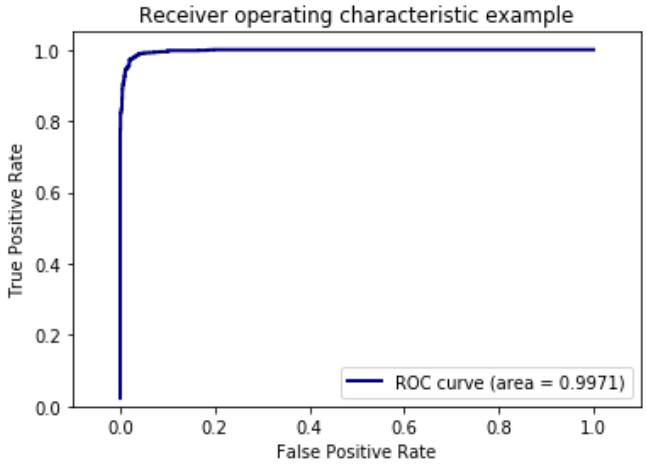
ROC Curves:

The ROC curves that we obtained are as follows:

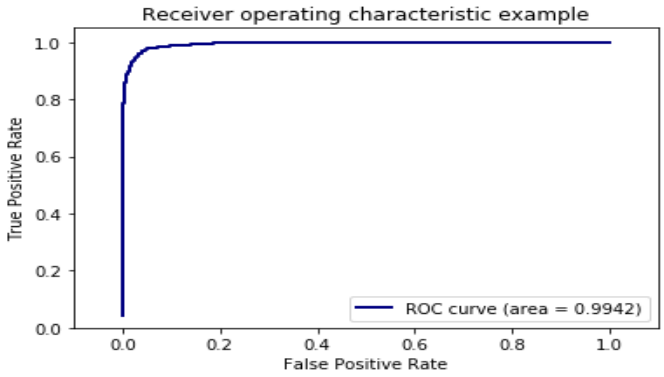
- SVM with cross validation using LSI for min_df=2



- SVM with cross validation using LSI for min_df=5



- SVM with cross validation using NMF for min_df=2



We find that LSI classifier reports similar high performance with 5-fold cross validation as it did for hard margin SVM for both values of min_df. However, the classification performance of NMF has increased with this technique.

PART G: Naive Bayes Classifier

In this task, we use the Naive Bayes Classifier to classify the documents. The Naive Bayes classification algorithm estimates the maximum likelihood probability of a class given a document with feature set, using the Bayes Rule. The accuracy, precision, recall and the confusion matrix that we obtained is reported in the table below:

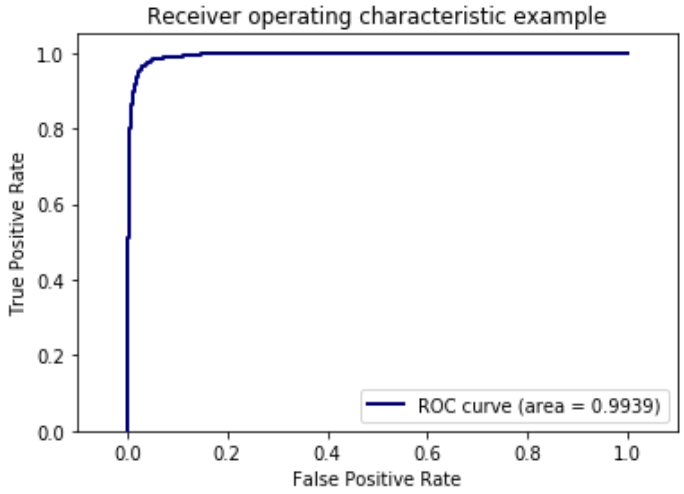
	LSI, min_df=2	LSI, min_df=5	NMF, min_df=2
Accuracy	81.1111111111	89.1428571429	93.4603174603
Precision	86.3844393593	91.006097561	93.8592311301
Recall	80.9294871795	89.0420899855	93.4161828737
Confusion Matrix	[[965 595] [0 1590]]	[[1224 336] [6 1584]]	[[1385 175] [31 1559]]

In the first part (using LSI), the predicted values are mapped into a range of (0,1) using MinMaxScaler to allow them to be used as input to MultinomialNB classifier.

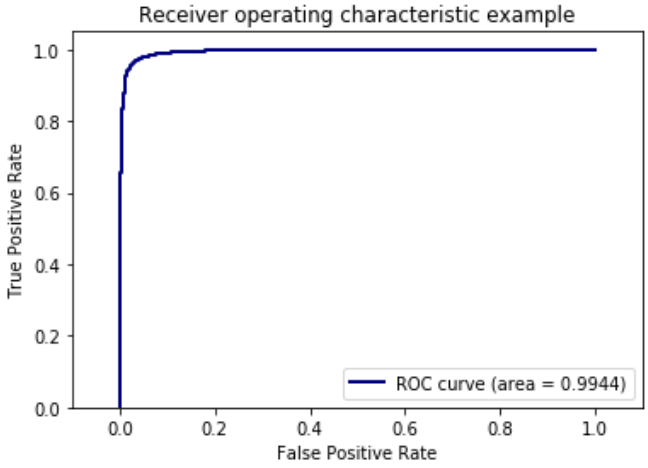
ROC Curves:

The ROC curves that we obtained are as follows:

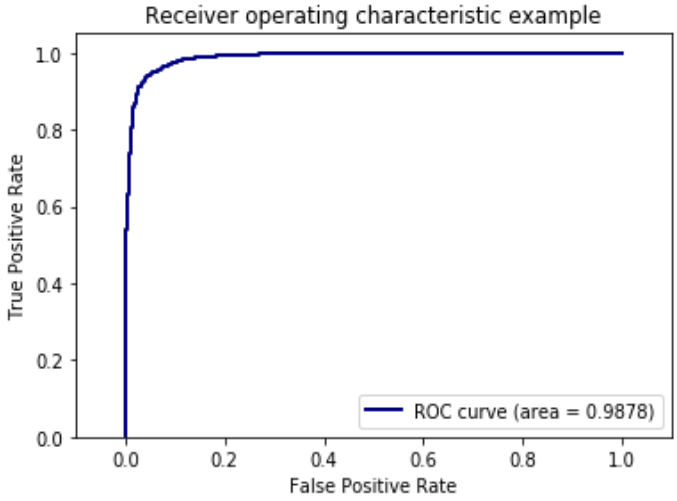
- MultinomialNB Classifier using LSI (min_df=2)



- MultinomialNB Classifier using LSI (min_df=5)



- MultinomialNB Classifier using NMF(min_df=2)



The area under the ROC curve has reduced in both LSI and NMF cases indicating that Naive Bayes Classifier (in our case, Mutinomial NB) places more records under the wrong classes than SVM. Also, LSI, with mapping to positive values still gives better performance than NMF.

Part H: Logistic Regression Classifier

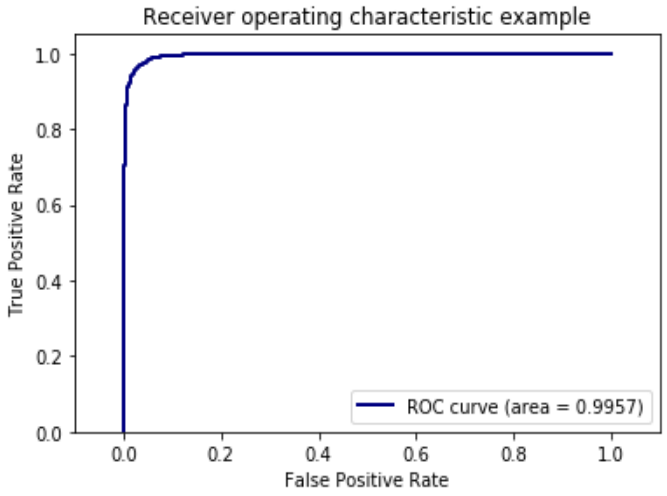
In this task, we use Logistic Regression Classifier to classify the documents and the results that we obtained as shown in the table below:

	LSI		**NMF**
	min_df=2	*min_df=5*	*min_df=2*
Accuracy	96.6031746032	96.8888888889	93.6825396825
Precision	96.6319738242	96.9260025765	93.792645097
Recall	96.5928882438	96.8771165941	93.6598935656
Confusion matrix	[[1490 70] [37 1553]]	[[1492 68] [30 1560]]	[[1424 136] [63 1527]]

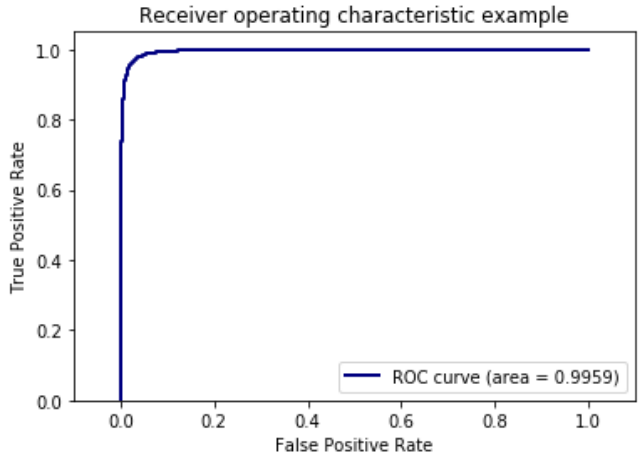
ROC Curves:

The ROC curves that we obtained in this task are as shown below:

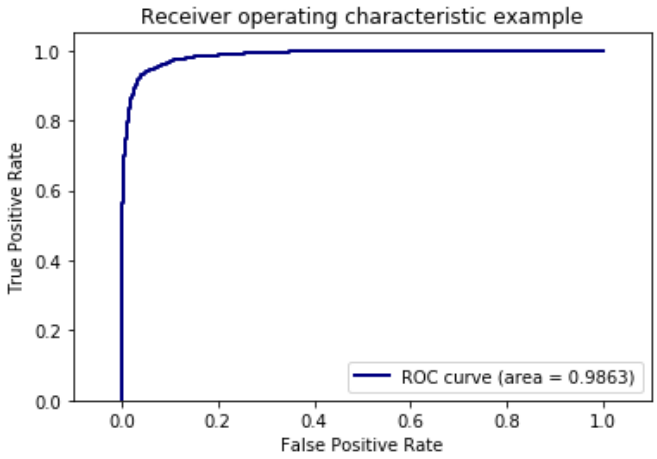
- Logistic Regression using LSI for min_df = 2



- Logistic Regression using LSI for min_df = 5



- Logistic Regression using NMF for min_df = 2



Again, all three cases present better performance for SVM and Naive Bayes over logistic regression as indicated by the progressive reduction in area under the ROC curve from SVM to Logistic Regression.

Part I: Logistic Regression with L1 and L2 regularization

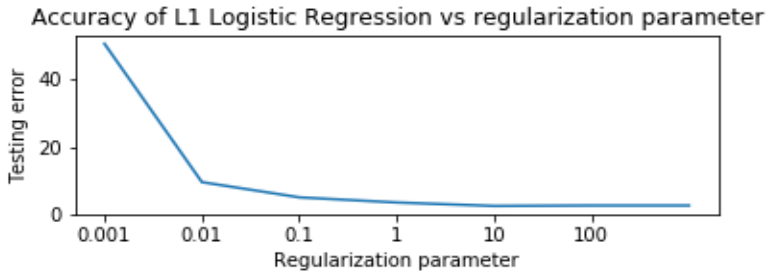
In this task, we used L1 and L2 regularization with the Logistic Regression and observed how the regularization parameter affects the test error.

Our observations for **L1 regularization** are as shown below:

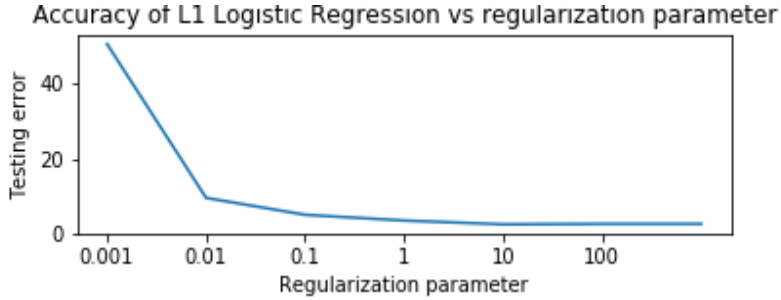
	LSI		**NMF**
	min_df=2	*min_df=5*	*min_df=2*
Accuracy	97.3968253968	97.3968253968	96.5079365079
Precision	97.405817885	97.405817885	96.5183808584
Recall	97.3917513304	97.3917513304	96.5021770682
Confusion matrix	[[1511 49] [33 1557]]	[[1511 49] [33 1557]]	[[1496 64] [46 1544]]

ROC curves

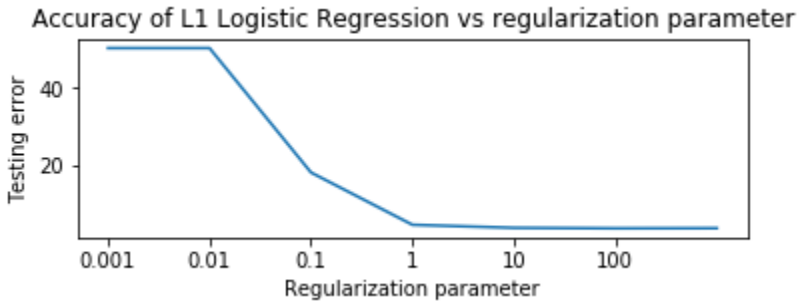
- Logistic Regression with L1 regularization using LSI for min_df = 2



- Logistic Regression with L1 regularization using LSI for min_df = 5



- Logistic Regression with L1 regularization using NMF for min_df = 2

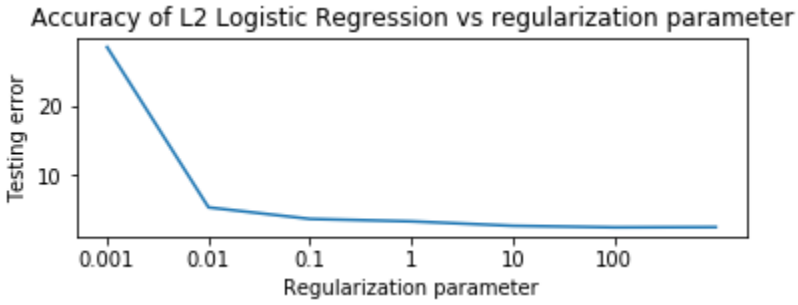


Our observation for **L2 regularization** are as shown in the table below:

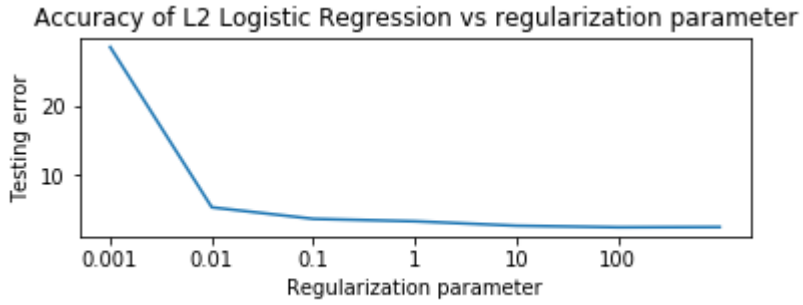
	LSI		**NMF**
	min_df=2	*min_df=5*	*min_df=2*
Accuracy	97.4285714286	97.4285714286	95.9365079365
Precision	97.4367117321	97.4367117321	95.9552535273
Recall	97.4238026125	97.4238026125	95.9282776971
Confusion matrix	[[1512 48] [33 1557]]	[[1512 48] [33 1557]]	[[1483 77] [51 1539]]

ROC curves

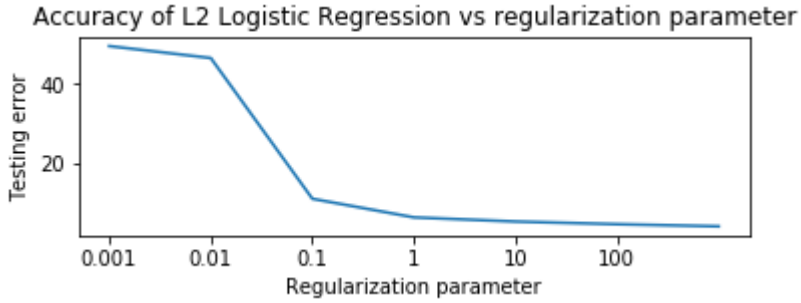
- Logistic Regression with L2 regularization using LSI for min_df = 2



- Logistic Regression with L1 regularization using LSI for min_df = 5



- Logistic Regression with L1 regularization using NMF for min_df = 2



The following table shows the effect of Regularization Parameter on the Test error

Parameter	**L1**	**L2**
K = -3	50.4761904762	49.5238095238
K = -2	50.4761904762	46.5079365079
k=-1	18.0317460317	11.0158730159
k=0	4.38095238095	6.31746031746
k=1	3.5873015873	5.26984126984
k=2	3.46031746032	4.60317460317
k=3	3.49206349206	4.06349206349

The following table shows the effect of Regularization Parameter on the coefficients:

Parameter	L1	L2
K = -3	0.0	-0.000180760118815
K = -2	0.0	-0.00211494708088
k=-1	0.12646196388	-0.0206550607308
k=0	-2.07837217448	-0.135053298217
k=1	-8.41309754096	-0.473942437742
k=2	-13.5355550305	-1.9932175072
k=3	-14.7498258665	-6.37144026103

Observation: L1 and L2 are regularization techniques that can be used to reduce model overfitting. It was observed that, for small values of the regularization parameter, excessive regularization took place, which leads to a high test error. However, as we increase the value of the regularization parameter, the test error decreases.

We would generally use L1 regularization when we want a robust solution and also have the computational power and are willing to tolerate multiple stable solutions. We would opt for L2 regularization if the dataset is large or if a single unstable solution would work. Also, we observed that the fitted hyperplane shifts away from the origin before coming closer again as the regularization parameter increases.

Part I (2): Multiclass SVM

In this task, we perform Naive Bayes classification and Multiclass SVM classification with One Vs One, which trains individual classes against one and other and One Vs Rest, which trains a single class against the rest, methods and the results obtained are as shown below:

One VS One Method:

	Naive Bayes		**Multiclass SVM**	
	LSI, min_df=2	*NMF, min_df=2*	*LSI, min_df = 2*	*NMF, min_df=2*
Accuracy	71.9488817891	79.9361022364	87.2843450479	77.7635782748
Precision	71.99458959	80.5062738714	87.4082815513	79.5179999171
Recall	71.7493724569	79.7832433479	87.2249391551	77.6733612762
Confusion Matrix	[[247 43 81 21] [82 184 104 15] [45 33 305 7] [0 1 7 390]]	[[315 30 42 5] [83 236 59 7] [49 12 310 19] [2 0 6 390]]	[[326 44 22 0] [47 315 23 0] [31 17 341 1] [8 1 5 384]]	[[246 48 98 0] [44 260 77 4] [38 10 342 0] [1 1 27 369]]

One VS Rest Method:

	Naive Bayes		**Multiclass SVM**	
	LSI, min_df=2	*NMF, min_df=2*	*LSI, min_df = 2*	*NMF, min_df=2*
Accuracy	72.9073482428	80.2555910543	88.3706070288	81.214057508
Precision	73.5184121207	80.3508301433	88.2595894428	80.8535008235
Recall	72.7136559899	80.113983737	88.3112229061	81.1145602423
Confusion Matrix	[[246 38 95 13] [74 190 109 12] [41 31 313 5] [0 1 5 392]]	[[305 39 41 7] [72 249 56 8] [47 14 309 20] [1 0 4 393]]	[[315 46 27 4] [33 322 28 2] [23 13 352 2] [2 0 2 394]]	[[268 66 28 30] [46 286 35 18] [34 16 321 19] [1 0 1 396]]