EE 219 Project 1 Report: Classification Analysis of Texture Data

Jessica Fu (805034901) & Jasmine Moreno (705035581)

Winter 2018

Introduction

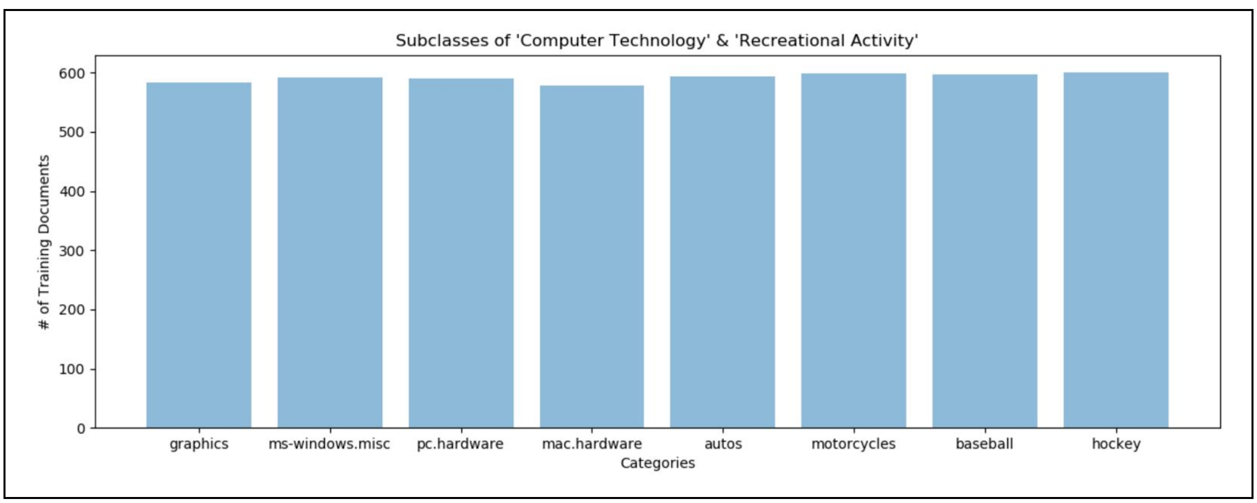
In this project, we tasked to identify a category from a predefined set. We are given the training data, which will be used in order to classify the data into the correct category. Classification is essential when it comes to data analysis, especially big data. This report presents different methods for classifying data from documents with its results and graph.

Dataset and Problem Statement

Part A

In this section, we are introduced to the “20 Newsgroups” dataset on Python. For this project we are only using eight subclasses of computer technology and recreational activity. In this section, we plotted a histogram of the training documents per class to check if they are evenly distributed. The results can we seen below.

Results for Part A



Histogram of training documents per class

As we can see from the histogram above, we can see that the number of training documents are evenly distributed. In general, if they are not evenly distributed, we would have to assign more weight to the classes with significantly lower quantity or down-sample the classes with more.

Modeling Text Data and Feature Extraction

Part B
The goal is to first tokenize each document into words and then create a Term Frequency-Inverse Document Frequency (TFxIDF) vector representation. In order to classify text, we must filter certain words and characters. Words with the same stem word must also be categorized as the same word. This will decrease the number of words/size of the data set.

Results for Part B
We used a Lemma tonkenizer to remove stop words, symbols, punctuation, and non-ascii characters. It also converts all letters to lower case and organizes the same-stem words. When min_df = 2, TF-IDF representation results in 25023 terms. It removed terms that only appear in one document. When min_df = 5, TF-IDF representation results in 10294 terms. It removed terms that appeared in less than 5 documents, reducing the number of terms.

Part C
We will use TFxICF to quantify the importance of a word to a class. The goal is to find the 10 most significant words to the classes: comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian.

Results for Part C
Similar to Part B, we tokenize each document, remove stop words and stemmed version of words. With min_df = 2, it ignores words appearing in less than two documents. Next we use the given TFxICF function to find words of most significance. Below is a table of the 10 most significant terms in each specified category.

Comp.sys.ibm.pc.hardware	Comp.sys.mac.hardware	misc.forsale	soc.religion.christian
eisa motherboard synchronous mfm ati bios conner jumbo scsi vram	khz motherboard cabling laserwriter adaptor hcf umcc ravi vram scsi	motherboard vhs packaging adaptor vcr bike adventure sunysb amp panasonic	ceremony spiritual geneva atheist hebrew christianity prophet prophecy scripture sinner

Table of top 10 most significant

Feature Selection

After the previous steps, the TFxIDF vectors (representation vectors) have high dimensions which cause poor computational performance.

Part D
To lower the dimensional space, we use Latent Semantic Indexing (LSI) and Non-Negative Matrix Factorization (NMF). We chose to map each document to a 50-dimensional vector by making k = 50.

Learning Algorithms

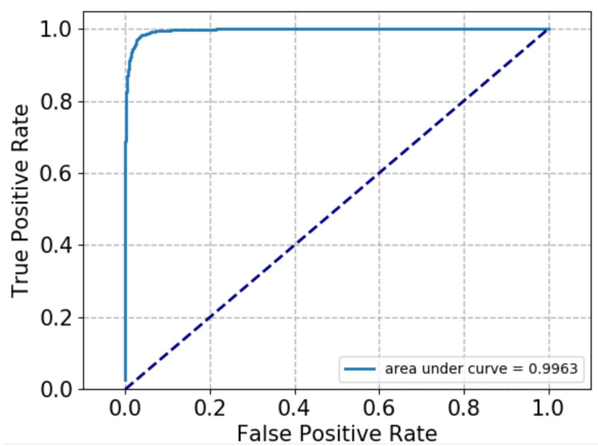
In these sections, we explore the different classifiers we can use to classify the documents into two categories “Computer Technology” vs “Recreational Activity.” The subclasses can be seen in the table below. For each classifier, we evaluate the accuracy, recall, and precision. Then we plot its receiver operating characteristic (ROC) curve and find its confusion matrix.

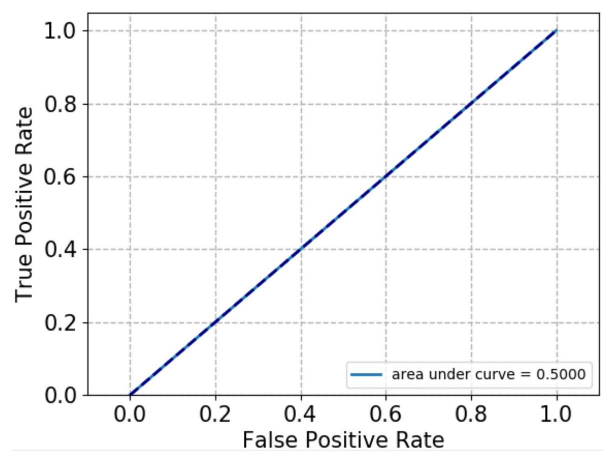
Computer Technology	**Recreational Activity**
comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey

Subclasses of classes “Computer Technology” and “Recreational Activity”

Part E
In order to classify documents into “Computer Technology” and “Recreational Activity” groups, we use a hard margin SVM classifier (SVC). A hard margin means to set γ >> 1 and misclassification is highly penalized. We also compared the results to a soft margin (γ << 1), which is very lenient if there are a few misclassifications.

Results for Part E (LSI, min_df = 2)
The results from using the LSI SVM classifier are shown below. min_df is set to equal to 2, meaning it removed terms appearing in less than 2 documents. Results using a hard margin differ significantly from results using a hard margin. A hard margin results in high percentage of accuracy, precision, and recall. The confusion matrix shows high true-positive and true-negative values and low false-positive and false-negative values. The soft margin, which is more lenient with inaccurate classifying, shows a much lower accuracy, precision, and recall. The confusion matrix shows all documents were classified as recreational activity.





Hard Margin	Soft Margin

LSI SVM receiver operating characteristic (ROC) graphs

Statistic	**Value**
Accuracy	97.0793650794
Precision	97.1166947021
Recall	97.0676100629

Testing LSI SVM Hard Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1495	65
True Recreational Activity	27	1563

LSI SVM Hard Margin Confusion Matrix

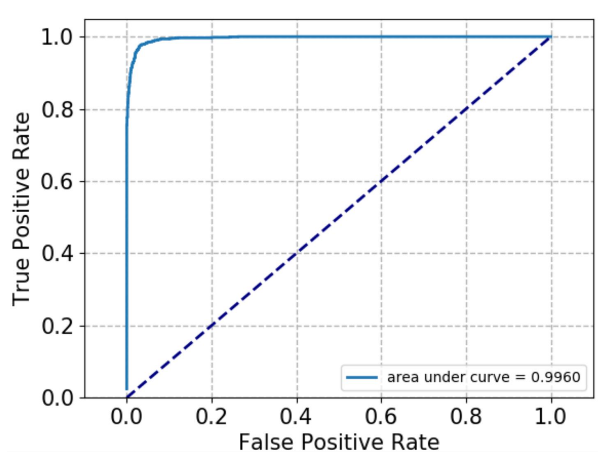
Statistic	**Value**
Accuracy	50.4761904762
Precision	25.2380952381
Recall	50.0

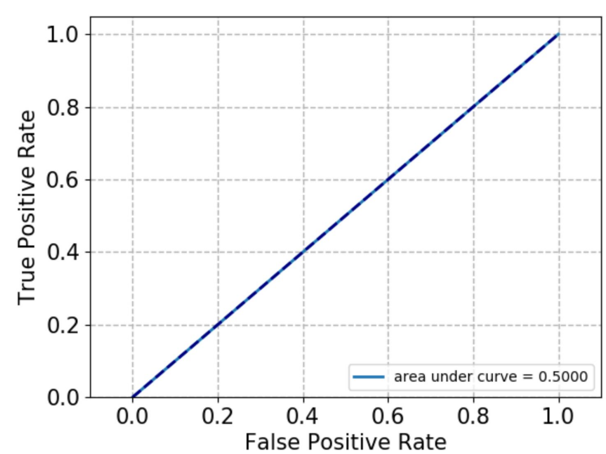
Testing LSI SVM Soft Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	0	1560
True Recreational Activity	0	1590

LSI SVM Soft Margin Confusion Matrix

Results for Part E (LSI, min_df = 5)
Now we have changed min_df to equal 5, meaning it removed terms appearing in less than 5 documents. The results are very similar to the the results above.





Hard Margin	Soft Margin

LSI SVM receiver operating characteristic (ROC) graphs

Statistic	Value
Accuracy	96.8571428571
Precision	96.8892812106
Recall	96.8462747944

Testing LSI SVM Hard Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1493	67
True Recreational Activity	32	1558

LSI SVM Hard Margin Confusion Matrix

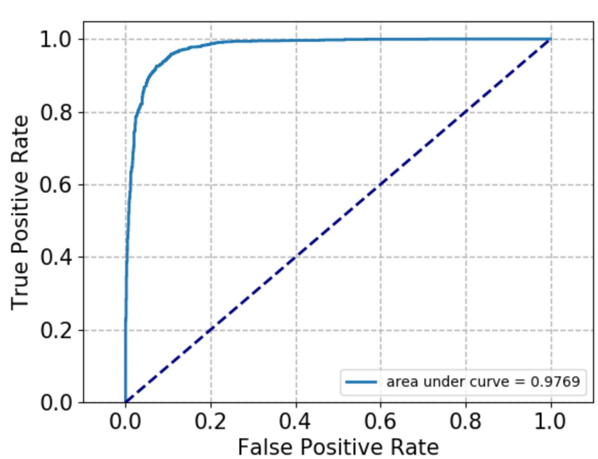
Statistic	Value
Accuracy	50.4761904762
Precision	25.2380952381
Recall	50.0

Testing LSI SVM Soft Margin

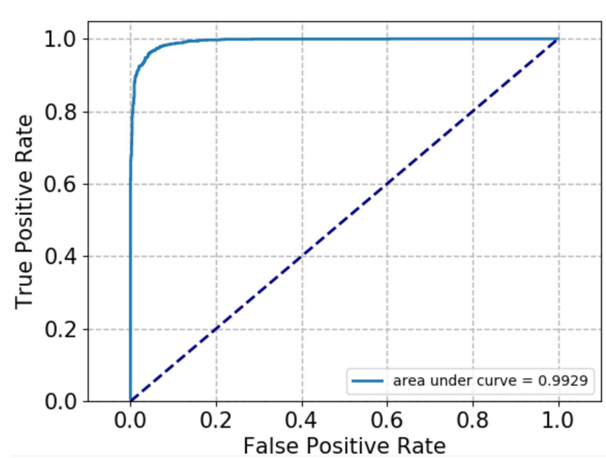
	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	0	1560
True Recreational Activity	0	1590

LSI SVM Soft Margin Confusion Matrix

Results for Part E (NMF, min_df = 2)
Here, instead of LSI, we use Non-Negative Matrix Factorization (NMF). NMF also reduces dimensionality, but only accepts positive entries. Again, the results are very similar to those above. Hard margins have a higher accuracy, precision, and recall and soft margins give much lower results.



Hard Margin



Soft Margin

NMF SVM receiver operating characteristic (ROC) graphs

Statistic	Value
Accuracy	96.8571428571
Precision	96.8892812106
Recall	96.8462747944

Testing NMF SVM Hard Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1493	67
True Recreational Activity	32	1558

NMF SVM Hard Margin Confusion Matrix

Statistic	Value
Accuracy	50.4761904762
Precision	25.2380952381
Recall	50.0

Testing NMF SVM Soft Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	0	1560
True Recreational Activity	0	1590

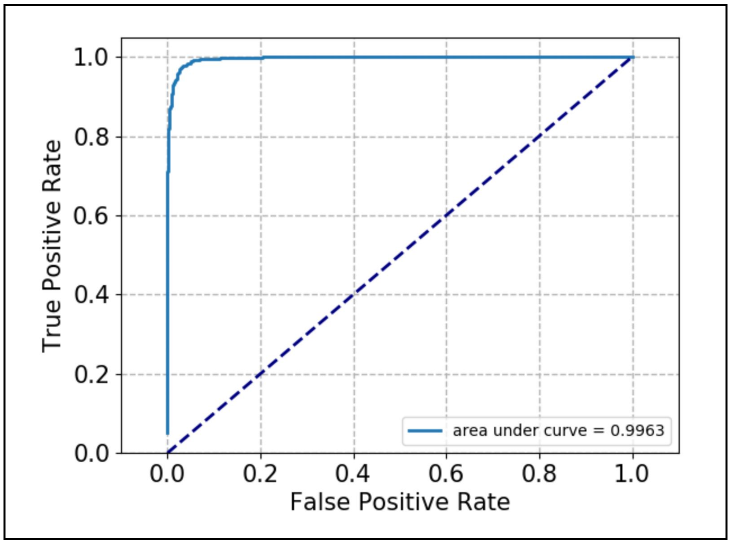
NMF SVM Soft Margin Confusion Matrix

Part F

In this section, we use a 5-fold cross-validation and the SVM method is used to find the best value of the parameter γ in the range $\{10^k \mid -3 \leq k \leq 3, k \in Z\}$.

Results for Part F (LSI, min_df = 2)

We result in the best score, γ = 0.9759, when k = 2. The results are positive, as shown by the high accuracy, precision, and recall values.



LSI SVM receiver operating characteristic (ROC) graph

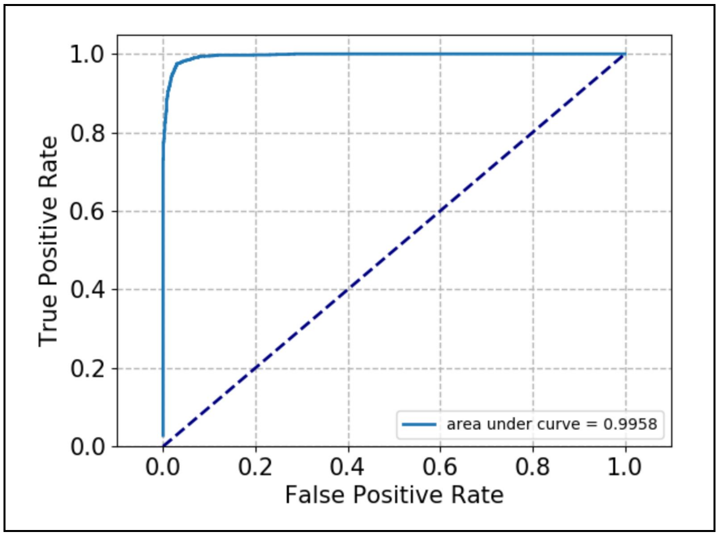
Statistic	Value
Accuracy	96.9206349206
Precision	96.9528373266
Recall	96.9097726173

Testing LSI SVM Hard Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1494	66
True Recreational Activity	31	1559

LSI SVM Hard Margin Confusion Matrix

Results for Part F (LSI, min_df = 5)
When min_df = 5, we result in the best score, γ = 0.9763, when k = 2. Again, the results are positive as shown in the data below.



LSI SVM receiver operating characteristic (ROC) graph

Statistic	**Value**
Accuracy	96.9523809524
Precision	96.983041081
Recall	96.9418238994

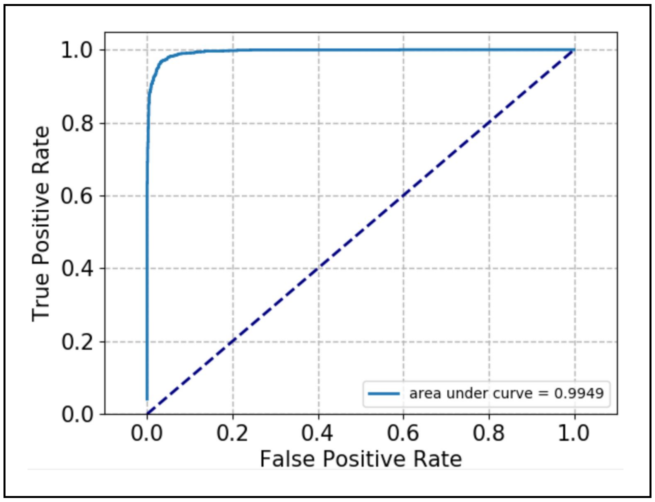
Testing LSI SVM Hard Margin

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1495	65
True Recreational Activity	31	1559

LSI SVM Hard Margin Confusion Matrix

Results for Part F (NMF, min_df = 2)

Now using the NMF matrix decomposition method, we find the best score, γ = 0.9699, when k = 3. Again, the results are positive as shown in the data below.



NMF SVM receiver operating characteristic (ROC) graph

Statistic	**Value**
Accuracy	96.4444444444
Precision	96.4746124644
Recall	96.4338413159

Testing NMF SVM Hard Margin

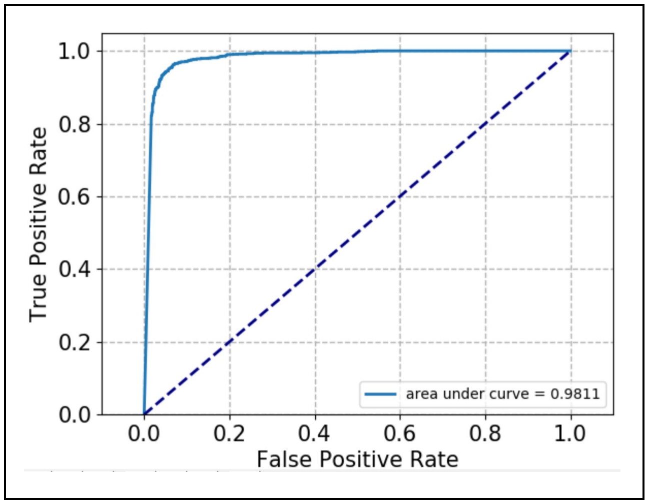
	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1487	73
True Recreational Activity	31	1551

NMF SVM Hard Margin Confusion Matrix

Part G

In this section, we use the Naive Bayes as our classifier. This classifier uses Bayes rule to maximum the likelihood probability of a class. In this section, we only use NMF and not LSI. The LSI data contains negative values and cannot be used to train in the Naive Bayes Classifier.

Results for Part G (NMF, min_df = 2)



Naive Bayes Classification ROC Graph

Statistic	**Value**
Accuracy	94.6349206349
Precision	94.6744667686
Recall	94.6220367683

Naive Bayes Classification

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1455	105
True Recreational Activity	64	1526

Naive Bayes Classification Confusion Matrix

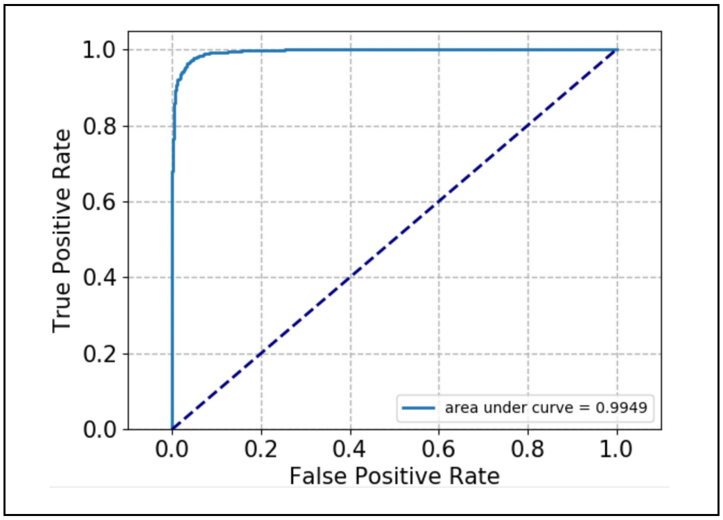
Observations: We only used the NMF in this section because NMF only takes in positive values. We can see that the Naive Bayes classifies really well. Its accuracy is above 90% therefore great for these test data.

Part H

Now we are using the logistic regression classifier to perform the same task as in Part G.

Results for Part H (LSI, min_df = 2)

The results are positive, as shown by the high accuracy, precision, and recall values.
The confusion matrix shows high true-positive and true-negative values and low false-positive and false-negative values.



Logistic Regression Classification ROC Graph

Statistic	Value
Accuracy	96.3492063492
Precision	96.398401308
Recall	96.3352685051

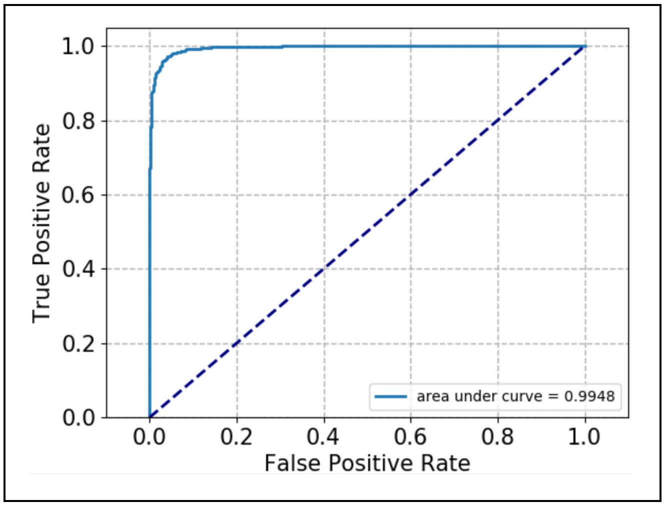
Logistic Regression Classification

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1480	80
True Recreational Activity	35	1555

Logistic Regression Classification Confusion Matrix

Results for Part H (LSI, min_df = 5)

Again, the results are positive as shown in the data below.



Logistic Regression Classification ROC Graph

Statistic	**Value**
Accuracy	96.4444444444
Precision	96.4918470626
Recall	96.4308176101

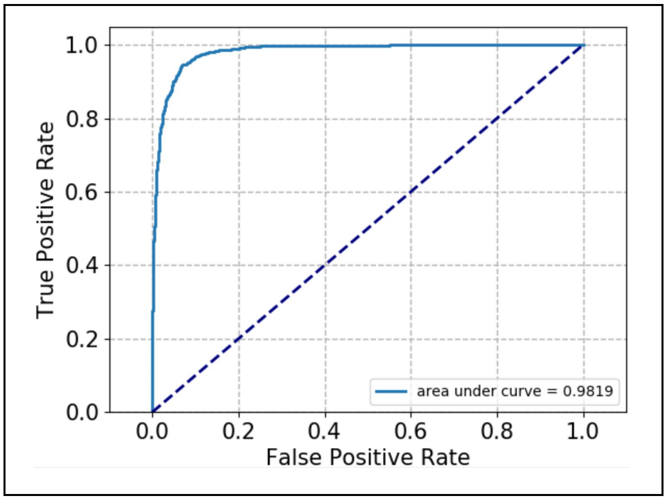
Logistic Regression Classification

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1482	78
True Recreational Activity	34	1556

Logistic Regression Classification Confusion Matrix

Results for Part H (NMF, min_df = 2)

Again, the results are positive as shown in the data below.



Logistic Regression Classification ROC Graph

Statistic	Value
Accuracy	93.4285714286
Precision	93.6536116324
Recall	93.3956216739

Logistic Regression Classification

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1403	157
True Recreational Activity	50	1540

Logistic Regression Classification Confusion Matrix

Part I

In this part, we are repeating the same tasks as in part h. However, this time we are going to try using the L_1 and L_2 norm regularization. We sweep through different regulation coefficients that range from (10^{-3} to 10^3). Then we compare all the accuracy for each coefficient and plot the ROC for the most accurate coefficient for each L_1 and L_2.

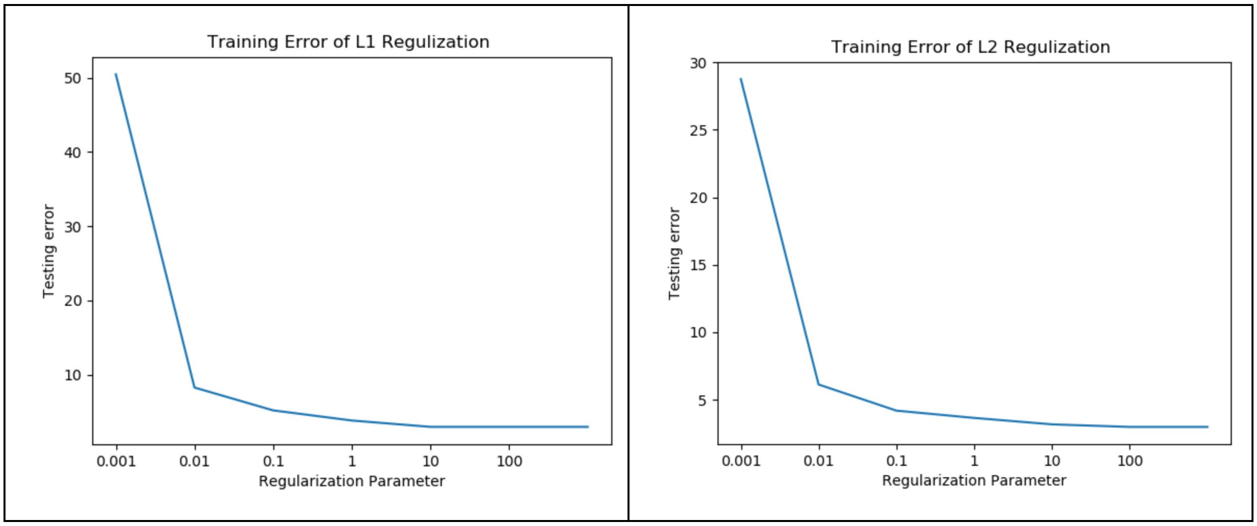
Results for Part I (LSI, min_df = 2)

Parameter k in 10^k	L1	L2
-3	50.4762	28.7619
-2	8.2539	6.1269
-1	5.1746	4.1904
0	3.8095	3.6507
1	2.9523	3.1746
2	2.9523	2.9841
3	2.9523	2.9841

Table for Training Errors

Parameter k in 10^k	L1	L2
-3	0	-0.0013
-2	-0.1155	-0.0121
-1	-0.5875	-0.0617
0	-0.6748	-0.1252
1	-0.4737	-0.1843
2	-0.4057	-0.2875
3	-0.4031	-0.3717

Table for Coefficient in Hyperplane



Training Error vs Regularization Parameter Graphs

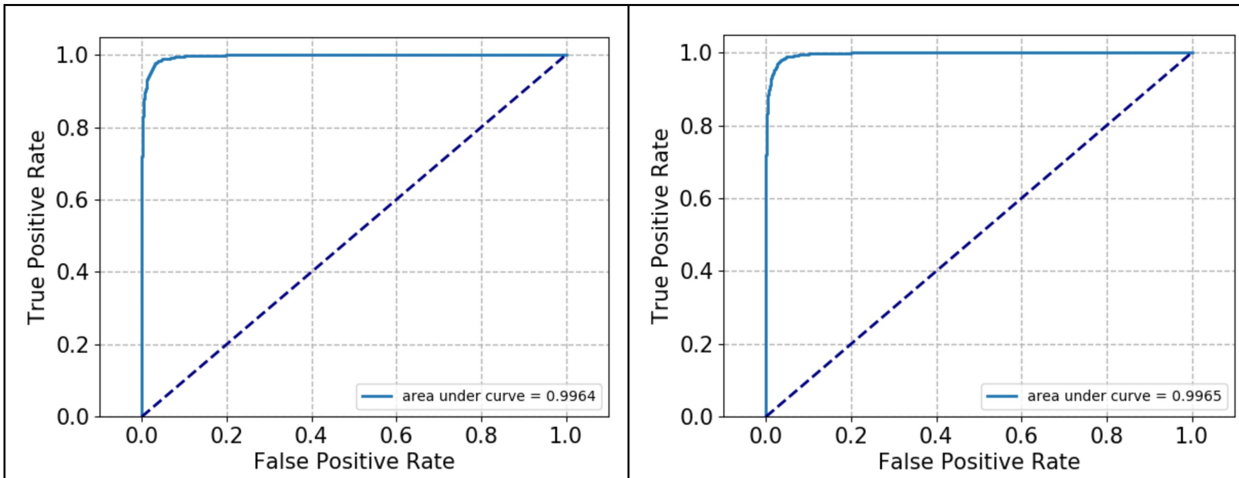
Statistic	**Value (L_1)**	**Value (L_2)**
Accuracy	97.0476190476	97.0158730159
Precision	97.0902191695	97.0602766798
Recall	97.0349540397	97.0029027576

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1493	67
True Recreational Activity	26	1564

L_1 Confusion Matrix

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1492	68
True Recreational Activity	26	1564

L_2 Confusion Matrix



L_1 ROC | *L_2 ROC*

<u>Observations:</u> How does the regularization parameter affect the test error? How are the coefficients of the fitted hyperplane affected? Why might one be interested in each type of Regularization?

From these results, we can see that excessive regularization takes place when the regularization parameter is really small. L_2 errors are less than L_1 for smaller parameters, and L_1 errors are less for larger parameters. Increasing both L_2 and L_1 parameters both seem to stabilize around a number.

We can see that the hyperplane shifts away from the origin, but then comes close to it again for L_1. For L_2 it just shifts away.

We can see from the statistical table, that L_1 is a little more accurate than L_2. We will typically use L_1 when we need a more concrete or robust solution. However, L_1 solution is not stable and maybe have more than one solution. We would use L_2 if we do not need a robust solution. Unlike L_1, L_2 has a stable unique solution.

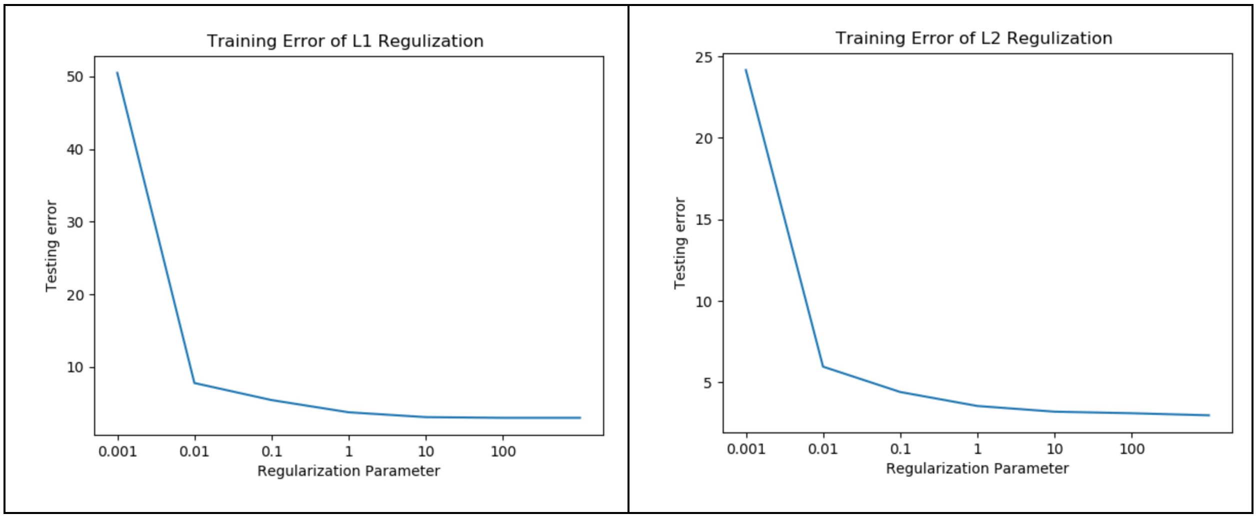
Results for Part I (LSI, min_df = 5)

Parameter k in 10^k	L1	L2
-3	50.4761	24.15873
-2	7.7777	5.9682
-1	5.4285	4.4126
0	3.7460	3.5555
1	3.0793	3.2063
2	2.9841	3.1111
3	2.9841	2.9841

Table for Training Errors

Parameter k in 10^k	L1	L2
-3	0.0	-0.0028
-2	-0.1349	-0.0268
-1	-0.6023	-0.1673
0	-1.2431	-0.5230
1	-2.7853	-1.1386
2	-3.5044	-2.2241
3	-3.6100	-3.2822

Table for Coefficient in Hyperplane



Training Error vs Regularization Parameter Graphs

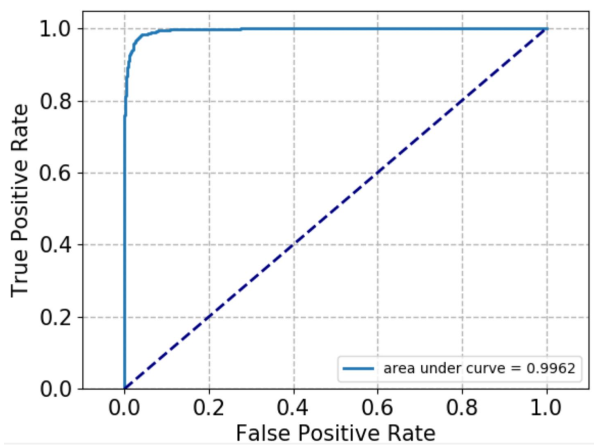
Statistic	**Value (L_1)**	**Value (L_2)**
Accuracy	97.0158730159	97.0158730159
Precision	97.0531306602	97.0531306602
Recall	97.00411224	97.00411224

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1494	66
True Recreational Activity	28	1562

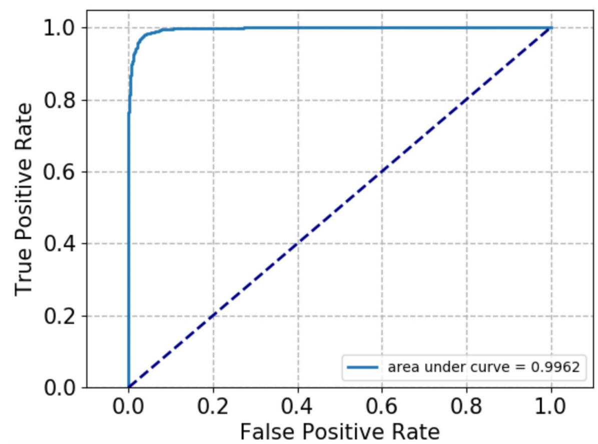
L_1 Confusion Matrix

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1494	66
True Recreational Activity	28	1562

L_2 Confusion Matrix



L_1 *ROC*



L_2 *ROC*

<u>Observations:</u> How does the regularization parameter affect the test error? How are the coefficients of the fitted hyperplane affected? Why might one be interested in each type of Regularization?

Just like the previous section, we can see that excessive regularization takes place when the regularization parameter is really small. L_2 errors are less than L_1 for smaller parameters, and L_1 errors are less for larger parameters. Increasing both L_2 and L_1 parameters both seem to stabilize around a number.

We can see that the hyperplane shifts away from the origin as the parameters increases for both L_1 and L_2.

Same Answer: We can see from the statistical table, that L_1 is a little more accurate than L_2. We will typically use L_1 when we need a more concrete or robust solution. However, L_1 solution is not stable and maybe have more than one solution. We would use L_2 if we do not need a robust solution. Unlike L_1, L_2 has a stable unique solution.

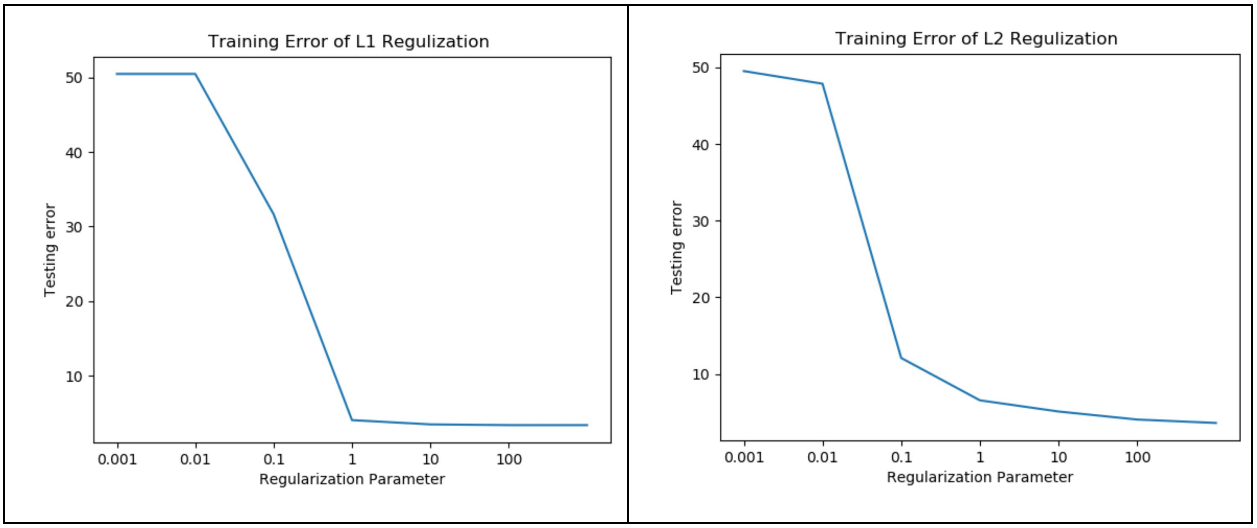
Results for Part I (NMF, min_df = 2)

Parameter k in 10^k	L1	L2
-3	50.4761	49.5238
-2	50.4761	47.8730
-1	31.6507	12.0952
0	4.0317	6.5714
1	3.4603	5.1111
2	3.3650	4.0634
3	3.3650	3.61904

Table for Training Errors

Parameter k in 10^k	L1	L2
-3	0.0	-0.0003
-2	0.0	-0.0040
-1	0.0825	-0.0389
0	-2.7815	-0.2759
1	-5.6595	-0.9756
2	-12.6432	-2.3789
3	-16.2790	-5.4974

Table for Coefficient in Hyperplane



Training Error vs Regularization Parameter Graphs

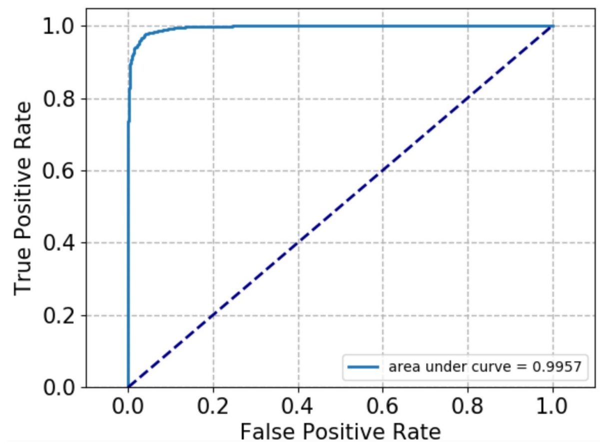
Statistic	**Value (L_1)**	**Value (L_2)**
Accuracy	96.6349206349	96.380952381
Precision	96.6752098805	96.4209315467
Recall	96.6225205612	96.3685292695

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1487	73
True Recreational Activity	33	1557

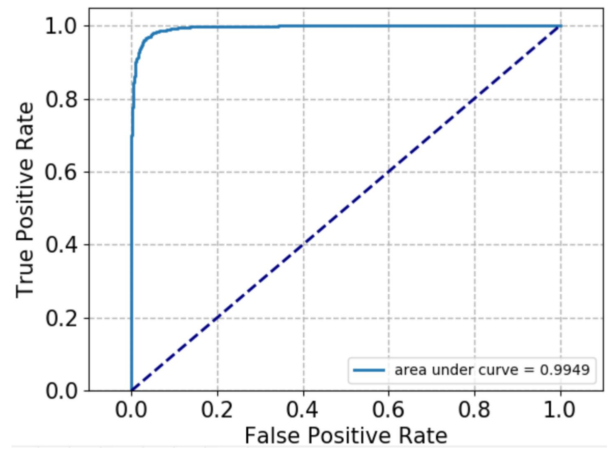
L_1 Confusion Matrix

	Predicted Computer Technology	Predicted Recreational Activity
True Computer Technology	1483	77
True Recreational Activity	37	1553

L_2 Confusion Matrix



L_1 ROC



L_2 ROC

<u>Observations:</u> How does the regularization parameter affect the test error? How are the coefficients of the fitted hyperplane affected? Why might one be interested in each type of Regularization?

From these results, we can see that excessive regularization takes place when the regularization parameter is really small. L_2 errors are less than L_1 for smaller parameters, and L_1 errors are less for larger parameters. Increasing both L_2 and L_1 parameters both seem to stabilize around a number. We also see that it for both L_1 and L_2 the error increases as the parameter decreases.

We can see that the hyperplane shifts away from the origin, but then comes close to it again for L_1. For L_2 it just shifts away.

Same answer: We can see from the statistical table, that L_1 is a little more acuarate than L_2. We will typically use L_1 when we need a more concrete or robust solution. However, L_1 solution is not stable and maybe have more than one solution. We would use L_2 if we do not need a robust solution. Unlike L_1, L_2 has a stable unique solution.

Multiclass Classification

In this section, we will be working with multiclass classification. Two classifiers that perform multiclass classification are the Naive Bayes and SVM, which we will explore in this section. We will use the methods One Vs One and One Vs Rest when classifying.
Given a document, one vs one method basically picks the class depending on majority vote. In case of a tie, the class that has the highest total classification confidence level is picked.
Given a document, one vs rest the class is fitted depending on the other classes. Therefore, unbalanced documents should be handle to use these methods.

Results for Part I (NMF, min_df = 2)

Statistic	**Value (L_1)**
Accuracy	75.9744408946
Precision	75.7249539956
Recall	75.8181310793

One vs One Classification using Naive Bayes

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	274	59	56	3
True mac_hardware	88	231	62	4
True misc_forsale	49	28	294	19
True religion_christian	2	3	3	390

One vs One Classification using Naive Bayes Confusion Matrix

Statistic	**Value (L_1)**
Accuracy	77.4440894569
Precision	77.2202152033
Recall	77.2975245023

One vs Rest Classification using Naive Bayes

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	266	66	57	3
True mac_hardware	75	237	66	7
True misc_forsale	46	20	317	7
True religion_christian	1	2	3	392

One vs Rest Classification using Naive Bayes Confusion Matrix

Observation: We can see that the One vs Rest Classification using Naive Bayes resulted in better accuracy than the One vs One Classification. Both of these classes were not the best results.

Results for Part I (NMF, min_df = 2, SVM)

Statistic	**Value (L_1)**
Accuracy	59.0415335463
Precision	74.1851054268
Recall	58.9135659447

One vs One Classification using SVM

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	117	33	242	0
True mac_hardware	11	137	237	0
True misc_forsale	16	9	365	0
True religion_christian	0	0	93	305

One vs One Classification using SVM

Statistic	**Value (L_1)**
Accuracy	79.4888178914
Precision	79.4112395033
Recall	79.3720267827

One vs Rest Classification using SVM

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	117	33	242	0
True mac_hardware	11	137	237	0
True misc_forsale	16	9	365	0
True religion_christian	0	0	93	305

One vs Rest Classification using SVM

<u>Observation: </u>Just like the Naive Bayes Classification, one vs rest yields better accuracy. In these results we can see there is a better advantage in using the one vs rest classification using SVM than the one vs one.

Results for Part I (LSI, min_df = 2, SVM)

Statistic	**Value (L_1)**
Accuracy	88.3067092652
Precision	88.568239572
Recall	88.2502631372

One vs One Classification using SVM

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	336	39	17	0
True mac_hardware	54	317	14	0
True misc_forsale	28	15	346	1
True religion_christian	10	1	4	383

One vs One Classification using SVM

Statistic	**Value (L_1)**
Accuracy	87.5399361022
Precision	87.4052166605
Recall	87.4743019902

One vs Rest Classification using SVM

	Predicted pc_hardware	Predicted mac_hardware	Predicted misc_forsale	Predicted religion_christian
True pc_hardware	307	52	31	2
True mac_hardware	42	313	27	3
True misc_forsale	19	12	357	2
True religion_christian	3	1	1	393

One vs Rest Classification using SVM

Observation: Unlike the NMF, we can see that one vs one classification yields a slightly better result than the one vs rest classification when using LSI. In addition, we can see that the using LSI with the SVM one vs one classification yielded the best results compared to the other methods and matrix decomposition technique (NMF).