EE 219 Project 2 - Report

February 15

Isha Verma	Heenal Doshi	Pranav Thulasiram Bhat
ishaverma@cs.ucla.edu	heenald@cs.ucla.edu	pranavtbhat@cs.ucla.edu
404761131	004758927	704741684

1 Introduction

In this project, we analysed the 20-NewsGroups dataset, which had some 20,000 documents, partitioned evenly across 20 distinct categories. The documents were mostly classified using several classifiers such as Naive Bayesian, SVMs and Logistic Regressors. The report below presents the techniques we used, as well as the plots and results we obtained.

2 Dataset

We obtained the dataset using sklearn's fetch_20newsgroups module.

The module had the following data items:

- target_names: The names of the 20 categories to which the documents belong.
- target: An integer mapping of the document categories.
- data: An array of document text.

Only a subset of the newgroup categories were considered here:

2.1 Computer Technology

- comp.graphics

- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- comp.sys.mac.hardware

2.2 Recreational Activity

- rec.autos
- rec.motorcycles
- rec.sport.baseball
- rec.sport.hockey

3 Question a

For any classification problem, it is essential for the input training dataset to be balanced, there are equal number of documents in each category. We plotted a histogram of the 8 selected categories.

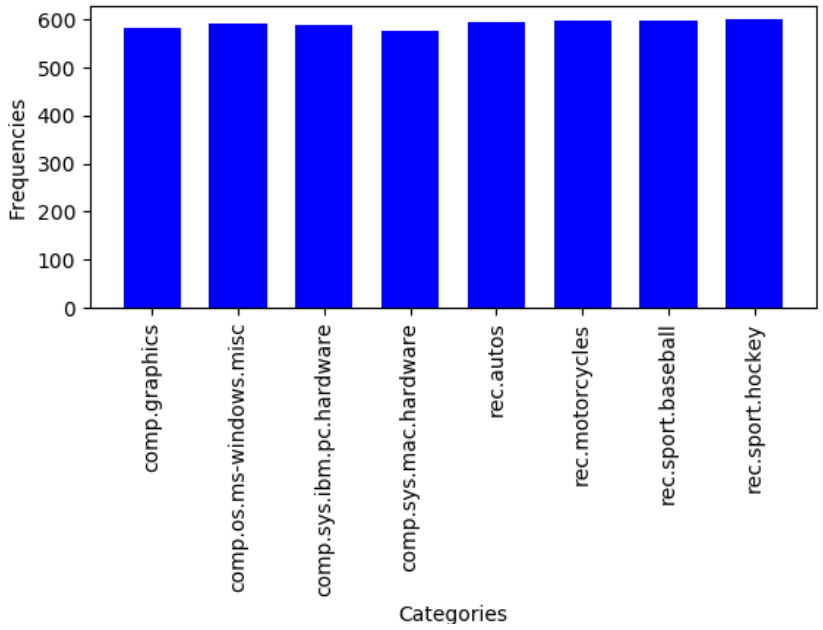

Figure 1: Histogram for the distribution of documents over categories

These eight categories were further partitioned into two separate classes: Computer Technology & Recreational Activity.

The number of documents in each class were as follows:

	Computer Technology	Recreational Activity
Training Dataset	2343	2389
Testing Dataset	1560	1590

Table 1: Distribution of documents across categories

4 Question b : TFxIDF matrices

The goal of this section was to first tokenize the documents, remove commonly occurring Stop-words, lemmatize words into their stems, and finally count occurances per document, and over the entire dataset.

We used the StemTokenizer module form the *nltk* package. Words were split using a regex tokenizer, splitting on whitespaces. A snowball stemmer was used to lemmatize the words and finally the Count Vectorizer was used to count word occurances and nltk stopwords were removed. A pipleine was built for the workflow mentioned, and the documents were processed.

We removed the terms that appeared in large number of documents and infrequent terms by setting max_df = 0.99 (Removing terms that appear in more than 99% of documents) and min_df = 2 (Removing terms that appeared in a single document). This largely reduced our corpus size.

A TFxIDF transformer from the *Sklearn* package was used to build the TFxIDF matrix. Our TFxIDF matrix had 22048 terms.

5 Question c : Class wise TFxIDF

In this task we found the top 10 terms in each of the following classes:

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

comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
scsi	quadra	wolverine	athos
ide	powerbook	hiram	clh
bios	centris	obo	christians
adaptec	c650	sabretooth	christianity
vlb	lciii	liefeld	jayne
motherboard	nubus	hobgoblin	scripture
isa	iisi	hulk	resurrection
aspi4dos	duo	de7	sabbath
irq	simms	02106	liturgy
floppy	adb	forsale	revelation

Table 2: The top 10 terms for each of the categories

6 Question d : LSI Decomposition

In this section, we reduced the number of features in our training and testing data, using the Latent Semantic Indexing technique. This dimensional reduction was necessary to improve the performance and effectiveness of the subsequent classification stages.

To apply LSI, we used the Truncated Singular Value Decompostition of the TFxIDF matrices obtained for the training and testing data. The best 50 dimensions in the feature space were used to represent the TFxIDF matrices for the subsequent sections. We used sublinear transform and l2 norm to calculate the tf-idf values.

7 Question e : Classification using SVM

In this question, we applied the Linear SVM classifier to classify the documents between two classes, Computer Technology and Recreational Activity. The sign of the $W^T\vec{x}^t + b$ was used to classify a document.

Statistic	Result
Accuracy	97.365
Precision	97.401
Recall	97.353

Table 3: Statistics for SVM Classifier

	Predicted CT	Predicted RA
Actual CT	1567	23
Actual RA	60	1500

Table 4: SVM Confusion Matrix (CT: Computer Technology RA: Recreational Activity)

We also plotted a ROC, or Receiver Operating Characteristic curve, to observe the trade-off between the two components of the predictions. We plotted the the probabilities of true positives against the false positives.

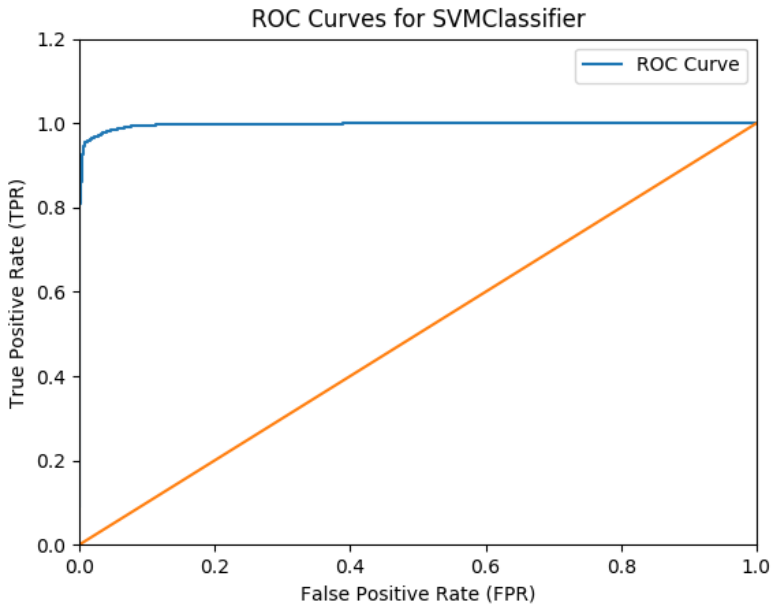

Figure 2: ROC curve for SVM

8 Question f: SVM with Cross Validation

In this task we used Soft Margin SVM, to classify the documents. Cross validation was used with 5 folds, to determine the correct value of the L2 regularization parameter. We determined that k = -2 gave the best training score, and used this parameter for the final classification.

Statistic	Result
Accuracy	97.365
Precision	97.401
Recall	97.353

Table 5: Statistics for SVM Classifier after CV

	Predicted CT	Predicted RA
Actual CT	1567	23
Actual RA	60	1500

Table 6: Confusion Matrix for SVM after CV

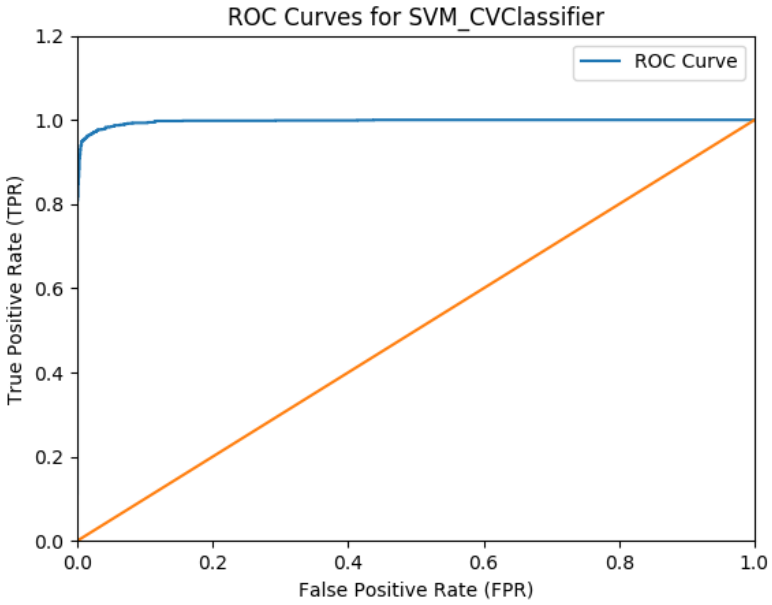

Figure 3: ROC curve for SVM with cross validation

9 Question g : Naive Bayes

In this task we used Naive Bayes algorithm for the same task as above. It calculates the maximum likelihood of probability of a class given a document with feature set X using Bayes rule based on the assumption that the features are independent given the class.

Statistic	Result
Accuracy	95.333
Precision	95.332
Recall	95.334

Table 7: Statistics for Naive Bayes Classifier

	Predicted CT	Predicted RA
Actual CT	1514	76
Actual RA	71	1489

Table 8: Confusion Matrix for Naive Bayes Classifier

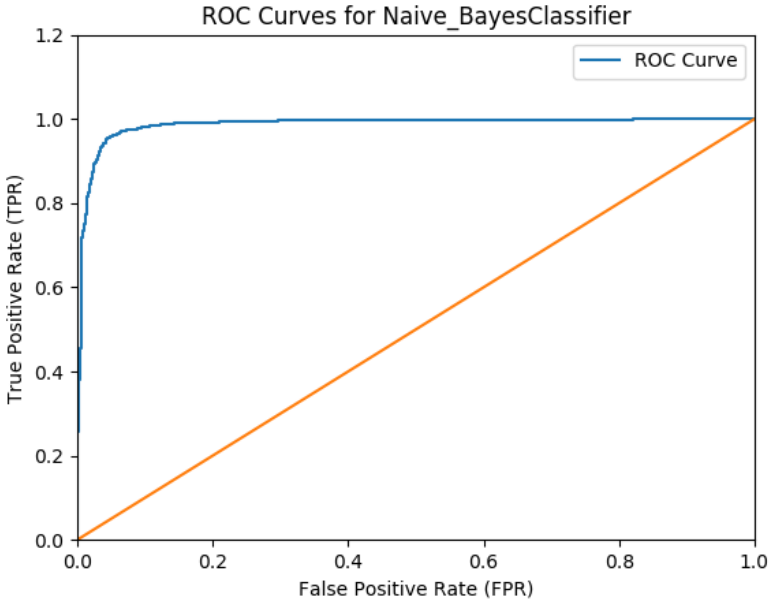

Figure 4: ROC curve for Naive Bayes Classifier

Statistic	Result
Accuracy	97.015
Precision	97.053
Recall	97.004

Table 9: Statistics for Logistic Classifier

	Predicted CT	Predicted RA
Actual CT	1562	28
Actual RA	66	1494

Table 10: Confusion Matrix for Logistic Classifier

10 Question h : Logistic Classifier

Logistic classifer is used next for the same task which basically derives a relationship between a depdendent variable and multiple independent variables, using a logistic function following a cumulative distribution.

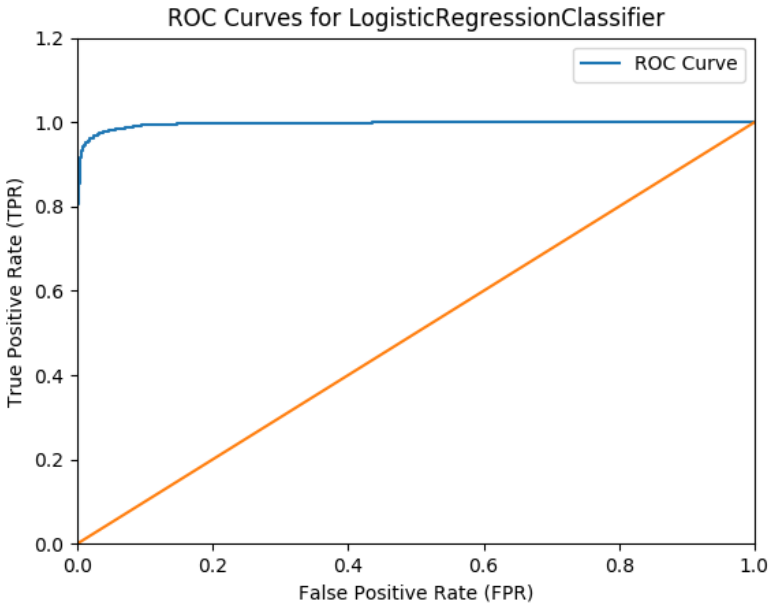

Figure 5: ROC curve for Logistic Classifier

11 Question i - Logistic Regression with Regularization

In this section, we applied L1 and L2 regularization on the Logistic classifiers used in the previous section. The regularization parameter was varied as $\{10^k | -3 <= k <= 3\}$.

Parameter	L1	L2
-3	49.523	25.015
-2	8.444	5.111
-1	5.206	3.523
0	3.142	2.984
1	2.253	2.412
2	2.349	2.222
3	2.412	2.317

Table 11: Testing error against regularization parameter

Parameter	L1	L2
-3	0.0	-0.003
-2	-0.127	-0.034
-1	-1.153	-0.222
0	-1.841	-0.644
1	-1.358	-1.121
2	-0.875	-1.261
3	-0.784	-0.941

Table 12: Coefficient values against regularization parameter

We observed that for very small values of the parameter, excessive regularization took place, and the testing error was very high. However, on increasing the parameter, the testing error steadily reduced before increasing again.

We would typically use L1 loss function when we need a robust solution, but have the computational power, and are willing to tolerate multiple stable solutions.

If the dataset is very large, or if a single unstable solution will work, we would use the L2 loss function.

We also observed that as the regularization parameter increased, the fitted hyperplane shifts away from the origin, before coming closer again.

12 Question j : Multiclass Classification

In this section, we had to classify documents into four separate categories:

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

We used two separate multiclass classification techniques. The OneVsOne technique and the OneVsRest

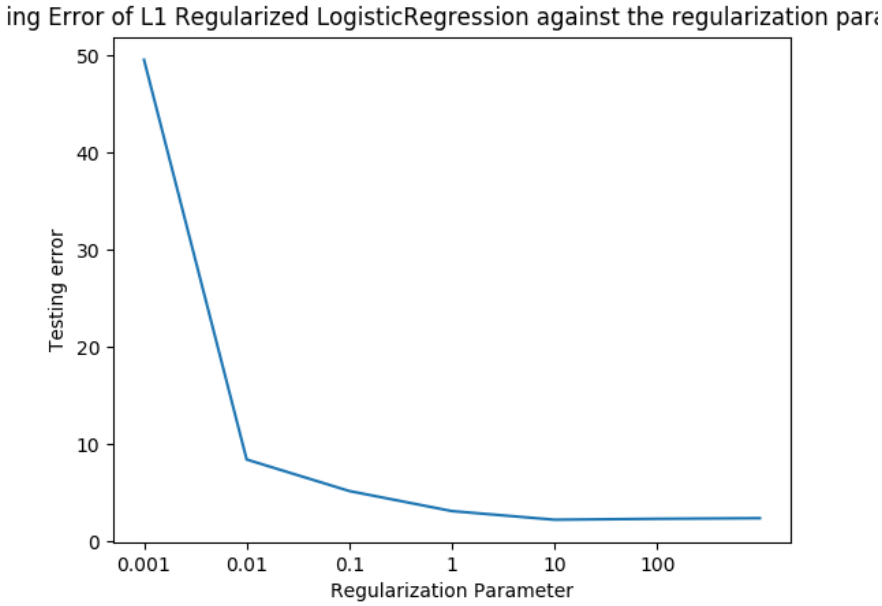

Figure 6: Plot of Testing error against the regularization parameter for L1 Regularization

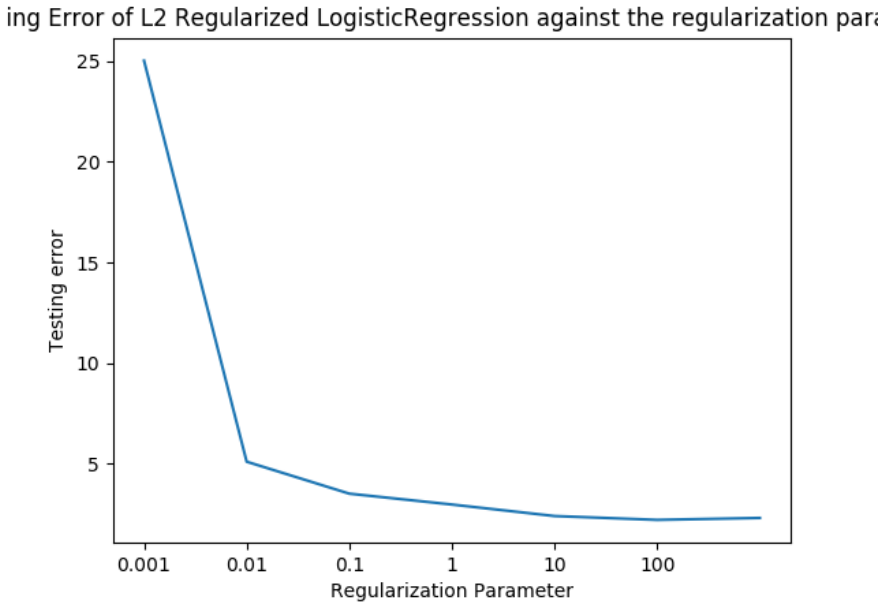

Figure 7: Plot of Testing error against the regularization parameter for L2 Regularization

technique. The OneVsOne technique trains $\binom{n}{2}$ separate classifiers, each one training individual classes against one another. The OneVsRest classifier trains just n classifiers, each one training a single class against the rest.

The two multiclass techniques were applied on the SVM and Naive Bayes classifiers used in the previous sections, the training technique being essentially the same.

Statistic	Result
Accuracy	89.329
Precision	89.439
Recall	89.279

Table 13: Statistics for $OneVsOne$ SVM Classifier

	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 1	339	39	14	0
Actual 2	39	327	19	0
Actual 3	27	17	346	0
Actual 4	6	2	4	386

Table 14: Confusion matrix for $OneVsOne$ SVM Classifier

Statistic	Result
Accuracy	89.329
Precision	89.270
Recall	89.280

Table 15: Statistics for $OneVsRest$ Classifier

	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 1	322	44	22	4
Actual 2	28	329	27	1
Actual 3	20	13	356	1
Actual 4	3	1	3	391

Table 16: Confusion matrix for $OneVsRest$ SVM Classifier

Statistic	Result
Accuracy	75.207
Precision	77.94
Recall	75.012

Table 17: Statistics for $OneVsRest$ Naive Bayes Classifier

	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 1	276	15	94	7
Actual 2	61	182	124	18
Actual 3	38	14	333	5
Actual 4	0	0	12	386

Table 18: Confusion matrix for $OneVsRest$ Naive Bayes Classifier

Statistic	Result
Accuracy	75.910
Precision	78.227
Recall	75.718

Table 19: Statistics for OneVsOne Naive Bayes Classifier

	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 1	283	14	84	11
Actual 2	72	186	110	17
Actual 3	40	14	333	3
Actual 4	0	0	12	386

Table 20: Confusion matrix for OneVsOne Naive Bayes Classifier