ECE219 Project 1: Classification Analysis on Textual Data

Larsan Aro Brian (904943186) and Firnaz Ahamed (104943091)

January 29, 2018

1 Introduction

Text classification is one of the common applications of supervised learning processes in natural language processing. It can be used in several applications such as spam filtering, sentiment analysis etc. In this project, we explore the classification of text into different predefined classes based on the Term Frequency Inverse Document Frequency (TF-IDF) feature which gives the relative importance of a term in a document as compared to the entire corpus as a whole. Dimensionality reduction was implemented to map the high dimensional matrices to lower dimensions for faster implementation. Several models such as Support Vector Machine (SVM), Naive Bayes classifier and logistic regression classifier were fitted and hyperparameter tuning was done to determine the optimal parameters. Throughout this project, we experimented with several parameters and models and analyzed how they affected the classification process.

2 Dataset

The dataset which was used for this project was the '20 Newsgroups' dataset. The dataset had a total of 18846 newsgroup documents which were classified evenly across 20 classes. The dataset was imported directly from the scikit-learn library. For a major part of this project, a subgroup of this dataset corresponding to 8 classes were used. The dataset corresponding to these 8 classes had 4732 training documents and 3150 testing documents.

3 Document Balance: Histogram

The training and test datasets of the 8 classes were converted into pandas dataframes and the histogram of the training set documents was plotted using the matplotlib library functions. The histogram (shown in figure 1) is evenly spread indicating that the dataset is balanced. Therefore we can directly proceed to build our model with this dataset without having to do any balancing actions.

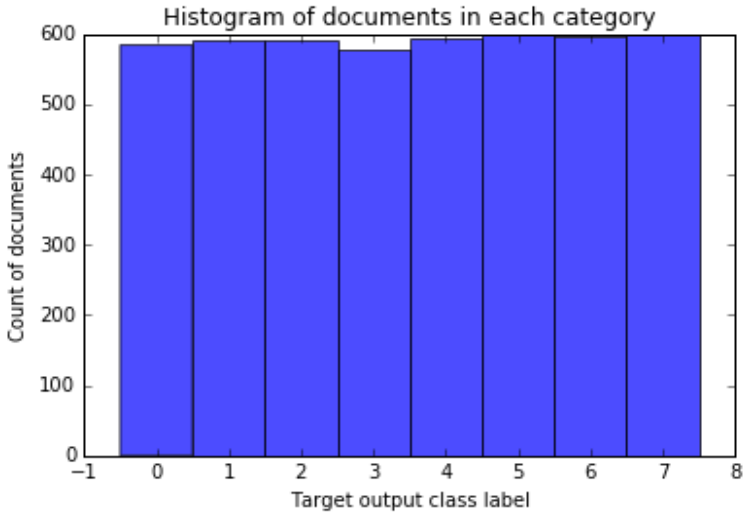

Figure 1: Histogram showing the distribution of documents in each of the 8 classes

4 Cleaning the Raw Text

In any text based mining problem, be it classification or clustering, the first task at hand is to clean the data. This means that we must convert the raw text into a form in which we would be able to extract useful inferences to begin our classification problem. The major tasks in this process would be to extract words from the raw text by eliminating punctuations and stop words, and trimming words into their stem-words. In the cleaning process, first all the punctuations and numbers were removed using regular expression. All the words were converted to lower case, stop words were removed and stemmed to their root word using PorterStemmer.

5 Creating the Bag of Words model

Once the text was cleaned, the bag of words model was created using the CountVectorizer scikit-learn package. The bag of words model refers to the matrix that represents the frequency of occurrence of each word in each document of the corpus. Each column in the matrix refers to an unique word from the documents and each row represents the document itself. The value present in the matrix refers to the frequency of occurrence of the word in that document. The min_df parameter is used to determine the minimum number of occurrence of a word in a document for it be considered in the matrix.

min_df	Number of terms
2	19320
5	8958

6 Term-Frequency Inverse Document Frequency (TF-IDF)

TF-IDF refers to a commonly used statistical measure that determines how important a word is to a document in a collection of documents or corpus. The importance is determined by a weight that is given based on the two components called term frequency and inverse document frequency. The basic idea behind tf-idf is that when a term occurs more frequently, it is more important since it carries the bulk of the information in the document. It is important as to how much information the word carries and contributes to the document, and based on this the weight should be adjusted. Thus tf-idf weight consists of the following components:

Term frequency The frequency of occurrence of a particular term in a document is called term frequency. The words with a higher term frequency will be given a higher weight.

Inverse document frequency This gives the measure of how important the word is to the document and how much information it provides to the document. It is found by taking the negative logarithm of the fraction of documents that contain the particular word out of all the documents in the corpus. The weight of the words that occur in most of the documents will be reduced, while the less commonly occurring words' weights will be increased.

We used the TfidfTransformer from scikit-learn library to generate the tfidf weights. The TfidfTransformer object takes the bag of words model matrix as input and returns a sparse matrix where the columns correspond to the words in the data and the rows correspond to the tfidf weights.

7 Term-Frequency Inverse Class Frequency (TF-ICF)

Similar to how TF-IDF is used to determine how important a word is to a document, TF-ICF is used to determine the significance of a word to a class. For this part of the assignment, we used the entire dataset which included all the 20 classes to determine the 10 most significant words in 4 of those classes. The documents were grouped based on the output classes and TfidfTransformer was used to determine the weights given to the words relative to the classes. The following table shows the 10 most significant words (the stemmed words) in each of those 4 classes.

Class	Most significant words
comp.sys.ibm.pc.hardware	scsi, edu, drive, com, line, ide, subject, use, organ, card
comp.sys.mac.hardware	edu, line, mac, subject, organ, use, appl, post, problem, drive
misc.forsale	edu, line, subject, sale, organ, post, univers, com, new, host
soc.religion.christian	god, christian, edu, church, subject, jesu, homosexu, peopl, line, sin

7.1 Word clouds

We created wordclouds to visually represent the most significant words in each of the classes. The 50 most significant words from each class were used to create these wordclouds.

7.1.1 IBM

Figure 2: Wordcloud for the comp.sys.ibm.pc.hardware class

7.1.2 Mac

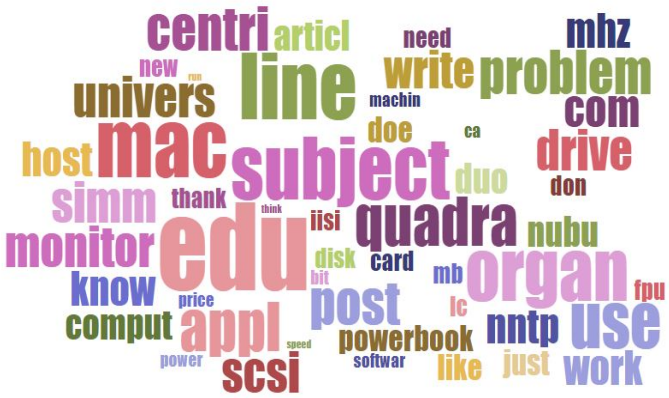

Figure 3: Wordcloud for the comp.sys.mac.hardware class

7.1.3 Misc

Figure 4: Wordcloud for the misc.forsale class

7.1.4 Christian

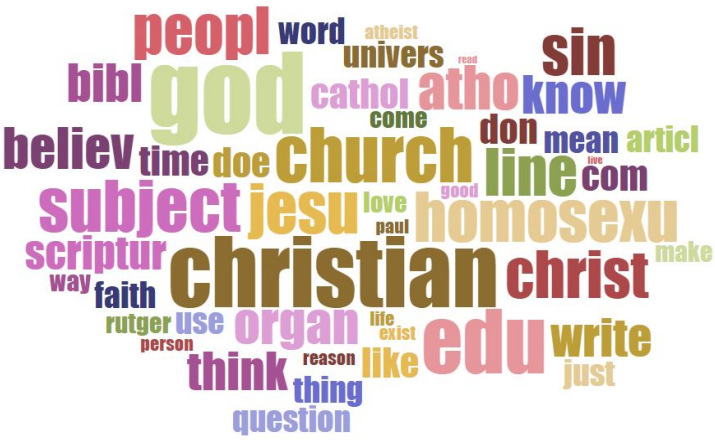

Figure 5: Wordcloud for the soc.religion.christian class

8 Dimensionality Reduction

Since we are dealing with high dimensional matrices, dimensionality reduction plays a crucial role in improving the speed and accuracy. The high dimensional vectors are mapped to lower dimensions that will best explain the variance in the data. Singular Value Decomposition (SVD) plays a significant role in the dimensionality reduction process. In this project, we have used two different dimensionality reduction techniques, namely Latent Semantic Indexing (LSI) and Non-Negative Matrix Factorization (NMF) to map the high dimensional vectors to 50 dimensions.

9 Latent Semantic Indexing

Latent semantic indexing (LSI) is a commonly used method to determine the relationship between terms and concepts in content. It can be used for dimensionality

reduction. All the following subsections have been implemented after applying LSI for dimensionality reduction.

9.1 Support Vector Machine

Support vector machine (SVM) is a commonly used supervised learning model that can be used to classify data using a hyperplane that achieves the largest margin. Linear SVMs have been efficient in dealing with sparse matrices such as the Tfidf matrix and hence can be an effective tool in text classification.

9.1.1 Hard margin SVM

In support vector machines, there is a penalty parameter γ that is used to determine how much a misclassification is penalized. Depending on the value of γ, the SVM can be classified as hard margin and soft margin. In hard margin SVM, any type of misclassification is highly penalized and contributes to a larger value in the loss function. Hard margin SVM was implemented using a γ of 1000 and the following results were achieved.

Min_df = 2:

		Predicted Class	
		0	1
Actual Class	0	1509	51
	1	25	1565

	Hard Margin Support Vector Machine
Accuracy	0.9758
Precision	0.9684
Recall	0.9842

Min_df = 5:

		Predicted Class	
		0	1
Actual Class	0	1504	56
	1	22	1568

	Hard Margin Support Vector Machine
Accuracy	0.9752
Precision	0.9655
Recall	0.9861

Interpretation As seen from the above results, the hard margin SVM has performed very well. Since, it penalizes all the misclassification it is able to fit the data well. As a result, the accuracy, precision and recall are exceptionally high. There is also no evident difference due to the min_df variation.

9.1.2 Soft margin SVM

In soft margin SVM, there is some leeway for misclassification. The misclassification is not as heavily penalized as in the case of hard margin SVM. In this part, we have used a γ of 0.001 to obtain the following results.

Min_df = 2:

		Predicted Class	
		0	1
Actual Class	0	0	1560
	1	0	1590

	Soft Margin Support Vector Machine
Accuracy	0.5047
Precision	0.5047
Recall	1.0000

Min_df = 5:

		Predicted Class	
		0	1
Actual Class	0	0	1560
	1	0	1590

	Soft Margin Support Vector Machine
Accuracy	0.5047
Precision	0.5047
Recall	1.0000

Interpretation As seen from the results above, the soft margin SVM does not perform well. In both the cases of min_df = 2 and min_df = 5, the classifier has predicted all the output values to be of class 1. This has led to a very poor accuracy and precision. Since soft SVM allows misclassification, the low γ translates to underfitting. As a result of the model underfitting the data, the classification results turn out to be poor.

9.1.3 Cross-validation to find γ

Since the penalty parameter γ plays a crucial role in determining the accuracy of the classifier, 5 fold cross validation was done to find the best value of γ. By using GridSearchCV from sklearn package, the penalty parameter was swept from 0.001 to 1000 to determine the best penalty parameter. The best value of γ was found to be 100 both in the case of min_df=2 and min_df=5. The confusion matrix and classification results that we obtained by setting γ as 100 are tabulated below.

Min_df=2:

		Predicted Class	
		0	1
Actual Class	0	1506	54
	1	26	1564

	Support Vector Machine with best value of γ
Accuracy	0.9746
Precision	0.9666
Recall	0.9836

Min_df=5:

		Predicted Class	
		0	1
Actual Class	0	1505	55
	1	23	1567

	Support Vector Machine with best value of γ
Accuracy	0.9752
Precision	0.9660
Recall	0.9855

Interpretation Firstly, we have found that the best value of γ is 100. We have ended with a penalty parameter that isn't too high or too small. This optimum penalty parameter is obtained to avoid the extreme cases of underfit and overfit. By using the optimum value of γ the classifier performs well and achieves good accuracy. There is no considerable difference in performance by changing the value of min_df.

9.2 Naive Bayes classifier

Naive Bayes classifier is a simple probabilistic classifier that using the Bayes theorem and a few naive independence assumptions to classify data. Generally multinomial Naive Bayes classifiers are used in the case of text classification. However, due to the presence of negative values in the tfidf matrix after the application of Latent Semantic Indexing, multinomial Naive Bayes classifier could not be fitted to the data. Hence, we used Gaussian Naive Bayes classifier instead.

Min_df=2:

		Predicted Class	
		0	1
Actual Class	0	1323	237
	1	76	1514

	Naive Bayes Classifier
Accuracy	0.9006
Precision	0.8646
Recall	0.9522

Min_df=5:

		Predicted Class	
		0	1
Actual Class	0	1216	344
	1	49	1541

	Naive Bayes Classifier
Accuracy	0.8752
Precision	0.8175
Recall	0.9691

Interpretation As seen from the results, the Gaussian Naive Bayes classifier does not perform as well as the other classifiers. In general, the multinomial Bayes classifier is applied for such data and it gives good results. Due to the presence of negative elements in the tfidf matrix after applying LSI, we had to opt for Gaussian Naive Bayes classifier and its performance is not as good as the others.

9.3 Logistic Regression classifier

Logistic Regression is a commonly used method in binary classification problems. Since this text classification problem has been adapted to become binary, logistic regression performs well on it.

Min_df=2:

		Predicted Class	
		0	1
Actual Class	0	1493	67
	1	33	1557

	Logistic Regression Classifier
Accuracy	0.9682
Precision	0.9587
Recall	0.9792

Min_df=5:

		Predicted Class	
		0	1
Actual Class	0	1486	74
	1	32	1558

	Logistic Regression Classifier
Accuracy	0.9663
Precision	0.9546
Recall	0.9798

Interpretation The logistic regression classifier gives good results on the text classification problem. Both min_df = 2 and min_df = 5 produce similar results. Logistic regression classifier works well on binary classification problems and it can be clearly seen from the exceptionally good accuracy, precision and recall.

9.3.1 Effect of regularization term

Regularization plays an important role in preventing overfitting. In this part of the project, we have varied the regularization coefficient using both $l1$ and $l2$ norms to determine the best regularization norm and its corresponding coefficient. By using a grid search through the different penalties, we found that $l1$ regularization with a regularization coefficient of 10 gave us the best results. In the case of $l2$ regularization, a coefficient of 100 gave the best result. In general, a higher regularization coefficient gave better result and $l1$ norm performed better than $l2$ norm. The ROC curves corresponding to the best coefficient for both values of min_df are presented at the end of the report along with the other ROC curves.

Min_df=2:

	$l1$ regularization	$l2$ regularization
Best coefficient	10	100
Best accuracy	0.9750	0.9756

Min_df=5:

	$l1$ regularization	$l2$ regularization
Best coefficient	10	100
Best accuracy	0.9746	0.9748

Observation and interpretation In the logistic regression class from the sklearn library, the parameter C refers to the inverse of the regularization strength. Smaller value of C refers to stronger regularization. By varying the value of C, we found out that for small values of C, the coefficients of the fitted hyperplane are very small and as the value of C increases, the coefficients become larger. Another observation was that as the value of C increases, the test error reduces. Regularization is used in order to avoid overfitting.

In general, the $l1$ regularization is preferred when the data is sparse. $l2$ regularization, on the other hand, is computationally efficient due to having analytical solutions but it does not perform as well as $l1$ on sparse data. $l1$ regularization also has a built-in feature selection which is not present in $l2$ regularization.

10 Non-negative matrix factorization (NMF)

Non-negative matrix factorization is a commonly used decomposition method for multivariate data. It decomposes the matrix into two matrices with non-negative values which allows better application. All the results in the following subsections were obtained using NMF as the dimensionality reduction technique.

10.1 Support Vector Machine(SVM)

Using a min_df value of 2, the following confusion matrices and accuracy metrics were determined, for both the hard margin SVM and soft margin SVM.

10.1.1 Hard margin SVM

		Predicted Class	
		0	1
Actual Class	0	1499	61
	1	56	1534

	Hard Margin Support Vector Machine
Accuracy	0.9628
Precision	0.9617
Recall	0.9647

Interpretation The hard margin SVM gives good results with high values for accuracy, precision and recall. The classification results are pretty similar to those obtained by using LSI for dimensionality reduction. By not allowing misclassification, the model is able to give high classification scores.

10.1.2 Soft margin SVM

		Predicted Class	
		0	1
Actual Class	0	0	1560
	1	0	1590

	Soft Margin Support Vector Machine
Accuracy	0.5047
Precision	0.5047
Recall	1.0000

Interpretation The results obtained for soft SVM for NMF turns out to be exactly the same as that obtained using LSI. Its performance is extremely bad compared to all the other classifiers since it allows a lot of misclassification. Both accuracy and precision values are very low.

10.1.3 Cross validation to find γ

By using a grid search on the penalty parameter and sweeping it from 0.001 to 1000, we found the best penalty parameter to be 100. The following confusion matrix and classification results were obtained by setting γ as 100.

		Predicted Class	
		0	1
Actual Class	0	1483	77
	1	59	1531

	Support Vector Machine with best value of γ
Accuracy	0.9568
Precision	0.9521
Recall	0.9628

Interpretation Even in this case of cross validation to finding γ, we have obtained the best γ value to be 100, similar to that obtained using LSI. This is neither too high nor too low. This value of γ gives good classification results similar to those obtained using LSI.

10.2 Naive Bayes classifier

A naive bayes classifier was fitted onto the tfidf matrix that was obtained after applying NMF and the following results were obtained.

		Predicted Class	
		0	1
Actual Class	0	1462	98
	1	86	1504

	Naive Bayes Classifier
Accuracy	0.9415
Precision	0.9388
Recall	0.9459

Interpretation The multinomial Naive Bayes classifier has performed decently well on this data. It has performed better than the Gaussian Niave Bayes classifier that we used in the case of LSI. However, it does not give as good a result as those obtained using the other classifiers.

10.3 Logistic Regression classifier

A logistic regression classifier was fitted onto the tfidf matrix that was obtained after applying NMF and the following results were obtained.

		Predicted Class	
		0	1
Actual Class	0	1358	202
	1	42	1548

	Logistic Regression Classifier
Accuracy	0.9225
Precision	0.8845
Recall	0.9735

Interpretation The logistic regression classifier too performs decently well on the data. However this logistic regression classifier does not perform as well as the logistic regression classifier that we obtained using LSI. The LSI seems to outperform NMF, at least in the case of Logistic regression classification.

10.3.1 Effect of regularization term

By using a grid search through the different penalties and coefficients, we found that the $l1$ regularization with a coefficient of 10 gave the best result and in the case of $l2$ regularization, a coefficient of 1000 gave the best result. The results are tabulated below.

	$l1$ regularization	$l2$ regularization
Best coefficient	10	1000
Best accuracy	0.9687	0.9678

Interpretation Without regularization, the logistic regression classifier did not obtain very good results. It had an accuracy of 0.9225 and that accuracy increased to the range of 0.96 by optimizing the hyperparameter of regularization coefficient. This shows the importance of hyperparameter tuning in improving the results.

11 Multi-class classification

In the first half of the project, we have been dealing with binary classification where the output was classified into Computer technology and Recreational activity. Now, we implement multi-class classification with 4 output classes. Naive Bayes classifier and SVM classifier are used in this section for classification. In SVM, both one-vs-one and one-vs-rest classifiers have been implemented and both LSI and NMF have been used for dimensionality reduction.

11.1 Latent Semantic Index (LSI)

The following results were obtained by using LSI as the technique for dimensionality reduction. In the case of Naive Bayes classifier using LSI, GaussianNB was used instead of MultinomialNB.

11.1.1 Naive Bayes Classifier

		Predicted Class			
		0	1	2	3
Actual Class	0	228	32	129	3
	1	73	158	150	4
	2	42	40	308	0
	3	0	0	27	371

Accuracy	Class	Precision	Recall
0.6805	0	0.6647	0.5816
	1	0.6870	0.4104
	2	0.5016	0.7897
	3	0.9815	0.9322
Average		0.7101	0.6805

Interpretation Here the Naive Bayes classifier does not perform really well, the main reason being the usage of Gaussian Naive Bayes instead of Multinomial Naive Bayes classifier. The accuracy, precision and recall values are pretty low compared to the other classifiers.

11.1.2 SVM classifier one vs one

		Predicted Class			
		0	1	2	3
Actual Class	0	339	30	23	0
	1	55	304	26	0
	2	33	14	342	1
	3	8	1	7	382

Accuracy	Class	Precision	Recall
0.8738	0	0.7793	0.8648
	1	0.8711	0.7896
	2	0.8593	0.8769
	3	0.9974	0.9598
Average		0.8773	0.8735

Interpretation The SVM classifier performs well on the data. The classification results are not as high as those obtained in the binary classification since we have more classes which increases the probability of misclassification. Nonetheless, these classification results are pretty good.

11.1.3 SVM classifier one vs rest

		Predicted Class			
		0	1	2	3
Actual Class	0	308	49	32	3
	1	33	321	30	1
	2	21	13	354	2
	3	2	1	2	393

Accuracy	Class	Precision	Recall
0.8792	0	0.8462	0.7857
	1	0.8359	0.8338
	2	0.8469	0.9077
	3	0.9850	0.9874
Average		0.8791	0.8792

Interpretation The One vs rest SVM classifier performs as well as the one vs one SVM classifier. There does not seem to be a significant difference in performance in the two types of SVM classifiers. Both the classifiers produce good results.

11.2 Non-negative matrix factorization (NMF)

The following results were obtained by using NMF as the technique for dimensionality reduction.

11.2.1 Naive Bayes Classifier

		Predicted Class			
		0	1	2	3
Actual Class	0	273	39	77	3
	1	84	223	74	2
	2	48	25	310	7
	3	0	1	3	394

Accuracy	Class	Precision	Recall
0.7667	0	0.6741	0.6964
	1	0.7743	0.5792
	2	0.6681	0.7949
	3	0.9657	0.9899
Average		0.7714	0.7668

Interpretation The Naive Bayes classifier here performs better than that of the LSI case only because this uses MultinomialNB while that uses GaussianNB. In general, the Naive Bayes classifier does not perform that well as the other classifiers.

11.2.2 SVM classifier one vs one

		Predicted Class			
		0	1	2	3
Actual Class	0	168	210	14	0
	1	24	351	10	0
	2	26	64	300	0
	3	0	24	3	371

Accuracy	Class	Precision	Recall
0.7603	0	0.7706	0.4286
	1	0.5408	0.9117
	2	0.9174	0.7692
	3	1.0000	0.9322
Average		0.8090	0.7604

Interpretation The SVM one vs one classifier in the NMF case does not perform as well as that obtained using LSI. Quite clearly the LSI outperforms the NMF. The SVM classifier in this case gives an average result.

11.2.3 SVM classifier one vs rest

		Predicted Class			
		0	1	2	3
Actual Class	0	285	45	34	28
	1	60	244	45	36
	2	28	8	342	12
	3	0	0	1	397

Accuracy	Class	Precision	Recall
0.8102	0	0.7641	0.7270
	1	0.8215	0.6338
	2	0.8104	0.8769
	3	0.8393	0.9975
Average		0.8089	0.8102

Interpretation The one vs rest SVM classifier performs equally badly as the one vs one SVM classifier in the NMF case. The accuracy is more in this case but the recall is lesser. The classifiers obtained using LSI performed better than those obtained using NMF.

12 Receiver Operating Characteristics (ROC) curves

12.1 Results using LSI

12.1.1 Min_df=2

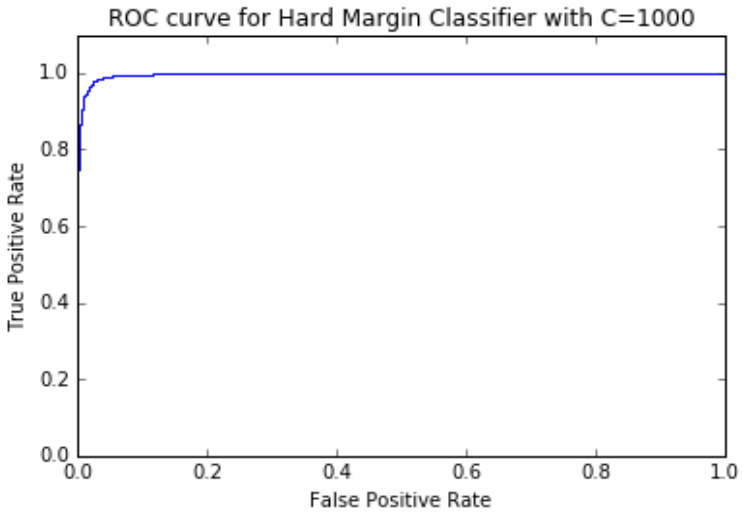

Figure 6: Receiver Operating Characteristics (ROC) curve for Hard Margin Classifier with with a coefficient of 1000

Interpretation The ROC curve for the Hard margin SVM classifier shows that it performs very well. The area under curve is very large and it shows the good performance of the classifier.

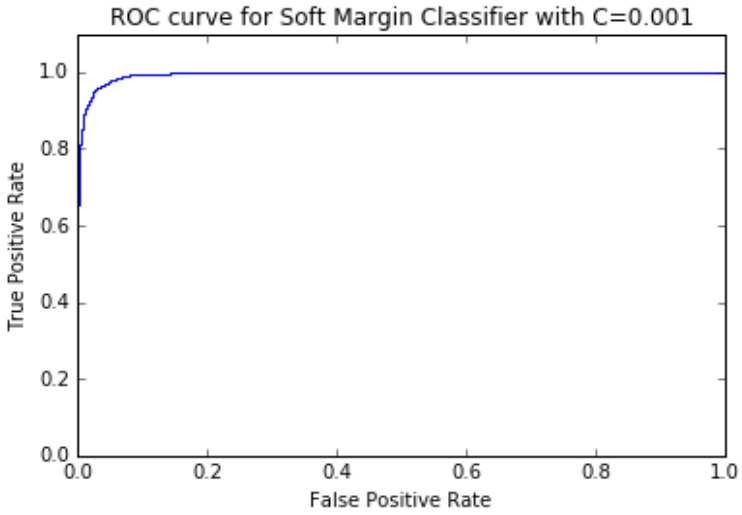

Figure 7: Receiver Operating Characteristics (ROC) curve for Soft Margin Classifier with coefficient of 0.001

Interpretation The soft margin SVM does not give as good a result as the hard margin SVM. The True Positive Rate (TPR) takes a longer time to reach the maximum than the Hard margin SVM.

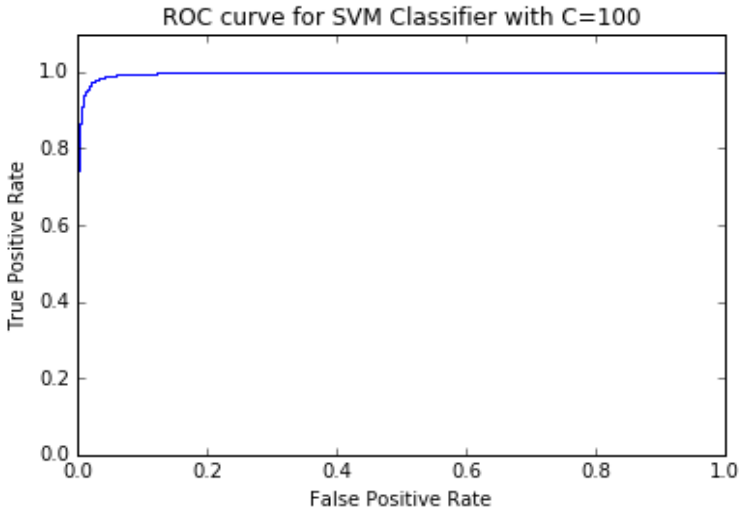

Figure 8: Receiver Operating Characteristics (ROC) curve for Support Vector Machine(SVM) with coefficient of 100

Interpretation By using the best value of C, the SVM classifier performs really well which can be clearly seen from this ROC curve which has a larger area under the curve (AUC).

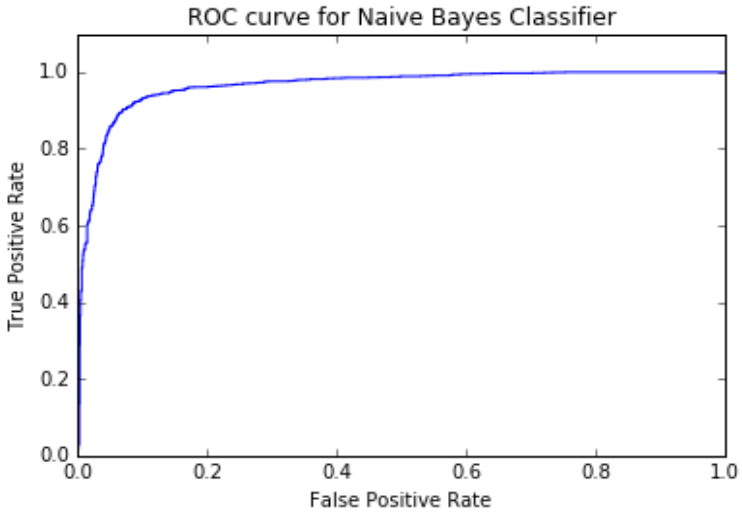

Figure 9: Receiver Operating Characteristics (ROC) curve for Naive Bias Classifier

Interpretation As seen from the ROC curve for Naive Bayes Classifier, it does not perform well on the data, the main reason being the usage of Gaussian Naive Bayes instead of Multinomial Naive Bayes classifier. The True Positive Rate (TPR) rises very slowly and the area under the curve is not large.

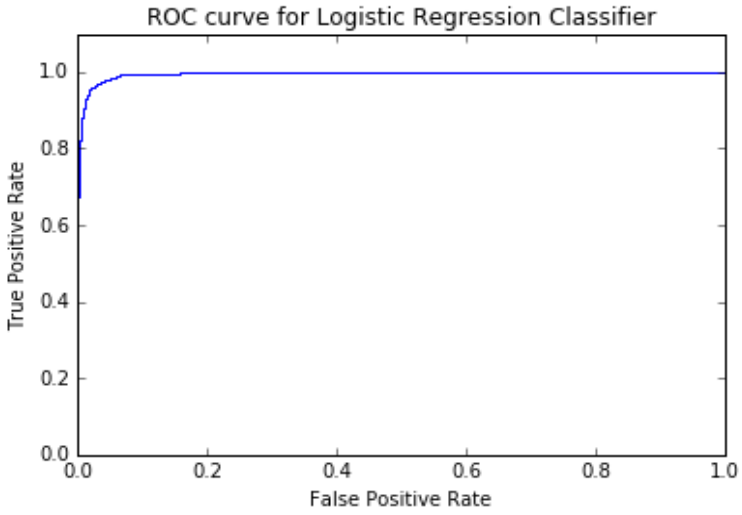

Figure 10: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier

Interpretation The logistic regression classifier performs well on this binary classification problem as seen from the good ROC curve.

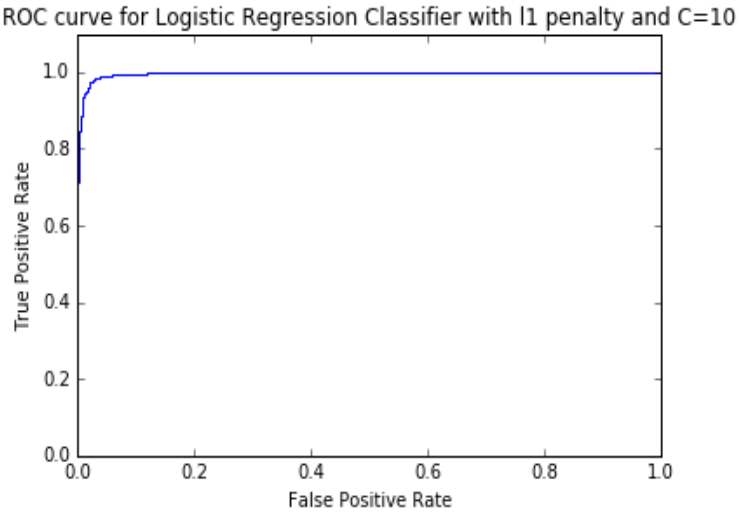

Figure 11: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l1 regularization with a coefficient of 10

Interpretation By optimizing the regularization coefficient and using the best value of regularization coefficient, the ROC curve has become better and shows great results.

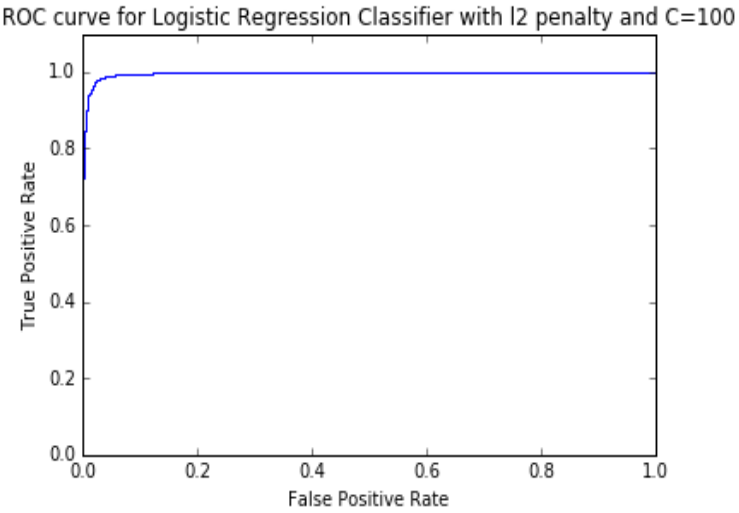

Figure 12: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l2 regularization with a coefficient of 100

Interpretation Even in this case of l2 regularization, the ROC curve is similarly good as in the case of l1 regularization and it does not show much difference between the two regularization norms.

12.1.2 Min_df=5

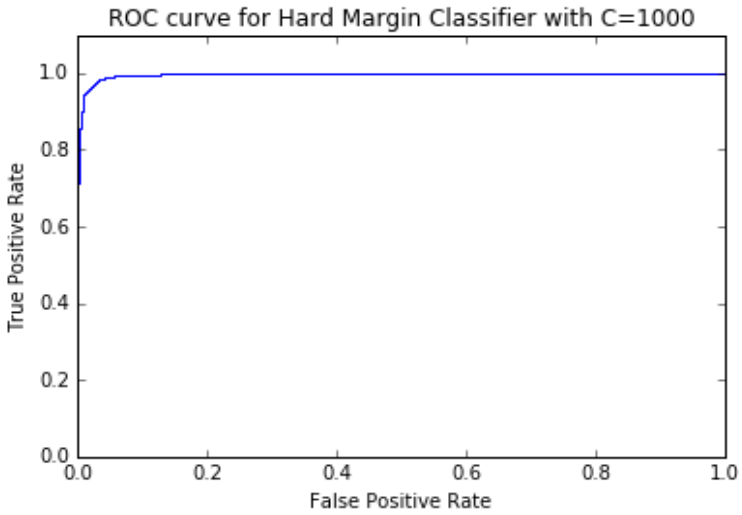

Figure 13: Receiver Operating Characteristics (ROC) curve for Hard Margin Classifier with a coefficient of 1000

Interpretation The ROC curve for the case of min_df = 5 shows similar results as those obtained using min_df = 2.

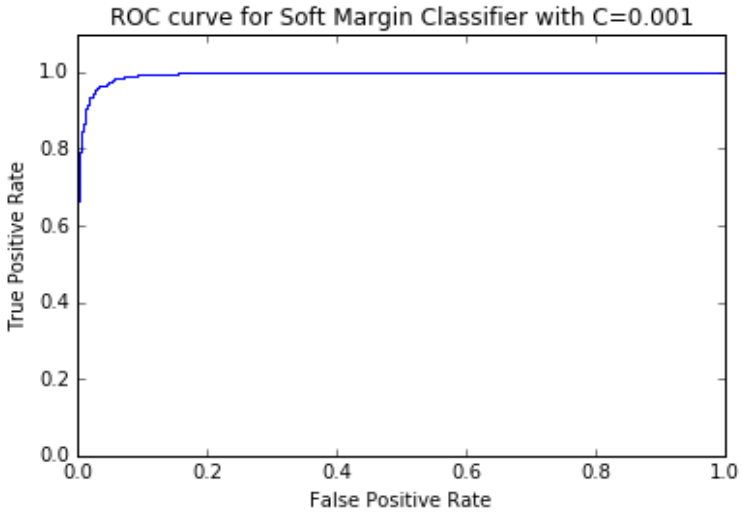

Figure 14: Receiver Operating Characteristics (ROC) curve for Soft Margin Classifier with a coefficient of 0.001

Interpretation Hard margin SVM classifier performs better than the soft margin SVM classifier as seen from the ROC curves, however there is no difference between the two min_df.

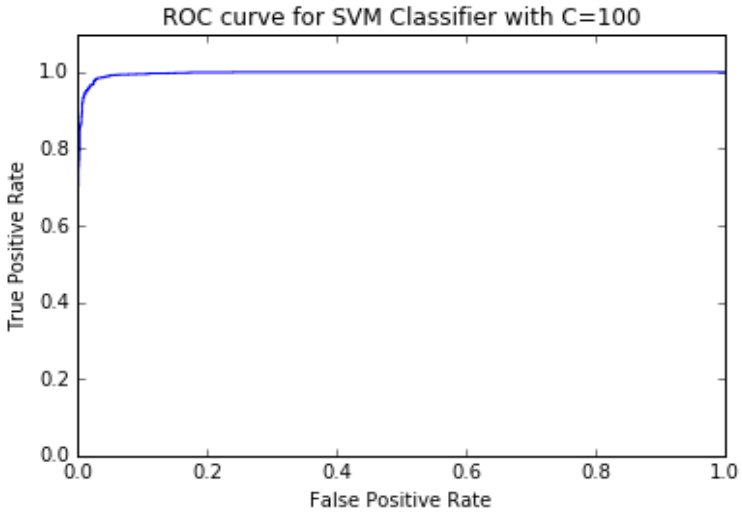

Figure 15: Receiver Operating Characteristics (ROC) curve for Support Vector Machine(SVM) with coefficient of 100

Interpretation The ROC curve for this value of C shows the best classifier results as seen from the ROC curves.

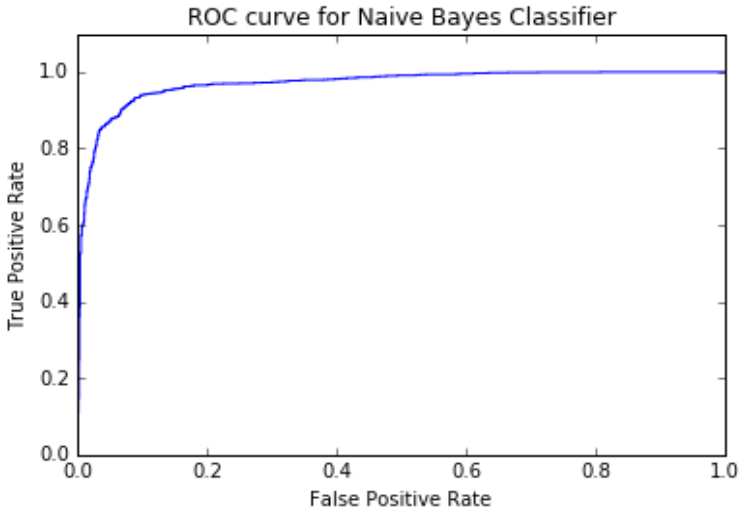

Figure 16: Receiver Operating Characteristics (ROC) curve for Naive Bias Classifier

Interpretation The ROC curve for Naive Bayes classifier shows poor results due to the use of Gaussian Naive Bayes classifier. However there is no difference between the two min_df.

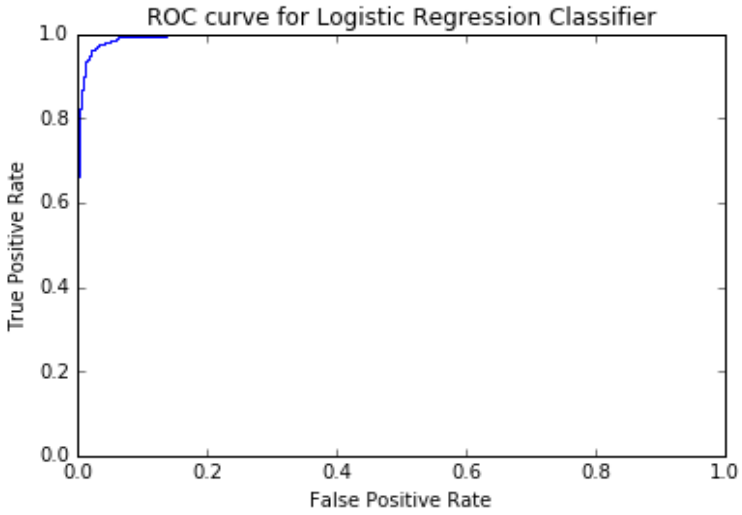

Figure 17: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier

Interpretation The logistic regression classifier shows good results which is expected since logistic regression classifier work well on binary classification problems.

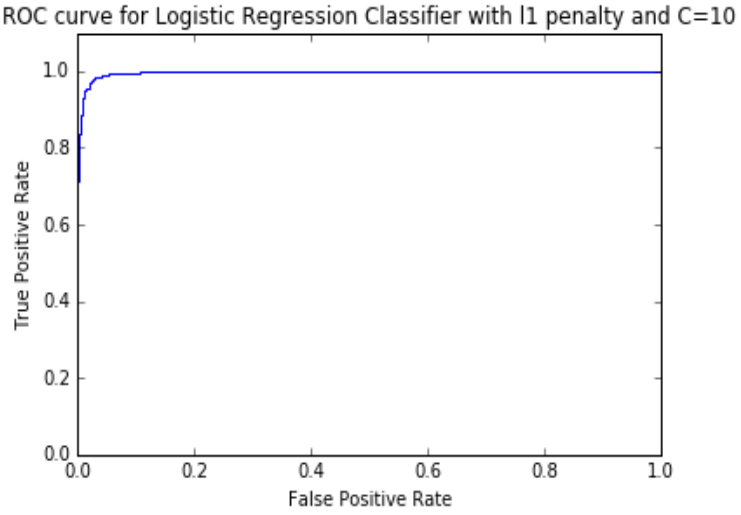

Figure 18: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l1 regularization with a coefficient of 10

Interpretation By adding a regularization term and by optimizing the coefficient, the logistic regression classifier performs better which can be seen from the curves.

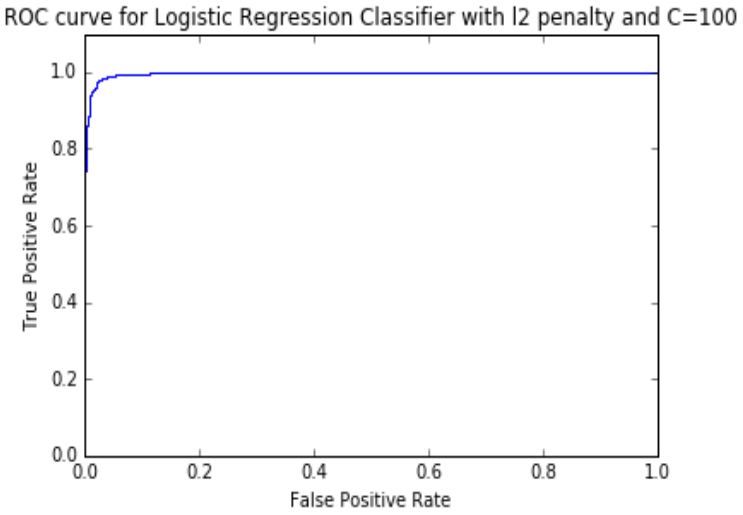

Figure 19: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l2 regularization with a coefficient of 100

Interpretation The l2 regularization shows similarly good results as those obtained using l1 regularization. The change in min_df term also does not affect the results much.

12.2 Results using NMF

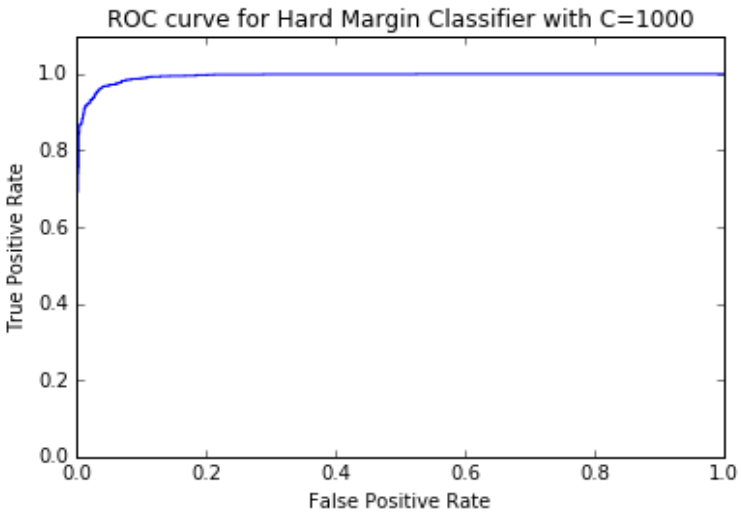

Figure 20: Receiver Operating Characteristics (ROC) curve for Hard Margin Classifier with a coefficient of 1000

Interpretation It can be seen by comparing the ROC curves for the LSI case with the NMF case that LSI produces better results and the area under the curve is larger for the LSI case.

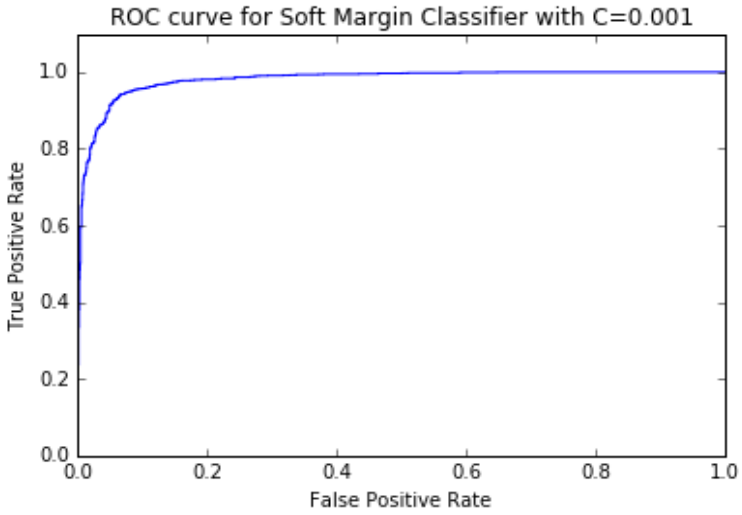

Figure 21: Receiver Operating Characteristics (ROC) curve for Soft Margin Classifier with a coefficient of 0.001

Interpretation As seen from the ROC curve, the soft margin classifier does not perform as well as the hard margin classifier since it allows a lot of misclassification.

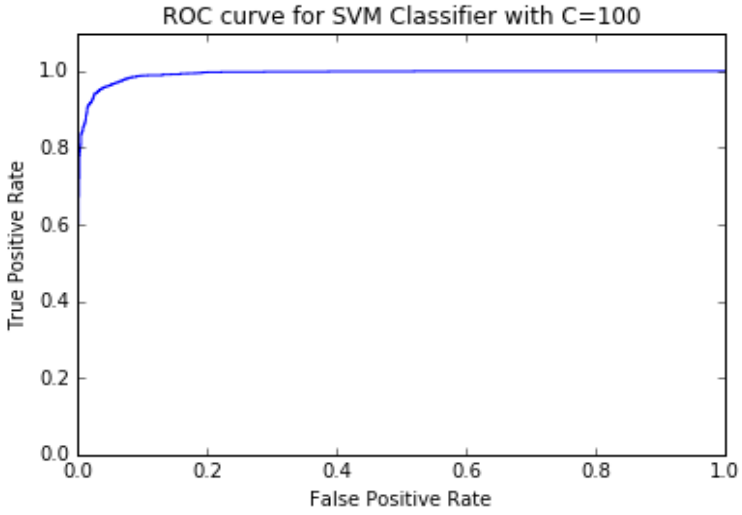

Figure 22: Receiver Operating Characteristics (ROC) curve for Support Vector Machine(SVM) with a coefficient of 100

Interpretation By using the best value of C, the ROC curve becomes a lot better which can be very clearly seen by comparing this ROC curve with the ROC curve above.

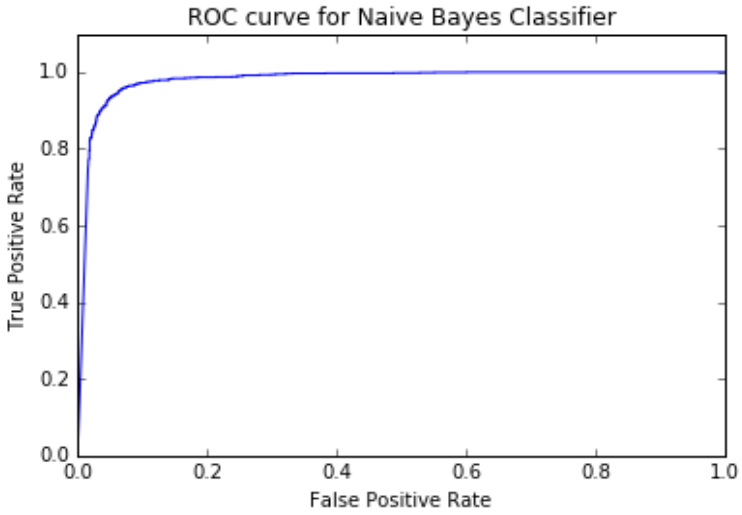

Figure 23: Receiver Operating Characteristics (ROC) curve for Naive Bias Classifier

Interpretation The ROC curve for the Naive Bayes classifier is not as good as the other classifiers such as the hard margin SVM classifier.

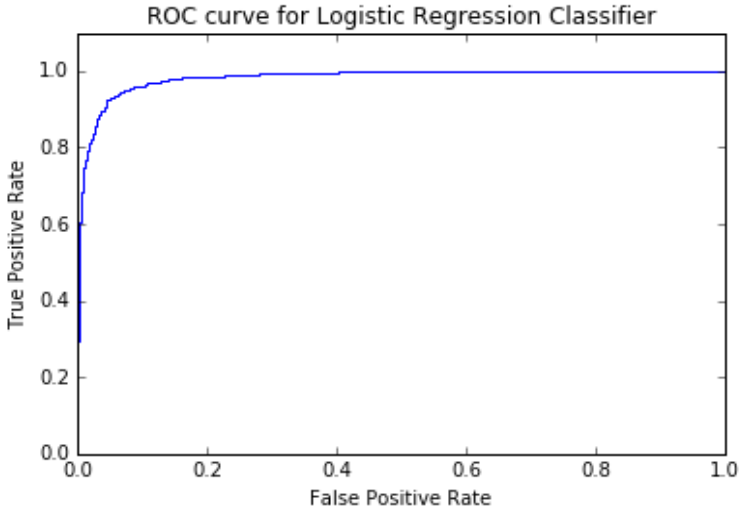

Figure 24: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier

Interpretation By comparing the logistic regression classifier's ROC curves for the cases of LSI and NMF, we find that the LSI performs better than NMF, since the area under the curve is more for the case of LSI.

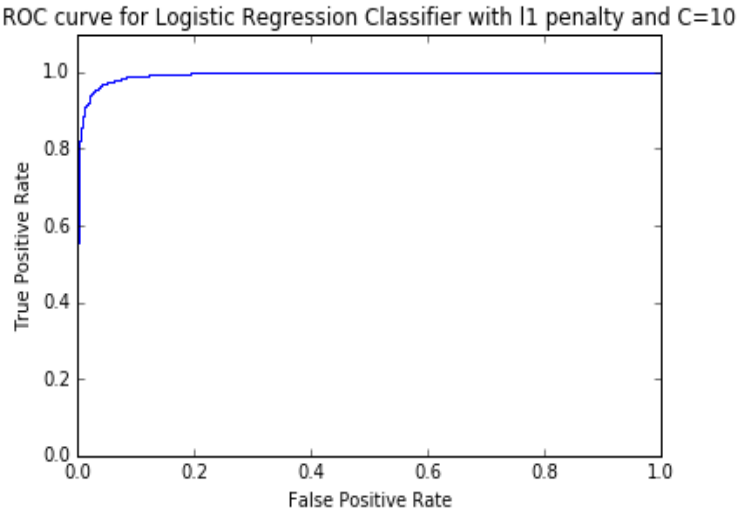

Figure 25: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l1 regularization with a coefficient of 10

Interpretation By adding a regularization term, the logistic regression classifier performs better which can be seen by comparing this ROC curve with the ROC curve above.

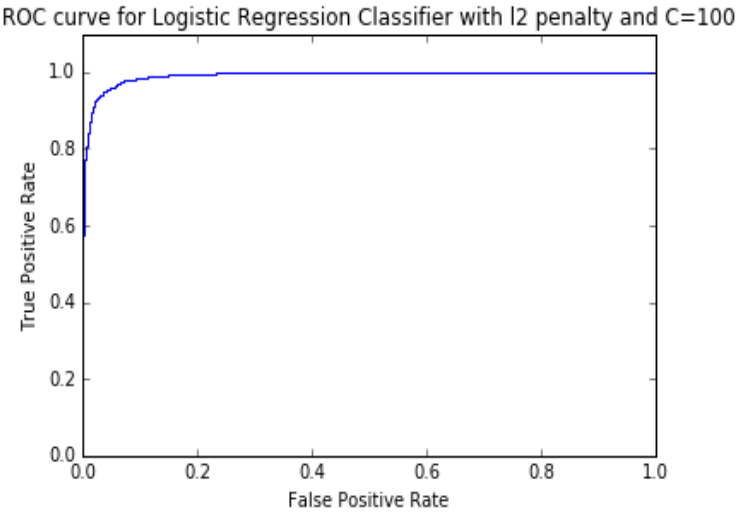

Figure 26: Receiver Operating Characteristics (ROC) curve for Logistic Regression classifier with l2 regularization with a coefficient of 100

Interpretation Both the l1 regularization and l2 regularization perform well on the data and give good results. The area under the curve has increased after adding the regularization term and by optimizing the regularization coefficient.

13 Conclusion

In this project, we implemented text classification using several models and techniques. We learned to clean the text and convert them to term frequency matrix.

TF-IDF was used to give weights to the words based on the frequency of occurrence of the word in the document as compared to the corpus. Two different dimensionality reduction techniques, latent semantic indexing (LSI) and non-negative matrix factorization (NMF) were used to map the high dimensional matrix to lower dimensions. Several models such as SVM classifier, logistic regression classifier and Naive bayes classifier were implemented for the classification process. Hyperparameter tuning was performed to determine the best values of the hyperparameters. Several experiments were conducted and inferences were drawn from them. The accuracy, precision, recall and ROC curve were used to compare the different models and the effect of different parameters.

References

[1] http://scikit-learn.org/

[2] https://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html

[3] TA Discussion material and class notes