ECE 219
Large Scale Data Mining: Models and Algorithms
Prof. Vwani Roychowdhury
UCLA, Department of ECE
Spring 2021

Project 1: Classification Analysis on Textual Data
Report

Kenny Huynh, 805436804
Shu-Yun Ku, 405515493
Ryan Li, 704142455
Osama Hassen, 105644793

QUESTION 1

QUESTION 1: To get started, plot a histogram of the number of training documents for each of the 20 categories to check if they are evenly distributed.

<u>Answer:</u>

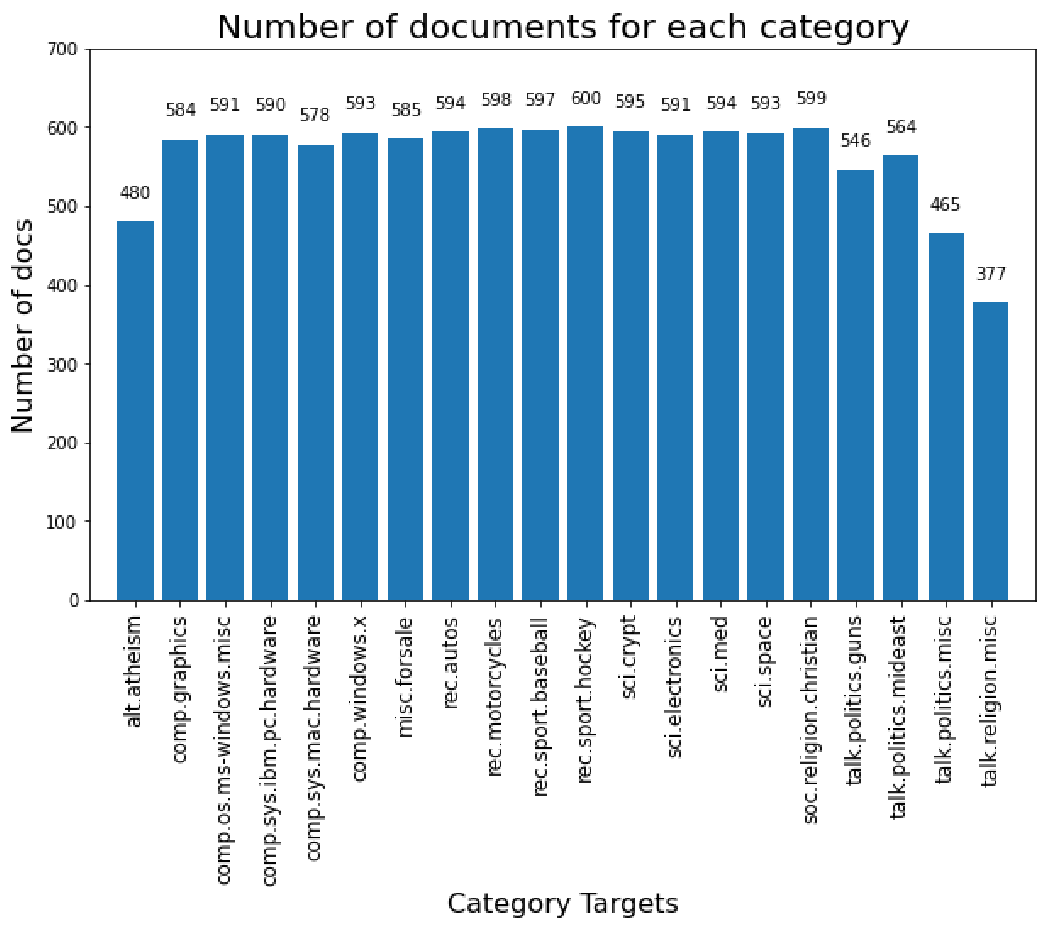


Figure 1: Number of Docs for each Category

The categories are generally evenly distributed with most having between 450 and 600 documents each. The only exception is category 19 (talk.religion.misc)

QUESTION 2

QUESTION 2: Use the following specs to extract features from the textual data:

- **Use the “english” stopwords of the CountVectorizer**
- **Exclude terms that are numbers (e.g. “123”, “-45”, “6.7” etc.)**
- **Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos tag**

- **Use min df=3**

Report the shape of the TF-IDF matrices of the train and test subsets respectively.

Answer:
In building the tf-idf matrix, we elected not to use max_df in CountVectorizer since most high-repeat words would be included in the stop words list. A custom stopword list was used that is a combination of 'english' from nltk and text.ENGLISH_STOP_WORDS from sklearn. This yielded a combined stop word list of 378 words (vs 318 from nltk's 'english'). Adding punctuations to the stop words was found not to affect the dimensions of the data sets so it was left off. Terms that numbers were also excluded per the spec. Original number of terms before lemmatizing is 20297.

The train dataset has dimensions of 4732 x 17389.
i.e. 4732 documents (as expected) and 17389 terms after lemmatizing and vectorizing with given parameters.

The test dataset has dimensions of 3150 x 17389.
i.e. 3150 documents (as expected) and 17389 terms (also as expected) after lemmatizing and vectorizing with given parameters.

It makes sense that the number of terms is the same in the 2 data sets as it is the same set of feature terms.

QUESTION 3

QUESTION 3: Reduce the dimensionality of the data using the methods above

- **Apply LSI to the TF-IDF matrix corresponding to the 8 categories with k=50; so each document is mapped to a 50-dimensional vector.**
- **Also reduce dimensionality through NMF (k=50) and compare with LSI:**

  Which one is larger, the in $||X-WH||^2_F$ NMF or the $||X - U_k\Sigma_k V^T_k||^2_F$ in LSI? Why is the case?

Answer:

Dimensionality Reduction using both LSI (SVD) and NMF yielded the following matrix shapes:

Train Matrix shape: 4732 x 50

Test Matrix shape: 3150 x 50

NMF error = 4154.33

LSI error = 4115.72

NMF error $||X-WH||^2_F$ is larger than the LSI error $||X - U_k\Sigma_k V^T_k||^2_F$. The added constraints in NMF dimensionality reduction of non-negative W & H restrict the search domain. LSI, having no such restrictions, can better minimize the error.

QUESTION 4

QUESTION 4: Hard margin and soft margin linear SVMs:

- **Train two linear SVMs and compare:**
  - **Train one SVM with γ = 1000 (hard margin), another with γ = 0.0001 (soft margin).**
  - **Plot the ROC curve, report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of both SVM classifier. Which one performs better?**
  - **What happens for the soft margin SVM? Why is the case?**
    - **Does the ROC curve of the soft margin SVM look good? Does this conflict with other metrics?**
- **Use cross-validation to choose γ (use average validation accuracy to compare):**
  - **Using a 5-fold cross-validation, find the best value of the parameter γ in the range {10k|−3 ≤ k ≤ 3, k ∈ Z}. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this best SVM.**

Answer:

Hard Margin Linear SVM:

Scores on Test Data: Hard Margin Linear SVM	
Accuracy Score	0.9762
Recall Score	0.9881
Precision Score	0.9656
F1 Score	0.9767

Table 1: Hard Margin Linear SVM Score Table

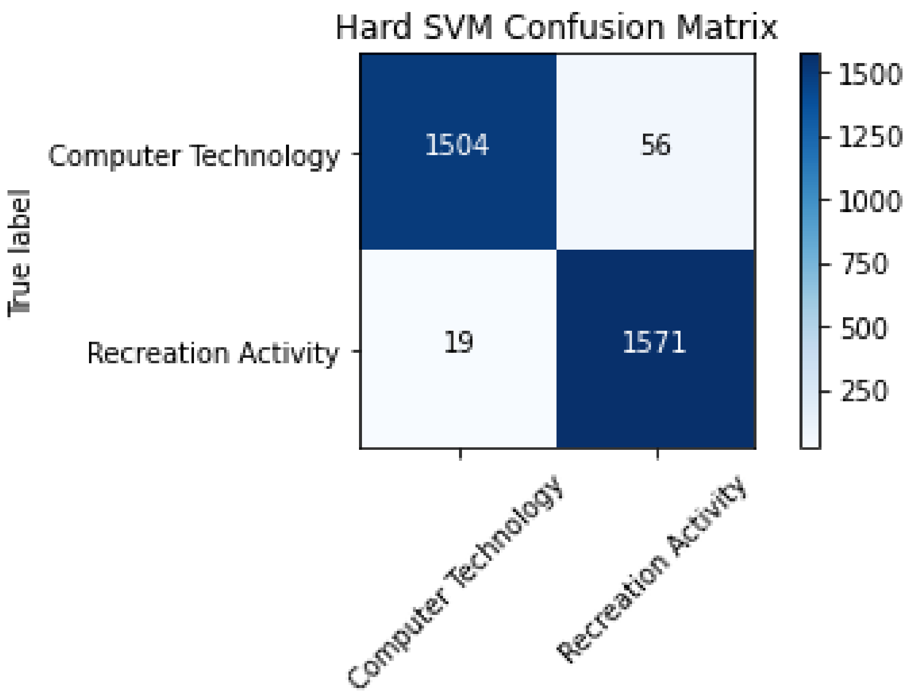


Figure 2: Hard SVM Confusion Matrix

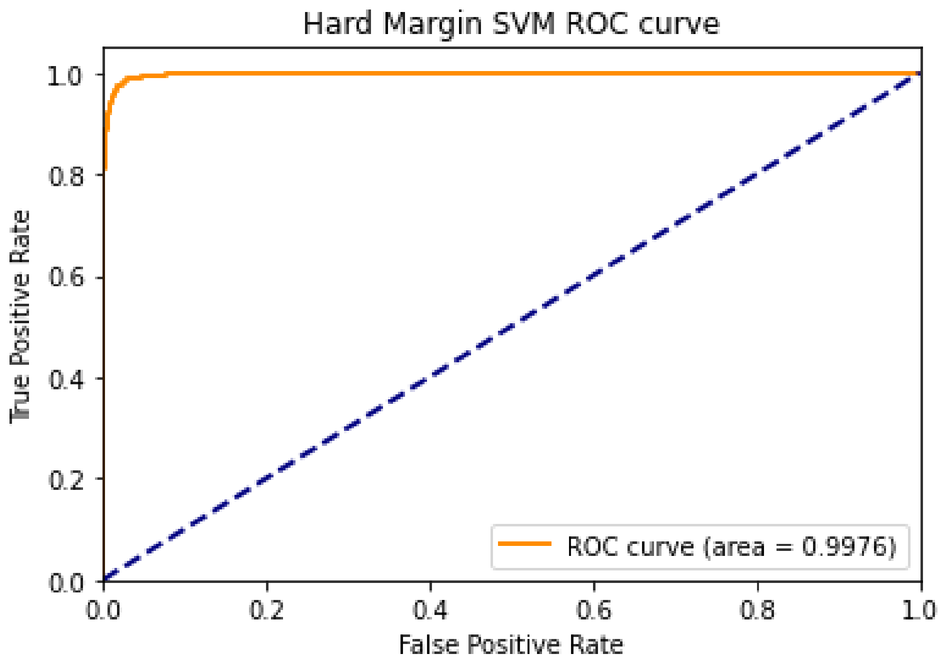


Figure 3: Hard Margin SVM ROC Curve

Soft Margin Linear SVM:

Soft Margin Linear SVM	
Accuracy Score	0.7444
Recall Score	1.0000
Precision Score	0.6639
F1 Score	0.7980

Table 2: Soft Margin Linear SVM Score Table

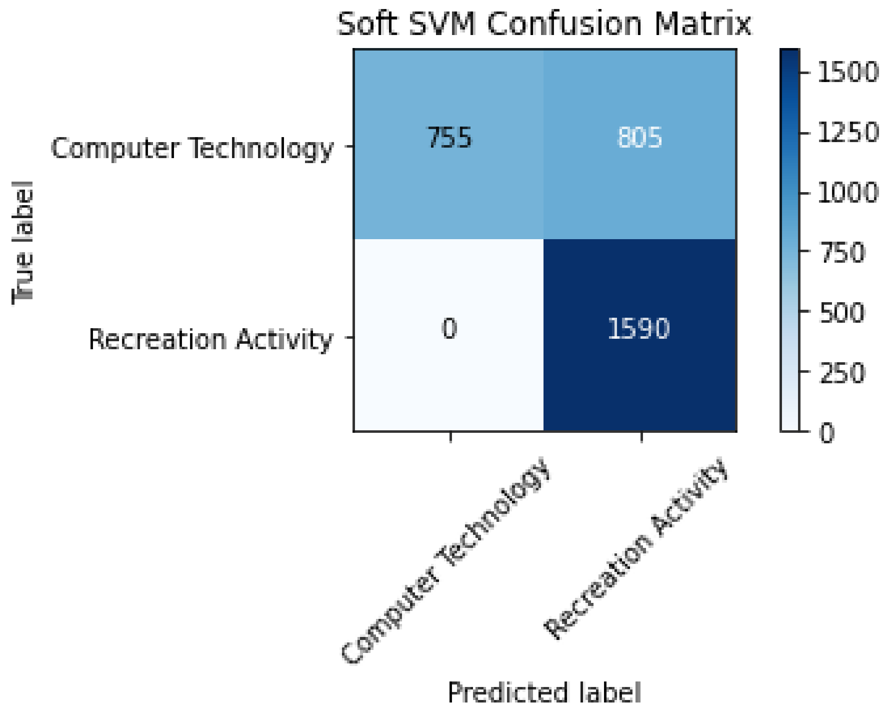


Figure 4: Hard SVM Confusion Matrix

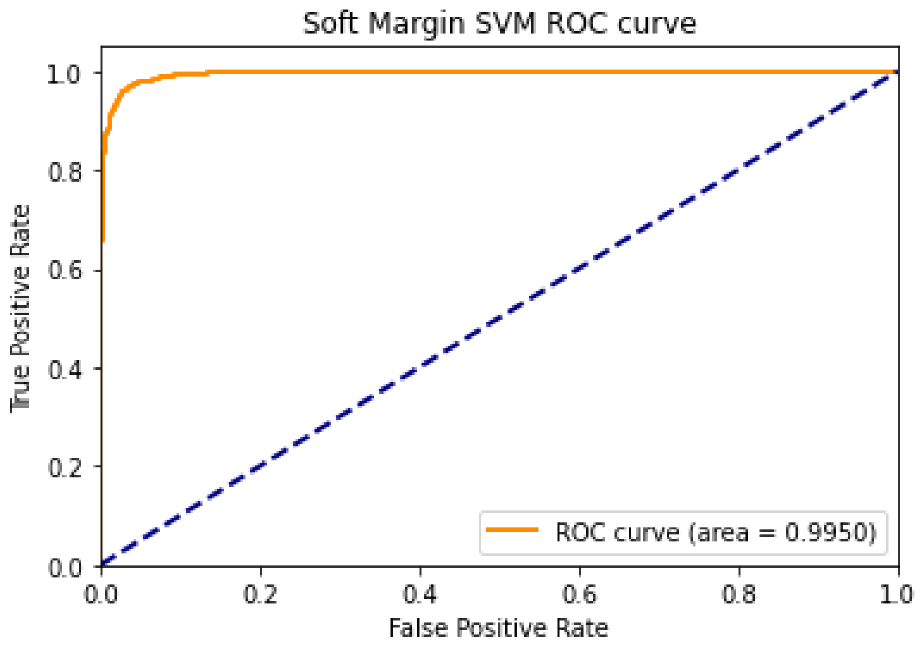


Figure 5: Soft Margin SVM ROC Curve

As you can see, the Hard Margin SVM performs better than the Soft Margin SVM. In addition, the ROC curve of Soft SVM and recall score looks good. However, this conflicts with the other metrics which are in the ranges of 0.6-0.75 and this might be due to the soft margin SVM being a separate hyperplane where the intercept b is not adjusted properly. In result, the SVM may have found the optimal w vector but not the b term, meaning that the margin may have not been fully maximized.

Next, a 5 fold cross validation was ran and it seems like the best value for $\gamma = 100$.The best accuracy score achieved by a linear SVM was 0.977175055979604

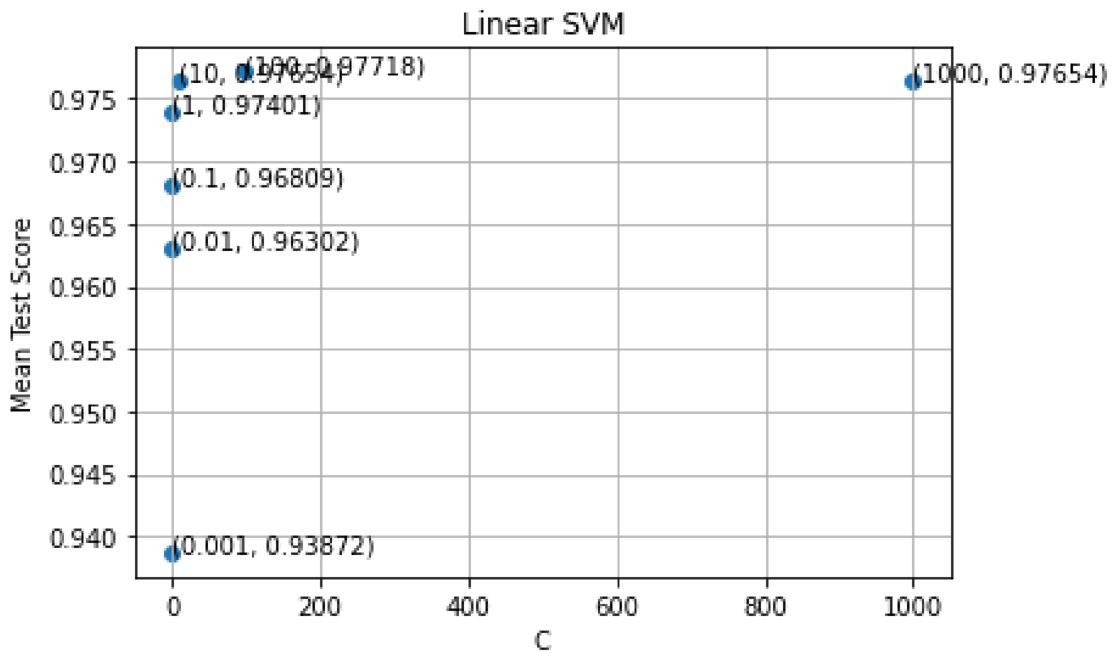


Figure 6: Linear SVM

Best Gamma Linear SVM, $\gamma = 100$	
Accuracy Score	0.9768
Recall Score	0.9893
Precision Score	0.9656
F1 Score	0.9773

Table 3: Best Gamma Linear SVM Score Table

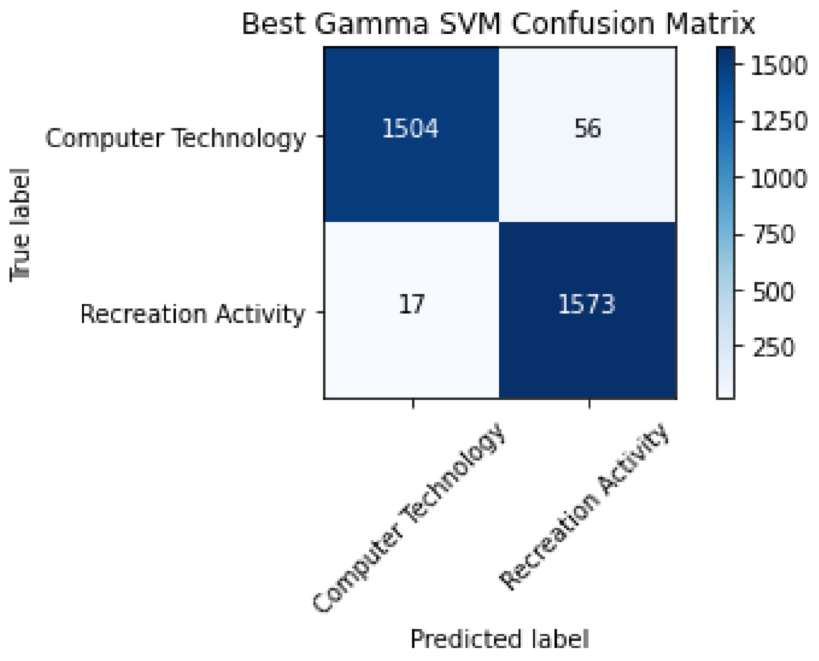


Figure 7: Best Gamma SVM Confusion Matrix

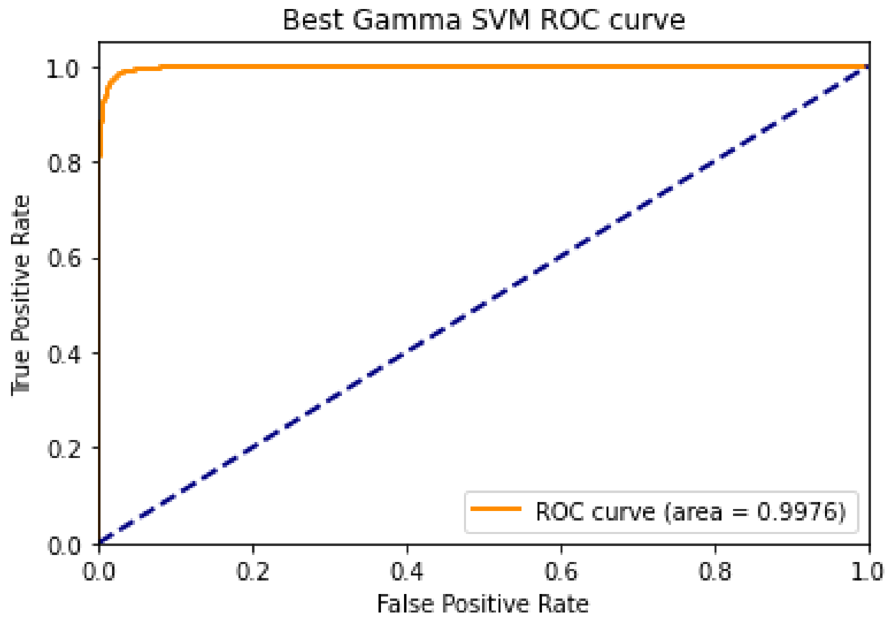


Figure 8: Best Gamma SVM ROC Curve

QUESTION 5

QUESTION 5: Logistic classifier

- **Train a logistic classifier without regularization (you may need to come up with some way to approximate this if you use sklearn.linear model.LogisticRegression); plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this classifier.**
- **Regularization:**
    - **Using 5-fold cross-validation on the dimension-reduced-by-svd training data, find the best regularization strength in the range {10k | − 3 ≤ k ≤ 3, k ∈ Z} for logistic regression with L1 regularization and logistic regression L2 regularization, respectively.**
    - **Compare the performance (accuracy, precision, recall and F-1 score) of 3 logistic classi- fiers: w/o regularization, w/ L1 regularization and w/ L2 regularization (with the best parameters you found from the part above), using test data.**
    - **How does the regularization parameter affect the test error? How are the learnt coeffi- cients affected? Why might one be interested in each type of regularization?**
    - **Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary, then what’s the difference between their ways to find this boundary? Why their performance differ?**

Answer:

Logistic Regression without Regularization:

Scores on Test Data: Logistic Regression without Regularization	
Accuracy Score	0.9771
Recall Score	0.9881
Precision Score	0.9674
F1 Score	0.9776

Table 4: Logistic Regression without Regularization Table

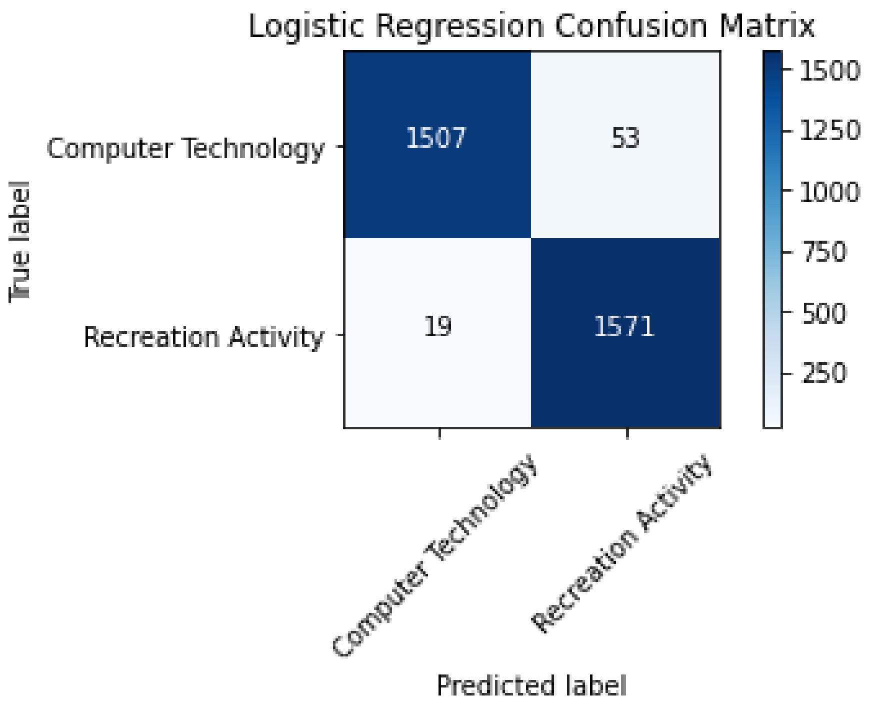


Figure 9: Logistic Regression Confusion Matrix

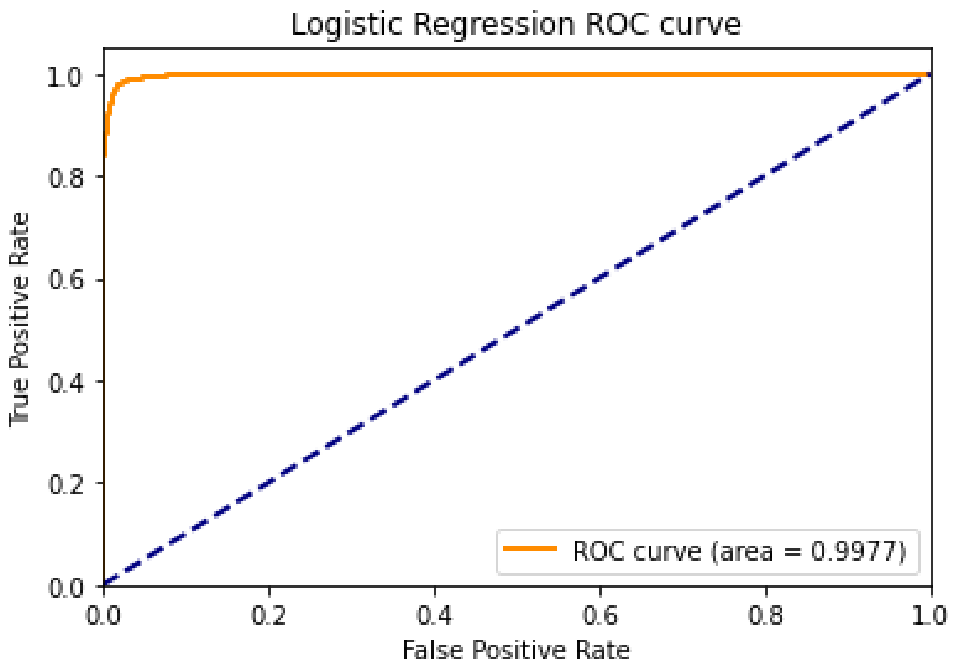


Figure 10: Logistic Regression ROC Curve

Logistic Regression with L1 Regularization:

Next, a 5 fold cross validation with L1 regularization was ran and it seems like the best value for $\gamma = 10$.The best accuracy score achieved during the 5-fold cross validation on training data was 0.977175055979604.

Scores on Test Data: Logistic Regression with L1 Regularization with best gamma, $\gamma = 10$	
Accuracy Score	0.9778
Recall Score	0.9887
Precision Score	0.9679
F1 Score	0.9782

Table 5: Logistic Regression with L1 Regularization Table

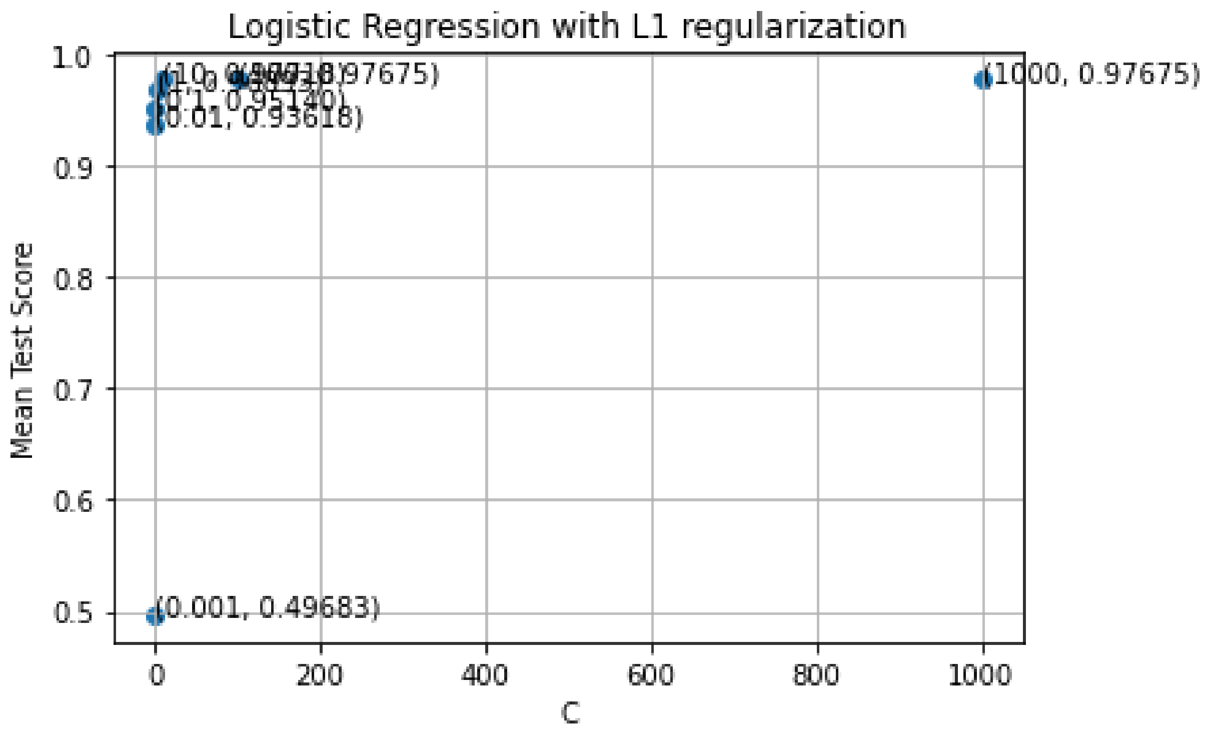


Figure 11: Logistic Regression with L1 Regularization ___

Logistic Regression with L2 Regularization:

Next, a 5 fold cross validation was ran with L2 Regularization and it seems like the best value for $\gamma = 100$.The best accuracy score achieved during the 5-fold cross validation on training data was 0.9769636394891178.

Scores on Test Data: Logistic Regression with L2 Regularization with best gamma, $\gamma = 100$	
Accuracy Score	0.9759
Recall Score	0.9874
Precision Score	0.9656
F1 Score	0.9764

Table 6: Logistic Regression with L2 Regularization Table

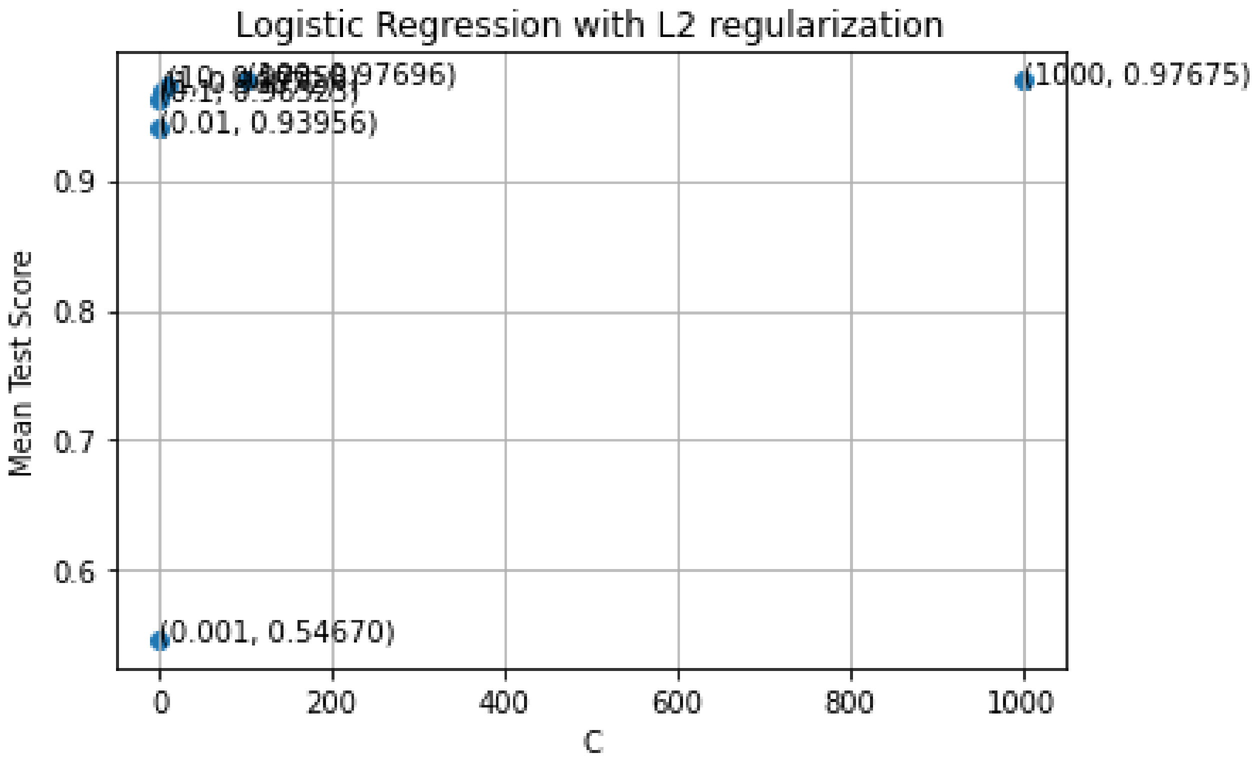


Figure 12: Logistic Regression with L2 Regularization

As you can see from the results, when comparing between the three Logistic Regression, Logistic Regression with L2 Regularization performs better because it has the highest accuracy and F1 score. Next, Logistic Regression without any regularization did the worse due the scores being the lowest in all categories. With that being said, Logistic Regression with L1 did the second best.

The regularization parameter decreased the test error in both L1 and L2.

- No Regularization - the learned coefficients was large in the order of 10^1. This type of regularization is useful when there is the need to build a complex model to deal with complex data and with no signs of overfitting.
- L1 Regularization - the learned coefficients in this regularization includes some being 0 and with most being in the order of 10^0. This type of regularization is useful when building a sparse model by removing features and keeps the most significant features in the model.
- L2 Regularization - most of the learned coefficients are in the order of 10^0which is smaller than the “No regularization”. This type of regularization is most useful when the learned coefficients are wanted to be small so that the model is not mostly dependent on a couple of new features. In result, this model is not sensitive to small changes in the feature vectors and makes the model more stable.

When comparing between logistic regression and linear SVM classifying data points using a linear decision boundary, we must keep in mind that logistic regression is a probabilistic classifier and SVM is a deterministic classifier. Logistic regression uses all data points to find the decision boundary, whereas, SVM only uses a small subset of data points called support vectors to find the decision boundary. In addition, logistic regression finds a decision boundary without considering the position of the hyperplane. However, SVm tries to find the position of the hyperplane which maximizes the margin. In conclusion, SVM finds a better discriminating hyperplane than logistic regression and thus, SVM has the optimal and better performance than logistic regression.

QUESTION 6

QUESTION 6: Na¨ıve Bayes classifier: train a GaussianNB classifier; plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of this classifier.

<u>Answer:</u>

We trained a GaussianNB classifier on the data that had been pre-processed with LSI using n_components=50.

Scores on Test Data: Gaussian Naive Bayes	
Accuracy Score	0.8530
Recall Score	0.9774
Precision Score	0.7845
F1 Score	0.8703

Table 7: Gaussian Naive Bayes Score Table

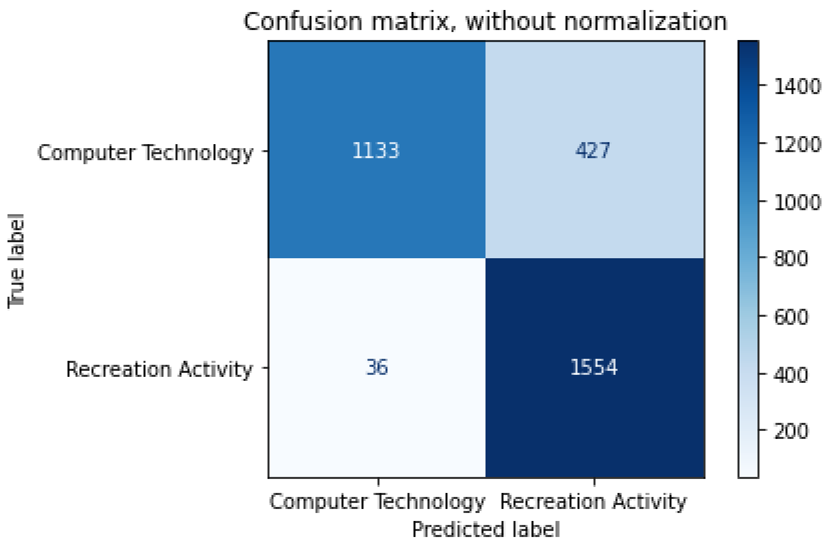


Figure 13: Gaussian NB Confusion Matrix (Without Normalization)

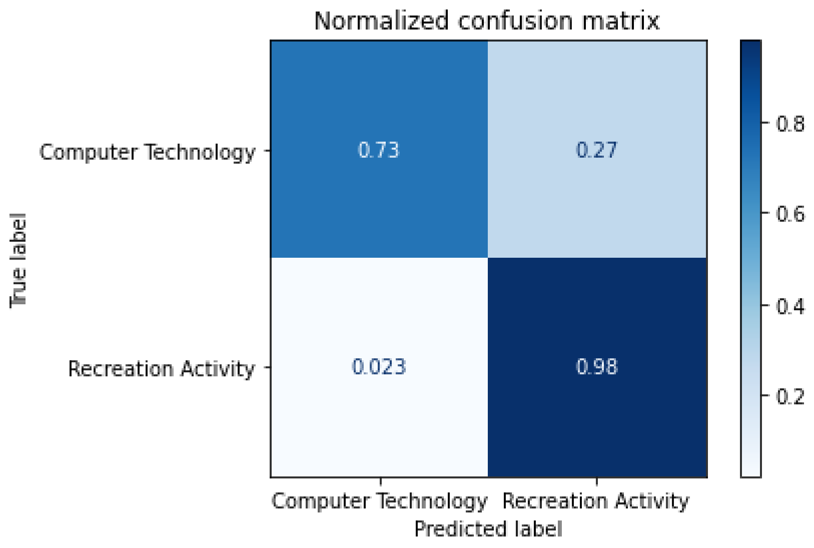


Figure 14: Gaussian NB Confusion Matrix (With Normalization)

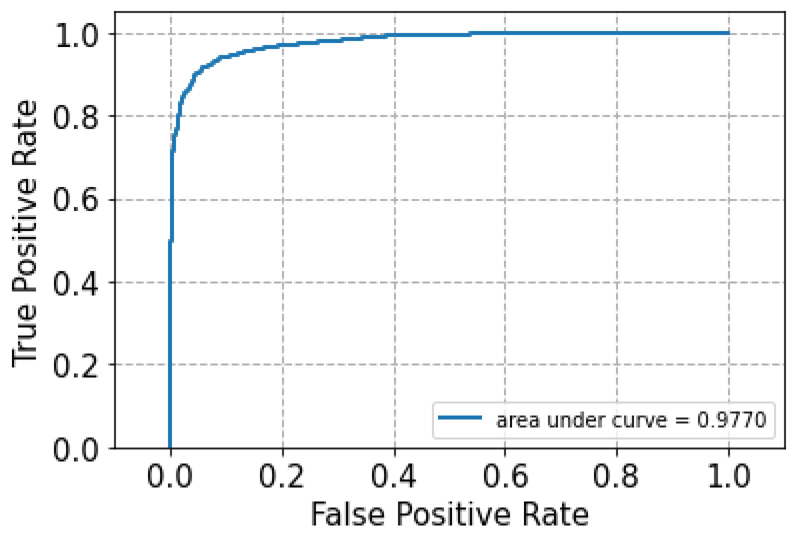


Figure 15: Gaussian NB ROC Curve

QUESTION 7

QUESTION 7: Grid Search of Parameters

- **Construct a Pipeline that performs feature extraction, dimensionality reduction and classification;**
- **Do grid search with 5-fold cross-validation to compare the following (use test accuracy as the score to compare):**

Procedure	**Options**
Loading Data	remove "headers" and "footers" vs not
Feature Extraction	`min_df = 3` vs 5; use lemmatization vs not
Dimensionality Reduction	LSI vs NMF
Classifier	SVM with the best γ previously found vs Logistic Regression: L1 regularization vs L2 regularization, with the best regularization strength previously found vs `GaussianNB`
Other options	Use default

- **What is the best combination?**

Answer:
We constructed a pipeline as described in the question:

- For loading data, there were 2 choices: removing headers and footers, not removing headers and footers
- In feature extraction, there were 2 sets of options, resulting in 4 total choices:
  - Lemmatization, no lemmatization
  - min_df = 3, min_df=5
- In dimensionality, there were 2 choices (both with n_components=50): LSI (TruncatedSVD), NMF
- In the classifier section, we had a total of 4 choices:
  - SVM (LinearSVC(C=100))
  - Logistic Regression - L1 regularization (LogisticRegression(penalty='l1',C=10))
  - Logistic Regression - L2 regularization (LogisticRegression(penalty='l2',C=100))
  - GaussianNB

Multiplying all these choices out, there are a total of 64 combinations. The results we found were as follows (The combinations are ranked by their mean_test_score as found during the 5-fold cross validation on training data):

Results With No Removal of Headers and Footers and No Lemmatization

min_df	dimensionality reduction	classifier	mean_test_score	rank_test_score
3	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9740	1
3	NMF(n_components=50)	LinearSVC(C=100)	0.9696	7
5	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9732	3
5	NMF(n_components=50)	LinearSVC(C=100)	0.9672	9
3	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9736	2
3	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9685	8
5	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9723	5
5	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9664	10
3	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9727	4
3	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9656	11
5	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9721	6
5	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9639	12
3	TruncatedSVD(n_components=50)	GaussianNB()	0.9104	16
3	NMF(n_components=50)	GaussianNB()	0.9444	13
5	TruncatedSVD(n_components=50)	GaussianNB()	0.9163	15
5	NMF(n_components=50)	GaussianNB()	0.9434	14

Table 8: Results With No Removal of Headers and Footers and No Lemmatization

Results WITH REMOVAL OF HEADERS AND FOOTERS and No Lemmatization

min_df	dimensionality reduction	classifier	mean_test_score	rank_test_score
3	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9715	3
3	NMF(n_components=50)	LinearSVC(C=100)	0.9681	7
5	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9704	5
5	NMF(n_components=50)	LinearSVC(C=100)	0.9645	10
3	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9717	1
3	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9677	8
5	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9710	4
5	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9649	9
3	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9704	5
3	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9613	11
5	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9717	1
5	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9582	12
3	TruncatedSVD(n_components=50)	GaussianNB()	0.8614	15
3	NMF(n_components=50)	GaussianNB()	0.9474	14
5	TruncatedSVD(n_components=50)	GaussianNB()	0.8512	16
5	NMF(n_components=50)	GaussianNB()	0.9522	13

Table 9: Results WITH REMOVAL OF HEADERS AND FOOTERS and No Lemmatization

Results With No Removal of Headers and Footers and LEMMATIZATION

min_df	dimensionality reduction	classifier	mean_test_score	rank_test_score
3	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9736	4
3	NMF(n_components=50)	LinearSVC(C=100)	0.9702	8
5	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9738	1
5	NMF(n_components=50)	LinearSVC(C=100)	0.9677	10
3	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9734	5
3	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9723	7
5	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9727	6
5	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9679	9
3	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9736	3
3	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9668	11

5	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9736	2
5	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9622	12
3	TruncatedSVD(n_components=50)	GaussianNB()	0.8929	16
3	NMF(n_components=50)	GaussianNB()	0.9480	13
5	TruncatedSVD(n_components=50)	GaussianNB()	0.8939	15
5	NMF(n_components=50)	GaussianNB()	0.9451	14

Table 10: Results With No Removal of Headers and Footers and LEMMATIZATION

Results WITH REMOVAL OF HEADERS AND FOOTERS and LEMMATIZATION

min_df	dimensionality reduction	classifier	mean_test_score	rank_test_score
3	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9683	5
3	NMF(n_components=50)	LinearSVC(C=100)	0.9645	9
5	TruncatedSVD(n_components=50)	LinearSVC(C=100)	0.9687	4
5	NMF(n_components=50)	LinearSVC(C=100)	0.9649	8
3	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9681	6
3	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9639	10
5	TruncatedSVD(n_components=50)	LogisticRegression(L1,C=10)	0.9702	1
5	NMF(n_components=50)	LogisticRegression(L1,C=10)	0.9653	7
3	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9696	2
3	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9582	11
5	TruncatedSVD(n_components=50)	LogisticRegression(L2,C=100)	0.9696	2
5	NMF(n_components=50)	LogisticRegression(L2,C=100)	0.9577	12
3	TruncatedSVD(n_components=50)	GaussianNB()	0.8375	16
3	NMF(n_components=50)	GaussianNB()	0.9413	13
5	TruncatedSVD(n_components=50)	GaussianNB()	0.8385	15
5	NMF(n_components=50)	GaussianNB()	0.9398	14

Table 11: Results WITH REMOVAL OF HEADERS AND FOOTERS and LEMMATIZATION

From the results of the previous 4 tables (Tables 8 - 11), we can see that depending on whether or not we choose to remove headers and footers and whether or not we choose to use lemmatization, the best choice of min_df and classifier to use can vary. However, in all cases, it appears that LSI (TruncatedSVD) performs better than NMF, which verifies what we discovered in Question 3.

It should be noted that the Gaussian Naive Bayes (GaussianNB) classifier performs the worst of all the classifiers in all cases. This makes sense because GaussianNB is perhaps the simplest out of all these classifiers with the least number of parameters to tune. Additionally, GaussianNB is a probabilistic model as opposed to one that tries to find an absolute solution like the other classifiers do.

Excluding GaussianNB, the performances of all the other classifiers in cross validation are all quite similar. Given that the difference in mean_test_score on training data is so small between so many combinations, it would not be that surprising if combinations that had a slightly lower score in the tables above performed slightly better when predicting on actual test data.

Now, we will compare the top performers from each of the above 4 tables. Since there was an exact tie in the second case (Table 9), we will pick the combination that is listed higher up on the table because this is the combination that GridSearchCV picks (Note that the combination that we did not pick is one that utilizes Logistic Regression with L2 Regularization, which is not represented below, but worked equally as well as the combination with Logistic Regression with L1 Regularization that we did pick). Looking at the mean_test_score obtained during the 5-fold cross validation on the training data, we see that the best performer is the combination that does not remove headers and footers, uses no lemmatization, uses min_df=3 in the Count Vectorizer, does dimensionality reduction with LSI (TruncatedSVD()), and uses SVM (LinearSVC(C=100)) as its classifier:

Combination	**mean_test_score**
(No removal of H&F, no lemmatization, min_df=3, TruncatedSVD(), LinearSVC(C=100))	0.9740
(Removal of H&F, no lemmatization, min_df=3, TruncatedSVD(), LogisticRegression(L1,C=10))	0.9717
(No removal of H&F, lemmatization, min_df=5, TruncatedSVD(), LinearSVC(C=100))	0.9738
(Removal of H&F, lemmatization, min_df=5, TruncatedSVD(), LogisticRegression(L1,C=10))	0.9702

Table 12: Comparison of Best Combinations Across Tables 8-11

However, evaluating these same 4 combinations by having them predict on actual test data, we see that another combination comes out on top, that is, the combination that does not remove the headers and footers, uses lemmatization, uses min_df=5 in the Count Vectorizer, does dimensionality reduction with LSI (TruncatedSVD()), and uses SVM (LinearSVC(C=100)) as its classifier:

Combination	**test_accuracy**
(No removal of H&F, no lemmatization, min_df=3, TruncatedSVD(), LinearSVC(C=100))	0.9705
(Removal of H&F, no lemmatization, min_df=3, TruncatedSVD(), LogisticRegression(L1,C=10))	0.9721
(No removal of H&F, lemmatization, min_df=5, TruncatedSVD(), LinearSVC(C=100))	0.9733
(Removal of H&F, lemmatization, min_df=5, TruncatedSVD(), LogisticRegression(L1,C=10))	0.9702

Table 13: Comparison of Best Combinations in Their Predictions on Actual Test Data

This change is not that unexpected since the mean_test_scores in the 5-fold cross validation on the training data were so close. Even the results in terms of test_accuracy on the actual test data are very close. It is totally within error and within the realm of possibility that a combination that performed a little better during cross validation on training data could perform a little worse than another combination when predicting on actual test data. Given this fact, it is possible that another combination outside of these 4 combinations could perform the best when predicting on actual test data, since the results in cross validation on training data were so close.

It should be noted that the first combination out of the top 4 had the biggest drop off from the cross validation on training data to the prediction on the actual test data. This suggests that lemmatization may help reduce variance, since lemmatization is the main difference between the first combination and the third combination. Also, in both the cross validation on training data and the prediction on actual test data, a combination that did not remove the headers and footers came out on top, suggesting that the content of the headers and footers may be helpful in classifying whether an email is in “Computer Technology” category or the “Recreational Activity” category.

QUESTION 8

QUESTION 8: Read the paper about GLoVE embeddings - found here and answer the following subquestions:

(a) Why are GLoVE embeddings trained on the ratio of co-occurrence probabilities rather than the probabilities themselves?

Answer:

By studying the ratio of their co-occurrence probability, the relationship between various word probes are considered. This way, it is better to distinguish relevant words from irrelevant words and it is also better able to discriminate between the two relevant words.

(b) In the two sentences: “James is running in the park.” and “James is running for the presidency.”, would GLoVE embeddings return the same vector for the word running in both cases? Why or why not?

Answer:

Yes, they will return the same vector. Because in GLoVE, it considers word-word co-occurrence matrices so that the distinction between a word and a context word is arbitrary. In this case, only the probability of occurrences to other neighboring words is taken into account.

(c) What do you expect for the values of,

$$\|GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]\|_2,$$
$$\|GLoVE["queen"] - GLoVE["king"]\|_2 \text{ and } \|GLoVE["wife"] - GLoVE["husband"]\|_2 \,?$$

Compare these values.

Answer:

Since the relationship between queen and king is similar to wife to husband, the value of $\|GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]\|_2$ should be zero (because it cancels out). The value of $\|GLoVE["queen"] - GLoVE["king"]\|_2$ and $\|GLoVE["wife"] - GLoVE["husband"]\|_2$ should be similar.

(**d) Given a word, would you rather stem or lemmatize the word before mapping it to its GLoVE embedding?**

Answer:

We would choose lemmatization over stemming before mapping to GLoVE embedding. Stemming is a way of vocabulary reduction where only a part of the vocabulary will be saved and the result might not be complete or meaningful, for example “cats” will be transformed to “cat” and “revival” to “reviv”. As for lemmatization, words will be transformed to its prototype, for example “drove” and “driving” will both be transformed to “drive”, and the result of lemmatization is always meaningful and complete, making the result more accurate.

QUESTION 9

QUESTION 9: For the binary classification task distinguishing the "Computer Technology" class and "Recreational Activity" class:
(a) Describe a feature engineering process that uses GLoVE word embeddings to represent each document. You have to abide by the following rules:

- **A representation of a text segment needs to have a vector dimension that CANNOT exceed the dimension of the GLoVE embedding used per word of the segment.**
- **You cannot use TF-IDF scores (or any measure that requires looking at the complete dataset) as a pre-processing routine.**
- **In each document, there are specific phrases that have more important topical words than others. They are highlighted by "Keywords: ..." or "Subject: ...". Use the average embedding of these words for each document representation.**
- **To aggregate these words into a single vector consider normalizing the final vectors.**

(b) Select a classifier model, train and evaluate it with your GLoVE-based feature.

Answer:

To do binary classification with GLoVE embedding, the process will be performed as below:

1. Data preparation such as feature extraction, dimensionality reduction and lemmatization.
2. Unnecessary words such as "From", "Reply-To", "Lines", and etc that may result in noises will be removed.
3. If the word exists in the embedding_dict, the GLoVE embedding for that word will be obtained, and sum all word vectors into a document vector.
4. Normalize the final vectors and use SVM as a classifier model, the result is shown in the below table.

SVM classifier with GLoVE embedding	
Accuracy Score	0.711746
Recall Score	0.711030
Precision Score	0.716157
F1 Score	0.733568

Table 14: SVM Classifier with GLoVE Embedding

QUESTION 10

QUESTION 10: Plot the relationship between the dimension of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task. Describe the observed trend. Is this trend expected? Why or why not?

Answer:

The accuracy increases as the dimension increases as expected and the increase in accuracies is more significant in dimensions lower than 200 . The reason for this is because when the dimension increases, the word can be categorized more precisely.

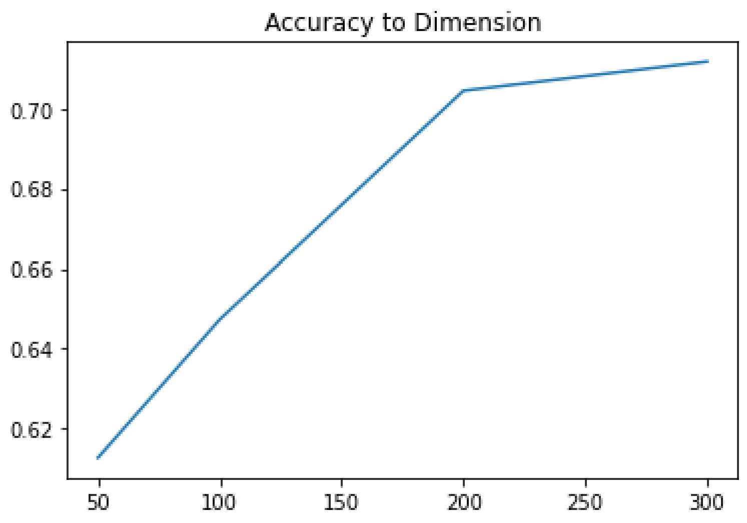


Figure 15: Accuracy vs. Dimension

QUESTION 11

QUESTION 11: Compare and contrast the two visualizations. Are there clusters formed in either or both of the plots?

Answer:

Clusters are formed in both plots. However the decision boundary seems to be clearer for GLoVE-based embeddings. In the plot of GLoVE-based embeddings, the upper-right is mostly purple which stands for label 1 whereas the red dots are clustered in the lower-left of the plot. Decision boundaries can be seen in plots of normalized random vectors too, but the distribution seems to be less in order as the first plot.

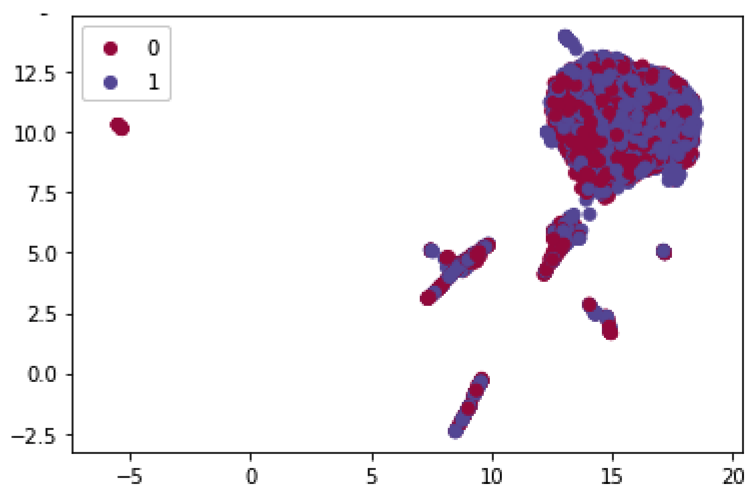
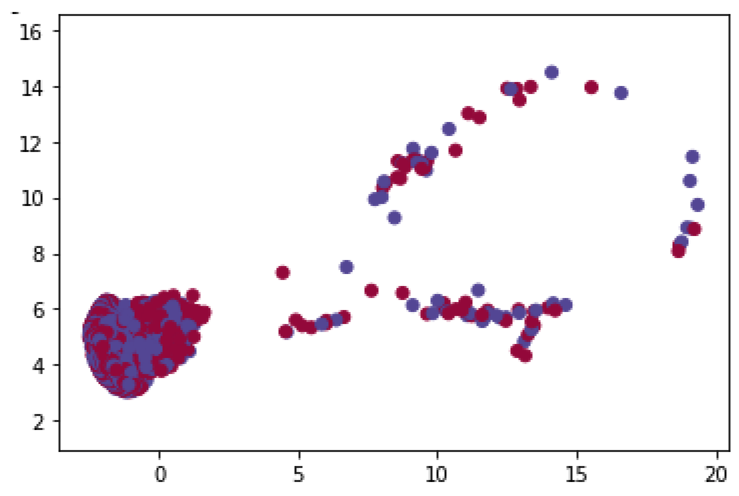
Normalized GLoVE-based embeddings	
Normalized random vectors	

Figure 16: Cluster Plots

QUESTION 12

QUESTION 12: Perform Na¨ıve Bayes classification and multiclass SVM classification (with both One VS One and One VS the rest methods described above) and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of your classifiers.

Answer:

LSI dimensionality reduction was chosen as it yields a lower error than NMF for this dataset. The following are the outcomes of Multiclass Gaussian Naïve Bayes, SVM One vs One and SVM One vs Rest classification methods.

Multiclass Gaussian Naïve Bayes

Multiclass Gaussian Naïve Bayes	
Accuracy Score	0.692013
Recall Score	0.692013
Precision Score	0.703850
F1 Score	0.685141

Table 15: Multiclass Gaussian Naïve Bayes Classification Table

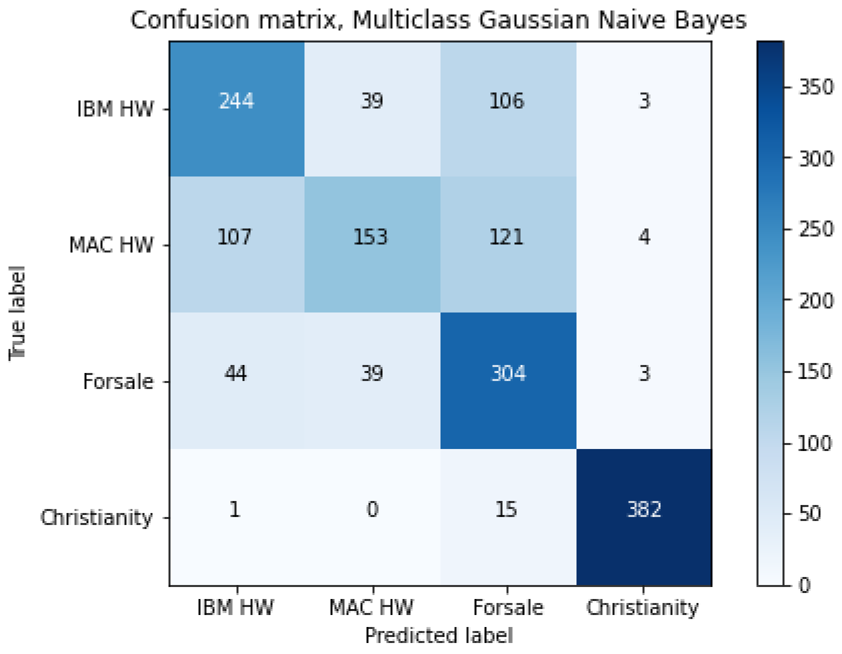


Figure 17: Confusion Matrix, Multiclass Gaussian Naive Bayes

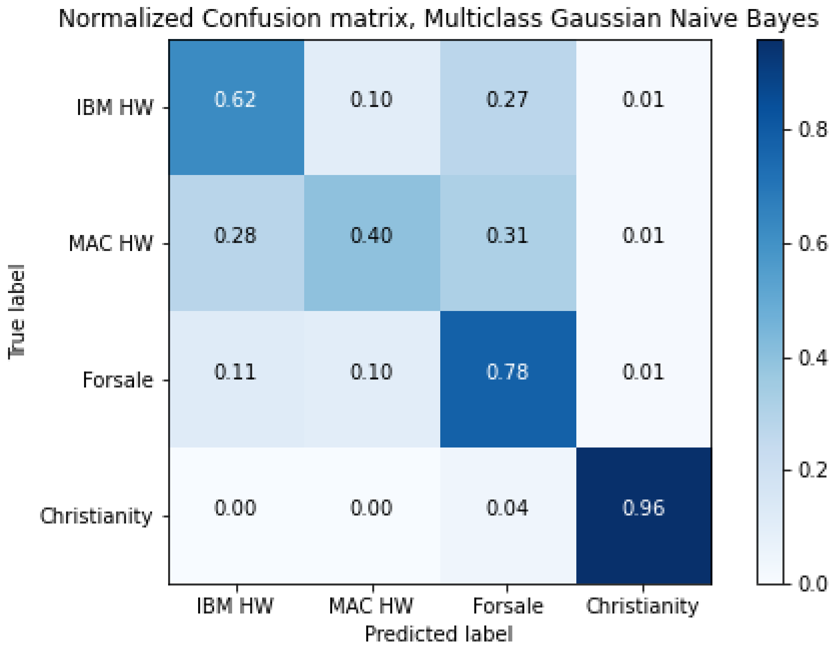


Figure 18: Normalized Confusion Matrix, Multiclass Gaussian Naive Bayes

One vs One SVM Classification

A five-fold grid search cross validation yielded the best gamma parameter as $\gamma = 10$.
It showed to be a better predictor than the Gaussian NB.

One vs One SVM Classification	
Accuracy Score	0.882428
Recall Score	0.882428
Precision Score	0.882771
F1 Score	0.882558

Table 16: One vs One SVM Classification Table

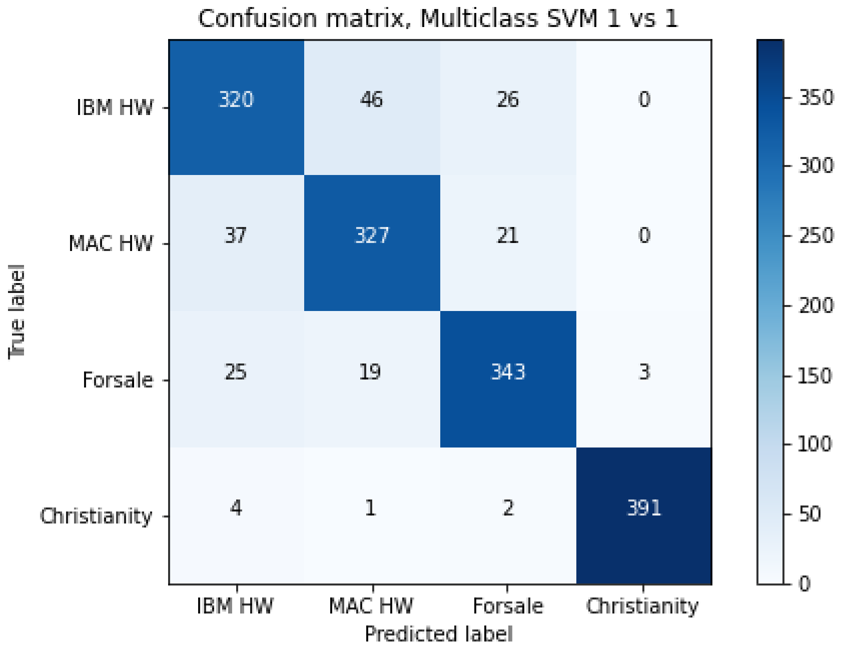


Figure 19: Confusion Matrix, Multiclass SVM 1 vs 1

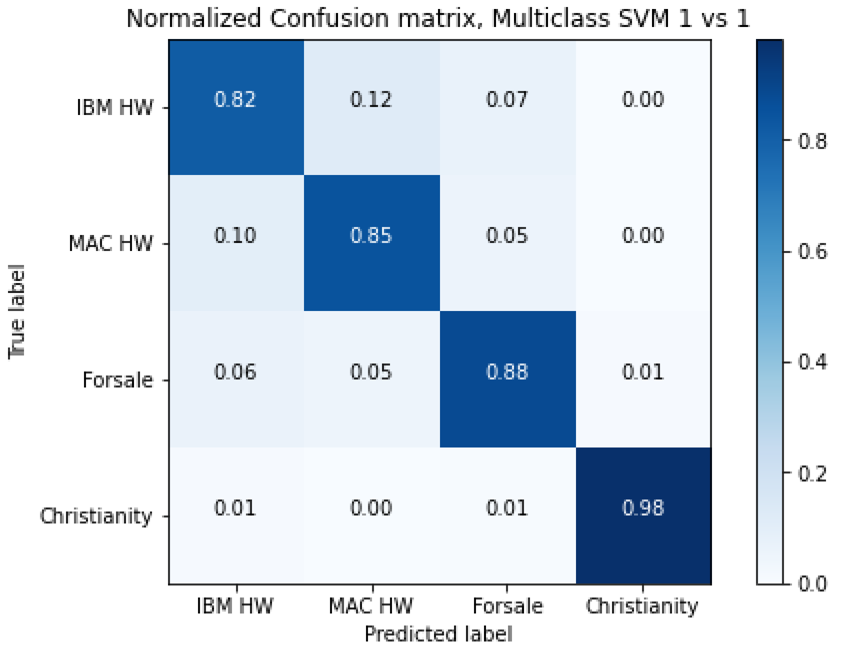


Figure 20: Normalized Confusion Matrix, Multiclass SVM 1 vs 1

One vs Rest SVM Classification

The best gamma parameter for the One vs Rest SVM classification also had a parameter of $\gamma = 10$. It showed an almost negligible improvement over the One vs One SVM.

One vs Rest SVM Classification	
Accuracy Score	0.884345
Recall Score	0.884345
Precision Score	0.884430
F1 Score	0.884098

Table 17: One vs Rest SVM Classification Table

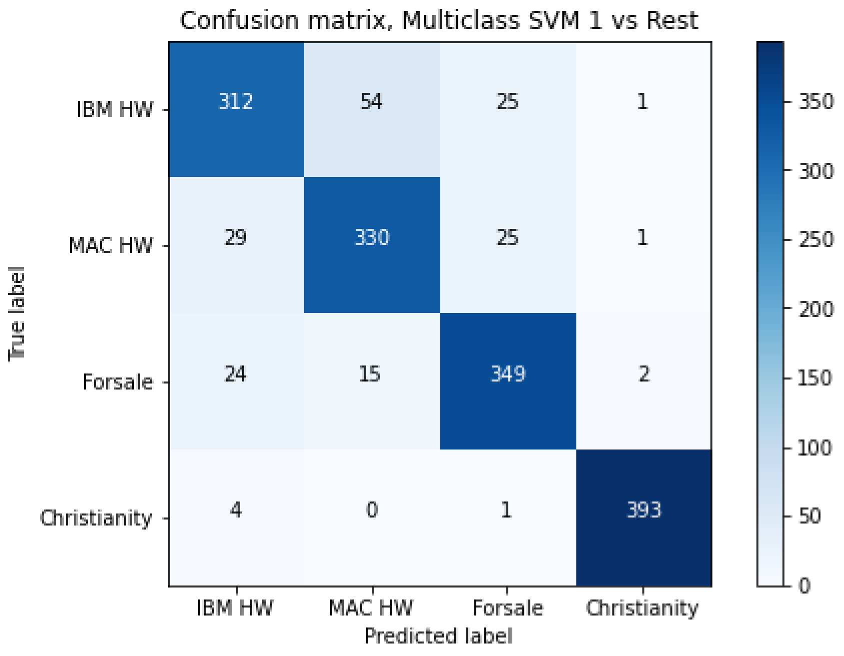


Figure 21: Confusion Matrix, Multiclass SVM 1 vs Rest

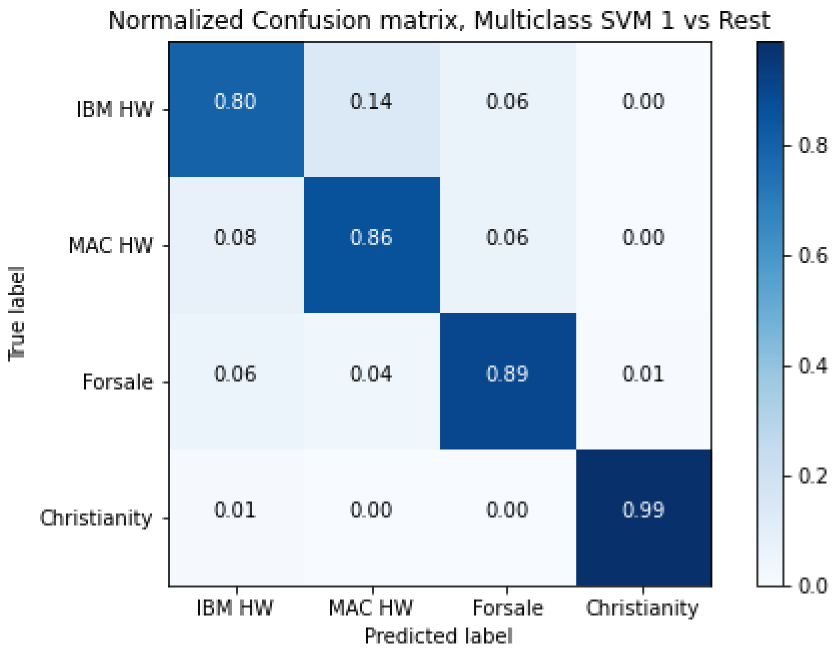


Figure 22: Normalized Confusion Matrix, Multiclass SVM 1 vs Rest