Project 1: Classification Analysis on Textual Data

Zeyu Bai, 904514671 Yi-Jui Chang, 804587644
Ruizhi Yang, 704514506 Nan Liu, 004434221

January 25, 2019

Introduction

In this project, a complete pipeline for classification problems is constructed, tested and discussed. Normalized and balanced data loaded from scikit-learn 20newsgroups is used for training and testing. Starting from text documents, feature extraction is done with TF-IDF matrix, then two methods(LSI, NMF) are applied to reduce matrix dimension. Multiple classifiers(SVM, logistic regression and naïve Bayes) are implemented to make predictions on document types. To evaluate these classifiers, featured figures(ROC plot, confusion matrix) together with metrics(accuracy, F-1 score etc.) are plotted and computed. Finally a grid search is done involving different parameters, dimension reduction methods and classifiers to find out optimal combination. Multiclass classification is also explored with naïve Bayes and SVM classifiers.

1 Feature Extraction

Question 1

By fetching all training document categories and plotting using histogram with 20 bins, we found in figure 1 that all 20 categories are roughly evenly balanced.

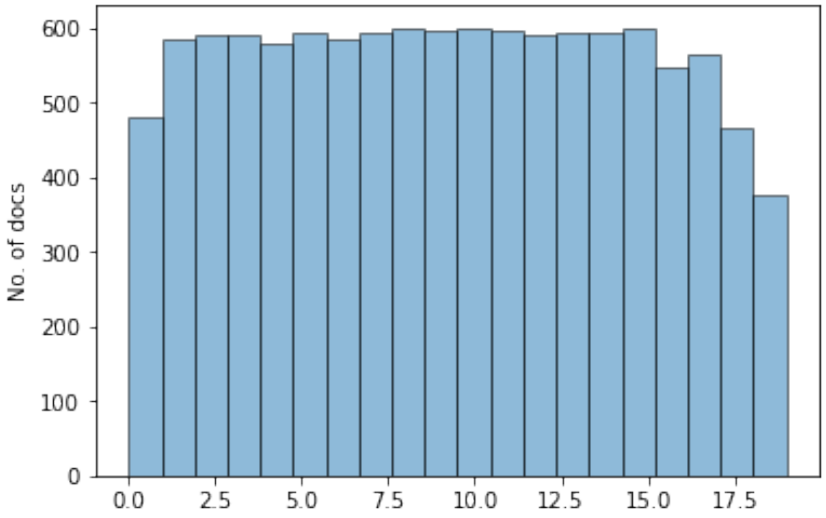


Figure 1: Histogram of training document number

Question 2

TF-IDF matrix size:

- train set: (4732, 16600)
- test set: (3150, 16600)

We constructed an analyzer by lowering words and lemmatizing based on part of speech tags, with pure numbers excluded. Stop words are chosen as "english" and minimum document frequency chosen as 3. We run the CountVectorizer first to get total counts and then apply the TF-IDF transformation. Note that the number of items in test set is the same as train set, which indicates a projection from test set to document-item space.

2 Dimensionality Reduction

Question 3

By importing functions of TruncatedSVD and NMF in sklearn.decomposition, we reduced the dimension of TF-IDF matrix of train dataset from size 4732×16600 to size 4732×50. To

measure the error from dimensionality reduction, we recovered the reduced matrix into the original document-item space through an inverse fitting function, and then computed the Frobenius norm between the recovered matrices and the original matrix corresponding to different methods as shown in table 1. The "error" from LSI is smaller than it from NMF, which meets our expectation since that in NMF, the matrices $\mathbf{W}$ and $\mathbf{H}$ have one strict requirement that each component inside matrices should be non-negative. On the other hand, the LSI kept the larger singular values but truncated the smaller ones, which means LSI actually preserved the significant features of the matrix.

Table 1: Frobenius Norm of error in dimensionality reduction

NMF	$\|\mathbf{X} - \mathbf{WH}\|_F^2 = 3940.3425$
LSI	$\left\|\mathbf{X} - \mathbf{U}_k\Sigma_k\mathbf{V}_k^T\right\|_F^2 = 3895.4187$

3 Classification Algorithms

Question 4: Linear SVM

In this question, we use linear SVM to classify the data into two categries: Computer Technology and Recreational Activity. In order to perform the binary classification, we first assign the Computer Technology documents with a "0" label, and the Recreational Activity documents with a "1" label. For part (i), we trained the data under two extreme cases: hard margin SVM with $\gamma = 1000$ and soft margin SVM with $\gamma = 0.0001$. Linear kernels are used in both cases. And in part (ii), we use a 5-fold cross-validation to choose the optimal γ and evaluate the model under best γ.

(i) In order to compare the performance of hard margin and soft margin linear SVMs. We plotted ROC curve under two cases in figure 2 and figure 3, and also reported the confusion matrices in figure 4 and figure 5. Further, we calculated the accuracy, recall, precision and F-1 score of both SVM classifiers (table 2).

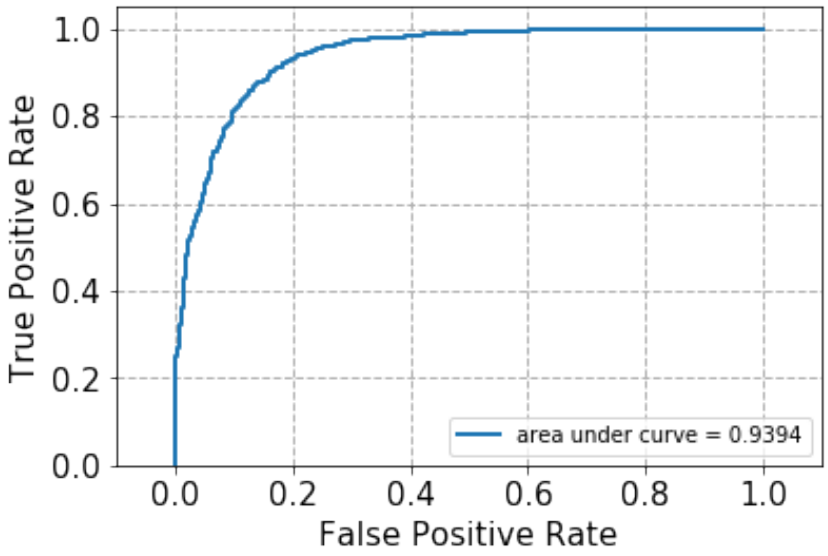


Figure 2: ROC curve of hard margin linear SVM classifier (γ=1000)

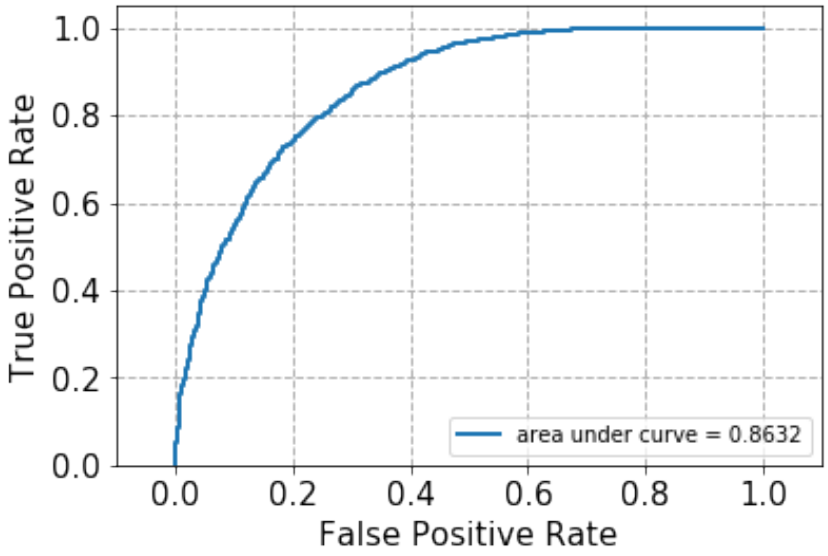
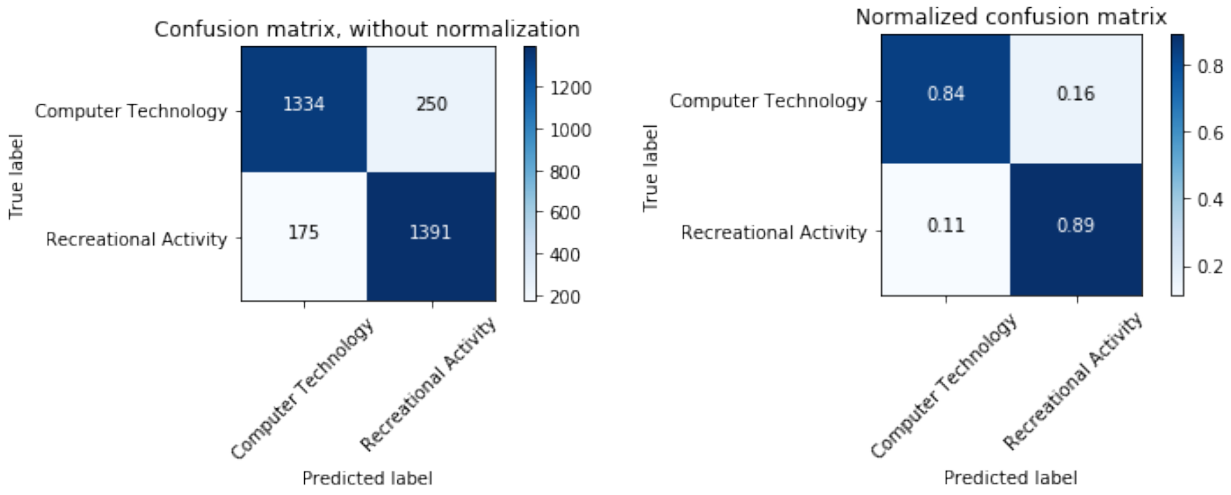
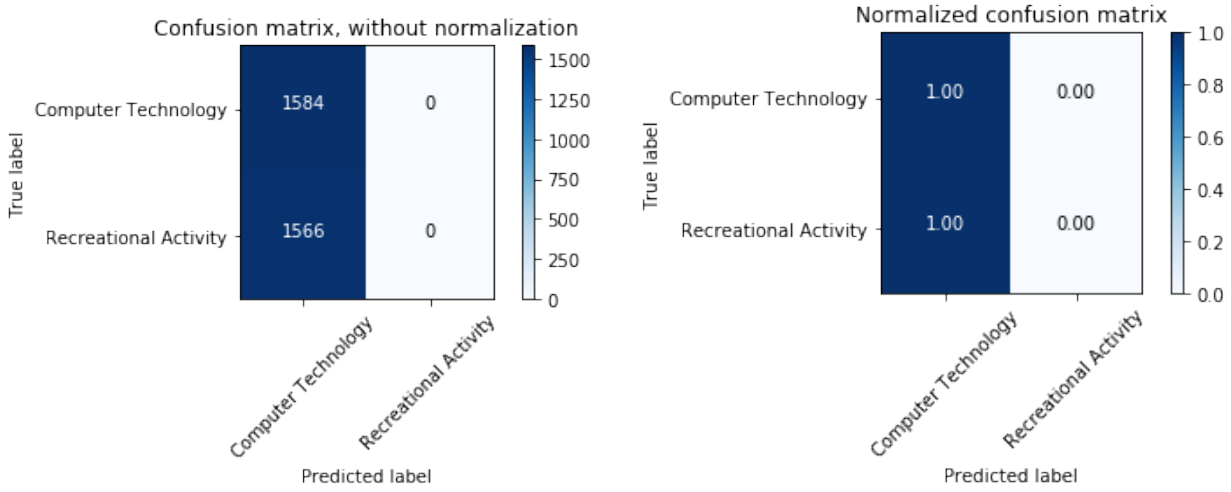


Figure 3: ROC curve of soft margin linear SVM classifier (γ=0.0001)



(a) (b)

Figure 4: Confusion matrix of hard margin linear SVM classifier (γ=1000) on test documents



(a) (b)

Figure 5: Confusion matrix of soft margin linear SVM classifier (γ=0.0001) on test documents

Table 2: Metrics of hard and soft SVMs on test documents

	Hard SVM	Soft SVM
accuracy	0.8651	0.5029
recall	0.8883	0.0
precision	0.8477	0.0
F-1 score	0.8675	0.0

From the results above, we learned that the hard margin performs better than the soft margin given on all the evaluation indicators.

Consider the confusion matrix in figure 5, the soft margin SVM classifies all the data points into the Computer Technology group - the negative class, which also explains why the recall, precision and F-1 score are all 0 for the soft SVM in table 2. The failure of the soft SVM is because that $\gamma = 0.0001$ is such a small restriction on the slack variable ϵ, thus surges the classification error.

The ROC curve of the soft margin SVM also does not look good and it is far from the upper left corner. This is consistent with the other metrics which all indicate the soft margin with $\gamma = 0.0001$ has poor performance.

(ii) In this part, we use the 5-fold cross-validation to find the best value of the parameter γ in the range $\{10^k \mid -3 \leq k \leq 3\}$. We experimented with 13 different values for γ including all the $\{10^k \mid -3 \leq k \leq 3,\ k \in Z\}$ and perform the grid search for these parameters. The result is presented in table 6.

By comparing the mean validation accuracy, we obtained the best γ equals to 33.0 under this case. Then, for this best SVM, we plotted the ROC curve and reported the confusion matrix and also calculated the accuracy, recall precision and F-1 score. The results are showed in the following figure 6, figure 7 and table 3.

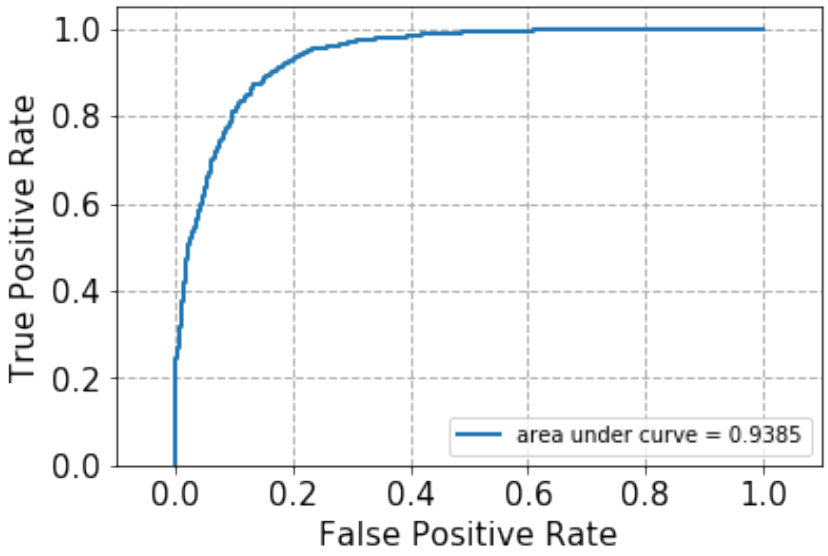
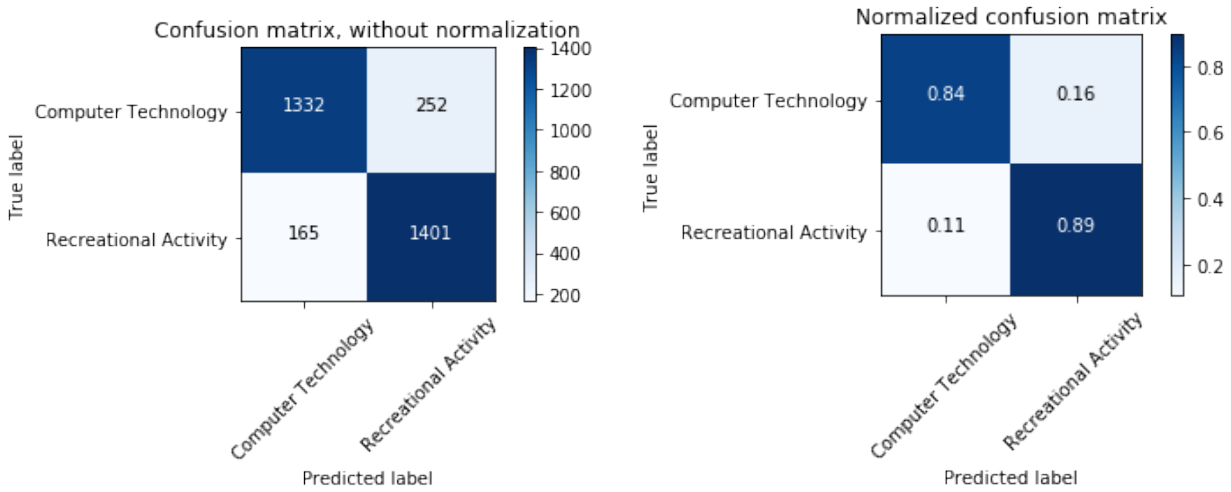


Figure 6: ROC curve of the best linear SVM classifier (γ=33)



(a) (b)

Figure 7: Confusion matrix of the best linear SVM classifier (γ=33) on test documents

Table 3: Metrics of the best SVM ($\gamma = 33.0$) on test documents

accuracy	0.8676
recall	0.8946
precision	0.8475
F-1 score	0.8705

Question 5: Logistic Classifier

In this question, we applied the logistic regression to classify the data into two groups. For part (i) and (ii), we considered the logistic classifier without regularization, and with L_1, L_2 regularization respectively. And then compared the performance of the aboved three logistic classifiers in part (iii). In part (iv), we discussed the effect of regularization parameters on test error and coefficients. And for part (v) we discussed the differences between logistic regression and the linear SVM classifier.

(i) To implement the logistic classifier in Python, we tried to use "LogisticRegression" from "sklearn.linear model" , but it doesn't have w/o regularization option. So we used the default L_2 regularization and make the inverse penalty $c = 10^4$ to approximate the w/o regularization case. For this classifier, we plotted the ROC curve in figure 8, reported the confusion matrix in figure 9, and also calculated the accuracy, recall precision and F-1 score which is showed in table 4.

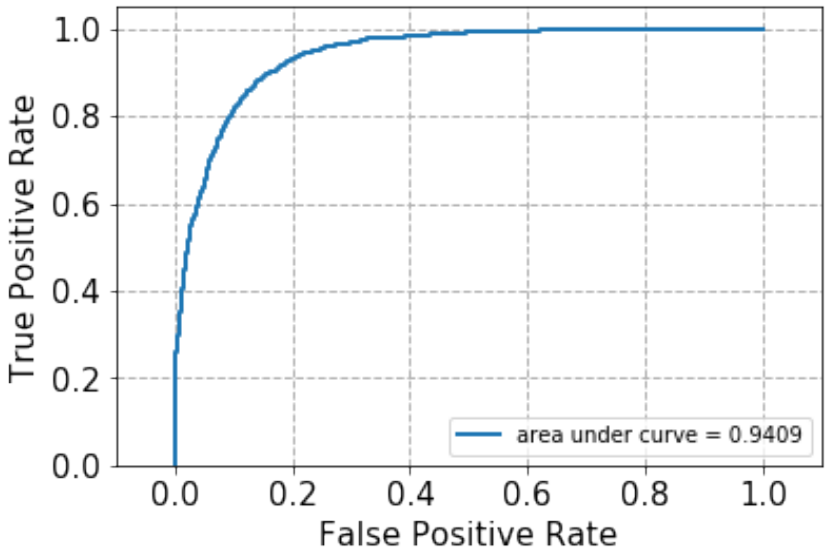
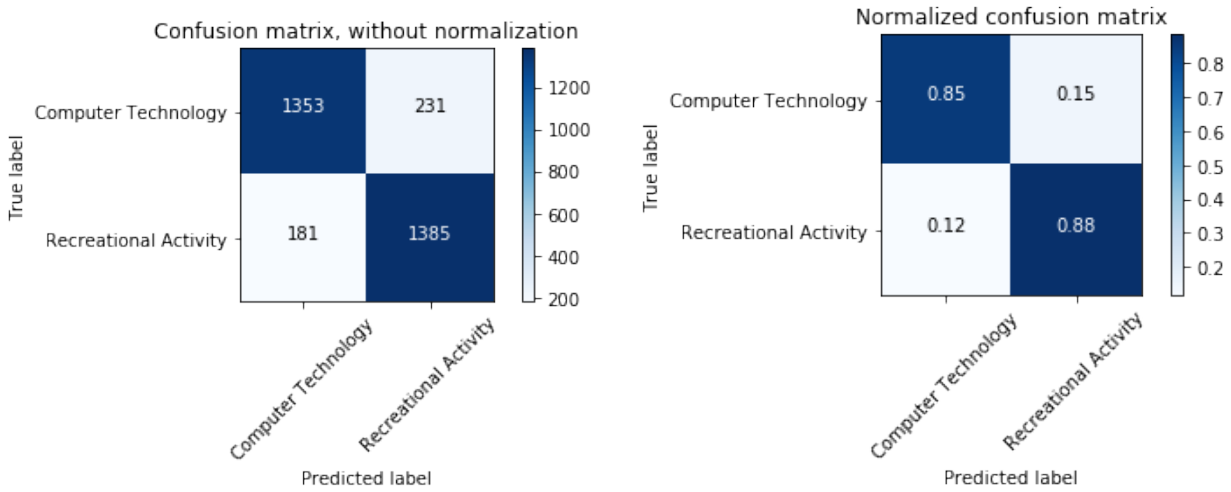


Figure 8: ROC curve of logistic regression classifier w/o regularization



(a) (b)

Figure 9: Confusion matrix of logistic regression classifier w/o regularization on test documents

Table 4: Metrics of the logistic regression classifier w/o regularization on test documents

accuracy	0.8692
recall	0.8844
precision	0.8571
F-1 score	0.8705

(ii) For logistic regression with L_1 and L_2 regularization, we need to choose the best regularization strength parameter c. Here we used the 5-fold cross-validation to search the regularization parameter c in the range of $\{10^k \mid -3 \leq k \leq 3\}$. The smaller the c is, the stronger regularization strength. We also used 13 different values including all the $k \in Z$ and perform the grid search for these parameters. The result is presented in table 6.

By comparing the mean test accuracy results for the 5-fold cross-validation, we obtained the best regularization parameters which are $c_1 = c_2 = 3.3$ for both the L_1 and L_2 regularizations. Then, we evaluate the performance of the classifiers with the best regularization parameter on the test dataset. We plotted the ROC curve and reported the confusion matrices (figure 10 and figure 12 for the L_1 penalty, and figure 11 and figure 13 for the L_2 penalty). Then we summarized the accuracy, recall precision and F-1 score of these two classifiers in table 5.

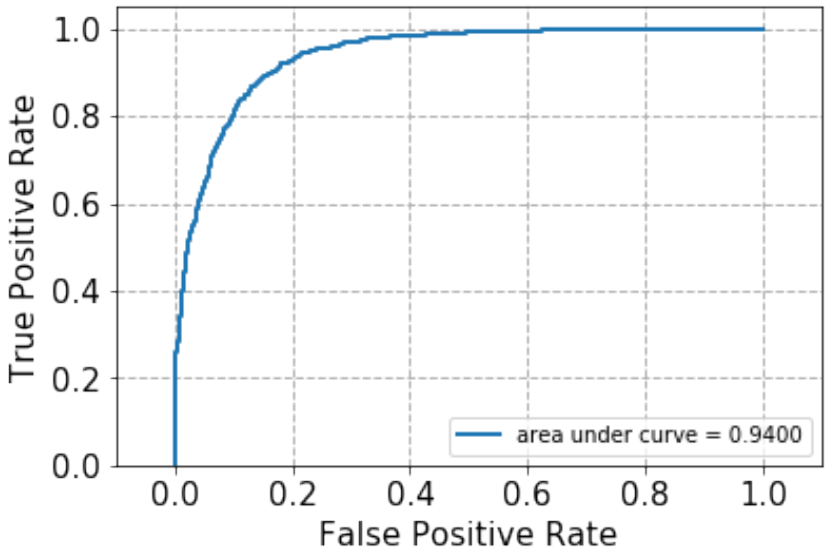


Figure 10: ROC curve of logistic regression classifier (L_1 regularization)

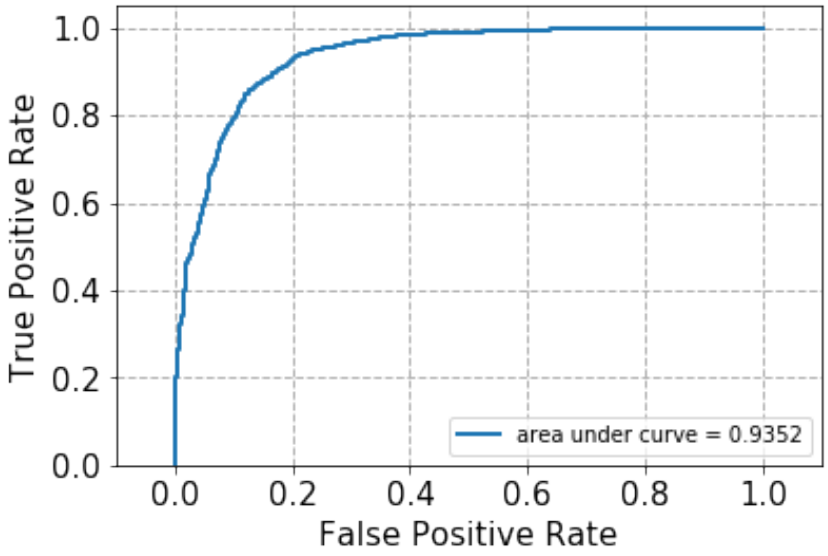
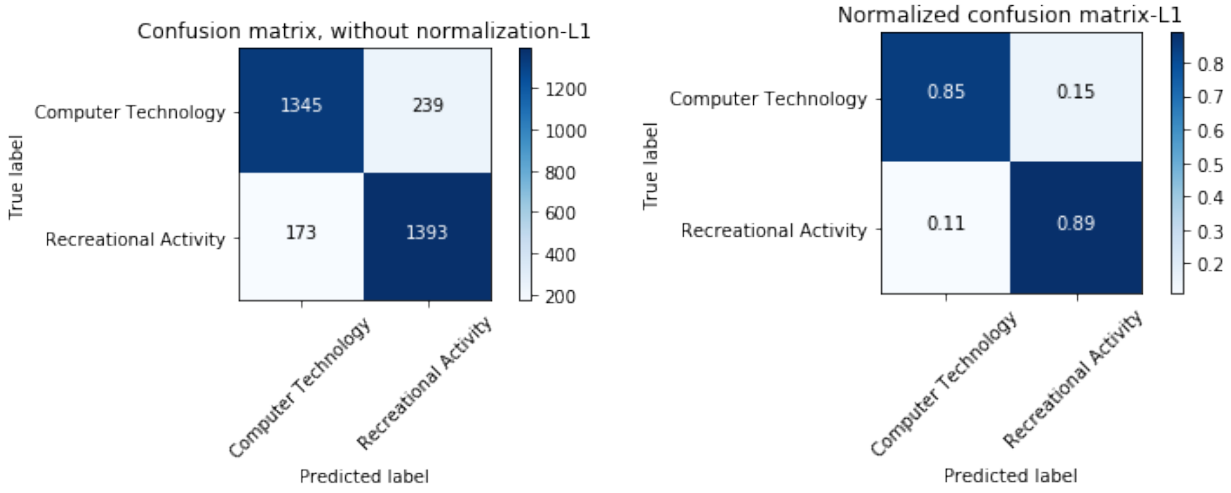
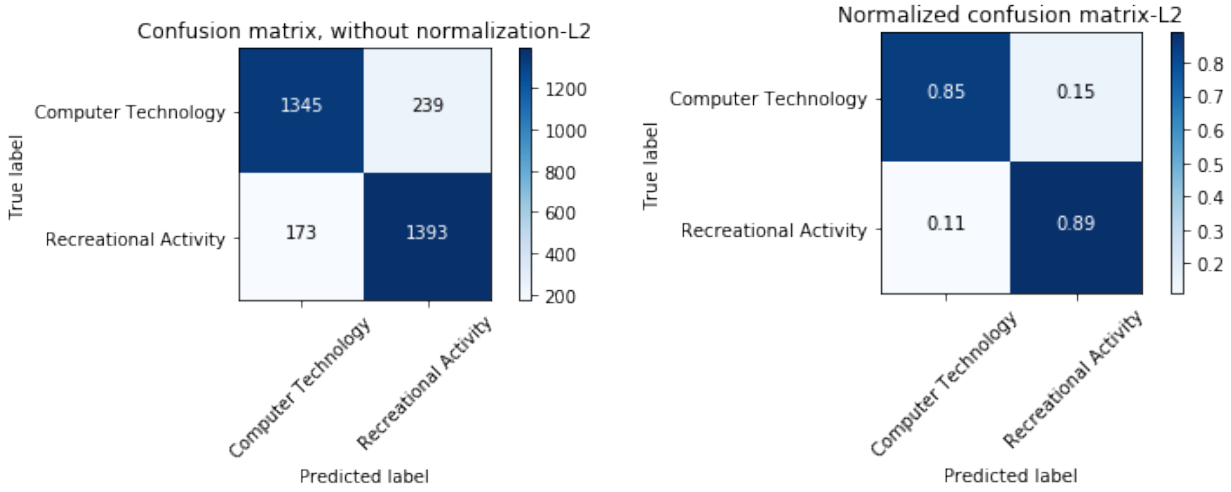


Figure 11: ROC curve of logistic regression classifier (L_2 regularization)



(a) (b)

Figure 12: Confusion matrix of logistic regression classifier (L_1 regularization) on test documents



(a) (b)

Figure 13: Confusion matrix of logistic regression classifier (L_2 regularization) on test documents

Table 5: Metrics of logistic regression with L_1 and L_2 regularization on test documents

	w/o regularization	L_1 regularization	L_2 regularization
accuracy	0.8692	0.8692	0.8641
recall	0.8844	0.8895	0.8959
precision	0.8571	0.8536	0.8411
F-1 score	0.8705	0.8712	0.8677

Table 6: Mean test score of 5-fold cross-validation for SVM and logistic regression

hyper parameter	SVM	Logistic Regression with L_1 regularization	Logistic Regression with L_2 regularization
0.001	0.50317	0.50317	0.57079
0.0033	0.50317	0.50317	0.75021
0.01	0.50317	0.50317	0.80896
0.033	0.75550	0.74937	0.82355
0.1	0.83412	0.81531	0.84172
0.33	0.86307	0.85503	0.86032
1	0.87638	0.87194	0.87342
3.3	0.87997	0.87722	0.87659
10	0.87870	0.87616	0.87637
33	0.88123	0.87489	0.87658
100	0.88102	0.87405	0.87532
330	0.88018	0.87468	0.87468
1000	0.87933	0.87468	0.87426

(iii) To compare the performance of the three logistic classfiers: w/o regularization, L_1 regularization and L_2 regularization, we consider the accuracy, precision, recall and F-1 score of the three models which are summarized in table 5.

In general, the logistic regression with L_1 regularization performs the best among the three models. However, the testing performance for w/o regularization is similar or even better

compared with the L_2 regularized model. The reason might be that we did dimension reduction using SVD in the first step. Thus adding L_2 penalty in the second step did not help too much to avoid overfitting, but only introduced extra estimation biases which increases the testing error.

(iv) The effect of regularization parameter on the test error should be U-shaped. For small values of c, the regularization strength is strong and the model underfits as it does not be complex enough. The estimation bias will be high, and thus we will have large testing error at small values of c. As c increases, the model complexity increases, it reduced the estimation bias thus test error decreases at first. But if the model becomes too complex, it also adds noise and overfitting occurs which leads to the increase in the test error after some point. Choosing the best value regularization parameter is a trade off between estimation bias and overfitting problem.

For the effect of regularization parameters on the learned coefficients, the L_1 penalty will drove some coefficients exactly to 0, and L_2 penalty will shrink all the coefficients toward 0. If we assume only few features in X are important, then we should add a L_1 penalty and it has the sparse outputs and can select some important features. And if we assume all the features have an effect on the Y variable, then we should use a L_2 penalty to shrink the coefficients in order to prevent the overfitting problem.

(v) Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary. But the logistic regression tries to maximize the probability of the data getting correclty classified. Thus all the data points matters for choosing the separating hyperplane (boundary). While the linear SVM tries to find the separating hyperplane by maximizing the distance between the margin of the closest support vectors which means only the data points on the support vectors matters.

The performance of the two methods depends on the properties of the dataset. Linear SVM is more robust to the outliers because it only depends on the data points on the support vectors. Thus if the dataset contains outliers, then SVM should perform better. And also SVM is itself regularized, and it should outperform the logistic regression for high dimensional data.

Question 6

For this question, the naïve Bayes classifier with the Gaussian probabilistic model describing the likelihood of each class is used for the binary classification problem. After fit the model on the training documents, it is then evaluated on the test documents. The ROC curve is shown in figure 14. Compared with the linear SVC or logistic regression classifier, the ROC curve shows that the Gaussian naïve Bayes classifier doesn't work well on the training documents because the ROC curve is far from the upper left corner. The area under curve is smaller as well.

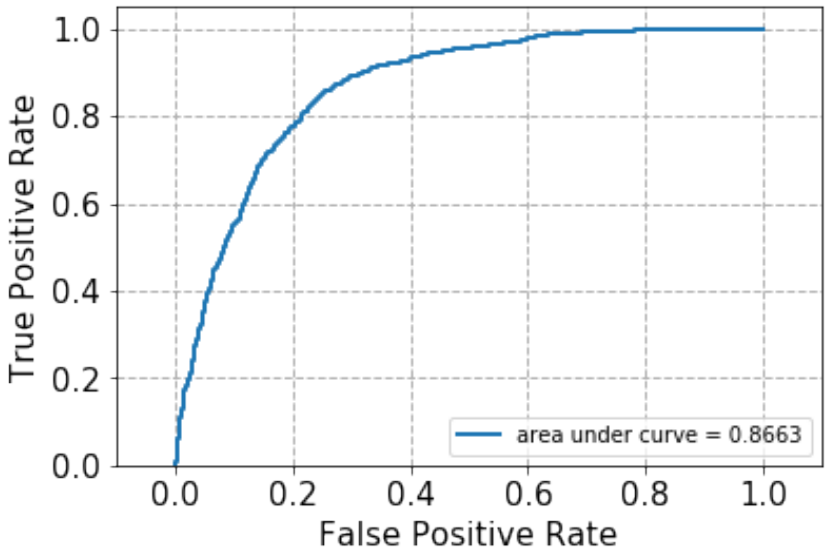


Figure 14: ROC curve of the Gaussian naïve Bayes classifier

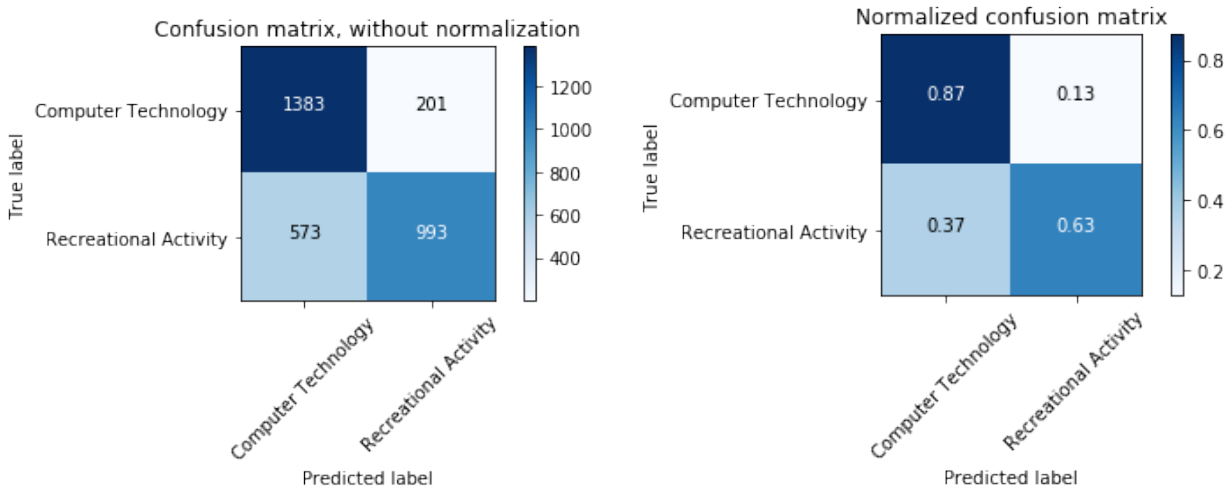
The accuracy, recall, precision and F-1 score are shown in table 7. The accuracy of 0.7543 indicates that the Gaussian naïve Bayes classifier can not predict the category of the testing sample very well. Moreover, a low recall of 0.6341 and a high precision of 0.8317 indicate that the Gaussian naïve Bayes classifier has bias to the negative class, which is the computer technology documents. In other words, it tends to give the prediction of computer technology documents. A F-1 score of 0.7196 shows that the Gaussian naïve Bayes classifier doesn't work very well, because the F-1 score is a overall measure that combines the precision and recall.

The confusion matrix and the normalized confusion matrix is presented in figure 15. It shows that the prediction accuracy for the computer technology documents is relatively high (0.87)

Table 7: Metrics of the Gaussian naïve Bayes classifier on test documents

accuracy	0.7543
recall	0.6341
precision	0.8317
F-1 score	0.7196

and is much lower for the recreational activity (0.63). This observation clearly shows that the naïve Bayes classifier has a bias to the computer technology class, which is consistent with the conclusion from table 7.



(a) (b)

Figure 15: Confusion matrix of Gaussian naïve Bayes classifier on test documents

Question 7

In this part, we follow the instruction in the project description to perform the grid search of hyper parameters. A pipeline is first constructed to combine the feature extraction, dimensionality reduction and classification. By changing the input parameters for the pipeline, different parameters in each of the 3 steps can be compared. The option of removing or keeping "headers" and "footers" is completed in the step of dataset loading. Then, the 5-fold cross-validation is

used to find the best combination of hyper parameters and options. The mean test score is used to in the comparison. To be specific, we focus on the averaged validation accuracy in each cross-validation process.

In total, 64 different options are compared. The best combination is listed in table 8. This model has exactly the same parameters as in Question 4, and it has a mean validation accuracy of 0.8783. The ROC curve, confusion matrix, and metrics of this model are already presented in figure 6, figure 7 and table 3.

Table 8: Best Combination of Parameters

Loading Data	keeping "headers" and "footers"
Feature Extraction	min_df = 3 and using lemmatization
Dimensionality Reduction	LSI using truncated SVD with 50 components
Classifier	linear SVM with $\gamma = 33.0$

To investigate the effect of hyper parameters, table 9 and table 10 present the parameters of the best 10 models and worst 10 models respectively. After observing these models, we find that using Gaussian naïve Bayes classifier leads to worst performance, which is consistent with the observation in Question 6. A possible explanation of the poor performance of naïve Bayes classifier is that the assumption of independence of features are not satisfied in the textual data. Also, the Gaussian distribution for the likelihood of each class may not be satisfied as well. On the other hand, using lemmatization and keeping the "headers" and "footers" in the documents always make the classification more accurate. The benefit of lemmatization is very obvious. By observing some textual data, we find that some important information such as "subject", "key words", "emails", "organizations" are given in the "headers" and "footers". Therefore, it is helpful to keep the "headers" and "footers" for better classification. In terms of other parameters, the performance of SVM, logistic regression with L_1 or L_2 regularization are very close. In fact, SVM with linear kernel are very similar with logistic regression. Using LSI or NMF for dimensionality reduction also leads to similar classification accuracy. Setting the min_df parameter to 3 or 5 has no obvious effect on the classification accuracy.

Table 9: Parameters of best 10 models

Rank	Loading Data	Feature Extraction	Dimensionality Reduction	Classifier	Mean Test Score
1	keep headers and footers	mindf=3 with lemma	SVM	LSI(k = 50)	0.8783
2	keep headers and footers	mindf=3 with lemma	Logistic Regression (L1)	NMF(k = 50)	0.8781
3	keep headers and footers	mindf=5 with lemma	SVM	LSI(k = 50)	0.8768
4	keep headers and footers	mindf=3 with lemma	Logistic Regression (L2)	LSI(k = 50)	0.8760
5	keep headers and footers	mindf=3 with lemma	Logistic Regression (L1)	LSI(k = 50)	0.8757
6	keep headers and footers	mindf=5 with lemma	Logistic Regression (L1)	NMF(k = 50)	0.8753
7	keep headers and footers	mindf=5 with lemma	SVM	NMF(k = 50)	0.8751
8	keep headers and footers	mindf=5 with lemma	Logistic Regression (L2)	LSI(k = 50)	0.8747
9	keep headers and footers	mindf=5 with lemma	Logistic Regression (L1)	LSI(k = 50)	0.8736
10	keep headers and footers	mindf=3 with lemma	SVM	NMF(k = 50)	0.8730

Table 10: Parameters of worst 10 models

Rank	Loading Data	Feature Extraction	Dimensionality Reduction	Classifier	Mean Test Score
55	keep headers and footers	mindf=5 no lemma	GaussianNB	NMF(k = 50)	0.8210
56	no headers and footers	mindf=5 no lemma	Logistic Regression (L2)	NMF(k = 50)	0.8191
57	keep headers and footers	mindf=3 no lemma	GaussianNB	LSI(k = 50)	0.7806
58	keep headers and footers	mindf=5 with lemma	GaussianNB	LSI(k = 50)	0.7604
59	keep headers and footers	mindf=5 no lemma	GaussianNB	LSI(k = 50)	0.7593
60	keep headers and footers	mindf=3 with lemma	GaussianNB	LSI(k = 50)	0.7576
61	no headers and footers	mindf=5 with lemma	GaussianNB	LSI(k = 50)	0.7052
62	no headers and footers	mindf=5 no lemma	GaussianNB	LSI(k = 50)	0.6887
63	no headers and footers	mindf=3 with lemma	GaussianNB	LSI(k = 50)	0.6868
64	no headers and footers	mindf=3 no lemma	GaussianNB	LSI(k = 50)	0.6712

4 Multiclass Classification

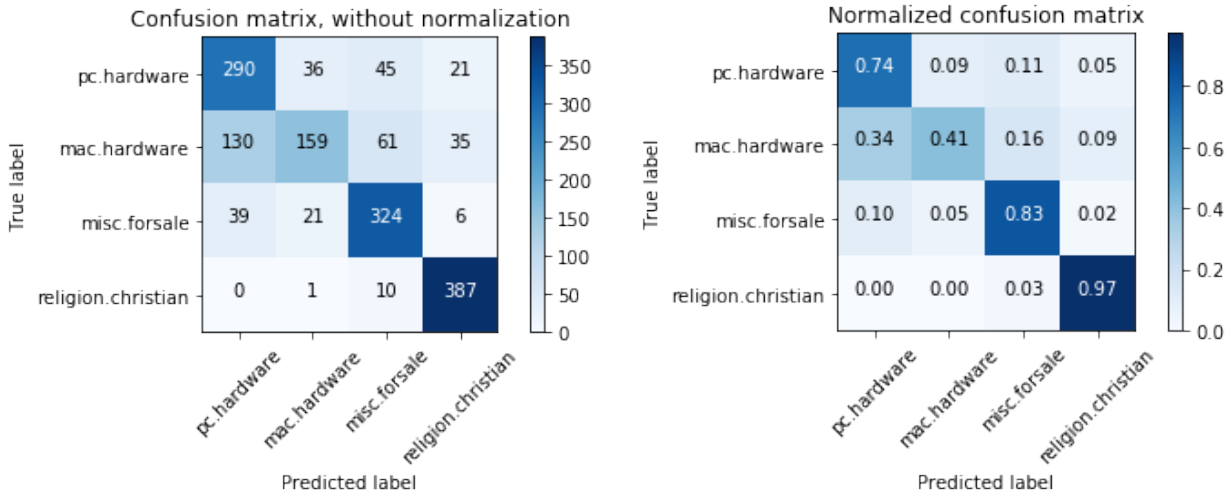
Question 8

In this part we use naïve Bayes, multiclass SVM with One vs One and One vs Rest for classifying multiple document classes. Confusion matrix of the above three cases are reported in figure 16, 17 and 18 respectively. Metrics results are shown in table 11. We can see that all metrics of naïve Bayes are not as good as SVM. This is possibly due to the independence assumption of naïve Bayes, since items could not be totally independent of each other in the same article with identical topic and it does not account for polysemy. Also the assumption of Gaussian distribution can be another source of inaccuracy. Again, with consistency to former parts, strong bias shows in the confusion matrix of naïve Bayes method. On the other hand, it is found that both metrics and confusion matrices performance by SVM(One vs One) and SVM(One

vs Rest) are pretty good, and they do not differ a lot from each other. This results possibly from the small number of classes that needs to be classified. For One vs One case, there are 6 comparison types in total, compared with the total number of 4 comparison types in the One vs Rest case. We propose that with the number of classes increasing, the difference between the two methods can be more obvious.

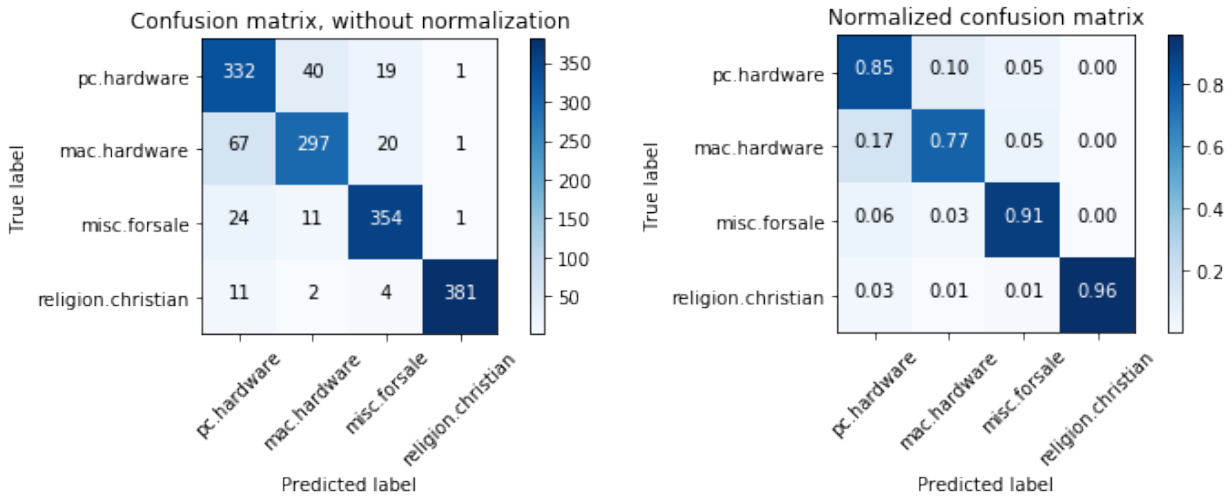
Table 11: Metrics of the naïve Bayes, multiclass SVM(OVO) and SVM(OVR)

	naïve Bayes	SVM(One versus One)	SVM(One versus Rest)
accuracy	0.7412	0.8716	0.8722
recall	0.7390	0.8708	0.8716
precision	0.7407	0.8744	0.8711
F-1 score	0.7261	0.8715	0.8713



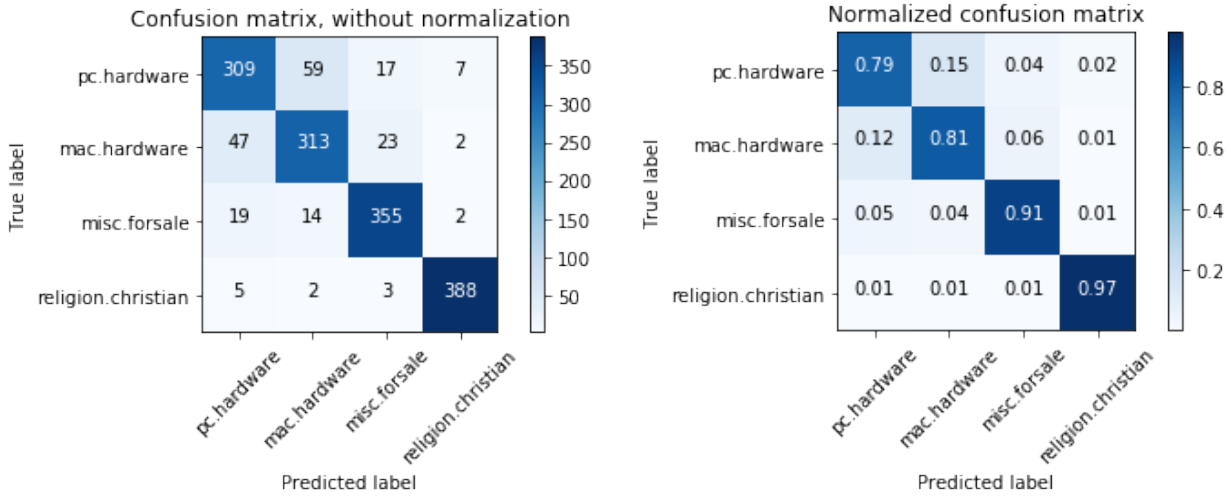
(a) (b)

Figure 16: Confusion matrix of naïve Bayes



(a) (b)

Figure 17: Confusion matrix of SVM One vs One



(a) (b)

Figure 18: Confusion matrix of SVM One vs Rest