Large-Scale Data Mining: Models and Algorithms **ECE 2019 Spring 2018**
Prof Vwani Roychowdhury TA: Tianyi Wang
UCLA, Department of ECE Behnam Shahbazi

Project 1: Classification Analysis on Textual Data

Due Friday Apr. 20, 2018 by 11:59 pm

Team Members

Brandon Hu, UID: 304743232
Jennifer MacDonald, UID: 604501712
Nguyen Nguyen, UID: 004870721
Sam Yang, UID: 604034791

Introduction

This project is designed to explore three different statistical classification algorithms including Linear Support Vector Machines (SVM), Logistics Regression, and Naïve Bayes. Statistical classification algorithms identify and categorize a predefined data set. This differs from clustering in which it only concerns about grouping of data points. The goal of this project is to learn and explore the construction of tf-idf textual data structure, familiarize with the listed algorithms above, evaluate and diagnose classification results, and learn PCA & NMF dimensionality reduction methods.

The advantages of Support Vector Machines are effective in high dimensional spaces, and cases where number of dimensions are greater than the number of samples. They are also memory efficient because they use a subset of training points in decision function which are called support vectors. They are versatile because they can use common kernel functions such as linear, polynomial, rbf, sigmoid, or a custom kernel. However, Support Vector Machines are not ideal when the number of features is much higher than the number of samples, thus requires extra care to avoid over-fitting in choosing the kernel function; regularization term is important. Another disadvantage is that SVMs do not provide probability estimates. They can only be obtained from expensive five-fold cross-validation.

Logistics Regression is a linear model for classification rather than regression. It is also widely known as logit regression. In this algorithm, the probabilities are the possible outcomes of a single trial using a logistic function. For the optimization problems, binary class L2 penalized tries to minimize the follow cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Likewise, L1 regularized calculates minimize the following optimization problem:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Naive Bayes is a supervised learning algorithm based on Bayes' theorem with the "naive" assumption that all features are independence between every pair. The Bayes' theorem states the following relationship below:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Maximum A Posteriori (MAP) estimation can be applied to estimate P(y) and P(x|y).

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Despite its relatively naive assumption, it has worked exceptionally well in many real-world scenarios such in document classification and spam filtering. They do require a small amount of training data to estimate the initial necessary parameters.

For this project, we used several methods as mentioned above to classify “20 Newsgroups” textual dataset. These include constructing tf-idf representations of the data, using some common classifiers found in the python machine-learning library scikit-learn, evaluating and diagnosing those classification methods, using the LSI and NMF dimensionality reduction methods, and using a complete pipeline of the textual data.

Getting familiar with the dataset

The “20 Newsgroups” dataset contains about 20,000 newsgroup posts split evenly among 20 topics, and individually is split into training and testing sets.

Question 1: To get started, plot a histogram of the number of training documents per category to check if they are evenly distributed.

Before we begin our classification, we need to set the training and testing datasets, and determine if the subclasses are evenly distributed. The purpose of the training dataset is to train the model since we are working with supervised training models; testing dataset is to determine how accurate and precise we are. We will discuss the accuracy, precision, recall, and F-1 score for each algorithm later in the report. These are the metrics that we can look at to determine how well each model performs. First, we split up the training and testing dataset into two main classes which are then used to train and test our classification models later. The two main classes are *Computer Technology* and *Recreational Activity*. Each main class contains 4 different individual subclasses as seen in Table 1 below.

Table 1: subclasses of categories

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

We shuffle the data, and set random_state parameter to 42 (a common seed number that is used in sklearn) in order to randomize the state of the machine. We also remove headers and footers since we find them to contain irrelevant and insignificant information. By plotting separate histogram as seen in Figure 1 and Figure 2 below, we can quickly see that the categories are evenly distributed for both training and testing dataset. However, if a dataset is not balanced, we would need to perform some kind of balancing to avoid overfitting. The training dataset has a total of 4,732 points, and testing dataset has a total of 3,150. In Figure 1, all the subclasses are within range between 578 and 600 documents (difference is only 22 documents). In Figure 2, all the subclasses are within 385 and 399 documents (difference is 14 documents). Thus, we can conclude that the datasets are evenly distributed and continue with our classification analysis.

Figure 1: Histogram of Topic-Documents for Train Dataset

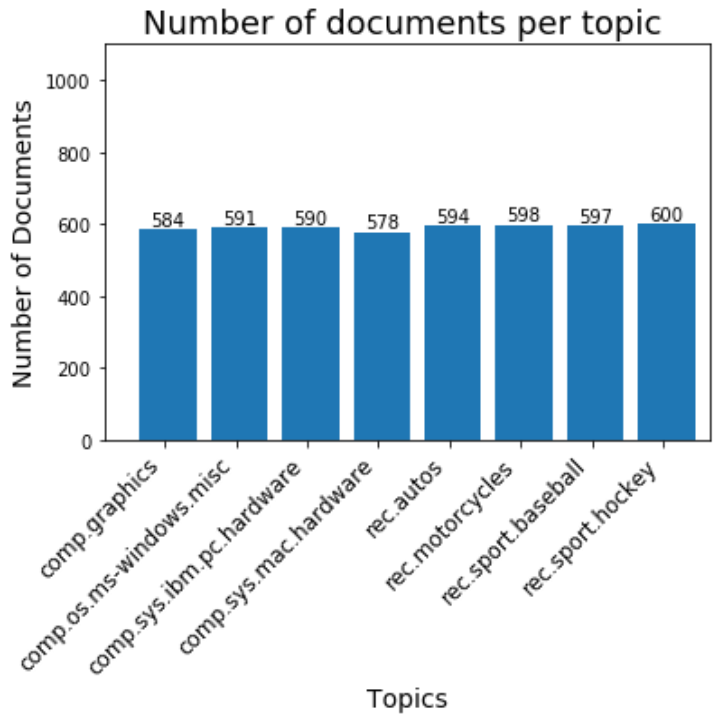
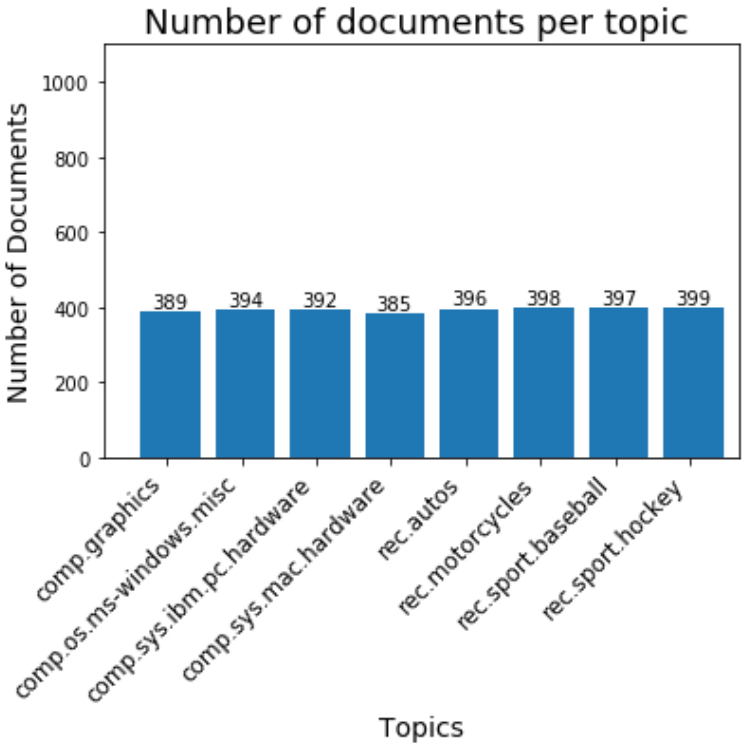


Figure 2: Histogram of Topic-Documents for Test Dataset



Binary Classification

Since we are only dealing with two classes, Computer Technology and Recreation Activity, we want turn our classes into binary classes. We define Computer Technology as 0, and Recreation Activity as 1. For any subclasses in Computer Technology, we set the target to 0; any subclasses in Recreation Activity, we set target to 1. For example,

Training Set
Original train_dataset.target: [6 7 4 2 1 3 0 7 5 3 0 5 5 5 3 1 3 0 0 2]
Binarized train_dataset.target:[1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0]

Test Set
Original test_dataset.target: [3 2 3 3 2 3 2 6 0 1 0 1 7 7 1 5 7 2 0 1]
Binarized test_dataset.target: [0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 1 0 0 0]

1 Feature Extraction

Feature extraction is a process to determine proper document representation. A good representation contains relevant information that is important for any classification models and to avoid any possible overfitting. One common representation that is used in this project is the "Bag of Words" representation, such that a document is represented as a matrix of term frequencies. For example,

Document-term Matrix

$$\begin{pmatrix} tf(d_1,t_1) & \cdots & tf(d_1,t_m) \\ tf(d_2,t_1) & \cdots & tf(d_2,t_m) \\ \vdots & \vdots & \vdots \\ tf(d_n,t_1) & \cdots & tf(d_n,t_m) \end{pmatrix}$$

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

- *Use the default 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.

Before we could vectorize our collection of documents, we had to lemmatize the words in both training and testing datasets. Lemmatization is a process to reduce any inflectional forms and related forms of a word to a common base word. For example,

am, are, is $\Rightarrow$ *be*

car, cars, car's, cars' $\Rightarrow$ *car*

The result of this mapping of text will be something like:

the boy's cars are different colors $\Rightarrow$ *the boy car be differ color*

We apply WordNetLemmatizer in nltk package (a popular natural language processing package in python) by tokenize words and apply pos_tag since this part of speech tag is required for lemmatizer process. We are looking only at words that are nouns, verbs, adjectives and adverbs. Additionally, we filtered any terms that are digits, but retained any terms that are alphanumeric.

After the lemmatizer process, we used CountVectorizer in sklearn to convert the collection of documents into a matrix form that we can easily used for classification later on. In our vectorizer method, we set the stop_words parameter to 'english'. Stop words are words that are commonly used in the english language and that do not add much value in our classification analysis. Some examples of the stop words include: a, an, and, are, is, in has, from, for, that, the, or, but, etc... Furthermore, we also set min_df parameter to 3. This is the cut-off parameter which is to disregard any terms that have frequencies less than 3. We also filtered We simply generate the TF-IDF matrix (Document-term Matrix) by using TfidfTransformer in sklearn for both training and testing datasets. We defined TF-IDF to be:

$$\text{tf-idf}(d,t) = \text{tf}(t,d) \times \text{idf}(t)$$

$$\text{idf}(t) = \log\left(\frac{n}{\text{df}(t)}\right) + 1$$

where 'tf' is the term frequency, and 'idf' is inverse document frequency.

Below are the shapes of the TF-IDF matrices:

X_train_tfidf: (4732, 14416)
X_test_tfidf: (3150, 14416)

There are more features (more than 3x) than samples available in these two matrices. If we apply a SVM classification model on these two datasets, then we will not be able to achieve high accuracy. This is a weakness of SVM as mentioned previously. Therefore, a dimensional reduction step must be applied which will be discussed below.

2 Dimensionality Reduction

Since the matrix dimensions of TF-IDF matrices above are very high, we need to reduce the number of features to avoid the "Curse of Dimensionality". High dimensions can definitely lead

to poor performance. Since our document-term TF-IDF matrix is sparse and low-rank, we can select a subset of the original features, which have more relevant information with respect to some performance measurements, and transform to a lower dimensional space. In this project, we explored and utilized two popular dimensionality reduction methods: Latent Semantic Indexing (LSI) and Non-negative Matrix Factorization (NMF).

LSI

We used TruncatedSVD in sklearn for LSI method which implements singular value decomposition (SVD) that computes only the *k* value (largest singular values) which is defined by the user. It is applied to our term-document matrices (returned by CountVectorizer and TfidfTransformer). This transformation converts our matrices to a "semantic" space of lower dimensions. LSI is known to remove effects of multiple meanings per word which can cause term-document matrices to be excessively sparse and have poor cosine similarity.

NMF

Likewise, Non-negative matrix factorization is an alternative method that assumes the data and components are non-negative, meaning that the data matrix do not contain negative values. It decomposes the samples by optimizing the distance between the samples and the matrix product. The most common distance is the squared Frobenius norm as seen below.

$$d_{\mathrm{Fro}}(X, Y) = \frac{1}{2}||X - Y||^2_{\mathrm{Fro}} = \frac{1}{2}\sum_{i,j}(X_{ij} - Y_{ij})^2$$

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 and compare with LSI: Which one is larger, the ∥X − WH∥ 2 F in NMF or the X − UkΣkVT k 2 F in LSI? Why is the case?*

First, we apply LSI and NWF to the training and testing datasets using TruncatedSVD and NMF respectively from sklearn, with parameters of k = 50, number of iterations = 10, and random_state = 42. Then, we transformed the data using fit_tranform().

The shape of the testing and training LSI and NMF matrices can be seen in Table 2 below.

Table 2: Shape of testing and training LSI and NMF Matrices

LSI train: (4732, 50)	LSI test: (3150, 50)
NMF train: (4732, 50)	NMF test: (3150, 50)

The shapes of the reduced training and testing datasets by using LSI and NMF are identical. This can mean that we did not have any negative values in our TF-IDF matrices.

Then we calculated the distance of Frobenius Norm for LSI and MWF which can be seen in Table 3 below.

Table 3: Frobenius Norm for LSI and NMF

Frobenius Norm for LSI:	63.8523
Frobenius Norm for NMF:	64.1921

From observation, LSI Frobenius distance is lower than NMF. LSI has an unique property such that SVD mathematically guarantees to have the lowest Frobenius distance error. NMF, on the other hand, is random and can reach a local minimum. However, it does not guarantee that it is the smallest distance, and therefore can converge early (https://arxiv.org/pdf/1407.7299.pdf). Overall, the two dimensional reduction methods are comparable and have similar performance.

3 Classification Algorithms

Now that we want have reduced our matrices to a lower dimension space using LSI and NMF, we're ready to begin the classification process using the dimension-reduced matrices to train different classifiers and then test those classifiers with the testing dataset.

Classification measures

To evaluate the quality of the classifier, we are going to be measuring the outcomes of the predictions on the testing data set in terms of precision, recall, F-score, and others.

- *Accuracy*, perhaps the most intuitive performance of measurement, is simply a ratio of correctly predicted observations.
- *Precision* looks at the ratio of correct positive observations.

  Precision = True Positives / (True Positives + False Positives)

- *Recall*, also known as sensitivity or true positive rate, is the ratio of correctly predicted positive events.

  Recall = True Positives / (True Positives + False Negatives)
- *F-Score* is the weighted average of Precision and Recall. This score takes both false positives and false negatives into account. Intuitively, it is not obvious as accuracy but it is more useful for uneven class distribution. It works best if false positives and false negatives have similar cost.

  F1-Score = 2*(Recall*Precision)/(Recall + Precision)
- *Confusion Matrix* is simply a matrix to display the numbers of correctly classified and misclassified.

SVM

Linear Support Vector Machine method is efficient at dealing with sparse and high dimensional (features) datasets, including textual data. It is usually good at producing general accuracy with low computational complexity. The algorithm learns the feature weights, *w,* and its intercept, *b,* from the training dataset. Once the weights are learned, the label of a data point is determined by $\bar{sign}(\mathbf{w}^\mathsf{T}\mathbf{x} + b)$ (a binary classification, either positive or negative in this case). The learning process of the parameter *w* and *b* is solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} \xi_i \\ s.t. \quad & y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \\ & \forall i \in \{1, \ldots, n\} \end{aligned}$$

In an optimization problem, minimizing the sum of the slack variables (a slack variable is a variable that is added to an inequality constraint to transform it into an equality) is to minimize the loss function on the training dataset. Likewise, minimizing the first term (regularization) is to maximize the margin between two classes. The tradeoff parameter γ controls the importance of two components. When γ is much greater than 1, also known as “Hard Margin SVM”, misclassification of an individual data point is heavily penalized. However, a “Soft Margin SVM” is when γ is much smaller than 1 but greater than 0 which is much more lenient to misclassification of a a few individual points as long the data points are well separated.

Question 4: Hard margin and soft margin linear SVMs:

- *Train two linear SVMs and compare:*
  - *Train one SVM with $\gamma = 1000$ (hard margin), another with $\gamma = 0.0001$ (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?*
- *Use cross-validation to choose γ:*
  Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10k \mid -3 \leq k \leq 3, k \in Z\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this best SVM.

We trained four different Linear SVMs; two is for using LSI with a hard margin and soft margin, and the other two are for NMF. We then calculated Accuracy, Precision, Recall, and F-1 Score values. We also plotted out a confusion matrix for each comparison.

Table 4: Overall Measurements for LSI and NMF using SVM

Method	gamma	Accuracy	Precision	Recall	F-Score
LSI	1000	0.9714	0.9681	0.9754	0.9718
LSI	0.0001	0.5047	0.5047	1.0	0.6708
NMF	1000	0.9663	0.9585	0.9754	0.9669
NMF	0.0001	0.5047	0.5047	1.0	0.6708
LSI	1000	0.9714	0.9681	0.9754	0.9718
NMF	1000	0.9663	0.9585	0.9754	0.9669

Table 5: Confusion Matrix of Hard and Soft Margin LSI

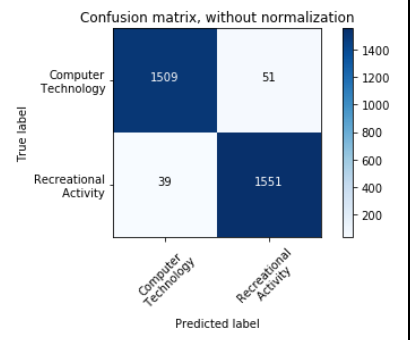
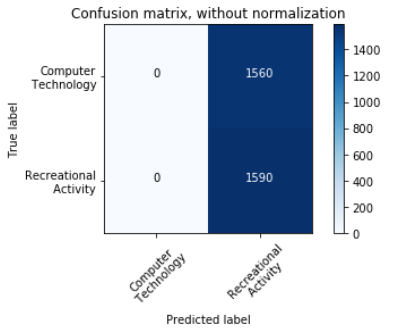
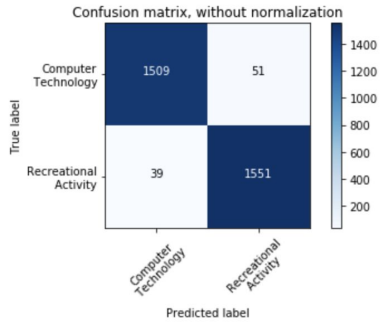
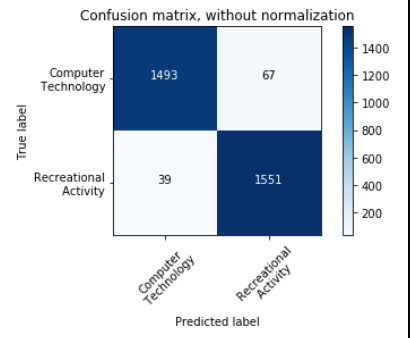
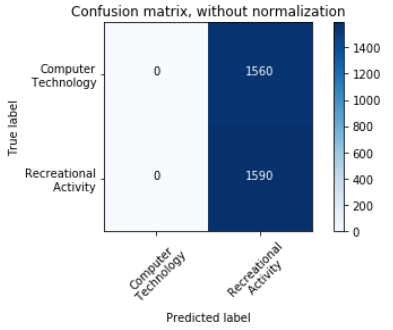
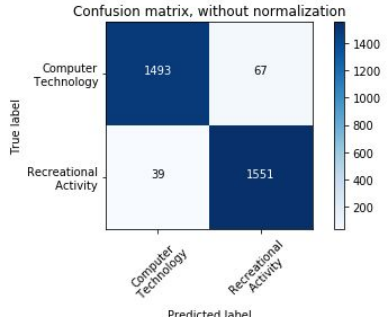
Hard Margin LSI	Soft Margin LSI	Optimized gamma LSI
		
Hard Margin NMF	**Soft Margin NMF**	**Optimized gamma NMF**
		

Table 4 above displays the Accuracy, Precision, Recall, and F-score for each comparison. It's obvious that using hard margin (γ = 1000) for both LSI and NMF has better accuracy and better precision so we would assume that is the best model. Not so fast! Intuitively, the gamma parameter defines how to penalize the error term. The behavior of SVM is very sensitive and greatly influence by the gamma parameter. Soft margin (γ = 0.0001) allows some examples to be "ignored" or placed in the wrong side of the margin. Hard margin puts a much stricter penalty on error term. That is why readers can see that accuracy for hard margin is much higher than accuracy for soft margin. Furthermore, the recall measurement for hard margin is 1, which means that all the terms are "correctly" classified. This could mean that the model might be overfitting using hard margin. Thus, if gamma is too large, then we need to make sure that the model is not overfitting. If gamma is too small, the model cannot capture the shape of the data, thus prone to underfitting.

Therefore, we performed a 5-fold validation method to validate and determine the best gamma. In our observation, we determined that the best gamma is 1000 for both LSI and 1000 for NMF, which is interesting. With a gamma at 1000, the measurements are exactly the same for both LSI and NMF. Table 4 provides all the performance measurements for hard and soft margin for both

LSI and NMF, as well as the best gamma parameter using 5-fold validation. Table 5 provides all the Confusion Matrices.

For our 5-fold cross validation, we found the best value of the parameter in the range of 10^k where k is between -3 and 3. We created a parameters object and set gamma to these values: parameters = {C:[10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]}. We then trained another SVC classifier and used GridSearchCV, which implements a fit and score method, and fit the training and binary targets to the classifier. To display the results, we used a DataFrame with the grid search cv classifier's cv_results_, displaying the parameter's gamma and corresponding mean test score for both LSI and NMF. The 5-fold validation tables can be found in Table 6 below.

Table 6: 5-fold Validation SVM

LSI

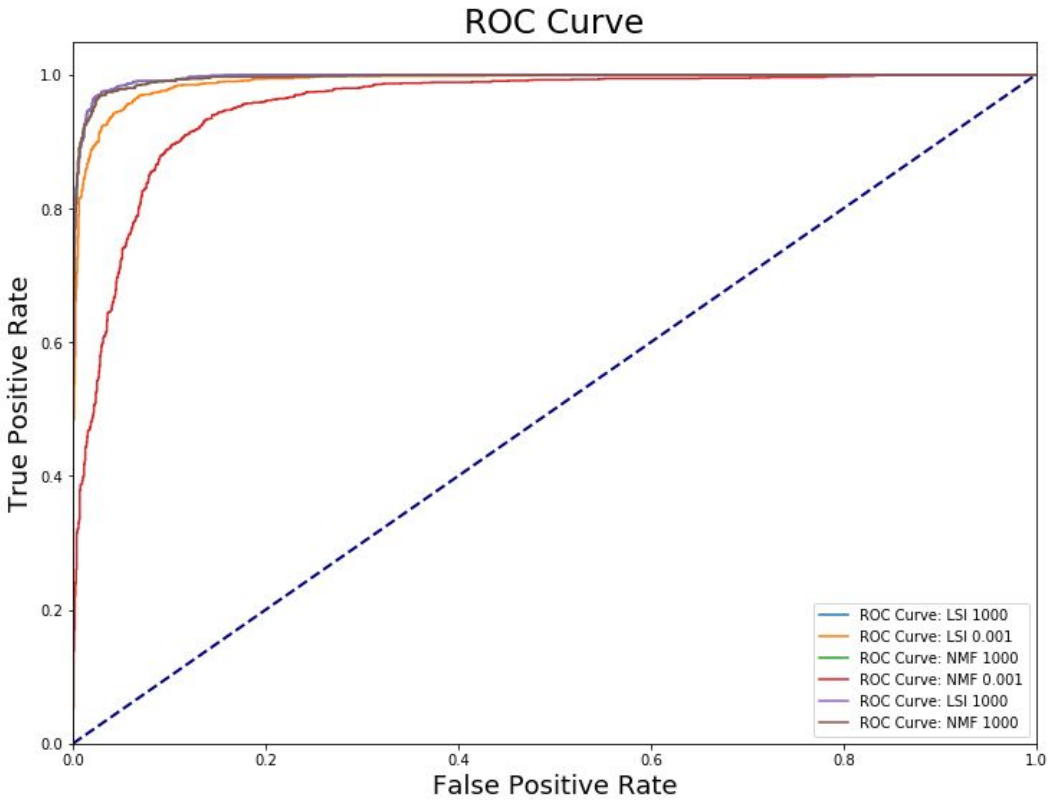
	param_C	mean_test_score
0	0.001	0.504861
1	0.01	0.504861
2	0.1	0.504861
3	1	0.930262
4	10	0.956889
5	100	0.964286
6	1000	0.969146

NMF

	param_C	mean_test_score
0	0.001	0.504861
1	0.01	0.504861
2	0.1	0.504861
3	1	0.504861
4	10	0.922232
5	100	0.944421
6	1000	0.952663

We also plotted an ROC curve with the combined curves of LSI with a hard margin, LSI with a soft margin, NMF with a hard margin, and NMF with a soft margin as seen in Figure 3 below.

Figure 3: ROC Curve for LSI and NMF with different gamma in SVM



Logistic Regression

Although logistic regression has a misleading name, it is a useful classifier for binary classification. It uses the input values to create a best-fit line equation as the representation. The goal for logistic regression is to maximize the likelihood of the training data while minimizing the regularization term. In logistic regression, the logistic function $(\sigma(\phi) = 1/(1+\exp(-\phi)))$ classifies a data point by calculating the probability of it belonging to a certain class. During the training process, *w* and *b* maximizes the likelihood of the training data. It can further include regularization term thus it can also minimize the regularization term. Also, adding regularization helps prevent overfitting.

Question 5: Logistic classifier:

- *Train a logistic classifier; 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, find the best regularization strength in the range $\{10k \mid -3 \leq k \leq 3, k \in Z\}$ 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 classifiers: w/o regularization, w/ L1 regularization and w/ L2 regularization, using test data. How does the regularization parameter affect the test error? How are the learnt coefficients affected? Why might one be interested in each type of regularization?*

First, we computed the logistic regression without regularization (leading to a very high c-value). We created a logistic regression classifier, setting c to 10**15 (basically setting a ridiculously high c-value to effectively have "no regularization"). We fit the classifier to the training data and use the predict function to predict the testing labels. We test our accuracy using the accuracy_score(), recall_score(), precision_score(), f1_score(), and the confusion_matrix() using the same parameters for LSI and NMF.

Table 7: 5-fold Validation for Best C Values of L1 and L2 Regularization for LSI and NMF

LSI with L1 Regularization

	param_C	mean_test_score
0	0.001	0.495139
1	0.01	0.919062
2	0.1	0.927303
3	1	0.964708
4	10	0.970626
5	100	0.969780
6	1000	0.969569

LSI with L2 Regularization

	param_C	mean_test_score
0	0.001	0.762468
1	0.01	0.926669
2	0.1	0.948859
3	1	0.957523
4	10	0.967033
5	100	0.971682
6	1000	0.969992

NMF with L1 Regularization

	param_C	mean_test_score
0	0.001	0.495139
1	0.01	0.495139
2	0.1	0.645604
3	1	0.950549
4	10	0.963652
5	100	0.961327
6	1000	0.961327

NMF with L2 Regularization

	param_C	mean_test_score
0	0.001	0.504861
1	0.01	0.560440
2	0.1	0.895604
3	1	0.927303
4	10	0.943998
5	100	0.954565
6	1000	0.961750

Next, we performed the same logistic regression with LSI using L1 regularization penalty. We set parameters= {'C':[10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]} and cv to 5. We also performed the same logistic regression with LSI using L2 regularization penalty. We set parameters= {'C':[10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3]} and cv to 5. Table 7 provides the table for 5-fold validation to determine best C values for L1 and L2 regularization for LSI and NMF.

Based on the cross-validation above, it seems that a C=10 leads to the highest validation mean accuracy with L1 regularization and C=100 leads to the highest validation mean accuracy with L2 regularization for LSI; C=10 with L1 regularization and C=1000 with L2 regularization for NMF.

We then calculate the performance measures and compare the logistic regression classifiers without regularization, with L1 regularization, and with L2 regularization.

Table 8: Overall Measurements for LSI and NMF using Logistics Regression

Method	C Value	Penalty	Accuracy	Precision	Recall	F-Score
LSI	10^15	None	0.9707	0.9681	0.9742	0.9711
NMF	10^15	None	0.9673	0.9598	0.9761	0.9678
LSI	10	L1	0.9692	0.9651	0.9742	0.9696
NMF	10	L1	0.9638	0.9544	0.9748	0.9645
LSI	100	L2	0.9698	0.9645	0.9761	0.9703
NMF	1000	L2	0.9622	0.9537	0.9723	0.9629

Table 9: Confusion Matrices of LSI and NMF with different Penalty

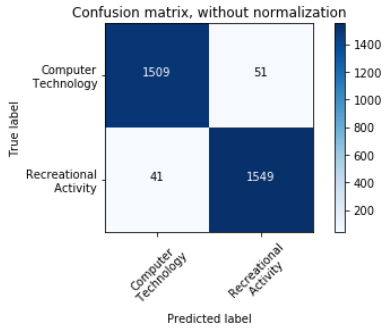
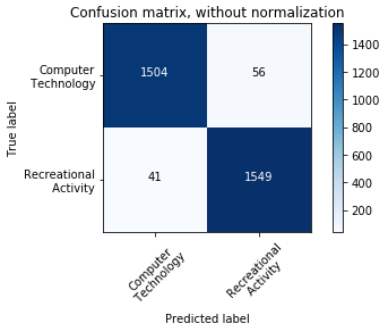
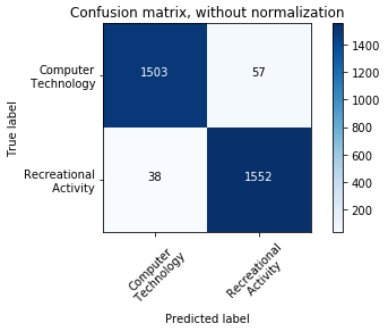
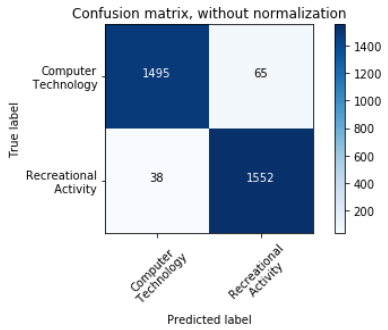
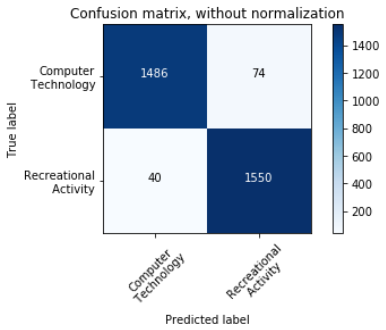
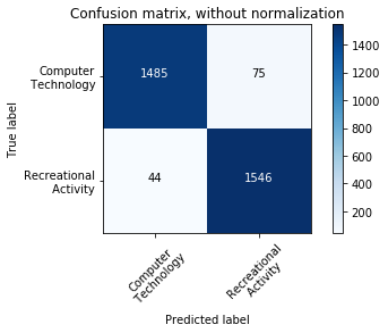
LSI (Default Penalty)	LSI using best L1	LSI using best L2
		
NMF (Default Penalty)	**NMF using best L1**	**NMF using best L2**
		

Table 8 and Table 9 provide a summary of measurements for LSI and NMF using Logistics Regression model. In this model, the success of the classifier increases proportionally with the C value (the inverse of the regularization strength). That is to say, less regularization is associated with a higher 5-fold validation accuracy. These results illustrate that when taking the sigmoid of the linear combination of the features, the weights should not be restricted in size in order to allow for the model to fit the target function accurately. The complexity of the target function matches the model-fitting capabilities of logistic regression without regularization. For this dataset, more regularization would lead to a model that underfits the data.

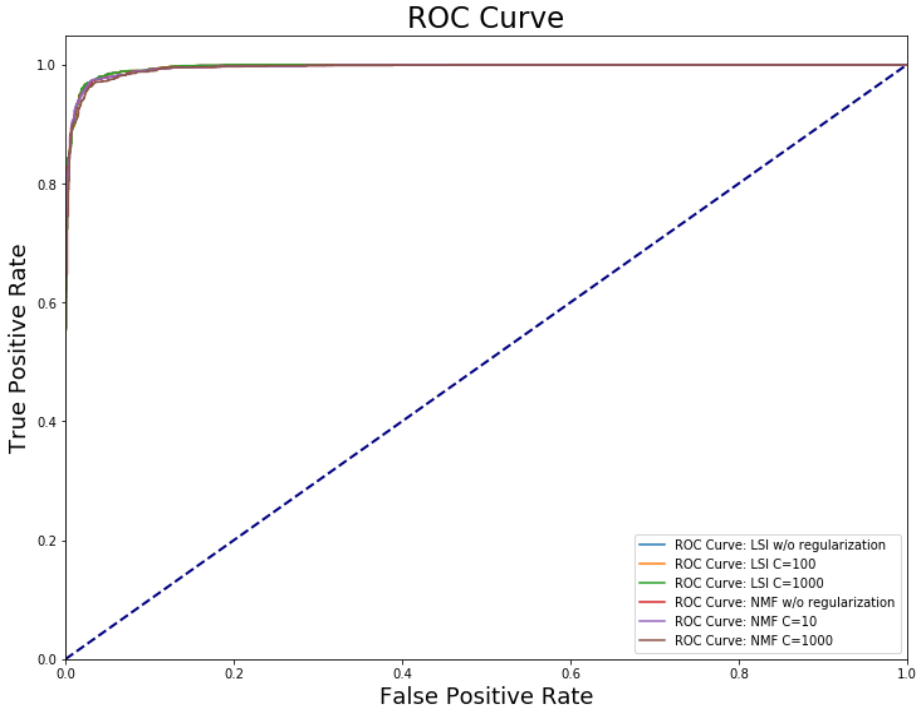
The learnt coefficients decrease in size with an increase in the regularization in power. In terms of the C-value, higher C means less regularization which means more freedom for the weights to increase in magnitude.

We see that L2 regularization leads to slightly smaller weights with a smaller standard deviation. This is because it punishes outlier weight values more heavily -- weights are then encouraged to remain small and close to each other. On the other hand, L1 regularization has a slightly higher mean weight value and a wider standard deviation compared to L2.

One may be interested in L2 regularization to keep the weights small and close together to prevent overfitting. L1 does a similar job, but is more robust to outliers. This often leads to the ability for weights to be sparse and able behave as feature selectors (with the zero-weights filtering out features and the high weight values indicating an important feature).

To compare, we plotted the ROC curves of the logistic regression classifiers LSI and NMF and without regularization which can be seen in Figure 4 below.

Figure 4: ROC Curve for LSI and NMF with different gamma in Logistics Regression



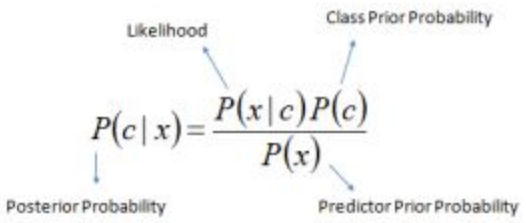
Naive Bayes

Naive Bayes assumes that features are statistically independent of each other. This simplifies the Maximum A Posteriori (MAP) estimation of the labels that is,

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m) = P(x_i \mid y) \qquad i \in \{1, \ldots, m\}$$

There are multiple types of Naive Bayes classifiers include MultinomialNB, BernoulliNB, and Gaussian NB. Each classifier uses different algorithm to classify. In this project, we used GaussianNB classifier.

The Naive Bayes classifier is based on Bayes Theorem, where the probability of an outcome is based on prior knowledge of conditions that are related to the event or outcome being measured (seen below).



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

The outcome is the measurement is the posterior probability, which is itself the prior probability of the class multiplied by likelihood over the probability of a predictor given the class.

The Naive Bayes modifies this theorem by assuming each feature in a class is unrelated to the other features and only calculates the posterior probability for each class. The class with the highest posterior probability would be the one with the highest likeliness. For example (https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/):

For example: Players will play if weather is sunny. Is this statement is correct?

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)
Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64
Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Question 6: Naïve 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.

We initialized arrays for false positive rates and true positive rates, and then used the data to train the GaussianNB classifier and predict the labels to calculate the accuracy, recall, precision, F-1 score, and confusion matrix for LSI and NMF matrices.

Table 10: Overall Measurements for LSI and NMF using Naive Bayes

Method	Accuracy	Precision	Recall	F-Score
LSI	0.8000	0.7254	0.9717	0.8306
NMF	0.9416	0.9140	0.9761	0.9440

Table 11: Confusion Matrices of LSI and NMF using Naive Bayes

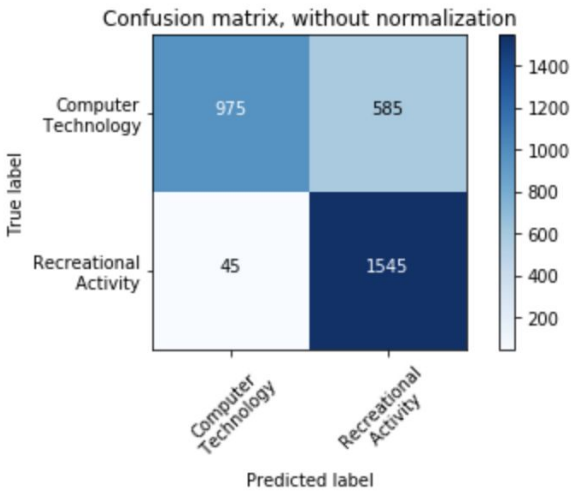
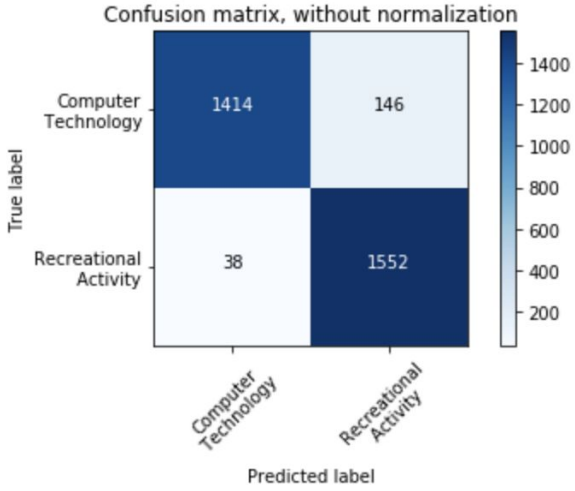
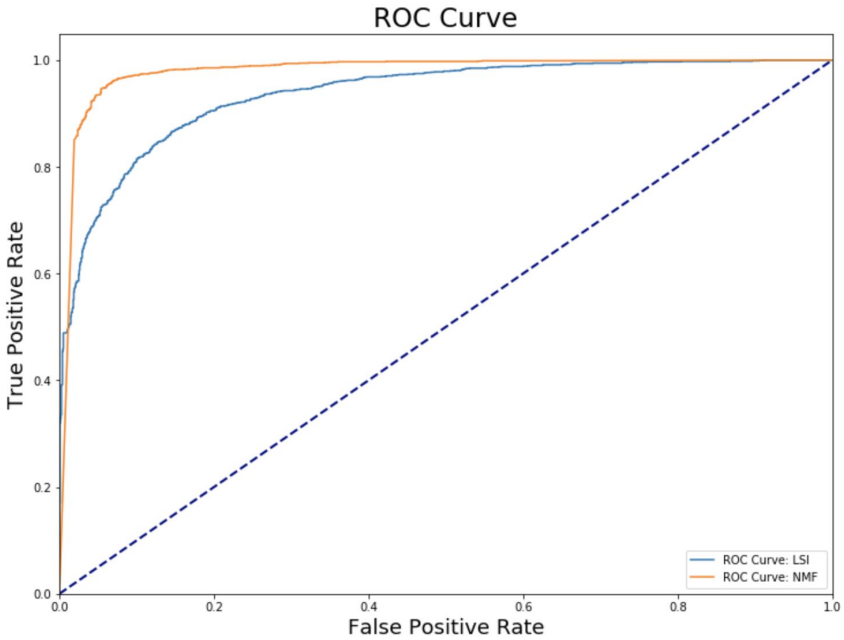
LSI	NMF
	

Table 10 provides a summary of measurements for LSI and NMF using Naive Bayes. As we can see, NMF is better at using Gaussian Distribution Naive Bayes model with an accuracy at 94.16%, while LSI is falling behind at 80%. Table 11 provides a visual of confusion matrices. The prediction statistics between both methods was greater using NMF, and can be visually seen in the ROC curve graph in Figure 5 below.

Figure 5: ROC Curve for LSI and NMF with different gamma in Naive Bayes



Grid Search of Parameters

In our analysis above, we have looked at different models and found ways to train and predict the data. However, we can further fine tune the parameters using Grid Search.

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):*

Table 2: Options 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?*

To accomplish the objective of question 7, our group constructed a pipeline that performs feature extraction, dimensionality reduction, and classification. Using the pipeline, 5-fold cross-validation was performed with the following parameters shown in the table above. The best parameters were chosen for the best test accuracy score. The best parameters that were found are:

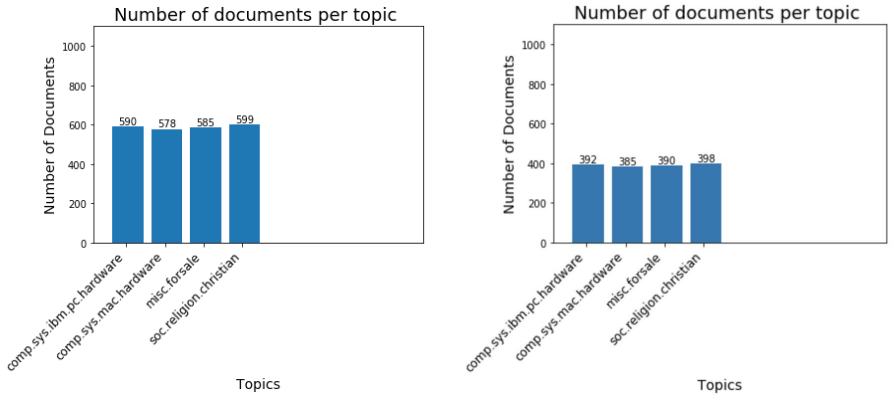
Table 12: Mean Test Score optimized parameters for Grid Search

mean_test_score	classifier	penalty	dim reduction	min df	lemm	rmv_headfoot
0.976754	Logistic	l1	SVD	3	TRUE	FALSE
0.976543	SVM	NaN	SVD	3	TRUE	FALSE
0.975909	Logistic	l2	SVD	3	TRUE	FALSE
0.975063	Logistic	l2	SVD	3	FALSE	TRUE
0.975063	Logistic	l1	SVD	5	TRUE	FALSE

Multiclass Classification

Question 8: In this part, we aim to learn classifiers on the documents belonging to the classes: comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, misc.forsale, soc.religion.christian Perform Naïve 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.

Initial inspection of the data indicates that the classes in the dataset are unbalanced as shown below and thus the class imbalance will need to be addressed. To address the class imbalance the majority classes would need to be down sampled to have the same number of instances as the minority class. However, since the dataset distribution is even downsampling is not needed. This was also confirmed with the TA in Office Hours on 4/17/2018.



Dataset distribution for Question 8. Training Set [Left] and Testing Set [Right]

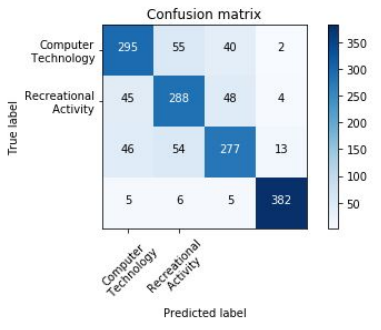
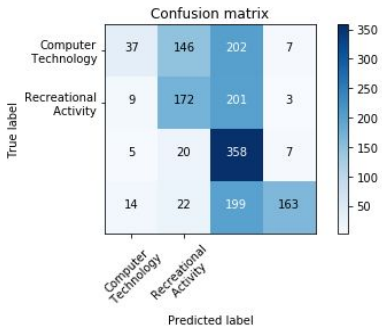
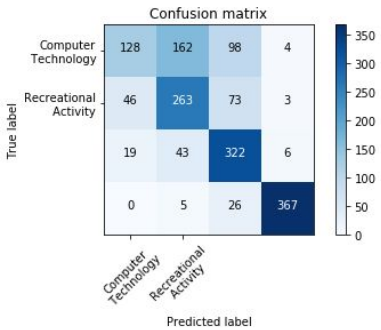
Our investigation of multiclass classification on the data set was performed in three main steps: Initializing data, fitting data + testing models, quantifying classifier statistics.
The three different models used in this multiclass classification activity was Naive Bayes, multiclass SVM One vs One (OvO), and multiclass SVM One vs Rest (OvR).
To prepare the dataset for classification we utilized the same approach as with the Computer Technology and Recreational Activity dataset by compressing the count vector into TFIDF, LSI, and NMF formats.

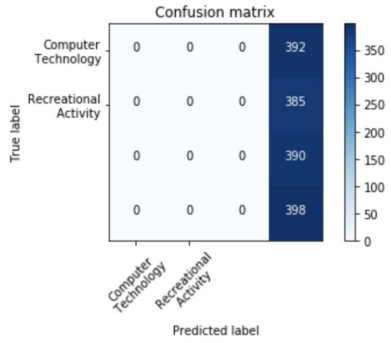
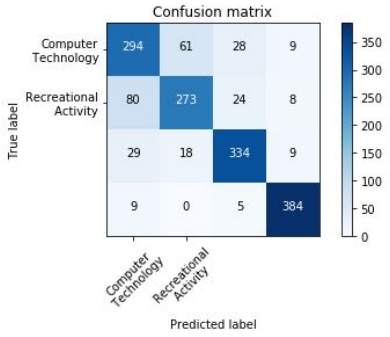
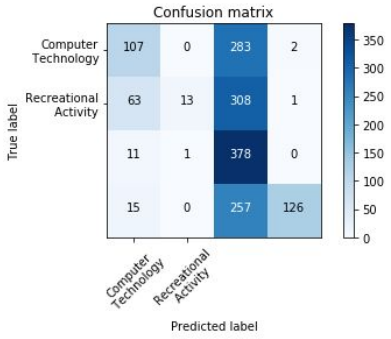
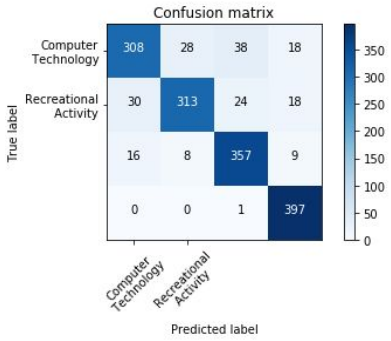
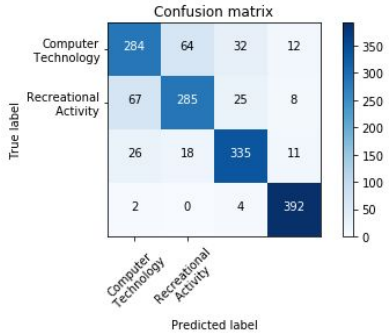
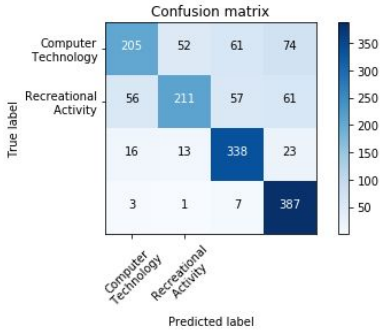
We evaluate the models using the following statistics: the confusion matrix, accuracy, recall, precision, and F-1 score. These statistics were calculated and are shown on the next few pages:

Table 12: Confusion Matrices of LSI and NMF using Naive Bayes

Method	Accuracy	Precision	Recall	F-Score
Naive Bayes Multiclass TFIDF	0.7936	0.7936	0.7936	0.7936
Naive Bayes Multiclass LSI	0.4664	0.4664	0.4664	0.4664
Naive Bayes Multiclass NMF	0.6900	0.6900	0.6900	0.6900
SVM OvO Multiclass TFIDF	0.2543	0.2543	0.2543	0.2543
SVM OvO Multiclass LSI	0.8210	0.8210	0.8210	0.8210
SVM OvO Multiclass NMF	0.3987	0.3987	0.3987	0.3987
SVM OvR Multiclass TFIDF	0.8785	0.8785	0.8785	0.8785
SVM OvR Multiclass LSI	0.8281	0.8281	0.8281	0.8281
SVM OvR Multiclass NMF	0.7290	0.7290	0.7290	0.7290

Table 13: Confusion Matrices of LSI and NMF using Naive Bayes

Naive Bayes		
Multiclass TFIDF	Multiclass LSI	Multiclass NMF
		

SVM OvO		
Multiclass TFIDF	Multiclass LSI	Multiclass NMF
		
SVM OvR		
Multiclass TFIDF	Multiclass LSI	Multiclass NMF
		

With the Naive Bayes model, the performance measurements (Accuracy, Recall, Precision, F1) were middle of the road for the TFIDF and NMF formats while it was the lowest amongst the three models for the LSI dataset format. The contributing factors to this performance can be attributed to the independence assumption of naive bayes as well as the guesswork included in a probabilistic model.

For the SVM OvO model, the performance measures on the TFIDF dataset was very poor. This is due to the sparse nature of the TFIDF format and when the SVM uses one-vs-one comparison then inaccuracies will occur. SVM OvO performs decently well on the LSI dataset, this is intuitive due to the LSI algorithm essentially creating automatic document categorizations which would aid in the correct classification.

Among the three different dataset formats (TFIDF, LSI, NMF), the SVM OvR model displayed highest measurement scores for all three formats. The SVM OvR approach performed similarly

to the SVM OvO approach to the LSI dataset. The one-vs-rest implementation of the SVM allows the SVM to compare the test data to all other labels in the dataset. This provides a bigger picture for the SVM which leads to lower error rates.

We can see that in an overall comparison between Naive Bayes, SVM OvO, and SVM OvR, Naive Bayes is not particularly the best in any of the dataset formats. SVM OvR performs best on the TFIDF and NMF formatted dataset. SVM OvO and SVM OvR perform similarly for the LSI format. The measurements scores for the models are the same due to the use of the 'micro' averaging option of the measurement scores. The 'micro' option calculates metrics globally by counting the total true positives, false negatives and false positives and thus the reason the measurement scores are the same. The other two options, 'macro' and 'weighted' could have also been used in accordance with the TA post on piazza. The 'macro' option computes the measurement scores for each class label without taking into account any class imbalances. The 'weighted' option is essentially the 'macro' option but accommodates for any imbalances of the class labels in the dataset. Either of these two options, 'macro' or 'weighted' would have produced different scores. However, each option is valid and dependent on the purpose of the measurements.

Overall, to fully utilize the classification effectiveness for the TFIDF, LSI, NMF data configurations, and ensemble method should be used to weight the models to their best classified data. For example, SVM OvR would be used to classify data in the TFIDF and NMF format, and either SVM OvR or SVM OvO would be used to classify data in the LSI format. For a specific dataset, if the SVM OvR and SVM OvO desired performance measures are the same, then the SVM OvR model is chosen due to its faster computation time.

Conclusion

This project provided an enlightening introduction to the possibilities of classification of textual data. Using the newsgroup dataset we were able to explore the nuance of data inspection, feature extraction, dimensionality reduction, use multiple classification algorithms though a data pipeline, optimizing our hyperparameters though gridsearch, and evaluate our models on their performance with common measurements used to quantify the efficacy of models.