Classification Analysis on Textual Data

University of California, Los Angeles
ECE 219 Project 1

Joshua Hannan (UID: 805221851)
James (Zach) Harris (UID: 105221897)
Dennis Wang (UID: 105229662)
Akash Deep Singh (UID: 405228294)

jhannan@g.ucla.edu
jzharris@g.ucla.edu
dmwang626@g.ucla.edu
akashdeepsingh@g.ucla.edu

1 Introduction

Classification of large data sets has become an increasingly important tool for data analysis, as it allows for models to make predictions in real-world applications. In this project, we classify textual data through a series of commonly-used techniques. First, we lemmatize the data and analyze it with a metric known as TF-IDF to process the words into words that have more significance to us. We further process the data by performing dimensionality reduction to circumvent the Curse of Dimensionality. With this dimension-reduced data, we train several classifiers and analyze how well they perform with various metrics. We then use 5-fold cross validation and a grid search of parameters to find the best-performing classifier.

1.1 Getting familiar with the dataset

The data set we will be working with in this project is the "20 Newsgroups" data set, which is a collection of approximately 20,000 documents spread across 20 different newsgroups. The 8 newsgroups we are interested in are:

- comp.graphics
- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- comp.sys.mac.hardware
- rec.autos
- rec.motorcycles
- rec.sport.baseball
- rec.sport.hockey

These are further combined into two superclasses, **Computer Technology** and **Recreational Activity**, which we will be using for binary classification.

Question 1 We see that the 20 newsgroups have a somewhat even spread, except for the first and last few categories. However, if we look at the categories we will be using in this project, as in Fig. 2, we see that these are much more balanced. In general, we need to consider any sort of imbalance in our data set, such as incurring a stronger penalty to classes with fewer data points or randomly removing points from larger classes to artificially balance the data. However, this is not necessary for this project, as the data set is evenly spread across the 8 categories.

2 Binary Classification

2.1 Feature Extraction

Classification of documents cannot be easily performed without first preprocessing the data. There are many reasons why this is the case, however the main

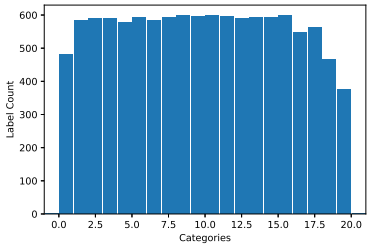

Fig. 1. Spread for all 20 newsgroups

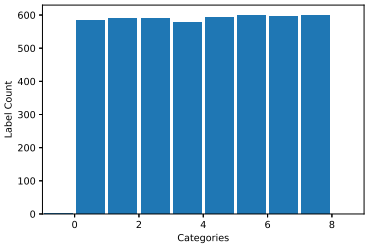

Fig. 2. Spread for the 8 newsgroups

reason is that the English language has many ways of conveying the same information. For example, the sentences "the boy ran across the lawn" and "the boy is running over the grass" have many similarities. Both cases are an example of physical activity, however a classifier would have trouble discerning this by using the raw document. Instead, it is easier for features to be extracted from the documents and used as markers defining what class they belong to. For example, a string of documents talking about physical activities may have a different feature "fingerprint" when compared to a document talking about historical literature.

Features are extracted through multiple processes. The document is first passed through a lemmatizer. The process of lemmatization uses vocabulary and morphological analysis to deconstruct each word into their base constituent, called a lemma. For example, when the sentence "A boy is running over the green and yellow hills" is passed through the lemmatizer, the output becomes "A boy be run over the green and yellow hill". Notice the lemmatizer replaces all past and future tense verbs with their present tense, and all plural nouns with their singular counterparts.

After each document is lemmatized, frequently occuring words are removed by a dictionary of words called "stopwords". The english stopwords collection consists of 318 words that are too common in the english language to be of any use for classification. This collection includes the words "no, to, or, not, very, rather" and many others. All 318 words are removed from the documents. The complete list of stopwords is given in the Appendix.

After each lemmatized document is filtered by stopwords, features can be extracted. The frequency of each word used in the document is analyzed using the "Term Frequency-Inverse Document Frequency (TF-IDF)" metric. This metric measures the number of times a word is used in a single document, normalized over a function of how many times the word appears in the entire corpus. Therefore the TF-IDF will attempt to quantify which words are unique for each document. The most unique words can be used to identify that document. If enough documents have the same set of words, it is straightforward to conclude

that they are of the same class of document. Classification of documents can be performed in this way. The specific equations used in this project are

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

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

Where $\text{tf}(t,d)$ is the term frequency of term t in document d and $\text{idf}(t)$ is the inverse document frequency of term t across the corpus. The inverse document frequency metric is calculated by taking the logarithm of the total number of documents, n, divided by the number of documents that contain term t, $\text{df}(t)$. Intuitively, $\text{idf}(t)$ gives larger weight to unique terms and smaller weight to terms that are frequent among many documents. A trailing 1 is added to the terms in the quotient of $\text{idf}(t)$ to prevent zero division from occurring.

Question 2 The following specs were used during lemmatization:

- Use the "english" stopwords of the CountVectorizer
- Use min_df = 3 for the CountVectorizer
- Exclude terms that are numbers (e.g. "123", "-45", "6.7", "5e07" etc.)
- Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos_tag

The resulting dimensions of the TF-IDF matrices are shown in Table 1.

Subset	**TF-IDF Matrix shape**
train	(4732, 16595)
test	(3150, 16595)

Table 1. Dimensions of TF-IDF matrices

2.2 Dimensionality Reduction

With large data sets, the dimensionality of the representation vectors will be extremely high, but our learning algorithms often are not efficient and do not perform well in high dimensions. This is referred to as the curse of dimensionality. To mitigate this, we can take advantage of the fact that our TF-IDF matrix is relatively sparse, so we can take a significantly smaller subset of data points that give us the most "information" about the data set as a whole. Two techniques that are often used are Latent Semantic Indexing (LSI) and Non-negative Matrix Factorization (NMF).

LSI Latent Semantic Indexing, or Latent Semantic Analysis (LSA), is done by using the Singular Value Decomposition to find a low-rank matrix that best approximates our data matrix. When we compute the SVD on our matrix X with d documents and t terms, we get

$$X = U\Sigma V^T = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix} \begin{bmatrix} - & v_1^T & - \\ & \vdots & \\ - & v_r^T & - \end{bmatrix}$$

where $\Sigma \in \mathbb{R}^{r \times r}$ contains the nonzero singular values, $U \in \mathbb{R}^{d \times r}$ contains the left singular vectors, and $V^T \in \mathbb{R}^{r \times t}$ the right singular vectors. We can interpret the singular values as how important each corresponding left and right singular vector are. Thus if we wish to find a rank k matrix that approximates X, we truncate U, Σ, and V^T to only contain the first k singular values and singular vectors. Then our approximation to the TF-IDF matrix will be

$$X_{reduced} = U_k\Sigma_k = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{bmatrix}$$

Note that $U_k\Sigma_k$ has the same number of rows as X; this is important because our data is stored in the rows of X. Sci-kit learn has a class TruncatedSVD that can perform this approximation for a specific rank k.

NMF Non-negative Matrix Factorization is another low-rank approximation method that attempts to find non-negative matrices W and H that minimize

$$\|X - WH\|_F^2$$

where $\|\cdot\|_F$ is the Frobenius norm, defined to be

$$\|X\|_F = \sqrt{\sum_{i,j} X_{ij}^2}$$

In other words, we are solving:

$$\min_{W,H} \|X - WH\|_F^2 \quad \text{s.t.} \begin{cases} W \geq 0 \\ H \geq 0 \end{cases}$$

Intuitively, this optimization problem is trying to best describe our data (the rows of X) as non-negative linear combinations of k "topics", or terms. We impose the non-negativity constraints because we want there to be physical meaning to the optimal matrices W, H that are found; having negative elements does not make any sense for text documents. Then we use

$$X_{reduced} = W^*$$

as the dimension-reduced data matrix. Again, notice that $X_{reduced}$ has the same number of rows as X, so we have not fundamentally lost any data; we are simply trying to describe it with fewer terms. Sci-kit learn also has a class NMF than can compute the NMF of a matrix for a specific rank k.

Question 3 To compare how each of these dimensionality-reducing algorithms performed, we compare their objective values: $\|X - U_k \Sigma_k V_k^T\|_F^2$ for LSI and $\|X - WH\|_F^2$ for NMF. For the training data, we get:

$$\begin{aligned} \text{LSI}_{\text{train}} &= 3895.3935173929685 \\ \text{NMF}_{\text{train}} &= 3939.889752677838 \end{aligned}$$

and for the testing data we get

$$\begin{aligned} \text{LSI}_{\text{test}} &= 2676.4613618492067 \\ \text{NMF}_{\text{test}} &= 2689.3020883815734 \end{aligned}$$

In both cases, we see that LSI performed better, as it is a closer approximation to X than NMF is. This may be the case because LSI does not have any non-negativity constraints, making it more flexible in the optimal $X_{reduced}$ found. In optimization terms, the LSI problem likely has a feasible set that contains a solution that is closer to the true X than the NMF problem, resulting in LSI being a better approximation. Furthermore, if there are a few ($\leq k$) terms that are significantly more important (i.e. have much higher singular values) than the others, then the SVD will give us a very accurate approximation to X because the information in X will be predominantly in the first k singular vectors and singular values. NMF will likely not be able to attain this level of accuracy because it is not directly computed using the SVD.

3 Classification Algorithms

In the following sections, we trained and tested different classifiers, namely soft and hard margin support vector machines, unregularized and regularized logistic classifiers, and Naïve Bayes classifiers, on our dimension-reduced data set. Instead of classifying on all 8 categories separately, we reduced it to binary classification by combining them into two super-classes: "Computer Technology" and "Recreational Activity." We then use various classification measures to determine how well each classifier performed.

3.1 Classification measures

To measure the performance of our classifiers, we look at the **confusion matrix**, **ROC curve**, **accuracy score**, **precision and recall scores**, and **F-1 score**.

In binary classification, the confusion matrix is a 2x2 matrix whose elements are found by counting how many times our classifier correctly classified the test data points.

$$\left[\begin{array}{c|c} \text{True Positive} & \text{False Positive} \\ \hline \text{False Negative} & \text{True Negative} \end{array}\right]$$

From this matrix, we can compute various other classification measures. The other measures we used were:

$$\begin{aligned} \text{Accuracy} &= \frac{\text{True Positive} + \text{True Negative}}{\text{Total Population}} \\ \text{Precision} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ \text{Recall} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\ \text{F-1} &= \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \end{aligned}$$

The ROC curve is found by plotting the true positive rate (TPR) against the false positive rate (FPR). The TPR and FPR values are found by varying the discrimination threshold value t from 0 to 1. Essentially, this is finding how well we classify for different thresholds. The greater the area under the ROC curve, the better our classifier performed because this tells us that the classifier was able to have a high true positive rate at low false positive rate.

4 SVM

A **Linear Support Vector Machine (SVM)** is a classifier that uses an optimal hyperplane to categorize data into separate classes. To determine the separating hyperplane, SVM attempts to learn a vector of weights $\boldsymbol{w}$, and an intercept b, based on a training data set. SVM calculates $\boldsymbol{w}$ and b by solving the optimization problem:

$$\begin{aligned} \underset{\boldsymbol{w},b}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \gamma \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1 - \xi_1, \\ & \xi_i \geq 0, \qquad\qquad \forall i \in 1, \ldots, n, \end{aligned}$$

where $\boldsymbol{x_i}$ is the ith data point and $y_i \in \{0, 1\}$ is the class label. Once all $\boldsymbol{x_i}$ have been determined, the SVM "thresholds" $\boldsymbol{w}^T\boldsymbol{x} + b$ with 0, such that the sign of the function determines to which class each point belongs.

By minimizing the first term, $\frac{1}{2}\|\boldsymbol{w}\|_2^2$, the **SVM seeks to maximize the margin between the two classes of data.** By minimizing the second term, $\gamma \sum_{i=1}^{n} \xi_i$, however, the SVM tries to minimize the loss function on the training

data. The parameter γ controls the SVM's tolerance for misclassification. **In this report, we analyze the effects of a hard margin (when $\gamma \gg 1$) and a soft margin (when $\gamma \ll 1$)**. Instituting a hard margin highly penalizes misclassifcation of individual points, making the SVM more susceptible to outliers. Conversely, using a soft margin allows the SVM to misclassify several points is the data points are generally well separated.

4.1 Hard Margin vs Soft Margin Linear SVM

To train two SVMS with a hard and soft margin, we used the **LinearSVC** classifier from the sci-kit learn machine learning library. The parameter C in LinearSVC is the error term or penalty parameter and corresponds to γ as stated in the problem statement.
For the hard margin linear SVM, the value of $\gamma = 1000$ and hence in LinearSVC, we specify the value of C as 1000. For the soft margin classifier, the value of $\gamma = C = 0.0001$.

The *max_iter* parameter is set to 100000 because for any lower number of iterations, the hard margin classifier does not converge.

The performance scores of both the classifiers are plotted and compared in Tables 2 and 3. Fig. 3 and Fig. 4 show the Confusion matrices for hard margin linear SVM and soft margin linear SVM respectively. Fig. 5 and Fig. 6 depict the ROC curves for hard margin linear SVM and soft margin linear SVM respectively. **From tables 2 and 2, we can infer that the hard margin SVM classifier performs better than the soft margin SVM because it has better accuracy, , better precision and better F1 score**. This can be largely attributed to the fact that since the penalization or error factor is very small in case of soft margin SVM classifier, it ends up with a hyperplane that wrongly classifies a large number of points and hence ends up with a low accuracy.

Accuracy score:	0.9721
Precision score:	0.9647
Recall score:	0.9805
F-1 score:	0.9726

Table 2. Hard Margin Linear SVM

Accuracy score:	0.6130
Precision score:	0.5660
Recall score:	1.0000
F-1 score:	0.7229

Table 3. Soft Margin Linear SVM

Comparing the two ROC curves, we can notice that the area under curve for the hard margin SVM is 0.9958 **whereas it is** 0.9685 **for the soft margin SVM classifier.**

The soft margin SVM has a low accuracy because it does not penalize wrong classification heavily. It's ROC curve has less area under it when compared to

that of the hard margin SVM. **This conflicts with the fact that soft margin SVM has a recall score 1**. Since, **F1 score takes into account both recall and precision, it is low because the precision is not very good.**

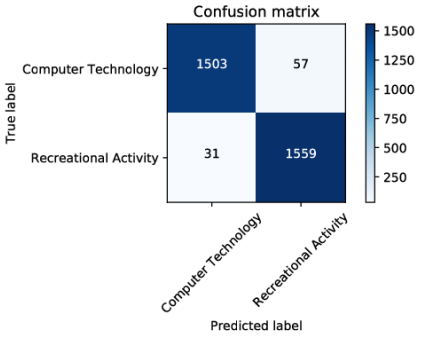

Fig. 3. Confusion matrix for linear SVM with hard margin ($\gamma = 1000$)

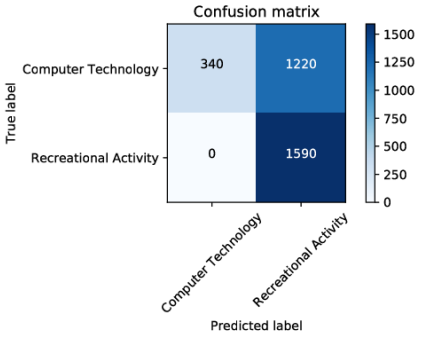

Fig. 4. Confusion matrix for linear SVM with soft margin ($\gamma = 0.0001$)

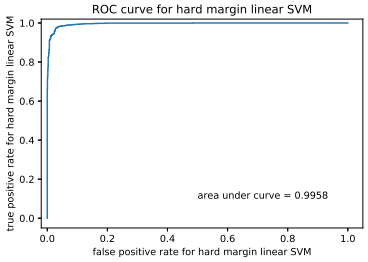

Fig. 5. ROC curve for linear SVM with hard margin ($\gamma = 1000$)

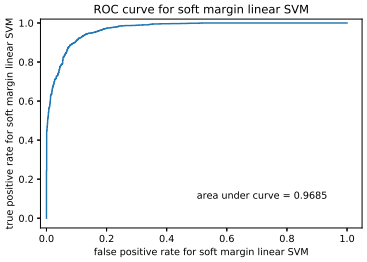

Fig. 6. ROC curve for linear SVM with soft margin ($\gamma = 0.0001$)

Cross Validation to choose best γ: In this part of the question, we have used 5-fold cross validation to choose the best value of γ on the basis of average validation accuracy (as stated in the question).

We use GridSearchCV() function from the sklearn library that allows us to search for the best parameters. Here, the problem states that we need to find the best value of γ for $\gamma = 10^{-k}$ where $k \in [-3, 3]$ and $k \in Z$. We do a grid search over 5-fold cross validation in order to look for the best value of γ on the basis of average validation accuracy. We optimize the value of γ using a 5-fold cross validated grid search and present the results in table 4.

Value of γ	Average Accuracy Score	Standard Deviation
0.001	0.505	0.000
0.01	0.506	0.002
0.1	0.965	0.012
1	0.973	0.013
10	0.976	0.009
100	0.973	0.008
1000	0.975	0.008

Table 4. 5-fold cross validation to find the best value of γ on the basis of average validation accuracy

Based on our analysis, **we found that $\gamma = 10$ produces the best average validation accuracy and hence we selected it to train another linear SVM classifier and plot its Confusion matrix** in Fig. 7. We plotted the ROC curve for the same in Fig. 8. The performance scores are analyzed and presented in table 5.

Accuracy score:	0.9711
Precision score:	0.9629
Recall score:	0.9805
F-1 score:	0.9716

Table 5. Performance analysis of the best linear SVM classifier on the basis of average validation accuracy

5 Logistic Regression

Logistic regression seeks to classify problems in a similar fashion to SVM classification—to find the "optimal" hyperplane that has the maximum probability of classifying unseen data points correctly. However, the definition of "optimal" differs between these two models. Unlike SVM classification which seeks to find a hyperplane with the greatest margin between separable data, logarithmic regression aims to calculate the probability that a data point belongs to class 1. To accomplish accurate classification, logistic regression takes the output of a linear function and maps the value to [0,1] using the sigmoid function, known as the logistic function:

$$\phi_{sig}(z) = \frac{1}{1 + \exp(-z)}.$$

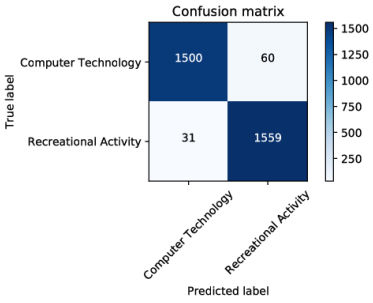

Fig. 7. Confusion matrix for linear SVM with best average validation accuracy ($\gamma = 10$)

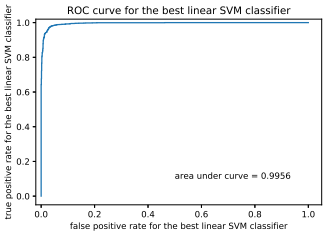

Fig. 8. ROC curve for linear SVM with best average validation accuracy ($\gamma = 10$)

5.1 Logistic Classification Without Regularization

Consequently, logistic regression seeks to find $\boldsymbol{w}$ and b to minimize the loss function $\phi_{sig}(z) = \frac{1}{1+\exp(-z)}$, where $z(x) = \boldsymbol{w}^T\boldsymbol{x} + b$, to best learn the data. Logistic regression minimizes a logistic function, which diverges more quickly than SVM's loss function. As a result, logistic regression is more sensitive to outliers than SVM. Additionally, logistic regression loss does not go to zero for correct classifications (unlike SVM, which produces 0 or 1), implying that logistic regression "assumes" that the data it's given is not complete.

5.2 Logistic Classification With Regularization

Logistic regression can be modified with the addition of a regularization term to the objective function. A regularization term is implemented to train models that don't overfit the training data to better predict unseen data. Logistic regression with regularization helps facilitates the balance between maximum likelihood and minimizing generalization error. Regularization also helps to simplify the model that is generated by penalizing complex equations higher order equations. Simplifying models helps to accurately generalize models despite the presence of many irrelevant features. Although several regularization equations exist, including Gauss and Laplace regularization, we focus on L1 and L2 regularization in this report. L1 regularization is given by

$$\phi_{sig}(\theta^T z) = \frac{1}{1+\exp(-\theta^T z)},$$
$$\text{subject to } \|\theta\|_1 \leq C,$$

where C is the regularization strength. Similarly, L2 regularization is given by

$$\phi_{sig}(\theta^T z) = \frac{1}{1+\exp(-\theta^T z)}.$$
$$\text{subject to } \frac{1}{2}\|\theta\|_2^2 \leq C.$$

Question 5 We trained an **unregularized logistic classifier** on our training data set reduced by LSI. Fig. 9 and Fig. 10 show the confusion matrix and and ROC curve of our classifier. In the confusion matrix, we notice that the true positive and true negative categories are both high, signifying a good classifier. This is reflected in the ROC curve, as the area under the curve is 0.9961. As for the different scores, this classifier achieved:

Accuracy score:	0.9711
Precision score:	0.9624
Recall score:	0.9811
F-1 score:	0.9717

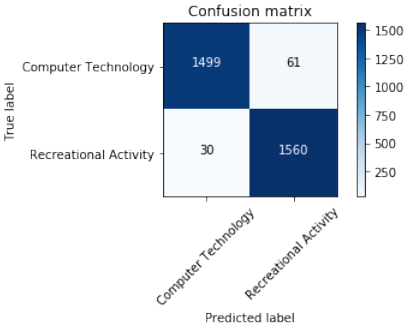

Fig. 9. Confusion matrix for unregularized Logistic Regression

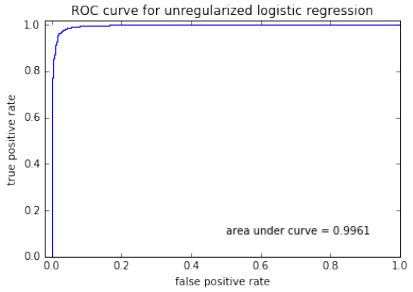

Fig. 10. ROC curve for unregularized Logistic Regression

We trained two **regularized logistic regressions** with **L1 and L2 regularization**. Using 5-fold cross validation on the dimension-reduced-by-SVD training data, we found the optimal regularization strength C in the range $\{10^k | -3 \leq k \leq 3, k \in \mathbb{Z}\}$ for logistic regression with L1 and L2 regularization. We found that the best **regularization strength for logistic regression with L1 regularization is 0.1**. We found that the best **regularization strength for logistic regression with L2 regularization is 0.01**.

Fig. 11 and Fig. 12 show the ROC curves of logistic regression with L1 and L2 regularization, respectively. The logistic regression with L1 regularization performed well, which is demonstrated in the ROC curve, as the area under the curve is 0.9958. The logistic regression with L2 regularization also performed well, which is demonstrated in the ROC curve, as the area under the curve is 0.9956. As for the different scores, **logistic regression with L1 regularization achieved:**

Accuracy score:	0.9708
Precision score:	0.9606
Recall score:	0.9824
F-1 score:	0.9714

Logistic regression with L2 regularization achieved:

Accuracy score:	0.9702
Precision score:	0.9600
Recall score:	0.9818
F-1 score:	0.9708

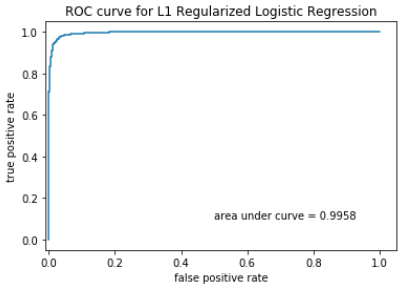

Fig. 11. ROC curve for L1 Regularized Logistic Regression

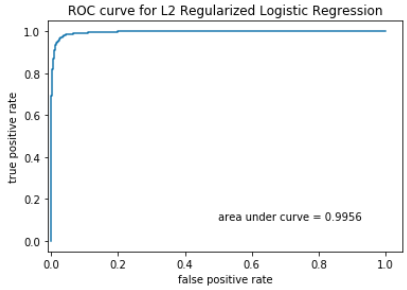

Fig. 12. ROC curve for L2 Regularized Logistic Regression

Logistic regression without regularization performed slightly better than L1 and L2 regularized classification (with parameters of 0.1 and 0.01, respectively) in the categories of accuracy, precision, and F-1 score. Additionally, regression without regularization had a higher area-under-ROC curve than regularized classification. However, both L1 and L2 regularized classifiers had a higher recall score compared to the unregularized classifier.

The addition of the regularization parameter marginally affects the test error compared to unregularized logistic regression, as shown above. The implementation of the regularization parameter "encourages" the model to simplify its learnt coefficients, such that more complex models are penalized. As a result, regularized logistic regression models are typically less overfit to the training data, since simplification helps models accurately generalize models despite the presence of many irrelevant features.

L1 regularization seeks to minimize the sum of absolute differences between the target value and the estimated values. L2 regularization minimizes the sum of the square of the differences between the target value and the estimated values. Consequently, **L1 regularization is more resistant to outliers compared**

to L2 regularization, since L2 regularization considers squared error. However, since L2 regularization is more sensitive to outliers, **L2 regularization is less susceptible to horizontal changes in the data compared to L1 regularization.**

Logistic regression seeks to classify problems in a similar fashion to SVM classification—to find the "optimal" line that has the maximum probability of classifying unseen data points correctly. However, the definition of "optimal" differs between these two models. As is aforementioned, **Unlike SVM classification which seeks to find a hyperplane with the greatest margin between separable data, logarithmic regression aims to calculate the probability that a data point belongs to class 1.** To accomplish accurate classification, logistic regression takes the output of a linear function and maps the value to [0,1] using the sigmoid function. Thus, logistic regression seeks to find $\boldsymbol{w}$ and b to minimize the loss function $\phi_{sig}(z) = \frac{1}{1+\exp(-z)}$, where $z(x) = \boldsymbol{w}^T\boldsymbol{x} + b$, to best learn the data. Logistic regression minimizes a logistic function, which diverges more quickly than SVM's loss function. As a result, **logistic regression is more sensitive to outliers than SVM.** Additionally, **logistic regression loss does not go to zero for correct classifications (unlike SVM, which produces 0 or 1), implying that logistic regression "assumes" that the data it's given is not complete.**

6 Naïve Bayes

The Naïve Bayes classifier is a type of probabilistic classifier where it is assumed that the probability of the document is from a particular class is independent from the other documents. In other words, the equation below holds true for the m documents in the corpus, and each document can be considered a separate classification problem.

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

This project uses the Maximum A Posteriori (MAP) decision rule to classify documents in the corpus. The category having the highest probability of being correct for a given x_i is the one that is assigned to x_i. Mathematically, the Naïve Bayes classifier implements the following decision rule, where K represents the number of unique categories in the corpus.

$$\hat{y} = \underset{k \in \{1,...,K\}}{\operatorname{argmax}} \; P(y) \prod_{i=1}^{m} P(x_i|y)$$

Question 6 A Gaussian Naïve Bayes (NB) classifier is trained on the train subset reduced using the LSI metric. Fig. 13 shows the **un-normalized confusion matrix** after fitting. Fig. 14 shows the **ROC of the classifier trained using the subset reduced by LSI**. The **scores** which the Gaussian NB classifier

achieved when fitted with the LSI-reduced subset is shown in Table 6.

The performance of the Gaussian NB does not come close to that of Logistic Regression or SVM. One reason why this might be the case is that the Gaussian NB assumes the data distribution to be fairly separable and to have a Gaussian probability distribution. Although these assumptions may be true in many real-world cases, we have not proven that they hold true for our corpus of documents.

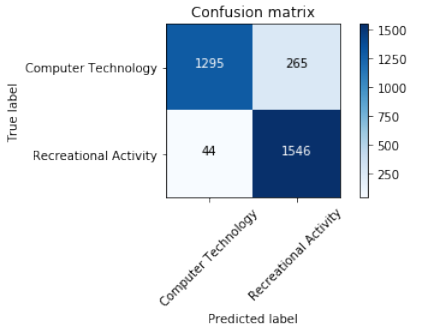

Fig. 13. Confusion matrix for Gaussian NB from LSI reduction.

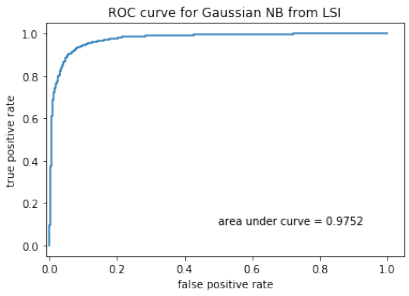

Fig. 14. ROC curve for Gaussian NB from LSI reduction.

Accuracy score:	0.9019
Precision score:	0.8532
Recall score:	0.9723
F-1 score:	0.9091

Table 6. Gaussian Naïve Bayes scores

7 Grid Search of Parameters

There are many parameters that can be tuned during the process of loading the data, extracting features, reducing the feature dimension, and classifying the documents using these features. In order to find the best combination of parameters, each combination is evaluated. A grid search process is used to maximize the efficiency of this test. The test accuracies were derived using 5-fold cross validation. The parameters that were tested are given in Table 7.

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 Gaussian Naïve Bayes classifier

Table 7. The parameters that were tuned, and options therein.

Question 7 The **best combination of parameters** is presented in Table 8. The complete results of the grid search is shown in the Appendix. It is evident that the SVM model with best γ outperformed the other classifiers, because the top 16 classifiers are the SVM model. Of those top 16 classifiers, the best 7 were trained using the LSI-reduced subset. The combination of keeping headers and footers with a min_df of 3 appears in the top 2 classifiers. It appears that the practice of lemmatization was not as important, because no pattern can be derived from the outcome of the grid search.

An interesting conclusion from this grid search can be made for the Naïve Bayes classifiers. The best-performing Gaussian NB classifiers were using the NMF-reduced subset. Although Logistic Regression and and SVM both benefited from using the LSI-reduced subset, Gaussian NB favored NMF. Perhaps when the data is reduced by NMF, it is more representative of what the Gaussian NB classifier is expecting.

Procedure	**Best option**
Loading data	keep headers and footers
Feature Extraction	min_df = 3 lemmatize
Dimensionality Reduction	LSI
Classifier	SVM with $\gamma = 1000$ (hard margin)

Table 8. The best parameters found via grid search.

8 Multiclass Classification

In the previous experiments, binary classification was performed to sort the corpus of documents into two categories. The following experiments will sort the corpus of documents into four distinct categories. The ROC curve is no longer a reasonable guideline since there are multiple False-Positive and False-Negative values. However, the confusion matrix, accuracy, precision, recall, and F1-score metrics can still be used to analyze multiclass classifiers if all of the False-Positives and False-Negatives are summed together.

The four categories being classified are "PC Hardware", "Mac Hardware", "Misc Forsale", and "Christian Religion". Although the classifiers were able to distinguish between "PC Hardware" and "Mac Hardware" very well, it becomes clear that adding two more classes further complicates the matter.

8.1 Multiclass Naïve Bayes

The NB classifier can naturally be expanded to fit however many classes the data contains. The equations described in Section 6 hold true for any value of K, as long as $K > 1$.

The **confusion matrix for the Gaussian NB classifier fit to the four classes** is shown in Fig. 15. The corresponding accuracy, precision, recall, and F-1 scores are given in Table 9. The classifier only achieves 67% accuracy on classifying all four categories. The conclusions drawn from Section 7 suggest that by using the NMF-reduced subset, the classifier scores may improve. Fig. 16 and Table 10 give these results, and show that there is an impressive 10% improvement when the Gaussian NB classifier fits on the training subset reduced by the NMF instead of the LSI method.

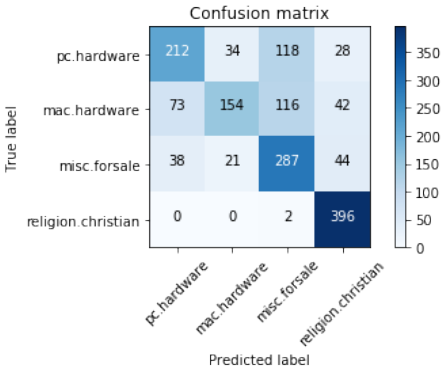

Fig. 15. Confusion matrix for Gaussian NB with LSI

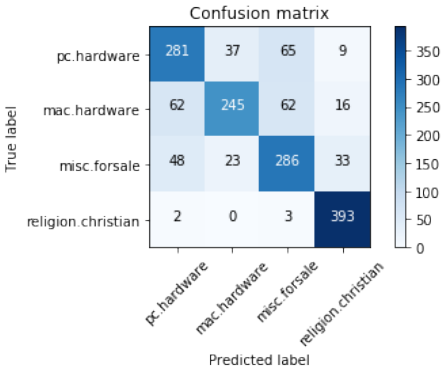

Fig. 16. Confusion matrix for Gaussian NB with NMF

Accuracy score:	0.6703
Precision score:	0.6796
Recall score:	0.6679
F-1 score:	0.6531

Table 9. Gaussian Naïve Bayes multiclass scores with LSI

Accuracy score:	0.7700
Precision score:	0.7693
Recall score:	0.7685
F-1 score:	0.7654

Table 10. Gaussian Naïve Bayes multiclass scores with NMF

As was previous discussed, the poor performance of the Gaussian NB classifier in comparison to SVM and Logistic Regression may be due to falsely assuming the dataset is a fairly separable set of gaussian distributions. Instead, let us consider the Bernoulli NB classifier for this case. Fig. 17 and Table 11 show that the Bernoulli NB classifier trained on the LSI-reduced subset achieves better scores than the Gaussian NB classifier trained on LSI or NMF. The Bernoulli NB classifier may therefore be a better model than Gaussian NB when categorizing this corpus of documents.

Accuracy score:	0.7744
Precision score:	0.7781
Recall score:	0.7735
F-1 score:	0.7730

Table 11. Bernoulli Naïve Bayes multiclass scores with LSI

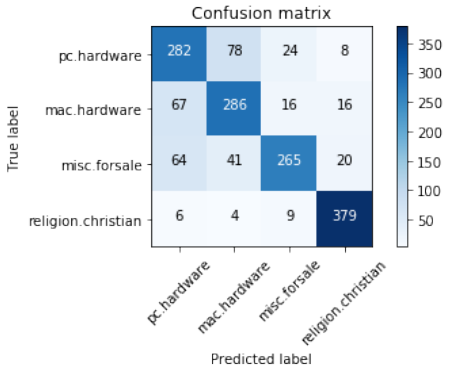

Fig. 17. Confusion matrix for Bernoulli NB with LSI

8.2 SVM Multiclass Classification

Theoretically, SVM is a binary classifier. This means it can choose one class out of two. In order to do a multiclass classification using SVM, there are two ways:

1. **One vs One Classification:** Let's say we start with k classes. In one vs one classification, SVM trains $\frac{k*(k-1)}{2}$ classifier. Each classifier trains data from 2 different classes. For example, if we have 4 classes, then a total of $\frac{4*3}{2} = 6$ classifiers will be trained between classes 1 and 2, classes 1 and 3, classes 1 and 4, classes 2 and 3, classes 2 and 4, classes 3 and 4.

2. **One vs Rest Classification:** In this method, SVM trains one classifier per class. So if we have k classes, k classifiers will be trained. For example, if we have 4 classes, we will have 4 classifiers between class 1 and the union of classes (2,3,4), class 2 and the union of classes (1,3,4), class 3 and the union of classes (1,2,4), class 4 and the union of classes (1,2,3). Hence, it is called a one vs rest classification because it creates one class out of all the remaining classes.

In our code, we have used OneVsOneClassifier() from sklearn to implement one vs one classification and OneVsRestClassifier() to implement one vs rest classification. Here's a code snippet depicting how these clasifiers look:

```
# One vs Rest classifier
clf_ovr = OneVsRestClassifier(LinearSVC(C=10, random_state=42))

# One vs one classifier
clf_ovo = OneVsOneClassifier(LinearSVC(C=10, random_state=42))
```

From the code snippet it is apparent that we have used the value of $C = \gamma = 10$ as this was the best performing value as shown in Question 4, table 4.

The OneVsRestClassifier() and OneVsOneClassifier() functions take in an estimator in the form of LinearSVC. Hence, here the classifier is linear SVM. The results for both the approaches are tabulated in the table] 12 and the Confusion matrix for one vs one classification is plotted in Fig. 18 while the Confusion matrix for one vs rest classification is plotted in Fig. 19.

From the results, **we conclude that one vs one classification performs better in case of linear SVM but only by a narrow margin**.

Scores	One vs One Linear SVM	One vs Rest Linear SVM
Accuracy Score	0.8900958466453675	0.8773162939297124
Precision Score	0.8902178899532479	0.8767027441500496
Recall Score	0.8895063359708012	0.8767403301412634
F1 Score	0.8897944756720788	0.8766248233516013

Table 12. Performance comparison of One vs One and One vs Rest linear SVM classifier.

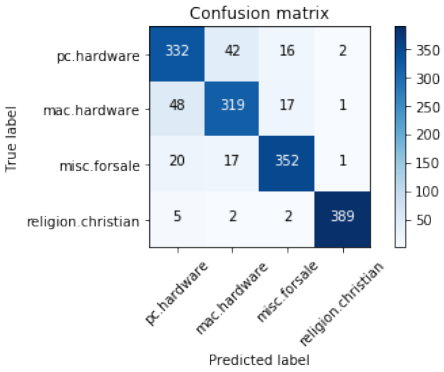

Fig. 18. Confusion matrix for linear one vs one SVM with $\gamma = 10$

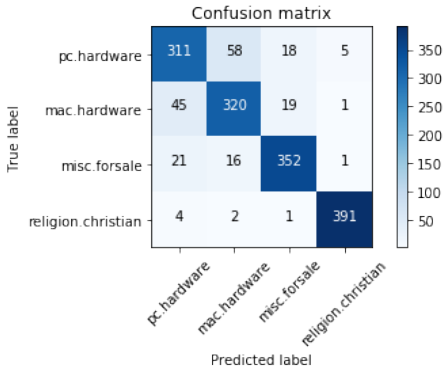

Fig. 19. Confusion matrix for linear one vs rest SVM with $\gamma = 10$

9 Conclusion

Based on our results, we find that linear SVM classifier performs the best on the given dataset. In order to get this result, the best gamma was experimentally found. This result was compared to the different combinations of Linear Regression and Gaussian Naïve Bayes models. The Gaussian NB classifier performed very poorly; it was found that the Bernoulli model was a better fit than the Gaussian model used in these experiments. This suggests that further analysis of the dataset may improve the scores of SVM, Logistic Regression, and Naïve Bayes classifiers.

10 Appendix

10.1 English stopwords

any, cry, her, itself, elsewhere, such, hundred, hers, out, back, get, anything, may, system, whole, yours, thru, along, not, ltd, when, anyway, further, detail, name, than, once, fire, whose, whither, their, neither, at, down, his, something, anyone, to, interest, are, first, etc, without, mill, nobody, nothing, often, during, about, were, while, becoming, all, many, very, then, too, wherever, keep, call, see, forty, you, nor, that, a, ie, eg, same, done, beside, these, via, your, take, whereas, what, bill, i, enough, besides, afterwards, although, must, our, until, give, whenever, above, bottom, been, somewhere, the, is, except, thereafter, again, formerly, himself, everything, for, cannot, have, so, over, them, through, would, someone, from, herself, being, whatever, mine, within, every, full, onto, couldnt, where, front, already, before, as, themselves, can, noone, amoungst, beforehand, empty, hereafter, nevertheless, much, most, latter, do, anywhere, and, con, could, other, sixty, beyond, sometimes, my, an, everyone, whoever, with, in, namely, him, who, move, others, ourselves, seemed, whom, how, never, thus, always, myself, per, twelve, un, below, there, everywhere, well, whereupon, here, meanwhile, somehow, towards, up, if, might, next, fifty, seeming, sometime, also, ten, together, top, under, several, six, indeed, he, more, amongst, former, latterly, or, co, rather, five, should, go, perhaps, side, though, whether, nowhere, inc, please, across, put, thick, serious, whereby, after, am, it, made, against, hence, will, few, they, each, hereupon, be, no, had, one, me, since, de, thereby, ever, third, eight, however, still, wherein, has, seem, its, thin, else, became, therefore, either, thereupon, amount, alone, into, by, own, those, among, twenty, was, upon, show, none, some, whence, which, yet, this, even, find, mostly, fill, re, she, anyhow, thence, cant, yourself, four, because, both, due, toward, yourselves, on, behind, two, around, three, but, us, herein, last, moreover, hasnt, least, only, fifteen, eleven, otherwise, another, become, found, less, now, of, ours, sincere, between, we, why, seems, therein, almost, becomes, throughout, describe, hereby, part, whereafter, off, nine

10.2 Results of Grid Search

#	mean_test_score	mean_train_score	param_clf	std_train_score	std_test_score	⋯
2	0.974852	0.978497	SVM, $\gamma = 10$	0.001463	0.004456	
0	0.974218	0.978445	SVM, $\gamma = 10$	0.001027	0.004084	
3	0.973584	0.978128	SVM, $\gamma = 10$	0.001025	0.005094	
8	0.972739	0.975909	SVM, $\gamma = 10$	0.001676	0.004964	
1	0.972527	0.977811	SVM, $\gamma = 10$	0.000768	0.004374	
9	0.971893	0.975116	SVM, $\gamma = 10$	0.001599	0.003371	
11	0.965131	0.967984	SVM, $\gamma = 10$	0.001728	0.004619	
5	0.964708	0.962806	SVM, $\gamma = 10$	0.001809	0.002706	
6	0.964497	0.964286	SVM, $\gamma = 10$	0.002141	0.003976	
10	0.964286	0.968354	SVM, $\gamma = 10$	0.002110	0.005401	⋯
4	0.963440	0.960218	SVM, $\gamma = 10$	0.001035	0.008078	
7	0.961116	0.965184	SVM, $\gamma = 10$	0.002485	0.002608	
13	0.960270	0.962331	SVM, $\gamma = 10$	0.001361	0.010736	
12	0.959637	0.959848	SVM, $\gamma = 10$	0.004019	0.006518	
14	0.955833	0.957418	SVM, $\gamma = 10$	0.002358	0.008580	
15	0.954987	0.957946	SVM, $\gamma = 10$	0.002782	0.007748	
16	0.955199	0.956942	LR, with 'L1'	0.001027	0.002529	
17	0.952240	0.955199	LR, with 'L1'	0.001517	0.004761	
61	0.949281	0.949387	Gaussian NB	0.003661	0.005299	
54	0.943576	0.940670	Gaussian NB	0.004120	0.008199	

	param_import__remove	param_reduce_dim	param_vect__analyzer	param_vect__min_df
	keep headers/footers	LSI	lemmatizer on	3
	keep headers/footers	LSI	lemmatizer off	3
	keep headers/footers	LSI	lemmatizer on	5
	remove headers/footers	LSI	lemmatizer off	3
	keep headers/footers	LSI	lemmatizer off	5
	remove headers/footers	LSI	lemmatizer off	5
	remove headers/footers	LSI	lemmatizer on	5
	keep headers/footers	NMF	lemmatizer off	5
	keep headers/footers	NMF	lemmatizer on	3
⋯	remove headers/footers	LSI	lemmatizer on	3
	keep headers/footers	NMF	lemmatizer off	3
	keep headers/footers	NMF	lemmatizer on	5
	remove headers/footers	NMF	lemmatizer off	5
	remove headers/footers	NMF	lemmatizer off	3
	remove headers/footers	NMF	lemmatizer on	3
	remove headers/footers	NMF	lemmatizer on	5
	keep headers/footers	LSI	lemmatizer off	3
	keep headers/footers	LSI	lemmatizer off	5
	remove headers/footers	NMF	lemmatizer off	5
	keep headers/footers	NMF	lemmatizer on	3

Bibliography

[1] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient l˜ 1 regularized logistic regression," in *AAAI*, vol. 6, pp. 401–408, 2006.

[2] V. Roychowdhury, "Classification analysis on textual data," University of California, Los Angeles, 2019.