Project 1: Classification Analysis on Textual Data

ECE 219 Spring 2020

Jake Tsuyemura, UID 705078434

Ryan Chau, UID 704286024

Introduction

This project is designed to complete textual data classification of the “20 Newsgroups” dataset utilizing learning algorithms such as Linear Support Vector Machines, Logistic Regression, and Naïve Bayes. For each of these methods, training and test documents from 20 Newsgroups are loaded and processed through tokenization, lemmatization and stemming, and eventually into tf-idf matrices. These matrices are then run through dimensionality reduction methods, such as Non-negative Matrix Factorization (NMF) and Latent Semantic Indexing (LSI), in order for the learning algorithms to perform well. In this project, we primarily employ LSI for dimensionality reduction. We train the learning algorithms using the training set and vary parameters such as regularization strength to optimize training accuracy. Finally, the algorithms are run on the test set so that we can collect and display data, such as the confusion matrix and ROC plot.

Question 1

The 20 Newsgroup data contains approximately 20,000 documents partitioned into 20 different newsgroups. We plot a histogram of the number of training documents in each category, in order to verify that the dataset has roughly the same number of documents per category.

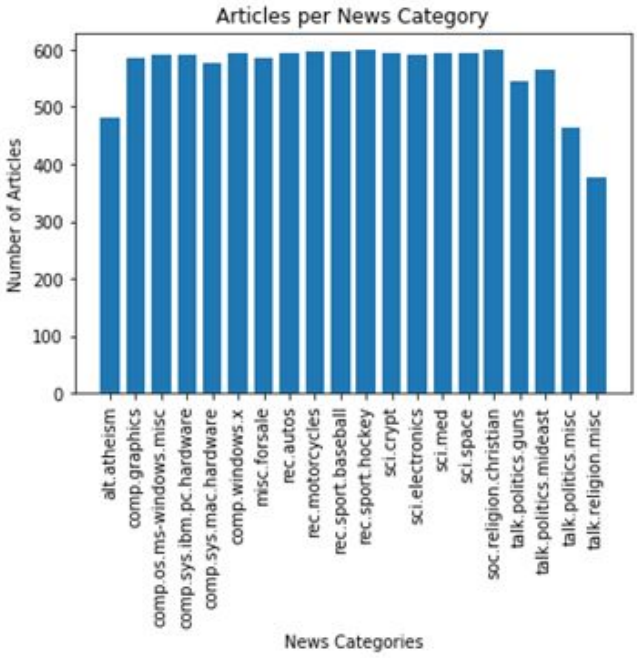

Figure 1.1: The number of documents in each Newsgroup category

From **Figure 1.1**, we see that most categories contain 500-600 documents in the training set, while only alt.atheism, talk.politics.misc, and talk.religion.misc have fewer than 500 documents.

Question 2

For Questions 2 through 7, we will only use 8 of the previously shown 20 Newsgroups. Those groups are shown below in Figure 2.1 and they are used to represent two larger classes of documents, “Computer Technology” and “Recreational Activity”

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

Figure 2.1: 8 categories from the 20 Newsgroups that will be used for training and testing

Note that the top row of **Figure 2.1** contains documents belonging to the “Computer Technology” class and the bottom row contains “Recreational Activity” class documents.

Next, we extract meaningful features from the textual data by running each document from the 8 categories through tokenization, then stemming and lemmatization. Tokenization is achieved by removing punctuation, numbers, and stopwords from each document. Sentences within each document are then separated, with parts of speech tags being paired with each word. Then, lemmatization and stemming are conducted to create a Bag of Words representation for each document. Finally, we use the TfidfTransformer function to create the TF-IDF matrix that will be passed into the next set of operations. The shape of the TF-IDF matrices for the train and test sets are given below in **Figure 2.2**.

TF-IDF Matrix	Shape
Training set	(4732, 10847)
Test set	(3150, 8290)

Figure 2.2: The TF-IDF matrix shape for each set of data

Question 3

Once the TF-IDF matrices are computed, we compare the effect of LSI on the TF-IDF matrix vs NMF. Both methods reduce each document within the TF-IDF matrix to a 50-dimensional vector, so the TF-IDF in whole is reduced from (4732, 10847) to (4732, 50) for the training set. Below, we see the following minimization values for LSI and NMF.

Minimization terms	Values
LSI:	4085.134
NMF:	4121.635

Figure 3.1. The values of each optimization problem for LSI and NMF

From **Figure 3.1**, we see that the objective function for NMF is larger.

Question 4

In this step, we use the LSI dimension-reduced TF-IDF matrix as input into the Linear SVM learning algorithm. The learning algorithm will perform binary classification of documents into either “Computer Technology” or “Recreational Activity” categories, referenced in **Figure 2.2**. To assess the difference in performance between hard margin SVMs and soft margin SVMs in our binary classification, we assign a gamma value of 1000 for our hard margin SVM and a gamma value of 0.0001 for our soft margin SVM. Confusion matrices for soft margin SVM and hard margin SVM are shown below, as well as a table comparing accuracy, precision, recall, and F1-score.

	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**0**	1560
True label: Recreational Activity	0	**1590**

Figure 4.1. The confusion matrix for the soft margin SVM.

	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1499**	61
True label: Recreational Activity	31	**1559**

Figure 4.2 The confusion matrix for the hard margin SVM.

Method	Accuracy	Precision	Recall	F1-score
Soft Margin SVM	0.50476	0.50476	1.00000	0.67089
Hard Margin SVM	0.97079	0.96234	0.98050	0.97134

Figure 4.3. Statistics gathered from confusion matrix data for soft and hard margin SVM

Figures 4.1, **4.2**, and **4.3** above show that the hard margin SVM performs much better than the soft margin SVM. From the confusion matrix in **Figure 4.1**, we can see that the soft margin SVM classifies every document as Recreational Activity. Although the soft margin model

has a recall of 1 in **Figure 4.3**, meaning it has correctly identified all of the true Computer Technology documents, the precision is low due to many false positives. The gamma value being as low as 0.0001 causes the SVM to have a very loose margin for classifying a document, and any errors are not penalized. As a result, the predictions are severely underfitted to the training data. On the other hand, the hard margin SVM using a high gamma value of 1000 means that errors are penalized more heavily, resulting in a stricter margin. **Figure 4.3** shows a high accuracy of 97.079% for hard margin SVM. Corresponding ROC plots for soft margin and hard margin SVM are shown below.

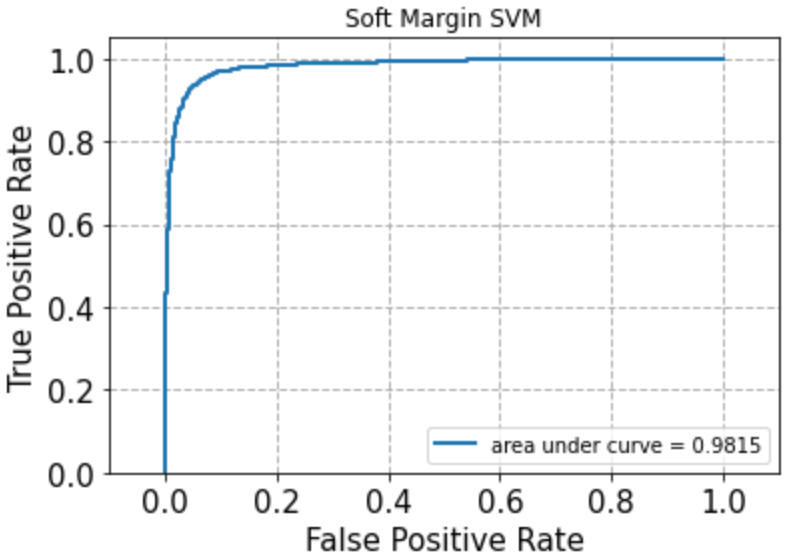

Figure 4.4. The ROC plot for the soft margin SVM.

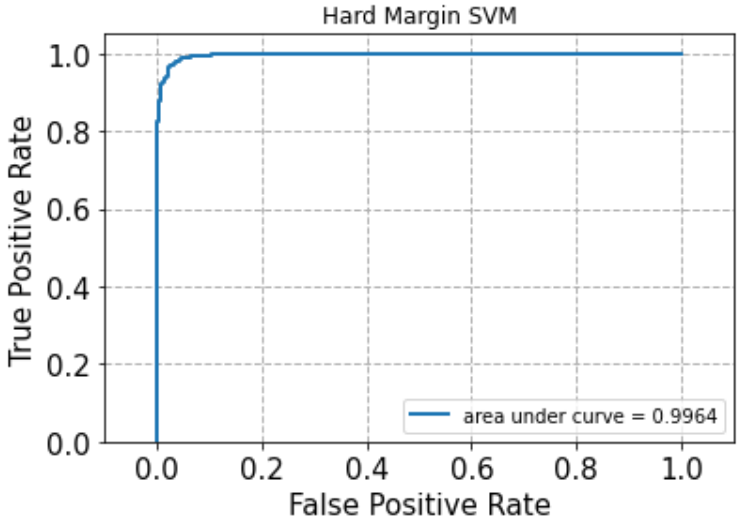

Figure 4.5. The ROC plot for the hard margin SVM.

The ROC plots for both the hard margin SVM and soft margin SVM look like plots from high performing algorithms. For the soft margin SVM in particular, the ROC plot in **Figure 4.4** appears to conflict with the low accuracy and precision values in **Figure 4.3**, but those values are not parameters of the ROC plot. We see that a rough value of 0.5 false positive rate calculated from the confusion matrix in **Figure 4.1** corresponds to a recall value near 1 on the ROC plot, which is consistent with the recall value of 1 in the **Figure 4.3** statistics.

The gamma value experiments so far support the use of higher gamma values, which corresponds to hard margin SVM. However, extremely high gamma values can lead the model to overfitting to the training data. To find the best value of gamma to use, we perform 5-fold cross validation and determine the best accuracy rate out of the following gamma values: 0.001, 0.01, 0.1, 1, 10, 100, 1000.

Gamma values	Mean Accuracy
0.001	0.57523
0.01	0.88989
0.1	0.94632
1	0.96196
10	0.97125
100	0.97559
1000	0.96703

Figure 4.6. The test accuracy corresponding to SVM with various gamma values

The highest mean accuracy from 5-fold validation was given by setting gamma = 100. The confusion matrix for gamma = 100 is shown below, as well as statistics gathered from the confusion matrix.

	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1494**	66
True label: Recreational Activity	31	**1559**

Figure 4.7. The confusion matrix corresponding to SVM with gamma = 100

Accuracy	Precision	Recall	F1-score
0.96921	0.95938	0.98050	0.97134

Figure 4.8. Statistics gathered from the confusion matrix for SVM with gamma = 100

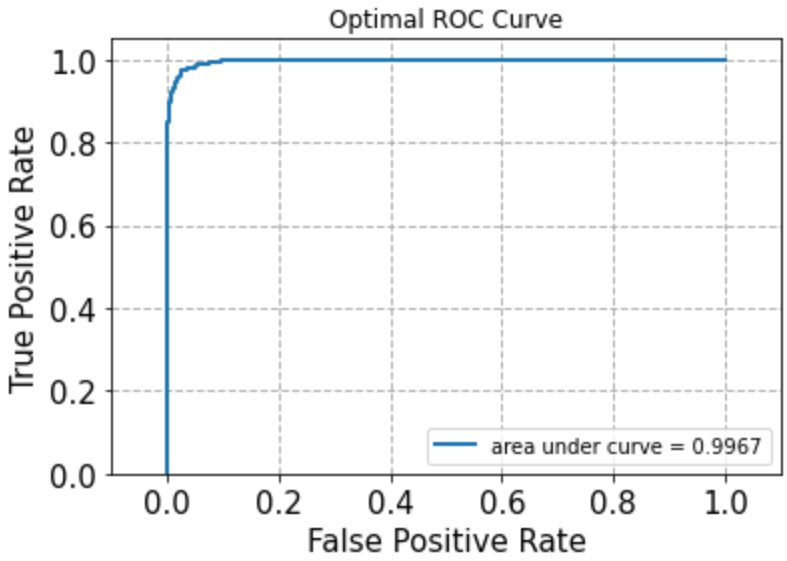

Figure 4.9 The ROC plot for SVM with gamma = 100.

For the value of gamma = 100 , we see high performance with an accuracy 96.921% from **Figure 4.8**, although this is lower than the mean accuracy of 97.559% with gamma = 100 during 5-fold validation.

Question 5

In this problem, we explored the effects of regularization on a logistic regression classification model. We find the optimal regularization strength (C-value) of each model using 5-fold cross validation. We then compare the results; accuracy, recall, precision, and F1-scores.

The settings used to perform Logistic Regression with/without regularization are shown below in **Figure 5.1**. *SkLearn LogisticRegression()* library was used to perform all data modeling. Data input, Vectorizer, TF-IDF, and Reducer variables were kept consistent throughout all modeling and validation testing.

Data	Vectorizer	TF-IDF Transformer	Reducer	Classifier
No Regularization				
Stemmed, Lemmatized, Punc. removed, lowercase	CountVectorizer (min_df=3)	TfidfTransformer (default)	NMF (n_comp=50)	Logistic Regression (penalty=none)
L1 - Regularization				
Stemmed, Lemmatized, Punc. removed, lowercase	CountVectorizer (min_df=3)	TfidfTransformer (default)	NMF (n_comp=50)	Logistic Regression (penalty=l1, C=1000)
L2 - Regularization				
Stemmed, Lemmatized, Punc. removed, lowercase	CountVectorizer (min_df=3)	TfidfTransformer (default)	NMF (n_comp=50)	Logistic Regression (penalty=l1, C=1000)

Figure 5.1 Logistic Regression Model Settings

We iterated over a range of *LogisticRegresstion() '*C-values' to find the optimal settings for both L1 & L2 regularization models. The results are shown below in **Figure 5.2**. The optimal C-values (regularization strength) for both penalty functions was C=1000.

Regularization Strength (C-values)	L1 Mean Accuracy	L2 Mean Accuracy
0.001	0.4951	0.5055
0.01	0.4951	0.6025
0.1	0.6927	0.8994
1	0.9643	0.9518
10	0.9742	0.9656
100	0.9742	0.9734
1000	0.9746	0.9763

Figure 5.2 Optimal L1 & L2 Regression Strength (C-values)

These findings imply that a high C-value (1000) will improve model accuracy and not lead to underfitting. It is likely that even larger C-values (outside the bounds of Q5) would result in higher average accuracy. Low C-Values (0.001) will cause large amounts of model overfitting and very low accuracy. This is particularly harmful in an experiment where there are many unique terms in each document (sparse matrix).

Logistic Regression (No Regularization)

The statistics for Logistic Regression without any regularization is shown below in **Figure 5.3**. The model was able to produce a classification accuracy of ~97% over 5-fold cross validation. The shape of the ROC curve in **Figure 5.5** also reflects a highly accurate model.

Accuracy	Precision	Recall	F1-score
0.97079	0.96813	0.97421	0.97116

Figure 5.3 Logistic Regression (No Regularization) Statistics

(No Regularization)	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1505**	55
True label: Recreational Activity	38	**1552**

Figure 5.4 Logistic Regression (No Regularization) Confusion Matrix

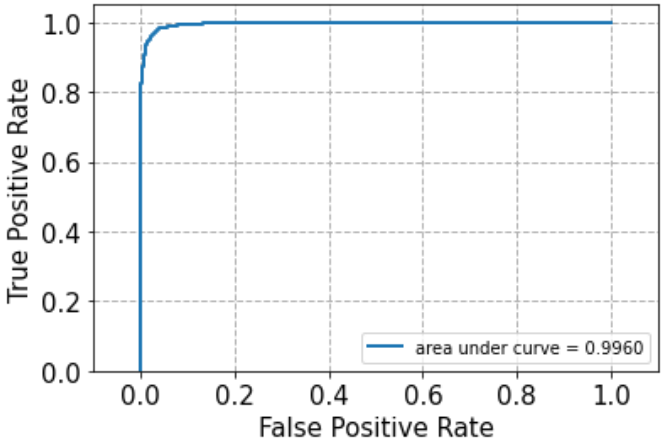

Figure 5.5 Logistic Regression (No Regularization) ROC-curve

Logistic Regression L1 & L2

Accuracy, precision, recall, and F1-scores were calculated using the optimal regularization strengths (C-values=1000) for both L1 and L2 regularization functions, shown below in **Figure 5.6**. Confusion matrices for both L1 & L2 regularization are shown in **FIgures 5.7-5.8** and ROC curves are shown in **Figure 5.9**.

Method	Accuracy	Precision	Recall	F1-score
L1 Regularization, C = 1000	0.97111	0.96698	0.9761	0.97152
L2 Regularization, C = 1000	0.96921	0.96166	0.97799	0.96975

Figure 5.6 Logistic Regression L1 & L2 Regularization Statistics

(L1 Regularization)	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1509**	51
True label: Recreational Activity	41	**1549**

Figure 5.7 Logistic Regression (L1) Confusion Matrix

(L2 Regularization)	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1498**	62
True label: Recreational Activity	35	**1555**

Figure 5.8 Logistic Regression (L2) Confusion Matrix

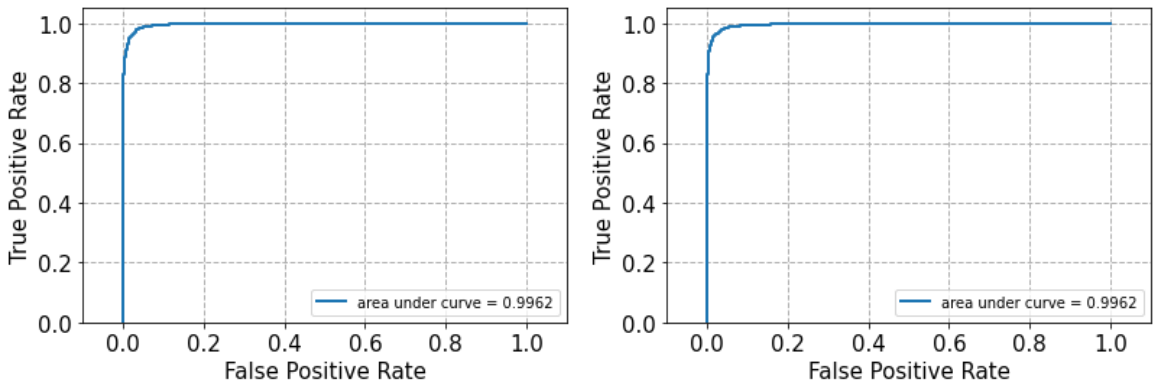

Figure 5.9 Logistic Regression ROC-curves: L1 (Left), L2 (Right)

Based on the above findings, when the regularization coefficient values for both L1 & L2 types are small (C=0.001), the classification accuracy is very low. This due to the large amounts of overfitting. Document-term matrices comprise of many individual features (words) that can form a very space feature matrix. Normalizing/regularizing the logistic regression model (both L1 & l2) with high C-values (1000) will reduce the impact of each specific feature (term), leading to higher test accuracy and better generalization capability.

L1 regularization outperforms L2 with an accuracy of ~97.1% vs. ~96.9%. Since the L1 penalty function applies regularization equally across all terms (unlike L2), it encourages feature sparsity by pushing small weights to zero. The feature matrix for this experiment is very sparse, with some words/terms only appearing in a small number of documents (df_min = 3).

Linear SVM decision boundaries are found by maximizing the margin between the nearest opposing classification points (support vectors). Logistic Regression fits a decision boundary based on maximum likelihood, resulting in a probability of classification between 0 and 1.

Logistic regression performs well when the classes are well defined, with data points that are grouped further away from the decision boundary. It returns the highest probability of a sample belonging to a specific class based on the probabilistic nature of document-term-matrix. Linear SVM works well with unstructured data, and focuses on only the support vector data points. As long as the support vectors for each class produce a large margin, Linear SVM will succeed. This allows the data within each class to have a wider feature variance. In this experiment, logistic regression with L1 regularization outperforms SVM based on scoring accuracy shown in in **Figures 5.6 & 4.3**.

Question 6

Naïve Bayes is another learning algorithm classifier, and we are specifically using GaussianNB for binary classification of the same documents into “Computer Technology” and “Recreational Activity” categories. The dimension-reduced TF-IDF matrix is fed into the Naive Bayes learning algorithm for this section. The confusion matrix, accuracy, and other statistics are shown below for this method.

	Predicted label: Computer Technology	Predicted label: Recreational Activity
True label: Computer Technology	**1474**	86
True label: Recreational Activity	40	**1550**

Figure 6.1. The confusion matrix for Naïve Bayes

Accuracy	Precision	Recall	F1-score
0.96000	0.94734	0.97484	0.93536

Figure 6.2: Statistics gathered from the confusion matrix for Naïve Bayes

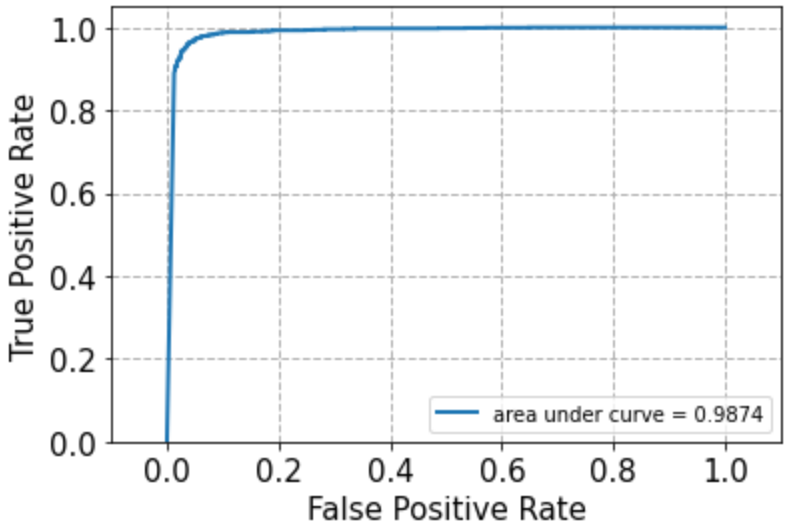

Figure 6.3: The ROC plot for Naïve Bayes binary classification

We see that the accuracy of this method is slightly lower than both linear SVM and logistic regression with optimal parameters, but an accuracy of 96% from **Figure 6.2** is still high-performing.

Question 7

Problems 1 thru 6 of this project explored the common feature-selection, dimensionality-reduction, and classification algorithms; each consisting of tunable parameters which affect the model's classification accuracy. *Sklearn Pipelines()* were constructed to easily compare the effects of these tunable parameters outlined below:

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 no lemmatization nor stemming
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

Figure 7.1: Options to Compare

The tunable parameters listed in **Figure 7.1** will result in 64 unique combinations. Each of these combinations are configured as independent *sklearn Pipelines()*. Performance statistics are gathered using 5-fold cross validation. Performance is assessed based on 'Mean Accuracy'. Statistics for all 64 pipeline iterations are listed in **Appendix 7A**. The Top 5 performing variable combinations are shown below in **Figure 7.2**.

Rank	Header Footer	Lemm & Stem	Classifier	Reducer	Mean Accuracy	Std Dev.
1	TRUE	TRUE	LogReg (C=100, penalty=l1)	TrunSVD (min_df=5)	0.87110	0.00471
2	TRUE	TRUE	LinSVC (C=100)	TrunSVD (min_df=5)	0.86996	0.00425
3	TRUE	TRUE	LogReg (C=100, penalty = l2)	TrunSVD (min_df=5)	0.86983	0.00416
4	TRUE	TRUE	LogReg (C=100, penalty = l2)	TrunSVD (min_df=3)	0.86869	0.00836
5	TRUE	TRUE	LogReg (C=100,	TrunSVD (min_df=3)	0.86843	0.00859

			penalty = l1)			

Figure 7.2: Top 5 Pipeline Combinations, Scoring = 'Mean Accuracy'

The findings in **Figure 7.2** suggest that all classification models are sensitive to the input data they receive. All top-5 models ingested data (news articles) that had undergone stemming, lemmatization, punctuation removal, lowercase-ing, and header/footer removal. This trend continues well beyond the top-5 performing models (See **Appendix 7A**).

After data ingestion, both *LinearSVD()* and *LogisitcRegression(l1 & l2)* classifiers performed well. All top-5 performance models used *TuncatedSVD()* as a feature-set reducer with *NFM()* being absent until rank #13. *Min_df* values of 5 performed better than 3, implying that a feature should be contained within a higher number of documents (5) before being considered significant.

The top performing model was able to achieve accuracy of ~87.1%, using 'cleaned' input data, higher minimum-document-frequency requirements (min_df), and a Logistic Regression classifier. Alternative configurations (NMF, min_df=3, raw input data) are found in the lower half of rankings shown in **Appendix 7A.**

Question 8

In this problem, we compare the multiclass classification abilities of the Naive Bayes, SVM one-to-one, and SVM one-to-rest algorithms. Accuracy, precision, recall, F1, and confusion matrices are calculated and compared. All fitting & training was validated with 5-fold cross validation. Average accuracy was used as the primary scoring metric.

Naive Bayes:

The Naive Bayes algorithm supports multiclass classification inherently; selecting the class with the highest maximum likelihood out of all possible classes. This allows us to use the *sklearn GaussianNB()* library to produce classification statistics. The settings for the CountVectorizer(), NMF() reducer, and GaussianNB() classifiers were chosen based on heuristics observed in problems 1 thru 7.

Data	Vectorizer	TF-IDF Transformer	Reducer	Classifier
Stemmed, Lemmatized, Punc. removed, lowercase	CountVectorizer (min_df=3)	TfidfTransformer (default)	NMF (n_comp=50)	GaussianNB (default)

Figure 8.1 **- Naive Bayes Multiclass Settings**

News Cat.	Precision	Recall	F1-score	Support
comp.sys.ibm.pc.hardware	0.66	0.70	0.68	392
comp.sys.mac.hardware	0.74	0.58	0.65	385
misc.forsale	0.67	0.76	0.71	390
soc.religion.christian	0.96	0.98	0.97	398
Accuracy			0.76	1565
Macro Avg	0.76	0.75	0.75	1565
Weighted Avg.	0.76	0.76	0.75	1565

Figure 8.2 **- Naive Bayes Statistics**

comp.sys.ibm.pc.hardware	**276**	42	66	8
comp.sys.mac.hardware	86	**222**	73	4
misc.forsale	56	34	**295**	5
soc.religion.christian	2	2	4	**390**
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian

Figure 8.3 **- Naive Bayes Multiclass Confusion Matrix**

Naive Bayes algorithm was able to achieve an average multiclass classification accuracy of ~76%. The confusion matrix in **Figure 8.3** shows *soc.religion.christian* as the class that was most accurate/easiest to classify.

SVM One-vs-One

Traditionally, SVM is a binary classification algorithm. To enable multiclass operation, SVM can be run multiple times in a one-vs-one configuration. Classification of the 4 news categories will require 6 individual comparisons. The *sklearn OneVsOneClassifier()* library made it easy to iterate over these combinations of binary comparisons. The settings for the *OneVsOneClassifier()* are shown below in **Figure 8.4**.

Data	Vectorizer	TF-IDF Transformer	Reducer	Classifier
Stemmed, Lemmatized, Punc. removed, lowercase	SVC (C=100, max_ter=5000, kernel=linear)	TfidfTransformer (default)	NMF (n_comp=50)	OneVsOne Classifier (default)

Figure 8.4 **- SVM One-vs-One Settings**

News Cat.	Precision	Recall	F1-score	Support
comp.sys.ibm.pc.hardware	0.76	0.79	0.76	392
comp.sys.mac.hardware	0.79	0.74	0.76	385
misc.forsale	0.87	0.88	0.87	390
soc.religion.christian	0.99	0.97	0.98	398
Accuracy			0.84	1565
Macro Avg	0.85	0.84	0.84	1565
Weighted Avg.	0.85	0.84	0.84	1565

Figure 8.5 **- SVM One-vs-One Statistics**

comp.sys.ibm.pc.hardware	**310**	57	25	0
comp.sys.mac.hardware	74	**284**	25	2

misc.forsale	33	15	**342**	0
soc.religion.christian	7	3	3	**385**
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian

Figure 8.6 **- SVM One-vs-One Confusion Matrix**

The One-vs-One SVM binary classification method significantly outperformed the Naive Bayes algorithm. Average classification accuracy for SVM was ~0.84% compared to ~76% for Bayes. The increase in accuracy can be attributed to additional computations and comparisons used to train 6 unique iterations of the SVC() classifier. Naive Bayes may also underperform due to it's assumption that all terms are independent, which is unlikely given the nature of human speech/sentence construction.

SVM One-Vs-Rest

Another way to use binary classification algorithms in a multiclass environment is the one-vs-rest approach. Each classifier is trained to pick between a single class (1), and a group consisting of the remaining classes (3). This requires fitting four unique models. The *sklearn OneVsRestClassifier()* was used. Settings are shown below in **Figure 8.7**. Statistics are shown in **Figures 8.8-8.9**.

Data	Vectorizer	TF-IDF Transformer	Reducer	Classifier
Stemmed, Lemmatized, Punc. removed, lowercase	SVC (C=100, max_ter=5000, kernel=linear)	TfidfTransformer (default)	NMF (n_comp=50)	OneVsRest Classifier (default)

Figure 8.7 **- SVM One-vs-Rest Settings**

News Cat.	Precision	Recall	F1-score	Support
comp.sys.ibm.pc.hardware	0.76	0.79	0.77	392
comp.sys.mac.hardware	0.80	0.75	0.77	385
misc.forsale	0.87	0.89	0.88	390

soc.religion.christian	0.98	0.98	0.98	398

Accuracy			0.85	1565
Macro Avg	0.85	0.85	0.85	1565
Weighted Avg.	0.85	0.85	0.85	1565

Figure 8.5 **- SVM One-vs-Rest Statistics**

comp.sys.ibm.pc.hardware	**309**	56	24	3
comp.sys.mac.hardware	67	**288**	26	4
misc.forsale	27	15	**347**	1
soc.religion.christian	4	2	1	**391**
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian

Figure 8.6 **- SVM One-vs-Rest Confusion Matrix**

The *OneVsRestClassifier()* was able to achieve an average classification accuracy of ~85%, slightly better than the *OneVsOneClassifier()* ~84%. Given the reduced number of fittings (4) required vs the *OneVsOneClassifier()* (6), the *OneVsRestClassifier()* is the better option. The *OneVsOneClassifier()* may be prone to overfitting due to the larger amount of model fittings and comparisons.

The confusion matrix across all models shows that the *soc.religion.christian* class is the easiest to classify, with *comp.sys.mac.hardware* being the least accurate. This illustrates how the models were able to associate different features (terms) in the documents with specific classes (news categories). All three models struggled with differentiating features between the *comp.sys.ibm.pc.hardware* & *comp.sys.mac.hardware* classes, implying that these two news categories use similar terms/sentence structures that can't be easily attributed to either category.

Appendix

Appendix 7A: Full Pipeline Statistics - 5-fold cross-validation, Scoring = 'Mean Test Score'

Rank	Header Footer	Lem Stem	Class.	C value	Penalty (L1, L2)	Dim. Red.	# Feat.	min_df	Mean Test Score:	Std Dev.
1	TRUE	TRUE	LogReg	100	l1	TrunSVD	50	5	0.87110	0.00471
2	TRUE	TRUE	LinSVC	100	N/A	TrunSVD	50	5	0.86996	0.00425
3	TRUE	TRUE	LogReg	100	l2	TrunSVD	50	5	0.86983	0.00416
4	TRUE	TRUE	LogReg	100	l2	TrunSVD	50	3	0.86869	0.00836
5	TRUE	TRUE	LogReg	100	l1	TrunSVD	50	3	0.86843	0.00859
6	TRUE	TRUE	LinSVC	100	N/A	TrunSVD	50	3	0.86818	0.00958
7	TRUE	FALSE	LogReg	100	l1	TrunSVD	50	3	0.85080	0.00843
8	TRUE	FALSE	LogReg	100	l2	TrunSVD	50	3	0.84978	0.00890
9	TRUE	FALSE	LinSVC	100	N/A	TrunSVD	50	3	0.84750	0.00950
10	TRUE	FALSE	LogReg	100	l1	TrunSVD	50	5	0.84712	0.00701
11	TRUE	FALSE	LogReg	100	l2	TrunSVD	50	5	0.84699	0.00571
12	TRUE	FALSE	LinSVC	100	N/A	TrunSVD	50	5	0.84433	0.00521
13	TRUE	TRUE	LogReg	100	l1	NMF	50	3	0.83557	0.00748
14	TRUE	TRUE	LogReg	100	l1	NMF	50	5	0.83443	0.00665
15	TRUE	TRUE	LinSVC	100	N/A	NMF	50	5	0.83278	0.00794
16	TRUE	TRUE	LinSVC	100	N/A	NMF	50	3	0.83253	0.00906
17	TRUE	TRUE	LogReg	100	l2	NMF	50	3	0.83177	0.00820
18	TRUE	TRUE	LogReg	100	l2	NMF	50	5	0.83037	0.00876
19	FALSE	TRUE	LogReg	100	l2	TrunSVD	50	3	0.81604	0.00580
20	FALSE	TRUE	LogReg	100	l1	TrunSVD	50	3	0.81553	0.00454
21	FALSE	TRUE	LinSVC	100	N/A	TrunSVD	50	3	0.81515	0.00473
22	FALSE	TRUE	LogReg	100	l1	TrunSVD	50	5	0.81388	0.00292
23	FALSE	TRUE	LogReg	100	l2	TrunSVD	50	5	0.81388	0.00374
24	FALSE	FALSE	LogReg	100	l1	TrunSVD	50	3	0.81325	0.00780
25	FALSE	TRUE	LinSVC	100	N/A	TrunSVD	50	5	0.81312	0.00344
26	FALSE	FALSE	LogReg	100	l2	TrunSVD	50	3	0.81134	0.00770
27	FALSE	FALSE	LinSVC	100	N/A	TrunSVD	50	3	0.81071	0.00800
28	FALSE	FALSE	LogReg	100	l1	TrunSVD	50	5	0.81071	0.00555

29	FALSE	FALSE	LogReg	100	l2	TrunSVD	50	5	0.81045	0.00546
30	FALSE	FALSE	LinSVC	100	N/A	TrunSVD	50	5	0.80779	0.00714
31	TRUE	FALSE	LogReg	100	l1	NMF	50	5	0.79269	0.01177
32	TRUE	FALSE	LogReg	100	l1	NMF	50	3	0.79066	0.01909
33	TRUE	FALSE	LinSVC	100	N/A	NMF	50	5	0.78381	0.01356
34	FALSE	FALSE	LogReg	100	l1	NMF	50	3	0.78267	0.00835
35	FALSE	FALSE	LogReg	100	l1	NMF	50	5	0.78191	0.00729
36	FALSE	TRUE	LogReg	100	l1	NMF	50	3	0.78153	0.00843
37	FALSE	TRUE	LogReg	100	l1	NMF	50	5	0.77950	0.00369
38	TRUE	FALSE	LinSVC	100	N/A	NMF	50	3	0.77899	0.01907
39	FALSE	FALSE	LinSVC	100	N/A	NMF	50	3	0.77671	0.00681
40	FALSE	TRUE	LinSVC	100	N/A	NMF	50	3	0.77569	0.00810
41	TRUE	FALSE	LogReg	100	l2	NMF	50	5	0.77493	0.01303
42	FALSE	FALSE	LinSVC	100	N/A	NMF	50	5	0.77430	0.00697
43	FALSE	TRUE	LinSVC	100	N/A	NMF	50	5	0.77277	0.00416
44	FALSE	FALSE	LogReg	100	l2	NMF	50	3	0.77074	0.00687
45	FALSE	TRUE	LogReg	100	l2	NMF	50	3	0.76947	0.01017
46	FALSE	FALSE	LogReg	100	l2	NMF	50	5	0.76935	0.00833
47	TRUE	FALSE	LogReg	100	l2	NMF	50	3	0.76706	0.01929
48	FALSE	TRUE	LogReg	100	l2	NMF	50	5	0.76681	0.00349
49	TRUE	TRUE	GausNB	N/A	N/A	TrunSVD	50	5	0.73471	0.01182
50	TRUE	TRUE	GausNB	N/A	N/A	TrunSVD	50	3	0.72938	0.01278
51	TRUE	TRUE	GausNB	N/A	N/A	NMF	50	3	0.72723	0.00875
52	TRUE	TRUE	GausNB	N/A	N/A	NMF	50	5	0.72609	0.00720
53	FALSE	FALSE	GausNB	N/A	N/A	TrunSVD	50	3	0.68802	0.01476
54	FALSE	FALSE	GausNB	N/A	N/A	TrunSVD	50	5	0.68663	0.00888
55	FALSE	TRUE	GausNB	N/A	N/A	TrunSVD	50	5	0.68625	0.00733
56	FALSE	TRUE	GausNB	N/A	N/A	TrunSVD	50	3	0.68447	0.00763
57	TRUE	FALSE	GausNB	N/A	N/A	TrunSVD	50	5	0.67419	0.01291
58	FALSE	TRUE	GausNB	N/A	N/A	NMF	50	5	0.66696	0.01080
59	FALSE	FALSE	GausNB	N/A	N/A	NMF	50	5	0.66696	0.00857
60	FALSE	TRUE	GausNB	N/A	N/A	NMF	50	3	0.66658	0.01605
61	FALSE	FALSE	GausNB	N/A	N/A	NMF	50	3	0.66049	0.01587
62	TRUE	FALSE	GausNB	N/A	N/A	TrunSVD	50	3	0.66036	0.01877

63	TRUE	FALSE	GausNB	N/A	N/A	NMF	50	5	0.65745	0.01607
64	TRUE	FALSE	GausNB	N/A	N/A	NMF	50	3	0.64133	0.02340