Project 1: End-to-End Pipeline to Classify News Articles

Prof. Vwani Roychowdhury

UCLA, Department of ECE

Team Members:
Rohan Mehta, UID: 205871841

Getting familiar with the dataset

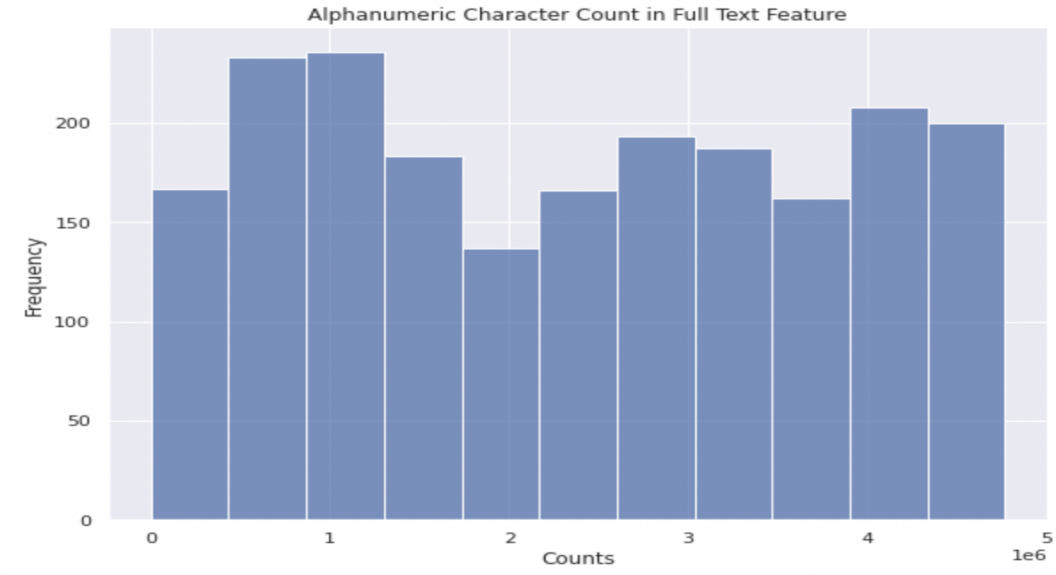
Question 1:

- ***Overview***

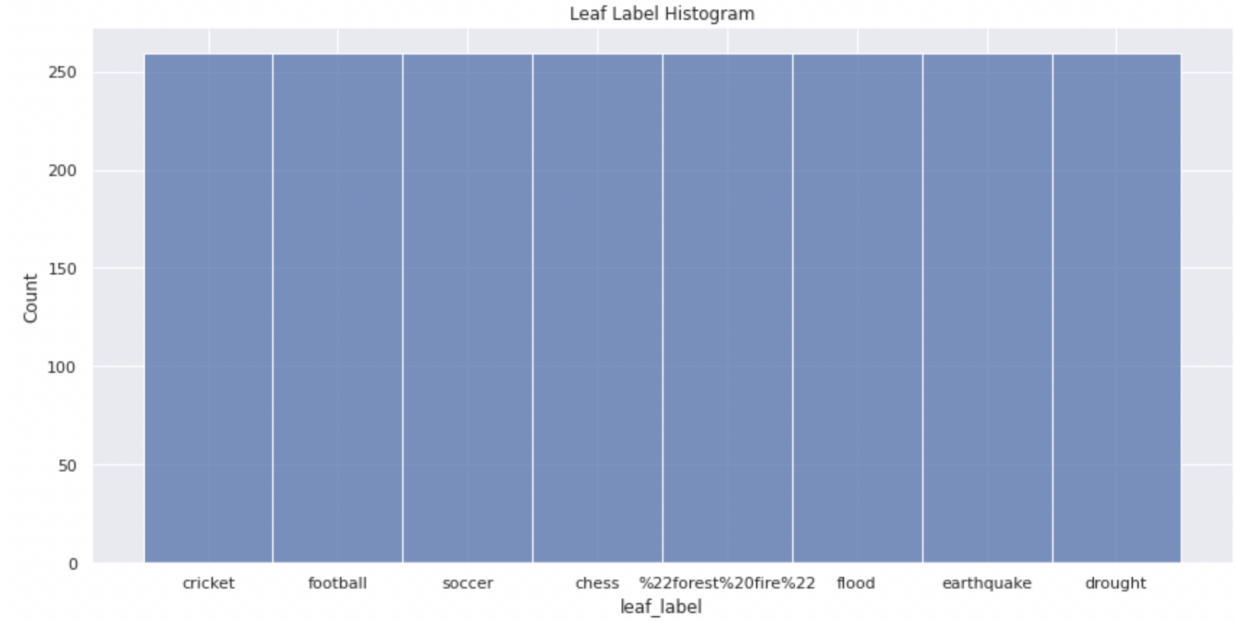
Table1: News Articles Dataset	
Samples	2072
Features	9

- ***Histograms***

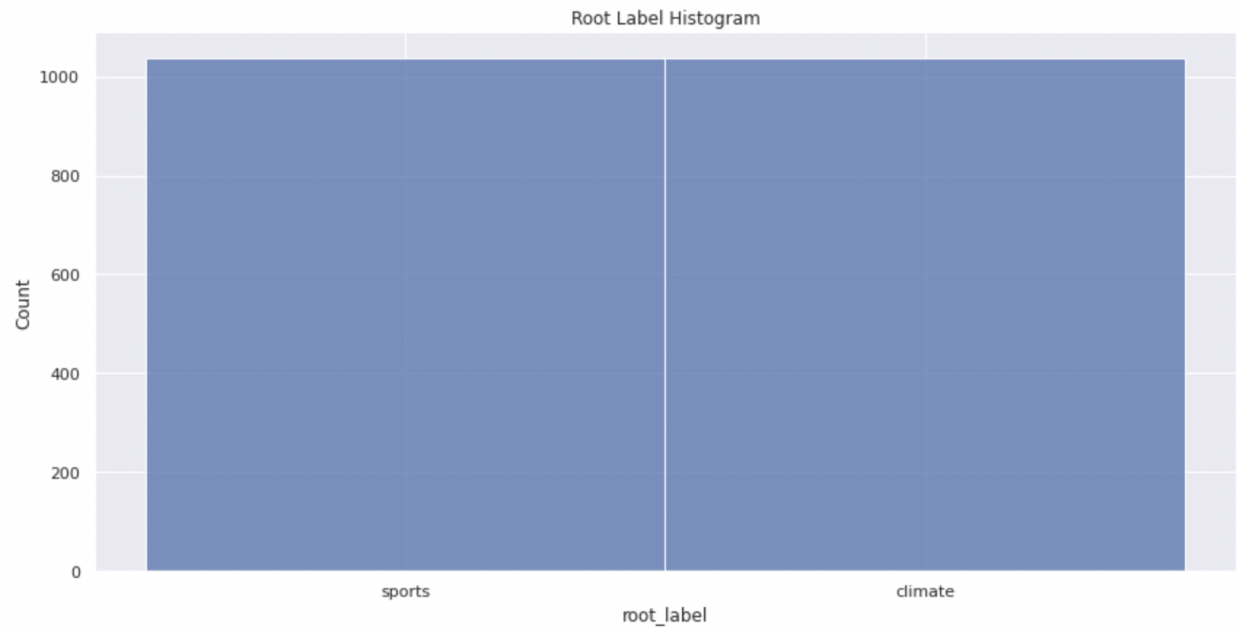
a. **Figure 1**



b. Figure 2



c. Figure 3



- ***Interpret Plots***
    a. There is non-uniformity amongst the number of alphanumeric characters present in the “full text” feature provided in the dataset. This means that when we proceed to embed the full text into feature vectors, certain samples will be providing less data than others to the vectorizer.
    b. The “leaf_label” feature in the dataset appears to have a very balanced spread of classes; if we were to utilize this feature as the output prediction for a classifier then we would not have the problem of having under or over representation of certain classes
    c. The “root_label” feature as a direct consequence of the above is also perfectly balanced; using this feature as the output of a classifier will also not require class-balancing.

Binary Classification

1. Splitting the entire dataset into training and testing data

Question 2

Table 2: # of Examples after Training/Testing Split (Test Size = 0.2)	
# Training Examples	**1657**
# Testing Examples	**415**

2. Feature Extraction

Question 3

- **What are the pros and cons of lemmatization versus stemming? How do these processes affect the dictionary size?**

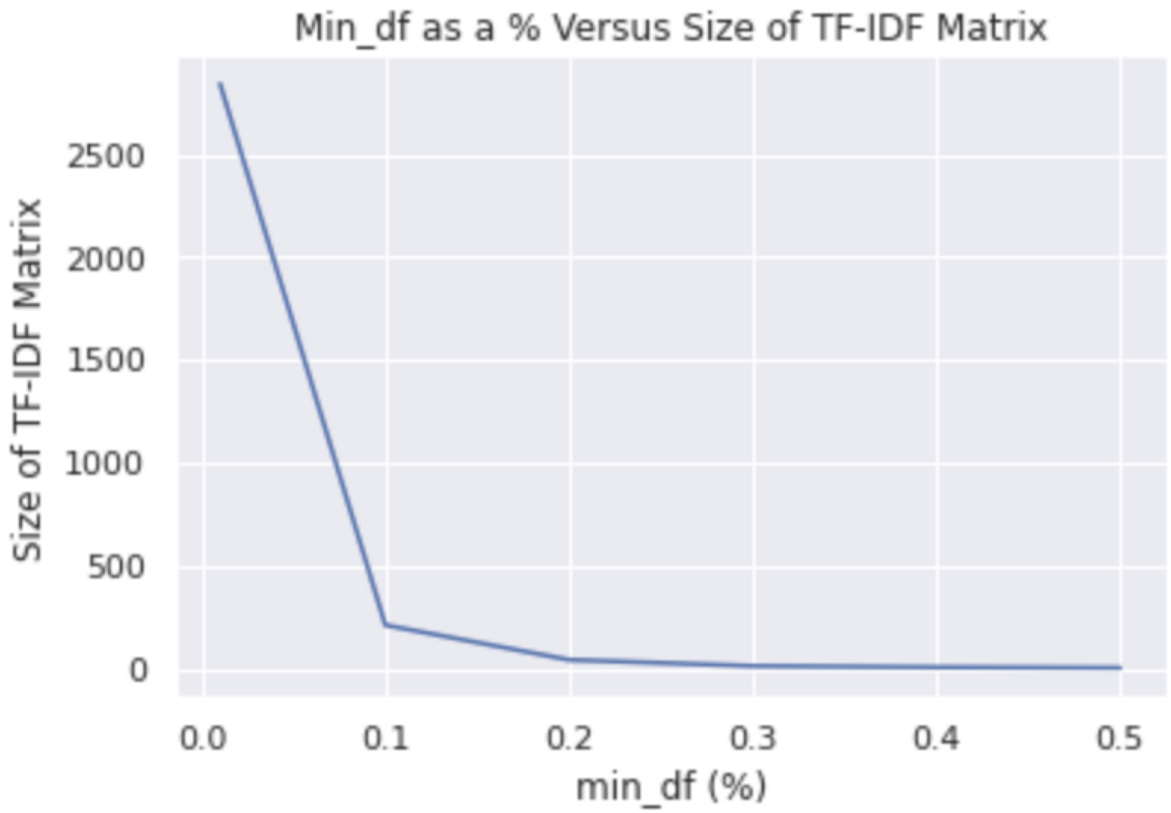
Lemmatization and stemming are both word normalization techniques, however they differ in terms of the output of each algorithm. Lemmatization requires more processing due to the part of speech tagging requirement, but will always produce a valid word; stemming, on the other hand, is a general set of rules which can be applied to any word for its output, but will not necessarily produce a valid word.

Lemmatization will produce better results for processing, but the dictionary size is larger with stemming due to the output not needing to be a valid dictionary word.

- **min_df means *minimum document frequency*. How does varying min_df change the TF-IDF Matrix?**

The min_df term is the lower-bound on how many occurrences of a term in a document is considered significant. As one increases this threshold, the resulting size of the TF-IDF Matrix decreases and only rarer words are considered in the output.

Figure 4



- **Should I remove stopwords before or after lemmatizing? Should I remove punctuations before or after lemmatizing? Should I remove numbers before or after lemmatizing?**

Stopwords may provide context which affects the part of speech tagging for lemmatization, thus stopword removal should be done after lemmatization.

After the sentences in a document have been tagged for parts of speech, punctuation should be removed prior to lemmatizing so that the output of lemmatization only includes valid words.

Numbers usually do not add context if they are not significant to a specific topic being analyzed. Removing numbers prior to lemmatization will provide a more relevant output.

- ***Shape of TF-IDF Processed Matrices***

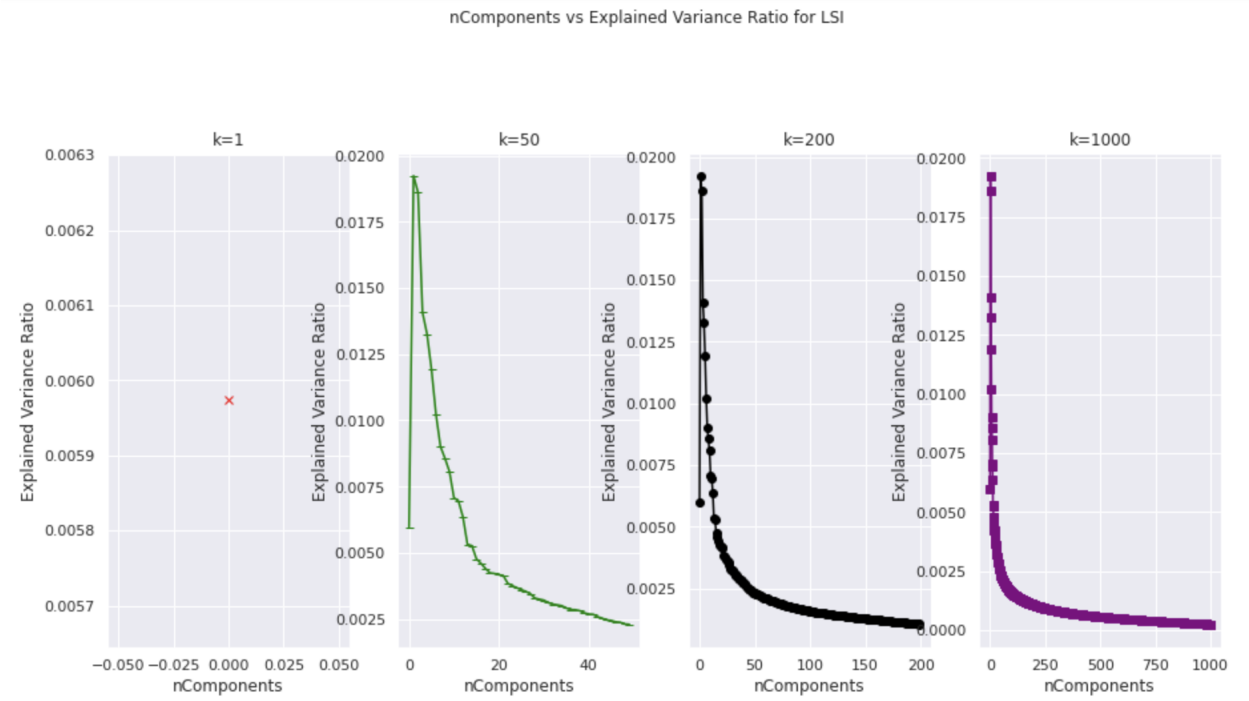
Table 3: TF-IDF Train and Test Matrices Shapes	
Train Matrix	(1657, 10441)
Test Matrix	(415, 10441)

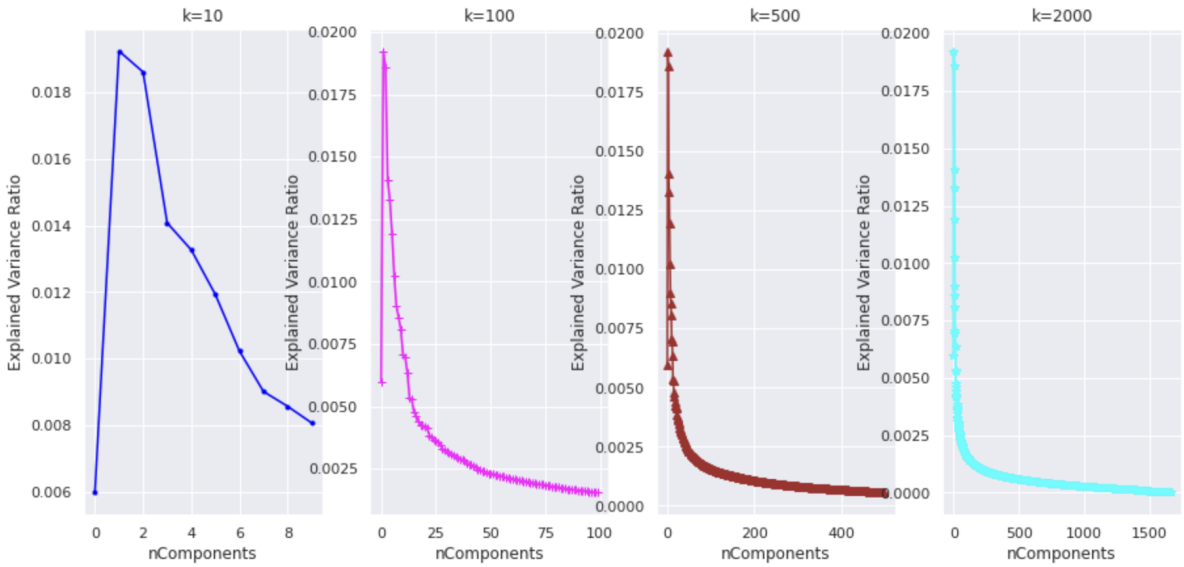
3. Dimensionality Reduction

Question 4

- *Explained Variance Ratio of LSI*

Figure 5





The plot's concavity suggests that the most important eigenvectors, which account for most of the information in the independent features, occur first in the LSI output. As the index of the eigenvector increases, the explained variance ratio decreases. Cumulatively, the more components we have the more information is captured of the original feature matrix.

- *Residual MSE Error of LSI and NMF*

Table 4: Frobenius Norm for LSI and NMF with k = 50	
Frobenius Norm LSI	34.390447642814436
Frobenius Norm NMF	34.72713958285293

Based on the results above in Table 4, it can be concluded that NMF has a larger residual MSE error after dimensionality reduction. The reason this could be the case is due to the convergence condition of NMF; the NMF algorithm will run for the maximum iterations specified even if the error is not below a threshold. In order to reach the ideal threshold within a tolerance level the computational cost would be more than what users are willing to pay.

4. Classification Algorithms

a. SVM

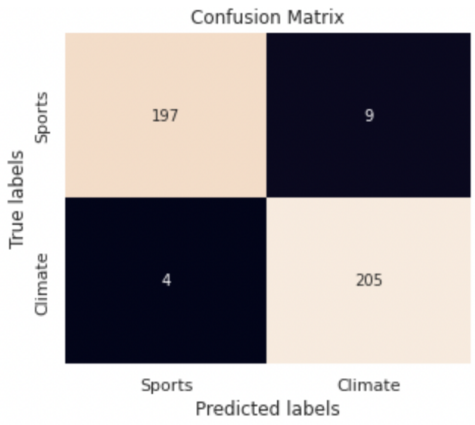
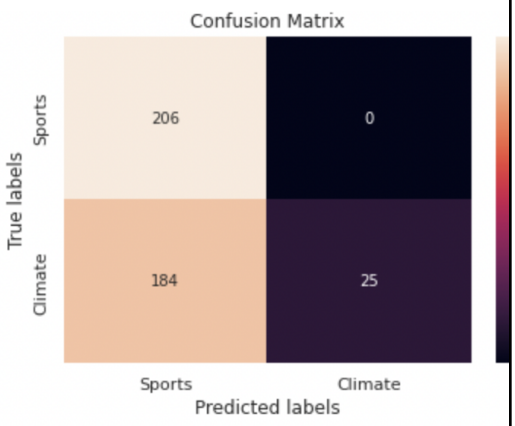
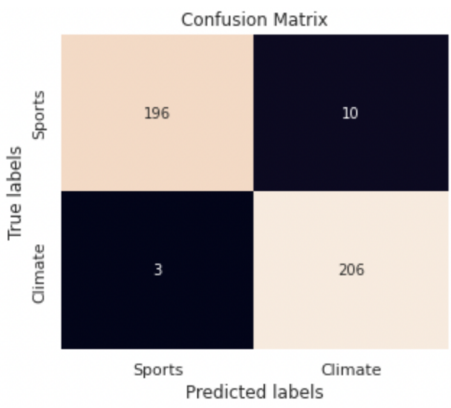
Hard Margin SVM	Soft Margin SVM	Hardest Margin SVM
		

Table 5: Comparing SVM with Soft/Hard/Hardest Margins ("C" Value)

	Accuracy	Precision	Recall	F1-Score
Soft Margin	55.663%	100%	11.962%	21.368%
Hard Margin	96.867%	95.794%	98.086%	96.927%
Hardest Margin	96.867%	95.370%	98.565%	96.941%

The soft margin SVM classifier performs very poorly as can be seen in Table 5. The confusion matrix reveals that the "Climate" category was predicted incorrectly as "Sports" for many examples in the test dataset. The margin is a hyperparameter which balances minimizing classification mistakes versus maximizing distance between the points and decision boundary between classes. When this margin (C-value) is very small, more classification mistakes are allowable under the formulation of the SVM algorithm. This explains the poor performance of the soft margin SVM classifier.

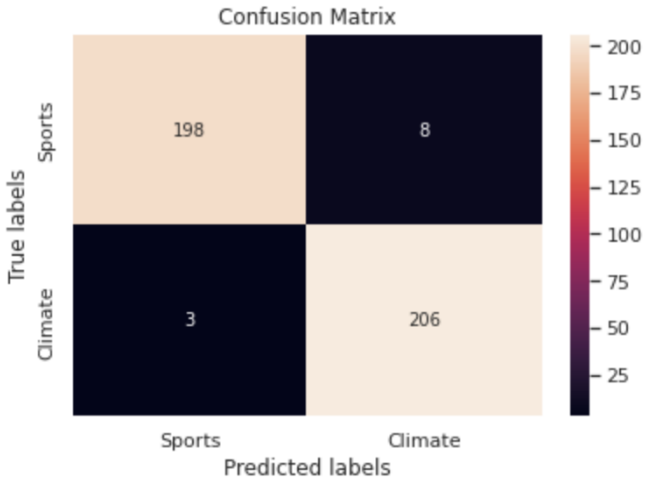
- *Cross Validation Procedure for Optimal Margin*

	param_C	mean_test_score
0	0.001	0.500906
1	0.01	0.701287
2	0.1	0.946888
3	1	0.949299
4	10	0.956539
5	100	0.954122
6	1000	0.949893
7	10000	0.949893
8	100000	0.949893
9	1000000	0.949893

When comparing the validation accuracy of each margin value, the optimal is C=10.

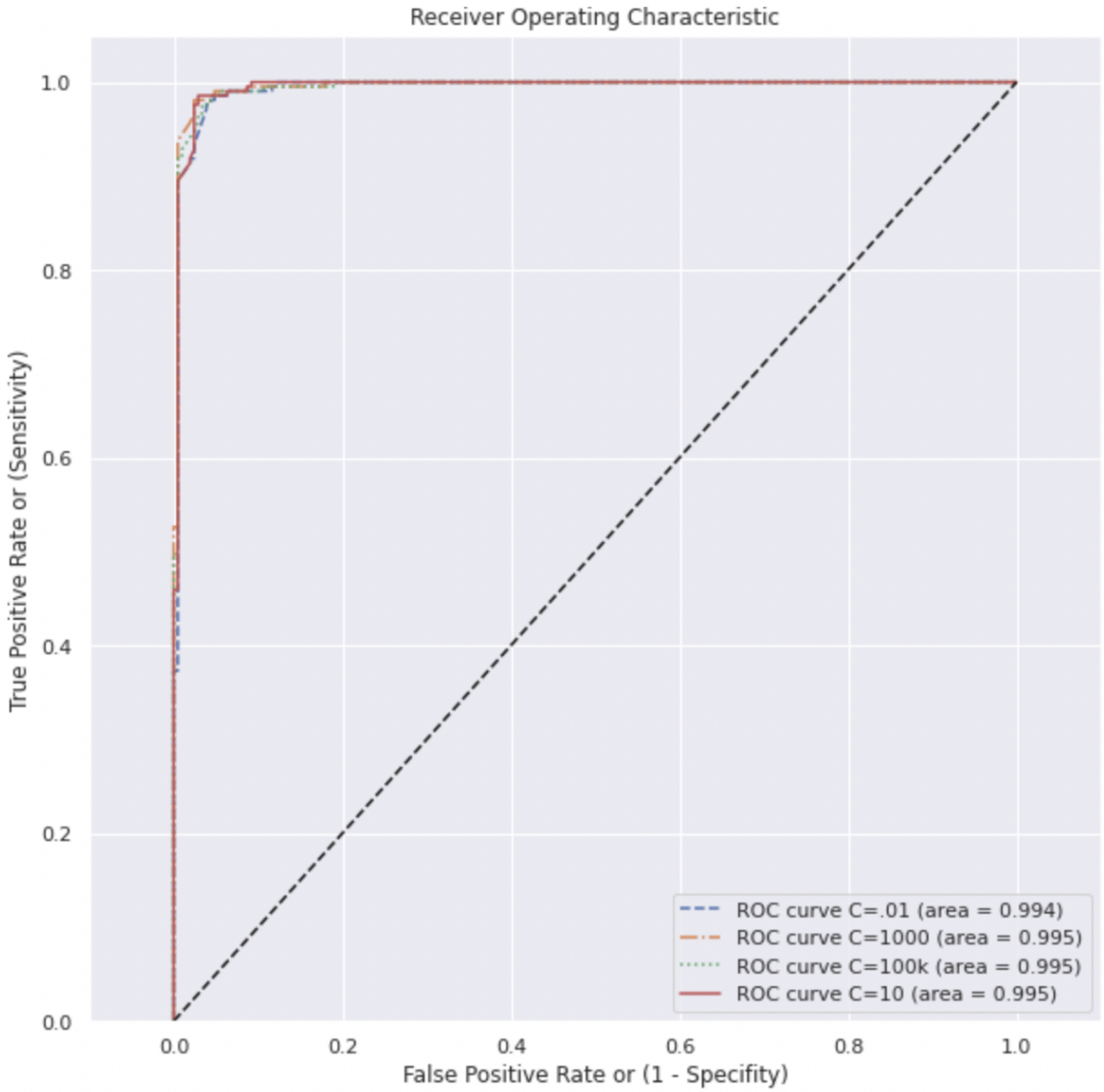
Table 6: SVM Optimal Margin Classifier (C = 10)			
Accuracy	Precision	Recall	F1-Score
97.349%	96.262%	98.565%	97.400%

Figure 9



- *ROC Curve Analysis*

Figure 10



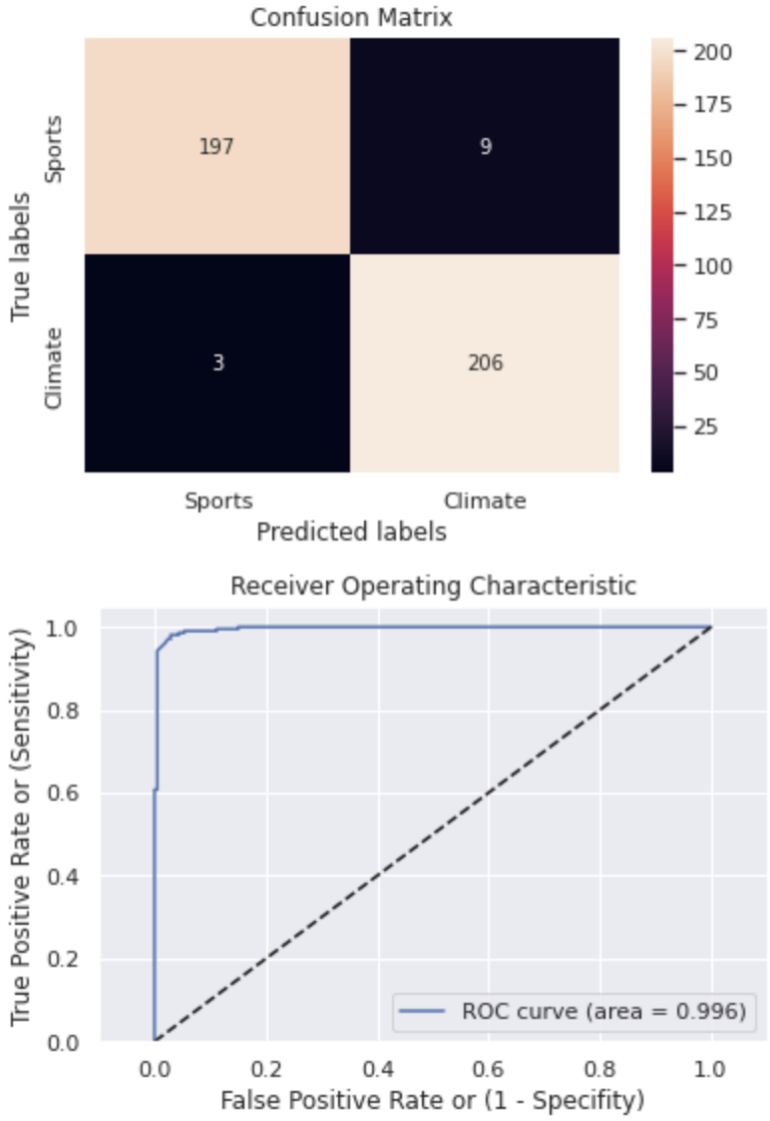
The soft margin (C=.01) ROC curve does not look competitive based on Figure 10 above. The blue dotted line represents the soft-margin classifier and of all the SVM classifiers trained it is the least ideal (furthest from area of 1).

b. Logistic Regression

- *No Regularization*

Table 7: Logistic Regression w/ No Regularization			
Accuracy	Precision	Recall	F1-Score
97.108%	95.814%	98.565%	97.170%

Figure 11



- *Finding Optimal Regularization Strengths for L1 & L2*

L1 Regularization Cross Validation

	param_C	mean_test_score
0	0.0001	0.500906
1	0.001	0.500906
2	0.01	0.500906
3	0.1	0.904639
4	1	0.955338
5	10	0.957151
6	100	0.954732
7	1000	0.952925
8	10000	0.952925

L2 Regularization Cross Validation

	param_C	mean_test_score
0	0.0001	0.500906
1	0.001	0.900451
2	0.01	0.941455
3	0.1	0.945676
4	1	0.952919
5	10	0.955340
6	100	0.954131
7	1000	0.955940
8	10000	0.954131

The optimal regularization strength based on the mean test score for the L1 regularization is C=10. The optimal regularization strength based on the mean test score for L2 regularization is C=1000.

- *Comparing Performance of Logistic Classifiers w/ Optimal Regularization Strength*

	Table 8: Logistic Classifiers			
Reg. Parameter	Accuracy	Precision	Recall	F1-Score
No Penalty	97.108%	95.814%	98.565%	97.170%
L1 Reg.	97.349%	96.698%	98.086%	97.387%
L2 Reg.	97.108%	96.244%	98.086%	97.156%

- **How does the regularization parameter affect the test error? How are the learnt coeffi- cients affected? Why might one be interested in each type of regularization?**

The test error is a measure of how well the model generalized to data it has never seen. The regularization parameter shrinks the weights in the model which in effect reduces the variance of the model; less variance in the model means it will generalize better to the test set.

The learned coefficients, as mentioned previously, shrink as a result of the regularization term introduced into the classifier. Thus, the regularization methods are also referred to as “shrinkage” methods.

The L1 regression term operates on the sum of the absolute values of the coefficients and differs slightly in its mathematical formulation from L2 regression. Under L1 regression, coefficients can be zero’d entirely from the model which can come in useful when one knows apriori to remove a certain dependency from the model. L2 regression, on the other hand, penalizes large coefficients. By shifting coefficients closer to zero, L2 regression reduces model complexity.

- **Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary. What is the difference between their ways to find this boundary? Why do their performances differ? Is this difference statistically significant?**

Logistic regression ingests an input feature vector and outputs the probability of belonging to a particular class. The model trains by attempting to minimize the error from the actual and predicted class utilizing the probability for each point. Thus, the linear boundary in a binary classification problem is the line at which the probability for a class would be 0.5; anything below or above that line will belong to two of the candidate classes.

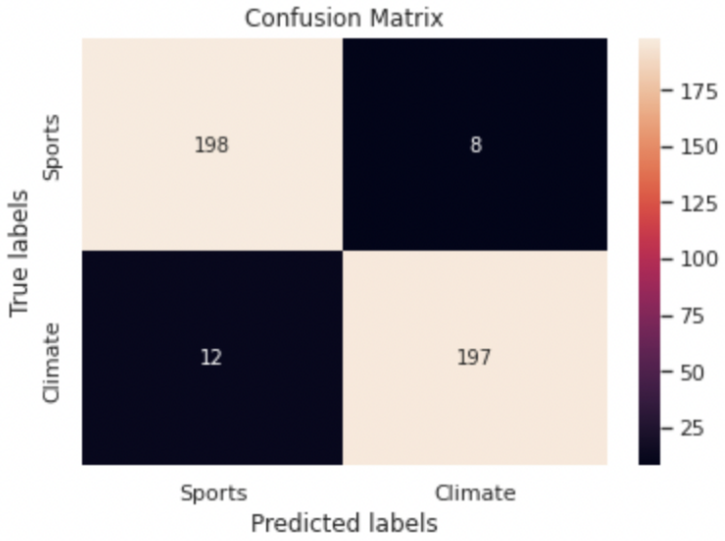
SVM begins by creating a hyperplane between the data points and adjusts the plane in an attempt to maximize the distance between the margin and “support vectors” (input feature vectors). SVM utilizes the geometry of the data points in space to determine the linear boundary as opposed to logistic regression, which relies on statistics.

The optimal SVM and Logistic Regression classifier both performed with the same accuracy of 97.349% but SVM had a slightly higher F1-Score of 97.400% as compared to Logistic Regression with 97.387%. This difference does not appear to

be statistically significant. In general though, SVM does not care about the statistical distribution of the data in order to optimize its linear boundary. For an unstructured dataset with text such as the news articles intuitively SVM would perform better.

c. Naive Bayesian Model

Figure 12



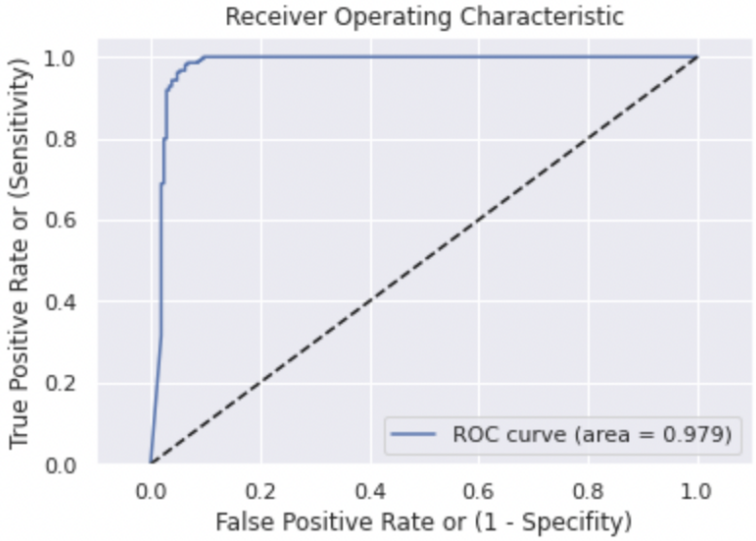
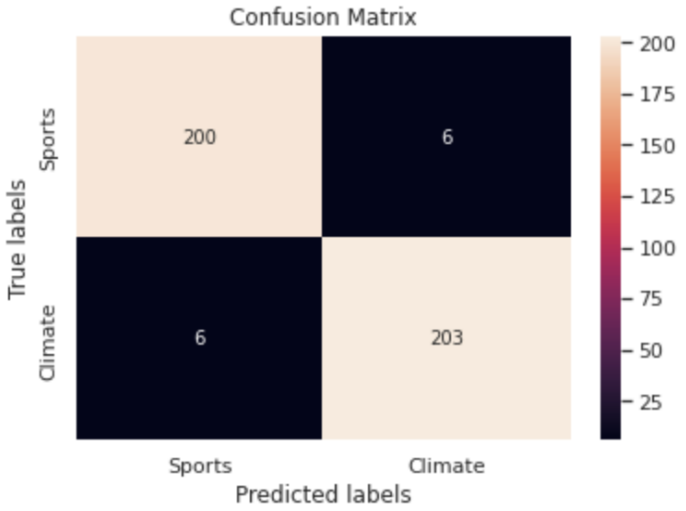


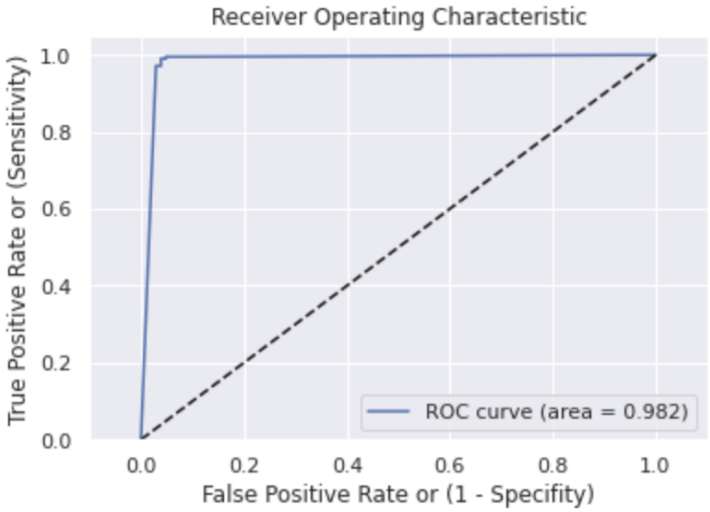
Table 9: GaussianNB Classifier			
Accuracy	Precision	Recall	F1-Score
95.181%	96.098%	94.258%	95.169%

d. Grid Search of Parameters

Grid Search Modules and Corresponding High Performing Options	
Module	**Best Option**
Loading Data	Cleaning
Feature Extraction	Min_df 5, Lemmatization
Dimensionality Reduction	NMF, k = 200
Classifier	GaussianNB()
Other Options	Default

Table 9: GaussianNB Classifier			
Accuracy	Precision	Recall	F1-Score
97.108%	97.129%	97.129%	97.129%

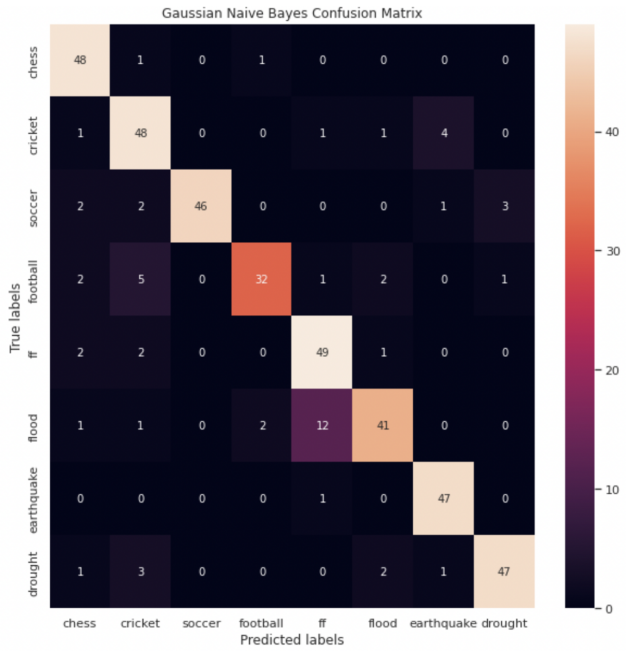
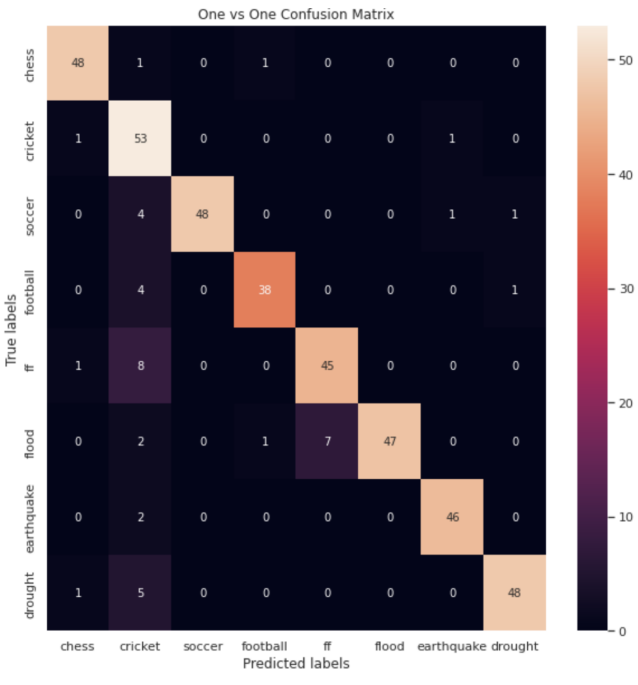

Confusion Matrix
200
6
6
203
Sports
Climate
True labels
Sports
Climate
Predicted labels
200
175
150
125
100
75
50
25


Receiver Operating Characteristic
True Positive Rate or (Sensitivity)
False Positive Rate or (1 - Specifity)
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.982)

5. MultiClass Classification

- *Confusion Matrices of MultiClass Classifiers*

For the following confusion matrices, “ff” signifies "%22forest%20fire%22"

Gaussian Naive Bayes	
SVM One vs. One	

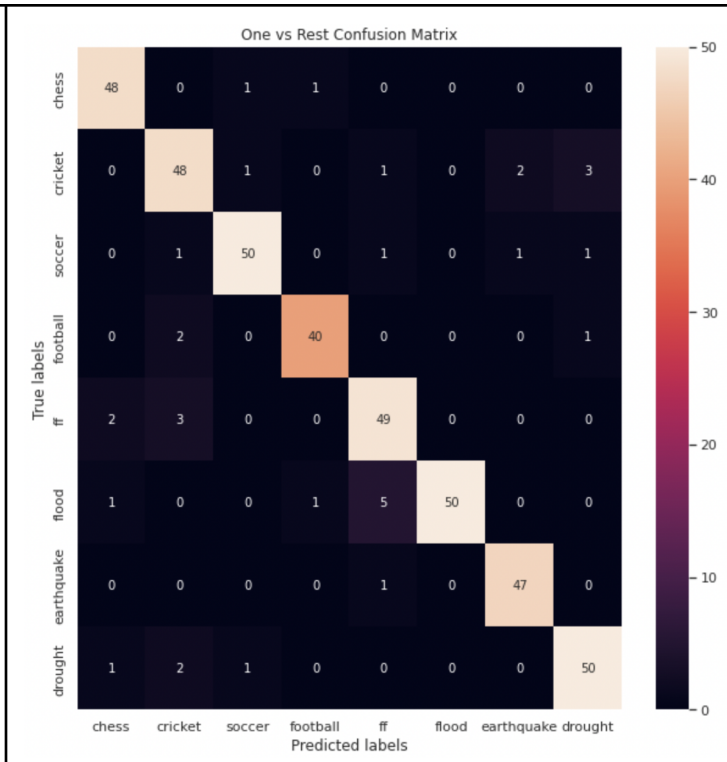
SVM One vs. Rest	

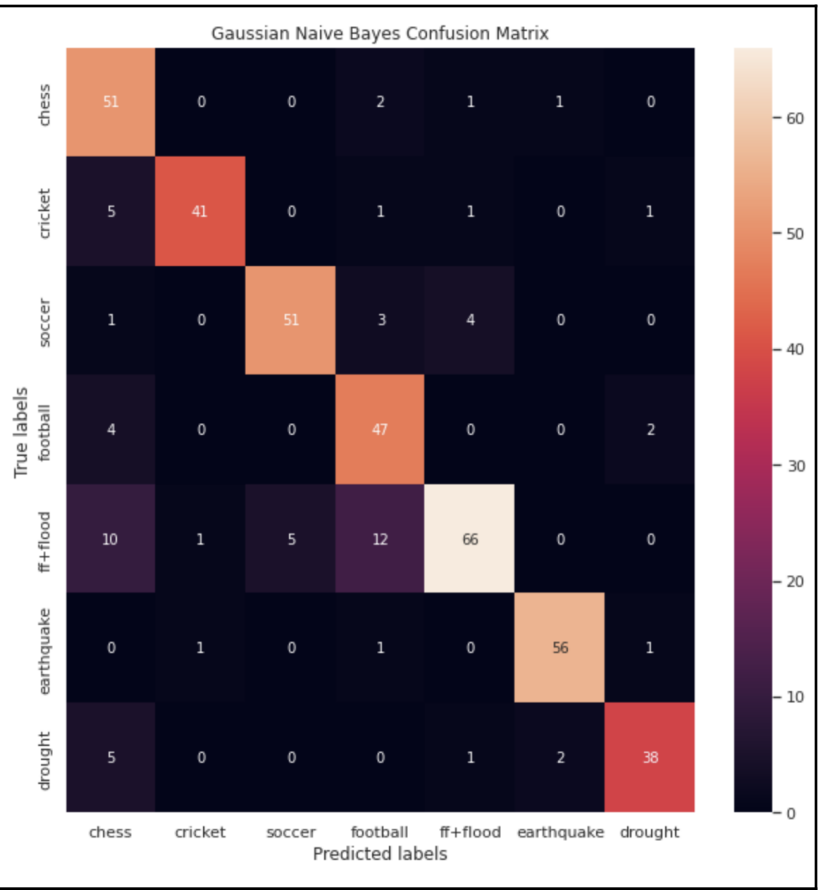
Table 10: Comparing MultiClass Classifiers

	Accuracy	Precision	Recall	F1-Score
Gaussian NB	86.265%	87.211%	86.313%	86.289%
One vs. One	89.880%	91.822%	90.017%	90.409%
One vs. Rest	92.048%	92.309%	92.232%	92.198%

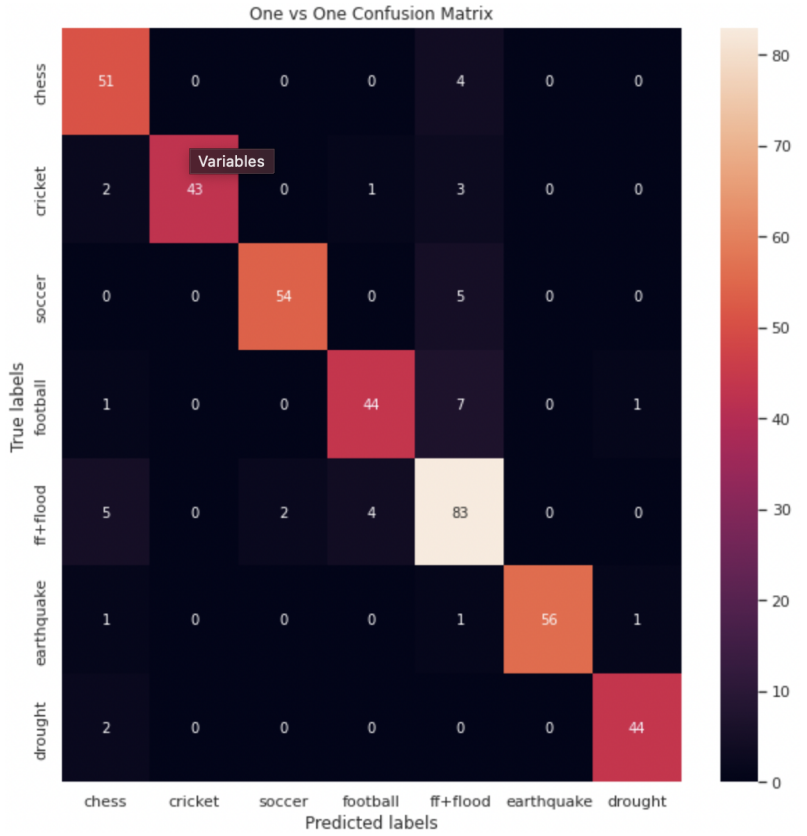
In the SVM One vs. Rest Classifier, we begin with an equal distribution of classes in the “leaf label” column of our dataset. When we begin training on individual classes and aggregate the rest, the training becomes imbalanced. This however is not an issue for SVM due to the fact that SVM implements the optimal slack penalty (“C” value) ,and the fact that we are training every One vs. Rest classifier in a similar fashion thus the error is averaged out to some degree. The accuracy for the One vs. Rest classifier trained above is the largest out of each of the models trained above.

In any of the confusion matrices listed above, a major block along the diagonal indicates correctly classified predicted labels when compared to the truth provided to the classifier. In the Naive Bayes confusion matrix in particular, the classifier seemed to miss classify many instances of a forest fire as a flood. The One vs. One and One vs. Rest classifier seemed to exhibit this pattern to a lesser degree. Both labels are a good candidate for merging and re-training the classifier and belong to the same hierarchy of “climate”.

- *Merging Classes and Retraining*

Gaussian Naive Bayes	

SVM One vs. One



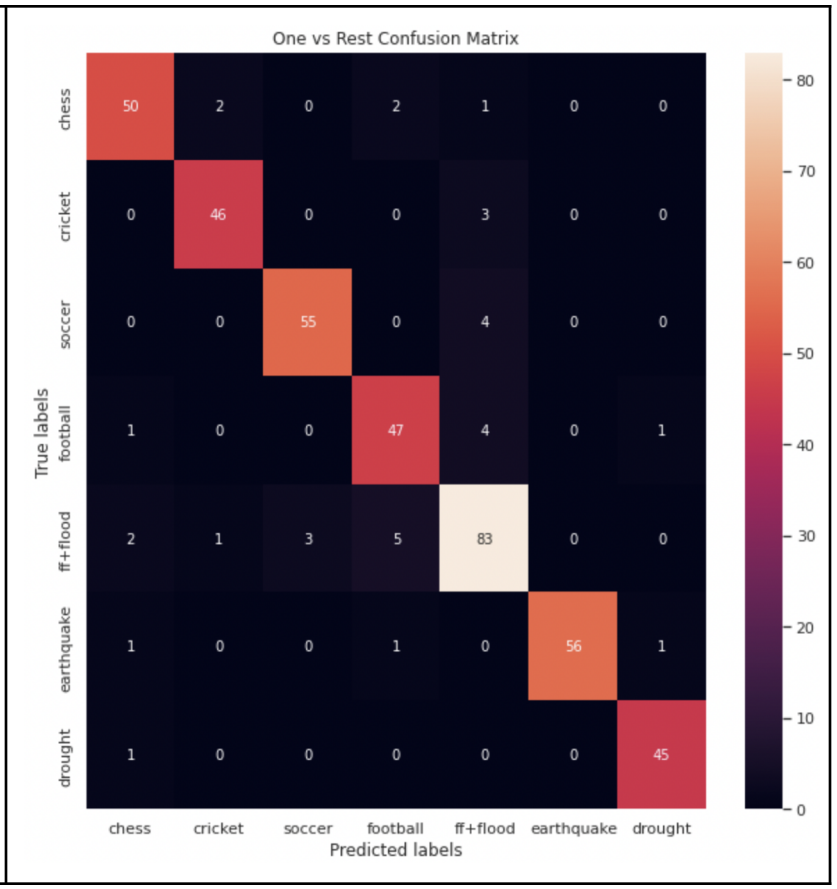
SVM One vs. Rest	

Table 11: Accuracies of Classifiers After Merging Labels and Retraining	
Naive Bayes	84.337%
SVM One vs. One	90.361%
SVM One vs. Rest	92.482%

Comparing Table 10 and 11, we can see that SVM One vs. One and SVM One vs. Rest performed better in terms of accuracy. The relabeling of the training data included merging the “flood” and “%22forest%20fire%22” class after analyzing the confusion matrix from the previous section.

There is a class imbalance issue once two labels are merged. The class with the merged labels is over-represented in the training dataset and this may create a

learned bias within the classifier. This can be seen by a simple value_counts() below:

```
ff+flood      94
drought       59
football      59
chess         55
earthquake    53
cricket       49
soccer        46
dtype: int64
```

To remedy this issue, I employed an undersampling method of the majority class such that all classes have equal representation after this algorithm is run. This method randomly removes samples from the majority class until the dataset achieves an equal amount of samples as the minority class, in this case football or drought (seen in Table 12). This method can create issues if the minority class is severely underrepresented, however in this case that is not the issue.

Table 12: Number of Samples per Class Before UnderSampling

ff+flood	soccer	cricket	earthquake	chess	football	drought
424	213	210	206	204	200	200

- *Results After Undersampling Method*

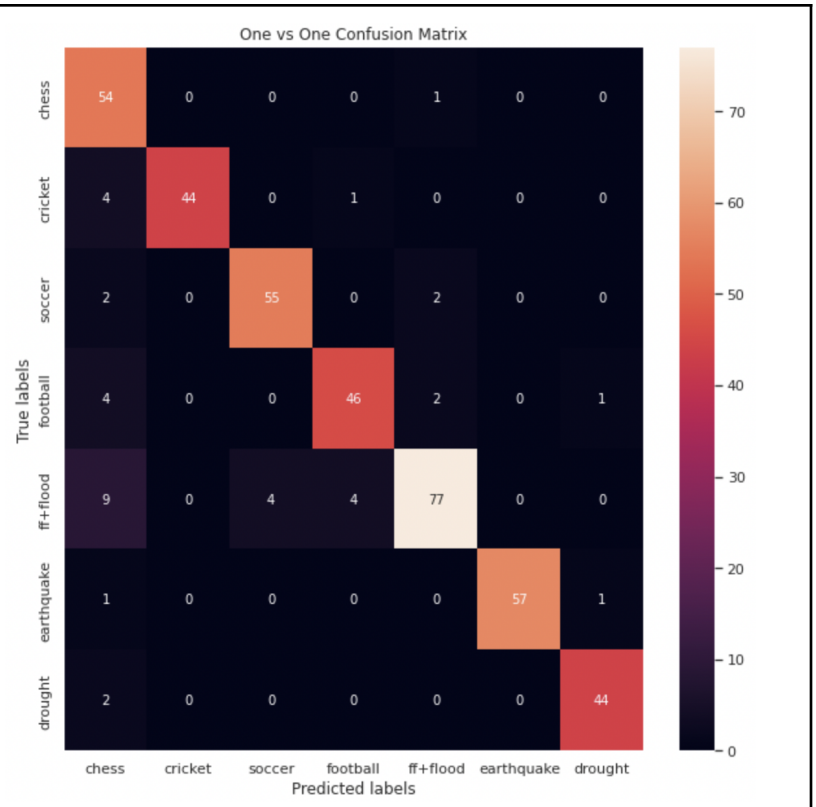
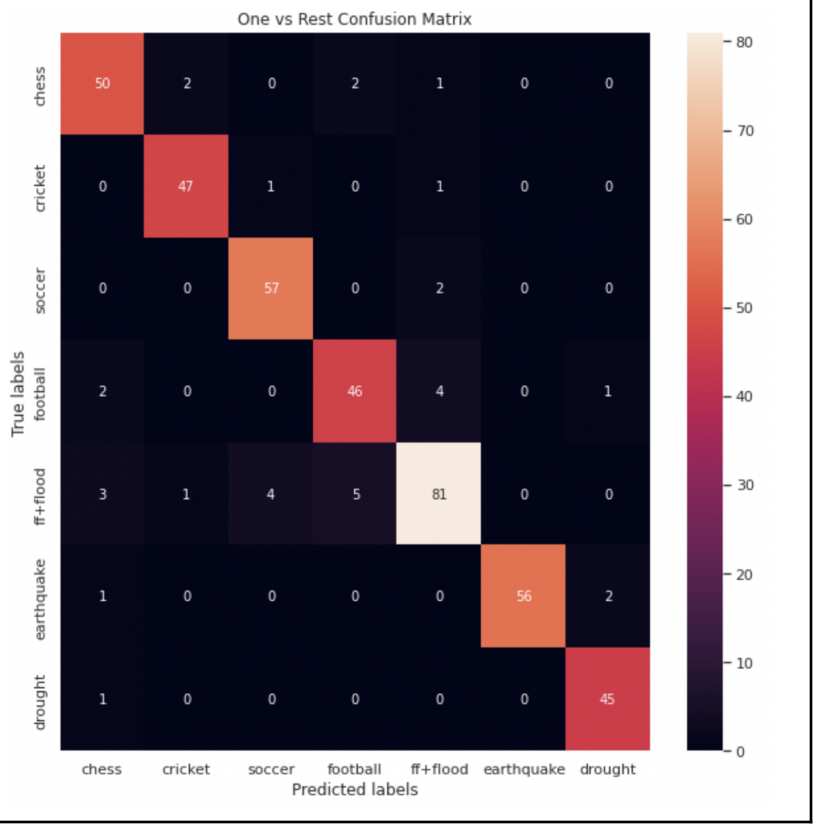
SVM One vs. One	
SVM One vs. Rest	

Table 13: Reported Accuracies After Undersampling	
One vs. One	90.843%
One vs. Rest	92.048%

After compensating for undersampling, it appears that the performance change is negligible for the One vs. One and One vs. Rest Classifiers when comparing Table 11 and Table 13.

6. Word Embeddings

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

If the ratio of the co-occurrence probabilities are used as opposed to the individual co-occurence probabilities, then the comparison of two test words to a third word can be performed. When performed over the entire dataset, the ratio of co-occurence probabilities will be highest for the words which showed up most frequently with the word under test.

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

If a pre-trained GLoVE embedding is provided such as the ones provided by Stanford University, then the term “running” will return the same word vector because context does not matter. The algorithm will produce the vector based on the words most frequently paired with “running” from its pre-trained corpus.

c. What do you expect for the values of, ||GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]||2, ||GLoVE["queen"] - GLoVE["king"]||2 and ||GLoVE["wife"] - GLoVE["husband"]||2 ? Compare these values.

The first part of this question can be interpreted as whether I expect the vector difference between the vector representation of “queen” and “king” to be the same as that of “wife” and “husband”. The words “queen” and “king” , although antonyms, would often show up in the same passages of text in a corpus due to their similarity in topic. However, there would be a small set of contextual words which are associated with one but not the other due to the gender differences between both words as well. The same logic can be applied to “wife” and “husband”. I would expect the value of the difference between the vector representations of “queen” and “king” as well as “wife” and “husband” to be a small non-zero value.

Table 14: Comparing L2 Norm of GLoVE Embedding Calculations	
||GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]||2	6.1650367
||GLoVE["queen"] - GLoVE["king"]||2	5.966258
||GLoVE["wife"] - GLoVE["husband"]||2	3.1520464

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

The GLoVE embedding is usually trained on a corpus only containing valid words. Thus, the sound choice is the method which produces valid words which is lemmatization. If stems are provided, GLoVE may have issues returning a vector representation.

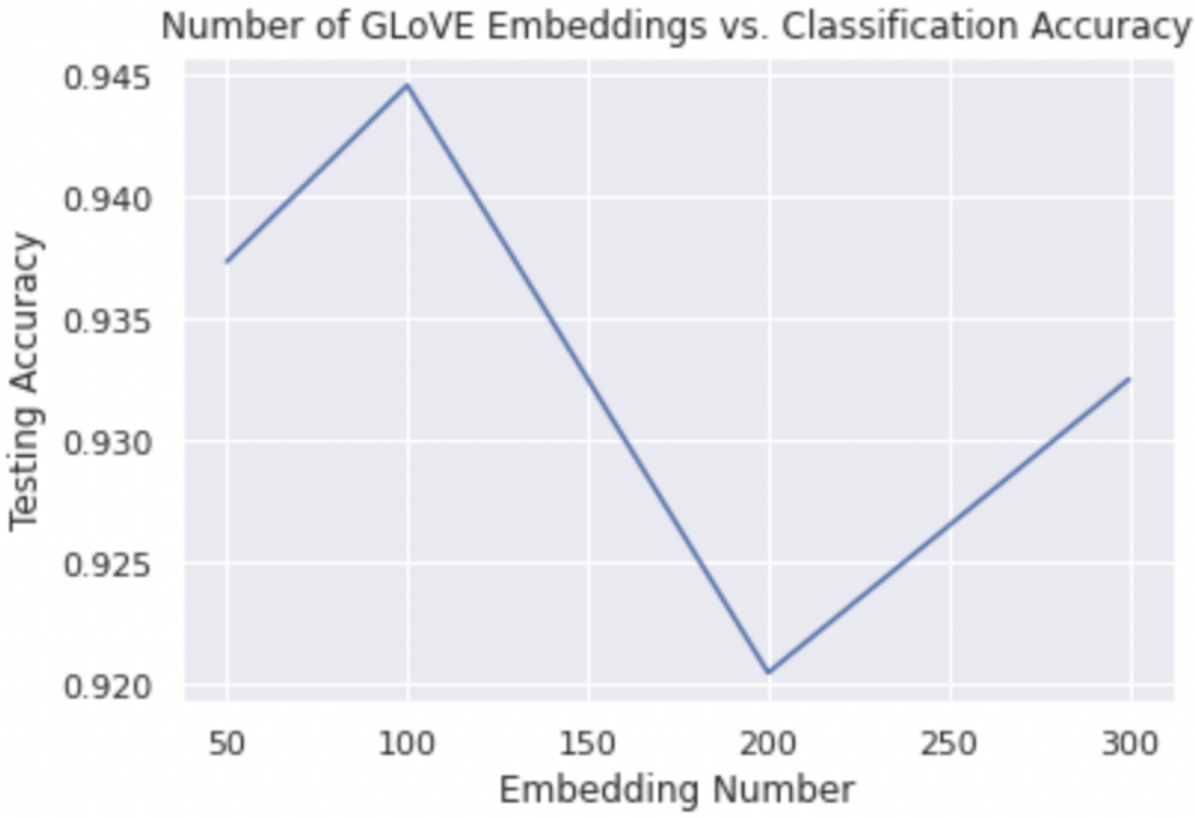
- *Feature Engineering Process for GLoVE Embeddings*

1. Due to ease of use for retrieving GLoVE Embeddings, I chose to use the “keywords” column of the dataset to perform training/testing for my binary classifier.

2. Each “keywords” row was first checked for validity as per the english dictionary and then lemmatized
3. I extracted GLoVE embeddings of size 300 for each keyword; if a “keywords” example contained four keywords, then I would have a matrix of (4, 300) for a single document in the training set.
4. In order to have a single (1, 300) word vector for each example in the dataset, I averaged the vectors for every keyword for a single document and then performed normalization on the resulting vector.

After following the steps above, I ended up with a GLoVE embedding vector of size 300 for each document in the dataset.

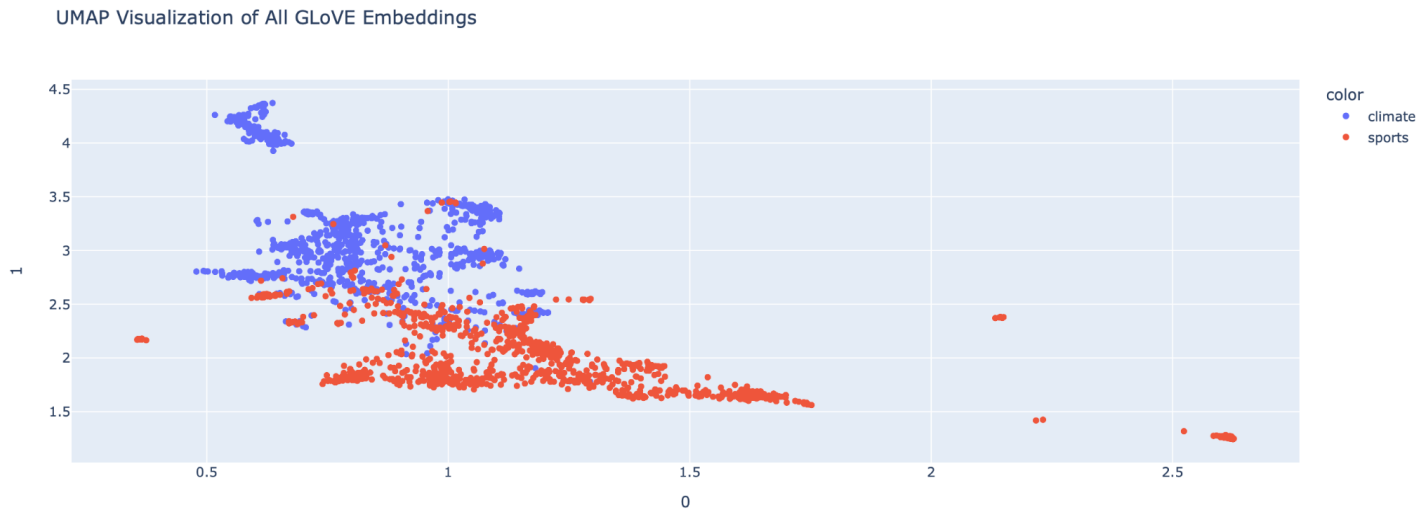
- *Plotting Different Embedding Sizes vs. Accuracy*

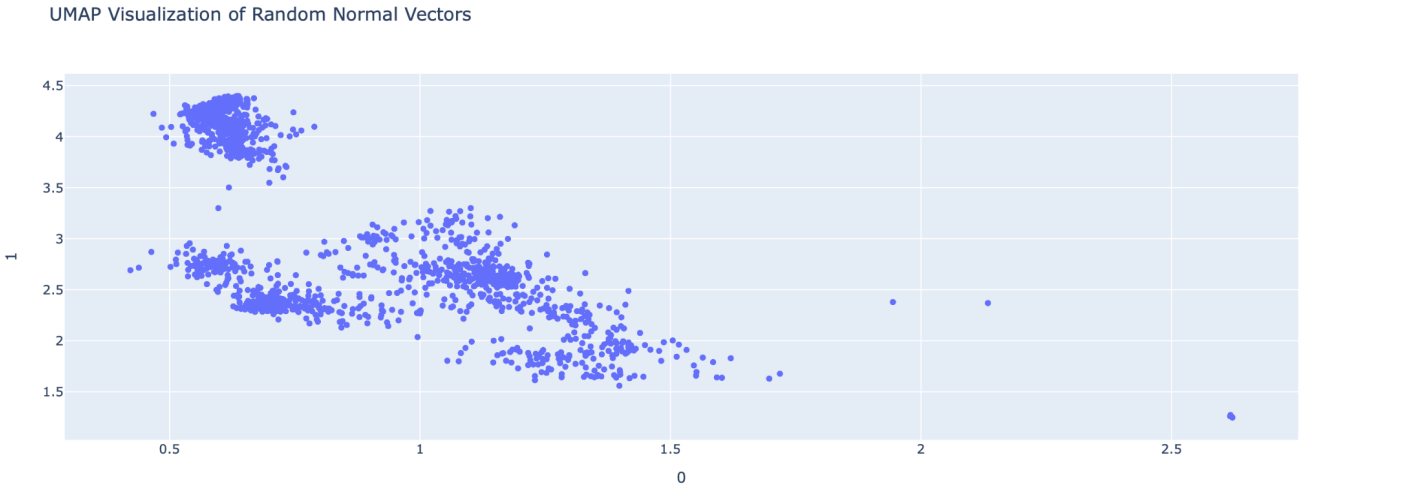


From the plot above it seems that there is an optimal number for the GLoVE embedding count of 100 for this particular dataset. The accuracy ranges between approximately 92% and 94.5% which is significant. Perhaps an embedding count of 100 provides the best separability between classes due to its reduction in noise; a

higher embedding count could potentially have relevant words which apply to all keywords in the dataset which creates confusion for the classifier.

UMAP Visualization





The UMAP representation based on the GLoVE embeddings with the class labels above shows two separate clusters being formed in the 2D space. There is overlap between points which may not be separable, however the far majority of points could have a linear decision boundary drawn between them. Another option would be to increase the dimension of the UMAP embedding and determine separability in three dimensions.

Random normal vectors of the same dimension, in comparison, show clusters being formed such as the one on the top left. This is however a random generation of normal vectors and would most likely change if generated for a number of iterations. There seem to be two major clusters in this plot which could have a boundary drawn between them.

All in all, both visualizations show potential clusters being formed however the GLoVE embeddings show many points overlapping the same region in the UMAP visualization. Perhaps as a follow up, the feature engineering process could be altered to create more separability to allow for better clustering.