ECE 219 Project 1: End-to-End Pipeline to Classify News Articles

Sarah Wilen
UCLA
ECE Department
swilen1@g.ucla.edu
UID: 305070897

Marco Venegas
UCLA
ECE Department
venmarco310@g.ucla.edu
UID: 805461921

Henry Peters
UCLA
CS Department
hpeters@g.ucla.edu
UID: 205704526

1. Question 1

1.1. A

There are 3150 data samples and there are 8 features in the dataset.

1.2. Histograms

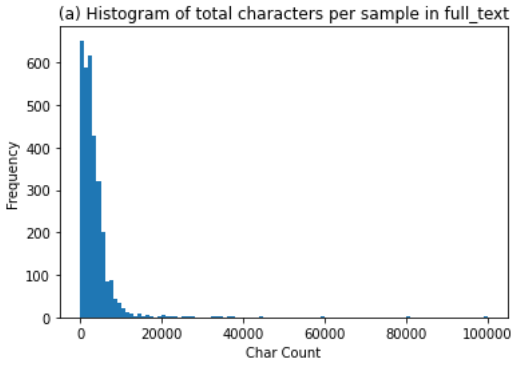

Figure 1. Alpha-numeric characters per data point

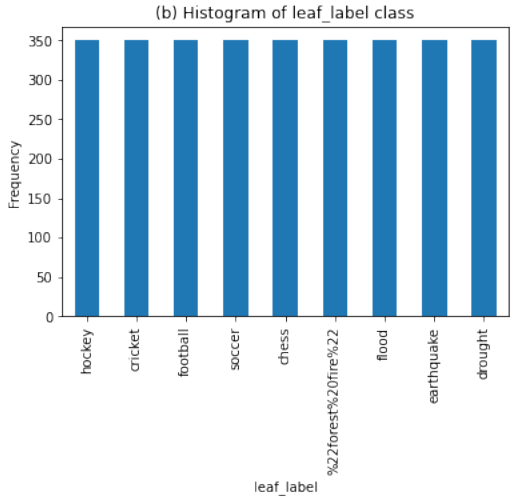

Figure 2. Leaf Label Histogram

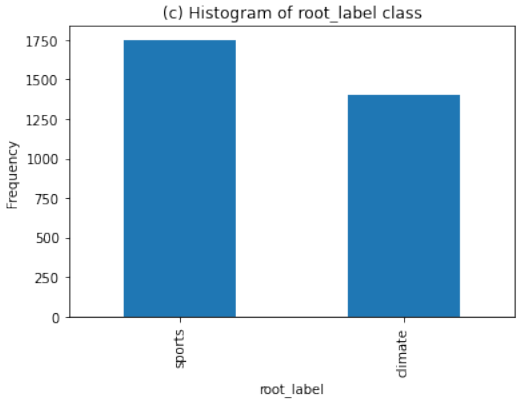

Figure 3. Root Label Histogram

1.3. Interpret Histograms

Looking at Figure 1, we see that the majority of samples have at most 10,000 characters in the text.

In figure 2, we see there is an even distribution between leaf nodes. If the leaf nodes were not balanced, then our classifier would have issues with overfitting to a specific leaf node. Furthermore, if there was an imbalance in leaf nodes, then some other approaches would be needed to mitigate this issue such as data augmentation, resampling, and oversampling/undersampling.

Figure 3 shows that the root label can be split into 2 categories: sports and weather. However, we see that the sports root label has more data points than the weather root label. The underlying data set is a bit imbalanced. There are about 500 more data points for sports, which can possibly lead to higher losses due to overfitting.

2. Question 2

Train1_test_split results in 2520 training samples and 630 test samples.

3. Question 3

3.1. Bullet 1

Stemming cuts off the suffixes that can be found in an inflected word to result in a stem. The benefits of stemming are that it's very simple and easy to run. It is not a very complex algorithm, so if a user were to have a large data set, it would be able to process it quickly. Stemming is mostly used to index documents. A disadvantage is that stemming crudely chops off the end of a word until a stem is reached, therefore, the stem is occasionally not a meaningful word.

Lemmatization uses the context of the word to return the pre-learned dictionary form of a word, the lemma. The benefit of lemmatization is that it is much more accurate than its stemming counterpart. Lemmatization is more informative than stemming because it uses surrounding text to determine a word's part of speech. Additionally, the returned lemma is an actual word. A disadvantage of lemmatization is that we need to have a detailed dictionary so that the algorithm can link the word to its base lemma. This is computationally slower than stemming.

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base.

Ideally, the dictionary size output would be the same because both lemmatization and stemming boil down the input to its inflection words. However, if there are inputs that heavily use pronouns, there the dictionary size for stemming would be smaller in size due to the process of inflection would remove them.

3.2. Bullet 2

min_df means that when building the vocabulary, ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. min_df essentially is used to return terms that are rare. The larger min_df is, the more that infrequent terms will be removed.

Varying min_df changes the number of columns in the TFIDF matrix. A higher min_df means that more words will fall below the threshold so there will be fewer columns in the final matrix. This hypothesis held when varying the min_df in our experiments.

3.3. Bullet 3

Since lemmatization tags the position of every word based on the sentence structure, our code removes stop words after lemmatization in case the stop words inform the context of the to-be-lemmatized words. However, since stop words are words used so commonly, they might not carry much useful information that informs the context of the lemmatized words, so we tried removing the stop words before and after lemmatization, and both had similar results.

Punctuation and numbers can be removed before lemmatization because these features do not provide more information about the inflection of words, so not much information would be lost when choosing to remove them.

3.4. Bullet 4

For the training set, after cleaning and lemmatization, TF-IDF processed data has a shape of (2520, 15076) For the testing set, after cleaning and lemmatization, TF-IDF processed data has a shape of (630, 15076) We see that we still retain the same amount of train and test samples from the train-test split.

4. Question 4

4.1. Bullet 1

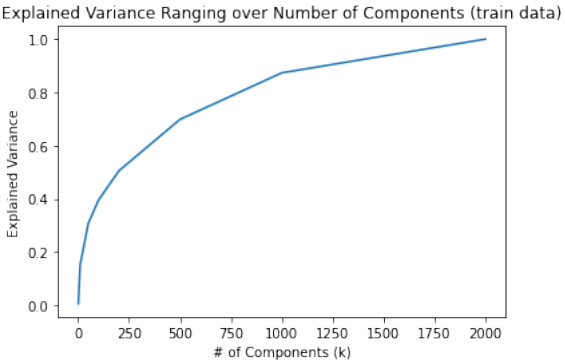

Figure 4. Explained Variance Ratio for different dimensions

The explained variance plot is expectantly concave down, increasing meaning that as we increase the number of principal components, the explained variance increases at a decreasing rate. This makes sense because as the number of components increase, we get closer to having the same amount of features as the original data matrix, so the explained variance converges towards 100%.

4.2. Bullet 2

MSE Train LSI: 1689 MSE Test LSI: 448
MSE Train NMF: 1713 MSE Test NMF: 454
The result of the MSE shows that the LSI reduction dimension representation seems to maintain more information about the input than the NMF method.

NMF works best for shorter texts. NMF calculates how well each doc fits each topic rather than assuming a document has multiple topics. LSI assumes each document has multiple topics and works best with long texts, which we are using in full_text.

5. Question 5

All scores reported in this section weight all classes equally, rather than weighting each class's contribution to the overall score based on the number of samples from each class (macro accuracy).

5.1. Linear SVM γ =1000 (hard margin)

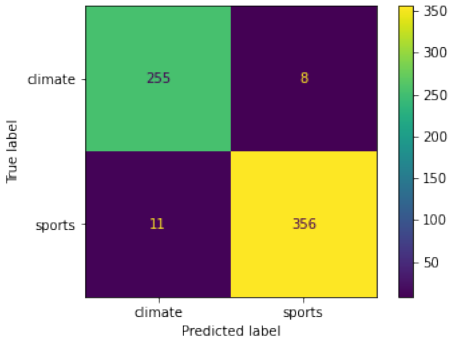

Figure 5. Confusion Matrix for SVM γ=1000

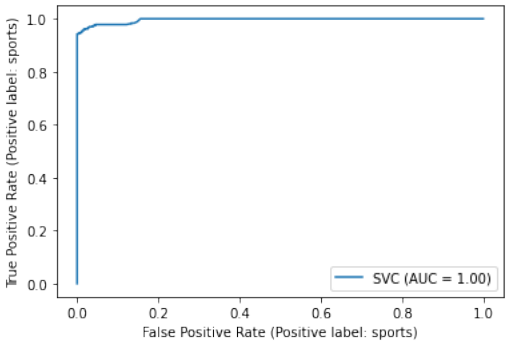

Figure 6. ROC for SVM γ=1000

Accuracy	0.97
Recall	0.97
Precision	0.97
F1 Score	0.97

5.2. Linear SVM γ =0.0001 (soft margin)

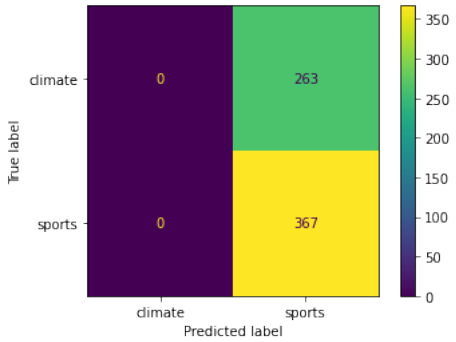

Figure 7. Confusion Matrix for SVM γ=0.0001

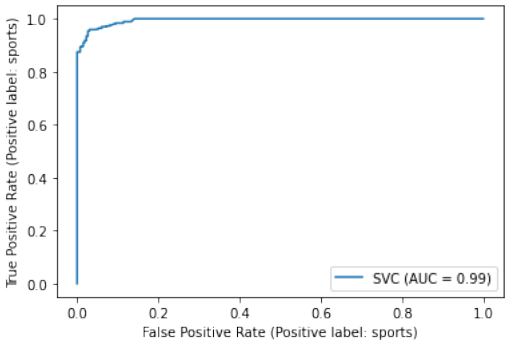

Figure 8. ROC for SVM γ=0.0001

Accuracy	0.58
Recall	0.5
Precision	0.29
F1 Score	0.37

5.3. Analysis for Soft SVM

We see that the Soft SVM performs poorly compared to the Hard SVM. Since the regularization parameter is set to 0.0001, the model does not penalize its prediction when making an error. Equally important, this goes to show that the model is underfitting since it cannot generalize the inputted data. This is due to the fact that the classifier is allowed to make a higher misclassification rate. So it can seem that the classifier is just classifying for a specific class, which we see it does. Furthermore, this means that all classes will be categorized into one class (the sports class). In terms of the confusion matrix, we see that our classifier is predicting the label to be '1' all of the time. This is not good because we are just leaving it up to luck to see if the label is correct. This supports our claim that the classifier is only classifying for one class since we do not have a tight error tolerance. This is further evidence that the classifier is underfitting the data since it cannot make a good prediction. The ROC does not accurately reflect the performance. While it is true that the ROC looks almost perfect for the sports category because our classifier is classifying all samples into sports, so TPR = 1 and FPR =0, we should not take this ROC as an indication that our classifier is perfect, obviously, from the confusion matrix we see that it is not. If we ran the ROC on the climate class, we could see that TPR = 0 and FPR = 0 as well because the soft classifier never classifies anything as climate.

5.4. Linear SVM γ =100000 (very-hard margin)

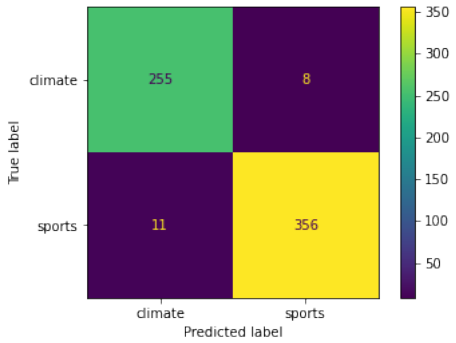

Figure 9. Confusion Matrix for SVM γ=100000

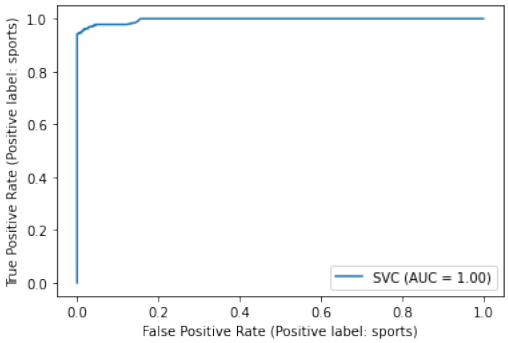

Figure 10. ROC for SVM γ=100000

Accuracy	0.97
Recall	0.97
Precision	0.97
F1 Score	0.97

5.5. Cross Validation

Using cross-validation we found that the best parameter for SVM training was $\gamma = 0.5$.

5.6. Linear SVM γ =0.5 (cross-validated)

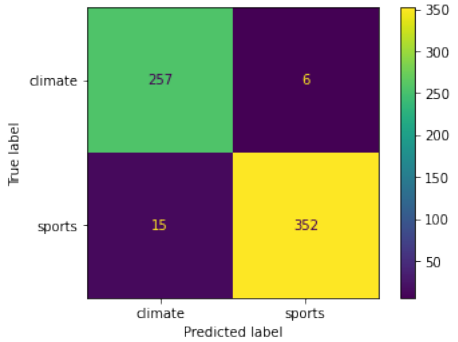

Figure 11. Confusion Matrix for SVM γ=0.5

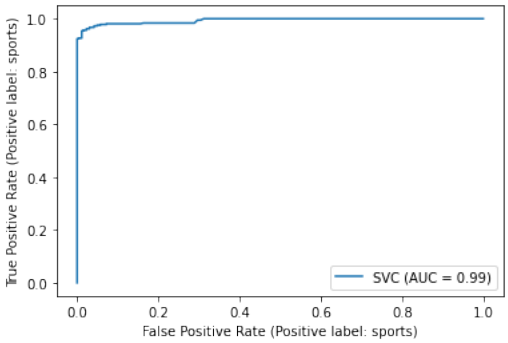

Figure 12. ROC for SVM γ=0.5

Accuracy	0.97
Recall	0.97
Precision	0.96
F1 Score	0.97

6. Question 6

6.1. Logistic Regression without Penalty

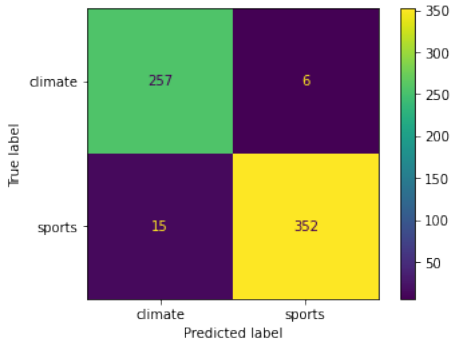

Figure 13. Confusion Matrix for Logistic Regression (no penalty)

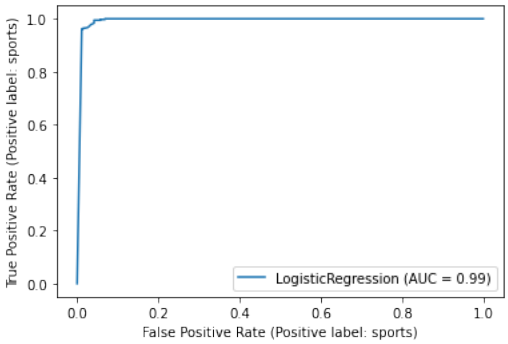

Figure 14. ROC for Logistic Regression (no penalty)

Accuracy	0.97
Recall	0.97
Precision	0.97
F1 Score	0.97

6.2. Logistic Regression with L1 Regularization (cross-validated)

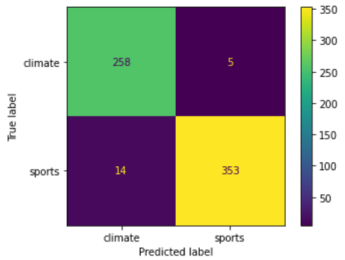

Figure 15. Confusion Matrix for Logistic Regression (L1, C=1/10)

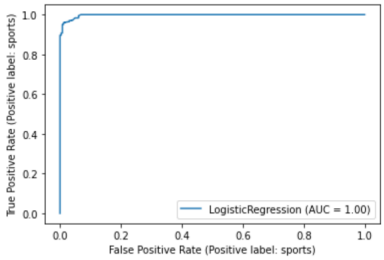

Figure 16. ROC for Logistic Regression (L1, C=1/10)

Accuracy	0.97
Recall	0.97
Precision	0.97
F1 Score	0.97

6.3. Logistic Regression with L2 Regularization (cross-validated)

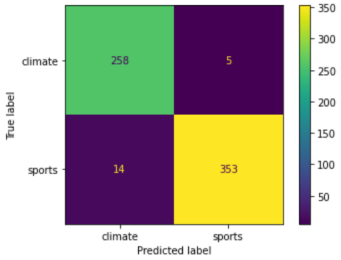

Figure 17. Confusion Matrix for Logistic Regression (L2, C=1/10)

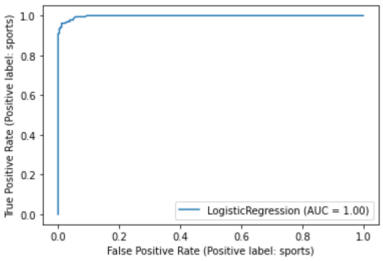

Figure 18. ROC for Logistic Regression (L2, C=1/10)

Accuracy	0.97
Recall	0.97
Precision	0.97
F1 Score	0.97

6.4. Regularization Parameter Comparison (Bullet 2)

Firstly, the way that the regularization parameters are calculated is different. L1 regularization penalizes on the basis of the sum of the absolute values of weights. L1 technique is good for feature selection and it has a sparse solution. Meaning,

where a feature has 0 weight to it, the L1 regularization will be able to discard it and do fine when training the classifier. One might be interested in this technique because it is robust to outliers and can generate simple, interpretable models.

L2 regularization penalizes on the basis of the sum of square weights. L2 technique does not necessarily do any feature selection, instead, it can give more accurate predictions when the output variable is the whole function of the whole input variables. One might be interested in this technique because it can help to learn more complex data patterns.

There is a huge difference in how logistic regression and linear SVM output their respective decision boundary. Logistic regression makes its decision on the probability that data input is likely to be a certain class, this can be a confidence level. Linear SVM essentially uses closely related data points to create a hyperplane that tries to separate different classes. There is obviously some margin with these data points (which are also called support vectors), that help with the generalization of data. So when a new data point is going to be classified, it is classified where the data point lies with respect to the hyperplane.

7. Question 7

7.1. Naive Bayes

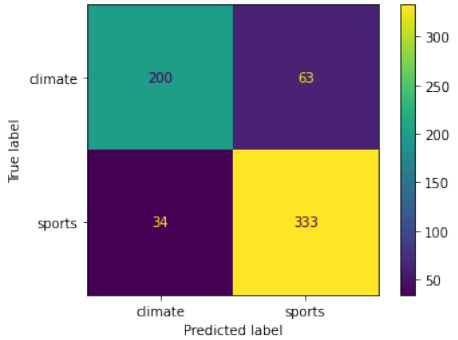

Figure 19. Confusion Matrix for Naive Bayes

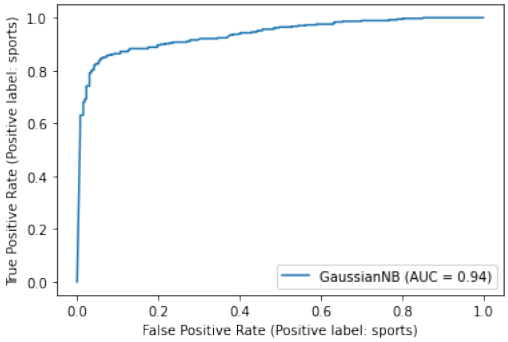

Figure 20. ROC for Naive Bayes

Accuracy	0.85
Recall	0.83
Precision	0.85
F1 Score	0.84

8. Question 8

8.1. 5 best combinations

We first individually lemmatized and stemmed the data, so our grid search would not have to implement those functions and take a long time to run.

1. Feature extraction: lemmatized, min_df=5, LSI (n_components=80), Model: LogisticRegression(C=10, penality='l1')... **accuracy = 0.9623**
2. Feature extraction: lemmatized, min_df=3, LSI (n_components=80), Model: LogisticRegression(C=10, penality='l1')... **accuracy = 0.9619**
3. Feature extraction: lemmatized, min_df=5, LSI (n_components=80), Model: SVM(C=0.5)... **accuracy = 0.960**
4. Feature extraction: lemmatized, min_df=3, LSI (n_components=80), Model: SVM(C=0.5)... **accuracy = 0.9595**
5. Feature extraction: lemmatized, min_df=3, NMF (n_components=80), Model: LogisticRegression(C=10, penality='l1')... **accuracy = 0.9595**

	mean_test_score	params	rank_test_score
21	0.956746	{'dimensionalityReduction': TruncatedSVD(n_com...	1
43	0.956746	{'dimensionalityReduction': NMF(n_components=8...	1
19	0.956746	{'dimensionalityReduction': TruncatedSVD(n_com...	1
18	0.955556	{'dimensionalityReduction': TruncatedSVD(n_com...	4
20	0.955159	{'dimensionalityReduction': TruncatedSVD(n_com...	5
42	0.953968	{'dimensionalityReduction': NMF(n_components=8...	6
47	0.952381	{'dimensionalityReduction': NMF(n_components=8...	7
46	0.952381	{'dimensionalityReduction': NMF(n_components=8...	7
38	0.951984	{'dimensionalityReduction': NMF(n_components=3...	9
17	0.951587	{'dimensionalityReduction': TruncatedSVD(n_com...	10
39	0.951190	{'dimensionalityReduction': NMF(n_components=3...	11
16	0.950397	{'dimensionalityReduction': TruncatedSVD(n_com...	12
34	0.949603	{'dimensionalityReduction': NMF(n_components=3...	13
35	0.948810	{'dimensionalityReduction': NMF(n_components=3...	14

Figure 21. Example Grid Search Table (Lemmatized)

	mean_test_score	params	rank_test_score
19	0.962302	{'dimensionalityReduction': TruncatedSVD(n_com...	1
18	0.961905	{'dimensionalityReduction': TruncatedSVD(n_com...	2
17	0.960317	{'dimensionalityReduction': TruncatedSVD(n_com...	3
16	0.959524	{'dimensionalityReduction': TruncatedSVD(n_com...	4
42	0.959524	{'dimensionalityReduction': NMF(n_components=8...	4
20	0.959127	{'dimensionalityReduction': TruncatedSVD(n_com...	6
43	0.958730	{'dimensionalityReduction': NMF(n_components=8...	7
21	0.957540	{'dimensionalityReduction': TruncatedSVD(n_com...	8
47	0.955952	{'dimensionalityReduction': NMF(n_components=8...	9
46	0.955952	{'dimensionalityReduction': NMF(n_components=8...	9
10	0.949603	{'dimensionalityReduction': TruncatedSVD(n_com...	11
40	0.949603	{'dimensionalityReduction': NMF(n_components=8...	11
35	0.949603	{'dimensionalityReduction': NMF(n_components=3...	11
38	0.949206	{'dimensionalityReduction': NMF(n_components=3...	14
33	0.948810	{'dimensionalityReduction': NMF(n_components=3...	15
11	0.948016	{'dimensionalityReduction': TruncatedSVD(n_com...	16
41	0.947619	{'dimensionalityReduction': NMF(n_components=8...	17
9	0.947619	{'dimensionalityReduction': TruncatedSVD(n_com...	17

Figure 22. Example Grid Search Table (Stemmed)

9. Question 9

9.1. Multiclass: Naive Bayes

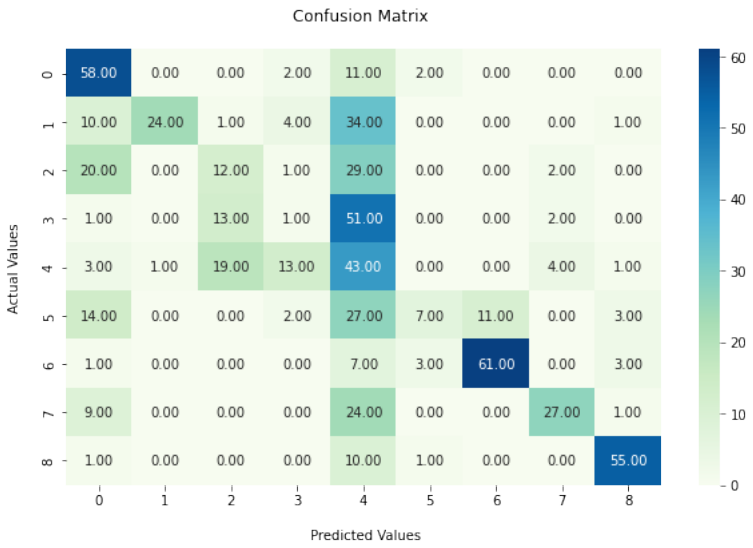

Figure 23. Confusion Matrix for Multiclass Naive Bayes

From the confusion matrix, you can observe that the blocks on the diaganol from the top left to the bottom right have the most values in it. It is clear that the diaganol values are the respective classes' true positive value. The more that the confusion matrix diagonal has values opposed to the rest of the matrix, then you know the accuracy is higher. However, for Naive Bayes, our accuracy was not that high, so you will notice in the matrix that ther are values populated all across the board and not just on the diagonal meaning there were inaccurate labelings.

Accuracy	0.46
Recall	0.45
Precision	0.55
F1 Score	0.45

9.2. SVM: One vs. One

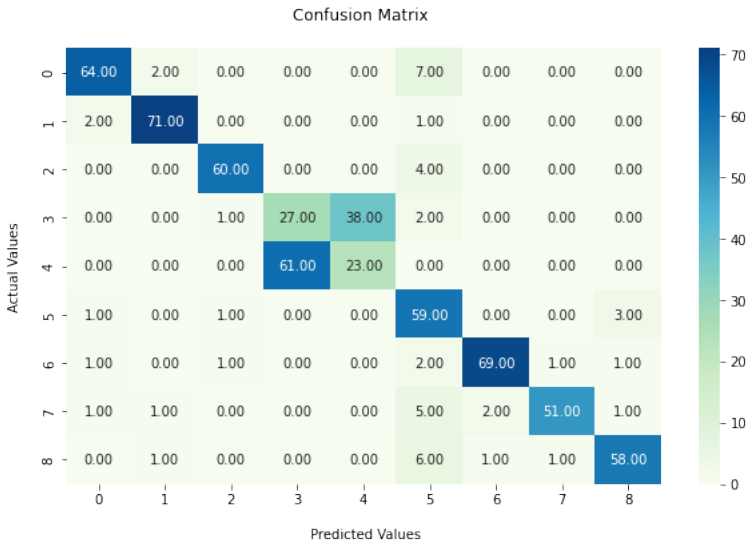

Figure 24. Confusion Matrix for SVM: One vs. One

Accuracy	0.77
Recall	0.78
Precision	0.78
F1 Score	0.78

9.3. SVM: One vs. Rest

We resolved the class imbalance issue for this classifier using random downsampling from the larger class. We picked a number of random samples from the majority class such that we trained on the same number of samples from each class.

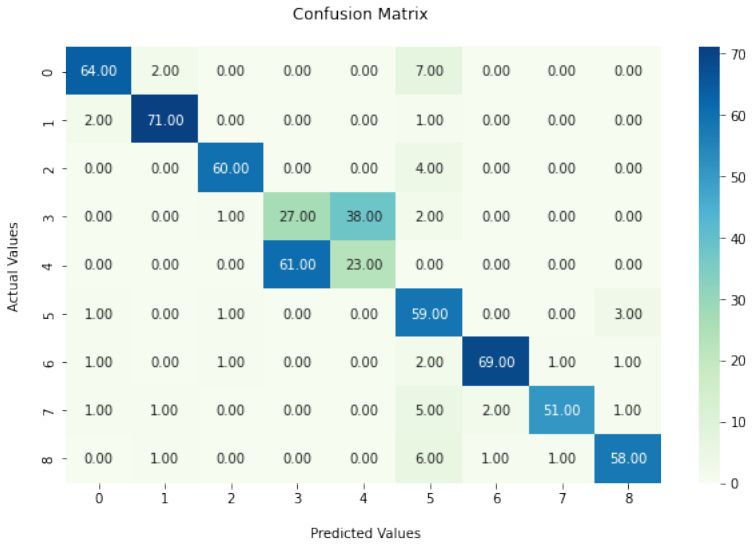

Figure 25. Confusion Matrix for SVM: One vs. Rest

Accuracy	0.77
Recall	0.78
Precision	0.78
F1 Score	0.78

9.4. Multiclass: New Labeling

Based on out 9x9 confusion label, we can see that the categories ”football” and ”soccer” have relatively close labeling rates, false positive rate, and true positive, with one another. This may be due to semantics where some papers refer to ”soccer” as ”football”. A suggestion we have is to combine these two categories into one larger one. After merging these two labels, the accuracy of the classifier jumped from 0.77 to 0.92 in both the One vs. One and the One vs. Rest models. We report the confusion matrices for both models below.

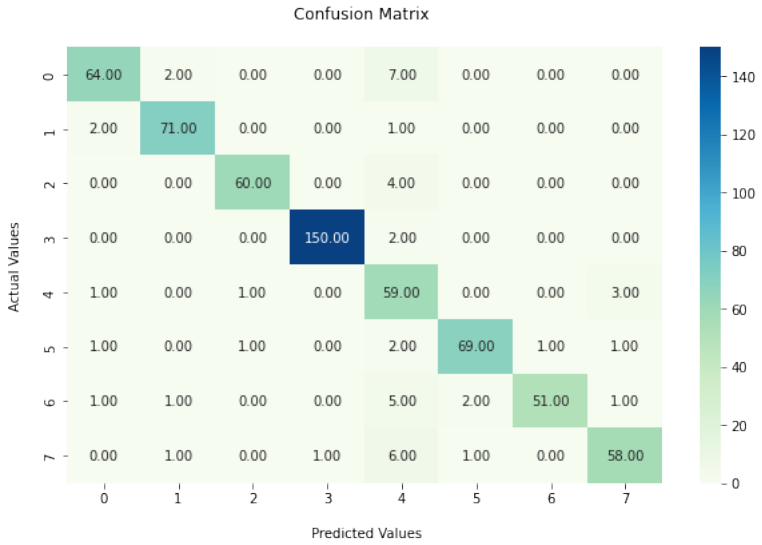

Figure 26. Confusion Matrix for SVM with Merged Labels: One vs. One

Accuracy	0.91
Recall	0.91
Precision	0.92
F1 Score	0.91

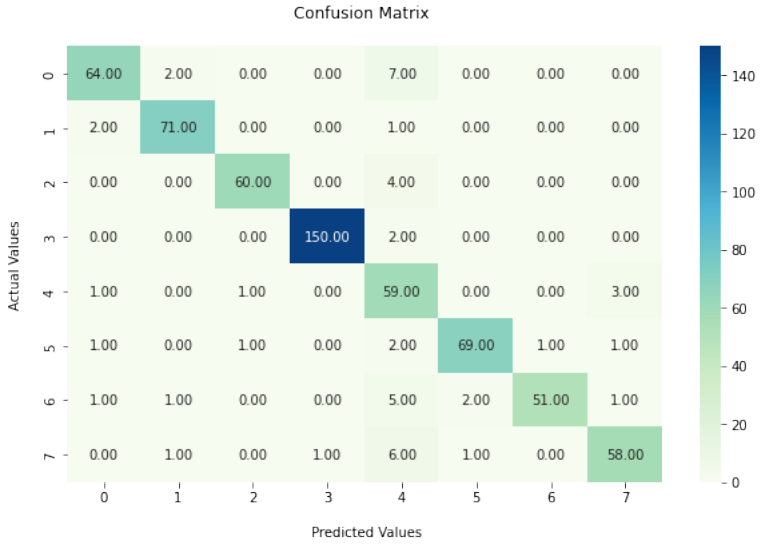

Figure 27. Confusion Matrix for SMV with Merged Labels: One vs. Rest

Accuracy	0.91
Recall	0.91
Precision	0.92
F1 Score	0.91

9.5. Multiclass: Imbalance Impact

In general, a class imbalance will affect the performance of the classification because the classifier will have more data points for this newly combined class. However, in our case, we did not see mich difference in accuracy for the imbalanced and balanced cases, so it did not affect us. We resolve this class imbalance with random downsampling from the majority class. After this downsampling, the accuracy is 0.92 for both models and the confusion matrices are shown below.

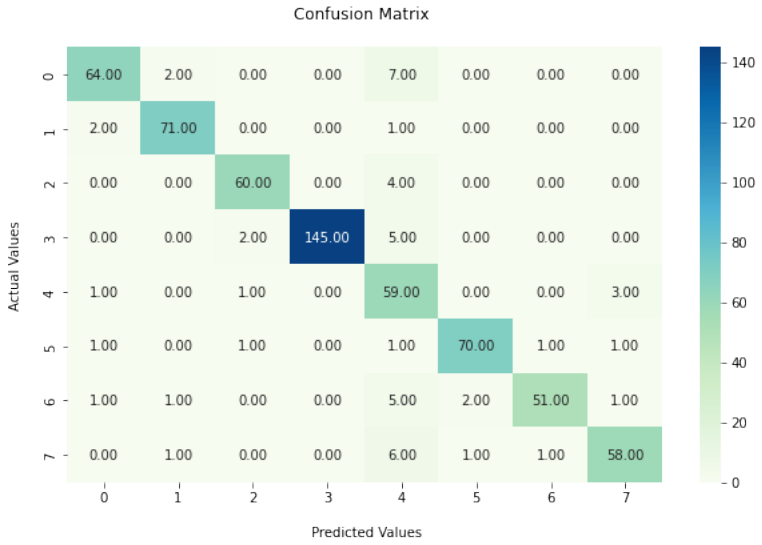

Figure 28. Confusion Matrix for SVM with Class Balance: One vs. One

Accuracy	0.91
Recall	0.91
Precision	0.91
F1 Score	0.90

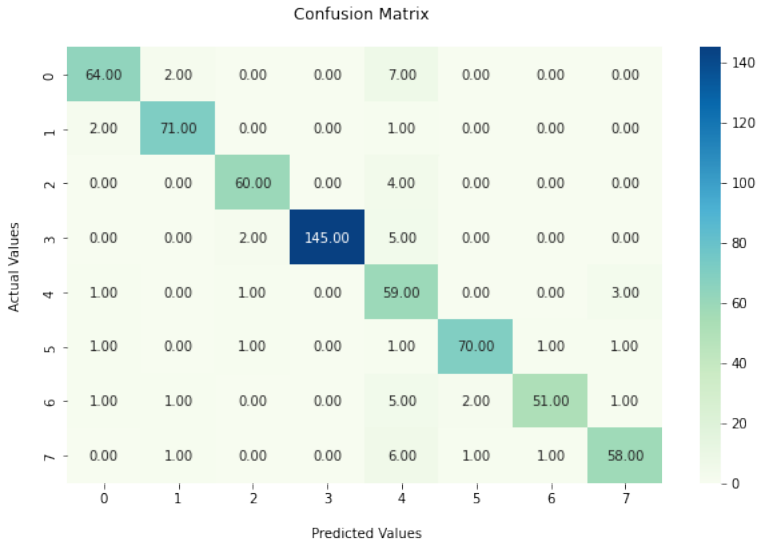

Figure 29. Confusion Matrix for SMV with Class Balance: One vs. Rest

Accuracy	0.91
Recall	0.91
Precision	0.91
F1 Score	0.91

10. Question 10

10.1. Question A

The ratio of co-occurrence conditional probabilities of a target word k with respect to a baseline pair of words (i,j) has more contextual information than the raw conditional probabilities of the word pair. As seen in the example, co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus. Only in the co-occurrence ratio does noise from non-discriminative words like water and fashion cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific to steam.

10.2. Question B

The outputs would be the same. During training, GLoVE embedding attempts to create word vector representations such that the co-occurrences of similar words are maximized. Once the GLoVE model is trained, the word representation of a target word is an average of all the different contexts that the word could've been in. Therefore, although running is used in different contexts in this case, the vector representation, which has been pre-calculated during training will be the same because, in training, the representation has already taken into account the different contexts.

10.3. Question C

For the first "equation", if we took queen - king (remove the regalness of the word), then add the marriage context of husband, then we would result in a vector value close to the vector representation of wife. Therefore, this whole equation should be somewhat close to zero.

Additionally, we may expect that the vector distance between queen and king is roughly comparable to wife and husband because these are common gender relationships. However, I expect that husband and wife are closer related to each other than King and Queen are because of how common it may be in the Wiki dictionary GLoVE training set.

10.4. Question D

It would be preferred to do lemmatization versus stemming. Stemming essentially is a very simple way to break down an input into its inflection points, which is sometimes not a real word, which can lead to a higher error rate due to trying to simplify as much as possible. Whereas lemmatization takes more into account such as context, and part of speech, among other factors, to help break down an input. It will make more meaningful lemmas (real words) when parsing up an input, which will retain more important information versus stemming.

11. Question 11

11.1. GLoVE Based Feature

```
def get_glove_text(row):
  vec = None
  count = 0
  for word in row.split():
    if count==0 and word in embeddings_dict:
      vec = embeddings_dict[word]
      count = count+1
    elif word in embeddings_dict:
      vec = vec + embeddings_dict[word]
      count = count+1
  return vec/count # averaging
```

Figure 30. Word Embedding Code

We tested both keyword and full_text features to embed but found that the full_text feature provides better accuracy in classification. Our GLoVE feature is fairly simple, we essentially put in the full_text from our data set and get its output embedded vector. The output embedded vectors will then be aggregated together and they will be averaged when returned.

11.2. Classifier

After cross-validating logistic regression and SVM with different hyperparameters, we resulted in using Hard SVM. Here are the following metrics.

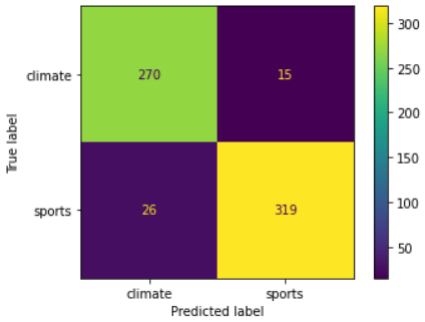

Figure 31. Confusion Matrix for SVM with GLoVE

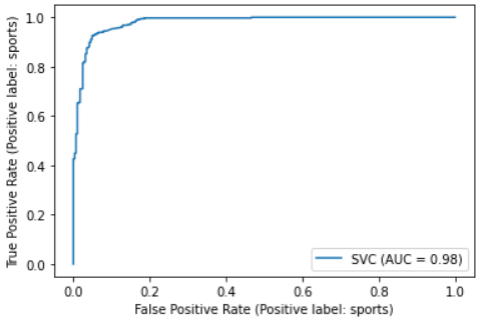

Figure 32. ROC for SVM with GLoVE

Accuracy	0.93
Recall	0.93
Precision	0.94
F1 Score	0.94

12. Question 12

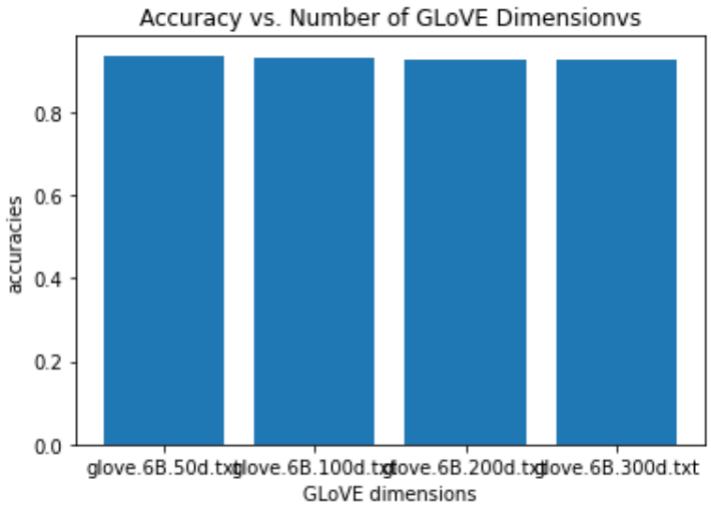

Figure 33. Accuracy vs. GLoVE Dimensions

Dimension	Accuracy
50	0.938
100	0.930
200	0.928
300	0.926

This trend that we see is not expected. Ideally what is expected is when the dimensions increase, the accuracy should increase as well. With increasing dimensions, we can capture more semantics and part of speech context from the input. However, one reason why we can get good results is from lemmatization. Lemmatization is another way for our model to keep important information, so when we pair it with the GLoVE feature, we have already captured a good amount of information.

13. Question 13

We are asked to picture both GLoVE word embeddings and a random vector on a 2D mapping using UMAP. We used dimensions set to 300. We can see in GLOVE-Based Embedding we can distinguish cluttering between both classes, which goes to show that the word embedding is pretty good to help a classifier learn, despite some outliers. Equally important, it looks fairly balanced between classes, meaning we will have good data for a classifier, and no other techniques would be needed to mitigate an imbalance. Looking at the random vector distribution we can see there is no clear pattern between the classes.

13.1. GLoVE-Based Embedding

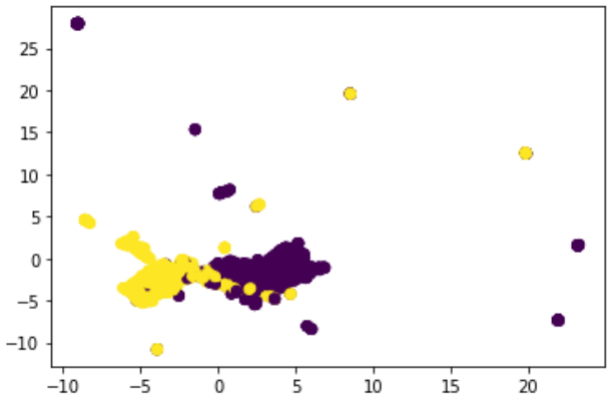

Figure 34. UMAP of GLove Based Embedding

13.2. Normal Random Vectors

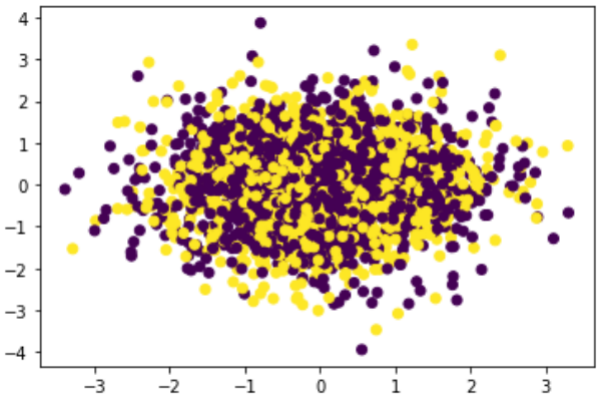

Figure 35. UMAP of Normal Random Vectors