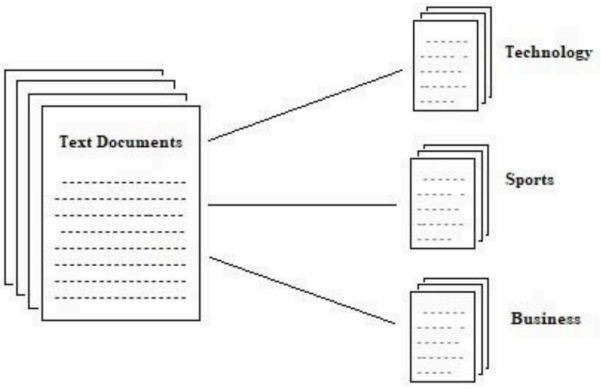
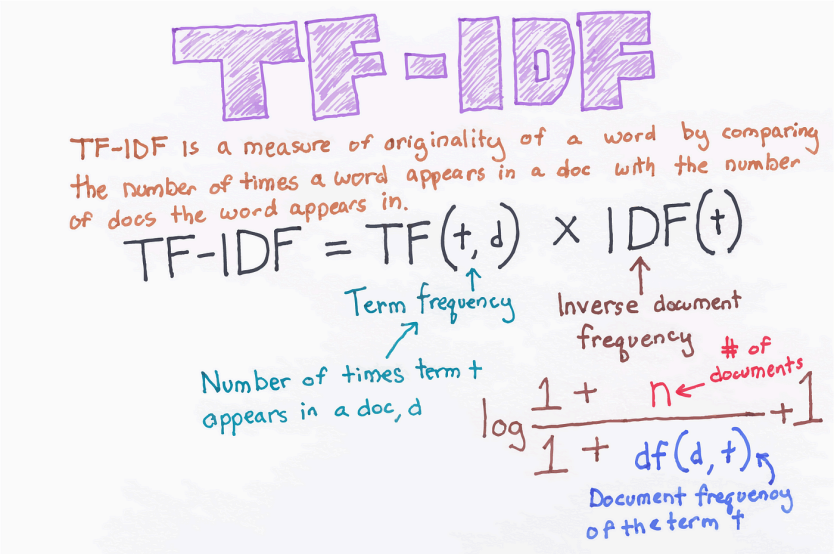
Project 1: End-to-End Pipeline to Classify News Articles Report

Authors: Alex Chen, Chang Yang Liu, Jiamin Xu

Electrical and Computer Engineering 219: Winter 2024

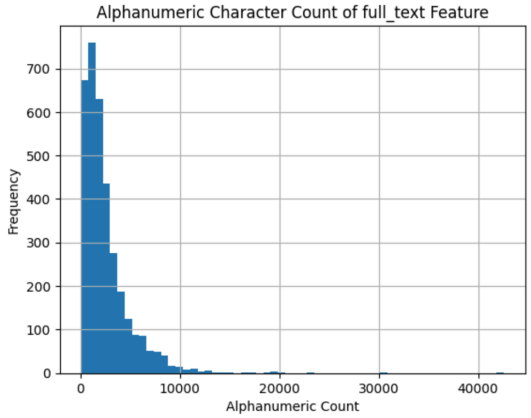
Professor Vwani Roychowdhury





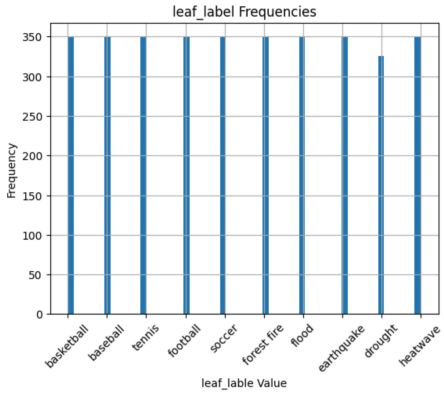
Question 1

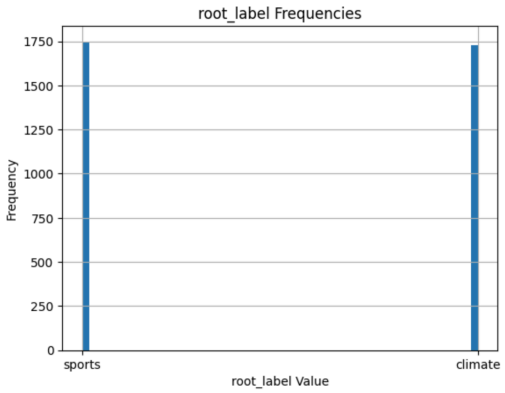
There are 3476 rows and 8 columns in the dataset. Three histograms are shown below to help gain a better understanding of the data.



The first histogram shows the count of alphanumeric characters in each datapoint's full_text feature. From this histogram, we can see that median character count for the news articles is around 2500. The data is skewed right, meaning that some articles are much longer in terms of character text (beyond 10000), increasing the mean beyond the median. This makes sense because some articles may be particularly interesting and have more relevant content to write about.

The next histogram shows the breakdown of leaf labels of news articles. The selected datapoints are pretty uniformly distributed here, as each of 10 leaf labels has around 350 articles, with the exception of droughts, which has slightly less. The last histogram does the same for root_label. There are only two root labels, sports and climate, each with around 1750 corresponding articles.





Question 2

There are 2780 training and 696 testing samples.

Question 3

The benefit of using lemmatization over stemming is semantically similar words are reduced. For example, “better” would be lemmatized into “good,” while “better” would be unchanged when stemmed. Thus, for our use case, words similar in meaning will be grouped together as much as possible before processing. A disadvantage of doing this is that sometimes the lemmatized form of a word has a different contextual meaning or implication. For example, the word “better” may be used commonly in sports, while “good” is more common in weather (hypothetically). If the words are combined together, we may lose important information. Further, stemming is likely more computationally inexpensive since there is less transformation that needs to occur.

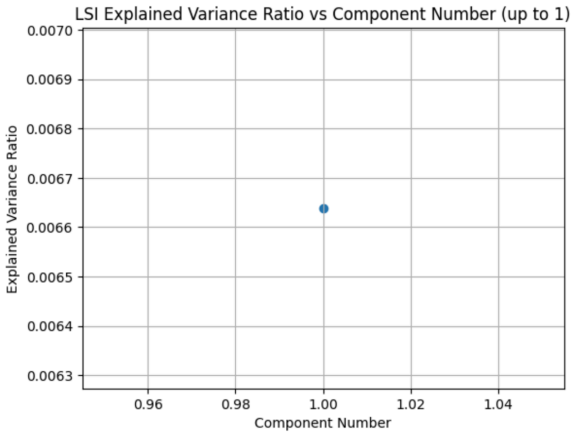
As we increase the minimum document frequency, the TF-IDF matrix decreases in size. Specifically, the columns decrease since we have less words that meet the minimum threshold to be considered. Rows stay same since rows represent documents which is unchanged.

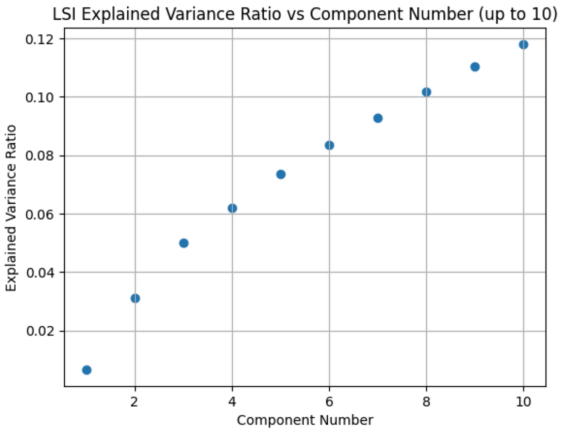
We should remove stopwords after lemmatizing since words that are lemmatized into a stopword should also be removed. Same with punctuations and numbers. Since we input the full sentence into the lemmatizer, we are lemmatizing first before using the count vectorizer which remove stopwords.

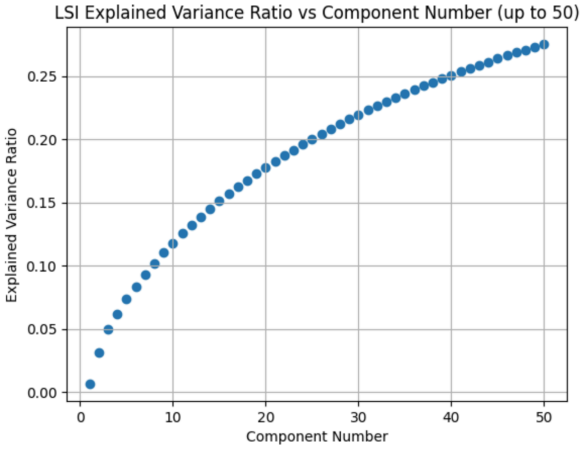
The TF-IDF processed train and test matrices are of size (2780, 12781) and (696, 12781), respectively.

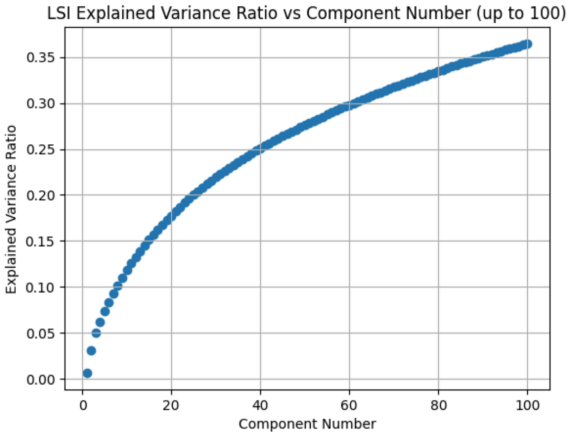
Question 4

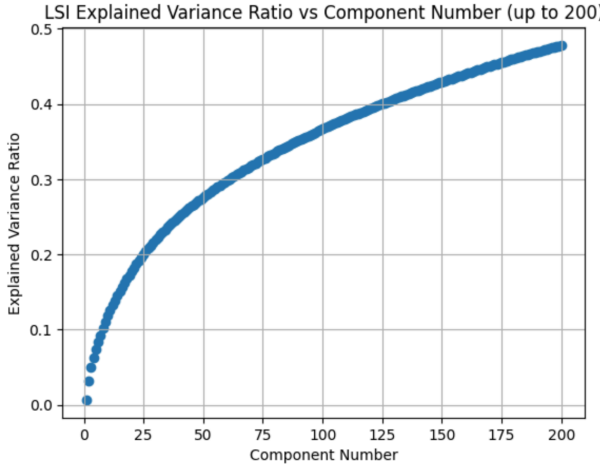
The following plots show how the explained variance ratio varies with component number, for several component counts used (k = [1, 10, 50, …, 2000]. Each plot has the cumulative variance sum plotted against the component number for the specific k value used in that LSI computation. The plot looks clearly concave, and as k increases, the concavity becomes more smooth. The explained variance only reaches near 1 as k is 2000. Furthermore, for each plot, the concavity suggests that as we add more components, the increase in the amount of variance explained decreases (rate of increase is decreasing, negative second derivative).

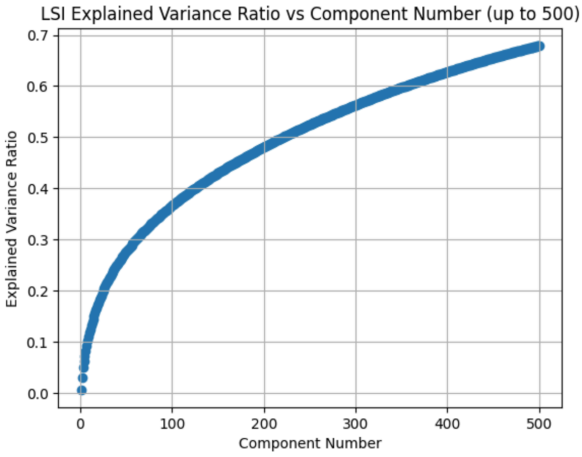

LSI Explained Variance Ratio vs Component Number (up to 1)
Explained Variance Ratio
Component Number

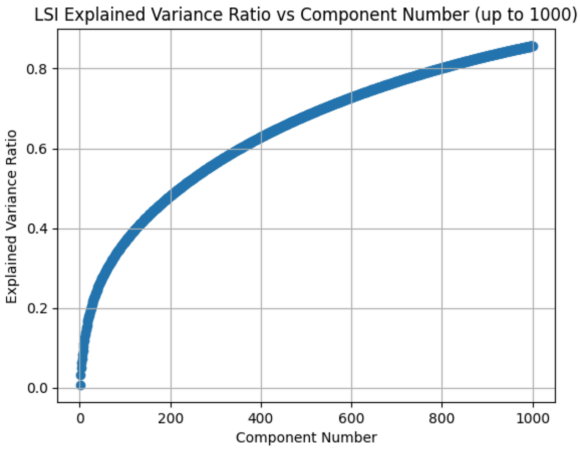

LSI Explained Variance Ratio vs Component Number (up to 10)
Explained Variance Ratio
Component Number

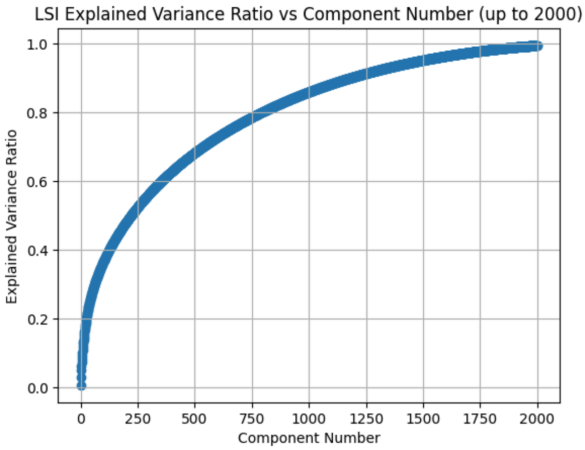

LSI Explained Variance Ratio vs Component Number (up to 50)
Explained Variance Ratio
Component Number


LSI Explained Variance Ratio vs Component Number (up to 100)
Explained Variance Ratio
Component Number

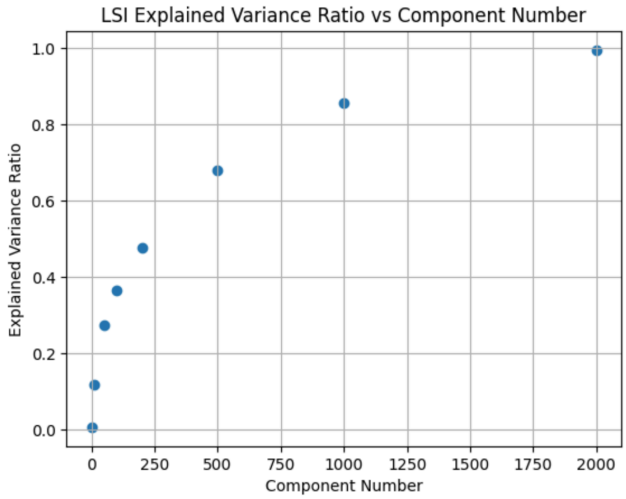

LSI Explained Variance Ratio vs Component Number (up to 200)
Explained Variance Ratio
Component Number


LSI Explained Variance Ratio vs Component Number (up to 500)
Explained Variance Ratio
Component Number



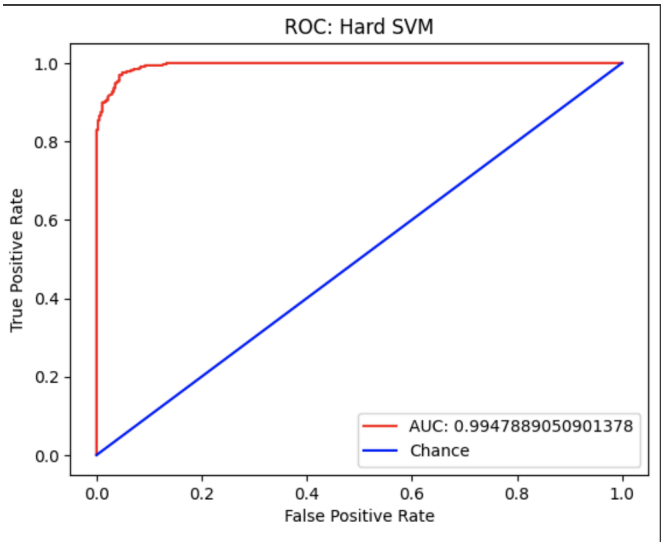


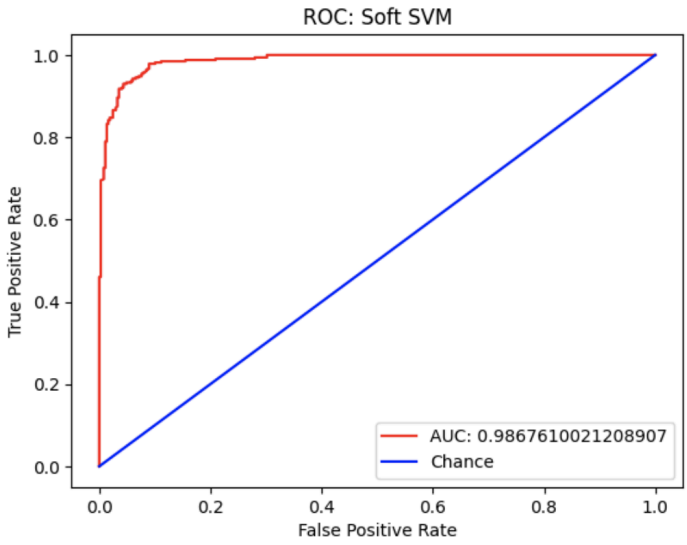
I have also plotted the total explained variance ratio versus the component number, rather than plotting the explained variance ratio for each component number. A similar trend exists; there is a point from which less increase / benefit comes from increasing component number.

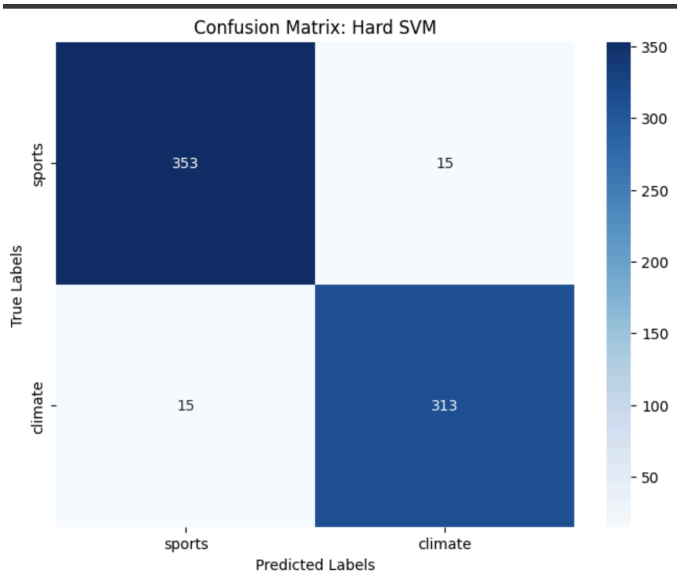


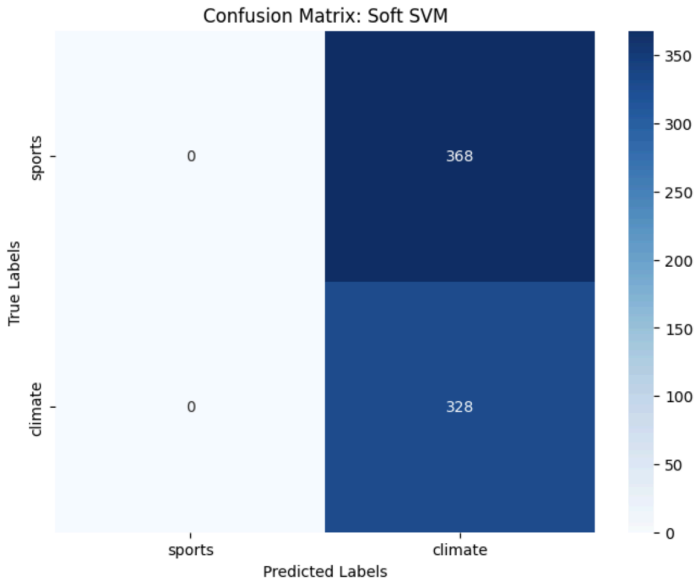
The error is lower for LSI than NMF (465.40 vs 520.86 for test data, 1949.00 vs 1979.19 for train data).. This is a result of LSI having more freedom in the values generated in the reduced feature matrix, while NMF has more constraints. For example, NMF only allows for positive values as well as the constraints that W >= 0 and H >= 0. Thus, LSI is able to better represent the original, sparse matrix and find the vectors that minimize error most.

Question 5





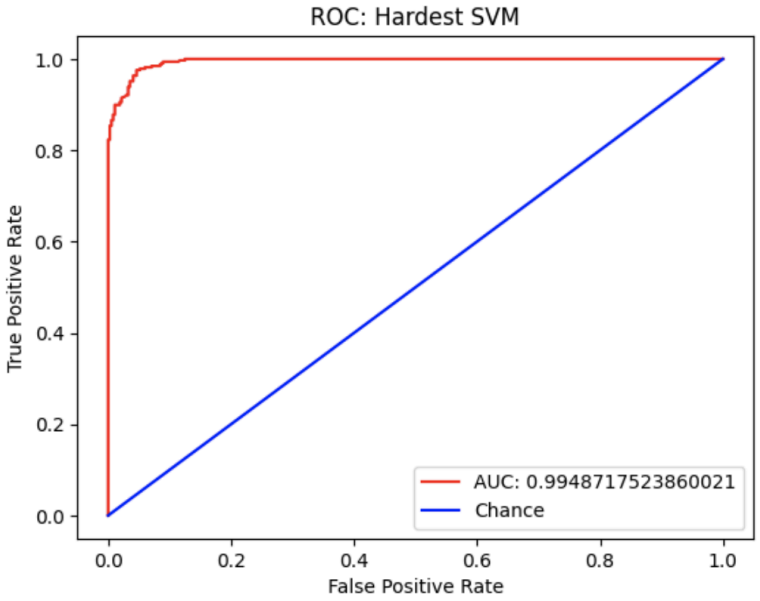


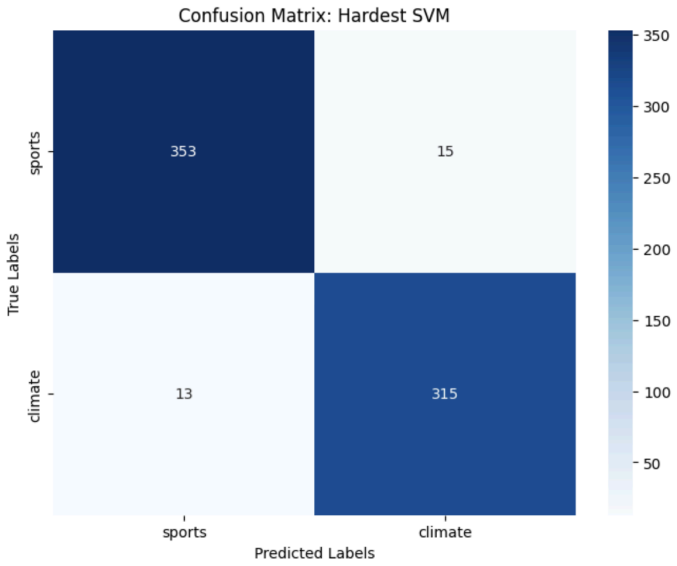


Accuracy hard SVM: 0.9568965517241379
Precision hard SVM: 0.9542682926829268
F1 Score hard SVM: 0.9542682926829268
Recall hard SVM: 0.9542682926829268

Accuracy soft SVM: 0.47126436781609193
Precision soft SVM: 0.47126436781609193
F1 Score soft SVM: 0.640625
Recall soft SVM: 1.0

The hard margin SVM performed significantly better than the soft margin one.





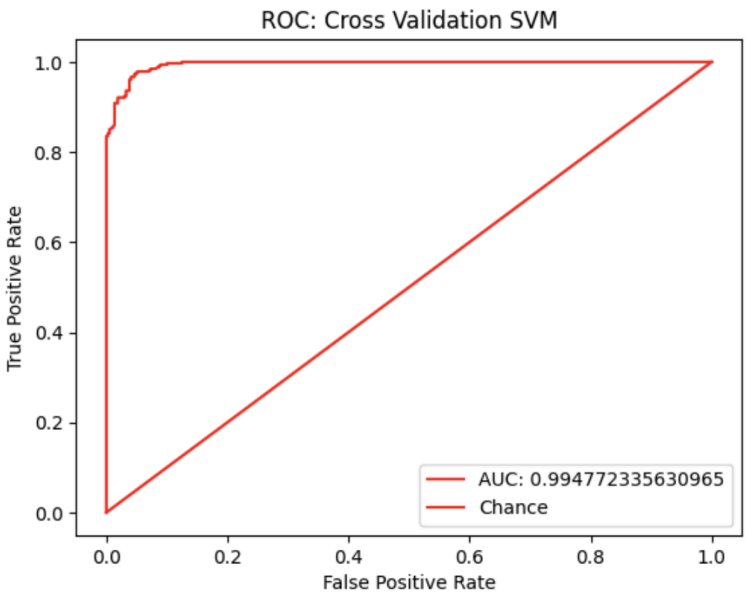
γ = 100000
Accuracy hardest SVM: 0.9597701149425287
Precision hardest SVM: 0.9545454545454546
F1 Score hardest SVM: 0.9574468085106382
Recall hardest SVM: 0.9603658536585366

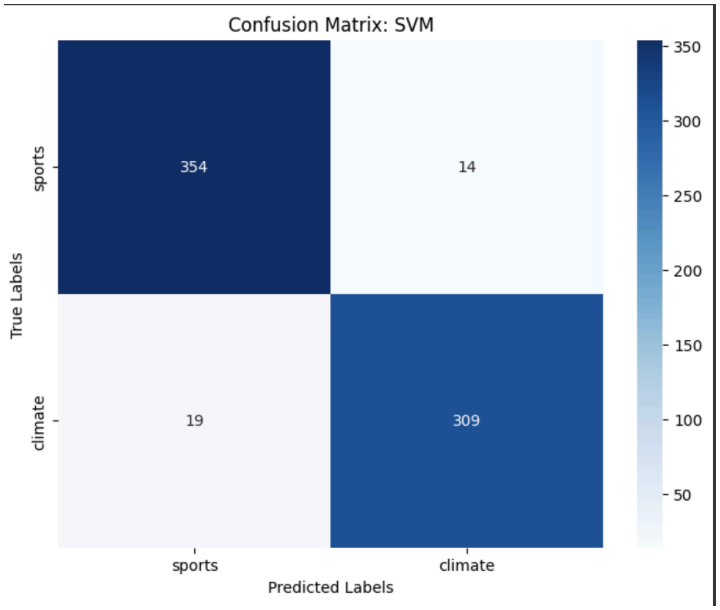
The SVM with γ = 100000 performed marginally better than the other hard margin one.

The soft margin SVM classifier allows for a wider margin when the data is linearly separable, to prevent the model from being prone to overfitting or being too sensitive to outliers, in order for it to generalize better. When γ is small, misclassifications aren't penalized as much. As shown in the soft-margin SVM confusion matrix above, the model predicted all the climate labels correctly, but misclassified all the sports labels.

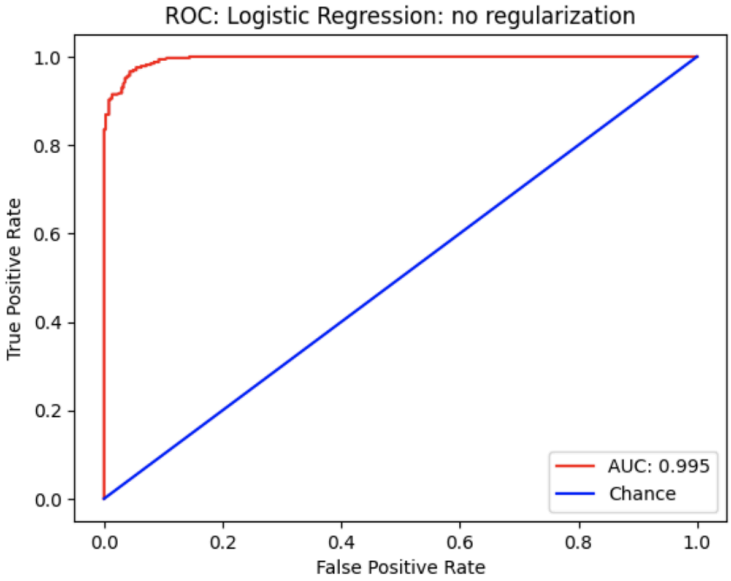
When examining the ROC curve, it does not reflect the performance of the SVM, as the AUC value is still very high and looks very similar to the hard margin SVMs.

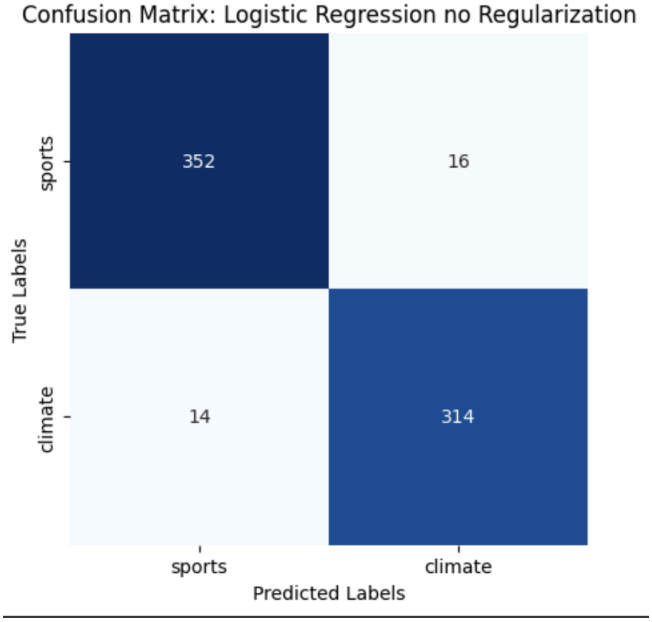
Cross Validation Best Gamma: 10





Accuracy SVM: 0.9525862068965517
Precision SVM: 0.9566563467492261
F1-Score SVM: 0.9493087557603687
Recall SVM: 0.9420731707317073





Question 6

Accuracy (Logistic Regression no regularization): 0.9568965517241379
Recall (Logistic Regression no regularization): 0.9573170731707317
Precision (Logistic Regression no regularization): 0.9515151515151515
F1-Score (Logistic Regression no regularization): 0.9544072948328268

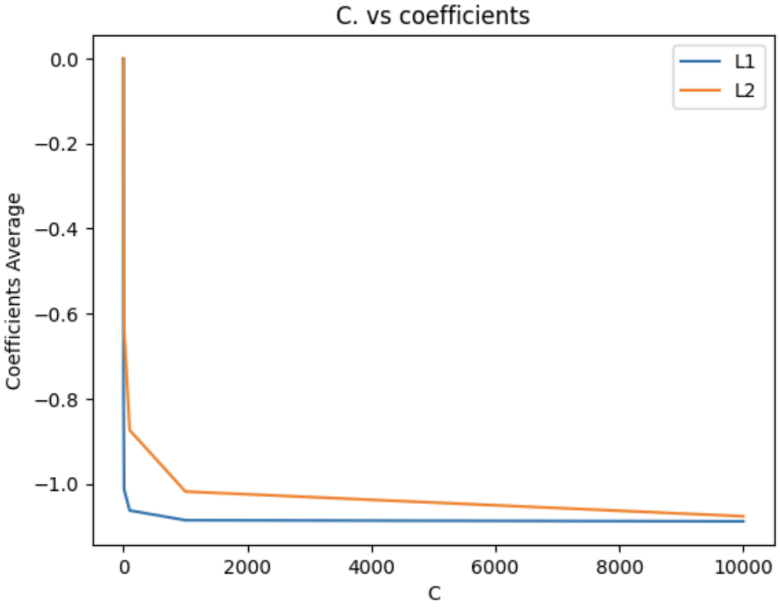
L1 Parameter: 1e-05 Average Cross Validation Accuracy: 0.4971223021582734
L1 Parameter: 0.0001 Average Cross Validation Accuracy: 0.4971223021582734
L1 Parameter: 0.001 Average Cross Validation Accuracy: 0.4971223021582734
L1 Parameter: 0.01 Average Cross Validation Accuracy: 0.4971223021582734
L1 Parameter: 0.1 Average Cross Validation Accuracy: 0.926978417266187
L1 Parameter: 1 Average Cross Validation Accuracy: 0.9485611510791367
L1 Parameter: 10 Average Cross Validation Accuracy: 0.9535971223021583
L1 Parameter: 100 Average Cross Validation Accuracy: 0.953956834532374
L1 Parameter: 1000 Average Cross Validation Accuracy: 0.9535971223021582
L1 Parameter: 10000 Average Cross Validation Accuracy: 0.9532374100719423
L1 Parameter: 100000 Average Cross Validation Accuracy: 0.9532374100719423

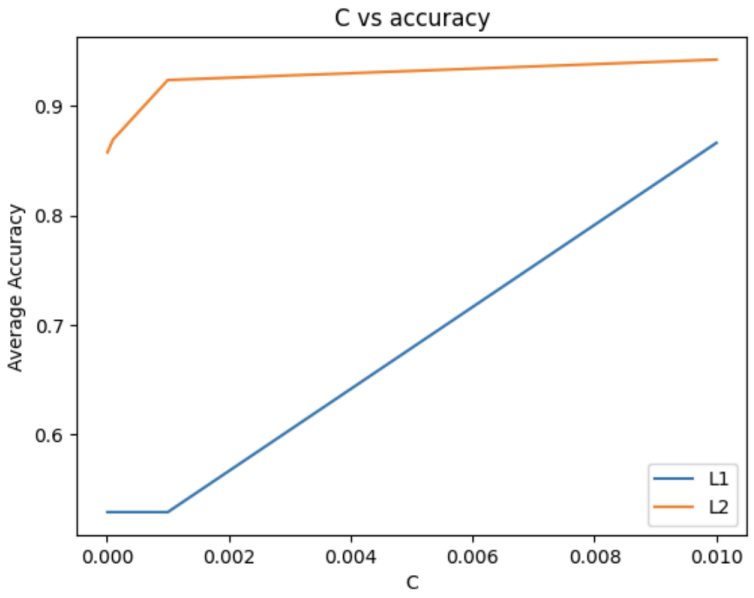
L2 Parameter: 1e-05 Average Cross Validation Accuracy: 0.8636690647482013
L2 Parameter: 0.0001 Average Cross Validation Accuracy: 0.8744604316546762
L2 Parameter: 0.001 Average Cross Validation Accuracy: 0.914388489208633
L2 Parameter: 0.01 Average Cross Validation Accuracy: 0.9410071942446043
L2 Parameter: 0.1 Average Cross Validation Accuracy: 0.943884892086331
L2 Parameter: 1 Average Cross Validation Accuracy: 0.9514388489208633
L2 Parameter: 10 Average Cross Validation Accuracy: 0.9557553956834532
L2 Parameter: 100 Average Cross Validation Accuracy: 0.9553956834532373
L2 Parameter: 1000 Average Cross Validation Accuracy: 0.9546762589928057
L2 Parameter: 10000 Average Cross Validation Accuracy: 0.9535971223021582
L2 Parameter: 100000 Average Cross Validation Accuracy: 0.9532374100719423

Best L1 Reg Parameter: 100
Best L2 Regularization Parameter: 10

L1 Accuracy: 0.9583333333333334
L1 Recall: 0.9573170731707317
L1 Precision: 0.9544072948328267
L1 F1-Score: 0.9558599695585996

L2 Accuracy: 0.9497126436781609
L2 Recall: 0.9420731707317073
L2 Precision: 0.9507692307692308
L2 F1-Score: 0.9464012251148545





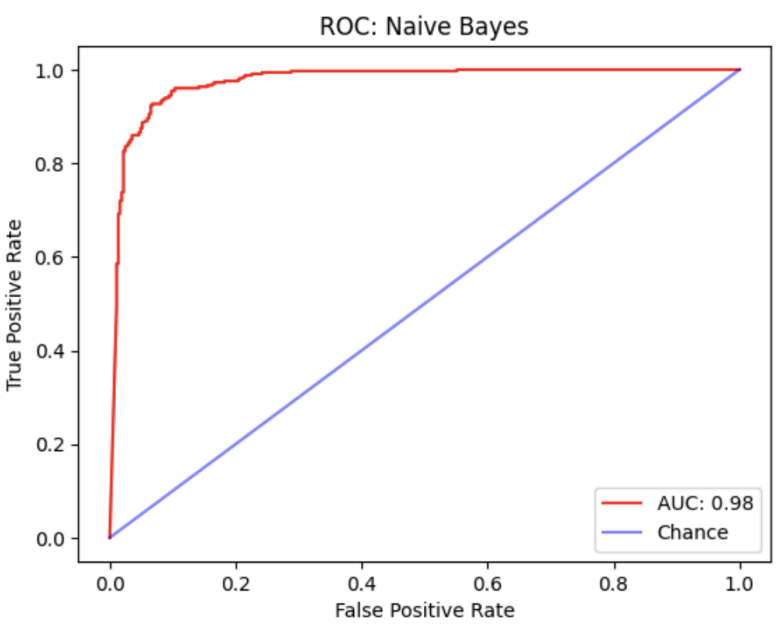
As regularization strength decreases, or as C increases, the coefficient average converges towards around -1. The value of C marginally affects the L2 regularization accuracy, but it does improve the L1 up to a similar accuracy when C > 0.010.

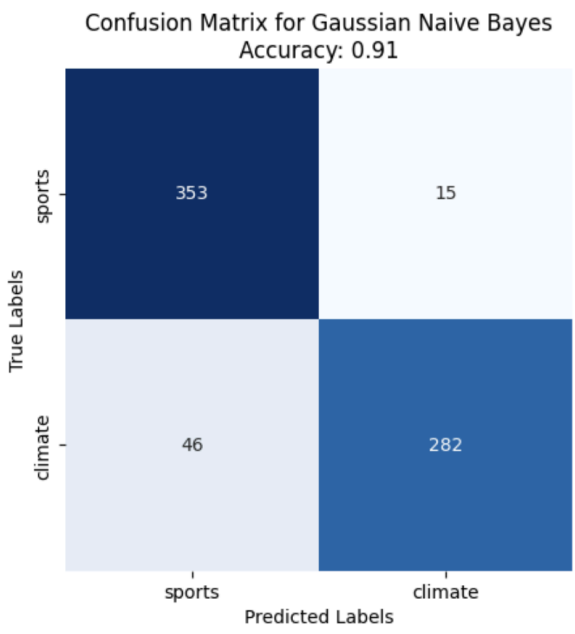
With models that don't need to worry about overfitting, no regularization is required. L1 regularization is suitable for feature selection and making simpler models. L2 regularization, which uses weights sampled from a Gaussian distribution, is used for reducing the effects of collinear features, which can lead to increased variance.

Logistic Regression and Support Vector Machines (SVM) Differences:

Logistic Regression uses logistic loss as its loss function, whereas linear SVMs use the hinge loss. Logistic Regression's decision boundary is chosen to maximize the likelihood of the observed data given the parameters of the model. It aims to model the probability that a given input belongs to a particular class. For SVMs, the decision boundary is chosen to maximize the margin between different classes. Logistic Regression outputs class probabilities, where linear SVMs output class labels directly based on the sign of the decision function. With performance, logistic regression is better when the classes are well-separated and the relationship between the features and the log-odds of the response is approximately linear. SVMs might perform better in situations where there is noise in the data or when the classes are not perfectly separable. The statistical significance of the performance difference would depend on the specific dataset, and would be seen through cross-validation or other evaluation metrics.

Question 7





Accuracy (Gaussian Naive Bayes): 0.9123563218390804
Recall (Gaussian Naive Bayes): 0.8597560975609756
Precision (Gaussian Naive Bayes): 0.9494949494949495
F1-Score (Gaussian Naive Bayes): 0.9024

Question 8

After running grid search on the specified hyperparameters, our top 5 parameters were as follows. The accuracies of testing the pipeline with the top parameters on the testing data are also included.

1)
   a) Feature Extraction: `min_df = 3` with lemmatization
   b) Dimensionality Reduction: `NMF(init='random', n_components=30, random_state=42)`
   c) Classifier: `SVC(C=10, random_state=42)`
   d) Test Set Accuracy: 0.950
2)
   a) Feature Extraction: `min_df = 5` with lemmatization
   b) Dimensionality Reduction: `NMF(init='random', n_components=30, random_state=42)`
   c) Classifier: `svm.SVC(C=10, random_state=42)`
   d) Test Set Accuracy: 0.958
3)

a) Feature Extraction: `min_df = 3` with stemmization
b) Dimensionality Reduction: `NMF(init='random', n_components=30, random_state=42)`
c) Classifier: `svm.SVC(C=10, random_state=42)`
d) Test Set Accuracy: 0.955

4)

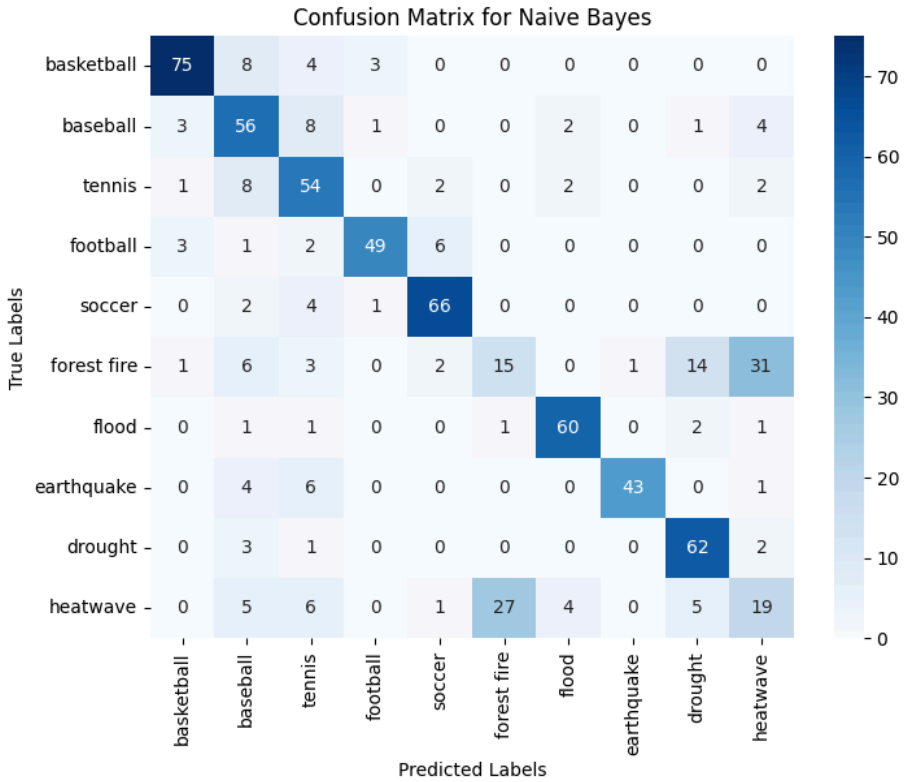
a) Feature Extraction: `min_df = 5` with stemmization
b) Dimensionality Reduction: `NMF(init='random', n_components=30, random_state=42)`
c) Classifier: `svm.SVC(C=10, random_state=42)`
d) Test Set Accuracy: 0.958

5)

a) Feature Extraction: `min_df = 3` with lemmatization
b) Dimensionality Reduction: `NMF(init='random', n_components=30, random_state=42)`
c) Classifier: `LogisticRegression(C=10, random_state=42)`
d) Test Set Accuracy: 0.934

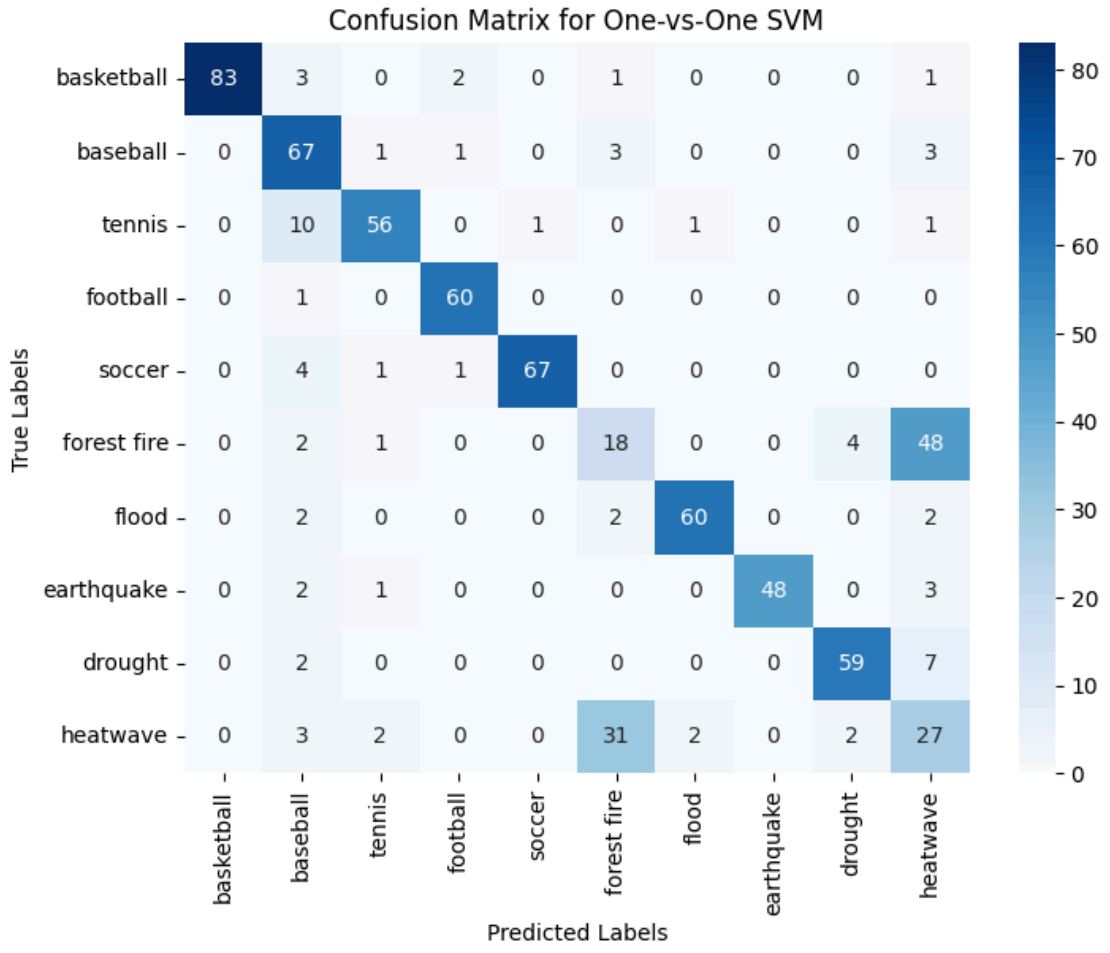
Question 9

The results of the multiclass classification for the three different classifiers are as follows.



Scores for multiclass Naive Bayes

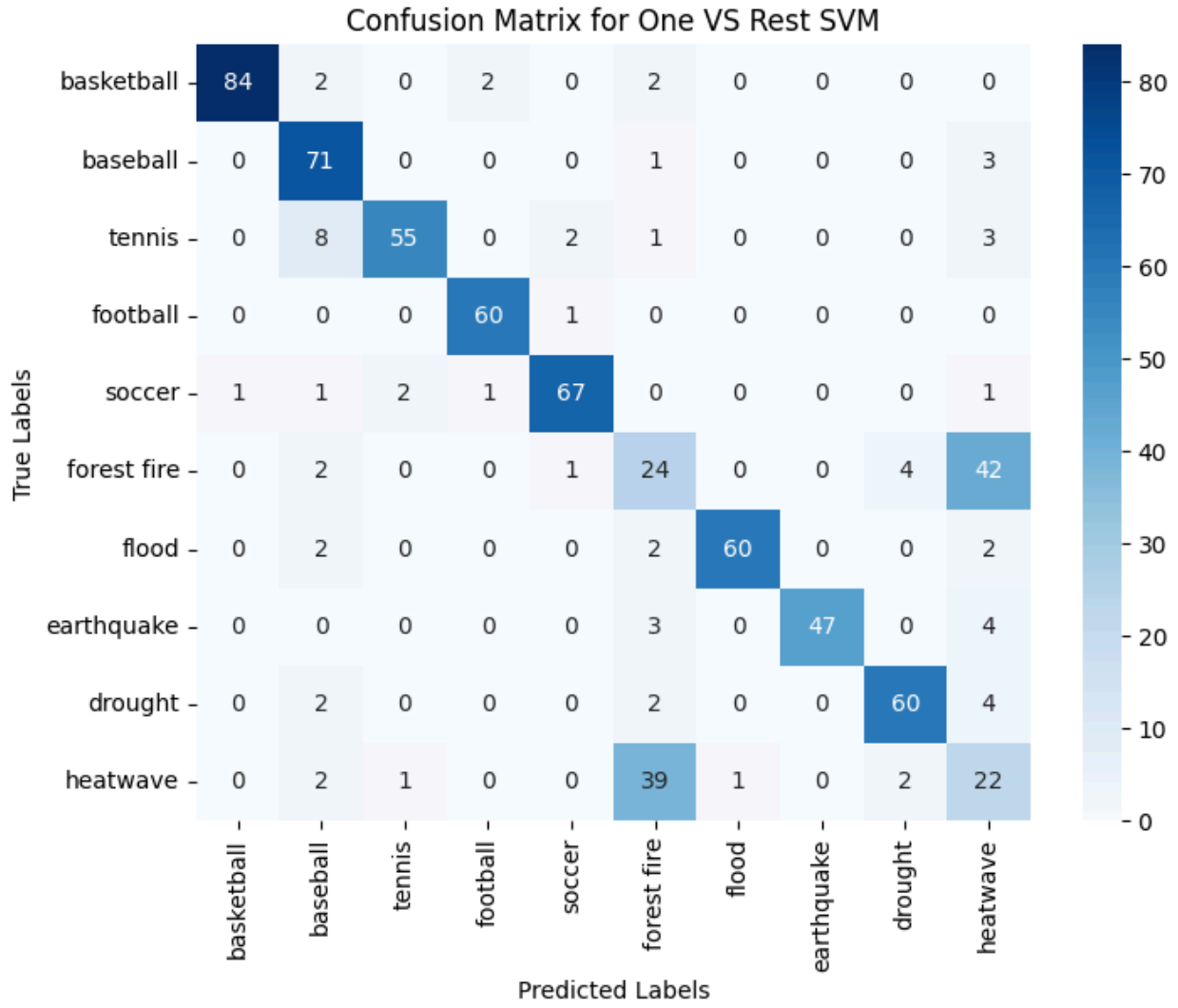
Accuracy	0.717
Recall	0.718
Precision	0.713
F1-Score	0.709



Scores for One-vs-One SVM

Accuracy	0.783
Recall	0.784
Precision	0.800
F1-Score	0.789

To resolve the class imbalance issue for the One-vs-Rest SVM, we used the “balanced” option for the class_weight argument in the linear SVM. This argument automatically assigns a higher class weight to the minority class inversely proportional to the class frequency, which mitigates class imbalance.



Scores for One-vs-Rest SVM

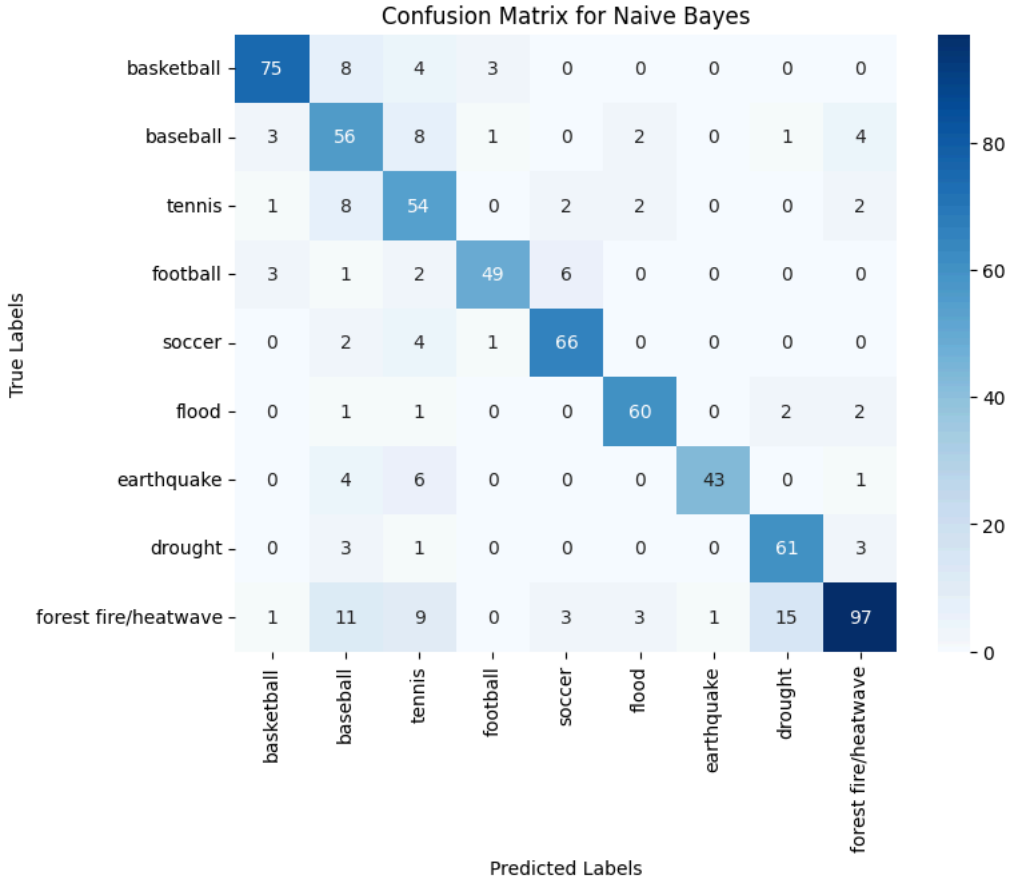
Accuracy	0.790
Recall	0.790
Precision	0.811
F1-Score	0.798

Merging classes

We observed that “forest fire” and “heatwave” seem to be in a distinct block on the diagonal. This means that the classifier often classifies “forest fire” articles as “heatwave” and vice-versa. This suggests that there is a lot of similarity in the word data for the two categories.

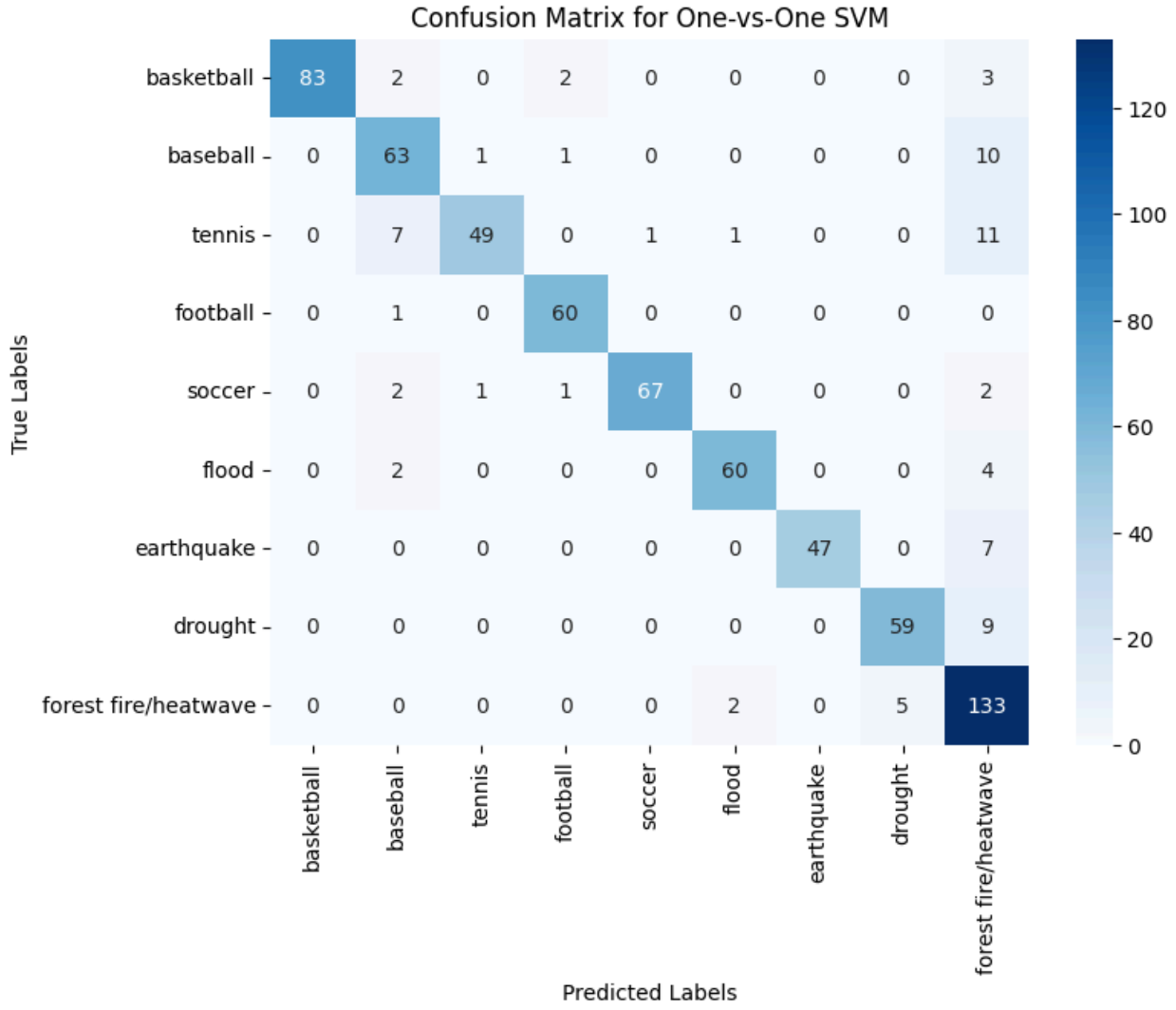
We can merge the two categories into a larger “forest fire/heatwave” class. This class makes sense, since forest fires are often caused by hot and dry weather conditions, which can occur during a heatwave.

The classifier results for the new categories are as follows.



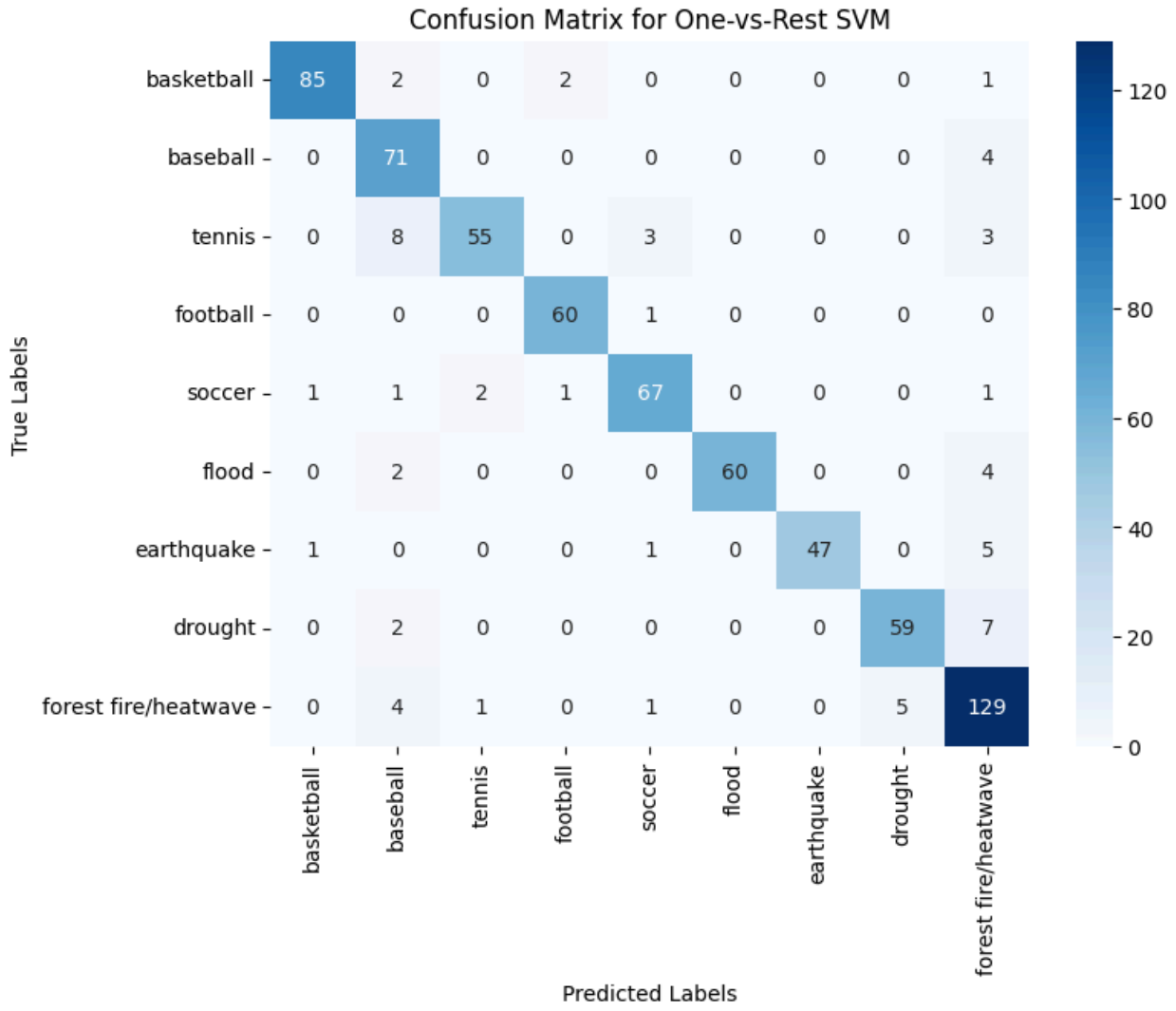
Scores for multiclass Naive Bayes after merging

Accuracy	0.806
Recall	0.818
Precision	0.823
F1-Score	0.815



Scores for One-vs-One SVM after merging

Accuracy	0.892
Recall	0.886
Precision	0.924
F1-Score	0.900



Scores for One-vs-Rest SVM after merging

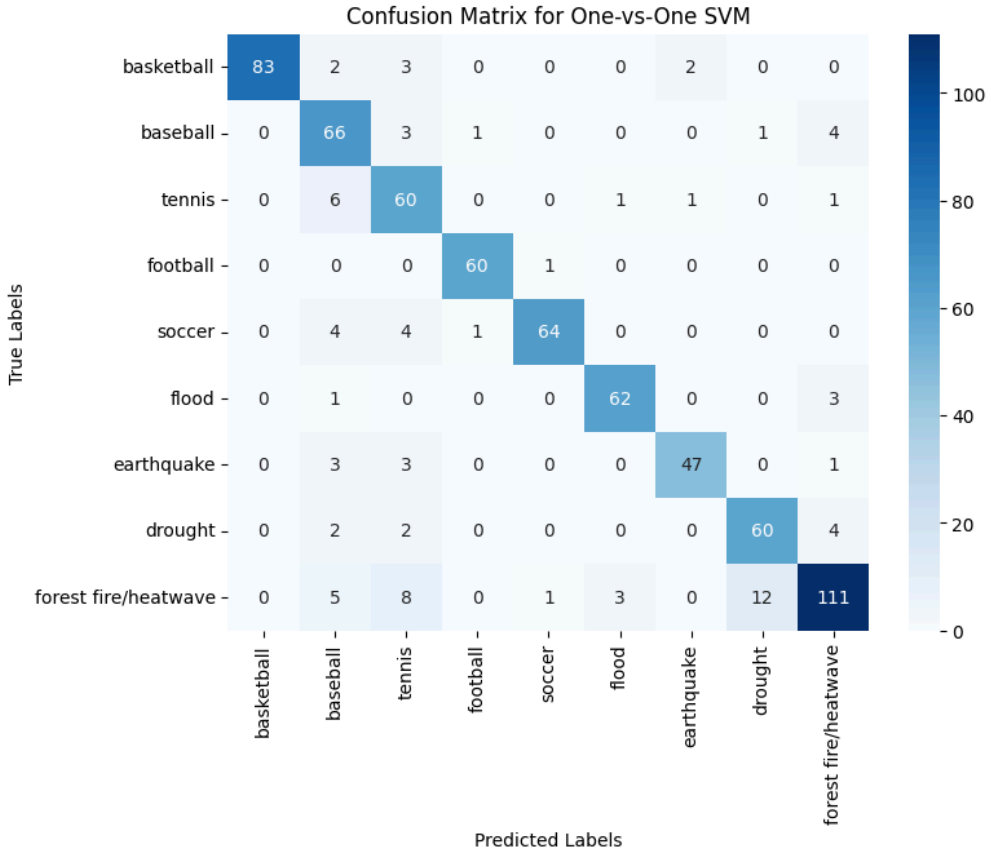
Accuracy	0.909
Recall	0.906
Precision	0.927
F1-Score	0.914

Dealing with imbalance

The accuracy for both One-vs-One and One-vs-Rest increased once the labels were merged. This is likely because of a decrease in misclassifications between “forest fire” and “heatwave” from the previous classifiers.

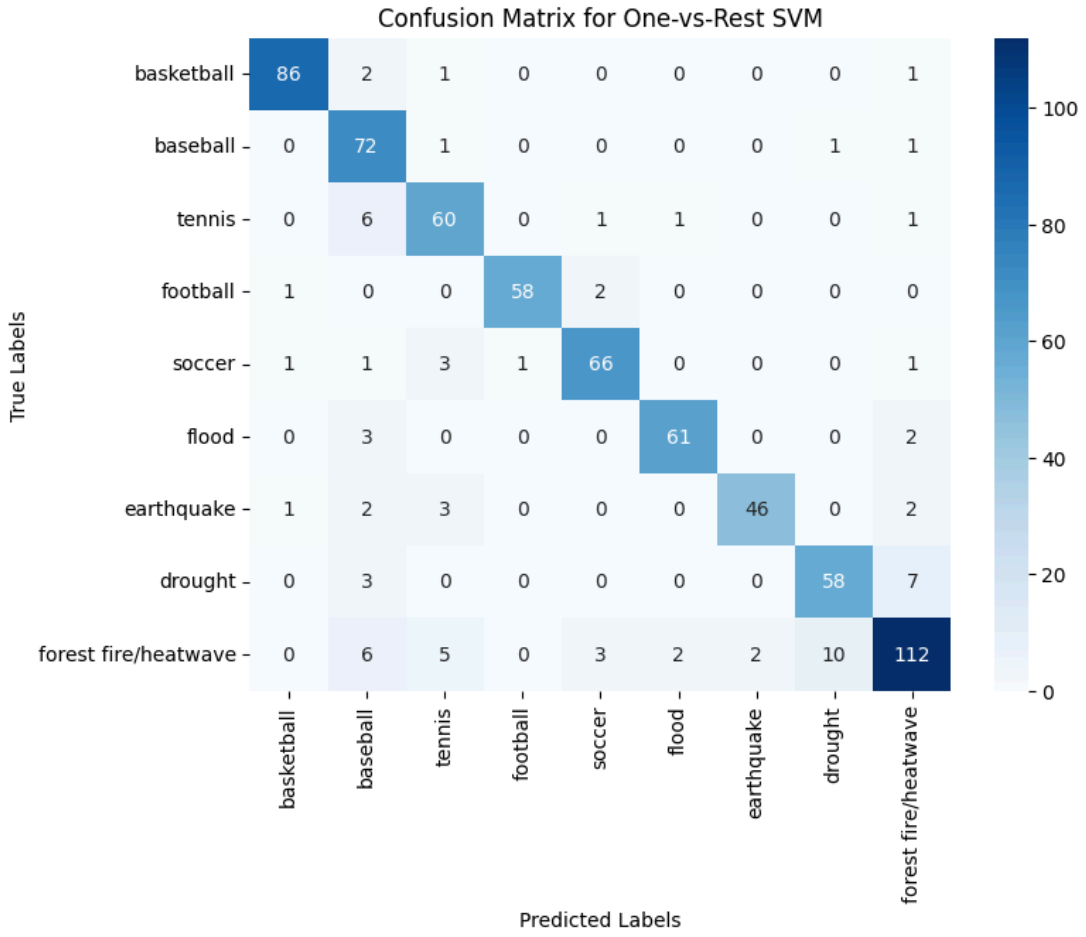
While accuracy has increased, there is class imbalance in the models, as there are twice as many samples for “forest fire/heatwave” compared to the other categories. This causes the model to be biased towards classifying samples into this category. This can be seen by how, for example, the One-to-One model is more likely to misclassify samples as “forest fire/heatwave”, even those from unrelated categories like “basketball” or “tennis”.

We can solve the class imbalance by downsampling the “forest fire/heatwave” category, randomly selecting half of the samples in the original data. By decreasing the data by half, we make the number of samples roughly equal to the other categories.The results for the One-vs-One and One-vs-Rest classifiers after downsampling are as follows.



Scores for One-vs-One SVM after downsampling

Accuracy	0.881
Recall	0.891
Precision	0.889
F1-Score	0.887



Scores for One-vs-Rest SVM after downsampling

Accuracy	0.889
Recall	0.897
Precision	0.898
F1-Score	0.895

Interestingly, the accuracy has gone down slightly after dealing with class imbalance. This is likely because there is also imbalance in the testing dataset, allowing the unbalanced classifiers to have a higher accuracy even while being biased. With a more balanced testing dataset, the new classifiers will likely perform better.

Question 10

(a) The ratio of co-occurrence probabilities helps to identify probe words that are closely related to one word of interest but not the other–these will either have a very large or very small ratio. For words that are related to both or none of the words, they will have a ratio close to 1, signifying that the probe word is not useful for differentiating the two words of interest. Using only the probabilities themselves, the raw value of the probability does not capture this information.

(b) Both words will return the same vector. This is because GLoVE embeddings are trained on the entire corpus, and each distinct word will have a single vector representing them based on information learned from the entire text.

(c) We would expect $|| GLoVE["woman"] - GLoVE["man"] ||_2$ to be close to the value of $|| GLoVE["wife"] - GLoVE["husband"] ||_2$, while the value of $|| GLoVE["wife"] - GLoVE["orange"] ||_2$ is likely to be larger. This is because "woman" - "man" and "wife" - "husband" should both represent the semantic difference between the concepts of "female" and "male", while the other characteristics shared between "woman" and "man" (e.g. human adults), or "wife" and "husband" (e.g. being married), should cancel out. On the other hand, since "wife" and "orange" have very little relation to each other, the vector difference should have a larger magnitude.

After computing the actual values, we find that

$$|| GLoVE["woman"] - GLoVE["man"] ||_2 = 4.75,$$

$$|| GLoVE["wife"] - GLoVE["husband"] ||_2 = 3.15,$$

$$|| GLoVE["wife"] - GLoVE["orange"] ||_2 = 8.67$$

The first two values are not equal, indicating that there is still some difference between the two pairs of words after subtraction. At the same time, the difference between "wife" and "orange" is larger than the first two values, which is as predicted.

(d) We would rather use lemmatization before mapping to GLoVE embeddings. Lemmatization is more accurate at reducing words to their base form, while stemming might result in nonsensical word fragments or different words with the same base word e.g. "is" and "be". Since GLoVE is trained on an entire corpus and tries to capture information about words in the context of others, having words that have no meaning or words not reduced to their base form in the dataset might make the embedding of other words inaccurate.

Question 11

(a) A possible feature engineering process is as follows

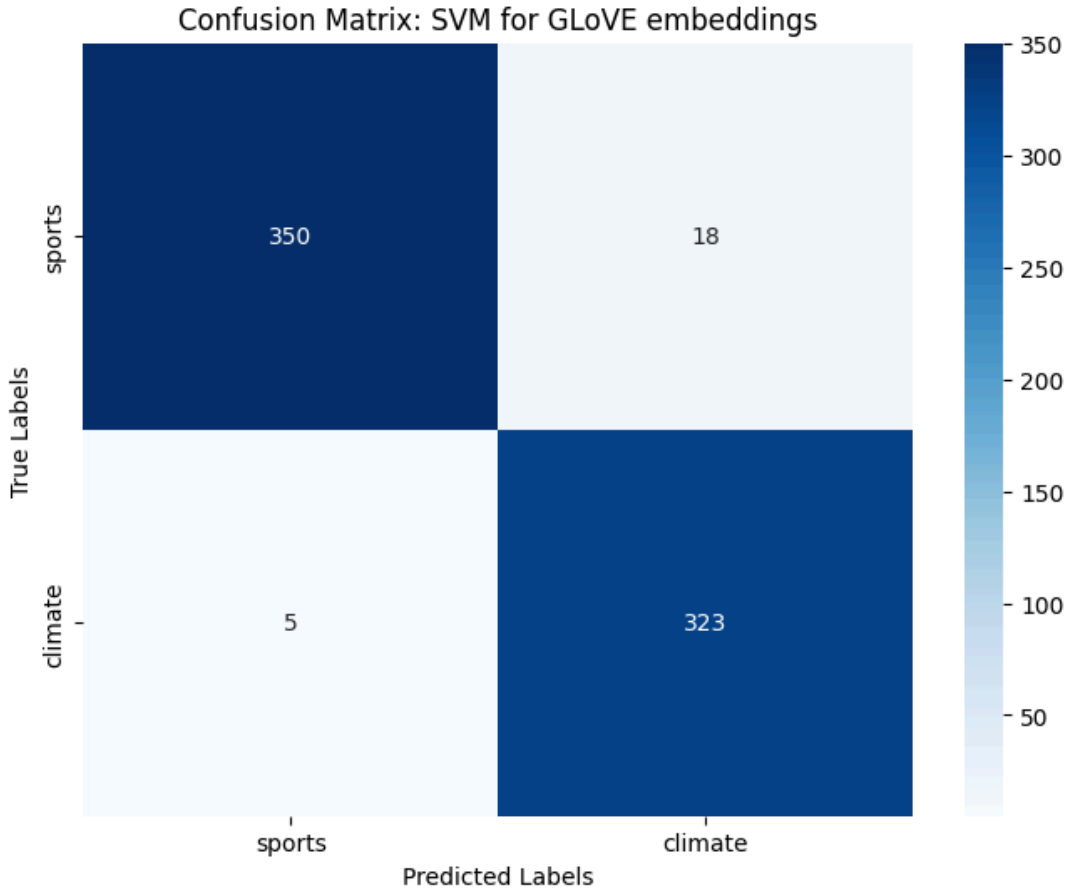
(i) Perform a train-test split, setting the “full_text” column as X and the “root_label” column as y

(ii) Clean the X data of any HTML artifacts

(iii) Split each document in X into individual words, and lemmatize each word

(iv) For each document, compute the GLoVE embedding of every word in the document, add them all together by vector addition, and divide them by the total number of words. If there are words that do not appear in the GLoVE data, ignore them. The result is a single vector representing each document, which has the same dimensions as the GLoVE embedding for a word.

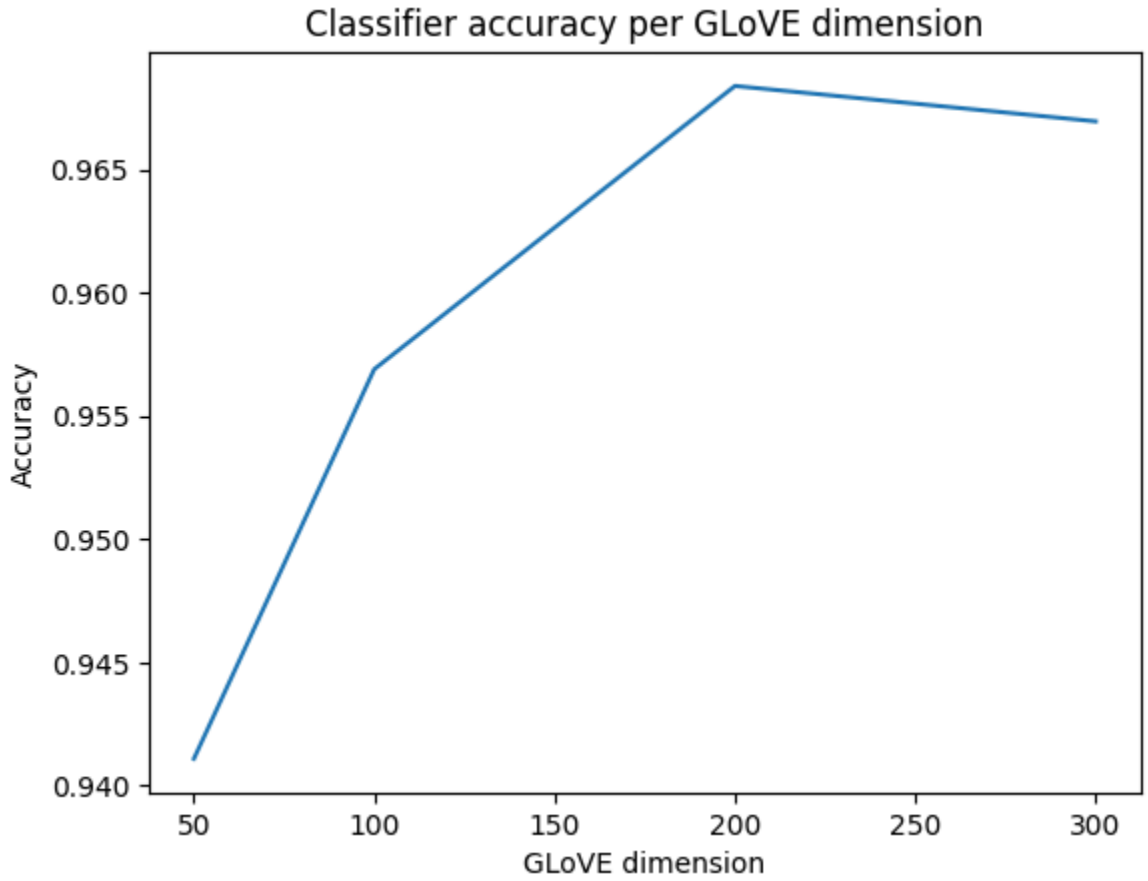
(b) We trained a linear SVM on the dataset after feature engineering, using GridSearch to find the ideal Gamma. The results of the classification are as follows



Accuracy: 0.967
Precision: 0.986
F1 Score: 0.968
Recall: 0.951

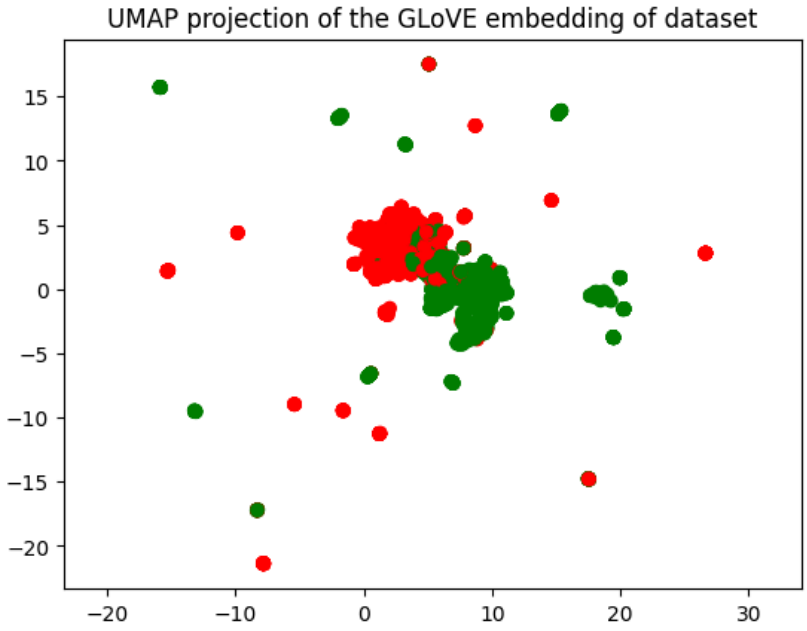
The high accuracy indicates that the classifier is able to perform well on the GLoVE embeddings of the documents.

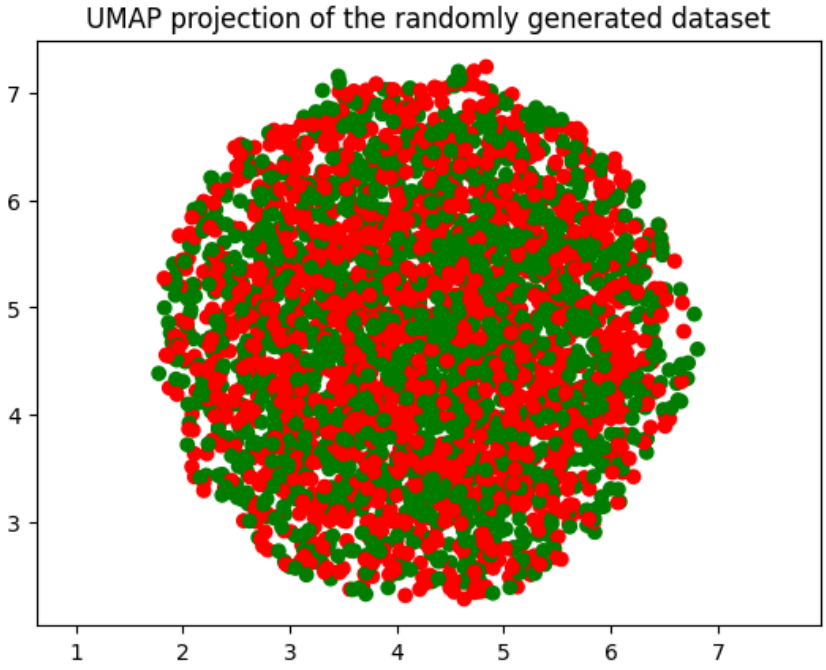
Question 12



The accuracy of the model increases with GLoVE dimension until 200, after which it decreases. This trend is expected, because, in general, increasing dimension should allow each individual word vector to encode more information, thus making it easier to differentiate between documents. However, past a certain point, there is likely to be overfitting due to too many dimensions, which causes performance to decrease.

Question 13





For the UMAP visualization of the GloVE embedding of the original dataset, most of the data points are in 2 distinct clusters, corresponding to the topic of the respective document (red is sports, green is climate). On the other hand, the UMAP visualization of the randomly generated data does not show any distinct clustering, and data from both labels seems to be uniformly distributed around the center.