Project 1: End to End Pipeline to Classify News Articles

Lauren Mizner (#005225768)

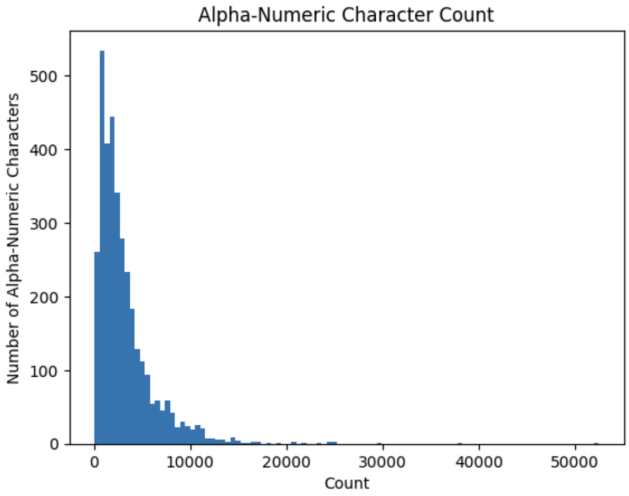
Question 1

How many rows (samples) and columns (features) are present in the dataset?

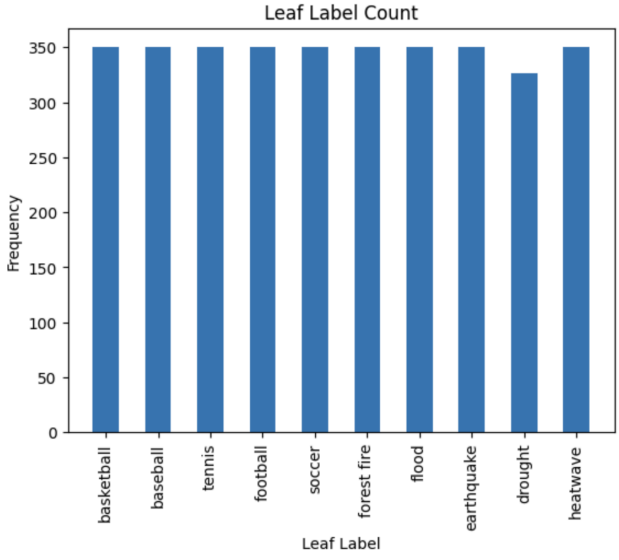
Number of rows/samples = 3476
Number of columns/features = 8

Plot 3 Histograms:

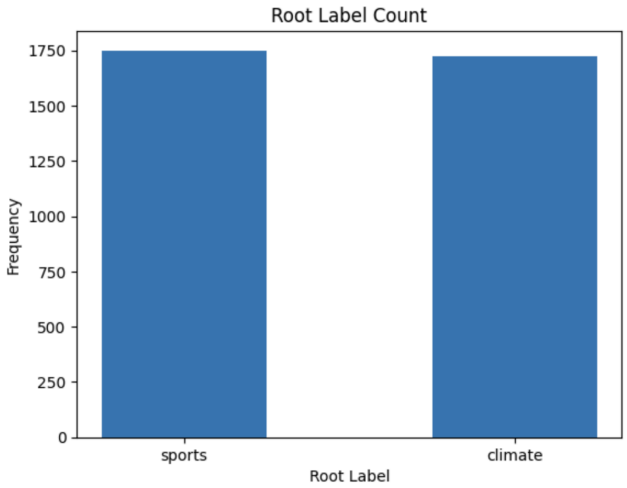
(a) The total number of alpha-numeric characters per data point (row) in the feature full_text (i.e. count on the x-axis and frequency on the y-axis)



(b) The column leaf_label - class on the x-axis



(c) The column root_label - class on x-axis



Interpret Plots: Provide qualitative interpretations of the histograms

From the root label count and leaf label count, there is an even distribution of each label type across the dataset, with the exception of the leaf label, drought, which has a slightly lower count than any of the other leaf labels. Overall, the even distribution of labels is desirable for classification tasks as this helps mitigate any model overfitting on any given label class with a higher number of data samples than others.

Question 2
Report the number of training and testing samples.

Number of training samples = 2780
Number of testing samples = 696

Question 3
What are the pros and cons of lemmatization versus stemming? How do these processes affect the dictionary size?

Stemming tends to be faster than lemmatization since it applies simpler rules and since stemming tends to result in a smaller dictionary, it is more space/memory efficient. Stemming does run into the issue of overstemming or understemming, making lemmatization more accurate compared to stemming.

Lemmatization reduces words to their canonical form, so it doesn't significantly affect the dictionary size compared to the original text, while stemming reduces words to their root form by removing prefixes and suffixes, meaning multiple related words can collapse into the same stem, resulting in a smaller dictionary size compared to the original text.

How does varying min.df change TF-IDF matrix?

min.df is used to specify the minimum number of texts that the word must appear in, in order to be included. A larger min.df value results in a smaller dictionary size, as less words would satisfy the frequency requirement, resulting in a TF-IDF matrix with fewer columns/features. On

the other hand, a smaller min.df value results in a larger dictionary size, as more words would satisfy the frequency requirement, resulting in a TF-IDF matrix with more columns/features.

Should I remove stop words before or after lemmatizing? Should I remove punctuations before or after lemmatizing? Hint: Recall that the full sentence is input into the Lemmatizer and the lemmatizer is tagging the position of every word based on the sentence structure.

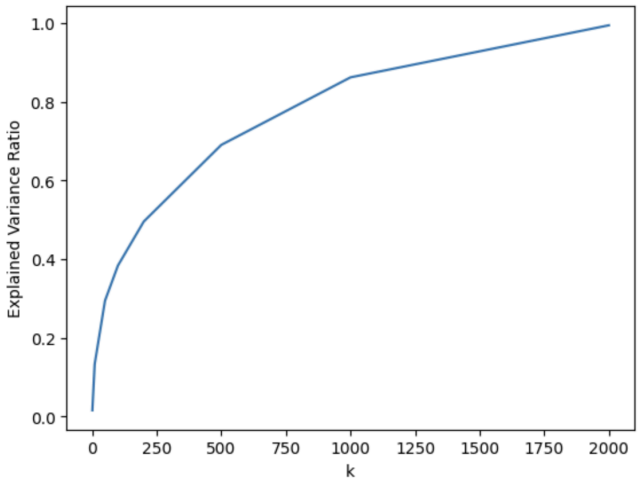
Stop words, punctuation, and numbers are all removed before lemmatizing within the data cleaning process.

Report the shape of the TF-IDF processed train and test matrices. The number of rows should match the results of Question 2. The number of columns should roughly be in the order of k x 10^3. This dimension will vary depending on your exact method of cleaning and lemmatizing and that is okay.

Shape of TF-IDF processed training matrix = (2780, 13413)
Shape of TF-IDF processed testing matrix = (696, 13413)

Question 4

Plot the explained variance ratio across multiple different k = [1, 10, 50, 100, 200, 500, 1000, 2000] for LSI and for the next few sections choose k = 50. What does the explained variance ratio plot look like? What does the plot's concavity suggest?



The plot suggests that as k increases, the rate of increase of the explained variance ratio decreases (i.e. the slope decreases as k increases). Initially adding more components, k, results in a significant increase in the explained variance ratio. As k gets significantly larger, adding more components, k, contributes less to the overall explained variance ratio. So as k gets larger we're capturing less essential information.

With k = 50 found in the previous sections, calculate the reconstruction residual MSE error when using LSI and NMF - they both should use the same k = 50. Which one is larger, the NMF or the LSI and why?

NMF Mean Squared Error = 4.595e-05
LSI Mean Squared Error = 7.360e-05

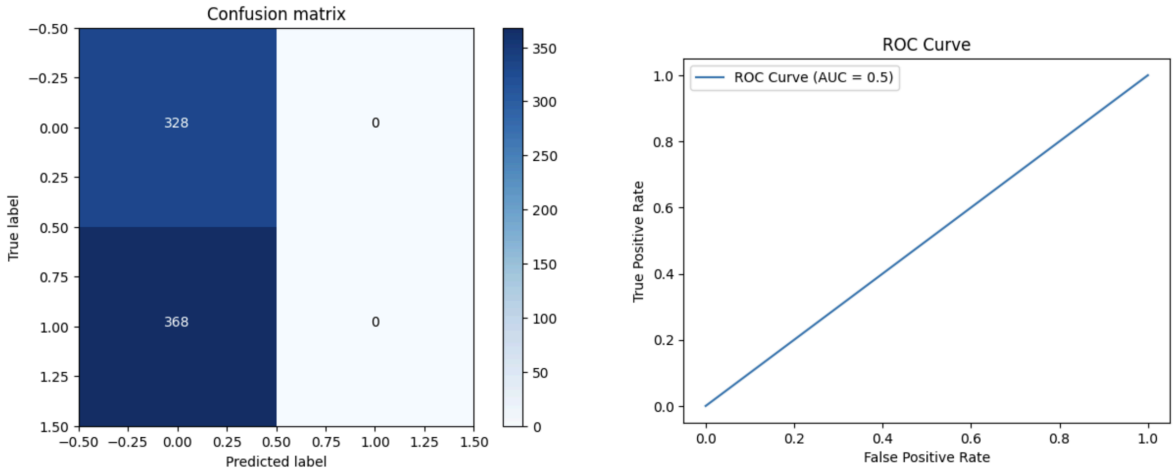
The results for NMF mean square error and LSI mean square error are very close in value. Typically, NMF tends to have a higher mean squared error than LSI , due to NMF enforcing non-negativity constraints on the vectors and coefficients.

Question 5

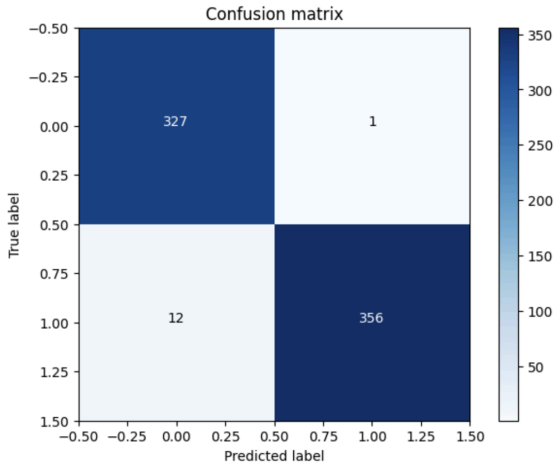
Train one SVM with γ = 1000 (hard margin), another with γ = 0.0001 (soft margin). Plot the ROC curve, report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of both SVM classifiers on the testing set. Which one performs better? What about for γ = 100000?

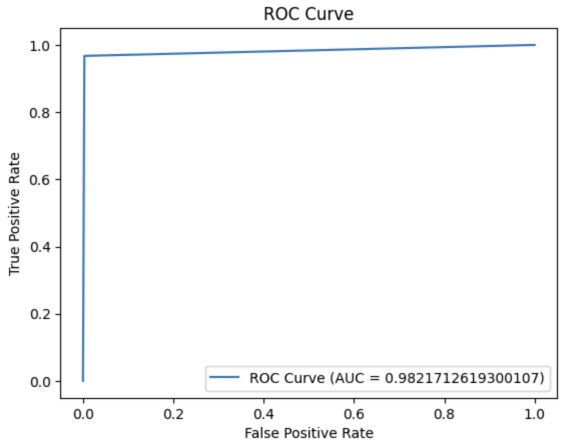
	Accuracy	Precision	Recall	F1-Score
γ = 0.0001	0.471	1.0	0.0	0.0
γ = 1000	0.981	0.997	0.967	0.982
γ = 100000	0.981	0.997	0.967	0.982

γ = 0.0001 (Soft Margin)

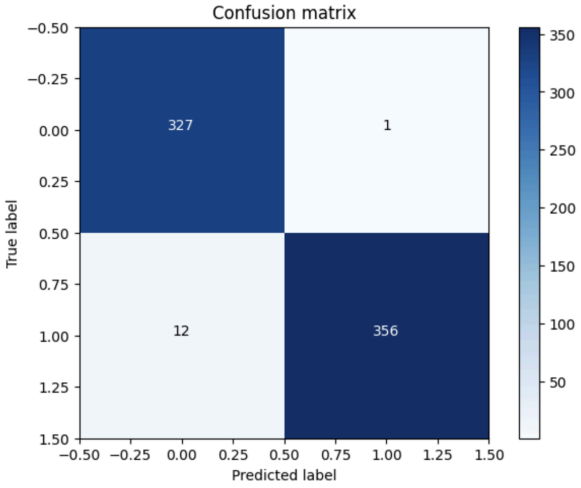


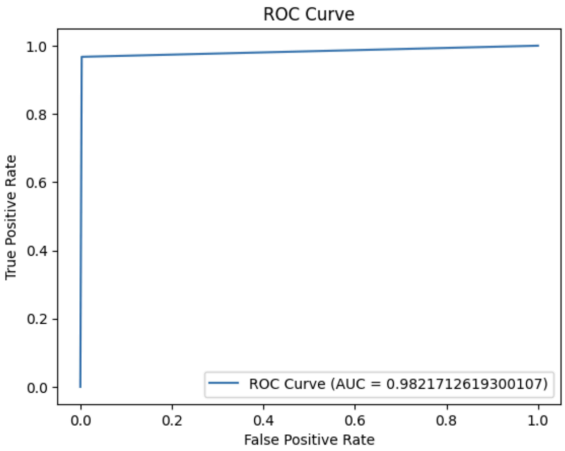
γ = 1000 (Hard Margin)





γ = 100000 (Highest Margin)





Between the hard margin of γ = 1000 and the soft margin of γ = 0.0001, the hard margin SVM classifier performs significantly better with an accuracy of 0.981. The highest margin of γ = 100000, performs on par with the hard margin SVM classifier, obtaining the same performance metrics.

What happens for the soft margin SVM? Why is this the case? Analyze in terms of the confusion matrix.

The soft margin SVM has a γ = 0.0001 and produces a confusion matrix with a true positive count of 328 and false negative count of 368. This indicates that the classifier is not performing well as it's misclassifying a significant number of instances from the positive class as negative.

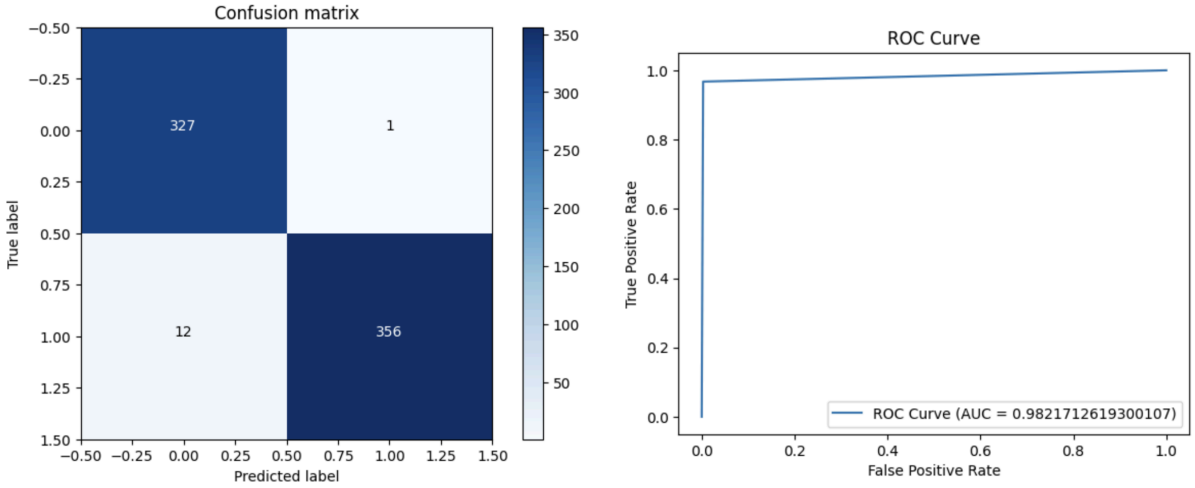
Does the ROC curve reflect the performance of the soft-margin SVM? Why?

The soft margin has a γ = 0.0001 and results in an accuracy of 0.471, which indicates that the classifier's performance is no better than random guessing. This aligns with the results we are seeing with the confusion matrix as well.

Use cross-validation to choose γ (use average validation accuracy to compare). Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10k \mid -3 \leq k \leq 6, k \in Z\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision, and F-1 score of this best SVM.

	Accuracy	Precision	Recall	F1-Score
γ = 10	0.981	0.997	0.967	0.982

γ = 10 (Best Margin)

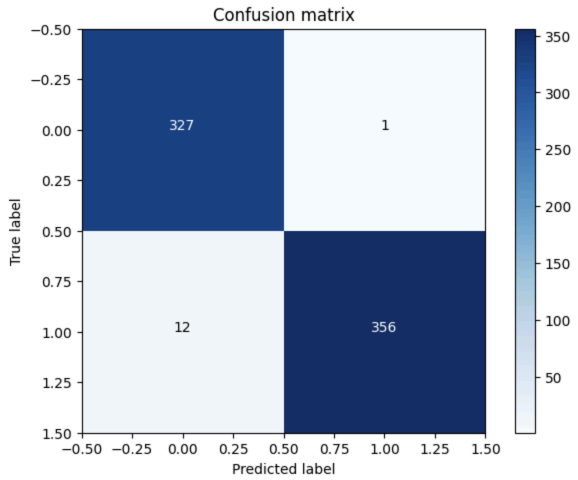


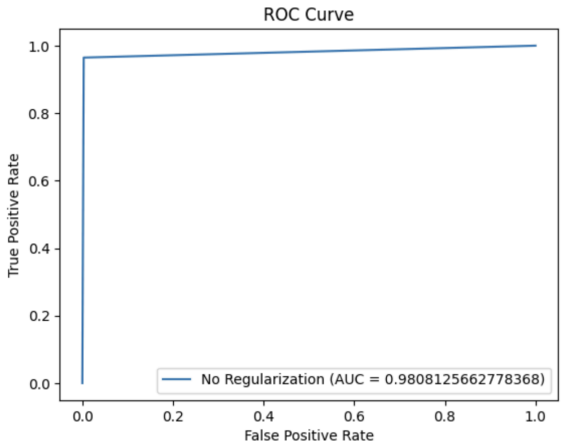
Question 6

Train a logistic classifier without regularization. Plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision, and F-1 score of this classifier on the testing set.

	Accuracy	Precision	Recall	F1-Score
L0	0.979	0.997	0.964	0.981

L0

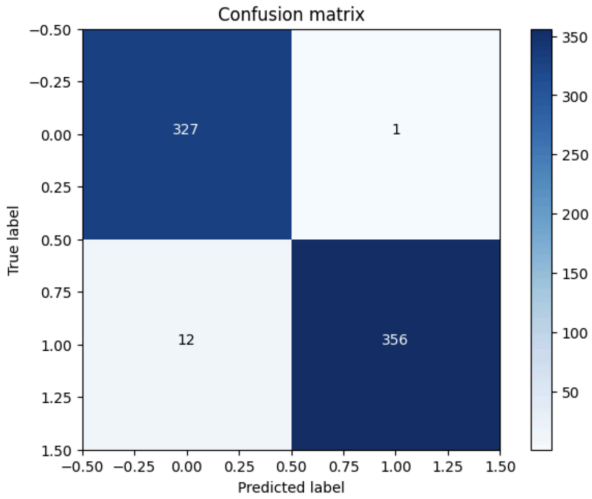


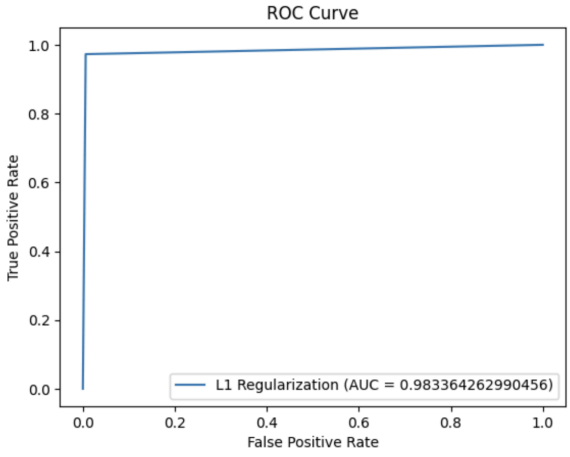


Using 5-fold cross-validation on the dimension-reduced-by-SVD training data, find the optimal regularization strength in the range $\{10k | -5 \leq k \leq 5, k \in Z\}$ for logistic regression with L1 regularization and logistic regression with L2 regularization, respectively.

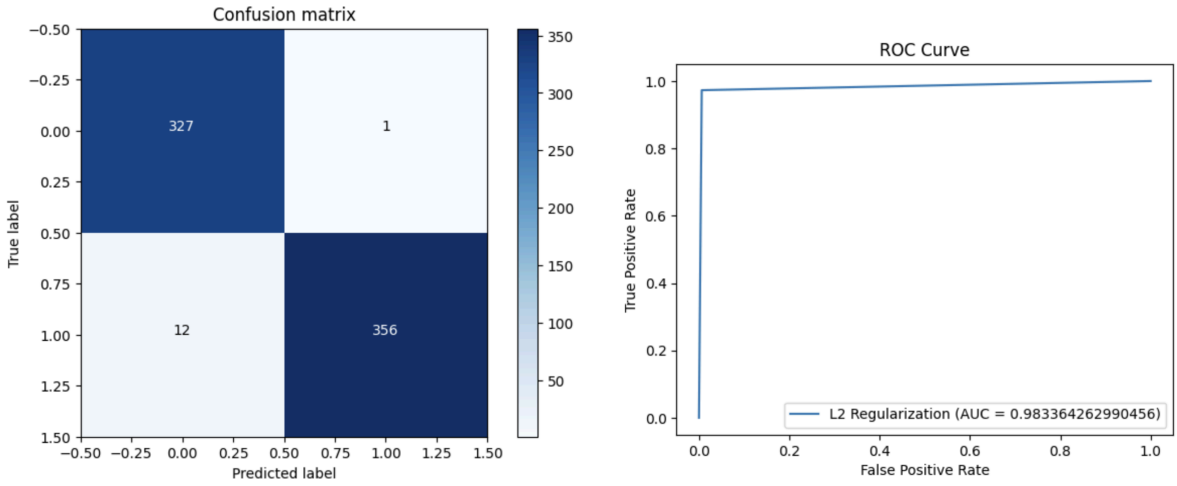
	Accuracy	Precision	Recall	F1-Score
L1 (100.0)	0.983	0.994	0.973	0.984
L2 (10.0)	0.983	0.994	0.973	0.984

L1





L2



Compare the performance (accuracy, precision, recall, and F-1 score) of 3 logistic classifiers: w/o regularization, w/ L1 regularization, and w/ L2 regularization (with the best parameters you found from the part above), using test data.

The best accuracy, recall, and F1-score occur with the logistic classifier with L1 and L2 regularization, with values of 0.983, 0.973, and 0.984, respectively. The best precision occurs with the logistic classifier without regularization with a value of 0.997.

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

A small regularization parameter, C, allows a model to fit to the training data more closely, potentially leading to overfitting, meaning that it will have poor generalization to unseen data and a higher test error. It also tends to assign larger coefficients to features in this case, again, leading to overfitting.

A large regularization parameter, C, allows the model to penalize more heavily for large coefficients, leading to smaller coefficient values. This helps mitigate overfitting, by creating simple models that generalize better to unseen data and produce a lower test error.

L1 Regularization penalizes the absolute values of the coefficients. This leads to sparsity in the model and tends to force some coefficients to zero. This is useful when dealing with high-dimensional data with many unimportant features, to select the important features and dispose of the rest.

L2 Regularization penalizes the squared magnitudes of the coefficients. This leads to smaller, but non-zero coefficients. This is useful when all features are potentially important, to shrink some feature coefficients without setting them to zero and disposing of them.

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?

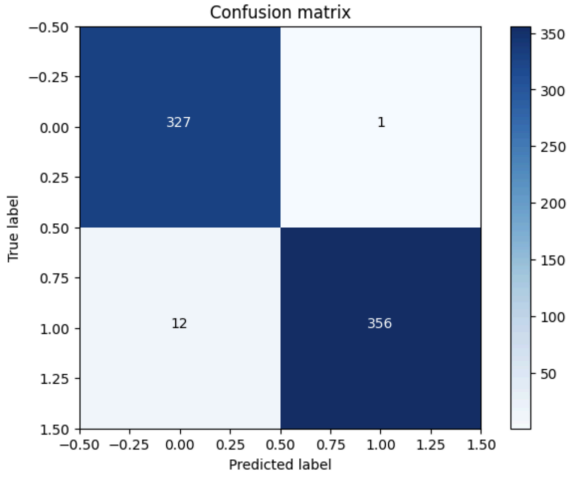
Logistic regression uses cross entropy loss as the loss function and works by modeling the probability that a data point belongs to a certain class and finds the best decision boundary that separates the classes based on these probabilities.

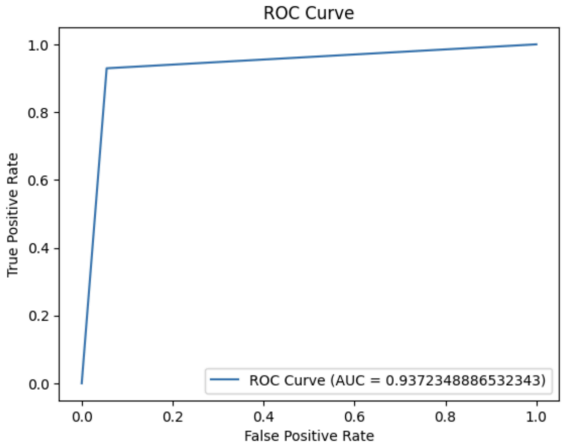
Linear SVM uses high loss as the loss function and works by finding the decision boundary that maximally separates the classes and penalizing points that fall on the wrong side of the margin.

Question 7

Evaluate and profile a Naive Bayes classifier. Train a GaussianNB classifier, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision, and F-1 score of this classifier on the testing set.

	Accuracy	Precision	Recall	F1-Score
NB	0.937	0.950	0.929	0.940





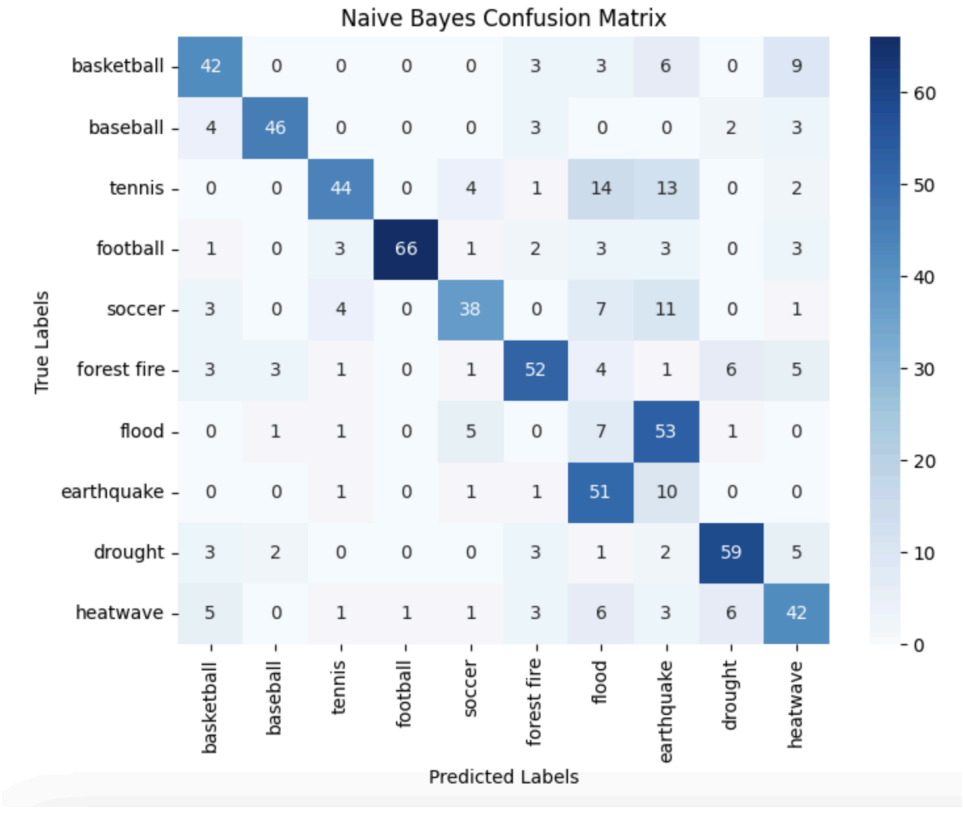
Question 8

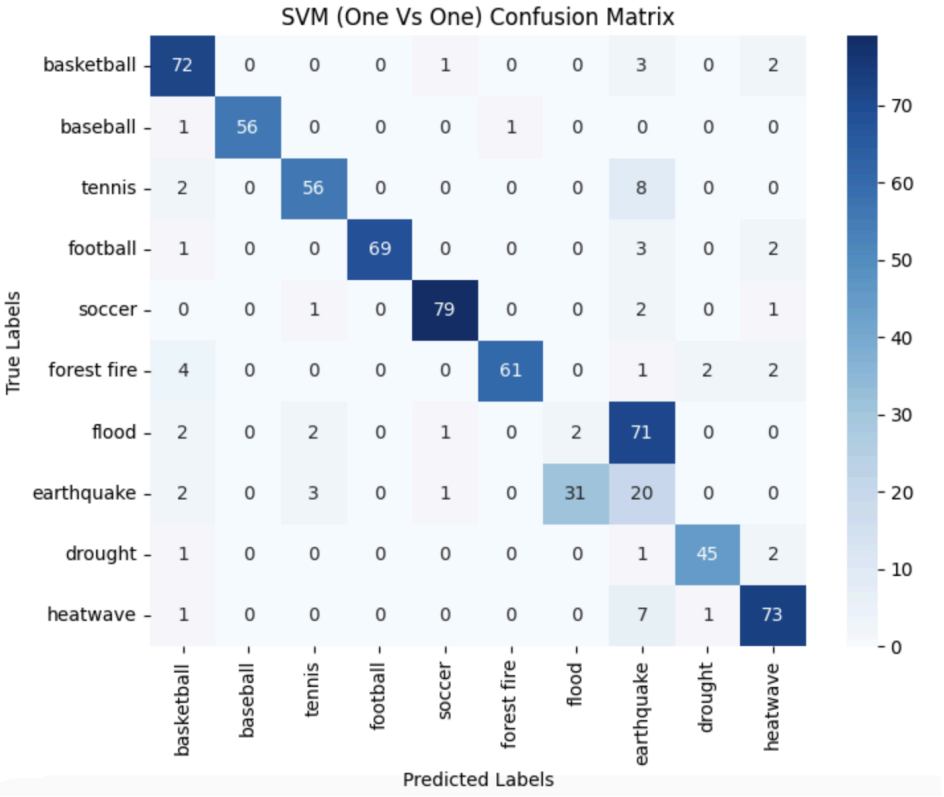
What are the best 5 combinations? Report their performances on the testing set.

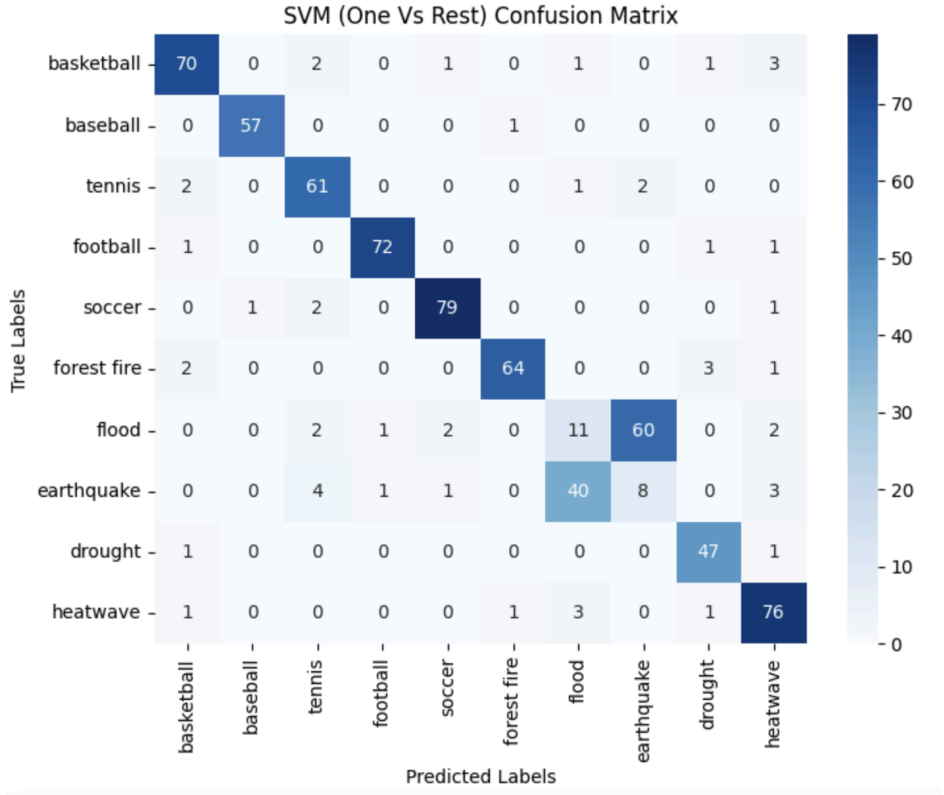
Best combinations:
Combination 1: {'svc__C': 0.1, 'svc__kernel': 'linear', 'svd__n_components': 2}
Combination 2: {'svc__C': 0.1, 'svc__kernel': 'linear', 'svd__n_components': 3}
Combination 3: {'svc__C': 0.1, 'svc__kernel': 'linear', 'svd__n_components': 4}
Combination 4: {'svc__C': 0.1, 'svc__kernel': 'rbf', 'svd__n_components': 2}
Combination 5: {'svc__C': 0.1, 'svc__kernel': 'rbf', 'svd__n_components': 3}
Test set accuracy for the best combination: 0.9310344827586207

Question 9

Perform Naive Bayes classification and multiclass SVM classification (with both One VS One and One VS the Rest methods) and report the confusion matrix and calculate the accuracy, recall, precision, and F-1 score of your classifiers.




SVM (One Vs One) Confusion Matrix
basketball 72 0 0 0 1 0 0 3 0 2
baseball 1 56 0 0 0 1 0 0 0 0
tennis 2 0 56 0 0 0 0 8 0 0
football 1 0 0 69 0 0 0 3 0 2
soccer 0 0 1 0 79 0 0 2 0 1
forest fire 4 0 0 0 0 61 0 1 2 2
flood 2 0 2 0 1 0 2 71 0 0
earthquake 2 0 3 0 1 0 31 20 0 0
drought 1 0 0 0 0 0 0 1 45 2
heatwave 1 0 0 0 0 0 0 7 1 73
basketball
baseball
tennis
football
soccer
forest fire
flood
earthquake
drought
heatwave
True Labels
Predicted Labels
70
60
50
40
30
20
10
0


SVM (One Vs Rest) Confusion Matrix
basketball 70 0 2 0 1 0 1 0 1 3
baseball 0 57 0 0 0 1 0 0 0 0
tennis 2 0 61 0 0 0 1 2 0 0
football 1 0 0 72 0 0 0 0 1 1
soccer 0 1 2 0 79 0 0 0 0 1
forest fire 2 0 0 0 0 64 0 0 3 1
flood 0 0 2 1 2 0 11 60 0 2
earthquake 0 0 4 1 1 0 40 8 0 3
drought 1 0 0 0 0 0 0 0 47 1
heatwave 1 0 0 0 0 1 3 0 1 76
basketball
baseball
tennis
football
soccer
forest fire
flood
earthquake
drought
heatwave
True Labels
Predicted Labels
70
60
50
40
30
20
10
0

	Accuracy	Precision	Recall	F1-Score
NB	0.583	0.654	0.583	0.613
SVM (OvO)	0.766	0.776	0.766	0.767
SVM (OvR)	0.783	0.777	0.783	0.779

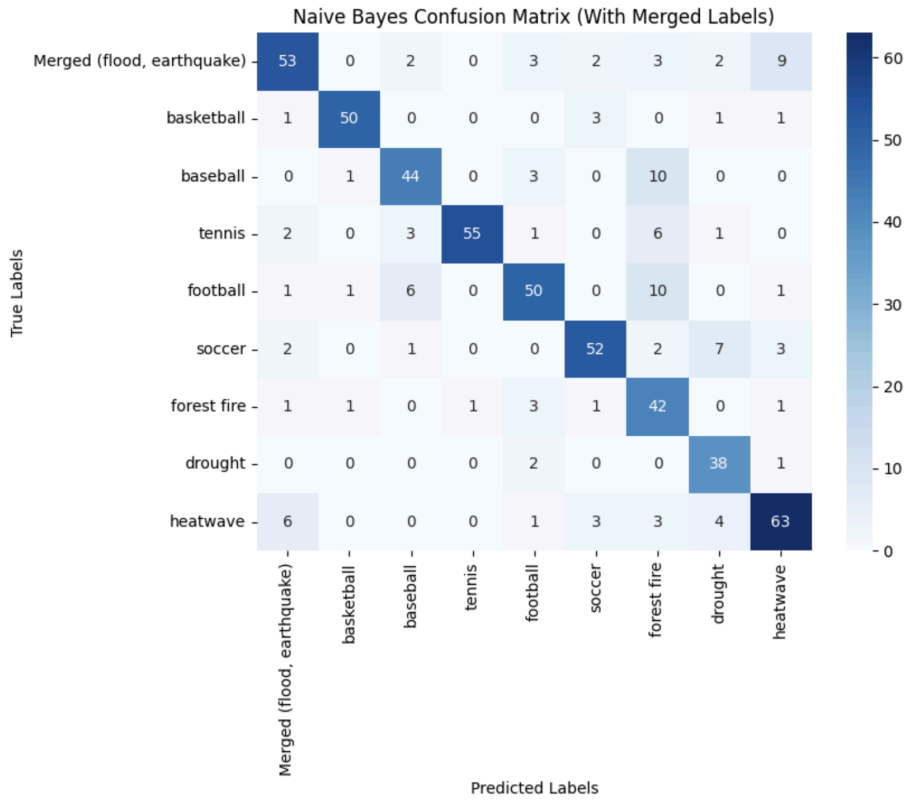
Do you observe any structure in the confusion matrix? Are there distinct visible blocks on the major diagonal? What does this mean?

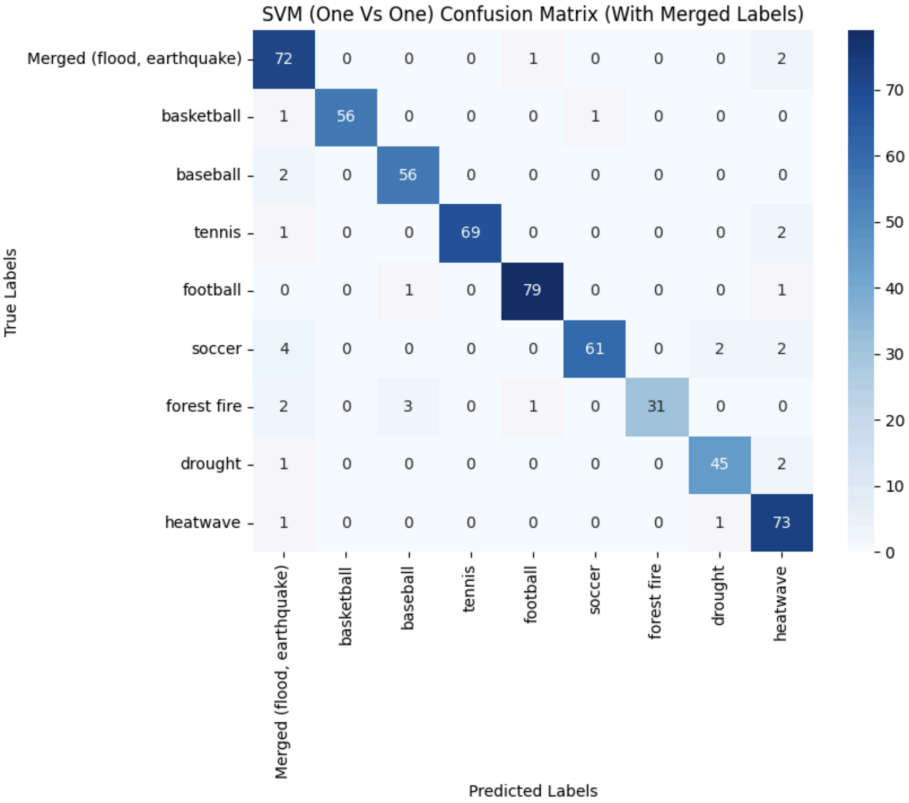
The diagonal of the confusion matrix represents the correctly classified instances for each class, while those off-diagonal represent misclassifications. From the results, we're seeing correct classification among most classes with the exception of flood and earthquake, which are being misclassified for each other.

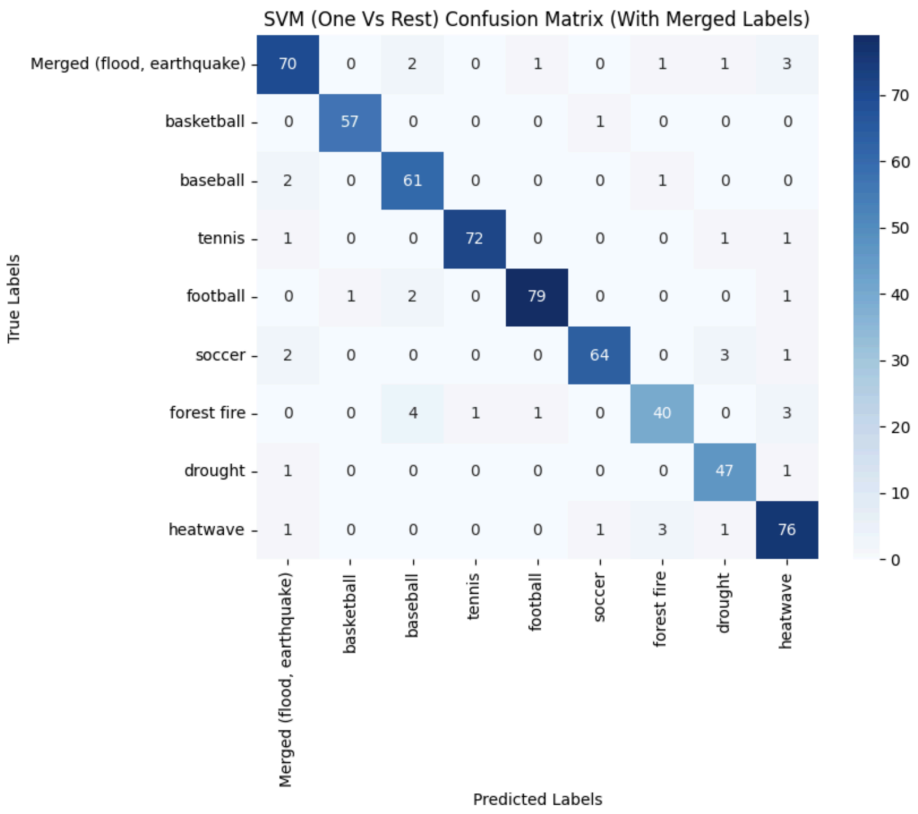
Based on your observation from the previous part, suggest a subset of labels that should be merged into a new larger label and recompute the accuracy and plot the confusion matrix. How did the accuracy change in One VS One and One VS the rest?

Based on the previous results, it would be advantageous to merge the flood and earthquake label data together as they are being misclassified for each other. This should help us improve our overall accuracy.

	NB	SVM (OvO)	SVM (OvR)
Accuracy	0.603	0.766	0.783


Naive Bayes Confusion Matrix (With Merged Labels)
True Labels
Predicted Labels
Merged (flood, earthquake)
basketball
baseball
tennis
football
soccer
forest fire
drought
heatwave
53 0 2 0 3 2 3 2 9
1 50 0 0 0 3 0 1 1
0 1 44 0 3 0 10 0 0
2 0 3 55 1 0 6 1 0
1 1 6 0 50 0 10 0 1
2 0 1 0 0 52 2 7 3
1 1 0 1 3 1 42 0 1
0 0 0 0 2 0 0 38 1
6 0 0 0 1 3 3 4 63
60
50
40
30
20
10
0

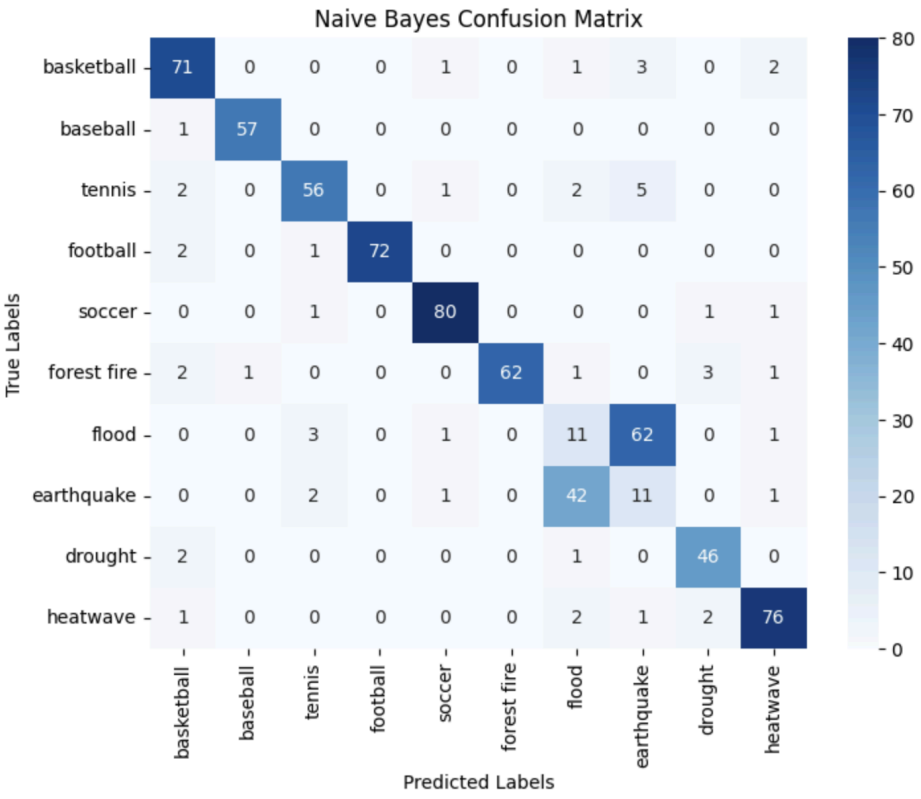

SVM (One Vs One) Confusion Matrix (With Merged Labels)
True Labels
Predicted Labels
Merged (flood, earthquake)
basketball
baseball
tennis
football
soccer
forest fire
drought
heatwave
72 0 0 0 1 0 0 0 2
1 56 0 0 0 1 0 0 0
2 0 56 0 0 0 0 0 0
1 0 0 69 0 0 0 0 2
0 0 1 0 79 0 0 0 1
4 0 0 0 0 61 0 2 2
2 0 3 0 1 0 31 0 0
1 0 0 0 0 0 0 45 2
1 0 0 0 0 0 0 1 73
70
60
50
40
30
20
10
0

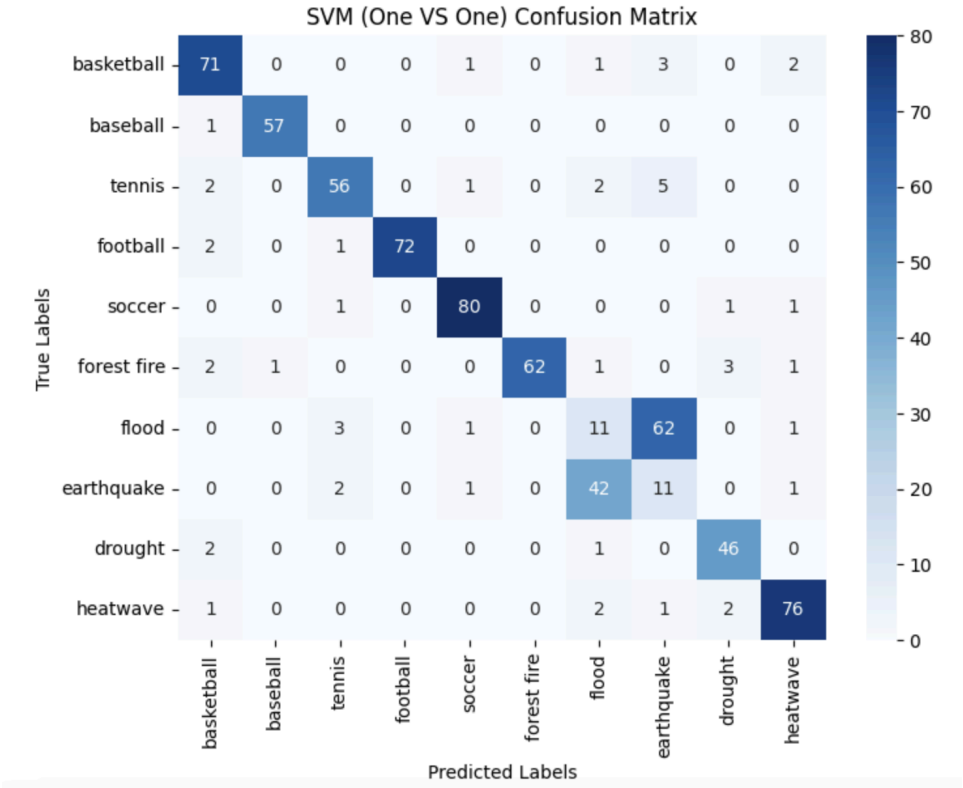


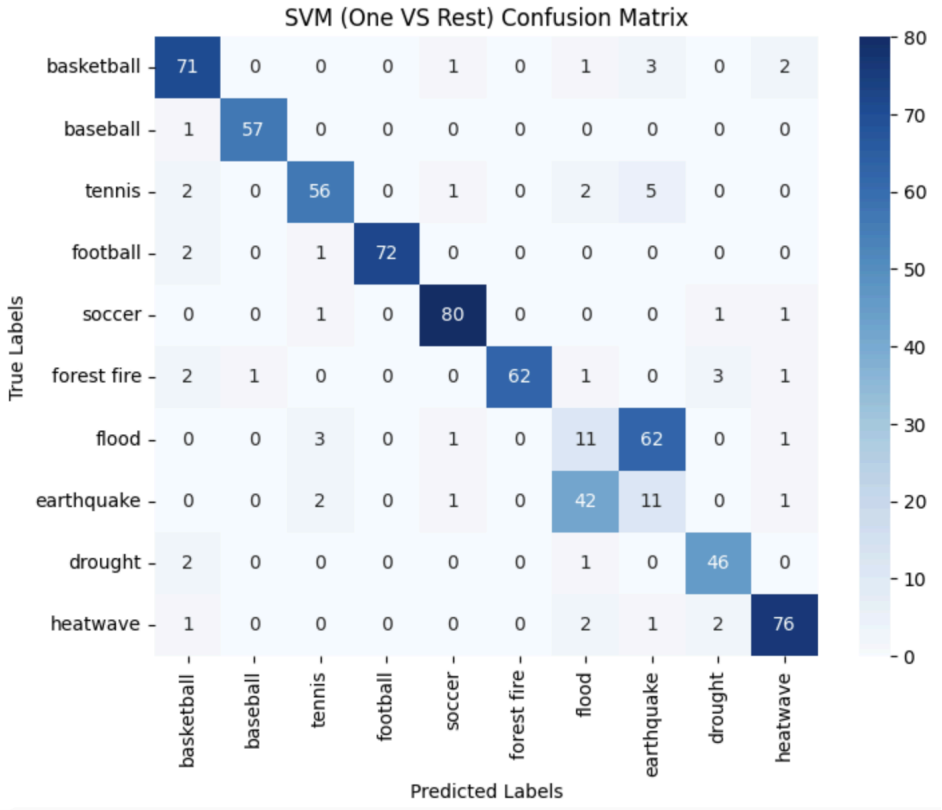
Does class imbalance impact the performance of the classification once some classes are merged? Provide a resolution for the class imbalance and recompute the accuracy and confusion matrix in One VS One and One VS the rest.

Class weights can be calculated to be inversely proportional to the class frequencies of the input data, in order to balance the impact of each class during the training of the model, and ideally help improve the overall accuracy.

	NB	SVM (OvO)	SVM (OvR)
Accuracy	0.603	0.766	0.779


Naive Bayes Confusion Matrix
basketball 71 0 0 0 1 0 1 3 0 2
baseball 1 57 0 0 0 0 0 0 0 0
tennis 2 0 56 0 1 0 2 5 0 0
football 2 0 1 72 0 0 0 0 0 0
soccer 0 0 1 0 80 0 0 0 1 1
forest fire 2 1 0 0 0 62 1 0 3 1
flood 0 0 3 0 1 0 11 62 0 1
earthquake 0 0 2 0 1 0 42 11 0 1
drought 2 0 0 0 0 0 1 0 46 0
heatwave 1 0 0 0 0 0 2 1 2 76
True Labels
basketball baseball tennis football soccer forest fire flood earthquake drought heatwave
Predicted Labels
80 70 60 50 40 30 20 10 0


SVM (One VS One) Confusion Matrix
basketball 71 0 0 0 1 0 1 3 0 2
baseball 1 57 0 0 0 0 0 0 0 0
tennis 2 0 56 0 1 0 2 5 0 0
football 2 0 1 72 0 0 0 0 0 0
soccer 0 0 1 0 80 0 0 0 1 1
forest fire 2 1 0 0 0 62 1 0 3 1
flood 0 0 3 0 1 0 11 62 0 1
earthquake 0 0 2 0 1 0 42 11 0 1
drought 2 0 0 0 0 0 1 0 46 0
heatwave 1 0 0 0 0 0 2 1 2 76
True Labels
basketball baseball tennis football soccer forest fire flood earthquake drought heatwave
Predicted Labels
80 70 60 50 40 30 20 10 0



Question 10

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

The co-occurrence probability is the probability of observing two words together in a given document. A ratio of co-occurrence probabilities greater than 1 indicates the two words occur together more frequently than individually, while a ratio less than 1 indicates that the co-occurrence happens less frequently than individual occurrence. The ratio of co-occurrence probabilities helps with capturing meaningful relationships between words.

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?

It would be reasonable to expect the GLoVE embedding to return different vectors for the two sentences as the meaning associated with the word “running” is different in each context. The co-occurrence patterns would also impact these results, as “running” is associated with “park” in one context and “presidency” in the other.

What do you expect for the values of,
||GLoVE["woman"] - GLoVE["man"]||2, ||GLoVE["wife"] - GLoVE["husband"]||2 and
||GLoVE["wife"] - GLoVE["orange"]||2 ? Compare these values.

||GLoVE["woman"] - GLoVE["man"]||2 = 4.754
||GLoVE["wife"] - GLoVE["husband"]||2 = 3.152
||GLoVE["wife"] - GLoVE["orange"]||2 = 8.668

The value between woman and man is more similar to the value between wife and husband, while we see a greater difference between wife and orange, implying the values are not as similar.

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

Lemmatization would be chosen over stemming as it is important to preserve the semantic meaning and context of the words with their relationships within GLoVE embedding. This is achieved through lemmatization as it reduces words to their canonical form, which holds a more meaningful representation.

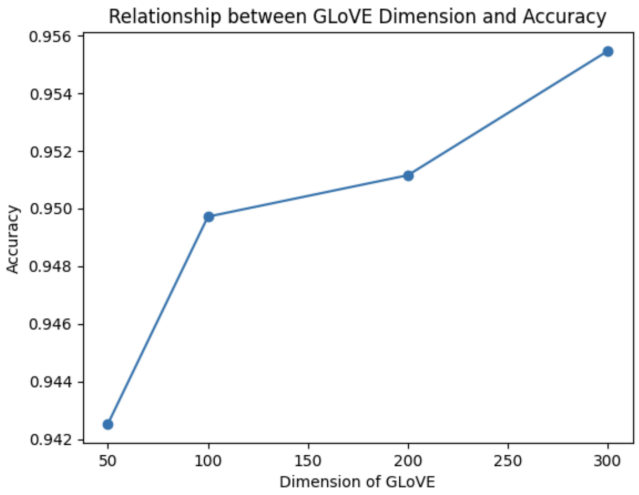
Question 11

Describe a feature engineering process that uses GLoVE word embeddings to represent each document. Select a classifier model, train, and evaluate it with your GLoVE-based feature.

The feature engineering process utilizes the GLoVE word embeddings to represent each document, with a logistic regression classifier utilized for training and testing the GLoVE features. This model results in an accuracy of 0.9368.

Question 12

Plot the relationship between the dimensions of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task. Describe the observed trend. Is this trend expected? Why or why not?

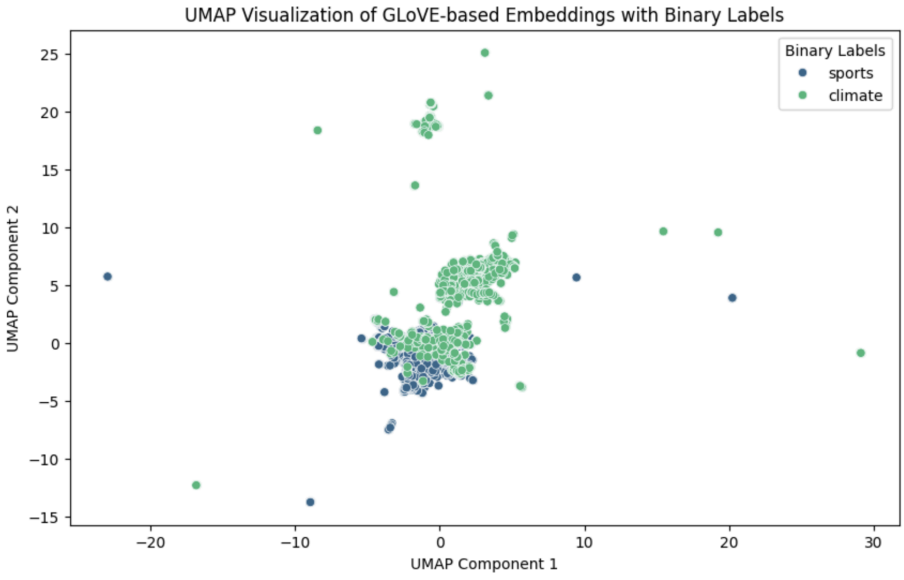


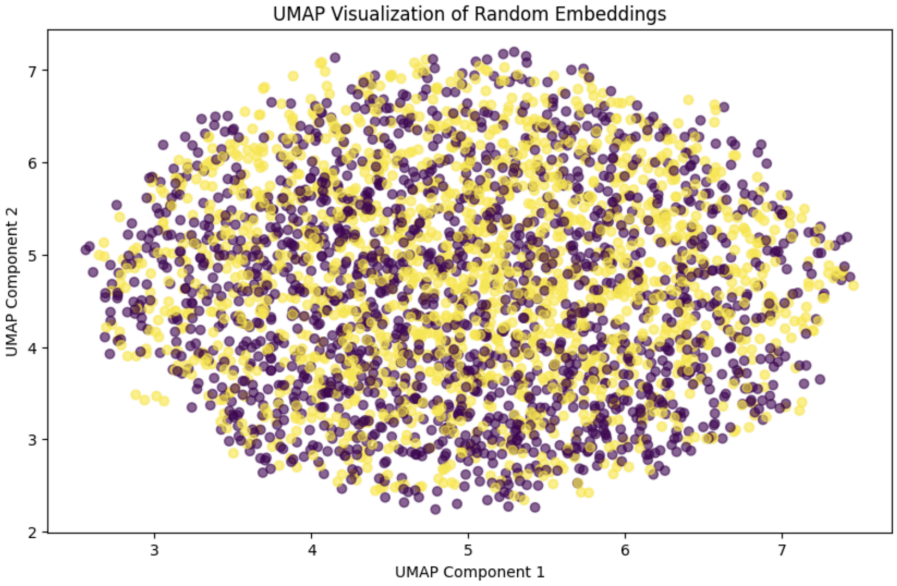
From the plot, it can be seen that as the dimensions of the GLoVE embeddings increase, the accuracy increases. This is expected as higher-dimensional embeddings should provide more information, which can assist the classifier in better modeling the relationships in the data, and lead to better overall performace.

Question 13

Compare and contrast the two visualizations. Are there clusters formed in either or both of the plots?

Within the plot of the GLoVE embeddings, clusters are formed for both sports and climate data with some outliers surrounding the clusters. For the plot of the random embeddings, no distinct clusters are formed. Instead the data points are evenly distributed across the plot.




UMAP Visualization of Random Embeddings
UMAP Component 2
UMAP Component 1
2
3
4
5
6
7
3
4
5
6
7