UCLA

ECE 219 - Large-Scale Data Mining: Models and Algorithms

Winter 2024

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

Sunday, January 28, 2024

Author:

Gian Zignago (UID: 706294998)

Question 1:

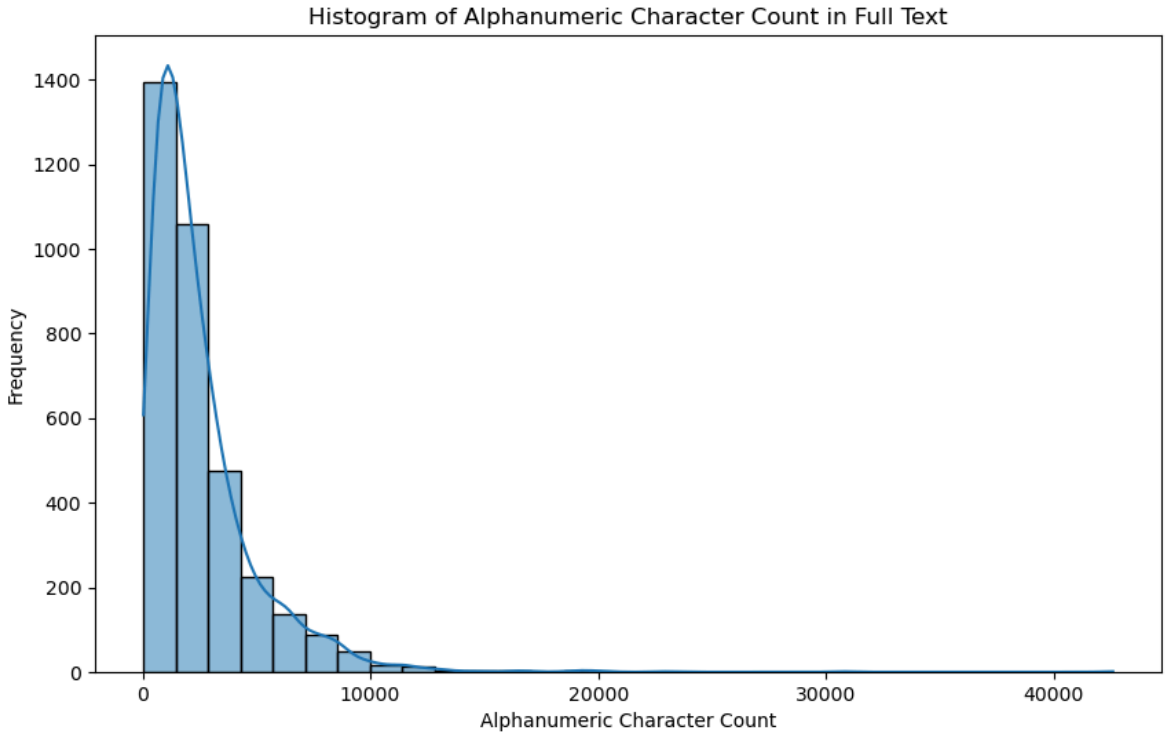
Overview: How many rows (samples) and columns (features) are present in the dataset?

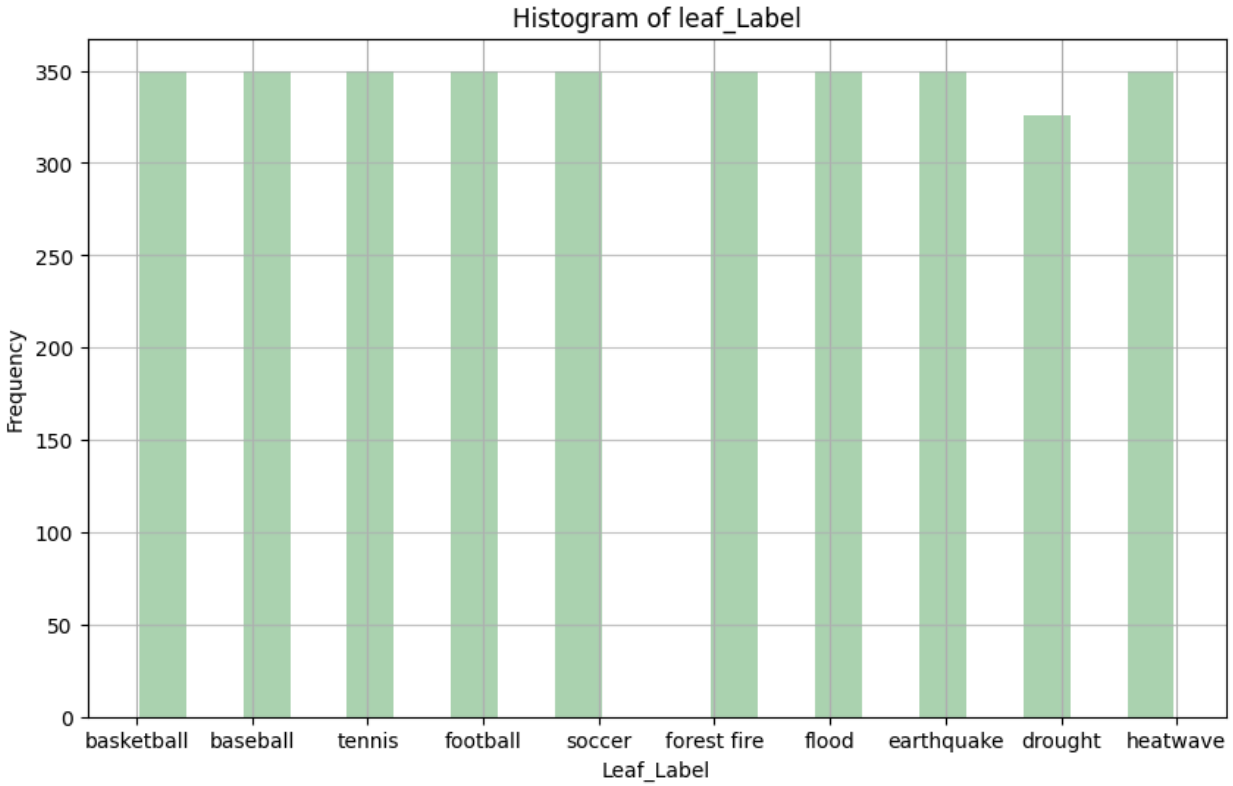
As described in the assignment, the dataset we are working with – titled "Project1-ClassificationDataset" – is a custom dataset created specifically for this quarter. The dataset is split into the following topics:

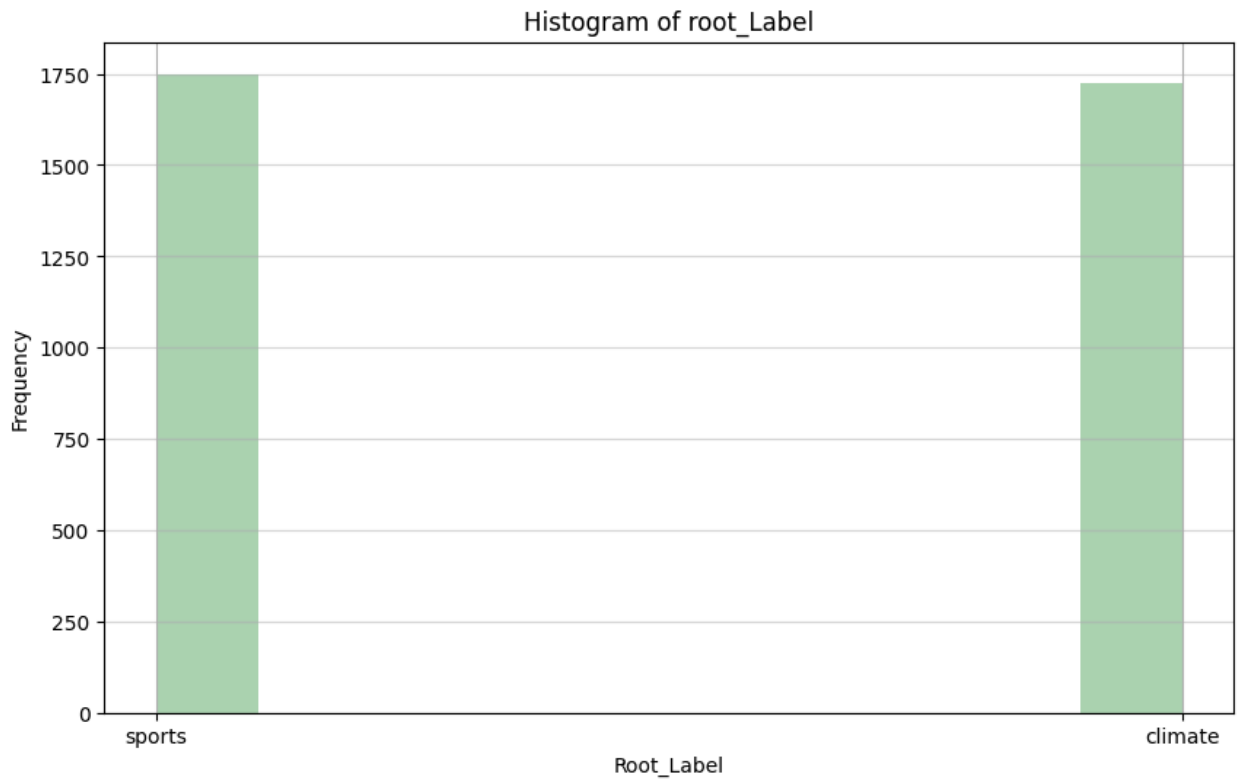
- 'basketball'
- 'baseball'
- 'tennis'
- 'football'
- 'soccer'
- 'forest fire'
- 'flood'
- 'earthquake'
- 'drought'
- 'heatwave'

There are 3476 rows (samples) and 8 columns (features) present in this custom dataset.

Histograms: Plot 3 histograms on : (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 the x-axis.


Histogram of Alphanumeric Character Count in Full Text
Frequency
Alphanumeric Character Count
0
200
400
600
800
1000
1200
1400
0
10000
20000
30000
40000


Histogram of leaf_Label
Frequency
Leaf_Label
0
50
100
150
200
250
300
350
basketball
baseball
tennis
football
soccer
forest fire
flood
earthquake
drought
heatwave



Interpretation of the Histograms:

The shape and spread of the alphanumeric character count histogram indicates the typical length of the text in each data point and how varied each text is. The high peak around the lower alphanumeric character count values suggests that most texts are of a similar short length. While there are a few texts with very large character counts, the histogram definitively shows that almost all texts in the dataset have less than 10000 alphanumeric characters (with most having a count significantly lower than 10000).

The histogram displaying frequency of leaf labels gives insights into the distribution of categories or types represented by these labels. A few high bars would suggest that some labels are much more common than others. In the given dataset, however, there is a near-even distribution of the frequency of each label. Notably, the “drought” label has a slightly lower frequency than the other leaf labels, meaning there are slightly fewer texts in the dataset classified under that label, but due to how similar the frequency distribution is, we can safely conclude that the dataset is already balanced.

The root label histogram visualizes how the dataset is distributed across different types. It reveals that, while the dataset is balanced between the “sports” and “climate” categories, the “sports” root label is slightly more overrepresented than the “climate” root label.

Question 2:

Report the number of training and testing samples.

There are 2,780 training samples and 696 testing samples.

Question 3:

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

Lemmatization tends to produce more accurate and meaningful results as it considers the context of the word in the sentence. It reduces words to their base or dictionary form (lemma). This process can potentially decrease the size of the vocabulary in the dataset, as multiple forms of a word are reduced to a single lemma. Stemming, on the other hand, simply removes affixes from words, often leading to non-words. It's generally faster but less accurate than lemmatization. The vocabulary size after stemming might be larger than with lemmatization, as it might not always group different forms of a word as effectively.

min_df means minimum document frequency. How does varying min df change the TF-IDF matrix?

The min_df parameter in feature extraction functions like TF-IDF sets the minimum frequency (document frequency) a term must have to be included in the vocabulary. A higher min_df value will result in a smaller vocabulary as it filters out rare terms, potentially reducing noise but also possibly omitting relevant but infrequent words. Conversely, a lower min_df value will include more terms, expanding the vocabulary and possibly capturing more nuances but increasing the risk of overfitting and noise.

Should I remove stopwords before or after lemmatizing? Should I remove punctuations before or after lemmatizing? Should I remove numbers 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.

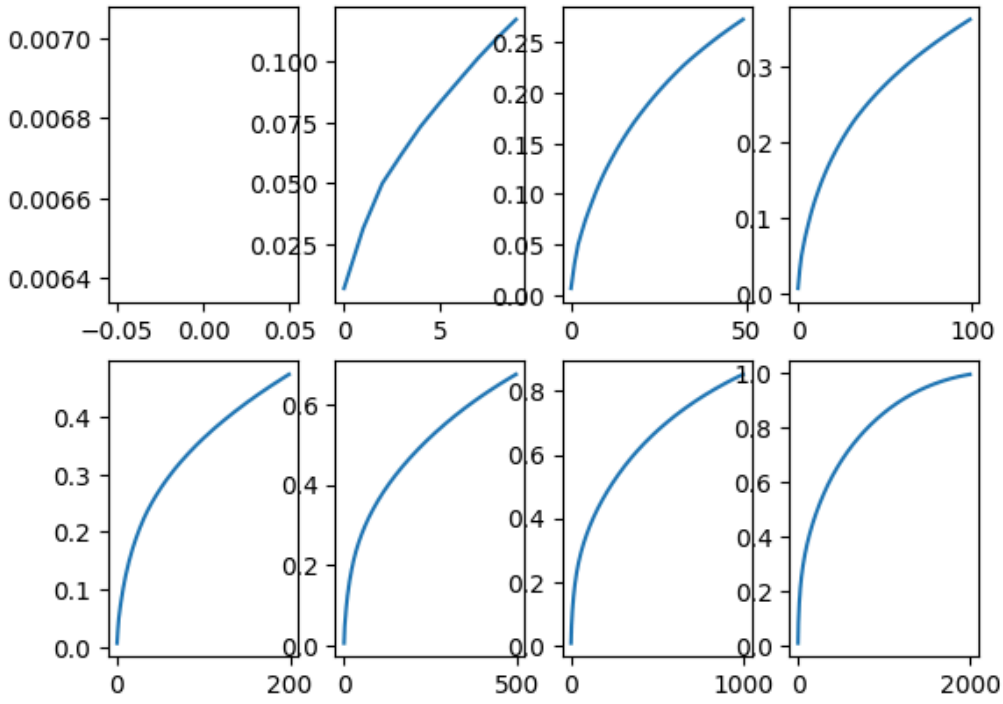
It is better to remove stopwords after lemmatization, as stopwords provide meaningful context necessary for accurate lemmatization. Punctuation should be removed before lemmatization, as it does not contribute to the meaning of words in the context of lemmatization. Numbers should also be removed before lemmatization, as they are not relevant in this context and do not need to be lemmatized.

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×103 . This dimension will vary depending on your exact method of cleaning and lemmatizing and that is okay.

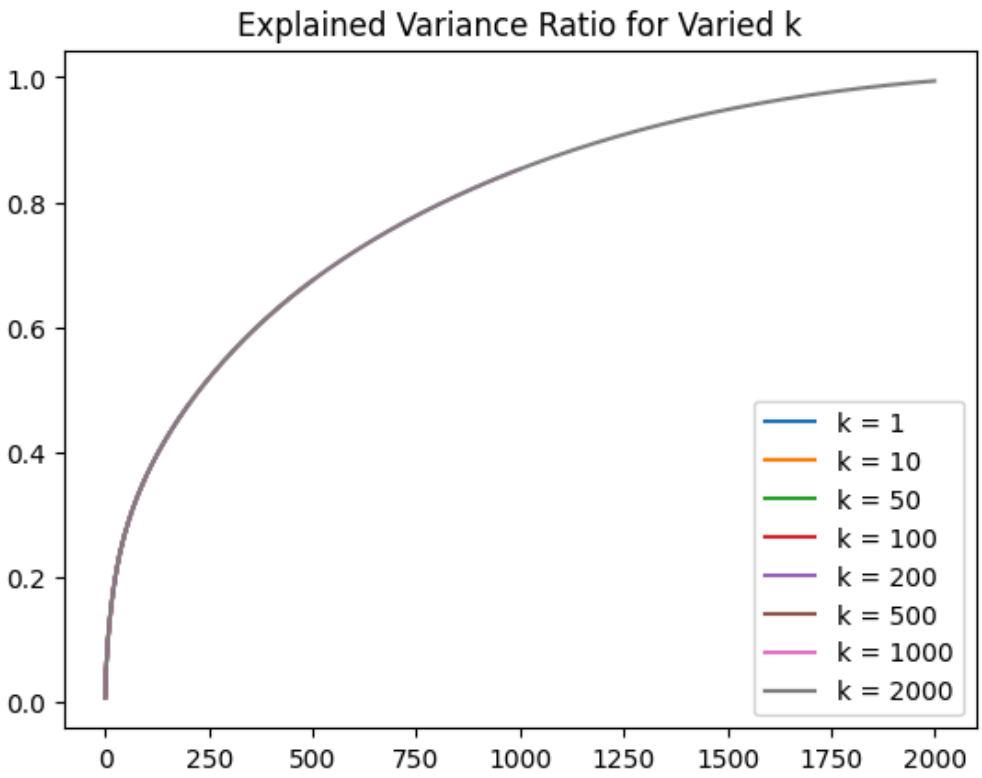
Train: (2780, 14150)
Test: (696, 14150)

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?



As the K increases the increase in explained variance reduces. We can see that approx 200 (10%) components explain 50% of the variance.



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 // X − WH // 2 F in NMF or the X − UkΣkVT k 2 F in LSI?

```
Error for NMF:  1987.6950728969205
Error for LSI:  1959.9979042690625
```

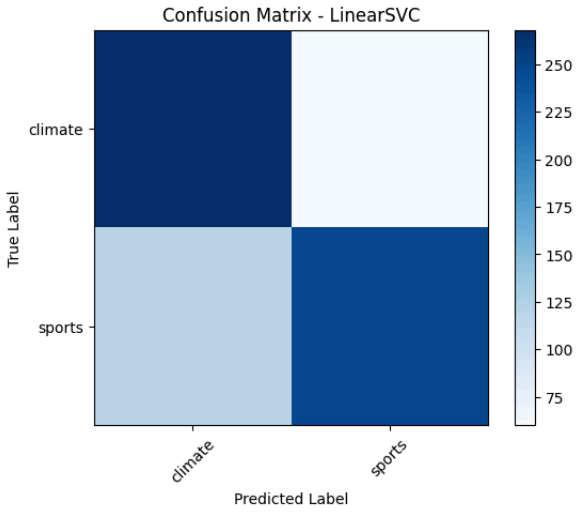
Error for NMF is larger than LSI

Question 5:

Compare and contrast hard-margin and soft-margin linear SVMs:

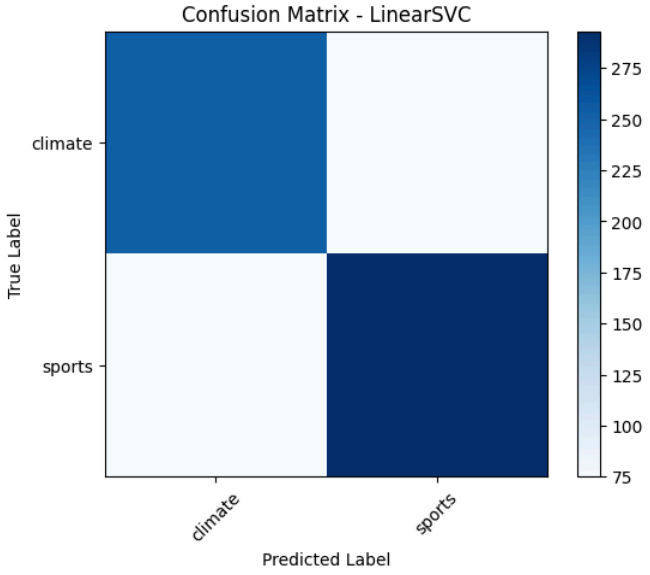
- Train two linear SVMs:
  - Train one SVM with γ = 1000 (hard margin), another with γ = 0.0001 (soft margin).
  - Plot the ROC curve, report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of both SVM classifiers on the testing set. Which one performs better? What about for γ = 100000?
  - What happens for the soft margin SVM? Why is the case? Analyze in terms of the confusion matrix.
    - Does the ROC curve reflect the performance of the soft-margin SVM? Why?

a) HARD SVM (C = 1000)



	precision	recall	f1-score
climate	0.69	0.82	0.75
sports	0.81	0.67	0.73

b) SOFT SVM(C = 0.001)

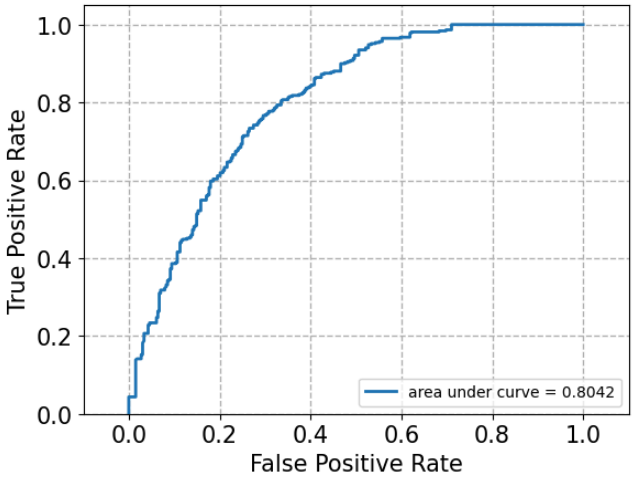


	precision	recall	f1-score
climate	0.77	0.77	0.77
sports	0.80	0.80	0.80

Also Tried non Linear SVm with RBF Kernel with gamma = 0.0001: Accuracy -> 0.772988505747126

- Use cross-validation to choose γ (use average validation 3 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.

```
Best C is 0.1
```



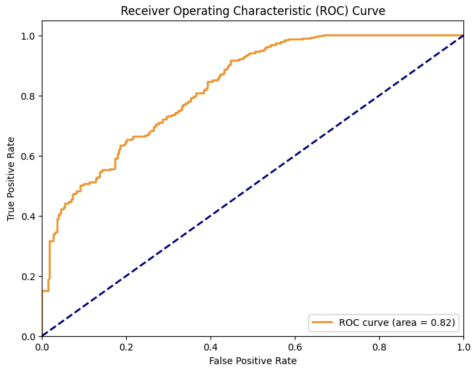
	precision	recall	f1-score
climate	0.69	0.75	0.72
sports	0.76	0.70	0.73

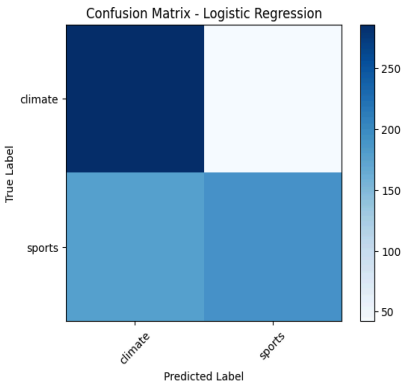
Question 6:

Evaluate a logistic classifier:

- Train a logistic classifier without regularization (you may need to come up with some way to approximate this if you use sklearn.linear model.LogisticRegression); 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.

ROC Curve, confusion matrix and metrics:






```
Accuracy: 0.6839080459770115
Recall: 0.6839080459770115
Precision: 0.7234938565200159
F1 Score: 0.6752234993614303
```

- Find the optimal regularization coefficient:

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

```
Best L1 Regularization Strength: 10
Best L2 Regularization Strength: 100
```

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 result w/o regularization, w/ L1 regularization and w/ L2 regularization respectively:

```
Accuracy: 0.6839080459770115
Recall: 0.6839080459770115
Precision: 0.7234938565200159
F1 Score: 0.6752234993614303

Accuracy: 0.6882183908045977
Recall: 0.6882183908045977
Precision: 0.7072882647426016
F1 Score: 0.6850274663543215

Accuracy: 0.6939655172413793
Recall: 0.6939655172413793
Precision: 0.714159702420891
F1 Score: 0.6906438303539765
```

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?

From the provided metrics, we can observe the following trends:

- Effect on Test Error:
    - l1 regularization: slight improvement compared to without regularization
    - l2 regularization: further improvement compared to without regularization

We can see that regularization helps in reducing overfitting and improving generalization to unseen data.

- Effect on Learnt Coefficients:
    - l1 regularization: tend to produce sparse coefficients by driving many of them to zero; help in feature selection by emphasizing the most important features while discarding the less important ones
    - l2 regularization: penalize the square of the magnitude of coefficients, leading to smaller but non-zero coefficients for all features; prevent overfitting by keeping all features in the model but reducing their magnitudes
- Interest in Each Type of Regularization:
    - L1 regularization may be preferred for feature selection and interpretability
    - L2 regularization can be preferred for overall performance improvement and stability in the model

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?

SVM and LR differ in optimization objectives and decision boundary definitions.

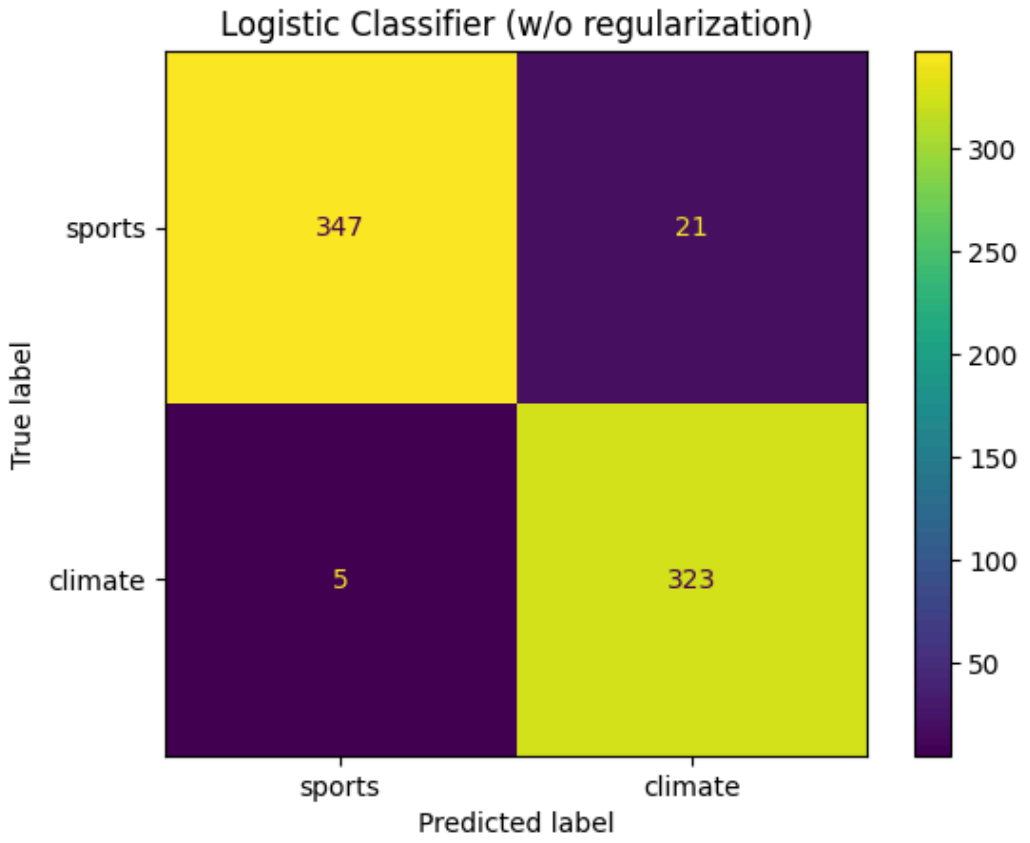
- Optimization Objectives:
  - Logistic Regression: Maximizes likelihood function, models class probabilities.
  - Linear SVM: Maximizes margin between support vectors, doesn't model probabilities directly.
- Decision Boundary:
  - Logistic Regression: Probability-based boundary.
  - Linear SVM: Margin-maximizing hyperplane.
- Performance differences arise due to:
  - Margin vs. Probability
  - Robustness to Outliers
  - Interpretability
- Statistical significance of performance difference depends on:
  - dataset and evaluation metrics, requiring rigorous testing like cross-validation or hypothesis testing

Question 7:

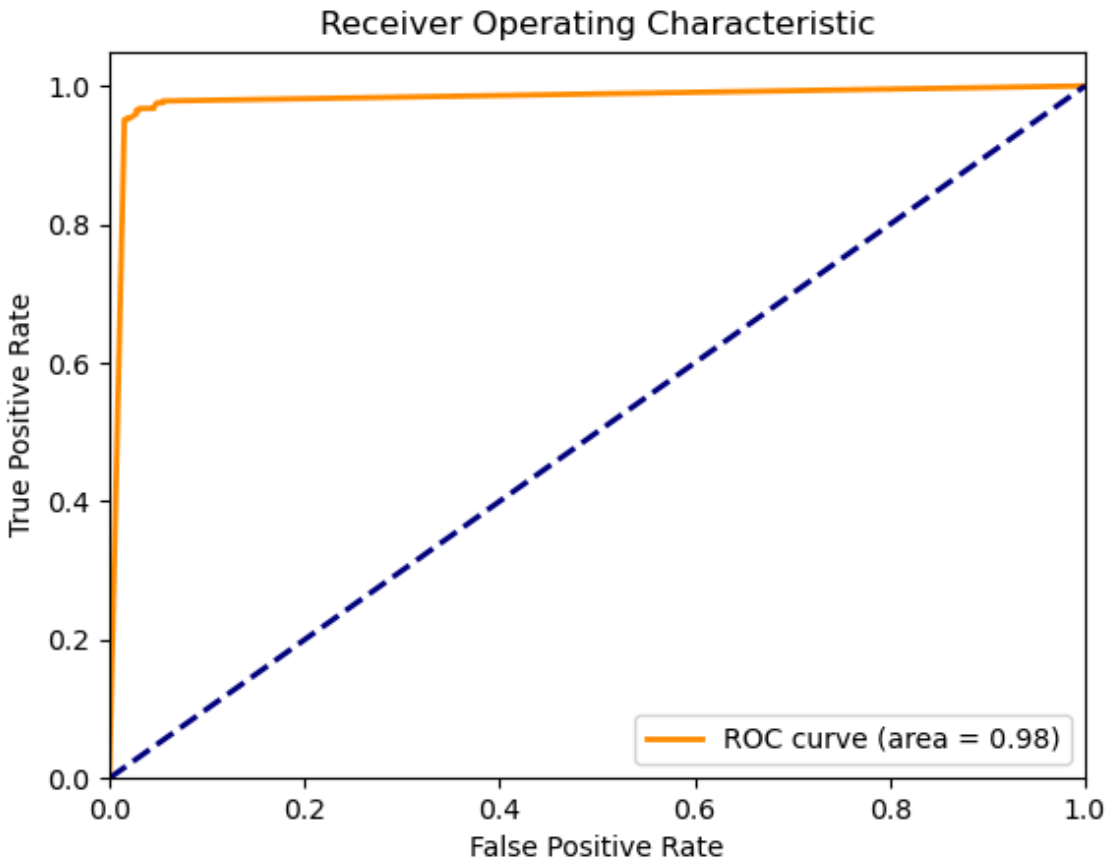
Evaluate and profile a Naïve Bayes classifier: Train a GaussianNB classifier; plot the ROC curve and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of this classifier on the testing set.

Accuracy: 0.9669540229885057
Recall: 0.9669540229885057
Precision: 0.9676768942776556
F-1 score: 0.9669780036576802

Confusion Matrix:



ROC Curve:



Area under ROC curve (AUC): 0.9802450623011663

Question 8:

In this part, you will attempt to find the best model for binary classification.

- Construct a Pipeline that performs feature extraction, dimensionality reduction and classification;
- The evaluation of each combination is performed with 5-fold cross-validation (use the average validation set accuracy across folds).
- What are the 5 best combinations? Report their performances on the testing set.

The 5 best combinations and their performances on the testing set are as follows:

1. Params: {'classifier': SVC(), 'dim_reduction': TruncatedSVD(n_components=80), 'dim_reduction__n_components': 80, 'vectorizer__min_df': 3}

   Mean CV Score: 0.9607913669064748

2. Params: {'classifier': SVC(), 'dim_reduction': TruncatedSVD(n_components=80), 'dim_reduction__n_components': 80, 'vectorizer__min_df': 5}

   Mean CV Score: 0.9561

3. Params: {'classifier': SVC(), 'dim_reduction': NMF(), 'dim_reduction__n_components': 80, 'vectorizer__min_df': 3}

   Mean CV Score: 0.9561

4. Params: {'classifier': SVC(), 'dim_reduction': TruncatedSVD(n_components=80), 'dim_reduction__n_components': 30, 'vectorizer__min_df': 5}

   Mean CV Score: 0.9547

5. Params: {'classifier': SVC(), 'dim_reduction': TruncatedSVD(n_components=80), 'dim_reduction__n_components': 30, 'vectorizer__min_df': 3}

   Mean CV Score: 0.9529

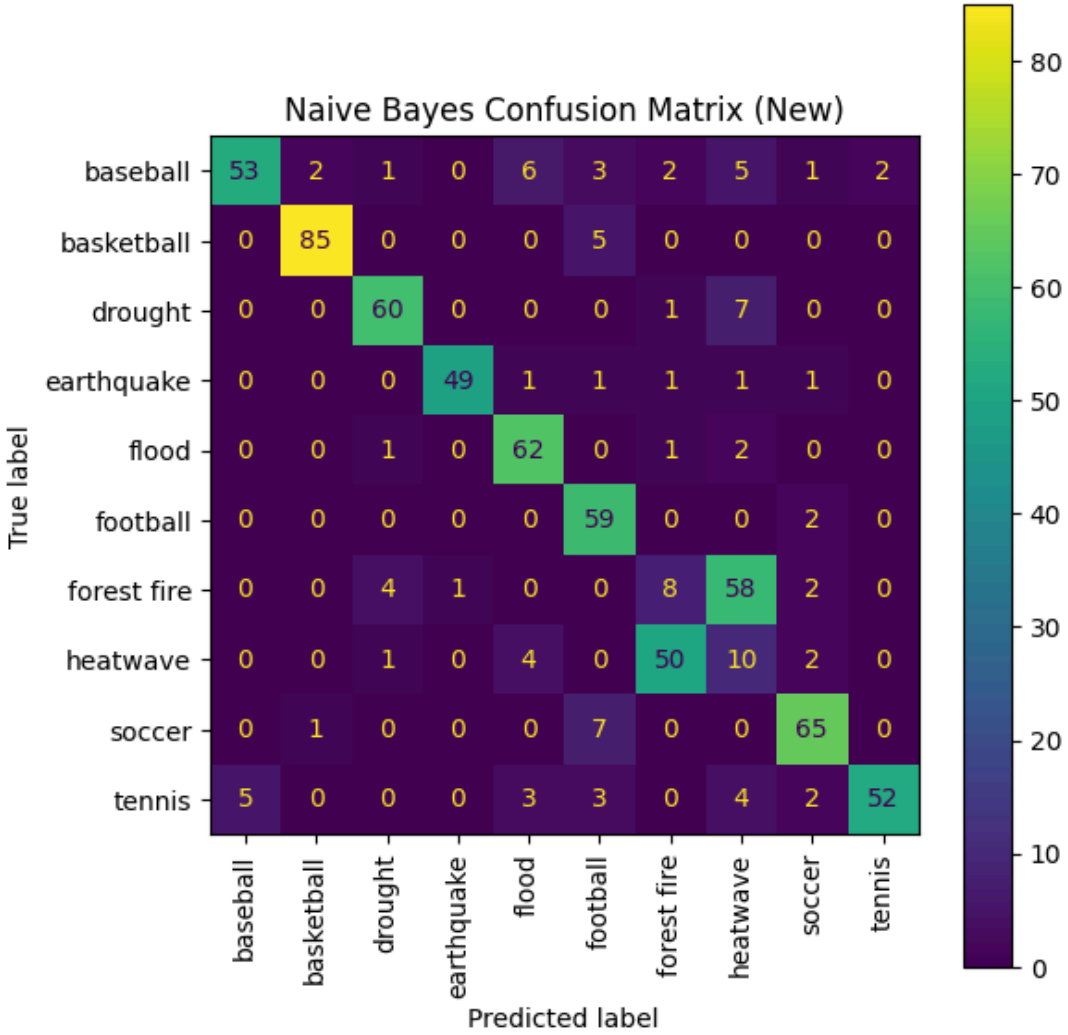
For context, the test score is 0.9640804597701149

Question 9:

In this part, we aim to learn classifiers on the documents belonging to unique classes in the column leaf label.

Perform Naive Bayes classification and multiclass SVM classification (with both One VS One and One VS the rest methods described above) and report the confusion matrix and calculate the accuracy, recall, precision and F-1 score of your classifiers. How did you resolve the class imbalance issue in the One VS the rest model?

Results of running Naive Bayes classification:
Accuracy: 0.7227011494252874
Recall: 0.7250355433592107
Precision: 0.7398980597860925
F1: 0.7276829564460531
Confusion Matrix:



Results of running SVM One Versus One classification:

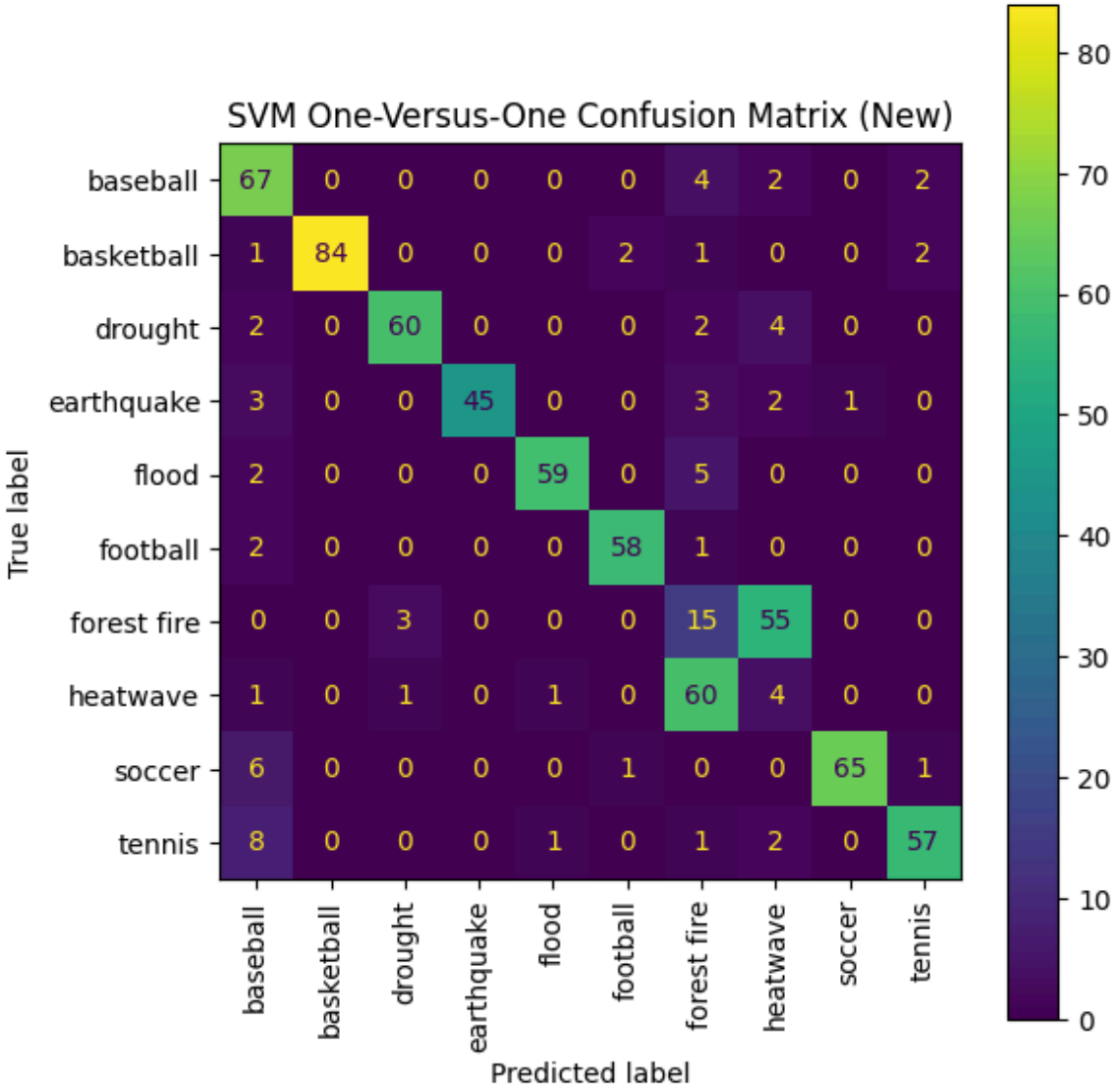
Accuracy: 0.7385057471264368

Recall: 0.7368790867264969

Precision: 0.7709011472762248

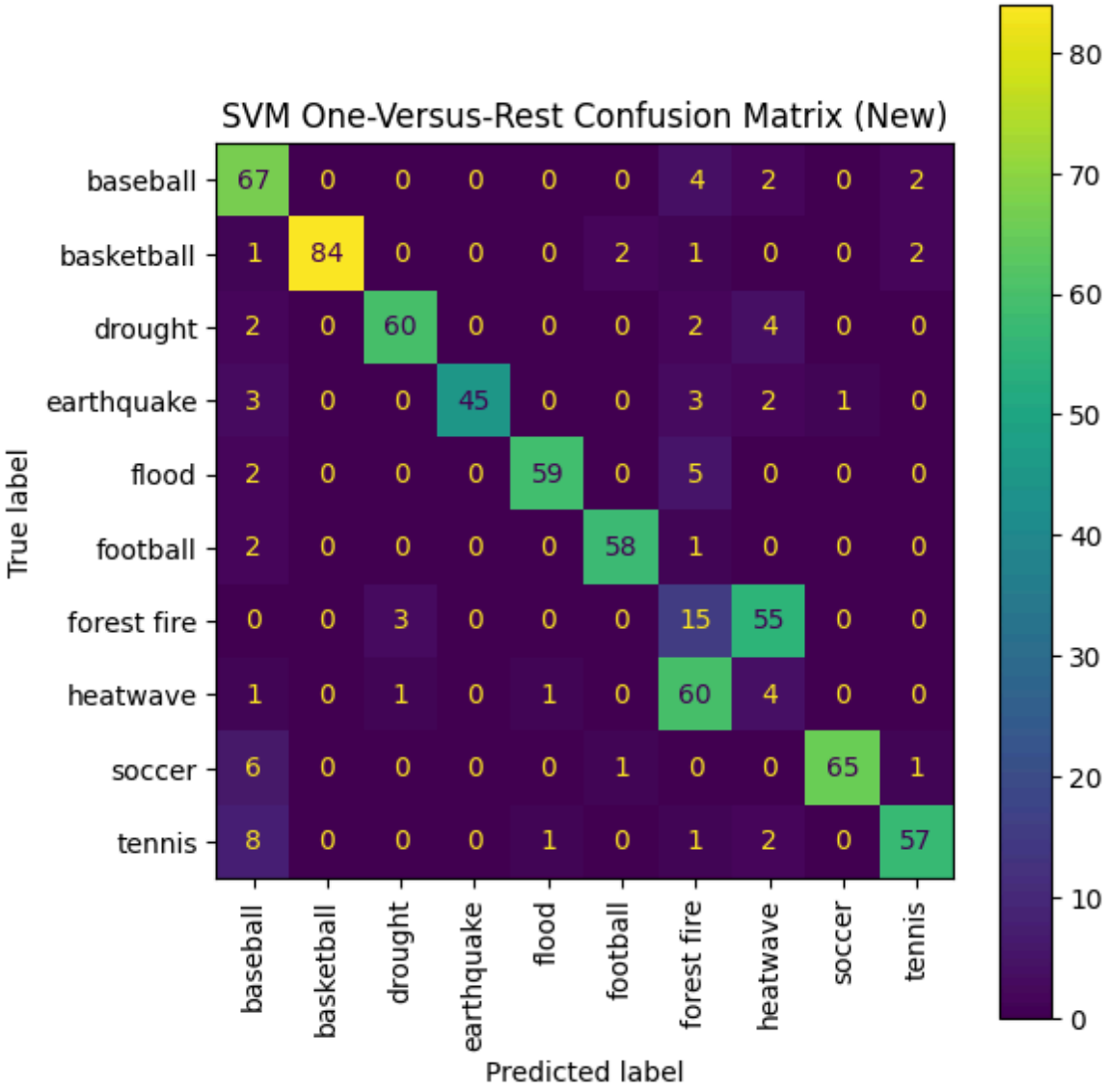
F1: 0.7512170316965516

Confusion Matrix:



Results of running SVM One Versus The Rest classification:

Accuracy: 0.7385057471264368
Recall: 0.7368790867264969
Precision: 0.7709011472762248
F1: 0.7512170316965516
Confusion Matrix:



Resolving the class imbalance issue in the Once VS the rest model:

To handle class imbalance for One-Versus-Rest SVM, we used the strategies of implementing class weights (setting class_weight='balanced' in SVC) and oversampling the minority class using the RandomOverSampler function from imblearn.over_sampling.

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

Naive Bayes Confusion Matrix:

There is a well-defined diagonal present in the confusion matrix, meaning that the accuracy is relatively high. This suggests that Naïve Bayes performs reasonably well for this dataset. However, looking at the confusion matrix, there are certain classes where Naïve Bayes is particularly confused, such as class 6 (earthquake) being predicted as class 7 (flood) 58 times. The F-1 score, which is a harmonic mean of precision and recall, is also decent, indicating good balance between precision and recall.

SVM (One-Versus-One and One-Versus-Rest):

The accuracy and other metrics for SVM are very similar in both One-Versus-One and One-Versus-Rest strategies, which is not always the case. Like in the Naive Bayes confusion matrix, both also have relatively well-defined diagonals present in their confusion matrices. There is a noticeable confusion between certain classes in SVM as well, like class 7 (flood) with class 6 (earthquake) and class 9 (heatwave) being confused with class 0 (basketball).

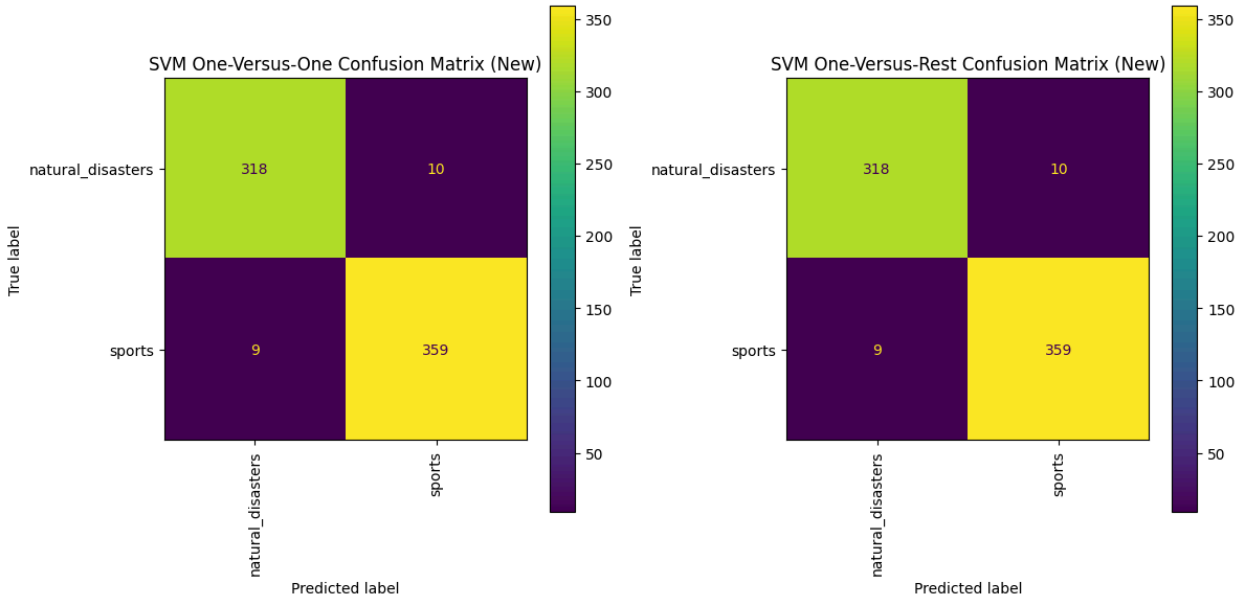
Based on your observation from the previous part, suggest a subset of labels that should be merged into a new larger label and recompute the accuracy and plot the confusion matrix. How did the accuracy change in One VS One and One VS the rest?

From our results, it seems that a small set of categories are consistently confused. Specifically, the ‘earthquake’ and ‘flood’ classes get mixed up by the classifiers often. Thus, we have decided to merge these two labels along with 'forest fire', 'drought', and 'heatwave' into a single label titled ‘natural_disasters’. Similarly, we decided to merge 'tennis', 'basketball', 'baseball', 'football', and 'soccer' into the label 'sports'. Here is the updated confusion matrices after making these label consolidations:

Changes in accuracy in One VS One and One VS the rest after label consolidation:

- SVM One-Versus-One Metrics (Old vs. New):
    - Old Accuracy: 0.7385057471264368
    - New (Larger Label) Accuracy: 0.9727011494252874
- SVM One-Versus-Rest Metrics (Old vs. New):
    - Old Accuracy: 0.7385057471264368
    - New (Larger Label) Accuracy: 0.9727011494252874

Updated confusion matrix:



The accuracy for SVM One-Versus-One increased notably from approximately 73.9% to 97.3% given the new larger labels. The accuracy for SVM One-Versus-Rest is the same as One-Versus-One, also increasing from 73.9% to 97.3%. This drastic improvement in accuracy is expected, as combining the subcategories into larger labels eliminates the confusion between specific sub-categories such as ‘flood’ and ‘earthquake’.

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 plot the confusion matrix in One VS One and One VS the rest?

Class Imbalance Impact:

The initial performance before merging showed some significant misclassifications, especially between classes like 'earthquake' and 'flood'. After merging, the class imbalance issue seems to be mitigated, and the model's ability to correctly classify has improved. The class imbalance impacts the performance of classification algorithms significantly; by merging the classes, we have balanced the class distribution, making the dataset easier for the models to learn from.

Resolution for Class Imbalance:

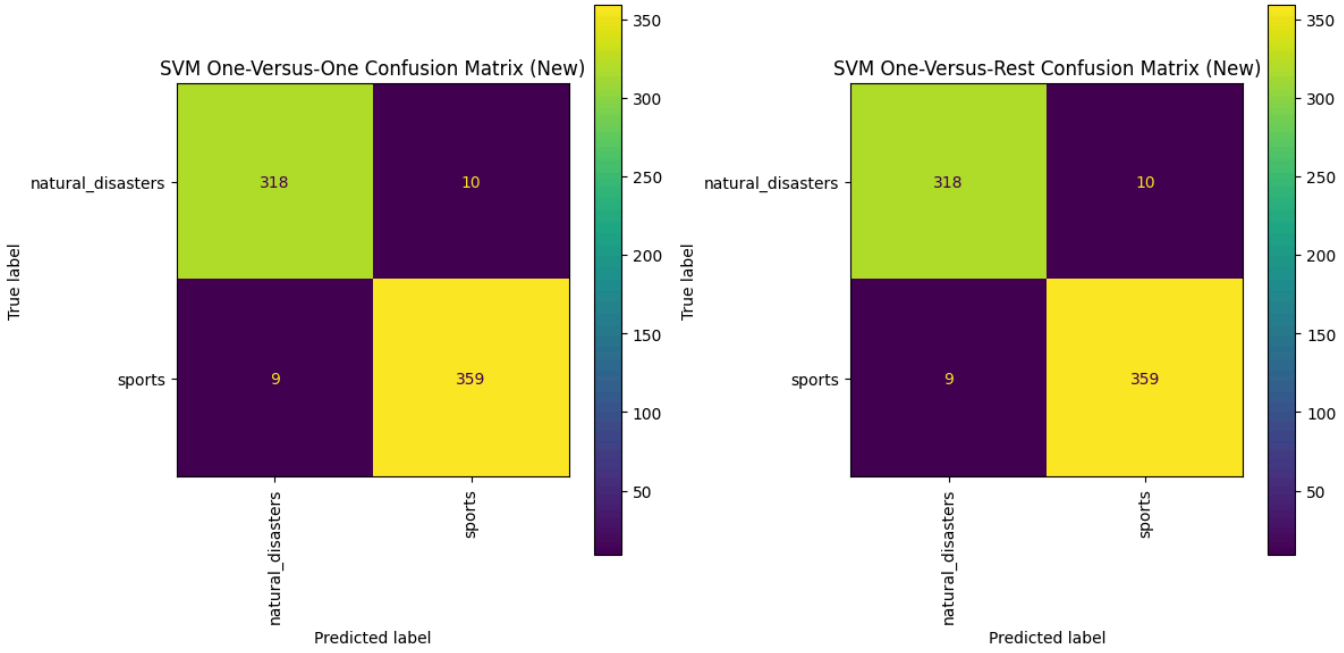
In the original setup, the class imbalance could have been resolved using techniques like oversampling the minority classes, undersampling the majority classes, or using class weights. However, by merging the classes into broader categories, the balance is naturally improved. While implementing these techniques will not diminish the accuracy, they will likely not improve the accuracy much if at all as there is not a significant extant class imbalance present in the current classification process.

Changes in accuracy in One VS One and One VS the rest after implementing techniques to resolve class imbalance:

- SVM One-Versus-One Metrics (Old vs. New):
    - Old (Larger Label) Accuracy: 0.9727011494252874
    - New (Resolved Imbalance) Accuracy: 0.9727011494252874
- SVM One-Versus-Rest Metrics (Old vs. New):
    - Old (Larger Label) Accuracy: 0.9727011494252874
    - New (Resolved Imbalance) Accuracy: 0.9727011494252874

As expected, the new accuracy resulting from implementing the techniques to resolve class imbalance described above is unchanged. While there was no loss in accuracy, there was also no gain.

Updated confusion matrix:



Question 10:

a) The ratio of co-occurrence probabilities in the GloVe model is a design choice that enables better capturing of word relationships, additivity of word vectors, and mitigation of biases introduced by frequent words. It helps create embeddings that reflect semantic relationships and capture more subtle linguistic patterns.

b) GloVe embeddings capture the semantic relationships between words based on their co-occurrence patterns, and words with different meanings or contexts are expected to have distinct vector representations. Therefore, I generally expect the GloVe embeddings for "running" in the two sentences to be different.

c) **||GLoVE['woman'] - GLoVE['man']||$_2$: 4.753939628601074**
||GLoVE['wife'] - GLoVE['husband']||$_2$: 3.1520464420318604
||GLoVE['wife'] - GLoVE['orange']||$_2$: 8.667715072631836

We can see that distance between woman and man is similar to distance between wife and husband while distance between wife and orange is much larger than the first two

d) If maintaining semantic consistency is crucial for your task, lemmatization is generally preferred. It ensures that words with similar meanings have similar vector representations.

Question 11:

a) The feature engineering approach employed involved computing the average matrix for each text segment within a document, fitted to a size of 300, aligning with the dimensions of a GLoVE vector. This process was applied to the keywords, with the resulting matrices averaged to obtain a consolidated matrix with dimensions (2780, 300) for training and (696, 300) for testing.

b) Based on the findings earlier, we used a Gaussian Naive Bayes classifier with the feature engineering described above. We computed and displayed the confusion matrix and accuracy on the corresponding test set, found below. Given the results, we decided that the best features to include were full text and keywords.

```
GLoVE clf Confusion Matrix:
[[224  44]
[ 97 265]]
GLoVE clf Accuracy: 0.7761904761904762
```

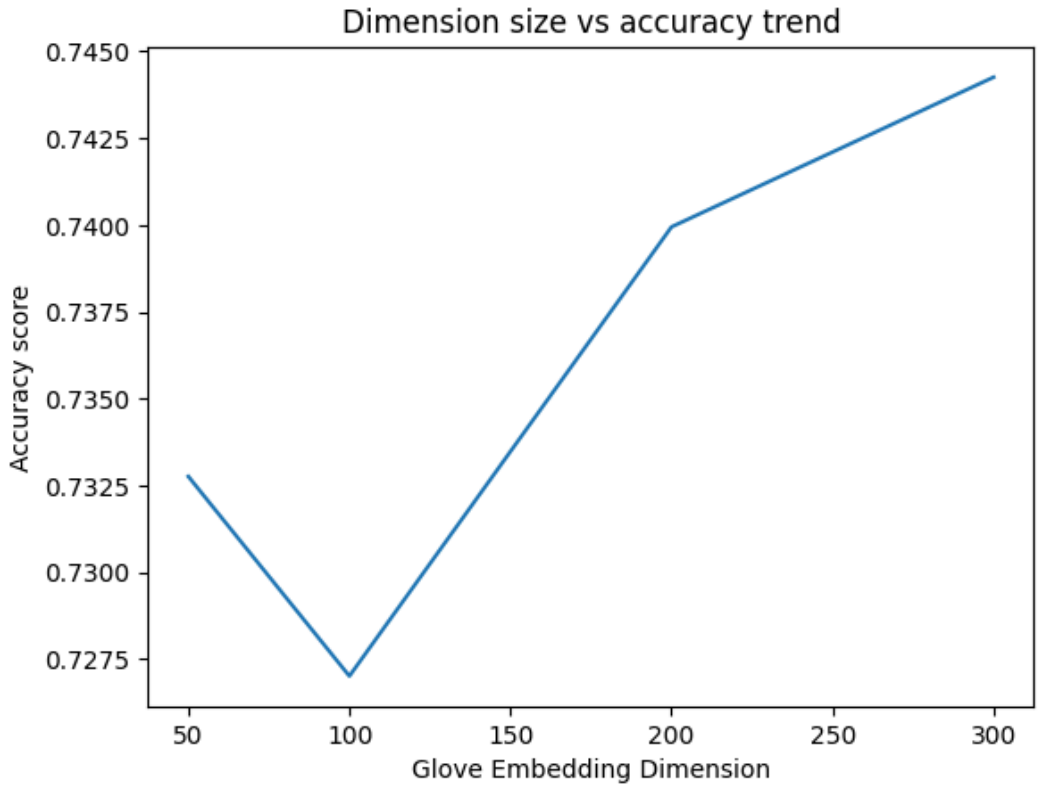
```
GLoVE clf Confusion Matrix:
[[253  97]
[ 93 253]]
GLoVE clf Accuracy: 0.7270114942528736
```

```
GLoVE clf Confusion Matrix:
[[268  82]
[ 99 247]]
GLoVE clf Accuracy: 0.7399425287356322
```

```
GLoVE clf Confusion Matrix:
[[270  80]
[ 98 248]]
GLoVE clf Accuracy: 0.7442528735632183
```

Question 12:

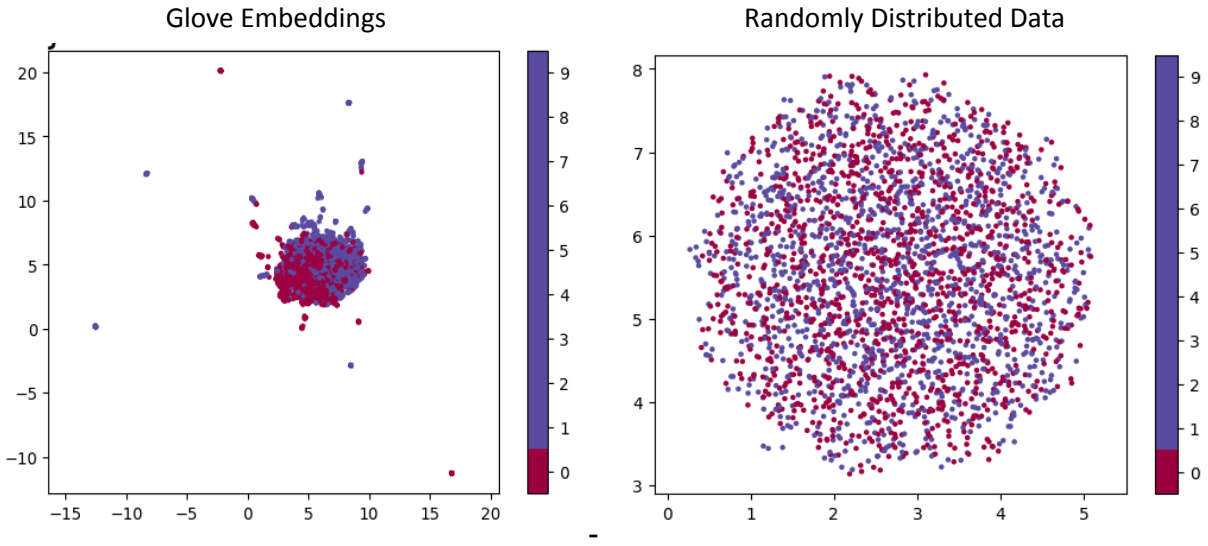
Plot the relationship between the dimension of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task. Describe the observed trend. Is this trend expected? Why or why not?



As the dimension of the GloVe embedding increases from 50 to around 150, there's a notable dip in accuracy. However, continuing from 150 to 300, the accuracy increases consistently. The initial decrease in accuracy as the dimension size moves from 50 to 100 is due to the fact that the model overfits with a 50-dimensional embedding, providing high accuracy on training data that doesn't generalize well. The increase in accuracy from 150 to 300 suggests that the additional dimensions provide more nuanced information that the model can leverage to make better predictions. As the dimensions increase, the complexity of patterns and relationships between words the embeddings can capture increases. This trend is expected, as larger embeddings capture more linguistic nuances.

Question 13:

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



The Glove Embeddings plot shows a clear cluster of points in the center with a high density of overlapping points. There are some sparse points scattered around the central cluster, with a few outliers far from the main group. The significant cluster near the center of the plot suggests a strong grouping within the data. Although there are a few outliers in the plot, the large group of points represents a cohesive cluster.

The randomly distributed data plot, as expected, depicts a more uniform, dispersed set of points in a large circular shape with no obvious grouping. Therefore, there is no cluster in the randomly distributed data plot.