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

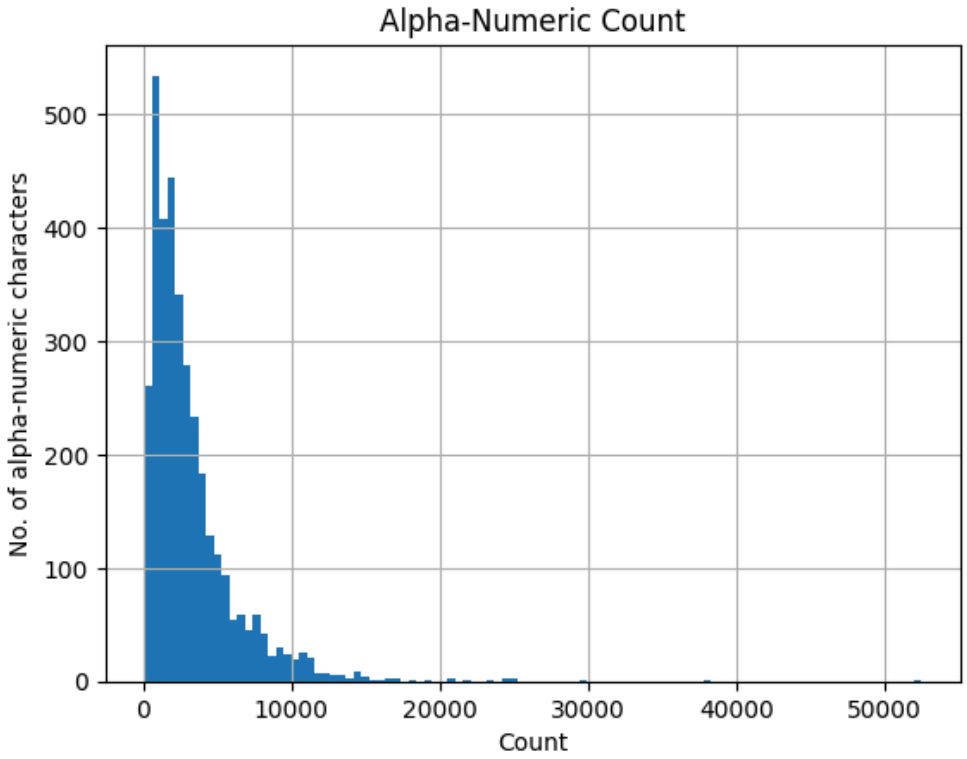
Tania Rajabally
UID: 806153219

QUESTION 1:

Provide answers to the following questions:

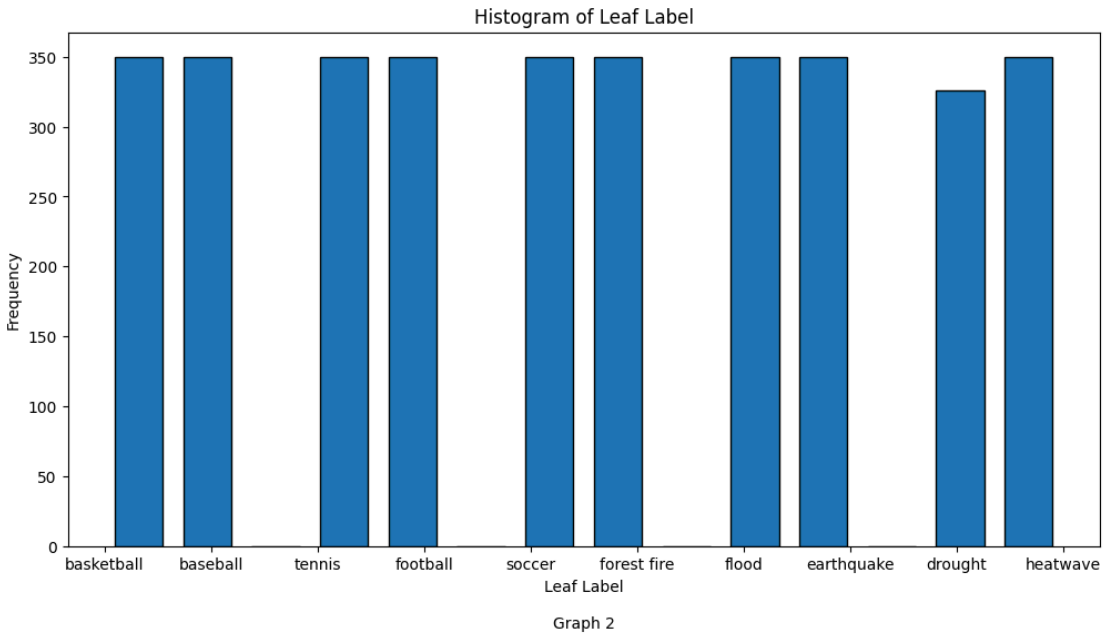
- **Overview: How many rows (samples) and columns (features) are present in the dataset?**
  There are 3476 rows (samples) and 8 columns (features)

- **Histograms: Plot 3 histograms on :**
  - **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;**



Frequency of Alpha-Numeric Characters per Data Point above represents this histogram.

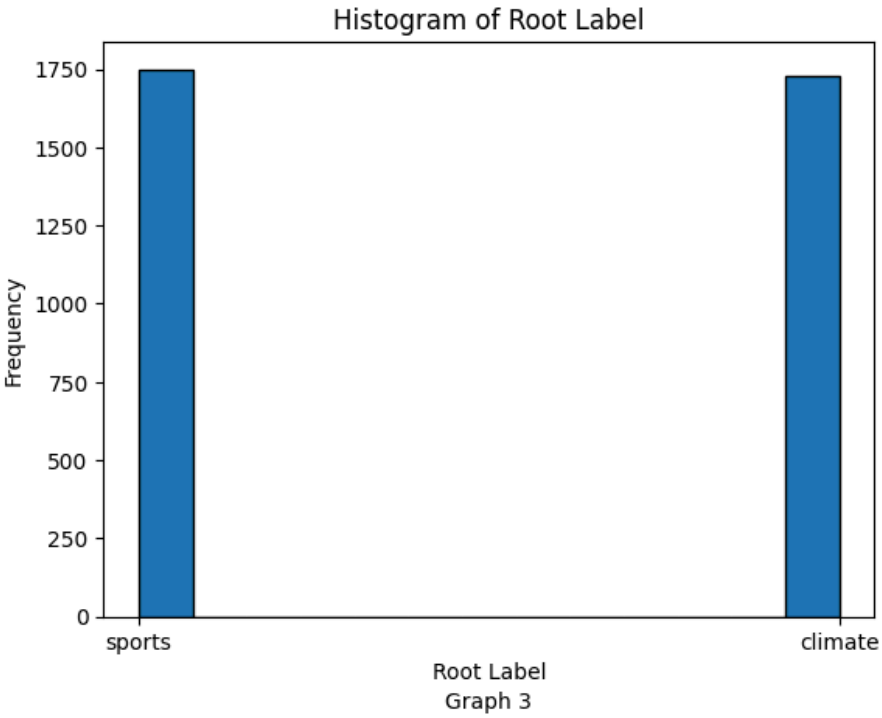
  - **The column leaf label – class on the x-axis;**



Graph 2

Histogram of Leaf Label above represents this histogram.

- **The column root label – class on the x-axis.**



Graph 3

Histogram of Root Label above represents this histogram.

- **Interpret Plots: Provide qualitative interpretations of the histograms.**

From the root label histogram, we can see that the data is evenly distributed i.e there are equal number of rows having root label as sports and equal number of rows having root label as climate. Even for the leaf labels, it is almost evenly distributed. Only drought has a little lower data as compared to the rest. But, all the other leaf labels have evenly distributed data. This even distribution is desired as it prevents the model from overfitting on classes that have more samples than the rest.

QUESTION 2:

Report the number of training and testing samples.

There are 2780 rows in the training samples.
There are 696 rows in the testing samples.

QUESTION 3:

Use the following specs to extract features from the textual data:
*** Before doing anything, please clean each data sample using the code block provided above. This function helps remove many but not all HTML artefacts from the crawler's output. You can also build your own cleaning module if you find this function to be ineffective.**
*** Use the "english" stopwords of the CountVectorizer**
*** Exclude terms that are numbers (e.g. "123", "-45", "6.7" etc.)**
*** Perform lemmatization with nltk.wordnet.WordNetLemmatizer and pos tag**
*** Use min df=3**

Please answer the following questions:

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

Both lemmatization and stemming are techniques used to reduce words to their base or root form.

Pros of lemmatization:
- It produces valid words. The base words can be found in the dictionary and are hence valid words.
- The lemmatized words are more interpretable and maintain the semantic meaning better than stemming.

Cons of lemmatization:
- It is computationally expensive since it involves dictionary lookups and morphological analysis.
- Lemmatization may not be able to deal with irregular forms.

Pros of stemming:
- Stemming is computationally efficient and is faster than lemmatization since it involves simpler rule based truncation.
- Stemming reduces words to their base or root form by removing common prefixes or suffixes, helping to group related words.

Cons of stemming:
- Stemming may produce words that do not exist.
- It might be overly aggressive in its truncation, leading to loss of information.

How these processes affect dictionary size:
- Lemmatization: lemmatization reduces the dictionary size compared to the original text. This happens because words are mapped to their base or dictionary forms, eliminating variations.

- Stemming: It can also reduce the dictionary size, but it may be less effective in eliminating variations compared to lemmatization. Stemmed words might still have variations that are not fully captured.

Lemmatization is often preferred when the preservation of valid words and semantic meaning is crucial, even at the cost of computational complexity. Stemming, on the other hand, is faster and can be suitable for applications where computational efficiency is a priority, and minor variations are acceptable.

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

min_df helps control the vocabulary size. It excludes terms that appear in a very small number of documents. Hence, changing the min_df parameter can have a significant impact on the resulting TF-IDF matrix. If we have a higher min_df, the words which appear in very few documents will be eliminated from the vocabulary. Hence, the vocabulary size will be smaller and the matrix will focus on the terms that occur frequently. If we have a smaller min_df, we will include the terms that do not occur frequently and hence will have a larger vocabulary size. The matrix may include more specific terms that are unique to certain documents, potentially capturing more nuanced information

- **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.

Generally stopwords, punctuations and numbers are removed before lemmatization. Stopwords do not carry significant meaning. Hence, removing them early in the preprocessing pipeline helps to reduce noise and focus on the more meaningful terms. Punctuation marks usually do not contribute to the meaning of words, and removing them early in the process helps to simplify the text. If numbers are not crucial to your analysis, removing them early can help streamline the text.

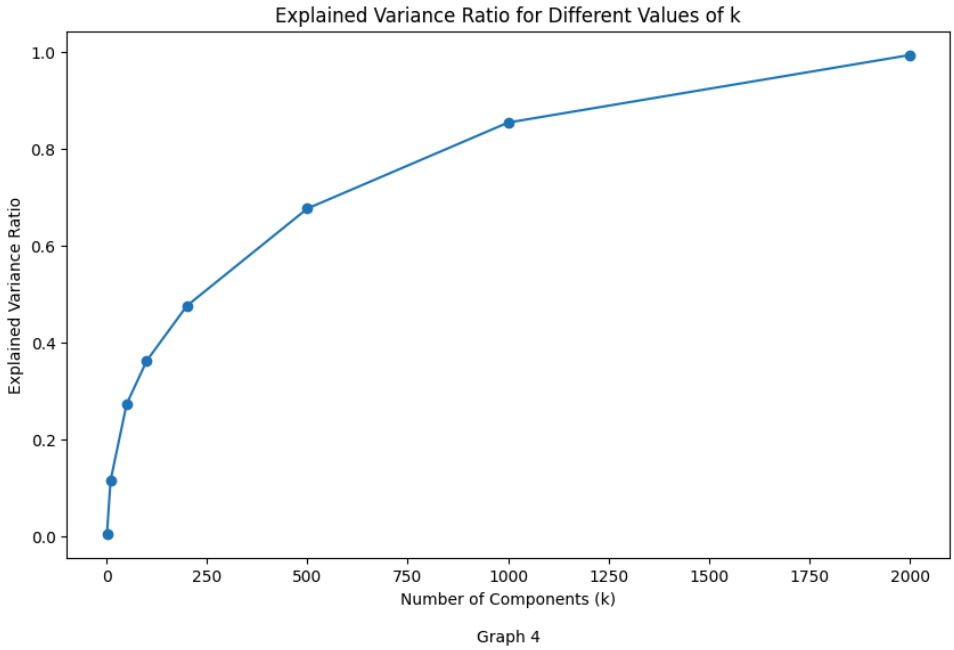
- **Report the shape of the TF-IDF-processed train and test matrices. The number of rows should match the results of Question 2. The number of columns should roughly be in the order of k×103. This dimension will vary depending on your exact method of cleaning and lemmatizing and that is okay.**

TF-IDF-processed train data shape : (2780, 13729)
TF-IDF-processed train data shape : (696, 13729)

<u>QUESTION 4:</u>

Reduce the dimensionality of the data using the methods above:

- **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?**



Graph 4

The figure above shows the Explained Variance Ratio for Different Values of k.

The explained variance ratio changes with increasing k. If the plot shows diminishing returns in terms of explained variance ratio as k increases, it suggests that adding more components beyond a certain point contributes less to explaining the variance in the data. The concavity may indicate a point of diminishing returns in terms of capturing additional information with more components.

- **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 // 2F in NMF or the X − UkΣkVTk 2F in LSI and why?**

Error for LSI: 1954.3696979505175
Error for NMF: 1985.128192630896

The error for NMF is larger. NMF has a larger reconstruction residual error. This depends on how well the factorization captures the original data structure. A higher MSE indicates a larger difference between the original and reconstructed matrices. It depends on the specific characteristics of the data and how well each method captures its underlying structure. NMF has more constraints than LSI. The added constraints are W≥0 and H≥0. This reduces the degrees of freedom and makes the search space more restrictive. Whereas, LSI has a much larger search space to explore. It has more freedom to find the best vectors that minimize the error and the best vectors can include negative elements. This is not possible in NMF.

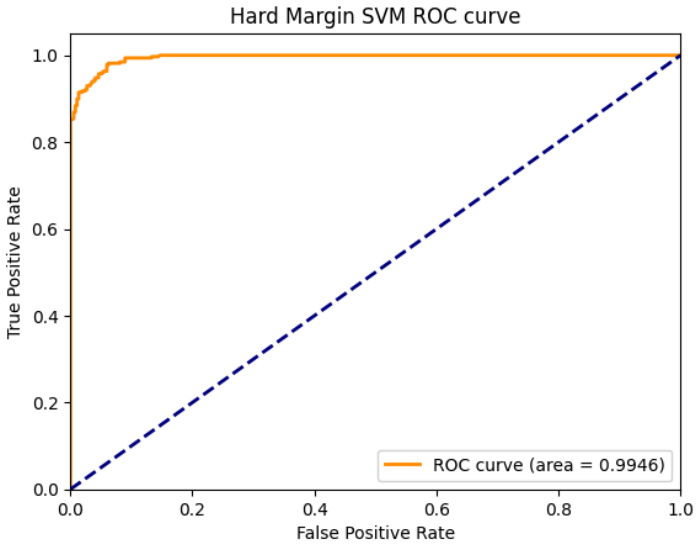
QUESTION 5:

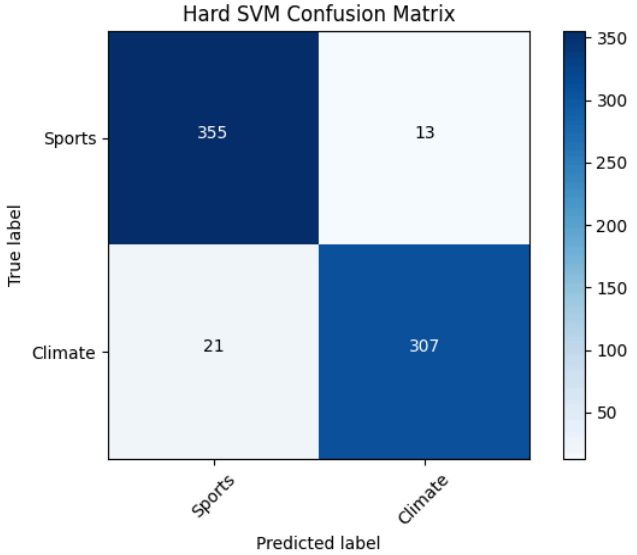
Compare and contrast hard-margin and soft-margin linear SVMs:
Train two linear SVMs:

– 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?

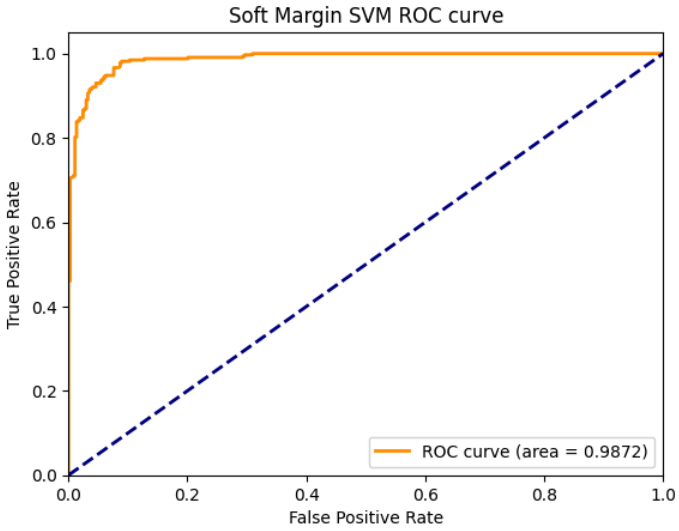
Hard Margin SVM(y=1000)

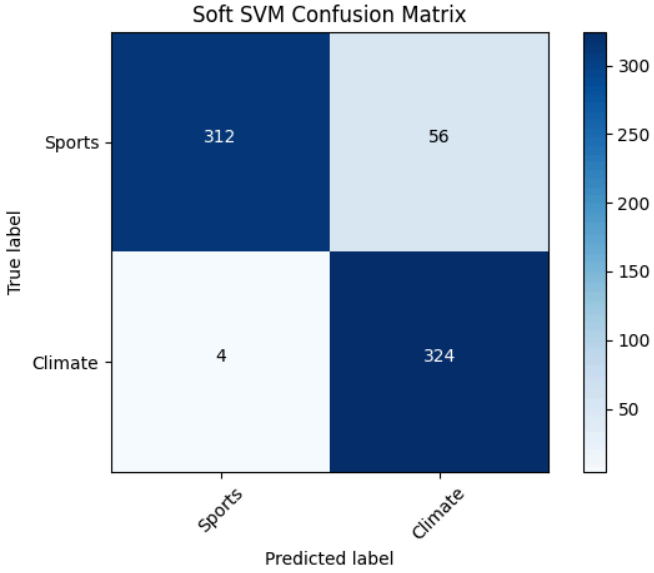




Accuracy score for Hard Margin SVM: 0.951149
Recall score for Hard Margin SVM: 0.935976
Precision score for Hard Margin SVM: 0.959375
F-1 score for Hard Margin SVM: 0.947531

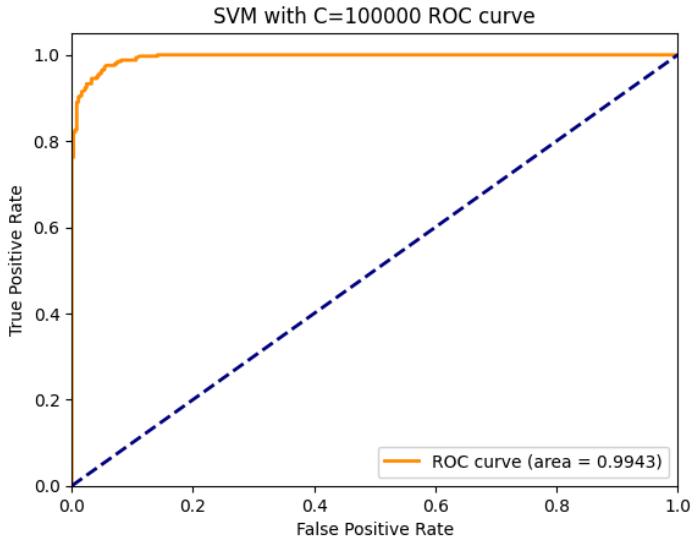
Soft Margin SVM (y=0.0001)

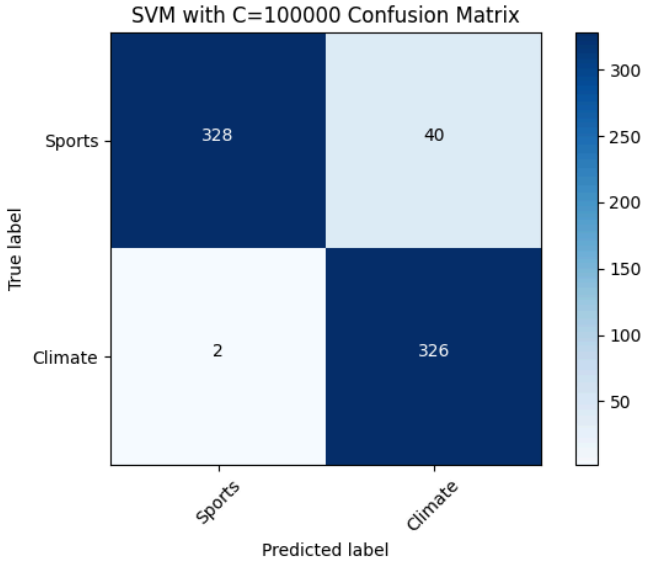




Accuracy score for Soft Margin SVM: 0.913793
Recall score for Soft Margin SVM: 0.987805
Precision score for Soft Margin SVM: 0.852632
F-1 score for Soft Margin SVM: 0.915254

SVM with y=100000





Accuracy score for SVM with C=100000: 0.939655
Recall score for SVM with C=100000: 0.993902
Precision score for SVM with C=100000: 0.890710
F-1 score for SVM with C=100000: 0.939481

Please refer to the graphs above for the ROC Curve, confusion matrix and metrics for SVMs with C=0.001,1000,100000.
From C=0.001 (soft-margin SVM) and C=1000 (hard-margin SVM), hard-margin SVM performs better. Hard-margin SVM has a higher accuracy score which means it correctly classifies a higher proportion of instances. It has a higher precision score which means it has a higher proportion of true positive predictions among instances predicted as positive. Hard-margin SVM has a higher F1-score as well. We can see that soft margin SVM under fits the data since it gives more scope for misclassification. For y=100000, it overfits the data and the accuracy drops. For hard margin SVM, it neither underfits nor overfits the data and hence that is the best choice.

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

We can see that soft margin SVM under fits the data since it gives more scope for misclassification. For y=100000, it overfits the data and the accuracy drops. Soft margin SVM has a low precision and hence it is likely that it is underfitting the data. Low precision indicates a high false positive rate.

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

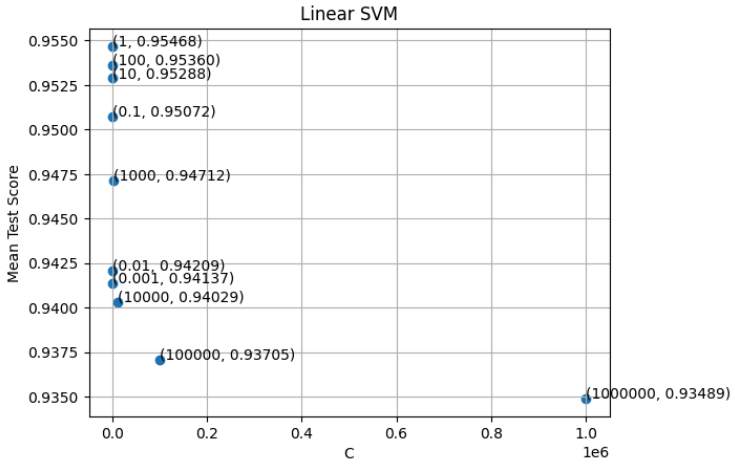
The soft margin SVM under fits the data. This is supported by the top left region of the ROC curve shown above. The ROC curve does not look competitive as the turn at the top left corner shows that it is underfitting the data and reflects the conclusion drawn.

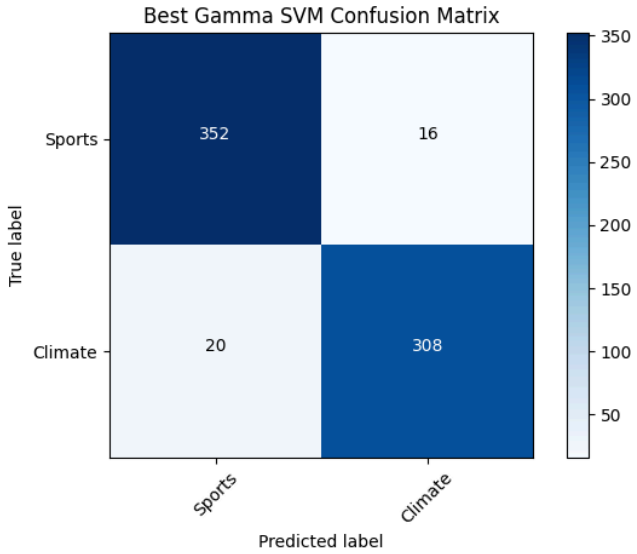
*** Use cross-validation to choose γ (use average validation 3 accuracy to compare): Using a 5-fold cross-validation, find the best value of the parameter γ in the range {10k| − 3 ≤ k ≤ 6, k ∈ Z}. Again,**

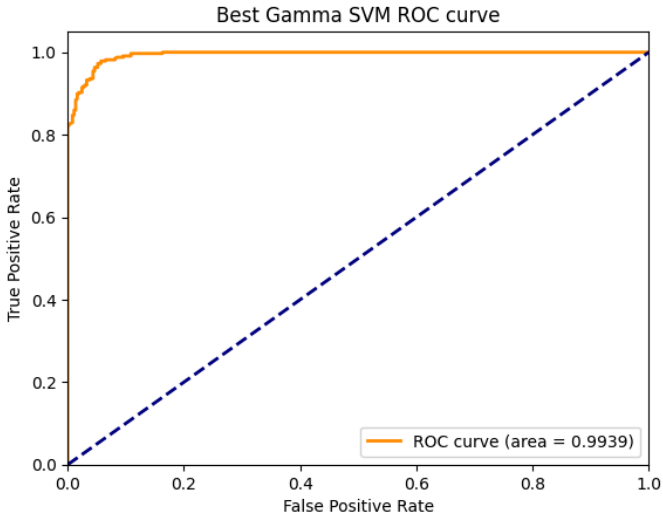
plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F-1 score of this best SVM.

The best value of y was found to be 1.

Best estimator for SVM: LinearSVC(C=1, random_state=42)
Best parameters for SVM: {'C': 1}
Best score for SVM: 0.9546762589928057
Best Gamma for SVM: 1
Accuracy score for Best Gamma SVM: 0.948276
Recall score for Best Gamma SVM: 0.939024
Precision score for Best Gamma SVM: 0.950617
F-1 score for Best Gamma SVM: 0.944785







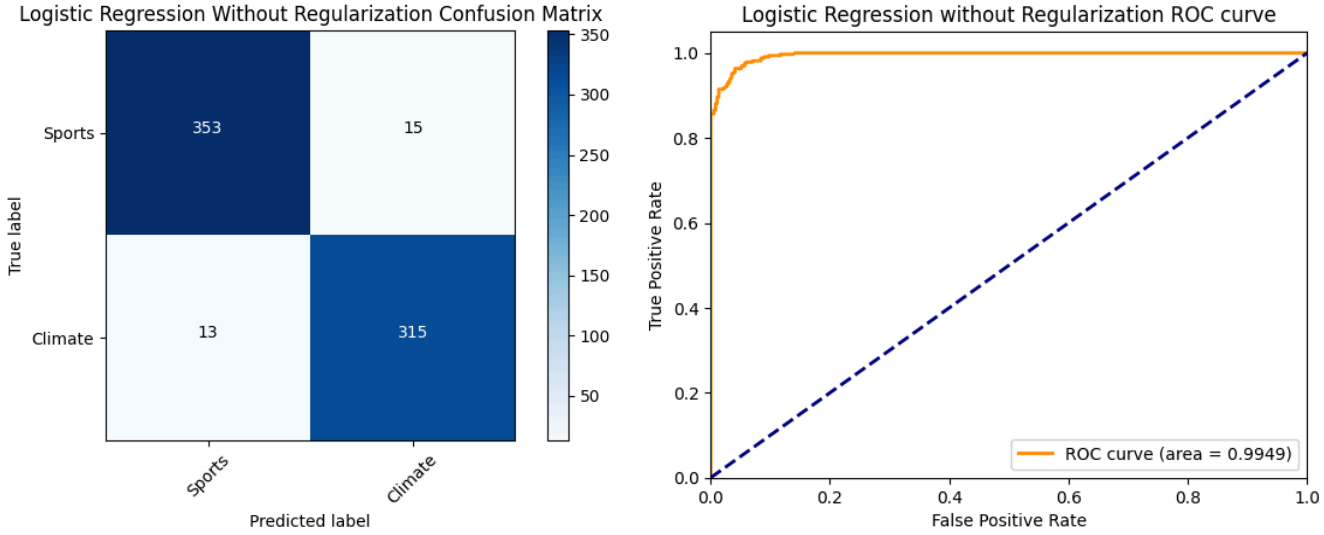
<u>QUESTION 6:</u>

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.

For logistic regression without regularization, the below metrics were found:
Accuracy score for Logistic Regression: 0.959770
Recall score for Logistic Regression: 0.960366
Precision score for Logistic Regression: 0.954545
F-1 score for Logistic Regression: 0.957447



The figures above show the confusion matrix and ROC curve.

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 {10k|−5 ≤ k ≤ 5, k ∈ Z} for logistic regression with L1 regularization and logistic regression with L2 regularization, respectively.

For L1 regularization - the optimal regularization strength is 100.
For L2 regularization - the optimal regularization strength is 1000.

– 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.

For Logistic Regression without regularization
Accuracy score for Logistic Regression: 0.959770
Recall score for Logistic Regression: 0.960366
Precision score for Logistic Regression: 0.954545
F-1 score for Logistic Regression: 0.957447

For Logistic Regression with L1 Regularization
Accuracy score for Logistic Regression with l-1 regularization: 0.958333
Recall score for Logistic Regression with l-1 regularization: 0.960366

Precision score for Logistic Regression with l-1 regularization: 0.951662
F-1 score for Logistic Regression with l-1 regularization: 0.955994

For Logistic Regression with L2 Regularization
Accuracy score for Logistic Regression with L2 regularization: 0.958333
Recall score for Logistic Regression with L2 regularization: 0.960366
Precision score for Logistic Regression with L2 regularization: 0.951662
F-1 score for Logistic Regression with L2 regularization: 0.955994

The accuracy scores for all three logistic classifiers are very close, with minimal differences. All models perform well in terms of overall accuracy. The recall scores, precision scores, and F-1 scores are also quite similar across the models. The differences are relatively small. The introduction of L1 or L2 regularization does not lead to significant changes in performance for this specific dataset and evaluation metric set. L1 regularization seemed to have a little higher accuracy and precision.

– 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?

The introduction of L1 or L2 regularization does not lead to significant changes in performance for this specific dataset and evaluation metric set. Generally, L1 regularization leads to logistic regression models with a higher accuracy and precision. L2 regularization prevents the model from overfitting. L2 regularization does not make any significant change in the test error for this dataset. A model with no regularization is normally used when we want to build complex models that follow the training data closely. L1 and L2 shrink the coefficients hence preventing overfitting. L1 regularization is good for feature selection as it can remove some features which are not important from the model. L2 regularization keeps all the features. Hence, L1 regularization is used when we only need to select a few important features of the dataset. Whereas we use L2 regularization when we want to avoid overfitting but also want to use all the features provided.

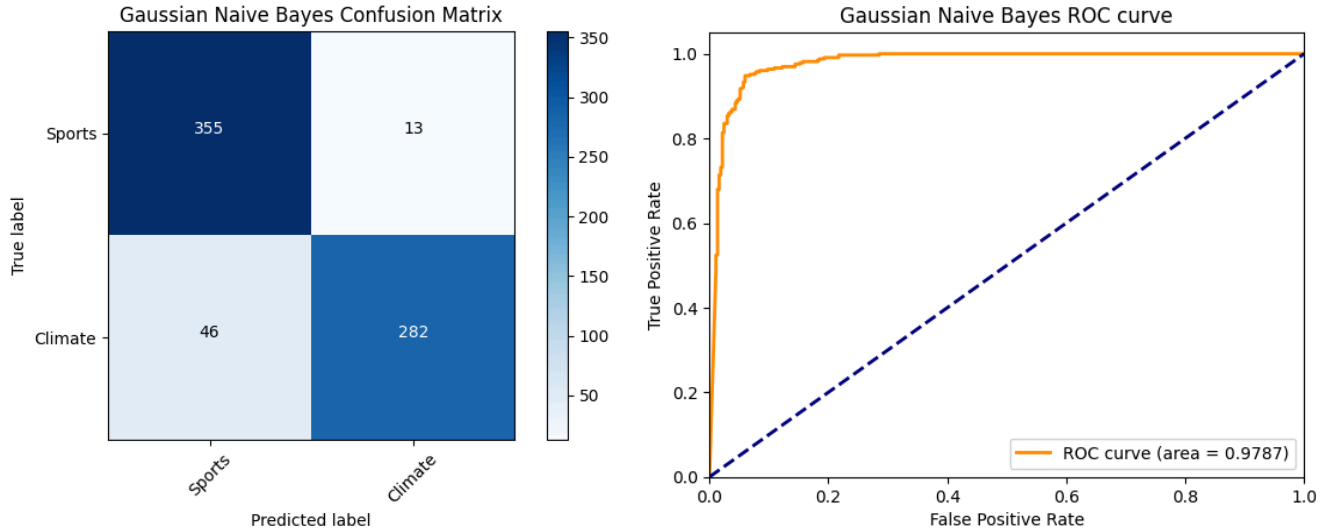
– Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary. What is the difference between their ways to find this boundary? Why do their performances differ? Is this difference statistically significant?

Logistic Regression models the probability that an instance belongs to a particular class. The decision boundary is determined by finding the weights (coefficients) that maximize the likelihood of the observed data under the logistic function. Logistic Regression uses the logistic loss function (also known as cross-entropy or log loss) to measure the difference between predicted probabilities and actual class labels. Whereas, Linear SVM aims to find a hyperplane that best separates the classes with the largest margin. The margin is the distance between the hyperplane and the nearest data points from each class. SVM uses a hinge loss function, which penalizes instances that are on the wrong side of the decision boundary. The objective is to minimize the hinge loss and maximize the margin. Logistic Regression minimizes the logistic loss, which penalizes deviations from true class probabilities. Linear SVM minimizes the hinge loss, which penalizes instances on the wrong side of the decision boundary. Linear SVM focuses on maximizing the margin, aiming for a clear separation between classes. Logistic Regression models the probabilities directly, aiming to model the likelihood of class membership. SVM is less sensitive to outliers due to the use of the hinge loss and the focus on maximizing the margin. Logistic Regression may be more influenced by outliers, especially if they affect the logistic loss. The choice between Logistic Regression

and Linear SVM depends on the characteristics of the data, interpretability, and the importance of robustness to outliers. SVM will perform better than Logistic Regression when we have high dimensional data and there are outliers.

QUESTION 7:

Evaluate and profile a Naıve 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.



Refer to the figures above for the ROC curve and confusion matrix.

Accuracy score for Gaussian Naive Bayes: 0.915230
Recall score for Gaussian Naive Bayes: 0.859756
Precision score for Gaussian Naive Bayes: 0.955932
F-1 score for Gaussian Naive Bayes: 0.905297

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).
• In addition to any other hyperparameters you choose, your gridsearch must at least include the table given in the pdf
• What are the 5 best combinations? Report their performances on the testing set.

Below are the best 5 combinations:

1. SVM(C=1) + LSI(k=80) + lemmatization(min_df=5)
2. Logistic Regression with L1 Regularization(C=100) + NMF(k=80) + lemmatization (min_df=5)
3. Logistic Regression with L2 Regularization(C=1000) + LSI(k=80) + stemming(min_df=3)

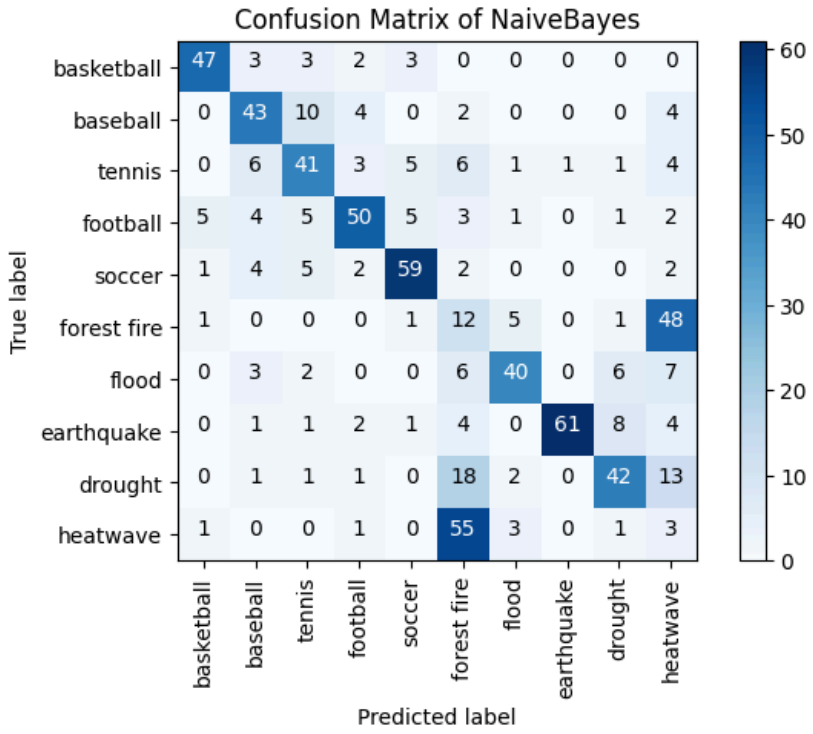
4. Logistic Regression with L2 Regularization(C=1000) + LSI(k=80) + lemmatization(min_df=3)
5. Logistic Regression with L1 Regularization(C=100) + LSI(k=80) + lemmatization (min_df=5)

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?

Naive Bayes Classification

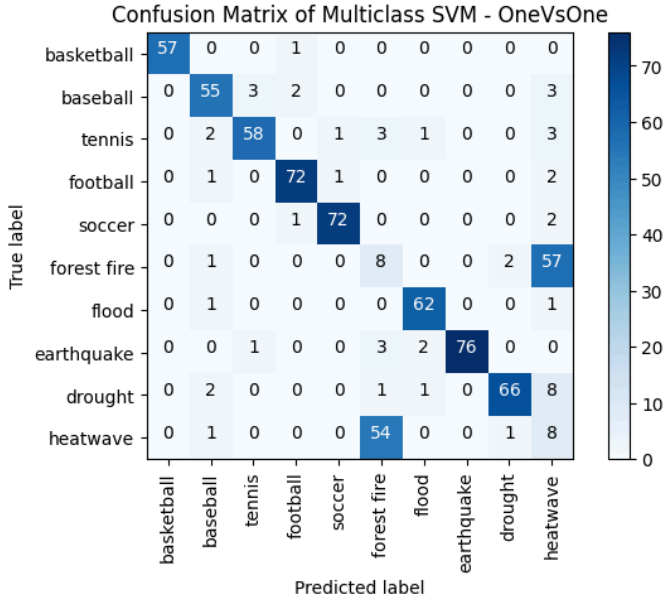
Multiclass NaiveBayes Accuracy Score: 0.5718390804597702
Multiclass NaiveBayes Recall Score: 0.5718390804597702
Multiclass NaiveBayes Precision Score: 0.5718390804597702
Multiclass NaiveBayes f1 Score: 0.5718390804597702



Multiclass SVM (One vs One):

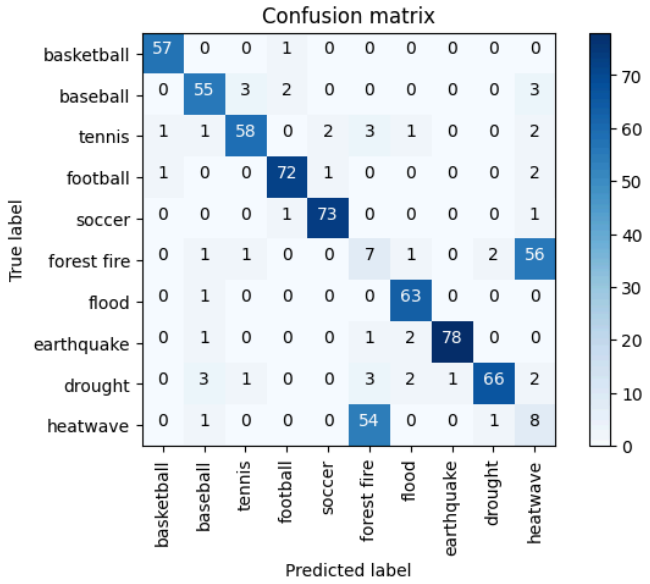
Multiclass SVM OneVsOne Accuracy Score: 0.7672413793103449
Multiclass SVM OneVsOne Recall Score: 0.7672413793103449
Multiclass SVM OneVsOne Precision Score: 0.7672413793103449

Multiclass SVM OneVsOne f1 Score: 0.7672413793103448



Multiclass SVM (One vs Rest):

Multiclass SVM OneVsRest Accuracy Score: 0.771551724137931
Multiclass SVM OneVsRest Recall Score: 0.771551724137931
Multiclass SVM OneVsRest Precision Score: 0.771551724137931
Multiclass SVM OneVsRest f1 Score: 0.771551724137931



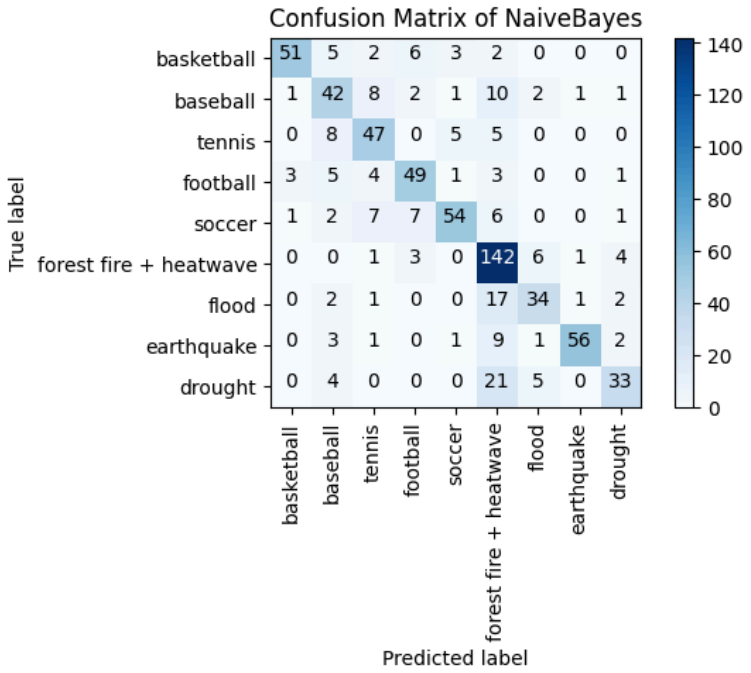
In addition, answer the following questions:
• In the confusion matrix you should have a 10 × 10 matrix where 10 is the number of unique labels in the column leaf label. Please make sure that the order of these labels is as follows: map_row_to_class = {0:"basketball", 1:"baseball", 2:"tennis", 3:"football", 4:"soccer", 5:"forest fire", 6:"flood", 7:"earthquake", 8:"drought", 9:"heatwave"}

Do you observe any structure in the confusion matrix? Are there distinct visible blocks on the major diagonal? What does this mean?
We can see that the class leaf_label is a little imbalanced.From all the results, we see that One vs Rest SVM classifier performs the best. From the confusion matrix, we can see that we have a distinct diagonal. The diagonal elements of a confusion matrix show the labels that are predicted correctly. The off diagonal elements are the misclassified labels. Except for forest fire and heatwave, we can see that the remaining have a perfect diagonal. The distinct visible diagonals in the block leads to a higher accuracy.

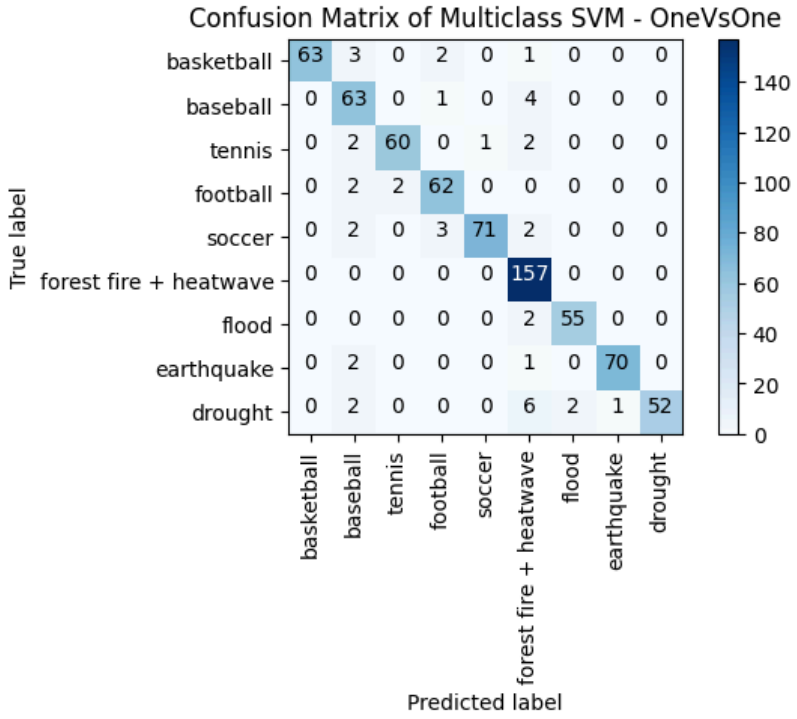
• Based on your observation from the previous part, suggest a subset of labels that should be merged into a new larger label and recompute the accuracy and plot the confusion matrix. How did the accuracy change in One VS One and One VS the rest?

We can see that forest fire and heatwave do not have a perfect diagonal meaning there are more misclassified labels. Merging forest fire and heatwave into a single label and checking gives a higher accuracy for all 3 models.

Naive Bayes:
Multiclass NaiveBayes with leaf labels merged Accuracy Score: 0.7298850574712644
Multiclass NaiveBayes with leaf labels merged Recall Score: 0.7298850574712644
Multiclass NaiveBayes with leaf labels merged Precision Score: 0.7298850574712644
Multiclass NaiveBayes with leaf labels merged f1 Score: 0.7298850574712644



Multiclass SVM (One vs One):
Multiclass SVM OneVsOne with leaf labels merged Accuracy Score: 0.9382183908045977
Multiclass SVM OneVsOne with leaf labels merged Recall Score: 0.9382183908045977
Multiclass SVM OneVsOne with leaf labels merged Precision Score: 0.9382183908045977
Multiclass SVM OneVsOne with leaf labels merged f1 Score: 0.9382183908045977



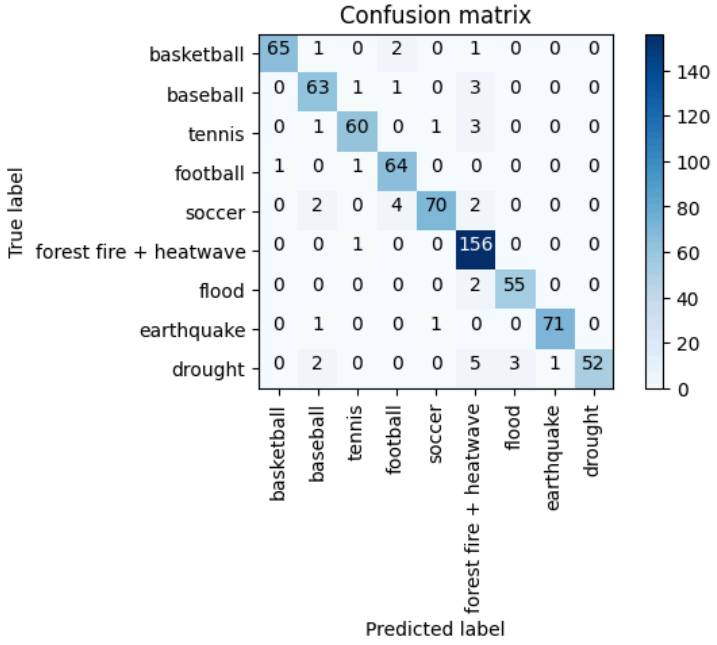
Multiclass SVM (One vs Rest):

Multiclass SVM OneVsRest with leaf labels merged Accuracy Score: 0.9425287356321839

Multiclass SVM OneVsRest with leaf labels merged Recall Score: 0.9425287356321839

Multiclass SVM OneVsRest with leaf labels merged Precision Score: 0.9425287356321839

Multiclass SVM OneVsRest with leaf labels merged f1 Score: 0.9425287356321839

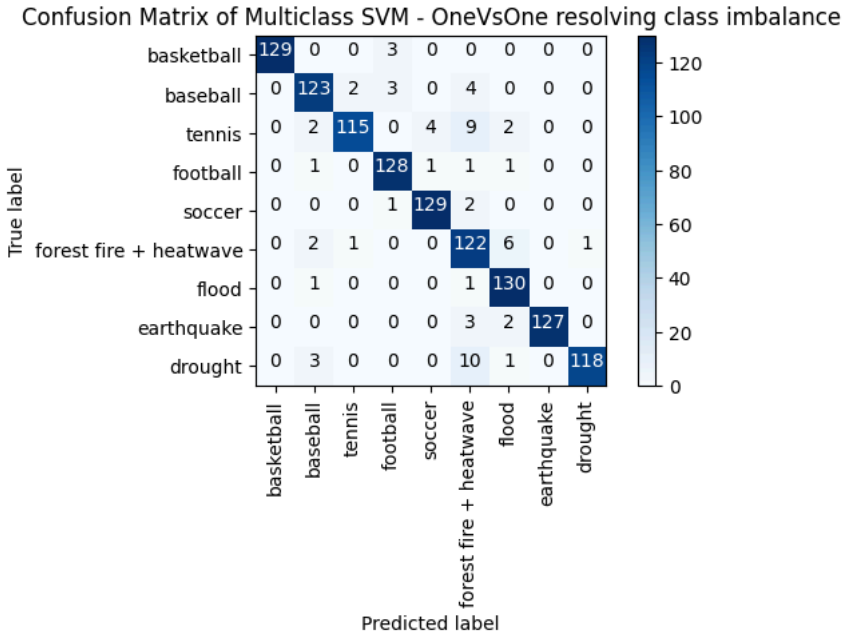


For all 3 models i.e Naive Bayes, OneVsOne SVM and OneVsRest SVM, the accuracy increased drastically and we get a perfect diagonal in the confusion matrix.

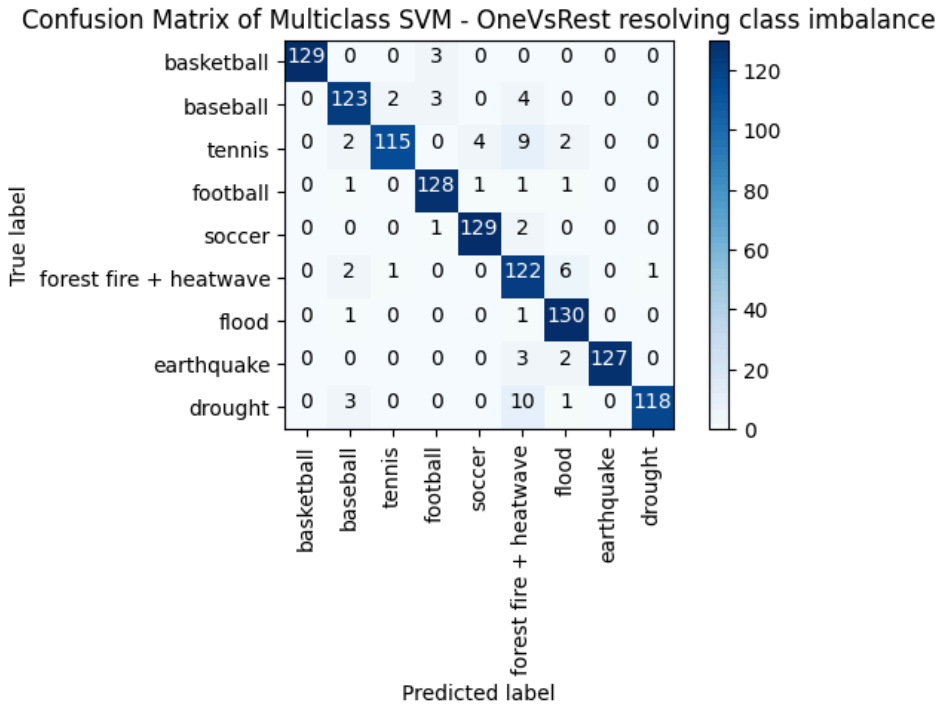
• Does class imbalance impact the performance of the classification once some classes are merged? Provide a resolution for the class imbalance and recompute the accuracy and plot the confusion matrix in One VS One and One VS the rest?.

Yes the class imbalance affects the classification since we have almost twice the number of data points as compared to the rest of the leaf labels. The solution to this is either double the samples for the remaining leaf labels or then reduce the sample size to half for this new merged leaf label (forest fire + heatwave). In this way we will be able to deal with the class imbalance. Using SMOTE can help resolve this class imbalance issue.

Multiclass SVM (One vs One):
Multiclass SVM OneVsOne with leaf labels merged and resolving class imbalance Accuracy Score: 0.9436026936026936
Multiclass SVM OneVsOne with leaf labels merged and resolving class imbalance Recall Score: 0.9436026936026936
Multiclass SVM OneVsOne with leaf labels merged and resolving class imbalance Precision Score: 0.9436026936026936
Multiclass SVM OneVsOne with leaf labels merged and resolving class imbalance f1 Score: 0.9436026936026936



Multiclass SVM (One vs Rest):
Multiclass SVM OneVsRest with leaf labels merged and resolving class imbalance Accuracy Score: 0.9461279461279462
Multiclass SVM OneVsRest with leaf labels merged and resolving class imbalance Recall Score: 0.9461279461279462
Multiclass SVM OneVsRest with leaf labels merged and resolving class imbalance Precision Score: 0.9461279461279462
Multiclass SVM OneVsRest with leaf labels merged and resolving class imbalance f1 Score: 0.9461279461279462



We can see that after dealing with the issue of class imbalance, the accuracy has increased for both OveVsOne SVM and OneVsRest SVM.

QUESTION 10:

Read the paper about GLoVE embeddings - found here and answer the following subquestions:

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

GLoVE embeddings are trained on the ratio of co-occurrence probabilities rather than the probabilities themselves because the ratio of probabilities has some advantages in capturing meaningful relationships between words. GloVe tries to capture the relation between words based on their co-occurrence probabilities. Using these ratios of probabilities, the model emphasizes relative comparisons between the probabilities of different word pairs rather than the absolute values. Using these ratios helps mitigate the impact of word frequency on the embeddings. The ratio formulation inherently normalizes the co-occurrence information. It captures semantic relationships between words. It captures meaningful semantic relationships between words while addressing challenges related to frequency effects, normalization, and sparsity.

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

No, we will not get the same vector for the word running in both the cases. This is because while forming the vector, GLoVE takes into consideration the neighboring words i.e the contextual dependencies of the

words in the sentence. Here, park and presidency are two different neighboring words and hence we get different vectors.

(c) 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.

Difference between 'woman' and 'man': 4.753939628601074
Similarity between 'wife' and 'husband': 0.8646390438079834

Difference between 'wife' and 'husband': 3.1520464420318604
Similarity between 'wife' and 'orange': 0.1162913516163826

Similarity between 'wife' and 'orange': 0.1162913516163826
Difference between 'wife' and 'orange': 8.667715072631836

We can see that from the three pairs above, wife and husband have the least difference i.e they are the most similar. This is followed by woman and man. Wife and orange have a high difference which means they are not very similar. Wife and husband have a strong semantic relationship. This is due to the spousal and gender relationship. Man and woman have a gender relationship. Wife and orange have a larger norm and are not related.

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

Generally, lemmatization is preferred over stemming. This is because lemmatization takes into consideration context, syntax, semantics and POS tokens and hence we get meaningful words. Whereas in stemming, it is not necessary that we get meaningful words. Although stemming is faster and computationally efficient, it has a higher error rate. Therefore, it is preferred to choose lemmatization for a broader, more meaningful words before mapping it to its GLoVE embedding.

QUESTION 11:

For the binary classification task distinguishing the “sports” class and “climate” class:
(a) Describe a feature engineering process that uses GLoVE word embeddings to represent each document. You have to abide by the following rules:
• A representation of a text segment needs to have a vector dimension that CANNOT exceed the dimension of the GLoVE embedding used per word of the segment.
• You cannot use TF-IDF scores (or any measure that requires looking at the complete dataset) as a pre-processing routine.
• Important: In this section, feel free to use raw features from any column in the original data file not just full text. The column keywords might be useful... or not. Make sure that your result achieves an accuracy of at least 92%.
• To aggregate these words into a single vector consider normalization the vectors, averaging across the vectors.

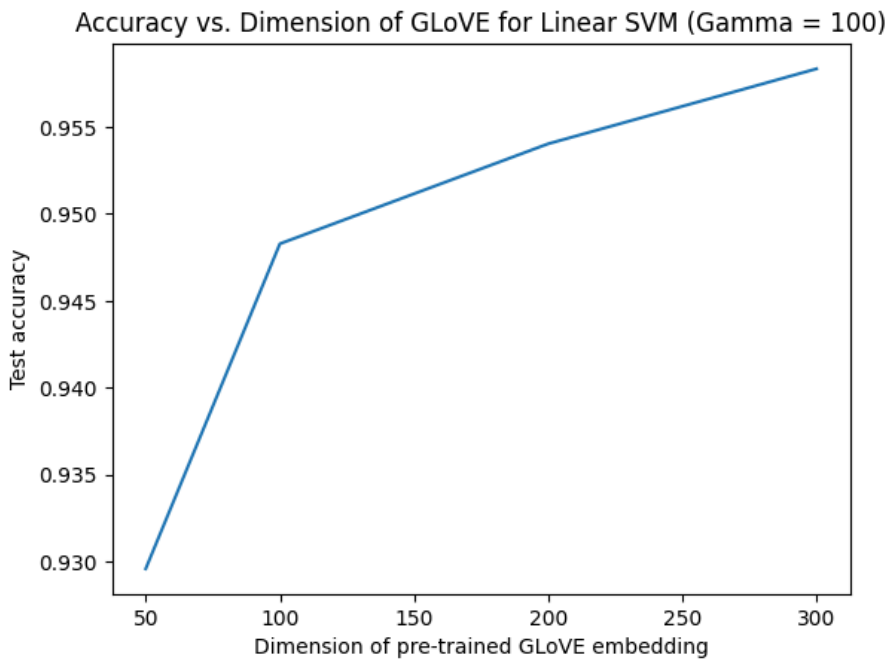
We convert the GLoVE word embeddings to a word2vec file by using a keyed vector model from Gensim. The words are then stored in embedded vector format. The data is cleaned and we remove the punctuations, digits and stopwords before lemmatizing it. Then each sentence is converted into a vector of size 300 as mentioned in the question. We then need to check if all the features which are there in the vector are based in the pretrained GLoVE embedding file. Then the vectors are normalized and aggregated into a single vector.

(b) Select a classifier model, train and evaluate it with your GLoVE-based feature. If you are doing any cross-validation, please make sure to use a limited set of options so that your code finishes running in a reasonable amount of time.

We chose a linear SVM and tried cross validation to find the value of C which is the best. Below were the metrics obtained.
LinearSVC(C=100, random_state=42)
Accuracy (Best GLoVE classifier): 0.9583333333333334
Recall (Best GLoVE classifier): 0.9695121951219512
Precision (Best GLoVE classifier): 0.9436201780415431
F1-Score (Best GLoVE classifier): 0.9563909774436089

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? In this part use the different sets of GLoVE vectors from the link.

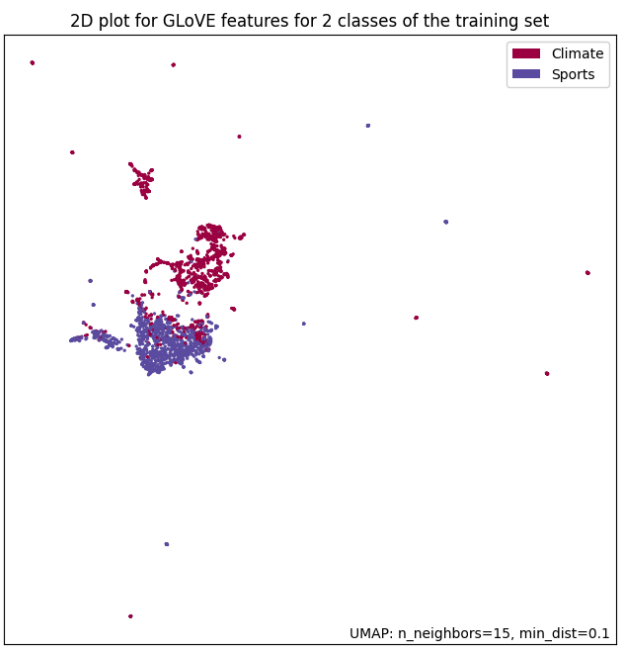


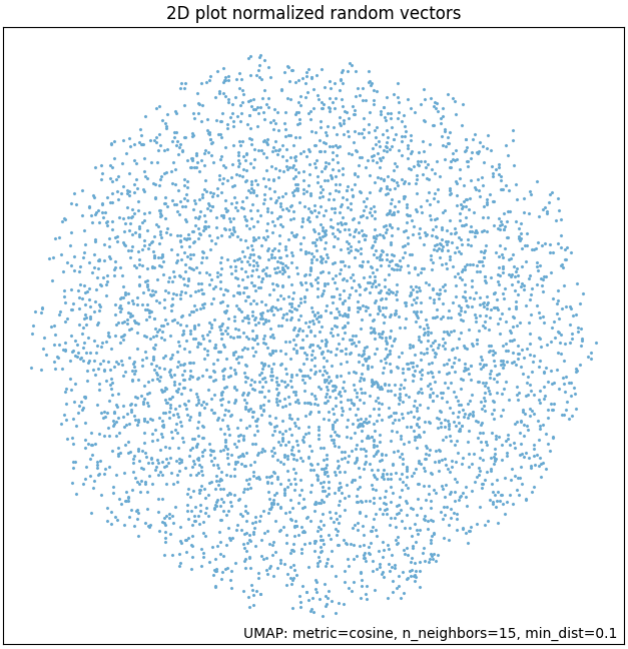
The figure above shows the graph between the dimensions of pretrained GLoVE embeddings and the test accuracy. The trend that can be observed is that as the dimensions increase, the accuracy on the test data

increases. More dimensions mean that there are more semantics and more information. Hence a better relation between target words and context words can be found. There we have better feature dependencies and hence more accuracy.

QUESTION 13:

Compare and contrast the two visualizations. Are there clusters formed in either or both of the plots? We will pursue the clustering aspect further in the next project.





We can observe that for the GLoVE model, distinct clusters are formed. Whereas for the random vectors, we cannot see separate clusters. Instead, we can observe one big cluster. This proves that GLoVE is able to accurately distinguish between the two dataset features i.e climate and sports. This shows successful learning of the classifier and that all the points are evenly distributed.