EC ENGR 219

2023 Winter

Project 1:

End-to-End Pipeline to Classify News Articles

Wenxin Cheng	706070535	wenxin0319@g.ucla.edu
Yuxin Yin	606073780	yyxyy999@g.ucla.edu
Yingqian Zhao	306071513	zhaoyq99@g.ucla.edu

- **QUESTION 1**

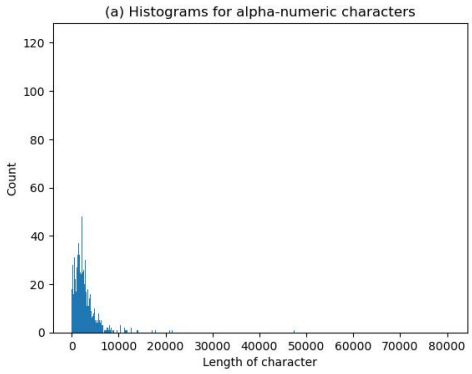
How many rows (samples) and columns (features) are present in the dataset?

Number of rows: **3150**

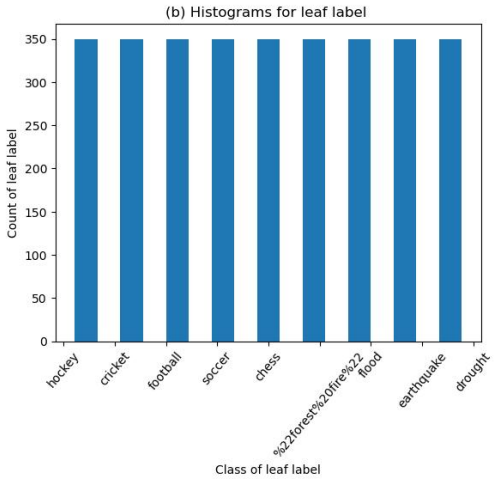
Number of columns: **8**

Histograms: Plot 3 histograms on :

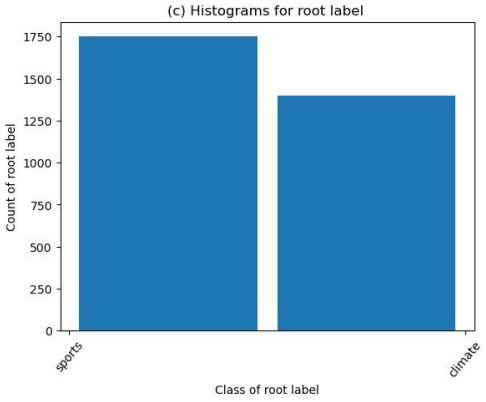
(a) The total number of alpha-numeric characters per data point (row) in the feature full text: i.e count on the x-axis and frequency on the y-axis;



(b) The column leaf label – class on the x-axis;



(c) The column root label – class on the x-axis.



- **QUESTION 2**

Report the number of training and testing samples.

Number of Training Samples: **2520**

Number of Testing Samples: **630**

- **QUESTION 3**

Use the following specs to extract features from the textual data:

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

Lemmatization uses a vocabulary and morphological analysis to reduce words to their base form. This method is **more accurate** than stemming, but also **more computationally expensive**. Lemmatization also tends to produce **more readable results**, since it converts words to their base forms while still maintaining their original meaning.

Stemming, on the other hand, uses simple heuristics to chop off the ends of words in an attempt to reduce them to their base forms. This method is less accurate than lemmatization, but also **less computationally expensive**. Stemming tends to produce **less readable results**, since it can chop off the ends of words in a way that changes the original meaning.

The dictionary size will be **smaller after stemming process**, because it reduces the words to their root form and remove the suffixes from the words. In contrast, **Lemmatization will not reduce the dictionary size as much**, because it tries to maintain the meaning of the words by reducing them to their base form.

• How does varying minimum document frequency change the TF-IDF matrix?

Varying the min_df parameter when creating a TF-IDF matrix will affect the number of terms that are included in the matrix.

A lower min_df value will include more terms in the matrix, while a higher min_df value will exclude less common terms from the matrix. This means that if we set min_df to a high value, it will exclude rare words from the matrix, which will make the matrix smaller and the computations faster. However, it also means that the rare words that could contain important information for the task will be excluded.

Overall, the effect of varying the min_df parameter on a TF-IDF matrix is a trade-off between the size and computational efficiency of the matrix and the amount of information included in the matrix.

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

Stopwords, punctuations and numbers should **be removed before lemmatizing**.

Punctuation, numbers, and stopwords should be handled by tokenization process which is done before the lemmatization, as the lemmatizer need the context of the sentence. The lemmatizer uses the context of the sentence and the POS tagging of the words to determine the correct base form of a word. So removing punctuations, numbers and stopwords before lemmatizing will not affect the accuracy of the lemmatization and will make the process more efficient.

• 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 $k \times 10^3$. This dimension will vary depending on your exact method of cleaning and lemmatizing and that is okay.

when min_df = 1
train matrix shape: (2520, 34769)
test matrix shape: (630, 34769)

when min_df = 2
train matrix shape: (2520, 19362)
test matrix shape: (630, 19362)

when min_df = 3
train matrix shape: (2520, 14076)
test matrix shape: (630, 14076)

when min_df = 4
train matrix shape: (2520, 11330)
test matrix shape: (630, 11330)

when min_df = 5
train matrix shape: (2520, 9644)
test matrix shape: (630, 9644)

when min_df = 6
train matrix shape: (2520, 8565)
test matrix shape: (630, 8565)

when min_df = 7
train matrix shape: (2520, 7412)
test matrix shape: (630, 7412)

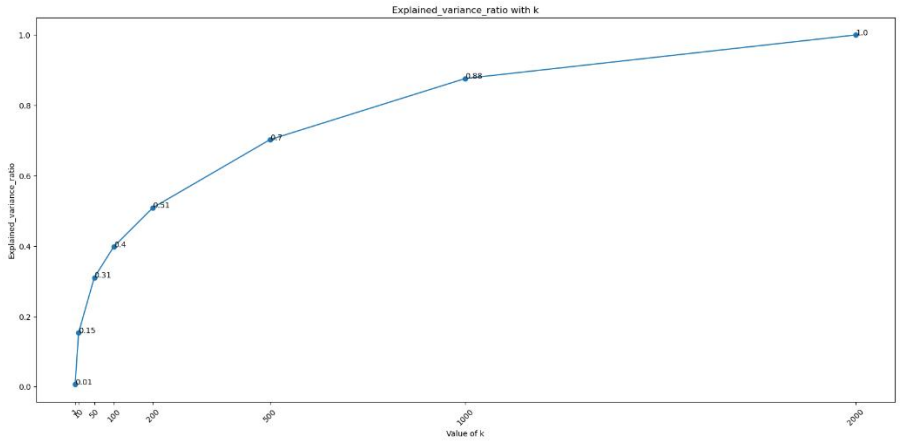
when min_df = 8
train matrix shape: (2520, 6747)
test matrix shape: (630, 6747)

when min_df = 9
train matrix shape: (2520, 6219)
test matrix shape: (630, 6219)

- **QUESTION 4**

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?



As we increase the number of components, the explained variance ratio will generally increase as well, meaning that more variance in the data is explained by the components. The curve will reach a point of diminishing returns, where the explained variance ratio plateaus or starts to decrease after a certain point, regardless of the number of components.

The concavity of the plot suggests that the explained variance ratio increases at a decreasing rate as k increases. In other words, as we increase the number of components, the explained variance ratio will increase at a slower rate.

For the LSI, when k = 50, the explained variance ratio is the highest among all the k values considered. So, this would be the optimal value for k, as it strikes a balance between explained variance ratio and computational complexity.

• 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 and why?

MSE Error for LSI is **41.02637078289104**

MSE Error for NMF is **41.37286730025996**

MSE Error for NMF is **larger.**

When using LSI with k = 50, the reconstruction residual MSE error will be the error obtained from reconstructing the original data matrix from the reduced-dimensional LSI matrix. Similarly, when using NMF with k = 50, the reconstruction residual MSE error will be the error obtained from reconstructing the original data matrix from the reduced-dimensional NMF matrix.

Because the reconstruction residual MSE Error when using LSI and NMF are given in the above table. NMF calculates how well each document fits each topic, rather than

assuming a document has multiple topics, due to which it works better with shorter texts like tweets.

● 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 γ= 1000:

accuracy= 96.67

recall= 97.28

precision= 97.01

f1= 97.14

Hard margin svm γ=100000:

accuracy= 90.32

recall= 100.00

precision= 85.75

f1= 92.33

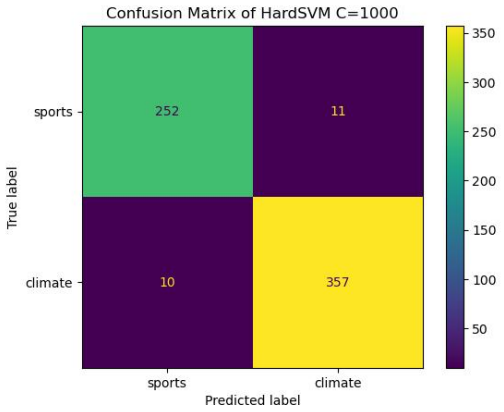
Soft margin svm γ=0.0001:

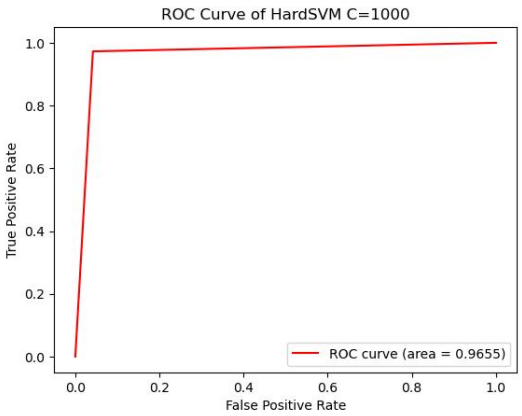
accuracy= 58.25

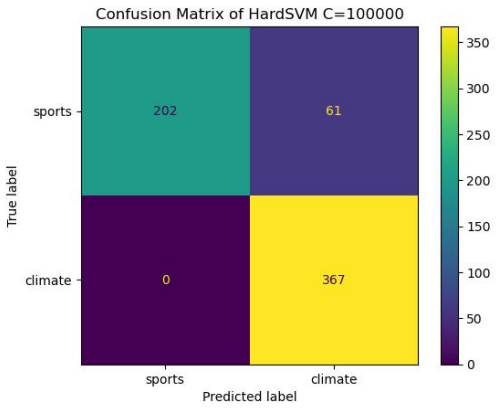
recall= 100.00

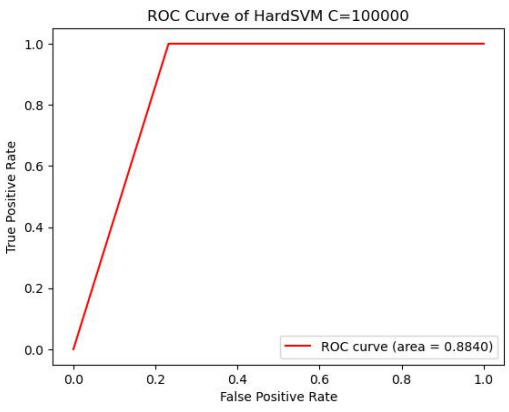
precision= 58.25

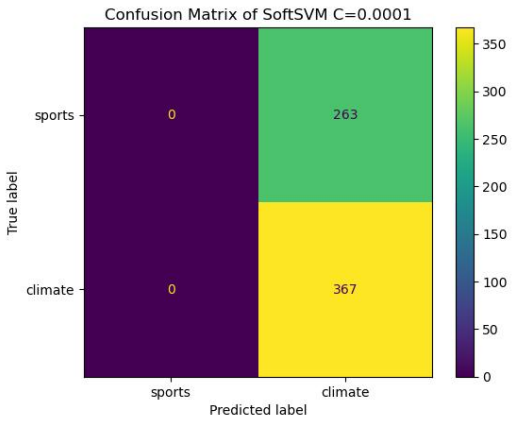
f1= 73.62

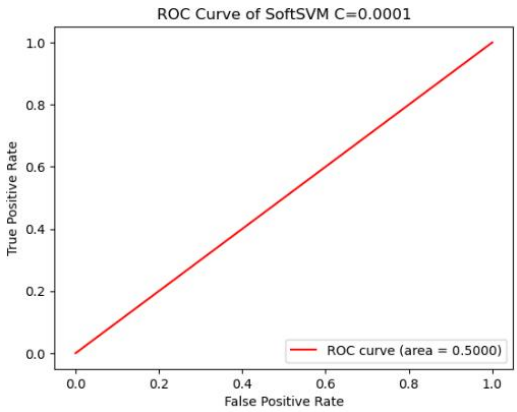












The best γ is 1000.

For γ = 100000, the SVM classifier is more likely to overfit the training data, leading to a poor performance on the testing set. This is because a high value of gamma implies a smaller radius of the Gaussian function, which in turn means a more complex decision boundary. This may lead to capturing noise in the training data, resulting in poor generalization performance on unseen data.

– 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?

In a soft margin SVM, the objective is to minimize the misclassification error while also allowing for some misclassifications. This is done by introducing a slack variable for each sample that measures the degree of misclassification. A penalty parameter C is used to control the trade-off between maximizing the margin and minimizing the misclassification error.

In terms of the confusion matrix, a soft-margin SVM allows for some misclassifications which can be reflected in the entries of the matrix. In particular, a larger value of C will result in a smaller number of misclassifications, as a larger penalty is placed on misclassifications, while a smaller value of C will result in a larger number of misclassifications as the penalty for misclassifications is smaller.

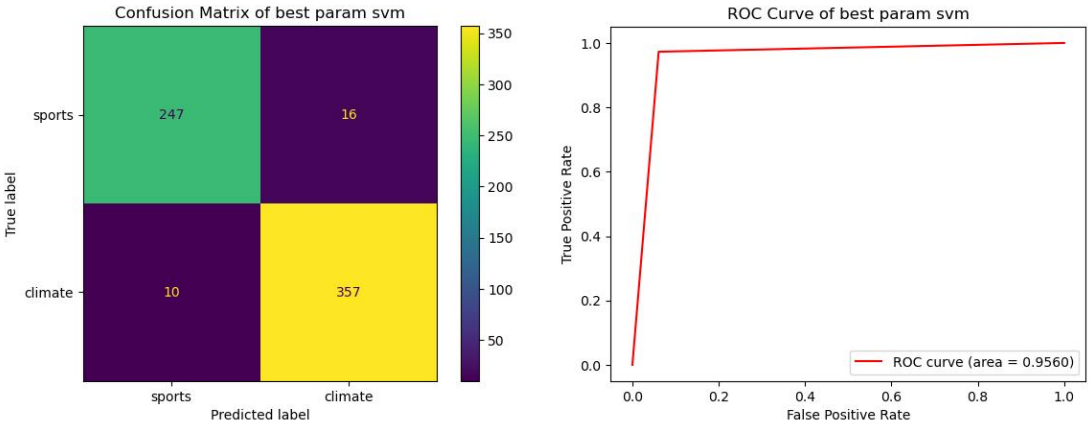
• Use cross-validation to choose γ (use average validation 3 accuracy to compare): Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10^k \mid -3 \leq k \leq 6, k \in Z\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall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

γ= 1000, mean_test_score = 95.39683
γ= 10000, mean_test_score = 95.39683
γ= 100000, mean_test_score = 95.39683
γ= 1000000, mean_test_score = 95.39683
γ= 100, mean_test_score = 95.19841
γ= 1, mean_test_score = 94.64286
γ= 10, mean_test_score = 94.64286

γ= 0.1, mean_test_score = 93.88889
γ= 0.01, mean_test_score = 91.94444
γ= 0.001, mean_test_score = 66.90476

Result for the best param svm:

accuracy= 95.87
recall= 97.28
precision= 95.71
f1= 96.49



- **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.

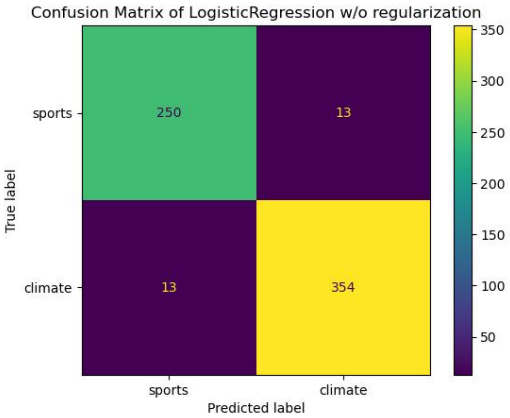
Logistic Regression without regularization:

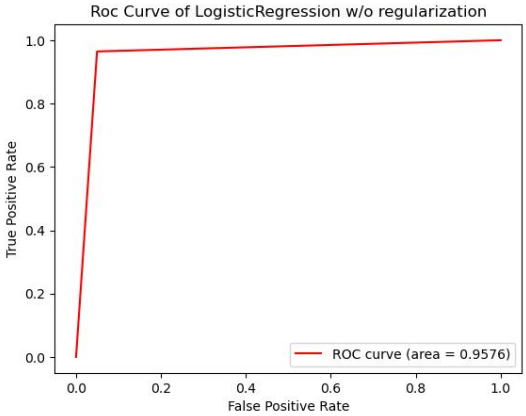
accuracy= 95.87

recall= 96.46

precision= 96.46

f1= 96.46





• 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.

γ= 100, mean_test_score = 95.59524

γ= 1000, mean_test_score = 95.59524

γ= 10000, mean_test_score = 95.55556

γ= 100000, mean_test_score = 95.55556

γ= 10, mean_test_score = 95.03968

γ= 1, mean_test_score = 94.40476

γ= 0.1, mean_test_score = 92.30159

γ= 1e-05, mean_test_score = 45.11905

γ= 0.0001, mean_test_score = 45.11905

γ= 0.001, mean_test_score = 45.11905

γ= 0.01, mean_test_score = 45.11905

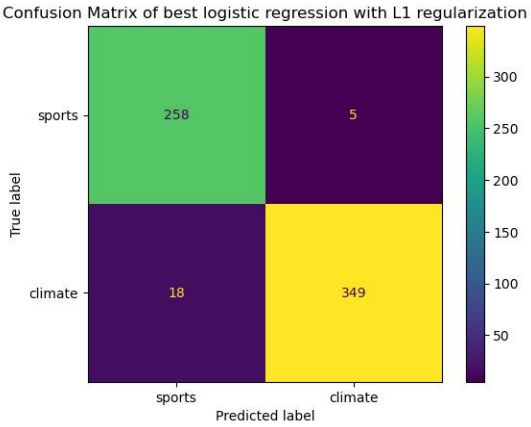
Result for the best logistic regression with L1 regularization:

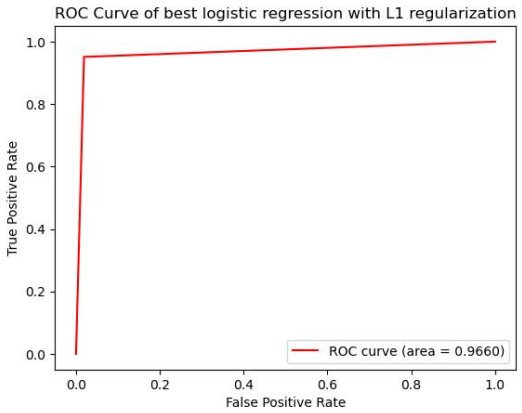
accuracy= 96.35

recall= 95.10

precision= 98.59

f1= 96.81

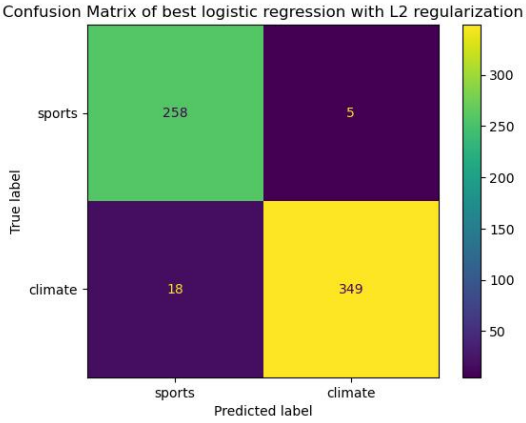


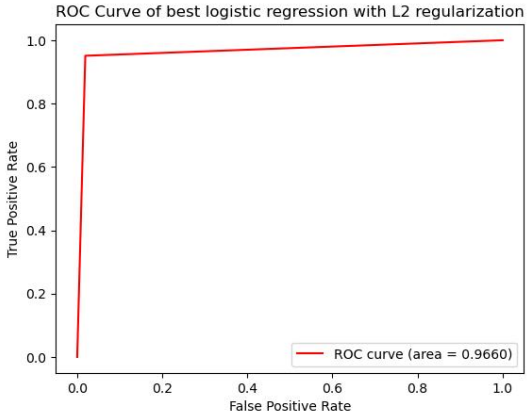


γ= 10000, mean_test_score = 95.59524
γ= 100000, mean_test_score = 95.55556
γ= 1000, mean_test_score = 95.39683
γ= 100, mean_test_score = 94.68254
γ= 10, mean_test_score = 94.68254
γ= 1, mean_test_score = 94.12698
γ= 0.1, mean_test_score = 92.22222
γ= 0.01, mean_test_score = 72.7381
γ= 1e-05, mean_test_score = 54.88095
γ= 0.0001, mean_test_score = 54.88095
γ= 0.001, mean_test_score = 54.88095

Result for the best logistic regression with L2 regularization:

accuracy= 96.35
recall= 95.10
precision= 98.59
f1= 96.81





– 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.

We can find that Logistic Regression with L1 Regularization is having a slight increase

in performance compared to the case without Regularization.

– 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 regularization parameter affects the test error by controlling the trade-off between fitting the training data and preventing overfitting. A larger value of the regularization parameter will result in a smaller model complexity and a lower test error, while a smaller value will result in a larger model complexity and a higher test error.

One might be interested in using L1 regularization when they are working with a high-dimensional dataset and want to reduce the number of features in the model. L1 regularization can be useful in feature selection as it can shrink the weights of less important features towards zero, effectively removing them from the model. On the other hand, L2 regularization is useful when the goal is to improve the generalization performance of the model without necessarily removing any features.

– 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?

They differ in the way they find this boundary and the criteria used to evaluate the performance of the model.

Logistic regression finds the decision boundary by maximizing the likelihood of the training data, under the assumption that the data follows a logistic distribution. It uses the maximum likelihood principle to estimate the model parameters and finds a linear boundary that separates the data points into different classes. The performance of logistic regression is evaluated using the likelihood ratio test or the deviance test.

Linear SVM, on the other hand, finds the decision boundary by maximizing the margin between the classes, which is the distance between the closest data points from different classes and the decision boundary. The optimization problem in linear SVM is a convex quadratic program and it is solved by Quadratic Programming (QP) algorithm. The performance of linear SVM is evaluated using the hinge loss function.

- **QUESTION 7**

Evaluate and profile a Na¨ıve 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.

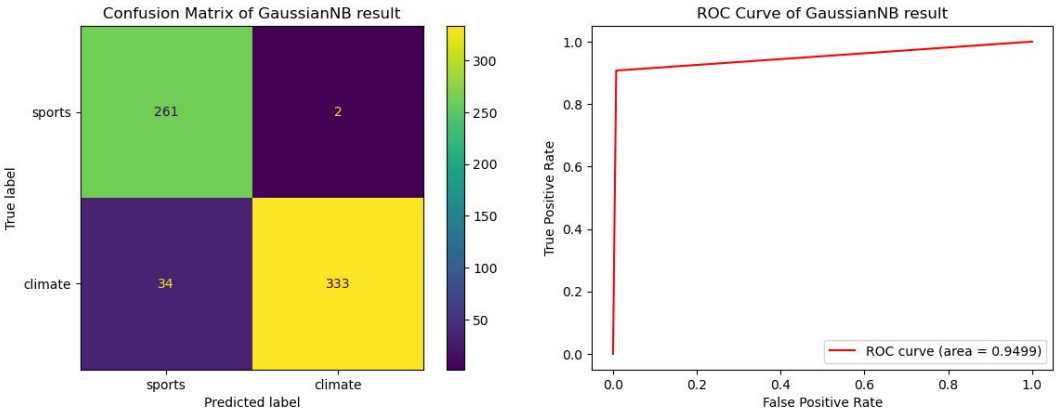
GaussianNB result:

accuracy= 94.29

recall= 90.74

precision= 99.40

f1= 94.87



- **QUESTION 8**

• Construct a Pipeline that performs feature extraction, dimensionality reduction and classification.

step1
Loading data whether clean data or not
step2
Compare four circumstances of min_df = 3 vs 5
lemmatized vs stemming
step3
Compare six circumstances of LSI (k = [5, 30, 80]) vs NMF (k = [5, 30, 80])
step4
SVM with the best γ previously found
vs
Logistic Regression: L1 regularization vs L2 regularization,
with the best regularization strength previously found
vs
GaussianNB
Memory set
Create a temporary folder to store the transformers of the pipeline

• The evaluation of each combination is performed with 5-fold cross-validation (use the average validation set accuracy across folds).

```
location = "cachedir"
memory = Memory(location=location, verbose=10)

cached_pipe = Pipeline(
    [
        ("countvector", CountVectorizer(stop_words='english')),
        ("tfidf", TfidfTransformer(use_idf = True)),
        ("reduce_dim", "passthrough"),
        ("classifier", None)
    ],
    memory=memory,
)

N_FEATURES_OPTIONS = [5, 30, 80]
param_grid = [
    {
        "countvector__min_df": (3,5),
        "reduce_dim": [TruncatedSVD(random_state=42), NMF(random_state=42,max_iter=10000)],
        "reduce_dim__n_components": N_FEATURES_OPTIONS,
        'classifier': (
```

```
                LinearSVC(C=grid.best_params_['classify__C'],random_state=42),
                LogisticRegression(penalty='l1',C=wl1_grid.best_params_['C'],random_state=42,solver='liblinear'),
                LogisticRegression(penalty='l2',C=wl2_grid.best_params_['C'],random_state=42,solver='liblinear'),
                GaussianNB(),
            ),
        }
    ]
```

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

Hence, the combination results are shown below:

	Classifier	Reduce_dim	Analyzer	Min_df	Cleaning	Test Score
1	**LogisticRegression L1**	**NMF 80**	**Lemmatization**	**5**	**Clean**	**0.962698**
2	LogisticRegression L1	NMF 80	Stemming	3	Clean	0.959921
3	LogisticRegression L1	NMF 80	Stemming	3	Unclean	0.959127
4	LogisticRegression L1	Truncated SVD 80	Lemmatization	3	Unclean	0.958333

The five best combinations are the situation of clean data with lemmatization.

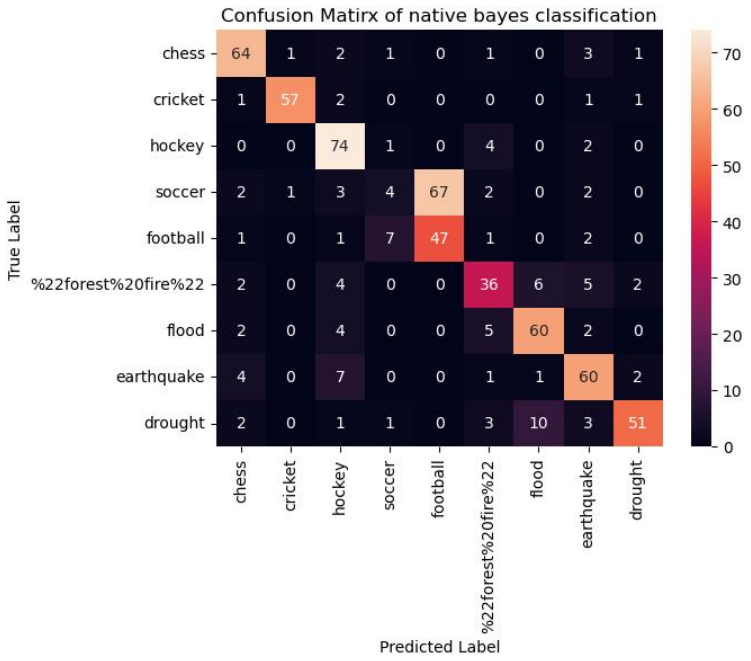
Which are **clean data - lemmatization - min_df=5 - NMF(k=80) - LogisticRegression L1(C=100) ,** the test score is 96.26%

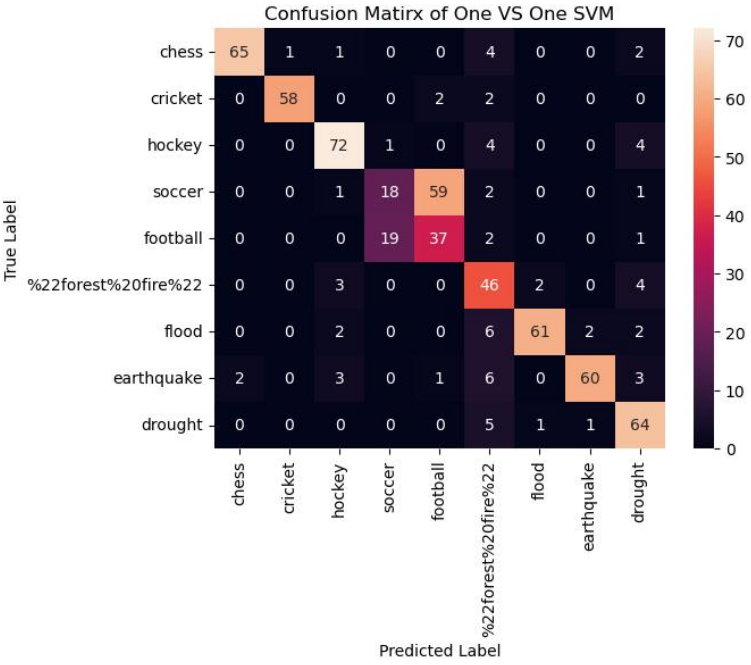
- **QUESTION 9**

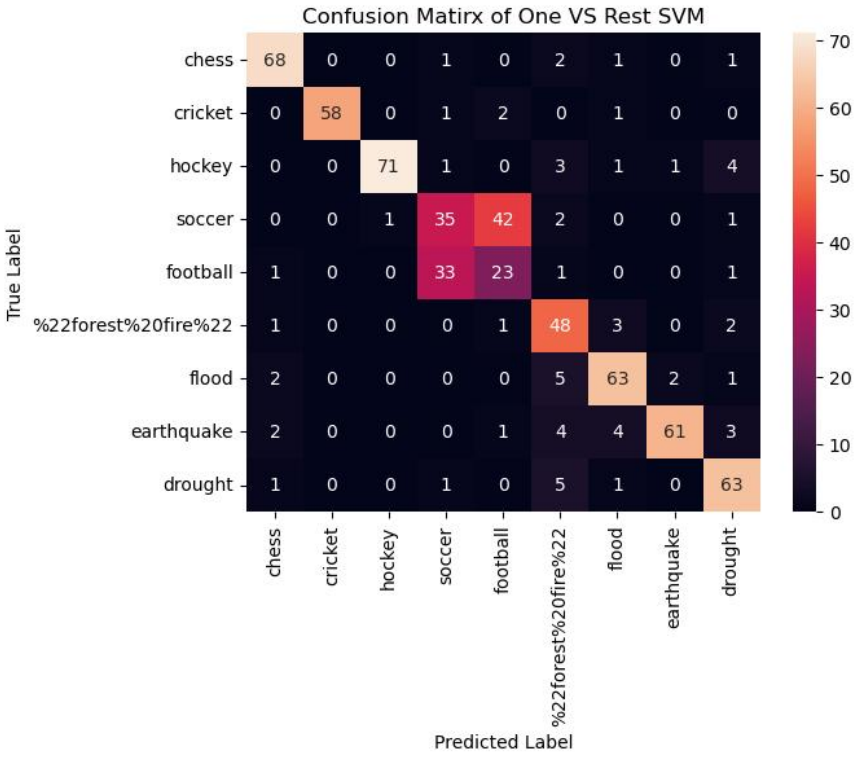
In this part, we aim to learn classifiers on the documents belonging to unique classes in the column leaf label.

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?

Here are the Confusion Matrix of Naive Bayes classification, One VS One SVM, One VS Rest SVM, respectively:







For the Naive Bayes classifier, accuracy = 71.90, recall = 71.90, precision = 71.90, F-1 score = 71.90.

For the One VS One SVM, accuracy = 76.35, recall = 76.35, precision = 76.35, F-1 score = 76.35.

For the One VS Rest SVM, accuracy = 77.78, recall = 77.78, precision = 77.78, F-1 score = 77.78.

As for the imbalance issue in the One VS Rest SVM model, we used an attribute called "class_weight" to resolve. By setting **class_weight = balanced**, each of class weights will become **inversely proportional** to their respective sample numbers or frequency. Therefore, the imbalance issue can be solved by multiplying accordingly new weights.

In addition, answer the following questions:

• In the confusion matrix you should have a 9 ×9 matrix where 9 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:"chess", 1:"cricket", 2:"hockey", 3:"soccer", 4:"football", 5:"%22forest%20fire%22", 6:"flood", 7:"earthquake", 8:"drought"}

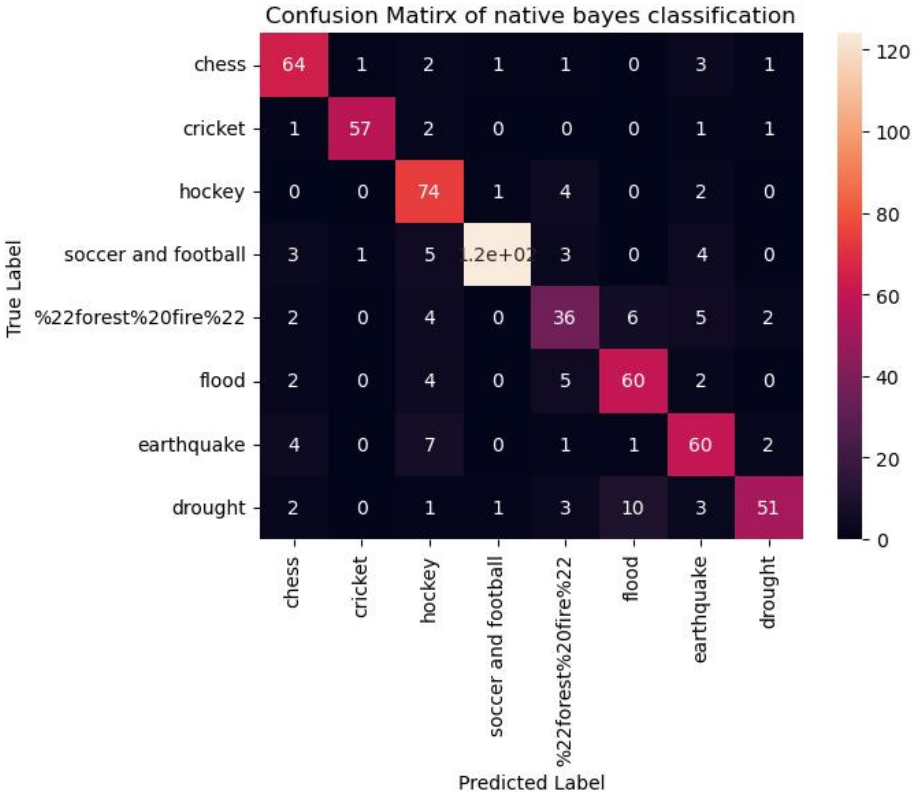
Do you observe any structure in the confusion matrix? Are there distinct visible blocks on the major diagonal? What does this mean?

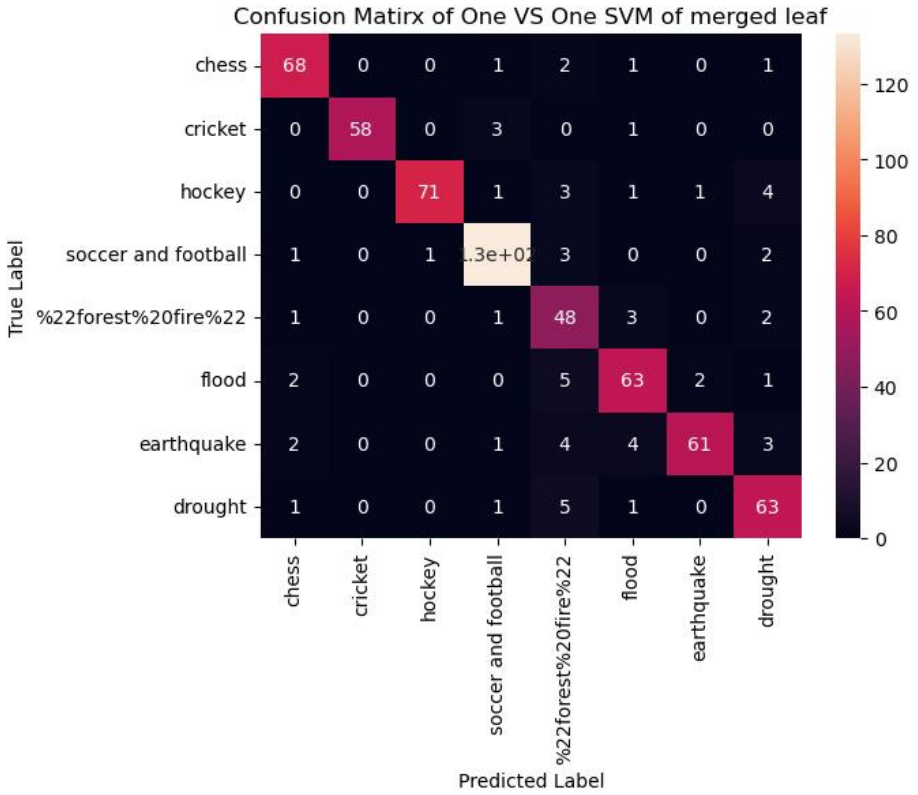
Based on our observations, the confusion matrix has one set of distinct visible blocks on the major diagonal. The diagonal entry is the prediction score for each class sample, which means most of the classes are predicted quite properly.

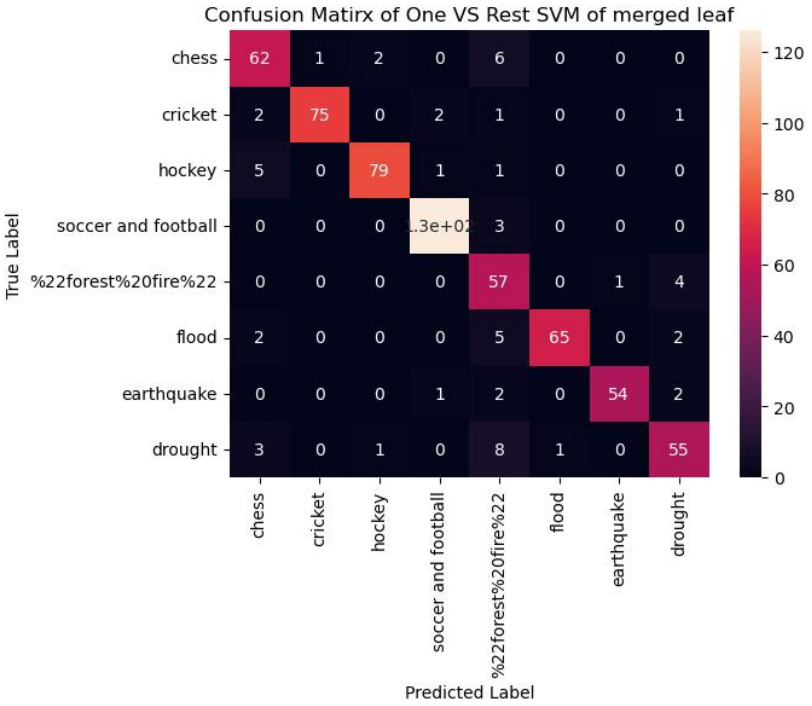
• Based on your observation from the previous part, suggest a subset of labels that

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?

Here are the **merged** Confusion Matrix of Naive Bayes classification, One VS One SVM, One VS Rest SVM, respectively:







Based on the observation above:

the accuracy of merged One VS One SVM is 90.32,

the accuracy of merged One VS Rest SVM is 90.95.

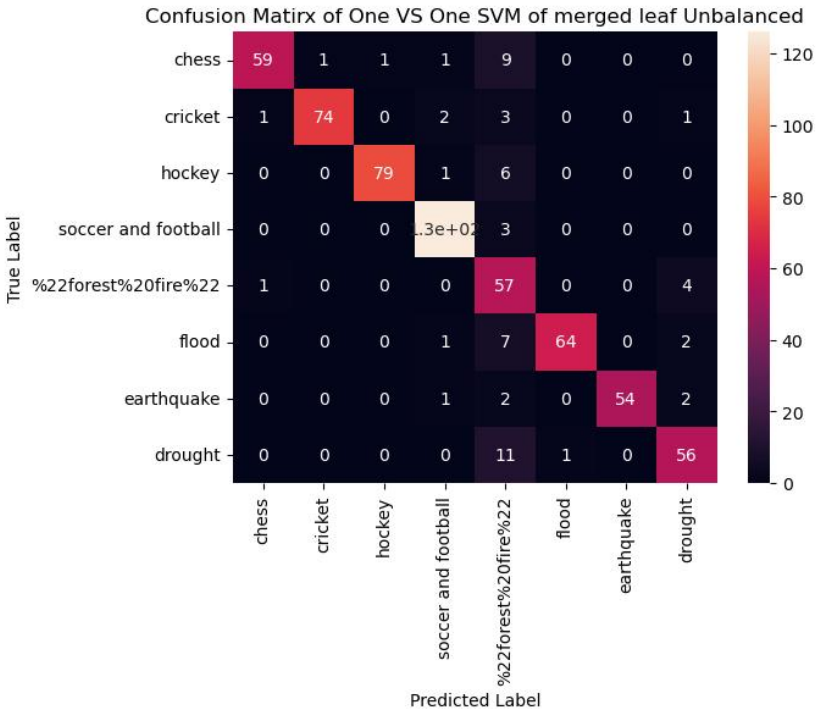
Compared with the original models discussed formerly, the accuracy of merged One VS One SVM and merged One VS Rest SVM are both **higher** than before.

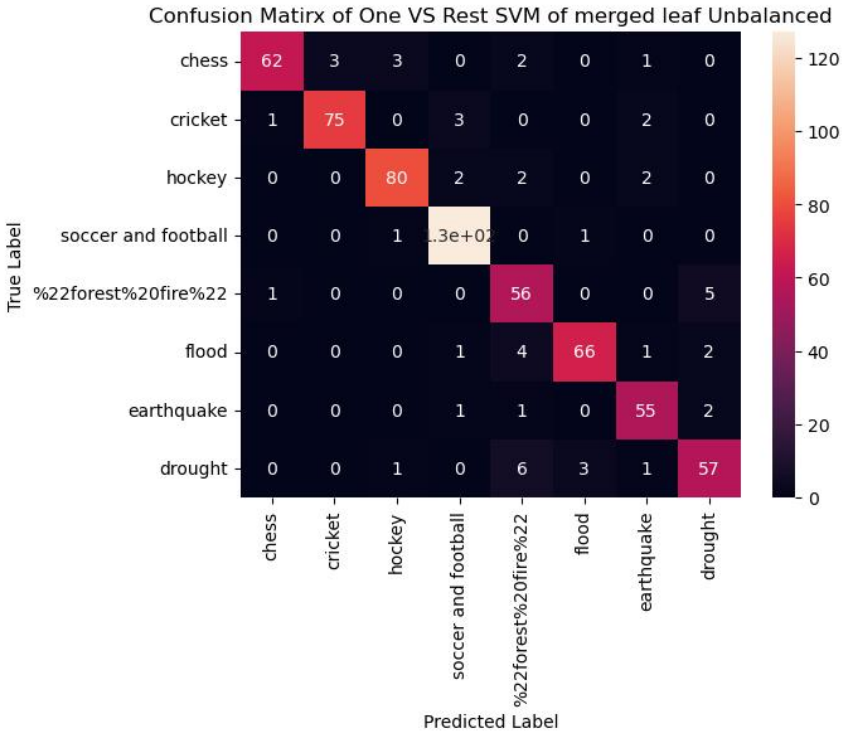
• Does class imbalance impact the performance of the classification once some classes are merged? Provide a resolution for the class imbalance and recompute the accuracy and plot the confusion matrix in One VS One and One VS the rest?

The class imbalance have a slight impact on the performance of the classification since some classes are merged.

We got the accuracy for One VS One = 90.32, which is exactly equal to the balanced model. And the accuracy for One VS the rest = 91.75, which is a little bit higher than the balanced model. We can still use class_weight = balanced here to resolve the imbalance issue.

Here are the plots of the confusion matrix in One VS One and One VS the rest respectively:


Confusion Matirx of One VS One SVM of merged leaf Unbalanced
chess 59 1 1 1 9 0 0 0
cricket 1 74 0 2 3 0 0 1
hockey 0 0 79 1 6 0 0 0
soccer and football 0 0 0 1.3e+02 3 0 0 0
%22forest%20fire%22 1 0 0 0 57 0 0 4
flood 0 0 0 1 7 64 0 2
earthquake 0 0 0 1 2 0 54 2
drought 0 0 0 0 11 1 0 56
True Label
chess
cricket
hockey
soccer and football
%22forest%20fire%22
flood
earthquake
drought
Predicted Label
120
100
80
60
40
20
0


Confusion Matirx of One VS Rest SVM of merged leaf Unbalanced
chess 62 3 3 0 2 0 1 0
cricket 1 75 0 3 0 0 2 0
hockey 0 0 80 2 2 0 2 0
soccer and football 0 0 1 1.3e+02 0 1 0 0
%22forest%20fire%22 1 0 0 0 56 0 0 5
flood 0 0 0 1 4 66 1 2
earthquake 0 0 0 1 1 0 55 2
drought 0 0 1 0 6 3 1 57
True Label
chess
cricket
hockey
soccer and football
%22forest%20fire%22
flood
earthquake
drought
Predicted Label
120
100
80
60
40
20
0

- **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?

In this paper, the author took two terms as examples, which are “ice” and “steam”. By introducing the value of ratio of co-occurrence probabilities, the relationship of these words can be learned by the ratio. Compared to the raw probabilities, the ratio is better able to **distinguish relevant words** (here, like solid and gas) **from irrelevant words** (e.g. water and fashion) and it is also better able to **discriminate between the two relevant words**.

(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. The GLoVE embeddings concern **contexts** even for the same words, which is represented by the co-occurrence matrix. Obviously, the context for “running” in these two sentences are totally different from each other.

(c) What do you expect for the values of,
||GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]||2,
||GLoVE["queen"] - GLoVE["king"]||2 and ||GLoVE["wife"] - GLoVE["husband"]||2 ?
Compare these values.

Here we used codes for above question:

```
1.  print(np.linalg.norm(embeddings_dict[b'queen']-embeddings_dict[b'
    king']-embeddings_dict[b'wife']+embeddings_dict[b'husband']))
2.  print(np.linalg.norm(embeddings_dict[b'queen']-embeddings_dict[b'
    king']))
3.  print(np.linalg.norm(embeddings_dict[b'wife']-embeddings_dict[b'h
    usband']))
```

And the result from above codes is 6.165036, 5.966258 and 3.1520464 respectively.

The first value should be equal to **0** because wife and husband is almost analogous to queen and king. For the same reason the values for the second and third one should be **close** to each other.

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

I would rather lemmatize than stem the word before mapping it to its GLoVE embedding. Because lemmatizing will not create some invalid words that never exist in the previously trained embedding sets to affect the outputs.

- **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.
• To aggregate these words into a single vector consider normalization the vectors, averaging across the vectors.

We decide to use the average glove embedding for the keywords for the train data

```
confuseWords = set()
def wordListToEmbeddng(wordlist, glove_dim):
    init_embedding, wordLen = np.zeros(glove_dim), 0
    for word in wordlist:
        if word not in embeddings_dict:
            confuseWords.add(word)
            continue
        init_embedding += embeddings_dict[word]
        wordLen += 1
    init_embedding /= wordLen
    return init_embedding.tolist()
```

(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 selected HarMargin_SVM model to train the binary classification. For the task distinguishing the "sports" class and "climate" class, we get the following results:

The HardMargin_SVM result for average GLoVE embedding:

accuracy = 93.81
recall = 93.16
precision = 96.46
F-1 score = 94.78

- **QUESTION 12**

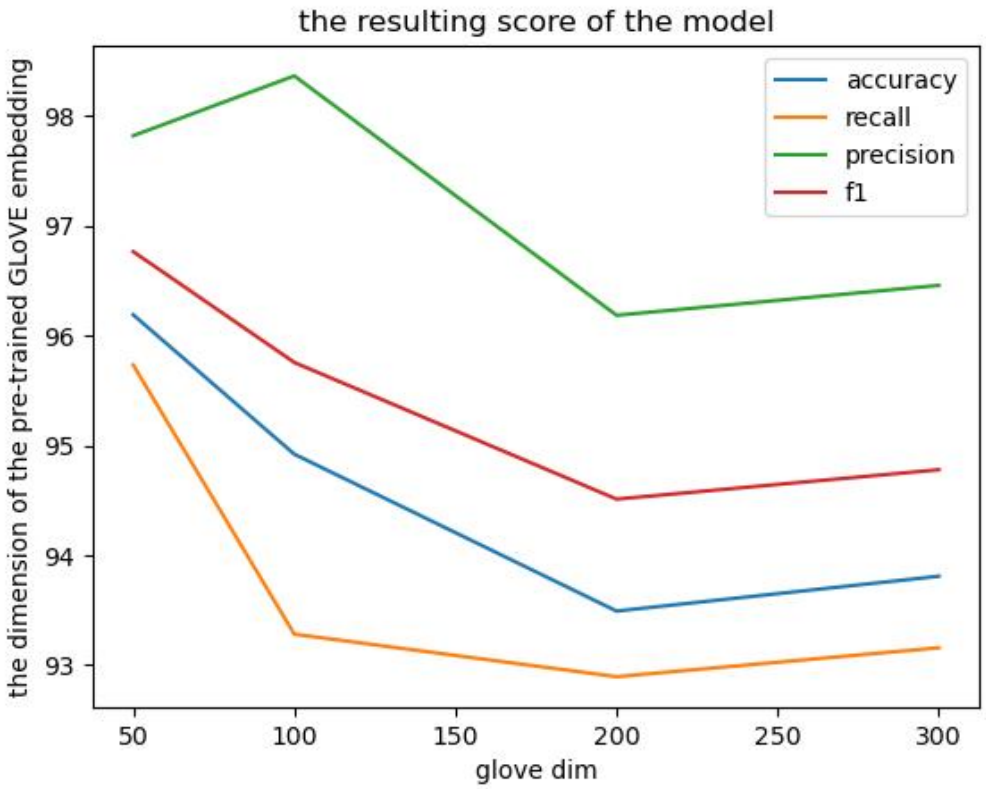
Plot the relationship between the dimension of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task. Describe the observed trend. Is this trend expected? Why or why not? In this part use the different sets of GLoVE vectors from the link.

Before plotting the relationship, we firstly used the following codes to retrieve each accuracy score of different dimensions including {50, 100, 200, 300} with HardMargin_SVM model:

```
1.    results = []
2.    dims = [50, 100, 200, 300]
3.    for dim in dims:
4.        model, model_name = LinearSVC(C=1000, random_state=42), f"Har
   dMargin_SVM_{dim}"
5.        model,result = train_model(train_data_glove, test_data_glove,
    model, model_name, dim)
6.        results.append(result)
```

we obtained the following results:
HardMargin_SVM_50: accuracy= 96.19, recall= 95.73, precision= 97.82, f1= 96.77
HardMargin_SVM_100: accuracy= 94.92, recall= 93.28, precision= 98.37, f1= 95.76
HardMargin_SVM_200: accuracy= 93.49, recall= 92.89, precision= 96.19, f1= 94.51
HardMargin_SVM_300: accuracy= 93.81, recall= 93.16, precision= 96.46, f1= 94.78

Hence, the relationship between the dimension of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task is shown below:



As the dimension of the GLoVE embedding increases, the trend for resulting accuracy firstly decreases and then increases at dim = 200.

And this resulting trend **is expected** because the model accuracy is not monotonous increasing or decreasing with the dimension's increase. At dim = 300, the model is much worse than that in dim = 200, because dim = 300 causes the model behave overfitting.

- **QUESTION 13**

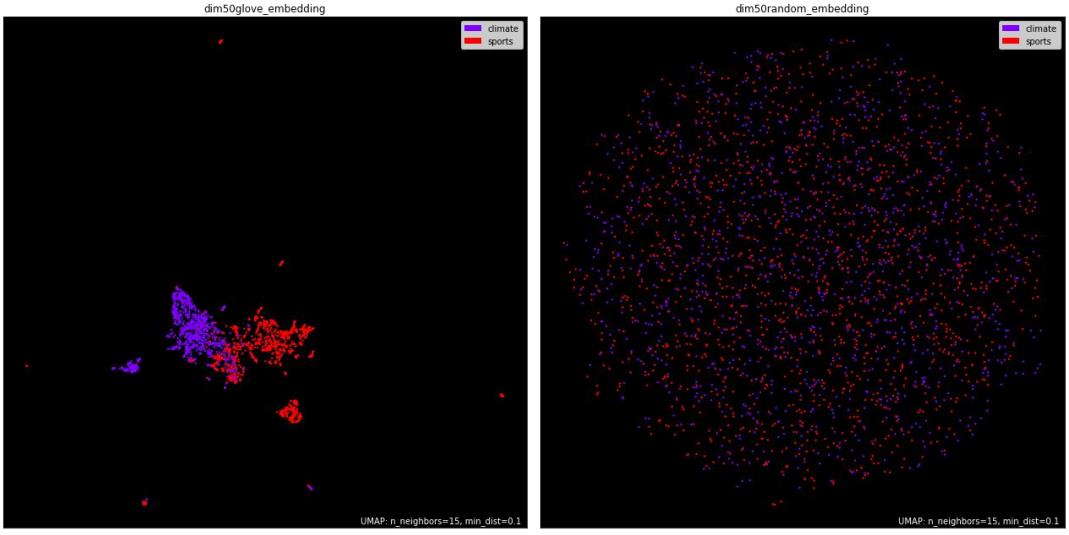
Compare and contrast the two visualizations. Are there clusters formed in either or both of the plots? We will pursue the clustering aspect further in the next project.

By using the codes shown as below:

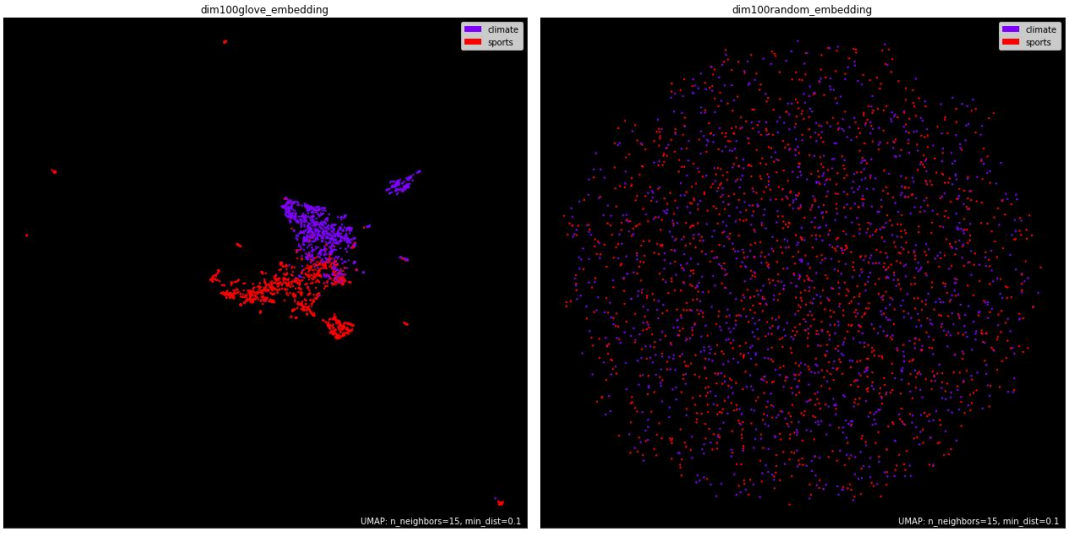
```
1.    uplot_embedding_labels("scaled_embedding_labels_dim50")
2.    uplot_embedding_labels("scaled_embedding_labels_dim100")
3.    uplot_embedding_labels("scaled_embedding_labels_dim200")
4.    uplot_embedding_labels("scaled_embedding_labels_dim300")
```

We get the following visualizations figures with dim = {50, 100, 200, 300} using GLoVE embedding and random embedding respectively, where on the left figure is GLoVE embedding visualizations, on the right is random embedding visualizations:

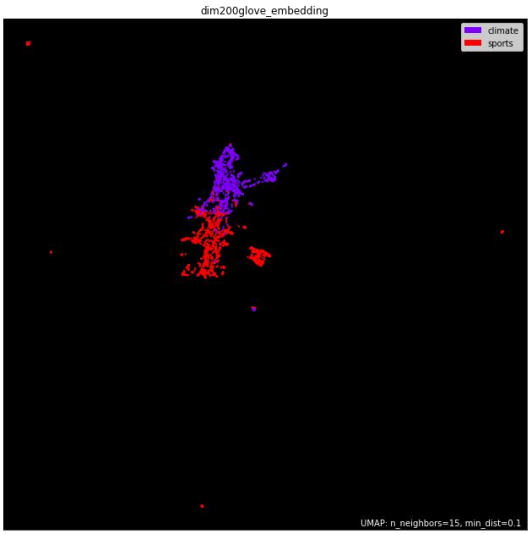
Dimension = 50:

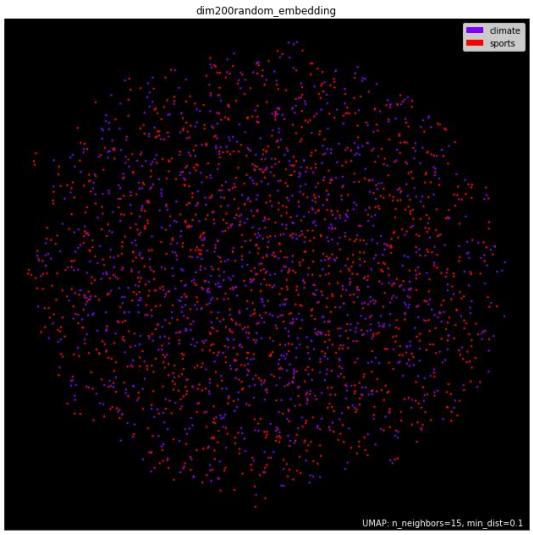


Dimension = 100:

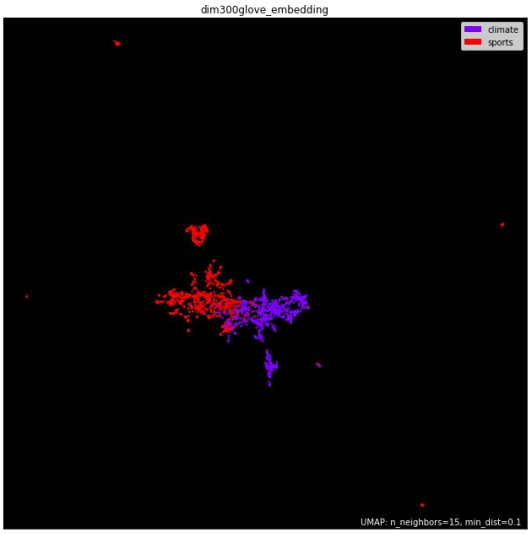


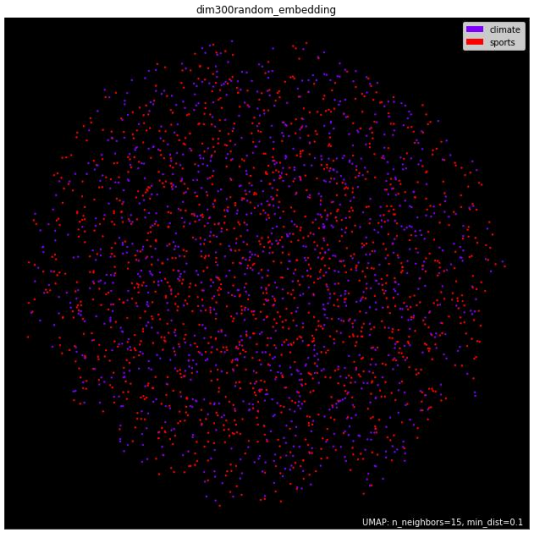
Dimension = 200:





Dimension = 300:





From the above 4 sets of visualizations, we can conclude that in the GLoVE embedding model, it forms two distinct clusters for two different labels which are apparently distant from each other. It indicates that GLoVE has a strong capacity for processing data sets. However, the random embedding's visualizations are sparse and two clusters are mixed together.

- **Conclusion**

In this project, we explore this field of unsupervised learning using textual data, which consists of building an end-to-end pipeline to classify samples of news articles and involves the some ML components.

Since we learn how to store memory of transformer, it only takes 15 minutes for the overall project code to run, we also encapsulate functions as much as possible makes our code clearer, easier to understand, and easier to reuse.

- **Work Distribution**

Our group members are evenly distributed.