Large-Scale Data Mining: Models and Algorithms ECE 219 Winter 2023
Prof. Vwani Roychowdhury
UCLA, Department of ECE

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

Sudeeksha Agrawal, Tazeem Khan, Vamsi Krishna Pamidi

UID: 305928941,105946724,805945580

Introduction:

This project aims to build an end-to-end pipeline for classifying news articles from the GDELT based dataset using machine learning techniques. The pipeline includes steps for feature extraction (using TF-IDF representations), dimensionality reduction (using PCA and NMF), applying simple classification models (such as logistic/linear classification and support vector machines), evaluating the pipeline using grid search and cross validation, and replacing corpus-level features with pre-trained features to evaluate performance.

Getting familiar with the dataset:

QUESTION 1: Provide answers to the following questions:

• Overview: How many rows (samples) and columns (features) are present in the 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;

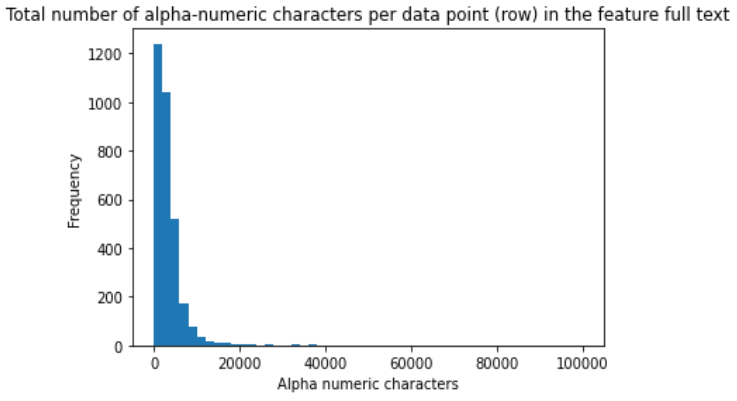
(c) The column root label – class on the x-axis.

• Interpret Plots: Provide qualitative interpretations of the histograms.

Answer 1:

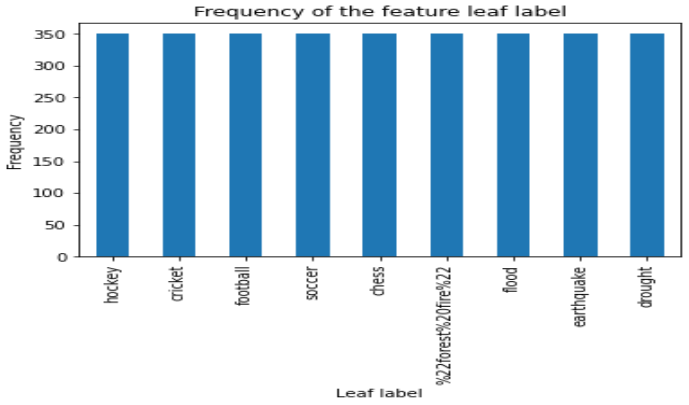
The data has 3150 rows (samples) and 8 columns (features).

(a) Histogram of 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 is below:



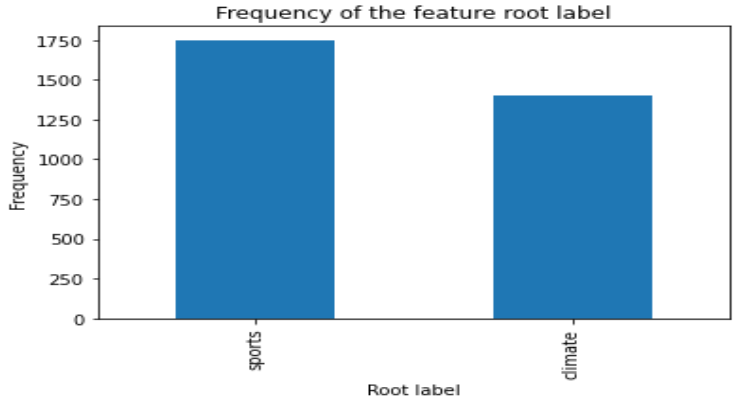
This histogram shows us that the maximum frequency of the alphanumeric characters is 1200 and the count of alphanumeric characters varies across documents from 53 to 100,002 characters. This implies we have a very diverse dataset in terms of size, which needs to be normalized before proceeding.

(b) Histogram of frequency of the feature leaf label text



This histogram shows us the distribution of data across classes and we see that it is equally distributed across the leaf labels at 350 documents for each class. Hence, we can directly use this data for classification as no class imbalance is present.

(c) Histogram of frequency of the feature root label text is below:



This histogram shows us the distribution of data across classes and we see that it is unequally distributed across the root labels at 1750 documents for sports class and 1400 for climate class. Here, we see a slight class imbalance of around 5:4 but since that's the ratio in which each of the root labels are split into the leaf labels which are balanced, we can use this dataset for classification.

Binary Classification:

Splitting the dataset into training/testing:

QUESTION 2: Report the number of training and testing samples?

Answer 2. The number of training samples are 2520 and the number of testing samples are 630.

Feature Extraction:

QUESTION 3: Provide answers to the following questions:

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

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

• Should I remove stop words 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.

• 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.

Answer 3.

- In natural language processing, both lemmatization and stemming are used to convert words to their basic form.
  Lemmatization uses a dictionary or morphological analysis to achieve this and is more precise as it takes into account the context and meaning of the word. For instance, "running" would be converted to "run" and "was" would be converted to "be". However, lemmatization requires more computational power and memory to store the morphological data. Additionally, it may not be effective for languages that do not have a complex morphology or for languages without a morphological dictionary.

  On the other hand, Stemming simplifies words by removing prefixes or suffixes and is a quick technique, but it may not always yield the correct basic form of the word. For example, "running" would be stemmed to "run" and "runner" would also be stemmed to "run".
  In conclusion, lemmatization is more accurate but computationally expensive, while stemming is faster but less accurate.

  Further, stemming has larger dictionary size by producing multiple variations of the same base form of a word while lemmatization reduces the size of the dictionary by producing one variation of the base form of a word. Example// flying and flies get changed to fly by lemmatization but under stemming, it becomes fly and fli respectively increasing dictionary size in stemming.

- min df or the minimum document frequency has a significant effect on the TF-IDF matrix as it changes the number of terms included in the matrix, which captures the importance of each term in the document corpus. In our example we take min df = 3, that implies words present in less than 3 documents will be excluded.
  Increasing the min df value filters out more terms resulting in a smaller matrix with fewer terms and can be useful when the corpus contains rare terms that are not informative.
  Decreasing the min df value includes more terms in the matrix resulting in a larger matrix with more terms, which can be useful when the corpus contains domain-specific terms that are informative.
  The min df value also affects the sparsity of the matrix, where a higher value results in a more sparse matrix with more zero entries and a lower value results in a denser matrix with fewer zero entries, which can affect the computational efficiency of downstream tasks.

- The order in which pre-processing steps are performed can impact the outcome of the analysis, specifically in the case of lemmatization.
  It's generally recommended to remove stop words before lemmatizing in order to decrease the noise and hence computational cost.
  Punctuation is usually removed after lemmatizing as it can improve the POS tagging accuracy which in turn improves the lemmatization results.
  Numbers are removed before lemmatizing as it also can decrease the computational cost by reducing noise.

- The shape of the TF-IDF train matrix is (2520, 15181) and that of the TF-IDF test matrix is (630, 15181).
  We notice that the number of rows in both matrices are similar to results in Q2 and the number of columns are of the order of 10^3 as expected.

Dimensionality Reduction:

QUESTION 4: Reduce the dimensionality of the data using the LSI, NMF methods:

- 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?
- 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 $\| X\text{-}WH \|_F^2$ in NMF or the $\| X\text{-} U_k\Sigma_k V_k^T \|_F^2$ in LSI and why?

Answer 4.

- Below are the figures for LSI corresponding to different k = [1, 10,100, 200, 500, 1000, 2000,50] respectively.

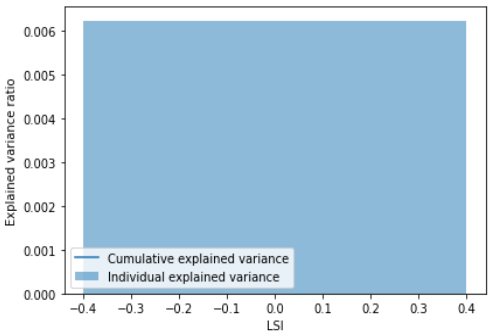


Figure 1 (k=1)

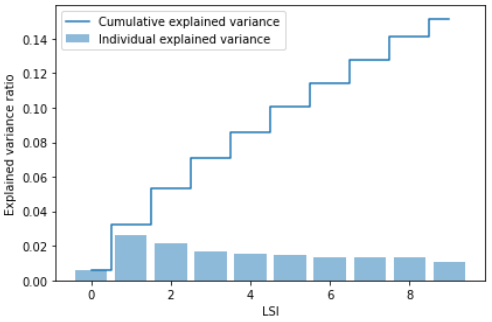


Figure 2 (k=10)

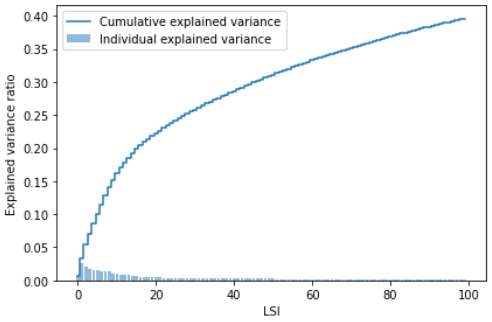


Figure 3 (k=100)

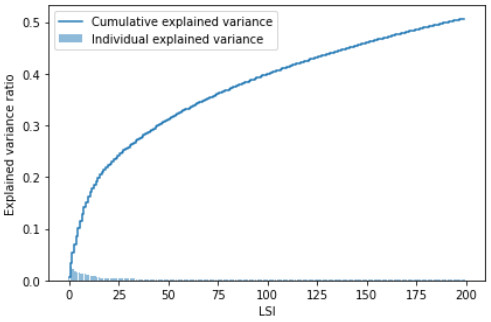


Figure 4 (k=200)

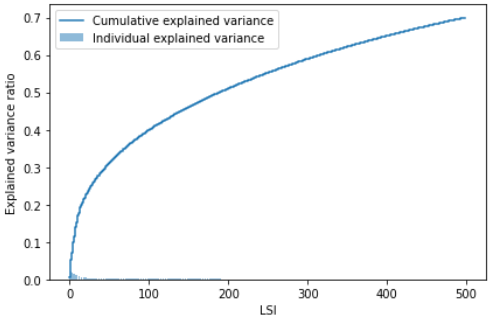


Figure 5 (k=500)

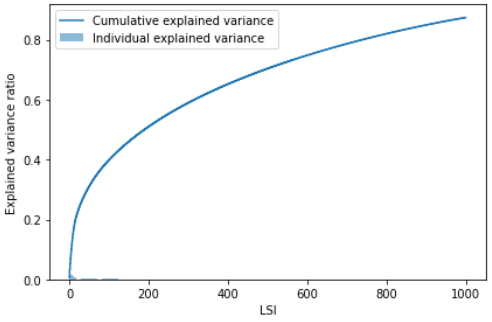


Figure 6 (k=1000)

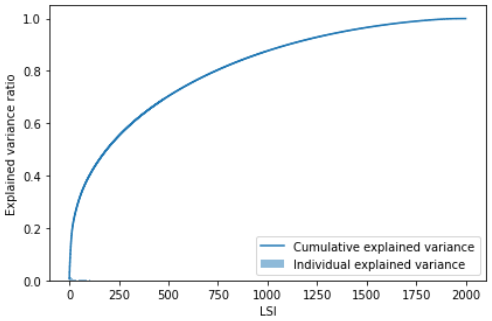


Figure 7 (k=2000)

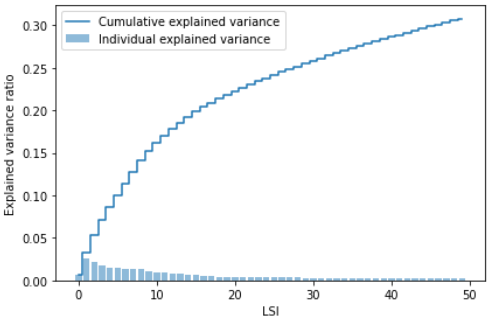


Figure 8 (k=50)

In all the above plots, the x-axis shows the values of k (number of components selected by LSI) and the y-axis shows the explained variance ratio (the % of variance explained by each component). The concavity of the variance plots suggests that as the k increases, the variance ratio also increases, but at a slower rate as k gets larger. For example the explained variance between k=100 and k=200 is larger than what we see for k=500 and k=600. Around 1000 features, or k = 1000, cover approx. 80% of the variance. This suggests that using only the top 1000 features would still provide good performance. The plot's shape demonstrates that it is not possible to exceed 100% in variance explanation.

- The reconstruction residual MSE error for k=50 are as follows:
  $\| X-WH \|_F^2$ in NMF = 1713.1452642145618
  $\| X- U_k\Sigma_k V_k^T \|_F^2$ in LSI = 1687.8485029615786

  Hence, we see that the error is larger for NMF, since NMF evaluates the level of relevance of each document to a specific topic, as opposed to assuming that a document encompasses multiple topics, this is why it is more effective for shorter texts, such as tweets.

Classification Algorithms:

Support Vector Machines (SVM):

QUESTION 5: Compare and contrast hard-margin and soft-margin linear SVMs:

• Train two linear SVMs:

– Train one SVM with $\gamma = 1000$ (hard margin), another with $\gamma = 0.0001$ (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 $\gamma = 100000$?

– What happens to the soft margin SVM? Why is this the case? Analyse in terms of the confusion matrix.

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

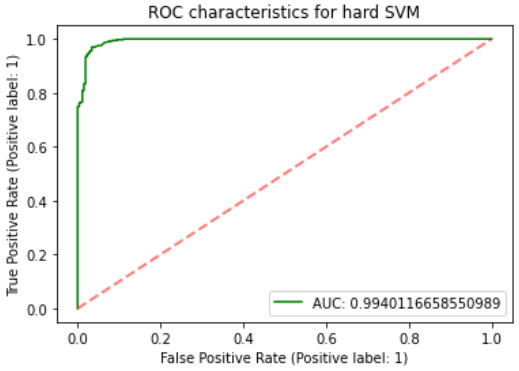
• Use cross-validation to choose γ (use average validation 3 accuracy to compare): Using a 5-fold cross-validation, find the best value of the parameter γ in the range $\{10^k | -3 \leq k \leq 6, k \in Z\}$. Again, plot the ROC curve and report the confusion matrix and calculate the accuracy, recall precision and F1 score of this best SVM.

Answer 5.

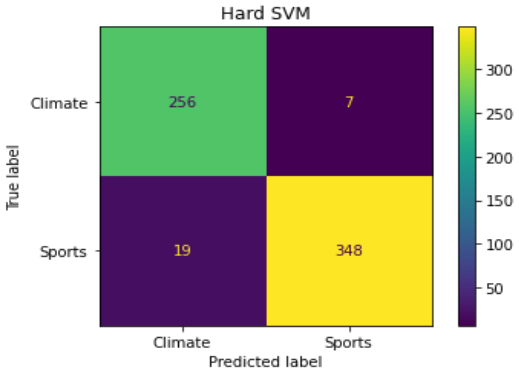
- In this we are comparing soft and hard margin SVM using linear kernels. The performance metrics as per the question are calculated for both the cases on the testing set, as follows:

Hard margin SVM:

Accuracy: 0.9587301587301588
Recall: 0.9482288828337875
Precision: 0.9802816901408451
F1 Score: 0.96398891966759
ROC Curve:

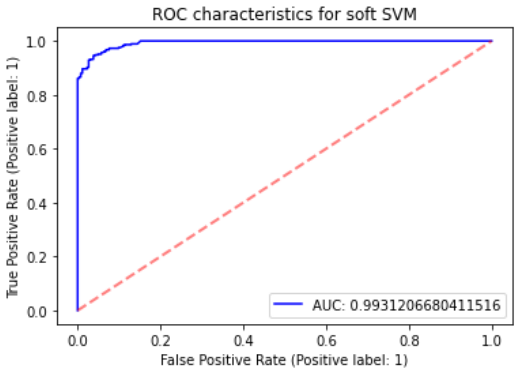


Confusion Matrix:

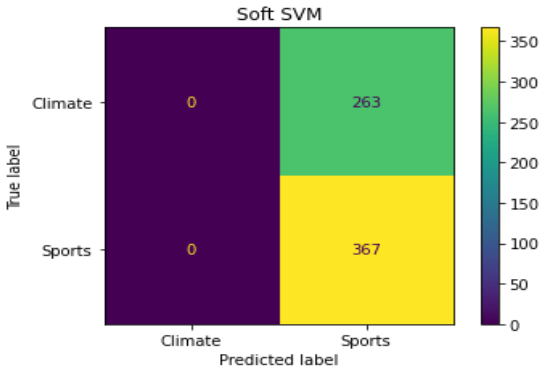


Soft margin SVM:

Accuracy: 0.5825396825396826
Recall: 1.0
Precision: 0.5825396825396826
F1 Score: 0.7362086258776329
ROC Curve:



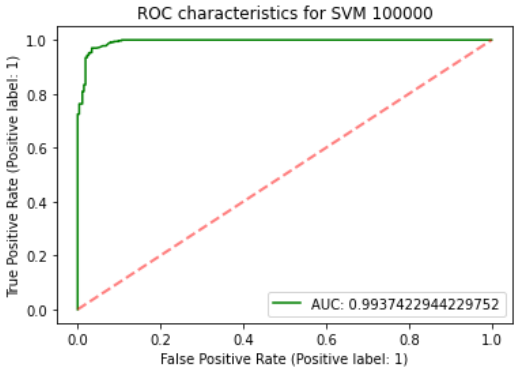
Confusion Matrix:



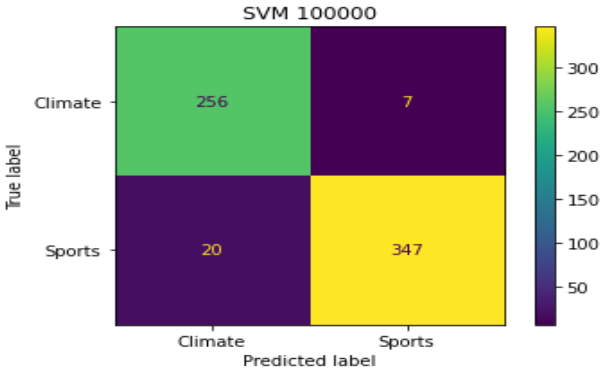
From above performance metrics we can see that Hard margin SVM performs better and gives improved results than Soft margin SVM.

If we change theγ = 100000, the performance metrics are as follows:

Accuracy: 0.9571428571428572
Recall: 0.9455040871934605
Precision: 0.980225988700565
F1 Score: 0.9625520110957004
ROC Curve:



Confusion Matrix:



From the above variation for Hard SVM using the γ = 100000, we see that the regularization strength being high causes the performance to become stable and not improve.

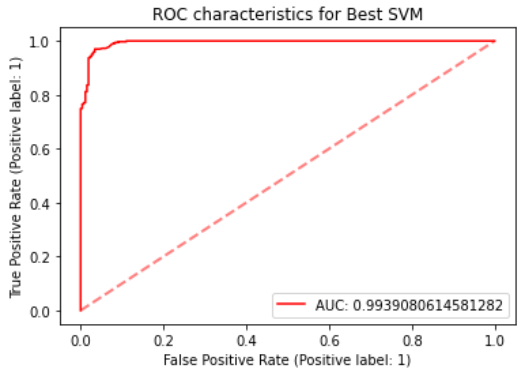
When a dataset is linearly separable, a hard margin SVM is used to minimize misclassifications. On the other hand, when a linear boundary is not possible, a soft margin SVM is used for better generalization across samples, even if it results in more misclassifications.

A soft margin is characterized by a low value of γ, which aims to make the decision surface smooth but leads to more errors. A hard margin, on the other hand, has a high value of γ and focuses on correctly classifying all training examples. This results in a high number of true predictions and a low number of false predictions in the confusion matrix.

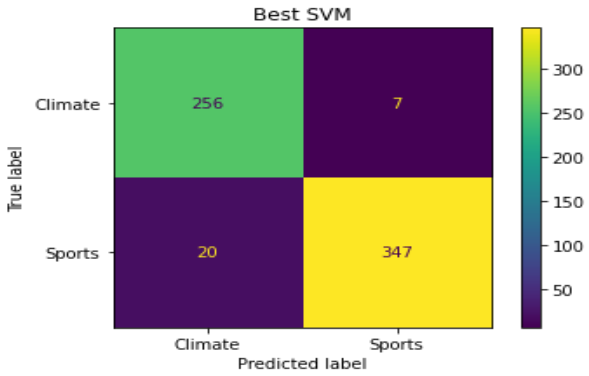
However, the ROC curve for a soft margin SVM may contradict the other metrics, such as confusion matrix, accuracy, precision, and F1, which indicate a high misclassification rate for one class and a low misclassification rate for the other class.

- Further, we use the 5-fold cross validation to choose the best γ (Regularization strength) in the given range in question. It is found to be for **γ=500.** The performance metrics for this best SVM is given below:

  Accuracy: 0.9571428571428572
  Recall: 0.9455040871934605
  Precision: 0.980225988700565
  F1 Score: 0.9625520110957004
  ROC Curve:



  Confusion Matrix:



Logistic Regression:

QUESTION 6: Evaluate a logistic classifier:

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

• 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 $\{10^k | -5 \leq k \leq 5, k \in Z\}$ for logistic regression with L1 regularization and logistic regression with L2 regularization, respectively.

– 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.

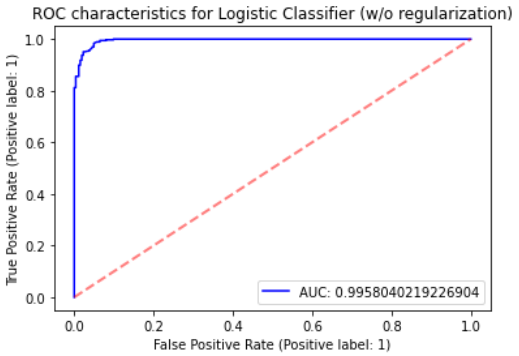
– How does the regularization parameter affect the test error? How are the learnt coefficients affected? Why might one be interested in each type of regularization?

– 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?

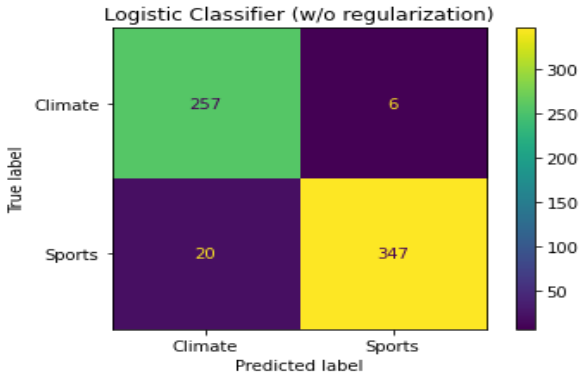
Answer 6.

- In the first part we train a logistic classifier without regularisation and obtain the below performance results:

  Accuracy: 0.9587301587301588
  Recall: 0.9455040871934605
  Precision: 0.9830028328611898
  F1 Score: 0.9638888888888889
  ROC Curve:



  Confusion Matrix:



- Then we find the optimal regularisation coefficient using 5-fold cross validation and obtain the performance metrics on Logistic regression with L1 regularisation as below:

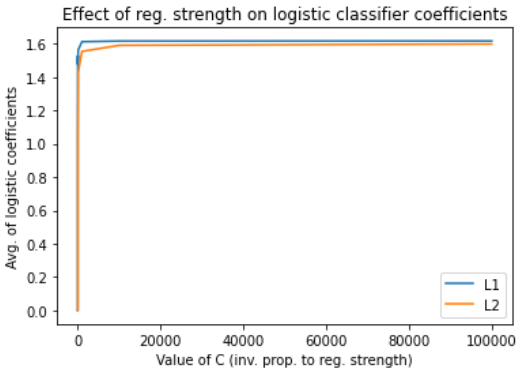
  Best regularisation coefficient, γ= **100000**
  Accuracy: 0.9603174603174603
  Recall: 0.9482288828337875
  Precision: 0.9830508474576272
  F1 Score: 0.9653259361997226

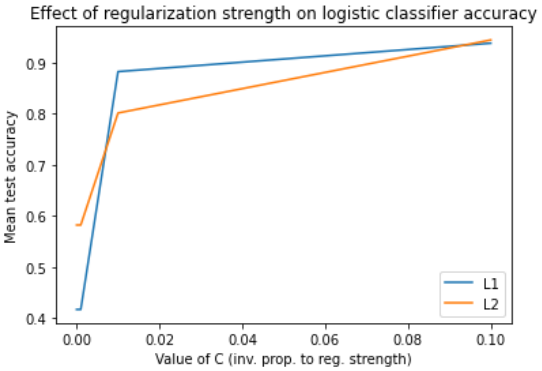
- Then, we find the optimal regularisation coefficient using 5-fold cross validation and obtain the performance metrics on Logistic regression with L2 regularization as below:

Best regularisation coefficient, **γ= 100**
Accuracy: 0.9587301587301588
Recall: 0.9509536784741145
Precision: 0.9775910364145658
F1 Score: 0.9640883977900553

- Comparing the performance metrics (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 found from the part above), using test data we observe that the Logistic regression with L1 regularization has slightly improved performance than the other 2 classifiers.

  The **choice of regularization method, L1 or L2**, can have a significant impact on the resulting model. **L1 regularization**, also known as Lasso regularization, is useful for feature selection as it tends to set the weights of unimportant features to zero due to its sign (w) sub-gradient nature based on ||w||, leading to a simpler and sparser model.
  L2 regularization, also known as Ridge regularization, on the other hand based on w^2, focuses on the magnitude of the weights and can be useful for decoupling the effects of correlated features. However, it is not as effective for feature selection and can lead to models with diffuse weights that are susceptible to outliers.





Based on the above plots for the effect of regularization strength on Logistic classifier coefficients on Mean Learnt Coefficients and Mean Test Accuracy highlight that:

- As the C/ γ (inversely proportional to regularization strength) increases, the mean value of logistic regression coefficients and test accuracy decrease.
  This is because more weights in the learned model are set to 0, which is a result of the differences in the formulations of L1 and L2 regularization, with L1 tending to set more weights to zero than L2 as it is based on ||w|| instead of w^2.
- As the complexity of the model increases during training, certain important weights that are necessary for differentiating features may be eliminated. This can lead to poor test accuracy if the regularization strength is set too high.
- By using moderate values for regularization, the model is able to generalize well when making predictions and avoids overfitting to the training data.
- Additionally, regularization can reduce the variance and make the model less sensitive to outliers, making it more stable overall.

- Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary but below are the differences between their ways:
  - The main difference between logistic regression and linear SVM in terms of finding a linear decision boundary is in the objective function they optimize. SVM is a geometric algorithm that focuses on a subset of data points, called support vectors that are closest to the decision

boundary. It uses these points to find an optimal hyperplane that maximizes the distance between the vectors and the margin.
On the other hand, Logistic Regression is a statistical algorithm that aims to maximize the probability of correctly classifying a random data point, known as Maximum Likelihood Estimation (MSE).

- In terms of performance the difference is seen in cases where the dataset is not linearly separable, Logistic Regression may not be successful, while SVM with a non-linear kernel (like RBF), can still find a decision boundary by transforming the data into a higher dimensional feature space.
- In terms of statistical approach, SVM aims to find the widest possible margin using non-linear kernels, while Logistic Regression optimizes the log likelihood function, with probabilities modelled by functions such as sigmoid. In general we can't comment on the statistical significance of the difference between the two models as it depends on the specific dataset and the evaluation metrics used. Commonly, cross-validation is used to compare the performance of the two models and find the best fit for a given dataset. Like in our case, by 5 fold cross validation we see that L1 Logistic regression performs better than linear SVM.

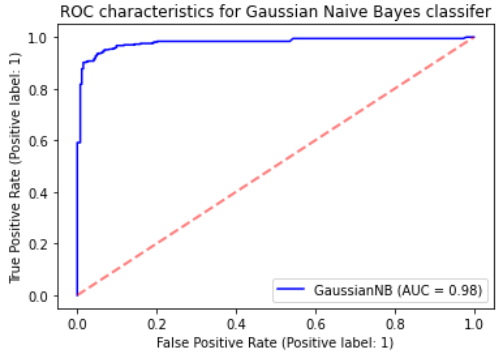
Naïve Bayes Model:

QUESTION 7: Evaluate and profile a Naïve Bayes classifier: Train a Gaussian NB 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.

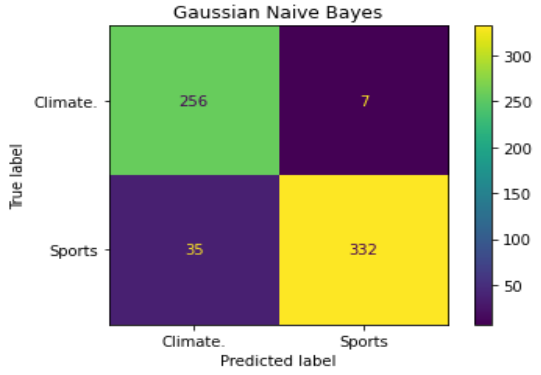
Answer 7.

In this part we train a Gaussian Naïve Bayes classifier and obtain the below performance results:

Accuracy: 0.9333333333333333
Recall: 0.9046321525885559
Precision: 0.9793510324483776
F1 Score: 0.9405099150141643
ROC Curve:



Confusion Matrix:



Grid Search of Parameters:

QUESTION 8: In this part, we will attempt to find the best model for binary classification using Pipeline creation.

• 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 hyper parameters you choose, your grid search must at least include: Loading Data: Clean the data; Feature Extraction: min df = 3 vs 5 while constructing the vocabulary; and use Lemmatization vs Stemming as a compression module; Dimensionality Reduction LSI (k = [5, 30, 80]) vs NMF (k = [5, 30, 80]); Classifiers: SVM with the best γ previously found vs Logistic Regression: L1 regularization vs L2 regularization, with the best regularization strength previously found vs Gaussian NB.

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

Answer 8.

In this part we construct a pipeline combining all the steps of extraction, dimensionality reduction and classification using above hyper parameters and obtain the below 5 best combinations with test set performance results:

	Classifier	Reduce_dim	Analyzer	min_df	mean_test_score	Test Scores
1	SVM(C=500, kernel='linear', random_state=42)	TruncatedSVD(n_components=80, random_state=42)	Lemmatized	5	0.958730	0.953968
2	SVM(C=500, kernel='linear', random_state=42)	TruncatedSVD(n_components=80, random_state=42)	Lemmatized	3	0.957937	0.952381
3	LogisticRegression(C=100, random_state=42, solver='liblinear')	TruncatedSVD(n_components=80, random_state=42)	Stemmed	5	0.957937	0.949206
4	LogisticRegression(C=100000, penalty='l1', random_state=42, solver='liblinear')	TruncatedSVD(n_components=80, random_state=42)	Stemmed	5	0.957937	0.947619
5	LogisticRegression(C=100000, penalty='l1', random_state=42, solver='liblinear')	TruncatedSVD(n_components=80, random_state=42)	Stemmed	3	0.957937	0.947619

Multiclass Classification:

QUESTION 9: Perform Naïve 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?

In addition, answer the following questions:

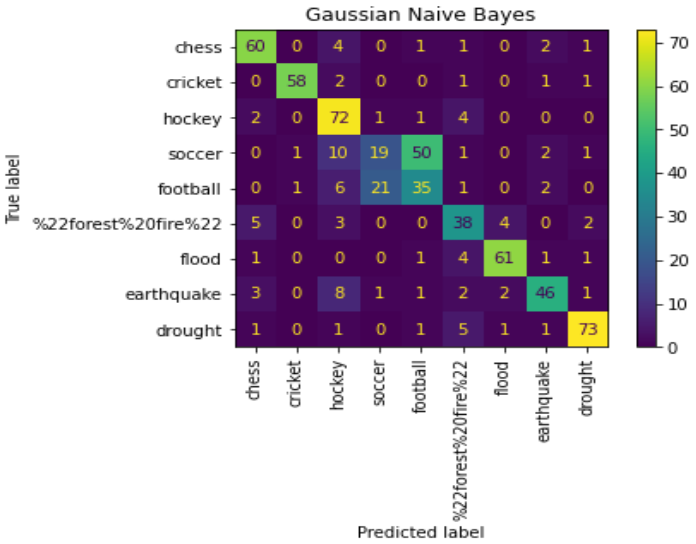
• In the confusion matrix you should have a 9 × 9 matrix where 9 is the number of unique labels in the column leaf label. Please make sure that the order of these labels is as follows: map_row_to_class = {0:"chess", 1:"cricket", 2:"hockey", 3:"soccer", 4:"football", 5:"%22forest%20fire%22", 6:"flood", 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 your observation from the previous part, suggest a subset of labels that should be merged into a new larger label and re compute the accuracy and plot the confusion matrix. How did the accuracy change in One VS One and One VS the rest?

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

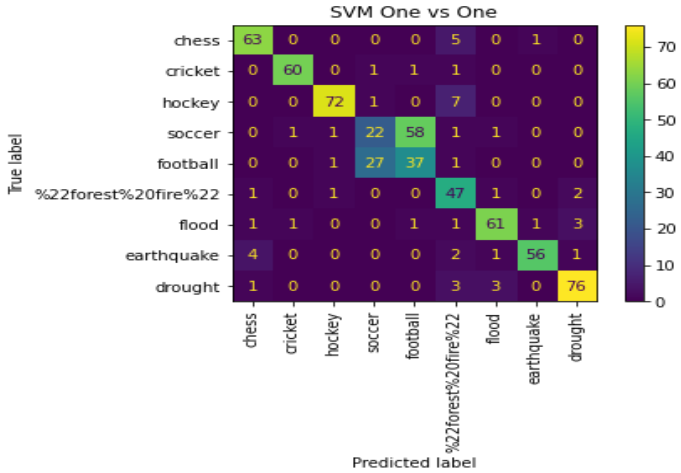
Answer 9.

- In this part we train a Gaussian Naïve Bayes classifier and obtain the below performance results:
  Accuracy: 0.7412698412698413
  Recall: 0.73
  Precision: 0.73
  F1 Score: 0.72
  Confusion Matrix:



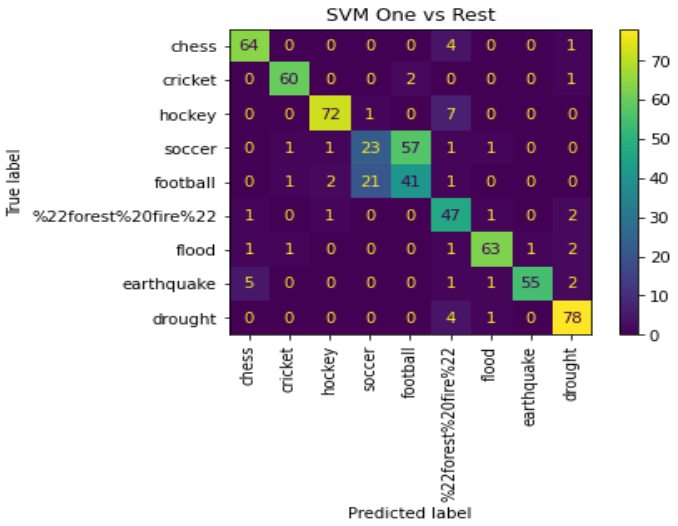
We also train a Multiclass SVM classifier (One vs One) and obtain the below performance results:

Accuracy: 0.7968253968253968
Recall: 0.93
Precision: 0.93
F1 Score: 0.93
Confusion Matrix:



We also train a Multiclass SVM classifier (One vs Many) and obtain the below performance results:

Accuracy: 0.7984126984126985
Recall: 0.80
Precision: 0.81
F1 Score: 0.79
Confusion Matrix:



In one vs one classification, n classifiers are trained to compare one class against another, resulting in n*(n-1)/2 models. In one vs rest classification, n classifiers are trained to compare each class against all other classes, resulting in n models. One vs rest classification can lead to imbalanced datasets as elements outside the class outweigh elements within it, which can be addressed by setting class weight to 'balanced' to give weight inversely proportional to the frequency of the classes. One vs one classification generally has better accuracy, but class imbalances may also play a role.

We resolve this class imbalance in the One to Many case of Multi class SVM by using the technique of sub-sampling.

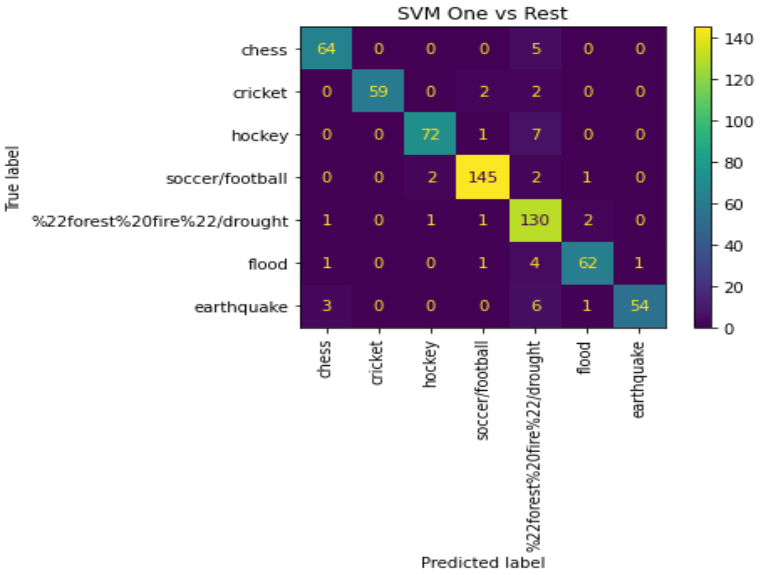
- In the confusion matrix we see that the diagonal elements have the maximum value. As there are distinct blocks present in the major diagonal of the confusion matrix, it indicates that there is a relatively high number of accurate predictions being made. However, using techniques such as combining labels and reducing the sample size could lead to even better performance.

- The one vs rest method of data partitioning and classification can lead to imbalanced datasets, resulting in lower accuracy. To overcome this, techniques such as merging related labels and reducing the number of negative clusters seen by each classifier can be used to improve performance. Analysing the above results we see that the ‘Merging Semantic Labels’ technique of merging related labels improves the performance of both One vs One and One vs Rest SVMs. The dimensions of the confusion matrix are also reduced due to reduction in classes for classification.We have combined the classes for soccer ,football together and forest fires with drought based on semantic similarity. The improved performance can be observed through the confusion matrix below.

- By using subsampling to balance the data, we are able to improve the performance of our classification pipeline. This is shown by an improvement in performance metrics and confirms that this is a promising approach to addressing the issue of imbalance in the data.

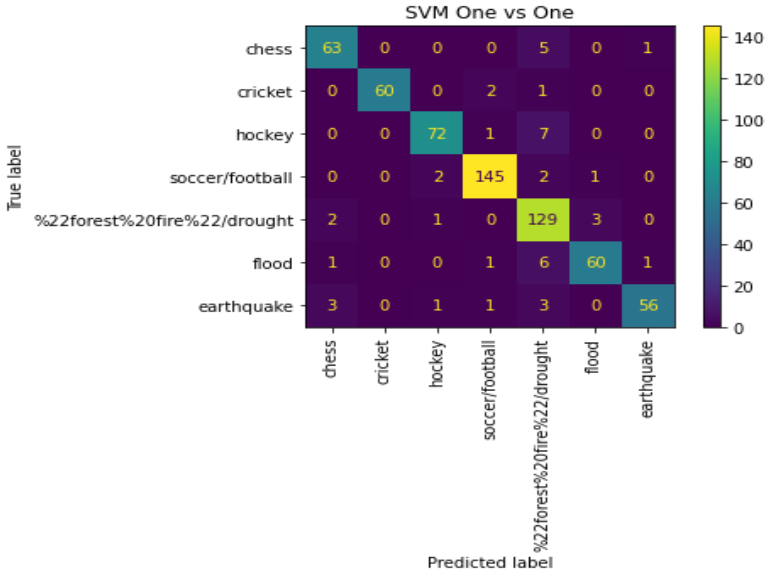
Multiclass SVM classifier (One vs Many) with subsampling we obtain the below performance results:

Accuracy: 0.93
Recall: 0.93
Precision: 0.93
F1 Score: 0.93
Confusion Matrix:



Multiclass SVM classifier (One vs One) with subsampling we obtain the below performance results:

Accuracy: 0.93
Recall: 0.93
Precision: 0.93
F1 Score: 0.93
Confusion Matrix:



Word Embedding:

QUESTION 10: Read the paper about GLoVE embedding and answer the following sub questions:

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

(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?

(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.

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

Answer 10.

a) GLoVE embeddings are trained on the ratio of co-occurrence probabilities rather than the probabilities themselves because the ratio of co-occurrence probabilities captures the relative significance of different word pairs in the corpus. Generally, the context words nearer to the focus word are more meaningful in context than the words farther away.This ratio is a measure of the strength of the association, which is important for the accuracy of the embeddings. Using the probabilities themselves would not capture the association as effectively.

b) In the two sentences given, “James is running in the park” and “James is running for the presidency”, the GLoVE embeddings would return the same vector for the word “running”. It highlights a limitation of GLoVE called polysemy. If GLoVE models are created from scratch instead of using a pretrained model, then the vectors would be returned differently. But since we are using a pretrained model where the scores for each word are predefined, we would be getting the same vector.

c) The values of the given combinations are, ||GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]||$_2$ is expected to be closer to 0 while for ||GLoVE["queen"] - GLoVE["king"]||$_2$ and ||GLoVE["wife"] - GLoVE["husband"]||$_2$ the values are mostly going to be similar since the relationship between the words is the same i.e. queen : king =wife : husband. Therefore, the vector representation of queen and king or wife and husband will be similar to that of woman and man, indicating the relationship of gender change.

d) Stemming simplifies words by cutting off the end of them, regardless of context. This can lead to meaningless or misspelled words. On the other hand, lemmatization reduces words to their base form by taking into account context, semantics, parts of speech, etc. Thus, if needed, lemmatization is a better choice than stemming. And these lemmatized words can be used to make a GLoVE score vector.

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 normalizing the vectors, averaging across the vectors.

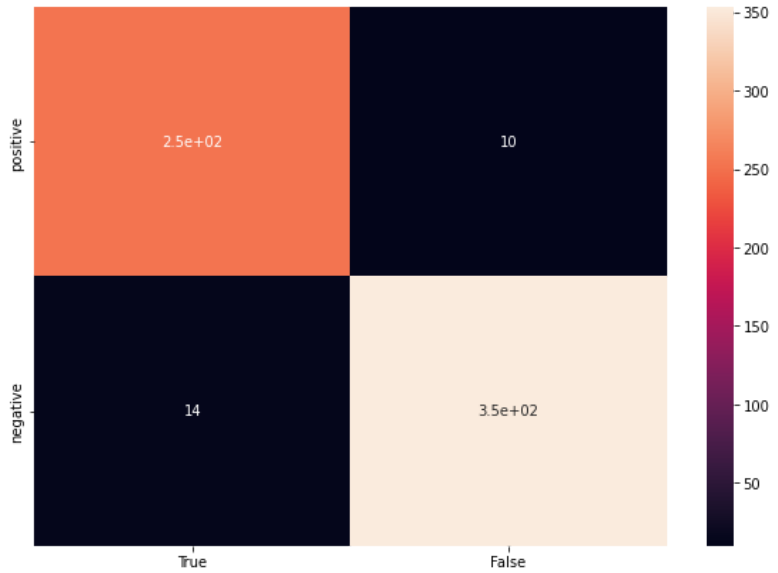
(b) Select a classifier model, train and evaluate it with your GLoVE-based feature. If you are doing any cross-validation, please make sure to use a limited set of options so that your code finishes running in a reasonable amount of time.

Answer 11.

a) A feature engineering process that uses GLoVE embeddings with the rules given, is described below:
- Firstly, the corpus can be cleaned as it was cleaned in the prior states like removing #, @, numbers, and other irrelevant things. Then using lemmatization to change words to their base forms .
- Now this input can be broken down into sentences containing separated words. Then, these separated words can be used to create GLoVE scores for each of these words.
- These scores can be averaged into a single score vector by averaging it for all the words in a sentence. This will give a score vector for every single data point.
- These vectors can now be used to train a model. For this question, the full_text column was used to make the GLoVE score vectors which were then used to create the training and testing data. These data were then trained on a SVM with different values of γ to choose which value of γ would be suitable for the model.

b) The model used is linear SVM for classifying and we applied 5 fold cross validation to get the best γ=10 with below performance metrics:

Accuracy: 0.962
Recall: 0.962
Precision: 0.972
F1 Score: 0.967
Confusion Matrix:

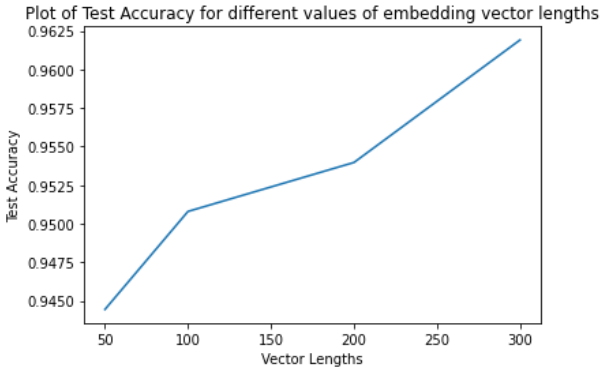


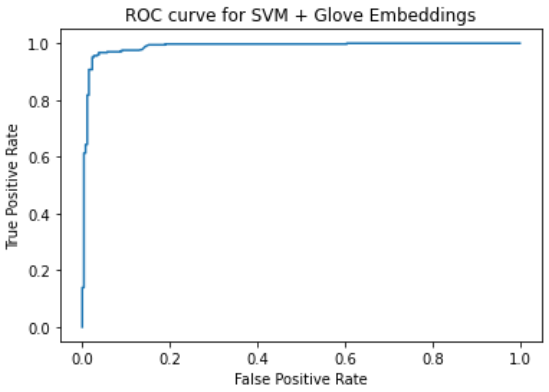
*** True, Positive and False,Negative labels correspond to the sports and climate classes respectively.*

QUESTION 12: Plot the relationship between the dimensions of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task. Describe the observed trend. Is this trend expected? Why or why not? In this part use the different sets of GLoVE vectors from the link.

Answer 12.

We tested a linear SVM classifier with a value of γ = 10 for different dimensions of pre-trained GLoVE embedding (50, 100, 200, 300) and found that as the dimension of the embedding increased, the test accuracy also increased. This is likely because higher-dimensional embedding contains more detailed semantic and contextual information about the words, which in turn allows the classifier to make more accurate predictions.

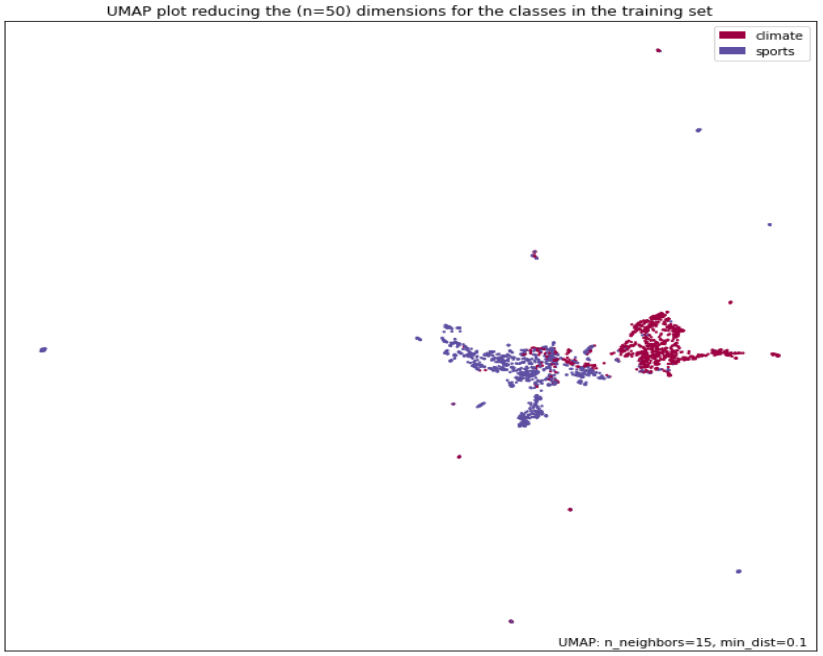




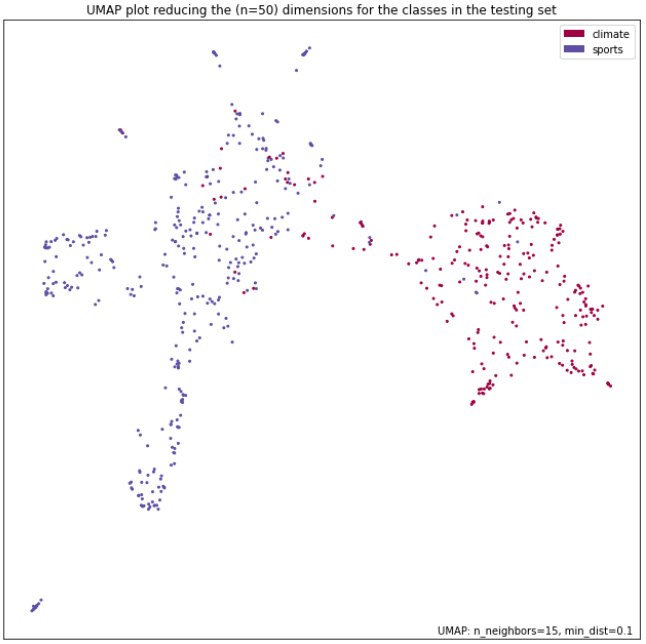
QUESTION 13: Compare and contrast the two visualizations. Are there clusters formed in either or both of the plots? We will pursue the clustering aspect further in the next project.

Answer 13.

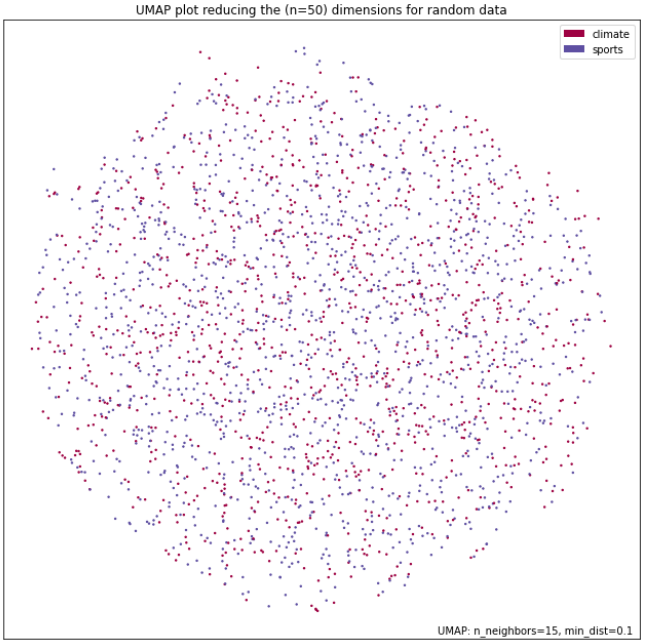
The clusters are only formed in the case of the training and testing data plots. The clusters can be seen in all the four different embedding length plots for both training and testing plots. The clusters can't be seen in the random number plot as there is no correlation between the numbers generated and the labels for each data point. This can be seen in the plots below:



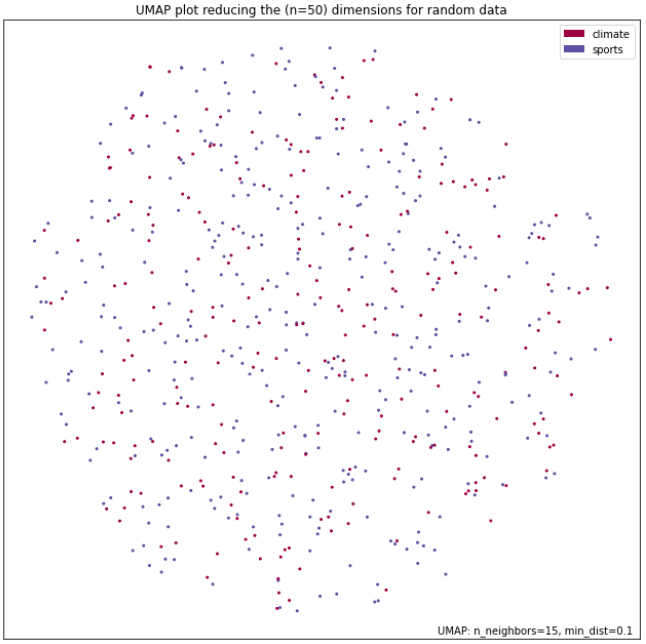
UMAP plot reducing the (n=50) dimensions for the classes in the testing set



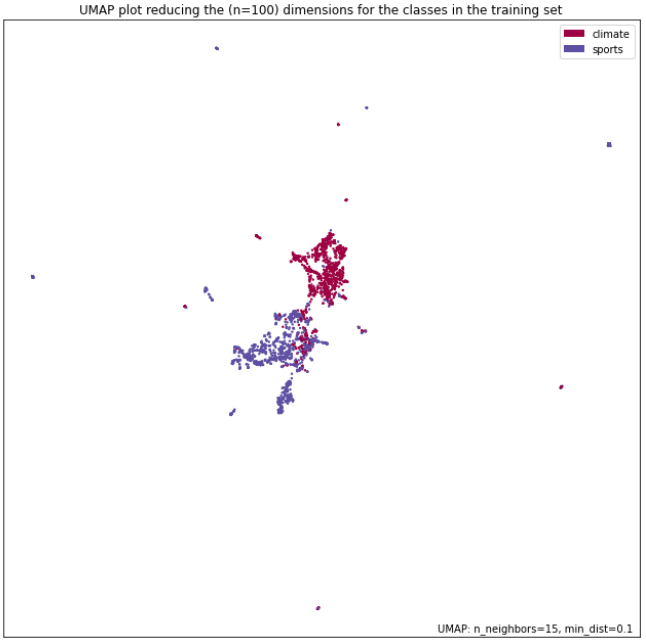
UMAP plot reducing the (n=50) dimensions for random data



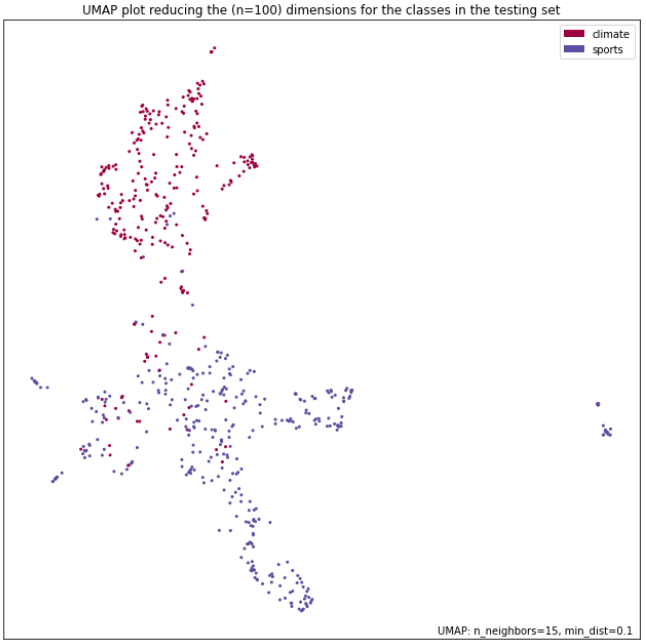
UMAP plot reducing the (n=50) dimensions for random data



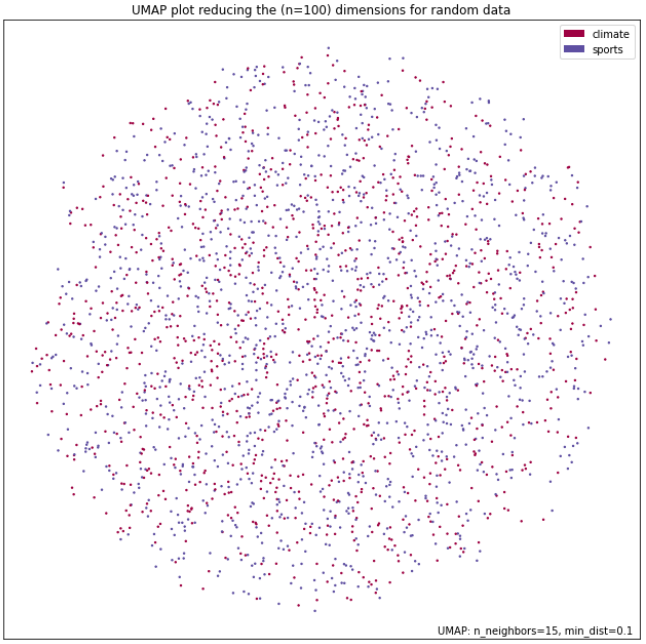
UMAP plot reducing the (n=100) dimensions for the classes in the training set



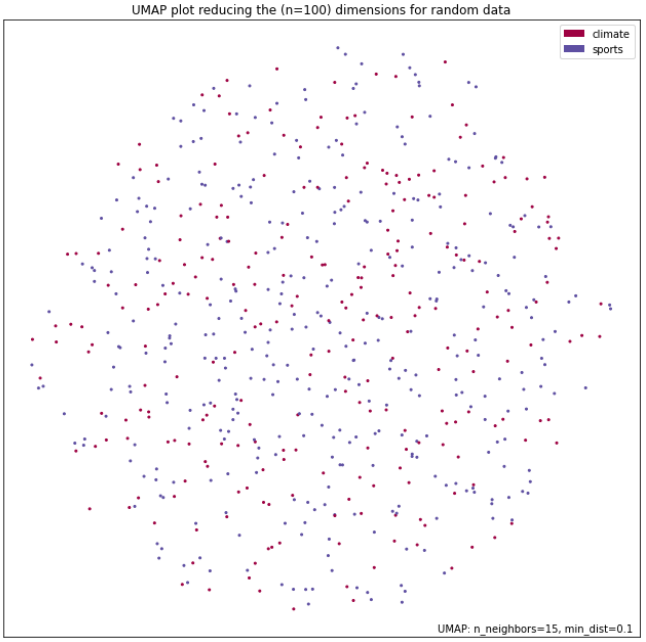
UMAP plot reducing the (n=100) dimensions for the classes in the testing set



UMAP plot reducing the (n=100) dimensions for random data



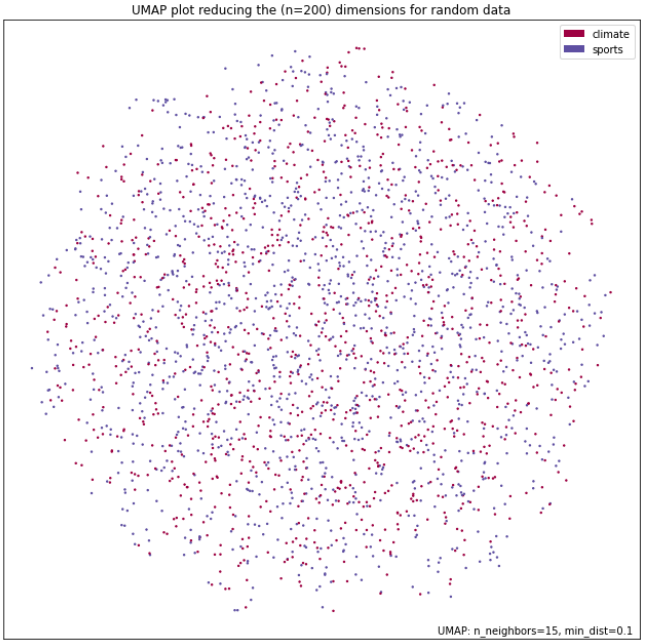
UMAP plot reducing the (n=100) dimensions for random data



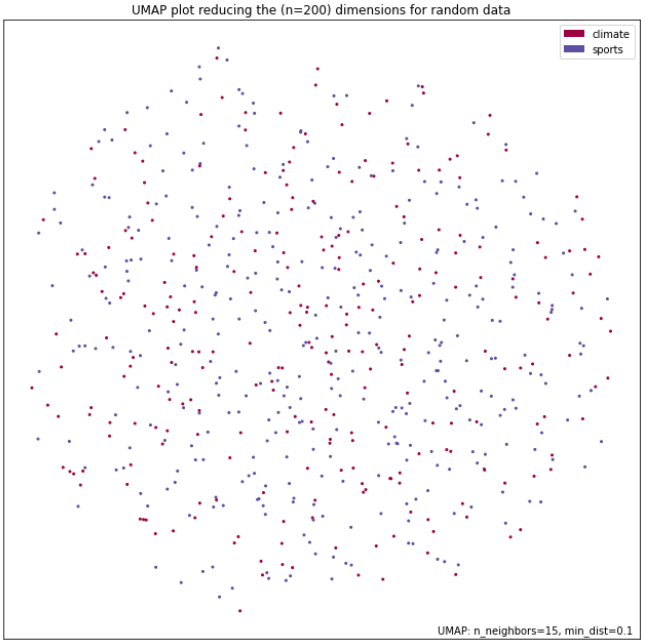
UMAP plot reducing the (n=200) dimensions for the classes in the training set

UMAP plot reducing the (n=200) dimensions for the classes in the testing set

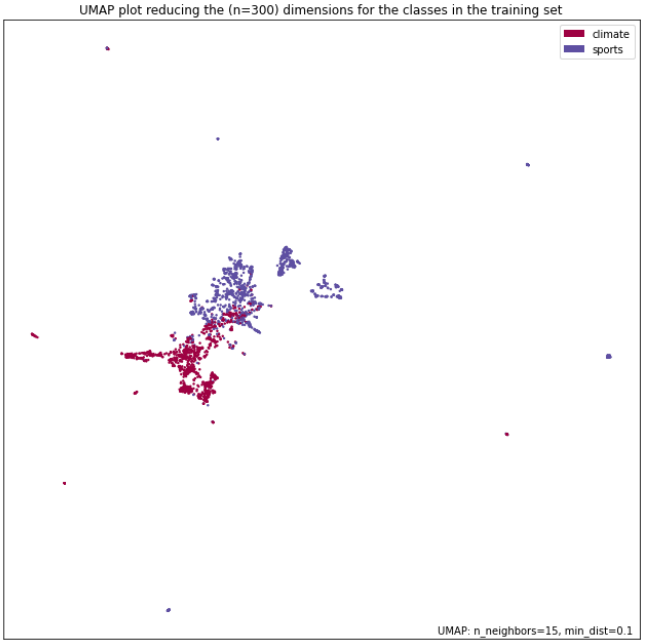
UMAP plot reducing the (n=200) dimensions for random data



UMAP plot reducing the (n=200) dimensions for random data

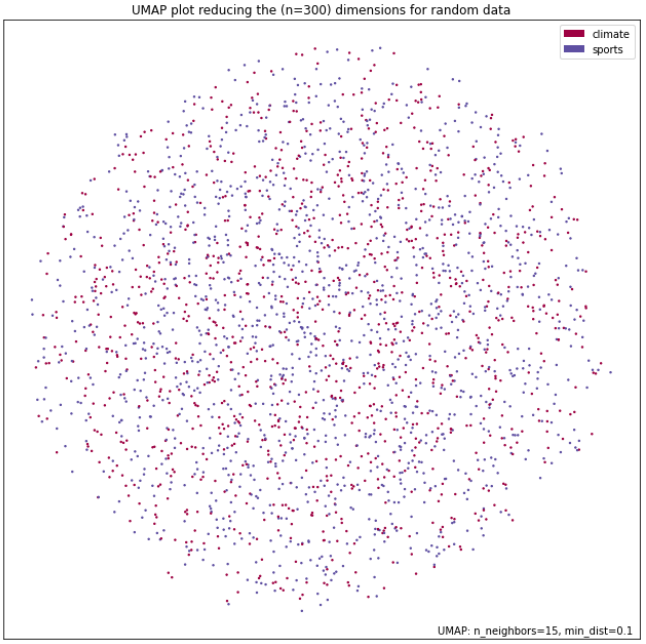


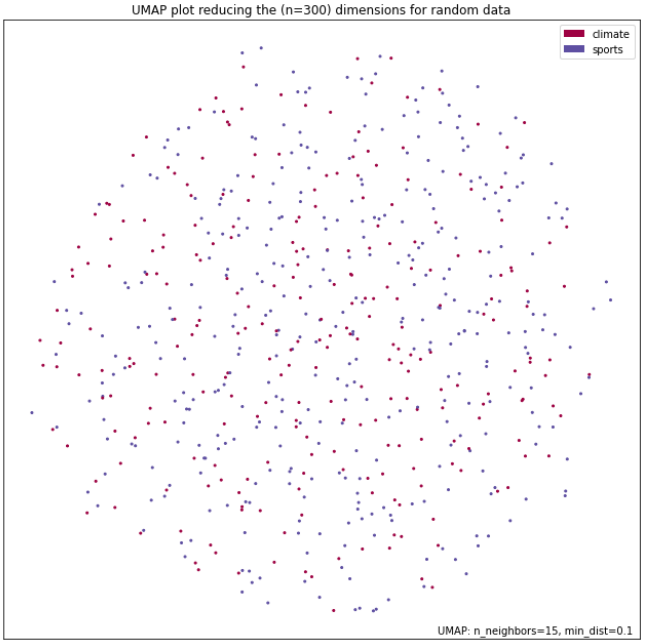
UMAP plot reducing the (n=300) dimensions for the classes in the training set



UMAP plot reducing the (n=300) dimensions for the classes in the testing set

UMAP plot reducing the (n=300) dimensions for random data




UMAP plot reducing the (n=300) dimensions for random data
climate
sports
UMAP: n_neighbors=15, min_dist=0.1