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

Sidarth Srinivasan (UID - 005629203), Shweta Katti (UID - 505604846)

January 25, 2022

Course : ECE 219 Large Scale Data Mining
Term : Winter 2022

The aim of this project is to build an end-to-end pipeline to classify the given dataset of news articles. The classification of this text data is done by first performing feature extraction using a TF-IDF matrix, the dimensionality of the matrix is then reduced using two methods: PCA and NMF. These extracted features are then classified using a few simple classification models such as SVM, Linear/Logistic classification. To evaluate and diagnose the classifiers, we plot the ROC plot and confusion matrix. We also use metrics such as accuracy, precision, and F1 score to evaluate the classifiers. We perform a grid search and cross-validation to find the optimal combination of hyperparameters. Multiclass classification and GLoVE word embeddings are also explored in this project.

1 Question 1

- **Overview:**
  For the given dataset we found 2072 samples (rows) with 9 categories for the features (columns).
- **Histograms:**

  Histogram 1a :Total number of alphanumeric characters per data point.

  In Figure 1, X axis represents the Count and the Y axis represents the Frequency.

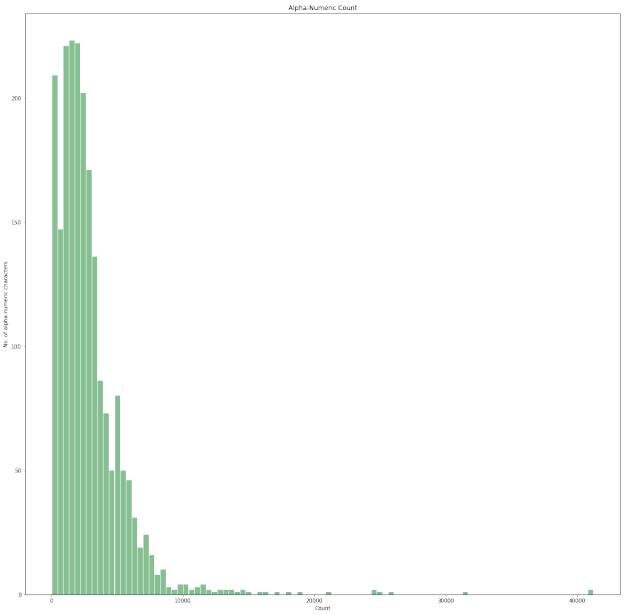


Figure 1: Total number of alphanumeric characters per data point.

Histogram 1b : Leaf Label - Class plot

In Figure 2, X axis represents the Leaf Label and the Y axis represents the Frequency of each leaf label.

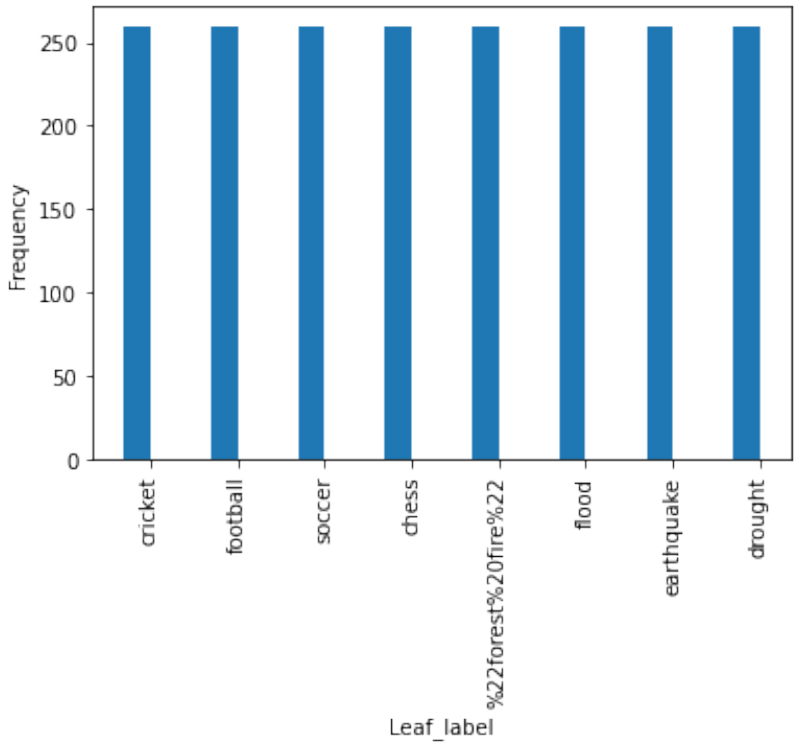


Figure 2: Leaf label - Class plot

Histogram 1c: Root label - Class plot

In Figure 3, X axis represents the Root Label and the Y axis represents the Frequency of each root label.

- **Qualitative Interpretations :**

From histogram 1b) and 1c), we observe that the dataset is evenly distributed, with each class containing roughly the same number of documents. This is desired for classification tasks so as to prevent the model from overfitting on classes that have a higher number of samples than the others. However, it is possible to be presented with a dataset with uneven class distributions and in that case the following techniques could be followed:

  - Oversampling/Data Augumentation: The technqiue that synthetically increases the number of training samples for the class with lower samples by adding slightly modified copies of the minority class samples.
  - Undersampling: A technique where we remove few samples from the majority class to equalize the overall class distribution. However, the downside of this method is that we lose information.
  - Choosing pertinent models and Loss functions: Decision Tress and Ensemble of one-vs-all classifiers performs well on unbalanced datasets. This is done so by modifying the Loss function that adds penalty terms for misclassfying minority class samples and ensure faster convergence. This method is also called as Cost Sensitive Learning.

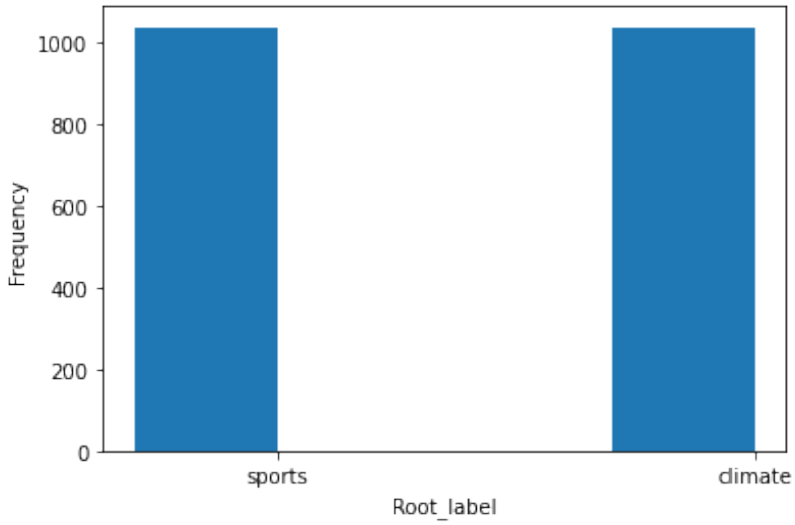


Figure 3: Root Label - Class plot

- Choosing appropriate performance metrics: Accuracy for unevenly distributed class sampls might not be accurate measure and hence we could use metrics such as F1 score, precision, recall and confusion matrix that includes class-dependent perspectives.

2 Question 2

- After splitting the dataset into training and test samples, we then obtain the following split
  - Number of data points in training set = 1657
  - Number of data points in test set = 415

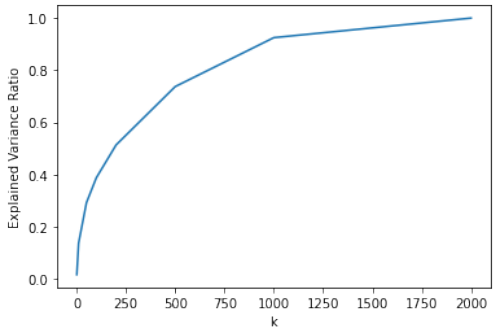
3 Question 3 - Feature Extraction

- Stemming and Lemmatization are techniques that are used for word normalization. Stemming removes the affixes of words in order to map the given word to a root word. Lemmatization on the other hand is a more advanced version of stemming as it converts the words to a more context specific root word called lemma. It uses POS (Parts-of-Speech) tags to map the words to the most appropriate lemma by using semantics and semantic context of the word in the sentence. Stemming is easier and faster to implement but is less accurate when compared to Lemmatization and hence, we prefer lemmatization.
- Minimum document frequency is used to remove terms that appear too infrequently. To remove such infrequent terms, we set min_df to 3. This removes all terms that have a count of less than 3 in the training set. Decreasing the value of min_df increases the size of the resulting bag-of-words matrix. For example, setting the min_df to 1 would result in ignoring no terms and hence more BOW compared to min_df set to 3.
- Stop words, punctuation and numbers are removed before the lemmatization process.
- Size of training data after lemmatization but before TF-IDF: (1657, 10159)
  Size of testing data after lemmatization but before TF-IDF: (415, 10159)

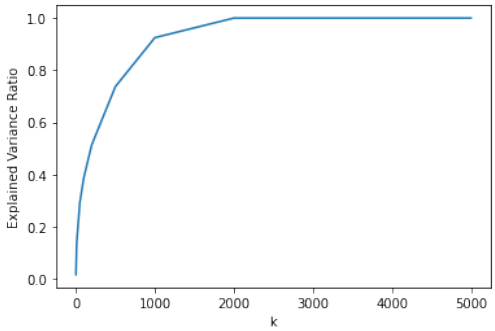
Shape of train TF-IDF matrix: (1657, 10159)
Shape of test TF-IDF matrix: (415, 10159)

4 Question 4 - Dimensionality Reduction

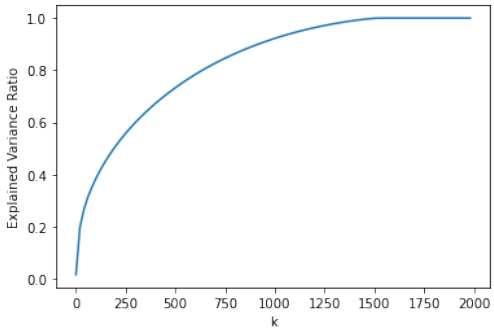
- **Explained Variance Ratio:**



(a) Plot for k = (1, 10, 50, 100, 200, 500, 1000, 2000)



(b) Plot for k = (1, 10, 50, 100, 200, 500, 1000, 2000, 2500, 3000, 5000)



(c) Plot for 200 values of k between 0 and 2000

Figure 4: Explained Variance Plots

As seen from the figure 4 , the plot of the Explained Variance Ratio for various values of k are shown. Figure 4a) and 4b), we can infer that the curve plateaus after beyond k = 2000. The main motive of performing LSI is to reduce the number of feature dimensions along with preserving class information. Hence any k greater than 2000 is not desired as the EVR remains a constant as the best 2000 components captures essential feature information. We could see that the explained variance ratio increases steeply at first and then gradually plateaus. In order to obtain a smoother plot, we increased the number of values of k ranging between 0 and 2000 to obtain plot 4c). Hence, we could say that the **nature of the plot is concave downwards** as the gradient increases and then decreases.

- The reconstruction residual MSE error with k=50 were as follows:
  - **LSI for train data = 31.79317655905423**

  - **NMF for train data = 32.1469440048339**

  - **LSI for test data = 15.032038613679044**
  - **NMF for test data = 16.465989044673552**

- The **MSE for NMF** was found to be **larger** than that of LSI. The reason being we know in NMF, the matrix R is expressed as product of two matrices W & H where R is a sparse matrix. Since NMF has a requirement that the components of the matrices be non-negative whereas LSI does not have any such requirement, preserving the significant features are preserved better in LSI. The TF-IDF matrix for our case is not sparse and due to the above reason, we observe better MSE error results for LSI than for NMF and hence we can conclude that LSI is better than NMF for this case.

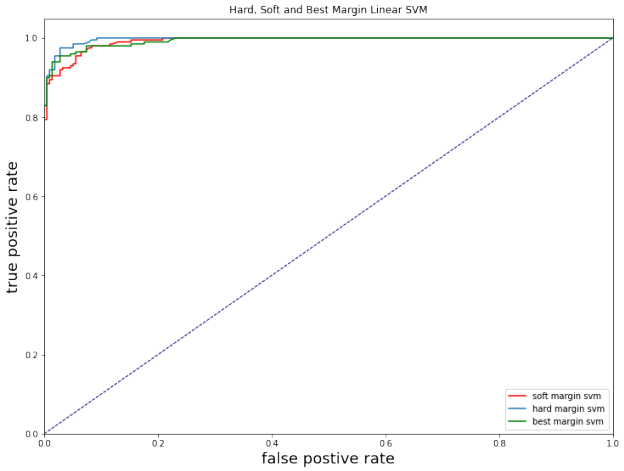
5 Question 5 - SVM

We trained 3 SVMs for various values of γ . We considered $\gamma = 1000$ for the hard margin SVM, $\gamma = 0.0001$ for the soft margin SVM and another SVM with a very large of $\gamma = 100000$. The observed results such as ROC curve, confusion are represented in the figures 6,7 and tabulated below.

- Based on the results, it can be inferred that the **Hard Margin SVM** performs **better** than the **Soft Margin SVM** model. This could be since SVMs are highly sensitive to the penalty term γ. In a soft margin SVM, the goal is to allow the classifier make some misclassifications by allowing a wider margin such that other samples can still be classified even when the classes are linearly inseparable. A higher recall for the Soft SVM can be explained to the reason that the soft margin classifier could correctly label all items to a certain class C, but it might also misclassify due to leniency in the misclassification allowance. A low γ could underfit the data as observed for the soft margin SVM. Similarly, we can observe that a higher γ is not always the best option as seen with $\gamma = 100000$. As in this case, the accuracy drops as a high γ overfits the data. Hence, we need to make sure to choose a γ value such that it neither overfits (very large γ value) nor underfits (small γ value) the data.
- The **Soft Margin** case clearly **underfits** the data. This could be supported by the confusion matrix (see figure 4b). The Soft Margin classifier either has a *high true positive rate* with a *low false positive* **or** a *low true positive rate* with a *high false positive rate* and hence the model fails to identify a state that has both good true positive rate and false positive rate. This is further supported by the top-left region of the ROC curve as seen in figure 7), the **ROC plot for the Soft Margin does not look competitive** as the turn at the top-left corner reflects our above argument.
- To find the **optimal value of** γ, we performed a 5-fold cross-validation on the dataset. As seen from figure 5, the best value for $\gamma = 10$ was observed. The ROC curve, performance metrics and the confusion matrix have all been tabulated in the following figures. It can be inferred that the γ found through Cross-validation is optimal as it neither overfits nor underfits the data unlike the Hard and Soft SVM cases.
- From the ROC curve, the Area under the curve were also calculated
  - Area under hard SVM curve = 0.9965452261306532
  - Area under soft SVM curve = 0.9923110924995349
  - Area under best SVM curve = 0.9957077051926299

	param_C	mean_test_score	rank_test_score
0	0.001	0.505129	10
1	0.01	0.619188	9
2	0.1	0.952331	8
3	1	0.960781	2
4	10	0.968025	1
5	100	0.958967	3
6	1000	0.958363	4
7	10000	0.958363	4
8	100000	0.958363	4
9	1000000	0.958363	4

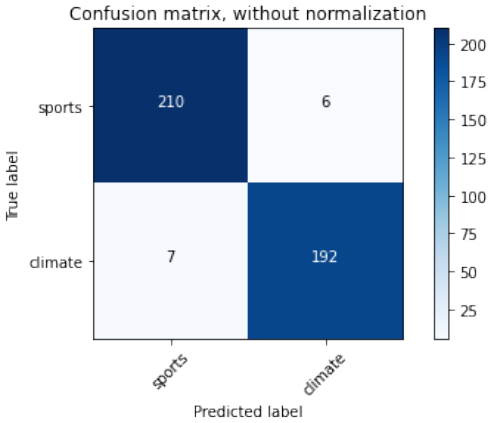
(a) Optimal γ value for SVM



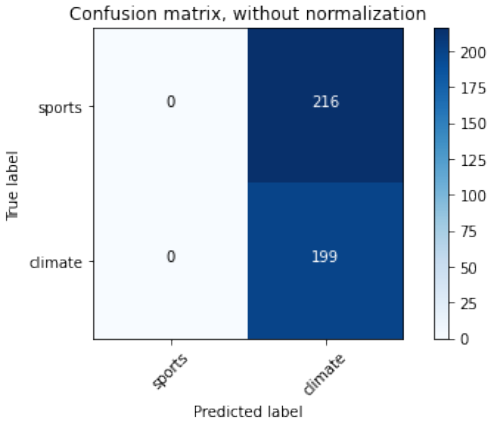
(b) ROC for Hard, Soft and Best Margin Linear SVM

Figure 5: Optimal value and ROC for SVM

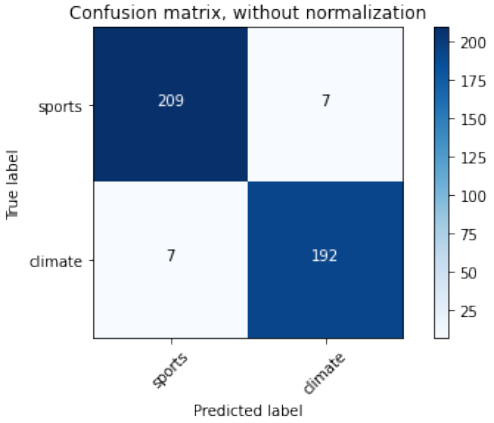
SVM WITH LSI				
Metrics	Hard SVM	Soft SVM	With $\gamma = 100000$	Optimal $\gamma = 10$
Accuracy	0.9686746987951808	0.4795180722891566	0.9662650602409638	0.963855421686747
Precision	0.9696969696969697	0.4795180722891566	0.964824120603015	0.9742268041237113
Recall	0.964824120603015	1.0	0.964824120603015	0.949748743718593
F1 Score	0.9672544080604534	0.6482084690553745	0.964824120603015	0.9618320610687022



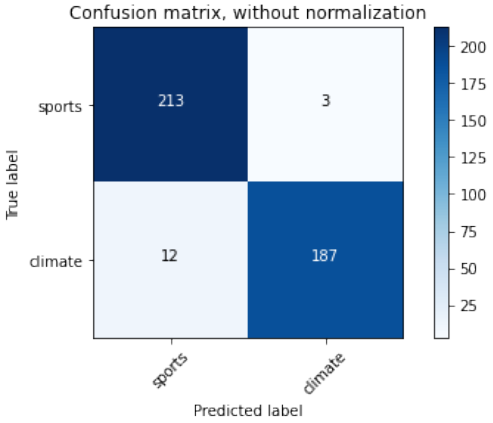
(a) HARD SVM



(b) SOFT SVM



(c) SVM WITH $\gamma = 100000$



(d) WITH CROSS VALIDATION BEST $\gamma = 10$

Figure 6: Confusion Matrices.

6 Question 6 - Logistic Regression

- We trained three Logistic Regression models using LSI as the dimensionality reduction method: with no regularization, L1 regularized and L2 regularized. To find the best C value, we perform 5-fold cross-validation and the results obtained were as shown in figure 8. Based on the average test scores, we got **C=10 for L1 regularization, C=100 for L2 regularization** and the average test score seemed to remain the same throughout various C values for no regularization.

- L1 regularized logistic regression model seemed to have a higher accuracy and precision when compared to L2 regularization and no regularization. We also note that the model with no regularization has similar performance metric values as the L2 regularized model. L2 regularization usually helps to prevent the model from over-fitting, but inclusion of the same, however does not make a significant difference in the test error for this dataset.

- A model with no regularization can be used when we would want to build complex models that follows the training data closely. Both L1 and L2 shrink the coefficients of the model to prevent overfitting. L1 regularization drives a few coefficients to be exactly 0. It is useful for feature selection as it can expel certain features (which are unimportant) from the model. On the other hand, though L2

performs a similar shrinking, it drives the coefficients close to 0 but not exactly to 0 i.e, it keeps all the variables/features. Hence, we may use L1 when we have sparse outputs and we need to select only a few important features of the data available to us. L2 could be used when we want to avoid overfitting but also when every available features are significant and need to be used.

- Logistic regression forms a decision boundary based on the probability of achieving maximum probability for the correct classification of datapoints and hence, it uses all the available data points. Linear SVM on the other hand, uses only those data points that are on the support vectors as it forms the decision boundary such that it maximizes the distance of the margin between the closest support vector. Based on the models' dependencies on data points, we can say that Linear SVM is robust to outliers and Logistic Regression is more likely to overfit the data. Hence, SVM will perform better than Logistic Regression when there are outliers and if we are dealing with high dimensional data.

	param_C	mean_test_score
0	0.0001	0.959569
1	0.001	0.959569
2	0.01	0.959569
3	0.1	0.959569
4	1	0.959569
5	10	0.959569
6	100	0.959569
7	1000	0.959569
8	10000	0.959569

(a) For no regularization

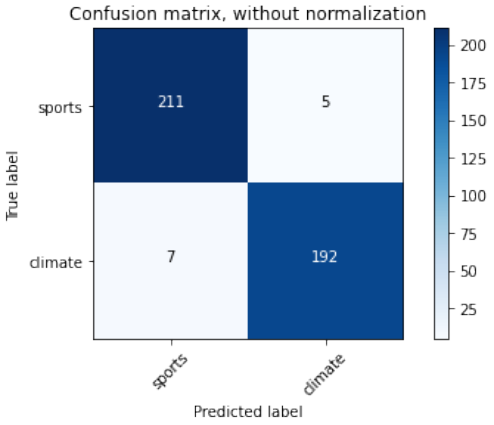
	param_C	mean_test_score
0	0.0001	0.494871
1	0.001	0.494871
2	0.01	0.494871
3	0.1	0.901625
4	1	0.946901
5	10	0.963799
6	100	0.961380
7	1000	0.959568
8	10000	0.960172

(b) For L1 regularization

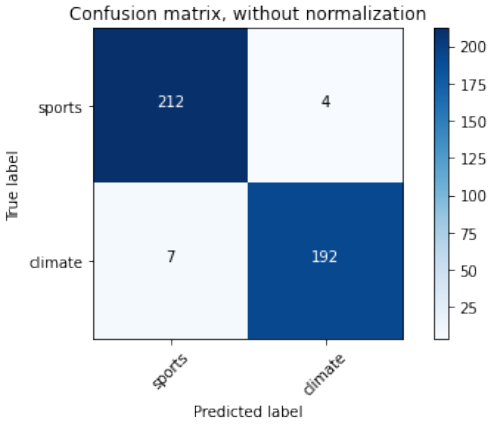
	param_C	mean_test_score
0	0.0001	0.701827
1	0.001	0.788141
2	0.01	0.923967
3	0.1	0.948102
4	1	0.955345
5	10	0.959573
6	100	0.962587
7	1000	0.965002
8	10000	0.958963

(c) For L2 regularization

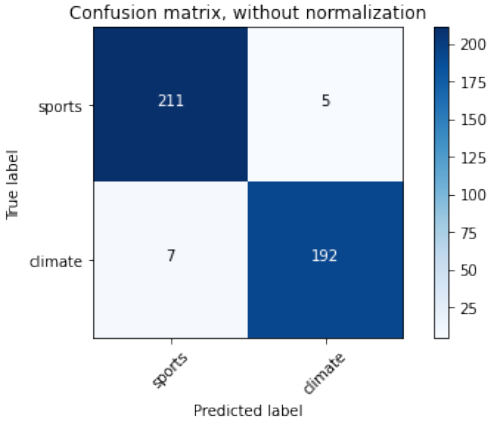
Figure 7: Best k values for Logistic Regression.



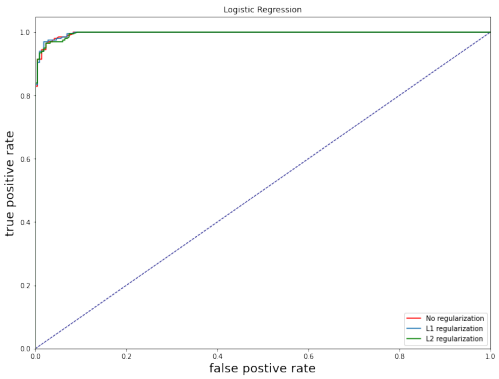
(a) For no regularization



(b) For L1 regularization



(c) For L2 regularization



(d) ROC Plot

Figure 8: Confusion Matrix and ROC Plot

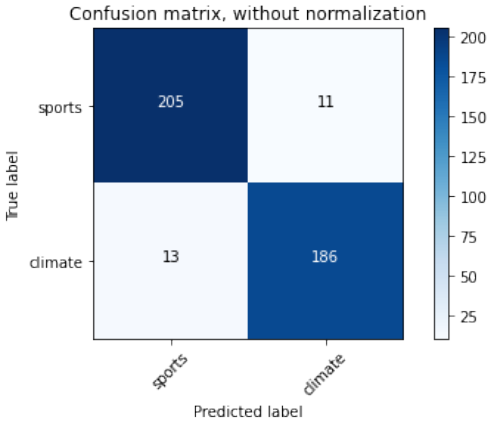
From the ROC Plot:

- area under no regularization 0.9967778708356597
- area under L1 regularization 0.9970105155406664
- area under L2 regularization 0.9965219616601525

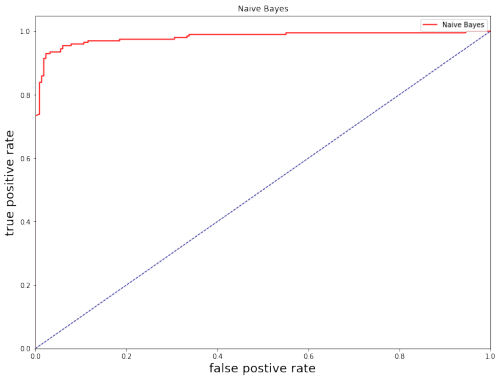
LOGISTIC REGRESSION WITH LSI			
Metrics	No regularization	L1 regularization	L2 regularization
Accuracy	0.9710843373493976	0.9734939759036144	0.9710843373493976
Precision	0.9746192893401016	0.9795918367346939	0.9746192893401016
Recall	0.964824120603015	0.964824120603015	0.964824120603015
F1 Score	0.9696969696969697	0.9721518987341772	0.9696969696969697

7 Question 7 - Gaussian Naive Bayes Classifier

From the obtained results, we can see the the Gaussian Naive Bayes is not the best classifier model for our dataset. The performance metrics had lower values when compared to the other models we have explored in the previous sections.



(a) Confusion Matrix



(b) ROC for Gaussian NB

NAIVE BAYES WITH LSI	
Metrics	Values
Accuracy	0.9421686746987952
Precision	0.9441624365482234
Recall	0.9346733668341709
F1 Score	0.9393939393939393

8 Question 8 - Grid Search

In order to find the best hyper-parameters, we constructed multiple pipelines for grid search. Pipelines for both cleaned and uncleaned data were built. We also used various feature extraction, dimensionality reduction and classifier models as given in the project specifications. To ensure easy debugging and parallel run time, multiple pipelines were built and run across different local machines.

The top 5 best parameters based on test score accuracies obtained were for cleaned data are reported below. We've also reported the best score for the unclean data pipeline.

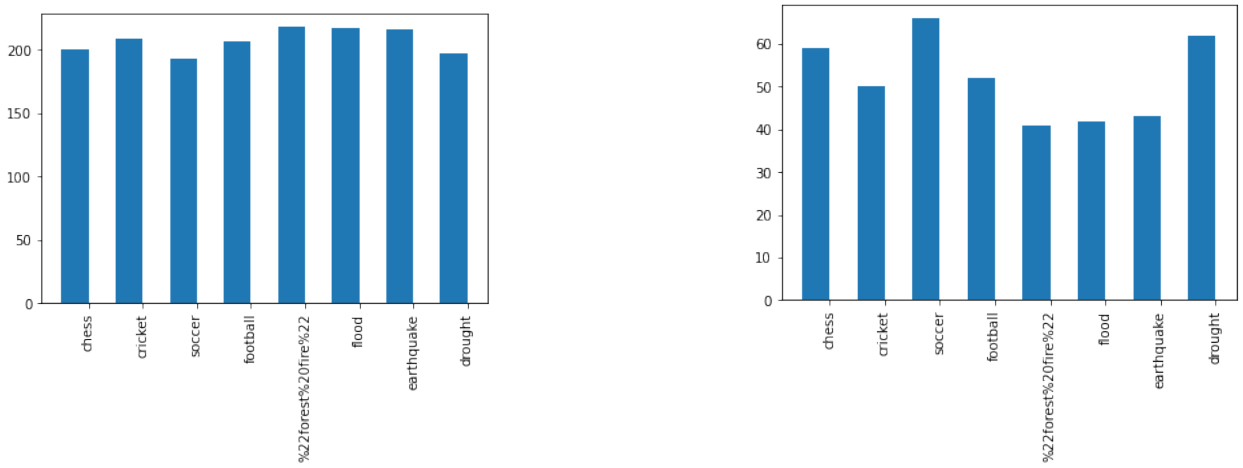
Clean Data, k = 500				
Classifier	Feature Extraction	Dimensionality Reduction	min_df	Mean Test Score
Logistic Regression (with L1)	Lemmatization	LSI	5	0.9776762639682600
Logistic Regression (with L2)	Lemmatization	LSI	3	0.9776726240308670
Logistic Regression (with L1)	NIL	LSI	5	0.9764659847850620
Logistic Regression (with L2)	Lemmatization	LSI	5	0.9764659847850620
Logistic Regression (with L2)	Stemming	LSI	5	0.9758672150839010

Unclean Data, k = 500				
Classifier	Feature Extraction	Dimensionality Reduction	min_df	Mean Test Score
Logistic Regression (with L2)	NIL	LSI	5	0.9728410217863214

9 Question 9 - Multiclass Classification

For the multi-class classification, we consider Naive Bayes, Linear SVM classifier models and perform the One vs One and One vs Rest techniques. We generate models for both LSI, NMF and with no feature reduction techniques (just tf-idf feature extraction) with setting min_df = 5.

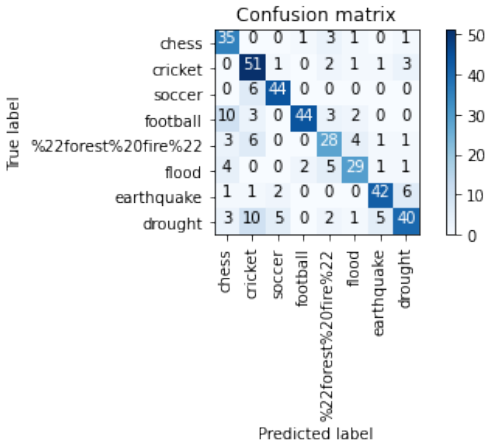
- The data distribution for both test and train data for leaf_label is as shown in Figure 9a) and 9b). We could observe that few leaf labels classes are imbalanced and hence to avoid the imbalance issue, downsampling could be performed.
- As seen in the results tabulated, SVM with One vs Rest performed the best, followed by SVM with One vs One. Both of which performed the best with no dimensionality reduction technique.
- From the confusion matrices of the various models, we notice a distinct visible diagonal. The diagonal elements of a confusion matrix represent the labels that are predicted correctly and the off-diagonal elements are the misclassified labels. Hence, we can say that distinct visible blocks in the matrix leads to higher accuracy of the model.
- When we merge the leaf_labels into a higher root_label, say sports and climate, we observe that the issue with class imbalance is dealt with and the same models give a better accuracy in that case.
- We can say that the Naive Bayes does not perform well across any formation. The SVM One vs One and SVM One vs Rest performs fairly similar as we the models use mirco averaging options. The micro averaging option calculates the metrics globally by considering the total true positives, false negatives, and false positives and thus the scores are fairly similar.



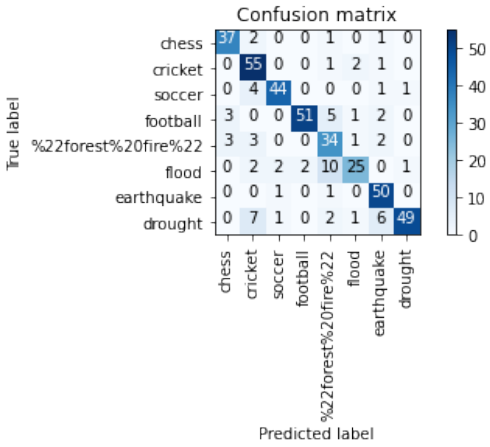
(a) For Train set

(b) For Test set

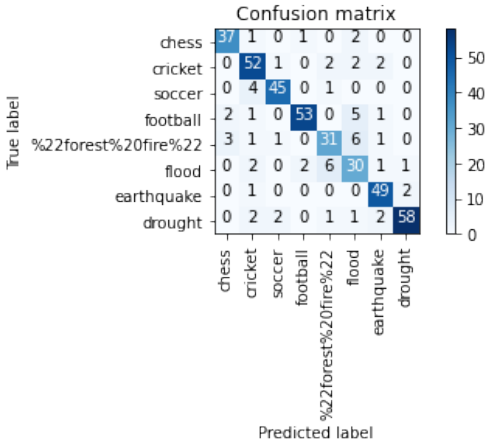
Figure 9: Leaf Label distribution for MultiClass Classification



(a) With tfidf

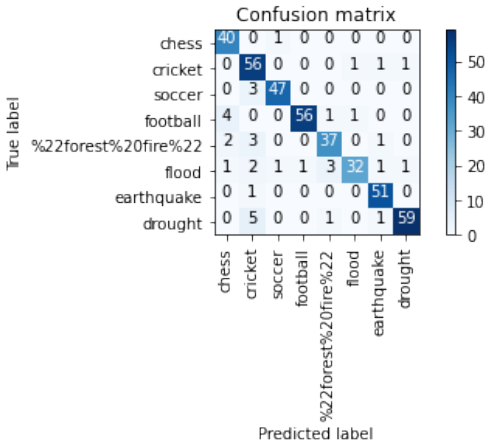


(b) With LSI

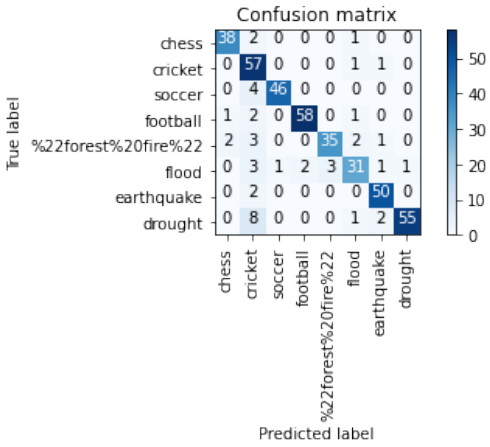


(c) With NMF

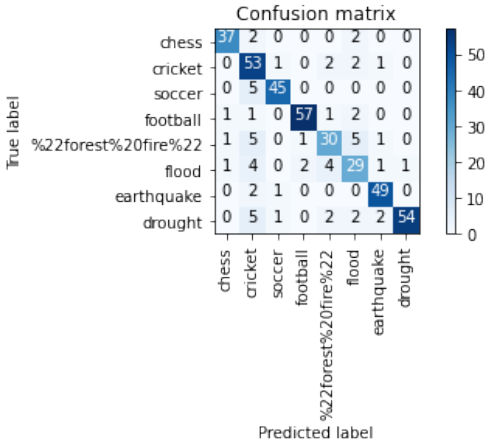
Figure 10: Confusion Matrix for Gaussian Naive Bayes Multiclass



(a) With tfidf

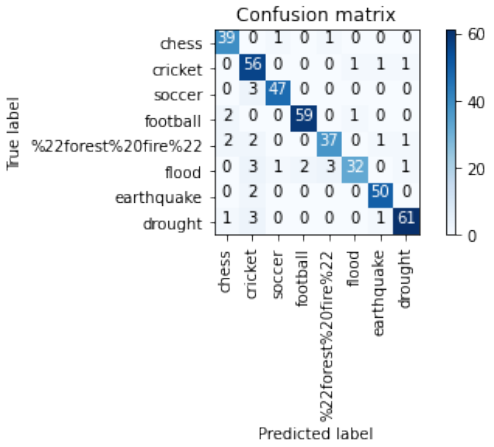


(b) With LSI

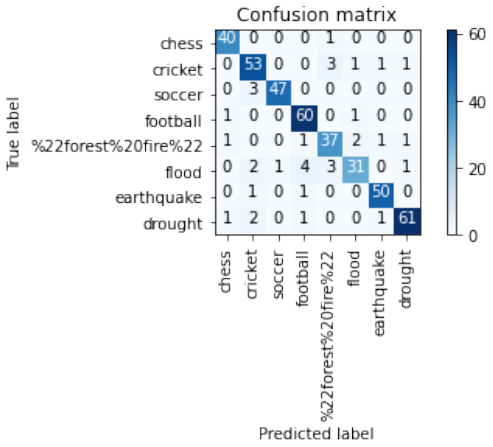


(c) With NMF

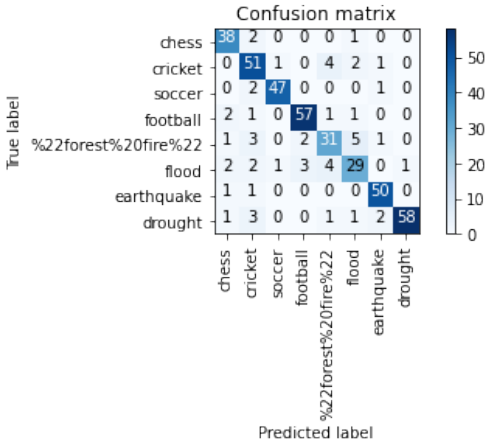
Figure 11: Confusion Matrix for One vs One SVM Multiclass



(a) With tfidf



(b) With LSI



(c) With NMF

Figure 12: Confusion Matrix for One vs Rest SVM Multiclass

MULTICLASS CLASSIFICATION									
Classifiers:	Gaussian NB			SVM One vs One			SVM One vs Rest		
Metrics	TFIDF	LSI	NMF	TFIDF	LSI	NMF	TFIDF	LSI	NMF
Accuracy	0.7542	0.8313	0.8554	0.9108	0.8916	0.8530	0.9181	0.9132	0.8698
Recall	0.7542	0.8313	0.8554	0.9108	0.8916	0.8530	0.9181	0.9132	0.8698
Precision	0.7542	0.8313	0.8554	0.9108	0.8916	0.8530	0.9181	0.9132	0.8698
F1 Score	0.7542	0.8313	0.8554	0.9108	0.8916	0.8530	0.9181	0.9132	0.8698

10 Question 10

- The group of contextual words that are relevant to the target words are separated better by using the ratio of co-occurrence probabilities. The following cases are to be considered :
  - The co-occurence ratio will be very large or small for those group of words that are related to any of the target words. In other words, we say that if a word is related to a target word that contributes to the numerator term of the co-occurence ratio, then the ratio would be large and similarly, if the word is related to a target word that contributes to the denominator term of the the co-occurrence ratio, then the ratio would be small.
  - For words that are not related to both target words or for words that are present in both target words (words that does not provide context), the ratio would be close to one.
- No, the GLoVE embedding does not give the same vector for "running" in this scenario. GLoVE takes into account the contextual dependencies of the word in the given sentence. Thus the neighbor words "park" and "presidency" would play a significant role in the vector representation of the word "running".
- The values we got by implementing the code given in the project specification were as follows.
  - $||GLoVE[queen]||_2 - ||GLoVE[king]||_2 - ||GLoVE[wife]||_2 + ||GLoVE[husband]||_2 = 6.165036$
  - $||GLoVE[queen]||_2 - ||GLoVE[king]||_2 = 5.966258$
  - $||GLoVE[wife]||_2 - ||GLoVE[husband]||_2 = 3.1520464$
  - We converted the glove embeddings to Word2Vec format and the similarity scores between 'queen' and 'king' and then 'husband' and 'wife' were calculated by finding out the closest word embedding of 'king' and 'wife'.
    - 'king' most similar word is 'queen', with a similarity score of 0.636.
    - 'wife' most similar word is 'husband', with a similarity score of 0.8646.
  - The spatial distance of 'king' from 'queen' is greater than that of 'husband' and 'wife' as explained by a higher L2 norm.
  - We could also observe that $||GLoVE[queen]||_2 - ||GLoVE[king]||_2 - ||GLoVE[wife]||_2 + ||GLoVE[husband]||_2$ is similar to $||GLoVE[queen]||_2 - ||GLoVE[king]||_2$ and this shows that analogical word embedding need not form perfect parallelograms.
- Lemmatization is chosen over stemming as the former considers the context, semantics and syntax, and POS tokens to derive meaningful root forms called lemmas which is not the case with stemming that does not perform any morphological analysis. This can lead to stemmed words having no meaning or incorrect spelling. Although stemming is faster and computationally efficient than stemming, it has a higher error rate. Thus it is better to choose lemmatization to ensure a broader context of words are passed onto the GLoVE embeddings.

11 Question 11

The feature engineering process is highlighted below :

- We first convert the GLoVE embeddings into a word2vec file buy using a keyed vector model from Gensim. This stores the GLoVE embedding words in an embedded vector format.
- We then clean the data by removing any redundant information such as punctuation, numbers, HTML links, etc.

- We then convert each sentence into vectors of size 300 as this is the size of the GLoVE embeddings file provided to us and which works the best. We then need to make sure that all the features which are described in the vectors and that are significant are based in the pre-trained GLoVE embeddings file.To do this, we take the average of the ith values of the vectors of the word embeddings of all the words in a given sentence.
- As a final step, we normalize the final vectors in order to aggregate them into a single vector as suggested in the project specification.

We chose a Linear SVM as our classifier with $\gamma = 10$. The following metrics were obtained :

- Accuracy : 0.9614457831325302
- Recall : 0.9669811320754716
- Precision : 0.9579439252336449
- F1-Score : 0.9624413145539905

12 Question 12

Based on the Accuracy vs Dimension plot for GLoVE using Linear SVM as a classifier, we can state that as the dimension of the GLoVE embedding increases, so does the accuracy of the test set. This is an expected result as more dimension suggests that there is more information and semantics available between the target words and the context words. Thus, providing us with better feature dependencies and hence, a more accurate model.

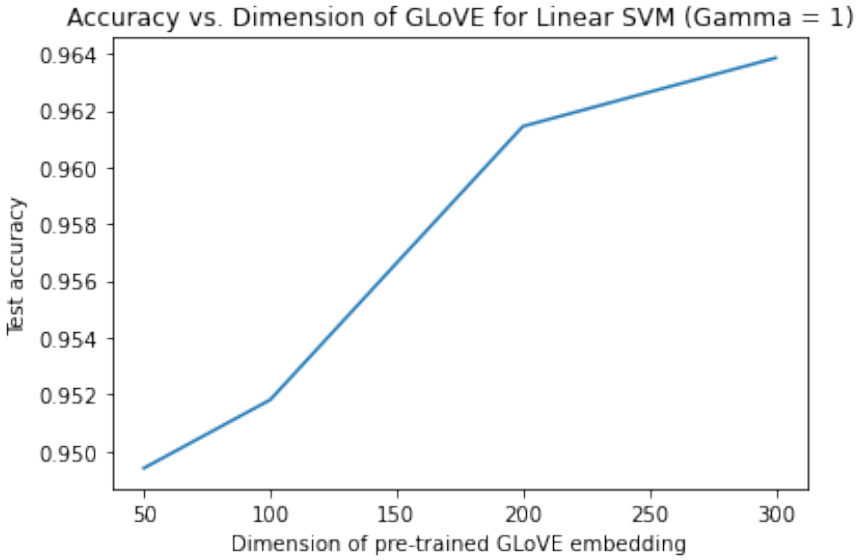
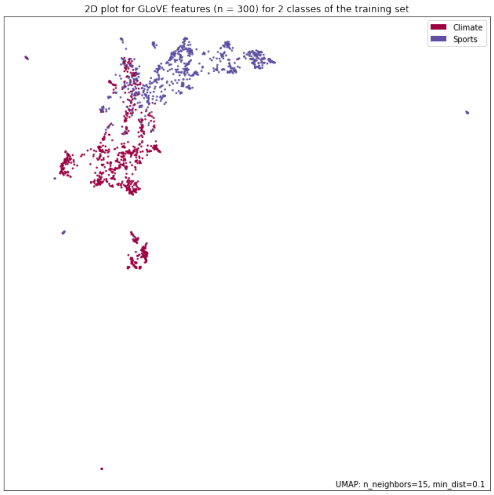


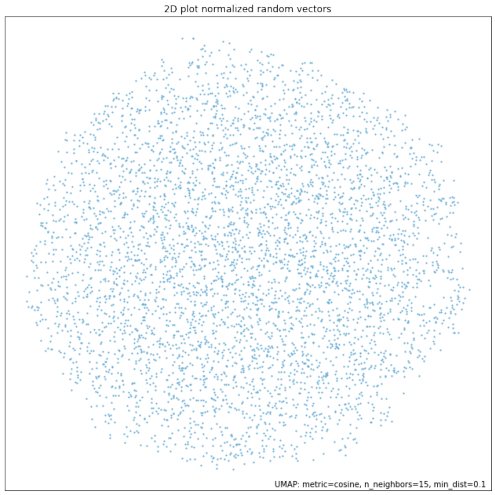
Figure 13: Accuracy vs Dimension plot for GLoVE

13 Question 13

On visualizing the 2D plots for GLoVE and random vectors with the same dimension as the GLoVE embeddings, we can see that distinct clusters are formed only in the GLoVE model (Figure 14.a) and not for the random vectors (Figure 14.b). This shows that GLoVE embedings are succesfully distinguishing the dataset features for the succesful learning of the classifiers and all the points are evenly distributed.



(a) GLoVE Clusters using UMAP



(b) Random Vectors of same dimensions as the GLoVE embeddings

Figure 14: Visualizing 2D plots