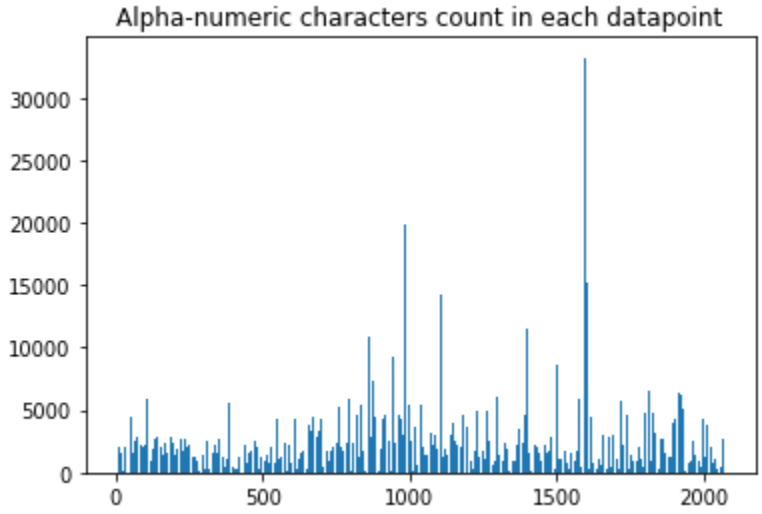
ECE-219

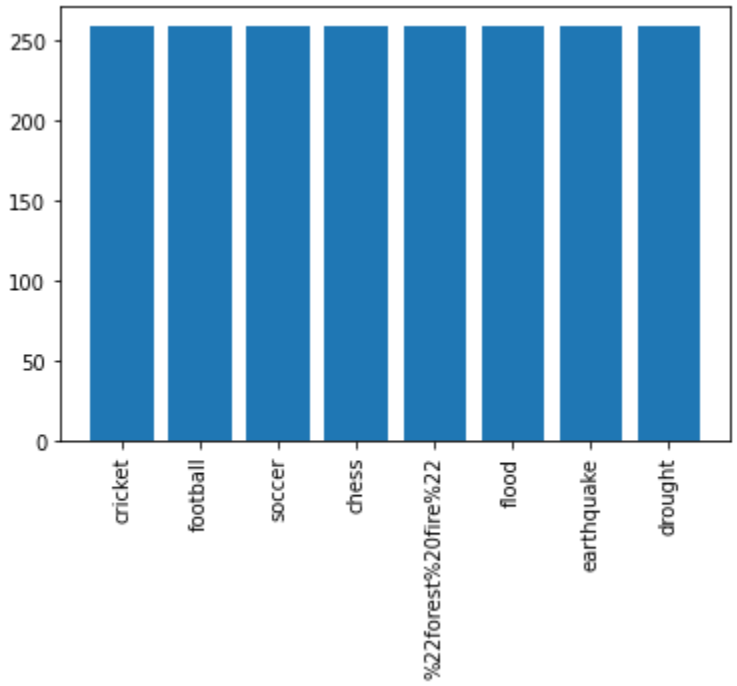
End-to-End Pipeline to Classify News Articles Report

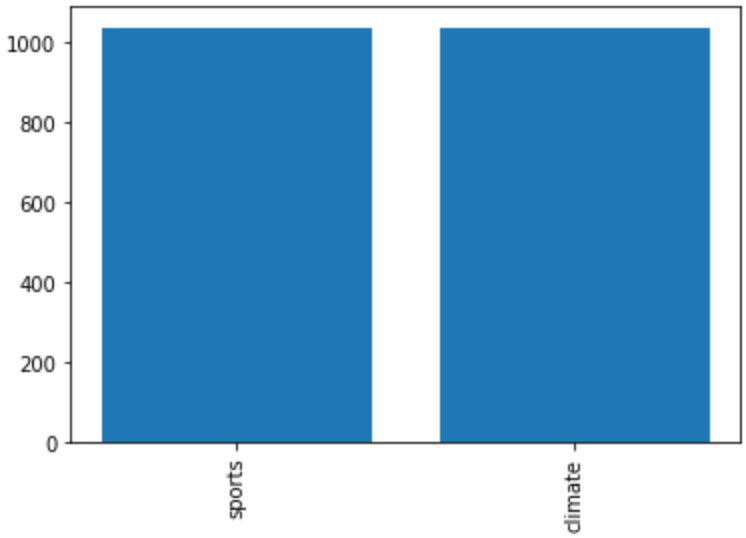
Team Member Names:

Tianpei Gu, 405863048
Yuxin Huang, 105711853
Yilin Xie, 405729012

Question 1

- The dataset has 2072 rows (samples) and 9 columns (features).
- Histograms & Interpretations
  - 

  - 



-
- Note that we only care about alpha-numeric characters which are used for the following section **feature extraction**, the rest will be cleaned.
    - From the histogram that counts the number of alpha-numeric characters, we can see that each row has enough information (over 1,000 characters) that can be used in **feature extraction**.
    - From the histogram that uses “leaf_label” and “root_label” on the x-axis, we can see that all categories have about the same number of data points.

Question 2

- The training data has 1657 samples and the testing data has 415 samples.

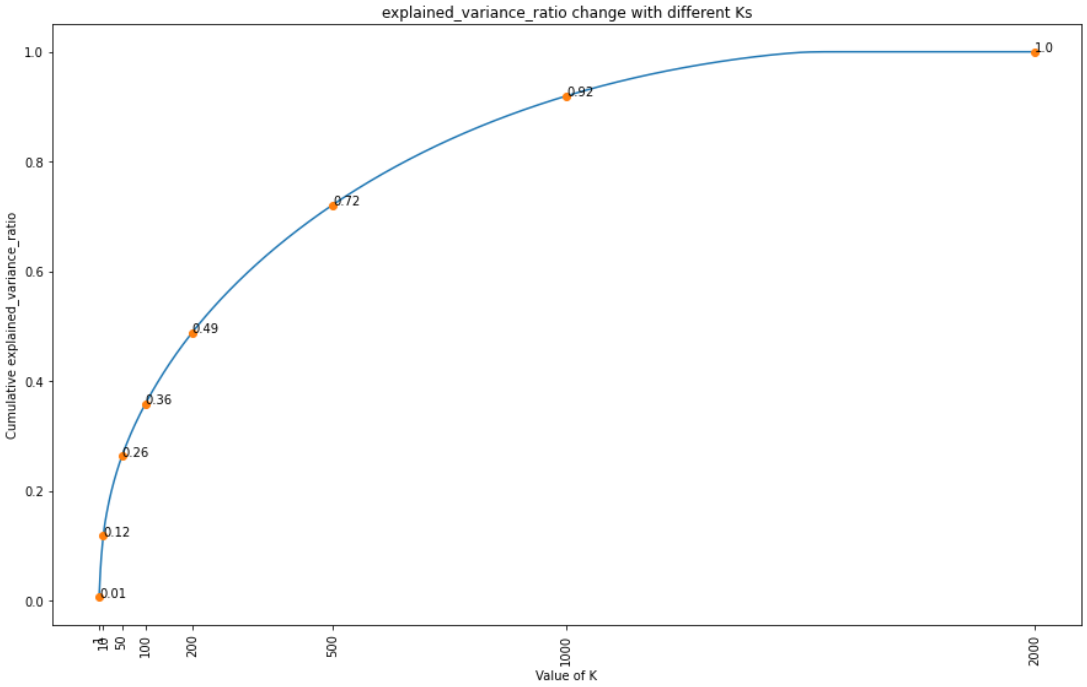
Question 3

- Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.
    - **Pros**: Less expensive and fast to run.
    - **Cons**: The output may be inaccurate.
- Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.
    - Pros: More sophisticated than stemming; produce accurate dictionary words by utilizing part-of-speech.
    - Cons: Slower and harder to implement.
- **Lemmatization needs a bigger dictionary size.**
- As min-df increases, the dimension of the TF-IDF matrix decreases.
- Punctuations and numbers should be handled before lemmatizing, stopwords will be handled after lemmatizing, and numbers should be removed before lemmatizing.

- Based on the print result above, the shape of the TF-IDF matrix on the training data is (1657, 9575) and that on the testing data is (415, 9575).

Question 4

- Explained Variance Ratio Plot for K = [1, 10, 50, 100, 500, 1000, 2000]

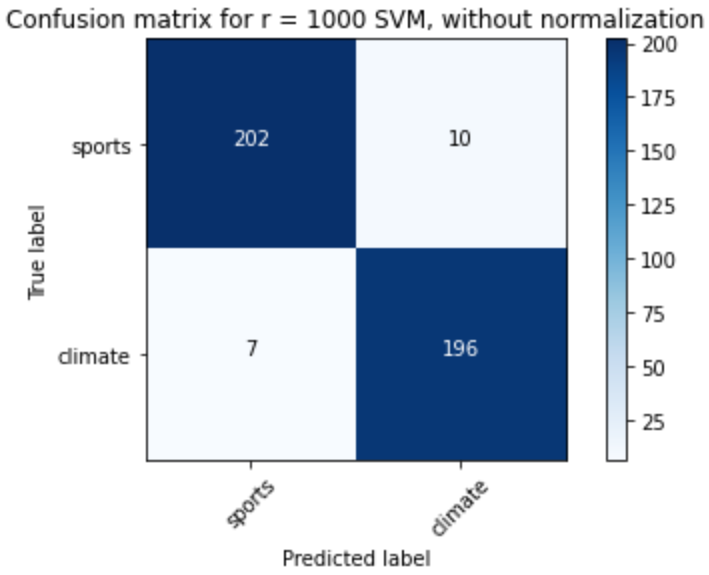


- The explained variance ratio is the percentage of variance that is attributed by each of the selected components. From the concavity plot, as more principal components are selected, the higher percentage of the variance of the data is explained. Besides, we observe that the curve has a negative second derivative, which means that as more dimensions are added, the less increase of information will be gained by the marginal dimensions.

- The error by LSI is slightly slower than NMF, this is expected since SVD guarantees to produce minimum MSE. On the other hand though, NMF has random factors and might not be optimal, and it requires w vector and H to be ≥ 0 which constrains the search space.

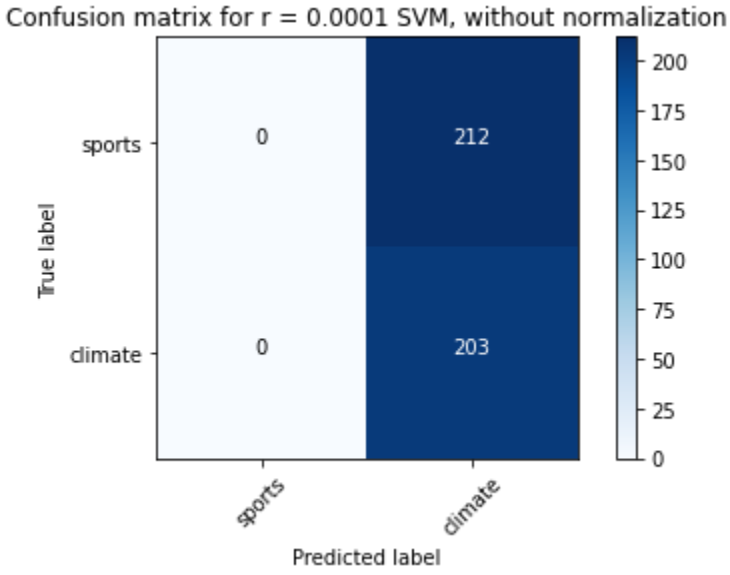
Question 5

- gamma: 1000
    - Accuracy: 0.9590361445783132
    - Precision: 0.9514563106796117
    - Recall: 0.9655172413793104
    - F-1 Score: 0.9584352078239609



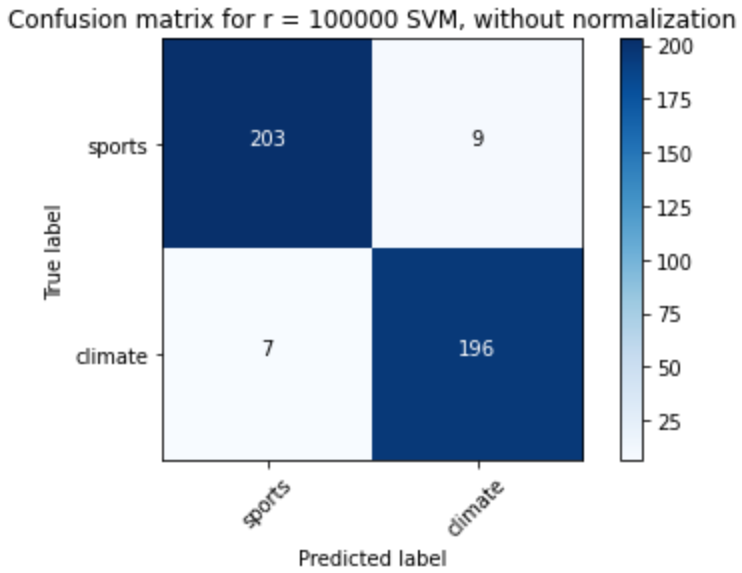
-

- gamma: 0.0001
    - Accuracy: 0.4891566265060241
    - Precision: 0.4891566265060241
    - Recall: 1.0
    - F-1 Score: 0.656957928802589



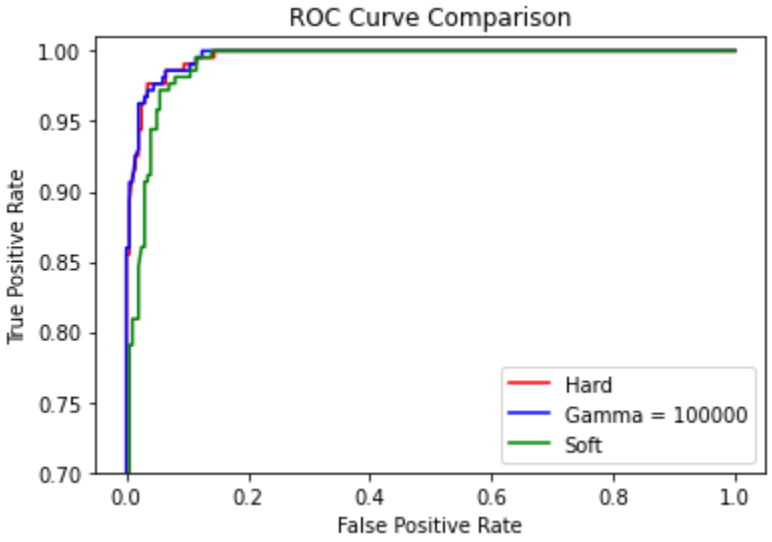
-

- gamma: 100000
    - Accuracy: 0.9614457831325302
    - Precision: 0.9560975609756097
    - Recall: 0.9655172413793104
    - F-1 Score: 0.9607843137254901



-

- ROC Comparison

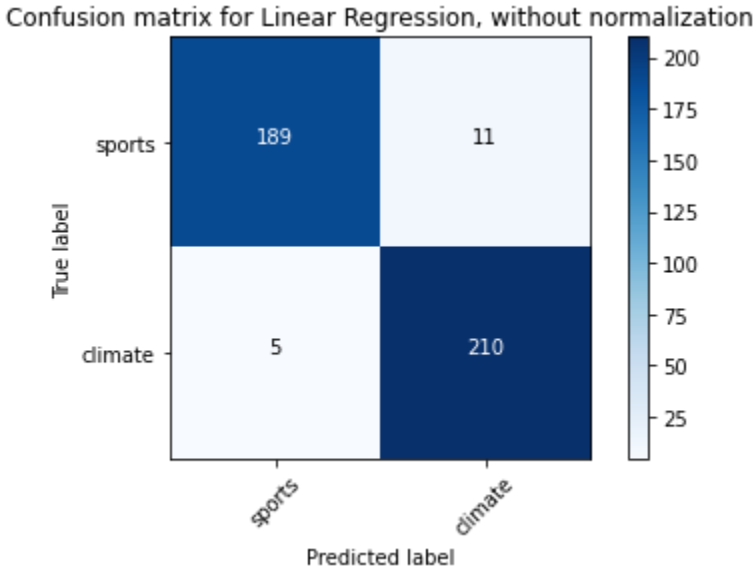


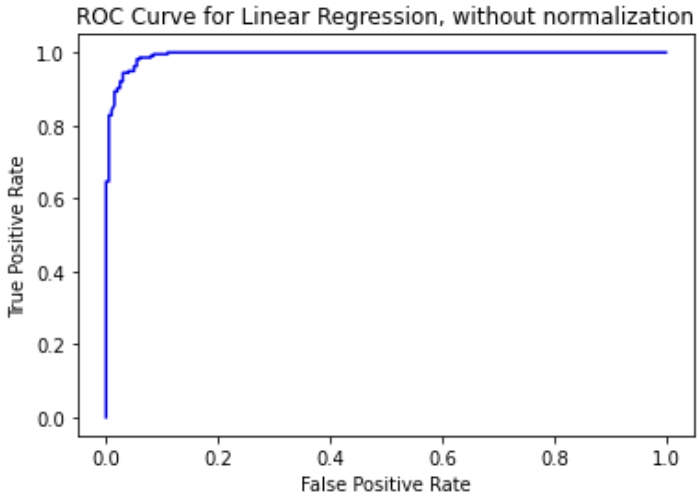
-

- Soft SVM does not perform well. This is because γ = 0.0001 is so small that the model is very lenient towards misclassification of quite a few individual points.
  In the aspect of confusion matrix, we can see that all climate label data points are misclassified. The ROC curve is not competitive because the area below the curve is less than the area below the curve when γ = 1000 or γ = 100000.
- The reason could be that the soft margin SVM has a hyperplane whose intercept b is miserably incorrect even if the w vector is optimized, which means that the margin is not fully maximized.
- The best γ using cross validation is γ = 10 according to the function **`svc_cv.best_params_['C']`**:
  - γ = 10
    Accuracy: 0.963855421686747
    Precision: 0.9532710280373832
    Recall: 0.9760765550239234

F-1 Score: 0.9645390070921985
Confusion matrix, without normalization:
[[196 10]
[5 204]]

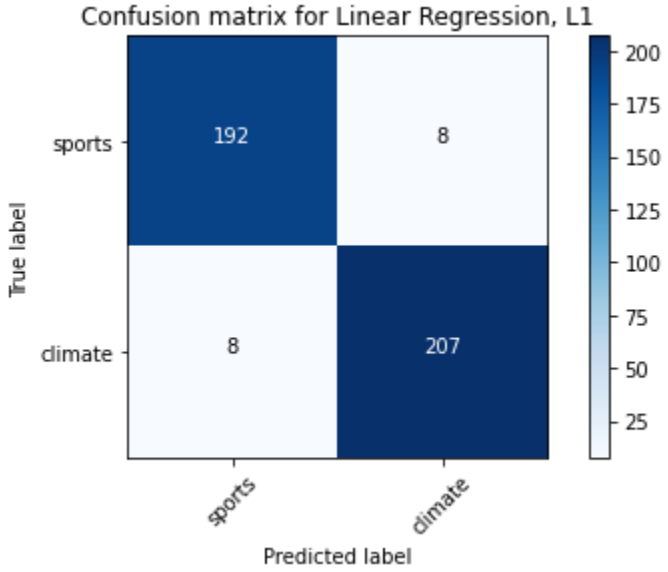
Question 6

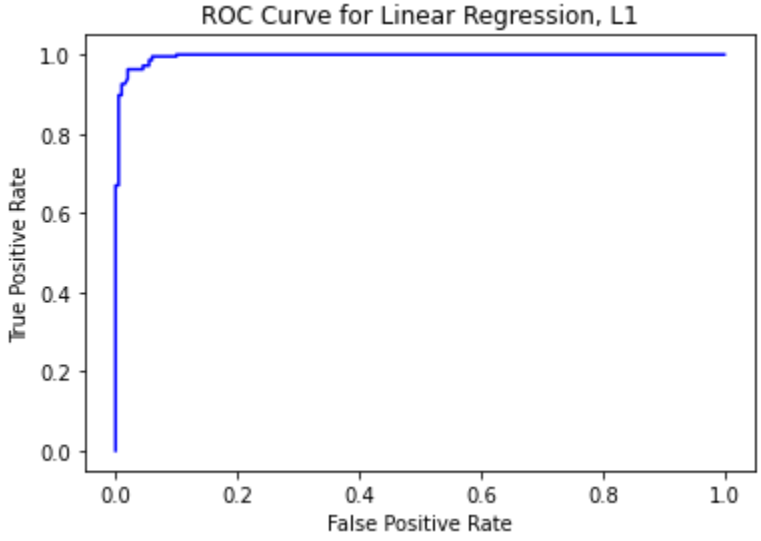
- Without regularization
  - Accuracy: 0.9614457831325302
  - Precision: 0.9502262443438914
  - Recall: 0.9767441860465116
  - F-1 Score: 0.963302752293578
  - CNF & ROC Curve
  - 

  - 

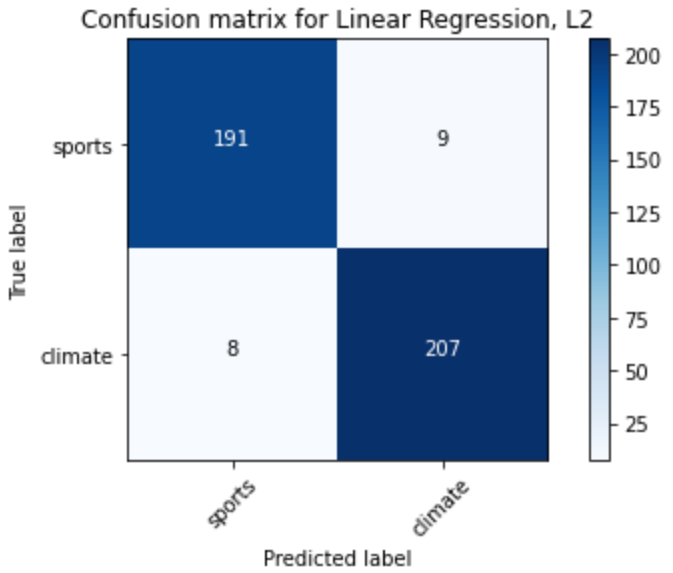
- L1 regularization
  - Best C using 5-fold cross-validation = 10
  - Accuracy: 0.9614457831325302
  - Precision: 0.9627906976744186

- Recall: 0.9627906976744186
- F-1 Score: 0.9627906976744186
- CNF & ROC Curve

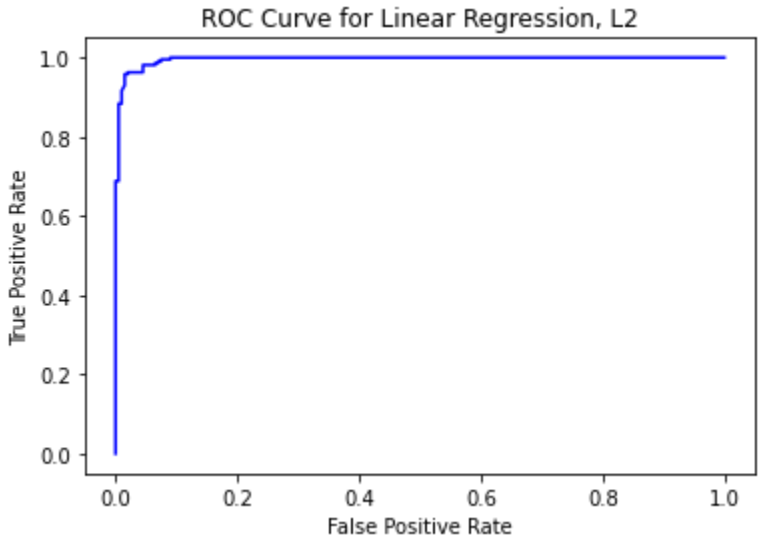




- L2 regularization
  - Best C using 5-fold cross-validation = 100
  - Accuracy: 0.9590361445783132
  - Precision: 0.9583333333333334
  - Recall: 0.9627906976744186
  - F-1 Score: 0.9605568445475637
  - CNF & ROC Curve



-



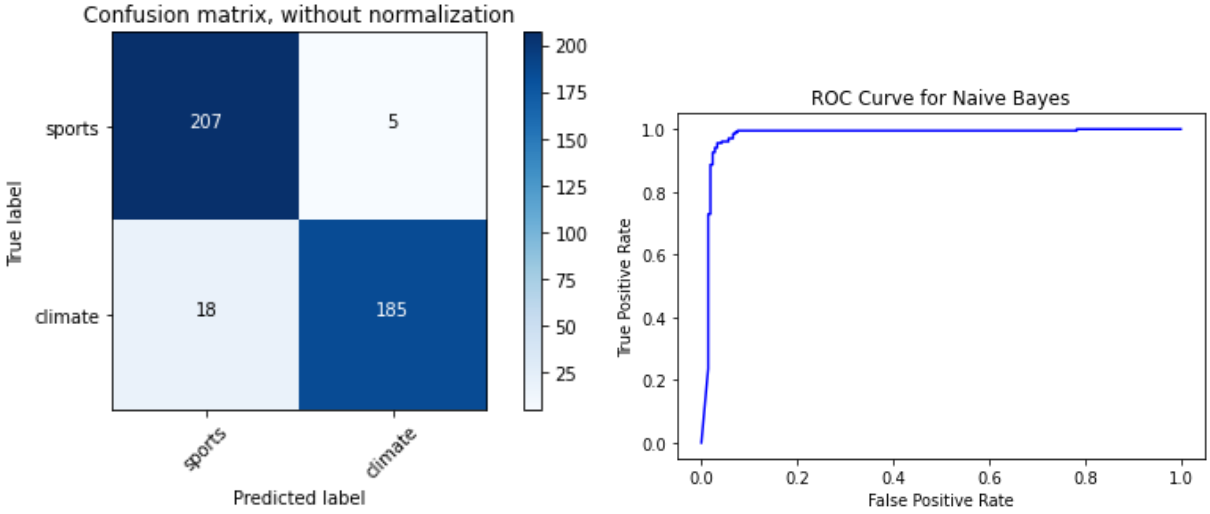
-

- Regularization prevents the model from overfitting. It significantly reduces the variance of the model, without substantial increase in its bias. For regularization parameters, Increasing lambda results in less overfitting but also greater bias.
  For the learnt coefficients, the smaller the coefficients, the slower the model learns the training set.
- L1 regularization is the preferred choice when having a high number of features as it provides sparse solutions. L2 Regularization shrinks all the weights to small values, preventing the model from overfitting.
- SVM uses the kernel trick to find the best line separator, yet logistic regression uses sigmoid function to find the relationship between variables.
  They perform differently because SVM tries to find the best margin that separates the classes and this reduces the risk of error on the data. Logistic regression, on the other hand, has different decision boundaries with different weights that are near the optimal point.

Logistic regression and SVM with a linear kernel have similar statistical performance but depending on the features, one may be more efficient than the other.

Question 7

- Accuracy: 0.944578313253012
- Precision: 0.9736842105263158
- Recall: 0.9113300492610837
- F-1 Score: 0.9414758269720103
- CNF & ROC Curve



Question 8

- Clean + Lemmatize:

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
6	LogisticRegression(C=100, random_state=42)	NMF(init='random', n_components=500, random_st...	3	0.974662
4	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.974058
0	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.972852
1	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.972852
9	LogisticRegression(C=10, penalty='l1', random_...	TruncatedSVD(n_components=500, random_state=42)	5	0.972245

- Clean + Nothing

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
5	SVC(C=10, random_state=42)	NMF(init='random', n_components=5, random_stat...	5	0.510578
30	GaussianNB()	NMF(init='random', n_components=50, random_sta...	3	0.508763
31	GaussianNB()	NMF(init='random', n_components=50, random_sta...	5	0.506958
9	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=5, random_state=42)	5	0.502126
1	SVC(C=10, random_state=42)	TruncatedSVD(n_components=5, random_state=42)	5	0.502115

- Clean + Stemming

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
28	GaussianNB()	NMF(init='random', n_components=5, random_stat...	3	0.520196
24	GaussianNB()	TruncatedSVD(n_components=5, random_state=42)	3	0.517186
25	GaussianNB()	TruncatedSVD(n_components=5, random_state=42)	5	0.516584
7	LogisticRegression(C=100, random_state=42)	NMF(init='random', n_components=500, random_st...	5	0.512973
30	GaussianNB()	NMF(init='random', n_components=50, random_sta...	3	0.511176

- No Clean + Lemmatization

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
0	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.976471
4	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.975869
5	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.974659
1	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.974056
7	LogisticRegression(C=100, random_state=42)	NMF(init='random', n_components=500, random_st...	5	0.971039

- No Clean + Nothing

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
30	GaussianNB()	NMF(init='random', n_components=50, random_sta...	3	0.505731
7	SVC(C=10, random_state=42)	NMF(init='random', n_components=50, random_sta...	5	0.504517
16	LogisticRegression(C=10, penalty='l1', random_...	TruncatedSVD(n_components=5, random_state=42)	3	0.502723
10	LogisticRegression(C=10, penalty='l1', random_...	NMF(init='random', n_components=500, random_st...	3	0.497902
8	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=5, random_state=42)	3	0.497894

- No Clean + Stemming

	param_classifier	param_dim_reduction	param_vect__min_df	mean_test_score
4	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.978273
5	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.978273
0	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.975258
9	LogisticRegression(C=10, penalty='l1', random_...	TruncatedSVD(n_components=500, random_state=42)	5	0.973443
1	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.972238

- **In all, the best five would be:**

  No Clean + Stemming

4	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.978273
5	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	5	0.978273

  No Clean + Lemmatization

0	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.976471
4	LogisticRegression(C=100, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.975869

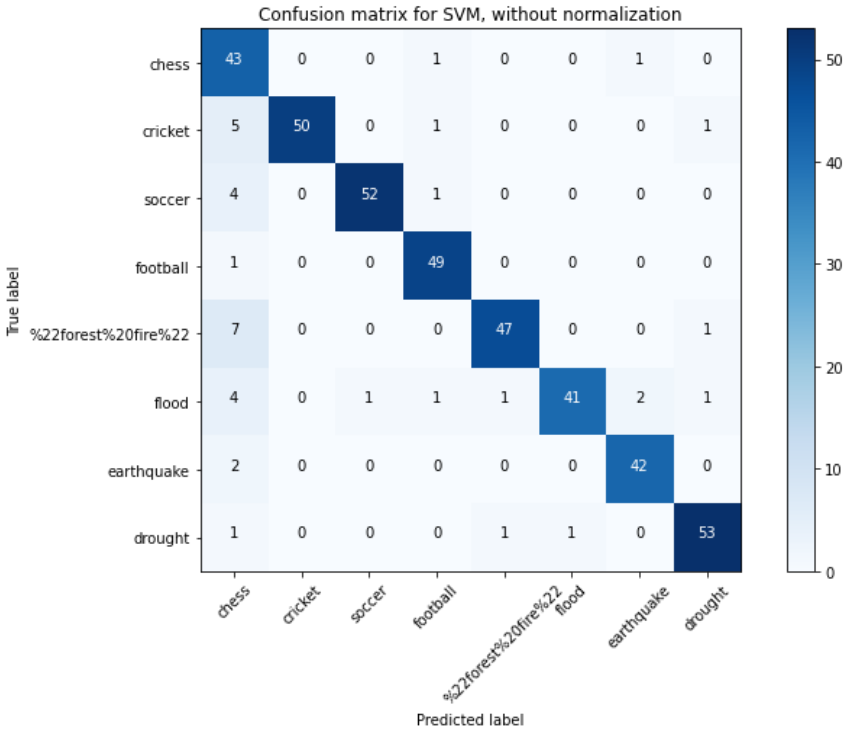
No Clean + Stemming

-

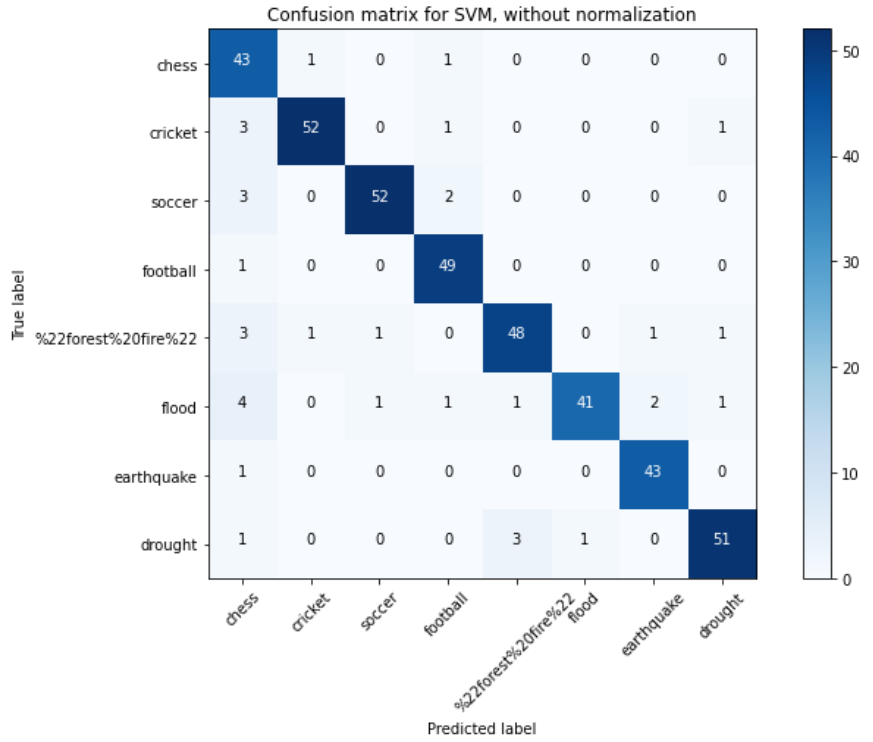
0	SVC(C=10, random_state=42)	TruncatedSVD(n_components=500, random_state=42)	3	0.975258

Question 9

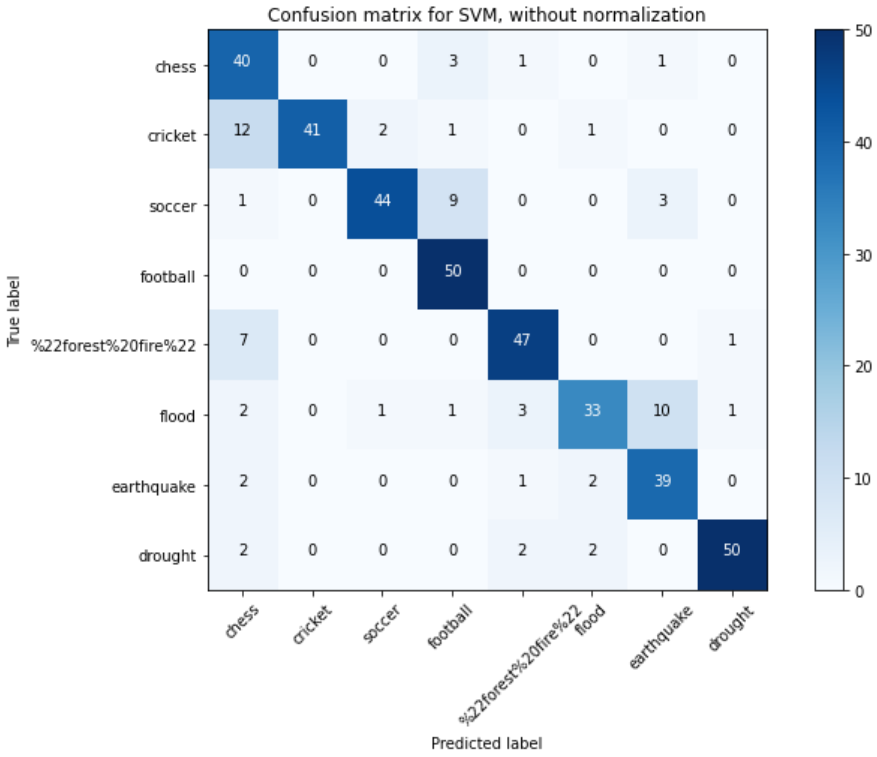
- To resolve the class imbalance issue in the One VS the rest model, we decided to use the “class_weight” attribute in the model. According to the document, The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))
- Based on the confusion matrix, there are distinct visible blocks on the major diagonal, which means that most of the categories or classes are correctly predicted.
- The One VS One and One VS Rest have comparatively better results than the GaussianNB. Therefore, based on the confusion matrix of GaussianNB, we can see a comparatively greater false positive on the intersection of class “soccer” & “football”, and “flood” & “earthquake”, and we can merge them into 2 larger classes respectively.
- Before merging
    - One VS One
        - Accuracy: 0.908433734939759
        - Precision: 0.908433734939759
        - Recall: 0.908433734939759
        - F-1 Score: 0.908433734939759



    - One VS Rest
        - Accuracy: 0.9132530120481928
        - Precision: 0.9132530120481928
        - Recall: 0.9132530120481928
        - F-1 Score: 0.9132530120481928

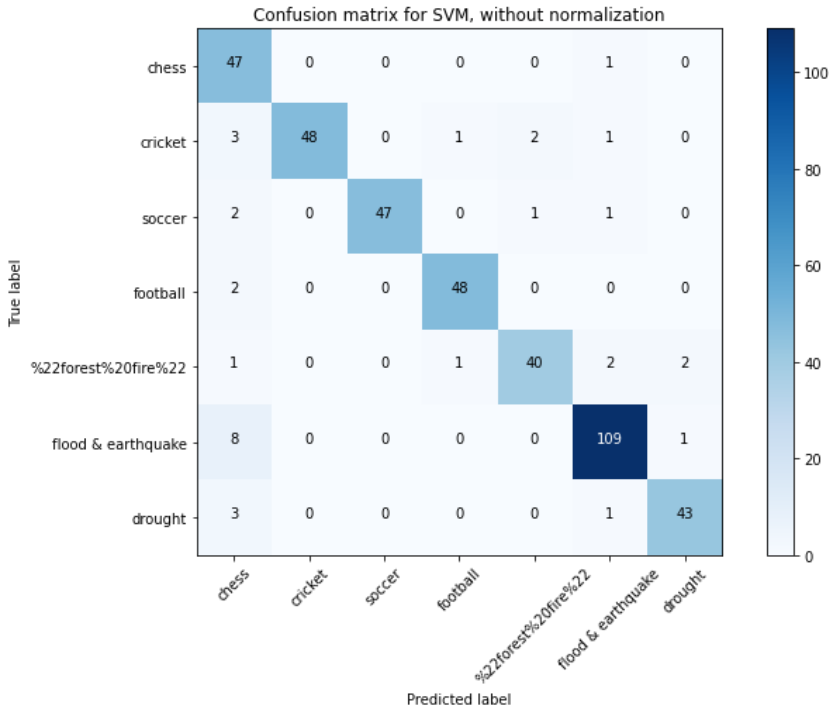


- GaussianNB
  - Accuracy: 0.8289156626506025
  - Precision: 0.8289156626506025
  - Recall: 0.8289156626506025
  - F-1 Score: 0.8289156626506025

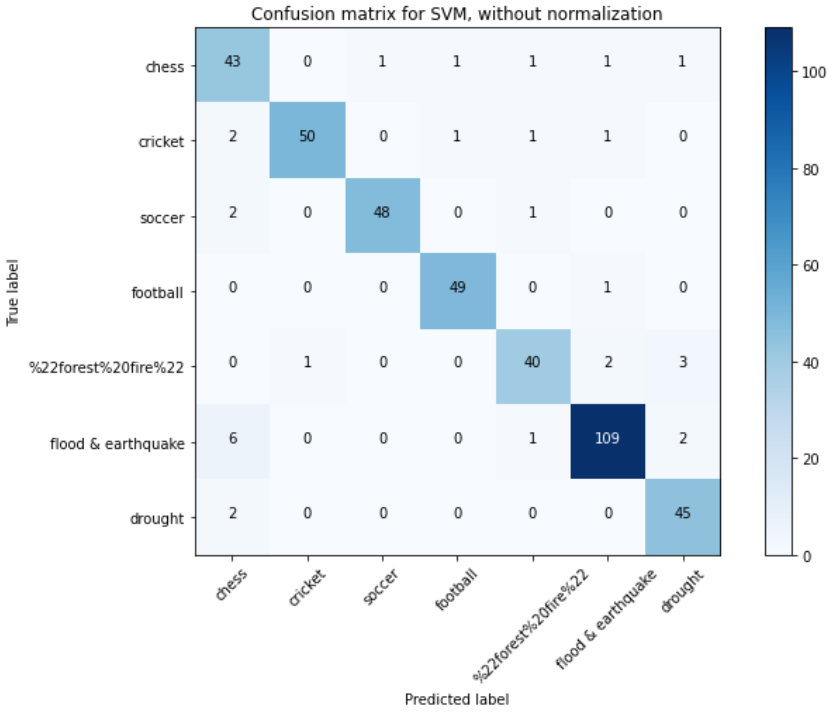


- After Merging

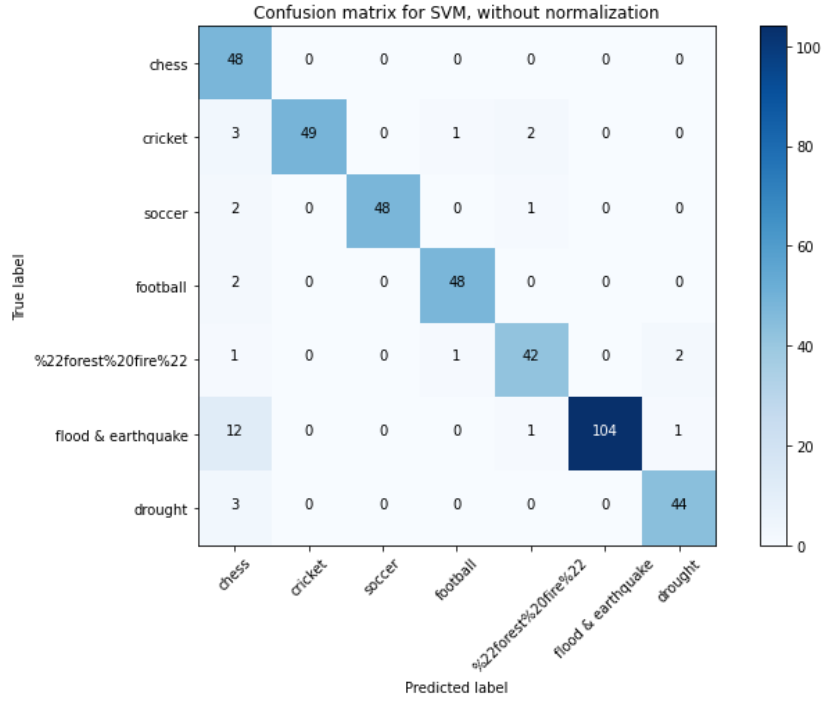
- Based on the results above, “soccer” and “football” can be merged into one class, and “flood” and “earthquake” can be merged into one class.
- One VS One
    - Accuracy: 0.9204819277108434
    - Precision: 0.9204819277108434
    - Recall: 0.9204819277108434
    - F-1 Score: 0.9204819277108434



- One VS Rest
    - Accuracy: 0.9253012048192771
    - Precision: 0.9253012048192771
    - Recall: 0.9253012048192771
    - F-1 Score: 0.9253012048192772

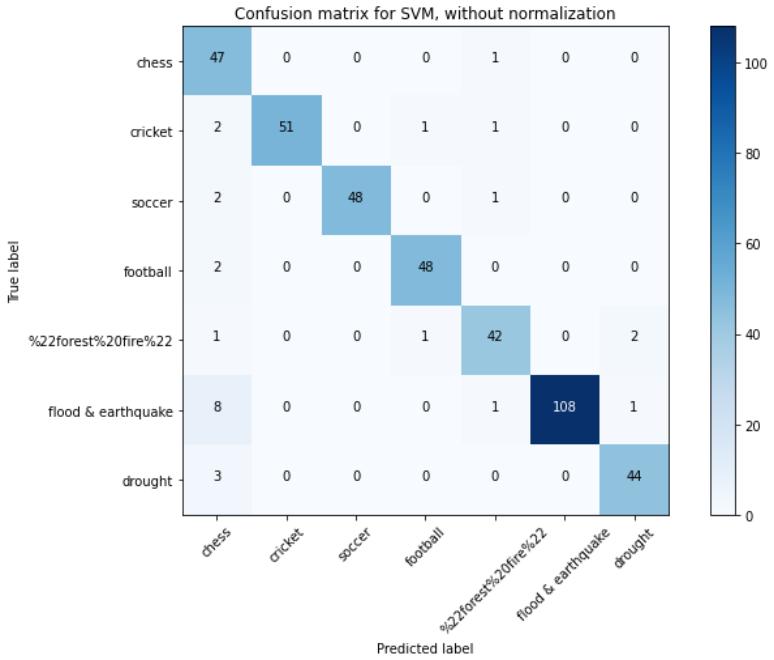


- One VS One (Balanced)
  - Accuracy: 0.9228915662650602
  - Precision: 0.9228915662650602
  - Recall: 0.9228915662650602
  - F-1 Score: 0.9228915662650602



- One VS Rest (Balanced)
  - Accuracy: 0.9349397590361446
  - Precision: 0.9349397590361446

- Recall: 0.9349397590361446
- F-1 Score: 0.9349397590361446



- After merging, the accuracy of One VS One, and One VS Rest models improved by approximately 1%.

Models	Before Merging		After Merging			
	Balanced		Imbalance		Balanced	
	OvO	OvR	OvO	OvR	OvO	OvR
Accuracy	0.9084	0.9133	0.9205	0.9253	0.9229	0.9349
Precision	0.9084	0.9133	0.9205	0.9253	0.9229	0.9349
Recall	0.9084	0.9133	0.9205	0.9253	0.9229	0.9349
F-1 Score	0.9084	0.9133	0.9205	0.9253	0.9229	0.9349

- Based on the data we got from after merging, we can see that the imbalance issue does have a slight impact on the accuracy. By setting the class_weight attribute to “balanced”, the accuracy improved a little bit.

Question 10

1. Why are GLoVE embeddings trained on the ratio of co-occurrence probabilities rather than the probabilities themselves?

- With co-occurrence information, global statistics take into effect for the GLoVE model and allows GLoVE to capture both local and global features. By applying ratio, relevant

words can be better distinguished by the ratio, while irrelevant words can be also better discriminated between two relevant words.

2. 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 vector GLoVE returns relies on the Co-ocurrence Matrix (Equation 7), and it changes since the context of running in the two sentences is different.

3. The expected value.
- The expected value of the first sentence should be close to 0, since the relationship is very similar between (queen, king) and (wife, husband). The expected value of the second sentence and the third sentence should be similar for the same reason.

4. Given a word, would you rather stem or lemmatize the word before mapping it to its GLoVE embedding?
- We would rather lemmatize the word. This is because stemming may create invalid words that don’t exist in GLoVE pretrained embeddings, and all words not found in the embedding will be all-zeros. This may cause the model to lose context information. Lemmatization, on the other hand, outputs valid words that wouldn't affect the embedding matrix.

Question 11

- In this process, we choose [“keywords”] column as the training set, and [“root_label”] column as the testing set.
    - Instead of using the TF-IDF matrix, we create an embedding matrix for the words we have in the dataset. The dimension of each datapoint after preprocessed by GLoVE embedding is (25,300), with 300 dimensions for each word in the “keywords”' list of maximum length 25.

- We selected the LSTM model to train the binary classification problem using GLoVE pre-trained word embeddings with dimension of 300.
    - Training Accuracy : 0.9988
    - Testing Accuracy : 0.9589

Question 12

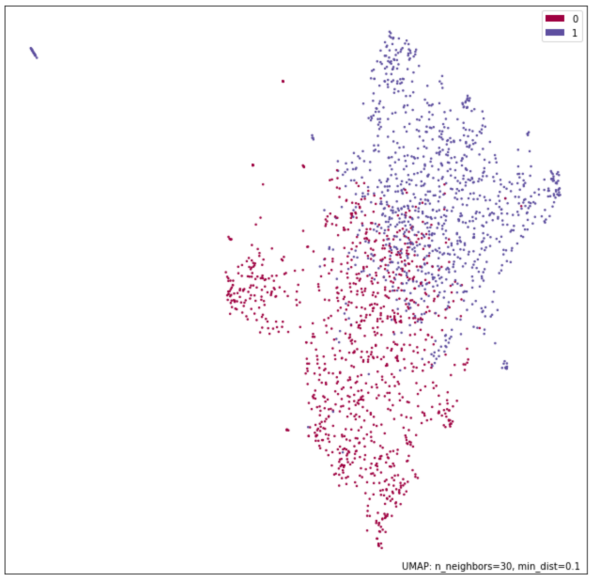
- Comparison of the dimension of the pre-trained GLoVE embedding and the resulting accuracy of the model in the classification task.

-

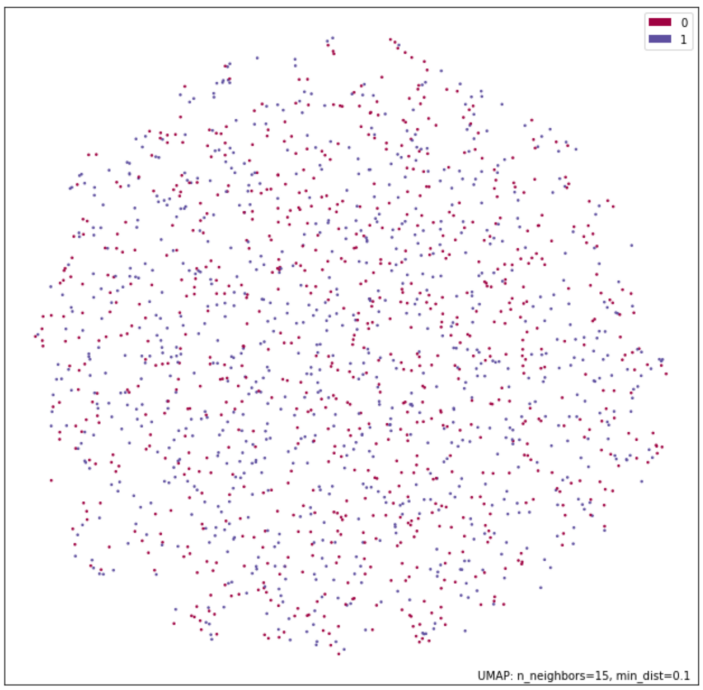
- We use GLoVe pre-trained word embeddings with dimensions = 50, 100, 200, 300 respectively to train the model.
- Generally in this classification problem, the validation accuracy with less dimension (i.e. dim=50, 100) is lower than the validation accuracy with higher dimension. This may because the model is underfitted.
- Yet it doesn't mean the higher the dimension the better the model will be. In our result, GLoVe with 200 dimensions performs slightly better than GLoVe with 300 dimensions. This may be caused by overfitting with 300 dimensions.

Question 13

- We visualize the GLoVe embedding using UMAP such that:



- And the random variable cluster using UMAP:



- We can observe the plot for GLoVe embedding is more concentrated and clearly formed two clusters for two labels. It indicates that GLoVe embedding has significant advantages for processing the data and projecting the data into higher dimensions to explore the feature. The plot for random variables is sparse and just randomly fills in the entire space.