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

Yi Xiao, Zeyu Tan, Binrui Shan

January 2023

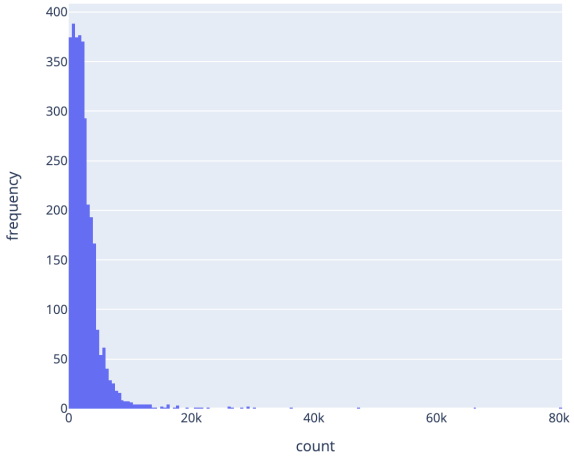
Question 1:

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

The number of samples in the dataset is **3150**
The number of features in the dataset is **8**

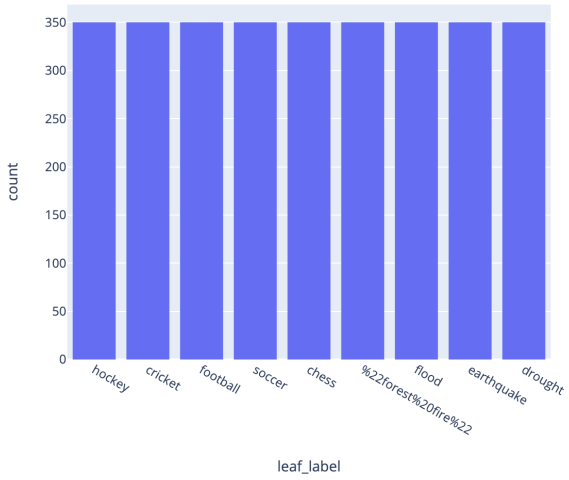
Histograms: Plot 3 histograms on (a) The total number of alpha-numeric characters per data point (row) in the feature full text: i.e count on the x-axis and frequency on the y-axis; (b) The column leaf label – class on the x-axis; (c) The column root label – class on the x-axis.

Figure 1: (a).



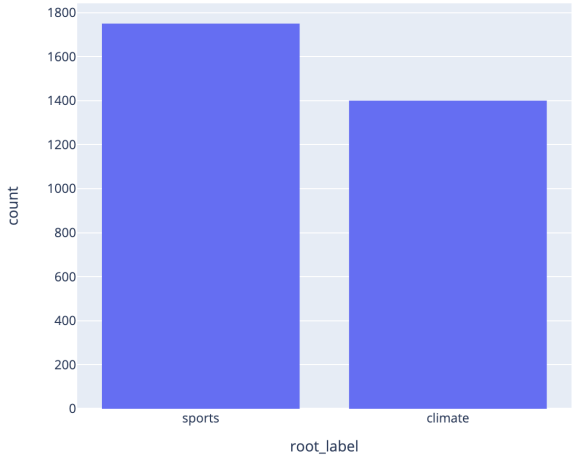
(a). From this plot, we can find that the character counts of most texts (data points) are less than 10000 alpha-numeric characters.

Figure 2: (b).



(b). All leaf classes have the same number of samples (the data points are evenly distributed in leaf classes).

Figure 3: (c).



(c). The number of texts labeled as sports is more than the number of texts labeled as climate (more data points belong to the sports class than the climate class).

Question 2:

Report the number of training and testing samples.

The training set has **2520** data samples.
The testing set has **630** data samples.

Question 3:

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

Stemming will remove the last few characters (e.g. adjectives/nouns/verbs ending) of the word, but does not care about the meaning of the words (which means that there will be no further adjustments after the removal). For example, stemming will change historical to histori. However, histori is meaningless in English. While Lemmatization can fix this problem, it does not only remove the last few characters but also trace the supposed word it should be (to make the words meaningful).
The pro of lemmatization is that it finds meaningful words or representations (higher accuracy), while the con of it is that it takes more time.
The pro of stemming is that it just needs to get a base word and takes less time, while the con of it is that it may cause meaningless words.
If we do not use these two methods for texts, the dictionary of terms might be large, and e.g. eating, eat, ate, through the meaning of the three words are the same (the tenses are not the same), but they are considered as different terms and being put to the dictionary. If we use the lemmatization, all three words will be converted to the same term (eat), so only 'eat' would be put into the dictionary. If we use stemming, the three words will be converted to eat, eat, ate, so we can use two terms to summarize the three words. Overall, lemmatization could decrease the dictionary size more than stemming.

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

Increasing the min_df will cause a reduction of TF-IDF matrix column number. Because higher minimum document frequency means more words will be affected and removed, fewer words will appear in the TF-IDF matrix(fewer columns). The row number will remain the same since no document will be removed.

Should I remove stopwords before or after lemmatizing? Should I remove punctuation 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.

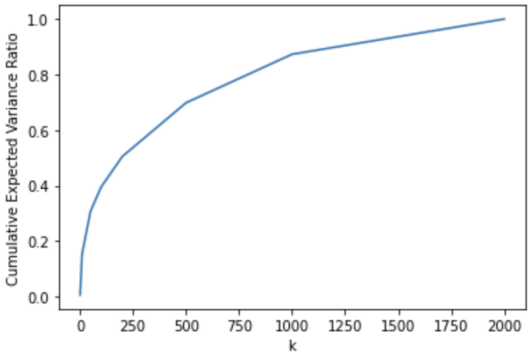
We think that we should remove stopwords before lemmatization because some words are meaningless for our model and we do not need to lemmatize them. e.g. 'is', 'are', 'was', and 'were', the four words all belong to stopwords and all of them can be converted to 'be' after lemmatization. While lemmatization is more complex than removing a word, which means if we remove stopwords before lemmatization, we can reduce the lemmatization time.
We should remove punctuation before lemmatization because the punctuation may affect the lemmatization and cause some words to miss the lemmatization process, e.g. "dimensions." would not be converted to "dimension" during the lemmatization process, but "dimensions" (does not have punctuation follows) should be converted to "dimension" during this process.
We should remove numbers before lemmatization because numbers do not change during the lemmatization process.

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

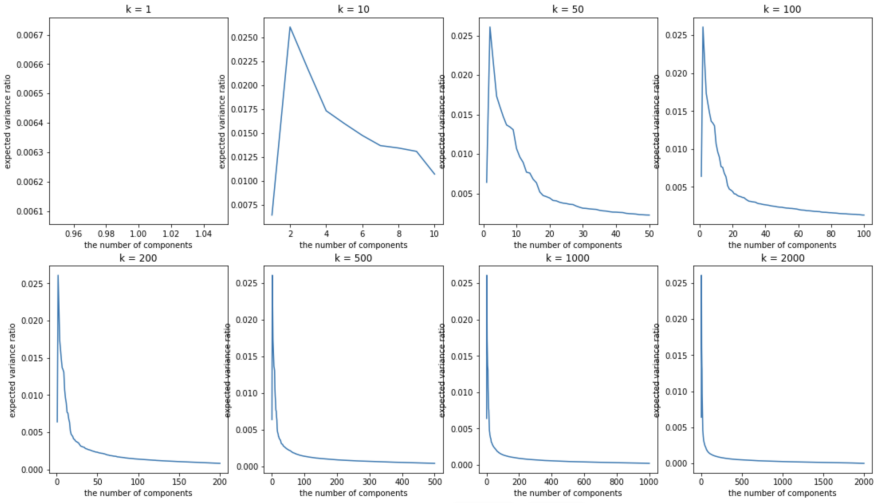
The shape of the TF-IDF processed train matrix is (2520, 14414).
The shape of the TF-IDF processed test matrix is (630, 14414).

Question 4.

The cumulative expected variance ratio across multiple different k for LSI is plotted as follows:



And the expected variance ratio plots for different k(s):



What does the explained variance ratio plot look like?
The expected variance ratio first increases and then decreases as the number of components increases.
What does the plot's concavity suggest?
The plot's concavity suggests that we do not need to retain all features (dimensions), we can reduce the dimensionality of the data without affecting too much the percentage variance that is explained by the data. **With k = 50 found in the previous sections, calculate the reconstruction residual MSE error when using LSI and NMF – they both should use the same k = 50. Which one is larger and why?**
We have the LSI error is ≈ 1680, and the NMF error is ≈ 1706.
The NMF error is larger. And this is because the NMF optimization objectives have more constraints than LSI. As stated in the declaration, we can find that the matrices W and H are non-negative, this prerequisite reduces the degrees of freedom. While LSI has a larger search space and a larger degree of freedom to find the best vectors that minimize the errors. (Since the negative elements might be in the best vectors).

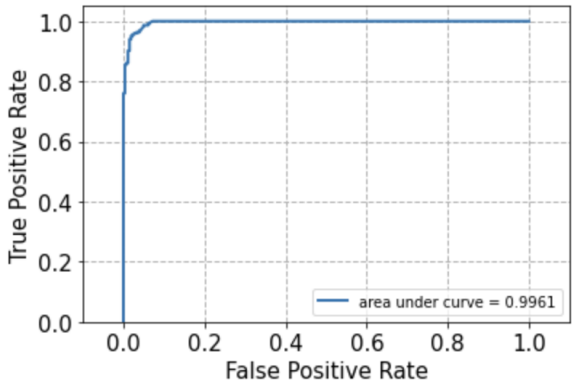
Question 5

The SVM with $\gamma = 1000$ (hard margin)
(pos_lable is "sports") The confusion matrix:

$$\begin{bmatrix} 257(true\_positive) & 6(false\_negative) \\ 18(false\_positive) & 349(true\_negative) \end{bmatrix}$$

The accuracy score: 0.9619047619047619

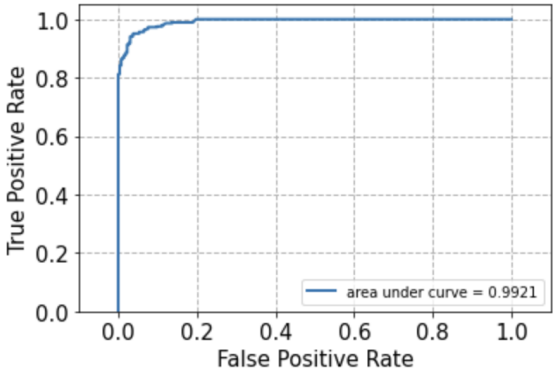
The recall score: 0.9619047619047619
The precision score: 0.962829583561978
The F1 score: 0.9620130349352152
And the ROC plot of it is:



The SVM with $\gamma = 0.0001$ (soft margin)
The confusion matrix:

$$\begin{bmatrix} 0 & 263 \\ 2 & 367 \end{bmatrix}$$

The accuracy score: 0.5825396825396826
The recall score: 0.5825396825396826
The precision score: 0.33935248173343413
The F1 score: 0.4288707392017322
And the ROC plot of it is:

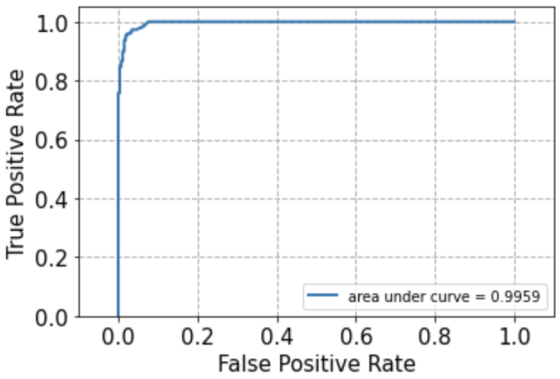


From the calculated scores above, we can infer that the hard margin one performs better.
The SVM with $\gamma = 100000$

The confusion matrix:

$$\begin{bmatrix} 254 & 9 \\ 13 & 354 \end{bmatrix}$$

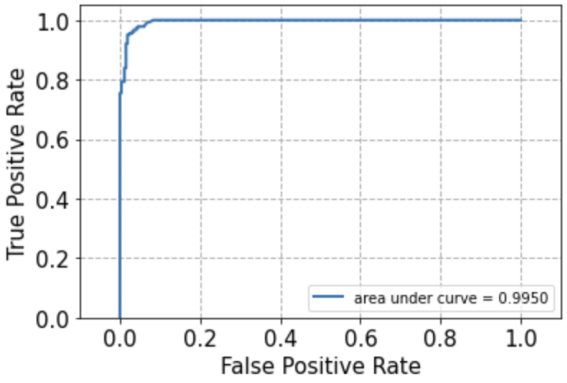
The accuracy score: 0.9650793650793651
The recall score: 0.9650793650793651
The precision score: 0.9652310841279175
The F1 score: 0.9651154680878123
And the ROC plot of it is:



What happens for the soft margin SVM? Why is the case? Analyze in terms of the confusion matrix.
The true positive value and false positive value are very low, while the true negative value and false negative value are very high. From the given info, we know that the "Soft Margin SVM" is very lenient towards the misclassification of a few individual points as long as most data points are well separated. This causes most "sports" data is also being classified as "climate."
Does the ROC curve reflect the performance of the soft-margin SVM? Why?
The ROC curve looks good and does not reflect the performance of the soft-margin SVM. This might be because the soft-margin SVM has found a separating hyperplane whose intercept term b is not appropriate, with only a few data points being classified as "sports".
Use cross-validation to choose γ (use average validation 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 F-1 score of this best SVM.
We found that when $\gamma = 100$, the SVM model has the highest mean test score.
The confusion matrix:

$$\begin{bmatrix} 257 & 6 \\ 17 & 350 \end{bmatrix}$$

The accuracy score: 0.9634920634920635
The recall score: 0.9634920634920635
The precision score: 0.9642810928466631
The F1 score: 0.9635882580132562
And the ROC plot of it is:

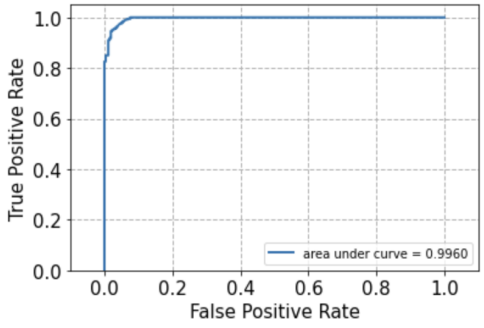


Question 6

The logistic classifier without regularization (we set $C = 10^{10}$)
The confusion matrix:

$$\begin{bmatrix} 257 & 6 \\ 18 & 349 \end{bmatrix}$$

The accuracy score: 0.9619047619047619
The recall score: 0.9619047619047619
The precision score: 0.962829583561978
The F1 score: 0.9620130349352152
And the ROC plot of it is:

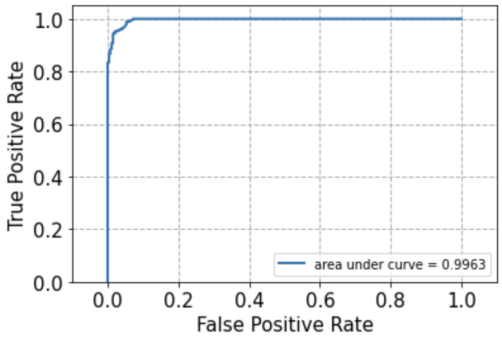


Find the optimal regularization coefficient

For logistic regression with L1 regularization, we have the optimal C = 10 (k = 1)
The confusion matrix:

$$\begin{bmatrix} 257 & 6 \\ 17 & 350 \end{bmatrix}$$

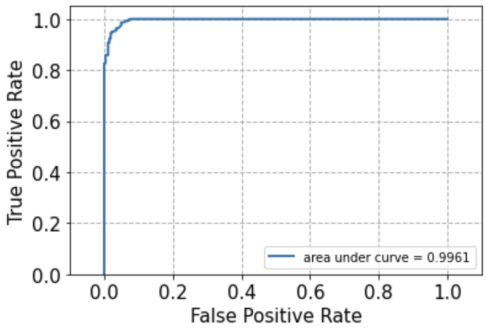
The accuracy score: 0.9634920634920635
The recall score: 0.9634920634920635
The precision score: 0.9642810928466631
The F1 score: 0.9635882580132562
And the ROC plot of it is:



For logistic regression with L2 regularization, we have the optimal C = 1000 (k = 3)
The confusion matrix:

$$\begin{bmatrix} 236 & 27 \\ 2 & 365 \end{bmatrix}$$

The accuracy score: 0.9619047619047619
The recall score: 0.9635312522663462
The precision score: 0.9490804535588518
The F1 score: 0.9447532639022
And the ROC plot of it is:



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.

Comparing the three logistic classifiers, we have the logistic regression with L1 regularization performs the best because it has the highest accuracy, precision score and F1-score, and the logistic regression with L2 regularization has the highest recall score. The logistic regression without any regularization performs the worst because all scores of it are the lowest.

The regularization parameter (both the L1 and L2 regularization) decreases the test error. For the model without regularization: its learned coefficients tend to be very large (with the order of magnitude of 10^1), which makes the model over-fitting during the training process. If someone wants to train a complex model that could deal with high-dimension data while there are no signs of over-fitting, he or she might be interested in building a model without regularization.

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

For the model with L1 regularization: some of the learned coefficients are 0 and most of the learned coefficients are with a magnitude of 10^0 (which is much smaller than the learned coefficients of the logistic regression model without regularization). Someone might be interested in this kind of regularization if he or she wants to build a sparse model by removing features (while the most significant features are kept in the model).

For the model with L2 regularization: most of the learned coefficients are with a magnitude of 10^0 (which also has smaller learned coefficients than the one without regularization). Someone might be interested in this type of regularization if the model is not heavily dependent on a few features (there are a few outliers in the dataset).

(L2 regularization takes the square of the weights, so the cost of outliers present in the data increases exponentially. L1 regularization takes the absolute values of the weights, so the cost only increases linearly).

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

Logistic regression is a probabilistic classifier, while the SVM is a deterministic classifier. The differences are 1. the logistic regression classifier uses all the data points to find the decision boundary, but the SVM classifier only uses a small subset of the data points (support vectors) to find the decision boundary. 2. The SVM tries to maximize the margin between the closest support vectors whereas logistic regression maximizes the posterior class probability.

For complex data, SVMs usually perform better in generating complex decision boundaries. (But SVMs are not suitable for large dataset).

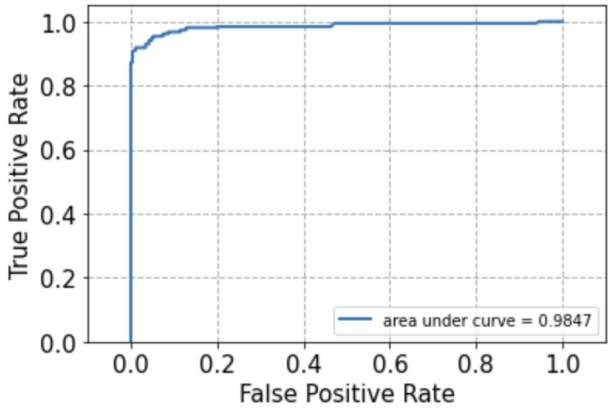
Question 7

(a).

The confusion matrix:

$$\begin{bmatrix} 260 & 3 \\ 32 & 335 \end{bmatrix}$$

The accuracy score: 0.9444444444444444
The recall score: 0.9444444444444444
The precision score: 0.962829583561978
The F1 score: 0.9620130349352152
And the ROC plot of it is:



Question 8

The best 5 combinations are as follows:

```
[ ] total_results.head()
```

	mean_test_score	param_clf	param_reduce_dim	param_vect__min_df	lemmatized	Stemmed
0	0.927381	LogisticRegression(C=1000, random_state=42, so...	TruncatedSVD(n_components=80, random_state=42)	5	True	False
1	0.926984	LogisticRegression(C=10, penalty='l1', random_...	TruncatedSVD(n_components=80, random_state=42)	5	True	False
2	0.924206	LogisticRegression(C=10, penalty='l1', random_...	TruncatedSVD(n_components=80, random_state=42)	5	False	True
3	0.924206	LogisticRegression(C=1000, random_state=42, so...	TruncatedSVD(n_components=80, random_state=42)	5	False	True
4	0.923413	LogisticRegression(C=10, penalty='l1', random_...	NMF(n_components=80, random_state=42)	5	True	False

(The performances of each model on the testing set are listed in the table above).

Question 9

Do you observe any structure in the confusion matrix? Are there distinct visible blocks on the major diagonal? What does this mean?
Yes, most samples are classified correctly, the numbers on the diagonal are obviously larger than the rest.
There is a distinct visible block on the major diagonal on the soccer and football columns and rows.
This means about one half of the data points that labelled as 'soccer' being incorrectly classified as 'football', and for one half of the data points that labelled as 'football' being incorrectly classified as 'soccer'. Our model could not effectively distinguish between data points that are labelled as 'football' or 'soccer'.

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

```
One VS One for multi stats:
confusion matrix
 [[61  2  1  0  0  5  0  2  0]
 [ 0 75  0  0  2  3  0  1  0]
 [ 1  0 78  2  0  5  0  0  0]
 [ 0  0  0 30 36  2  0  0  0]
 [ 0  0  0 37 23  1  0  0  0]
 [ 0  0  0  0  0 58  1  0  3]
 [ 1  0  0  1  0  7 63  0  2]
 [ 0  0  0  0  0  1  0 56  2]
 [ 0  0  1  0  0 11  1  0 55]]
accuracy score:  0.792063492063492
recall score:  0.792063492063492
precision score:  0.8100751063807898
f1 score:  0.79673431420768
----------------------------------------
```

```
One VS One for multi stats:
confusion matrix
 [[ 59   2   1   0   8   0   1   0]
 [  0  74   0   2   3   0   1   1]
 [  1   0  79   1   5   0   0   0]
 [  0   0   0 126   3   0   0   0]
 [  0   0   0   0  56   1   1   4]
 [  0   0   0   1   7  64   0   2]
 [  0   0   0   0   1   0  56   2]
 [  0   0   0   0  12   1   0  55]]
accuracy score:  0.9031746031746032
recall score:  0.9031746031746032
precision score:  0.9228306367122157
f1 score:  0.9083062388230142
----------------------------------------
```

Since we observed distinct visible blocks at "soccer" and "football" columns,

we merged the 'football' and 'soccer' data points into a new larger label. The accuracy improved from 0.79 to 0.90 in one VS one and from 0.83 to 0.91 in one VS the rest.

```
One VS the rest for multi stats:
confusion matrix
 [[64  2  2  0  0  3  0  0  0]
 [ 1 77  0  0  1  1  0  1  0]
 [ 1  0 80  1  0  4  0  0  0]
 [ 2  0  0 30 34  2  0  0  0]
 [ 2  0  1 26 30  1  0  0  1]
 [ 0  0  0  0  0 57  1  0  4]
 [ 0  0  0  1  0  6 64  1  2]
 [ 0  0  0  0  0  1  0 56  2]
 [ 0  0  1  0  0 12  1  0 54]]
accuracy score:  0.8126984126984127
recall score:  0.8126984126984127
precision score:  0.8201970977689629
f1 score:  0.8133638639002908
---------------------------------------
```

```
One VS the rest for multi stats:
confusion matrix
 [[ 65   2   1   0   3   0   0   0]
 [  1  75   0   3   1   0   1   0]
 [  0   0  80   2   4   0   0   0]
 [  0   0   0 127   2   0   0   0]
 [  2   0   0   0  55   0   0   5]
 [  0   0   0   1   5  65   1   2]
 [  0   0   0   1   1   0  55   2]
 [  0   0   1   0  12   1   0  54]]
accuracy score:  0.9142857142857143
recall score:  0.9142857142857143
precision score:  0.923977854102336
f1 score:  0.9167460922982267
---------------------------------------
```

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

Once we merged 'soccer' and 'football' classes, we can see a drastic performance improvement based on the metrics, having f1 score increase about 0.2. This is due to the better prediction in the merged class 'football'. However, there is still minor issue in the 'football' class as this class also drag in more inaccurate classes than the previous 'soccer' and 'football' combining together (7 Vs 3). Our resolution for imbalance class is to double the size of classes other than the merged class, making then relatively balanced in size. We later reached a decent metric score by improve them by at least 1 percent.

```
One VS the rest for multi stats:
confusion matrix
 [[128   2   2   0   6   0   2   2]
 [  0 156   0   2   2   0   0   2]
 [  0   2 160   2   8   0   0   0]
 [  0   0   0 127   1   0   0   1]
 [  4   0   2   0 112   0   0   6]
 [  0   0   2   2  10 130   0   4]
 [  0   0   0   0   0   0 114   4]
 [  0   0   0   0  18   2   0 116]]
accuracy score:  0.9221927497789567
recall score:  0.9221927497789567
precision score:  0.9298421789029521
f1 score:  0.9241553776367073
---------------------------------------
```

Question 10

(a).

Because co-occurrence probability can reflect the relationship between two words. For example, gas is related to steam, so its ratio is higher. gas is oppositely related to water, so its ratio is lower. This ratio can help distinguish relevant words from irrelevant words. It also helps distinguish between two relevant words.

(b).

GLoVE embeddings will not return the same vector for the word running because the context of the two words is different. If running is close to 'in the park', then the model will consider that run as a sport, because the co-occurrence probability will show that that run is relative to a place, then run means a kind of sport. If the run is close to a position("president"), then the co-occurrence probability will inform the model that "run" means differently. Since two words have different meanings, they will have a different vectors.

(c).

$$||GLoVE[\text{“}queen\text{”}]-GLoVE[\text{“}king\text{”}]||_2-||GLoVE[\text{“}wife\text{”}]+GLoVE[\text{“}husband\text{”}]||_2 = 0 \tag{1}$$

$$||GLoVE[\text{“}queen\text{”}]-GLoVE[\text{“}king\text{”}]||_2 = ||GLoVE[\text{“}wife\text{”}]-GLoVE[\text{“}husband\text{”}]||_2 \tag{2}$$

The analogy "king is to queen as man is to woman" will be encoded in the vector space by the vector equation king - queen = man - woman.

(d).

I would rather lemmatize the word before mapping it to its GLoVE because lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma. So that two words will have similar vectors and that will increase the overall accuracy. Stemming is a process that stems or removes the last few characters from a word, often leading to incorrect meanings and spelling. For example, stemming the word 'Caring' would return 'Car'. So that two words will have similar vectors since the spelling or meaning of the word is changed and overall accuracy will be lower.

Question 11

(a).

From glove300d txt, we can get a word embedding dictionary, which is a dictionary that has vocabularies as keys and a 300 dimension vector as the corre-

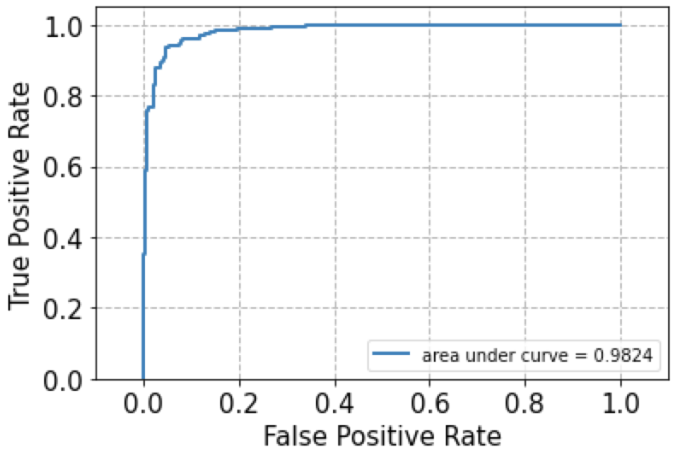
sponding value. We found that each data point/document has a list of keywords that has a similar length(so there won't be data imbalance problems), and by representing those keywords with word embedding we can obtain a matrix that represents the data set's features.
The TF-IDF method will generate a D x N matrix, N is the number of vocabulary words we pick and D is the number of data points. By using glove pre-trained word embedding, we can obtain a matrix of size D x 300. D is the number of data points and 300 is the number of dimensions. Each row is the sum of word vectors of all keywords of that data point. For instance, if a data point has 5 different keywords, the feature vector = v1 + v2 + ... + v5. This D x 300 matrix is the result of our feature engineering process that uses GLove word embeddings to represent each document.

(b).

We decided to use the SVM classifier and the result we get is:
And the results for 300 dimensions are as follows
The confusion matrix:

$$\begin{bmatrix} 240 & 23 \\ 15 & 352 \end{bmatrix}$$

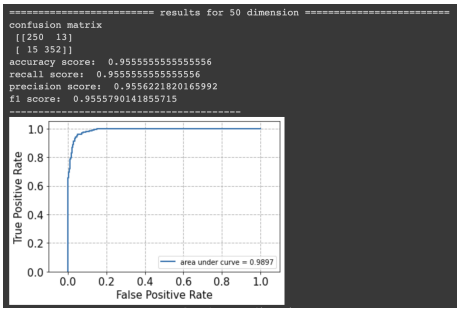
The accuracy score: 0.9396825396825397
The recall score: 0.9396825396825397
The precision score: 0.9397144102085279
The F1 score: 0.9395419292993417
And the ROC plot of it is:

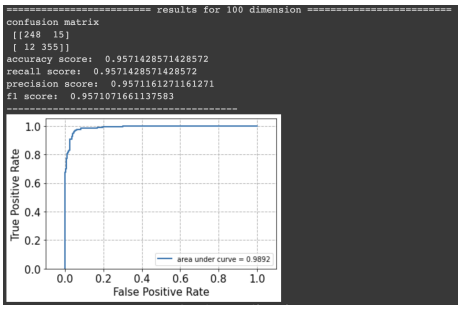


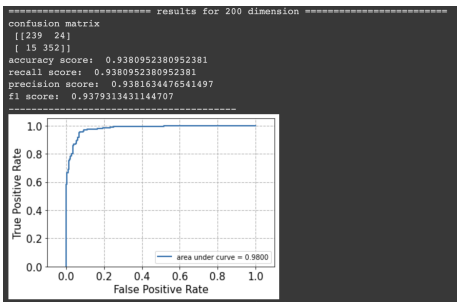
Question 12

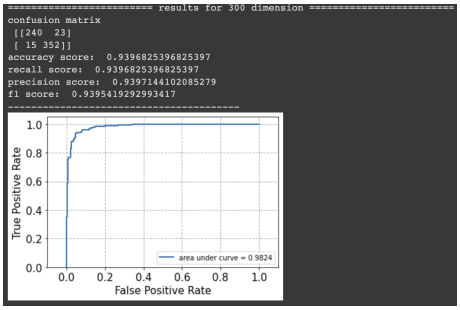
We used all 4 different word embedding dictionaries and they have 50, 100, 200, and 300 dimensions respectively. The results of the accuracy are plotted below.

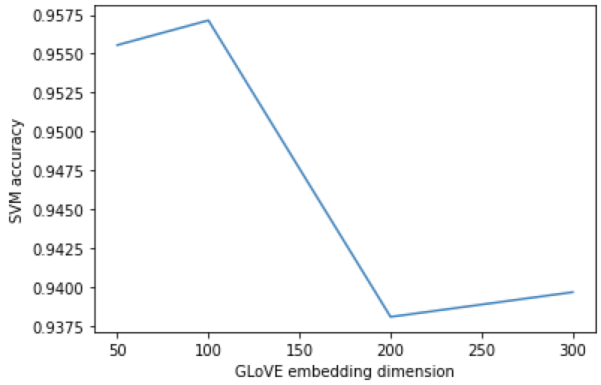
From the results, we can see that the accuracy of the model that evaluates









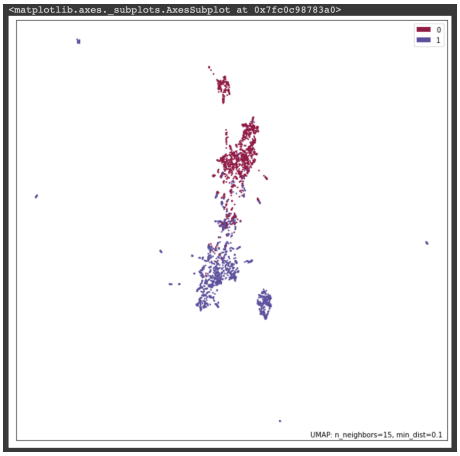


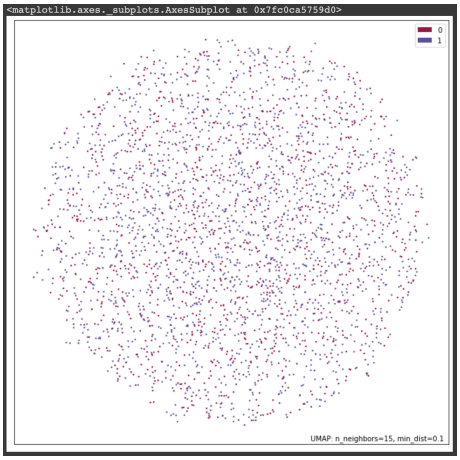
GLoVE increases at first and then decreases. 100 dimension has the best results among all of the 4 results.

This trend is expected because we believe a 50-dimension feature matrix will lead to some extent of under-fitting, while a high-dimension feature matrix will

lead to over-fitting, which will also cause a lower accuracy. Therefore, compared to the 50-dimension and 200/300-dimension feature matrices, the 100-dimension feature matrix has the best performance.

Question 13





The first plot visualizes the set of normalized GLoVE-based embeddings of the documents with their binary labels in a 2D plane using the UMAP library. Similarly, the second plot generates a set of normalized random vectors of the same dimension as GLoVE and visualize these in a 2D plane with UMAP.
We can clearly see clustering in the first plot and there are two major clusters, and we cannot see clear sign of clustering in the second plot.