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

ECE 219 - Large Scale Data Mining
University of California, Los Angeles

Member 1:	Shruti Mohanty (705494615)
Member 2:	Kimaya Kulkarni (805528337)
Member 3:	Mohan S Acharya (905627884)

1 Question 1

1.1 Number of samples

The news dataset has 2072 samples, and 8 feature columns corresponding to - full_text, summary, keywords, publish_date, authors, url, leaf_label and root_label respectively.

1.2 Histogram Plots

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 -

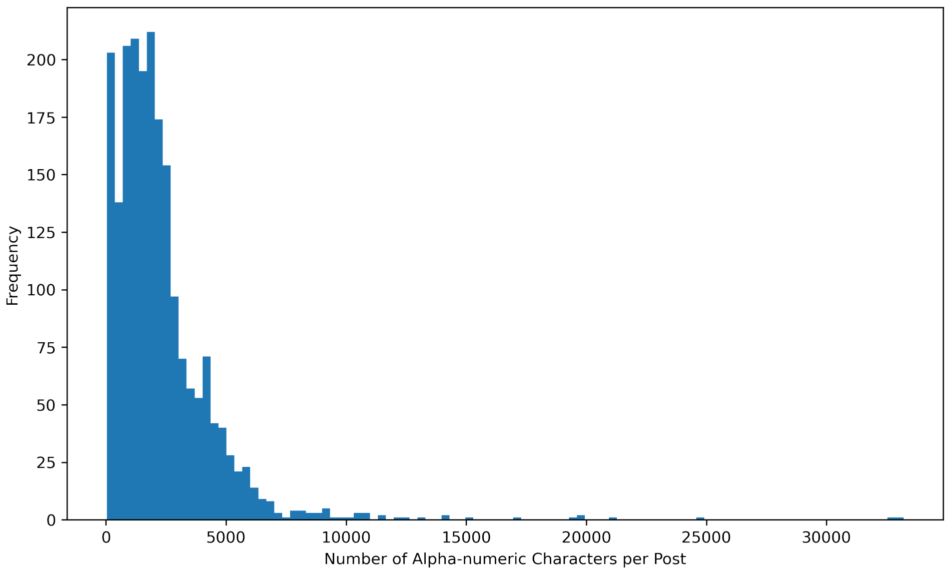
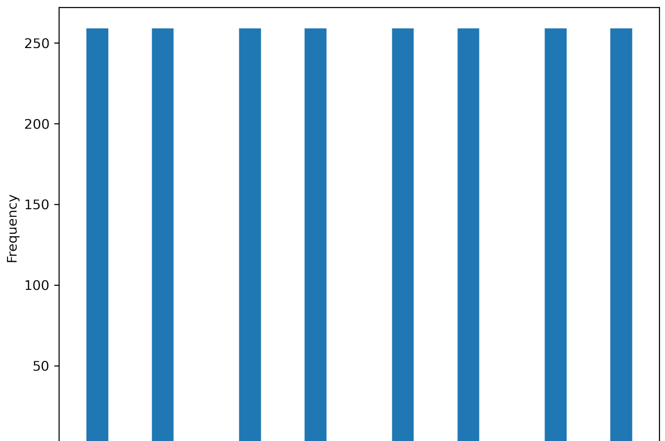


Figure 1:
Histogram plot of number of alpha-numeric characters per post

The column leaf_label – class on the x-axis-



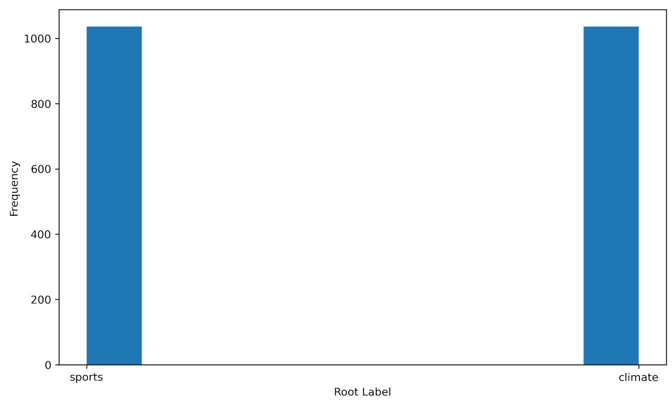


Figure 3:
Histogram plot of number of posts with Root labels

1.3 Qualitative Analysis

From Figure 1, we deduce that there are higher number of posts in the dataset with number of alpha numeric characters in the range till 5000. After that, there are very few posts with alpha-numeric characters beyond 10000.

From Figure 2, we deduce that the dataset is balanced as all the eight classes have equal number of posts - close to 250. This will help in the multi class classification task, and will prevent the classifier from over-fitting. If the dataset was unbalanced, the classifier might perform well on certain classes because of higher samples in those classes. However, due to the even distribution of our data we don't have to mitigate any imbalance using data augmentation or sampling procedures.

From Figure 3, we can similarly deduce that the dataset is perfectly balanced for the task of binary classification. Both the labels sports and climate have close to 1000 data points each.

2 Question 2

Report the number of training and testing samples -
We have 1657 posts for training, and 415 posts for testing, using the train_test_split function in python with test_size of 0.2 on the dataset.

3 Question 3

Feature Extraction Performed - lemmatization, remove punctuation, html artefacts, and numerical characters. The ”english” stopwords of CountVectorizer have been used, with min_df parameter set to 3, along with a TFidfTransformer. A fit_transform is performed on the training set, and transform on the testing set.

3.1

What are the pros and cons of lemmatization versus stemming? How do these processes affect the dictionary size?
Both lemmatization, and stemming are word normalisation techniques used in NLP.
Stemming - Will convert a word into its stem(root form). The PorterStemmer algorithm will strip the suffix from the word. This algorithm doesn't follow linguistic, but it follows a set of rules for stripping the suffix. PorterStemmer is famously used for its simplicity and speed. The stemmed word may not be from the same language. Hence, it is often used in applications like data indexing.

Lemmatization-This algorithm takes into consideration the morphological analysis of words. It depends on part-of-speech information, and different rules are applied based on the semantic information of the word - whether it is a noun, verb or adjective. This is used in various NLP applications, where getting valid words as output is important for word distinguishment.

Lemmatization reduces variability of different forms of the same stem word, and hence decreases the size of the feature vector. If speed and simplicity is of focus than stemming can be used, otherwise lemmatization can be employed for better qualitative analysis. These word normalisation processes will reduce the vocab dictionary size that we initially start with, as they will either remove the suffix or convert the words into a simpler semantic word in most cases, i.e they will reduce the inflection forms.

3.2

min_df means minimum document frequency. How does varying min df change the TF-IDF matrix?
The min_df parameter will remove rare words from the vocabulary. It ignores the terms that have a frequency lesser than the given parameter in the document. If we increase min_df, more words will get ignored, so it will reduce the size of the TF-IDF matrix. On the other-hand, reducing min_df will increase the size of TF-IDF matrix. In our case, we have set min_df to 3 initially.

3.3

Should I remove stopwords before or after lemmatizing? Should I remove punctuations before or after lemmatizing? Should I remove numbers before or after lemmatizing?
In our code, we have performed lemmatization first. Punctuations are removed only after lemmatization, as they serve useful context for the words which can help in POS tagging, and hence generate correct lemmas. Numbers can be removed before or after lemmatization as per choice, as they don't have any affect on the algorithm to understand context. We have performed all pre-processing after lemmatization for more clarity. The stopwords called in CountVectorizer should be removed after lemmatization, as they are seemingly not useful words for text processing, but in certain cases they can also help in providing clarity in terms of context. So the stopwords parameters along with min_df is used after lemmatization to reduce the vocabulary size further.

3.4

Report the shape of the TF-IDF-processed train and test matrices-
TF-IDF Processed Train matrix size - (1657, 10440)
TF-IDF Processed Test matrix size - (415, 10440)

4 Question 4

4.1

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?

Explained Variance Ratio Plot for train dataset -

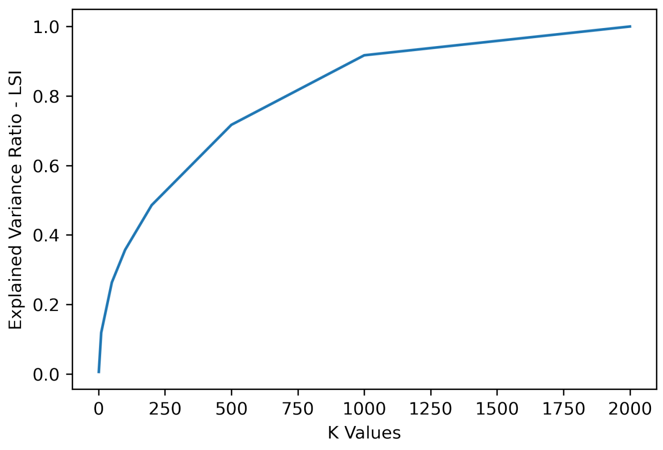


Figure 4:
Explained variance ratio v/s K for train

Explained Variance Ratio Plot for test dataset -

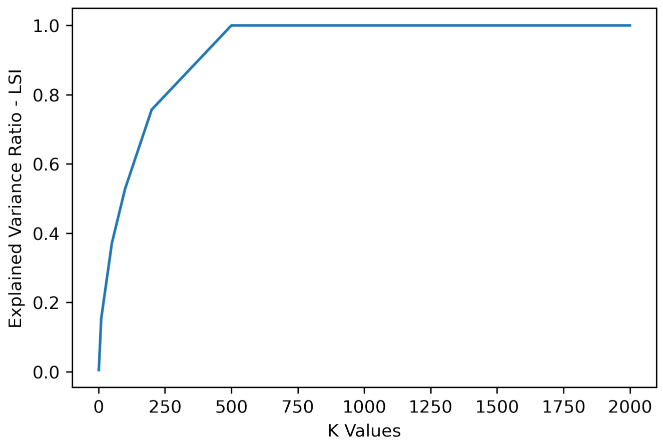


Figure 5:
Explained variance ratio v/s K for test

The plots for train and test dataset have a concave nature (downward concavity). where they are increasing with increase in K value, and saturate at 1. By definition of a concave plot - A function of a single variable is concave if every line segment joining two points on its graph does not lie above the graph at any point. This nature can be seen in the graph till it saturates for the test set. For the train curve it is seen till the end. For the train plot it saturates at 1 close to K=1750, and for the test dataset it saturates at 1 much faster, at k= 500.

The plots concavity suggests that as the number of feature dimensions increases, the explained variance percentage will increase for the data. A higher explained variance indicates more association, which can potentially lead to better predictions. However, this is not the only criteria that can be used for classification tasks as discussed in class with 2 adjoining ellipses. A dimension with more variance might not be a perfect classifier, and hence we employ a grid search to identify the optimal K value.

4.2

Reconstruction Residual Error for k=50

Error values for LSI - 1182.66 (train), 253.60 (test)
Error values for NMF - 1204.24(train), 307.53 (test)

The reconstruction error is greater for NMF in both training and testing. NMF has more constraints than LSI, one of them being that it allows only positive entries in the reduced rank matrix. LSI on the otherhand, doesn't have any such constraints. So, NMF has lesser degrees of freedom, as the search space for optimal parameters is much lesser. LSI can better represent higher dimensional data, by providing a better factorisation with lesser loss when compared to NMF.

5 Question 5

5.1

In this question we implemented two linear SVMs with soft and hard margins. We then compared the two with respect to the following metric:

- *ROC Curve*: It is a graphical plot which describes the diagnostic ability of a binary classifier as its discriminatory threshold is varied. It is a plot between TP rate vs FP rate.
- *Confusion Matrix*: The matrix true positives (TP), true negatives (TN), false positives (FP), false negatives (FN). The true values are represented on y-axis whereas predicted values are represented on x-axis.
- *Accuracy*: It is a measure of the correctness of prediction of both positive and negative values. The equation for the same is as follows:
  $Acc = \frac{tp+tn}{tp+tn+fp+fn}$
- *Recall*: It is a measure of positive class predictions out of all positive examples.
  $Recall = \frac{tp}{tp+fn}$
- *Precision*: Precision measures positive predictions that are actually positive. It is a measure of exactness of classifier.
  $Precision = \frac{tp}{tp+fp}$
- *F1 Score*: Its the harmonic mean between precision and recall. F1 score is a better metric in case the dataset is not balanced.

For a given training set $D = (x_i, y_i)$ where i ranges from 1 to n. x belongs to real numbers set and y is either -1 or +1, the goal is to find a hyperplane decision boundary H which separates all -1s from +1s. SVM tries to classify the dataset in such a way that the euclidean distance between training samples and decision boundary is maximised. The difference between hard and soft margin lies in the separability of data. Hard margin requires all points be separated(classified) correctly. Whereas, soft margin classification allows misclassification due to noisy data.
Figure 6 shows roc curve of hard and soft margin and figure 7 shows confusion matrix for hard margin svm and soft margin svm.
In case of Soft SVM, the idea is to allow wide margin allowing the classifier to misclassify such that the classes are still linearly separable. In case of lesser value of γ misclassfication is penalised less harshly. Hence all class 1 examples will be classified into class 1 but even class -1 may be assigned to class 1 itself. As can be seen by the confusion matrix for soft margin SVM, all 206 climate examples have also been assigned to sports class. Hence the soft margin classifier has very low misclassification rate for one class and very high misclassification rate for the other one. If we observe ROC curve for soft SVM, it conflicts with all other performance metric. From the confusion matrix we know that misclassification rate for one class is very low while very high for the other one. This imples that the model is failing to notice the differences in classes due to wider margin.

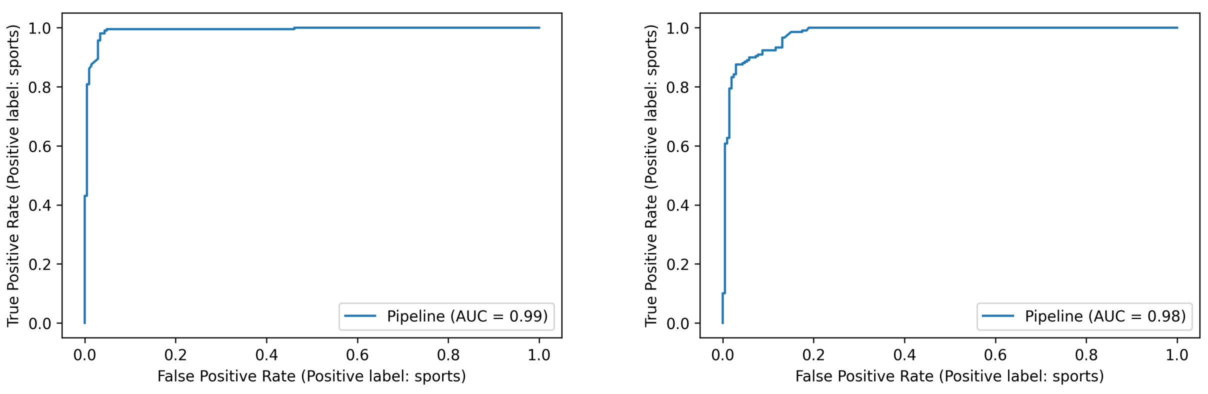


Figure 6: ROC curve for (Left) Hard margin SVM with $\gamma = 1000$ (Right) Soft margin SVM with $\gamma = 0.0001$

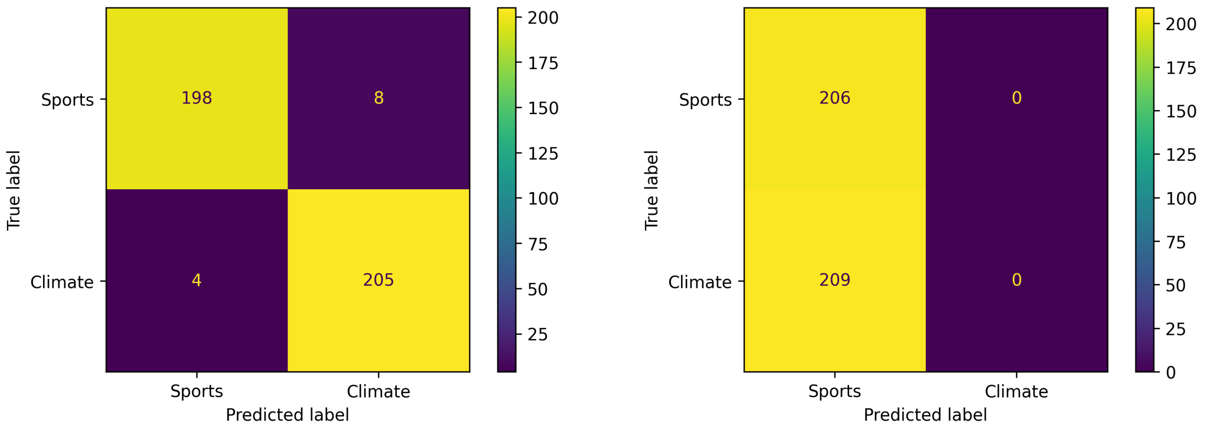


Figure 7: Confusion Matrix for (Left) Hard margin SVM with $\gamma = 1000$ (Right) Soft margin SVM with $\gamma = 0.0001$

From table 1 given below, we can see that hard margin performs better than soft margin.

Table 1: Performance Metric for Hard Margin SVM and Soft Margin SVM

Performance Metric	Hard Margin SVM	Soft Margin SVM
Accuracy	0.9710	0.4963
Recall	0.9710	0.5
Precision	0.9713	0.2481
F1 Score	0.9710	0.3317

From figure 8 we can observe ROC curve and confusion matrix for $\gamma = 100000$.

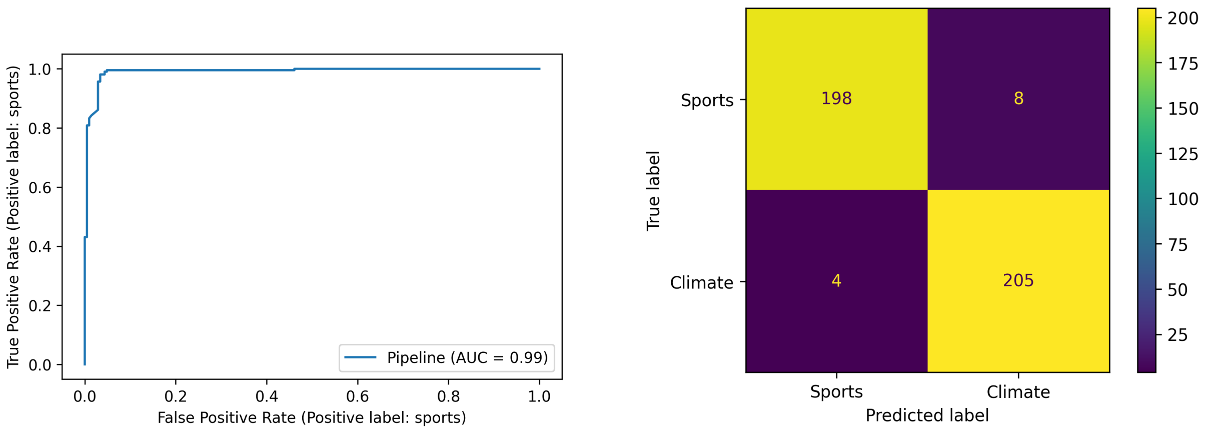


Figure 8: (Left) ROC curve for $\gamma = 100000$ (Right)Confusion Matrix for $\gamma = 100000$

From table 2, we can see the metric values for $\gamma = 100000$. For such a huge value of gamma we still get similar results to that of $\gamma = 1000$.

Table 2: Performance Metric for Hard Margin SVM and Soft Margin SVM

Performance Metric	Hard Margin SVM
Accuracy	0.9710
Recall	0.9710
Precision	0.9713
F1 Score	0.9710

5.2

5-folds cross validation was used to find the best value of hyperparameter(γ), to as a result find the best model. The following procedure was used for the same.

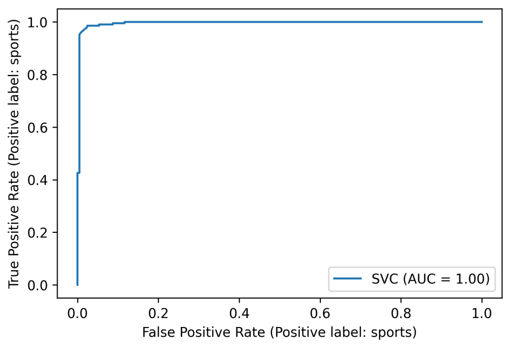
- Split the dataset in 5 folds (equal parts).
- Choose 4 parts as training set, one as testing set.
- Train the model on training set. Validate on test set.
- During each iteration of cross validation we will use different data as training set.
- After all the iterations, we will have results for each train-test combination for that specified number of training and testing samples.
- At the end, we will average the 5 results obtained using all combinations for 5-fold cross validation.

To find an optimal value of γ cross validation with a search space of $\gamma = [0.001, 0.01, 0.1, 1, 10, 100, 300, 350, 400, 450, 500, 700, 1000, 10000, 100000, 500000]$ with linear kernel was used. The average accuracy values for the same can be found in the table.

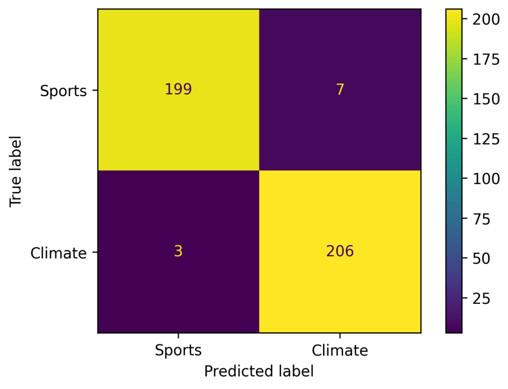
Table 3: Average validation accuracies for different γ Values

γ	Average Validation Accuracy
0.001	0.5009
0.01	0.5009
0.1	0.9426
1	0.9408
10	0.9559
100	0.9577
300	0.9589
350	0.9577
400	0.9577
450	0.9571
500	0.9577
700	0.9553
1000	0.9547
10000	0.9547
100000	0.9547
500000	0.9541

The best value for γ is 300. The ROC curve and confusion matrix for the same is as shown.



(a) label 1



(b) label 2

Figure 9: ROC Curve and Confusion Matrix for $\gamma = 300$

6 Question 6

6.1

Logistic regression models multiple predictor variables to binary output. The logistic function is basically a sigmoid function given by the following equation:
$g(z) = \frac{1}{1+e^{-z}}$ where $z = \theta^T x$
The loss function used is a cross entropy loss. The equation for the same is as follows:
$J(\theta) = \frac{1}{m}(-y^T log(g(z)) - (1 - y^T)log(1 - g(z))$
Fig 10 shows ROC curve and confusion matrix for logistic regression without any regularization. Table 4 shows performance metric for the same.

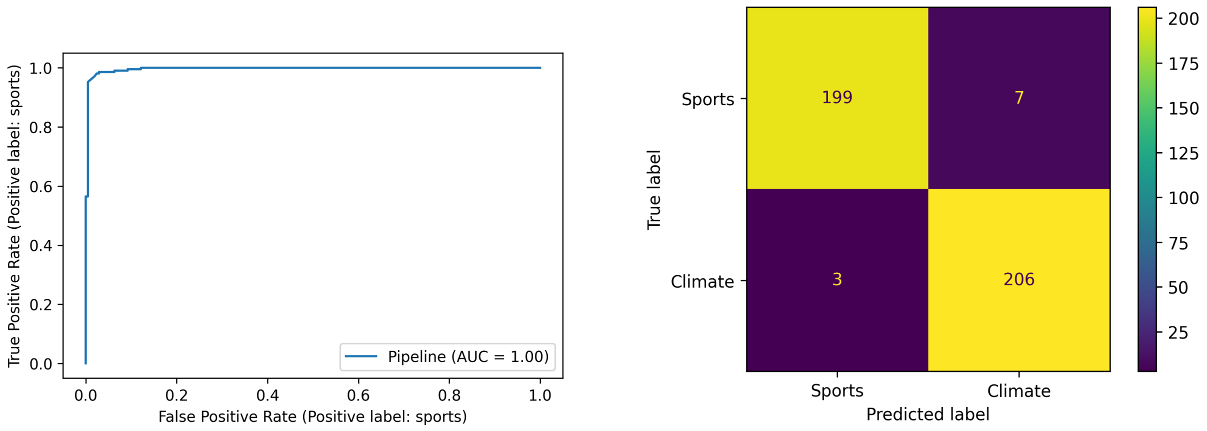


Figure 10: ROC Curve and Confusion Matrix for Logistic Regression Without Regularization

Table 4: Performance Metric for Logistic Regression without Regularization

Performance Metric	Logistic Regression without Regularization
Accuracy	0.9759
Recall	0.9758
Precision	0.9761
F1 Score	0.9758

6.2

Regularization is used to train models such that models generalize better on new data hence avoiding overfitting. The regularization term is added to the loss function as shown.
$J(\theta) = \frac{1}{m}(-y^T log(g(z)) - (1 - y^T)log(1 - g(z)) + \lambda R(f)$
R(f) being the regularizing term.
We tested two regularization techniques namely, L1 regularization and L2 regularization. The definitions of both are as follows:
L1 norm: $R(f) = \sum_{i=1}^{n} |f_i|$
L2 norm: $R(f) = \frac{1}{2}\sum_{i=1}^{n} |f_i|$
We implemented 5-folds cross validation to find best regularization parameters for both methods. Table 5 and 6 give average validation accuracies for L1 and L2 regularization parameters respectively.

Table 5: Average validation accuracies for L1 Regularizarion

L1 Regularization Parameter	Average Validation Accuracy
0.0001	0.5009
0.001	0.5009
0.01	0.5009
0.1	0.9100
1	0.9541
10	0.9589
30	0.9589
40	0.9571
50	0.9571
100	0.9571
200	0.9571
250	0.9571
500	0.9571
2000	0.9571
4000	0.9565
10000	0.9565

The best regularization strength for L1 logistic classifier is 10.

Table 6: Average validation accuracies for L2 Regularizarion

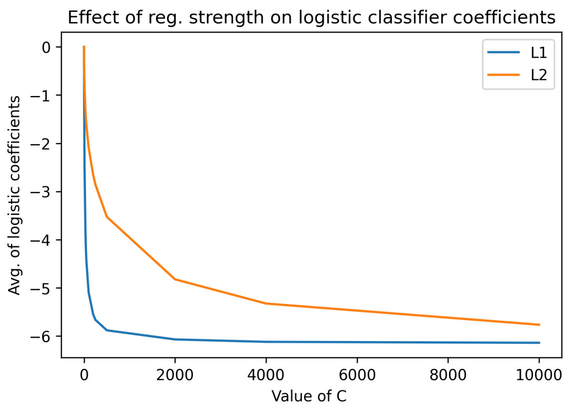
L2 Regularization Parameter	Average Validation Accuracy
0.0001	0.9209
0.001	0.9432
0.01	0.9480
0.1	0.9468
1	0.9523
10	0.9553
30	0.9559
40	0.9553
50	0.9559
100	0.9589
200	0.9619
250	0.9625
500	0.9577
2000	0.9583
4000	0.9583
10000	0.9583

The best regularization strength for the L2 logistic classifier is 250. Table 7 compared performance of Logistic regression without regularization, with L1 regularization and with L2 regularization in terms of accuracy, recall, precision and F1 score.

Table 7: Performance Metric for Logistic Regression

Performance Metric	Logistic Regression (w/o Regularization)	Logistic Regression (L1 Reg. C = 10)	Logistic Regression (L2 Reg. C = 250)
Accuracy	0.9759	0.9759	0.9759
Recall	0.9758	0.9758	0.9758
Precision	0.9761	0.9761	0.9761
F1 Score	0.9758	0.9758	0.9758

Figure 11 shows effects of regularization strength on learned coefficients and classifier accuracy. Regularization strength is inversely proportional to C value.



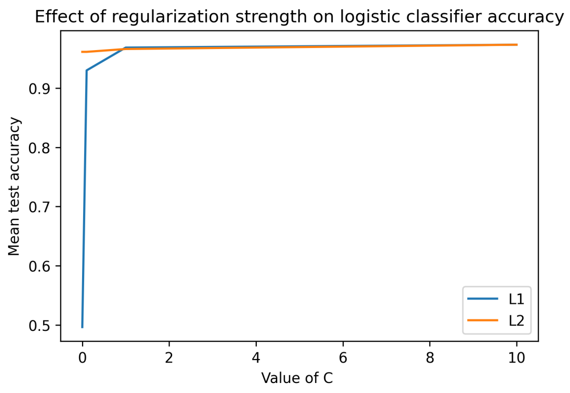


Figure 11: (Left) Effect of Reg. strength on coefficients (Right) Effect Reg. strength on Test Accuracy

As can be seen by figure 11, as regularization strength increases avg coefficients and test accuracy decreases. Models lose most significant weigths to differentiate between the classes as the models become straightforward. As a result, larger value of reg. strength will cause accuracy to drop. Although, finite values of regularization prevent overfitting.

If we know the process of data generation then we can estimate model parameters according to this and use logistic regression without regularization. If the model is not flexible there won't be a need to regularize but it will not generalize well too.
L1 regularization is preferred when there are high number of features as it provides sparse solutions. There is also a computational advantage as features with zero coeffiecients can be avoided.
L2 regularization can deal with multicollinearity(multiple variables highly correlated) problem through constricting the coefficient and by keeping all the variables. L2 regression can be used to estimate the significance of predictors and based on that it can penalize the insignificant predictors.

Logistic regression uses probability between 0 to 1, and this probability with a predefined threshold is used to find the decision boundary between two classes. In SVM, we find points closest to the line from both classes. These points are called support vectors and the SVM tries to make the decision boundary such that the distance between two classes is maximum. From these two methods we can say that SVM will always find a decision boundary that will reduce misclassification rate. Logistic regression may not always find the most optimal decision boundary.
Logistic regression is more prone to outliers as logistic regression diverges faster than SVM. Logistic regression gives an input between 0 and 1 and does not give an exact number which may be considered as an advantage. Logistic regression classifies data away from the margins effectively but fails near the margins. SVM generalizes better and is better at classifying more complex unstructured data. Whereas, logistic regression is more suited for structured data.

7 Question 7

Naive Bayes classifiers are a family of probabilistic classifiers which apply Bayes theorem with the assumption of strong independence between features. In Gaussian Naive Bayes classifier the continuous features are assumed to follow a gaussian distribution. When plotted, its a bell shape curve symmetric about the mean of the feature value. The conditional probability in this case is given by the following equation.

$P(x_i|y) = \frac{1}{2\pi{\sigma_y}^2} exp(-\frac{(x_i-\mu_i)^2}{2{\sigma_y}^2})$

The loss function for Naive Bayes algorithm is given as follows:

$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^{n} P(x_i|y)$

Figure 12 shows the ROC curve and Confusion matrix for Naive bayes algorithm. Table 7 reports accuracy, recall, precision and F1 score values for the algorithm.

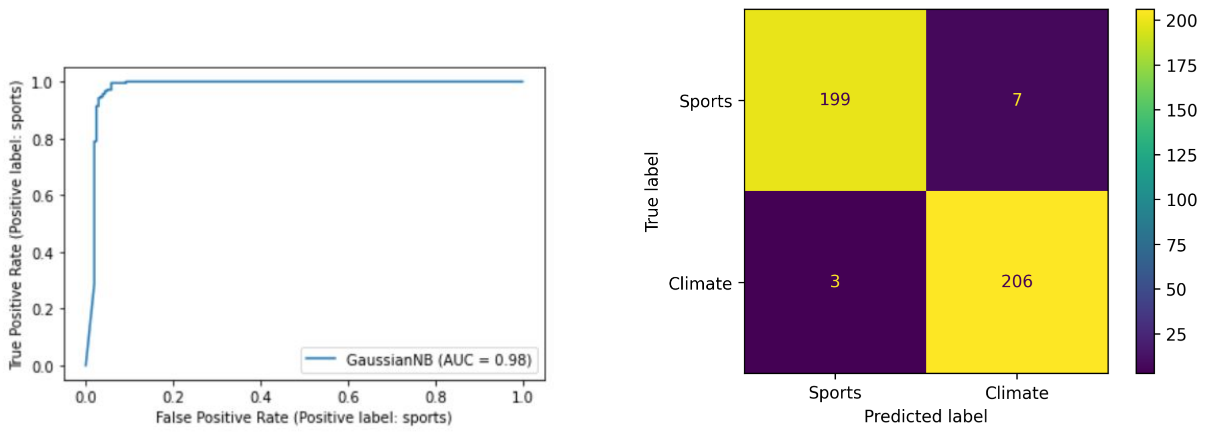


Figure 12: ROC Curve and Confusion Matrix for Naive Bayes

Table 8: Performance Metric for Gaussian Naive Bayes

Performance Metric	Gaussian Naive Bayes
Accuracy	0.9566
Recall	0.9566
Precision	0.9566
F1 Score	0.9567

8 Question 8

Construct a Pipeline that performs feature extraction, dimensionality reduction and classification. - In this section, we combine everything we have done in previous sections into a GridSearch task. The combinations employed in the grid search are :

1. Loading Data - Cleaned Data (Removing Punctuation , Numericals, HTML artefacts) v/s not cleaned data (Not removing Punctuation , Numericals, HTML artefacts)
2. Feature Extraction - min_df = 3 vs 5 while constructing the vocabulary; AND used Lemmatization vs Stemming vs Nothing as a compression module
3. Dimensionality Reduction - LSI (k = [5, 50, 500]) vs NMF (k = [5, 50, 500])
4. Classifier - SVM with C = 300, Gaussian Naive Bayes, Logistic L1 regression with C=10, Logistic L2 regression with C=250.

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

Table 9: Grid Search Results for Cleaned Data

Rank	CLF	Reduced Dim.	Vect_Analyzer	Min_df	Train_Score	Test_Score
1	L2 Logistic Regression	SVD(K=500)	Stemming	5	0.944	0.985
2	L1 Logistic Regression	SVD(K=500)	No normalization	5	0.943	0.976
3	L2 Logistic Regression	SVD(K=500)	Lemmatized	5	0.943	0.981
4	L1 Logistic Regression	SVD(K=500)	No normalization	3	0.943	0.971
5	L1 Logistic Regression	SVD(K=500)	Lemmatized	5	0.942	0.978

Table 10: Grid Search Results for Not Cleaned Data

Rank	CLF	Reduced Dim.	Vect_Analyzer	Min_df	Train_Score	Test_Score
1	L2 Logistic Regression	SVD(K=500)	Stemming	5	0.945	0.983
2	L2 Logistic Regression	SVD(K=500)	Lemmatized	5	0.944	0.978
3	L2 Logistic Regression	SVD(K=500)	No normalization	5	0.943	0.971
4	L1 Logistic Regression	SVD(K=500)	No normalization	3	0.943	0.968
5	L1 Logistic Regression	SVD(K=500)	Lemmatized	3	0.943	0.976

Table 11: Best combination of Grid Search Results

Rank	Clean	CLF	Reduced Dim.	Vect_Analyzer	Min_df	Train_Score	Test_Score
1	Not Clean	L2 Logistic Regression	SVD(K=500)	Stemming	5	0.945	0.983
2	Clean	L2 Logistic Regression	SVD(K=500)	Stemming	5	0.944	0.985
3	Not Clean	L2 Logistic Regression	SVD(K=500)	Lemmatized	5	0.944	0.978
4	Clean	L2 Logistic Regression	SVD(K=500)	Lemmatized	5	0.944	0.978
5	Clean	L2 Logistic Regression	SVD(K=500)	No normalization	5	0.943	0.971

The test score results for cleaned and not cleaned data are approximately the same, with the highest score of 0.985 for cleaned data, and 0.983 for non cleaned data. The test score pattern doesn't exactly match the pattern for the score results reported on the training set.

9 Question 9

For Naive Bayes, we used Gaussian Naive Bayes classifier.
For SVM, as mentioned, there are two approaches.

- In one v/s one classification, the entire dataset is partitioned into n! binary datasets and n! classifiers are trained on the dataset.
- In one v/s rest classification, the entire dataset is partitioned into n datasets, with each dataset containing a positive class and the remaining negative. There are n classifiers trained on the entire dataset.

The F-1, accuracy, precision and recall scores for all three methods are shown in the table below.

Table 12: Performance Metric for Multi-class Classification

Performance Metric	Gaussian Naive Bayes	SVM One v/s One	SVM One v/s Rest
Accuracy	0.8626	0.9181	0.9205
Recall	0.8640	0.9180	0.9216
Precision	0.8710	0.9174	0.9209
F1 Score	0.8628	0.9174	0.9200

The confusion matrix in each multi-class method has a distinct feature in that the major diagonal of the confusion matrix has the largest values indicating that true label and predicted label in the three methods match. This means that the accuracy of the model is high since there are very few misclassifications. We decided to merge the 'cricket', 'soccer', 'football' and 'chess' into one category and the rest into another. We observed that confusion matrix values for all these labels were quite similar with very few misclassifications. After the classes are merged, we observed that the distribution of data points became more balanced in terms of positives and negatives (0s and 1s) When we used the new merged labels, we observed an increase in accuracy and other metric values. This may be attributed to the fact that the number of labels are now fewer and the model can only fall into so many categories. However, this would not mean that the model is more accurate simply by looking at its value.

Some of the techniques used to overcome class imbalance are considering alternative performance metrics like ROC curves instead of accuracy and F1 score, penalizing using loss function, and generating additional synthetic samples.

The new performance metrics after merging labels are shown in the table below.

Table 13: Performance Metric for Multi-class Classification with merged labels

Performance Metric	Gaussian Naive Bayes	SVM One v/s One	SVM One v/s Rest
Accuracy	0.9566	0.9759	0.9759
Recall	0.9566	0.9758	0.9758
Precision	0.9566	0.9761	0.9761
F1 Score	0.9566	0.9759	0.9759

The confusion matrices are shown below :

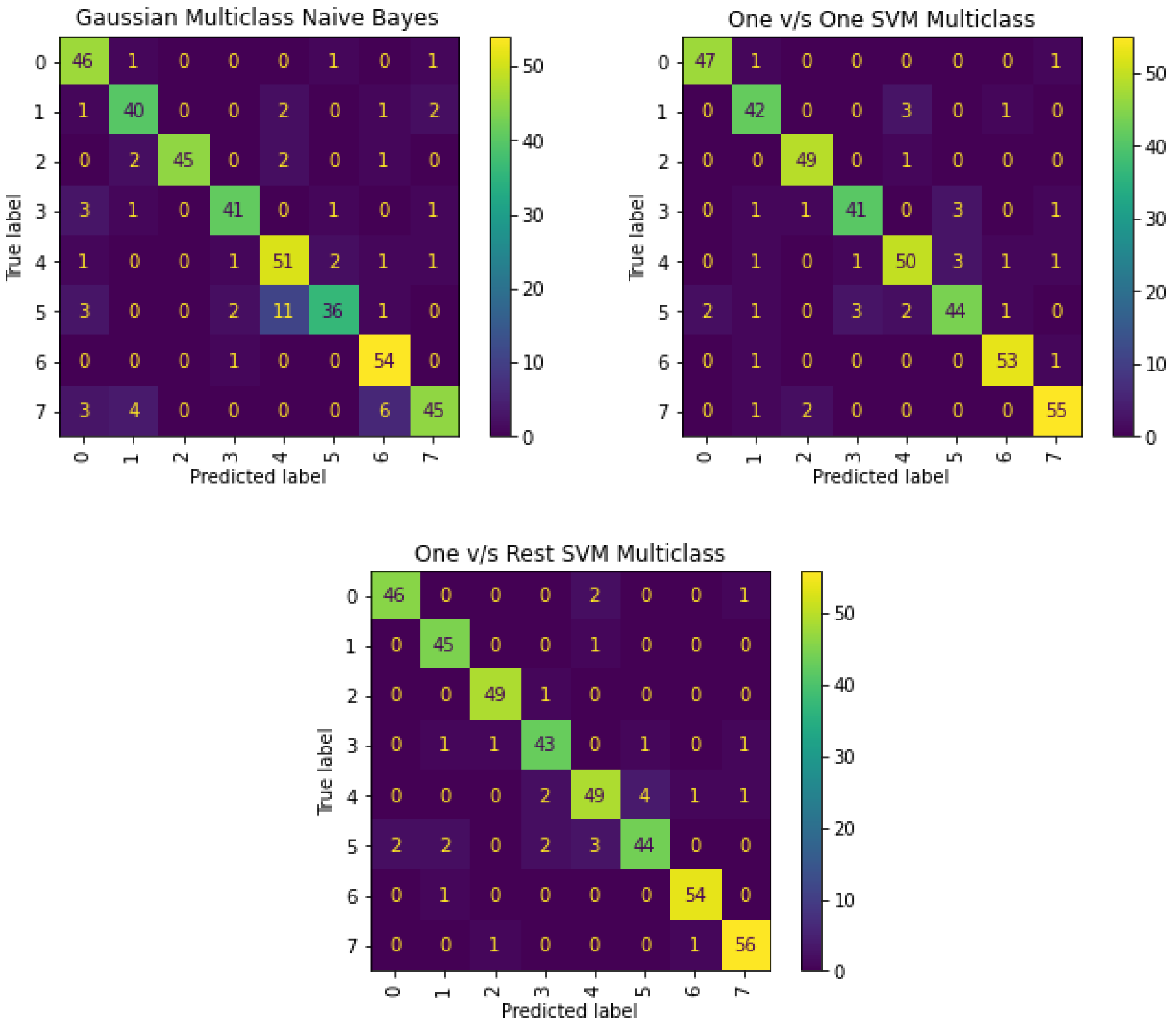


Figure 13: Confusion Matrices for (left) Naive Bayes Multi-Class, (right) SVM One v/s One Multi-Class and (center) SVM One v/s Rest Multi-Class with the labels in order (0-7)

The confusion matrices with merged labels are shown below:

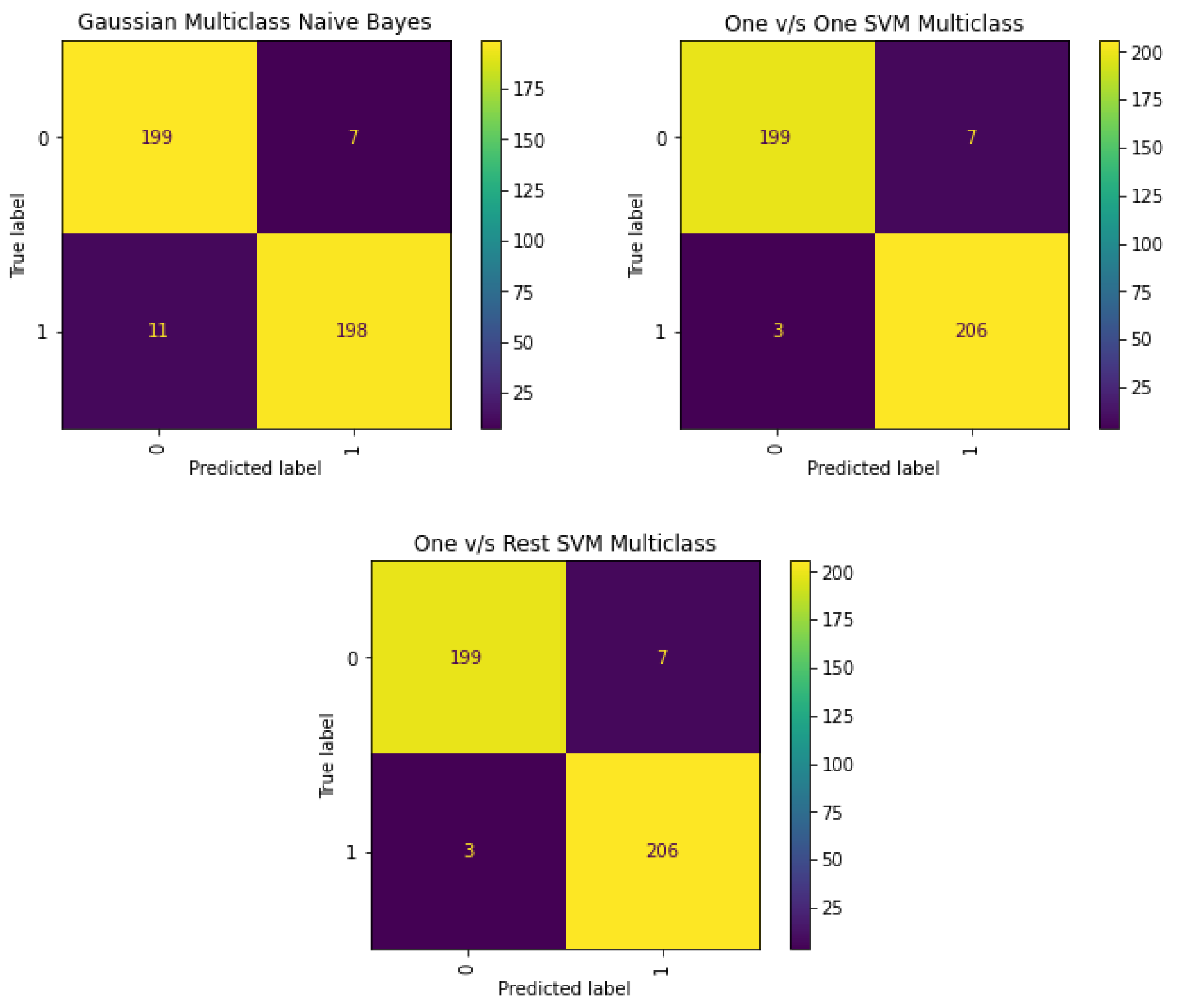


Figure 14: Confusion Matrices for (left) Naive Bayes Multi-Class, (right) SVM One v/s One Multi-Class and (center) SVM One v/s Rest Multi-Class for merged labels (root label)

10 Question 10

10.1 (a)

From the paper, consider two target words i and j used with probe words k. The authors consider a few scenarios to determine whether the raw probability is a good measure to train GLoVE embeddings or a ratio of co-occurrence probabilities are needed.

- Probe words k that are related to i but not j are expected to have a large co-occurrence probability ratio $(P_{ik}/P+jk)$. This is evident from the table that shows the results. Similarly, for probe words k that are more related to j but not i have a low co-occurrence probability ratio.
- As expected, probe words k that are either related to both i and j or have no relation with either i or j show a co-occurrence probability close to **1**

From the table, it is clear that raw probabilities for the above values are not distinguishable between each other. The ratio is better able to differentiate between relevant and non-relevant words. It is also able to discriminate between two relevant words. This is why GLoVE embeddings are trained on the ratio of co-occurrence probabilities rather than the probabilities themselves.

10.2 (b)

No, it does not return the same vector for the word **”running”** in both cases. The reason is the GLoVE embeddings use a distributed representation that has some dependency on the context words, in this case **”park”**

and **"presidency"**. It does not use an independent distribution scheme and instead relies on conditional probability ratios that represent relationships with different words.

10.3 (c)

The glove embeddings were loaded using the code give in the question. We obtain the following results:
1.$||queen - king - wife + husband||_2 = 6.1650$
2.$||queen - king||_2 = 5.966$
3.$||wife - husband||_2 = 3.152$
A possible explanation for the values obtained is that the words 'wife' and 'husband' are more analogous than the words 'queen' and 'king'. This explains the fact that the L_2 norm of the former is lower than queen and king. This means that in the vector space, the spatial distance of king from queen is greater than husband from wife.
The difference of the two terms (i.e. the first term) is slightly larger than $||queen - king||_2$. The reason for this is since one of the temrs is more analogous than the other, the resulting vector from this operation only marginally increases in the direction of the first term ($||queen - king||_2$. This explains the fact that the first terms is only slightly larger than than the second term.

10.4 (d)

- Stemming crudely removes the last few characters of a word with the aim of simplifying the word into its base form. It does not take into account the context of the word or any morphological analysis. This often leads to words without meaning or context.
- Lemmatization on the other hand, considers context,syntax, and semantics to convert the word into meaningful root forms. (called lemmas) Lemmatization deals only with inflectional variance. Sometimes, the same word can have multiple lemmas as well. Lemmatization in general is a more sophisticated process than stemming.

Although stemming is faster than lemmatization, it usually has a much higher error rate than lemmatization. Thus, in order to preserve context and push it into GLoVE embeddings, lemmatization makes more sense.

11 Question 11

11.1

We used the following feature engineering process for Glove:

- We used glove.6B.100d.txt which contains glove vectors trained on Wikipedia and Gigaword dataset.
- For convinience, we converted glove file with word embeddings into word2vec format. glove2word2vec from gensim library was used for this. This helps in representing the embedding as a mapping key which would map each word to the corresponding vector.
- After loading our dataset, we cleaned the data, removed punctuation marks and numbers from the data.
- Once we had cleaned data in the correct format, we need to assign a vector per sentence. We already have a vector per word. Now, we find the n^{th} value of sentence embedding by taking mean of all the n^{th} values in word embeddings for all words in the sentence. This helps in highlighting important topical words as mentioned in the question.
- The class in our code basically transforms each sentence into a vector as described above.
- Then, we create a vectorizer which transforms every sentence into a numerical value, a vector. The length of vector in this case would be 300 as we chose the dimension of glove as 300.

11.2

We used SVM as the classifier for this section. We used a 5 fold cross validation pipeline to find the best value of γ which we found out to be 20. The performance metrics for the same can be found in table 13.

Table 14: Performance Metric for Linear SVM Classifier with GLOVE embeddings

Performance Metric	Gaussian Naive Bayes
Accuracy	0.9542
Recall	0.9271
Precision	0.9794
F1 Score	0.9526

12 Question 12

We have retrained the SVM classifier used in the previous question with C = 5 for the different glove dimensions - 50,100,200, and 300.

From the plot below we see that as the dimension of the Glove embedding increases, the test accuracy also increases. This is the expected trend. The higher dimensions of these Glove embeddings capture more semantic and complex information regarding the distributive representation of words. They also provide more information on the co-occurrence probability ratios between the target and context words in the latent space, which provide more features that can contribute to a higher test accuracy.

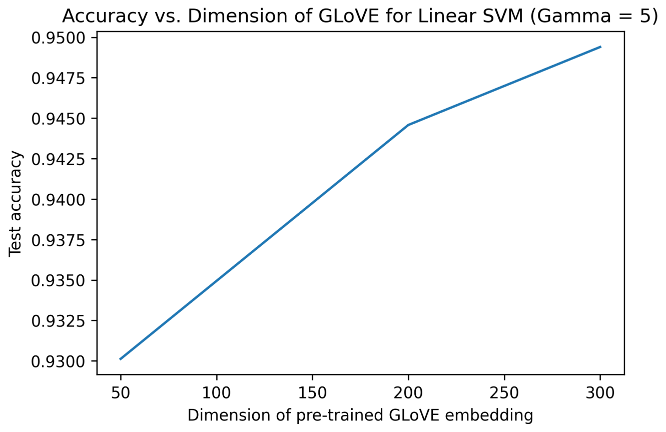


Figure 15: Plot of Accuracy vs Dimension of GLoVE for best SVM estimator

13 Question 13

Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation and also for general non-linear dimension reduction.

The 2 visualizations of the normalized GLoVE based features are shown in the figure below.

Figure on the left shows the plot for GLoVE features in 2D generated using the UMAP library for n = 300 for the 2 classes - sports and climate. The figure on the right shows the plot for random vectors in a 2D plane. We used the Euclidean distance metric in UMAP, with 15 neighbours being used to approximate the distance with minimum distance between points as 0.1. Random variables are generated from a normal gaussian distribution with mean 0 and standard deviation 1.
Clusters are only seen in the first plot for the 2 categories climate and sports.

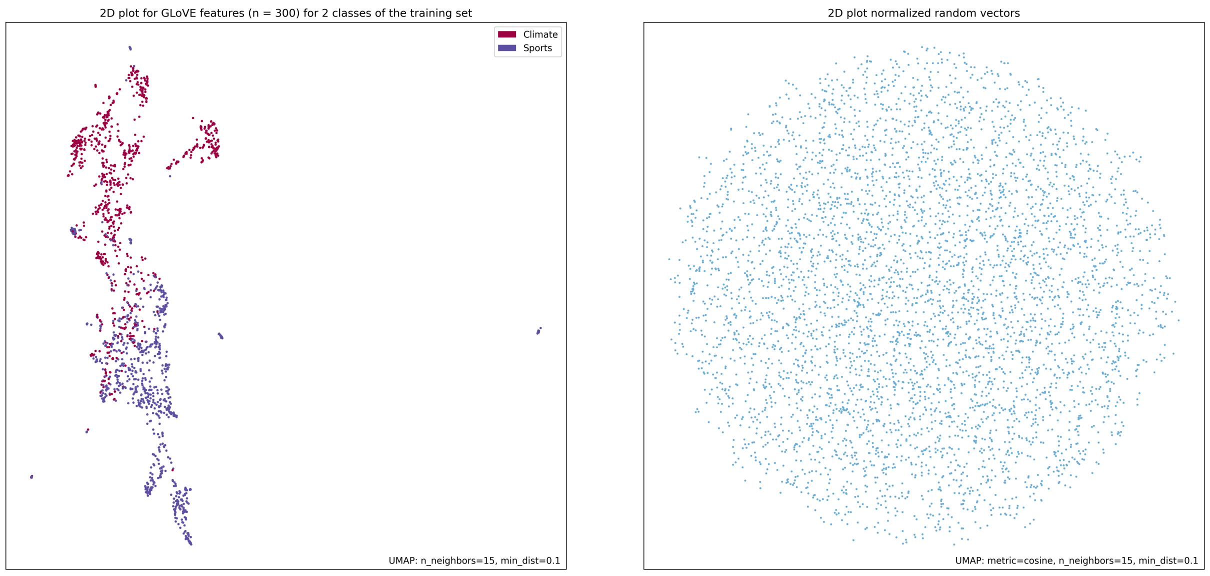


Figure 16: (Left) 2D visualization of GLoVE embeddings (n = 300) for two classes in the news dataset. (Right) 2D visualization of random vectors of the same dimension as GLoVE embeddings