Project 1 report

Hexi Meng (406200552),Zhanhong Liu(206152835)

- **Introduction**

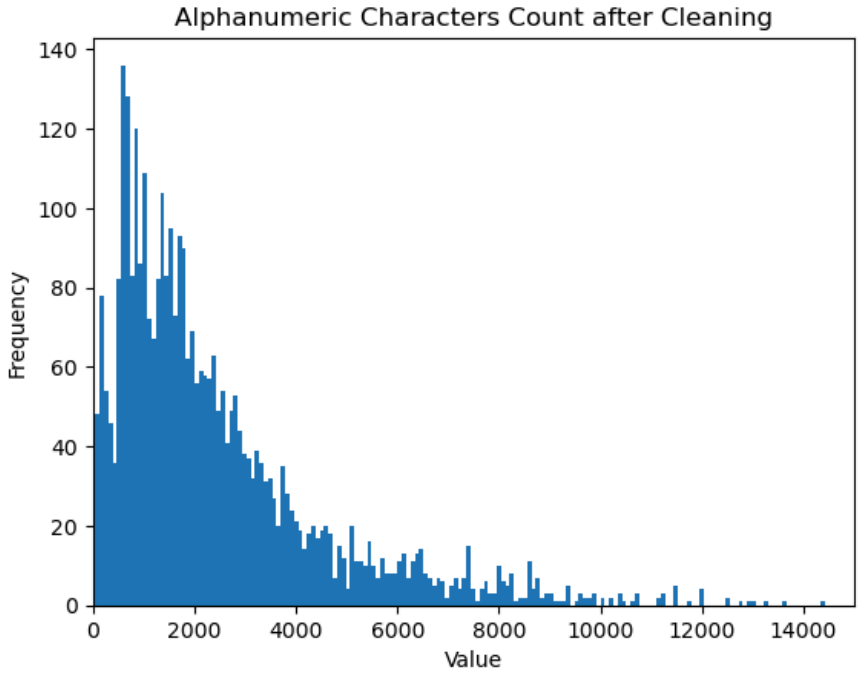
  This project delves into the realm of machine learning with a focus on classifying textual content, specifically targeting news articles. Our project is structured around the development of an end-to-end pipeline designed for the categorization of text data, employing a suite of machine learning methodologies. The primary objective is to accurately assign predefined categories to individual news articles. The pipeline comprises the following key components.This report presents a comprehensive exploration of these components, each contributing uniquely to the accurate classification of news articles. Through this project, we aim to showcase the effectiveness of combining traditional machine learning techniques with advanced pre-trained models in the field of text classification.

- **Getting familiar with the dataset**

  Question 1

  – There are 3476 rows and 8 columns in the datasheet

  (a)

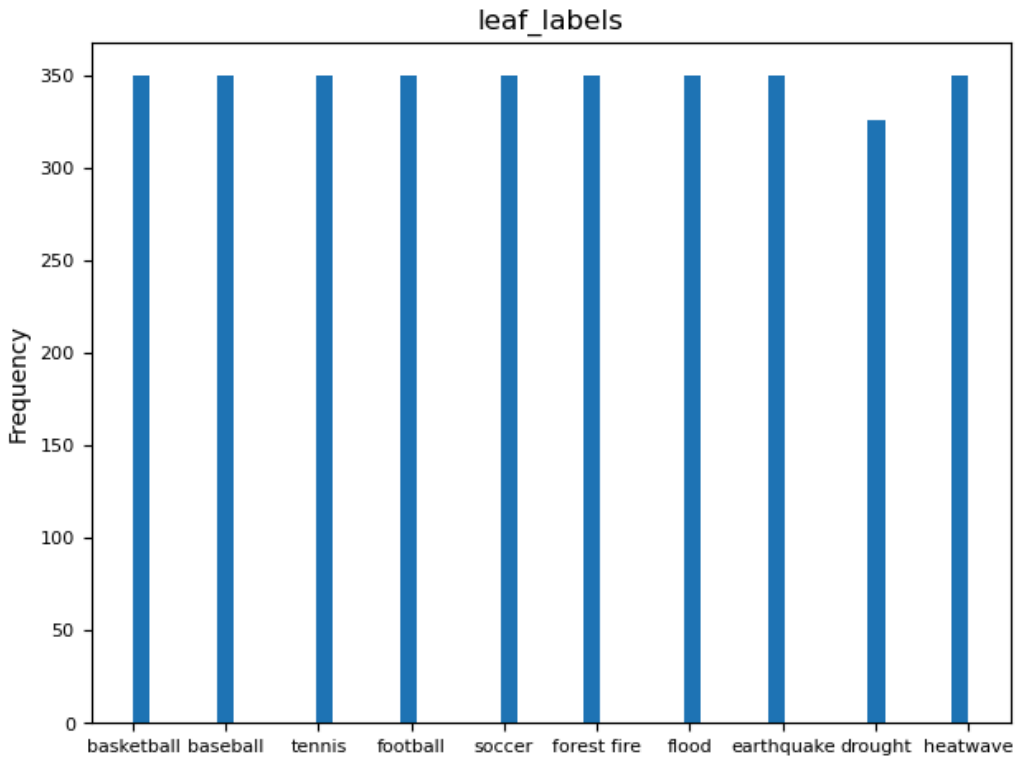


Interpretation of the plot

1.Most articles have no more than 8000 alpha-numeric characters,only few article have more than 10000 alpha-numeric characters

2.The distribution of alphanumeric character counts is right-skewed, as most of the data points are clustered on the left side of the histogram

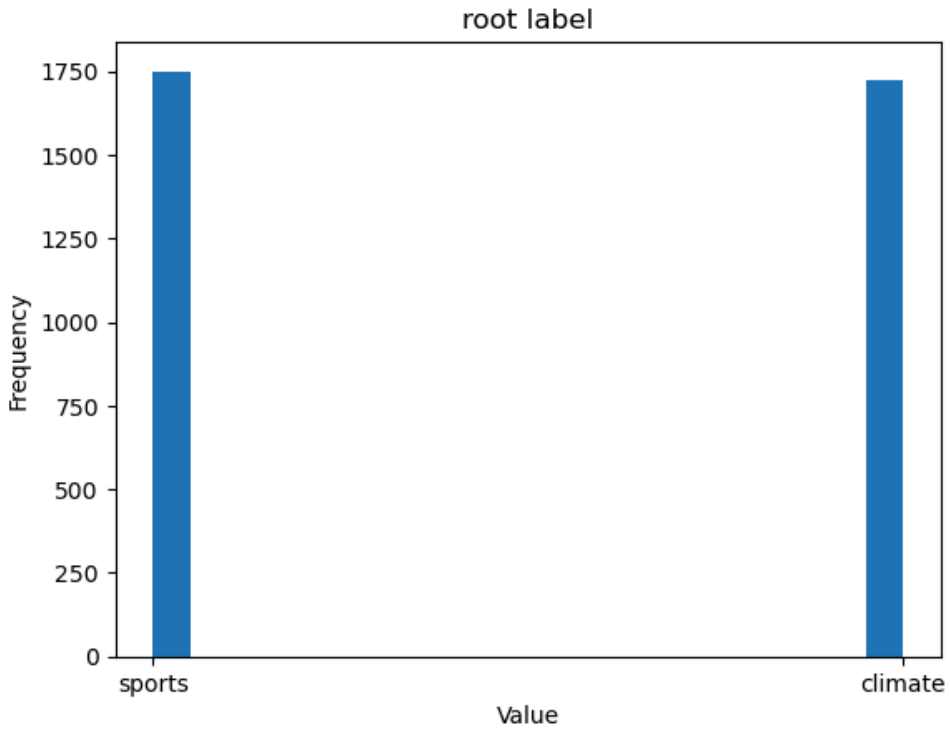
(b)



Interpretation of the plot

1.All the leaf label sports occurs about same amount time in the dataset,roughly 350 times ,except the drought

(c)



Interpretation of the plot
1.There are same number of articles related to sports as related to climate,both 1750 articles related to each of the two topic

- **Binary Classification**
  Question 2
  number of training samples : 2780
  number of test samples : 696
- **Feature Extraction**
  Question 3
  1.Lemmatization
  Pros:
  Precision:Lemmatization has higher precision that results in more precise words since it involves a deep linguistic understanding, often using vocabularies and morphological analysis
  Accuracy :It converts the words into same meaningful base form

Cons:
Computationally heavy:It requires more computational resources because it involves understanding the context and the grammatical rules of the language.
Language Dependency:Highly dependent on the language's morphology and requires extensive language-specific resources.
– Steamming
Pros:
Speed:It is way faster than lemmatization since only use heuristic logistics to cuts off the end of words
Simplicity: Easier to implement ,doesn't require knowledge about the language
Cons:
Lack of Accuracy:It use simple heuristics so produces non actual words

2.Increasing min_df will reduce the column number of the tfidf matrix since this reduces the overall size of the vocabulary,more words will be filtered out.

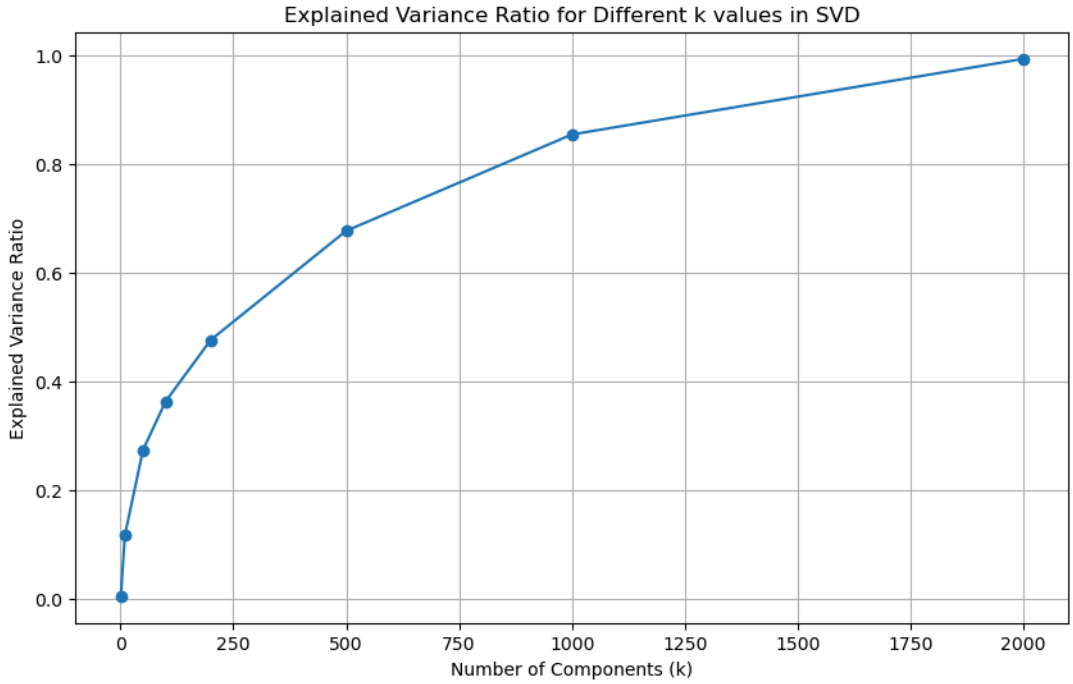
3.One should remove punctuation first as they might be considered part of a word, leading to incorrect lemmatization, then lemmatize the text,then remove stopwords to avoid missing lemmatization and finally remove the number.

4.The size of the TF-IDF-processed train matrix is 2780 x 13635
The size of the TF-IDF-processed test matrix is 696 x 13635

- **Dimensionality Reduction**

  Question 4

  – Plot of explained variance ratio across multiple different k

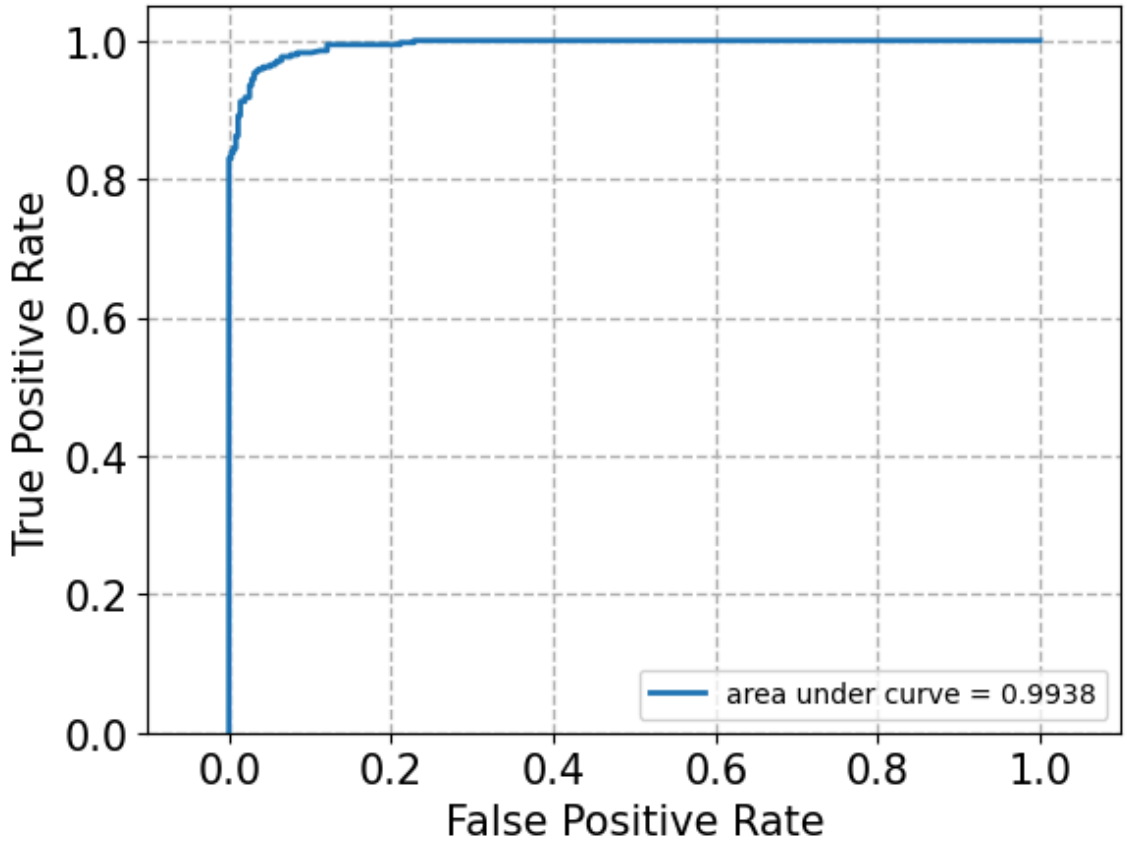


The graph looks like a log exponential curve,increase rapidly then converging,concavity suggesting that as K increases explained variance ratio saturated

– MSE for LSI: 5.221120137248087e-05
MSE for NMF: 5.293630859956141e-05

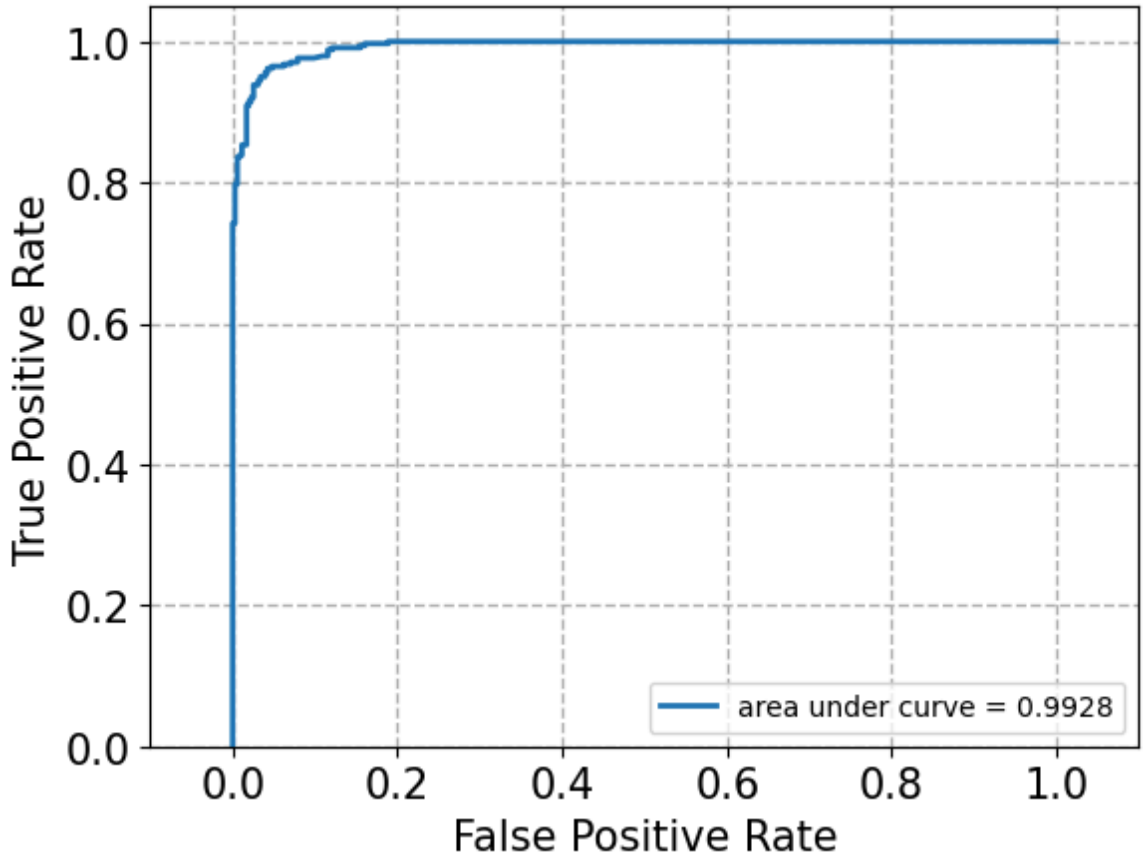
LSI has smaller MSE than NMF , because LSI, via SVD, captures the best lower-rank approximation in the least-squares sense. NMF, on the other hand, imposes non-negativity constraints which may lead to a less optimal fit in terms of least squares.

- **Classification Algorithms**
  Question 5
  roc plot
  – γ = 1000 (hard margin)



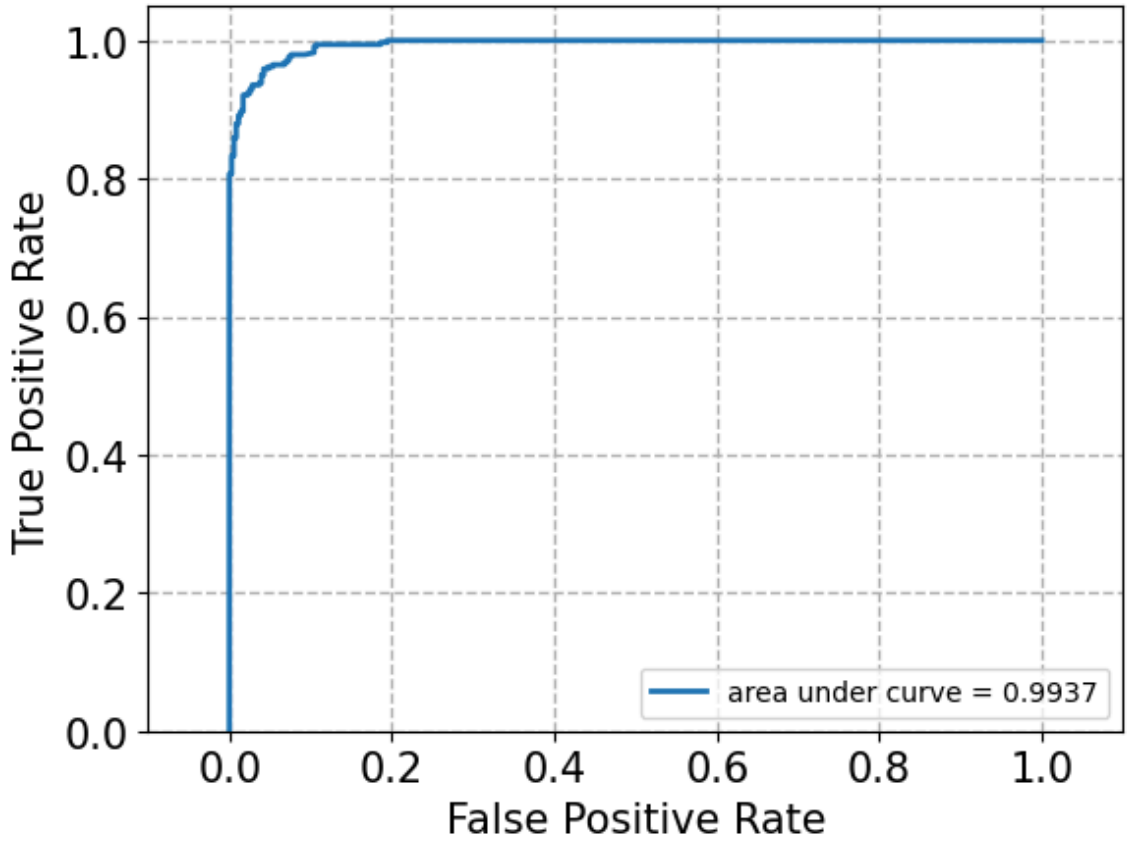

```
Confusion matrix is:
 [[325  30]
 [ 26 315]]
Accuracy:  0.9195402298850575
Recall:  0.9237536656891495
Precision:  0.9130434782608695
F-1 score:  0.9183673469387754
```

– γ = 0.0001 (soft margin)




```
Confusion matrix is:
 [[  0 355]
 [  0 341]]
Accuracy:  0.4899425287356322
Recall:  1.0
Precision:  0.4899425287356322
F-1 score:  0.6576663452266153
```

– γ = 100000




```
Confusion matrix is:
 [[339  16]
 [ 16 325]]
Accuracy:  0.9540229885057471
Recall:  0.9530791788856305
Precision:  0.9530791788856305
F-1 score:  0.9530791788856305
```

hard margin performs better, for γ = 100000 , the accuracy decreases possibly due to training data overfitting

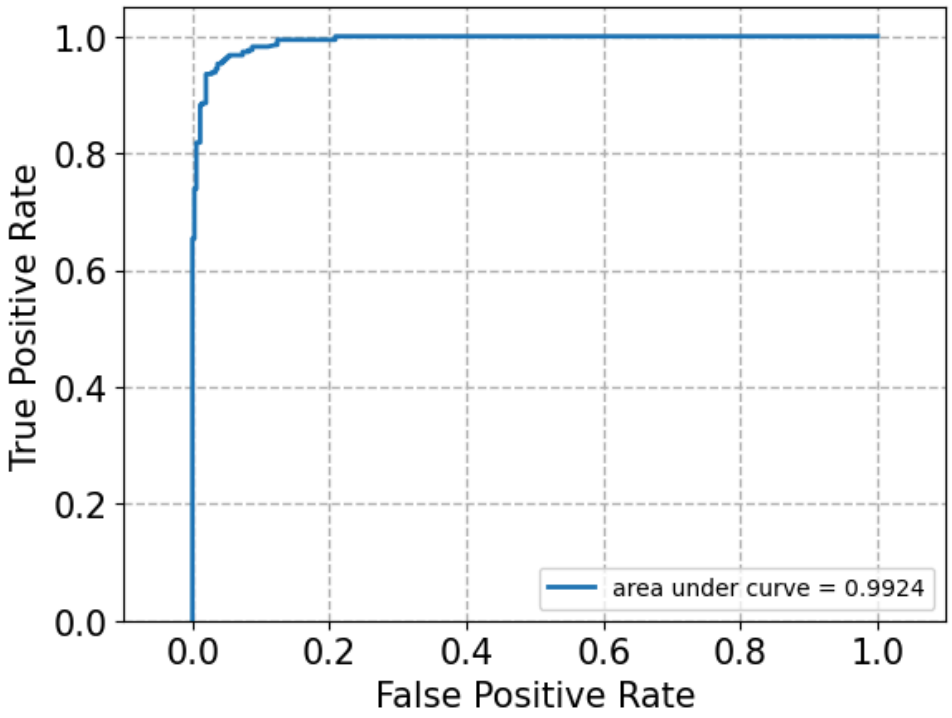
– we used following code to find best performance c value is 100

```
c=[10**-3, 10**-2, 10**-1, 10, 100, 1000, 10000, 100000, 100000]


validation_accuracy=[]

for i in c:
  svm_opt = SVC(C=i)
  scores = cross_val_score(svm_opt, X_train_svd, X_train_label, cv=5)
  scores = scores.mean()
  validation_accuracy.append(scores)

print(validation_accuracy)
max_accuracy = max(validation_accuracy)
max_index = validation_accuracy.index(max_accuracy)
optimal_c = c[max_index]

print("Optimal C value:", optimal_c)
print("Maximum accuracy:", max_accuracy)
```

Output:

```
 [0.5068345323741006, 0.8870503597122303, 0.9510791366906475,
0.9651079136690648, 0.966546762589928, 0.9593525179856115, 0.9593525179856115,
0.9593525179856115, 0.9593525179856115]
Optimal C value: 100
Maximum accuracy: 0.966546762589928
```

– γ = 100



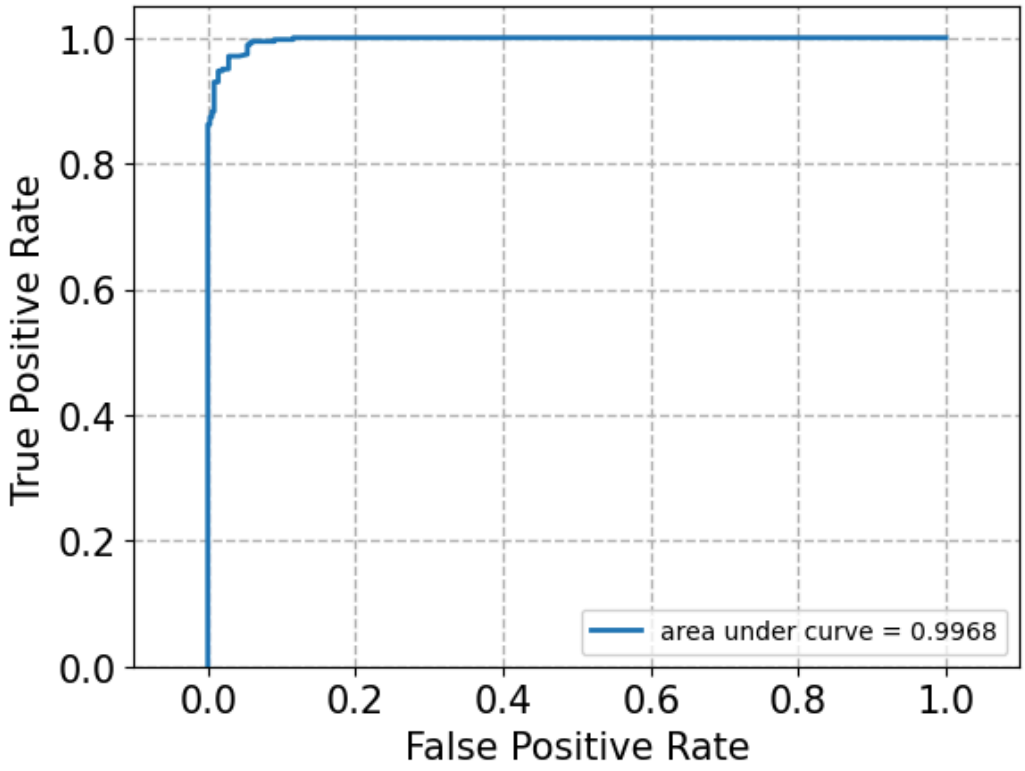

```
Confusion matrix is:
 [[337  18]
```

```
 [ 11 330]]
Accuracy:  0.9583333333333334
Recall:  0.967741935483871
Precision:  0.9482758620689655
F-1 score:  0.9579100145137881
```

- **Logistic regression**

  Question 6

  -Roc plot




```
Confusion matrix is:
 [[344  11]
 [ 10 331]]
Accuracy is: 0.9698275862068966
Recall is: 0.9706744868035191
Precision is: 0.9678362573099415
F1 score is: 0.9692532942898975
```

-performance data

```
Results for No Regularization:
Accuracy: 0.9554597701149425
Precision: 0.9532163742690059
Recall: 0.9560117302052786
F1 Score: 0.9546120058565154


Results for L1 Regularization:
Accuracy: 0.9568965517241379
Precision: 0.9533527696793003
Recall: 0.9589442815249267
F1 Score: 0.956140350877193


Results for L2 Regularization:
Accuracy: 0.9583333333333334
Precision: 0.9534883720930233
Recall: 0.9618768328445748
F1 Score: 0.9576642335766424
```

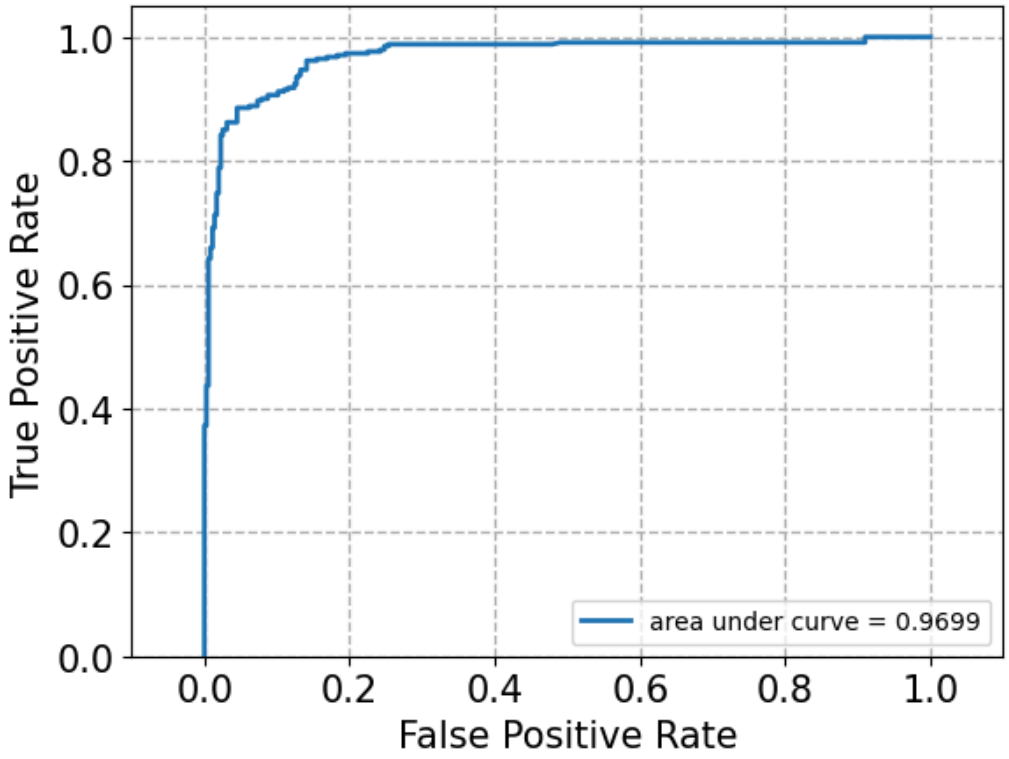
- Best parameters (L1): 10.0
  Best parameters (L2): 100.0
- See output above for performance regarding w/o regularization, w/ L1 regularization and w/ L2 regularization
- Higher regulating parameters reduces testing error, however, if 'C' is too high, the model may start overfitting, and the test error could increase.**L1** can shrink some coefficients to zero, effectively removing those features from the model and make the model simpler therefore it is preferred when we need a sparse model with fewer features whereas **L2** does not result in feature elimination. It can be preferred when all features are expected to contribute,also decreasing the model complexity and increase the stability of the model
- Logistic Regression vs SVM: Logistic regression derive the decision boundary based on the probability estimates(minimum log likelihood), linear SVM's decision boundary is determined by the support vectors and the margin, As a result SVMs can perform better when there's a clear margin of separation in the

data, whereas logistic regression can be more robust to overlapping class distributions.Therefore their performance really depend on the types of dataset,if the dataset is large, logistic regression might be more suitable, whereas if the set is high-dimensional data where a clear margin is present, an SVM might offer better performance.

- **Naive Bayes Model**

  Question 7

  -Roc plot of GaussianNB classifier




```
Confusion matrix is:
 [[299  56]
 [ 13 328]]
Accuracy is: 0.9008620689655172
Recall is: 0.9618768328445748
```

```
Precision is: 0.8541666666666666
F1 score is: 0.9048275862068966
```

- **Grid Search of Parameters**

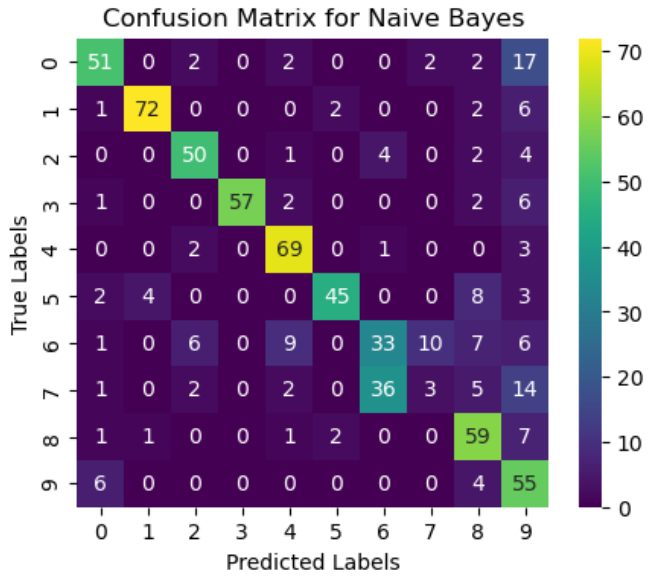
  Question 8

```
Top 5 Pipeline Configurations Based on Accuracy:
 min_df  k (n_components)          Classifier  Accuracy  Precision    Recall  F1 Score
      5                 5  LogisticRegression  0.913793   0.874667  0.961877  0.916201
      5                 5  LogisticRegression  0.913793   0.874667  0.961877  0.916201
      5                 5          GaussianNB  0.906609   0.867021  0.956012  0.909344
      3                 5          GaussianNB  0.875000   0.796729  1.000000  0.886866
      3                 5          GaussianNB  0.875000   0.796729  1.000000  0.886866
```

- **Multiclass Classification**

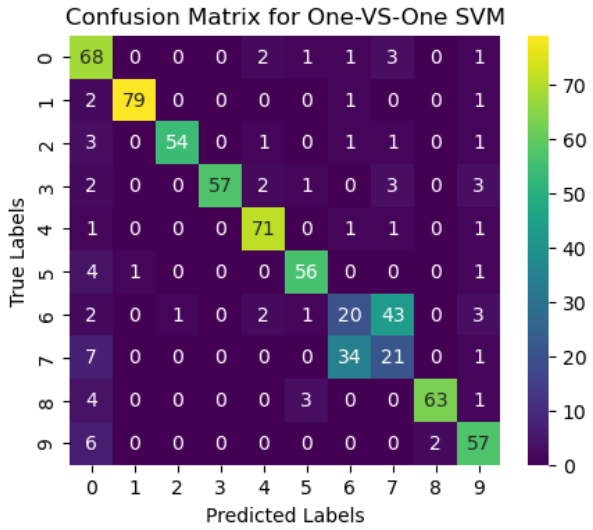
  Question 9

```
Naive Bayes Classifier:
```



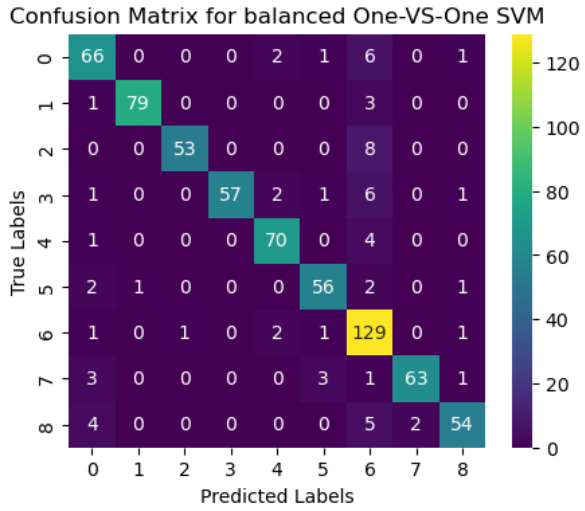

```
Accuracy: 0.7097701149425287
Recall: 0.7025328530574296
Precision: 0.7007927525145334
F1 Score: 0.6870304009240783
```

One-VS-One SVM Classifier:



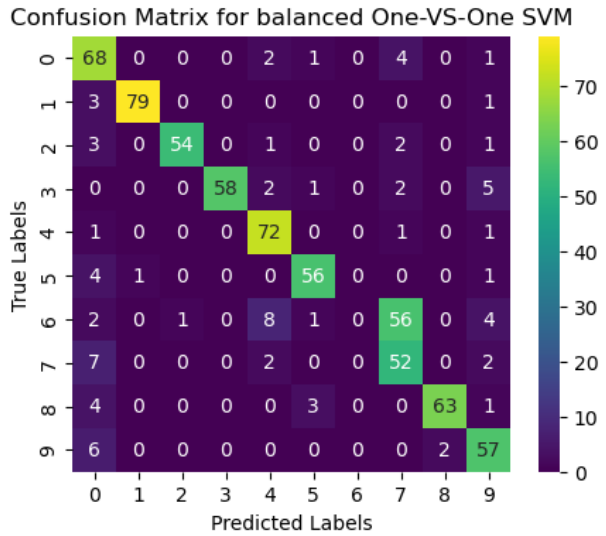
Accuracy: 0.7844827586206896
Recall: 0.7795275871592358
Precision: 0.7889679821784938
F1 Score: 0.7810551326373879

One-VS-One SVM Classifier (Merged):



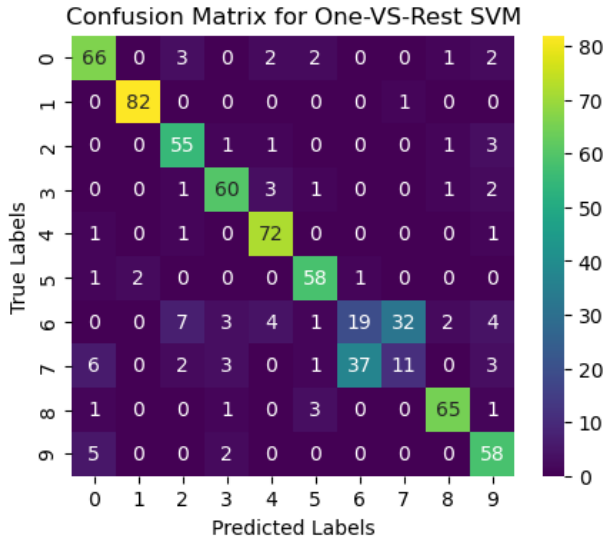
Accuracy: 0.9008620689655172
Recall: 0.7795275871592358
Precision: 0.7889679821784938
F1 Score: 0.7810551326373879

One-VS-One SVM Classifier (Balanced):



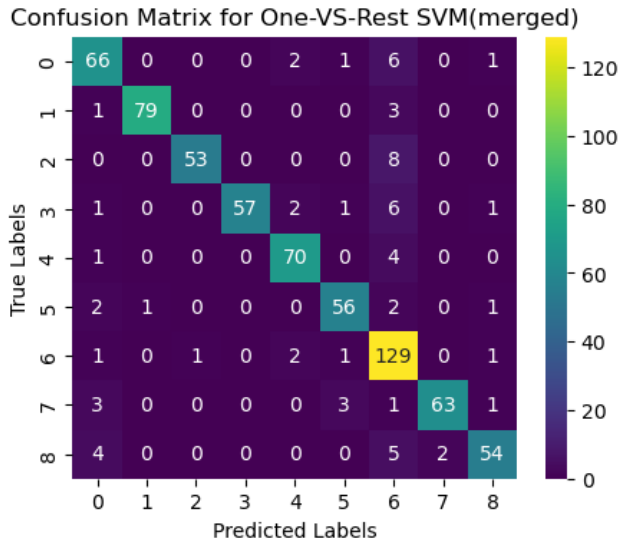
Accuracy: 0.8031609195402298
Recall: 0.8037600801564345
Precision: 0.7577953230132239
F1 Score: 0.7719110698415734

One-VS-Rest SVM Classifier:



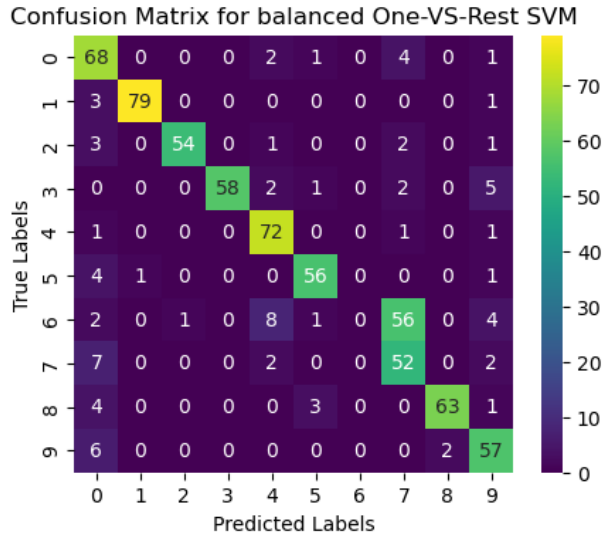
Accuracy: 0.7844827586206896
Recall: 0.7782141729813238
Precision: 0.7507959987572924
F1 Score: 0.7624055599821932

One-VS-Rest SVM Classifier(merged):



Accuracy: 0.9008620689655172
Recall: 0.7782141729813238
Precision: 0.7507959987572924
F1 Score: 0.7624055599821932

One-VS-Rest SVM Classifier (Balanced):



Accuracy: 0.8031609195402298
Recall: 0.8037600801564345
Precision: 0.7577953230132239
F1 Score: 0.7719110698415734

- see confusion matrix and performance data above, the imbalance problem can be solved through setting the class's weight,which adjusts weights inversely proportional to class frequencies.**(performance data and confusion matrix plot shown above)**
- The blocks are distinct on the major diagonal ,suggesting that the classifier is performing well in distinguishing between different classes. Each block represents a high number of correct predictions for a particular class.**(performance data and confusion matrix plot shown above)**
- Based on the confusion matrix in previous part, the label flood and earthquake are should be merged together, as the label been merged ,the accuracy increases for One VS One and One VS the rest classifiers **(performance data and confusion matrix plot shown above)**
- After merging,balancing the class actually decreases the performance ,this might due to the potential overfitting of the minority class.**(performance data and confusion matrix plot shown above)**

- **Word Embedding**

(a) Capturing Word Relationships:Glove uses co-occurrence probabilities ratio so it can capture relationships between words. Ratios help differentiate between relevant and irrelevant associations. For example similar words have closer vectors. The model can therefore learn the relative importance of different words instead of the absolute frequencies of co-occurrences in a different context and captures the meaning of words with effective word embeddings.

- Dimension reduction:co-occurrence probabilities ratio could also allow the word vector to preserve more information when being reduced to lower dimensions,preserving semantic and syntactic information.

(b)No. the GLoVE embeddings will not return the same vector for the word running in both cases.Because it is learning on the ratios of co-occurrence probabilities rather than the probabilities themselves.It will read the word next to 'running',like the first sentence the 'running' is followed by 'in the park' and the second is followed by 'for the presidency'. As a result, vectors will also be different.

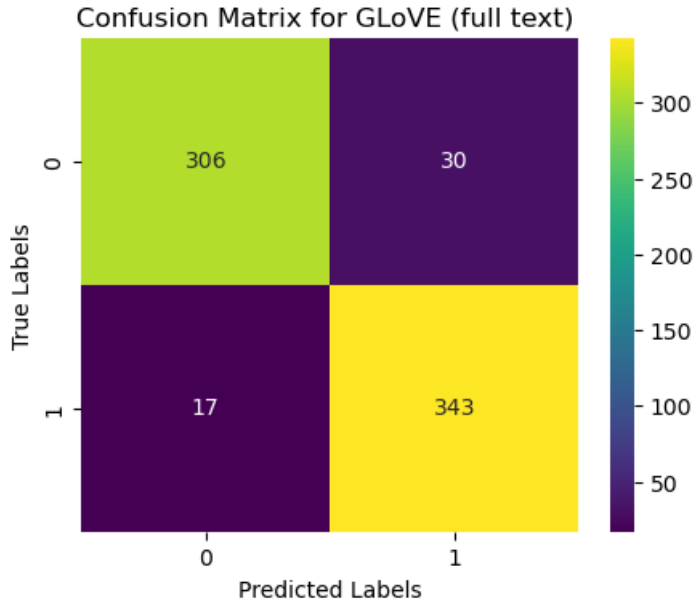
(c)The resulting euclidean distance is shown :

```
Euclidean Distance between 'woman' and 'man': 0.7747692

Euclidean Distance between 'wife' and 'husband': 0.5203094

Euclidean Distance between 'wife' and 'orange': 1.3294425
```

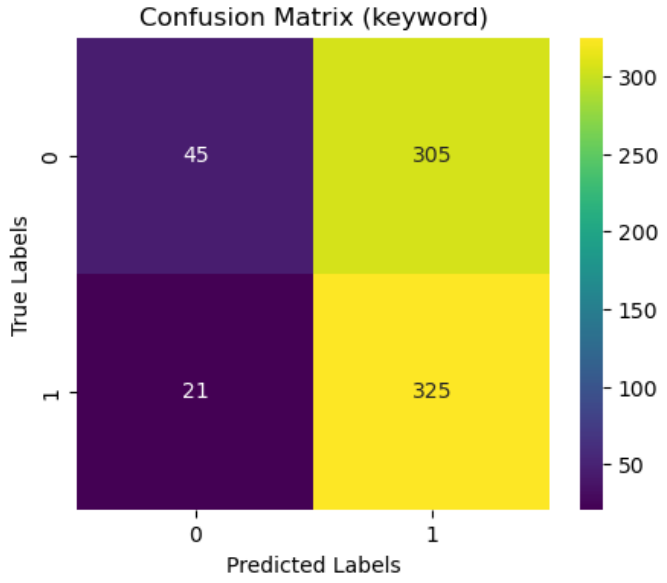
because 'wife' and 'husband' are more closely related to each other therefore the distance between their word vectors is smaller whereas the 'wife' and 'orange' are not so related so it has a larger distance.

(d)I will choose lemmatization rather than stem. Stemming is a process that stems or removes the last few characters from a word based on the heuristic logic, often leading to incorrect meanings and spelling. However, lemmatization preserves the context and converts the word to its meaningful base form. Using stem will be useless for GLoVE because we need the context of words to be correct, stem will decrease the accuracy.

Question 11



Accuracy: 0.9324712643678161
Recall: 0.9527777777777777
Precision: 0.9195710455764075
F1 Score: 0.9358799454297407



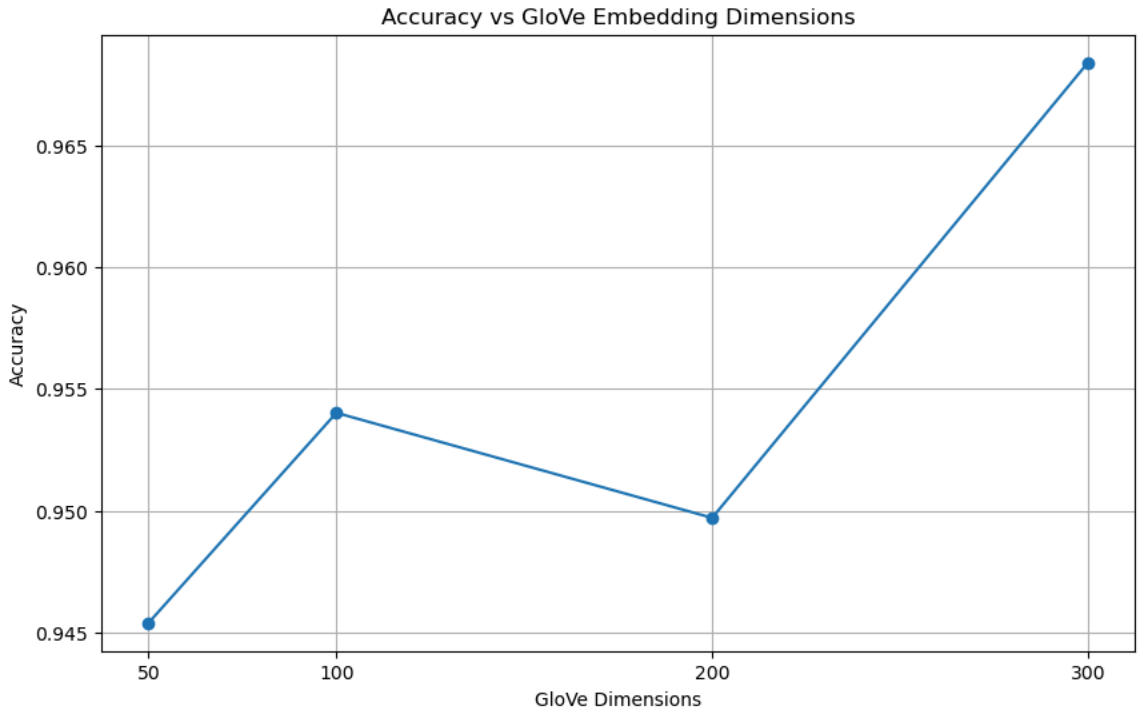
Accuracy: 0.5316091954022989
Recall: 0.9393063583815029
Precision: 0.5158730158730159
F1 Score: 0.6659836065573771

(a)

- Featuring engineering process
  1.Spilt train and test data set by using train_test_split
  2.clean and lemmatize the text by apply corresponding function
  3.apply glove_conv to processes the representation of the text
  4.aggregating word vectors into a single vector by normalizing and averaging vectors
- The function glove_conv read sentence from the input,breaks Them into Words (Tokenization) and compare them with the dictionary(in this case 'glove.6B.200d.txt') ,If a word in the sentence is found in the GloVe dictionary, it is represented by its corresponding embedding (a 300-dimensional vector)

(b)We selected Linear SVM model with γ=10 for better runtime efficiency

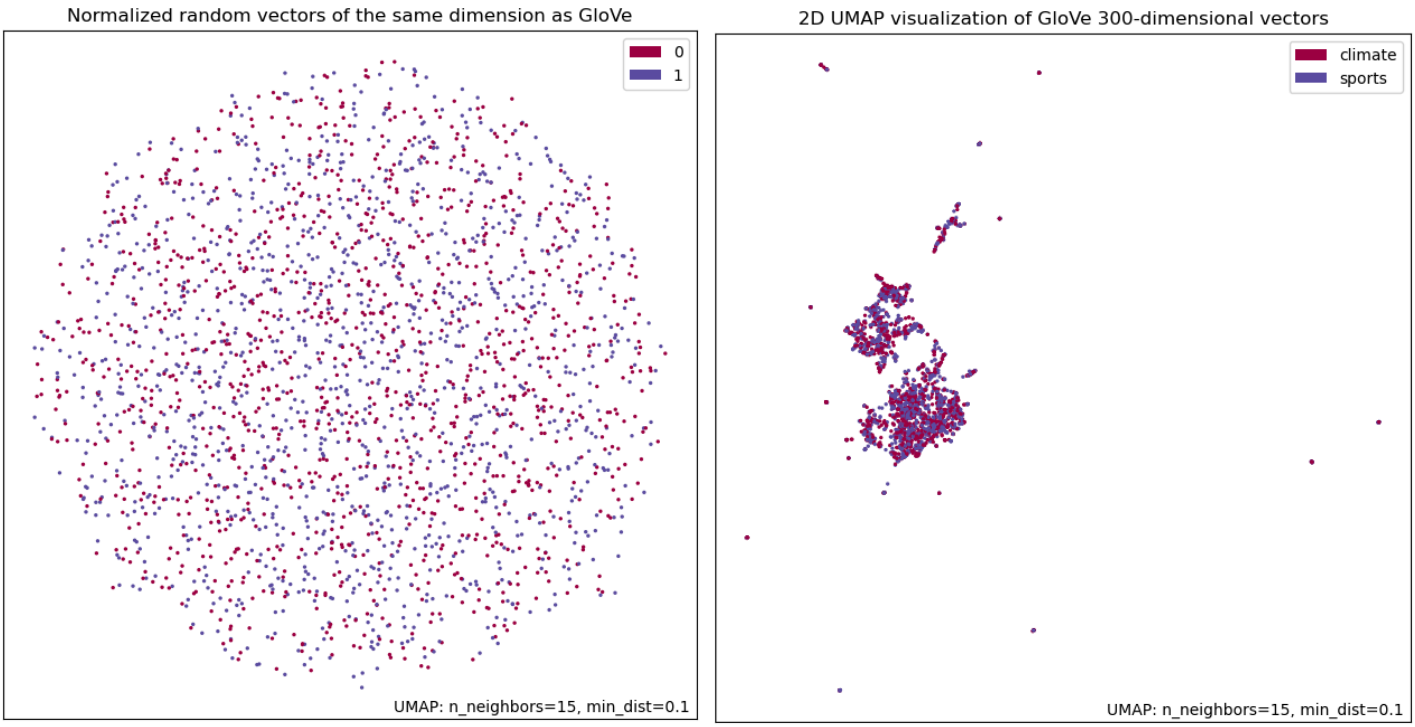
Question 12




```
Accuracy with GloVe 50d: 0.9497126436781609
Accuracy with GloVe 100d: 0.9626436781609196
Accuracy with GloVe 200d: 0.9468390804597702
Accuracy with GloVe 300d: 0.9482758620689655
```

- Above is plot of the dimension of GLoVE embedding and the resulting accuracy of the model, which differs from our expectation because the accuracy drops for GloVe dimension = 200, this might be due to over-fitting caused by higher dimension or it introduces more noise in higher higher dimension for this data.

Question 13



the right plot demonstrates the power of word embeddings like GloVe to capture and represent the semantic meaning of words, as opposed to the unstructured randomness of the vectors in the left plot. This structured representation is crucial for many NLP tasks, such as classification, where the proximity of word vectors can significantly impact the performance of machine learning models.