ECE 219 Large Scale Data Mining - Project #1

Reema Kumari - 905727807, Zhaoliang Zheng - 605432345,
Shivam Kumar Panda - 105730045

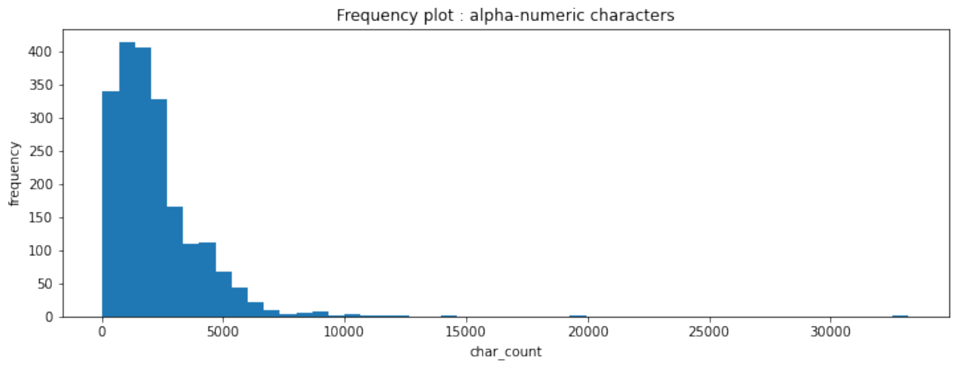
January 24, 2022

End to End Pipeline to classify news articles

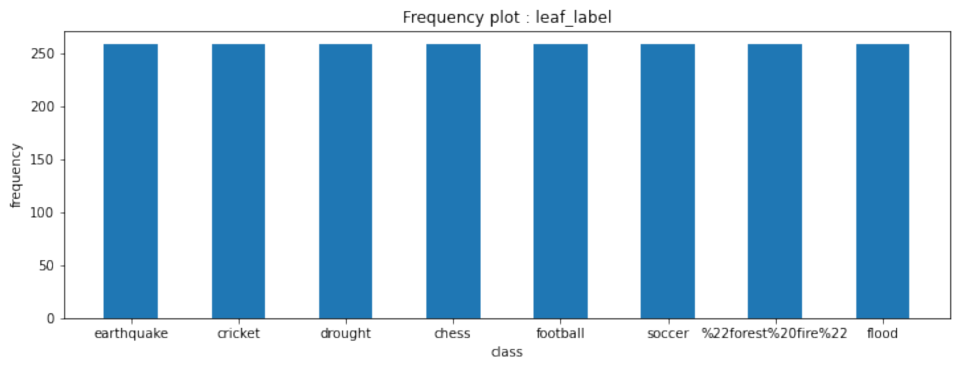
Getting familiar with the dataset

Question 1

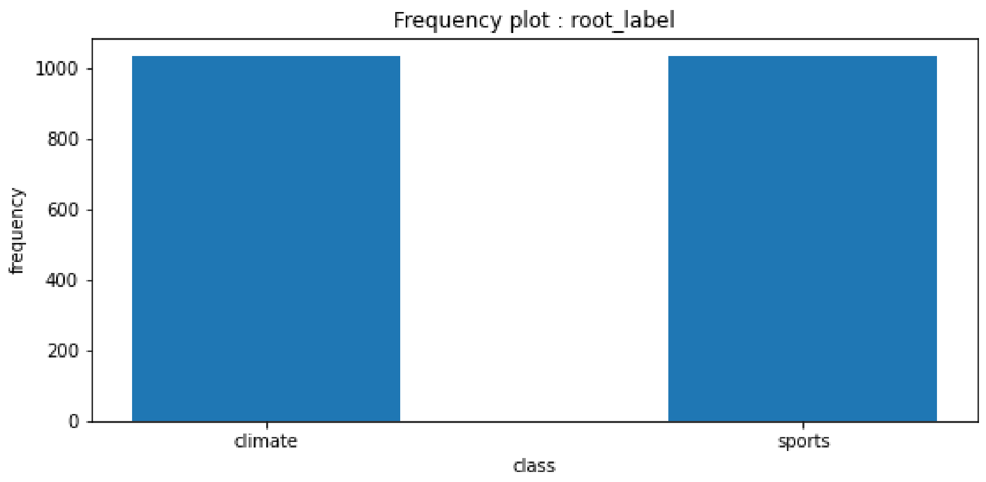
There are in total 2072 rows or samples in the dataset. There are in total 9 columns, however the features describing the data are full_text, summary, keywords, publish_date, authors and url (6 in total). Leaf_label and root_label are the dependent variables or classes. Unnamed: 0 is an index variable.



The frequency plot for total number of alpha-numeric characters suggest that it follows a long right-skewed distribution. Most of the documents consist of less than 5000 alpha-numeric characters. Few outlier documents have more than 15000 characters as well.



The frequency plot for leaf_label suggests that the leaf_label classes are balanced. The data is equally distributed among all the classes.



The frequency plot for root_label suggests that the binary root_label classes are balanced. The data is equally distributed among the classes - sports and climate with each.

Question 2

There are 1657 training samples and 415 test samples with a 80-20 split between training and hold-out (test) set.

Question 3

In the process of stemming, it just removes the affixes from the word in attempt of finding the root of the word, the root might not be a word on itself. Stemming may be beneficial when faster computation is important and meaning of the word is not required. Whereas in lemmatization, it identifies the ”lemma” or the ”base” form of the word. Lemmatization utilize some complexities of english language such as parts of speech in order to return better root words. Hence, lemmatization is much more accurate than stemming, however stemming is faster in computation. A downside for lemmatization is that it might end up giving a larger feature set as it parses the text according to context/meaning compared to stemming.

The dictionary size for the lemmatization process is 10,614 where as the dictionary size for stemming is 9,351.

Increasing minimum document frequency i.e, minimum number of documents in which the word should appear reduces the size of the vocabulary by removing words that are not descriptive of the set of documents as they occur in very few documents. It should also reduce some high idf (inverse document frequency) values associated with lower minimum document frequency as min_df is increased.

Punctuations and numbers can be removed from the text before lemmatizing since lemmatization only operates on dictionary words. It is a good practice to remove the stopwords before lemmatizing, since lemmatization may result in the base form of a word which may not be present in the stopwords list. For example, ”is”, ”am” and ”are” would become ”be” after lemmatization but now ”be” might not be present in the stopwords list. Another example is ’doe’ from ’does’. Further after removing the stopwords, there would be lesser words to be lemmatized, making it computationally faster.

The TF-IDF-processed train matrix is of the shape $(1657, 10614)$ and the TF-IDF-processed test matrix is of the shape $(415, 10614)$. Both matrices have same number of columns as should be the case.

Question 4

The explained variance ratio plot:

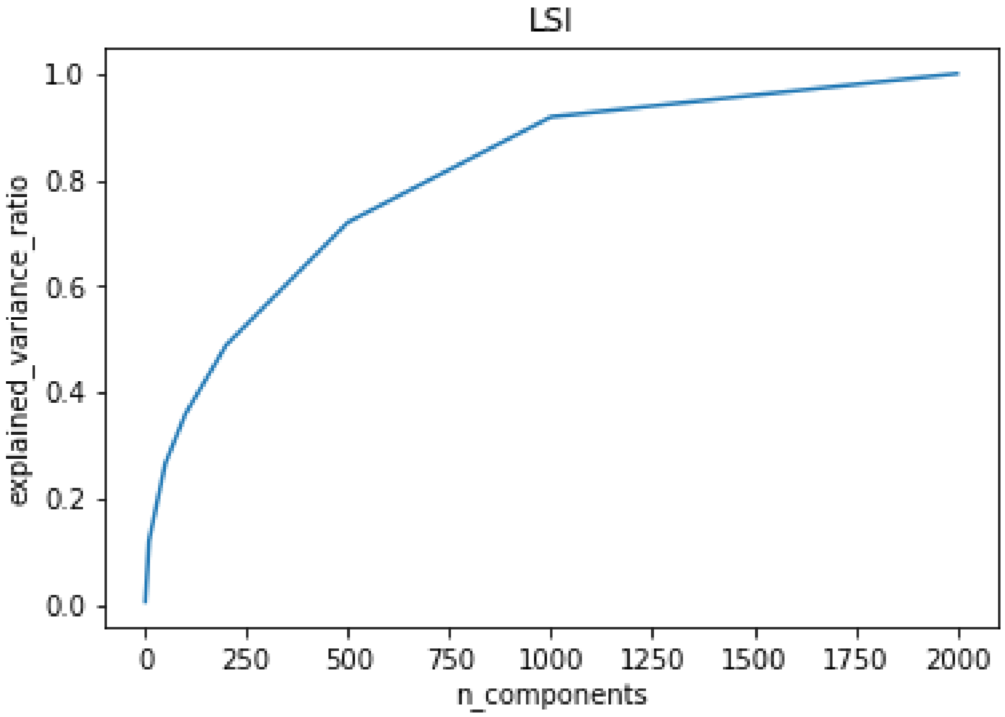


Figure 1: The explained variance ratio plot across multiple different k = [1, 10, 50, 100, 200, 500, 1000, 2000]

The explained variance ratio plot is a concave curve. The plot's concavity suggests that as we increase dimensions, the explained variance ratio first increases quickly and then the rate of increase in explained variance ratio decreases with increasing number of dimensions (diminishing returns with increasing dimensions).

Reconstruction MSE (mean squared error) on train set for NMF is : 6.88544602260256e-05
Reconstruction MSE (mean squared error) on train set for SVD is : 6.766065165465466e-05

Reconstruction MSE (mean squared error) on test set for NMF is : 6.964009639149524e-05
Reconstruction MSE (mean squared error) on test set for SVD is : 6.887313300219922e-05

The loss of NMF is larger. Both NMF and SVD represent a set of vectors in a given basis. The basis in NMF is composed of vectors with positive elements only while the basis in SVD can have both positive or negative values. The difference then is that NMF reconstructs each vector as a positive summation of the basis vectors, in other words you take a little (some positive weight) of each vector in the basis to reconstruct your data. In SVD the data is modeled as a linear combination of the basis you can add or subtract vectors as needed. Because of the added restriction it becomes harder for NMF to reconstruct the matrix and hence has a larger loss.

Question 5

Train two linear SVMs

(a) Plot the ROC curve between hard margin SVM and soft margen SVM

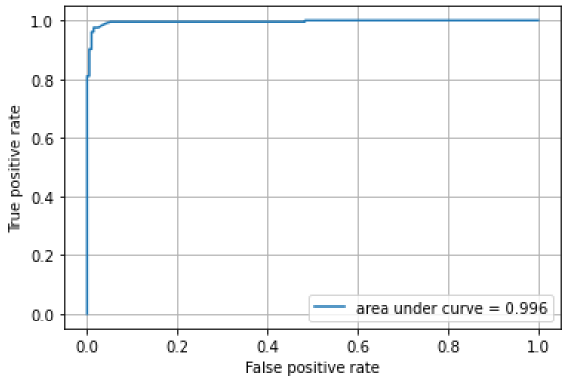


Figure 2: ROC curve of hard margin SVM C=1000

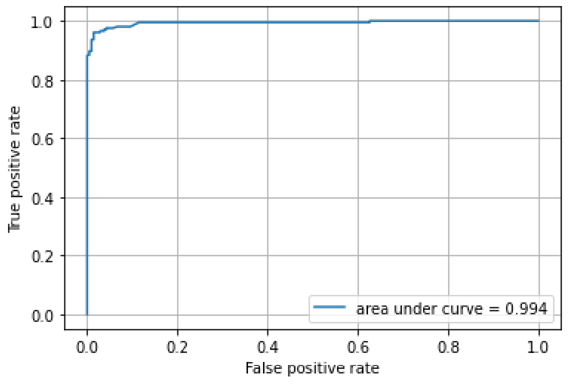


Figure 3: ROC curve of soft margin SVM C=0.0001

Confusion matrix for hard SVM on test set is:

$$\begin{bmatrix} 199 & 7 \\ 3 & 206 \end{bmatrix}$$

Confusion matrix for soft SVM on test set is:

$$\begin{bmatrix} 206 & 0 \\ 209 & 0 \end{bmatrix}$$

Performance metrics for hard SVM on test set is:
Precision is: 0.9671361502347418
Recall is: 0.9856459330143541
Accuracy is: 0.9759036144578314
F1 score is: 0.976303317535545

Performance metrics for soft SVM on test set is:
Precision is: 0.0
Recall is: 0.0
Accuracy is: 0.4963855421686747
F1 score is: 0.0

Reporting metrics for $\gamma = 100000$:

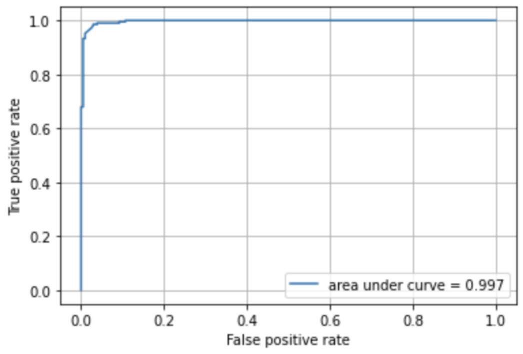


Figure 4: ROC curve of very high margin SVM C=100000

Confusion matrix for very high γ SVM on test set is:

$$\begin{bmatrix} 199 & 7 \\ 3 & 206 \end{bmatrix}$$

Performance metrics for this SVM on test set is:
Precision is: 0.9671361502347418
Recall is: 0.9856459330143541
Accuracy is: 0.9759036144578314
F1 score is: 0.976303317535545

From ROC plots, you can basically tell the different between soft and hard margin plot are small, all most close. But from the confusion matrix, accuracy, recall, precision and F-1 score. It clearly shows that hard SVM is better than soft SVM.

This is because, in the soft margin SVM, we set $\gamma = 0.0001$. Since for SVM classifier, we used numerical computation to find an optimal solution, and the way of finding it is a linear optimization, thus it will guarantee to find out the optimal point. However, once we set $\gamma = 0.0001$, for the soft SVM, the gradient around the optimal point would be small and therefore, the numerical method may be terminated before finding the optimal solution, and instead, it will find a very close decision boundary. Low gamma value leads to under-fitting whereas very high gamma value starts to over-fit the data.

Hence, even though the ROC curve looks competitive, from the confusion matrix we can see that all test records are classified in a single class in case of soft margin SVM.

(b) Use cross-validation to choose γ

In this question, we used mean cross validation accuracy to evaluate the performance of the SVM classifier. Since, both the classes are equally important to be identified, accuracy is a good performance metric as it weighs type 1 error and type 2 error equally. The range of γ is: $\gamma \in 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000, 1000000$. The result of this could be found in the Table .

γ	Accuracy
0.001	0.4852200342154115
0.01	0.4852200342154115
0.1	0.9402358679430713
1	0.9468823936228297
10	0.9559312779820187
100	0.9547300986423034
1000	0.9511138208422816
10000	0.9462836239216685
100000	0.9480981327121174
1000000	0.9480981327121174

Table 1: Mean Accuracy score on CV set for different γ

Thus, the best γ for SVM in this case is 10.

(c) Evaluation on best $\gamma = 10$

ROC plot for best $\gamma = 10$:

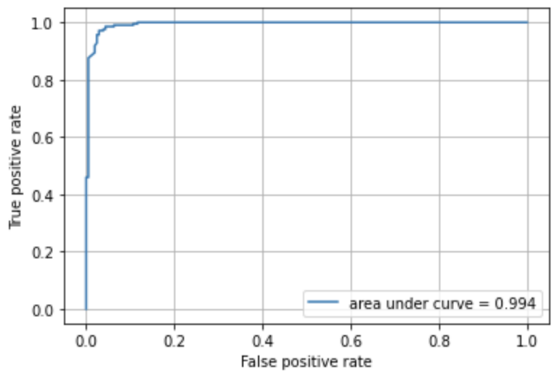


Figure 5: ROC curve of optimal γ SVM, C=10

Confusion matrix for optimal SVM on test set is:

$$\begin{bmatrix} 199 & 7 \\ 3 & 206 \end{bmatrix}$$

Performance metrics for optimal SVM on test set is:
Precision is: 0.9671361502347418
Recall is: 0.9856459330143541
Accuracy is: 0.9759036144578314
F1 score is: 0.976303317535545

Question 6

(a) Logistic regression without regularisation

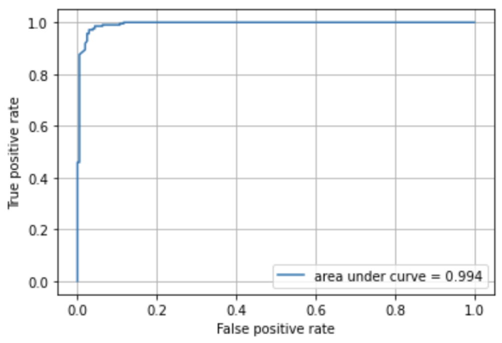


Figure 6: ROC curve of Logistic regression without regularisation

Confusion matrix for Logistic regression without regularisation on test set is:

$$\begin{bmatrix} 199 & 7 \\ 3 & 206 \end{bmatrix}$$

Performance metrics for Logistic regression without regularisation on test set is:
Precision is: 0.9671361502347418
Recall is: 0.9856459330143541
Accuracy is: 0.9759036144578314
F1 score is: 0.976303317535545

(b) Logistic regression with regularisation

Cross-validation to find optimal shrinkage parameter:

regularisation_strength	Accuracy(L1)	Accuracy(L2)
0.0001	0.48401157500091	0.4852200342154115
0.001	0.5147799657845885	0.7243075019109672
0.01	0.4852200342154115	0.9154970334510246
0.1	0.9076438685254613	0.9420503767335202
1	0.9547246387362136	0.9559312779820187
10	0.962574163724384	0.9583463764423252
100	0.9547300986423032	0.9613711644159719
1000	0.9547319186109998	0.9559349179194117
10000	0.9547319186109998	0.9541276890037491

Table 2: Mean Accuracy score on CV set for different γ

Optimal regularisation strength for Logistic regression with L1 regularisation is 10
Optimal regularisation strength for Logistic regression with L2 regularisation is 100

Comparison of performance between logistic regression with no regularisation, L1 regularisation and L2 regularisation:

model	Precision	Recall	Accuracy	F1 Score
No regularisation	0.9671361502347418	0.9856459330143541	0.9759036144578314	0.976303317535545
L1 regularisation	0.9716981132075472	0.9856459330143541	0.9783132530120482	0.9786223277909739
L2 regularisation	0.976303317535545	0.9856459330143541	0.980722891566265	0.9809523809523809

Table 3: Performance across regularisation

The performance improves as we add regularisation. As we go from no regularisation to L1 regularisation to L2 regularisation, higher weights start getting penalised according to the corresponding norm ans shrinkage parameter. By putting a penalty for high weights, regularisation reduces over-fitting in the data and makes the model more generalizable. The L1 norm penalises the sum of absolute values of the weight where as the L2 norm penalises the squared sum of weights. On increasing the value of the shrinkage parameter, penalty increases, hence the value of learnt coefficients decrease. This results in reduced over-fitting to training set and the accuracy of the model improves.

In case of L2 regularization, it would shrink the learnt coefficients for least important predictors, very close to zero, but it would never make them exactly zero, where as in L1 regularisation some of the learnt coefficients become zero due to shrinkage. So the final model would include all predictors in L2 regularisation but not in L1 regularisation. In the case of the L1 regularization, the shrinkage penalty has the effect of forcing some of the learnt coefficient estimates to be exactly equal to zero when the shrinkage is sufficiently large. Therefore, the L1 regularisation performs variable selection and is said to yield sparse models. L1 regularization can be helpful in features selection by eradicating the unimportant features, whereas, L2 regularization is not recommended for feature selection.

L2 has a solution in closed form as it's a square of a weight, on the other side, L1 doesn't have a closed form solution since it includes an absolute value and it is a non-differentiable function. Due to this reason, L1 regularization is relatively more expensive in computation. (source)

Linear SVM finds the best margin, from the distance between the line or hyperplane and the support vectors, that separates the classes. Hence this reduces risks of error. Whereas in case of logistic regression, it has different decision boundaries with different weights that are near the optimal point. So linear SVM is more geometrically motivated whereas logistic regression is more statistically motivated.

Question 7

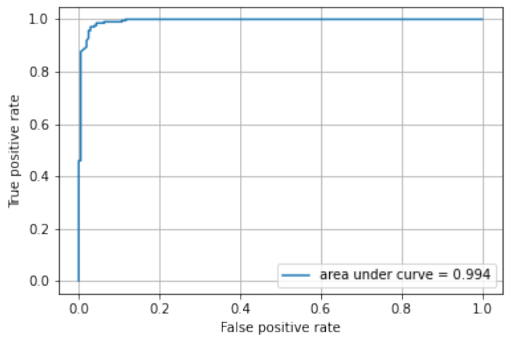


Figure 7: ROC curve of Gaussian Naive Bayes

Confusion matrix for Naive Bayes:

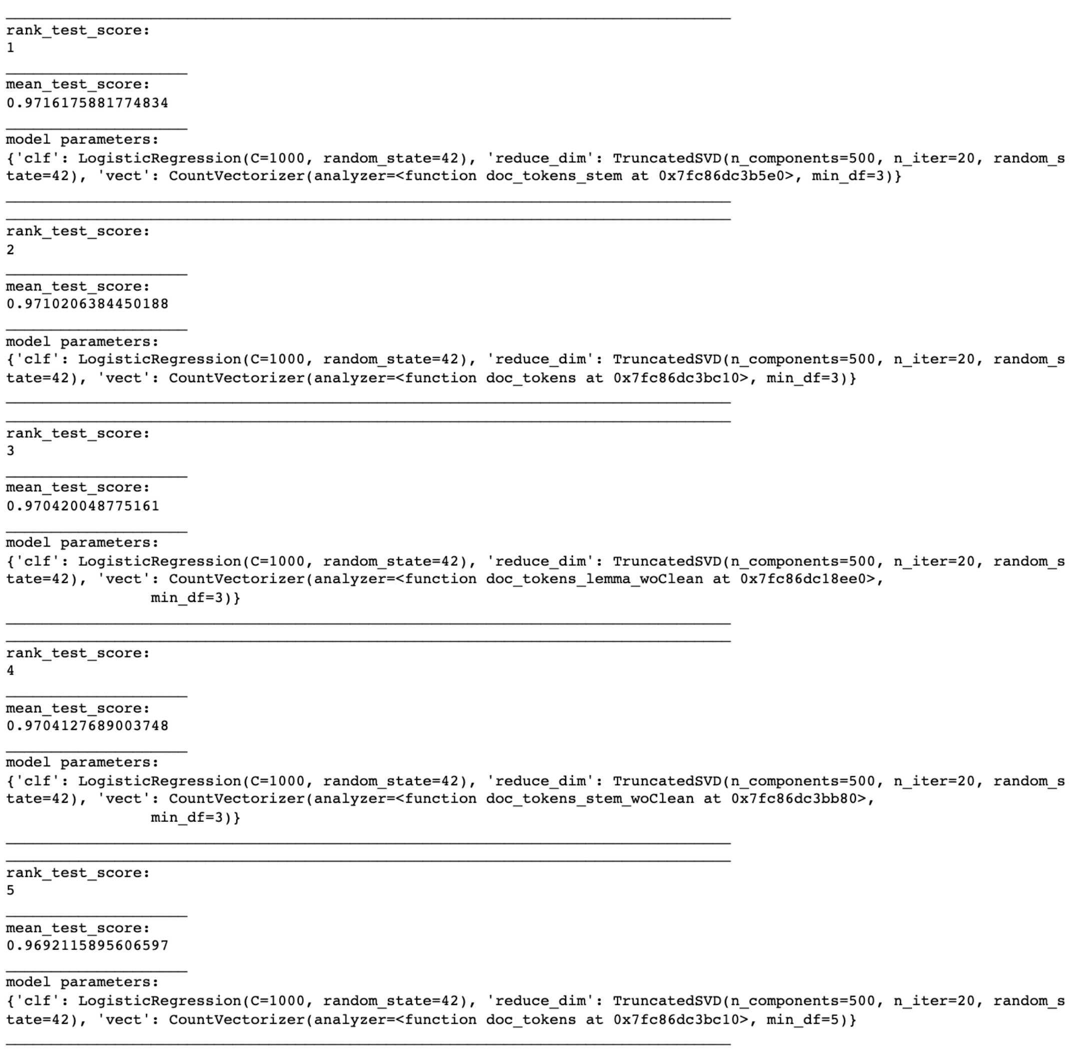
$$\begin{bmatrix} 197 & 9 \\ 13 & 196 \end{bmatrix}$$

Performance metrics for Naive Bayes on test set is:
Precision is: 0.9560975609756097
Recall is: 0.937799043062201
Accuracy is: 0.946987951807229
F1 score is: 0.9468599033816425

On the given dataset, naive bayes performs worse compared to SVM and logistic regression.

Question 8

We used the mean accuracy on the validation set with 5 fold cross validation to quantify the performance of various models and the top 5 best models are shown in the Fig. 8



```
________________________________________________________________________________
rank_test_score:
1
____________________
mean_test_score:
0.9716175881774834
____________________
model parameters:
{'clf': LogisticRegression(C=1000, random_state=42), 'reduce_dim': TruncatedSVD(n_components=500, n_iter=20, random_s
tate=42), 'vect': CountVectorizer(analyzer=<function doc_tokens_stem at 0x7fc86dc3b5e0>, min_df=3)}
________________________________________________________________________________
________________________________________________________________________________
rank_test_score:
2
____________________
mean_test_score:
0.9710206384450188
____________________
model parameters:
{'clf': LogisticRegression(C=1000, random_state=42), 'reduce_dim': TruncatedSVD(n_components=500, n_iter=20, random_s
tate=42), 'vect': CountVectorizer(analyzer=<function doc_tokens at 0x7fc86dc3bc10>, min_df=3)}
________________________________________________________________________________
________________________________________________________________________________
rank_test_score:
3
____________________
mean_test_score:
0.970420048775161
____________________
model parameters:
{'clf': LogisticRegression(C=1000, random_state=42), 'reduce_dim': TruncatedSVD(n_components=500, n_iter=20, random_s
tate=42), 'vect': CountVectorizer(analyzer=<function doc_tokens_lemma_woClean at 0x7fc86dc18ee0>,
                min_df=3)}
________________________________________________________________________________
________________________________________________________________________________
rank_test_score:
4
____________________
mean_test_score:
0.9704127689003748
____________________
model parameters:
{'clf': LogisticRegression(C=1000, random_state=42), 'reduce_dim': TruncatedSVD(n_components=500, n_iter=20, random_s
tate=42), 'vect': CountVectorizer(analyzer=<function doc_tokens_stem_woClean at 0x7fc86dc3bb80>,
                min_df=3)}
________________________________________________________________________________
________________________________________________________________________________
rank_test_score:
5
____________________
mean_test_score:
0.9692115895606597
____________________
model parameters:
{'clf': LogisticRegression(C=1000, random_state=42), 'reduce_dim': TruncatedSVD(n_components=500, n_iter=20, random_s
tate=42), 'vect': CountVectorizer(analyzer=<function doc_tokens at 0x7fc86dc3bc10>, min_df=5)}
________________________________________________________________________________
```

Figure 8: 5 best combinations results for grid search

The performance on test set for the 5 top models for overall grid-search is:

Model-1: Logistic regression with SVD 500 components, Using doc cleaning with stemming and min doc frequency = 3

Confusion matrix on test set is:

$$\begin{bmatrix} 199 & 7 \\ 1 & 208 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9674418604651163
Recall is: 0.9952153110047847
Accuracy is: 0.980722891566265
F1 score is: 0.981132075471698

Model-2: Logistic regression with SVD 500 components, Using doc cleaning without any compression and min doc frequency = 3

Confusion matrix on test set is:

$$\begin{bmatrix} 198 & 8 \\ 2 & 207 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9627906976744186
Recall is: 0.9904306220095693
Accuracy is: 0.9759036144578314
F1 score is: 0.9764150943396227

Model-3: Logistic regression with SVD 500 components, Using lemmatization without doc cleaning and min doc frequency = 3

Confusion matrix on test set is:

$$\begin{bmatrix} 198 & 8 \\ 0 & 209 \end{bmatrix}$$

Performance metrics on test set is: Precision is: 0.9631336405529954
Recall is: 1.0
Accuracy is: 0.980722891566265
F1 score is: 0.9812206572769953

Model-4: Logistic regression with SVD 500 components, Using stemming without doc cleaning and min doc frequency = 3

Confusion matrix on test set is:

$$\begin{bmatrix} 199 & 7 \\ 0 & 209 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9675925925925926
Recall is: 1.0
Accuracy is: 0.983132530120482
F1 score is: 0.9835294117647059

Model-5: Logistic regression with SVD 500 components, Using doc cleaning without any compression and min doc frequency = 5

Confusion matrix on test set is:

$$\begin{bmatrix} 198 & 8 \\ 1 & 208 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9629629629629629
Recall is: 0.9952153110047847
Accuracy is: 0.9783132530120482
F1 score is: 0.9788235294117646

Question 9

(a) Naive Bayes multi-class classification

Confusion matrix for Naive Bayes on test set :

$$\begin{bmatrix} 44 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2 & 48 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 48 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 53 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 47 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 2 & 39 & 6 & 2 \\ 7 & 0 & 1 & 1 & 0 & 1 & 41 & 6 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 43 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.8849429560952038
Recall is: 0.8796211127964779
Accuracy is: 0.8746987951807229
F1 score is: 0.8753806083569236

(b) SVM classification with one vs one classifiers

Confusion matrix for SVM classification with one vs one classifiers on dataset:

$$\begin{bmatrix} 44 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 49 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 51 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 53 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 46 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 2 & 43 & 2 & 2 \\ 2 & 0 & 1 & 1 & 1 & 2 & 49 & 1 \\ 2 & 0 & 1 & 0 & 2 & 1 & 0 & 41 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9114274627953882
Recall is: 0.9076942826067975
Accuracy is: 0.9060240963855422
F1 score is: 0.9061690492596456

(c) SVM classification with one vs rest classifier with class balancing

Confusion matrix for SVM classification with one vs one classifiers on dataset:

$$\begin{bmatrix} 44 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 49 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 53 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 55 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 47 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 44 & 3 & 2 \\ 1 & 0 & 1 & 1 & 0 & 2 & 51 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 42 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9272661264595534
Recall is: 0.928505132334405
Accuracy is: 0.927710843373494
F1 score is: 0.9268708929843031

SVM classification with one vs rest classifier without class balancing:

$$\begin{bmatrix} 44 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 49 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 53 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 55 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 47 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 44 & 3 & 2 \\ 1 & 0 & 1 & 1 & 0 & 2 & 50 & 2 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 42 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9246694691681691
Recall is: 0.9263121498782647
Accuracy is: 0.9253012048192771
F1 score is: 0.9244431217357528

The diagonal values are large suggesting correct classification. The confusion matrix is slightly spread out in the lower-right part of the matrix, suggesting that the features or predictors for these classes - pertaining to documents with various climates are somewhat less distinguishing.

On the basis of the confusion metrics observed for various classes, we suggest to merge the last 3 classes into 1. Following are the results for one-vs-one classifier and one-vs-rest classifier on the basis of the suggested merged classes:

One vs one classifier test results:

$$\begin{bmatrix} 42 & 0 & 0 & 1 & 0 & 3 \\ 0 & 48 & 0 & 0 & 0 & 2 \\ 6 & 1 & 50 & 0 & 0 & 1 \\ 1 & 0 & 0 & 52 & 0 & 2 \\ 0 & 0 & 1 & 0 & 43 & 5 \\ 0 & 0 & 2 & 2 & 2 & 151 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.93364545476997
Recall is: 0.9199835748552111
Accuracy is: 0.9301204819277108
F1 score is: 0.9259944213728057

One vs rest classifier test results:

$$\begin{bmatrix} 44 & 0 & 0 & 1 & 0 & 1 \\ 0 & 49 & 0 & 0 & 0 & 1 \\ 4 & 1 & 53 & 0 & 0 & 0 \\ 0 & 0 & 0 & 55 & 0 & 0 \\ 0 & 0 & 1 & 0 & 47 & 1 \\ 4 & 0 & 2 & 2 & 2 & 147 \end{bmatrix}$$

Performance metrics on test set is:
Precision is: 0.9433403255201286
Recall is: 0.957634041422029
Accuracy is: 0.9518072289156626
F1 score is: 0.9496789115178382

Test accuracy in both one vs one classifier and balanced one vs rest classifier improved by 2%

The class imbalance is resolved by the sklearn library in one vs rest model by adjusting 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)). Since all the classes have approximately same number of data points in training set, one vs classifier is already balanced. We share the performance for one vs rest SVM classifier on a balanced data set as well as unbalanced dataset. The performance in this case does not change much, however the balanced class classifier performs better on test set.

Question 10

(a) Why are GLoVE embeddings trained on the ratio of co-occurrence probabilities rather than the probabilities themselves?

Ratio of the co-occurence probabilities helps determine the words in context which are relevant in distinguishing two different words and also helps determining which context word is more important for which key word. For example, let the context words be solid, gas, water and fashion being studied to understand the target words ice and steam. The probability of solid given ice (P(solid | ice)) and (P(solid | steam)) by itself does not provide much information but if we take the ratio (P(solid | ice) / (P(solid | steam)) given a corpus, we can see that solid is a relevant word in context of ice compared to steam and can help distinguish between ice and steam.

(b) 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?

Since GLoVE is a global vector representation, after being trained on a large corpus using the co-occurence matrix of words from the corpus, the vector embeddings of the word remain unique. Hence, GLoVE will return same vector for the word ”running” in the two sentences.

(c) What do you expect for the values of,

$$||GLoVE[\text{"}queen\text{"}] - GLoVE[\text{"}king\text{"}] - GLoVE[\text{"}wife\text{"}] + GLoVE[\text{"}husband\text{"}]||_2,$$
$$||GLoVE[\text{"}queen\text{"}] - GLoVE[\text{"}king\text{"}]||_2 \ and \ ||GLoVE[\text{"}wife\text{"}] - GLoVE[\text{"}husband\text{"}]||_2?$$

Compare these values.

The distance between the embeddings of queen and king compared to the distance between the embeddings of wife and husband should be of similar value as they both have same semantic relationship. The difference between the queen and king and wife and husband should be close to zero, as the semantic relationship between captured by the vectors should cancel each other out. Queen is to king as wife is to husband. Word embeddings should capture these relationships.

(d) Given a word, would you rather stem or lemmatize the word before mapping it to its GLoVE embedding?

Lemmatization would be preferred over stemming before mapping the word to its GLoVE embedding. This is because lemmatization with pos tagging is word base reduction on the basis of context of word and returns a valid word. Also, lemmatization ensures the reduced word is actually an english word with close meaning to original word. Hence, any model trained on large corpus of documents such as GLoVE embeddings would work better with lemmatization. Stemming, on the other hand might return invalid base words and lose the document context by not having a match with any GLoVE keyword.

Question 11

(a) Feature engineering process

Since each word embedding from GLoVE provides the general context representation of the word, the sentence can be represented as having the average embedding of the constituent words. Each embedding dimension is like weight on a contextual property of the word. We represent a document by the average embedding of clean set of words in the documents.

The process is first create a set of keywords representing the document and then take average across all the word embeddings.

(b) Train and evaluate a classifier on GLoVE based features

Since logistic regression have comparable performance to SVM and is more simplistic i.e, less sensitive to parameters, we train a logistic regression model. We do not use regularisation as we want to compare the difference between various embedding sizes.

Confusion matrix for Logistic regression without regularisation on test set is:
$$\begin{bmatrix} 193 & 13 \\ 1 & 208 \end{bmatrix}$$

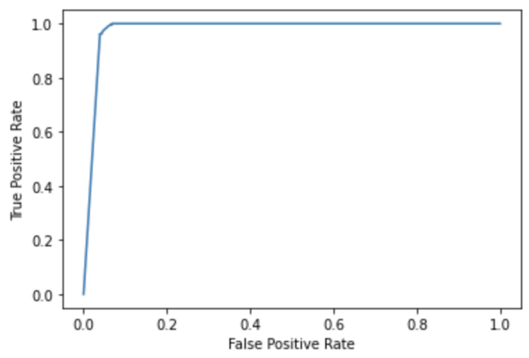


Figure 9: ROC curve of Logistic regression with no regularisation

Performance metrics for Logistic regression without regularisation on test set is:
Precision is: 0.9411764705882353
Recall is: 0.9952153110047847
Accuracy is: 0.9662650602409638
F1 score is: 0.9674418604651163

Question 12

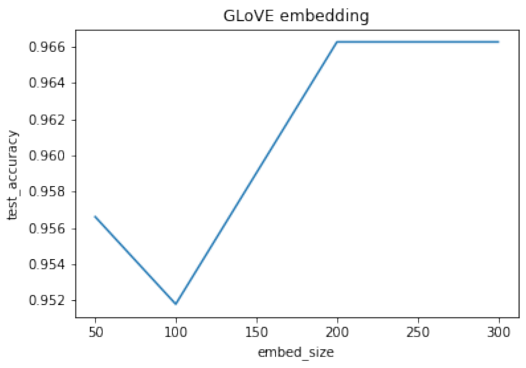


Figure 10: Plot of test accuracy with various Glove embedding sizes

From embedding_size 50 to 100, the test accuracy decreases, this is because the higher dimensional data leads to overfitting. Training accuracy improves but do not generalise well to test_accuracy due to generalisation error. For 200 and 300, both the training and test accuracies improve as the model features become more descriptive and outweighs the overfitting issues with large dimension. 200 and 300 embedding sizes have same accuracies suggesting that additional dimensions are not adding any value.

Question 13

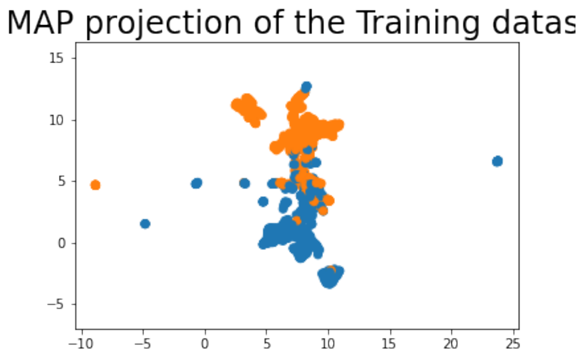


Figure 11: UMAP: Training data

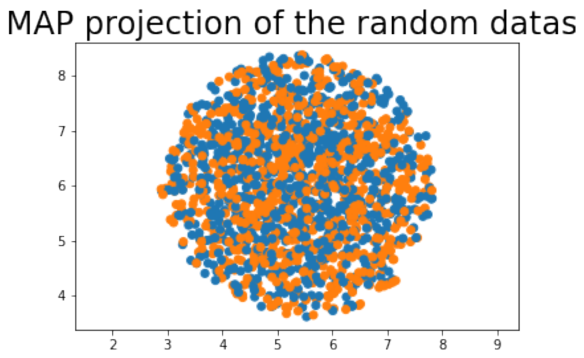


Figure 12: UMAP: Random data

We can observe some cluster in the training data_set suggesting that the lower dimension representation of the embeddings have distinct clusters for the two classes. For the random feature set, the umap projection is randomly distributed between the 2 classes.