COL774 Assignment 2

Aman Hassan
2021CS50607

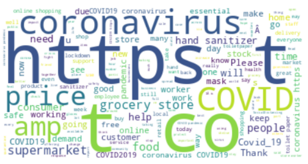
October 2023

1 Naive Bayes

(a) Here we implemented Naive Naive Bayes that uses the **Multinoulli distribution/Multinomial distribution**

i.
- Time taken for parameter finding:4.943481206893921
- Time taken for prediction:25.203924417495728
- Accuracy in Training set using naive bayes = 0.8504648214663004
- Time taken for parameter finding:5.051173686981201
- Time taken for prediction:2.271172523498535
- Accuracy in Validation set using naive bayes = 0.6705132098390525

ii. The following word clouds were obtained for each class



Positive

Neutral

Negative

(b) With Random Guessing, we should expect a 33% accuracy on average which is also what we observe (A particular run gave 33.73823261463711% accuracy). By predicting positive always we see that prediction accuracy is 43.85059216519891 because of the higher number of positive tweets over others.

The Part a) of Naive Bayes gives us 98.73987398739874% gain over Random Guessing (i.e. Naive Bayes is almost twice better than random) and gives 52.908587257617704% gain over simply predicting positive.

(c)
- The confusion matrices for training set are as follows:

NaiveBayes

	AP	ANu	AN
PP	15711	2158	1078
PNu	177	3574	171
PN	714	1364	12917

Random

	AP	ANu	AN
PP	5385	2415	4747
PNu	5595	2359	4703
PN	5622	2322	4716

All Positive

	AP	ANu	AN
PP	16602	7096	14166
PNu	0	0	0
PN	0	0	0

- The confusion matrices for validation set are as follows:

NaiveBayes

	AP	ANu	AN
PP	1121	271	272
PNu	77	174	55
PN	246	174	905

Random

	AP	ANu	AN
PP	479	206	415
PNu	468	186	225
PN	497	409	408

All Positive

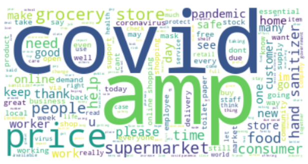
	AP	ANu	AN
PP	1444	617	1232
PNu	0	0	0
PN	0	0	0

We can see that for Naive Bayes classifier, the number of correctly predicted samples (i.e. the trace of confusion matrix) is higher than the others which indicates that Naive Bayes classifier has a higher prediction accuracy than random or positive guessing

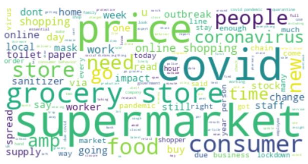
Note that on adding the values in the entire matrix we get back total number of training examples and adding up a particular column value gives us the total number of actual training examples in that class.

(d) The following transformations were done in this part:

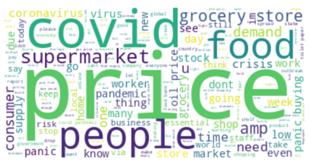
- Remove #'s and @'s (and their corresponding tags)
- Remove special chars like !, . , * etc
- Stopword removal
- Lemmatize



Positive



Neutral



Negative

The output accuracy produced was:

- Naive Bayes prediction accuracy with basic data cleaning = 0.6680838141512299
- Naive Bayes prediction accuracy with better data cleaning (stemming) = 0.7060431217734588
- Naive Bayes prediction accuracy with better data cleaning (lemmatizing) = 0.7127239599149712

Some observations to note:

- The word clouds show the frequency of the various lemma of words rather than the frequency of the words themselves.
- Accuracy increases since the data has lesser noise compared to earlier which allows for better predictions
- Lemmatizing has better accuracy over stemming

(e) Bigrams has not been implemented:

(f) Using Domain adaptation the following results were obtained:

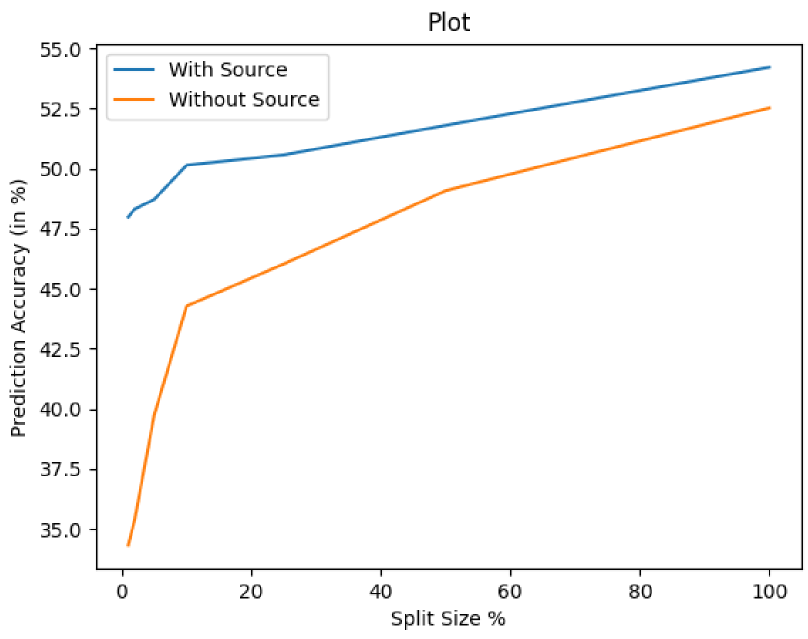
i. Using Source Data:

- The prediction accuracy when using split 1 with source domain is 47.97879390324719
- The prediction accuracy when using split 2 with source domain is 48.31013916500994
- The prediction accuracy when using split 5 with source domain is 48.70775347912525
- The prediction accuracy when using split 10 with source domain is 50.132538104705105
- The prediction accuracy when using split 25 with source domain is 50.56328694499669
- The prediction accuracy when using split 50 with source domain is 51.789264413518886
- The prediction accuracy when using split 100 with source domain is 54.208084824387015

ii. Without using Source Data:

- The prediction accuracy when using split 1 is 34.327369118621604
- The prediction accuracy when using split 2 is 35.387673956262425
- The prediction accuracy when using split 5 is 39.72829688535454
- The prediction accuracy when using split 10 is 44.26772697150431
- The prediction accuracy when using split 25 is 46.02385685884692
- The prediction accuracy when using split 50 is 49.07223326706428
- The prediction accuracy when using split 100 is 52.51822398939695

iii. The obtained graph is as follows:



iv. Observations to note:

- We see that the prediction accuracy using source data is higher than the prediction accuracy without source data
- Initially the model without source data performs considerably worse compared to the one using source data. This can possibly be attributed to the fact that the intial splits have very less data because of which intial model without souce data cannot predict accurately.
- As the split size increases the model without source data approaches the accuracy of the one with source data. This can possibly be attributed to the fact that the model without source data is more generalised and can hence predict better compared to the source data model which has been trained against Corona specific tweets.
- Both models reach just above 50% accuracy.This can possibly be attributed the fact that there are not info datapoints to train the model accurately (ie small size of general tweet training set compared to Corona training set). Possibly also due to the fact that many tweets in the general tweet training set are repeated and just subsets of other tweets.

2 Binary SVM

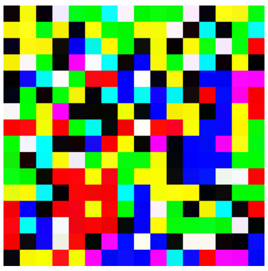
(a) i. With a threshold of $\alpha_i > 10^{-6}$ for the support vectors. We obtain the following results for a linear kernel

- No of support vectors: 673
- % of support vectors wrt training examples: 14.138655462184873%

ii. The test accuracy we obtain is 94.0%

iii. The top 6 support vectors are:

The weight image obtained is:



(b) i. With a threshold of $\alpha_i > 10^{-6}$ for the support vectors. We obtain the following results for a Gaussian Kernel

- No of support vectors: 1067
- % of support vectors wrt training examples: 22.415966386554622%
- Number of support vectors that match in both linear and gaussian: 571

ii. The test accuracy we obtain is 94.25%

iii. The top 6 support vectors are:

iv. We see that the validation accuracy obtained in Gaussian kernel is slightly higher than that of linear kernel (0.25% higher)

(c) i. Number of Support Vectors obtained is as follows

- Linear Kernel: sklearn_svm has 669 SV's over CVXOPT's 673 SV's
- Gaussian Kernel: sklearn_svm has 1056 SV's over CVXOPT's 1067 SV's

		CVXOPT	
		lin	rbf
Scikit	lin	669	569
	rbf	566	1056

ii. Comparison of weight and bias in linear kernel:

- CVXOPT: b = -4.0888424817486575
- sklearn_svm: b = -4.0881856782113015
- norm(w_cv - w_skl) = 0.015257845622513772

iii. Validation set accuracy is as follows:

- Linear Kernel: sklearn_svm obtains 94.75% accuracy over CVXOPT's 94% accuracy
- Gaussian Kernel: sklearn_svm obtains 93.75% accuracy over CVXOPT's 94.25% accuracy

iv. The training times are given below:

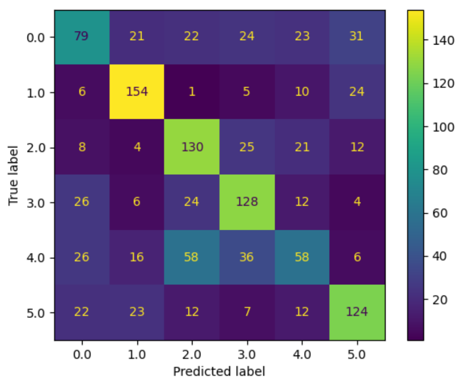
Scikit RBF	1.649s (+ 23.210s for data retrieval)
Scikit linear	1.944s (+ 22.590s for data retrieval)
CVXOPT RBF	154.257s (+ 23.080s for data retrieval)
CVXOPT linear	81.493s (+ 24.363s for data retrieval)

3 Multiclass SVM

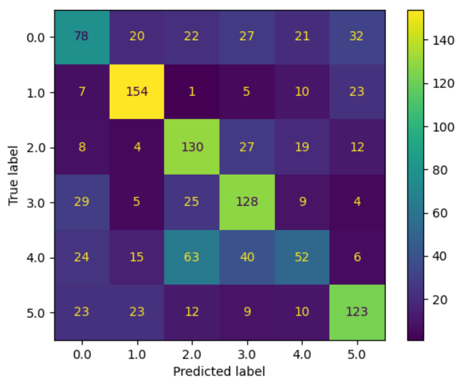
(a) Using the previously created CVXOPT Solver we obtain 55.414% accuracy in Multi-class classification

(b) Using the scikit SVM library we obtain 56.083% accuracy on the same test set. We also see that in comparison to CVXOPT, the scikit SVM library is much faster. The exact time taken to train the model is as follows 2028.495s ($\tilde{3}$3 mins) for CVXOPT implementation in comparison to 53.316s ($\tilde{1}$ min) for scikit library. Note the accuracy rate is almost similar (differing only by $\tilde{0}$.53%)

(c) The confusion matrices for both implementations is as follows:



Scikit



CVXOPT

We observe that classes 0 and 4 are the most misclassified classes with class 4 getting misclassified as class 2 or class 3 and class 0 getting misclassified as class 5. Below, we plot first 12 examples which are misclassified by Scikit and CVXOPT (both turn out to be the same, with their predicted and original labels respectively

SK: 0 5 0 5 0 4 0 3 0 5 0 2 0 3 0 4 0 3 0 3 0 3 0 5

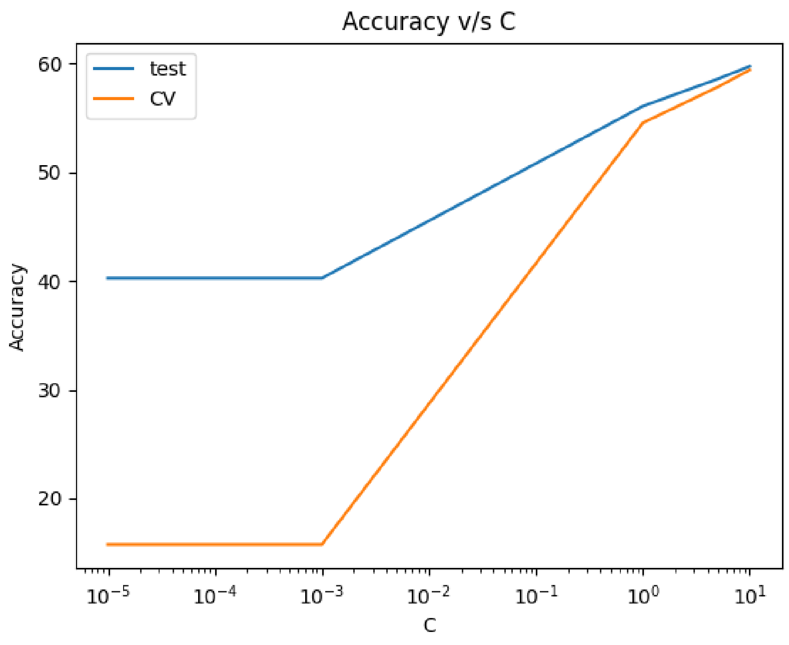
CVXOPT: 0 5 0 5 0 4 0 3 0 5 0 2 0 3 0 4 0 3 0 3 0 3 0 5

The classes of the Intel Image dataset are Building(0), Forest(1), Glacier(2), Mountain(3), Ocean(4) and Road(5). The model(s) misclassify several examples in the Building and Road categories amongst each other, which makes sense because most of the images with roads are surrounded by buildings. Similarly, Ocean seems to be misclassified into Glacier's as well, which also makes sense since the backgrounds are similar.

(d) The obtained accuracies are as follows:

- K-fold cross verification: [15.342%, 15.671%, 54.263%,56.132%,58.679%]
- Validation accuracy [41.240%, 41.368%, 53.482%,56.726%,58.629%]

(e) The obtained graph is given in the following page. The value of determines the number of misclassifications, a higher value of C would provide less misclassification but with the downside that it would tend to overfit. Smaller values of C underfit the model and causes a poor decision boundary which is exactly what we obtained as per the graph. We also see that both K-fold and validation classify better the higher the value of C.


Accuracy v/s C
test
CV
Accuracy
C
60
50
40
30
20
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}