COL 774 Assignment 2 (Semester I, 2021-2022)

1) Text Classification -

a) Naive Bayes-

Made a class to implement the Naive Bayes algorithm from scratch, there is also a method named *make_term_dict()* which makes the term dictionary class wise for the complete dataset.

With alpha = 1.0
Output --
Time Taken to make Term Dictionary: 03:35 (MM:SS)
Length of vocab – 90061

Accuracy: 68.17 % ------ On Training Data
Accuracy: 61.82857142857143 % ------- On Testing Data

b) Random and Majority Predictions-

Using Random Predictions and Majority predictions, although it looks like we are getting better prediction with majority prediction, but it only predicts one class and no other, so it’s hardly of any relevence to us , and highly depends on the data provided.

Output --

For Random prediction
Accuracy: 20.032

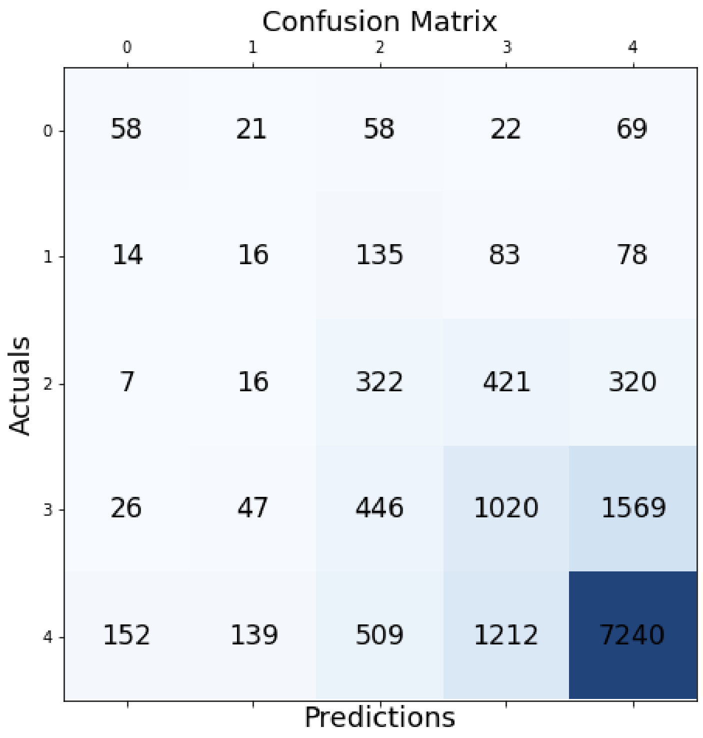
For Majority Prediction
Most Occurring label: 5.0
Accuracy: 66.08571428571429

c)Making the confusion matrix with the result got in part a --

Looking at the confusion matrix we can see the because significant amount of reviews being 5.0 most or the data is predicted near it, so taking a lower values of alpha might.

Output --

Accuracy: 61.82857142857143 %



Confusion Matrix

Actuals \ Predictions	0	1	2	3	4
0	58	21	58	22	69
1	14	16	135	83	78
2	7	16	322	421	320
3	26	47	446	1020	1569
4	152	139	509	1212	7240

d) Preprocessing part --

- Tokenized the words using nktl *word_tokenizer()*
- Made all words in lower case
- Removed numbers
- Here I removed the stopwords
- Removed punctuations
- Performed stemming

Output --
Time Taken to make Term Dictionary: 0:05:18.200181

Accuracy: 69.168 % -- (On Training Data)
Accuracy: 65.53571428571429 % -- (On Testing Data)

As we can see that although the processing time increased but there is some improvement in accuracy even with *alpha =1*

e) Feature Engineering --

Bigrams – Using bigrams gives us a lot better accuracy of about – 65.65%

But still the there is hardly any difference (0.1%) for this model

They took – 0:48:55.865315(MM:SS) just to train over the entire dataset
Accuracy: 84.618 %
F1 Score for classes:
1.0 : 0.48125184529081777
2.0 : 0.317733205985355
3.0 : 0.7584354180098861
4.0 : 0.8641737390208651
5.0 : 0.9026109158316283
Macro F1 Score: 0.6648410248277105
Accuracy: 65.65714285714286 %
F1 Score for classes:
1.0 : 0.061302681992337155
2.0 : 0.011594202898550725
3.0 : 0.04765687053216838
4.0 : 0.30450987700335447
5.0 : 0.8026385478357166
Macro F1 Score: 0.24554043605242545

Birgams with lots of preprocessing – Here I used

- Bigrams
- Removed punctuations
- Performed stemming
- Made all words in lower case
- Removed numbers
- Removed the stopwords

Alfter all this the the Train accuracy is still great but the test accuracy decreases and n
no significant change, so bigrams don't seem to be a great choice after all because of heavy computation and not great accuracy.
But the runtime was more than 2hrs just to make term dictionary for complete dataset
Accuracy – 59.3% (ran on 10,000 files)

f) F1 score and Best Model --

The best model was the processed model in part (d) and it makes sense because the stopwords and all the irrelevant things are removed and we use it to predict the overall ratings.
Macro F1 Score – 0.261 for part(d) model
accuracy – 61.1%

g) Using Summary field -

Using the summary completely instead of whole reviews seems to increase the accuracy significantly, this is rather afaster and more effictive approach to use the summary field as the summary is also sufficient to predict the overall ratings.

We can mostly reading few lines predict the overall ratings and same goes for naive bayes predictions as well.

Accuracy = 69.4%

2) MNIST Digit Classification-

This section is in Q2.py file.

Preprocessing – divided the X with 255 to make data values <=1

a, i)Binary classification Using Linear Kernel--

A python class for binary classification using SVM where the dual objective is trained using *CVXOPT* , The weights are converted in the required format using the method *make_parameters()* which is a common method and is used for Gaussian as well with just 1 if condition.

D = 5
so using data with label 5,6

Output -
Linear SVM:
Time to learn: 0:01:17.973444
nSV: 232 With threshold: 0.00099999999989637
Primal objective: -53.67457701101186
Prediction:
Training data Accuracy: 99.75%
Testing data Accuracy: 96.75%

So we can clearly see SVM can very easily classify binary data.

a,ii) Binary Classification using Gaussian Kernel -

Using gaussian kernel we cant directly store weights so we are calculating this in the prediction

Gaussian SVM:

Time to learn: 0:02:46.42 (HH:MM:SS)
nSV: 1480 With threshold: 0.0009999999978204917
Primal objective: -227.04500162471632
Prediction:
Training data Accuracy: 100.0
Test data Accuracy: 99.1891891891892

So we can clearly see the increased accuracy using gaussian kernels.

a,iii) Binary Classification using Libsvm-

Using Libsvm – There is significant increase in the performance as the learning and predictions are way faster, also the accuracy is improved slightly (although it seems like there is no room) compared to linear kernel. Although the Gaussian are almost same except minor difference in support vector count.

Libsvm:

Linear

```
Time to learn: 0:00:00.906429
optimization finished, #iter = 9820
nu = 0.022865
obj = -53.674532, rho = 1.624403
nSV = 233, nBSV = 23
Total nSV = 233

predictions on train and test
Accuracy = 99.9% (3996/4000) (classification)
Accuracy = 97.2973% (1800/1850) (classification)
```

Gaussian

```
Time to learn: 0:00:04.174591
optimization finished, #iter = 2350
nu = 0.110834
obj = -227.045015, rho = 0.140648
nSV = 1478, nBSV = 42
Total nSV = 1478

predictions on train and test
Accuracy = 100% (4000/4000) (classification)
Accuracy = 99.1892% (1835/1850) (classification)
```

b,i) One Vs One Classifier for *CVXOPT* --

This takes way lot time (Learning – 2hrs, prediction ~3hrs)
Here prediction is taking way to much because of non parallel computation which is the in libsvm(but not here).

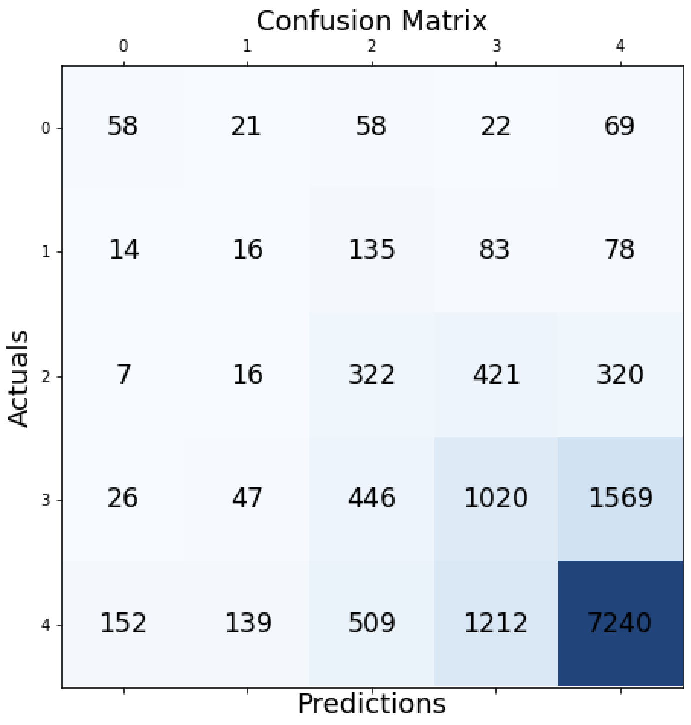
Training set accuracy – 99.4%
Testing set accuracy – 96.5%

b,ii) One Vs One Classifier using Libsvm -

This seems way faster implementation as the training took way lesser time and the accuracy is rather imroved a bit.

```
One vs One Classifier for livsvm
Learning models...
Time to learn: 0:03:00.79 (HH:MM:SS)
Time to predict: 0:07:30.169375
Accuracy: 97.27
```

b,iii) Confusion matrix for One vs One classifier is given below --



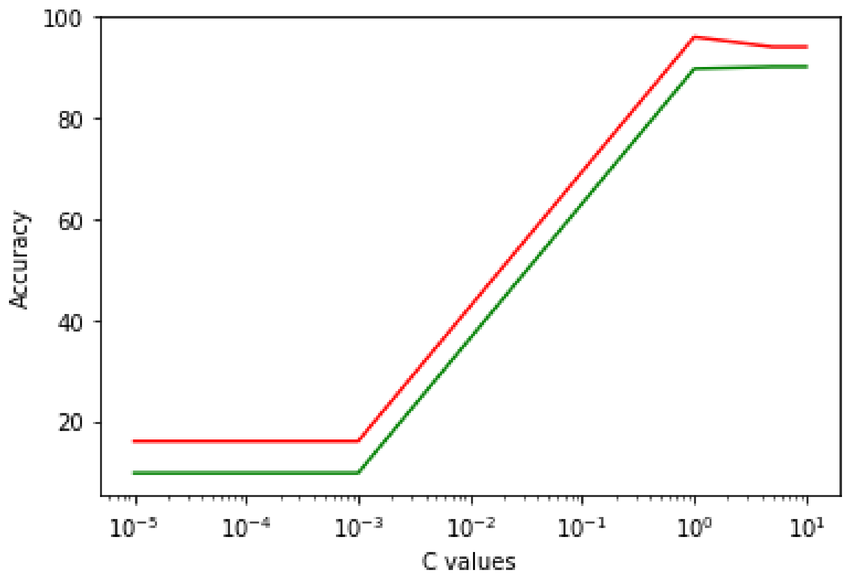
Confusion Matrix

Actuals \ Predictions	0	1	2	3	4
0	58	21	58	22	69
1	14	16	135	83	78
2	7	16	322	421	320
3	26	47	446	1020	1569
4	152	139	509	1212	7240

b,iv) Validation set --

Using K fold cross validation method we found that c = 5 gives the best result for our model.

C = 10**-5 and 10**-3 are very bad and gives poor accuracy
This complete plot this as follows



100
80
60
40
20
Accuracy
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
C values