REPORT
AnshPrakash
2016CS10367

NaiveBayes:

Part(a)

Confusion matrix
[[59508 12395   4775   2161   1452]
 [11314 14832 10870   5199   1328]
 [5739   4533 24809 19928   3704]
 [4500   1900   5169 80183 25842]
 [11230   804   1858 54668 166171]]

F1 array

[0.68962001 0.38027357 0.4672392 0.57328238 0.76712955]
F1 macro 0.5755089408065754

Training Set accuracy 64.59545461344023

Confusion matrix
[[14630 3369 1213   565   392]
 [2947 2919 3241 1321   410]
 [1430 1439 4729 5969   964]
 [1110   550 2060 18382 7256]
 [3091   257   450 14929 40095]]
F1 array
[0.67455103 0.30136279 0.36066199 0.52129771 0.74291961]
F1 macro 0.5201586272558972

T**est Set accuracy** 60.39201902511255
Time taken by the code is 189.27828645706177

Part(b)

Confusion matrix by random prediction
[[16121 16031 15876 16106 16157]
 [8896 8763 8672 8542 8670]
 [11632 11794 11756 11920 11611]
 [23246 23434 23589 23635 23690]
 [47071 46790 47102 46921 46847]]
accuracy by random prediction 0.20027595387307617
F1 score by random prediction policy
F1 macro 0.1830360115455228
F1 array
[0.17218048 0.11656413 0.14188814 0.21035253 0.27419478]

Confusion matrix by max repeated element policy
[[0 0 0 0 80291]
[0 0 0 0 43543]
[0 0 0 0 58713]
[0 0 0 0 117594]
[0 0 0 0 234731]]
Accuracy by max_prediction policy 0.43885452968186783
F1 score by max prediction policy
F1 macro 0.12200108367560938
F1 array
[0. 0. 0. 0. 0.61000542]

43% accuracy by max prediction cretia .
If I randomly predict classes I get 20% accuracy

My algorithms accury is 64% on training data which is 44% more than random prediction and 20% more than max cretian.

Part(c)
Confusion matrix for test set
[[14630 3369 1213 565 392]
[2947 2919 3241 1321 410]
[1430 1439 4729 5969 964]
[1110 550 2060 18382 7256]
[3091 257 450 14929 40095]]
5 star is the category which maximum diagonal entries
It means Training data have lot of instances of 5 star
Observations:
Classes which are close by gets mis predicted into other class nearest to it.

Part(d)
Time Taken for loading data 13.486944437026978
Stemming Start
Stemming Done
Time taken for Stemming 1359.182552576065
[80291.0, 43543.0, 58713.0, 117594.0, 234731.0]
[[59552 12171 4381 2378 1809]
[11859 14061 10107 5703 1813]
[6270 4002 23248 20315 4878]
[4965 1534 4916 77445 28734]
[11515 706 1705 52581 168224]]
Training Set accuracy 64.03962069429695
[[14664 3245 1114 624 522]
[3104 2748 2966 1478 542]
[1550 1311 4469 5954 1247]
[1216 482 1951 17809 7900]
[3130 244 484 14395 40569]]
Test Set accuracy 60.02108915777981

The accuracy decreases by a amount on the test data.

Observation:
Stemming does not effect the prediction much.

Part(e)
bi-grams
Time Taken for loading data 7.9173266887664795
Bigramming Started Start

Bigramming Done
Time taken for Bigraming 61.18964767456055

[80291.0, 43543.0, 58713.0, 117594.0, 234731.0]
[0.15011255029240642, 0.08140826216365785, 0.10977018800759808, 0.21985446985446985, 0.43885452968186783]

[[74183 993 1758 2598 759]
 [11198 21131 2293 7515 1406]
 [4925 101 34327 15542 3818]
 [2233 25 98 95751 19487]
 [2205 54 260 12130 220082]]

Training Set accuracy 83.28609461702987

[[16846 1063 799 1014 447]
 [4224 968 2160 2998 488]
 [1743 285 2163 9123 1217]
 [802 69 478 18411 9598]
 [1316 68 195 10088 47155]]

Test Set accuracy 63.97268879283268

Using bigrams accuarcy over Test set increased by almost 3%.
Observation:
Two words together can carry some meaning i.e context and this can help the model to improve.

Lemmitization:
Lemmatizer Started Start
Lemmatizer Done
Time taken by Lemmatizer 232.10045170783997
[80291.0, 43543.0, 58713.0, 117594.0, 234731.0]
[0.15011255029240642, 0.08140826216365785, 0.10977018800759808, 0.21985446985446985, 0.43885452968186783]
[[59388 12419 4820 2194 1470]
 [11461 14423 11052 5281 1326]
 [5863 4484 24438 20138 3790]
 [4604 1851 5296 79691 26152]
 [11394 790 1846 54778 165923]]
F1 array

[0.68656251 0.37215843 0.46037771 0.56988086 0.7656948]
F1 macro 0.5709348621816492
Training Set accuracy 0.6428883919891114
[[14653 3334 1225   561   396]
 [2974 2879 3246 1326   413]
 [1434 1433 4692 5973   999]
 [1126   555 2060 18328 7289]
 [3140   263   443 14952 40024]]
F1 array
[0.6737631 0.29831106 0.35820896 0.51995801 0.74157657]
F1 macro 0.5183635403080867
Test Set accuracy 0.60258155222184

Observation
Supprisingly Lemmitization didn't help that much.

Part(f)
For the best performing mertic i.e,simple Naive Bayes 1(a) my case is
F1 array
[0.67455103 0.30136279 0.36066199 0.52129771 0.74291961]
F1 macro 0.5201586272558972
(g)
Ran on Training Set Time is clock time here
Prediction done on test.json
Confusion matrix
[[14381 3867 1180   404   337]
 [2669 3634 3265   938   332]
 [1311 1578 5977 4822   843]
 [1089   743 2297 18609 6620]
 [3077   354   538 14848 40005]]

F1 array
[0.67364624 0.34586466 0.43018569 0.53955552 0.74804364]
F1 macro 0.547459151263009
Test Set accuracy 0.6177627544534019
Time taken by the code is 867.2274990081787s

Observation
Memory needs to be managed properly(haha)
The Prediction even though increased by 1% but it is not a significant improvement on test set as if the model have reached its asymtotic limit for training accuracy.

SVM
1
(a)
Confusion Matrix for Linear Kernel on Training Set
[[2000   0]
 [0 2000]]
Training accuracy for Linear Kernel 100.0

test set accuracy with Linear Kernel: 98.45154845154845

Confusion Matrix for Linear Kernel on Test data
[[959 15]
 [16 1012]]

Linear Kernel No. of support Vectors =152

bias=1.72(Linear Kernel)

1(b)

Confusion Matrix for Gaussian Kernel on Training Set
[[2000   0]
 [0 2000]]
Training set accuracy by Gaussian Kernel 100.0

Confusion Matrix for Gaussian Kernel on Test set
[[970   4]
 [10 1018]]
Accuracy of the set : 99.3006993006993
Test set accuracy with GaussianKernel: 99.3006993006993

Gaussian Kernel No. Of Support Vectors =1337
bias_g=0.37(Gaussian Kernel)

Both Linear and Gaussian Kernel are performing well on test and training set and Gaussian kernel give little bit improvement over Linear Kernel on Test Set

1(c)
LibSVm

```
Training accuracy using Linear Kernel 100.0%
Confusion matrix for Linear Kernel
[[2000    0]
 [   0 2000]]
nSV  = 159(LibSVM) nSV= 152(cvxopt)
bias = 1.79(Linear LibSVM)
bias = 1.72(cvxopt )
w: too big too show here

Test Set accuracy using Linear Kernel 98.45154845154845
Confusion matrix for Linear Kernel
[[ 959   15]
 [  16 1012]]
training time : 12.32s
faster than CVXopt training time

Gaussian Kernel SVM
Total nSV = 1323
Bias term for gaussian kernel is  0.37992479567147613
weight vector for b is infinity dimension vector,So it canno be explicitly
calculated
```

```
Accuracy = 100% (4000/4000) (classification)
Training Accuracy using Gassian Kerenl  100.0
Confusion MAtrix for Gaussian Kernel
[[2000    0]
 [   0 2000]]

Accuracy = 99.2507% (1987/2002) (classification)
Test Accuracy using Gassian Kerenl  99.25074925074925
Confusion Matrix for Gaussian Kernel
[[ 970    4]
 [  11 1017]]
training time :15.123s
faster than CVXopt training time
```

2

(a)

```
Training set Accuracy : 0.99835
[[2000 0 0 0 0 0 0 0 0 0]
[ 0 1998 0 0 1 0 0 0 0 1]
[ 0 1 1996 0 2 0 0 0 0 1]
[ 0 0 0 1996 0 0 0 1 0 3]
[ 0 2 0 0 1997 0 0 0 0 1]
[ 0 0 0 0 5 1995 0 0 0 0]
[ 0 0 0 0 1 0 1999 0 0 0]
[ 0 3 1 0 2 0 0 1993 0 1]
[ 0 1 0 0 0 0 0 0 1999 0]
[ 0 0 0 0 5 0 0 1 0 1994]]
```

```
Test Set Accuracy: 0.9492
[[ 970 0 0 0 1 0 5 1 3 0]
 [ 0 1126 2 1 1 0 3 0 1 1]
 [ 8 1 937 7 43 0 3 7 24 2]
 [ 1 0 4 976 6 3 0 7 8 5]
 [ 0 0 0 0 976 0 2 0 1 3]
 [ 6 0 0 6 27 721 13 1 97 21]
 [ 5 4 0 0 19 0 927 0 3 0]
 [ 1 9 7 1 17 0 0 973 5 15]
 [ 5 0 1 11 7 0 0 2 943 5]
 [ 5 5 0 4 39 0 1 4 8 943]]
Time taken : 5122.4583559036255s
```

2(b)

Train Set Accuracy 99.92%
Confusion MAtrix for train data

```
[[2000    0    0    0    0    0    0    0    0    0]
 [   0 1997    1    0    1    0    0    1    0    0]
 [   0    0 2000    0    0    0    0    0    0    0]
 [   0    0    0 1999    0    0    0    1    0    0]
 [   0    0    0    0 1999    0    0    0    0    1]
 [   0    0    0    0    0 2000    0    0    0    0]
```

[0 0 0 0 1 0 1999 0 0 0]
[0 2 1 0 1 0 0 1995 0 1]
[0 1 0 0 0 0 0 0 1999 0]
[0 0 0 0 2 0 0 2 0 1996]]
Accuracy = 97.23% (9723/10000) (classification)
Test Set Accuracy 97.23%
Confusion MAtrix for test data
[[969 0 1 0 0 3 4 1 2 0]
[0 1121 3 2 1 2 2 0 3 1]
[4 0 1000 4 2 0 1 6 15 0]
[0 0 8 985 0 4 0 6 5 2]
[0 0 4 0 962 0 6 0 2 8]
[2 0 3 6 1 866 7 1 5 1]
[6 3 0 0 4 4 939 0 2 0]
[1 4 19 2 4 0 0 987 2 9]
[4 0 3 10 1 5 3 3 942 3]
[4 4 3 8 13 4 0 9 12 952]]
Time Taken 625.535702

The Training Time for my implementation take 6 times more time than LibSVM implementation
Accuracy of LibSVM is also better on test set by 3% than my implementaion
2(c)
Confusion Matrix have been print in their corresponding part.I can't see any particular class getting mis-classified significantly into other in my predictions.

(d)
Validation set accuracy
{1e-05: 8.95, 0.001: 8.95, 1.0: 97.85000000000001, 5.0: 97.89999999999999, 10: 97.89999999999999}
Test set accuracy
{1e-05: 9.8, 0.001: 9.8, 1.0: 97.21, 5.0: 97.34, 10: 97.34}