

Class Priors:

```
src — -bash — 134x40
Sumons-MacBook-Pro:src sumon$ clear
Sumons-MacBook-Pro:src sumon$ javac *.java
Sumons-MacBook-Pro:src sumon$ java NaiveBayes vocabulary.txt map.csv train_label.csv train_data.csv test_label.csv test_data.csv

Class Priors:
P(Omega = 1) = 0.0426
P(Omega = 2) = 0.0516
P(Omega = 3) = 0.0508
P(Omega = 4) = 0.0521
P(Omega = 5) = 0.0510
P(Omega = 6) = 0.0525
P(Omega = 7) = 0.0516
P(Omega = 8) = 0.0525
P(Omega = 9) = 0.0529
P(Omega = 10) = 0.0527
P(Omega = 11) = 0.0531
P(Omega = 12) = 0.0527
P(Omega = 13) = 0.0524
P(Omega = 14) = 0.0527
P(Omega = 15) = 0.0526
P(Omega = 16) = 0.0532
P(Omega = 17) = 0.0484
P(Omega = 18) = 0.0500
P(Omega = 19) = 0.0412
P(Omega = 20) = 0.0334
```

In our result P_{BE} is always greater than P_{MLE} . Because when we calculate P_{MLE} , only one word with zero occurrence can make the whole posterior to zero. Therefore, often it can not provide the correct result. In case of P_{BE} , we counter this by adding 1 with the number of occurrences.

Performance on Training Data:

In our result, we the overall accuracy on training data is more than accuracy on the test data. For training data it is about 94% and in case of test data is about 78%.

First, we use Bayesian Estimator to measure the performance. Later we also used Maximum Likelihood to measure. The performance using BE on training data is shown below:

Overall Accuracy for Training Data = 0.9411

Class Accuracy for Training Data:

Group 1 = 0.9667
Group 2 = 0.9191
Group 3 = 0.8794
Group 4 = 0.9302
Group 5 = 0.9409
Group 6 = 0.9493
Group 7 = 0.7749
Group 8 = 0.9662
Group 9 = 0.9631
Group 10 = 0.9714
Group 11 = 0.9783
Group 12 = 0.9798
Group 13 = 0.9239
Group 14 = 0.9764
Group 15 = 0.9781
Group 16 = 0.9833
Group 17 = 0.9853
Group 18 = 0.9681
Group 19 = 0.9698
Group 20 = 0.7606

Confusion Matrix for Training Data:

464	0	0	0	0	0	0	0	0	0	1	0	0	0	0	11	0	1	1	2
1	534	6	15	1	9	2	0	1	0	0	2	1	1	2	4	0	0	2	0
1	10	503	23	1	20	2	0	0	0	0	7	1	1	0	2	0	0	1	0
0	10	4	546	4	4	6	2	0	0	0	0	3	0	1	2	0	2	2	1
2	5	2	7	541	3	1	0	2	0	0	2	1	2	2	3	0	1	1	0
0	11	8	1	2	562	0	0	1	1	0	2	0	1	1	0	1	0	1	0
2	3	2	34	6	2	451	17	1	3	3	16	15	5	4	5	5	1	7	0
1	0	0	3	1	2	3	572	1	1	0	1	0	0	0	1	1	1	3	1
0	1	0	1	1	0	4	1	574	0	0	0	0	2	0	2	6	1	3	0
0	3	0	1	0	1	1	3	0	577	4	0	0	1	0	1	2	0	0	0
1	0	1	2	0	1	0	2	0	0	585	1	0	0	0	1	0	2	2	0
0	2	0	0	0	0	0	0	0	0	0	582	0	1	0	0	3	1	5	0
1	4	0	15	5	0	3	2	0	0	1	5	546	2	2	1	2	0	2	0
0	1	0	0	0	1	0	1	0	0	0	1	2	580	0	5	2	0	1	0
2	2	0	1	0	1	0	1	0	0	0	1	0	2	580	1	0	0	2	0
0	0	0	2	0	1	0	0	0	0	1	0	0	0	0	589	1	3	2	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	537	2	3	0
1	1	0	0	0	0	0	0	0	1	1	1	0	2	0	6	0	546	5	0
2	2	0	0	0	0	0	0	0	1	0	3	0	1	0	1	2	2	450	0
25	0	0	0	0	0	0	0	2	0	0	0	0	0	1	39	15	4	4	286

Performance using BE on test data is shown below:

```
src — -bash — 134x44
Overall Accuracy for Test Data = 0.7811
Class Accuracy for Test Data:
Group 1 = 0.7390
Group 2 = 0.7609
Group 3 = 0.5294
Group 4 = 0.7781
Group 5 = 0.7128
Group 6 = 0.7846
Group 7 = 0.5916
Group 8 = 0.9013
Group 9 = 0.8892
Group 10 = 0.8690
Group 11 = 0.9549
Group 12 = 0.9139
Group 13 = 0.6590
Group 14 = 0.8244
Group 15 = 0.8546
Group 16 = 0.9472
Group 17 = 0.8929
Group 18 = 0.8644
Group 19 = 0.5935
Group 20 = 0.3546
```

```
src — -bash — 134x44
Confusion Matrix for Test Data:
235  0  0  0  0  1  0  0  0  0  1  1  1  2  3  45  3  10  7  9
 3 296  6 12  7 22  1  3  2  0  0 17  4  4  7  4  0  0  1  0
 3  33 207 58 11 31  0  2  2  2  1 17  1  4  4  5  0  0  9  1
 0  8 15 305 21  2  4  6  0  0  1  6 23  0  1  0  0  0  0  0
 0  8 10  37 273  3  4  4  1  1  0  6 17  8  2  0  3  0  6  0
 0 42  7 10  2 306  1  0  2  1  0 10  0  0  3  2  1  1  2  0
 0  8  4 50 20  1 226 33  5  0  1  3 11  2  3  4  2  3  6  0
 1  1  0  2  0  1  5 356  4  2  0  1  4  0  2  1  4  2  9  0
 0  1  0  0  0  0  0 26 353  2  0  1  1  1  0  1  4  2  5  0
 4  1  0  1  1  2  3  3  1 345 17  2  2  0  0  3  1  2  9  0
 2  0  0  0  0  0  1  1  0  4 381  1  0  2  1  2  0  1  3  0
 0  4  1  1  2  1  1  0  0  0  0 361  3  2  0  2  8  0  8  1
 2 18  0 27  8  3  1 10  2  0  0 46 259  6  3  6  0  2  0  0
10  7  1  3  0  0  0  4  0  1  0  1  3 324  3 17  3  6 10  0
 3  7  0  0  0  2  0  0  1  0  1  4  4  4 335  5  1  2 22  1
 7  2  1  0  1  2  0  0  0  0  0  1  0  1  0 377  2  2  1  1
 1  0  0  0  1  0  1  2  1  1  1  3  0  1  2  3 325  2 16  4
12  1  0  0  0  0  0  2  1  1  1  4  0  0  0  8  3 325 18  0
 6  1  0  0  0  1  0  1  0  0  0  3  0  3  7  3 95  5 184  1
47  3  0  0  0  0  0  0  1  0  0  1  0  3  5 70 19  5  8 89
```

When we use MLE the performance dropped significantly. The overall accuracy it gives is only 12%. The reason is that for a given class only one word with zero count can make the whole posterior to zero. The performance when we use MLE is given by:

```
src — -bash — 134x44
Overall Accuracy for Test Data using MLE = 0.1233
Class Accuracy for Test Data using MLE:
Group 1 = 0.6667
Group 2 = 0.0308
Group 3 = 0.0307
Group 4 = 0.0536
Group 5 = 0.1567
Group 6 = 0.1564
Group 7 = 0.1728
Group 8 = 0.0228
Group 9 = 0.0907
Group 10 = 0.0277
Group 11 = 0.2506
Group 12 = 0.1089
Group 13 = 0.1196
Group 14 = 0.0509
Group 15 = 0.1429
Group 16 = 0.0729
Group 17 = 0.0522
Group 18 = 0.0532
Group 19 = 0.0710
Group 20 = 0.2749
```

```
src — -bash — 134x44
Confusion Matrix for Test Data using MLE:
212  2   7   3   1   2   1   1   2   3   1   4   0   1   2  11   3   3   3  56
246 12  15  10   8  17  11   5   2   5   4   7   9   6   9   7   3  10   2   1
245  0  12  10  27  12  13   8   7   2   4   5   5  12  10   7   4   1   3   4
214  0   1  21  45   9  23  10   9   6   2   6  16   8   6   5   3   3   4   1
223  0   0   0  60   7   7  12   8   7   4   4  13   8  11   3   6   0   4   6
247  0   1   1   0  61  12   8   5   5   9   6   8   6   7   2   1   6   3   2
183  0   0   2   2   0  66  21  16  14  10   5  14   8  11   9   3   8   7   3
269  1   0   0   0   0   0   9  37  19   3   1   6   9  14   3   6   6   6   6
244  0   0   0   0   0   0   0  36  30  15   5   8  10   8   7  13  12   6   3
272  0   0   0   0   0   0   1   0  11  53   5   4   8   5   8   7   8  13   2
252  0   0   0   0   0   0   0   0   0   0  100   6   3  10   4   5   2  10   3   4
234  1   0   0   2   0   0   0   0   1   0  43  18  12   7   4  23  10  17  23
264  2   1   0   4   1   3   4   1   1   1   1  47  13  23  11   5   2   2   7
320  0   0   0   0   0   1   1   0   0   1   0   0  20   6   9  10   9   9   7
276  0   0   0   0   0   0   0   0   0   0   0   0   1   0  56  19   4  12  17   7
255  1   1   1   0   0   1   0   0   0   0   0   0   1   0  29   4   8   2  95
257  0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0  19  11  27  49
299  0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1  20  23  31
219  0   0   0   0   0   0   0   0   0   0   1   0   0   0   1   0   1   2  22  64
169  1   1   0   0   0   1   0   0   0   0   0   0   0   0   1   6   1   0   2  69
```