CS60050
MACHINE LEARNING

Assignment 2

Animesh Jha - 19CS10070

Nisarg Upadhyaya - 19CS30031

1. Code

There are 3 files -

1. main.py

This file contains code to run the experiments. It has the following methods:

- read_data(): this function reads "train.csv" and returns a pandas dataframe frame
- get_M_matrix(): this function first counts the number of unique words which are more than two letters long, not stopwords and match the given regular expression. After this it constructs the r X c binary matrix M, with M[i][j]=1 iff j-th word is present in the text of the i-th example
- train_test_split(): this generates stratified 70:30 splits on M and author names.
- run_experiment(): this runs the experiment for given test train split and laplace correction factor(alpha)
- Label_encoder(): we use this library function to match author names to the whole numbers 0,1,2 for easier predictions

2. model.py

This file contains the class NaiveBayes, it implements the algorithm, it has the following methods

- __init__(): constructor of the class, stores alpha and number of classes, and initializes label counts, label_word and total_label_word counts to all zeros
- fit(): this method fits the naive bayes model to the given train dataset. It computes label_total_text_counts, label_total_word_counts and label_word_counts (Counts how many words per label, the frequency of the word for a label, and number of words for a label)
- log_p_doc(): this method gives the log of the probability P(word+alpha|label+vocab*alpha) for given word and label. This is used for predictions
- prior(): computes the prior for a given label
- predict(): predicts the labels for a given test dataset, returns the predictions and the probability of the chosen prediction.

3. train.csv

The given dataset. It has the following attributes

1) Text
2) Author – Label

Procedure followed

We use the standard Naive Bayes algorithm. One thing to note is that, we compute the log probabilities for numerical stability, it is better to work with the sum of logs than to work with the multiplication of a bunch of small numbers. However when there is no laplace correction, for a few examples the probabilities become 0, to handle this we have defined log(0) to be a very large negative number (-1000000).

Class mapping:

0 -> EAP
1 -> HPL
2 -> MWS

Total samples: 19579
Vocab size: 24787
Shape X train (13705, 24787)
Shape y train (13705,)
Shape X test (5874, 24787)
Shape y test (5874,)

How to run the code?

After installing all dependencies just run `python main.py` and enjoy! README has more details.

2. Experiments

```
Class mapping:
      0 -> EAP
      1 -> HPL
      2 -> MWS

Total samples: 19579
Vocab size: 24787
Shape X train (13705, 24787)
Shape y train (13705,)
Shape X test (5874, 24787)
Shape y test (5874,)
```

a. Without Laplace Correction

```
                    CONFUSION MATRIX
      True values along the rows, predictions along the columns
```

```
      ['EAP' 'HPL' 'MWS']
EAP: [1913  202  255]
HPL: [ 193 1369  129]
MWS: [ 222  126 1465]
```

```
Accuracy on test split = 0.8081375553285666
95% confidence interval of accuracy on test split = 0.010069936500308747
Hence, accuracy is between 0.7980676188282578 and 0.8182074918288753
```

```
Results for class EAP:
      True positives: 1913
      True negatives: 3089
      False positives: 415
      False negatives: 457
      Sensitivity/Recall: 0.8071729957805908
      Specificity: 0.8815639269406392
      Precision: 0.8217353951890034
      F-score: 0.8143891017454237
```

```
Results for class HPL:
      True positives: 1369
      True negatives: 3855
      False positives: 328
      False negatives: 322
      Sensitivity/Recall: 0.8095801301005322
      Specificity: 0.9215873774802773
      Precision: 0.8067177371832646
      F-score: 0.80814639905549

Results for class MWS:
      True positives: 1465
      True negatives: 3677
      False positives: 384
      False negatives: 348
      Sensitivity/Recall: 0.8080529509100938
      Specificity: 0.9054420093573011
      Precision: 0.7923201730665225
      F-score: 0.8001092299290005
```

b. With Laplace Correction (alpha = 1)

```
                              CONFUSION MATRIX
      True values along the rows, predictions along the columns
```

```
      ['EAP' 'HPL' 'MWS']
EAP: [1944  151  275]
HPL: [ 175 1372  144]
MWS: [ 158   94 1561]
```

```
Accuracy on test split = 0.8302689819543753
95% confidence interval of accuracy on test split = 0.009600174231490202
Hence, accuracy is between 0.8206688077228851 and 0.8398691561858654

Results for class EAP:
      True positives: 1944
      True negatives: 3171
      False positives: 333
      False negatives: 426
      Sensitivity/Recall: 0.8202531645569621
      Specificity: 0.9049657534246576
      Precision: 0.8537549407114624
      F-score: 0.8366688185926404
```

```
Results for class HPL:
      True positives: 1372
      True negatives: 3938
      False positives: 245
      False negatives: 319
      Sensitivity/Recall: 0.8113542282672974
      Specificity: 0.9414295959837438
      Precision: 0.8484848484848485
      F-score: 0.8295042321644499

Results for class MWS:
      True positives: 1561
      True negatives: 3642
      False positives: 419
      False negatives: 252
      Sensitivity/Recall: 0.861003861003861
      Specificity: 0.8968234425018469
      Precision: 0.7883838383838384
      F-score: 0.8230951753229634
```

c. With Laplace Correction (alpha = 10)

```
                    CONFUSION MATRIX
 True values along the rows, predictions along the columns

      ['EAP' 'HPL' 'MWS']
EAP: [2015  120  235]
HPL: [ 310 1238  143]
MWS: [ 267  113 1433]
```

```
Accuracy on test split = 0.797752808988764
95% confidence interval of accuracy on test split = 0.010272224929825926
Hence, accuracy is between 0.7874805840589381 and 0.8080250339185899

Results for class EAP:
      True positives: 2015
      True negatives: 2927
      False positives: 577
      False negatives: 355
      Sensitivity/Recall: 0.8502109704641351
      Specificity: 0.8353310502283106
      Precision: 0.777391975308642
      F-score: 0.8121725110842403
```

```
Results for class HPL:
      True positives: 1238
      True negatives: 3950
      False positives: 233
      False negatives: 453
      Sensitivity/Recall: 0.7321111768184506
      Specificity: 0.9442983504661726
      Precision: 0.8416043507817811
      F-score: 0.7830487033523086

Results for class MWS:
      True positives: 1433
      True negatives: 3683
      False positives: 378
      False negatives: 380
      Sensitivity/Recall: 0.7904026475455047
      Specificity: 0.9069194779610933
      Precision: 0.7912755383765875
      F-score: 0.7908388520971303
```

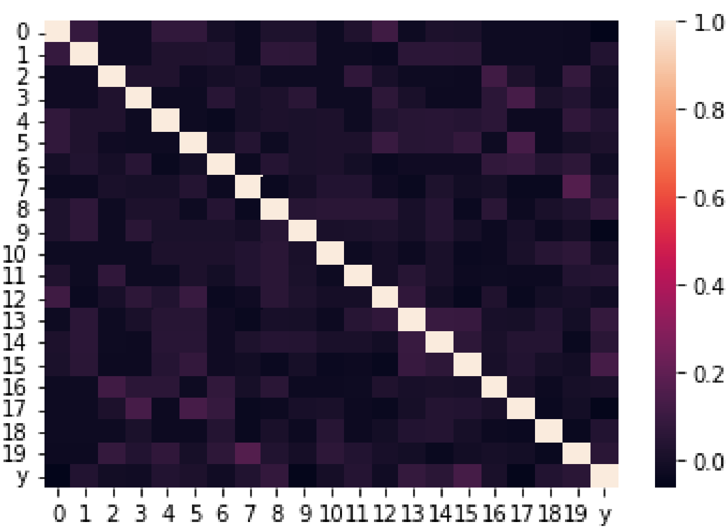
3. Analysis

1. Laplace Correction

As alpha increases, the likelihood probability moves towards uniform distribution.
Since we are not getting much information from that, it is not preferable. Therefore, it is not preferred to use very large values of alpha. Generally, taking alpha = 1 suffices as it smoothens the likelihood while maintaining the original distribution. Hence, the slight reduction in accuracy taking alpha = 10.
Also, on smoothening we can observe a slight increase in the accuracy on the test set. This is because smoothening maintains the likelihood but numerically the values are better distributed (not 0) and hence we get better results.

2. Correlation and Dependence of Features

(Code for this part of the analysis can be found in the file test_and_corr.py)
We wanted to test the naive bayes assumption, because the results of the model are quite good. First observe that if two variables are correlated they can not be independent of each other[1]. Note that zero correlation does not imply independence, instead we focus on the fact that two correlated variables will never be independent.
Therefore to check independence we check correlation of the features. To this end we randomly sample 5% of the dataset, off the 979 samples we get we then count the 20 most commonly appearing words. On this reduced dataset we compute the correlation matrix, we observe that for most part the correlation between two different features is really small (in the figure purple color denotes low values) and close to zero. We also compute the correlation of each word with the label and observe that for most part the correlation is really small. Therefore the data while not completely independent is "nearly" independent which is why the naive bayes model is able to do well even though it makes wrong assumptions about the dataset.



Correlation matrix for the 20 most common words in the randomly sampled 5% dataset

0	-0.063543
9	-0.053651
17	-0.052063
12	-0.016584
6	-0.015318
3	-0.014600
2	-0.010554
10	-0.004828
16	0.010618
5	0.017574
7	0.029308
1	0.033603
4	0.033662
18	0.034168
11	0.041597
19	0.054157
14	0.055277
8	0.079333
13	0.082493
15	0.127957
y	1.000000

Correlation of the 20 most common words with the label

Reference:
[1] Proof specified in
https://stats.stackexchange.com/questions/113417/does-non-zero-correlation-imply-dependence