Naive Bayes

Introduction

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback. Sentiment analysis is used in various places, like categorizing reviews into good or bad on online shopping websites or categorizing movie reviews, etc.
The problem is to classify the sentences according to their sentiments. According to the data set used, 1 is used to represent positive sentiment and 0 represents negative sentiment. The data set is a .txt file.

Problem Definition and Algorithm

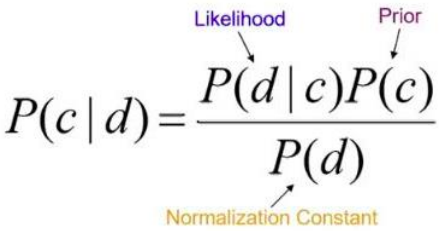
Problem : Using the data to train the model and classify the sentences from the test set as positive or negative sentiment.

Input:

- A document d - a1_d3.txt
- A fixed set of classes C = {c_1, c_2} which in this case is {0,1}

Output: A predicted class c ∈ C

Using Naive Bayes Classification for sentiment analysis:



Calculating the likelihood probability using:

$$P(x_1,...,x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet ... \bullet P(x_n \mid c)$$

We made predictions according to Naive Bayes algorithm and used Laplace Smoothing to smooth categorical data. Laplace Smoothing is introduced to solve the problem of zero probability. This handled out-of-vocabulary words. By applying this method, prior probability and conditional probability was calculated as: where V is the length of the vocabulary.

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

Design Decisions and Result:

Case with no text Processing :

```
In [4]: df = pd.DataFrame(clean_data, columns =['Review', 'Sentiment'])
        print(df)
```

```
                                                Review Sentiment
0    So there is no way for me to plug it in here i...         0
1                          Good case, Excellent value.         1
2                               Great for the jawbone.         1
3    Tied to charger for conversations lasting more...         0
4                                    The mic is great.         1
..                                                 ...       ...
995  The screen does get smudged easily because it ...         0
996  What a piece of junk.. I lose more calls on th...         0
997                         Item Does Not Match Picture.         0
998  The only thing that disappoint me is the infra...         0
999  You can not answer calls with the unit, never ...         0

[1000 rows x 2 columns]
```

```
Accuracy
75.7   ±   3.0757112998459397

F-score
0.7755810590421597   ±   0.023217857350729
```

Model using Text processing as converting the sentences to lower case but does not remove punctuation and symbols:

```
In [13]: df = pd.DataFrame(clean_data, columns =['Review', 'Sentiment'])
         print(df)
```

```
                                                Review Sentiment
0    so there is no way for me to plug it in here i...         0
1                          good case, excellent value.         1
2                               great for the jawbone.         1
3    tied to charger for conversations lasting more...         0
4                                    the mic is great.         1
..                                                 ...       ...
995  the screen does get smudged easily because it ...         0
996  what a piece of junk.. i lose more calls on th...         0
997                         item does not match picture.         0
998  the only thing that disappoint me is the infra...         0
999  you can not answer calls with the unit, never ...         0

[1000 rows x 2 columns]
```

```
Accuracy
77.6   ±   1.6852299546352716

F-score
0.7895244858650496   ±   0.022418735931030213
```

Model using Text processing as removing punctuations but the sentences are not converted to lowercase:

```
Accuracy
77.5   ±   1.7320508075688772

F-score
0.7921120549197084   ±   0.01345271509437666
```

Model using Text Processing: Punctuations / symbols are removed and sentences are converted to lowercase

```
In [14]: df = pd.DataFrame(clean_data, columns =['Review', 'Sentiment'])
         print(df)
```

	Review	Sentiment
0	so there is no way for me to plug it in here i...	0
1	good case excellent value	1
2	great for the jawbone	1
3	tied to charger for conversations lasting more...	0
4	the mic is great	1
..	...	...
995	the screen does get smudged easily because it ...	0
996	what a piece of junk i lose more calls on this...	0
997	item does not match picture	0
998	the only thing that disappoint me is the infra...	0
999	you can not answer calls with the unit never w...	0

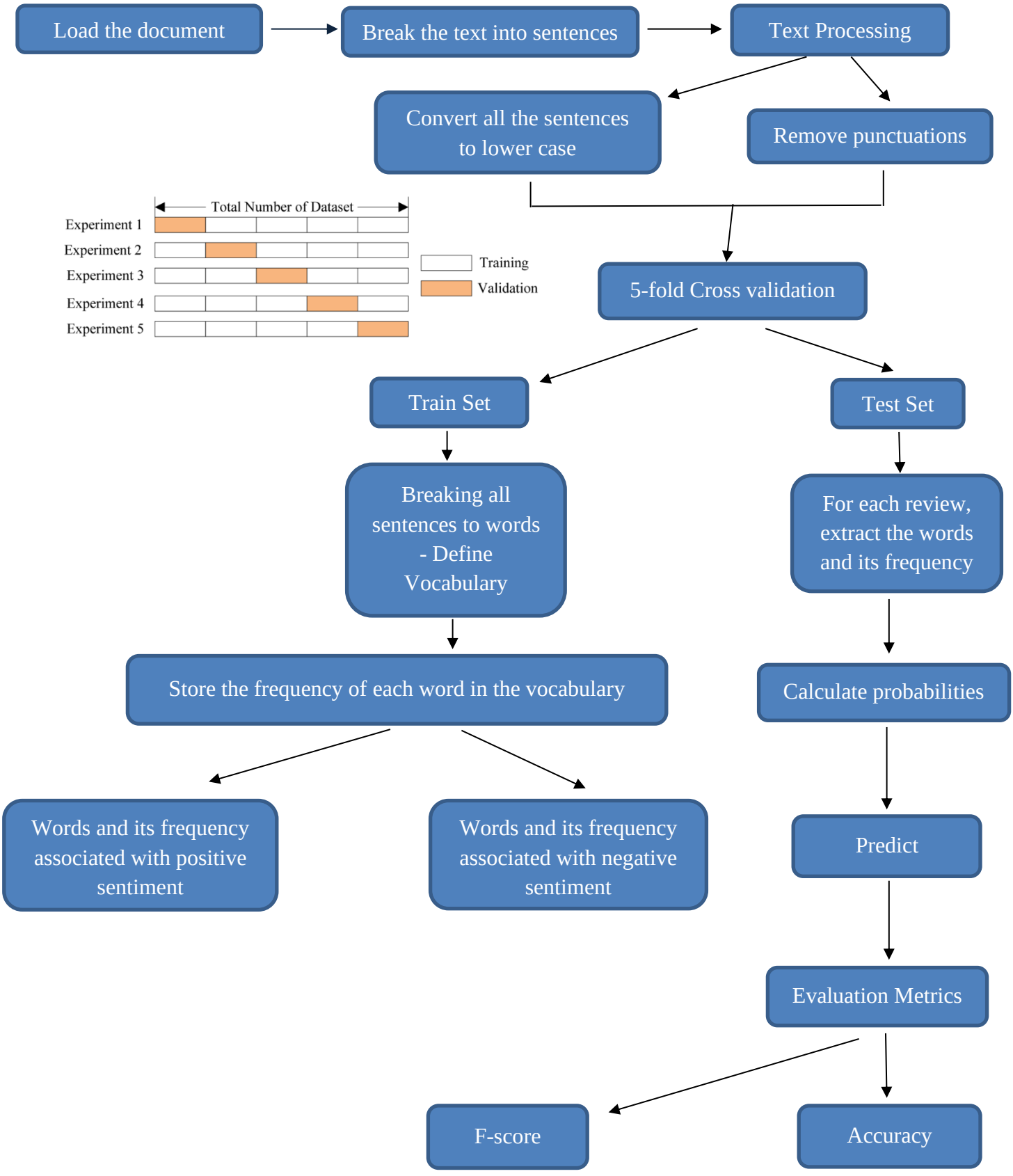
[1000 rows x 2 columns]

```
Accuracy
82.0   ±   1.9748417658131499

F-score
0.828594432942259   ±   0.024980819025272 08
```

It can be seen that the model performed better with text processing, and this was the final accuracy and F-score of the model.

Algorithm:



Results:

The model performed well. Final evaluation metrics -

```
Accuracy
82.0   ±   1.9748417658131499

F-score
0.828594432942259   ±   0.024980819025272 08
```