System Setup and Configuration:

CPU : Ryzen 7

Memory: 16 gb GPU: Nvidia gtx 3070 Library Used:

NLTK, Pandas, SKLEARN, MATPLOTLIB Training Dataset:

Review Dataset:

Positive Review Count – 1000

Negative Review Count – 1000

1) Data Preprocessing:

Performed - tokenization, lowercasing, Lemmatization and removed stop word and numbers in the 3rd Question. For 1st and 2nd question used the data from the sentiment reader file.

Document-term Matrix Representation:

array([[0., 1., 2., ..., 0., 0., 0.],

[0., 0., 1., ..., 0., 0., 0.],

[0., 0., 3., ..., 0., 0., 0.],

...,

[0., 1., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.]])

Word2Vec Representation with Mean Pooling:

array([[ 0.08483946, 0.01732197, 0.02084173, ..., -0.07800613,

0.00803693, 0.03369416],

[ 0.04032861, 0.02374937, -0.03442801, ..., -0.02238353, -0.00908531, -0.02877547],

[ 0.04445129, 0.01227202, 0.00535108, ..., -0.05605149, 0.0227796 , -0.00781159],

[-0.00379793, -0.01923468, 0.00137676, ..., -0.02183986, 0.00983723, 0.00140602],

[ 0.04913921, 0.00857398, -0.01460519, ..., -0.03292092, 0.03446993, -0.00181115]])

Data Split and Class Imbalance:

No of instances/ records 2000

No of Features/vocabulary 13989

Training data

Class 0 788

Class 1 812

Test data

Class 0 212

Class 1 188

When used Sentiment reader to split the data, it imbalances the classes.

2) Modeling:

1. Building multiple models with different hyperparameters:

i. Number of Hidden layers with neurons: (100),(100,100) , (100,100,100) ii. Activation Functions used: Logistic(Sigmoid), Tanh, Relu

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 2 | 1 | No of hidden layer/activation fucntion |
| A white paper with black text  Description automatically generated | A white screen shot of a computer  Description automatically generated | A white paper with black text  Description automatically generated | Si  gmoid |
|  |  |  | Tanh |
|  |  |  | Relu |

1. Performance Metrics for the different combinations

We can see that from above table the accuracy, precision and recall are almost similar to one another but when we look in to the confusion matrix we can see that the last configuration performed better and balances among both the classes. If we also look in to the average precision score (AP) it is higher for the last configuration which is three hidden layers with relu activation function.

This is also due to the numbers and other unnecessary characters that is present inside the datas. These are removed for the next methods.

3)

Now for the third question we are using the word2vec pretrained model from google dataset which comprised of about 100 billion words. It generates 300-dimensional vector for words and phrases.

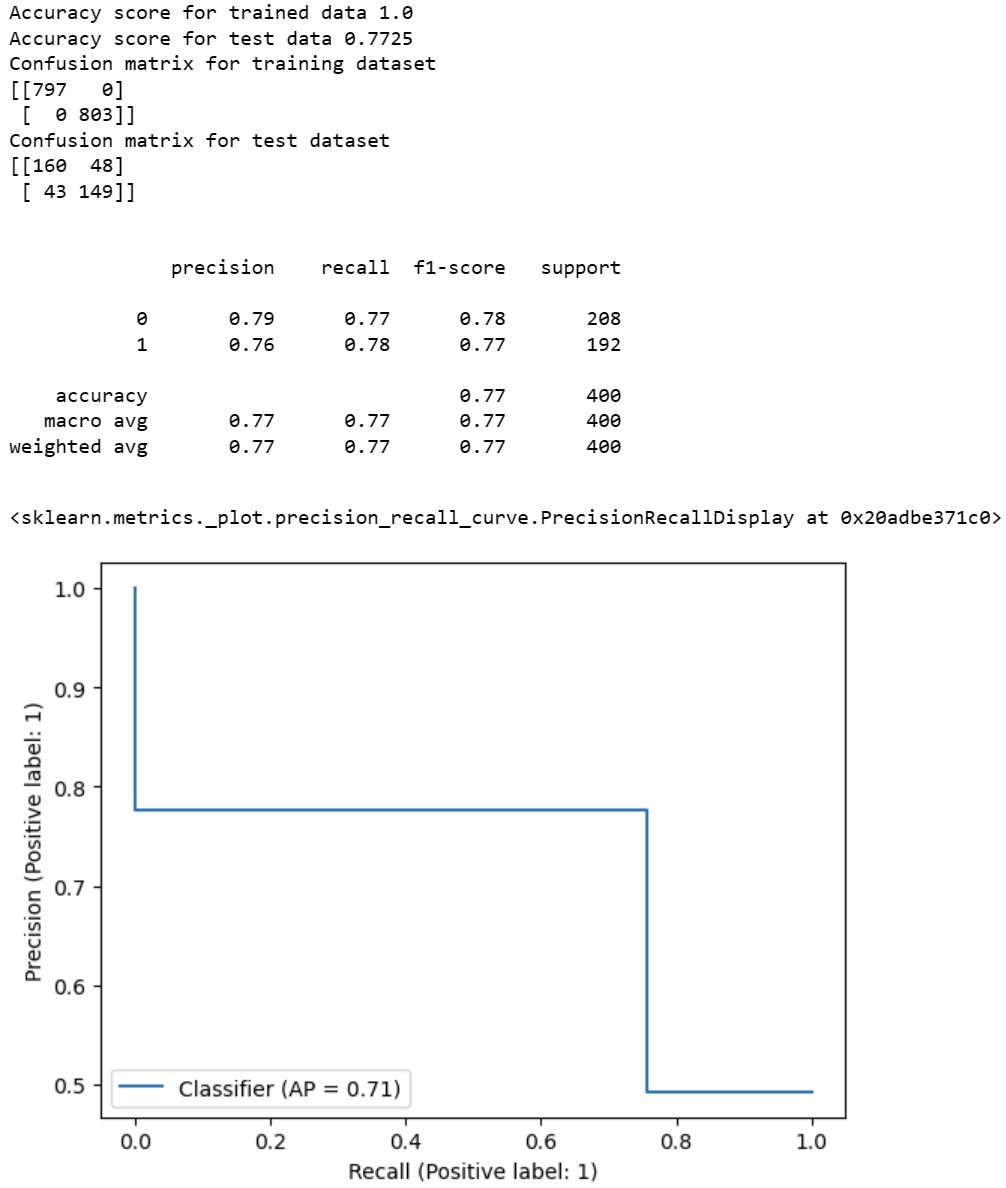
Size of the pretrained model : 1.5GB

Initially when words are sent it doesn’t able to generate the value for it, since it is not part of its vocabulary itself which we can consider it as junk data. Then preprocessed it and used try catch method to eliminate the junk data.

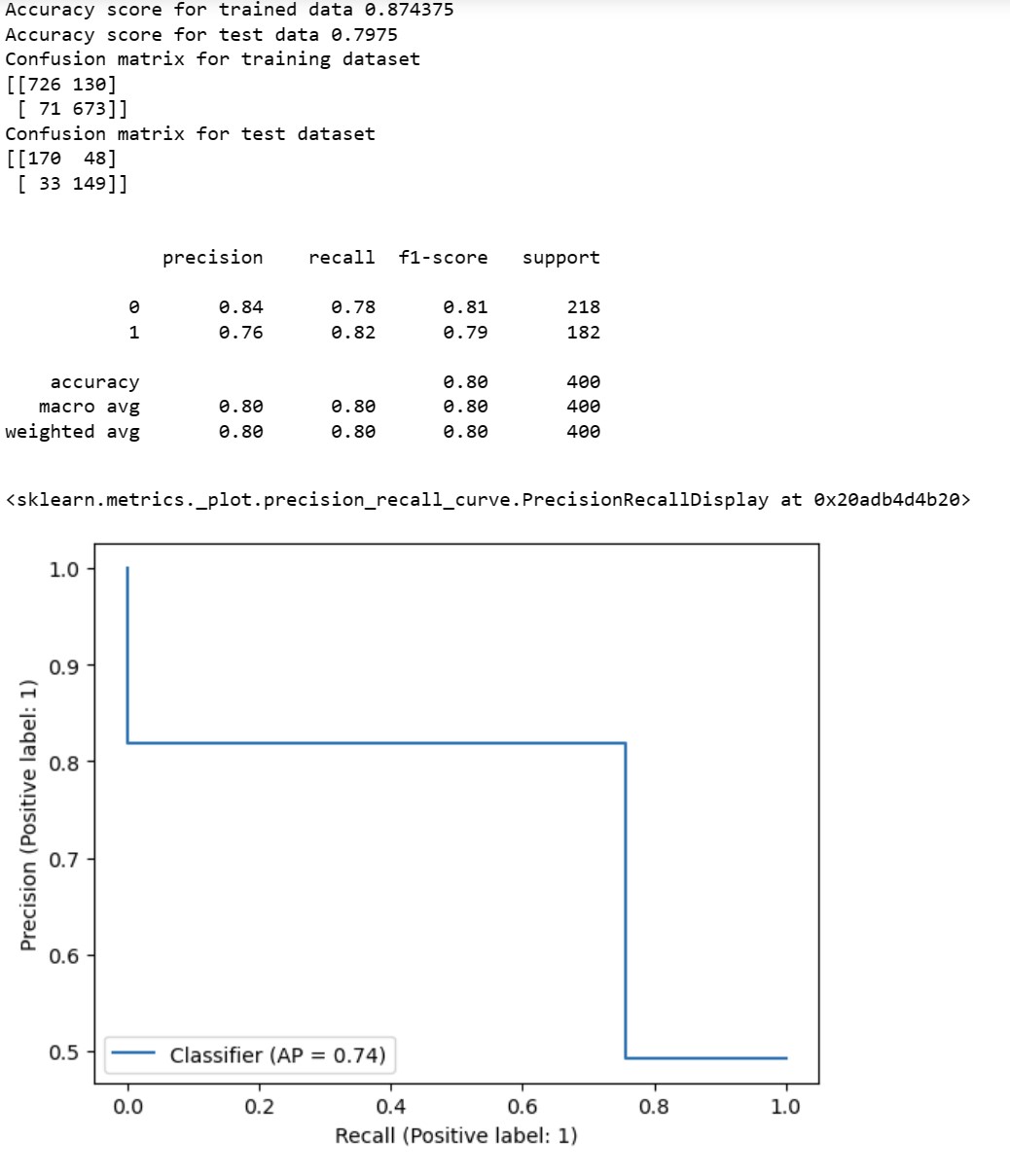
Once the vectors values are generated for each words, it is summed up and divided by the total number of words present in that sentence.

Performance output for top 2 models from the previous evaluation:

1. Configuration: 3 hidden layers(100,100,100) with Relu as activation function.



1. Configuration: 3 hidden layers(100,100,100) with Tanh as activation function.



As we can see eventhough Relu activation function performed well with document matrix as input , Tanh activation function with word embedding performed well.

Another important observation is with word embedding the number of input neurons is reduced which results in decreased in runtime and increases the efficiency.