IR ASSIGNMENT 2 - Report

Janak Kapuriya (MT22032)
Kirtirajsinh Mahida(MT22104)
Mansi Patel(MT22109)

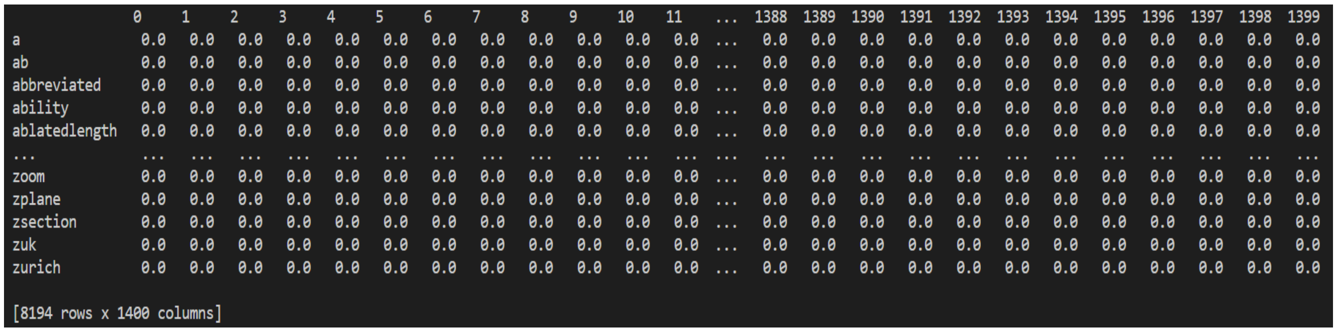
Relevant data extraction and Preprocessing

- Done in the earlier assignment

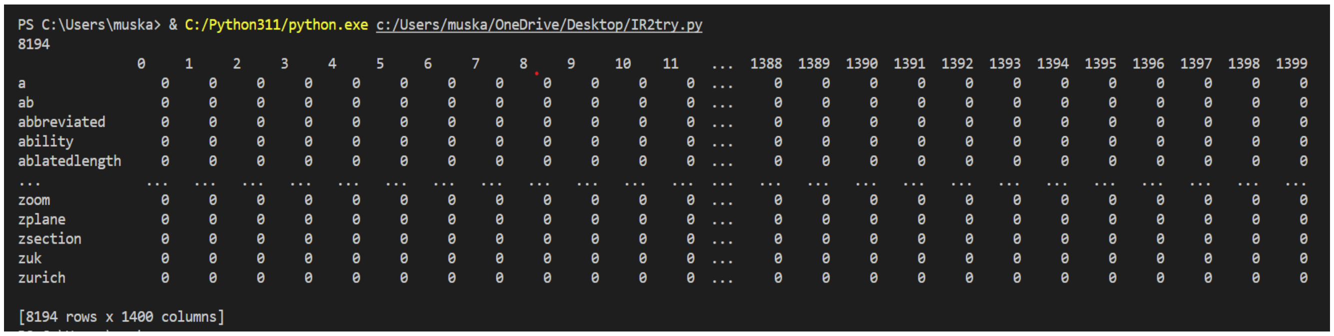
TFIDF Matrix

We created five different matrices for different TF weighting schemes, namely binary, raw count, term frequency, log normalization, and double normalization.
The results are shown below:

1) Binary TF and TF-IDF matrix:

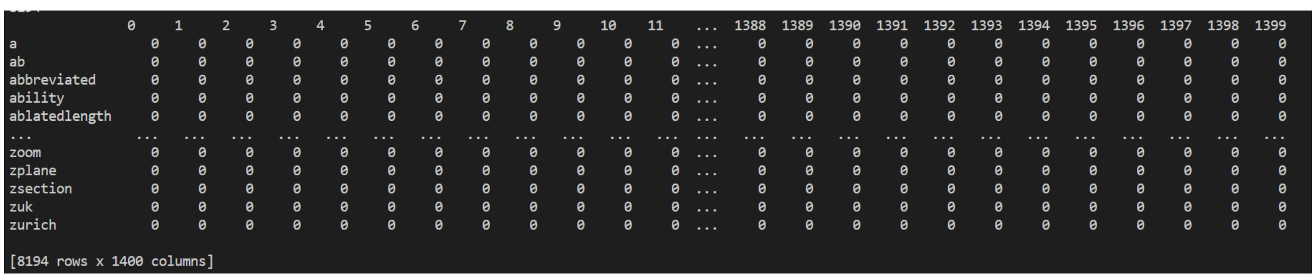


```
                0    1    2    3    4    5    6    7    8    9    10   11   ...  1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399
a               0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated     0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...             ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane          0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich          0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

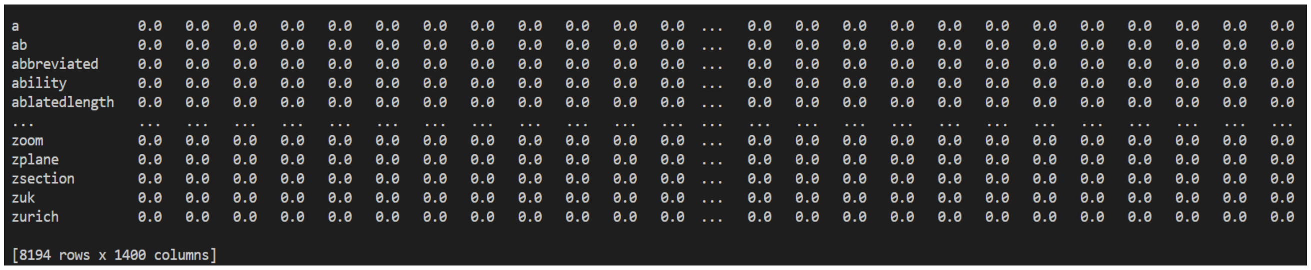


```
PS C:\Users\muska> & C:/Python311/python.exe c:/Users/muska/OneDrive/Desktop/IR2try.py
8194
                0    1    2    3    4    5    6    7    8    9    10   11   ...  1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399
a                  0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
ab                 0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
abbreviated        0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
ability            0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
ablatedlength      0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
...              ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom               0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
zplane             0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
zsection           0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
zuk                0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0
zurich             0    0    0    0    0    0    0    0    0    0    0    0 ...     0    0    0    0    0    0    0    0    0    0    0    0

[8194 rows x 1400 columns]
```

2) **Raw Count TF and TF-IDF matrix:**

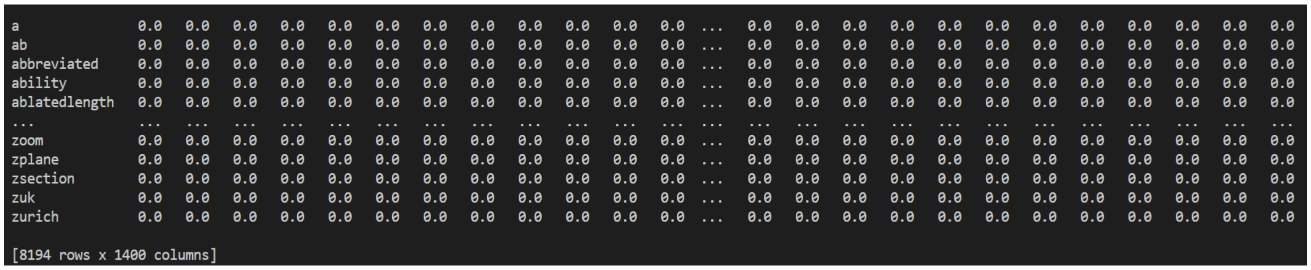


```
                 0    1    2    3    4    5    6    7    8    9    10   11   ...  1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399
a                0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
ab               0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
abbreviated      0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
ability          0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
ablatedlength    0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
...             ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom             0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
zplane           0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
zsection         0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
zuk              0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0
zurich           0    0    0    0    0    0    0    0    0    0    0    0   ...   0    0    0    0    0    0    0    0    0    0    0    0

[8194 rows x 1400 columns]
```

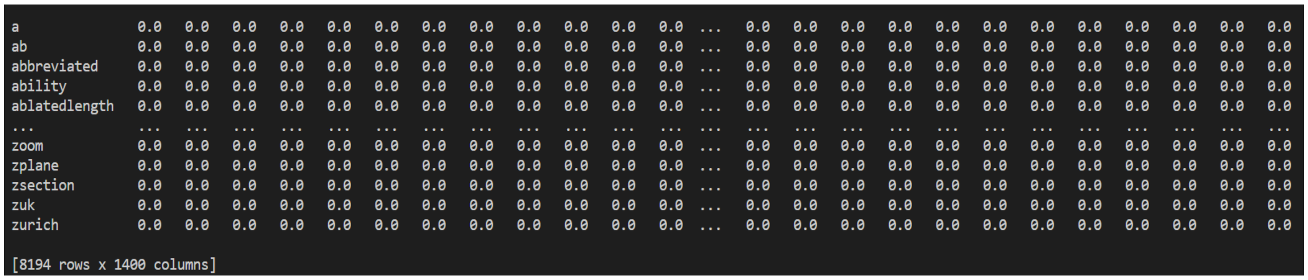


```
a              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection       0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

3) **Term Frequency TF and TF-IDF matrix:**

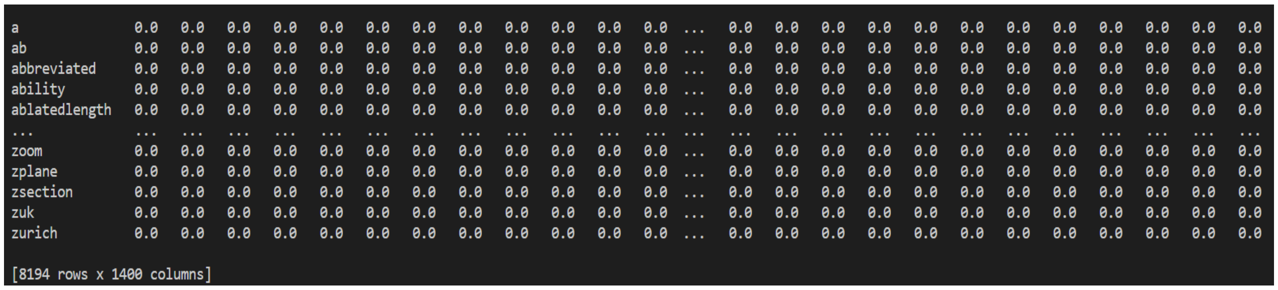


```
a              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection       0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

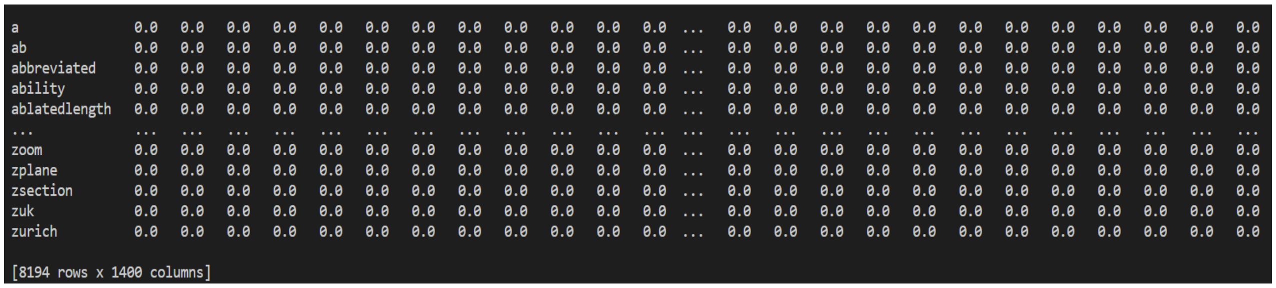


```
a              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection       0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

4) Log Normalization TF and TF-IDF matrix:

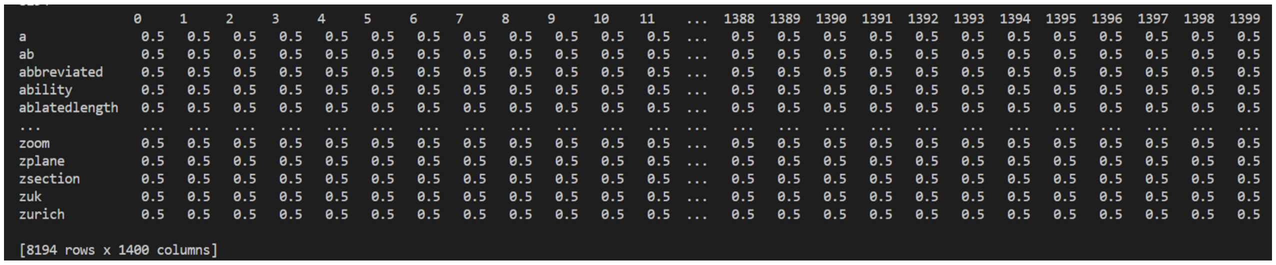


```
a              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection       0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

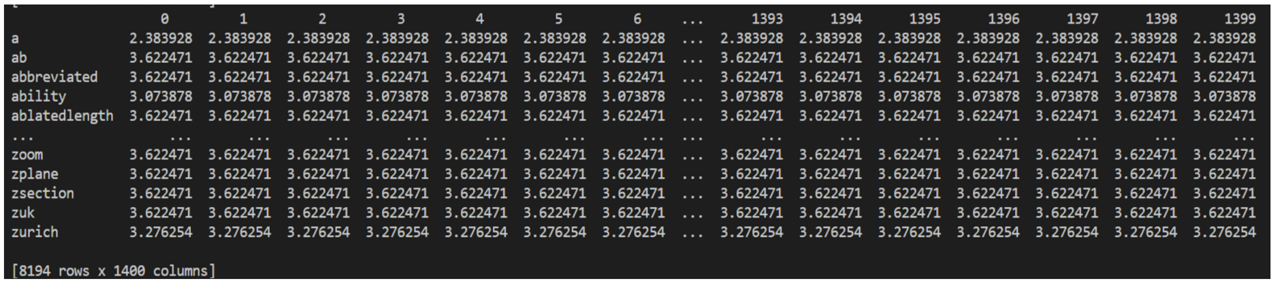


```
a              0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ab             0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
abbreviated    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ability        0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
ablatedlength  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zplane         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zsection       0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zuk            0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
zurich         0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[8194 rows x 1400 columns]
```

5) Log Normalization TF and TF-IDF matrix:

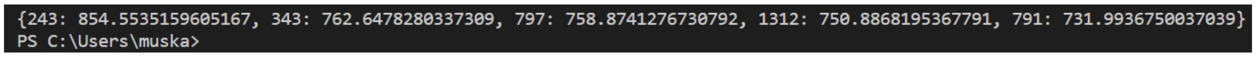


```
                0    1    2    3    4    5    6    7    8    9    10   11   ...  1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399
a              0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
ab             0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
abbreviated    0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
ability        0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
ablatedlength  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
...            ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
zoom           0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
zplane         0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
zsection       0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
zuk            0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
zurich         0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  ...  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5

[8194 rows x 1400 columns]
```

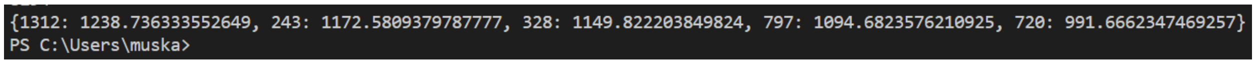


```
                      0         1         2         3         4         5         6  ...      1393      1394      1395      1396      1397      1398      1399
a              2.383928  2.383928  2.383928  2.383928  2.383928  2.383928  2.383928  ...  2.383928  2.383928  2.383928  2.383928  2.383928  2.383928  2.383928
ab             3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
abbreviated    3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
ability        3.073878  3.073878  3.073878  3.073878  3.073878  3.073878  3.073878  ...  3.073878  3.073878  3.073878  3.073878  3.073878  3.073878  3.073878
ablatedlength  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
...                 ...       ...       ...       ...       ...       ...       ...  ...       ...       ...       ...       ...       ...       ...       ...
zoom           3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
zplane         3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
zsection       3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
zuk            3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  ...  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471  3.622471
zurich         3.276254  3.276254  3.276254  3.276254  3.276254  3.276254  3.276254  ...  3.276254  3.276254  3.276254  3.276254  3.276254  3.276254  3.276254

[8194 rows x 1400 columns]
```

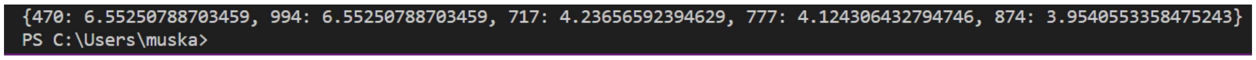
After finding all metrics, we need to make a query of size vocabulary and return the top 5 documents which contain that vocabulary. so we used all metrics and found the top 5 documents, and here is the result :



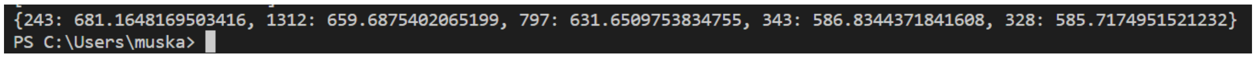
```
{243: 854.5535159605167, 343: 762.6478280337309, 797: 758.8741276730792, 1312: 750.8868195367791, 791: 731.9936750037039}
PS C:\Users\muska>
```



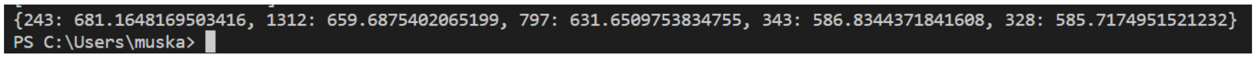
```
{1312: 1238.736333552649, 243: 1172.5809379787777, 328: 1149.822203849824, 797: 1094.6823576210925, 720: 991.6662347469257}
PS C:\Users\muska>
```



```
{470: 6.55250788703459, 994: 6.55250788703459, 717: 4.23656592394629, 777: 4.124306432794746, 874: 3.9540553358475243}
PS C:\Users\muska>
```



```
{243: 681.1648169503416, 1312: 659.6875402065199, 797: 631.6509753834755, 343: 586.8344371841608, 328: 585.7174951521232}
PS C:\Users\muska>
```



```
{243: 681.1648169503416, 1312: 659.6875402065199, 797: 631.6509753834755, 343: 586.8344371841608, 328: 585.7174951521232}
PS C:\Users\muska>
```

Pros and Cons of Weighting Scheme:

1) **Binary :**

Pros:

1. Binary TF-IDF is simpler and computationally less expensive than the standard TF-IDF scheme, as it only counts the presence of a term in a document rather than the frequency of the term.
2. The binary TF-IDF scheme works well for short documents or documents with very low term frequencies, as the term frequency is normalized to either 0 or 1.
3. Binary TF-IDF is particularly useful for tasks like document classification, where the presence or absence of certain terms is more important than their frequency.

Cons:

1. Binary TF-IDF ignores the frequency of a term within a document, which can lead to a loss of information.
2. The binary TF-IDF scheme may not work well for long documents, where the presence or absence of a term may not be as informative as the actual frequency of the term.
3. Binary TF-IDF may not perform as well as the standard TF-IDF scheme in tasks like information retrieval, where term frequency is an important factor in ranking search results.

2) Raw count :

Pros :

1. The raw count TF-IDF scheme is simple and straightforward to implement, and it is widely used in many information retrieval systems.
2. The raw count TF-IDF scheme takes into account the frequency of a term within a document, which can help capture the importance of terms that appear multiple times in a document.
3. The raw count TF-IDF scheme works well for longer documents where the frequency of a term may be more informative than its presence or absence.

Cons :

1. The raw count TF-IDF scheme is sensitive to document length and can result in a bias towards longer documents. Longer documents tend to have higher raw counts of terms, which can lead to the over-representation of those terms in the TF-IDF scores.
2. The raw count TF-IDF scheme can also suffer from the problem of high-frequency terms dominating the score. Common terms that appear in many documents (such as "the" and "and") may have high raw counts, leading to high TF-IDF scores even though they may not be very informative for a given document.
3. The raw count TF-IDF scheme may not work well for short documents or documents with very low term frequencies, where the raw counts are too small to provide meaningful information.

3) Term frequency :

Pros :

1. The term frequency (TF) TF-IDF scheme takes into account the frequency of a term within a document, which can help capture the importance of terms that appear multiple times in a document.
2. The term frequency (TF) TF-IDF scheme is less sensitive to document length than the raw count TF-IDF scheme. This is because the term frequency is normalized by the total number of

terms in the document, which helps to reduce the impact of longer documents on the TF-IDF scores.

3. The term frequency (TF) TF-IDF scheme works well for longer documents where the frequency of a term may be more informative than its presence or absence.

Cons :

1. The term frequency (TF) TF-IDF scheme is still sensitive to high-frequency terms, which can dominate the TF-IDF scores even if they are not very informative. This is because the term frequency is still based on the raw count of the term in the document.
2. The term frequency (TF) TF-IDF scheme may not work well for short documents or documents with very low term frequencies, where the TF scores are too small to provide meaningful information.
3. The term frequency (TF) TF-IDF scheme may not perform as well as the logarithmic TF-IDF scheme (which uses a logarithmic function to scale the term frequency) or the augmented TF-IDF scheme (which normalizes the term frequency by the maximum term frequency in the document) in certain tasks, such as document classification.

4) Log normalization :

Pros :

1. The logarithmic normalization TF-IDF scheme reduces the impact of high-frequency terms, which can dominate the TF-IDF scores even if they are not very informative. This is because the logarithmic function scales the term frequency logarithmically, which results in a more balanced distribution of term frequencies.
2. The logarithmic normalization TF-IDF scheme works well for longer documents where the frequency of a term may be more informative than its presence or absence.
3. The logarithmic normalization TF-IDF scheme is less sensitive to document length than the raw count TF-IDF scheme, as the

logarithmic function helps to reduce the impact of longer documents on the TF-IDF scores.

Cons :

1. The logarithmic normalization TF-IDF scheme still suffers from the problem of high-frequency terms dominating the score, although to a lesser extent than the raw count and term frequency TF-IDF schemes.
2. The logarithmic normalization TF-IDF scheme may not work well for short documents or documents with very low term frequencies, where the TF scores are too small to provide meaningful information.
3. The logarithmic normalization TF-IDF scheme may not be as suitable for tasks like document classification, where the presence or absence of certain terms is more important than their frequency.

5) Double normalization :

Pros **:**

1. The logarithmic normalization TF-IDF scheme reduces the impact of high-frequency terms, which can dominate the TF-IDF scores even if they are not very informative. This is because the logarithmic function scales the term frequency logarithmically, which results in a more balanced distribution of term frequencies.
2. The logarithmic normalization TF-IDF scheme works well for longer documents where the frequency of a term may be more informative than its presence or absence.
3. The logarithmic normalization TF-IDF scheme is less sensitive to document length than the raw count TF-IDF scheme, as the logarithmic function helps to reduce the impact of longer documents on the TF-IDF scores.

<u>Cons</u> **:**

1. The logarithmic normalization TF-IDF scheme still suffers from the problem of high-frequency terms dominating the score, although to a lesser extent than the raw count and term frequency TF-IDF schemes.
2. The logarithmic normalization TF-IDF scheme may not work well for short documents or documents with very low term frequencies, where the TF scores are too small to provide meaningful information.
3. The logarithmic normalization TF-IDF scheme may not be as suitable for tasks like document classification, where the presence or absence of certain terms is more important than their frequency.

Jaccard Coefficient

In the Jaccard coefficient, we have taken the user query and preprocessed it the same as we did on 1400 files. After that, the Jaccard coefficient of each file with a user query was computed, and the top 10 Jaccard coefficient documents were returned.

User Query: player at boundary missed simple catch during cricket match

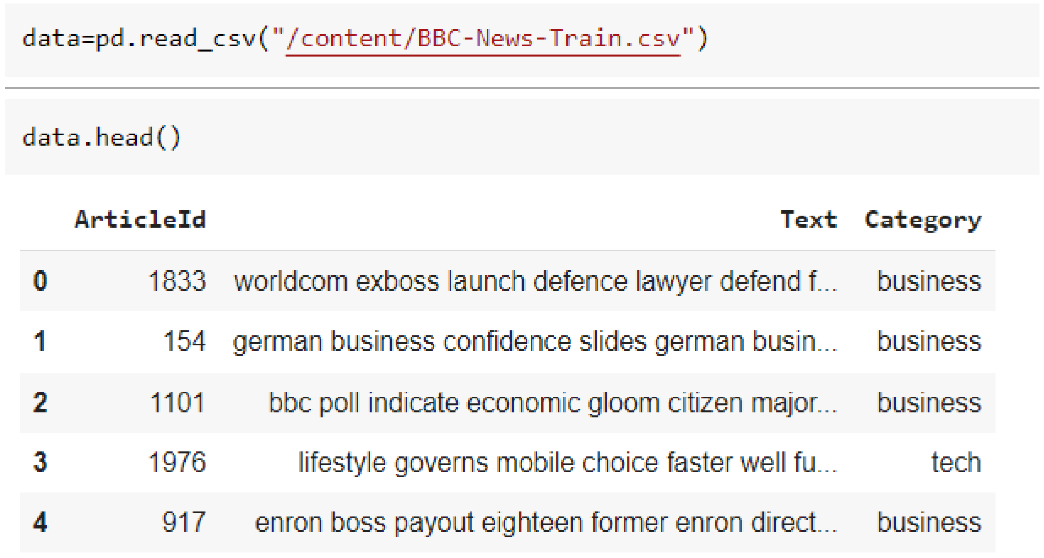
Result :

Cranfield0003	0.09090909090909091
Cranfield0291	0.05128205128205128
Cranfield0324	0.05128205128205128
Cranfield0320	0.047619047619047616
Cranfield0336	0.044444444444444446
cranfield1311	0.044444444444444446
cranfield0652	0.043478260869565216
Cranfield0648	0.0425531914893617
cranfield0359	0.0392156862745098
cranfield0343	0.037037037037037035

Question 2 :

1) <u>Preprocessing</u> :

- In this phase first of all using pandas we read the csv file and after then we apply some preprocessing steps like tokenization , remove stop word , remove punctuation and also provide lemmetization for understanding the meaning of the word
- After applying preprocessing on data we overwrite into the file
- Here are the code and result of that steps :
- After preprocessing we got 20590 unique word that's called our vocabulary and that we mentioned below:



```
data=pd.read_csv("/content/BBC-News-Train.csv")
```

```
data.head()
```

	ArticleId	Text	Category
0	1833	worldcom exboss launch defence lawyer defend f...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicate economic gloom citizen major...	business
3	1976	lifestyle governs mobile choice faster well fu...	tech
4	917	enron boss payout eighteen former enron direct...	business

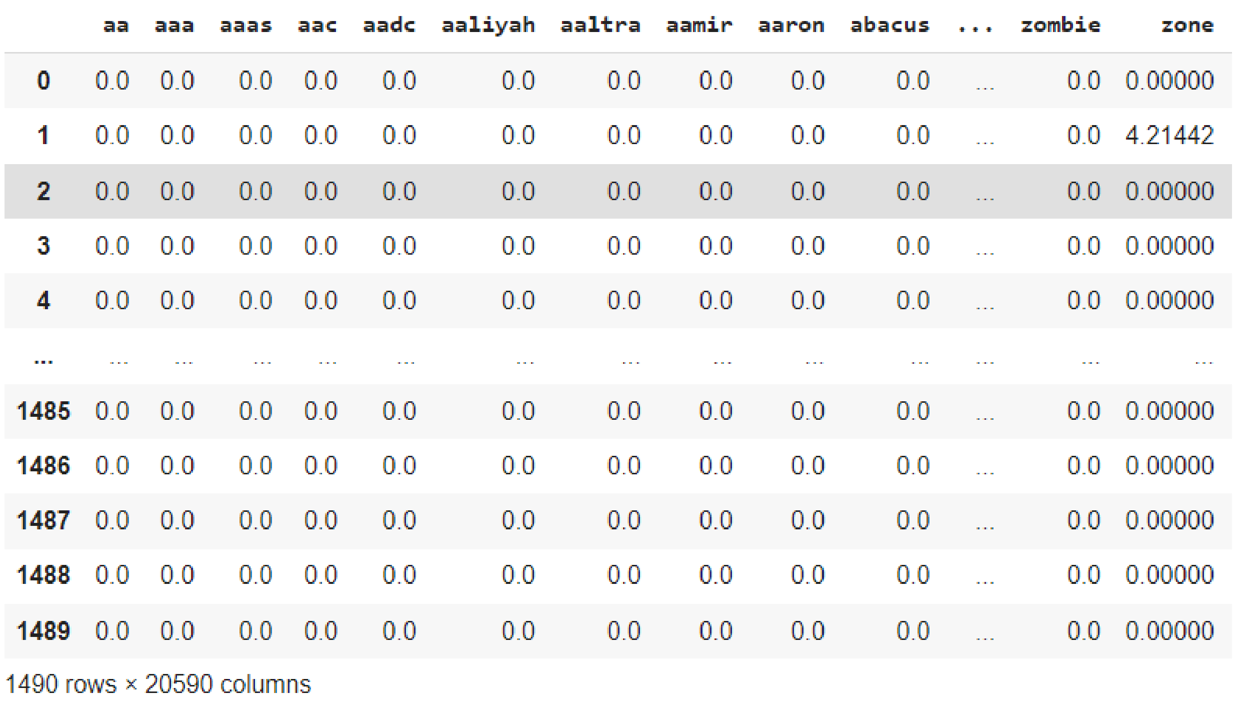
```python
unique_word=[]
for n,i in enumerate(data['Text']):
  s=(i.translate(str.maketrans('', '', string.punctuation))).lower()
  tok=lemmatize_sentence(s)
  tok=[x for x in tok if not any(c.isdigit() for c in x)]
  word = set(stopwords.words('english'))

  rw=[]
  for w in tok:
      if w not in word:
          rw.append(w)
  unique_word=unique_word+rw
  sen=''
  for x in rw:
      sen += x+' '
  data['Text'][n]=sen

corpus_np=np.array(unique_word)
unique_words=np.unique(corpus_np)
print(len(unique_words))
```

```
20590
```

- After that we find the tf-icf matrix of the data using the class frequency and term frequency and then we store that value into the matrix of size of vocabulary and size of number of document.
- Finally we list the all the metrics that we store into the dictionary formate

```python
train_dict=create_dict(unique_words,x,y)
train_tf=term_frequency(unique_words,train_dict)
train_cf=class_frequency(train_tf)
train_icf=icf(train_cf)
train_tf_icf=tf_icf(train_tf,train_icf)
```

```
{'business': ['worldcom exboss launch defence lawyer defend former worldcom chief bernie ebbers battery fra
{'aa': [7, 5, 3, 36, 6], 'aaa': [0, 1, 0, 5, 0], 'aaas': [0, 1, 0, 0, 0], 'aac': [0, 0, 0, 2, 0], 'aadc': [
{'aa': 5, 'aaa': 2, 'aaas': 1, 'aac': 1, 'aadc': 1, 'aaliyah': 1, 'aaltra': 1, 'aamir': 1, 'aaron': 1, 'aba
{'aa': 0.3010299956639812, 'aaa': 0.5440680443502757, 'aaas': 0.7781512503836436, 'aac': 0.778151250383643
{'aa': [2.1072099696478683, 1.505149978319906, 0.9030899869919435, 10.837079843903323, 1.806179973983887],
```



	aa	aaa	aaas	aac	aadc	aaliyah	aaltra	aamir	aaron	abacus	...	zombie	zone
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.21442
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1485	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
1486	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
1487	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
1488	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000
1489	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.00000

1490 rows × 20590 columns

- So this is the database that we created using tf-icf value of each term and after then we converted it into the dataftame for training and testing purpose

2) **Split the dataset** :

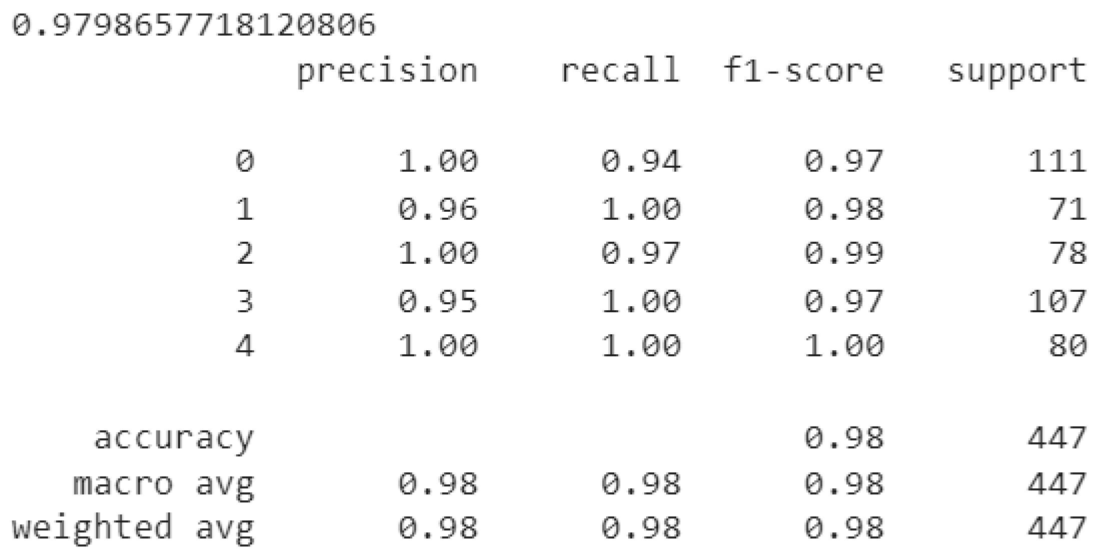
After getting the dataframe using sklearn library we split the dataset into 2 part raining and testing in the ratio of 70:30

3) **Training of dataset using naive bayes algorithm** :

For this steps also we used our if-icf metrix and Guassian Naive Bayes algorithm that will train and test the model after that it will give some accuracy , precision call and f1-score to measure how much our model work perfectly.

4) **Calculate accuracy and classification metrix** :

We got 97.98 % accuracy after applying the model on our dataset and also classification report that contain precision , recall and f1-score also mentioned below :



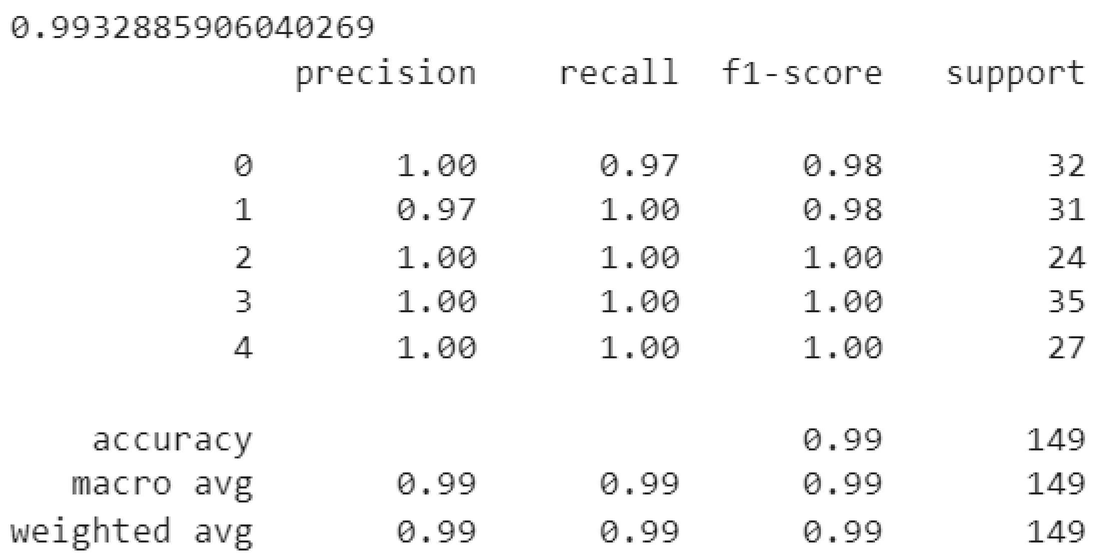
```
0.9798657718120806
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       111
           1       0.96      1.00      0.98        71
           2       1.00      0.97      0.99        78
           3       0.95      1.00      0.97       107
           4       1.00      1.00      1.00        80

    accuracy                           0.98       447
   macro avg       0.98      0.98      0.98       447
weighted avg       0.98      0.98      0.98       447
```

5) **Improve the performance by applying different preprocessing** :

For improving our model we need to change some preprocessing scheme and also some parameter tuning so first of all we change the ratio of training and testion dataset and then measure the accuracy of our model so that are all mentioned below

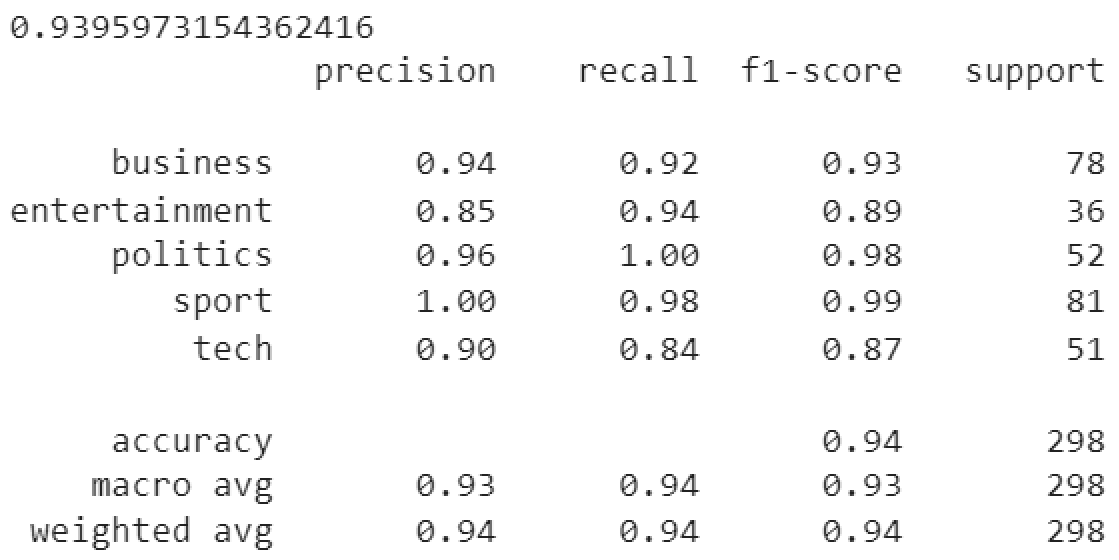
```
0.9832214765100671
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       120
           1       0.97      1.00      0.98       123
           2       0.97      0.97      0.97       117
           3       0.98      1.00      0.99       139
           4       1.00      0.97      0.98        97

    accuracy                           0.98       596
   macro avg       0.98      0.98      0.98       596
weighted avg       0.98      0.98      0.98       596
```

```
0.9664429530201343
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       161
           1       0.91      1.00      0.95       134
           2       0.98      0.94      0.96       139
           3       0.96      0.99      0.98       172
           4       0.99      0.95      0.97       139

    accuracy                           0.97       745
   macro avg       0.97      0.97      0.97       745
weighted avg       0.97      0.97      0.97       745
```



```
0.9932885906040269
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        32
           1       0.97      1.00      0.98        31
           2       1.00      1.00      1.00        24
           3       1.00      1.00      1.00        35
           4       1.00      1.00      1.00        27

    accuracy                           0.99       149
   macro avg       0.99      0.99      0.99       149
weighted avg       0.99      0.99      0.99       149
```

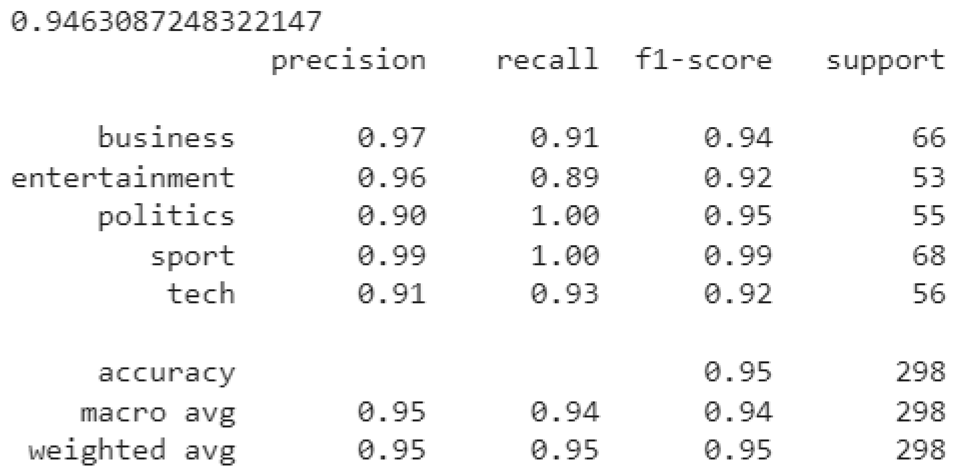
- After that we apply tf-idf vectorization and ngram with naive bayes and here are the accuracy and classification report of that model



```
0.9395973154362416
               precision    recall  f1-score   support

     business       0.94      0.92      0.93        78
entertainment       0.85      0.94      0.89        36
     politics       0.96      1.00      0.98        52
        sport       1.00      0.98      0.99        81
         tech       0.90      0.84      0.87        51

     accuracy                           0.94       298
    macro avg       0.93      0.94      0.93       298
 weighted avg       0.94      0.94      0.94       298
```

```
0.930648769574944
               precision    recall  f1-score   support

     business       0.95      0.93      0.94       111
entertainment       0.94      0.86      0.90        87
     politics       0.93      0.96      0.94        77
        sport       0.99      0.96      0.97       113
         tech       0.80      0.95      0.87        59

     accuracy                           0.93       447
    macro avg       0.92      0.93      0.92       447
 weighted avg       0.93      0.93      0.93       447
```

```
0.9429530201342282
               precision    recall  f1-score   support

     business       0.95      0.86      0.90        63
entertainment       0.94      0.97      0.95        62
     politics       0.89      0.94      0.91        50
        sport       1.00      1.00      1.00        73
         tech       0.92      0.94      0.93        50

     accuracy                           0.94       298
    macro avg       0.94      0.94      0.94       298
 weighted avg       0.94      0.94      0.94       298
```


```
0.9463087248322147
               precision    recall  f1-score   support

     business       0.97      0.91      0.94        66
entertainment       0.96      0.89      0.92        53
     politics       0.90      1.00      0.95        55
        sport       0.99      1.00      0.99        68
         tech       0.91      0.93      0.92        56

     accuracy                           0.95       298
    macro avg       0.95      0.94      0.94       298
 weighted avg       0.95      0.95      0.95       298
```

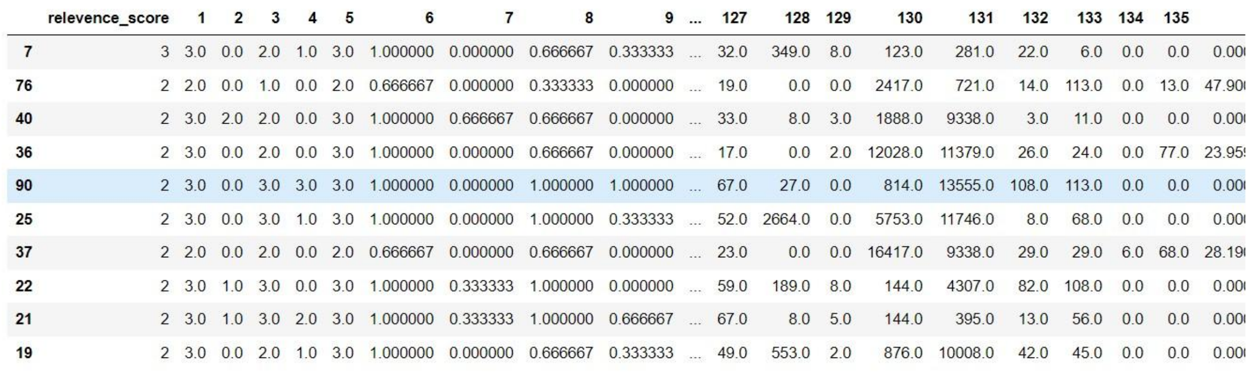
6) <u>Conclusion</u> :

- In the conclusion we follow all the methods and steps that mentioned above and also attach some screenshots of our outputs
- So if we compared all the result then we can say that we got the maximum accuracy when we apply naive bayes with tf-icf vectorization because it divide the data into classes and then find the frequency of the term for getting the priority of that term in the documents.
- So that will give us the if-icf vector and from that vector we can easily find the priority of the terms to identify the classes of thet text.
- We also apply document vise frequency but that will give us 0.94 accuracy while our model has 0.97 accuracy
- Also we apply n gram string materic in tf-idf vectorization that also give 0.95 accuracy
- So at the end we conclude that our model that build with the tf-icf vector and naive bayes will predict the correct class of text more accurately compare to other method and model.

Question 3

Objective 1

Save files in order of maximum DCG of the dataset



	relevence_score	1	2	3	4	5	6	7	8	9	...	127	128	129	130	131	132	133	134	135	
7	3	3.0	0.0	2.0	1.0	3.0	1.000000	0.000000	0.666667	0.333333	...	32.0	349.0	8.0	123.0	281.0	22.0	6.0	0.0	0.0	0.000
76	2	2.0	0.0	1.0	0.0	2.0	0.666667	0.000000	0.333333	0.000000	...	19.0	0.0	0.0	2417.0	721.0	14.0	113.0	0.0	13.0	47.90
40	2	3.0	2.0	2.0	0.0	3.0	1.000000	0.666667	0.666667	0.000000	...	33.0	8.0	3.0	1888.0	9338.0	3.0	11.0	0.0	0.0	0.000
36	2	3.0	0.0	2.0	0.0	3.0	1.000000	0.000000	0.666667	0.000000	...	17.0	0.0	2.0	12028.0	11379.0	26.0	24.0	0.0	77.0	23.95
90	2	3.0	0.0	3.0	3.0	3.0	1.000000	0.000000	1.000000	1.000000	...	67.0	27.0	0.0	814.0	13555.0	108.0	113.0	0.0	0.0	0.000
25	2	3.0	0.0	3.0	1.0	3.0	1.000000	0.000000	1.000000	0.333333	...	52.0	2664.0	0.0	5753.0	11746.0	8.0	68.0	0.0	0.0	0.000
37	2	2.0	0.0	2.0	0.0	2.0	0.666667	0.000000	0.666667	0.000000	...	23.0	0.0	0.0	16417.0	9338.0	29.0	29.0	6.0	68.0	28.19
22	2	3.0	1.0	3.0	0.0	3.0	1.000000	0.333333	1.000000	0.000000	...	59.0	189.0	8.0	144.0	4307.0	82.0	108.0	0.0	0.0	0.000
21	2	3.0	1.0	3.0	2.0	3.0	1.000000	0.333333	1.000000	0.666667	...	67.0	8.0	5.0	144.0	395.0	13.0	56.0	0.0	0.0	0.000
19	2	3.0	0.0	2.0	1.0	3.0	1.000000	0.000000	0.666667	0.333333	...	49.0	553.0	2.0	876.0	10008.0	42.0	45.0	0.0	0.0	0.000

The total files which have maximum DCS are

15388893622806154783964672633468962481061616562754466351013516109251784866051703
26155612828485828968844216913791477960601289924496793568264113015889676630677782
528

Objective 2

Compute nDCG at 50

- 0.3521042740324887

Compute nDCG over the whole dataset

- 0.5979226516897831

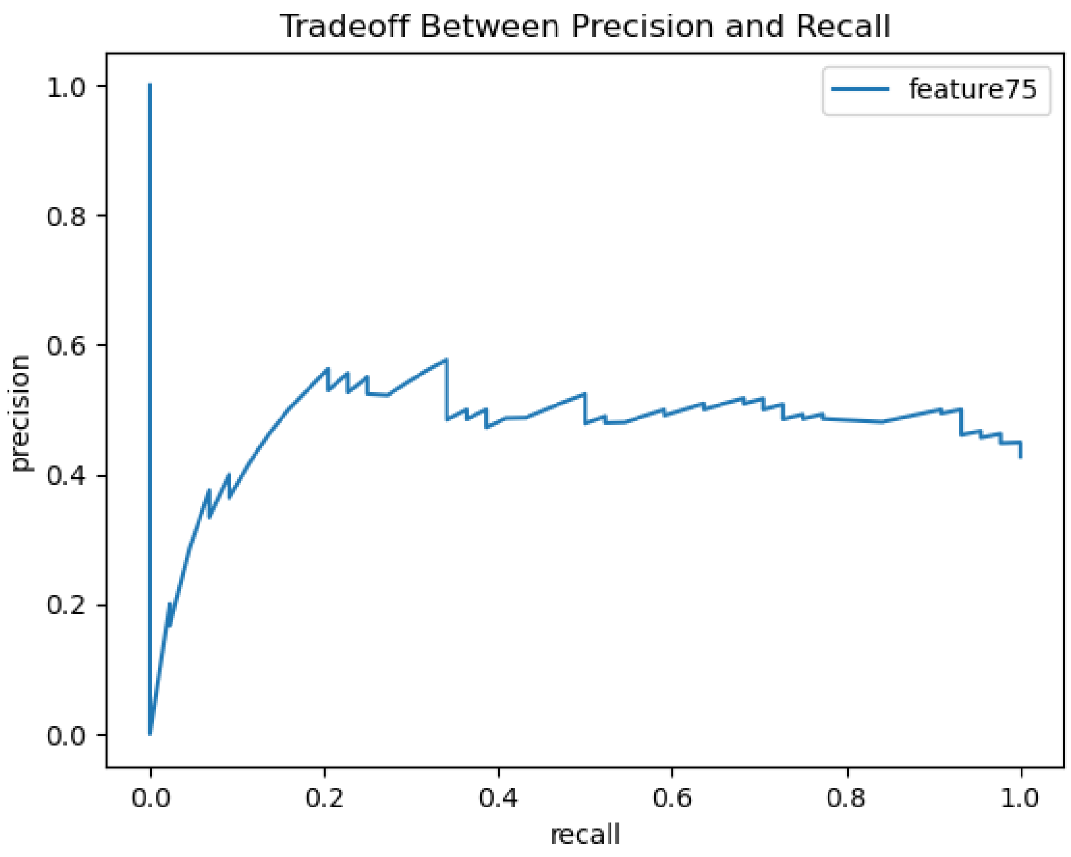
Objective 3

Tradeoff between Precision and Recall

- We say that document is relevant if its value is greater than or equal to the threshold and nonrelevant if the value is less than the threshold.

- When the threshold value is very high, then very few documents are considered relevant, so precision is high because all retrieved documents are relevant but recall is very low because not all relevant documents are retrieved by a high threshold.

- When the threshold value is low, then most of the documents are considered relevant, so precision is low because it retrieved documents that are not relevant and recall is very high because all relevant documents are retrieved by a low threshold. We can see those behaviors in the below graph.



<u>Resources :</u>

Youtube

Google

https://betterprogramming.pub/twitter-sentiment-analysis-using-naive-bayes-and-n-gram-5df42ae4bfc6

https://iq.opengenus.org/naive-bayes-on-tf-idf-vectorized-matrix/

Some online blogs

<u>Contribution</u> :

All the members are contributed equally.