INFORMATION RETRIEVAL(CSE508)

ASSIGNMENT 2

Group Member 1: Adarsh Singh Kushwah **RollNo:** MT21111
Group Member 2: Charisha Phirani **RollNo:** MT21117

Note: The supporting code is present in .pynb file

LIBRARIES USED:

1. **os**: for importing dataset folder from directory.

2. **nltk**: for performing operations like tokenization, stemming, detecting stop words etc.

3. **pandas**: for implementing dataFrames.

4. **string**: for working with text data present in dataset files.

5. **re**: Provides operations of detecting patterns with the help of regular expressions
6. **joblib**: Provides lightweighting pipeline
7. **icecream**: helps in printing debugging in a readable format.
8. **operator**: to perform various mathematical operations.

PREPROCESSING

DATA UPLOADING

1. Used google.colab to import files from the device

DATA UNDERSTANDING

1. Read data using os.listdir(“file location”).

2. Checked the total count of files in the folder. The number of files in the folder are 1133.

Ans 1a:

PRE-PROCESSING STEPS ON THE DATASET

1. Removed the special characters and ignored the ASCII characters.
2. Converted the document text to the lower case.
3. Removed stopwords from the textfiles and performed tokenization and lemmatization on the text.
4. Maintained a list of unique words.

METHODOLOGY

1. Performed the intersection of the document token and query token using a function in which 2 lists are passed, one for document tokens and the other for query tokens.
2. Similarly performed the union of the document token and query token using a function in which 2 list are passed, one for document tokens and the other for query tokens.
3. For calculating the jaccard's coefficient we divide the output of intersection by union obtained from the above steps.
4. For each document we calculate the jaccard's coefficient by taking the intersection and union with the query.
5. We return the documents which have top 5 values for jaccard's coefficient.
6. The ones with the high jaccard value are the most relevant documents to the query.

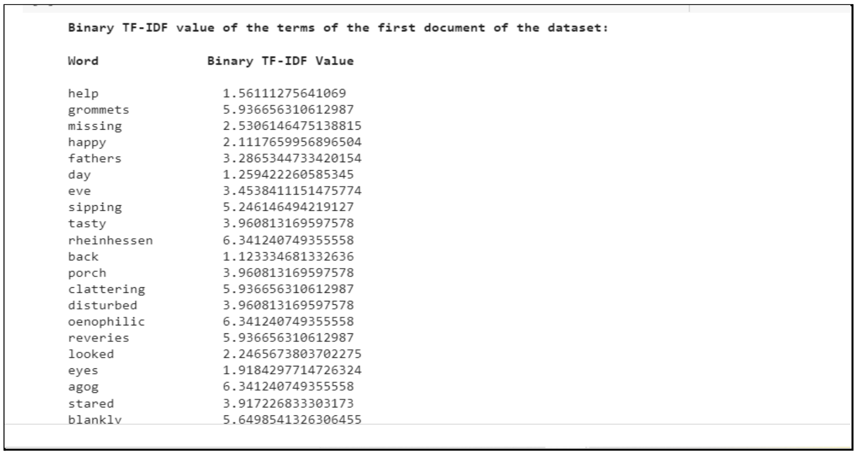
Ans 1b:

METHODOLOGY

1. Created a dictionary for storing the unique words.
2. Calculated the term frequency against each document and inverse document frequency.
3. For different weighting schemes, calculated the tf-idf score for each scheme.

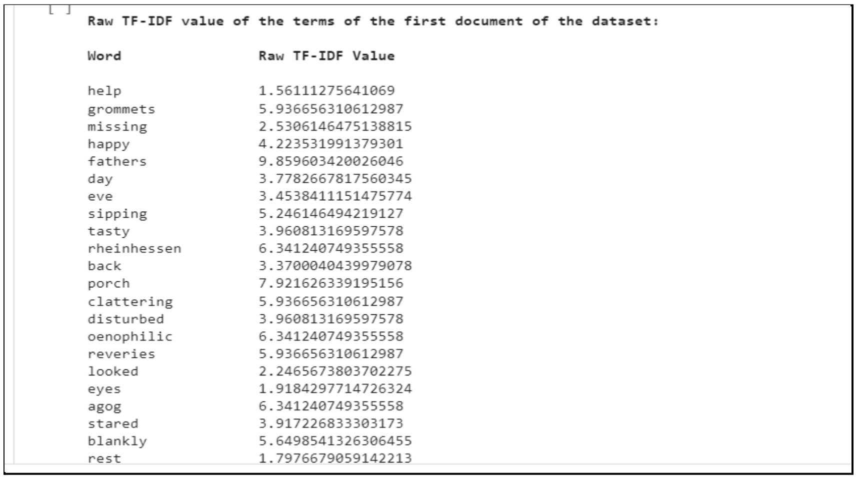
Results for Each Scheme

1. Binary TF-IDF



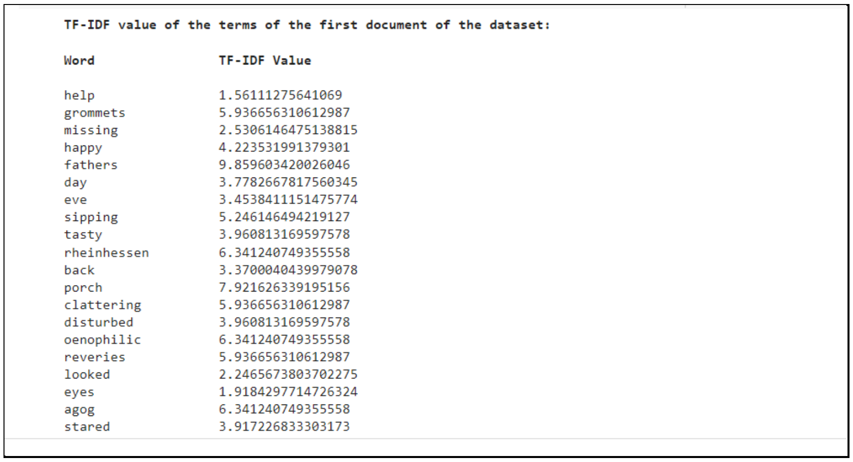
```
Binary TF-IDF value of the terms of the first document of the dataset:

Word              Binary TF-IDF Value

help                1.56111275641069
grommets            5.936656310612987
missing             2.5306146475138815
happy               2.1117659956896504
fathers             3.2865344733420154
day                 1.259422260585345
eve                 3.4538411151475774
sipping             5.246146494219127
tasty               3.960813169597578
rheinhessen         6.341240749355558
back                1.123334681332636
porch               3.960813169597578
clattering          5.936656310612987
disturbed           3.960813169597578
oenophilic          6.341240749355558
reveries            5.936656310612987
looked              2.2465673803702275
eyes                1.9184297714726324
agog                6.341240749355558
stared              3.917226833303173
blankly             5.6498541326306455
```

2. Raw Count



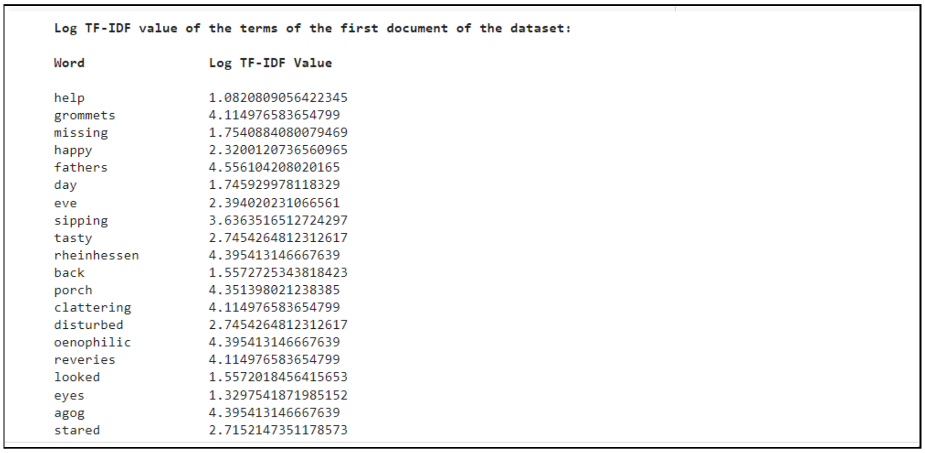
```
[ ]
  Raw TF-IDF value of the terms of the first document of the dataset:

  Word                 Raw TF-IDF Value

  help                 1.56111275641069
  grommets             5.936656310612987
  missing              2.5306146475138815
  happy                4.223531991379301
  fathers              9.859603420026046
  day                  3.7782667817560345
  eve                  3.4538411151475774
  sipping              5.246146494219127
  tasty                3.960813169597578
  rheinhessen          6.341240749355558
  back                 3.3700040439979078
  porch                7.921626339195156
  clattering           5.936656310612987
  disturbed            3.960813169597578
  oenophilic           6.341240749355558
  reveries             5.936656310612987
  looked               2.2465673803702275
  eyes                 1.9184297714726324
  agog                 6.341240749355558
  stared               3.917226833303173
  blankly              5.6498541326306455
  rest                 1.7976679059142213
```

3. Term frequency



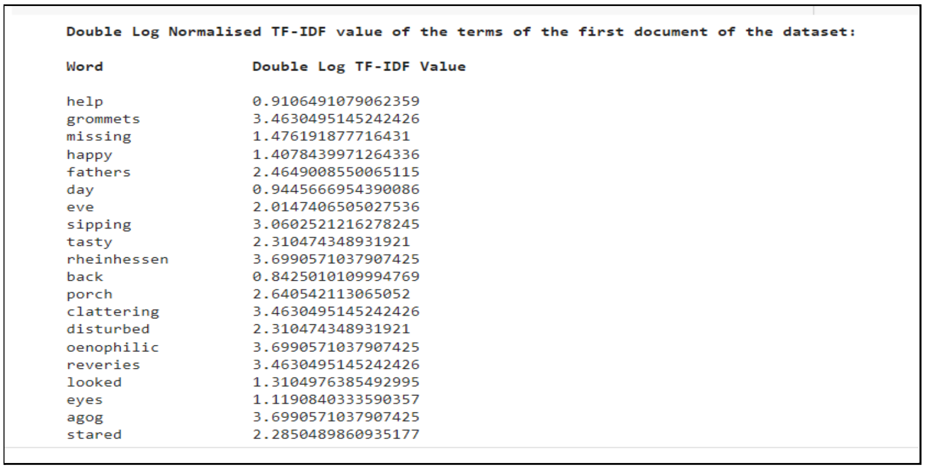
```
TF-IDF value of the terms of the first document of the dataset:

Word                TF-IDF Value

help                1.56111275641069
grommets            5.936656310612987
missing             2.5306146475138815
happy               4.223531991379301
fathers             9.859603420026046
day                 3.7782667817560345
eve                 3.4538411151475774
sipping             5.246146494219127
tasty               3.960813169597578
rheinhessen         6.341240749355558
back                3.3700040439979078
porch               7.921626339195156
clattering          5.936656310612987
disturbed           3.960813169597578
oenophilic          6.341240749355558
reveries            5.936656310612987
looked              2.2465673803702275
eyes                1.9184297714726324
agog                6.341240749355558
stared              3.917226833303173
```

4. Log Normalization

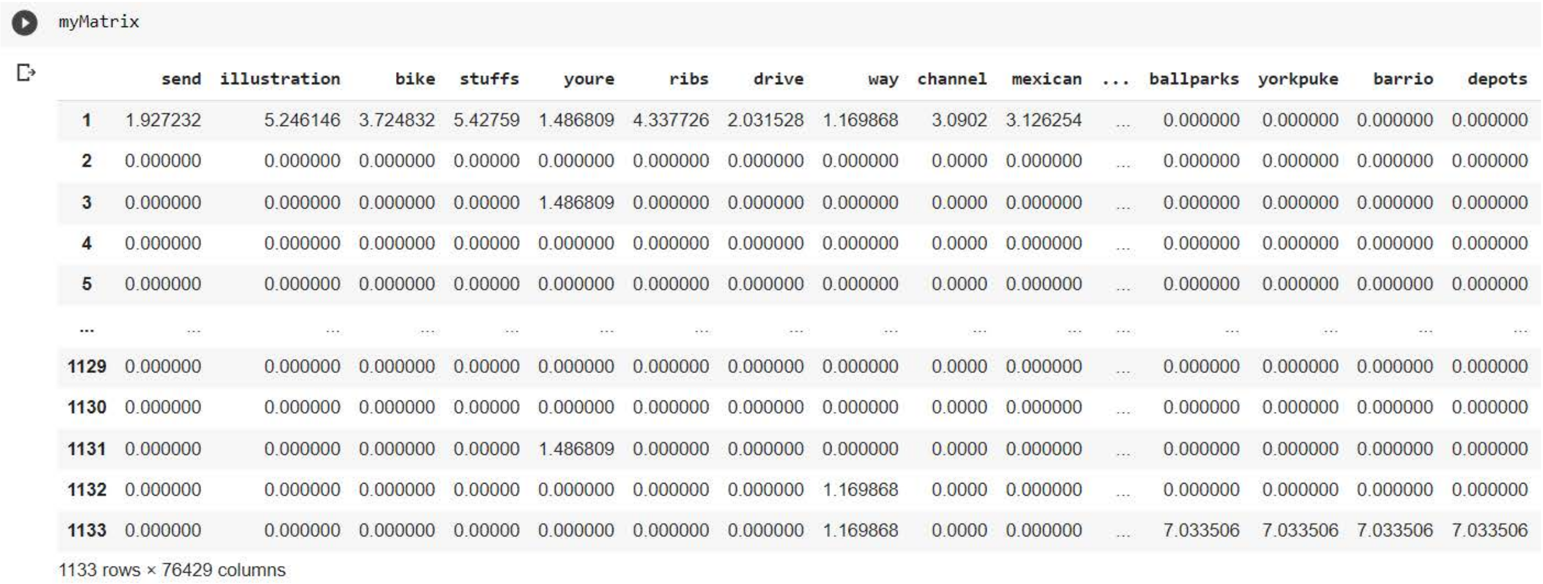


```
Log TF-IDF value of the terms of the first document of the dataset:

Word                Log TF-IDF Value

help                1.0820809056422345
grommets            4.114976583654799
missing             1.7540884080079469
happy               2.3200120736560965
fathers             4.556104208020165
day                 1.745929978118329
eve                 2.394020231066561
sipping             3.6363516512724297
tasty               2.7454264812312617
rheinhessen         4.395413146667639
back                1.5572725343818423
porch               4.351398021238385
clattering          4.114976583654799
disturbed           2.7454264812312617
oenophilic          4.395413146667639
reveries            4.114976583654799
looked              1.5572018456415653
eyes                1.3297541871985152
agog                4.395413146667639
stared              2.7152147351178573
```

5. Double Normalisation



```
Double Log Normalised TF-IDF value of the terms of the first document of the dataset:

Word                    Double Log TF-IDF Value

help                    0.9106491079062359
grommets                3.4630495145242426
missing                 1.476191877716431
happy                   1.4078439971264336
fathers                 2.4649008550065115
day                     0.9445666954390086
eve                     2.0147406505027536
sipping                 3.0602521216278245
tasty                   2.310474348931921
rheinhessen             3.6990571037907425
back                    0.8425010109994769
porch                   2.640542113065052
clattering              3.4630495145242426
disturbed               2.310474348931921
oenophilic              3.6990571037907425
reveries                3.4630495145242426
looked                  1.3104976385492995
eyes                    1.1190840333590357
agog                    3.6990571037907425
stared                  2.2850489860935177
```

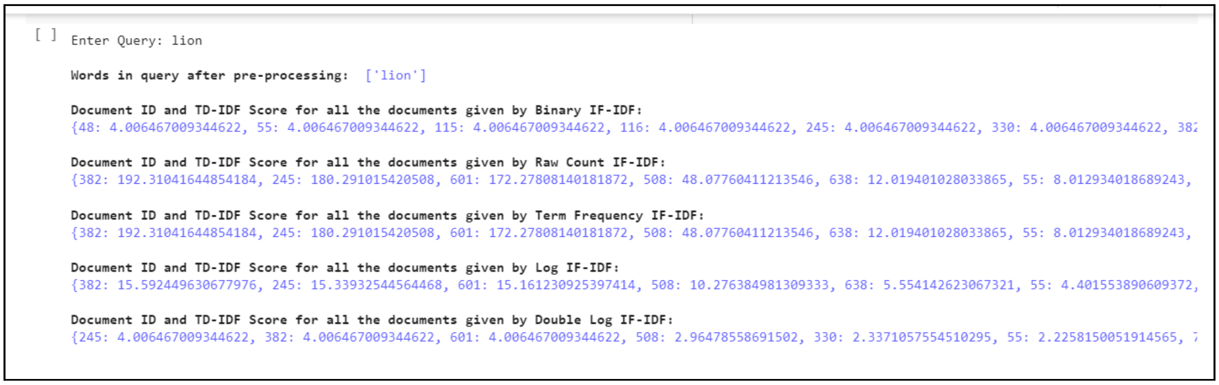


```
myMatrix
```

	send	illustration	bike	stuffs	youre	ribs	drive	way	channel	mexican	...	ballparks	yorkpuke	barrio	depots
1	1.927232	5.246146	3.724832	5.42759	1.486809	4.337726	2.031528	1.169868	3.0902	3.126254	...	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.00000	1.486809	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1129	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
1130	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
1131	0.000000	0.000000	0.000000	0.00000	1.486809	0.000000	0.000000	0.000000	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
1132	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	1.169868	0.0000	0.000000	...	0.000000	0.000000	0.000000	0.000000
1133	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	1.169868	0.0000	0.000000	...	7.033506	7.033506	7.033506	7.033506

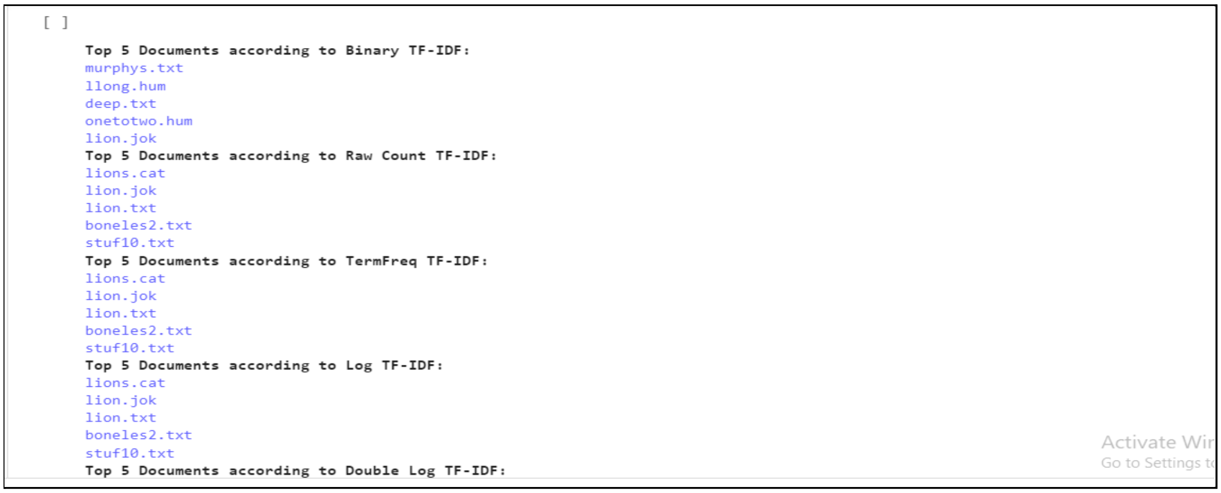
1133 rows × 76429 columns

THE QUERY ENTERED IS LION, THE TF-IDF SCORE OF EACH SCHEME IS:



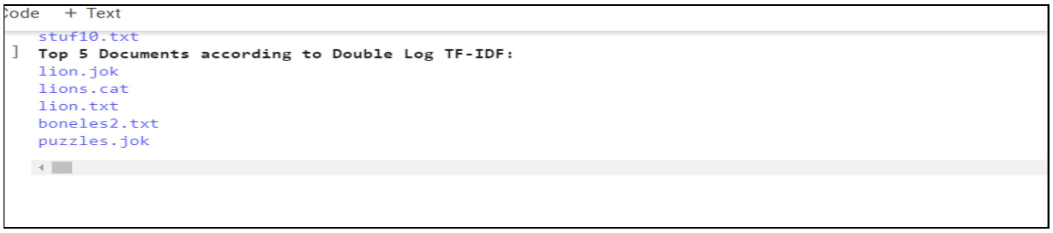
```
[ ]  Enter Query: lion

     Words in query after pre-processing:  ['lion']

     Document ID and TD-IDF Score for all the documents given by Binary IF-IDF:
     {48: 4.006467009344622, 55: 4.006467009344622, 115: 4.006467009344622, 116: 4.006467009344622, 245: 4.006467009344622, 330: 4.006467009344622, 382

     Document ID and TD-IDF Score for all the documents given by Raw Count IF-IDF:
     {382: 192.31041644854184, 245: 180.291015420508, 601: 172.27808140181872, 508: 48.07760411213546, 638: 12.019401028033865, 55: 8.012934018689243,

     Document ID and TD-IDF Score for all the documents given by Term Frequency IF-IDF:
     {382: 192.31041644854184, 245: 180.291015420508, 601: 172.27808140181872, 508: 48.07760411213546, 638: 12.019401028033865, 55: 8.012934018689243,

     Document ID and TD-IDF Score for all the documents given by Log IF-IDF:
     {382: 15.592449630677976, 245: 15.33932544564468, 601: 15.161230925397414, 508: 10.276384981309333, 638: 5.554142623067321, 55: 4.401553890609372,

     Document ID and TD-IDF Score for all the documents given by Double Log IF-IDF:
     {245: 4.006467009344622, 382: 4.006467009344622, 601: 4.006467009344622, 508: 2.96478558691502, 330: 2.3371057554510295, 55: 2.2258150051914565, 7
```

THE RESULTS FOR EACH SCHEME ARE:



```
[ ]
     Top 5 Documents according to Binary TF-IDF:
     murphys.txt
     llong.hum
     deep.txt
     onetotwo.hum
     lion.jok
     Top 5 Documents according to Raw Count TF-IDF:
     lions.cat
     lion.jok
     lion.txt
     boneles2.txt
     stuf10.txt
     Top 5 Documents according to TermFreq TF-IDF:
     lions.cat
     lion.jok
     lion.txt
     boneles2.txt
     stuf10.txt
     Top 5 Documents according to Log TF-IDF:
     lions.cat
     lion.jok
     lion.txt
     boneles2.txt
     stuf10.txt
     Top 5 Documents according to Double Log TF-IDF:
```

Activate Wir
Go to Settings t



```
  stuf10.txt
] Top 5 Documents according to Double Log TF-IDF:
  lion.jok
  lions.cat
  lion.txt
  boneles2.txt
  puzzles.jok
```

The pros and cons of each scoring scheme:

PROS OF JACCARD COEFFICIENT

1. For Jaccard calculation between two sets the 2 sets may/may not be of the same size.
2. It always outputs the value between 0 and 1.
3. In this categorical and continuous values can be used.

CONS OF JACCARD COEFFICIENT

1. It does not take care of how many times a term is occurring in a document i.e the term frequency.

2. Jaccard does not consider that terms occurring in a low frequency are more informative than terms occurring frequently.

PROS OF TF-IDF

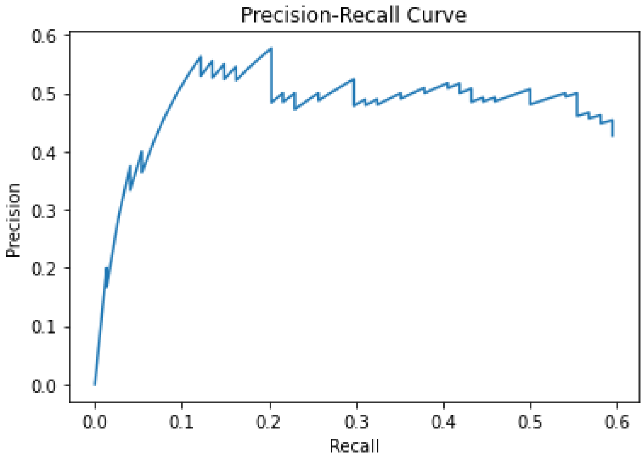
1. Easy for computing similarity between the 2 documents.
2. Helps to extract the most descriptive terms from the document.

CONS OF TF-IDF

1. This method is based on the Bag of Words model so it does not consider the ordering of words in the document.
2. Does not take care of semantics and position of the terms in the document.

Ans 2:

1. Read data using os.listdir(“file location”).
2. Stored the queries with the qid:4 in a dictionary and .txt file.
3. Rearranged the query-url pairs on the basis of max-dcg
4. Counted the total number of files.
5. Calculated the nDCG at 50 and for the whole dataset.
6. Plotted a precision recall curve on the basis of value of feature 75 on qid:4.



Ans 3:

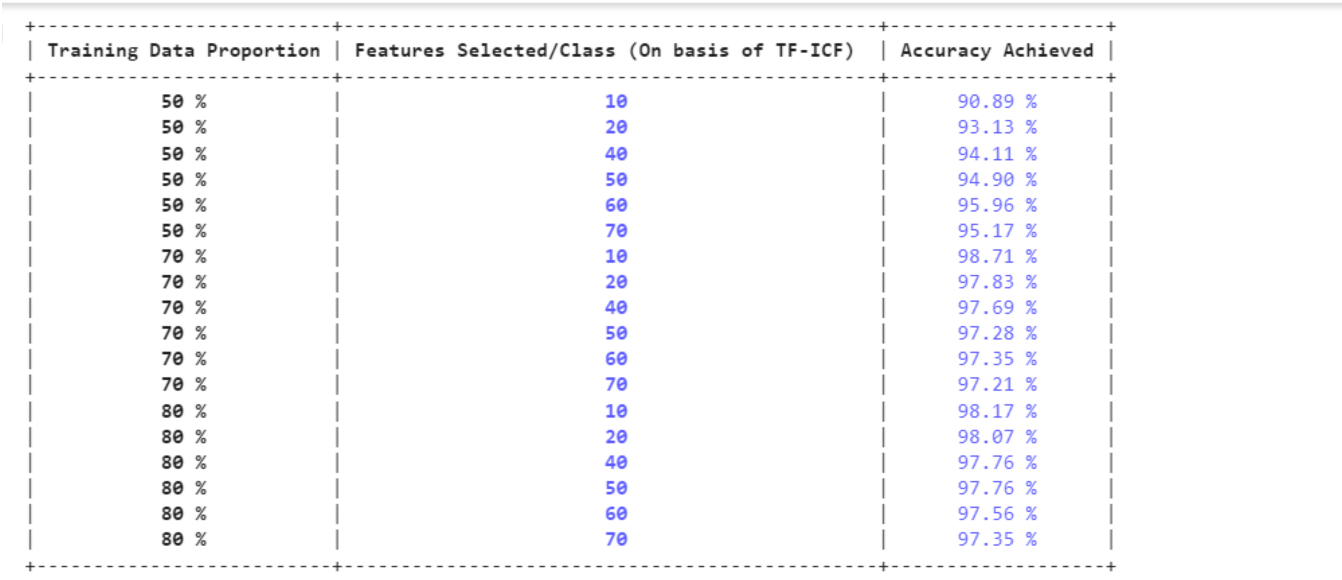
DATA PREPROCESSING

1. Stripped the data for each files
2. Converted the text to lower case
3. Tokenized the data and generated tokens
4. Converting the number into word like 12 -> twelve
5. Removed punctuation marks

6. Removed single alphabet terms
7. Removed stop words
8. Applied lemmatization

METHODOLOGY

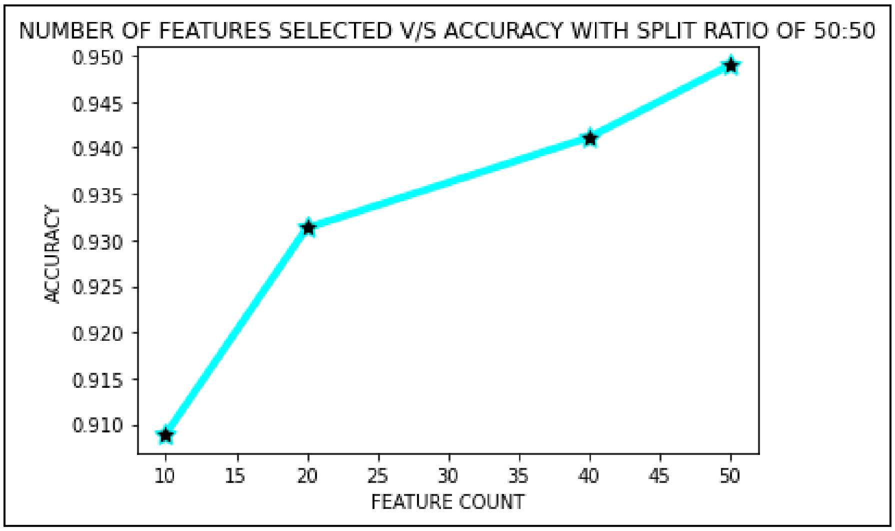
1. Created a dataframe containing the tokens in the corresponding files.
2. Created a class list for the files comp.graphics, sci.med, talk.politics.misc, rec.sport.hockey, sci.space respectively.
3. Counted the number of words and unique words in each of the 5 classes.
4. Also calculated the class frequency of each word..
5. Calculated the tf-icf value.
6. Found top k features using values of tf-icf.
7. Evaluated class frequencies.
8. Created function for Naive-Bayes algorithm and evaluation of accuracy and creating the confusion matri



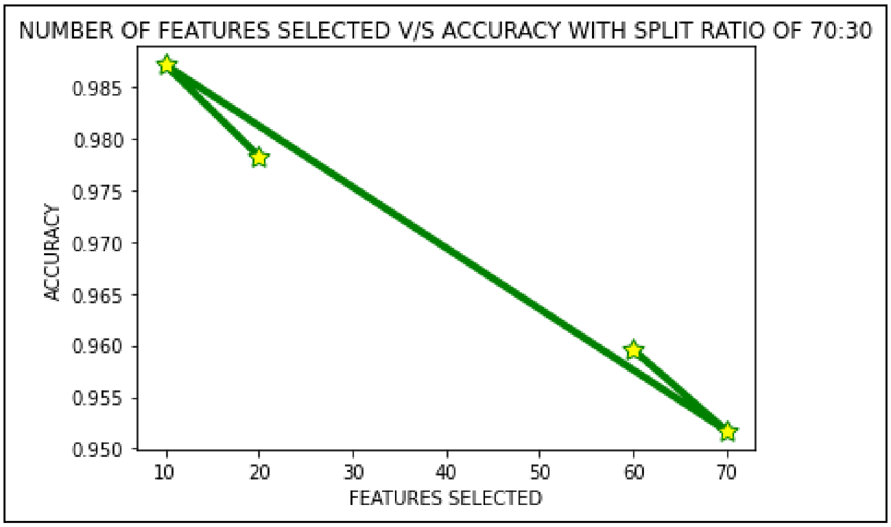
```
+--------------------------+---------------------------------------------+------------------+
| Training Data Proportion | Features Selected/Class (On basis of TF-ICF) | Accuracy Achieved |
+--------------------------+---------------------------------------------+------------------+
|           50 %           |                      10                     |      90.89 %     |
|           50 %           |                      20                     |      93.13 %     |
|           50 %           |                      40                     |      94.11 %     |
|           50 %           |                      50                     |      94.90 %     |
|           50 %           |                      60                     |      95.96 %     |
|           50 %           |                      70                     |      95.17 %     |
|           70 %           |                      10                     |      98.71 %     |
|           70 %           |                      20                     |      97.83 %     |
|           70 %           |                      40                     |      97.69 %     |
|           70 %           |                      50                     |      97.28 %     |
|           70 %           |                      60                     |      97.35 %     |
|           70 %           |                      70                     |      97.21 %     |
|           80 %           |                      10                     |      98.17 %     |
|           80 %           |                      20                     |      98.07 %     |
|           80 %           |                      40                     |      97.76 %     |
|           80 %           |                      50                     |      97.76 %     |
|           80 %           |                      60                     |      97.56 %     |
|           80 %           |                      70                     |      97.35 %     |
+--------------------------+---------------------------------------------+------------------+
```

9. The plots for accuracy vs feature for different train and test ratios are provided below.

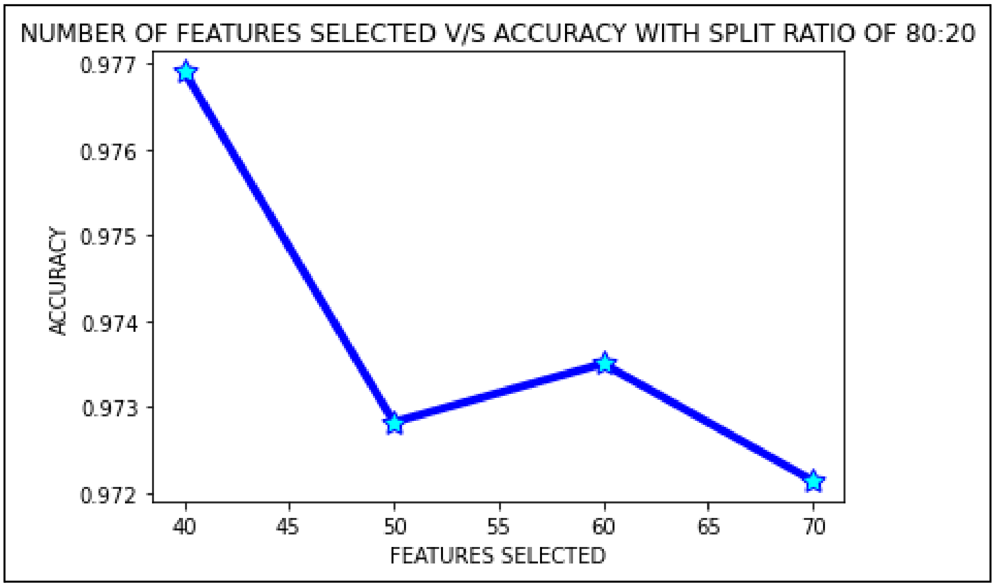
1. **50:50**



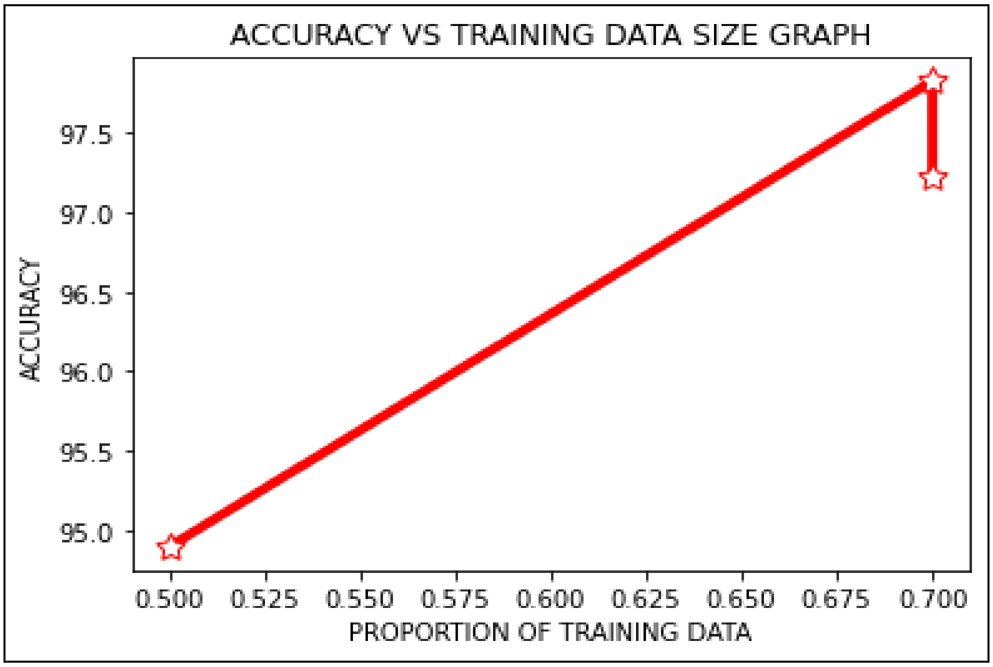
2. **70:30**



3. **80:20**



The graph for accuracy vs training is:


ACCURACY VS TRAINING DATA SIZE GRAPH
ACCURACY
PROPORTION OF TRAINING DATA