TF-IDF metric :

First run the pre-processing part by the running the main function

After that we need to convert TF value with using 5 different weight scheme by running the function

That will give you the total 11 metrics of the tf-idf scores

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 3:

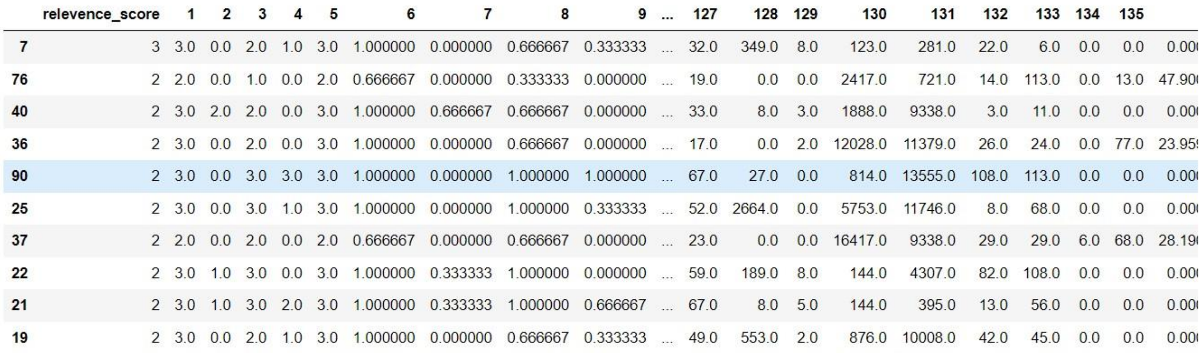
- After that we find the tf-icf matrix of the data using the class frequency and term frequency and then we store that value into the matrix of size of vocabulary and size of number of document.
- Finally we list the all the metrics that we store into the dictionary formate
- So this is the database that we created using tf-icf value of each term and after then we converted it into the dataftame for training and testing purpose
- After getting the dataframe using sklearn library we split the dataset into 2 part raining and testing in the ratio of 70:30
- 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.

- 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 :
- 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

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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00
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	0.00

The total files which have maximum DCS are

153888936228061547839646726334689624810616165627544663510135161092517848660517032615561282848582896884421691379147796060128992449679356826411301588967663067778 2528

Objective 2

Compute nDCG at 50

- 0.3521042740324887

Compute nDCG over the whole dataset

- 0.5979226516897831