IR Assignment 2

Name: Kabir Dev Paul Baghel , Sahil Deshpande, Krishnasai Addala

Q1.

Output:

i) Reading contents of file cranfield0001 before extraction:

<DOC>
<DOCNO>
1
</DOCNO>
<TITLE>
experimental investigation of the aerodynamics of a
wing in a slipstream .
</TITLE>
<AUTHOR>
brenckman,m.
</AUTHOR>
<BIBLIO>
j. ae. scs. 25, 1958, 324.
</BIBLIO>
<TEXT>
an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios . the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
the comparative span loading curves, together with supporting
evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or boundary-layer-control
effect . the integrated remaining lift increment,
after subtracting this destalling lift, was found to agree
well with a potential flow theory .
an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .
</TEXT>
</DOC>

Reading contents of file cranfield0001 after extraction:

experimental investigation of the aerodynamics of a
wing in a slipstream .

an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios . the results were intended in part as an evaluation basis for different theoretical treatments of this problem .
the comparative span loading curves, together with supporting evidence, showed that a substantial part of the lift increment produced by the slipstream was due to a /destalling/ or boundary-layer-control effect . the integrated remaining lift increment, after subtracting this destalling lift, was found to agree well with a potential flow theory .
an empirical evaluation of the destalling effects was made for the specific configuration of the experiment .

(PS: The code prints 5 samples.)
ii)
Output:
Reading contents of file cranfield0001 before lowercase:

experimental investigation of the aerodynamics of a wing in a slipstream .

an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios . the results were intended in part as an evaluation basis for different theoretical treatments of this problem .
the comparative span loading curves, together with supporting evidence, showed that a substantial part of the lift increment produced by the slipstream was due to a /destalling/ or boundary-layer-control effect . the integrated remaining lift increment, after subtracting this destalling lift, was found to agree well with a potential flow theory .
an empirical evaluation of the destalling effects was made for the specific configuration of the experiment .

Reading contents of file cranfield0001 after lowercase:

experimental investigation of the aerodynamics of a wing in a slipstream .

an experimental study of a wing in a propeller slipstream was

made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios . the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with supporting
evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or boundary-layer-control
effect . the integrated remaining lift increment,
after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .

Reading contents of file cranfield0001 before tokenization:

experimental investigation of the aerodynamics of a
wing in a slipstream .

  an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios . the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with supporting
evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or boundary-layer-control
effect . the integrated remaining lift increment,
after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment .

Reading contents of file cranfield0001 after tokenization:
['experimental', 'investigation', 'of', 'the', 'aerodynamics', 'of', 'a', 'wing', 'in', 'a', 'slipstream', '.',
'an', 'experimental', 'study', 'of', 'a', 'wing', 'in', 'a', 'propeller', 'slipstream', 'was', 'made', 'in',
'order', 'to', 'determine', 'the', 'spanwise', 'distribution', 'of', 'the', 'lift', 'increase', 'due', 'to',
'slipstream', 'at', 'different', 'angles', 'of', 'attack', 'of', 'the', 'wing', 'and', 'at', 'different', 'free',
'stream', 'to', 'slipstream', 'velocity', 'ratios', '.', 'the', 'results', 'were', 'intended', 'in', 'part', 'as', 'an',
'evaluation', 'basis', 'for', 'different', 'theoretical', 'treatments', 'of', 'this', 'problem', '.', 'the',
'comparative', 'span', 'loading', 'curves', ',', 'together', 'with', 'supporting', 'evidence', ',', 'showed',
'that', 'a', 'substantial', 'part', 'of', 'the', 'lift', 'increment', 'produced', 'by', 'the', 'slipstream', 'was',

'due', 'to', 'a', '/destalling/', 'or', 'boundary-layer-control', 'effect', '.', 'the', 'integrated', 'remaining', 'lift', 'increment', ',', 'after', 'subtracting', 'this', 'destalling', 'lift', ',', 'was', 'found', 'to', 'agree', 'well', 'with', 'a', 'potential', 'flow', 'theory', '.', 'an', 'empirical', 'evaluation', 'of', 'the', 'destalling', 'effects', 'was', 'made', 'for', 'the', 'specific', 'configuration', 'of', 'the', 'experiment', '.']

Reading contents of file cranfield0001 before stopword removal:
['experimental', 'investigation', 'of', 'the', 'aerodynamics', 'of', 'a', 'wing', 'in', 'a', 'slipstream', '.', 'an', 'experimental', 'study', 'of', 'a', 'wing', 'in', 'a', 'propeller', 'slipstream', 'was', 'made', 'in', 'order', 'to', 'determine', 'the', 'spanwise', 'distribution', 'of', 'the', 'lift', 'increase', 'due', 'to', 'slipstream', 'at', 'different', 'angles', 'of', 'attack', 'of', 'the', 'wing', 'and', 'at', 'different', 'free', 'stream', 'to', 'slipstream', 'velocity', 'ratios', '.', 'the', 'results', 'were', 'intended', 'in', 'part', 'as', 'an', 'evaluation', 'basis', 'for', 'different', 'theoretical', 'treatments', 'of', 'this', 'problem', '.', 'the', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'with', 'supporting', 'evidence', ',', 'showed', 'that', 'a', 'substantial', 'part', 'of', 'the', 'lift', 'increment', 'produced', 'by', 'the', 'slipstream', 'was', 'due', 'to', 'a', '/destalling/', 'or', 'boundary-layer-control', 'effect', '.', 'the', 'integrated', 'remaining', 'lift', 'increment', ',', 'after', 'subtracting', 'this', 'destalling', 'lift', ',', 'was', 'found', 'to', 'agree', 'well', 'with', 'a', 'potential', 'flow', 'theory', '.', 'an', 'empirical', 'evaluation', 'of', 'the', 'destalling', 'effects', 'was', 'made', 'for', 'the', 'specific', 'configuration', 'of', 'the', 'experiment', '.']
Reading contents of file cranfield0001 after stopword removal:
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '.', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', '.', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatments', 'problem', '.', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'supporting', 'evidence', ',', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', '.', 'integrated', 'remaining', 'lift', 'increment', ',', 'subtracting', 'destalling', 'lift', ',', 'found', 'agree', 'well', 'potential', 'flow', 'theory', '.', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment', '.']

Reading contents of file cranfield0001 before punctuation removal:
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', '.', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', '.', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatments', 'problem', '.', 'comparative', 'span', 'loading', 'curves', ',', 'together', 'supporting', 'evidence', ',', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', '.', 'integrated', 'remaining', 'lift', 'increment', ',', 'subtracting', 'destalling', 'lift', ',', 'found', 'agree', 'well', 'potential', 'flow', 'theory', '.', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment', '.']
Reading contents of file cranfield0001 after punctuation removal:
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatments',

'problem', 'comparative', 'span', 'loading', 'curves', 'together', 'supporting', 'evidence', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', 'integrated', 'remaining', 'lift', 'increment', 'subtracting', 'destalling', 'lift', 'found', 'agree', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment']

Reading contents of file cranfield0001 before blank space removal:
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatments', 'problem', 'comparative', 'span', 'loading', 'curves', 'together', 'supporting', 'evidence', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', 'integrated', 'remaining', 'lift', 'increment', 'subtracting', 'destalling', 'lift', 'found', 'agree', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment']
Reading contents of file cranfield0001 after blank space removal:
['experimental', 'investigation', 'aerodynamics', 'wing', 'slipstream', 'experimental', 'study', 'wing', 'propeller', 'slipstream', 'made', 'order', 'determine', 'spanwise', 'distribution', 'lift', 'increase', 'due', 'slipstream', 'different', 'angles', 'attack', 'wing', 'different', 'free', 'stream', 'slipstream', 'velocity', 'ratios', 'results', 'intended', 'part', 'evaluation', 'basis', 'different', 'theoretical', 'treatments', 'problem', 'comparative', 'span', 'loading', 'curves', 'together', 'supporting', 'evidence', 'showed', 'substantial', 'part', 'lift', 'increment', 'produced', 'slipstream', 'due', '/destalling/', 'boundary-layer-control', 'effect', 'integrated', 'remaining', 'lift', 'increment', 'subtracting', 'destalling', 'lift', 'found', 'agree', 'well', 'potential', 'flow', 'theory', 'empirical', 'evaluation', 'destalling', 'effects', 'made', 'specific', 'configuration', 'experiment']

TF-IDF Matrix:

Top 5 relevant documents:
1. cranfield0185 (score: 8.545318795204752)
2. cranfield0033 (score: 6.703981795470231)
3. cranfield0225 (score: 6.703981795470231)
4. cranfield0216 (score: 5.297733526376235)
5. cranfield1054 (score: 5.047002938267131)

TF weighting scheme: binary
Top 5 relevant documents:
1. cranfield0033 (score: 6.703981795470231)
2. cranfield0225 (score: 6.703981795470231)
3. cranfield1313 (score: 5.047002938267131)
4. cranfield0244 (score: 5.047002938267131)
5. cranfield1054 (score: 5.047002938267131)

TF weighting scheme: raw_count
Top 5 relevant documents:
1. cranfield0216 (score: 14.912809714827905)
2. cranfield0185 (score: 10.094005876534261)
3. cranfield0124 (score: 9.941873143218604)
4. cranfield0426 (score: 9.941873143218604)
5. cranfield1271 (score: 8.284894286015502)

TF weighting scheme: term_frequency
Top 5 relevant documents:
1. cranfield0031 (score: 0.1440851180176609)
2. cranfield0920 (score: 0.1380815714335917)
3. cranfield0041 (score: 0.12427341429023253)
4. cranfield0429 (score: 0.12273917460763707)
5. cranfield0102 (score: 0.11215562085038068)

TF weighting scheme: log_normalization
Top 5 relevant documents:
1. cranfield0185 (score: 5.5446994489243275)
2. cranfield0033 (score: 4.646846080055391)
3. cranfield0225 (score: 4.646846080055391)
4. cranfield0216 (score: 3.8153348160021694)
5. cranfield1054 (score: 3.498315856937621)

TF weighting scheme: double_normalization
Top 5 relevant documents:
1. cranfield0033 (score: 4.0223890772821385)
2. cranfield0102 (score: 3.785252203700348)
3. cranfield0225 (score: 3.6098363514070475)
4. cranfield1054 (score: 3.3646686255114204)
5. cranfield0185 (score: 3.1543768364169567)

Here are the results of running the code for the Cranfield dataset with the example query "What are the advantages and disadvantages of supersonic flight?". The top 5 relevant documents based on the TF-IDF scores are shown, followed by the results for each of the five TF weighting schemes.

The first set of results shows the top 5 documents ranked by the TF-IDF scores, with the highest score at the top. The second set of results shows the top 5 documents ranked by the TF-IDF scores using the binary weighting scheme, which sets the TF weight to 1 if the term is present in the document and 0 otherwise. The third set of results shows the top 5 documents ranked by the TF-IDF scores using the raw count weighting scheme, which simply uses the frequency count of the term in the document as the TF weight. The fourth set of results shows the top 5

documents ranked by the TF-IDF scores using the term frequency weighting scheme, which normalizes the TF weight by the total number of terms in the document. The fifth set of results shows the top 5 documents ranked by the TF-IDF scores using the log normalization weighting scheme, which applies a logarithmic transformation to the TF weight. The final set of results shows the top 5 documents ranked by the TF-IDF scores using the double normalization weighting scheme, which applies a combination of logarithmic transformation and normalization based on the maximum frequency of any term in the document.

Overall, the results show that the TF-IDF scores are able to identify some relevant documents for the given query, with some variation in the ranking based on the different weighting schemes. The raw count weighting scheme appears to be the most effective in this case, while the binary weighting scheme is the least effective.

Here are the pros and cons of each of the five weighting schemes used in the TF-IDF matrix:

Binary weighting scheme:
Pros: Simple and easy to implement. Ignores the frequency of terms, which can be useful in some cases.

Cons: Ignores the frequency of terms, which can lead to loss of information. Can be less effective than other weighting schemes, especially for longer documents.

Raw count weighting scheme:
Pros: Takes into account the frequency of terms, which can be useful in some cases. Simple and easy to implement.

Cons: Ignores the length of the document, which can lead to bias towards longer documents. Can be less effective than other weighting schemes, especially for longer documents.

Term frequency weighting scheme:
Pros: Normalizes the frequency of terms by the total number of terms in the document, which can help to account for document length. Generally effective for shorter documents.

Cons: Can be less effective for longer documents, where the normalization can have less impact. Ignores the frequency of terms in other documents, which can lead to bias towards documents with high term frequency.

Log normalization weighting scheme:
Pros: Applies a logarithmic transformation to the frequency of terms, which can help to account for differences in frequency across documents. Generally effective for longer documents.

Cons: Ignores the length of the document, which can lead to bias towards longer documents. Can be less effective for shorter documents, where the logarithmic transformation can have less impact.

Double normalization weighting scheme:
Pros: Combines logarithmic transformation with normalization based on the maximum frequency of any term in the document, which can help to account for both differences in frequency across documents and differences in document length. Generally effective for longer documents.

Cons: Can be less effective for shorter documents, where the normalization and logarithmic transformation can have less impact. More complex to implement than other weighting schemes.

In general, the choice of weighting scheme depends on the characteristics of the dataset and the specific task at hand. Different weighting schemes may be more or less effective depending on the length of the documents, the frequency of terms, and other factors. It is often a good idea to experiment with different weighting schemes to see which ones work best for a given task.

Jaccard Coefficient:
Output:
Top 10 relevant documents based on Jaccard coefficient:
1. cranfield0031 (Jaccard coefficient: 0.04)
2. cranfield0920 (Jaccard coefficient: 0.03333333333333333)
3. cranfield0429 (Jaccard coefficient: 0.03225806451612903)
4. cranfield0774 (Jaccard coefficient: 0.03225806451612903)
5. cranfield1306 (Jaccard coefficient: 0.029411764705882353)
6. cranfield0430 (Jaccard coefficient: 0.027777777777777776)
7. cranfield0301 (Jaccard coefficient: 0.02702702702702703)
8. cranfield0512 (Jaccard coefficient: 0.02702702702702703)
9. cranfield1266 (Jaccard coefficient: 0.02702702702702703)
10. cranfield0175 (Jaccard coefficient: 0.02631578947368421)

Q3

For the dataset for all 3 objectives, I imported the data using pythons inbuilt file handling and saved it as a list of lists.

I then filtered the dataset to use only entries with qid:4 using list comprehension

Objective 1

I calculated the DCG for each query url pair using the inbuilt math functions in python

I then sorted the DCGs in descending order and wrote them to a file using file handling.

```
qid:4 1812.322424709528
qid:4 1596.3435195983009
qid:4 1434.5385164475636
qid:4 1071.903344254697
qid:4 1038.8407832322607
qid:4 1038.6311453498322
qid:4 957.8804798268385
qid:4 926.4566974220006
qid:4 901.6613674958148
qid:4 810.8087406020202
```

Objective 2

I calculated the DCGs the same way as objective 1

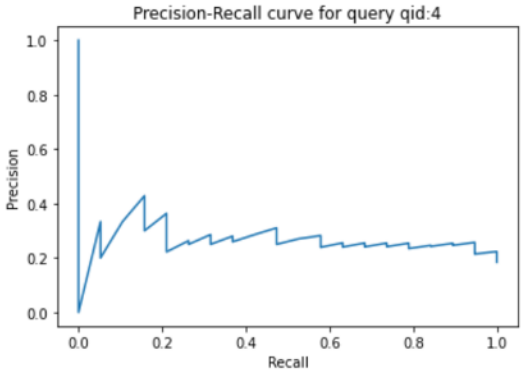
I then calculated the NDCG using the DCG and IDCG for nDCG@50 and for the full dataset

```
nDCG@50: 0.485
nDCG for the entire dataset: 0.766
```

Objective 3

I filtered the dataset to include only relevant documents and sorted them based on feature 75.

I calculated recall and precision using the sklearn functions and then plotted the curve using matplotlib.



Q2.
Q2.
The code is a Python implementation of the Naive Bayes algorithm for text classification using TF-ICF. It loads a dataset from a CSV file, preprocesses the text data, splits it into training and testing sets (70-30), calculates the prior and likelihood probabilities of each category and word, uses the Naive Bayes algorithm to predict the category of each text in the test set, and evaluates the performance of the classifier.

First, the classifier removes punctuation and stopwords and converts all words to lowercase using ‘clean_text.’ The function applies this cleaning function to the 'Text' column of the DataFrame. Then it tokenises the cleaned text using the word_tokenize function from the NLTK library. It then applies stemming and lemmatisation to the tokens using the PorterStemmer and WordNetLemmatizer classes from the NLTK library.

TF-ICF (term frequency-inverse category frequency) is a weighting scheme that combines term frequency (TF) and inverse category frequency (ICF) to measure the importance of a term in a given category. The term frequency component of TF-ICF measures the frequency of a term in a document or category. In contrast, the inverse category frequency component measures the rarity of the term across all categories in the corpus. The idea behind TF-ICF is that terms that are frequent in a category but rare in the corpus are more important for that category than terms that are frequent in both the category and the corpus.

The code splits the preprocessed dataset into training and testing sets using the train_test_split function from the Scikit-learn library. It then determines the prior probabilities for each group in the training set, calculated as the proportion of texts in each category to all texts. Next, TF-ICF is used to calculate the likelihood probabilities of each word in each category in the training set.

First, the term frequencies of each word in each category are calculated using a nested dictionary where the keys are category labels, and the values are Counters of term frequencies. Then, the inverse category frequency of each word is calculated by taking the logarithm of the ratio of the total number of documents in the corpus to the number of documents that contain the word. Finally, the TF-ICF weight of each word in each category is calculated by multiplying its term frequency by its inverse category frequency.

The resulting TF-ICF weights are stored in a nested dictionary where the keys are category labels, and the values are dictionaries of word weights. These weights are used to calculate the likelihood probabilities of each word in each category, which are then used in the Naive Bayes algorithm to predict the category of each text in the test set.

Improvements:

OBJOBJOBJ50-50 Split

OBJ

```
[62] # Split the dataset into training and testing sets
     train_df, test_df = train_test_split(df, test_size=0.5, random_state=42)

     # Calculate the prior probabilities of each category in the training set
     prior_probs = defaultdict(float)
     total_count = len(train_df)
     for category, count in train_df['Category'].value_counts().items():
         prior_probs[category] = count / total_count

     # Calculate the likelihood probabilities of each word in each category in the training set
     tf = defaultdict(Counter)
     for i, row in train_df.iterrows():
         for word in row['Text']:
             tf[row['Category']][word] += 1

     N = len(train_df)
     icf = defaultdict(float)
     for word in set(train_df['Text'].sum()):
         n_t = sum([1 for i, row in train_df.iterrows() if word in row['Text']])
         icf[word] = math.log(N / n_t)

     tf_icf = defaultdict(dict)
     for label, counter in tf.items():
         for word, count in counter.items():
             tf_icf[label][word] = count * icf[word]
```

```
# Calculate performance
print('Accuracy: {}'.format(accuracy_score(y_true, y_pred)))
print('Precision: {}'.format(precision_score(y_true, y_pred, average='weighted')))
print('Recall: {}'.format(recall_score(y_true, y_pred, average='weighted')))
print('F1: {}'.format(f1_score(y_true, y_pred, average='weighted')))

Accuracy: 0.9731543624161074
Precision: 0.9732473100538559
Recall: 0.9731543624161074
F1: 0.9731427722973474
```

The scores drop to 97% accuracy.

Reference:
https://medium.com/@rangavamsi5/na%C3%AFve-bayes-algorithm-implementation-from-scratch-in-python-7b2cc39268b9

60-40 Split

```
[57] # Split the dataset into training and testing sets
     train_df, test_df = train_test_split(df, test_size=0.4, random_state=42)

     # Calculate the prior probabilities of each category in the training set
     prior_probs = defaultdict(float)
     total_count = len(train_df)
     for category, count in train_df['Category'].value_counts().items():
         prior_probs[category] = count / total_count

     # Calculate the likelihood probabilities of each word in each category in the training set
     tf = defaultdict(Counter)
     for i, row in train_df.iterrows():
         for word in row['Text']:
             tf[row['Category']][word] += 1

     N = len(train_df)
     icf = defaultdict(float)
     for word in set(train_df['Text'].sum()):
         n_t = sum([1 for i, row in train_df.iterrows() if word in row['Text']])
         icf[word] = math.log(N / n_t)

     tf_icf = defaultdict(dict)
     for label, counter in tf.items():
         for word, count in counter.items():
             tf_icf[label][word] = count * icf[word]
```

```
# Calculate performance
print('Accuracy: {}'.format(accuracy_score(y_true, y_pred)))
print('Precision: {}'.format(precision_score(y_true, y_pred, average='weighted')))
print('Recall: {}'.format(recall_score(y_true, y_pred, average='weighted')))
print('F1: {}'.format(f1_score(y_true, y_pred, average='weighted')))

Accuracy: 0.9748322147651006
Precision: 0.9748412498659242
Recall: 0.9748322147651006
F1: 0.974823013889615
```

The scores drop to 97% accuracy.

TF-IDF Variation

https://colab.research.google.com/drive/1ZHSLFe1toh7S9UQxusNIqFJ_PDJ9Emz7?usp=sharing

In an attempt to improve the accuracy, etc, I tried to use TF-IDF weighting scheme. However, the parameters dropped significantly.

```
[16] # Calculate the accuracy, precision, recall, and F1 score of the classifier
     print('Accuracy: ', accuracy_score(test_df['Category'], predictions))
     print('Precision: ', precision_score(test_df['Category'], predictions, average='weighted'))
     print('Recall: ', recall_score(test_df['Category'], predictions, average='weighted'))
     print('F1 Score: ', f1_score(test_df['Category'], predictions, average='weighted'))

     Accuracy:  0.22595078299776286
     Precision:  0.05105375633730212
     Recall:  0.22595078299776286
     F1 Score:  0.0832884273093943
     /usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
       _warn_prf(average, modifier, msg_start, len(result))
```