JNAN VIKAS MANDAL'S

PADMASHREE DR. R.T.DOSHI DEGREE COLLEGE OF
INFORMATION TECHNOLOGY
MOHANLAL RAICHAND MEHTA COLLEGE OF COMMERCE
DIWALIMAA DEGREE COLLEGE OF SCIENCE
AMRATLAL RAICHAND MEHTA COLLEGE OF ARTS
JVM'S DEGREE COLLEGE OF INFORMATION TECHNOLOGY
AIROLI, NAVI MUMBAI – 400708
NAAC Reaccredited Grade 'A+' (CGPA- 3.31, 3rd Cycle)

CERTIFICATE

This is to certify that the **Mr.______________________** of T.Y.B.Sc.CS Semester-VI has completed the practical work in the subject of **Information Retrieval** during the Academic year **2024-25** under the guidance of **PROF.SARITA SARANG** being the partial requirement for the fulfillment of the curriculum of Degree of Bachelor of Science in Computer Science, University of Mumbai.

Place: **Date:**

--- ---
Sign of Subject In Charge Sign of External Examiner

Sign of Incharge / H.O.D

INDEX

Sr.No.	Name of Practicals	Date	Signature
1	Document Indexing and Retrieval • Implement an inverted index construction algorithm. • Build a simple document retrieval system using the constructed index.	08/01/2025	
2	Retrieval Models • Implement the Boolean retrieval model and process queries. • Implement the vector space model with TF-IDF weighting and cosine similarity	15/01/2025	
3	Spelling Correction in IR Systems • Develop a spelling correction module using edit distance algorithms. • Integrate the spelling correction module into an information retrieval system.	22/01/2025	
4	Evaluation Metrics for IR Systems • Calculate precision, recall, and F-measure for a given set of retrieval results. • Use an evaluation toolkit to measure average precision and other evaluation metrics.	22/01/2025	
5	Text Categorization • Implement a text classification algorithm (e.g., Naive Bayes or Support Vector Machines). • Train the classifier on a labelled dataset and evaluate its performance.	29/01/2025	
6	Clustering for Information Retrieval • Implement a clustering algorithm (e.g., K-means or hierarchical clustering). • Apply the clustering algorithm to a set of documents and evaluate the clustering results.	05/02/2025	
7	Web Crawling and Indexing • Develop a web crawler to fetch and index web pages. • Handle challenges such as robots.txt, dynamic content, and crawling delays.	12/02/2025	
8	Link Analysis and PageRank • Implement the PageRank algorithm to rank web pages based on link analysis. • Apply the PageRank algorithm to a small web graph and analyze the results.	05/03/2025	

Practical No: 1

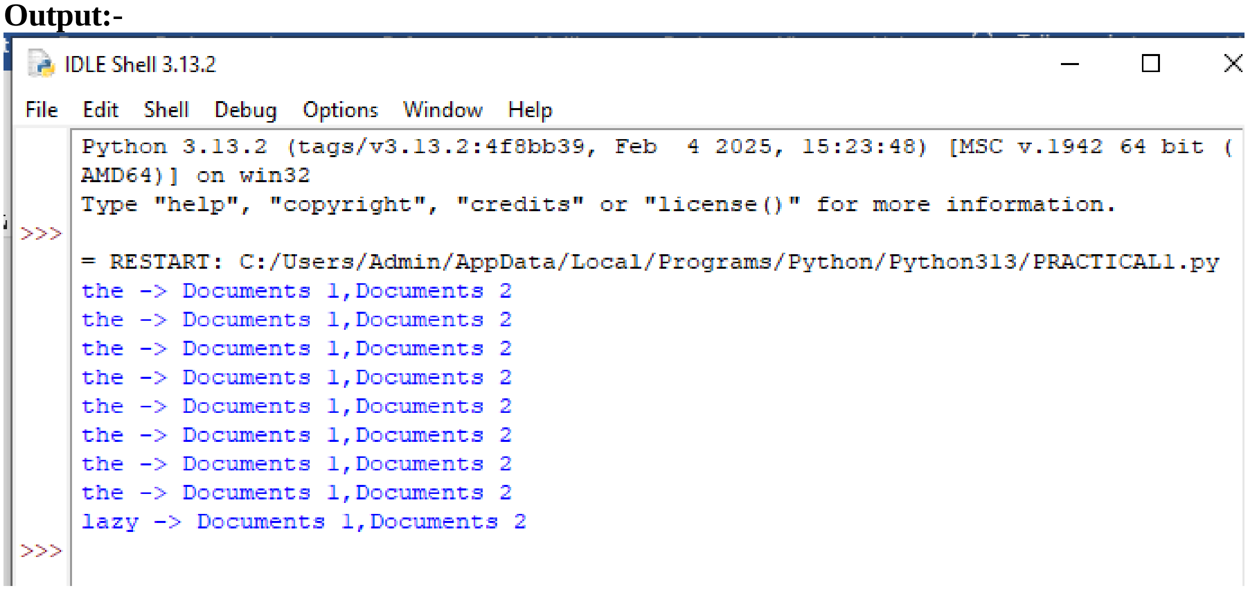
Aim: Document Indexing and Retrieval

- **Implement an inverted index construction algorithm.**
- **Build a simple document retrieval system using the constructed index.**

A:

```
document1 ="The quick brown fox jumped over the lazy dog."
document2 = "The lazy dog slept in the sun."
tokens1 = document1.lower().split()
tokens2 = document2.lower().split()
terms = list(set(tokens1 + tokens2))
inverted_index = {}
for term in terms:
    documents = []
    if term in tokens1:
        documents.append("Documents 1")
        if term in tokens2:
            documents.append("Documents 2")
            inverted_index[term] = documents
        for term, documents in inverted_index.items():
            print(term, "->",",".join(documents))
```

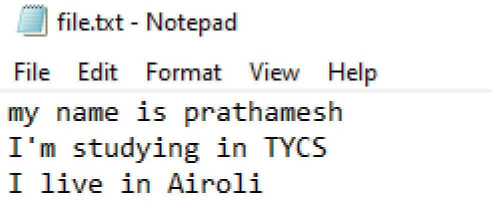
Output:-



```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python313/PRACTICAL1.py
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
the -> Documents 1,Documents 2
lazy -> Documents 1,Documents 2
>>>
```

B:

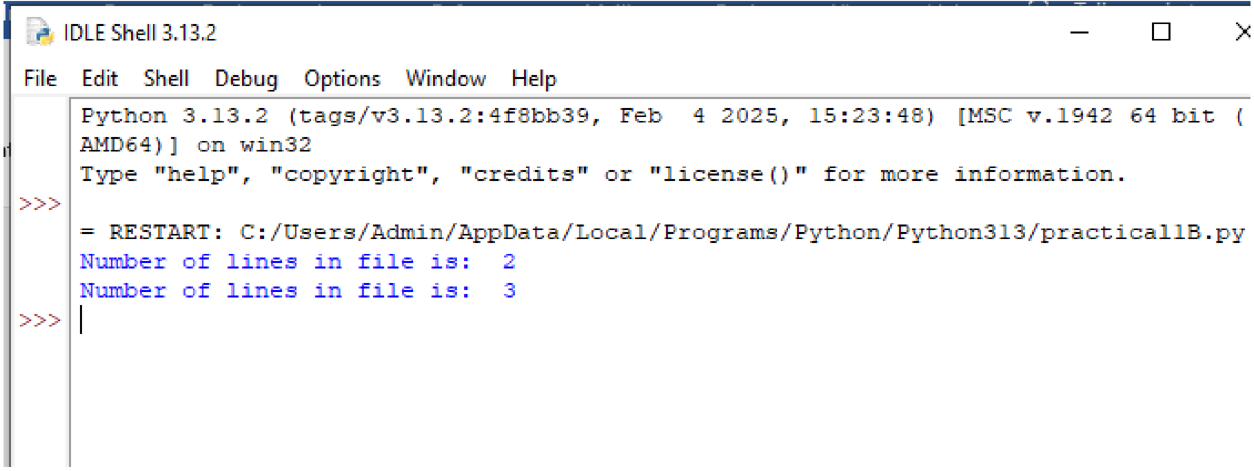
File.txt



```
my name is prathamesh
I'm studying in TYCS
I live in Airoli
```

```
# this will open the file
file = open('D:/file.txt', encoding='utf8')
read = file.read()
file.seek(0)
read
# to obtain the
# number of lines
# in file
line = 1
for word in read:
    if word == '\n':
        line += 1
        print("Number of lines in file is: ", line)
# create a list to
# store each line as
# an element of list
array = []
for i in range(line):
    array.append(file.readline())
```

Output:-

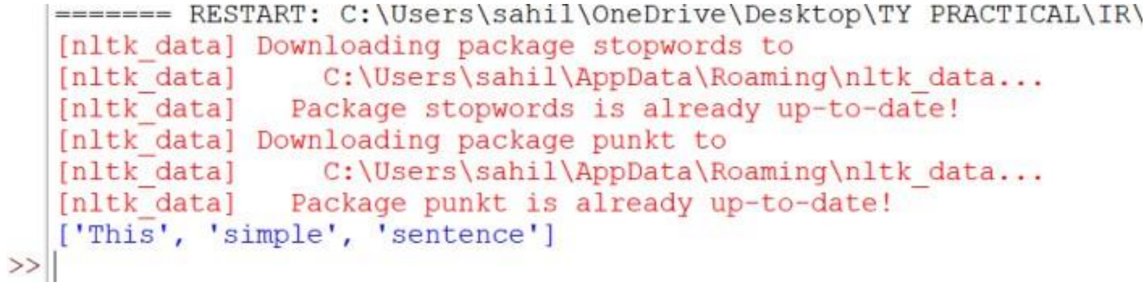


```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python313/practical1B.py
Number of lines in file is:  2
Number of lines in file is:  3
>>> |
```

C:

```
 from nltk.tokenize import word_tokenize
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
for i in range(1):
    read="This is a simple sentence"
    text_tokens = word_tokenize(read)
    tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
    print(tokens_without_sw)
```

Output:-



```
======= RESTART: C:\Users\sahil\OneDrive\Desktop\TY PRACTICAL\IR\
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sahil\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\sahil\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['This', 'simple', 'sentence']
>>
```

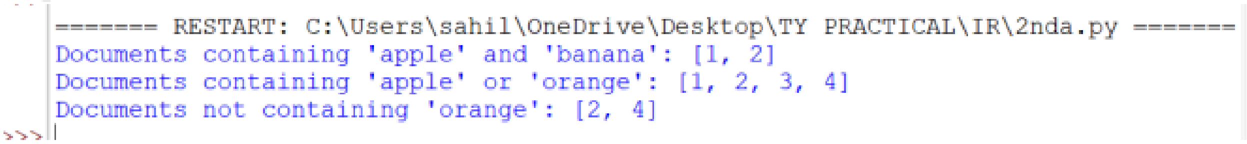
Practical No: 2

Aim: Retrieval Models

A)

```
documents = {
    1: "apple banana orange",
    2: "apple banana",
    3: "banana orange",
    4: "apple"
    }
def build_index(docs):
    index = {}
    for doc_id, text in docs.items():
        terms = set(text.split())
        for term in terms:
            if term not in index:
                index[term] = {doc_id} # Set with document ID
                else:
                    index[term].add(doc_id) # Add to existing set
                    return index
                inverted_index = build_index(documents)
                def boolean_and(operands, index):
                    if not operands:
                        return list(range(1, len(documents) + 1))
                    result = index.get(operands[0], set()) # First operand
                    for term in operands[1:]:
                        result = result.intersection(index.get(term, set())) # Intersection with sets of document IDs
                        return list(result)
                    def boolean_or(operands, index):
                        result = set()
                        for term in operands:
                            result = result.union(index.get(term, set()))
                            return list(result)
                        def boolean_not(operand, index, total_docs):
                            operand_set = set(index.get(operand, set())) # Set for the operand
                            all_docs_set = set(range(1, total_docs + 1)) # IDs for all documents
                            return list(all_docs_set.difference(operand_set)) # Return documents not in the operand set
                        # Example queries
                        query1 = ["apple", "banana"] # Query for documents containing both "apple"and "banana"
                        query2 = ["apple", "orange"] # Query for documents containing "apple" or"orange"
                        # Performing Boolean Model queries using inverted index
                        result1 = boolean_and(query1, inverted_index) # Get documents containing both terms
                        result2 = boolean_or(query2, inverted_index) # Get documents containing either
of the terms
                        result3 = boolean_not("orange", inverted_index, len(documents)) # Getdocuments
not containing "orange"
                        # Printing results
                        print("Documents containing 'apple' and 'banana':", result1)
                        print("Documents containing 'apple' or 'orange':", result2)
                        print("Documents not containing 'orange':", result3)
```

Output:-

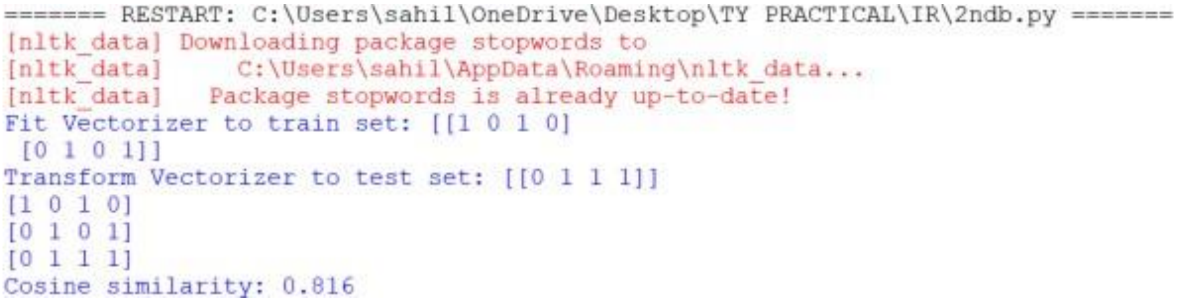


```
======= RESTART: C:\Users\sahil\OneDrive\Desktop\TY PRACTICAL\IR\2nda.py =======
Documents containing 'apple' and 'banana': [1, 2]
Documents containing 'apple' or 'orange': [1, 2, 3, 4]
Documents not containing 'orange': [2, 4]
>>>
```

B)

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
import nltk
from nltk.corpus import stopwords
import numpy as np
from numpy.linalg import norm
train_set = ["The sky is blue.", "The sun is bright."] # Documents
test_set = ["The sun in the sky is bright."] # Query
nltk.download('stopwords')
stopWords = stopwords.words('english')
vectorizer = CountVectorizer(stop_words=stopWords)
transformer = TfidfTransformer()
trainVectorizerArray = vectorizer.fit_transform(train_set).toarray()
testVectorizerArray = vectorizer.transform(test_set).toarray()
print('Fit Vectorizer to train set:', trainVectorizerArray)
print('Transform Vectorizer to test set:', testVectorizerArray)
cx = lambda a, b: round(np.inner(a, b) / (norm(a) * norm(b)), 3)
for vector in trainVectorizerArray:
    print(vector)
    for testV in testVectorizerArray:
        print(testV)
        cosine = cx(vector, testV)
        print(f"Cosine similarity: {cosine}")
```

Output:-



```
======= RESTART: C:\Users\sahil\OneDrive\Desktop\TY PRACTICAL\IR\2ndb.py =======
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sahil\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Fit Vectorizer to train set: [[1 0 1 0]
 [0 1 0 1]]
Transform Vectorizer to test set: [[0 1 1 1]]
[1 0 1 0]
[0 1 0 1]
[0 1 1 1]
Cosine similarity: 0.816
```

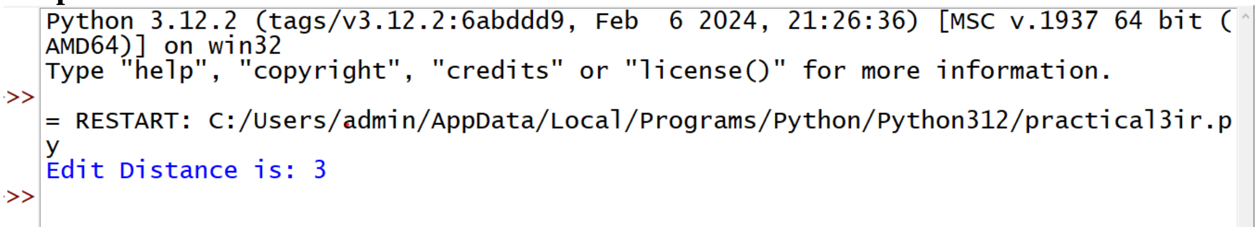
Practical No: 3

Aim: Spelling Correction in IR Systems

A)

```
def editDistance(str1,str2,m,n):
    if m==0:
        return n
    if n==0:
        return m
    if str1[m-1]==str2[n-1]:
        return editDistance(str1,str2,m-1,n-1)
    return 1+min(editDistance(str1,str2,m,n-1),
                editDistance(str1,str2,m-1,n),
                editDistance(str1,str2,m-1,n-1)
                )
str1="sunday"
str2="saturday"
print('Edit Distance is:', editDistance(str1,str2,len(str1),len(str2)))
```

Output:-



```
    Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>
    = RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python312/practical3ir.p
    y
    Edit Distance is: 3
>>
```

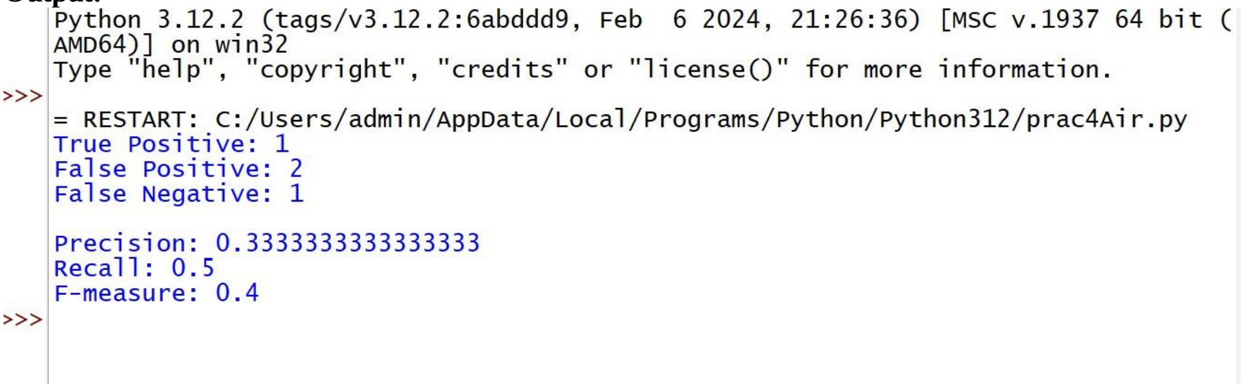
Practical No: 4

Aim: Evaluation Metrics for IR Systems

A) input:-

```
def calculate_metrics(retrieved_set, relevant_set):
    true_positive = len(retrieved_set.intersection(relevant_set))
    false_positive = len(retrieved_set.difference(relevant_set))
    false_negative = len(relevant_set.difference(retrieved_set))
    print("True Positive:", true_positive,
          "\nFalse Positive:", false_positive,
          "\nFalse Negative:", false_negative, "\n")
    precision = true_positive / (true_positive + false_positive)
    recall = true_positive / (true_positive + false_negative)
    f_measure = 2 * precision * recall / (precision + recall)
    return precision, recall, f_measure
retrieved_set = set(["doc1", "doc2", "doc3"]) # Predicted set
relevant_set = set(["doc1", "doc4"]) # Actually Needed set (Relevant)
precision, recall, f_measure = calculate_metrics(retrieved_set, relevant_set)
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F-measure: {f_measure}")
```

Output:-

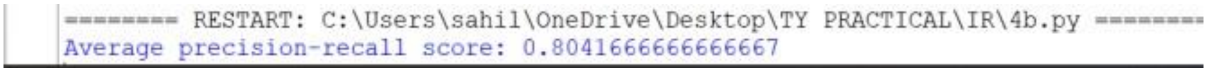


```
    Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>> 
    = RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python312/prac4Air.py
    True Positive: 1
    False Positive: 2
    False Negative: 1

    Precision: 0.3333333333333333
    Recall: 0.5
    F-measure: 0.4
>>> 
```

B) input:-

```
from sklearn.metrics import average_precision_score
y_true = [0, 1, 1, 0, 1, 1]
y_scores = [0.1, 0.4, 0.35, 0.8, 0.65, 0.9]

average_precision = average_precision_score(y_true, y_scores)
print(f" Average precision-recall score: {average_precision}")
```

Output:-



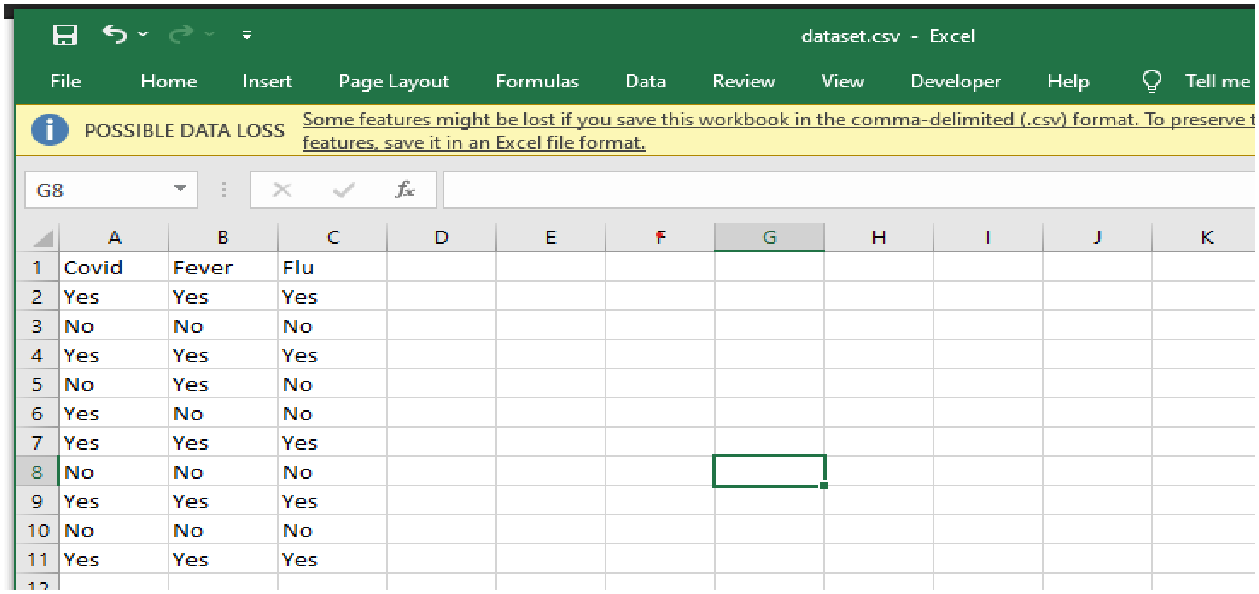
```
======== RESTART: C:\Users\sahil\OneDrive\Desktop\TY PRACTICAL\IR\4b.py ========
Average precision-recall score: 0.8041666666666667
```

Practical No: 5

Aim: Text Categorization

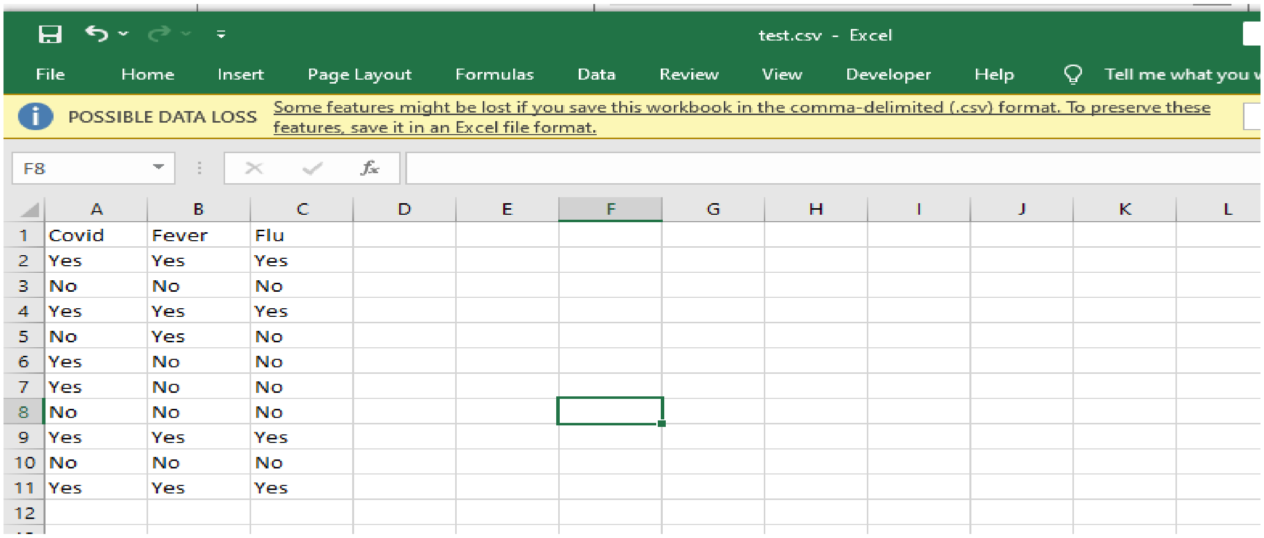
A) Implement a text classification algorithm (e.g., Naive Bayes or Support Vector Machines).

B) Train the classifier on a labelled dataset and evaluate its performance.



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve t features, save it in an Excel file format.

	A	B	C
1	Covid	Fever	Flu
2	Yes	Yes	Yes
3	No	No	No
4	Yes	Yes	Yes
5	No	Yes	No
6	Yes	No	No
7	Yes	Yes	Yes
8	No	No	No
9	Yes	Yes	Yes
10	No	No	No
11	Yes	Yes	Yes



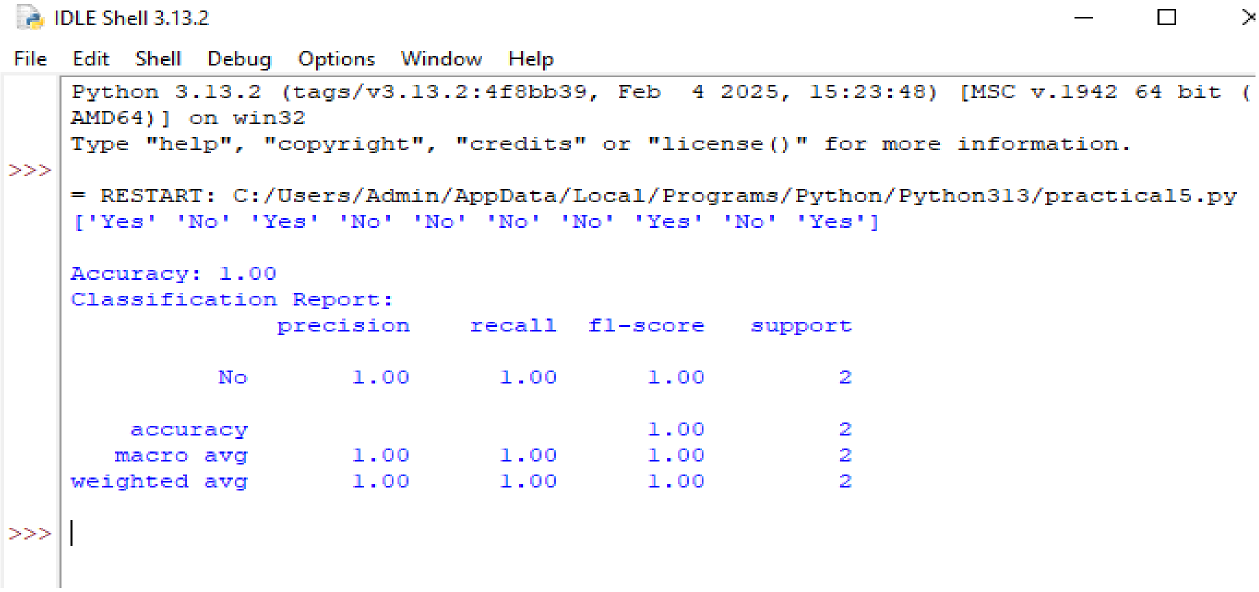
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C
1	Covid	Fever	Flu
2	Yes	Yes	Yes
3	No	No	No
4	Yes	Yes	Yes
5	No	Yes	No
6	Yes	No	No
7	Yes	No	No
8	No	No	No
9	Yes	Yes	Yes
10	No	No	No
11	Yes	Yes	Yes

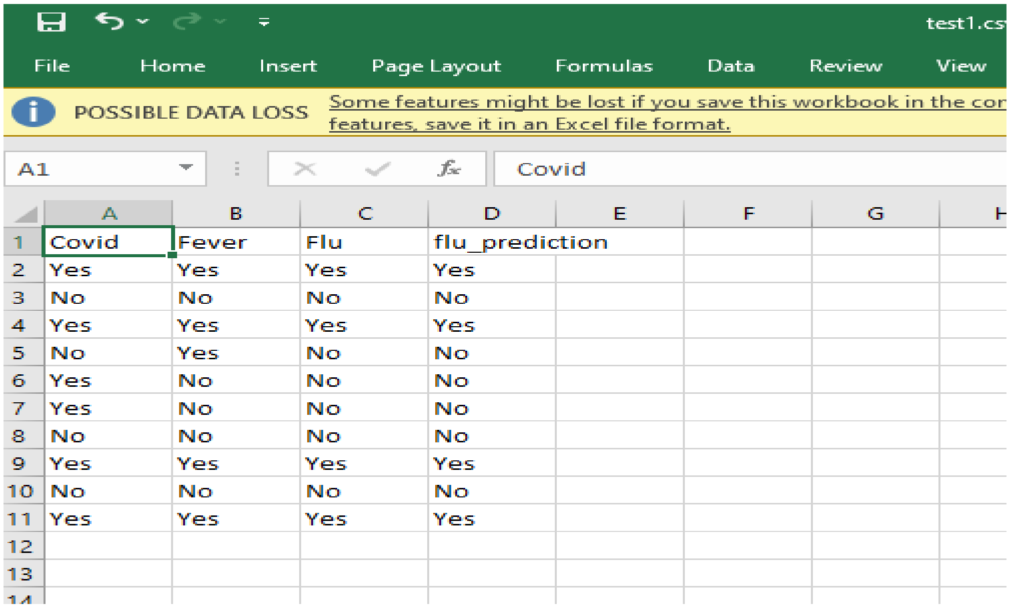
```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv(r"C:/Users/Admin/Documents/dataset.csv")
data = df["Covid"] + " " + df["Fever"]
X = data.astype(str)
y = df['Flu']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = CountVectorizer()

X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

classifier = MultinomialNB()
classifier.fit(X_train_counts, y_train)

data1 = pd.read_csv(r"C:/Users/Admin/Documents/test.csv")
new_data = data1["Covid"] + " " + data1["Fever"]
new_data_counts = vectorizer.transform(new_data.astype(str))

predictions = classifier.predict(new_data_counts)

new_data = predictions
print(new_data)

accuracy = accuracy_score(y_test, classifier.predict(X_test_counts))
print(f"\nAccuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, classifier.predict(X_test_counts)))
predictions_df = pd.DataFrame(predictions,columns = ['flu_prediction'])
data1 = pd.concat([data1,predictions_df],axis = 1)
data1.to_csv(r"C:/Users/Admin/Documents/test1.csv",index=False)
```

Output:-



```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python313/practical5.py
['Yes' 'No' 'Yes' 'No' 'No' 'No' 'No' 'Yes' 'No' 'Yes']

Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

          No       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

>>> |
```


test1.cs
File Home Insert Page Layout Formulas Data Review View
POSSIBLE DATA LOSS
Some features might be lost if you save this workbook in the cor
features, save it in an Excel file format.
A1
Covid

	A	B	C	D	E	F	G	H
1	Covid	Fever	Flu	flu_prediction				
2	Yes	Yes	Yes	Yes				
3	No	No	No	No				
4	Yes	Yes	Yes	Yes				
5	No	Yes	No	No				
6	Yes	No	No	No				
7	Yes	No	No	No				
8	No	No	No	No				
9	Yes	Yes	Yes	Yes				
10	No	No	No	No				
11	Yes	Yes	Yes	Yes				
12								
13								
14								

Practical No: 6

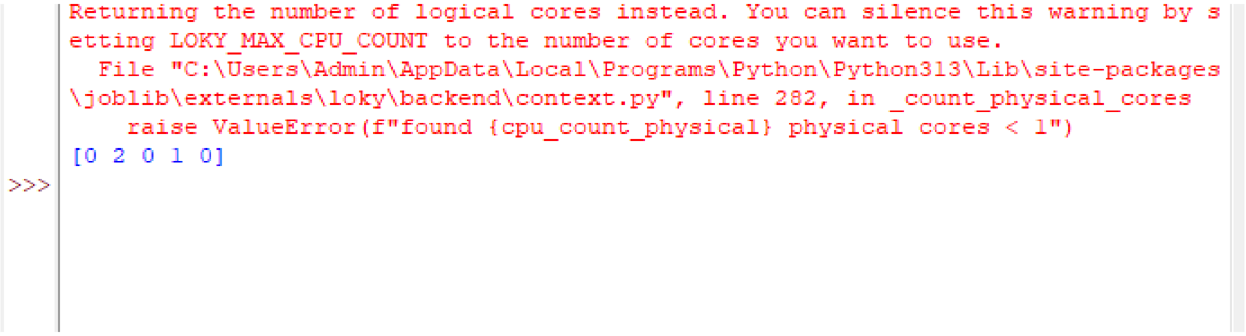
Aim: Clustering for Information Retrieval

Implement a clustering algorithm (e.g., K-means or hierarchical clustering).

Apply the clustering algorithm to a set of documents and evaluate the clustering results.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans


documents=["Cats are known for their agility and grace",
"Dogs are often called 'man's best friend'.",
"Some dogs are trained to assist people with disabilities.",
"The sun rises in the east and sets in the west.",
"Many cats enjoy climbing trees and chasing toys."
]
vectorizer=TfidfVectorizer(stop_words='english')
X=vectorizer.fit_transform(documents)
kmeans=KMeans(n_clusters=3,random_state=0).fit(X)
print(kmeans.labels_)
```

Output:-



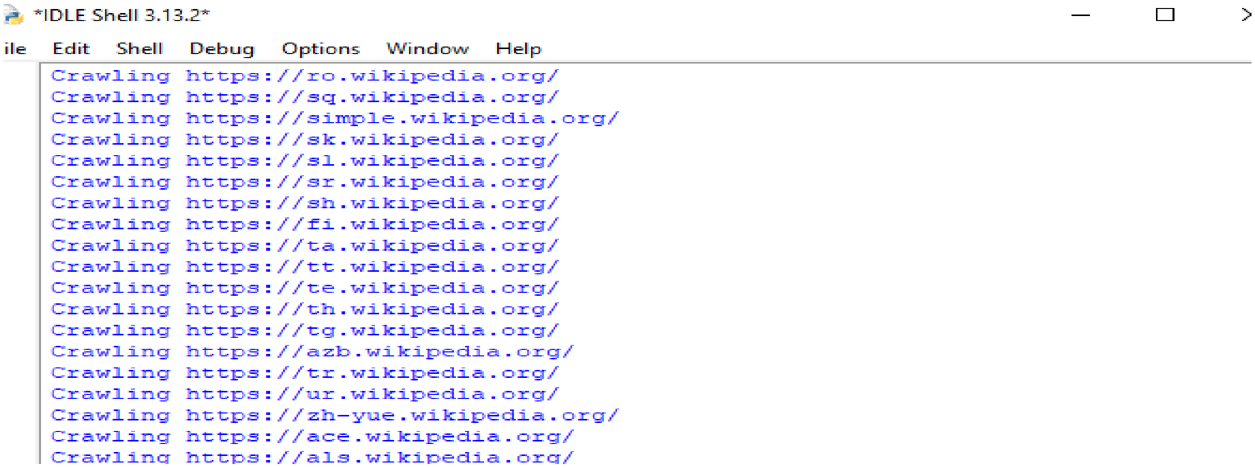
```
Returning the number of logical cores instead. You can silence this warning by s
etting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  File "C:\Users\Admin\AppData\Local\Programs\Python\Python313\Lib\site-packages
\joblib\externals\loky\backend\context.py", line 282, in _count_physical_cores
    raise ValueError(f"found {cpu_count_physical} physical cores < 1")
[0 2 0 1 0]
>>>
```

Practical No: 7

Aim: Web Crawling and Indexing

```
import requests
from bs4 import BeautifulSoup
import time
from urllib.parse import urljoin
from urllib.robotparser import RobotFileParser

def get_html(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'
    }
    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        return response.text
    except requests.exceptions.HTTPError as errh:
        print(f"HTTP Error: {errh}")
    except requests.exceptions.RequestException as err:
        print(f"Request Error: {err}")
    return None

def save_robots_txt(url):
    try:
        robots_url = urljoin(url, '/robots.txt')
        robots_content = get_html(robots_url)
        if robots_content:
            with open('robots.txt', 'w', encoding='utf-8-sig') as file:
                file.write(robots_content)
    except Exception as e:
        print(f"Error saving robots.txt: {e}")

def load_robots_txt():
    try:
        with open('robots.txt', 'r', encoding='utf-8-sig') as file:
            return file.read()
    except FileNotFoundError:
        return None

def extract_links(html, base_url):
    soup = BeautifulSoup(html, 'html.parser')
    links = []
    for link in soup.find_all('a', href=True):
        absolute_url = urljoin(base_url, link.get('href'))
        links.append(absolute_url)
    return links

def is_allowed_by_robots(url, robots_content):
    parser = RobotFileParser()
    parser.parse(robots_content.split('\n'))
    return parser.can_fetch('*', url)

def crawl(start_url, max_depth=3, delay=1):
    visited_urls = set()

    def recursive_crawl(url, depth, robots_content):
        if depth > max_depth or url in visited_urls or not is_allowed_by_robots(url, robots_content):
            return
        visited_urls.add(url)
        time.sleep(delay)

        html = get_html(url)
        if html:
            print(f"Crawling {url}")
            links = extract_links(html, url)

            for link in links:
                recursive_crawl(link, depth + 1, robots_content)

    save_robots_txt(start_url)
    robots_content = load_robots_txt()
    if not robots_content:
        print("Unable to retrieve robots.txt. Crawling without restrictions.")

    recursive_crawl(start_url, 1, robots_content)

    print("Performed by 740_Pallavi & 743_Deepak")
crawl('https://wikipedia.com', max_depth=2, delay=2)
```

Output:-



```
Crawling https://ro.wikipedia.org/
Crawling https://sq.wikipedia.org/
Crawling https://simple.wikipedia.org/
Crawling https://sk.wikipedia.org/
Crawling https://sl.wikipedia.org/
Crawling https://sr.wikipedia.org/
Crawling https://sh.wikipedia.org/
Crawling https://fi.wikipedia.org/
Crawling https://ta.wikipedia.org/
Crawling https://tt.wikipedia.org/
Crawling https://te.wikipedia.org/
Crawling https://th.wikipedia.org/
Crawling https://tg.wikipedia.org/
Crawling https://azb.wikipedia.org/
Crawling https://tr.wikipedia.org/
Crawling https://ur.wikipedia.org/
Crawling https://zh-yue.wikipedia.org/
Crawling https://ace.wikipedia.org/
Crawling https://als.wikipedia.org/
```

Practical No: 8

Aim: Link Analysis and PageRank

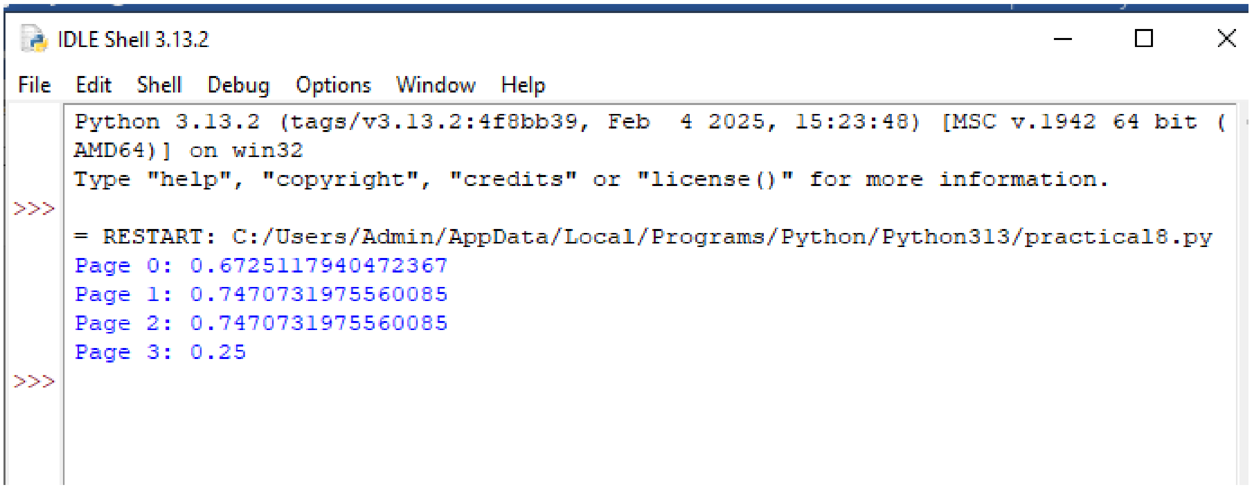
A) Implement the PageRank algorithm to rank web pages based on link analysis.
B) Apply the PageRank algorithm to a small web graph and analyse the results.

Input:-

```
import numpy as np


def page_rank(graph, damping_factor=0.85, max_iterations=100, tolerance=1e-6):
    num_nodes = len(graph)
    page_ranks = np.ones(num_nodes) / num_nodes


    for _ in range(max_iterations):
        prev_page_ranks = np.copy(page_ranks)
        for node in range(num_nodes):
            # Calculate the contribution from incoming links
            incoming_links = [i for i, v in enumerate(graph) if node in v]
            if not incoming_links:
                continue


            page_ranks[node] = (1 - damping_factor) / num_nodes + \
                damping_factor * sum(prev_page_ranks[link] / len(graph[link])
                                     for link in incoming_links)


        # Check for convergence
        if np.linalg.norm(page_ranks - prev_page_ranks, 2) < tolerance:
            break


    return page_ranks
```

```
if _name___== "_main_":
    web_graph = [
        [1, 2],
        [0, 2],
        [0, 1],
        [1, 2],
    ]

    result = page_rank(web_graph)

    for i, pr in enumerate(result):
        print(f"Page {i}: {pr}")
```

Output:-



```
    Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    = RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python313/practical8.py
    Page 0: 0.6725117940472367
    Page 1: 0.7470731975560085
    Page 2: 0.7470731975560085
    Page 3: 0.25
>>>
```