**NLP ASSIGNMENT – 1**

**NAME** : D. Hima Varshitha  **RegNo** : 22091A3318 **Class** : CSE(AI&ML) III B.Tech

**TITLE**

**PLAGIARISM DETECTION USING NLP TECHNIQUES**

**PROBLEM STATEMENT**

**Objective:** To develop an automated system that detects instances of plagiarism in academic and literary texts by leveraging advanced Natural Language Processing (NLP) techniques.

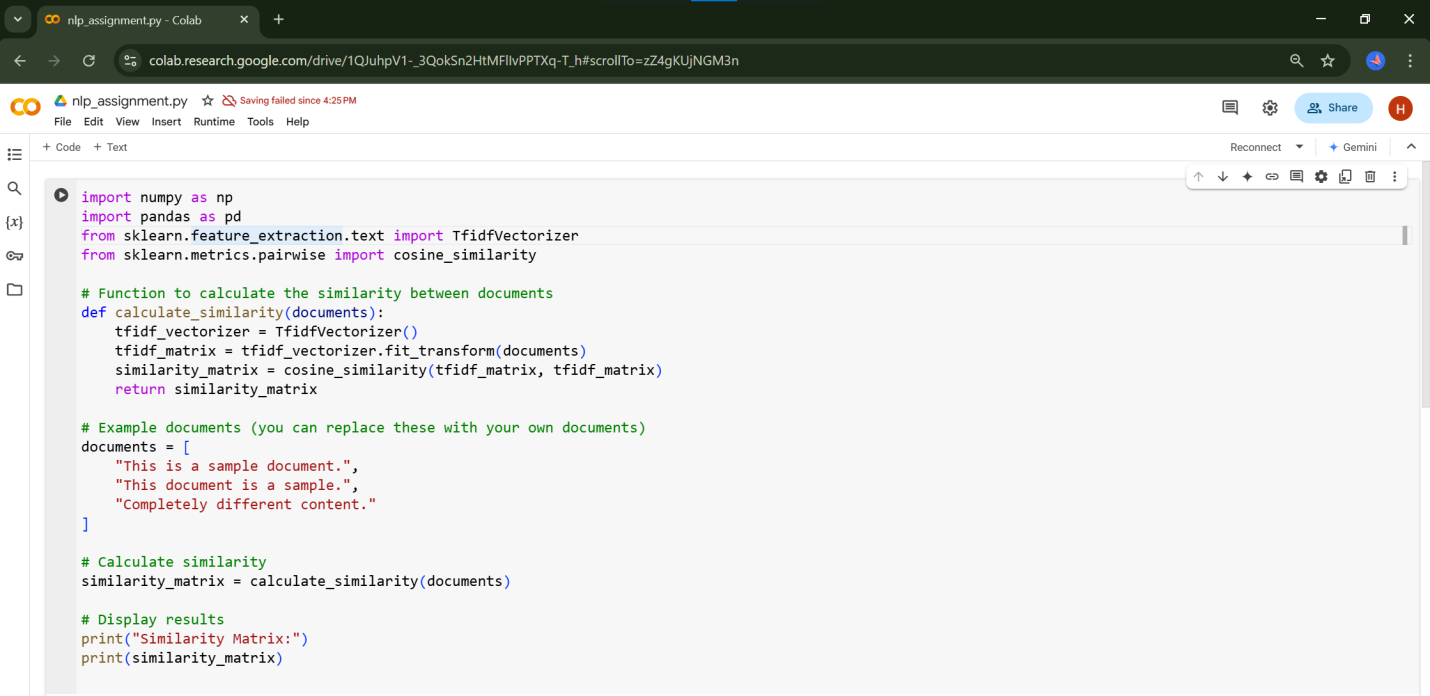
**Background:** Plagiarism is a significant issue in academia and the literary world, where individuals may copy or closely paraphrase content without proper attribution. Traditional methods of detection, such as manual checking, are time-consuming and inefficient. An automated system utilizing NLP can greatly enhance the accuracy and efficiency of plagiarism detection.

**SOLUTION DESCRIPTION**

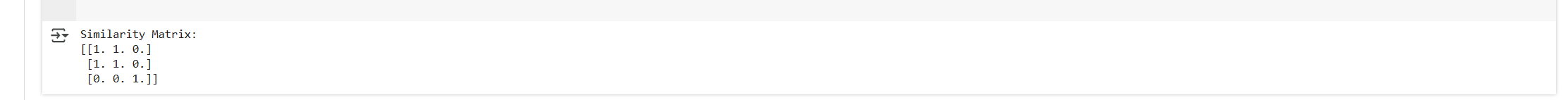
**Outcome:** The project aims to deliver a reliable, efficient, and scalable plagiarism detection system that significantly reduces the time and effort required for manual checking. The system should be capable of identifying various forms of plagiarism with high accuracy.

This provides a clear, structured approach to tackling the problem of plagiarism detection using NLP techniques.

**PROGRAM**

****

**OUTPUT**



**PROGRAM EXPLANATION**

.

**Step 1: Importing Libraries**

python

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

* numpy and pandas are general-purpose libraries for numerical and data manipulation.
* TfidfVectorizer from sklearn converts a collection of raw documents to a matrix of TF-IDF features.
* cosine\_similarity computes the cosine similarity between vectors.

**Step 2: Defining a Function to Calculate Similarity**

python

def calculate\_similarity(documents):

tfidf\_vectorizer = TfidfVectorizer()

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(documents)

similarity\_matrix = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

return similarity\_matrix

This function performs the following:

1. **Vectorization:**
   * TfidfVectorizer() initializes the vectorizer.
   * tfidf\_vectorizer.fit\_transform(documents) transforms the text documents into a matrix of TF-IDF features.
2. **Similarity Calculation:**
   * cosine\_similarity(tfidf\_matrix, tfidf\_matrix) computes pairwise cosine similarity between the vectors.
   * The similarity matrix is returned.

**Step 3: Example Documents**

python

documents = [

"This is a sample document.",

"This document is a sample.",

"Completely different content."

]

* Three example text documents for testing the similarity calculation.

**Step 4: Calculate Similarity**

python

similarity\_matrix = calculate\_similarity(documents)

* Calls the calculate\_similarity function with the example documents and stores the similarity matrix.

**Step 5: Display Results**

python

print("Similarity Matrix:")

print(similarity\_matrix)

* Prints the similarity matrix to the console.

**Explanation of the Output:**

Cosine similarity scores range from -1 to 1:

* 1 means the documents are identical.
* 0 means no similarity.
* -1 means they are completely opposite.

For your example, the output might look like:

Similarity Matrix:

[[1. 1. 0. ]

[1. 1. 0. ]

[0. 0. 1. ]]

* **1**: Diagonal elements are 1 because each document is identical to itself.
* **1:** Shows a high similarity between "This is a sample document." and "This document is a sample."
* **0**: "Completely different content." has no similarity with the other two documents.

**OUTPUT EXPLANATION**

The output of the code you provided is calculated on the basis of **words**, not sentences.

* **TF-IDF Vectorization:** The TfidfVectorizer converts the documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. In this process, it treats each document as a collection of words, and each word's importance is calculated based on its frequency in the document and its rarity across all documents.
* **Cosine Similarity:** Once the documents are represented as TF-IDF vectors, the cosine\_similarity function calculates the similarity between these vectors. This calculation is based on the vector representation of the documents, which are derived from the words they contain.

Essentially, the code breaks down each document into words, converts these words into numerical values (TF-IDF scores), and then calculates the similarity based on these numerical representations.