**AY: 2025-26**

| **Class:** | **BE- CSE(DS)** | **Semester:** | **VII** |
| --- | --- | --- | --- |
| **Course Code:** | **CSDOL7011** | **Course Name:** | **NLP Lab** |

| **Name of Student:** | Soham Shivpuje |
| --- | --- |
| **Roll No. :** | 50 |
| **Experiment No.:** | **2** |
| **Title of the Experiment:** | **Text Preprocessing and Feature Engineering using Bag-of-Words and TF-IDF** |
| **Date of Performance:** |  |
| **Date of Submission:** |  |

# **Evaluation**

| **Performance Indicator** | **Max. Marks** | **Marks Obtained** |
| --- | --- | --- |
| Performance | 5 |  |
| Understanding | 5 |  |
| Journal work and timely submission | 10 |  |
| Total | 20 |  |

| **Performance Indicator** | **Exceed Expectations (EE)** | **Meet Expectations (ME)** | **Below Expectations (BE)** |
| --- | --- | --- | --- |
| Performance | 4-5 | 2-3 | 1 |
| Understanding | 4-5 | 2-3 | 1 |
| Journal work and timely submission | 8-10 | 5-8 | 1-4 |

**Checked by**

**Name of Faculty :**

**Signature :**

**Date :**

**Aim:** To apply text preprocessing techniques and extract features from text using Bag-of- Words and TF-IDF methods.

**Objective:** To understand and apply basic text preprocessing and feature extraction techniques like Bag-of-Words and TF-IDF.

## **Tools Required:**

1. Python (preferably via Jupyter Notebook or Google Colab)
2. NLTK (Natural Language Toolkit)
3. Scikit-learn (sklearn)
4. Pandas

## **Procedure:**

1. Import necessary libraries:
   1. nltk, sklearn.feature\_extraction.text, pandas, re
2. Load or define a small sample text dataset.
3. Perform the following text preprocessing:
   1. Convert text to lowercase
   2. Remove punctuation and special characters
   3. Tokenize the text
   4. Remove stop words
   5. Apply stemming or lemmatization
4. Feature Extraction:
   1. Apply Bag-of-Words (BoW) vectorization using CountVectorizer.
   2. Apply TF-IDF vectorization using TfidfVectorizer.
5. Display the resulting feature matrices.
6. Compare and interpret the outputs of BoW and TF-IDF.

## **Description of the Experiment:**

This experiment demonstrates how raw text is cleaned, processed, and converted into structured numerical features using common feature engineering techniques. Students will understand the significance of preprocessing before feeding text data into machine learning models. Both BoW and TF-IDF representations help in quantifying the textual content into a usable format.

## **Detailed Description of the NLP Technique:**

1. Text Preprocessing:

Text preprocessing is a vital step in NLP that transforms unstructured textual data into a clean, machine-readable format. Common steps include:

* + **Tokenization**: Splitting text into words or tokens.
  + **Stop Word Removal**: Removing common words that don’t add significant meaning (e.g., "is", "the", "and").
  + **Stemming**: Reducing words to their root form (e.g., "playing" → "play").
  + **Lemmatization**: Reducing words to their dictionary form using context (e.g., "better"

→ "good").

1. Bag-of-Words (BoW):

BoW represents text by counting the frequency of each word in the document. It creates a vocabulary of known words and represents documents using word occurrence counts.

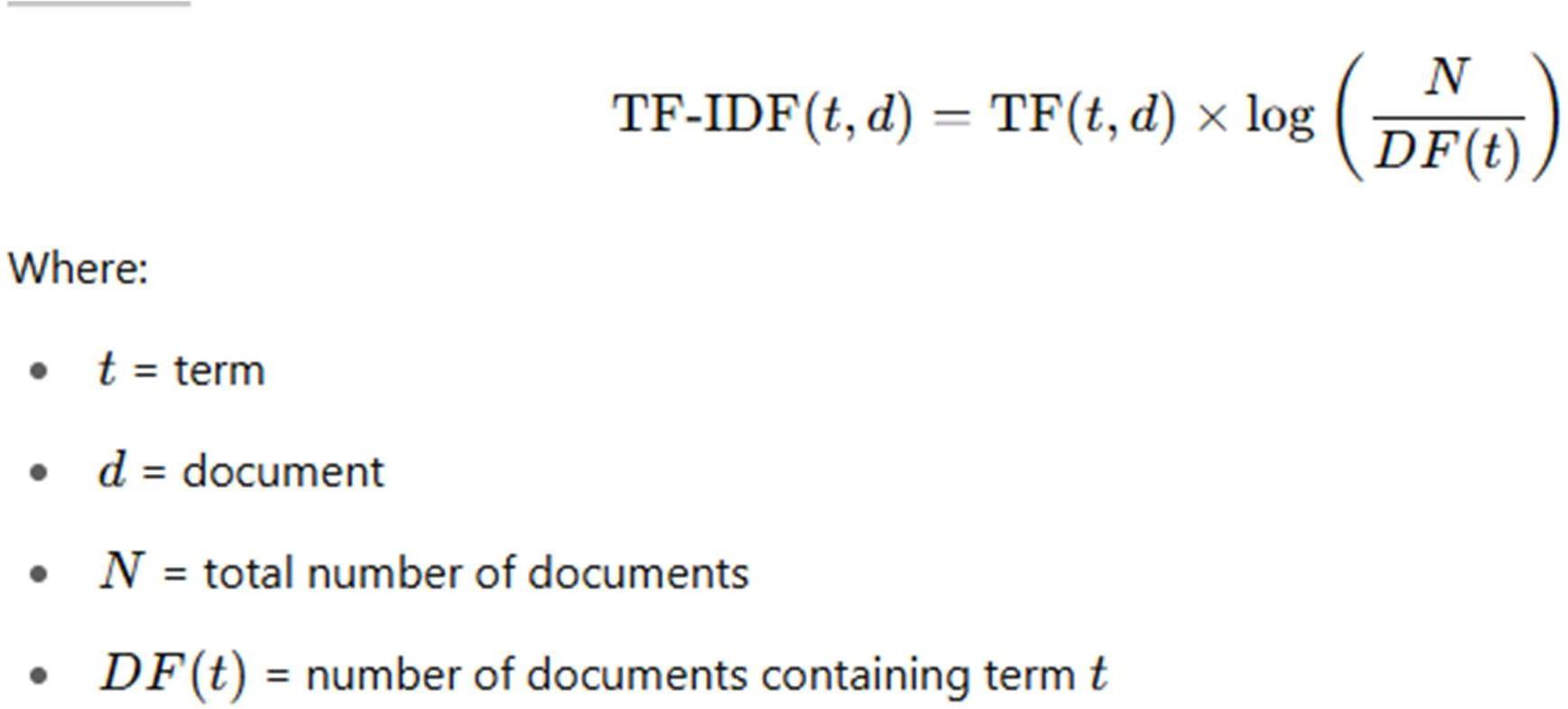
Pros: Simple and interpretable.

Cons: Ignores word order and semantic meaning.

1. TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF improves on BoW by assigning weights to words based on their importance in a document relative to the entire corpus.

TF: How often a word appears in a document. IDF: How rare the word is across all documents. Formula:



**Code and Output:**

**1. Import Required Libraries and Download NLTK Resources**

import nltk

import pandas as pd

import re

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer, WordNetLemmatizer

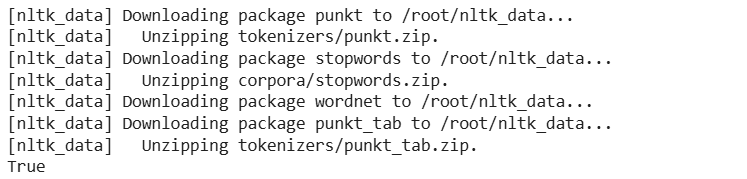
# Download necessary NLTK resources

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('punkt\_tab') # resolves LookupError



**2. Create a Sample Dataset**

# Sample text data

data = {

'Text': [

"Cats are playing in the garden.",

"Dogs bark loudly at strangers.",

"Birds are flying in the sky.",

"Cats and dogs are friendly pets.",

"The garden has many beautiful flowers."

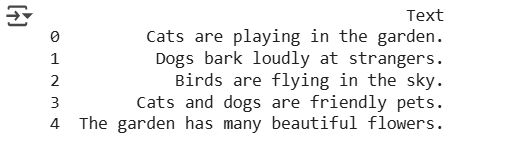
]

}

# Create a DataFrame

df = pd.DataFrame(data)

print(df)



**3. Text Preprocessing Function**

# Define stopwords, stemmer, and lemmatizer

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()

# Define a text preprocessing function

def preprocess(text):

text = text.lower() # Lowercase

text = re.sub(r'[^a-z\s]', '', text) # Remove punctuation

tokens = nltk.word\_tokenize(text) # Tokenize

tokens = [word for word in tokens if word not in stop\_words] # Remove stopwords

# Apply lemmatization

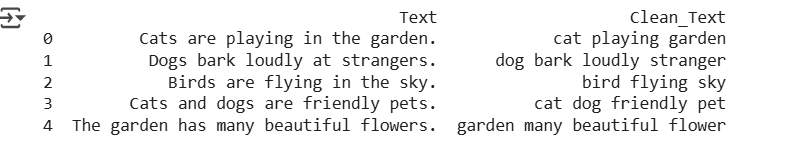
tokens = [lemmatizer.lemmatize(word) for word in tokens]

return ' '.join(tokens)

# Apply preprocessing

df['Clean\_Text'] = df['Text'].apply(preprocess)

print(df)



**4. Bag-of-Words (BoW) Vectorization**

# Initialize CountVectorizer

bow\_vectorizer = CountVectorizer()

# Fit and transform the clean text

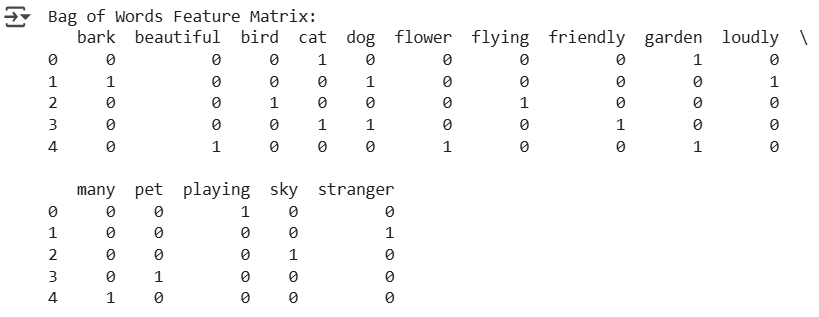
bow\_matrix = bow\_vectorizer.fit\_transform(df['Clean\_Text'])

# Convert to DataFrame for visualization

bow\_df = pd.DataFrame(bow\_matrix.toarray(), columns=bow\_vectorizer.get\_feature\_names\_out())

print("Bag of Words Feature Matrix:")

print(bow\_df)



**5. TF-IDF Vectorization**

# Initialize TF-IDF Vectorizer

tfidf\_vectorizer = TfidfVectorizer()

# Fit and transform the clean text

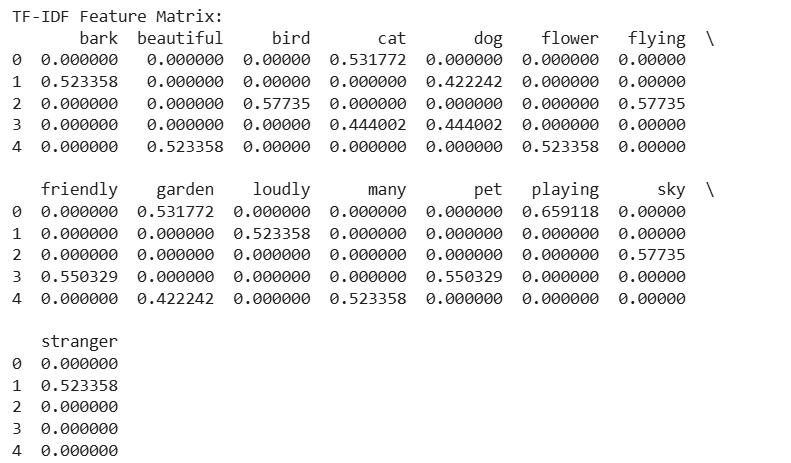
tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['Clean\_Text'])

# Convert to DataFrame for visualization

tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=tfidf\_vectorizer.get\_feature\_names\_out())

print("TF-IDF Feature Matrix:")

print(tfidf\_df)



**6. Interpretation BOW and TF-IDF**

import pandas as pd

# Display the BoW matrix

print("\nBag of Words Matrix:")

bow\_df = pd.DataFrame(bow\_matrix.toarray(), columns=bow\_vectorizer.get\_feature\_names\_out())

print(bow\_df)

# Display the TF-IDF matrix

print("\nTF-IDF Matrix:")

tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=tfidf\_vectorizer.get\_feature\_names\_out())

print(tfidf\_df)

# Compare word presence vs importance

print("\nTop words based on TF-IDF importance in each document:")

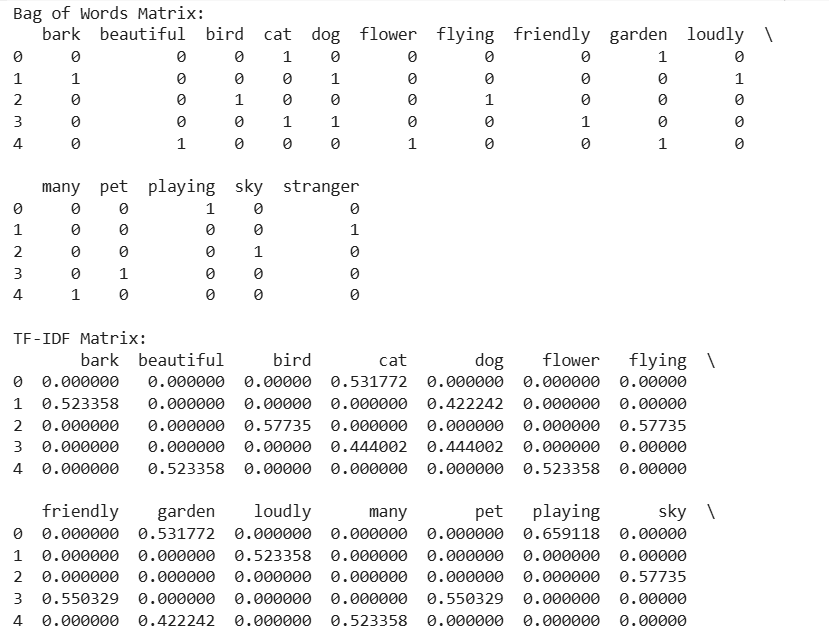
for i, row in tfidf\_df.iterrows():

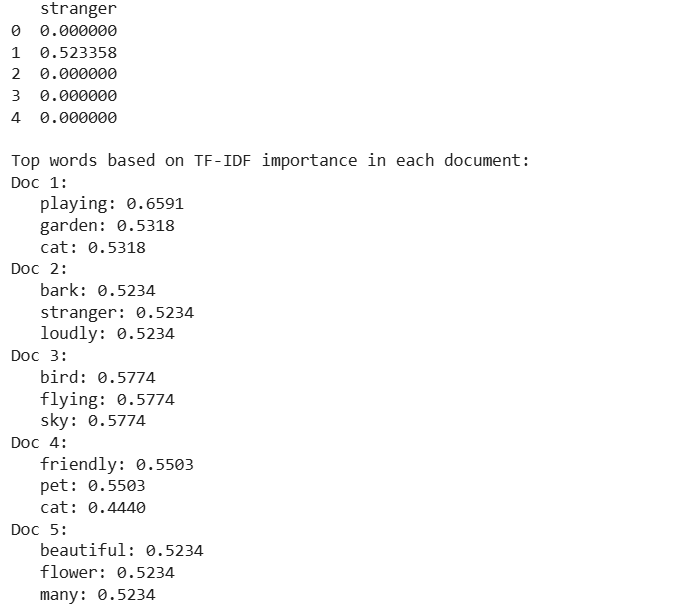
top\_words = row.sort\_values(ascending=False).head(3)

print(f"Doc {i+1}:")

for word, score in top\_words.items():

print(f" {word}: {score:.4f}")





## **Conclusion:**

Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) methods transformed raw textual data into numerical formats that can be used for machine learning and natural language processing tasks.

* BoW simply counts word occurrences, treating all words equally regardless of their significance across the corpus.
* TF-IDF, on the other hand, not only considers term frequency but also adjusts for how common or rare a word is across all documents, thus assigning higher weights to more informative and unique words.