**PRACTICAL NO : 07**

**TEXT ANALYTICS**

**CODE :**

text="Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization."

from nltk.tokenize import sent\_tokenize

tokenized\_text=sent\_tokenize(text)

print(tokenized\_text)

from nltk.tokenize import word\_tokenize

tokenized\_word=word\_tokenize(text)

print(tokenized\_word)

from nltk import re

from nltk.corpus import stopwords

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

print(stop\_words)

text= "How to remove stop words with NLTK library in Python?"

text= re.sub('[^a-zA-Z]', ' ',text)

tokens = word\_tokenize(text.lower())

filtered\_text=[]

for w in tokens:

if w not in stop\_words:

filtered\_text.append(w)

print("Tokenized Sentence:",tokens)

print("Filterd Sentence:",filtered\_text)

from nltk.stem import PorterStemmer

e\_words=["wait","waitng","waited","waits"]

ps=PorterStemmer()

for w in e\_words :

rootWord=ps.stem(w)

print(rootWord)

import nltk

from nltk.stem import WordNetLemmatizer

wordnet\_lemmatizer = WordNetLemmatizer()

text = "studies studying cries cry"

tokenization = nltk.word\_tokenize(text)

for w in tokenization:

print("Lemma for {} is {}".format(w,wordnet\_lemmatizer.lemmatize(w)))

import nltk

from nltk.tokenize import word\_tokenize

data="The pink sweater fit her perfectly"

words=word\_tokenize(data)

for w in words:

print(nltk.pos\_tag([w]))

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

documentA = 'Jupiter is the largest Planet'

documentB = 'Mars is the fourth planet from the Sun'

bagOfWordsA = documentA.split(' ')

bagOfWordsB = documentB.split(' ')

uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))

print(uniqueWords)

numOfWordsA = dict.fromkeys(uniqueWords, 0)

for word in bagOfWordsA:

numOfWordsA[word] += 1

numOfWordsB = dict.fromkeys(uniqueWords, 0)

for word in bagOfWordsB:

numOfWordsB[word] += 1

def computeTF(wordDict, bagOfWords):

tfDict = {}

bagOfWordsCount = len(bagOfWords)

for word, count in wordDict.items():

tfDict[word] = count / float(bagOfWordsCount)

return tfDict

tfA = computeTF(numOfWordsA, bagOfWordsA)

tfB = computeTF(numOfWordsB, bagOfWordsB)

def computeIDF(documents):

import math

N = len(documents)

idfDict = dict.fromkeys(documents[0].keys(), 0)

for document in documents:

for word, val in document.items():

if val > 0:

idfDict[word] += 1

for word, val in idfDict.items():

idfDict[word] = math.log(N / float(val))

return idfDict

idfs = computeIDF([numOfWordsA, numOfWordsB])

**OUTPUT :**

