```
import nltk
from nltk.tokenize import word_tokenize  # Importing the function to
split text into words (tokens)
from nltk.corpus import stopwords  # Importing the list of common
stopwords
from nltk.stem import PorterStemmer  # Importing Porter Stemmer for
stemming words
from nltk.stem import WordNetLemmatizer  # Importing WordNet
Lemmatizer for lemmatizing words
from sklearn.feature_extraction.text import TfidfVectorizer  #
Importing TfidfVectorizer to calculate TF-IDF

# Download necessary NLTK data for tokenization, stopwords, POS
tagging, and lemmatization
nltk.download('punkt')  # For tokenizing words in the text
nltk.download('stopwords')  # For removing stopwords (common words
like 'the', 'is', 'and')
nltk.download('averaged_perceptron_tagger')  # For Part of Speech
(POS) tagging
nltk.download('wordnet')  # For Lemmatization (reducing words to their
dictionary form)

# Step 1: Read the content from the SpaceX.txt file
file_path = r"C:\Users\ASUS\Documents\pythonStack\DS_PR\SpaceX.txt"  #
Path to the text file

# Open and read the file
with open(file_path, 'r', encoding='utf-8') as file:  # Open file in
read mode
    document = file.read()  # Read all the content of the file into a
string

# Step 2: Tokenization
# Tokenization is the process of splitting text into individual words
or tokens
tokens = word_tokenize(document)  # Tokenizing the content of the
document
print("Tokens:", tokens)  # Print the list of tokens (individual
words)

# Step 3: POS Tagging
# POS tagging labels each token with its part of speech (e.g., noun,
verb, etc.)
pos_tags = nltk.pos_tag(tokens)  # Get the part of speech tags for
each token
print("POS Tags:", pos_tags)  # Print each token with its POS tag

# Step 4: Stop Words Removal
# Stop words are common words like 'the', 'is', 'in', which do not
carry meaningful information
```

```
stop_words = set(stopwords.words('english'))  # Get a set of English
stopwords
# Filter out the stop words from the tokens
filtered_tokens = [word for word in tokens if word.lower() not in
stop_words]
print("Filtered Tokens (Stop Words Removed):", filtered_tokens)  #
Print tokens after removing stop words

# Step 5: Stemming
# Stemming is the process of reducing words to their root form (e.g.,
'running' -> 'run')
stemmer = PorterStemmer()  # Initialize the Porter Stemmer
# Apply the stemming process to each filtered token
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
print("Stemmed Tokens:", stemmed_tokens)  # Print the stemmed tokens

# Step 6: Lemmatization
# Lemmatization reduces words to their base or dictionary form (e.g.,
'better' -> 'good')
lemmatizer = WordNetLemmatizer()  # Initialize the Lemmatizer
# Apply lemmatization to each filtered token
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in
filtered_tokens]
print("Lemmatized Tokens:", lemmatized_tokens)  # Print the lemmatized
tokens

# Step 7: Term Frequency (TF) and Inverse Document Frequency (IDF)
# TF-IDF is a statistic that reflects the importance of a word in a
document within a corpus
vectorizer = TfidfVectorizer()  # Initialize the TfidfVectorizer
# Fit the vectorizer to the document and transform it into a matrix of
TF-IDF values
X = vectorizer.fit_transform([document])

# Display the TF-IDF scores for each word in the document
print("TF-IDF Scores:")
for word, idx in vectorizer.vocabulary_.items():  # Iterate through
each word in the vocabulary
    print(f"{word}: {X[0, idx]}")  # Print the word and its
corresponding TF-IDF score
```

```
Tokens: ['SpaceX', 'is', 'an', 'American', 'aerospace',
'manufacturer', 'and', 'space', 'transportation', 'company',
'founded', 'by', 'Elon', 'Musk', 'in', '2002', '.', 'Its', 'goal',
'is', 'to', 'reduce', 'space', 'transportation', 'costs', 'and',
'enable', 'the', 'colonization', 'of', 'Mars', '.', 'SpaceX', 'has',
'achieved', 'significant', 'milestones', 'with', 'its', 'reusable',
'rocket', 'technology', ',', 'making', 'space', 'missions', 'more',
'affordable', 'and', 'sustainable', '.', 'Famous', 'Projects', ':',
'Falcon', '1', ':', 'The', 'first', 'privately', 'developed', 'liquid-
fueled', 'rocket', 'to', 'reach', 'orbit', '.', 'Falcon', '9', ':',
'A', 'reusable', 'rocket', 'designed', 'to', 'transport',
'satellites', ',', 'cargo', ',', 'and', 'crew', 'to', 'the',
'International', 'Space', 'Station', '(', 'ISS', ')', '.', 'Known',
'for', 'landing', 'its', 'first', 'stage', 'back', 'on', 'Earth',
'after', 'launching', '.', 'Dragon', ':', 'A', 'spacecraft', 'that',
'delivers', 'cargo', 'to', 'the', 'ISS', ',', 'with', 'a', 'version',
'designed', 'to', 'carry', 'crew', 'as', 'well', '(', 'Crew',
'Dragon', ')', '.', 'Starship', ':', 'A', 'fully', 'reusable',
'spacecraft', 'designed', 'for', 'missions', 'to', 'the', 'Moon', ',',
'Mars', ',', 'and', 'beyond', '.', 'It', "'s", 'intended', 'to', 'be',
'the', 'largest', 'and', 'most', 'powerful', 'rocket', 'ever',
'built', '.', 'Upcoming', 'Projects', ':', 'Starship', 'Mars',
'Mission', ':', 'SpaceX', "'s", 'goal', 'to', 'transport', 'humans',
'to', 'Mars', ',', 'paving', 'the', 'way', 'for', 'interplanetary',
'colonization', '.', 'Starlink', ':', 'A', 'satellite',
'constellation', 'providing', 'global', 'internet', 'coverage', ',',
'especially', 'in', 'remote', 'areas', '.', 'Lunar', 'Gateway', ':',
'SpaceX', 'is', 'a', 'key', 'partner', 'in', 'NASA', "'s", 'Artemis',
'program', 'to', 'land', 'humans', 'on', 'the', 'Moon', 'by', '2024',
'using', 'the', 'Starship']

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!

----------------------------------------------------------------------
-----
LookupError                               Traceback (most recent call
last)
Cell In[39], line 28
     24 print("Tokens:", tokens)  # Print the list of tokens
(individual words)
     26 # Step 3: POS Tagging
     27 # POS tagging labels each token with its part of speech (e.g.,
noun, verb, etc.)
---> 28 pos_tags = nltk.pos_tag(tokens)  # Get the part of speech tags
for each token
     29 print("POS Tags:", pos_tags)  # Print each token with its POS
```

```
tag
     31 # Step 4: Stop Words Removal
     32 # Stop words are common words like 'the', 'is', 'in', which do
not carry meaningful information

File ~\anaconda3\Lib\site-packages\nltk\tag\__init__.py:168, in
pos_tag(tokens, tagset, lang)
    143 def pos_tag(tokens, tagset=None, lang="eng"):
    144     """
    145     Use NLTK's currently recommended part of speech tagger to
    146     tag the given list of tokens.
   (...)
    166     :rtype: list(tuple(str, str))
    167     """
--> 168     tagger = _get_tagger(lang)
    169     return _pos_tag(tokens, tagset, tagger, lang)

File ~\anaconda3\Lib\site-packages\nltk\tag\__init__.py:110, in
_get_tagger(lang)
    108     tagger = PerceptronTagger(lang=lang)
    109 else:
--> 110     tagger = PerceptronTagger()
    111 return tagger

File ~\anaconda3\Lib\site-packages\nltk\tag\perceptron.py:183, in
PerceptronTagger.__init__(self, load, lang)
    181 self.classes = set()
    182 if load:
--> 183     self.load_from_json(lang)

File ~\anaconda3\Lib\site-packages\nltk\tag\perceptron.py:273, in
PerceptronTagger.load_from_json(self, lang)
    271 def load_from_json(self, lang="eng"):
    272     # Automatically find path to the tagger if location is not
specified.
--> 273     loc = find(f"taggers/averaged_perceptron_tagger_{lang}/")
    274     with open(loc + TAGGER_JSONS[lang]["weights"]) as fin:
    275         self.model.weights = json.load(fin)

File ~\anaconda3\Lib\site-packages\nltk\data.py:579, in
find(resource_name, paths)
    577 sep = "*" * 70
    578 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579 raise LookupError(resource_not_found)

LookupError:
**********************************************************************
  Resource averaged_perceptron_tagger_eng not found.
  Please use the NLTK Downloader to obtain the resource:
```

```
  >>> import nltk
  >>> nltk.download('averaged_perceptron_tagger_eng')

  For more information see: https://www.nltk.org/data.html

  Attempted to load taggers/averaged_perceptron_tagger_eng/

  Searched in:
    - 'C:\\Users\\ASUS/nltk_data'
    - 'C:\\Users\\ASUS\\anaconda3\\nltk_data'
    - 'C:\\Users\\ASUS\\anaconda3\\share\\nltk_data'
    - 'C:\\Users\\ASUS\\anaconda3\\lib\\nltk_data'
    - 'C:\\Users\\ASUS\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************
```