In [ ]:
```
"""
Text Analytics
1. Extract Sample document and apply following document preprocessing methods:
Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document
Frequency.

"""
```

In [ ]:
```
"""
1) Text Preprocessing:
    1. Sentence/word tokenization
    2. Removing Stopwords
    3. Stemming
    4. Lematization

2) TF_IDF



"""
```

In [1]:
```
import re
```

In [13]:
```
!pip install nltk
import nltk

nltk.download('punkt')
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nltk in /home/ubuntu/.local/lib/python3.10/site-packages (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in /home/ubuntu/.local/lib/python3.10/site-packages (from n
ltk) (2023.3.23)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from nltk) (8.0.3)
Requirement already satisfied: joblib in /home/ubuntu/.local/lib/python3.10/site-packages (from nltk) (1.
2.0)
Requirement already satisfied: tqdm in /home/ubuntu/.local/lib/python3.10/site-packages (from nltk) (4.64.
1)

[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: python3 -m pip install --upgrade pip

[nltk_data] Downloading package punkt to /home/ubuntu/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[13]:
```
True
```

In [41]:
```
# python3 -m pip install --upgrade pip
```

In [ ]:
```
nltk.download('all')
```

In [11]:
```
from nltk.tokenize import sent_tokenize
```

In [53]:
```
# this df is not considered in this notebook, but can be used for preprocessing purposes

# df = pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/amazon.csv')
```

In [52]:
```
# df
```

In [9]:
```
# sample text

text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
```

In [14]:
```
tokenized = sent_tokenize(text)
print(tokenized)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.', 'The sky is pi
nkish-blue.', "You shouldn't eat cardboard"]
```

In [15]:
```
from nltk.tokenize import word_tokenize
tokenized_words = word_tokenize(text)
print(tokenized_words)
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great
', ',', 'and', 'city', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n'
t", 'eat', 'cardboard']
```

In [17]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/ubuntu/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[17]:
```
True
```

In [18]:
```
from nltk.corpus import stopwords
stopwords = set(stopwords.words('english'))
```

In [19]:
```
print(stopwords)
```

```
{'under', 'am', 'have', 'did', 'out', 'so', 'aren', 'by', 'she', 'some', 'do', 'should', 'there', 'on', 'w
hen', 'during', 'down', 'from', 'you', 'at', 'then', 'are', 'up', 't', 'they', 'doesn', 're', "doesn't", '
ours', "shan't", 'himself', 'has', "weren't", 'does', 'to', 'weren', 'mightn', "you'd", 'shouldn', 'not',
'those', 'below', 'after', 'whom', 'before', 'him', 'shan', 'don', 'further', 'a', 'that', 'more', 'these
', 'into', 'as', 'was', 'the', 'while', 'just', 'ourselves', "needn't", 'until', 'such', 'but', 'isn', 'fo
r', 'here', 'it', 'myself', "mustn't", "you're", "shouldn't", 'didn', 'doing', 'with', 'o', 'yours', 'y',
"hasn't", "she's", 'which', "hadn't", 'because', 'he', 'm', 'its', 'themselves', 'very', "mightn't", 'abov
e', 'i', 'once', 'my', 'wasn', 'most', 'all', 'an', 'wouldn', 'll', "wasn't", 'your', "aren't", 'their', "
you've", "you'll", 'nor', 'her', 'hers', 'hasn', 'each', 'needn', 'we', 'same', 'can', 've', "wouldn't", '
his', 'and', 'both', 'any', 'in', "should've", 'd', "don't", 'own', 'only', "couldn't", "isn't", "won't",
'mustn', 'of', "it's", 'won', 'against', "didn't", 'or', 'about', 'theirs', 'our', 'me', 'itself', 'throug
h', 'herself', 'who', 'had', 'how', "haven't", 'over', 'be', 'few', 'being', 'why', 'again', 'too', 'than
', 'this', 'other', 'were', 'if', 'where', 'is', 'yourselves', 'now', 'no', 's', 'having', 'off', 'couldn
', 'haven', 'yourself', "that'll", 'ain', 'hadn', 'what', 'them', 'been', 'between', 'ma', 'will'}
```

In [24]:
```
filtered_sent = []

for w in tokenized_words:
    if w not in stopwords:
        filtered_sent.append(w)
print("Filtered_Sentence: ")
print(filtered_sent)
```

```
Filtered_Sentence:
['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'great', ',', 'city', 'awesome', '.', 'The
', 'sky', 'pinkish-blue', '.', 'You', "n't", 'eat', 'cardboard']
```

In [23]:
```
from nltk.stem import PorterStemmer
ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Stemmed Sentence:",stemmed_words)
```

```
Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?', 'the', 'weather', 'great', ',', 'citi', 'aw
esom', '.', 'the', 'sky', 'pinkish-blu', '.', 'you', "n't", 'eat', 'cardboard']
```

In [29]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /home/ubuntu/nltk_data...
```

Out[29]:
```
True
```

In [30]:
```
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

lemmatized_words = []

for w in filtered_sent:
    lemmatized_words.append(lem.lemmatize(w))

print("lemmatized Sentence: ", lemmatized_words)
```

```
lemmatized Sentence:  ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'great', ',', 'city',
'awesome', '.', 'The', 'sky', 'pinkish-blue', '.', 'You', "n't", 'eat', 'cardboard']
```

In [40]:
```
# pos_tag(tokenized_words)
```

In [ ]:
```
!pip install spacy
import spacy
```

In [37]:
```
nlp = spacy.load('en_core_web_sm')
```

In [44]:
```
doc = nlp(text)
```

In [42]:
```
for token in doc:
    print(token, "|", token.pos_,"|", spacy.explain(token.pos_),"|",token.tag_, spacy.explain(token.tag_))
    print("")
```

```
Hello | INTJ | interjection | UH interjection

Mr. | PROPN | proper noun | NNP noun, proper singular

Smith | PROPN | proper noun | NNP noun, proper singular

, | PUNCT | punctuation | , punctuation mark, comma

how | SCONJ | subordinating conjunction | WRB wh-adverb

are | AUX | auxiliary | VBP verb, non-3rd person singular present

you | PRON | pronoun | PRP pronoun, personal

doing | VERB | verb | VBG verb, gerund or present participle

today | NOUN | noun | NN noun, singular or mass

? | PUNCT | punctuation | . punctuation mark, sentence closer

The | DET | determiner | DT determiner

weather | NOUN | noun | NN noun, singular or mass

is | AUX | auxiliary | VBZ verb, 3rd person singular present

great | ADJ | adjective | JJ adjective (English), other noun-modifier (Chinese)

, | PUNCT | punctuation | , punctuation mark, comma

and | CCONJ | coordinating conjunction | CC conjunction, coordinating

city | NOUN | noun | NN noun, singular or mass

is | AUX | auxiliary | VBZ verb, 3rd person singular present
```

```
awesome | ADJ | adjective | JJ adjective (English), other noun-modifier (Chinese)
. | PUNCT | punctuation | . punctuation mark, sentence closer

 | SPACE | space | _SP whitespace
The | DET | determiner | DT determiner
sky | NOUN | noun | NN noun, singular or mass
is | AUX | auxiliary | VBZ verb, 3rd person singular present
pinkish | NOUN | noun | NN noun, singular or mass
- | PUNCT | punctuation | HYPH punctuation mark, hyphen
blue | ADJ | adjective | JJ adjective (English), other noun-modifier (Chinese)
. | PUNCT | punctuation | . punctuation mark, sentence closer
You | PRON | pronoun | PRP pronoun, personal
should | AUX | auxiliary | MD verb, modal auxiliary
n't | PART | particle | RB adverb
eat | VERB | verb | VB verb, base form
cardboard | NOUN | noun | NN noun, singular or mass
```

TF_IDF

In [46]:
```
# import required module
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [ ]:

In [48]:
```
# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(lemmatized_words)
```

In [49]:
```
# get idf values
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ':', ele2)
```

```
idf values:
awesome : 3.3978952727983707
blue : 3.3978952727983707
cardboard : 3.3978952727983707
city : 3.3978952727983707
eat : 3.3978952727983707
great : 3.3978952727983707
hello : 3.3978952727983707
mr : 3.3978952727983707
pinkish : 3.3978952727983707
sky : 3.3978952727983707
smith : 3.3978952727983707
the : 2.992430164690206
today : 3.3978952727983707
weather : 3.3978952727983707
you : 3.3978952727983707

/home/ubuntu/.local/lib/python3.10/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please
use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [50]:
```
# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

# in matrix form
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

```
Word indexes:
{'hello': 6, 'mr': 7, 'smith': 10, 'today': 12, 'the': 11, 'weather': 13, 'great': 5, 'city': 3, 'awesome
': 0, 'sky': 9, 'pinkish': 8, 'blue': 1, 'you': 14, 'eat': 4, 'cardboard': 2}

tf-idf value:
  (0, 6)        1.0
  (1, 7)        1.0
  (2, 10)       1.0
  (4, 12)       1.0
  (6, 11)       1.0
  (7, 13)       1.0
  (8, 5)        1.0
  (10, 3)       1.0
  (11, 0)       1.0
  (13, 11)      1.0
  (14, 9)       1.0
  (15, 1)       0.7071067811865475
  (15, 8)       0.7071067811865475
  (17, 14)      1.0
  (19, 4)       1.0
  (20, 2)       1.0

tf-idf values in matrix form:
[[0.         0.         0.         0.         0.         0.
  1.         0.         0.         0.         0.         0.
  0.         0.         0.        ]
```

```
[0.         0.         0.         0.         0.         0.
 0.         1.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         1.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 1.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         1.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         1.         0.        ]
[0.         0.         0.         0.         0.         1.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         1.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[1.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         1.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         1.         0.         0.
```

```
 0.         0.         0.        ]
[0.         0.70710678 0.         0.         0.         0.
 0.         0.         0.70710678 0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         1.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.         0.         1.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         1.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]]
```

In [ ]: