In [5]:
```
import nltk
import string
import re
```

In [6]:
```
def text_lowercase(text):
    return text.lower()

input_str="Hi My Name is Yadnesh Wani"
text_lowercase(input_str)
```

Out[6]: 'hi my name is yadnesh wani'

In [7]:
```
def remove_numbers(text):
    result=re.sub(r'\d+','',text)
    return result

input_str="Hello my Name is Yadnesh Wani and My Age is 20"
remove_numbers(input_str)
```

Out[7]: 'Hello my Name is Yadnesh Wani and My Age is '

In [8]:
```
def remove_punctuation(text):
    translator=str.maketrans('','',string.punctuation)
    return text.translate(translator)

input_str="What is Your Name ? My Name is Yadnesh ! "
remove_punctuation(input_str)
```

Out[8]: 'What is Your Name My Name is Yadnesh '

In [9]:
```
def remove_whitespace(text):
    return " ".join(text.split())

input_str="Hello       World    !"
remove_whitespace(input_str)
```

Out[9]: 'Hello World !'

In [10]:
```
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


def remove_stopwords(text):
    stop_words=set(stopwords.words("english"))
    word_token=word_tokenize(text)
    filtered_text=[word for word in word_token if word not in stop_words]
    return filtered_text

example_text="This is a sample sentence and we are going to remove the stopwor
remove_stopwords(example_text)
```

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

Out[10]:
```
['This', 'sample', 'sentence', 'going', 'remove', 'stopwords']
```

In [11]:
```
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer=PorterStemmer()


def stem_words(text):
    word_tokens=word_tokenize(text)
    stems=[stemmer.stem(word) for word in word_tokens]
    return stems

text="Data Science use Scientific methods algorithms and many types of Process
stem_words(text)
```

Out[11]:
```
['data',
 'scienc',
 'use',
 'scientif',
 'method',
 'algorithm',
 'and',
 'mani',
 'type',
 'of',
 'process']
```

In [12]:
```
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer=WordNetLemmatizer()


def lemmatize_word(text):
    word_tokens=word_tokenize(text)
    lemmas=[lemmatizer.lemmatize(word,pos='v') for word in word_tokens]
    return lemmas

text="Data Science use Scientific methods algorithms and many types of Process
lemmatize_word(text)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\acer\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\acer\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[12]:
```
['Data',
 'Science',
 'use',
 'Scientific',
 'methods',
 'algorithms',
 'and',
 'many',
 'type',
 'of',
 'Process']
```

In [ ]:

In [ ]:

In [ ]:

In [13]:
```
import pandas as pd
import sklearn as sk
import math
```

In [14]:
```
first_sentence="Data Science is one of the top job of the 21st century"
second_sentence="Machine Learning is the key for Data Science"
```

In [15]:
```
first_sentence=first_sentence.split(" ")
second_sentence=second_sentence.split(" ")

total=set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'the', 'key', 'for', 'Data', '21st', 'Science', 'century', 'of', 'Learning',
'Machine', 'one', 'job', 'top', 'is'}
```

In [16]:
```
wordDictA=dict.fromkeys(total,0)
wordDictB=dict.fromkeys(total,0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1
```

In [17]:
```
pd.DataFrame([wordDictA,wordDictB])
```

Out[17]:

	the	key	for	Data	21st	Science	century	of	Learning	Machine	one	job	top	is
0	2	0	0	1	1	1	1	2	0	0	1	1	1	1
1	1	1	1	1	0	1	0	0	1	1	0	0	0	1

In [25]:
```
def computeTF(wordDict,doc):
    tfDict={}
    corpusCount=len(doc)
    for word,count in wordDict.items():
        tfDict[word]=count/float(corpusCount)
    return(tfDict)

tfFirst=computeTF(wordDictA,first_sentence)
tfSecond=computeTF(wordDictB,second_sentence)

tf=pd.DataFrame([tfFirst,tfSecond])
```

In [26]:
```
tf
```

Out[26]:

	the	key	for	Data	21st	Science	century	of	Learning	Machine
0	0.166667	0.000	0.000	0.083333	0.083333	0.083333	0.083333	0.166667	0.000	0.000
1	0.125000	0.125	0.125	0.125000	0.000000	0.125000	0.000000	0.000000	0.125	0.125

In [27]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=set(stopwords.words('english'))
filtered_sentence=[w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['key', 'Data', '21st', 'Science', 'century', 'Learning', 'Machine', 'one',
'job', 'top']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\acer\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [29]:
```
def computeIDF(docList):
    idfDict={}
    N=len(docList)

    idfDict=dict.fromkeys(docList[0].keys(),0)
    for word,val in idfDict.items():
        idfDict[word]=math.log10(N/(float(val)+1))

    return(idfDict)
idfs=computeIDF([wordDictA,wordDictB])
```

In [33]:
```
def computeTFIDF(tfBow,idfs):
    tfidf={}
    for word,val in tfBow.items():
        tfidf[word]=val*idfs[word]
    return(tfidf)

idfFirst=computeTFIDF(tfFirst,idfs)
idfSecond=computeTFIDF(tfSecond,idfs)

idf=pd.DataFrame([idfFirst,idfSecond])
print(idf)
```

```
        the       key       for      Data      21st   Science   century  \
0  0.050172  0.000000  0.000000  0.025086  0.025086  0.025086  0.025086
1  0.037629  0.037629  0.037629  0.037629  0.000000  0.037629  0.000000

         of  Learning   Machine       one       job       top        is
0  0.050172  0.000000  0.000000  0.025086  0.025086  0.025086  0.025086
1  0.000000  0.037629  0.037629  0.000000  0.000000  0.000000  0.037629
```

In [35]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
firstV = "Data Science is one of the top job of 21st Century"
secondV= "Machine Learning is the key for data science"

vectorize=TfidfVectorizer()

response=vectorize.fit_transform([firstV,secondV])
```

In [36]:
```
print(response)
```

```
  (0, 1)        0.3011696304075596
  (0, 0)        0.3011696304075596
  (0, 5)        0.3011696304075596
  (0, 13)       0.3011696304075596
  (0, 12)       0.21428467250457112
  (0, 9)        0.6023392608151192
  (0, 10)       0.3011696304075596
  (0, 4)        0.21428467250457112
  (0, 11)       0.21428467250457112
  (0, 2)        0.21428467250457112
  (1, 3)        0.40740123733358447
  (1, 6)        0.40740123733358447
  (1, 7)        0.40740123733358447
  (1, 8)        0.40740123733358447
  (1, 12)       0.28986933576883284
  (1, 4)        0.28986933576883284
  (1, 11)       0.28986933576883284
  (1, 2)        0.28986933576883284
```

In [ ]: