In [1]:
```
import nltk
nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("averaged_perceptron_tagger")
```

```
[nltk_data] Downloading package punkt to /home/student/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/student/nltk_dat
a...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[1]:
```
True
```

In [2]:
```
from nltk.tokenize import sent_tokenize, word_tokenize
```

In [3]:
```
corpus="Sachin was the GOAT of the previous generation. Virat is the
```

In [4]:
```
print(word_tokenize(corpus))
```

```
['Sachin', 'was', 'the', 'GOAT', 'of', 'the', 'previous', 'generati
on', '.', 'Virat', 'is', 'the', 'GOAT', 'of', 'this', 'generation',
'.', 'Shubman', 'will', 'be', 'the', 'GOAT', 'of', 'the', 'next',
'generation']
```

In [5]:
```
print(sent_tokenize(corpus))
```

```
['Sachin was the GOAT of the previous generation.', 'Virat is the G
OAT of this generation.', 'Shubman will be the GOAT of the next gen
eration']
```

In [6]:
```
from nltk.corpus import stopwords
```

In [7]:
```
stop_words=set(stopwords.words("english"))
```

In [8]:
```
print(stop_words)
```

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

In [9]:
```
tokens = word_tokenize(corpus)
cleaned_tokens = []
for token in tokens:
  if (token not in stop_words):
    cleaned_tokens.append(token)
print(cleaned_tokens)
```

```
['Sachin', 'GOAT', 'previous', 'generation', '.', 'Virat', 'GOAT',
'generation', '.', 'Shubman', 'GOAT', 'next', 'generation']
```

In [10]:
```
from nltk.stem import PorterStemmer
```

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

In [17]:
```
stemmer = PorterStemmer()
```

In [18]:
```
stemmed_tokens = []
for token in cleaned_tokens:
  stemmed = stemmer.stem(token)
  stemmed_tokens.append(stemmed)
print(stemmed_tokens)
```

```
['sachin', 'goat', 'previou', 'gener', '.', 'virat', 'goat', 'gener
', '.', 'shubman', 'goat', 'next', 'gener']
```

In [19]:
```
from nltk.stem import WordNetLemmatizer
```

In [20]:
```
from nltk.stem.wordnet import WordNetLemmatizer
```

In [21]:
```
lemmatizer = WordNetLemmatizer()
```

In [22]:
```
lemmatized_tokens = []
for token in cleaned_tokens:
  lemmatized = lemmatizer.lemmatize(token)
  lemmatized_tokens.append(lemmatized)
print(lemmatized_tokens)
```

```
['Sachin', 'GOAT', 'previous', 'generation', '.', 'Virat', 'GOAT',
'generation', '.', 'Shubman', 'GOAT', 'next', 'generation']
```

In [23]:
```
from nltk import pos_tag
```

In [25]:
```
tokens = word_tokenize(corpus)
print(pos_tag(tokens))
```

```
[('Sachin', 'NNP'), ('was', 'VBD'), ('the', 'DT'), ('GOAT', 'NNP'),
('of', 'IN'), ('the', 'DT'), ('previous', 'JJ'), ('generation', 'N
N'), ('.', '.'), ('Virat', 'NNP'), ('is', 'VBZ'), ('the', 'DT'),
('GOAT', 'NNP'), ('of', 'IN'), ('this', 'DT'), ('generation', 'N
N'), ('.', '.'), ('Shubman', 'NNP'), ('will', 'MD'), ('be', 'VB'),
('the', 'DT'), ('GOAT', 'NNP'), ('of', 'IN'), ('the', 'DT'), ('nex
t', 'JJ'), ('generation', 'NN')]
```

In [26]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [30]:
```
corpus = [
    "Sachin was the GOAT of the previous generation",
    "Virat is the GOAT of the this generation",
    "Shubman will be the GOAT of the next generation"
]
```

In [31]:
```
vectorizer = TfidfVectorizer()
```

In [34]:
```
matrix = vectorizer.fit(corpus)
matrix.vocabulary_
```

Out[34]:
```
{'sachin': 7,
 'was': 12,
 'the': 9,
 'goat': 2,
 'of': 5,
 'previous': 6,
 'generation': 1,
 'virat': 11,
 'is': 3,
 'this': 10,
 'shubman': 8,
 'will': 13,
 'be': 0,
 'next': 4}
```

In [35]:
```
tfidf_matrix = vectorizer.transform(corpus)
print(tfidf_matrix)
```

```
  (0, 12)       0.4286758743128819
  (0, 9)        0.5063657539459899
  (0, 7)        0.4286758743128819
  (0, 6)        0.4286758743128819
  (0, 5)        0.25318287697299496
  (0, 2)        0.25318287697299496
  (0, 1)        0.25318287697299496
  (1, 11)       0.4286758743128819
  (1, 10)       0.4286758743128819
  (1, 9)        0.5063657539459899
  (1, 5)        0.25318287697299496
  (1, 3)        0.4286758743128819
  (1, 2)        0.25318287697299496
  (1, 1)        0.25318287697299496
  (2, 13)       0.39400039808922477
  (2, 9)        0.4654059642457353
  (2, 8)        0.39400039808922477
  (2, 5)        0.23270298212286766
  (2, 4)        0.39400039808922477
  (2, 2)        0.23270298212286766
  (2, 1)        0.23270298212286766
  (2, 0)        0.39400039808922477
```

In [36]:
```
print(vectorizer.get_feature_names_out())
```

```
['be' 'generation' 'goat' 'is' 'next' 'of' 'previous' 'sachin' 'shu
bman'
 'the' 'this' 'virat' 'was' 'will']
```

In [38]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [39]:
```
from sklearn import metrics
```

In [ ]:

In [ ]: