```
import pandas as pd

import numpy as np

text='''Real madrid is set to win the UCL for the season . Benzema
might win Balon dor . Salah might be the runner up'''

text

{"type":"string"}

import nltk

nltk.download ('punkt')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

True
```

tokenization

```
from nltk import sent_tokenize , word_tokenize

sent_tokenize(text)

['Real madrid is set to win the UCL for the season .',
 'Benzema might win Balon dor .',
 'Salah might be the runner up']

word_list=word_tokenize(text)

word_list

['Real',
 'madrid',
 'is',
 'set',
 'to',
 'win',
 'the',
 'UCL',
 'for',
 'the',
 'season',
 '.',
 'Benzema',
 'might',
 'win',
 'Balon',
 'dor',
 '.',
 'Salah',
 'might',
 'be',
 'the',
 'runner',
 'up']
```

Stop Words Removal

```
from nltk.corpus import stopwords

nltk.download('stopwords')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

True

 stopword_list = stopwords.words('english')

stopword_list

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

```
word_list=[word.lower() for word in word_list]
```

```
filterword_list=[]
for word in word_list:
  if word not in stopword_list:
    filterword_list.append(word)
```

```
filterword_list
```

```
['real',
 'madrid',
 'set',
 'win',
 'ucl',
 'season',
 '.',
 'benzema',
 'might',
 'win',
 'balon',
 'dor',
 '.',
 'salah',
 'might',
 'runner']
```

```
w1=[word for word in word_list if word not in stopword_list ]
```

```
w1
```

```
['real',
 'madrid',
 'set',
 'win',
 'ucl',
 'season',
 '.',
 'benzema',
 'might',
 'win',
 'balon',
 'dor',
 '.',
 'salah',
 'might',
 'runner']
```

stemming

```
from nltk.stem import PorterStemmer

stemmer= PorterStemmer()

stem_words=[stemmer.stem (word) for word in filterword_list]

stem_words
```

```
['real',
 'madrid',
 'set',
 'win',
 'ucl',
 'season',
 '.',
 'benzema',
 'might',
 'win',
 'balon',
 'dor',
 '.',
 'salah',
 'might',
 'runner']
```

Lemmatization

```
from nltk.stem import WordNetLemmatizer

lemmatizer=WordNetLemmatizer()

nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
True
```

```
lemma_words=[lemmatizer.lemmatize(word) for word in filterword_list ]

lemma_words
```

```
['real',
 'madrid',
 'set',
 'win',
 'ucl',
 'season',
 '.',
 'benzema',
 'might',
 'win',
 'balon',
 'dor',
 '.',
 'salah',
 'might',
 'runner']
```

Part of Speech Tagging

```
from nltk import pos_tag

 nltk.download('averaged_perceptron_tagger')

[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!

True

tag_list=pos_tag(lemma_words)

tag_list

[('real', 'JJ'),
 ('madrid', 'NN'),
 ('set', 'VBN'),
 ('win', 'VBP'),
 ('ucl', 'JJ'),
 ('season', 'NN'),
 ('.', '.'),
 ('benzema', 'NN'),
 ('might', 'MD'),
 ('win', 'VB'),
 ('balon', 'NN'),
 ('dor', 'NN'),
 ('.', '.'),
 ('salah', 'NN'),
 ('might', 'MD'),
 ('runner', 'VB')]
```

Bag of Words

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer=CountVectorizer()

sentence_list=sent_tokenize(text)
```

```
sentence_list
```

```
['Real madrid is set to win the UCL for the season .',
 'Benzema might win Balon dor .',
 'Salah might be the runner up']
```

```
vectorizer.fit(sentence_list)
```

```
CountVectorizer()
```

```
print('Vocabulary',vectorizer.vocabulary_)
```

```
Vocabulary {'real': 8, 'madrid': 6, 'is': 5, 'set': 12, 'to': 14,
'win': 17, 'the': 13, 'ucl': 15, 'for': 4, 'season': 11, 'benzema': 2,
'might': 7, 'balon': 0, 'dor': 3, 'salah': 10, 'be': 1, 'runner': 9,
'up': 16}
```

Encoding

```
vector=vectorizer.transform(sentence_list)


print (vector.toarray())
```

```
[[0 0 0 0 1 1 1 0 1 0 0 1 1 2 1 1 0 1]
 [1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0]]
```

```
features=vectorizer.get_feature_names_out()
```

```
doc_list=['doc1','doc2','doc3']
```

```
df=pd.DataFrame(vector.toarray(),index=sorted(doc_list),columns=featur
es)
```

```
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"balon\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"be\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"benzema\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"dor\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"for\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"is\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"madrid\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"might\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"real\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"runner\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"salah\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"season\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"set\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
```

```
[\n          0,\n          1\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     },\n     {\n
\"column\": \"the\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          2,\n          0\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     },\n     {\n
\"column\": \"to\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     },\n     {\n
\"column\": \"ucl\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     },\n     {\n
\"column\": \"up\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     },\n     {\n
\"column\": \"win\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n         ],\n         \"semantic_type\":
\"\",\n         \"description\": \"\"\n       }\n     }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

Term Frequency and Inverse Document Frequency

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer1=TfidfVectorizer()

vectorizer1.fit(sentence_list)

TfidfVectorizer()

print ('Vocabulary',vectorizer1.vocabulary_)

Vocabulary {'real': 8, 'madrid': 6, 'is': 5, 'set': 12, 'to': 14,
'win': 17, 'the': 13, 'ucl': 15, 'for': 4, 'season': 11, 'benzema': 2,
'might': 7, 'balon': 0, 'dor': 3, 'salah': 10, 'be': 1, 'runner': 9,
'up': 16}

vector=vectorizer1.transform(sentence_list)

print (vector.toarray())
```

```
[[0.         0.         0.         0.         0.30300252 0.30300252
  0.30300252 0.         0.30300252 0.         0.         0.30300252
  0.30300252 0.46088245 0.30300252 0.30300252 0.         0.23044123]
 [0.49047908 0.         0.49047908 0.49047908 0.         0.
  0.         0.37302199 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.37302199]
 [0.         0.44036207 0.         0.         0.         0.
  0.         0.3349067  0.         0.44036207 0.44036207 0.
  0.         0.3349067  0.         0.         0.44036207 0.        ]]
```

```
print ('Features',vectorizer1.get_feature_names_out)
```

```
Features <bound method CountVectorizer.get_feature_names_out of
TfidfVectorizer()>
```

```
features=vectorizer.get_feature_names_out()

doc_list=['doc1','doc2','doc3']

df=pd.DataFrame(vector.toarray(),index=sorted(doc_list),columns=featur
es)

df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n
{\n      \"column\": \"balon\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.2831782312982749,\n
\"min\": 0.0,\n        \"max\": 0.49047908420610337,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
0.49047908420610337,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"be\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25424315805691916,\n
\"min\": 0.0,\n        \"max\": 0.4403620672313486,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
0.4403620672313486,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"benzema\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.2831782312982749,\n        \"min\": 0.0,\n        \"max\":
0.49047908420610337,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.49047908420610337,\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"dor\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.2831782312982749,\
n        \"min\": 0.0,\n        \"max\": 0.49047908420610337,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
0.49047908420610337,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"for\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.17493858549561722,\n
```

```
\"min\": 0.0,\n        \"max\": 0.30300251828264085,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0.0,\n
0.30300251828264085\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"is\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"madrid\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"might\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.20524809225067647,\n        \"min\": 0.0,\n        \"max\":
0.3730219858594306,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          0.0,\n          0.3730219858594306\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"real\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"runner\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.25424315805691916,\n        \"min\": 0.0,\n        \"max\":
0.4403620672313486,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.4403620672313486,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"salah\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.25424315805691916,\n        \"min\": 0.0,\n        \"max\":
0.4403620672313486,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.4403620672313486,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"season\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"set\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
```

```
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"the\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.23820335204341941,\n        \"min\": 0.0,\n        \"max\":
0.4608824503623661,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          0.4608824503623661,\n          0.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"to\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"ucl\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.17493858549561722,\n        \"min\": 0.0,\n        \"max\":
0.30300251828264085,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.0,\n          0.30300251828264085\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"up\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25424315805691916,\n
\"min\": 0.0,\n        \"max\": 0.4403620672313486,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
0.4403620672313486,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"win\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.18822762528080447,\n
\"min\": 0.0,\n        \"max\": 0.3730219858594306,\n
\"num_unique_values\": 3,\n        \"samples\": [\n
0.23044122518118304,\n          0.3730219858594306\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```