Data preprocessing in python Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

NLP Techniques in data science:

1. Tokenize text using NLTK in python
2. Removing stop words with NLTK in Python
3. Lemmatization with NLTK
4. Stemming words with NLTK

```
!pip install nltk -U
!pip install bs4 -U
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
para = 'In the magical world of Harry Potter, Hogwarts School of Witchcraft and Wizardry stands as a beacon of hope and learning for young witches and wi
print(para)
```

```
In the magical world of Harry Potter, Hogwarts School of Witchcraft and Wizardry stands as a beacon of hope and learning for young witches and wiza
```

```
para.split()
```

```
['In',
 'the',
 'magical',
 'world',
 'of',
 'Harry',
 'Potter,',
 'Hogwarts',
 'School',
 'of',
 'Witchcraft',
 'and',
 'Wizardry',
 'stands',
 'as',
 'a',
 'beacon',
 'of',
 'hope',
 'and',
 'learning',
 'for',
 'young',
 'witches',
 'and',
 'wizards.']
```

```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
```

```
#number of sentences
sent = sent_tokenize(para)
sent[0]
```

```
'In the magical world of Harry Potter, Hogwarts School of Witchcraft and Wizardry stands as a beacon of hope and learning for young witches and wi
zards.'
```

```
words = word_tokenize(para)
words
```

```
['In',
 'the',
 'magical',
 'world',
 'of',
 'Harry',
 'Potter',
 ',',
 'Hogwarts',
 'School',
 'of',
 'Witchcraft',
 'and',
 'Wizardry',
 'stands',
 'as',
 'a',
 'beacon',
 'of',
 'hope',
 'and',
 'learning',
 'for',
 'young',
 'witches',
 'and',
 'wizards',
 '.']
```

```
from nltk.corpus import stopwords
swords = stopwords.words('english')
swords
```

```
      '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_without_swords=[word for word in words if word not in swords]
word_without_swords
```

```
     ['In',
      'magical',
      'world',
      'Harry',
      'Potter',
      ',',
      'Hogwarts',
      'School',
      'Witchcraft',
      'Wizardry',
      'stands',
      'beacon',
      'hope',
      'learning',
      'young',
      'witches',
      'wizards',
      '.']
```

```
x = [word for word in words if word.lower() not in swords]
x
```

```
['magical',
 'world',
 'Harry',
 'Potter',
 ',',
 'Hogwarts',
 'School',
 'Witchcraft',
 'Wizardry',
 'stands',
 'beacon',
 'hope',
 'learning',
 'young',
 'witches',
 'wizards',
 '.']
```

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
ps.stem('working')
```

```
'work'
```

```
y=[ps.stem(word) for word in x]
y
```

```
['magic',
 'world',
 'harri',
 'potter',
 ',',
 'hogwart',
 'school',
 'witchcraft',
 'wizardri',
 'stand',
 'beacon',
 'hope',
 'learn',
 'young',
 'witch',
 'wizard',
 '.']
```

```
from nltk.stem import WordNetLemmatizer
wnl  = WordNetLemmatizer()
wnl.lemmatize('working', pos = 'v')
```

```
'work'
```

```
print(ps.stem('went'))
print(wnl.lemmatize('went',pos = 'v'))
```

```
went
go
```

```
z= [wnl.lemmatize(word , pos = 'v') for word in x]
z
```

```
['magical',
 'world',
 'Harry',
 'Potter',
 ',',
 'Hogwarts',
 'School',
 'Witchcraft',
 'Wizardry',
 'stand',
 'beacon',
 'hope',
 'learn',
 'young',
 'witch',
 'wizards',
 '.']
```

```
import string
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
t= [word for word in words if word not in string.punctuation]
t
```

```
['In',
 'the',
 'magical',
 'world',
 'of',
 'Harry',
 'Potter',
 'Hogwarts',
 'School',
 'of',
 'Witchcraft',
 'and',
 'Wizardry',
 'stands',
 'as',
 'a',
 'beacon',
 'of',
 'hope',
 'and',
 'learning',
 'for',
 'young',
 'witches',
 'and',
 'wizards']
```

```
from nltk import pos_tag
pos_tag(t)
```

```
[('In', 'IN'),
 ('the', 'DT'),
 ('magical', 'JJ'),
 ('world', 'NN'),
 ('of', 'IN'),
 ('Harry', 'NNP'),
 ('Potter', 'NNP'),
 ('Hogwarts', 'NNP'),
 ('School', 'NNP'),
 ('of', 'IN'),
 ('Witchcraft', 'NNP'),
 ('and', 'CC'),
 ('Wizardry', 'NNP'),
 ('stands', 'VBZ'),
 ('as', 'IN'),
 ('a', 'DT'),
 ('beacon', 'NN'),
 ('of', 'IN'),
 ('hope', 'NN'),
 ('and', 'CC'),
 ('learning', 'NN'),
 ('for', 'IN'),
 ('young', 'JJ'),
 ('witches', 'NNS'),
 ('and', 'CC'),
 ('wizards', 'NNS')]
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
v= tfidf.fit_transform(t)
v.shape
```

```
(26, 21)
```

```
import pandas as pd
pd.DataFrame(v)
```

	0
0	(0, 7)\t1.0
1	(0, 14)\t1.0
2	(0, 9)\t1.0
3	(0, 19)\t1.0
4	(0, 10)\t1.0
5	(0, 4)\t1.0
6	(0, 11)\t1.0
7	(0, 5)\t1.0
8	(0, 12)\t1.0
9	(0, 10)\t1.0
10	(0, 15)\t1.0
11	(0, 0)\t1.0
12	(0, 17)\t1.0
13	(0, 13)\t1.0
14	(0, 1)\t1.0
15	
16	(0, 2)\t1.0
17	(0, 10)\t1.0
18	(0, 6)\t1.0
19	(0, 0)\t1.0
20	(0, 8)\t1.0
21	(0, 3)\t1.0
22	(0, 20)\t1.0
23	(0, 16)\t1.0
24	(0, 0)\t1.0
25	(0, 18)\t1.0