```
# visualization --> matplotlib / seaborn
# dataset --> pandas
# multi-dimensional array --> numpy

# nlp --> nltk --> natural language tool kit
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\hp\anaconda3\lib\site-
packages (3.8.1)
Requirement already satisfied: click in c:\users\hp\anaconda3\lib\
site-packages (from nltk) (8.0.4)
Requirement already satisfied: joblib in c:\users\hp\anaconda3\lib\
site-packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\hp\
anaconda3\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\hp\anaconda3\lib\site-
packages (from nltk) (4.65.0)
Requirement already satisfied: colorama in c:\users\hp\anaconda3\lib\
site-packages (from click->nltk) (0.4.6)
```

Tokenization

```
# breaking the sentence into words

sentence = "my name is munira fatima."

print(sentence.split())
```

```
['my', 'name', 'is', 'munira', 'fatima.']
```

```
import nltk
from nltk import word_tokenize, sent_tokenize

# corpora --> body of text --> News/ English language
# corpus --> collection of sentence -> [s1, s2, s3, s4....]

nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-
pages/index.xml
```

```
True
```

```
nltk.download("brown")
```

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

```
True
```

```
nltk.download("punkt")
```

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

```
True
```

```
print(word_tokenize(sentence))
```

```
['my', 'name', 'is', 'munira', 'fatima', '.']
```

```
# tokenization --> split() // word_tokenize?
# list -- > np.array --> faster
# split()
```

```
sentences = "My name is munira fatima. I am a data scientist. I am a
passionate data science mentor."
```

```
print(sent_tokenize(sentences))
```

```
['My name is munira fatima.', 'I am a data scientist.', 'I am a
passionate data science mentor.']
```

```
print(word_tokenize(sentences))
```

```
['My', 'name', 'is', 'munira', 'fatima', '.', 'I', 'am', 'a', 'data',
'scientist', '.', 'I', 'am', 'a', 'passionate', 'data', 'science',
'mentor', '.']
```

Stopwords

```
nltk.download("stopwords")
```

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

```
True
```

```
from nltk.corpus import stopwords

sw = set(stopwords.words("english"))
print(sw)
```

```
{'ours', 'a', "needn't", 'own', 'hasn', 'aren', 'over', "couldn't",
"mightn't", 'yourselves', 'we', 'which', 'he', 'are', 'should',
'before', "that'll", 'it', 'by', 'their', "should've", 'just', 'who',
'because', 'of', 'them', 'to', 'is', "hadn't", 'have', 'so', 'your',
"she's", 'isn', "it's", 'hadn', 'whom', 'then', "won't", 'above',
"didn't", "isn't", 'has', 'no', 'once', 'same', 'on', 'll', 'both',
```

```
'than', 'be', 'where', 'having', 'these', 'further', 'was', 'were',
'did', 'most', "you'd", 'him', "wouldn't", 'do', 'up', 'in', 'me',
'his', 'for', 'when', 'other', 'why', "shouldn't", 'that', 'shouldn',
'those', 'against', 'the', 'theirs', 'mustn', 'her', 't', 'weren',
'my', 'ourselves', 'our', 'its', 'herself', 'as', 'they', 'such',
'if', 'an', 'm', 'or', 'more', 'after', "you're", 've', 'didn',
'mightn', 'at', 'o', 'with', 'will', 'doing', 'am', 'out', 'you',
'himself', 'into', 'now', 'and', 'does', 'couldn', 'haven', 'hers',
'about', 'off', "aren't", 'all', 'during', 'wasn', 'shan', "don't",
'don', 'this', 'few', "you've", 'ma', "weren't", 'but', 'being',
'how', 'been', 'under', 'yours', 'only', 'myself', 'themselves',
'any', 'won', 'while', 'each', 'can', 'down', 'yourself', 'y',
'wouldn', "doesn't", 'had', 'below', 'ain', 'what', 'from', 'itself',
's', "shan't", 'some', 'until', 'between', "wasn't", 'she', 'here',
're', "you'll", 'd', 'i', 'there', 'doesn', 'needn', 'very', 'again',
'not', "hasn't", "mustn't", 'nor', 'through', 'too', "haven't"}

print(sentences)
```

```
My name is munira fatima. I am a data scientist. I am a passionate
data science mentor.
```

```
review = "Sunsilk is a good shampoo. Sunsilk is a bad shampoo. I donot
like shampoo"

sent_token = sent_tokenize(review)
print(sent_token)
```

```
['Sunsilk is a good shampoo.', 'Sunsilk is a bad shampoo.', 'I donot
like shampoo']
```

```
word_token = word_tokenize(review)
print(word_token)
```

```
['Sunsilk', 'is', 'a', 'good', 'shampoo', '.', 'Sunsilk', 'is', 'a',
'bad', 'shampoo', '.', 'I', 'donot', 'like', 'shampoo']
```

```
filter_word_token = []
for word in word_token:
    if word not in sw:
        filter_word_token.append(word)
print(filter_word_token)
```

```
['Sunsilk', 'good', 'shampoo', '.', 'Sunsilk', 'bad', 'shampoo', '.',
'I', 'donot', 'like', 'shampoo']
```

```
filter_word = [word for word in word_token if word not in sw]

print(filter_word)
```

```
['Sunsilk', 'good', 'shampoo', '.', 'Sunsilk', 'bad', 'shampoo', '.',
'I', 'donot', 'like', 'shampoo']
```

```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

```
True
```

Stemming

```
from nltk.stem import PorterStemmer

porter = PorterStemmer()

my_words = ["change","changes","changing","changed"]

for word in my_words:
    print(porter.stem(word))
```

```
chang
chang
chang
chang
```

Lemmatization

```
nltk.download("wordnet")
```

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

```
True
```

```
from nltk.stem.wordnet import WordNetLemmatizer

lemm= WordNetLemmatizer()

for word in my_words:
    print(lemm.lemmatize(word))
```

```
change
change
```

```
changing
changed
```

```
name = ["leafs","leaves"]
for word in name:
    print(lemm.lemmatize(word))
```

```
leaf
leaf
```

```
name = ["run","ran","runs","running"]
for word in name:
    print(lemm.lemmatize(word))
```

```
run
ran
run
running
```

```
# v1 v2 v3 v4 v5
# run ran runned running runs


word_list = ["go","went","gone","going","goes"]

for word in word_list:
    print(lemm.lemmatize(word))
```

```
go
went
gone
going
go
```

```
# How we can use lemmatization in real world problem?

sentences = "My name is munira fatima and I have achieve so many
medals in my career. My name is munira fatima and I have achieves so
many medals in my career. My name is munira fatima and I have achieved
so many medals in my career. My name is munira fatima and I have
achieve so many medals in my career. My name is munira fatima and I
have achieved so many medals in my career. My name is munira fatima
and I have achieving so many medals in my career. My name is munira
fatima and I have achievement so many medals in my career. My name is
munira fatima and I have achieves so many medals in my career. My name
is munira fatima and I have achieving so many medals in my career. "

print(sentences)
```

```
My name is munira fatima and I have achieve so many medals in my
career. My name is munira fatima and I have achieves so many medals in
my career. My name is munira fatima and I have achieved so many medals
```

```
in my career. My name is munira fatima and I have achieve so many
medals in my career. My name is munira fatima and I have achieved so
many medals in my career. My name is munira fatima and I have
achieving so many medals in my career. My name is munira fatima and I
have achievement so many medals in my career. My name is munira fatima
and I have achieves so many medals in my career. My name is munira
fatima and I have achieving so many medals in my career.

word_token = word_tokenize(sentences)
print(word_token)

['My', 'name', 'is', 'munira', 'fatima', 'and', 'I', 'have',
'achieve', 'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My',
'name', 'is', 'munira', 'fatima', 'and', 'I', 'have', 'achieves',
'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is',
'munira', 'fatima', 'and', 'I', 'have', 'achieved', 'so', 'many',
'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is', 'munira',
'fatima', 'and', 'I', 'have', 'achieve', 'so', 'many', 'medals', 'in',
'my', 'career', '.', 'My', 'name', 'is', 'munira', 'fatima', 'and',
'I', 'have', 'achieved', 'so', 'many', 'medals', 'in', 'my', 'career',
'.', 'My', 'name', 'is', 'munira', 'fatima', 'and', 'I', 'have',
'achieving', 'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My',
'name', 'is', 'munira', 'fatima', 'and', 'I', 'have', 'achievement',
'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is',
'munira', 'fatima', 'and', 'I', 'have', 'achieves', 'so', 'many',
'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is', 'munira',
'fatima', 'and', 'I', 'have', 'achieving', 'so', 'many', 'medals',
'in', 'my', 'career', '.']

new_word = []
for each_word in word_token:
    if(each_word.startswith("ach")):
        each_word = each_word.replace(each_word,"achieve")
    new_word.append(each_word)

print(new_word)

['My', 'name', 'is', 'munira', 'fatima', 'and', 'I', 'have',
'achieve', 'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My',
'name', 'is', 'munira', 'fatima', 'and', 'I', 'have', 'achieve', 'so',
'many', 'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is',
'munira', 'fatima', 'and', 'I', 'have', 'achieve', 'so', 'many',
'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is', 'munira',
'fatima', 'and', 'I', 'have', 'achieve', 'so', 'many', 'medals', 'in',
'my', 'career', '.', 'My', 'name', 'is', 'munira', 'fatima', 'and',
'I', 'have', 'achieve', 'so', 'many', 'medals', 'in', 'my', 'career',
'.', 'My', 'name', 'is', 'munira', 'fatima', 'and', 'I', 'have',
'achieve', 'so', 'many', 'medals', 'in', 'my', 'career', '.', 'My',
'name', 'is', 'munira', 'fatima', 'and', 'I', 'have', 'achieve', 'so',
'many', 'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is',
```

```
'munira', 'fatima', 'and', 'I', 'have', 'achieve', 'so', 'many',
'medals', 'in', 'my', 'career', '.', 'My', 'name', 'is', 'munira',
'fatima', 'and', 'I', 'have', 'achieve', 'so', 'many', 'medals', 'in',
'my', 'career', '.']
```

```
words = ["run","achieve","change","beauty"]
```

Parts of Speech pos_tags

```
from nltk.corpus import state_union

from nltk.tokenize import PunktSentenceTokenizer


nltk.download("state_union")
```

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

```
True
```

```
print(sentences)
```

```
My name is munira fatima and I have achieve so many medals in my
career. My name is munira fatima and I have achieves so many medals in
my career. My name is munira fatima and I have achieved so many medals
in my career. My name is munira fatima and I have achieve so many
medals in my career. My name is munira fatima and I have achieved so
many medals in my career. My name is munira fatima and I have
achieving so many medals in my career. My name is munira fatima and I
have achievement so many medals in my career. My name is munira fatima
and I have achieves so many medals in my career. My name is munira
fatima and I have achieving so many medals in my career.
```

```
from nltk import pos_tag

sentence = "I am enjyoing my vacation"

token = sentence.split()



new_sentence = []
for value in pos_tag(token):
    if(value[1].startswith("VB") or value[1].startswith("RB") or
value[1].startswith("JJ")):
        new_sentence.append(value[0])
```

```
new_sent = " ".join(new_sentence)
print(new_sent)
```

```
am enjyoing
```

```
# JJ - Adjective
# NN - Nouns
# RB - Adverbs
# PRP - Pronouns
# VB - Verbs

for value in pos_tag(token):
    print(value)
```

```
('I', 'PRP')
('am', 'VBP')
('enjyoing', 'VBG')
('my', 'PRP$')
('vacation', 'NN')
```

```
# new_word
# JJ //

sentences = "Sunsilk is a nice product. The imported oranges are
heavily toxic. Apple iphone has improved battery life."
word_token = word_tokenize(sentences)
sent_token = sent_tokenize(sentences)


for sent in sent_token:
    value = sent.split()
    val = pos_tag(value)
    for i in val:
        if(i[1].startswith("JJ") or i[1].startswith("VB")):
            print(i)
```

```
('is', 'VBZ')
('nice', 'JJ')
('imported', 'VBN')
('are', 'VBP')
('toxic.', 'JJ')
('has', 'VBZ')
('improved', 'VBN')
```

```
preprocessed_sentences = " is nice . imported are toxic. has improved
"

word_
```