



NEWS CLASSIFICATION USING Natural language processing

CONCEPTS INVOLVED

- Supervised Learning
- Text pre-processing
- Vectorization
- Scikit learn NLP imports
- Natural Language Tool Kit (NLTK)
- ML Classifiers

OBJECTIVE

This project aims at building a model which classifies whether the news given is genuine news or false news by using Natural Language Processing .



DETAILS OF DATA SET

We used datasets

- fake.csv
- true .csv

COLUMNS :

[TITLE, TEXT ,SUBJECT, DATE]

NO-of- rows :

Fake.csv- 23481

True.csv- 21417

THERE are different subjects involved

-----false

News	9050
politics	6841
left-news	4459
Government News	1570
US_News	783
Middle-east	778

-----True

politicsNews	11272
worldnews	10145

SUPERVISED LEARNING

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

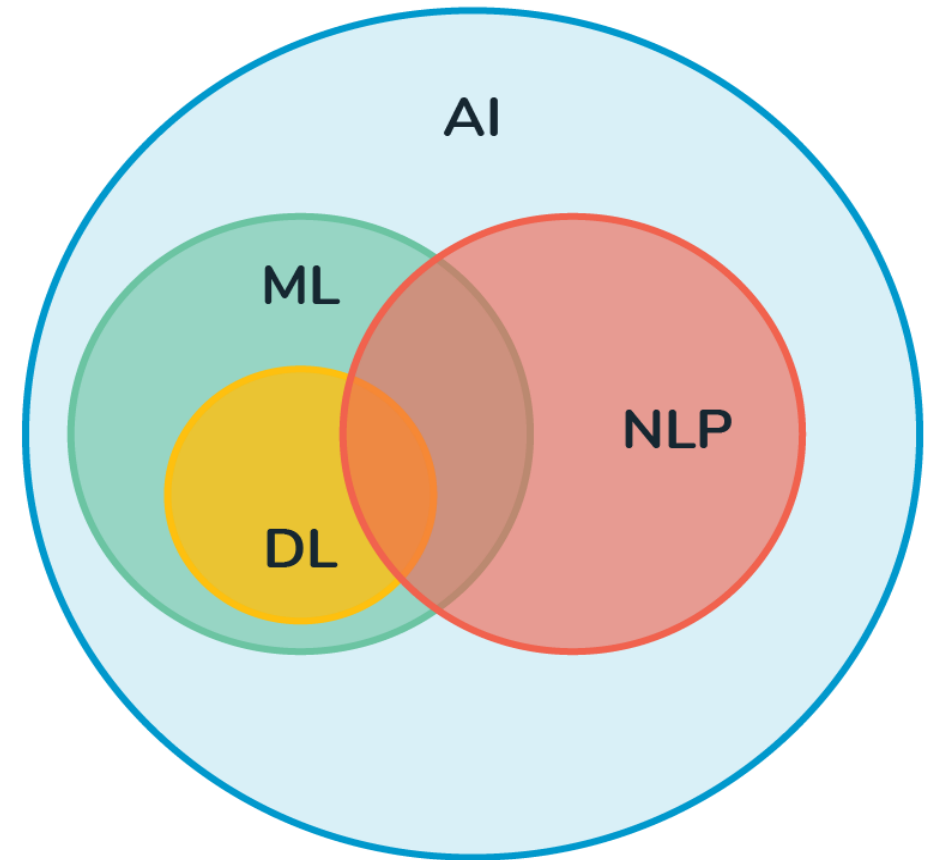
The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.


All classifications comes under supervised learning and this project will also be supervised learning i.e all category outputs comes under this

“THIS PROJECT CLASSIFIES WHETHER NEWS IS GENUINE OR NOT “

NLP – NATURAL LANGUAGE PROCESSING

- **NLP** is a branch of **Artificial Intelligence (AI)** that studies how machines understand human language.
- Its goal is to build systems that can make sense of text and perform tasks like translation, grammar checking, or topic classification.



- 
- This is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. Natural languages could be any human languages like English, Hindi, Telugu, Tamil.
 - **This** project undergoes **text analytics** which is an artificial intelligence (AI) technology that uses **natural language processing (NLP)** to transform the free (unstructured) **text** in documents and databases into normalized, structured data suitable for **analysis** or to drive machine learning (ML) algorithms.
 - We have different packages and libraries for Natural Language Processing

NLP Libraries

- **NLTK**

Natural Language Tool Kit (NLTK) is an essential library supports tasks such as classification, stemming, tagging, parsing, semantic reasoning, and tokenization in Python. It's basically your main tool for natural language processing and machine learning.



- **SCI-KIT LEARN**

This handy NLP library provides developers with a wide range of algorithms for building machine learning models. It offers many functions for using the bag-of-words method of creating features to tackle text classification problems.



NATURAL LANGUAGE TOOL KIT

NLTK is a really powerful tool to preprocess text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with. NLTK has been called “a wonderful tool for teaching and working in computational linguistics using Python,” and “an amazing library to play with natural language.”

REQUIREMENTS :

- **Install NLTK**

```
>>>pip install nltk
```

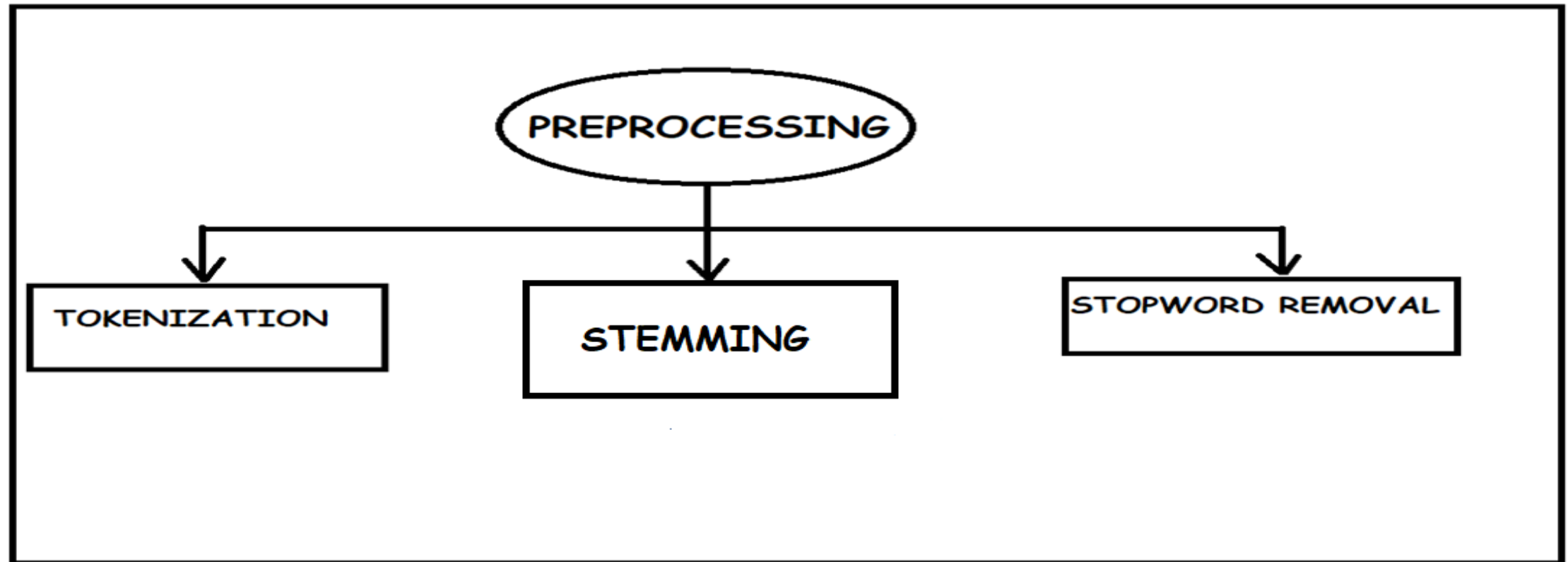
- **Import NLTK**

```
>>>import nltk
```

- **Download NLTK**

```
>>>nltk.download()
```

TEXT PREPROCESSING



TOKENIZATION

This process divides a large piece of continuous text into distinct units or tokens basically this process is often known as tokenization.

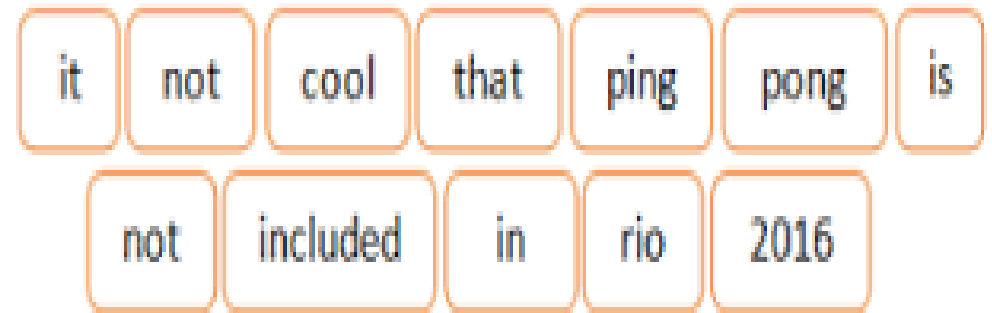
NLTK provides a **function** called **word_tokenize()** for splitting strings into tokens (nominally words). It splits tokens based on white space and punctuation.

Requirement:
`from nltk.tokenize import word_tokenize`

it not cool that ping pong is not included in rio 2016



Tokenization



STEMMING

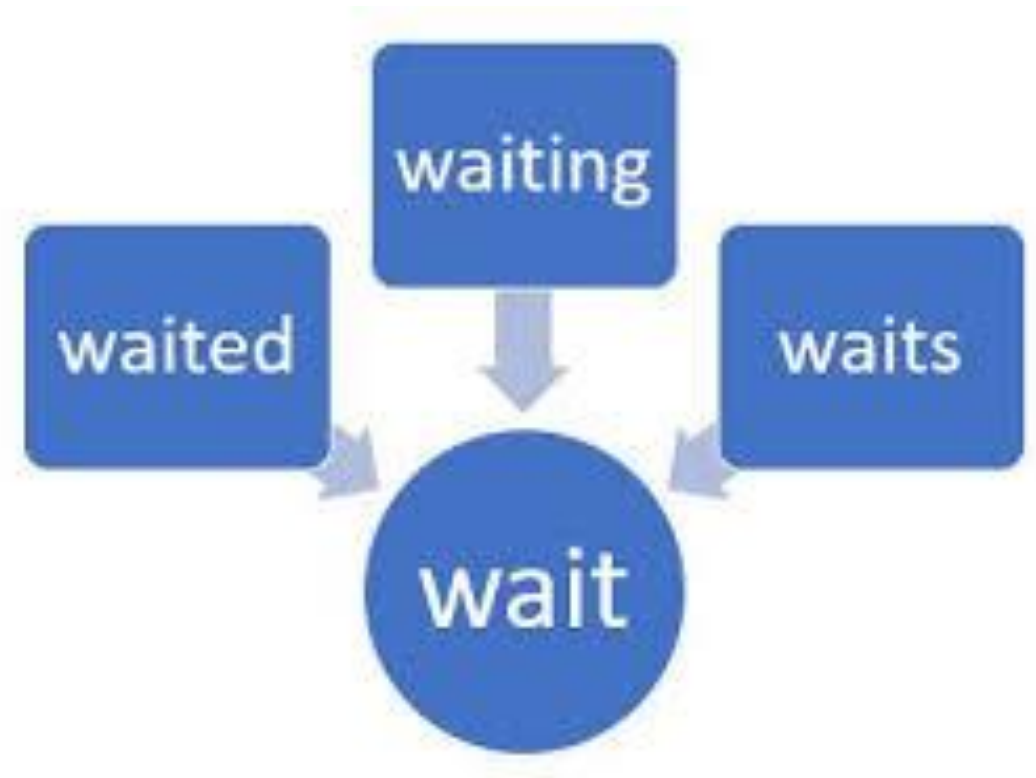
This is the idea of removing the suffix of a word and reducing different forms of a word to a core root.

REQUIREMENTS :

- **stemmer from nltk.stem package**

There are different stemmers in this package

- 1) Snowball
- 2) Porter
- 3) Lancaster



STOPWORDS REMOVAL

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore.

REQUIREMENTS :

- Import stopwords from nltk.corpus

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

```
from nltk.corpus import stopwords  
nltk.download('stopwords')
```

```
language = "english"  
stop_words = set(stopwords.words(language))  
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to  
[nltk_data] /Users/shubham/nltk_data...
```

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

```
[nltk_data] Unzipping corpora/stopwords.zip.
```

VECTORIZATION

The **scikit-learn** library offers easy-to-**use** tools to perform feature extraction of your text data that is Vectorization.

The vectorization is a technique used to convert textual data to numerical format. Using vectorization, a matrix is created where each column represents a feature and each row represents an individual review.

TF (Term Frequency)

Term Frequency is defined as how frequently the word appear in the document .

$$TF = \frac{\text{No of time word appear in the document}}{\text{Total no of word in the document}}$$

Term Frequency-Inverse Document Frequency(TF-IDF)

TD-IDF basically tells importance of the word in the corpus or dataset

- It is the combination of Term frequency and **Inverse Document Frequency** .
- Inverse Document frequency is another concept which is used for finding out importance of the word. It is based on the fact that less frequent words are more informative and important.

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

- $w_{i,j}$ =weight which signifies how important a word is for individual text message

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

Requirements:

Import TfidfVectorizer from sklearn

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

Documents



Vector-space representation

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term-document matrix

Count Vectorizer

	blue	bright	sky	sun
Doc1	1	0	1	0
Doc2	0	1	0	1

TD-IDF Vectorizer

	blue	bright	sky	sun
Doc1	0.707107	0.000000	0.707107	0.000000
Doc2	0.000000	0.707107	0.000000	0.707107

CLASIFICATION ALGORITHMS

Passive-aggressive classification is one of the available incremental learning algorithms and it is very simple to implement

Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.

- **Aggressive:** If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Logistic Regression

- Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.
- *logistic function* $1 / (1 + \exp(-x))$ normalizes everything to be between 0 and 1. Then we can interpret the number you get as a probability.

REQUIREMENTS:

- `Import Logistic regression from sklearn.linear_model`
`>>> from sklearn.linear_model import`
`LogisticRegression`

