

A Socially Relevant Project Report

On

FAKE NEWS DETECTION

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

BY

YAMANURI PRASANTH

(19BQ1A12I6)

YATHIRAJULA ANIL KUMAR

(19BQ1A12I8)

PULIPATI KARUN JOEL

(19BQ1A12D4)

VEMURI RAVI ARYAN

(19BQ1A12I0)

Under the esteemed guidance of

MR. N. ASHOK M. TECH (Ph. d)

Assistant Professor of IT



DEPARTMENT OF INFORMATION TECHNOLOGY

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Approved by AICTE and an Autonomous Institution affiliated to JNTUK,

Accredited by NAAC with 'A' grade, Accredited by NBA for 3 years

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508

Tel no:0863-2118036, [url:www.vvitguntur.com](http://www.vvitguntur.com)

SEPTEMBER 2021

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
NAMBUR



BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **YAMANURI PRASANTH, YATHIRAJULA ANIL KUMAR, PULIPATI KARUN JOEL, VEMURI RAVI ARYAN** Reg No. 19BQ1A12I6 , 19BQ1A12I8 , 19BQ1A12D4 , 19BQ1A12I0 Who carried out the project entitled **“FAKE NEWS DETECTION”** under our supervision during the year **2020-2021**.

PROJECT GUIDE

MR. N. ASHOK M. TECH (Ph. d)

Assistant Professor of IT

HEAD OF THE DEPARTMENT

DR. KALAVATHI ALLA, Ph. D

Professor

External Viva voice conducted on _____

Internal Examiner

External Examiner

DECLARATION

We **YAMANURI PRASANTH , YATHIRAJULA ANIL KUMAR , PULIPATI KARUN JOEL, VEMURI RAVI ARYAN** hereby declare that the project report entitled "**FAKE NEWS DETECTION**" done by us under the guidance of **Mr. N. ASHOK**, M . Tech (Ph. d), assistant Professor in **VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY** is submitted in fulfilment of their acquirements for the award of degree in **INFORMATION TECHNOLOGY**.The results embodied in this project have not been submitted to any other university or college for the award of any degree or diploma.

DATE:

PLACE: NAMBUR

SIGNATURE OF THE CANDIDATES

(YAMANURI PRASANTH-19BQ1A12I6)

(YATHIRAJULA ANIL KUMAR-19BQ1A12I8)

(PULIPATI KARUN JOEL-19BQ1A12D4)

(VEMURI RAVI ARYAN-19BQ1A12I0)

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people , who made this project work easier with words of encouragement , motivation , discipline , and faith by offering different places to lookto expand my ideas and helpedme towards the successful completion of this project work.

First and foremost,we express our deep gratitude and appreciationto **Sri.Vasireddy Vidyasagar Chairman**, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the infor-Mation Technology Program.

We express our sincere thanks to **Dr. Y. Mallikarjuna Reddy, Principal**, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the Information Technology program.

We express our sincere gratitude to **Dr. A. Kalavathi, Professor & HOD**,Information Tech-nology, Vasireddy Venkatadri Institute of Technology for her constant Encouragement, motivation and faith by offering different places to look to expand my ideas. We would like to express our sincere g ratefulness to our guide **MR.N.ASHOK Assistant Professor** for his insightful advice,motivating suggestions, invaluable guidance help and support in successful completion of this project And also our project co-ordinator **Ms. J. Sudeepthi Assistant Professor** for heradvice and support.We would like to take this opportunity to express our thanks to the Teaching and non teaching staff in Department of Information Technology , VVIT for their invaluable help and support.

ABSTRACT

FAKE NEWS DETECTION

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in the human history before. With the current usage of social media platforms, consumers are creating and sharing more information than ever before, some of which are misleading with no relevance to reality. Automated classification of a text article as misinformation or disinformation is a challenging task.

Even an expert in a particular domain has to explore multiple aspects before giving a verdict on the truthfulness of an article. In this work, we propose to use machine learning ensemble approach for automated classification of news article. Our study explores different textual properties that can be used to distinguish fake contents from real.

By using those properties, we train a combination of different machine learning algorithms using various ensemble methods and evaluate their performance on 4 real world data sets. Experimental evaluation confirms the superior performance of our proposed ensemble learner approach in comparison to individual learners.

LIST OF CONTENTS

CONTENT	PAGE NO.
1.INTRODUCTION	1-3
1.1-INTRODUCTION TO PROJECT	
1.2-EXISTING SYSTEM	
1.3-PROPOSED SYSTEM	
1.4-OBJECTIVE	
1.5-HARDWARE REQUIREMENTS	
1.6-SOFTWARE REQUIREMENTS	
2.NATURAL LANGUAGE PROCESSING	4-5
2.1-INTRODUCTION TO NLP	
2.2-NLTK	
3.NLP PIPELINE	6-22
3.1-TOKENIZATION	
3.2-TEXT CLEANING	
3.3-VECTORIZATION	
3.4-MACHINE LEARNING	

4. LIBRARIES	23-41
4.1-NUMPY	
4.2-PANDAS	
4.3-MATPLOTLIB	
4.4-SEABORN	
4.5-SCIKIT LEARN	
5. MACHINE LEARNING ALORITHMS	42-43
5.1-NAIVE BAYES ALGORITHM	
5.2-PASSIVE AGGRESSIVE ALGORITHM	
6. DATA FLOW DIAGRAM	44-45
7. IMPLEMENTATION	46-47
7.1-PSEUDO CODE	
7.2-OUTPUTS	
8. CONCLUSION	48

LIST OF FIGURES

FIG 1.1	6
FIG 1.2	44
FIG 1.3	47

ABBREVIATIONS

NLP- NATURAL LANGUAGE PROCESSING

NLTK- NATURAL LANGUAGE TOOL KIT

1.1-Introduction To Project

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in the human history before. Besides other use cases, news outlets benefitted from the widespread use of social media platforms by providing updated news in near real time to its subscribers. The news media evolved from newspapers, tabloids, and magazines to a digital form such as online news platforms, blogs, social media feeds, and other digital media formats . It became easier for consumers to acquire the latest news at their fingertips. Facebook referrals account for 70% of traffic to news websites . These social media platforms in their current state are extremely powerful and useful for their ability to allow users to discuss and share ideas and debate over issues such as democracy, education, and health. However, such platforms are also used with a negative perspective by certain entities commonly for monetary gain and in other cases for creating biased opinions, manipulating mindsets, and spreading satire or absurdity. The phenomenon is commonly known as fake news.

There has been a rapid increase in the spread of fake news in the last decade, most prominently observed in the 2016 US elections . Such proliferation of sharing articles online that do not conform to facts has led to many problems not just limited to politics but covering various other domains such as sports, health, and also science . One such area affected by fake news is the financial markets , where a rumor can have disastrous consequences and may bring the market to a halt.

Our ability to take a decision relies mostly on the type of information we consume; our world view is shaped on the basis of information we digest. There is increasing evidence that consumers have reacted absurdly to news that later proved to be fake . One recent case is the spread of novel corona virus, where fake reports spread over the Internet about the origin, nature, and behavior of the virus . The situation worsened as more people read about the fake contents online. Identifying such news online is a daunting task.

Fortunately, there are a number of computational techniques that can be used to mark certain articles as fake on the basis of their textual content. Majority of these techniques use fact checking websites such as “PolitiFact” and “Snopes.” There are a number of repositories maintained by researchers that contain lists of websites that are identified as ambiguous and fake . However, the problem with these resources is that human expertise is required to identify articles/websites as fake. More importantly, the fact checking websites contain articles from particular domains such as politics and are not generalized to identify fake news articles from multiple domains such as entertainment, sports, and technology.

The World Wide Web contains data in diverse formats such as documents, videos, and audios. News published online in an unstructured format (such as news, articles, videos, and audios) is relatively difficult to detect and classify as this strictly requires human expertise. However, computational techniques such as natural language processing (NLP) can be used to detect anomalies that separate a text article that is deceptive in nature from articles that are based on facts . Other techniques involve the analysis of propagation of fake news in contrast with real news . More specifically, the approach analyzes how a fake news article propagates differently on a network relative to a true article. The response that an article gets can be differentiated at a theoretical level to classify the article as real or fake.

1.2-Existing System

Using Bag of words, stop words techniques in Natural language processing, and using Machine learning naïve bayes algorithm to get most fake words and most real words. Fake words are having higher negative coefficient it means any sentence or text contain that particular word may have higher chances of being faked...

1.3-Proposed System

News Authenticator

New authenticator follows some steps to check whether the news is true or false. It will compare news which is given by our side with different websites and various news sources if that news is found on any news website then it shows the given news is true, else it shows there has been no such news in last few days. This can help us from fake news. These days "fake news" spread very fast because of social media and the internet. So, news authenticator helps us to detect either the given news is fake or real.

News Suggestion / Recommendation System

News suggestion suggests recent news and suggests the news related to the news which the user has given for authentication. If the news is fake, then this news suggestion gives the related news on that topic. The news suggestion suggests the news based on keywords which you give in your news based on keywords which you give in your news based on keywords which you give in your news which you wish to authenticate

1.4-Objective

The main objective is to detect the fake news, which is a classic text classification problem with a straight forward proposition. It is needed to build a model that can differentiate between "Real" news and "Fake" news. This leads to consequences in social networking sites like content. Secondly, the trolls are real humans who "aim to disrupt online communities" in hopes of provoking social media users into an emotional response. Other one is, Cyborg. Cyborg users are the combination of "automated activities with human input." Humans build accounts and use programs to perform activities in social media. For false information detection, there are two categories: Linguistic Cue and Network Analysis approaches. The methods generally used to do such type of works are Naïve Bayes Classifier and Support Vector Machines (SVM).

1.5-Hardware Requirements

Processor : Ryzen 5 or intel 5 and above

RAM: 1 GB or above

Hard Disk : 10 GB HDD

Graphics Card: Nvidia GTX 1650 and above

1.6-Software Requirements

Operating system : Windows 10

Programming Language: Python and its Libraries

Technology : Machine Learning

Coding Platform: Jupyter Notebook

2.1-Introduction to Natural Language Processing

So the first question that comes in our mind is **What is NLP** ? Why is it so important and so much famous these days.

To understand it's importance, let's look at some promising examples :-

So have you ever wondered when we are using famous messaging apps like Whats App, Messenger, Hike etc, they suggest meaningful words before letting you complete the sentence. Another example would be like the SPAM or junk folder in your email, Chat Bots, Google Translation and so much more !

So yeah, these were some cool examples of NLP or Natural Language Processing.

So the term Natural Language Processing can be defined as field concerned with the ability of a computer to understand, analyze, manipulate and potentially generate human language (or close to human language). It can be any language English, Hindi, French, Spanish etc.

Real Life Examples :-

- Auto – Complete
- Auto – Correct
- Spam Detection
- Translation of one Language to Another
- Conversational Chat Bots

Areas of NLP :-

- **Sentiment Analysis :-**
Sentiment Analysis is a natural language processing technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. For example :- Movie reviews sentiment analysis, tweets analysis etc.
- **Topic Modeling :-**
A topic model is one that automatically discovers topics occurring in a collection of documents. A trained model may then be used to discern which of these topics occur in new documents. The model can also pick out which portions of a document cover which topics.
- **Text Classification :-**
Text clarification is the process of categorizing the text into a group of words. By using NLP, text classification can automatically analyze text and then assign a set of predefined tags or categories based on its context.

2.2-NLP ToolKit – NLTK

Now that we have some clue about what's going on and what is Natural Language Processing, we will continue the NLP WITH PYTHON

NLTK i.e. Natural Language Processing Tool Kit is a suite of open-source tools created to make NLP processes in Python easier to build.

In the above lesson we have seen that how NLP has revolutionized many areas of language such as sentiment analysis, part-of-speech tagging, text classification, language translation, topic modeling, language generation and many many more. So there are many in-built functions and libraries that are included inside this NLTK library.

```
!pip install nltk

import nltk

nltk.download()

# this will allow you to download all the necessary tools present in the library
```

This library let us do all the necessary preprocessing on our text data without any pain :D, some of the components of this library are :- **stemming, lematizing, tokenizing, stop-words removal** and so many more...

NLP PipeLine

Whenever we work on *raw text* data, python does not understand words, it just sees a stream of characters and for it, all the characters are same having no meaning. Any Machine Learning algorithm/programming language only understands **Numbers/Vectors** and not words so in order to make it understand, we need to perform the above NLP pipeline as shown in the image above.

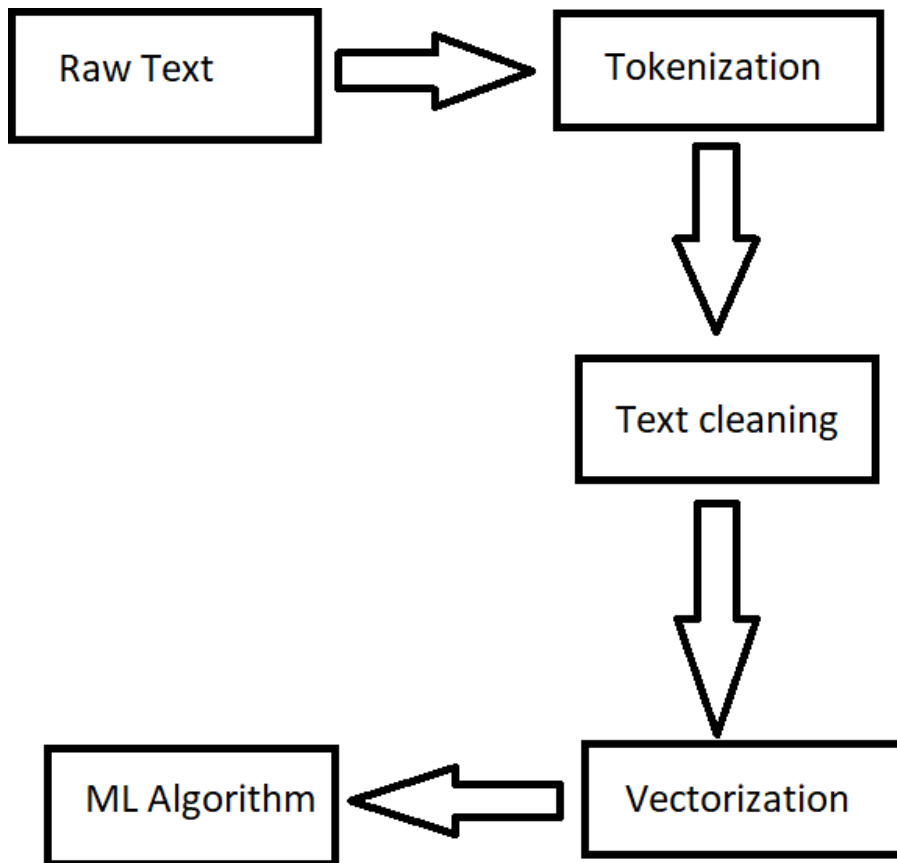


Fig 1.1

3.1-Tokenization

Tokenization is simply splitting the text/corpus into words or sentences.

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

```
sample_text = 'Hi my name is Yash'
```

```
tokenized_text = word_tokenize(sample_text)
```

```
print(tokenized_text)
```

```
output : ['Hi', 'my', 'name', 'is', 'Yash']
```


3.2-Text Cleaning

Text cleaning basically refers to the functions applied to the raw text **in** order to remove unnecessary words, punctuation, extra white spaces, **and** giving the text more meaning **in** order to be processed by our ML algorithm.

There are various Text Pre-processing/Cleaning techniques like :

- Punctuation Removal
- Stop Words Removal
- Extra white space Removal
- Emoji Removal
- Emoticons Removal
- HTML tags Removal
- URLs Removal
- Conversion to Lower case
- Numbers Removal
- Expanding Contractions **and** so many more !

Lower case conversion

importing necessary libraries

```
import os
import numpy as np
import pandas as pd
```

```
import nltk
nltk.download('punkt')
```

importing/reading raw text data

```
data = pd.read_csv("/content/SMSSpamCollection", sep="\t", header=None)
```

```
data.columns = ["category", "text"]
```

```
data.head()
```

```
category text
0 ham    Go until jurong point, crazy.. Available
1 ham    Ok lar... Joking wif u oni...
2 spam   Free entry in 2 a wkly comp to win FA Cup
3 ham    U dun say so early hor... U c already
4 ham    Nah I don't think he goes to usf, he
```

```
def convert_to_lower(text):
    return text.lower()
```

```
sample_text = 'MY NAME IS YASH'
```

```
lowered = convert_to_lower(sample_text)
```

```
print(lowered)
```

output : my name is yash

Removal of HTML tags

```
import re
```

```
punc = list(string.punctuation)
```

```
def remove_html_tags(text):
```

```
    html_pattern = r'<.*?>'
```

```
    without_html = re.sub(pattern=html_pattern, repl=' ', string=text)
```

```
    return without_html
```

```
sample_text = 'Do you know that <my name> is <yash>'
```

```
print(remove_html_tags(sample_text))
```

output : Do you know that is

Removal of Numbers

```
import re
```

```
def remove_numbers(text):
```

```
    number_pattern = r'\d+'
```

```
    without_number = re.sub(pattern=number_pattern, repl=" ", string=text)
```

```
    return without_number
```

```
sample_text = 'My number is 98274610010 and pincode is 230012'
```

```
print(remove_numbers(sample_text))
```

output : My number is and pincode is

Converting numbers to words

```
# !pip install num2words

from num2words import num2words

def convert_num_2_words(text):
    splittedText = text.split()
    for i in range(len(splittedText)):
        if splittedText[i].isdigit():
            splittedText[i] = num2words(splittedText[i])
    num_2_words = ' '.join(splittedText)
    return num_2_words

sample_text = 'My lucky number is 7'

print(convert_num_2_words(sample_text))
```

output : My lucky number is seven

Converting accented characters to ASCII characters

```
# !pip install unidecode

import unidecode

def convert_accented_2_ascii(text):
    return unidecode.unidecode(text)

example_text = "This is an example text with accented characters like dèèp lèarning ánd cömputer vísiön etc"

print(f"Original sentence: {example_text}")
print(f"Converted sentence: {convert_accented_2_ascii(example_text)}")
```

output :

Original sentence: This is an example text with accented characters like dèèp lèarning ánd cömputer vísiön etc

Converted sentence: This is an example text with accented characters like deep learning and computer vision etc

Expanding contractions

```
# !pip install contractions
```

```
import contractions
```

```
def expand_contractions(text):  
    expanded_text = []  
    for word in text.split():  
        expanded_text.append(contractions.fix(word))  
    return " ".join(expanded_text)
```

```
example_text = "Sometimes our mind doesn't work properly. I've tried everything."  
print(f"Original text: {example_text}")  
print(f"Expanded text: {expand_contractions(example_text)}")
```

output :

Original text: Sometimes our mind doesn't work properly. I've tried everything.

Expanded text: Sometimes our mind does not work properly. I have tried everything.

Stemming

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.

```
from nltk.stem import PorterStemmer
from nltk import word_tokenize
```

```
def stemming(text):
    stemmer = PorterStemmer()
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        stem_word = stemmer.stem(tokens[i])
        tokens[i] = stem_word
    return " ".join(tokens)
```

```
sample_text = 'i love gaming and I recently visited my friend's place'
```

```
print(stemming(sample_text))
```

output : i love gam and I recent visite my friend place

Lemmatizing

Lemmatization takes into consideration the morphological analysis of the words. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma.

Basic difference between stemming and lemmatizing is that in **stemming**, the removal of suffix takes place without any meaning. On the other hand, **lemmatizing** takes morphological and lexical meaning into consideration and then returns a much more meaning full 'lemma'.

```
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize

nltk.download("wordnet")

def lemmatizing(text):
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        lemma_word = lemmatizer.lemmatize(tokens[i])
        tokens[i] = lemma_word
    return " ".join(tokens)

sample_text = 'i love gaming and I recently visited my friend's place'

print(lemmatizing(sample_text))

output : i love game and I recent visit my friend place
```

Emoji removal

```
import re

def remove_emoji(text):
    emoji_pattern = re.compile("[
        u\"\\U0001F600-\\U0001F64F\" # emoticons
        u\"\\U0001F300-\\U0001F5FF\" # symbols & pictographs
        u\"\\U0001F680-\\U0001F6FF\" # transport & map symbols
        u\"\\U0001F1E0-\\U0001F1FF\" # flags (iOS)
        u\"\\U00002500-\\U00002BEF\" # chinese char
        u\"\\U00002702-\\U000027B0\"
        u\"\\U00002702-\\U000027B0\"
        u\"\\U000024C2-\\U0001F251\"
        u\"\\U0001f926-\\U0001f937\"
        u\"\\U00010000-\\U0010ffff\"
        u\"\\u2640-\\u2642\"
        u\"\\u2600-\\u2B55\"
    ]")
```

```
u"\u200d"
u"\u23cf"
u"\u23e9"
u"\u231a"
u"\ufe0f" # dingbats
u"\u3030"
"]+", flags=re.UNICODE)
removeEmoji = emoji_pattern.sub(r"", text)
return removeEmoji
```

```
example_text = "This is a test 🍌 "
print(f"Original text: {example_text}")
print(f"Removed emoji: {remove_emoji(example_text)}")
```

output :

Original text: This is a test 🍌
Removed emoji: This is a test

Punctuation Removal

```
import string

def remove_punctuation(text):
    return text.translate(str.maketrans("", "", string.punctuation))

example_text = 'i love gaming, programming, and what more ? .'

print(remove_punctuation(example_text))
```

output : i love gaming programming and what more

Stop words removal

The words which are generally filtered out before processing a natural language are called **stop words**. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”.

Why do we need to remove stop-words ?

Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information. In other words, we can say that the removal of such words does not show any negative consequences on the model we train for our task.

Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.

We do not always remove the stop words. The removal of stop words is highly dependent on the task we are performing and the goal we want to achieve.

```
from nltk.corpus import stopwords
from nltk import word_tokenize
```

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

```
def remove_stopwords(text):
    removed = []
    stop_words = list(stopwords.words("english"))
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        if tokens[i] not in stop_words:
            removed.append(tokens[i])
    return " ".join(removed)
```

```
example_text = 'The movie was not good at all.'
print(remove_stopwords(example_text))
```

output : movie good

Removal of extra white space

```
def remove_extra_white_spaces(text):  
    single_char_pattern = r'\s+[a-zA-Z]\s+'  
    without_sc = re.sub(pattern=single_char_pattern, repl=" ", string=text)  
    return without_sc
```

```
example_text = 'i love food it makes me happy'
```

```
print(remove_extra_white_spaces(example_text))
```

output : i love food it makes me happy

So these were some of the most important text cleaning/pre-processing techniques which are applied on raw text data

3.3-Vectorization

As we discussed earlier that in order to make our machine learning algorithm make sense of text data, we need to convert the characters/words/sentences into numbers/vectors.

So what exactly is **Vectorization** ?

Vectorization is the process of conversion of words into numbers/vectors.

```
sample_text = 'My name is Yash'
```

```
vectorized_text = [123, 32, 15, 107]
```

Types of Vectorization methods :-

- Count Vectorization
- N-Gram Vectorization
- TF-IDF Vectorization
- Word2Vec
- BERT (It is basically Word2Vec with Context) (state-of-the-art, Advance topic)

In this Project we use Count Vectorization and N-Gram Vectorization. Let us discuss about these two Techniques

Count Vectorization Technique :-

It is the simplest of all vectorization techniques.

- It creates a **Document Term Matrix**, now a doc-term matrix is a matrix whose **rows** are every single element our list, for example, we have **4** elements/docs in our 'sample_text', **columns** are all the unique words from our whole document/whole 'sample_text'. Each cell of our Doc-Term matrix represents the **frequency** of that word in current cell.
- Now let's code what we discussed above ! (It's really simple!).

```
cv = CountVectorizer()  
X = cv.fit_transform(sample_text)  
X = X.toarray()
```

```
print(X.shape)  
print(X)
```

Output:
(4, 20)

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

Let's also print out the unique words:-

```
cols = cv.get_feature_names()  
print(cols)
```

```
['am', 'badminton', 'favourite', 'final', 'games', 'graduation', 'in', 'is', 'kelkar', 'like', 'my', 'name', 'of',  
'outdoor', 'play', 'sport', 'to', 'video', 'yash', 'year']
```

As you can see, we had 4 documents in our 'sample_text' and the Count Vectorization technique has calculated 20 unique words from our whole documents, so therefore the shape : (4, 20). Now Each cell represents the frequency of that word, for example lets say for the cell row1 and col1 we have 0, this 0 represents that in our first doc i.e. 'My name is Yash Kelkar', 'My' word is not a unique word (we can verify this from above), so that is why it's 0. Similarly for all the docs and words the process goes same. Hence we have our Count Vectorized matrix ready to be fed into our Machine Learning model.

N-gram Vectorization Technique :-

It also creates a Document Term matrix.

Columns represent all columns of adjacent words of length 'n'.

Rows represent each document in our sample_text.

Cell represent Count.

When n = 1, it is called Uni-gram, which is basically Count Vectorizer. Example = "my", "name", "is", "yash".

When n = 2, it is called Bi-gram, Example = "my name", "is yash".

When n = 3, it is called Tri-gram, Example = "my name is", "yash".

So now let's discuss the code

```
ngram = CountVectorizer(ngram_range=(1,3)) # (1,3) means we will consider all grams i.e. uni, bi and tri.
```

```
X = ngram.fit_transform(sample_text) # returns a sparse matrix, so we need to convert it to array.
```

```
X = X.toarray()
```

```
print(X.shape)
```

```
print(X)
```

Output:

```
(4, 54)
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0]
 [1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0]
 [0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1
  1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0]]
```

As you can see, it creates a combination of 54 grams, so let's take a look at these n-grams created.

```
ngrams = ngram.get_feature_names()
```

```
print(ngrams)
```

['am', 'am in', 'am in my', 'badminton', 'favourite', 'favouriteoutdoor', 'favourite outdoor sport', 'final', 'final year', 'final year of', 'games', 'graduation', 'in', 'in my', 'in my final', 'is', 'is badminton', 'is yash', 'is yash kelkar', 'kelkar', 'like', 'like to', 'like to play', 'my', 'my favourite', 'my favourite outdoor', 'my final', 'my final year', 'my name', 'my name is', 'name', 'name is', 'name is yash', 'of', 'of graduation', 'outdoor', 'outdoor sport', 'outdoor sport is', 'play', 'play video', 'play video games', 'sport', 'sport is', 'sport is badminton', 'to', 'to play', 'to play video', 'video', 'video games', 'yash', 'yash kelkar', 'year', 'year of', 'year of graduation']

So, this was our n-grams and 'X' our n-gram doc-term matrix which is ready to be fed in our Machine Learning Model

3.4-Machine Learning

After we have successfully converted our raw text data into vectors, we are now ready to feed this vectorized data and its corresponding label like Fake or Real in to our machine learning algorithm.

There are many Machine Learning algorithms out there like Decision Trees, Random Forest Classifier, KNN, Logistic Regression etc, but the algorithms which works better on textual data are **Naive Bayes Algorithm** which works on the concept of Conditional Probability

After training with any of the mentioned Machine Learning/Deep Learning algorithms, We can now say that we have successfully trained our Fake News Classifier model which can detect whether a message/email is Fake or not!

This is our NLP pipeline.

```
from sklearn.feature_extraction.text import CountVectorizer
```

In This project I will be using n-gram Vectorization technique because it gave me around 90% accuracy

Now the interesting part – **Model Building**

I will be using **Naive Bayes Algorithm** as it considered to be good when dealing with text data.

4-Libraries

4.1-Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Why is NumPy Fast?

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just “behind the scenes” in optimized, pre-compiled C code. Vectorized code has many advantages, among which are:

- vectorized code is more concise and easier to read
- fewer lines of code generally means fewer bugs
- the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)
- vectorization results in more “Pythonic” code. Without vectorization, our code would be littered with inefficient and difficult to read `for` loops.

Broadcasting is the term used to describe the implicit element-by-element behavior of operations; generally speaking, in NumPy all operations, not just arithmetic operations, but logical, bit-wise, functional, etc., behave in this implicit element-by-element fashion, i.e., they broadcast. Moreover, in the example above, `a` and `b` could be multidimensional arrays of the same shape, or a scalar and an array, or even two arrays of with different shapes, provided that the smaller array is “expandable” to the shape of the larger in such a way that the resulting broadcast is unambiguous. For detailed “rules” of broadcasting see *basics.broadcasting*.

Who Else Uses NumPy?

NumPy fully supports an object-oriented approach, starting, once again, with *ndarray*. For example, *ndarray* is a class, possessing numerous methods and attributes. Many of its methods are mirrored by functions in the outer-most NumPy namespace, allowing the programmer to code in whichever paradigm they prefer. This flexibility has allowed the NumPy array dialect and NumPy *ndarray* class to become the *de-facto* language of multi-dimensional data interchange used in Python.

Creating NumPy Arrays, Loading and Saving Files

NumPy works with Python objects called multi-dimensional **arrays**. Arrays are basically collections of values, and they have one or more dimensions. NumPy array data structure is also called *ndarray*, short for n-dimensional array. An array with one dimension is called a **vector** and an array with two dimensions is called a **matrix**. Datasets are usually built as matrices and it is much easier to open those with NumPy instead of working with list of lists, for example.

Turning a list to a NumPy array is pretty simple:

```
numpy_array = np.array(list)
```

And printing/displaying the array will look like this:

```
array([[ 7.4 ,  0.7 ,  0. , ...,  0.56 ,  9.4 ,  5. ],
       [ 7.8 ,  0.88 ,  0. , ...,  0.68 ,  9.8 ,  5. ],
       [ 7.8 ,  0.76 ,  0.04 , ...,  0.65 ,  9.8 ,  5. ],
       ...,
       [ 6.3 ,  0.51 ,  0.13 , ...,  0.75 , 11. ,  6. ],
       [ 5.9 ,  0.645,  0.12 , ...,  0.71 , 10.2 ,  5. ],
       [ 6. ,  0.31 ,  0.47 , ...,  0.66 , 11. ,  6. ]])
```

Another option is to open a CSV file using the [np.genfromtxt\(\)](#) function:

```
numpy_array = np.genfromtxt("file.csv", delimiter=";", skip_header=1)
```

The argument inside the brackets are the file name (and path, if needed), the delimiter set to ';' to make sure it's parsed correctly — you can use different characters to parse (like ',' for example); and skip_header set to '1' will make the csv load to an array without the header row. You can just not include it if you do want the headers (as the default is zero).

You can also save NumPy arrays to files by using np.savetxt(). For

example, np.savetxt('file.txt', arr, delimiter=' ') will save to a text file

and np.savetxt('file.csv', arr, delimiter=',') will save to a CSV file.

Another cool feature is the ability to create different arrays like random

arrays: `np.random.rand(3, 4)` will create a 3x4 array of random numbers between 0 and 1 while `np.random.rand(7, 6) * 100` will create a 7x6 array of random numbers between 0 to 100; you can also define the size of the array in a different way: `np.random.randint(10, size=(3, 2))` creates an array the size of 3x2 with random numbers between 0 and 9. Remember that the last digit (10) is not included in the range when you use this syntax.

It's also possible to create an array of all zeros: `np.zeros(4, 3)` (4x3 array of all zeros) or ones `np.ones((4))` (4x1 array of ones); you can the command `np.full((3, 2), 8)` to create a 3x2 array full of 8. You can, of course, change each and every one of these numbers to get the array you want.

Working and Inspecting Arrays

Now that you have your array loaded, you can check its size (number of elements) by typing `array.size` and its shape (the dimensions — rows and columns) by typing `array.shape`. You can use `array.dtype` to get the data types of the array (floats, integers etc — see more in the [NumPy documentation](#)) and if you need to convert the datatype you can use the `array.astype(dtype)` command. If you need to convert a NumPy array to a Python list, there is a command for that too: `array.tolist()`.

Indexing and Slicing

Indexing and slicing NumPy arrays works very similarly to working with Python lists: `array[5]` will return the element in the 5th index, and `array[2, 5]` will return the element in index[2][5]. You can also select the first five elements, for example, by using a colon (:). `array[0:5]` will return the first five elements (index 0–4) and `array[0:5, 4]` will return the first five elements in column 4. You can use `array[:2]` to get elements from the beginning until index 2 (not including index 2) or `array[2:]` to return from the 2nd index until the end of the array. `array[:, 1]` will return the elements at index 1 on all rows.

Assigning values to a NumPy array is, again, very similar to doing so in Python lists: `array[1]=4` will assign the value 4 to the element on index 1; you can do it to multiple values: `array[1,5]=10` or use slicing when assigning values: `array[:,10]=10` will change the entire 11th column to the value 10.

Sorting and Reshaping

`array.sort()` can be used to sort your NumPy array — you can pass different arguments inside the brackets to define what you want to sort on (by using the argument ‘order=string/list of strings’, for example. See more examples in the [documentation](#)). `array.sort(axis=0)` will sort specific axis of the array — rows or columns. `two_d_arr.flatten()` will flatten a 2 dimensional array to a 1 dimensional array. `array.T` will transpose an array — meaning columns will become rows and vice versa. `array.reshape(x,y)` would reshape your array to the size you set with x and y. `array.resize((x,y))` will change the array shape to x and y and fill new values with zeros.

Combining and Splitting

You can use `np.concatenate((array1,array2),axis=0)` to combine two NumPy arrays — this will add array 2 as rows to the end of array 1 while `np.concatenate((array1,array2),axis=1)` will add array 2 as columns to the end of array 1. `np.split(array,2)` will split the array into two sub-arrays and `np.hsplit(array,5)` will split the array horizontally on the 5th index.

Adding and Removing Elements

There are, of course, commands to add and remove elements from NumPy arrays:

- `np.append(array,values)` will append values to end of array.
- `np.insert(array, 3, values)` will insert values into array before index 3
- `np.delete(array, 4, axis=0)` will delete row on index 4 of array
- `np.delete(array, 5, axis=1)` will delete column on index 5 of array

Descriptive Statistics

You can use NumPy methods to get descriptive statistics on NumPy arrays:

- `np.mean(array, axis=0)` will return mean along specific axis (0 or 1)
- `array.sum()` will return the sum of the array
- `array.min()` will return the minimum value of the array
- `array.max(axis=0)` will return the maximum value of specific axis
- `np.var(array)` will return the variance of the array
- `np.std(array, axis=1)` will return the standard deviation of specific axis
- `array.corrcoef()` will return the correlation coefficient of the array
- `numpy.median(array)` will return the median of the array elements

Doing Math with NumPy

Any tutorial to NumPy would not be complete without the numerical and mathematical operations you can do with NumPy! Let's go over them:

`np.add(array, 1)` will add 1 to each element in the array and `np.add(array1, array2)` will add array 2 to array 1. The same is true to `np.subtract()`, `np.multiply()`, `np.divide()` and `np.power()` — all these commands would work in exactly the same way as described above.

You can also get NumPy to return different values from the array, like:

- `np.sqrt(array)` will return the square root of each element in the array
- `np.sin(array)` will return the sine of each element in the array

Fake News Detection

- `np.log(array)` will return the natural log of each element in the array
- `np.abs(arr)` will return the absolute value of each element in the array
- `np.array_equal(arr1, arr2)` will return `True` if the arrays have the same elements and shape

It is possible to round different values in array: `np.ceil(array)` will round up to the nearest integer, `np.floor(array)` will round down to the nearest integer and `np.round(array)` will round to the nearest integer.

4.2-Pandas

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

All properties and methods of the DataFrame object, with explanations and examples:

Property/Method	Description
abs()	Return a DataFrame with the absolute value of each value
add()	Adds the values of a DataFrame with the specified value(s)
add_prefix()	Prefix all labels
add_suffix()	Suffix all labels
agg()	Apply a function or a function name to one of

the axis of the DataFrame

[`aggregate\(\)`](#)

Apply a function or a function name to one of
the axis of the DataFrame

`align()`

Aligns two DataFrames with a specified join method

[`all\(\)`](#)

Return True if all values in the DataFrame are True
, otherwise False

[`any\(\)`](#)

Returns True if any of the values in the DataFrame
are True, otherwise False

[`append\(\)`](#)

Append new columns

[`applymap\(\)`](#)

Execute a function for each element in the DataFrame

[`apply\(\)`](#)

Apply a function to one of the axis of the DataFrame

[`assign\(\)`](#)

Assign new columns

[`astype\(\)`](#)

Convert the DataFrame into a specified dtype

[`at`](#)

Get or set the value of the item with the specified label

[axes](#)

Returns the labels of the rows and the columns of the DataFrame

[bfill\(\)](#)

Replaces NULL values with the value from the next row

[bool\(\)](#)

Returns the Boolean value of the DataFrame

[columns](#)

Returns the column labels of the DataFrame

[combine\(\)](#)

Compare the values in two DataFrames, and let a function decide which values to keep

[combine_first\(\)](#)

Compare two DataFrames, and if the first DataFrame has a NULL value, it will be filled with the respective value from the second DataFrame

`compare()`

Compare two DataFrames and return the differences

[convert_dtypes\(\)](#)

Converts the columns in the DataFrame into new dtypes

[corr\(\)](#)

Find the correlation (relationship) between each column

[count\(\)](#)

Returns the number of not empty cells for each column/row

[`cov\(\)`](#)

Find the covariance of the columns

[`copy\(\)`](#)

Returns a copy of the DataFrame

[`cummax\(\)`](#)

Calculate the cumulative maximum values of the DataFrame

[`cummin\(\)`](#)

Calculate the cumulative minimum values of the DataFrame

[`cumprod\(\)`](#)

Calculate the cumulative product over the DataFrame

[`cumsum\(\)`](#)

Calculate the cumulative sum over the DataFrame

[`describe\(\)`](#)

Returns a description summary for each
column in the DataFrame

[`diff\(\)`](#)

Calculate the difference between a value and the value
of the same column in the
previous row

[`div\(\)`](#)

Divides the values of a DataFrame with the specified
value(s)

[`dot\(\)`](#)

Multiplies the values of a DataFrame with values from
another array-like object,
and add the result

[drop\(\)](#)

Drops the specified rows/columns from the DataFrame

[drop_duplicates\(\)](#)

Drops duplicate values from the DataFrame

[droplevel\(\)](#)

Drops the specified index/column(s)

[dropna\(\)](#)

Drops all rows that contains NULL values

[dtypes](#)

Returns the dtypes of the columns of the DataFrame

[duplicated\(\)](#)

Returns True for duplicated rows, otherwise False

[empty](#)

Returns True if the DataFrame is empty, otherwise False

[eq\(\)](#)

Returns True for values that are equal to the specified value(s), otherwise False

[equals\(\)](#)

Returns True if two DataFrames are equal, otherwise False

[eval](#)

Evaluate a specified string

[explode\(\)](#)

Converts each element into a row

[ffill\(\)](#)

Replaces NULL values with the value from the previous row

[fillna\(\)](#)

Replaces NULL values with the specified value

[filter\(\)](#)

Filter the DataFrame according to the specified filter

[first\(\)](#)

Returns the first rows of a specified date selection

[floordiv\(\)](#)

Divides the values of a DataFrame with the specified value(s), and floor the values

[ge\(\)](#)

Returns True for values greater than, or equal to the specified value(s), otherwise False

[get\(\)](#)

Returns the item of the specified key

[groupby\(\)](#)

Groups the rows/columns into specified groups

[gt\(\)](#)

Returns True for values greater than the specified value(s), otherwise False

[head\(\)](#)

Returns the header row and the first 10 rows, or the specified number of rows

[iat](#)

Get or set the value of the item in the specified position

[idxmax\(\)](#)

Returns the label of the max value in the specified axis

[idxmin\(\)](#)

Returns the label of the min value in the specified axis

[iloc](#)

Get or set the values of a group of elements in the specified positions

[index](#)

Returns the row labels of the DataFrame

[infer_objects\(\)](#)

Change the dtype of the columns in the DataFrame

[info\(\)](#)

Prints information about the DataFrame

[insert\(\)](#)

Insert a column in the DataFrame

[interpolate\(\)](#)

Replaces not-a-number values with the interpolated method

[isin\(\)](#)

Returns True if each elements in the DataFrame is in the

	specified value
<u>isna()</u>	Finds not-a-number values
<u>isnull()</u>	Finds NULL values
<u>items()</u>	Iterate over the columns of the DataFrame
<u>iteritems()</u>	Iterate over the columns of the DataFrame
<u>iterrows()</u>	Iterate over the rows of the DataFrame
<u>itertuples()</u>	Iterate over the rows as named tuples
<u>join()</u>	Join columns of another DataFrame
<u>last()</u>	Returns the last rows of a specified date selection
<u>le()</u>	Returns True for values less than, or equal to the specified value(s), otherwise False
<u>loc</u>	Get or set the value of a group of elements specified using their labels

[lt\(\)](#)

Returns True for values less than the specified value(s), otherwise False

[keys\(\)](#)

Returns the keys of the info axis

[kurtosis\(\)](#)

Returns the kurtosis of the values in the specified axis

[mask\(\)](#)

Replace all values where the specified condition is True

[max\(\)](#)

Return the max of the values in the specified axis

[mean\(\)](#)

Return the mean of the values in the specified axis

[median\(\)](#)

Return the median of the values in the specified axis

[melt\(\)](#)

Reshape the DataFrame from a wide table to a long table

[memory_usage\(\)](#)

Returns the memory usage of each column

[merge\(\)](#)

Merge DataFrame objects

[min\(\)](#)

Returns the min of the values in the specified axis

[mod\(\)](#)

Modules (find the remainder) of the
values of a DataFrame

[mode\(\)](#)

Returns the mode of the values in the specified axis

[mul\(\)](#)

Multiplies the values of a DataFrame with
the specified
value(s)

[ndim](#)

Returns the number of dimensions of the DataFrame

4.3-Matplotlib

What is Matplotlib?

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

4.4-Seaborn

Seaborn Vs Matplotlib

It is summarized that if Matplotlib “tries to make easy things easy and hard things possible”, Seaborn tries to make a well-defined set of hard things easy too.”

Seaborn helps resolve the two major problems faced by Matplotlib; the problems are –

- Default Matplotlib parameters
- Working with data frames

As Seaborn compliments and extends Matplotlib, the learning curve is quite gradual. If you know Matplotlib, you are already half way through Seaborn.

Important Features of Seaborn

Seaborn is built on top of Python’s core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in –

- Built in themes for styling matplotlib graphics
- Visualizing univariate and bivariate data
- Fitting in and visualizing linear regression models
- Plotting statistical time series data
- Seaborn works well with NumPy and Pandas data structures
- It comes with built in themes for styling Matplotlib graphics

4.5-Scikit-Learn

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

In this tutorial we will learn to code python and apply Machine Learning with the help of the scikit-learn library, which was created to make doing machine learning in Python easier and more robust.

To do this, we'll be using the [Sales Win Loss data](#) set from IBM's Watson repository. We will import the data set using pandas, explore the data using pandas methods like `head()`, `tail()`, `dtypes()`, and then try our hand at using plotting techniques from Seaborn to visualize our data.

Then we'll dive into scikit-learn and use `preprocessing.LabelEncoder()` in scikit-learn to process the data, and `train_test_split()` to split the data set into test and train samples. We will also use a cheat sheet to help us decide which algorithms to use for the data set. Finally we will use three different algorithms (Naive-Bayes, LinearSVC, K-Neighbors Classifier) to make predictions and compare their performance using methods like `accuracy_score()` provided by the scikit-learn library. We will also visualize the performance score of different models using scikit-learn and Yellowbrick visualization.

- pandas fundamentals
- Seaborn and matplotlib basics

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

- **Supervised learning algorithms:** Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.
- **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.
- **Unsupervised learning algorithms:** Again there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- **Various toy datasets:** This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- **Feature extraction:** Scikit-learn for extracting features from images and text (e.g. Bag of words)

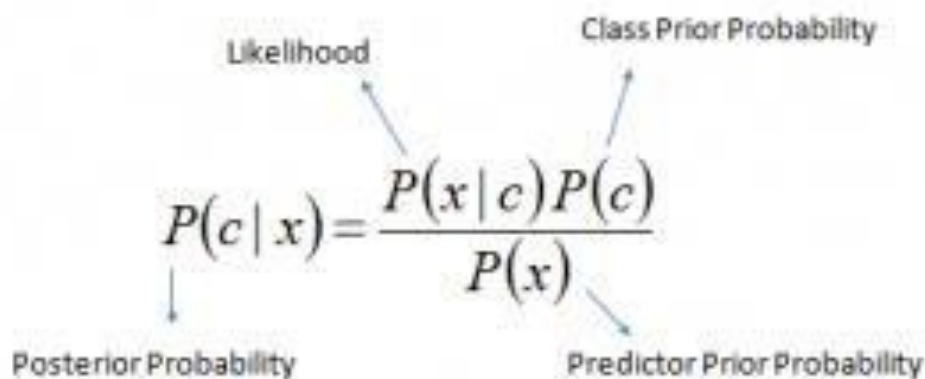
5.1-Naive Bayes Algorithm

It is a [classification technique](#) based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with arrows pointing from labels to the corresponding parts of the equation. 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

5.2-Passive Aggressive Algorithm

The Passive-Aggressive algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications.

Note: This is a high-level overview of the algorithm explaining how it works and when to use it. It does not go deep into the mathematics of how it works.

Passive-Aggressive algorithms are generally used for large-scale learning. It is one of the few '**online-learning algorithms**'. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated step-by-step, as opposed to batch learning, where the entire training dataset is used at once. This is very useful in situations where there is a huge amount of data and it is computationally infeasible to train the entire dataset because of the sheer size of the data. We can simply say that an online-learning algorithm will get a training example, update the classifier, and then throw away the example.

A very good example of this would be to detect fake news on a social media website like Twitter, where new data is being added every second. To dynamically read data from Twitter continuously, the data would be huge, and using an online-learning algorithm would be ideal.

Passive-Aggressive algorithms are somewhat similar to a Perceptron model, in the sense that they do not require a learning rate. However, they do include a regularization parameter.

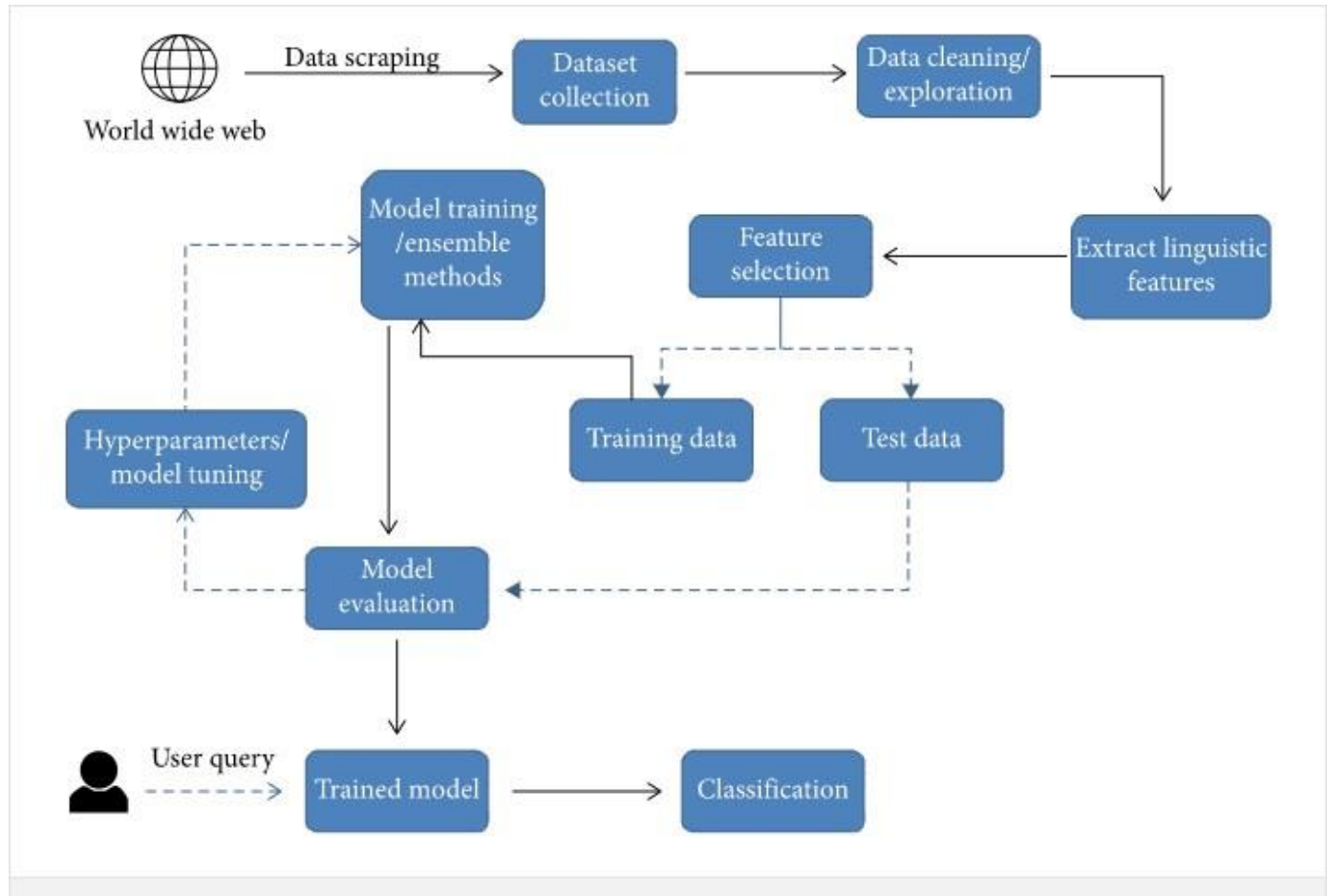
How Passive-Aggressive Algorithms Work:

Passive-Aggressive algorithms are called so because :

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.

Data Flow Diagram



In This Project we collect data set From Kaggle website

Data Cleaning

Data cleaning is the process of preparing raw **data** for analysis by removing bad **data**, organizing the raw **data**, and filling in the null values. Ultimately, **cleaning data** prepares the **data** for the process of **Machine learning** when the most valuable information can be pulled from the **data** set.

Exploration

-Data exploration definition:

Data exploration refers to the initial step in data analysis in which data we use [data visualization](#) and statistical techniques to describe dataset characterizations, such as size, quantity, and accuracy, in order to better understand the nature of the data.

Data exploration techniques include both manual analysis and automated data exploration software solutions that visually explore and identify relationships between different data variables, the structure of the dataset, the presence of outliers, and the distribution of data values in order to reveal patterns and points of interest, enabling data analysts to gain greater insight into the raw data.

After Exploration We extract important features this is called feature selection and split the data set in to training data and testing data

Model Training

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

Model Evaluation

Model evaluation aims to estimate the generalization accuracy of a **model** on future (unseen/out-of-sample) data.

Model Tuning

Model tuning helps to increase the accuracy of a machine learning **model**.

Explanation: **Tuning** can be defined as the process of improvising the performance of the **model** without creating any hype or creating over fitting of a variance.

After Model training our Fake news model detects correctly fake news and real news.

7.1-Pseudo Code

```
corpus=[]

sentences=[]

for i in range(0,len(messages)):

    review=re.sub('[^a-zA-Z]', '', messages['title'][i])

    review=review.lower()

    list=review.split()

    review=[ps.stem(word) for word in list if word not in set(stopwords.words('english'))]

    sentences=' '.join(review)

    corpus.append(sentences)


from sklearn.feature_extraction.text import CountVectorizer

cv=CountVectorizer(max_features=5000,ngram_range=(1,3))

X=cv.fit_transform(corpus).toarray()

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)

from sklearn.naive_bayes import MultinomialNB

classifier=MultinomialNB()

classifier.fit(X_train,y_train)

pred=classifier.predict(X_test)

from sklearn import metrics

metrics.accuracy_score(y_test,pred)
```

7.2-Outputs

```
In [65]: ### Most 20 real values  
sorted(zip(classifier.coef_[0],feature_names),reverse=True)[0:20]
```

```
Out[65]: [(-3.9648951809317863, 'trump'),  
          (-4.272721819476034, 'hillari'),  
          (-4.368759007672977, 'clinton'),  
          (-4.861090048802803, 'elect'),  
          (-5.219261999009128, 'new'),  
          (-5.230561554263062, 'comment'),  
          (-5.269176390390841, 'video'),  
          (-5.355472203843678, 'war'),  
          (-5.372788653855138, 'hillari clinton'),  
          (-5.394864605554338, 'us'),  
          (-5.412883111057016, 'fbi'),  
          (-5.483500678270969, 'vote'),  
          (-5.483500678270969, 'email'),  
          (-5.559486585248892, 'obama'),  
          (-5.570068694579429, 'world'),  
          (-5.718914322176994, 'donald'),  
          (-5.743915624382411, 'donald trump'),  
          (-5.8229040357010415, 'russia'),  
          (-5.864868234800074, 'presid'),  
          (-5.872036724278686, 'america')]
```



```
In [66]: ###Most 20 fake words  
sorted(zip(classifier.coef_[0],feature_names))[0:20]
```

```
Out[66]: [(-10.806510657409378, 'abroad'),  
          (-10.806510657409378, 'abus new'),  
          (-10.806510657409378, 'abus new york'),  
          (-10.806510657409378, 'accid'),  
          (-10.806510657409378, 'act new'),  
          (-10.806510657409378, 'act new york'),  
          (-10.806510657409378, 'adopt'),  
          (-10.806510657409378, 'advic'),  
          (-10.806510657409378, 'advis new'),  
          (-10.806510657409378, 'advis new york'),  
          (-10.806510657409378, 'age new'),  
          (-10.806510657409378, 'age new york'),  
          (-10.806510657409378, 'agenda breitbart'),  
          (-10.806510657409378, 'aleppo new'),  
          (-10.806510657409378, 'aleppo new york'),  
          (-10.806510657409378, 'ali'),  
          (-10.806510657409378, 'america breitbart'),  
          (-10.806510657409378, 'america new york'),  
          (-10.806510657409378, 'american breitbart'),  
          (-10.806510657409378, 'american new')]
```

Future Enhancement

In this Project We use Machine learning algorithms . In future Enhancement We use deep learning algorithms,Advanced NLP Techniques for better accuracy.

Conclusion

Fake Words are having higher negative coefficient it means any sentence or text contain that particular word may have higher chances of being faked....

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques. The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news. The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models. The learning models were trained and parameter-tuned to obtain optimal accuracy. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners.

Fake news detection has many open issues that require attention of researchers. For instance, in order to reduce the spread of fake news, identifying key elements involved in the spread of news is an important step. Graph theory and machine learning techniques can be employed to identify the key sources involved in spread of fake news. Likewise, real time fake news identification in videos can be another possible future direction.