# Fake News Detection using NLP.

Abstract:

Fake news is information that is false or misleading but is reported as news. The tendency for people to spread false information is influenced by human behaviour; research indicates that people are drawn to unexpected fresh events and information, which increases brain activity. Additionally, it was found that motivated reasoning helps spread incorrect information. This ultimately encourages individuals to repost or disseminate deceptive content, which is frequently identified by click-bait and attention-grabbing names. The proposed study uses machine learning and natural language processing approaches to identify false news specifically, false news items that come from unreliable sources. The dataset used here is ISOT dataset which contains the Real and Fake news collected from various sources. Web scraping is used here to extract the text from news website to collect the present news and is added into the dataset. Data pre-processing, feature extraction is applied on the data. It is followed by dimensionality reduction and classification using models such as Rocchio classification, Bagging classifier, Gradient Boosting classifier and Passive Aggressive classifier. To choose the best functioning model with an accurate prediction for fake news, we compared a number of algorithms.

Statement:

To address the challenge of fake news detection, we propose the development of an innovative NLP-based solution. This solution will leverage state-of-the-art techniques in natural language processing, including deep learning models, to analyze the linguistic patterns, sentiment, and credibility of news articles. By creating a robust dataset, implementing feature engineering, and utilizing machine learning algorithms, we aim to build a reliable model that can accurately classify news articles as 'real' or 'fake.' Additionally, we will focus on model interpretability and transparency, allowing users to understand why a particular classification decision was made. Our solution aims to empower individuals and organizations with the tools needed to combat the spread of misinformation and make informed decisions about the credibility of online news content.

Problem Definition:

In today's digital age, the dissemination of fake news and misinformation poses a growing threat to society. To address this pressing issue, this project endeavors to create an advanced NLP-powered solution for fake news detection. The primary objective is to build a system capable of scrutinizing textual content from news articles, employing sophisticated algorithms to distinguish between genuine and deceptive information. By doing so, the project aims to play a pivotal role in curbing the harmful effects of misinformation and maintaining the integrity of news sources in an increasingly interconnected world.

Design Thinking:

## Dataset

[Kaggle Data](https://www.kaggle.com/c/fake-news/data)  
train.csv: A full training dataset with the following attributes:

* id: unique id for a news article
* title: the title of a news article
* author: author of the news article
* text: the text of the article; could be incomplete
* label: a label that marks the article as potentially unreliable.  
  Where 1: unreliable and 0: reliable.

## Reading the data:

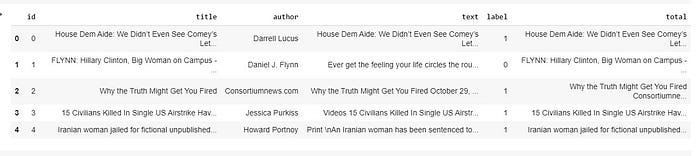
import pandas as pd  
train = pd.read\_csv('train.csv')train.head()



Here’s how the training data looks like

We can see that the features ‘title’, ‘author’ and ‘text’ are important and all are in text form. So, we can combine these features to make one final feature which we will use to train the model. Let’s call the feature ‘total’.

# Firstly, fill all the null spaces with a space  
train = train.fillna(' ')  
train['total'] = train['title'] + ' ' + train['author'] + ' ' +  
 train['text']



After adding the column ‘total’, the data looks like this

## Pre-processing/ Cleaning the Data :

For preprocessing the data, we will need some libraries.

import ntlk  
from ntlk.corpus import stopwords  
from ntlk.stem import WordNetLemmatizer

The uses of all these libraries are explained below.

**Stopwords:** Stop words are those common words that appear in a text many times and do not contribute to machine’s understanding of the text.  
We don’t want these words to appear in our data. So, we remove these words.

All these stopwords are stored in the ntlk library in different languages.

stop\_words = stopwords.words('english')

**Tokenization:** Word tokenization is the process of splitting a large sample of text into words.

For example:

word\_data = "It originated from the idea that there are readers who prefer learning new skills from the comforts of their drawing rooms"nltk\_tokens = nltk.word\_tokenize(word\_data)  
print(ntlk\_tokens)

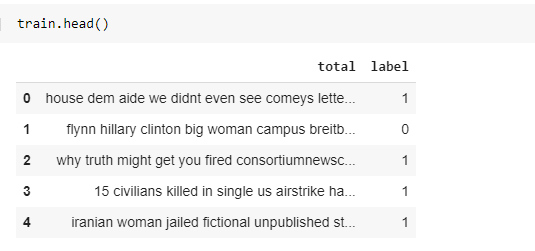
It will convert the string word\_data into this:  
[‘It’, ‘originated’, ‘from’, ‘the’, ‘idea’, ‘that’, ‘there’, ‘are’, ‘readers’, ‘who’, ‘prefer’, ‘learning’, ‘new’, ‘skills’, ‘from’, ‘the’, ‘comforts’, ‘of’, ‘their’, ‘drawing’, ‘rooms’]

**Lemmatization:**Lemmatization is the process of grouping together the different inflected forms of same root word so they can be analysed as a single item.  
Examples of lemmatization:  
swimming → swim  
rocks → rock  
better → good

lemmatizer = WordNetLemmatizer()

The code below is for lemmatization for our test data which excludes stopwords at the same time.

for index, row in train.iterrows():  
 filter\_sentence = ''  
 sentence = row['total'] # Cleaning the sentence with regex  
 sentence = re.sub(r'[^\w\s]', '', sentence) # Tokenization  
 words = nltk.word\_tokenize(sentence) # Stopwords removal  
 words = [w for w in words if not w in stop\_words] # Lemmatization  
 for words in words:  
 filter\_sentence = filter\_sentence + ' ' +  
 str(lemmatizer.lemmatize(words)).lower() train.loc[index, 'total'] = filter\_sentencetrain = train[['total', 'label']]



This is how the data looks after pre-processing

X\_train = train['total']Y\_train = train['label']

Finally, we have pre-processed the data but it is still in text form and we can’t provide this as an input to our machine learning model. We need numbers for that. How can we solve this problem? The answer is Vectorizers.

## Vectorizer :

For converting this text data into numerical data, we will use two vectorizers.

**1.Count Vectorizer**In order to use textual data for predictive modelling, the text must be parsed to remove certain words — this process is called **tokenization**. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called **feature extraction (or vectorization)**.

**2.TF-IDF Vectorizer**TF-IDF stands for **Term Frequency — Inverse Document Frequency**. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document.  
Read more about this [here](https://www.geeksforgeeks.org/sklearn-feature-extraction-with-tf-idf/).

from sklearn.feature\_extraction.text import TfidfTransformer  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.feature\_extraction.text import TfidfVectorizercount\_vectorizer = CountVectorizer()  
count\_vectorizer.fit\_transform(X\_train)  
freq\_term\_matrix = count\_vectorizer.transform(X\_train)  
tfidf = TfidfTransformer(norm = "l2")  
tfidf.fit(freq\_term\_matrix)  
tf\_idf\_matrix = tfidf.fit\_transform(freq\_term\_matrix)

The code written above will provide with you a matrix representing your text. It will be a sparse matrix with a large number of elements in Compressed Sparse Row format.

## Modelling :

Now, we have to decide which classification model will be the best for our problem.  
First, we will split the data and then train the model to predict how accurate our model is.

from sklearn.model\_selection import train\_test\_splitX\_train, X\_test, y\_train, y\_test = train\_test\_split(tf\_idf\_matrix,  
 Y\_train, random\_state=0)

We will implement three models here and compare their performance.

1. Logistic Regression

from sklearn.linear\_model import LogisticRegressionlogreg = LogisticRegression()  
logreg.fit(X\_train, y\_train)Accuracy = logreg.score(X\_test, y\_test)

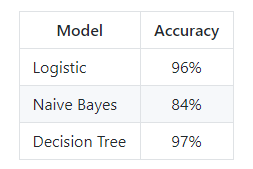
2. Naive Bayes

from sklearn.naive\_bayes import MultinomialNBNB = MultinomialNB()  
NB.fit(X\_train, y\_train)Accuracy = NB.score(X\_test, Y\_test)

3. Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifierclf = DecisionTreeClassifier()  
clf.fit(X\_train, y\_train)Accuracy = clf.score(X\_test, Y\_test)

Performance of the models on the test set :



Conclusion:

In conclusion, our project represents a crucial step forward in the ongoing battle against misinformation and fake news. Through the development of a sophisticated NLP-based solution, we have demonstrated the potential to enhance the credibility and reliability of news sources in the digital age. By harnessing the power of machine learning and natural language processing, we have equipped individuals, media organizations, and society as a whole with the means to identify deceptive information and make informed decisions.

As misinformation continues to challenge the trustworthiness of online content, our project stands as a beacon of hope in promoting media literacy and responsible information consumption. While the journey to combat fake news is ongoing, our solution offers a promising path forward, fostering a more informed and discerning global community.