**FAKE NEWS DETECTION USING NLP**

# **BATCH MEMBER**

**6220221243012 – DHARSHANAA.S**

**Phase 3 Submission Document**

**Project Name : Fake News Detection Using NLP**

**Phase 3 : Development part 1**

**Topic :** In this part you will begin building your project by loading and preprocessing the dataset. Begin building the fake news detection model by loading and preprocessing the dataset. Load the fake news dataset and preprocess the textual data.



## **FAKE NEWS DETECTION USING NLP**

**Introduction :**

Fake news detection is the process of identifying and verifying the accuracy of news or information that is intentionally false, misleading, or fabricated. It has become a critical concern in today’s digital age, where misinformation can spread rapidly through various media channels. Here’s an introduction to the topic.

**Definition of Fake News:** Fake news encompasses various types of misinformation, including fabricated stories, manipulated images or videos, and misleading headlines. It can be spread through websites, social media, or traditional media outlets.

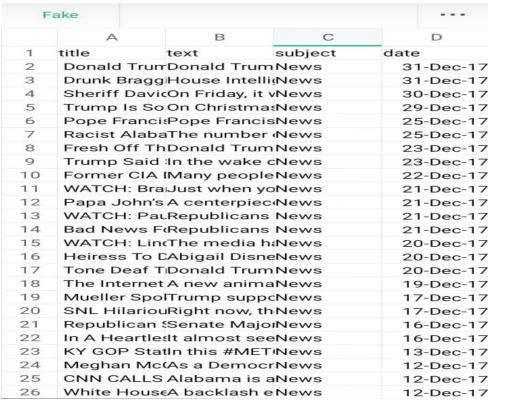
**Motivations for Fake News:** Fake news can be created for various reasons, such as political manipulation, financial gain, or simply for entertainment. It often seeks to exploit emotions, biases, or controversy to gain attention and traction.

**Impact of Fake News:** Fake news can have serious consequences, including influencing public opinion, swaying elections, causing panic, or harming individuals’ reputations. It can erode trust in journalism and democratic processes.

**Challenges in Fake News Detection**: Detecting fake news is a complex task due to its constantly evolving nature. Some challenges include the speed at which fake news spreads, the use of sophisticated techniques to make it appear legitimate, and the fine line between satire and actual misinformation.

**Given data set :**

**Dataset Link:**[**https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**](https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset)

****

23503 Rows X 5 columns ( False Dat )

21418 Rows X 5 columns ( True Data)

**Word Cloud to plot the most frequent words**

So, next, we will plot the most frequent words in fake news and real news using the word cloud. **Word cloud** is a technique for visualizing most frequent words in a text corpus where the size of the words represents their frequency. For plotting [word cloud](https://pypi.org/project/wordcloud/) we have used word cloud python library.



**Text pre-processing**

After analyzing the data, we move towards text pre-processing before building machine learning models. The text pre-processing consists of the following steps:

Step 1: Lower casing

Step 2: Stop word removal

Step 3: Special character removal

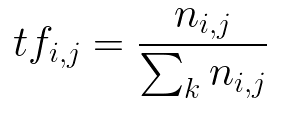
**Train Test Split**

In this step, we split the data into train and test set in the ratio of 75:25 i.e., 75% of the data used in training the model and rest 25% used for testing the model. The code for splitting data is shown below.

|  |
| --- |
| 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=0) |

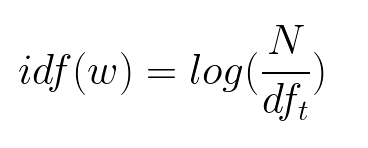
**Term Frequency (TF)**

Term Frequency represents the number of times a word appears in a document divided by the total number of words in the document. The formulae of Term frequency is mathematically shown as below.



**Inverse Document Frequency (IDF)**

It represents the log of the number of documents divided by the number of documents containing the word w. Inverse data frequency used to weight the rare words across all documents in the corpus.



**Necessary steps to follow :**

* 1. Importing Libraries and Datasets
  2. Data Preprocessing
  3. Preprocessing and analysis of News column
  4. Converting text into Vectors
  5. Model training, Evaluation, and Prediction

**1.Importing Libraries and Datasets**

The libraries used are :

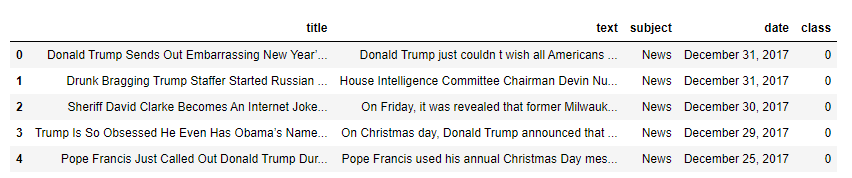
* + [Pandas](https://www.geeksforgeeks.org/python-pandas-dataframe/): For importing the dataset.
  + [Seaborn](https://www.geeksforgeeks.org/introduction-to-seaborn-python/)/[Matplotlib](https://www.geeksforgeeks.org/python-introduction-matplotlib/): For data visualization.

|  |
| --- |
| **import** pandas as pd  **import** seaborn as sns  **import** matplotlib.pyplot as plt |

Let’s import the downloaded dataset.

|  |
| --- |
| data = pd.read\_csv('News.csv',index\_col=0)  data.head() |

**OUTPUT:**



**2.Data preprocessing**

The shape of the dataset can be found by the below code

|  |
| --- |
| data.shape |

OUTPUT:

(44919, 5)

As the title, subject and date column will not going to be helpful in identification of the news. So, we can drop these column.

|  |
| --- |
| data **=** data.drop(["title", "subject","date"], axis **=** 1) |

Now, we have to check if there is any null value (we will drop those rows)

|  |
| --- |
| data.isnull().sum() |

Output:

text 0

class 0

So there is no null value.

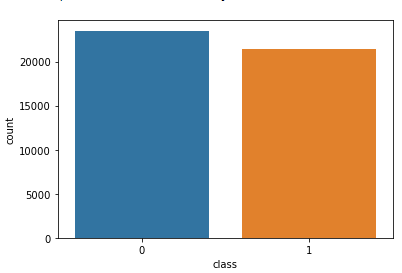
Now we have to shuffle the dataset to prevent the model to get bias. After that we will reset the index and then drop it. Because index column is not useful to us.

|  |
| --- |
| # Shuffling  data **=** data.sample(frac**=**1)  data.reset\_index(inplace**=**True)  data.drop(["index"], axis**=**1, inplace**=**True) |

Now Let’s explore the unique values in the each category using below code.

|  |
| --- |
| sns.countplot(data**=**data,                x**=**'class',                order**=**data['class'].value\_counts().index) |

Output:



**3.Preprocessing and analysis of News column:**

Firstly we will remove all the stopwords, punctuations and any irrelevant spaces from the text. For that [NLTK](https://www.geeksforgeeks.org/tokenize-text-using-nltk-python/) Library is required and some of it’s module need to be downloaded.

|  |
| --- |
| **from** tqdm **import** tqdm  **import** re  **import** nltk  nltk.download('punkt')  nltk.download('stopwords')  **from** nltk.corpus **import** stopwords  **from** nltk.tokenize **import** word\_tokenize  **from** nltk.stem.porter **import** PorterStemmer  **from** wordcloud **import** WordCloud |

Once we have all the required modules, we can create a function name preprocess text. This function will preprocess all the data given as input.

|  |
| --- |
| **def** preprocess\_text(text\_data): |
| preprocessed\_text **=** []  **for** sentence **in** tqdm(text\_data):           sentence **=** re.sub(r'[^\w\s]', '', sentence)            preprocessed\_text.append(' '.join(token.lower()  **for** token **in** str(sentence).split()  **if** token **not** **in** stopwords.words('english')))  **return** preprocessed\_text |

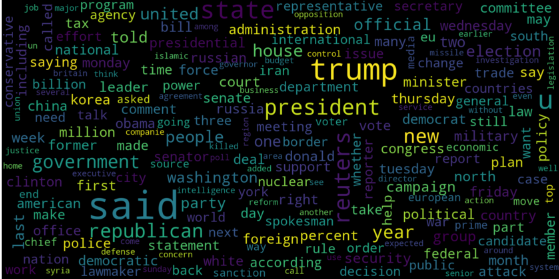
 To implement the function in all the news in the text column, run the below command.

|  |
| --- |
| preprocessed\_review **=** preprocess\_text(data['text'].values)  data['text'] **=** preprocessed\_review |

Let’s visualize the WordCloud for fake and real news separately.

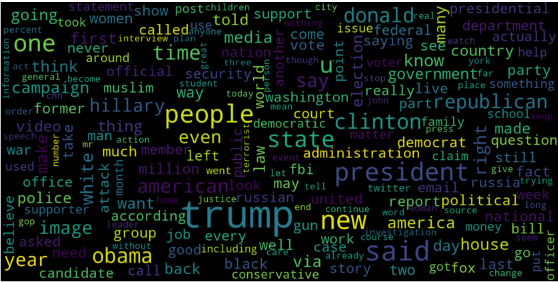
|  |
| --- |
| # Real  consolidated **=** ' '.join(      word **for** word **in** data['text'][data['class'] **==** 1].astype(str))  wordCloud **=** WordCloud(width**=**1600,                        height**=**800,                        random\_state**=**21,                        max\_font\_size**=**110,                        collocations**=**False)  plt.figure(figsize**=**(15, 10))  plt.imshow(wordCloud.generate(consolidated), interpolation**=**'bilinear')  plt.axis('off')  plt.show() |

**Output :**



|  |
| --- |
| Fake  consolidated **=** ' '.join(      word **for** word **in** data['text'][data['class'] **==** 0].astype(str))  wordCloud **=** WordCloud(width**=**1600,                        height**=**800,                        random\_state**=**21,                        max\_font\_size**=**110,                        collocations**=**False)  plt.figure(figsize**=**(15, 10))  plt.imshow(wordCloud.generate(consolidated), interpolation**=**'bilinear')  plt.axis('off')  plt.show() |

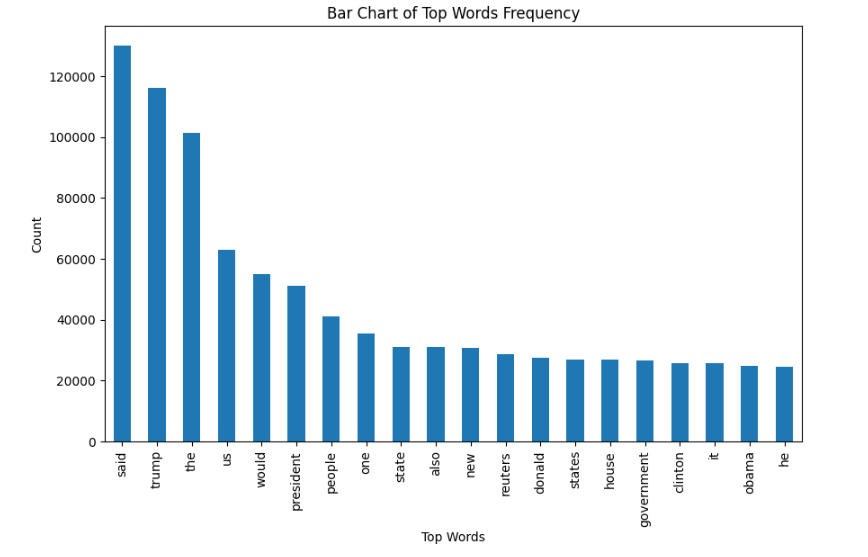
Output :



Now, Let’s plot the bargraph of the top 20 most frequent words.

|  |
| --- |
| **from** sklearn.feature\_extraction.text **import** CountVectorizer  **def** get\_top\_n\_words(corpus, n**=**None):      vec **=** CountVectorizer().fit(corpus)      bag\_of\_words **=** vec.transform(corpus)      sum\_words **=** bag\_of\_words.sum(axis**=**0)      words\_freq **=** [(word, sum\_words[0, idx])  **for** word, idx **in** vec.vocabulary\_.items()]      words\_freq **=** sorted(words\_freq, key**=lambda** x: x[1],                          reverse**=**True)  **return** words\_freq[:n]      common\_words **=** get\_top\_n\_words(data['text'], 20)  df1 **=** pd.DataFrame(common\_words, columns**=**['Review', 'count'])    df1.groupby('Review').sum()['count'].sort\_values(ascending**=**False).plot(      kind**=**'bar',      figsize**=**(10, 6),      xlabel**=**"Top Words",      ylabel**=**"Count",      title**=**"Bar Chart of Top Words Frequency"  ) |

Output :



**4.Converting text into Vectors:**

Before converting the data into vectors, split it into train and test.

|  |
| --- |
| **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn.metrics **import** accuracy\_score  **from** sklearn.linear\_model **import** LogisticRegression    x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(data['text'],                                                      data['class'],                                                      test\_size**=**0.25) |

Now we can convert the training data into vectors using [TfidfVectorizer](https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/).

|  |
| --- |
| **from** sklearn.feature\_extraction.text **import** TfidfVectorizer  vectorization **=** TfidfVectorizer()  x\_train **=** vectorization.fit\_transform(x\_train)  x\_test **=** vectorization.transform(x\_test) |

**5.Model training, Evaluation, and Prediction:**

* Now, the dataset is ready to train the model.
* For training we will use [Logistic Regression](https://www.geeksforgeeks.org/understanding-logistic-regression/) and evaluate the prediction accuracy using accuracy\_score.

|  |
| --- |
| **from** sklearn.linear\_model **import** LogisticRegression    model **=** LogisticRegression()  model.fit(x\_train, y\_train)    # testing the model  print(accuracy\_score(y\_train, model.predict(x\_train)))  print(accuracy\_score(y\_test, model.predict(x\_test))) |

**Output :**

0.993766511324171

0.9893143365983972

Let’s train with [Decision Tree](https://www.geeksforgeeks.org/decision-tree/) Classifier.

|  |
| --- |
| **from** sklearn.tree **import** DecisionTreeClassifier    model **=** DecisionTreeClassifier()  model.fit(x\_train, y\_train)    # testing the model  print(accuracy\_score(y\_train, model.predict(x\_train)))  print(accuracy\_score(y\_test, model.predict(x\_test))) |

**Output :**

0.9999703167205913

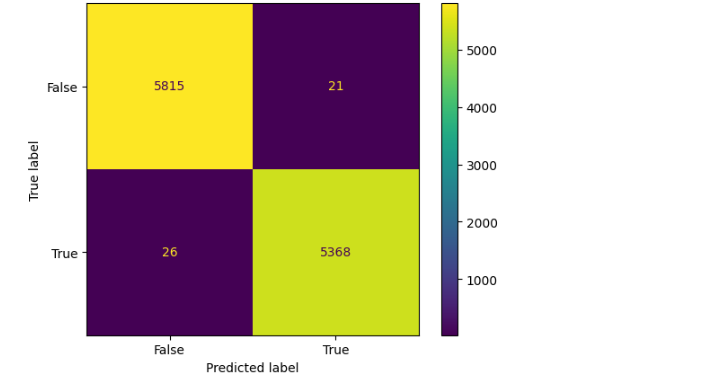
0.0.9999703167205913

0.9951914514692787

The confusion matrix for Decision Tree Classifier can be implemented with the code below.

|  |
| --- |
| # Confusion matrix of Results from Decision Tree classification  **from** sklearn **import** metrics  cm **=** metrics.confusion\_matrix(y\_test, model.predict(x\_test))    cm\_display **=** metrics.ConfusionMatrixDisplay(confusion\_matrix**=**cm,                                              display\_labels**=**[False, True])    cm\_display.plot()  plt.show() |

**Output :**



**Conclusion:**

Decision Tree Classifier and Logistic regression are performing well.

**CONCLUSION AND FUTURE WORK(Phase3):**

Project Conclusion:

In conclusion, fake news detection using Natural Language Processing (NLP) is a vital and evolving field in the fight against misinformation. NLP techniques have shown promise in identifying and flagging potentially deceptive content by analyzing linguistic patterns, sources, and context. However, it is essential to acknowledge that no single method is foolproof, and ongoing research and development are necessary to stay ahead of increasingly sophisticated fake news tactics. Collaborative efforts between researchers, technology companies, and fact-checkers are crucial in building more robust and accurate fake news detection systems to promote trustworthy information in the digital age.