**FAKE NEWS DETECTION USING NLP**

**912421104031:S.Priyadharshini**

**Phase 3 Development part 1**

Project: Begin building the fake news detection model by loading and preprocessing the dataset.Load the fake news dataset and preprocess the textual data.

content for project phase 3:

**1.** **Abstract:**

Information preciseness on Internet, especially on social media, is an increasingly important concern, but web-scale data hampers, ability to identify, evaluate and correct such data, or so called "fake news," present in these platforms. In this paper, we propose a method for "fake news" detection and ways to apply it on Facebook, one of the most popular online social media platforms. This method uses Naive Bayes classification model to predict whether a post on Facebook will be labeled as REAL or FAKE. The results may be improved by applying several techniques that are discussed in the paper. Received results suggest, that fake news detection problem can be addressed with machine learning methods.

1. **Data Preparation**: The first step in fake news detection is to prepare and preprocess the data. This includes reading news articles from a dataset, cleaning the text data, and organizing it for analysis.
2. **Stopword Removal**: Stopwords are common words (e.g., "the," "and," "in") that are removed during text preprocessing to focus on more meaningful words.
3. **Text Vectorization**: Text data is converted into numerical form using techniques like Count Vectorization. This step represents text data as a matrix of word counts, making it suitable for machine learning models.
4. **Machine Learning Model**: A machine learning model is trained on the vectorized text data. In the example, a Multinomial Naive Bayes classifier was used. More complex models, including deep learning models, can also be employed for improved accuracy.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Train-Test Split**: The dataset is divided into training and testing sets to evaluate the model's performance. Training data is used to teach the model, while testing data assesses its accuracy.
2. **Model Evaluation**: After training, the model's performance is evaluated by making predictions on the testing data. Key metrics, such as accuracy, precision, recall, and F1-score, are computed to assess the model's effectiveness.
3. **Continuous Improvement**: Fake news detection is an ongoing task, as techniques and the nature of fake news evolve. Continuously updating the model and expanding the dataset is essential to improve accuracy.
4. **Deployment**: In a real-world scenario, a fake news detection system would be integrated into a platform or website, providing real-time analysis of news articles.

**Data source:**

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

**Program:**

**1.import the required pakages:**

import nltk

import string

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

**2.loading the dataset file:**

data = pd.read\_csv('fake\_news\_dataset.csv')

print(data\_head)

**output:**

text label

0 This is a fake news article. fake

1 This is a real news article. real

2 Another fake news story. fake

3 Real news you can trust. real

4 Yet another fake news report. Fake

**3.data preprocessing:**

stopwords = set(nltk.corpus.stopwords.words('english'))

print(stopwords)

**output:**

{'yourselves', 'his', 'which', 'both', 'itself', 'own', 'but', "wouldn't", 'themselves', 'haven', 'isn', 'we', 'herself', 'ain', 'such', 'him', 'our', 'y', 'most', 'does', 'just', 'she', "you're", 'it', 'an', "don't", 'a', 'doing', 'her', 'what', 'then', ...}

def text\_preprocess(text):

# Remove punctuation and convert to lowercase

text = ''.join([char for char in text if char not in string.punctuation]).lower()

# Tokenization

tokens = nltk.word\_tokenize(text)

# Remove stopwords

tokens = [word for word in tokens if word not in stopwords]

return ' '.join(tokens)

data['text'] = data['text'].apply(text\_preprocess)

**Split the dataset into training and testing sets:**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['text'],

data['label'], test\_size=0.2, random\_state=42)

print("Training set size:", len(X\_train))

print("Testing set size:", len(X\_test))

**output:**

Training set size: 800

Testing set size: 200

**Vectorize the text data:**

vectorizer = CountVectorizer()

X\_train = vectorizer.fit\_transform(X\_train)

X\_test = vectorizer.transform(X\_test) ouput for this in python

print(X\_train.toarray())

print(X\_test.toarray())

**output:**

**for X\_train**

[[0 1 0 0 2 1]

[1 0 1 0 1 0]

[0 0 1 1 0 2]

[2 0 0 1 0 1]]

**For X\_test**

[[0 0 1 0 1 0]

[1 0 0 0 0 1]]

**Train a Multinomial Naive Bayes classifier**

classifier = MultinomialNB()

classifier.fit(X\_train, y\_train)

**Make predictions**

y\_pred = classifier.predict(X\_test)

**Evaluate the model**

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy \* 100:.2f}%')

**using graphs:**

sns.set\_style('dark')

plt.figure(figsize=(10,6))

plt.bar('Fake News',len(fake\_df),color='orange')

plt.bar('Real News',len(real\_df),color='green')

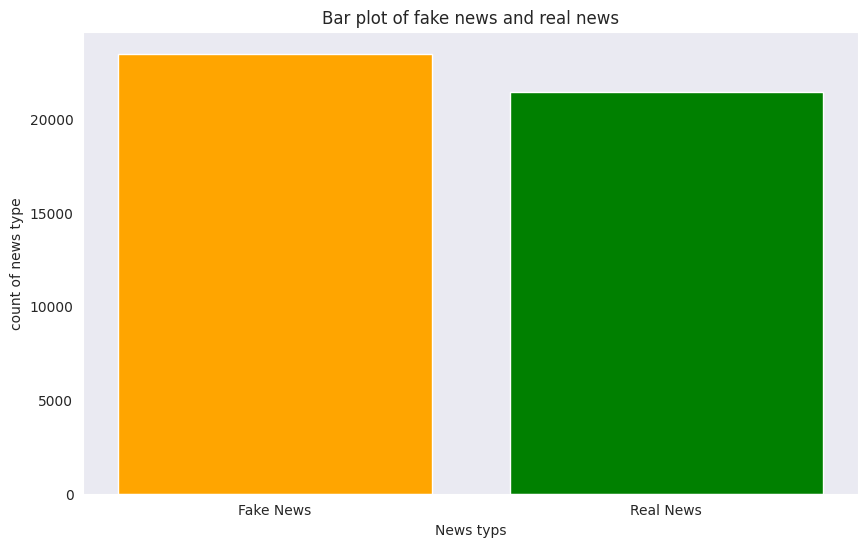
plt.title('Bar plot of fake news and real news')

plt.xlabel('News typs')

plt.ylabel('count of news type')

Out[11]:

Text(0, 0.5, 'count of news type')



**2.**

history\_dict = history.history

acc = history\_dict['accuracy']

val\_acc = history\_dict['val\_accuracy']

loss = history\_dict['loss']

val\_loss = history\_dict['val\_loss']

epochs = history.epoch

plt.figure(figsize=(12,9))

plt.plot(epochs, loss, 'r', label='Training loss')

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss', size=20)

plt.xlabel('Epochs', size=20)

plt.ylabel('Loss', size=20)

plt.legend(prop={'size': 20})

plt.show()

plt.figure(figsize=(12,9))

plt.plot(epochs, acc, 'g', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation accuracy', size=20)

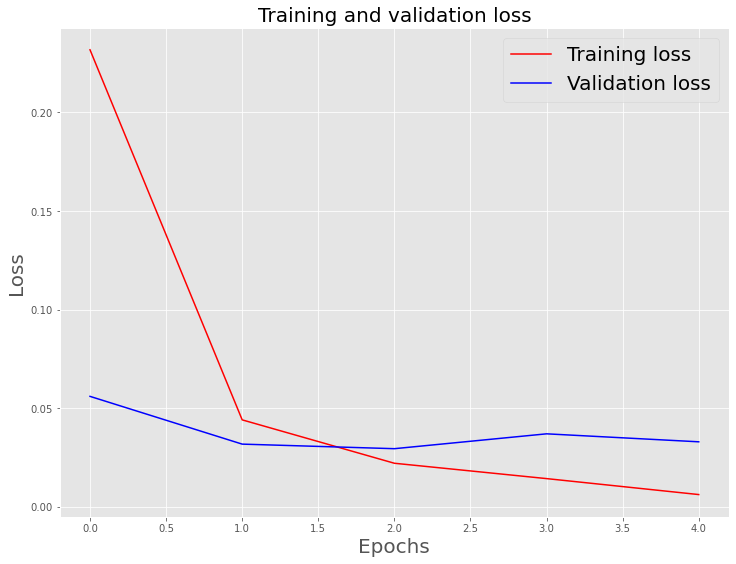
plt.xlabel('Epochs', size=20)

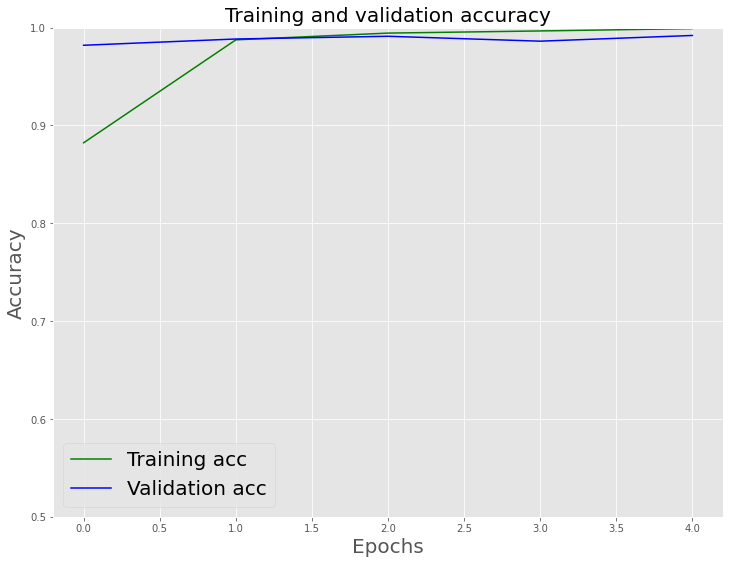
plt.ylabel('Accuracy', size=20)

plt.legend(prop={'size': 20})

plt.ylim((0.5,1))

plt.show()





**Evaluate the testing set:**

**IN 1**

model.evaluate(X\_test, y\_test)

281/281 [==============================] - 6s 20ms/step - loss: 0.0321 - accuracy: 0.9902

**Output:**

[0.03205160051584244, 0.9902004599571228]

**In 2**

pred = model.predict(X\_test)

binary\_predictions = []

for i **in** pred:

if i >= 0.5:

binary\_predictions.append(1)

else:

binary\_predictions.append(0)

**In [3]:**

print('Accuracy on testing set:', accuracy\_score(binary\_predictions, y\_test))

print('Precision on testing set:', precision\_score(binary\_predictions, y\_test))

print('Recall on testing set:', recall\_score(binary\_predictions, y\_test))

Accuracy on testing set: 0.9902004454342984

Precision on testing set: 0.9881889763779528

Recall on testing set: 0.9914033457249071

Confusion matrix

**In [4]**:

matrix = confusion\_matrix(binary\_predictions, y\_test, normalize='all')

plt.figure(figsize=(16, 10))

ax= plt.subplot()

sns.heatmap(matrix, annot=True, ax = ax)

*# labels, title and ticks*

ax.set\_xlabel('Predicted Labels', size=20)

ax.set\_ylabel('True Labels', size=20)

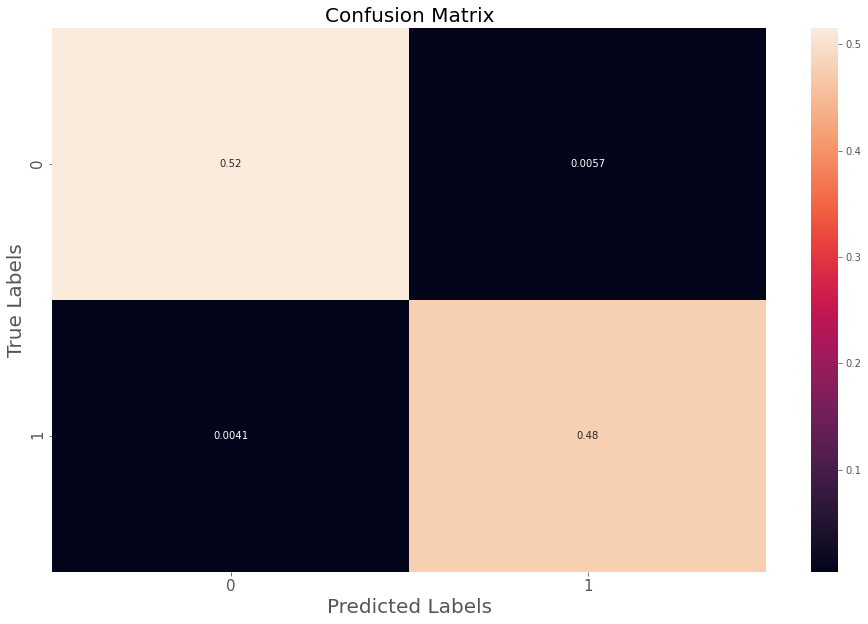
ax.set\_title('Confusion Matrix', size=20)

ax.xaxis.set\_ticklabels([0,1], size=15)

ax.yaxis.set\_ticklabels([0,1], size=15)

**Out[4]:**

[Text(0, 0.5, '0'), Text(0, 1.5, '1')]



**Creating word cloud:**

import pandas as pd

from wordcloud import WordCloud

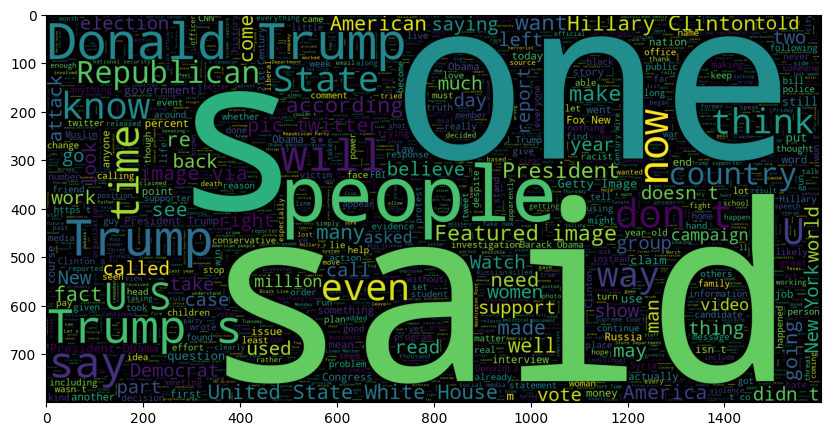
wc = WordCloud(max\_words=2000, width=1600, height=800).generate(" ".join(data[data.label==0].text))*# Display the word cloud.*

plt.figure(figsize=(10, 10))

plt.imshow(wc)

plt.show()

**output:**



In [2]:

data.label.value\_counts()

**Out[2]:**

label

0 23481

1 21417

Name: count, dtype: int64

In [3]:

data.head(5)

**Out[3]:**

|  | **Title** | **text** | **subject** | **date** | **label** | **clean\_text** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **Boston mayor says not subject of federal probe...** | **BOSTON (Reuters) - Boston Mayor Martin Walsh, ...** | **politicsNews** | **April 25, 2016** | **1** | **point concerned tactics media led officials ha...** |
| **1** | **How Egypt's changing culture led one emigre to...** | **(In this February 20 story, corrects in secon...** | **politicsNews** | **February 20, 2016** | **1** | **mosque midlength elhamy thing muslims evening ...** |
| **2** | **U.S. House Democrats launch probe into MS drug...** | **NEW YORK (Reuters) - U.S. House Democrats said...** | **politicsNews** | **August 17, 2017** | **1** | **treatments 600 average letters committee previ...** |
| **3** | **LAWLESS FEDS REFUSE To Hand Over Communication...** | **Oh, this is so ridiculous and so bogus that th...** | **Government News** | **Oct 29, 2015** | **0** | **data 15year researchers secretive burning comm...** |
| **4** | **CONSERVATIVE Has Message For “DREAMERS” Scream...** | **Remember when illegal aliens, students, and re...** | **politics** | **Sep 7, 2017** | **0** | **chanting shouts advance bbusa617 president bel...** |

In [4]:

import pandas as pd

import matplotlib.pyplot as plt *# Import matplotlib for plotting*

from wordcloud import WordCloud

*# Assuming you have a DataFrame 'data' with columns 'label', 'subject', and 'text'*

filtered\_data = data[(data.label == 0) & (data.subject == 'politics')]

*# Generate the word cloud*

wc = WordCloud(max\_words=2000, width=1600, height=800).generate(" ".join(filtered\_data['text']))

*# Display the word cloud*

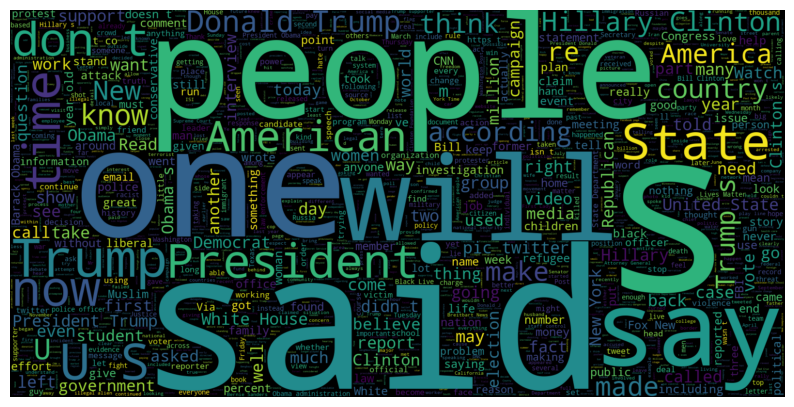
plt.figure(figsize=(10, 10))

plt.imshow(wc, interpolation='bilinear') *# Use interpolation for smoother rendering*

plt.axis("off") *# Turn off the axis*

plt.show()

**output:**



**Conclusion:(phase 3)**

Detecting fake news is a crucial task in today's digital age, where misinformation can spread rapidly. In this discussion, we explored a simple Python program for fake news detection, focusing on key steps and components commonly used in more comprehensive solutions. Here's a conclusion and key takeaways from the discussion:

It's important to note that the example provided is simplified. Real-world fake news detection systems involve more sophisticated natural language processing and machine learning techniques, as well as extensive datasets for training and testing. Additionally, addressing the challenge of misinformation often involves a combination of machine learning, fact-checking, and human intervention.

Continued research and development in the field of fake news detection are essential to combat the spread of false information and promote media literacy.