FAKE NEWS DETECTION USING NLP

# PHASE-5 PROJECT SUBMISSION DOCUMENT

## Author: R Sreeja

# INTRODUCTION:

In today's digital age, the rapid proliferation of information online has led to an increased spread of misinformation and fake news. Fake news is a term often used to refer to fabricated stories, hoaxes, or misleading information disguised as news spreading via traditional news media or online social media. The consequences of consuming and acting upon such fake news can be dire, affecting political, social, and economic landscapes. Thus, detecting and combating fake news has become a major concern.

Natural Language Processing (NLP), a branch of artificial intelligence (AI) concerned with enabling machines to understand, interpret, and generate human language, provides the tools and techniques required to automatically detect fake news. This involves analyzing the linguistic structures, semantic meaning, and even the context within which the information is presented.

# KEY FACTORS TO CREATE FAKENEWS DETECTION USING NLP

**Data Collection:** Gather a dataset of labeled news articles, where each article is marked as either "real" or "fake." Several datasets are available online for this purpose.

**Text Preprocessing:** Clean and preprocess the text data. This involves tasks like removing punctuation, stop words, and stemming or lemmatizing words.

**Feature Extraction:** Transform the text data into numerical features. Common methods include TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings (e.g., Word2Vec or GloVe).

**Model Selection:** Choose a machine learning or deep learning model for classification. Common choices include Logistic Regression, Random Forest, Naive Bayes, or neural networks like LSTM or BERT.

**Training:** Train the selected model on your labeled dataset, using the extracted features.

**Evaluation:** Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score. Cross-validation can help ensure robustness.

**Tune Hyperparameters:** Optimize the model by fine-tuning hyperparameters, such as learning rates or the number of hidden layers.

**Testing:** Use the trained model to predict the authenticity of new news articles.

**Deployment:** Implement the model in a real-world application, such as a web browser extension or a news aggregator, to help users identify fake news.

**Continuous Learning:** Continuously update and retrain the model with new data to keep it up to date with evolving fake news patterns.

Innovative for

Detecting fake news using NLP (Natural Language Processing) involves innovative techniques and approaches. Here are some ideas:

**Semantic Analysis:** Develop models that analyze the meaning and context of words in news articles to detect inconsistencies or misleading information.

**Source Credibility Analysis:** Use NLP to assess the credibility of news sources based on their historical accuracy and reputation.

**Contextual Sentiment Analysis:** Incorporate sentiment analysis into fake news detection to identify emotionally charged language that may indicate bias or manipulation.

**Deep Learning:** Explore deep learning techniques like Recurrent Neural Networks (RNNs) or Transformers for more accurate language understanding and pattern recognition.

**Multimodal Analysis:** Combine text analysis with image and video analysis to detect inconsistencies between text and accompanying media.

**Fact-Checking Integration:** Integrate fact-checking databases into the NLP pipeline to verify claims made in news articles.

**User Behavior Analysis:** Analyze user engagement and sharing patterns on social media to identify potentially viral fake news stories.

**Domain-Specific Models:** Train NLP models on domain-specific data (e.g., healthcare, politics) to improve accuracy in specific contexts.

**Adversarial Training:** Develop models that can adapt to evolving techniques used by fake news creators by using adversarial training.

**Explainable AI:** Ensure that the models provide explanations for their predictions, which can help users understand why a piece of news is considered fake or real.

**Cross-Language Detection:** Extend fake news detection to multiple languages to address the global nature of misinformation.

**Human-in-the-Loop Systems:** Create systems that combine AI with human expertise for more accurate and nuanced detection.

**Bias Detection:** Incorporate algorithms that detect bias in news articles, helping users understand potential editorial slants.

**Real-Time Monitoring:** Implement systems that continuously monitor news sources and social media for emerging fake news trends.

**Collaborative Filtering:** Use collaborative filtering techniques to identify users with a history of sharing fake news and provide them with educational resources.

**Data Augmentation:** Augment training data with synthetic examples of fake news to improve model robustness.

**Privacy-Preserving Techniques:** Develop methods that protect user privacy while still allowing for effective fake news detection in social media.

**Neccessary steps to follow:**

**1.import libraries:**

Start by importing the neccessary libraries

**Program:**

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import re

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

import matplotlib.pyplot as plt

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from keras.layers import TextVectorization

from keras.utils import pad\_sequences

from xgboost import XGBClassifier

from scipy.sparse import hstack

import random

**Load the dataset:**

**Program:**

real\_df = pd.read\_csv('/kaggle/input/fake-and-real-news-dataset/True.csv')

fake\_df = pd.read\_csv('/kaggle/input/fake-and-real-news-dataset/Fake.csv')

class 'pandas.core.frame.DataFrame'>

RangeIndex: 21417 entries, 0 to 21416

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 title 21417 non-null object

1 text 21417 non-null object

2 subject 21417 non-null object

3 date 21417 non-null object

dtypes: object(4)

memory usage: 669.4+ KB

list(real\_df.sample(5).title)

Out[5]:

['Renegade colonel surrenders in eastern Congo after clashes, seven dead',

'Factbox: Trump meetings include rapper Kanye West, Microsoft founder Bill Gates',

"At under $5 each, Trump's votes came cheap",

"Israeli air strike hits near Syria's Homs",

"Trump calls storm over Russia hacking 'political witchhunt': NYT"]

In [6]:

list(real\_df.sample(1).text)

Out[6]:

['WASHINGTON (Reuters) - The U.S. Congress, bitterly divided for years along party lines, may be mapping a bipartisan path forward that skirts around President Donald Trump when he refuses to engage constructively with lawmakers, Democrats and some lobbyists said on Monday. The path was discernible in a nearly $1.2 trillion federal spending deal carved out over the weekend to avert a government shutdown. It had Democratic fingerprints all over it, even though Republicans control Congress and the White House. White House budget director Mick Mulvaney said Trump will sign the 2017 budget bill when he receives it from Congress on Thursday or Friday. Trump, in an interview with Bloomberg on Monday, said he was “very happy” with the deal announced late on Sunday. Democrats claimed victory on issue after issue in the agreement, which will keep the lights on in Washington through the end of the federal fiscal year on Sept. 30, provided it holds up and wins final approval as expected. Trump scored a partial win, getting a commitment for up to $15 billion in additional funding for a military buildup. That was about half of what he originally asked for. No money was included for Trump’s proposed U.S.-Mexico border wall. Democratic opposition to it was solid and support from Trump’s fellow Republicans was soft. Mulvaney said Trump will seek wall funding in a budget proposal coming in late May. At a White House briefing, Mulvaney defended the concessions Trump made to reach an agreement, saying Democrats gave up on some items they had wanted as well in order to find a compromise. “Everything we got in this deal ... lines up perfectly with the president’s priorities,” he said. Democrats took an opposite view. Describing the work on Capitol Hill that went into the temporary spending pact, Senate Democratic leader Chuck Schumer told reporters: “Democrats and Republicans in the House and Senate were closer to one another than we were to the president on so many of the different issues.” Schumer and Senator Patrick Leahy, the senior Democrat on the Appropriations Committee, said they were bolstered in negotiations by the fact that several Republican senators opposed funding for Trump’s wall and his call for deep domestic spending cuts. Schumer and Leahy said the White House never tried to work with Democrats in the process. Trump treats engaging with lawmakers on legislation as “an afterthought,” said Doug Heye, a Republican strategist who worked in Congress as an aide to former House Republican leader Eric Cantor. “The power of the Oval Office can provide a lot of leverage when trying to move something on Capitol Hill,” Heye said. “We just haven’t see that level of engagement from Trump, whether it’s healthcare, or building a wall, or tax reform.” But John Feehery, a Republican strategist in Washington, said the spending bill may not be indicative of Trump’s ability to negotiate with Congress because the legislative body should have dealt with this year’s funding months ago and never have been allowed to go into the current year. “He knows that this bill was probably not going to reflect his priorities because it was old business,” Feehery said. “When it comes to new business, he has a lot more leverage in getting his priorities accomplished.“ The spending deal preserved funding for healthcare provider Planned Parenthood, which has drawn Republican ire because it performs abortions; for the Obamacare healthcare law; and for an array of environmental and other domestic programs Trump wanted to slash. White House spokesman Sean Spicer said because the legislation needs to win a supermajority of 60 votes in the 100-member Senate that cannot be achieved without Democratic support, “we couldn’t have our entire way” on the deal. Spicer said the “president’s priorities will be reflected much more” in spending yet to be worked out for the 2018 fiscal year that begins on Oct. 1. He said Trump was pleased to see the increase in military spending, a “down payment” on border security and money for scholarships to help low-income children in Washington attend private schools. The fiscal 2017 funds, which should have been locked into place seven months ago, would pay for federal programs from airport and border security operations to soldiers’ pay, medical research, foreign aid, space exploration and education. The Pentagon would win a $12.5 billion increase in defense spending for the fiscal year, with the possibility of an additional $2.5 billion contingent on Trump delivering a plan to Congress for defeating the Islamic State militant group. Congressional negotiators settled on $1.5 billion more for border security, including money for new technology and repairing existing infrastructure. Under the deal, Puerto Rico would get an emergency injection of $295 million for its Medicaid health insurance program for the poor. The impoverished U.S. territory faces a severe Medicaid funding shortfall. The U.S. government and coal companies would be required to pay out healthcare to retired coal miners, guaranteeing benefits to workers even as coal companies face bankruptcy, under the spending agreement. The deal also would reimburse New York City for money spent securing Trump and his family at Trump Tower in Manhattan. ']

In [7]:

list(fake\_df.sample(5).title)

Out[7]:

['RUSSIA HAMMERS ISIS…Kills 600 Jihadis, As China Reportedly Join Forces With Putin To Wipe Out ISIS',

'Boiler Room #106 – Did Israel Attack Damascus? + Bill Nye The PsyOp Guy',

'PAUL JOSEPH WATSON Exposes Media’s Obsession With Trump’s Call From Taiwan Leader In 14 Seconds [VIDEO]',

'U.S. Debt DECREASED By $68 BILLION In First Month Of Trump Presidency…Guess Who DOUBLED U.S. Debt During 8 Years In Office?',

'WAKE UP AMERICA! SOMALI CANDIDATES IN MINNESOTA Only Speak In Somali At Caucus…Guess Who They’re Voting For [Video]']

**Importance of loading and processing dataset:**

Loading and preprocessing the dataset is an important first step in building any machine learning model.However it isespecially important for fakenews detection.

**Data visualization:**

fig = plt.figure(figsize=(5, 5))

labels = 'Real', 'Fake'

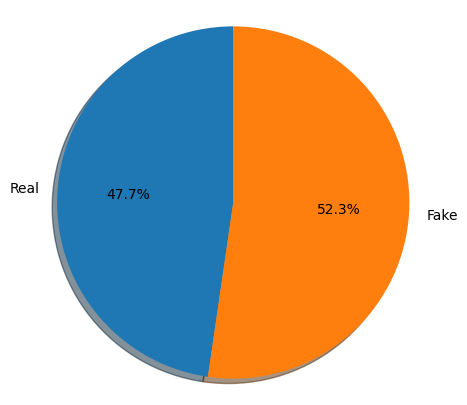
sizes = [len(real\_df), len(fake\_df)]

plt.pie(sizes, labels=labels, autopct='**%1.1f%%**',

shadow=True, startangle=90)

plt.axis('equal')

plt.show()

****

def process\_news(news):

*"""Process news function.*

*Input:*

*news: a string containing a news' text or title*

*Output:*

*newss\_clean: a list of words containing the processed news' text or title*

*"""*

stemmer = PorterStemmer()

stopwords\_english = stopwords.words('english')

*# remove hyperlinks*

news = re.sub(r'https?://[^\s\n\r]+', '', news)

*# tokenize news*

*#tokenizer = word\_tokenize*

news\_tokens = word\_tokenize(news)

news\_clean = []

for word **in** news\_tokens:

if (word **not** **in** stopwords\_english **and** *# remove stopwords*

word **not** **in** string.punctuation): *# remove punctuation*

stem\_word = stemmer.stem(word) *# stemming word*

news\_clean.append(stem\_word)

return news\_clean

news\_title = np.concatenate((positive\_titles, negative\_titles), axis=0)

In [15]:

positive\_y = np.array(real\_df.Fake\_news)

negative\_y = np.array(fake\_df.Fake\_news)

y = np.concatenate((positive\_y, negative\_y), axis=0)

In [16]:

linkcode

X\_train, X\_test, y\_train, y\_test = train\_test\_split(news\_title, y, test\_size=0.20, random\_state=0)

# **Building of the words frequency dictionary:**

def build\_freqs(news, ys):

*"""Build frequencies.*

*Input:*

*news: a list of news title or texts*

*ys: an m x 1 array with the real/fake label of each title/news*

*(either 0 or 1)*

*Output:*

*freqs: a dictionary mapping each (word, real/fake) pair to its*

*frequency*

*"""*

*# Convert np array to list since zip needs an iterable.*

*# The squeeze is necessary or the list ends up with one element.*

*# Also note that this is just a NOP if ys is already a list.*

yslist = np.squeeze(ys).tolist()

*# Start with an empty dictionary and populate it by looping over all news*

*# and over all processed words in each news.*

freqs = {}

for y, new **in** zip(yslist, news):

for word **in** process\_news(new):

pair = (word, y)

if pair **in** freqs:

freqs[pair] += 1

else:

freqs[pair] = 1

**Training of the model:**

def train\_naive\_bayes(freqs, train\_x, train\_y):

*'''*

*Input:*

*freqs: dictionary from (word, label) to how often the word appears*

*train\_x: a list of news*

*train\_y: a list of labels correponding to the news (0,1)*

*Output:*

*logprior: the log prior. (equation 3 above)*

*loglikelihood: the log likelihood of you Naive bayes equation. (equation 6 above)*

*'''*

loglikelihood = {}

logprior = 0

*# calculate V, the number of unique words in the vocabulary*

vocab = set([pair[0] for pair **in** freqs.keys()])

V = len(vocab)

*# calculate N\_pos, N\_neg, V\_pos, V\_neg*

N\_pos = N\_neg = 0

for pair **in** freqs.keys():

if pair[1] > 0:

*# Increment the number of positive words by the count for this (word, label) pair*

N\_pos += freqs[pair]

else:

*# increment the number of negative words by the count for this (word,label) pair*

N\_neg += freqs[pair]

*# Calculate D, the number of documents*

D = len(train\_y)

*# Calculate D\_pos, the number of positive documents*

D\_pos = np.sum((train\_y == 1))

*# Calculate D\_neg, the number of negative documents*

D\_neg = np.sum((train\_y == 0))

*# Calculate logprior*

logprior = np.log(D\_pos) - np.log(D\_neg)

*# For each word in the vocabulary...*

for word **in** vocab:

*# get the positive and negative frequency of the word*

freq\_pos = freqs.get((word,1),0)

freq\_neg = freqs.get((word,0),0)

*# calculate the probability that each word is positive, and negative*

p\_w\_pos = (freq\_pos + 1)/(N\_pos +V)

p\_w\_neg = (freq\_neg + 1)/(N\_neg +V)

*# calculate the log likelihood of the word*

loglikelihood[word] = np.log(p\_w\_pos) - np.log(p\_w\_neg)

return logprior, loglikelihood

# Building the fake news detection model:

**1. Data Collection:**

The first step involves gathering a diverse dataset of news articles or reports, which are already labeled as 'fake' or 'real'. This dataset serves as the foundation for training, validating, and testing the detection models.

**2. Pre-processing and Cleaning:**

Raw data often contains noise or irrelevant information. Pre-processing involves:

* Removing special characters and numbers.
* Converting all text to a uniform case (e.g., lowercase).
* Tokenizing: Breaking down the text into words or phrases.
* Removing stop words: Words like 'and', 'the', 'is' which don't provide significant meaning in the context of fake news detection.
* Lemmatization or stemming: Reducing words to their base or root form.

# Feature Engineering:

Feature engineering is one of the most crucial steps in building a robust machine learning model. For fake news detection, it entails transforming raw data (textual content) into a format that can be analyzed and used by machine learning algorithms. Here's a breakdown of feature engineering techniques specifically tailored for fake news detection:

**1. Basic Text Features:**

* **Word Count:** Total number of words in the content.
* **Character Count:** Total number of characters in the content.
* **Sentence Count:** Total number of sentences.
* **Average Word Length:** Sum of the length of all the words divided by the word count.
* **Average Sentence Length:** Sum of the word count of all the sentences divided by the sentence count.

**2. Lexical Features:**

* **Stop Words Count:** Count of commonly used words that are generally ignored in text processing (e.g., 'and', 'the').
* **Numerics Count:** Count of numbers present in the content.
* **Uppercase Word Count:** Count of words written in uppercase, which might indicate emphasis or shouting.

**3. Syntactic Features:**

* **POS Tagging:** Assign parts of speech labels to words, which can help in understanding the grammatical structure of the text.
* **NER (Named Entity Recognition):** Identify named entities (e.g., persons, organizations) in the text. The frequency or presence of certain entities may indicate the nature of the news.

**4. Sentiment Analysis:**

* **Polarity Score:** Measure the sentiment of the content. This could help in identifying exaggerated or overly emotional fake news stories.

**5. Frequency-Based Features:**

* **Bag of Words (BoW):** Represents the text based on the frequency of words.
* **TF-IDF (Term Frequency-Inverse Document Frequency):** Gives weight to terms based on their importance in a particular document relative to a set of documents.

**6. Word Embeddings:**

Word embeddings are dense vector representations of words that capture semantic meanings. These can be especially useful for fake news detection:

* **Word2Vec:** Uses neural networks to learn word associations from a large corpus of text.
* **GloVe (Global Vectors for Word Representation):** An unsupervised learning algorithm that learns vector representations for words by aggregating global word-word co-occurrence statistics from a corpus.

**7. Stance Detection:**

* Comparing the stance of the given article with the stance of multiple articles on the same topic can offer insights into its authenticity.

**8. Metadata and External Features:**

* **Author Credibility:** Track record or credibility of the news author.
* **Source Reputation:** Past reliability and bias of the publishing source or website.
* **Content Publishing Time:** The time at which the content was published can be crucial, especially when cross-referenced with real-world events or announcements.
* **External Source Verification:** Cross-check facts or claims in the content with external and credible databases or sources.

**9. Advanced Language Models:**

With the rise of transformer-based models like BERT, GPT-2, and RoBERTa, feature extraction becomes inherently built into the model. Fine-tuning these models on a fake news dataset can yield rich feature representations directly.

# Feature Extraction:

Transforming text data into a format that machine learning models can understand is crucial. Some common techniques include:

* **Bag of Words (BoW):** Represents text based on the frequency of words.
* **TF-IDF (Term Frequency-Inverse Document Frequency):** Weighs terms based on their importance in the document relative to a collection of documents.
* **Word Embeddings:** Representations like Word2Vec or GloVe that capture semantic relationships between words.

# Feature Selection:

Feature selection is the process of narrowing down the most relevant features from a given set, aiming to reduce the model's complexity and potentially enhance its performance. In the context of fake news detection, feature selection is pivotal because while the raw textual data can provide numerous features (like words or n-grams), not all of them contribute significantly to the task of classifying news as real or fake.

Here's an overview of feature selection techniques suitable for fake news detection:

**1. Filter Methods:**

* **Chi-Squared Test:** Assesses the dependency between each feature and the target class. If the feature is independent of the target class, it is considered irrelevant.
* **Information Gain:** Measures the reduction in entropy achieved by partitioning the observations based on a feature.
* **Mutual Information:** Quantifies the amount of information obtained about one variable through observing the other variable.
* **Term Frequency (TF):** You can select features based on their frequency, filtering out words or n-grams below a certain threshold.

**2. Wrapper Methods:**

* **Recursive Feature Elimination (RFE):** An iterative method that trains the model multiple times, eliminating the least impactful features in each iteration.
* **Forward Selection:** Begins with no features and adds them one by one, based on their contribution to the model's performance.
* **Backward Elimination:** Begins with all features and eliminates them one by one, based on their least contribution to the model's performance.

**3. Embedded Methods:**

* **LASSO Regression:** A linear regression model that employs L1 regularization. This regularization technique can shrink some coefficients to zero, effectively selecting a subset of the provided features.
* **Decision Trees:** Trees (like Decision Trees, Random Forests) inherently perform feature selection by splitting nodes based on features' importance.
* **Regularized Deep Learning Models:** Deep learning models with regularization techniques can inherently emphasize more informative features while minimizing the impact of irrelevant ones.

**4. Dimensionality Reduction:**

While not strictly feature selection, dimensionality reduction techniques can transform the original feature set into a reduced set while preserving as much relevant information as possible.

* **Principal Component Analysis (PCA):** A technique that transforms the original features into orthogonal components, which are linear combinations of the original features.
* **Truncated Singular Value Decomposition (SVD) for TF-IDF:** Useful for reducing the dimensionality of TF-IDF vectors.

**5. Hybrid Methods:**

* **Boruta Algorithm:** An ensemble method based on the Random Forest classifier. It compares the importance of real features with the importance of random shadow features and retains only those genuine features that have higher importance than the best shadow feature.

**6. Domain Knowledge:**

* Sometimes, domain expertise can guide feature selection. For instance, if experts believe that certain terms or phrases are often indicative of fake news, these can be explicitly highlighted or weighted in the feature set.

# PROGRAM:

* import pandas as pd
* import numpy as np
* import matplotlib.pyplot as plt
* import seaborn as sns
* from plotly.offline import init\_notebook\_mode, iplot
* import plotly.graph\_objs as go
* init\_notebook\_mode(connected=True)
* from wordcloud import WordCloud, STOPWORDS
* import re, string
* from bs4 import BeautifulSoup
* import unicodedata
* *# Feature Extraction*
* from sklearn.feature\_extraction.text import TfidfVectorizer
* *# Count Vectorizer*
* from sklearn.feature\_extraction.text import CountVectorizer
* *# Feature Selection*
* from sklearn.feature\_selection import chi2, SelectKBest, RFE, mutual\_info\_classif
* *# Feature Importance*
* from sklearn.ensemble import ExtraTreesClassifier
* *# Univariate*
* from sklearn.feature\_selection import SelectPercentile, f\_classif
* from sklearn.utils import shuffle
* In [2]:
* *# Classification Models*
* from sklearn.ensemble import RandomForestClassifier
* from sklearn.model\_selection import train\_test\_split
* from sklearn.linear\_model import LogisticRegression
* import xgboost as xgb
* from sklearn.tree import DecisionTreeClassifier
* from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay, accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score
* In [3]:
* true = pd.read\_csv("/kaggle/input/fake-and-real-news-dataset/True.csv")
* false = pd.read\_csv("/kaggle/input/fake-and-real-news-dataset/Fake.csv")
* In [4]:
* *# Data Preprocessing*
* import nltk
* from nltk.tokenize import word\_tokenize
* from nltk.corpus import stopwords, words
* stop\_words = set(stopwords.words('english'))
* *# Stemming and Lemmatization*
* from nltk.stem import SnowballStemmer, WordNetLemmatizer
* In [5]:
* true['category'] = 1
* false['category'] = 0
* In [6]:
* true.head()
* Out[6]:

|  | title | text | subject | date | category |
| --- | --- | --- | --- | --- | --- |
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 | 1 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 | 1 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 | 1 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 | 1 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 | 1 |

# Model training:

Model training for fake news detection using NLP involves a series of steps to teach a machine learning or deep learning model to distinguish between real and fake news articles based on their content. Here's a comprehensive breakdown:

**1. Data Preprocessing:**

* **Tokenization:** Divide the text into sentences and the sentences into words. Convert everything to lowercase and remove punctuation.
* **Stopwords Removal:** Remove common words (e.g., "and", "is", "in") that might not carry significant meaning in the context of fake news detection.
* **Stemming/Lemmatization:** Reduce words to their root form.
* **Vectorization:** Convert the text data into numerical format using techniques like Bag of Words, TF-IDF, or word embeddings like Word2Vec or GloVe.

**2. Dataset Splitting:**

* Split your data into training, validation, and test sets. A common split ratio might be 80% for training, 10% for validation, and 10% for testing.

**3. Model Selection:**

Choose a machine learning or deep learning model based on the dataset's size, feature complexity, and available computational resources:

* **Traditional Machine Learning Models:** SVM (Support Vector Machines), Naive Bayes, Logistic Regression, Random Forests, Gradient Boosted Trees, etc. These are good starting points and can be highly effective, especially with TF-IDF features.
* **Neural Network Models:** Multi-layer Perceptrons (MLP) can be used for a start, but more complex architectures might yield better results.
* **Recurrent Neural Networks (RNNs):** LSTMs or GRUs can be useful given their ability to capture sequential information in text data.
* **Transformer-based Models:** BERT, GPT-2, RoBERTa, etc. These models have shown state-of-the-art performance on various NLP tasks, including text classification like fake news detection.

**4. Model Training:**

* **Loss Function:** Since it's a binary classification problem (real vs. fake), Binary Cross-Entropy can be used as the loss function.
* **Optimizer:** Algorithms like Adam, RMSprop, or SGD to update network weights iteratively based on training data.
* **Regularization:** Techniques like dropout or L1/L2 regularization can help prevent overfitting.
* **Batch Training:** For large datasets, train the model in batches to make efficient use of memory.
* **Epochs:** An epoch is one forward pass and one backward pass of all the training examples. The number of epochs is the number of times the learning algorithm will work through the entire training dataset.

**5. Model Validation:**

* After each epoch, validate the model's performance on the validation set to ensure it's not just memorizing the training data (overfitting).
* Monitor metrics like accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC-ROC).

**6. Hyperparameter Tuning:**

* Adjust model parameters, like learning rate, batch size, number of layers, number of units in layers, dropout rate, etc., to improve performance.
* Tools like grid search, random search, or Bayesian optimization can automate this process.

**7. Model Evaluation:**

* Once you're satisfied with the model's performance on the validation set, evaluate it on the test set to gauge how it might perform in real-world scenarios.

**8. Iterative Refinement:**

* Use feedback from model evaluation to refine and retrain the model. Also, consider augmenting the training data or incorporating new data sources to improve performance.

**9. Deployment:**

* After obtaining a satisfactory model, deploy it in a real-world setting, be it a news aggregator, a browser plugin, or integrated into social media platforms.

# Example:

# Random forest:

import re

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from wordcloud import WordCloud

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import train\_test\_split, cross\_validate, StratifiedKFold

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix, ConfusionMatrixDisplay

from sklearn.feature\_extraction.text import TfidfVectorizer, ENGLISH\_STOP\_WORDS

from sklearn.ensemble import RandomForestClassifier

# Dataset Exploration

In [2]:

fake\_df = pd.read\_csv('../input/fake-and-real-news-dataset/Fake.csv')

fake\_df['label'] = 0

fake\_df.head()

Out[2]:

|  | title | text | subject | date | label |
| --- | --- | --- | --- | --- | --- |
| 0 | Donald Trump Sends Out Embarrassing New Year’... | Donald Trump just couldn t wish all Americans ... | News | December 31, 2017 | 0 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 | 0 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | News | December 30, 2017 | 0 |
| 3 | Trump Is So Obsessed He Even Has Obama’s Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 | 0 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 | 0 |

In [3]:

true\_df = pd.read\_csv('../input/fake-and-real-news-dataset/True.csv')

true\_df['label'] = 1

true\_df.head()

Out[3]:

|  | title | text | subject | date | label |
| --- | --- | --- | --- | --- | --- |
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 | 1 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 | 1 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 | 1 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 | 1 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 | 1 |

In [4]:

df = true\_df.copy(deep=True)

df = df.append(fake\_df, ignore\_index=True)

df

Out[4]:

|  | title | text | subject | date | label |
| --- | --- | --- | --- | --- | --- |
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 | 1 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 | 1 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 | 1 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 | 1 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 | 1 |
| ... | ... | ... | ... | ... | ... |
| 44893 | McPain: John McCain Furious That Iran Treated ... | 21st Century Wire says As 21WIRE reported earl... | Middle-east | January 16, 2016 | 0 |
| 44894 | JUSTICE? Yahoo Settles E-mail Privacy Class-ac... | 21st Century Wire says It s a familiar theme. ... | Middle-east | January 16, 2016 | 0 |
| 44895 | Sunnistan: US and Allied ‘Safe Zone’ Plan to T... | Patrick Henningsen 21st Century WireRemember ... | Middle-east | January 15, 2016 | 0 |
| 44896 | How to Blow $700 Million: Al Jazeera America F... | 21st Century Wire says Al Jazeera America will... | Middle-east | January 14, 2016 | 0 |
| 44897 | 10 U.S. Navy Sailors Held by Iranian Military ... | 21st Century Wire says As 21WIRE predicted in ... | Middle-east | January 12, 2016 | 0 |

44898 rows × 5 columns

It seems that subject column does not give us any significant value here. Fake and real news have completely different subject names. One can exploit this in order to classify fake news without even using ML. Although, we will try to solve this problem using a ML classifier.

In [5]:

print(f"Dataset subject unique values: {df['subject'].unique()}")

Dataset subject unique values: ['politicsNews' 'worldnews' 'News' 'politics' 'Government News'

'left-news' 'US\_News' 'Middle-east']

No empty cells are detected.

In [6]:

print(df.columns[df.isnull().any()])

Index([], dtype='object')

The dataset is pretty balanced. The number of fake news is almost equal to the real ones.

In [7]:

sns.countplot(x=df['label'], data=df)

Out[7]:

<AxesSubplot:xlabel='label', ylabel='count'>

### Word Cloud

In [8]:

fake\_text = ' '.join(fake\_df['title']) + ' '.join(fake\_df['text'])

true\_text = ' '.join(true\_df['title']) + ' '.join(true\_df['text'])

wordcloud\_fake = WordCloud(stopwords=ENGLISH\_STOP\_WORDS,

background\_color='white',

width=1200, height=1000).generate(fake\_text)

wordcloud\_true = WordCloud(stopwords=ENGLISH\_STOP\_WORDS,

background\_color='white',

width=1200, height=1000).generate(true\_text)

plt.figure(figsize = [8, 7])

plt.imshow(wordcloud\_fake)

plt.axis('off')

plt.title('Fake News')

plt.show()

plt.figure(figsize = [8, 7])

plt.imshow(wordcloud\_true)

plt.axis('off')

plt.title('Real News')

plt.show()

# Data Pre-processing

In this step we should clean up our data from:

* redundant columns: subject, date
* stopwords
* punctuation
* urls

In [9]:

*# Concatenate titles & text*

X = df['title'] + ' ' + df['text']

y = df['label']

punctuation\_regex = re.compile(r'[^\w\s]+')

urls\_regex = re.compile(r'(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9][a-zA-Z0-9-]+'

r'[a-zA-Z0-9]\.[^\s]{2,}|www\.[a-zA-Z0-9][a-zA-Z0-9-]+['

r'a-zA-Z0-9]\.[^\s]{2,}|https?:\/\/(?:www\.|(?!www))[a-'

r'zA-Z0-9]+\.[^\s]{2,}|www\.[a-zA-Z0-9]+\.[^\s]{2,})')

*# Apply data cleaning*

X = X.apply(lambda x: urls\_regex.sub('', str(x)))

X = X.apply(lambda x: ' '.join([item for item **in** x.split() if item **not** **in** ENGLISH\_STOP\_WORDS]))

X = X.apply(lambda x: punctuation\_regex.sub('', str(x)))

*# Split data to 80/20 ratio*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size=0.2,

random\_state=10)

# Random Forests + TF-IDF

In [10]:

*# Set up the model pipeline*

*# Note: the parameters are extracted through offline gridsearch param tuning*

*# We are using TF-IDF vectorizer in order to transform the text.*

pipeline = Pipeline(

[

('vect', TfidfVectorizer(lowercase=True, max\_features=10000, ngram\_range=(1,2))),

('clf', RandomForestClassifier(max\_features='sqrt', n\_estimators=1000, n\_jobs=-1))

]

)

*# Creating a StratifiedKFold object with 5 splits*

folds = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=10)

scores = cross\_validate(pipeline, X\_train, y\_train,

scoring=['accuracy', 'precision\_macro', 'recall\_macro', 'f1\_macro'],

cv=5,

n\_jobs=-1,

return\_train\_score=False)

print('Cross validation scores', scores)

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

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

print(classification\_report(y\_test, y\_pred))

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred)}")

cm = confusion\_matrix(y\_test, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=pipeline.classes\_)

disp.plot()

plt.show()

Cross validation scores {'fit\_time': array([652.38056231, 652.81378508, 654.09425163, 655.61904001,

179.9200809 ]), 'score\_time': array([14.12934446, 13.50879884, 12.8683238 , 12.27255249, 5.25449824]), 'test\_accuracy': array([0.99860802, 0.99819042, 0.99860802, 0.99832939, 0.99791174]), 'test\_precision\_macro': array([0.99858097, 0.99815648, 0.99859283, 0.99835155, 0.99787737]), 'test\_recall\_macro': array([0.99863022, 0.99821819, 0.99861776, 0.99830091, 0.99793887]), 'test\_f1\_macro': array([0.99860529, 0.99818685, 0.99860522, 0.99832592, 0.99790764])}

precision recall f1-score support

0 1.00 1.00 1.00 4714

1 1.00 1.00 1.00 4266

accuracy 1.00 8980

macro avg 1.00 1.00 1.00 8980

weighted avg 1.00 1.00 1.00 8980

Accuracy: 0.999554565701559

# Feature selection:

import pandas as pd

import numpy as np

In [2]:

df\_true = pd.read\_csv('../input/fake-and-real-news-dataset/True.csv')

df\_fake = pd.read\_csv('../input/fake-and-real-news-dataset/Fake.csv')

In [3]:

df\_true.head()

Out[3]:

|  | title | text | subject | date |
| --- | --- | --- | --- | --- |
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 |

In [4]:

df\_true.subject.unique()

Out[4]:

array(['politicsNews', 'worldnews'], dtype=object)

In [5]:

df\_true.shape,df\_fake.shape

Out[5]:

((21417, 4), (23481, 4))

In [ ]:

from nltk.tokenize import word\_tokenize

import nltk

nltk.download('all')

In [7]:

all\_news = []

words = []

for i **in** df\_true['title']:

all\_news.append((i,'true'))

words.extend(word\_tokenize(i))

for i **in** df\_fake['title']:

all\_news.append((i,'fake'))

words.extend(word\_tokenize(i))

all\_words = []

allowed\_word\_types = {'J','N','R','V'} *# adj,noun,adverb,verb*

for word **in** words:

pos = nltk.pos\_tag([word])[0][1][0]

if pos **in** allowed\_word\_types:

all\_words.append(word.lower())

*# all\_words = list((map(lambda x: x.lower(), all\_words)))*

all\_words = nltk.FreqDist(all\_words)

*#select only 5K words*

word\_features = list(all\_words.keys())[:5000]

In [8]:

def find\_features(news):

words = word\_tokenize(news)

features = {}

for w **in** word\_features:

features[w] = (w **in** words)

return features

import random

*#Training with little data only - Session crashed with whole data[10k->21417(true)-> 30k(fake)]*

featuresets = [(find\_features(news),category) for (news,category) **in** all\_news[10000:30000]]

print(len(featuresets))

random.shuffle(featuresets)

training\_set = featuresets[:10000]

testing\_set = featuresets[10000:]

20000

In [9]:

from nltk.classify.scikitlearn import SklearnClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

SVC\_classifier = SklearnClassifier(SVC(kernel='linear'))

SVC\_classifier.train(training\_set)

LogisticRegression\_classifier = SklearnClassifier(LogisticRegression())

LogisticRegression\_classifier.train(training\_set)

Out[9]:

<SklearnClassifier(LogisticRegression())>

In [10]:

LogisticRegression\_classifier.classify(find\_features('Donald Trump is a good businessman'))

Out[10]:

'true'

In [27]:

SVC\_classifier.classify(find\_features('Good leadership motivates others'))

Out[27]:

'true'

In [12]:

LogisticRegression\_classifier.classify(find\_features('I will make you intelligent'))

Out[12]:

'fake'

In [13]:

SVC\_classifier.classify(find\_features('You will make me a monk'))

Out[13]:

'fake'

In [14]:

nltk.classify.accuracy(SVC\_classifier,testing\_set)\*100

Out[14]:

96.66

In [15]:

nltk.classify.accuracy(LogisticRegression\_classifier,testing\_set)\*100

Out[15]:

96.46000000000001

In [16]:

import pickle

filename = 'LogisticRegression\_classifier.pickle'

pickle.dump(LogisticRegression\_classifier, open(filename, 'wb'))

filename = 'SVC\_classifier.pickle'

pickle.dump(SVC\_classifier, open(filename, 'wb'))

save\_word\_features = open('word\_features5k.pickle','wb')

pickle.dump(word\_features,save\_word\_features)

save\_word\_features.close()

In [ ]:

!pip install gradio

In [20]:

linkcode

import gradio as gr

import pickle

from nltk.tokenize import word\_tokenize

import nltk

*# nltk.download('all')*

with open('LogisticRegression\_classifier.pickle','rb') as fp:

LogisticRegression\_classifier = pickle.load(fp)

with open('word\_features5k.pickle','rb') as fp:

word\_features = pickle.load(fp)

def find\_features(news):

words = word\_tokenize(news)

features = {}

for w **in** word\_features:

features[w] = (w **in** words)

return features

def fn(news):

return LogisticRegression\_classifier.classify(find\_features(news))

iface = gr.Interface(

fn = fn,

inputs = 'text',

outputs = 'text'

)

url = iface.launch(share=True)

# Data cleaning:

In [17]:

*#Choosing the language as english*

stop = set(stopwords.words('english'))

*#Removing the stopwords from text*

def remove\_stopwords(text):

final\_text = []

text = text.lower()

for i **in** text.split():

if i.strip() **not** **in** stop:

final\_text.append(i.strip())

return " ".join(final\_text)

*#Removing the noisy text*

def clean\_text(text):

text = remove\_stopwords(text)

return text

df\_dataset['text'] = df\_dataset['text'].apply(clean\_text)

In [18]:

linkcode

print(df\_dataset['text'][0])

# Modeling:

## Methods for demonstrate the different performance with models

In [30]:

linkcode

*# This function will print the metrcis for diffeent model*

def print\_model\_report(y\_test, prediction, ML\_modelName):

print("Model report for: "+ ML\_modelName + "**\n**")

print(classification\_report(y\_test, prediction, digits=4))

*# Show confusion matrix plot*

def plot\_confusion\_matrix(y\_test, prediction, ML\_modelName, cmap):

cm = confusion\_matrix(y\_test, prediction)

ax = sns.heatmap(cm,

annot=True,

annot\_kws={'size':18,'weight':'normal'},

fmt='.20g',

cmap=cmap,

cbar\_kws={'shrink':1},

linewidths=2)

plt.title("Confusion Matrix for: " + ML\_modelName)

plt.ylabel("Actual Label")

plt.xlabel("Predict Label")

cbar = ax.collections[0].colorbar

plt.show()

*# Show ROC curve plot*

def plot\_ROC(pred\_models):

plt.figure(1)

plt.plot([0, 1], [0, 1], 'k--')

for i **in** pred\_models.index:

plt.plot(pred\_models.loc[i]['fpr'],

pred\_models.loc[i]['tpr'],

label=pred\_models.loc[i]['classifiers'] +", AUC={:.3f}".format(pred\_models.loc[i]['auc']))

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('ROC curve')

plt.legend(loc='best')

plt.show()

*# Show ROC curve plot*

def plot\_ROC\_zoom(pred\_models):

*# Zoom in view of the upper left corner.*

plt.figure(1)

plt.xlim(0, 0.2)

plt.ylim(0.8, 1)

plt.plot([0, 1], [0, 1], 'k--')

for i **in** pred\_models.index:

plt.plot(pred\_models.loc[i]['fpr'],

pred\_models.loc[i]['tpr'],

label=pred\_models.loc[i]['classifiers'] +", AUC={:.3f}".format(pred\_models.loc[i]['auc']))

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('ROC curve (zoomed in at top left)')

plt.legend(loc='best')

plt.show()

## Words to be tokens and words embeddings

In [31]:

*# Derive maxium length of token, then it will be used in word embeddings.*

max\_len = -1

for idx **in** df\_dataset['clean']:

if (len(idx)>max\_len):

max\_len = len(idx)

print(f"The maximum number of words in text is = {max\_len}")

The maximum number of words in text is = 4701

In [32]:

fig = px.histogram(x = [len(i) for i **in** df\_dataset['clean']], nbins = 150)

fig.show()

# Data Split:

In [33]:

*# Identify the independent and dependent variables!*

X = df\_dataset['clean\_joined']

y = df\_dataset['target']

In [34]:

*# Train and Test data splits!*

*# With pipelineLR approach!*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)

In [35]:

*#Create sequences of tokenized words*

MAX\_NUM\_WORDS = uni\_words *#4701*

tokenizer = Tokenizer(num\_words = MAX\_NUM\_WORDS)

tokenizer.fit\_on\_texts(X\_train)

*# Transform each token in texts to a sequence of integers!*

train\_sequences = tokenizer.texts\_to\_sequences(X\_train)

test\_sequences = tokenizer.texts\_to\_sequences(X\_test)

In [36]:

print(f"Sequence training dataset length: {len(train\_sequences)}")

print(f"Sequence testing dataset length: {len(test\_sequences)}")

Sequence training dataset length: 35918

Sequence testing dataset length: 8980

In [37]:

linkcode

*#The index number corresponds to the original words in dictionary*

for seq **in** train\_sequences[:1]:

print([tokenizer.index\_word[idx] for idx **in** seq])

# Logistic Regression Model Creation:

In [48]:

pipeline\_LR = Pipeline([('vect', CountVectorizer()),

('tfidf', TfidfTransformer(norm='l2')),

('model', LogisticRegression())])

### Logistic Regression Model Training

In [49]:

*# Fitting the Logistic Regression to the Training set!*

*# With Pipeline option!*

model\_LR = pipeline\_LR.fit(X\_train, y\_train)

In [50]:

*# Perform the training using KFolds cross validation method!*

scores = cross\_val\_score(model\_LR, X\_train, y\_train, scoring='accuracy', cv=SKF, n\_jobs=-1)

print(f" Accuracy of Logistic Regression : {round(np.mean(scores) ,4) \* 100} %")

Accuracy of Logistic Regression : 98.49 %

### Predictions using Logistic Regression

In [51]:

*# y\_pred With Pipeline*

pred\_LR = model\_LR.predict(X\_test)

In [52]:

*# Print model report for LogisticRegression*

print\_model\_report(y\_test, pred\_LR, "Logistic Regression")

Model report for: Logistic Regression

precision recall f1-score support

0 0.9891 0.9817 0.9854 4696

1 0.9801 0.9881 0.9841 4284

accuracy 0.9847 8980

macro avg 0.9846 0.9849 0.9847 8980

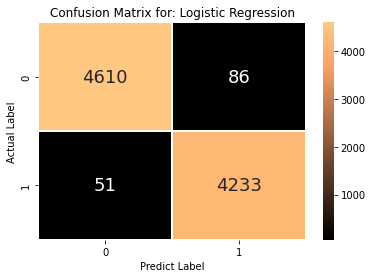
weighted avg 0.9848 0.9847 0.9847 8980

In [53]:

linkcode

*# Visualize the Confusion Matrix for LogisticRegression!*

plot\_confusion\_matrix(y\_test, pred\_LR, "Logistic Regression", 'copper')



# CONCLUSION:

Fake news detection using Natural Language Processing (NLP) represents a confluence of technological advancement and societal necessity. In an era where misinformation can spread rapidly and have profound real-world implications, leveraging NLP techniques offers a promising avenue to mitigate the challenges posed by fake news. By analyzing the linguistic patterns, context, and semantics of textual information, NLP provides tools to automatically sift through vast amounts of data and discern potential falsehoods.

However, the task is far from trivial. The nuances of human language, the evolution of misinformation tactics, and the inherent biases in data and algorithms make fake news detection a moving target. It's crucial to approach this problem with a comprehensive strategy that encompasses robust data collection, advanced modeling techniques, continuous model evaluation, and feedback mechanisms. Furthermore, collaboration across disciplines, from linguistics and journalism to computer science and ethics, is vital to ensure the holistic and fair detection of fake news.

Lastly, while NLP can serve as a powerful tool in this battle, it's essential to remember that technology alone cannot eradicate the problem. An informed and critical readership, combined with transparent and accountable information dissemination practices, will play a pivotal role in fostering a trustworthy information ecosystem. As the digital landscape continues to evolve, the interplay between NLP and fake news will remain a critical area of research, innovation, and societal discourse.