FAKE NEWS DETECTION

Project Members:

1. **Sanju Kumari**
   Roll Number: 23MCA048
   Contribution: Data Preprocessing, Training of Model, making predictive system, Report, Presentation

2. **Akansha Bharadwaj**
   Roll Number: 23MCA004
   Contribution: Data Preprocessing, Model Evaluation, Report, Presentation

Background and Problem

In today's digital world, information spreads quickly, especially on social media. However, not all news is true. Fake news—false or misleading information—can confuse people, influence opinions, and even harm society. This project aims to build a system that can automatically tell whether a news article is real or fake using advanced computer techniques like machine learning and Natural Language Processing (NLP).

Objectives of the Project

- **Analyse and Preprocess the Data**: First, we will look at a collection of news articles, clean and prepare the data so that it can be used by the computer for analysis.
- **Train a Machine Learning Model**: We will create a model using machine learning that can learn patterns in the news articles and use them to decide whether a news article is real or fake.
- **Evaluate the Model**: Once the model is trained, we will test how well it performs using different metrics (like accuracy), and make sure it works correctly to classify news articles.

Libraries Used

1. **NumPy**: Helps with number calculations and working with arrays, making it easier to handle data.
2. **Pandas**: Used to load, organize, and clean up data in a table format, which is easy to work with.
3. **NLTK**: Provides tools for understanding and processing text, like breaking down sentences and finding root words.
4. **scikit-learn**: Includes tools to build machine learning models and check how well they perform

[1]:
```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords #words that are of no importance
from nltk.stem import PorterStemmer #root word for a particular word
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Dataset Description

The dataset used in this project comes from **Kaggle** and contains various news articles. Each article has the following details:

- **id**: A unique identifier for each news article.
- **title**: The title of the news article.
- **author**: The name of the person who wrote the article.
- **text**: The full text of the article (sometimes it might be incomplete).
- **label**: This shows whether the article is real or fake:
    - **1**: Fake news
    - **0**: Real news

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Exploratory Data Analysis (EDA)

EDA is a crucial step to better understand the dataset before building a machine learning model. Here's what we looked at:

1. **About Dataset:**

```
news_dataset.shape
```

```
(20800, 5)
```

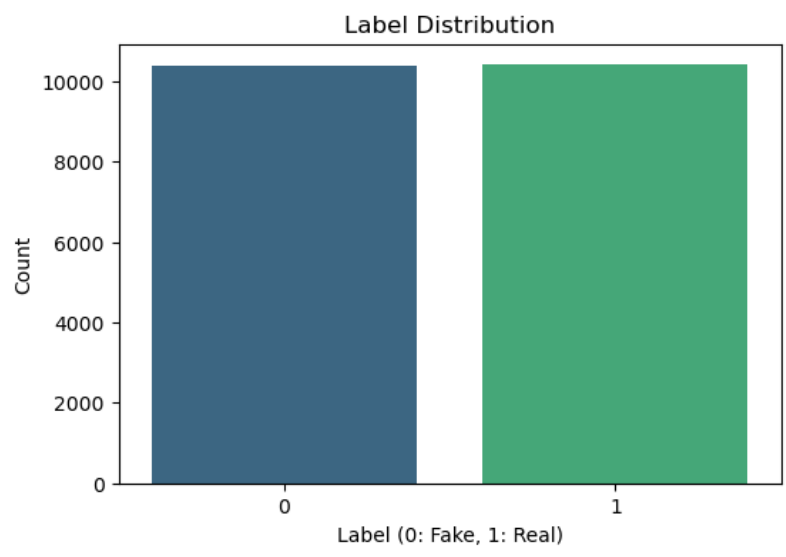
```
# Dataset Overview
news_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

```
#counting the number of missing values in each column
news_dataset.isnull().sum()
```

```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

2. **Class Distribution**: We examined how the two categories—real news and fake news—are distributed in the dataset. It's important for the dataset to be balanced, meaning there should be roughly equal numbers of real and fake news articles. If one category is overrepresented, the model may become biased and perform poorly on the underrepresented class.



Data Preprocessing

Data preprocessing helps improve the quality of the data before it is used to train the model. The following steps were performed on the news articles:

1. **Stopwords Removal**: We removed common words like "the," "is," and "in" that don't add much meaning to the analysis.

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

```
[nltk_data] Downloading package stopwords to C:\Users\Sanju
[nltk_data]     Kumari\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

```
#printing the stopwords in English
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', '
e', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'he
hat', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these
'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and'
'about', 'against', 'between', 'into', 'through', 'during', 'be
```

2. **Stemming**: We reduced words to their basic form. For example, "running" became "run." This helps the model focus on the core meaning of words, reducing unnecessary complexity.

```
port_stem = PorterStemmer()
```

```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ', content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

```
news_dataset['content'] = news_dataset['content'].apply(stemming)
```

3. **Text Vectorization**: We converted the text into numbers using a method called **TF-IDF** (Term Frequency-Inverse Document Frequency). This technique helps the model understand which words are more important in the context of all the articles, focusing on the key features.

```
# converting the textual data to numerical data
vectorizer = TfidfVectorizer() #term frequency
vectorizer.fit(X)

X = vectorizer.transform(X)
```

Methodology

The methodology for this project involves several key steps, each designed to prepare the data and train the model effectively:

1. **TF-IDF Vectorization**: This step converts the text data into numbers. It calculates the importance of each word in relation to all the other words in the dataset, which helps the model understand the key terms that are most useful for classifying real and fake news.
2. **Logistic Regression**: We use Logistic Regression because it's a simple and effective method for binary classification (deciding between two options—in this case, real or fake). It takes the numerical features from TF-IDF and predicts whether a news article is real or fake.

```
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
#accuracy score on the traing data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.9865985576923076
```

```
#accuracy score on the test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

Overall Workflow:

1. **Data Loading**: The dataset is imported and prepared in a format that can be easily analysed.
2. **Preprocessing**: The steps like stopword removal, stemming, and other text cleaning techniques are applied to make the text ready for analysis.
3. **Feature Extraction**: TF-IDF is used to convert the processed text into numerical features, which will serve as input for the model.
4. **Model Training**: The Logistic Regression model is trained using the processed and transformed data to learn the patterns in the news articles.
5. **Model Evaluation**: After training, the model is tested on a separate set of data (the test set) to check how well it predicts real vs. fake news. This ensures that the model is reliable and can generalize well to new, unseen data.

Experimental Results and Discussion

The model's performance was evaluated using a test set, focusing on the accuracy metric, which reflects the proportion of correctly classified articles.

Evaluation

```
#accuracy score on the traing data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
#accuracy score on the test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the training data :  0.9865985576923076
Accuracy score of the test data :  0.9790865384615385
```

Results Summary

Metric	Value
Accuracy	97.9%

Observations

- **Accuracy**: The Logistic Regression model achieved an impressive accuracy of **97.9%**, indicating that it correctly identified the class (real or fake news) in 93.5% of the test samples. This suggests that the feature engineering (e.g., TF-IDF vectorization) and model choice were effective in distinguishing between real and fake news articles.

- **Feature Effectiveness**: The TF-IDF vectorization method, which captures the importance of words in a document relative to the entire corpus, proved to be a good feature extraction method for this task. It helped the model identify important terms that separate fake news from real news.

Potential Enhancements

While the current results are promising, there are several areas where the model can be improved:

- **Advanced Models**: While Logistic Regression performed well, exploring more complex models, such as **neural networks** could potentially lead to better results. Neural networks, especially deep learning architectures, can capture intricate patterns in text and improve classification performance.

- **Additional Preprocessing Techniques**: Techniques such as **n-grams** (sequences of n consecutive words) could enhance the model's ability to capture context and the relationships between consecutive words in a sentence
- **Hyperparameter Tuning**: The model's performance could be enhanced further by tuning hyperparameters like **regularization strength** in Logistic Regression, or trying **ensemble models** like **Random Forests** or **Gradient Boosting Machines (GBMs)** for better generalization.

Conclusions

This project successfully developed a machine learning pipeline for classifying news articles as real or fake using a **Logistic Regression** model. The **TF-IDF vectorizer** was effective in transforming the textual data into numerical features that the model could use to make predictions. The model's **accuracy of 97.9%** demonstrates that the approach works well for this classification task, and the results are promising for distinguishing between real and fake news.

The project demonstrates that machine learning, specifically **Logistic Regression** with **TF-IDF vectorization**, can be highly effective for classifying news articles as real or fake. With further enhancements, especially with deep learning models and additional features, this system could evolve into a powerful tool for real-time fake news detection, contributing to mitigating the spread of misinformation.

Bibliography

- Various online resources on Logistic Regression and TF-IDF from scikit-learn documentation.
- **Text Mining: IBM** and **Qualtrics** emphasize the use of text mining techniques, such as NLP and ML, to transform unstructured text (like news articles and social media posts) into structured data. These methods help identify patterns and trends in text, making it easier to detect fake news.
- Detection of Fake News Text Classification on COVID-19 Using Deep Learning Approaches by Andrei Korobeinikov, article on Wiley online library website.