FAKENEWS DETECTION USING PYTHON AND MACHINE LEARNING:

Introduction:

Fake news detection is the process of identifying and classifying false or misleading information presented as news. With the widespread availability of social media and the internet, fake news has become a growing problem that can have serious consequences. It can spread quickly, influence public opinion, and even impact political and social outcomes.

The detection of fake news is crucial in maintaining the integrity of the news and information we receive. Detecting fake news involves using specialized techniques and algorithms to identify patterns and markers that are indicative of fake news. These methods involve analysing language, content, and sources to determine the credibility and accuracy of the information presented.

Overall, fake news detection is an essential tool in protecting the public from misinformation and ensuring that news and information are accurate and trustworthy.

In this report, we will analyse the given dataset of news articles and perform

- EDA and machine learning modelling on it
- Preprocess the text data
- Apply feature extraction techniques
- Build a logistic regression model to predict the class of news articles (fake or real).
- Evaluate the performance of the model using accuracy score and confusion matrix.

Finally, we will conclude with future ideas.

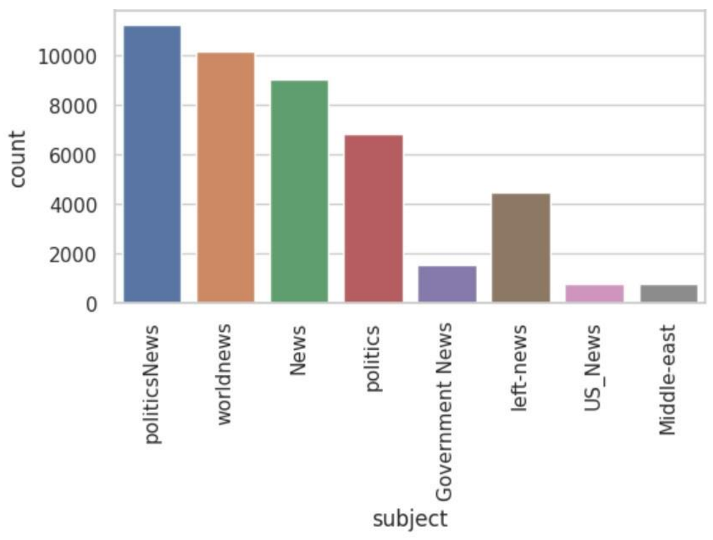
References:

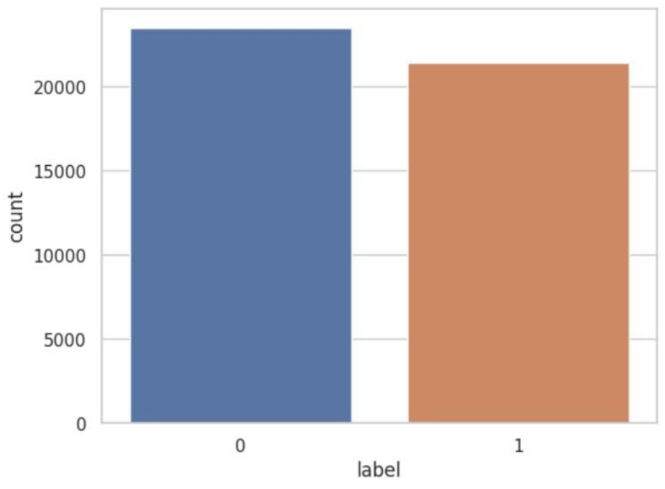
https://www.youtube.com/watch?v=nacLBdyG6jE

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-errormetrics/#Confusion_Matrix https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/

Data Analysis:

We have used two datasets, one with real news articles and the other with fake news articles. We have concatenated these datasets and added a new column "label" to identify whether the news article is real or fake. We have visualized the distribution of news articles across different subjects using a count plot.





<u>EDA in Machine Learning:</u>

We have pre-processed the text data by converting it to lowercase, removing numbers and punctuation, and removing extra whitespaces. We have used the TfidfVectorizer feature extraction technique to convert the textual data to numerical data. We have split the dataset into training and testing sets and used logistic regression as our machine learning model.

Logistic regression is a popular model for binary classification problems. It uses a logistic function to model the probability of the output class.

We have used logistic regression because it is a simple and effective model for binary classification problems. It is also easy to interpret and understand the results.

We have used the accuracy score to evaluate the performance of the model on both the training and testing sets. We have also used the confusion matrix to understand the performance of the model in classifying the news articles as real or fake. It is a useful tool for evaluating the performance of a model, especially when dealing with imbalanced datasets.

Code:

```
import pandas as pd                                   #import necessary libraries

import seaborn as sns import matplotlib.pyplot as plt

from nltk.corpus import stopwords      #Downloads necessary NLTK resources using the nltk.download()

import string

from sklearn.feature_extraction.text import TfidfVectorizer

from tqdm import tqdm import re import nltk import os

nltk.download('punkt')

nltk.download('stopwords') from

nltk.tokenize import word_tokenize from

sklearn.model_selection import

train_test_split from sklearn.metrics

import accuracy_score from

sklearn.linear_model import

LogisticRegression

data_directory = 'data/'    #load the dataset if not

os.path.exists(data_directory):

   !mkdir data/

   !wget https://onlineacademiccommunity.uvic.ca/isot/wp-content/uploads/sites/7295/2023/03/News-_dataset.zip -
-directory-prefix=data/

   !unzip data/News-_dataset.zip -d data/

fake_data = pd.read_csv('data/Fake.csv')  #Sets up a data directory and downloads a dataset of fake and real news

fake_data.head()

 true_data = pd.read_csv('data/True.csv')

true_data.head()

true_data["label"] = 1    #Adds a "label" column to indicate whether each article is

fake_data["label"] = 0   # fake (0) or real (1).

data = pd.concat([true_data, fake_data], axis=0) # Concatenates the fake and true datasets into a single DataFrame

plt.figure(figsize = (6,3))                           # Plots a countplot to visualize the distribution of subjects

 sns.set(style = "whitegrid",font_scale = 1.0) chart = sns.countplot(x = "subject", data = data)

chart.set_xticklabels(chart.get_xticklabels(),rotation=90)
```

```
data['text'] = data['title'] +' '+data['text']
del data['title']     # del keyword to remove the ‘title’ from the data dataset
del data['subject']   # del keyword to remove the ‘subject’ from the data dataset
del data['date']  # del keyword to remove the ‘date’ from the data dataset
data.head()
data.shape data.isnull().sum()
data = data.sample(frac=1).reset_index(drop=True) data.head()
sns.countplot(data=data,
            x='label',
order=data['label'].value_counts().index) def
preprocess_text(text):
   text = text.lower()                                         # Convert the text to lowercase
   text = re.sub(r'\d+', '', text)                          # Remove numbers from the text     text =
   text.translate(str.maketrans('', '', string.punctuation))       # Remove punctuation from the text

   text = re.sub(r'\s+', ' ', text)                            # Remove extra whitespaces from the text
   text = text.strip()
return text
data['text'] = data['text'].apply(preprocess_text) print(data['text'])
X = data['text'].values                                       #separating the data and label
Y = data['label'].values
vectorizer = TfidfVectorizer()                                # converting the textual data to numerical data
vectorizer.fit(X)
X = vectorizer.transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify=Y, random_state=2)
model = LogisticRegression() model.fit(X_train, Y_train)
X_train_prediction = model.predict(X_train)                    # accuracy score on the training data
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data: {:.2%}'.format(training_data_accuracy))
X_test_prediction = model.predict(X_test)          # accuracy score on the test data
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : {:.2%}'.format(test_data_accuracy))
```

```
X_new = X_test[0]

prediction = model.predict(X_new)

print(prediction)

if (prediction[0]==0):

        print('The news is Fake')

else:

        print('The news is Real')

from sklearn.metrics import confusion_matrix                    #DataFlair - Build confusion matrix

matrix = confusion_matrix(X_test_prediction,Y_test, labels=[1,0])

 print(matrix)
```

Future Ideas:

In the future, we can explore other feature extraction techniques such as word embeddings and topic modelling. We can also try other machine learning models such as support vector machines, random forests, and neural networks. We can also collect more data and perform deep learning-based models to improve the accuracy of the classification. Additionally, we can perform sentiment analysis to understand the emotions and opinions expressed in the news articles.