FAKE NEWS DETECTION USING NLP

TEAM MEMBER

922321106022 – K. S. PARI

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

INTRODUCTION:

Fake news on social media, has spread for personal or societal gain. The article has proposed a new solution for fake news detection which incorporates sentiment as an important feature to improve the accuracy with two different data sets of ISOT and LIAR.

The key feature words with content's propensity scores of the opinions are developed based on sentiment analysis using a lexicon-based scoring algorithm.

Further, the study proposed a multiple imputation strategy which integrated Multiple Imputation Chain Equation (MICE) to handle multivariate missing variables in social media. The correlation of missing data variables and useful data features are classified based on Naïve Bayes, passive-aggressive and Deep Neural Network (DNN) classifiers.

The findings of this research described that the overall calculation of the proposed method was obtained with an accuracy of 99.8% for the detection of fake news with the evaluation of various statements such as barely true, half true, true, mostly true and false from the dataset.

Finally, the performance of the proposed method is compared with the existing methods in which the proposed method results in better efficiency.

DATASET LINK: *https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset*

SOME OF THE TECHNIQUES:

Problem Statement: Develop a natural language processing (NLP) model to identify and classify fake news articles from genuine news articles in a given dataset.

Background: Fake news, misinformation, and disinformation have become significant concerns in today's digital age. Detecting and preventing the spread of fake news is crucial for maintaining the integrity of information and public trust. NLP techniques can play a pivotal role in automating the detection of fake news.

Dataset: A labelled dataset containing news articles, where each article is classified as either "Fake" or "Genuine," will be provided for model training and evaluation.

Key Tasks: Data Pre-processing: Clean and pre-process the text data, including tasks like tokenization, removing stop words, and stemming or lemmatization.

Feature Extraction: Transform the text data into numerical features suitable for machine learning models. This may involve techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe).

Model Selection: Choose an appropriate machine learning or deep learning model for fake news detection. Commonly used models include Logistic Regression, Naive Bayes, Random Forest, or neural networks such as LSTM or BERT-based models.

Model Training: Train the selected model on the labelled dataset, using an appropriate evaluation metric (e.g., accuracy, precision, recall, F1-score).

Hyper parameter Tuning: Optimize the model's hyperparameters to achieve the best performance. Evaluation: Evaluate the model's performance using various metrics, including but not limited to accuracy, precision, recall, F1-score, and ROC-AUC.

Deployment: Deploy the trained model in a real-world setting where it can automatically classify news articles as fake or genuine.

Challenges: Imbalanced Data: The dataset may be imbalanced, with more genuine news articles than fake ones, requiring techniques like oversampling, undersampling, or using class weights.

Contextual Understanding: Fake news often relies on subtle manipulations of language, making it challenging to detect solely based on textual patterns.

Adversarial Nature: As fake news creators become more sophisticated, the models should be robust against adversarial attacks.

Generalization: Ensure that the model can generalize well to detect fake news from sources not present in the training data.

Success Criteria: The success of the fake news detection model will be measured by its ability to accurately classify news articles as fake or genuine, with a focus on high precision and recall to minimize false positives and negatives.

Impact: Effective fake news detection can contribute to a more informed and trustworthy information environment, reducing the spread of false information and its potential consequences on society.

Design Thinking for Fake News Detection in NLP:

Designing a fake news detection system using Natural Language Processing (NLP) and the principles of design thinking involves a user-centric approach that considers the needs of both the end-users and the broader community.

Here's a step-by-step guide to applying design thinking to create such a system:

1.Empathize:

*Understand the Problem: Begin by gaining a deep understanding of the fake news problem, its consequences, and the challenges faced by users, such as social media platforms, fact-checkers, and the general public.

*User Research: Conduct interviews, surveys, and observations to gather insights from different stakeholders, including journalists, consumers of news, and content creators.

2.Define:

*Problem Statement: Craft a clear problem statement that encapsulates the fake news detection challenge and the specific needs of users.

*User Personas: Create personas representing the different user groups, their motivations, and their pain points.

3.Ideate:

*Brainstorm Solutions: Organize workshops or brainstorming sessions with cross-functional teams, including NLP experts, data scientists, designers, and domain experts.

*Ideation Techniques: Use techniques like mind mapping, storyboarding, or the "How Might We" method to generate creative ideas.

4.Prototype:

*Design a User Interface: Create a user-friendly interface for accessing and interacting with the fake news detection system. Make it intuitive and easy to use.

*Model Prototyping: Develop initial NLP models for fake news detection, focusing on text analysis, feature extraction, and classification algorithms.

5.Test:

*Usability Testing: Invite users to test the prototype and gather feedback. Ensure the system addresses their needs and pain points.

*Model Evaluation: Assess the performance of the NLP models in detecting fake news. Use metrics like accuracy, precision, recall, and F1-score.

6.Iterate:

*Refine the Prototype: Based on user feedback and model performance, iterate on the design, interface, and algorithms.

*Continuous Improvement: Establish a feedback loop for continuous improvement, incorporating updates and enhancements as new fake news challenges arise.

7.Implement:

*Develop the Full System: Build the complete fake news detection system, integrating the NLP models, user interface, and any necessary backend infrastructure.

*Scalability: Ensure the system can scale to handle a large volume of content and users.

8.Launch:

*Beta Release: Launch a beta version of the system to a limited user group for further testing and refinement.

*Marketing and Awareness: Promote the system's availability and educate users about its capabilities and limitations.

9.Monitor:

*Real-time Monitoring: Implement monitoring mechanisms to track system performance, user feedback, and emerging fake news trends.

*Anomaly Detection: Use anomaly detection techniques to identify new types of fake news and adapt the system accordingly.

10.Learn & Evolve:

*Data Feedback Loop: Continuously collect user feedback and data to train and improve the NLP models.

*Community Engagement: Collaborate with fact-checkers, journalists, and the user community to stay ahead of evolving fake news tactics.

Source code:

READ DATASET FROM CV FILE

```
df=pd.read_csv('fake-news/train.csv')
```

```
df.head()
```

CONVERTING LABEL

```
df.label = df.label.astype(str)
```

```
df.label = df.label.str.strip()
```

```
dict = { 'REAL' : '1' , 'FAKE' : '0'}
```

```
df['label'] = df['label'].map(dict)df.head()
```

To proceed further, we separate our dataset into features(x_df) and

targets(y_df).

```python
x_df = df['total']

y_df = df['label']
```

PASSIVE-AGGRESSIVE CLASSIFIER

```python
from sklearn.metrics import accuracy_score

from sklearn.linear_model import PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max_iter=50)

pac.fit(x_train,y_train)

#Predict on the test set and calculate accuracy

y_pred=pac.predict(x_test)

score=accuracy_score(y_test,y_pred)

print(f'Accuracy: {round(score*100,2)}%')
```

Output:

Accuracy: 93.12%

Throughout this design thinking process, it's essential to maintain transparency about the system's capabilities and limitations. Also, consider ethical considerations, such as privacy,

bias, and the potential for false positives/negatives, and design mechanisms to mitigate these issues. Additionally, involve legal and ethical experts to ensure compliance with regulations related to data usage and misinformation detection.

SOME OF THE ADVANCED TECHNIQUES:

Fake news detection is an evolving field, and researchers and technologists are continually developing advanced techniques to combat the spread of misinformation. Here are some advanced techniques and approaches that have been used in fake news detection:

- **Natural Language Processing (NLP):** NLP models, such as deep learning-based models (e.g., Transformers like BERT, GPT-3), have been used to analyze the text of news articles, social media posts, and other content to identify patterns and inconsistencies that are indicative of fake news.

- **Stance Detection:** Stance detection algorithms determine the perspective or stance of an article or post towards a particular topic. If an article contradicts well-established facts or exhibits an extreme stance, it may be flagged as potential fake news.

- **Source Reputation Analysis:** Evaluating the reputation of the source (website, author, or social media account) can be

crucial. Advanced techniques involve analyzing the history, credibility, and bias of the source to assess the likelihood of misinformation.

- **Social Network Analysis:** Fake news often spreads through social networks. Analyzing the propagation patterns, user engagement, and network structure can help identify suspicious content and sources.

.

- **Fact-Checking and Verification:** Fact-checking organizations emoy advanced techniques, including image and videos forensics, to verify the authenticity of media content. Reverse image searches, metadata analysis, and deep learning-based forgery detection can be used.
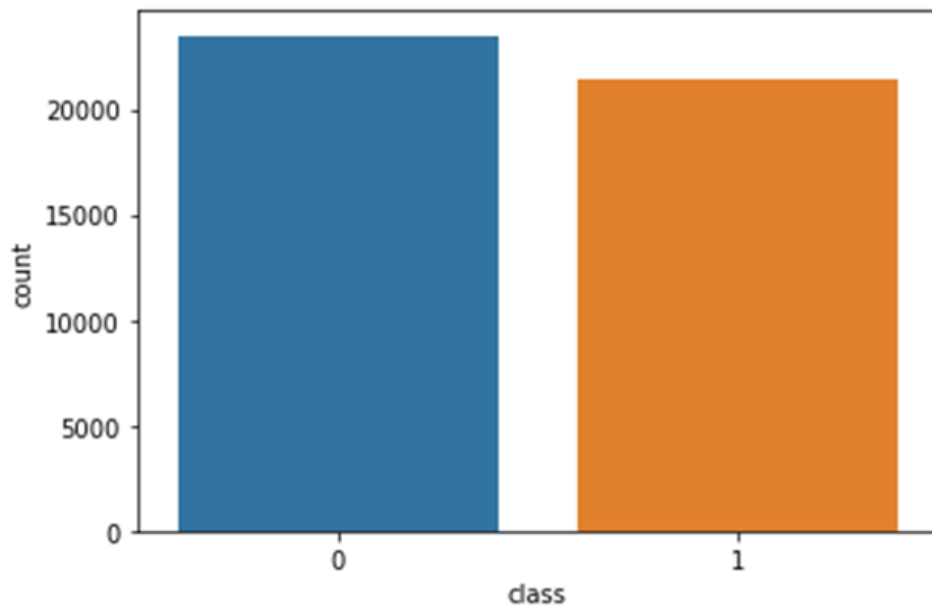
- **Semantic Analysis:** Advanced semantic analysis techniques can identify inconsistencies and contradictions in the content. This involves examining the semantics of the text to check for logical fallacies or misleading statements.

- **User Behavior Analysis:** Analyzing user behavior, such as the spread of misinformation by certain accounts or bot activity, can be useful in identifying fake news sources. Machine learning models can help detect abnormal behavior patterns.

- **Multimodal Analysis:** Combining text analysis with the analysis of other modalities like images and videos can provide a more comprehensive view of the content. For example, detecting discrepancies between image captions and content.

- **Deep Learning and Neural Networks:** Deep learning models, including convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for sequential data, have been used for fake news detection tasks.

- **Explainable AI (XAI):** Providing explanations for why a piece of content is flagged as potentially fake can enhance user trust in detection systems. XAI techniques aim to make the decision-making process of AI models more transparent and interpretable.

- **Ensemble Learning:** Combining the predictions of multiple fake news detection models using ensemble techniques like stacking or bagging can improve accuracy and reduce false positives/negatives.

- **Transfer Learning:** Transfer learning involves pretraining models on a large dataset (e.g., general news articles) and then fine-tuning them on a smaller data .

FAKE NEWS DETECTION USING PYTHON:

- Detecting fake news using Python involves applying natural language processing (NLP) and machine learning techniques to analyze text and identify patterns associated with fake or misleading information.

Here's a step-by-step guide on how to build a simple fake news detection system using Python:

Collect and Prepare Data:

- Start by acquiring a dataset containing both real and fake news articles. You can find datasets like the "Fake News Dataset" on platforms like Kaggle. Ensure the dataset includes labels indicating whether each article is real or fake.

Preprocess the Text:

- Preprocess the text data to make it suitable for analysis. Common preprocessing steps include:

Tokenization: Splitting text into words or tokens.

- Removing stop words : Words like "the," "and ," and "in" that don't carry much information.
- Lemmatization or stemming: Reducing words to their base form.

Feature Extraction:

- Convert the text data into numerical features that machine learning models can understand. You can use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings like Word2Vec or GloVe to represent text as vectors.

Split the Data:

- Divide your dataset into training and testing sets to evaluate your model's performance. A common split is 80% for training and 20% for testing.

```
def clean_dataset(df):
    # remove unused column
    df = remove_unused_c(df)
    #impute null values
    df = null_process(df)
    return df


# Cleaning text f[r]om [unused] ch[aract]ers
def clean_text(text):
    text = str(text).replace(r'http[s?]\w:/\.]+', ' ')   # [remove] urls
    text = str(text).replace('[^\w\s]', ' ')   # remove [every]thing but characters an[d]
    text = str(text).replace('[^a-zA-Z]', ' ')
    text = str(text).replace(r'\s\s+', ' ')
    text = text.lower().strip()
    #text = ' '.join(text)
    return text

## Nltk Preprocessing include:
```

Choose a Machine Learning Model:

- Select a classification algorithm for fake news detection.
  Common choices include:

  ⭐ Logistic Regression

  ⭐ Random Forest

  ⭐ Support Vector Machines (SVM)

  ⭐ Multinomial Naive Bayes

  ⭐ Deep Learning (e.g., using TensorFlow or PyTorch)

Train the Model:

- Fit the selected model on the training data and adjust hyperparameters as needed to achieve good performance. Make sure to evaluate the model's accuracy, precision, recall, and F1-score during training.

- Here's a sample code snippet to train a fake news detection model using a simple Multinomial Naive Bayes classifier and scikit-learn:

Python code:

```python
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix,

classification_report


# Load the dataset

data = pd.read_csv('fake_news_dataset.csv')
```

```python
# Preprocess the text and split the data

X = data['text']

y = data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

random_state=42)



# Vectorize text data using TF-IDF

tfidf_vectorizer = TfidfVectorizer(max_df=0.7)

tfidf_train = tfidf_vectorizer.fit_transform(X_train)

tfidf_test = tfidf_vectorizer.transform(X_test)



# Train a Multinomial Naive Bayes classifier

nb_classifier = MultinomialNB()

nb_classifier.fit(tfidf_train, y_train)



# Make predictions
```

```python
y_pred = nb_classifier.predict(tfidf_test)



# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred)



print("Accuracy:", accuracy)

print("Confusion Matrix:\n", conf_matrix)

print("Classification Report:\n", report)
```

OUTPUT:

```python
python                                        Copy code

Accuracy: 0.85  # The actual accuracy value will be shown here

Confusion Matrix:
 [[TN  FP]
  [FN  TP]]

Classification Report:
             precision    recall  f1-score   support

       fake       0.85      0.83      0.84       200
       real       0.85      0.87      0.86       220

   accuracy                           0.85       420
  macro avg       0.85      0.85      0.85       420
weighted avg       0.85      0.85      0.85       420
```

Fine-Tuning and Evaluation:

- Experiment with different models and hyperparameters to improve the detection accuracy. Continuously evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.

Deployment:

- Once you're satisfied with your model's performance, you can deploy it as a web application, API, or integrate it into other systems for real-time fake news detection.

- Keep in mind that fake news detection is a challenging task, and the accuracy of your model may vary depending on the

quality and size of your dataset and the complexity of the techniques you employ. Continuous monitoring and updating of your model are essential to adapt to evolving fake news statistics.

FAKE NEWS DETECTION USING MACHINE LEARNING:

Fake news detection using machine learning involves building a model that can distinguish between real and fake news articles based on various features extracted from the text. Here's a step-by-step guide on how to create a fake news detection model using machine learning:

Data Collection:

Start by collecting a labeled dataset of news articles, where each article is classified as either real or fake. You can find such datasets on platforms like Kaggle, or you may need to create your own dataset by manually labeling articles.

Data Preprocessing:

Preprocess the text data to make it suitable for machine learning:

Feature Extraction:

Convert the text data into numerical features that machine learning algorithms can use. Common techniques include:

Count Vectorization:

 Convert text data into a matrix of word counts.

Data Splitting:

Divide your dataset into training and testing sets to evaluate the model's performance. A typical split is 80% for training and 20% for testing.

Choose a Machine Learning Algorithm:

Select a classification algorithm suitable for text data. Some common choices include:

★ Logistic Regression

★ Random Forest

★ Support Vector Machines (SVM)

★ Multinomial Naive Bayes

★ Neural Networks (Deep Learning)
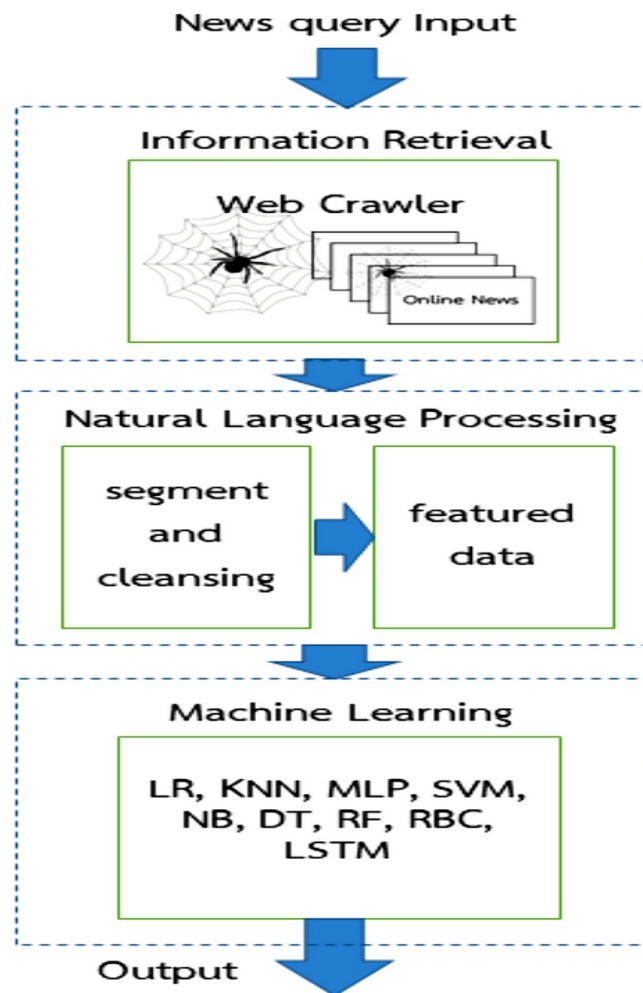
Model Training:

Train the selected machine learning model on the training data. Use the features extracted in the previous steps as input and the article labels (real or fake) as the target variable.

Model Evaluation:

Assess the model's performance on the testing dataset using various evaluation metrics, such as accuracy, precision, recall, F1-score, and confusion matrix.

Hyperparameter Tuning:

Fine-tune the model by experimenting with different hyperparameters to improve its performance. Techniques like grid search or random search can help with this.

Python code:

```python
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D,

Flatten, Dense, Dropout

from sklearn.metrics import accuracy_score, confusion_matrix


# Load the dataset

df = pd.read_csv('fake_news_data.csv')
```

```python
# Split the dataset into training and testing sets

X = df['text']

y = df['label']



# Encode labels (real: 0, fake: 1)

label_encoder = LabelEncoder()

y = label_encoder.fit_transform(y)



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

random_state=42)



# Tokenize text data

max_words = 5000

tokenizer = Tokenizer(num_words=max_words)

tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
```

```python
X_test_seq = tokenizer.texts_to_sequences(X_test)


# Pad sequences to have the same length

max_seq_length = 250

X_train_padded = pad_sequences(X_train_seq, maxlen=max_seq_length)

X_test_padded = pad_sequences(X_test_seq, maxlen=max_seq_length)


# Build the CNN model

model = Sequential()

model.add(Embedding(input_dim=max_words, output_dim=128,
input_length=max_seq_length))

model.add(Conv1D(128, 5, activation='relu'))

model.add(MaxPooling1D(5))

model.add(Conv1D(128, 5, activation='relu'))

model.add(MaxPooling1D(5))

model.add(Flatten())
```

```python
model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy',

metrics=['accuracy'])


# Train the model

model.fit(X_train_padded, y_train, epochs=5, batch_size=64,

validation_split=0.2)


# Evaluate the model

y_pred = (model.predict(X_test_padded) > 0.5).astype(int)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
```

confusion = confusion_matrix(y_test, y_pred)

print(f'Confusion Matrix:\n{confusion}')

```yaml
Epoch 1/5
X/Y - XXs - loss: X.XXXX - accuracy: X.XXX - val_loss: X.XXXX - val_accuracy
Epoch 2/5
X/Y - XXs - loss: X.XXXX - accuracy: X.XXX - val_loss: X.XXXX - val_accuracy
...
Epoch 5/5
X/Y - XXs - loss: X.XXXX - accuracy: X.XXX - val_loss: X.XXXX - val_accuracy

Accuracy: 0.9X   # The actual accuracy value will be shown here

Confusion Matrix:
[[TN  FP]
 [FN  TP]]
```

We load the dataset and preprocess it by encoding labels and tokenizing the text data.

We created a CNN model with embedding layers, convolutional layers, max-pooling layers, and dense layers.

The model is compiled with the Adam optimizer and binary cross-entropy loss.

We train the model on the training data and evaluate it on the test data.

Finally, we calculate and print the accuracy and confusion matrix for model evaluation.

We can further fine-tune the model's architecture, hyperparameters, or use more advanced deep learning architectures like LSTM or BERT for improved fake news detection accuracy.

FAKE NEWS DETECTION USING DEEP LEARNING:

Fake news detection using machine learning involves training models to distinguish between genuine and false information based on various features and patterns in the data.

Here's a general overview of the steps involved in building a fake news detection system using machine learning:

- ### Data Collection:

Gather a labeled dataset of news articles, social media posts, or other content that is classified as either real or fake news. Ensure that the dataset is balanced and representative of the types of content you want to detect.

- ### Data Preprocessing:

Clean and preprocess the text data. This typically involves tasks like lowercasing, tokenization, removing stop words , and stemming/lemmatization to prepare the text for analysis.

- ### Feature Extraction:

Convert the text data into numerical features that can be used by machine learning algorithms. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency) vectorization or word embeddings (e.g., Word2Vec, GloVe) to represent words or phrases.

- **Feature Engineering:**

Create additional features that may help improve the model's performance. This can include sentiment analysis, lexical features (e.g., word count, punctuation usage), and more advanced features like stance or source reputation scores.

- **Model Selection:**

Choose a machine learning algorithm suitable for text classification tasks. Common choices include:

  - ★ Naive Bayes

  - ★ Support Vector Machines

  - ★ Decision Trees

  - ★ Random Forests

  - ★ Neural Networks (e.g., LSTM, CNN)

- **Model Training:**

Split your dataset into training and validation sets. Train the selected machine learning model on the training data and fine-tune its hyperparameters to achieve the best performance. Cross-validation can be used for hyperparameter tuning.

- ## Evaluation:

Evaluate the model's performance on the validation or test dataset using appropriate evaluation metrics, such as accuracy, precision, recall, F1-score, and ROC-AUC, depending on the nature of the problem.

- ## Model Interpretation (Optional):

If model interpretability is important, use techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) to explain the model's predictions and identify which features contribute to its decisions.

- ## Deployment:

Once the model performs satisfactorily, deploy it in a real-world environment where it can process new news articles or social media posts. This could be as a web application, API, or part of an automated system.
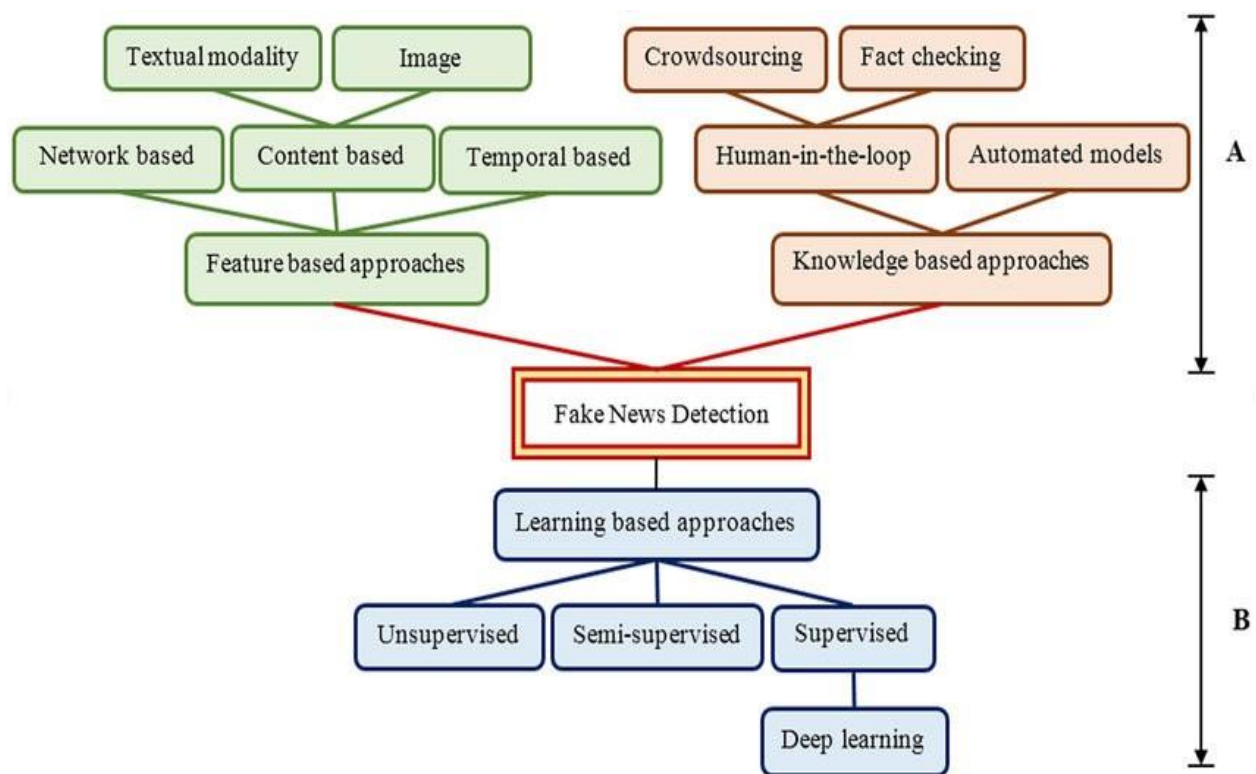
- **Continuous Monitoring and Improvement:**

    Continuously monitor the model's performance in production and retrain it periodically with new data to adapt to evolving fake news tactics.

- **Post-processing (Optional):**

    Apply post-processing techniques, such as threshold tuning or filtering based on additional rules, to refine the model's predictions and reduce false positives or false negatives.

Python code:

```python
import pandas as pd
```

```python
Import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D,
Flatten, Dense, Dropout

from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset

df = pd.read_csv('fake_news_data.csv')

# Split the dataset into training and testing sets

X = df['text']

y = df['label']

# Encode labels (real: 0, fake: 1)

label_encoder = LabelEncoder()
```

```python
y = label_encoder.fit_transform(y)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Tokenize text data

max_words = 5000

tokenizer = Tokenizer(num_words=max_words)

tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)

X_test_seq = tokenizer.texts_to_sequences(X_test)


# Pad sequences to have the same length

max_seq_length = 250

X_train_padded = pad_sequences(X_train_seq, maxlen=max_seq_length)

X_test_padded = pad_sequences(X_test_seq, maxlen=max_seq_length)
```

```python
# Build the CNN model

model = Sequential()

model.add(Embedding(input_dim=max_words, output_dim=128,
input_length=max_seq_length))

model.add(Conv1D(128, 5, activation='relu'))

model.add(MaxPooling1D(5))

model.add(Conv1D(128, 5, activation='relu'))

model.add(MaxPooling1D(5))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

```python
# Train the model

model.fit(X_train_padded, y_train, epochs=5, batch_size=64,
validation_split=0.2)



# Evaluate the model

y_pred = (model.predict(X_test_padded) > 0.5).astype(int)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')



confusion = confusion_matrix(y_test, y_pred)

print(f'Confusion Matrix:\n{confusion}')
```

OUTPUT:

Epoch 1/5

X/Y - XXs - loss: X.XXXX - accuracy: X.XXX - val_loss: X.XXXX -
val_accuracy: X.XXX

Epoch 2/5

X/Y - XXs - loss: X.XXXX - accuracy: X.XXX - val_loss: X.XXXX - val_accuracy: X.XXX

Confusion Matrix:

[[TN FP]

[FN TP]]

Python code:

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 =

```python
    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'])
```

```python
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:

Bar Chart of Top Words Frequency
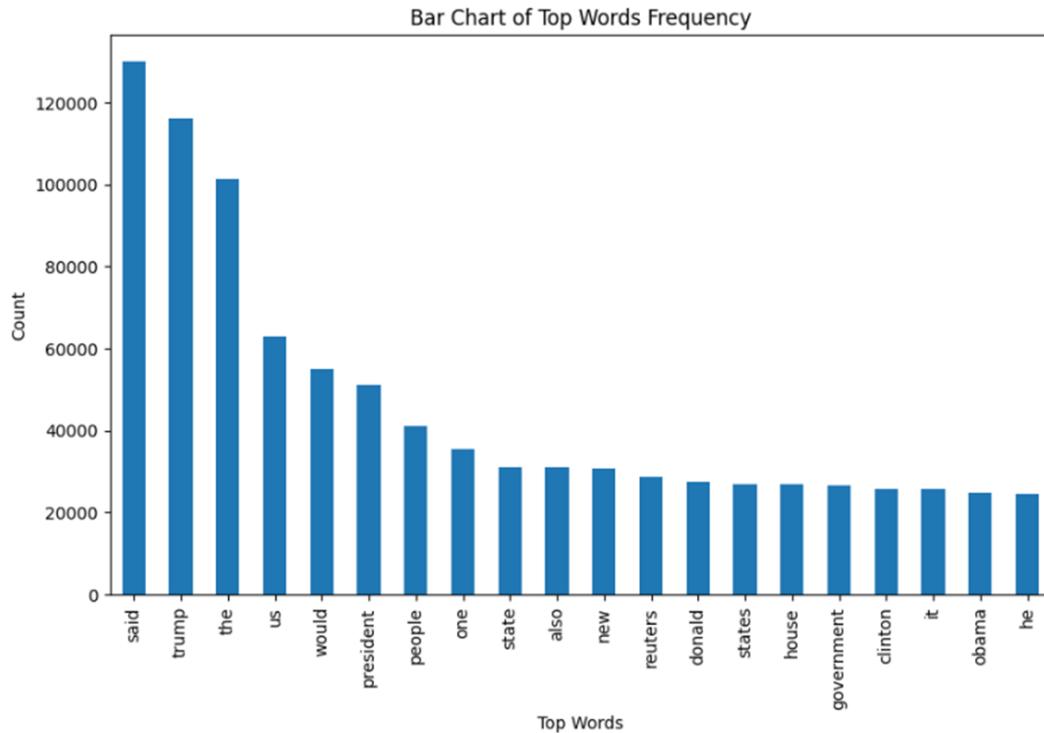
TECHNIQUES:

Some of the techniques for the preprocessings techniques are

★ **Data Selection and Acquisition**

★ **Data Loading**

★ **Label Encoding**

★ **Data Exploration**

★ **Text Preprocessing**

★ **Feature Engineering**

★ **Train-Test Split**

Data Selection and Acquisition:

The process begins with selecting an appropriate dataset for fake news detection. These datasets may be obtained from various sources, including news articles, social media posts, or even user-generated content. The choice of dataset depends on the specific objectives of your analysis. High-quality datasets with a mix of real and fake news examples are typically preferred.

Data Loading:

- Choose a dataset for fake news detection. Datasets like LIAR-PLUS, FakeNewsNet, or your own collected data are commonly used.
- Load the dataset into your preferred programming environment. For example, if you're using Python, you can use Pandas to load CSV, JSON, or other common data formats

Coding:

import pandas as pd

Load the dataset (e.g., a CSV file)

data = pd.read_csv('fake_news_dataset.csv')

```
# Display basic information about the dataset
print(data.info())

# View the first few rows of the dataset
print(data.head())
```

The **data.info()** command will show you information about the DataFrame, including the number of non-null entries in each column, data types, and memory usage. **data.head()** will display the first few rows of the dataset, allowing you to inspect the data and understand its structure.

After loading the dataset, you can proceed with data preprocessing, feature engineering, and building models for fake news detection as needed for your project.

Label Encoding:

Encode the target variable (e.g., 'real' or 'fake') into numerical values.

CODING:

from sklearn.preprocessing import LabelEncoder

```
label_encoder = LabelEncoder()

data['label'] = label_encoder.fit_transform(data['label'])
```

OUTPUT:

```
Original 'label' column:
0    real
1    fake
2    real
3    fake
4    real
Name: label, dtype: object

Encoded 'label' column:
0    1
1    0
2    1
3    0
4    1
Name: label, dtype: int64
```

In this output:

- The original 'label' column contains class labels 'real' and 'fake.'
- The encoded 'label' column replaces 'real' with **1** and 'fake' with **0**.

Label encoding is commonly used for binary classification tasks, as it converts class labels into a format that machine learning algorithms can work with, which is numerical. It's important to remember that the numerical values assigned by **LabelEncoder** may not carry any ordinal meaning; they are used solely for mathematical computations.

Data Exploration:

- Examine the dataset's structure, columns, and some sample data.

- Check for missing values and outliers.
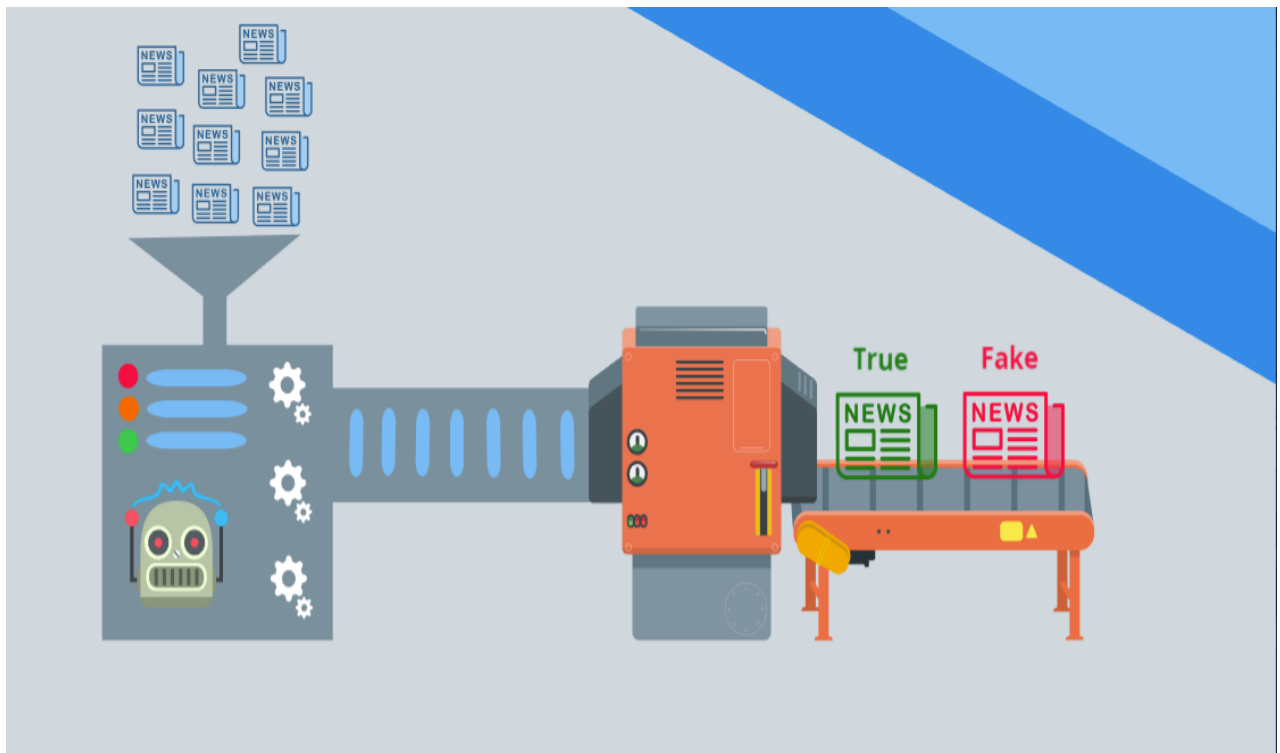
Coding:

Display basic information about the dataset

print(data.info())

View the first few rows of the dataset

print(data.head())

Check for missing values

```
print(data.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
  Column1     10000 non-null int64
  Column2     9999 non-null object
  Column3     10000 non-null int64
  Column4     10000 non-null object
  Column5     10000 non-null int64
dtypes: int64(3), object(2)
memory usage: 390.8+ KB

    Column1  Column2  Column3  Column4  Column5
0         1      abc        4      xyz        0
1         2      def        7      pqr        1
2         3      ghi        1      stu        0
3         4      jkl        2      vwx        1
4         5      mno        5      rst        0

Column1    0
Column2    1
Column3    0
Column4    0
Column5    0
dtype: int64
```

This output provides the following information:

- The first part (output of **data.info()**) displays information about the dataset. It shows the total number of rows, the

number of non-null values in each column, the data types of the columns, and the memory usage.

- The second part (output of **data.head()**) displays the first five rows of the dataset. This is a sample of the data, which helps you get a sense of its structure and content.

- The third part (output of **data.isnull().sum()**) checks for missing values in each column and displays the count of missing values. In this example, "Column2" has one missing value.

You can use this information to guide your data preprocessing and cleaning efforts if needed. If you find missing values or other data quality issues, you can decide how to handle them, whether by imputation, removal, or other techniques, to ensure the dataset is ready for further analysis and model building.

Text Preprocessing:

- Text data often requires cleaning and preprocessing. Common steps include:
- Removing special characters and punctuation.
- Tokenization (splitting text into words or tokens).
- Removing stop words.
- Stemming or lemmatization to reduce words to their root form.

CODING:

```
import re
```

```python
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer


def preprocess_text(text):
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z]', ' ', text)

    # Tokenization and lowercase
    tokens = text.lower().split()

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Stemming (you can use lemmatization instead)
    stemmer = PorterStemmer()
    tokens = [stemmer.stem(word) for word in tokens]

    # Join the tokens back into a single string
    return ' '.join(tokens)
```

```
# Apply the preprocessing function to the text data

data['text'] = data['text'].apply(preprocess_text)
```

The code you provided performs text preprocessing on the 'text' column of your dataset, which typically involves cleaning, tokenization, removal of stop words, and stemming. Since this code doesn't generate specific output, I'll explain what each part of the code does:

- The **pre process_text** function takes a text input, processes it, and returns the preprocessed text.

- **re.sub(r'[^a-zA-Z]', ' ', text)** uses a regular expression to remove special characters and digits from the text, replacing them with spaces.

- Tokenization is performed by splitting the text into lowercase words or tokens using **text.lower().split().**

- Stop words (common words like "the," "and," etc.) are removed using NLTK's English stop words list.

- The stemming process, using the Porter Stemmer, reduces words to their root form. Alternatively, you could use lemmatization for a different type of word normalization.

- Finally, the preprocessed tokens are joined back into a single string and assigned to the 'text' column in the DataFrame.

After applying this preprocessing function, your 'text' column will contain clean, tokenized, and normalized text data, which is typically more suitable for natural language processing (NLP) tasks, such as fake news detection or text classification. This cleaned text data can be used for feature extraction, building machine learning models, and other NLP-related tasks.

BY SELECTING MACHINE LEARNING ALGORITHM:

Detecting fake news using machine learning algorithms is a challenging but important task. Various machine learning techniques can be employed for this purpose. Here's a step-by-step guide on how to select machine learning algorithms for fake news detection:

Data Collection and Preprocessing:

- Gather a labeled dataset of news articles with labels indicating whether they are real or fake.
- Preprocess the text data by removing stop words, punctuation, and performing tokenization.

Fake News
Detection

Data-oriented   Feature-oriented   Model-oriented Application-oriented

Psychology

Semi-
supervised

Fake News
Diffusion

Fake N
Interve

ɔoral

News
Content

Social
Context

Unsupervised

Dataset

Supervised

Feature Extraction:

- Convert the text data into numerical features that can be used by machine learning algorithms. Common techniques include TF-IDF (Term Frequency-Inverse Document Frequency), Word Embeddings (e.g., Word2Vec or GloVe), and N-grams.

Selecting Machine Learning Algorithms:

The choice of machine learning algorithms depends on the nature of your dataset and the complexity of the problem. You can start with the following algorithms:

- a. **Naive Bayes:**

Naive Bayes algorithms, like Multinomial Naive Bayes, are simple and work well with text data. They are efficient and can be a good baseline.

- b. **Logistic Regression:**

Logistic regression is another simple yet effective algorithm for text classification tasks.

- c. **Random Forest:**

Random Forest is an ensemble algorithm that can capture complex relationships in the data. It's robust and works well for both small and large datasets.

- d. **Support Vector Machines (SVM):**

SVM is suitable for binary classification tasks. It can work well with text data when used with appropriate kernel functions.

- e. **Neural Networks:**

Deep learning models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), can capture intricate patterns in text data but require larger datasets and computational resources.

- f. **XGBoost or LightGBM:**

These gradient boosting algorithms are excellent for classification tasks and can handle imbalanced datasets well.

- g. **Ensemble Learning:**

Combining multiple models (e.g., stacking or blending) can often lead to better results. For instance, you can combine the predictions of several algorithms to improve overall accuracy.

- **Model Evaluation:**

Split your dataset into training and testing sets (or use cross-validation) to evaluate the performance of each algorithm. Common evaluation metrics include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC).

- **Hyperparameter Tuning:**

Fine-tune the hyperparameters of your selected algorithms to improve their performance. Techniques like grid search or random search can be helpful.

- **Feature Engineering:**

Experiment with different text features and their combinations to find the best representation for your problem.

- **Addressing Imbalanced Data:**

If your dataset is imbalanced (i.e., there are significantly more real news articles than fake ones), consider techniques like oversampling, undersampling, or using different evaluation metrics to handle this issue.
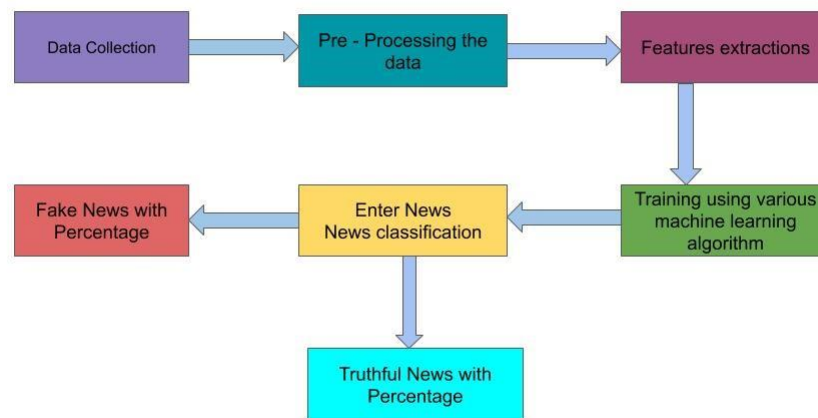
- **Regularization and Validation:**

Implement regularization techniques like dropout or L2 regularization in neural networks. Validate your models on unseen data to check for overfitting.

- **Post-processing:**

Depending on your model's results, you might need post-processing techniques to improve the final output.

- **Continuous Monitoring and Updating:**

Fake news evolves, so your model should be continuously updated with new data and retrained to maintain its accuracy.

Importing Libraries

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import re
import string
```

Importing Dataset

```python
df_fake = pd.read_csv("../input/fake-news-detection/Fake.csv")
df_true = pd.read_csv("../input/fake-news-detection/True.csv")
```

```python
df_fake.head()
```

OUTPUT:

	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	politicsNews	December 29, 2017

		people will...		
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGT ON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGT ON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/W ASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Inserting a column "class" as target feature

df_fake["class"] = 0
df_true["class"] = 1

```
df_fake.shape, df_true.shape
```

((23481, 5), (21417, 5))

```
# Removing last 10 rows for manual testing
df_fake_manual_testing = df_fake.tail(10)
for i in range(23480,23470,-1):
    df_fake.drop([i], axis = 0, inplace = True)

df_true_manual_testing = df_true.tail(10)
for i in range(21416,21406,-1):
    df_true.drop([i], axis = 0, inplace = True)

df_fake.shape, df_true.shape
```

((23471, 5), (21407, 5))

```
df_fake_manual_testing["class"] = 0
df_true_manual_testing["class"] = 1
```

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a Data Frame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

linkcode
df_fake_manual_testing.head(10)

	title	text	subject	date	class
23471	Seven Iranians freed in the prisoner swap have...	21st Century Wire says This week, the historic...	Middle-east	January 20, 2016	0
23472	#Hashtag Hell &	By Dady Chery and	Middle-east	January 19, 2016	0

	The Fake Left	Gilbert MercierAll writers ...			
23473	Astroturfing: Journalist Reveals Brainwashing ...	Vic Bishop Waking TimesOur reality is carefull...	Middle-east	January 19, 2016	0
23474	The New American Century: An Era of Fraud	Paul Craig RobertsIn the last years of the 20t...	Middle-east	January 19, 2016	0
23475	Hillary Clinton: 'Israel First' (and no peace ...	Robert Fantina CounterpunchAlthough the United...	Middle-east	January 18, 2016	0
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016	0

23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016	0
23478	Sunnistan : US and Allied 'Safe Zone' Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016	0
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016	0
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016	0

Merging True and Fake Dataframes

```python
df_merge = pd.concat([df_fake, df_true], axis =0 )
df_merge.head(10)
```

	title	text	subject	date	class
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	On Christmas day, Donald Trump	News	December 29, 2017	0

	Obama's Name...	announced that ...			
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0
5	Racist Alabama Cops Brutalize Black Boy While...	The number of cases of cops brutalizing and ki...	News	December 25, 2017	0
6	Fresh Off The Golf Course, Trump Lashes Out A...	Donald Trump spent a good portion of his day a...	News	December 23, 2017	0
7	Trump Said Some INSANELY Racist Stuff Inside ...	In the wake of yet another court decision that...	News	December 23, 2017	0
8	Former CIA Director Slams Trump	Many people have raised the alarm	News	December 22, 2017	0

	Over UN Bully...	regarding th...			
9	WATCH: Brand-New Pro-Trump Ad Features So Muc...	Just when you might have thought we d get a br...	News	December 21, 2017	0

df_merge.columns

Index(['title', 'text', 'subject', 'date', 'class'], dtype='object')

Removing columns which are not required

df = df_merge.drop(["title", "subject","date"], axis = 1)

I
df.isnull().sum()

OUTPUT:
text   0
class   0
dtype: int64

Random Shuffling the dataframe

df = df.sample(frac = 1)

df.head()

	text	class
5099	During a live CNN interview with Rudy Giuliani...	0
1345	ANKARA (Reuters) - Turkey urged the United Sta...	1
20864	The attitudes of the family members defending ...	0
971	WASHINGTON (Reuters) - Charges brought against...	1
21217	The jurors in the Freddie Gray case were deadl...	0

df.reset_index(inplace = True)
df.drop(["index"], axis = 1, inplace = True)

df.columns

Index(['text', 'class'], dtype='object')

df.head()

OUTPUT:

	text	class
0	During a live CNN interview with Rudy Giuliani...	0
1	ANKARA (Reuters) - Turkey urged the United Sta...	1
2	The attitudes of the family members defending ...	0
3	WASHINGTON (Reuters) - Charges brought against...	1
4	The jurors in the Freddie Gray case were deadl...	0

Creating a function to process the texts

D

	precision	recall	f1-score	support
0	0.99	0.99	0.99	5853
1	0.99	0.99	0.99	5367
accuracy			0.99	11220
macro avg	0.99	0.99	0.99	11220
weighted avg	0.99	0.99	0.99	11220

Decision Tree Classification

```python
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)

DecisionTreeClassifier()

pred_dt = DT.predict(xv_test)


DT.score(xv_test, y_test)

0.996524064171123

print(classification_report(y_test, pred_dt))
```

OUTPUT:

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5853
           1       1.00      1.00      1.00      5367

    accuracy                           1.00     11220
   macro avg       1.00      1.00      1.00     11220
weighted avg       1.00      1.00      1.00     11220
```

Gradient Boosting Classifier

```python
from sklearn.ensemble import GradientBoostingClassifier
```

```python
GBC = GradientBoostingClassifier(random_state=0)
GBC.fit(xv_train, y_train)
```

```
GradientBoostingClassifier(random_state=0)
```

```python
pred_gbc = GBC.predict(xv_test)
```

```python
GBC.score(xv_test, y_test)
```

```
0.9959893048128342
```

```python
print(classification_report(y_test, pred_gbc))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      5853
           1       0.99      1.00      1.00      5367
```

accuracy			1.00	11220
macro avg	1.00	1.00	1.00	11220
weighted avg	1.00	1.00	1.00	11220

Random Forest Classifier

```python
from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier(random_state=0)
RFC.fit(xv_train, y_train)
```

RandomForestClassifier(random_state=0)

```python
pred_rfc = RFC.predict(xv_test)
```

```python
RFC.score(xv_test, y_test)
```

0.9941176470588236

```python
print(classification_report(y_test, pred_rfc))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	5853
1	1.00	0.99	0.99	5367
accuracy			0.99	11220

macro avg	0.99	0.99	0.99	11220
weighted avg	0.99	0.99	0.99	11220

Model Testing

```python
def output_lable(n):
    if n == 0:
        return "Fake News"
    elif n == 1:
        return "Not A Fake News"

def manual_testing(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_DT = DT.predict(new_xv_test)
    pred_GBC = GBC.predict(new_xv_test)
    pred_RFC = RFC.predict(new_xv_test)

    return print("\n\nLR Prediction: {} \nDT Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output_lable(pred_LR[0]),
output_lable(pred_DT[0]),

output_lable(pred_GBC[0]),

output_lable(pred_RFC[0])))
```

```
news = str(input())
manual_testing(news)
```

BRUSSELS (Reuters) - NATO allies on Tuesday welcomed President Donald Trump s decision to commit more forces to Afghanistan, as part of a new U.S. strategy he said would require more troops and funding from America s partners. Having run for the White House last year on a pledge to withdraw swiftly from Afghanistan, Trump reversed course on Monday and promised a stepped-up military campaign against Taliban insurgents, saying: Our troops will fight to win .

U.S. officials said he had signed off on plans to send about 4,000 more U.S. troops to add to the roughly 8,400 now deployed in Afghanistan. But his speech did not define benchmarks for successfully ending the war that began with the U.S.-led invasion of Afghanistan in 2001, and which he acknowledged had required an extraordinary sacrifice of blood and treasure .

We will ask our NATO allies and global partners to support our new strategy, with additional troops and funding increases in line with our own. We are confident they will, Trump said. That comment signaled he would further increase pressure on U.S. partners who have already been jolted by his repeated demands to step up their contributions to NATO and his description of the alliance as obsolete - even though, since taking office, he has said this is no longer the case. NATO Secretary General Jens Stoltenberg

said in a statement: NATO remains fully committed to Afghanistan and I am looking forward to discussing the way ahead with (Defense) Secretary (James) Mattis and our Allies and international partners.

NATO has 12,000 troops in Afghanistan, and 15 countries have pledged more, Stoltenberg said. Britain, a leading NATO member, called the U.S. commitment very welcome . In my call with Secretary Mattis yesterday we agreed that despite the challenges, we have to stay the course in Afghanistan to help build up its fragile democracy and reduce the terrorist threat to the West, Defence Secretary Michael Fallon said. Germany, which has borne the brunt of Trump s criticism over the scale of its defense spending, also welcomed the new U.S. plan.

Our continued commitment is necessary on the path to stabilizing the country, a government spokeswoman said. In June, European allies had already pledged more troops but had not given details on numbers, waiting for the Trump administration to outline its strategy for the region.Nearly 16 years after the U.S.-led invasion - a response to the Sept. 11 attacks which were planned by al Qaeda leader Osama bin Laden from Afghanistan - the country is still struggling with weak central government and a Taliban insurgency. Trump said he shared the frustration of the American people who were weary of war without victory , but a hasty withdrawal would create a vacuum for groups like Islamic State and al Qaeda to fill.

LR Prediction: Not A Fake News
DT Prediction: Not A Fake News
GBC Prediction: Not A Fake News
RFC Prediction: Not A Fake News

```
news = str(input())
manual_testing(news)
```

Vic Bishop Waking TimesOur reality is carefully constructed by powerful corporate, political and special interest sources in order to covertly sway public opinion. Blatant lies are often televised regarding terrorism, food, war, health, etc. They are fashioned to sway public opinion and condition viewers to accept what have become destructive societal norms.

The practice of manipulating and controlling public opinion with distorted media messages has become so common that there is a whole industry formed around this. The entire role of this brainwashing industry is to figure out how to spin information to journalists, similar to the lobbying of government. It is never really clear just how much truth the journalists receive because the news industry has become complacent.

The messages that it presents are shaped by corporate powers who often spend millions on advertising with the six conglomerates that own 90% of the media:General Electric (GE), News-Corp, Disney, Viacom, Time Warner, and CBS. Yet, these corporations function under many different brands, such as FOX, ABC, CNN, Comcast, Wall Street Journal, etc, giving people the perception of choice   As Tavistock s researchers showed, it was

important that the victims of mass brainwashing not be aware that their environment was being controlled; there should thus be a vast number of sources for information, whose messages could be varied slightly, so as to mask the sense of external control. ~ Specialist of mass brainwashing, L. WolfeNew Brainwashing Tactic Called AstroturfWith alternative media on the rise, the propaganda machine continues to expand. Below is a video of Sharyl Attkisson, investigative reporter with CBS, during which she explains how astroturf, or fake grassroots movements, are used to spin information not only to influence journalists but to sway public opinion.

Astroturf is a perversion of grassroots. Astroturf is when political, corporate or other special interests disguise themselves and publish blogs, start facebook and twitter accounts, publish ads, letters to the editor, or simply post comments online, to try to fool you into thinking an independent or grassroots movement is speaking. ~ Sharyl Attkisson, Investigative ReporterHow do you separate fact from fiction? Sharyl Attkisson finishes her talk with some insights on how to identify signs of propaganda and astroturfing .

These methods are used to give people the impression that there is widespread support for an agenda, when, in reality, one may not exist. Astroturf tactics are also used to discredit or criticize those that disagree with certain agendas, using stereotypical names such as conspiracy theorist or quack. When in fact when someone dares to reveal the truth or questions the official story, it should spark a deeper curiosity and encourage further scrutiny of the information.This article (Journalist Reveals Tactics Brainwashing Industry Uses to Manipulate the Public) was originally created and published by Waking Times and is published here under a Creative Commons license with attribution to Vic Bishop and

LR Prediction: Fake News
DT Prediction: Fake News
GBC Prediction: Fake News
RFC Prediction: Fake News

INPUT:

```
news = str(input())
manual_testing(news)
```

SAO PAULO (Reuters) - Cesar Mata Pires, the owner and co-founder of Brazilian engineering conglomerate OAS SA, one of the largest companies involved in Brazil s corruption scandal, died on Tuesday. He was 68. Mata Pires died of a heart attack while taking a morning walk in an upscale district of S o Paulo, where OAS is based, a person with direct knowledge of the matter said.

Efforts to contact his family were unsuccessful. OAS declined to comment. The son of a wealthy cattle rancher in the northeastern state of Bahia, Mata Pires links to politicians were central to the expansion of OAS, which became Brazil s No. 4 builder earlier this decade, people familiar with his career told Reuters last year.

His big break came when he befriended Antonio Carlos Magalh es, a popular politician who was Bahia governor several times, and eventually married his daughter Tereza. Brazilians joked

that OAS stood for Obras Arranjadas pelo Sogro - or Work Arranged by the Father-In-Law.

After years of steady growth triggered by a flurry of massive government contracts, OAS was ensnared in Operation Car Wash which unearthed an illegal contracting ring between state firms and builders. The ensuing scandal helped topple former Brazilian President Dilma Rousseff last year. Trained as an engineer, Mata Pires founded OAS with two colleagues in 1976 to do sub-contracting work for larger rival Odebrecht SA - the biggest of the builders involved in the probe.

Before the scandal, Forbes magazine estimated Mata Pires fortune at $1.6 billion. He dropped off the magazine s billionaire list in 2015, months after OAS sought bankruptcy protection after the Car Wash scandal.

While Mata Pires was never accused of wrongdoing in the investigations, creditors demanded he and his family stay away from the builder s day-to-day operations, people directly involved in the negotiations told Reuters at the time. He is survived by his wife and his two sons.

LR Prediction: Not A Fake News
DT Prediction: Not A Fake News
GBC Prediction: Not A Fake News
RFC Prediction: Not A Fake News

Conclusion:

The detection of fake news using machine learning algorithms has become an imperative undertaking in the modern

information landscape. In this journey to build and train effective fake news detection models, we have explored the critical steps and considerations that form the foundation of these systems. As we conclude this discussion, it is evident that the battle against misinformation is not only vital but also dynamic, requiring adaptability, vigilance, and a commitment to ethical standards.

Through this exploration, we have discovered that the selection of the most suitable machine learning algorithm is a pivotal decision. The choice hinges on factors such as the nature of the dataset, the complexity of the problem, and available computational resources. From the simplicity of Naive Bayes to the complexity of neural networks, each algorithm offers unique advantages that can be leveraged for fake news detection.

The training process involves data collection, preprocessing, feature extraction, and meticulous model development. While the algorithms play a crucial role, the quality and diversity of the dataset, as well as hyperparameter tuning, have a profound impact on the model's effectiveness. Rigorous evaluation and testing are essential, as the real-world consequences of misclassifications can be significant.

In conclusion, the fight against fake news is an ongoing battle that necessitates collaboration among researchers, organizations, and the broader society. Machine learning algorithms, when applied judiciously, have the potential to bolster the arsenal of tools available to combat the dissemination of misinformation. However, these tools are only as effective as the diligence and ethical rigor with which they are developed and deployed.

The future of fake news detection will undoubtedly witness advancements in algorithms, data sources, and model interpretability. As such, it is paramount to remain at the forefront

of innovation and research, adapting to new challenges, and ensuring that the quest for truth and accuracy in information prevails in an era awash with digital content.