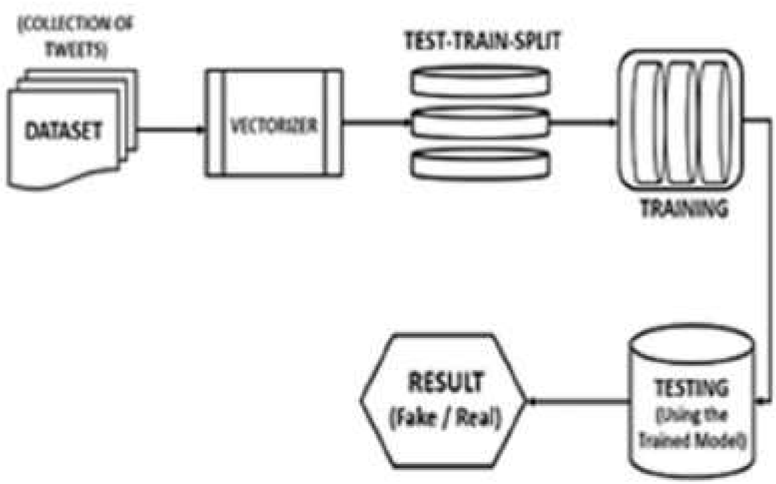
FAKE NEWS DETECTION USING NLP

Technology: ARTIFICIAL INTELLIGENCE

NAME: HARRIS JOSHUA S
NM ID: au311121205022
REG. NO:311121205022

PHASE 4 - PROJECT SUBMISSION

Phase 4: Development Part 2



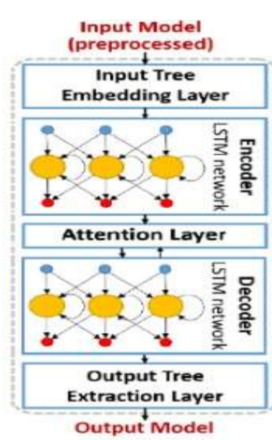


Figure 1. Generic LSTM architecture.

PHASE 3 GUIDELINES:(GIVEN)

Phase 4: Development Part 2
In this part you will continue building your project.
Continue building the fake news detection model by applying NLP techniques and training a classification model.
Text Preprocessing and Feature Extraction
Model training and evaluation

INTRODUCTION:

In phase 4 of this project, we continue with the further development of our fake news detection project. In the last phase of development we focused on loading the dataset and preprocessing the textual dataset. In this phase, we will proceed further with processes like: **Text Preprocessing, Feature Extraction, Applying NLP techniques and training a classification model, Model training and evaluation.**

Let us begin with the project first by seeing the steps we have already seen before moving on to the rest of the steps.

STEP 1: IMPORTING THE LIBRARIES.

AI_Phase4.ipynb

```
# Import necessary libraries for the project
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
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
nltk.download('punkt')
```

Here, we import the necessary libraries and modules for the project. Let's break down their roles:

- numpy and pandas are fundamental for data manipulation and numerical operations.
- matplotlib and seaborn are used for data visualization, providing tools to create plots and graphs.
- nltk (Natural Language Toolkit) is a library for natural language processing (NLP) tasks, including text processing.
- nltk.corpus is used to access stopwords, which are common words to be excluded in text preprocessing.
- TfidfVectorizer from sklearn.feature_extraction.text is a tool to convert text data into numerical features using TF-IDF.
- train_test_split from sklearn.model_selection is used to split the data into training and testing sets.
- MultinomialNB from sklearn.naive_bayes is a Naive Bayes classifier suitable for text classification.
- accuracy_score, confusion_matrix, and classification_report from sklearn.metrics are for model evaluation.
- nltk.download('punkt'): This command downloads the 'punkt' resource for NLTK. The 'punkt' resource includes data files used for tokenization, which is the process of splitting text into individual words or tokens.

STEP 2: DATA LOADING:

AI_Phase4.ipynb

```
# Load the dataset from Kaggle
fake_news = pd.read_csv("/Fake.csv")
real_news = pd.read_csv("/True.csv")
```

DATASETS:

True.csv

	A	B	C	D
1	title	text	subject	date
2	As U.S. budget fight looms, Republicans flip their fiscal script	WASHINGTON (Reuters) - The head of a conservative Republican faction in the U.S. Cong	politicsNews	December 31, 2017
3	U.S. military to accept transgender recruits on Monday: Pentagon	WASHINGTON (Reuters) - Transgender people will be allowed for the first time to enlist i	politicsNews	December 29, 2017
4	Senior U.S. Republican senator: 'Let Mr. Mueller do his job'	WASHINGTON (Reuters) - The special counsel investigation of links between Russia and F	politicsNews	December 31, 2017
5	FBI Russia probe helped by Australian diplomat tip-off: NYT	WASHINGTON (Reuters) - Trump campaign adviser George Papadopoulos told an Austral	politicsNews	December 30, 2017
6	Trump wants Postal Service to charge 'much more' for Amazon shipments	SEATTLE/WASHINGTON (Reuters) - President Donald Trump called on the U.S. Postal Ser	politicsNews	December 29, 2017
7	White House, Congress prepare for talks on spending, immigration	WEST PALM BEACH, Fla./WASHINGTON (Reuters) - The White House said on Friday it wa	politicsNews	December 29, 2017
8	Trump says Russia probe will be fair, but timeline unclear: NYT	WEST PALM BEACH, Fla (Reuters) - President Donald Trump said on Thursday he believes	politicsNews	December 29, 2017
9	Factbox: Trump on Twitter (Dec 29) - Approval rating, Amazon	The following statementsÂ were posted to the verified Twitter accounts of U.S. Presiden	politicsNews	December 29, 2017
10	Trump on Twitter (Dec 28) - Global Warming	The following statementsÂ were posted to the verified Twitter accounts of U.S. Presiden	politicsNews	December 29, 2017
11	Alabama official to certify Senator-elect Jones today despite challenge: CNN	WASHINGTON (Reuters) - Alabama Secretary of State John Merrill said he will certify Der	politicsNews	December 28, 2017
12	Jones certified U.S. Senate winner despite Moore challenge	(Reuters) - Alabama officials on Thursday certified Democrat Doug Jones the winner of tl	politicsNews	December 28, 2017
13	New York governor questions the constitutionality of federal tax overhaul	NEW YORK/WASHINGTON (Reuters) - The new U.S. tax code targets high-tax states and	politicsNews	December 28, 2017
14	Factbox: Trump on Twitter (Dec 28) - Vanity Fair, Hillary Clinton	The following statementsÂ were posted to the verified Twitter accounts of U.S. Presiden	politicsNews	December 28, 2017
15	Trump on Twitter (Dec 27) - Trump, Iraq, Syria	The following statementsÂ were posted to the verified Twitter accounts of U.S. Presiden	politicsNews	December 28, 2017
16	Man says he delivered manure to Mnuchin to protest new U.S. tax law	(In Dec. 25 story, in second paragraph, corrects name of Strongâ€™s employer to Menta	politicsNews	December 25, 2017
17	Virginia officials postpone lottery drawing to decide tied statehouse election	(Reuters) - A lottery drawing to settle a tied Virginia legislative race that could shift the st	politicsNews	December 27, 2017
18	U.S. lawmakers question businessman at 2016 Trump Tower meeting: sources	WASHINGTON (Reuters) - A Georgian-American businessman who met then-Miss Univers	politicsNews	December 27, 2017
19	Trump on Twitter (Dec 26) - Hillary Clinton, Tax Cut Bill	The following statementsÂ were posted to the verified Twitter accounts of U.S. Presiden	politicsNews	December 26, 2017
20	U.S. appeals court rejects challenge to Trump voter fraud panel	(Reuters) - A U.S. appeals court in Washington on Tuesday upheld a lower courtâ€™s dec	politicsNews	December 26, 2017
21	Treasury Secretary Mnuchin was sent gift-wrapped box of horse manure: reports	(Reuters) - A gift-wrapped package addressed to U.S. Treasury Secretary Steven Mnuchin	politicsNews	December 24, 2017
22	Federal judge partially lifts Trump's latest refugee restrictions	WASHINGTON (Reuters) - A federal judge in Seattle partially blocked U.S. President Dona	politicsNews	December 24, 2017
23	Exclusive: U.S. memo weakens guidelines for protecting immigrant children in court	NEW YORK (Reuters) - The U.S. Justice Department has issued new guidelines for immigr	politicsNews	December 23, 2017
24	Trump travel ban should not apply to people with strong U.S. ties: court	(Reuters) - A U.S. appeals court on Friday said President Donald Trumpâ€™s hotly contes	politicsNews	December 23, 2017
25	Second court rejects Trump bid to stop transgender military recruits	WASHINGTON (Reuters) - A federal appeals court in Washington on Friday rejected a bid	politicsNews	December 23, 2017
26	Failed vote to oust president shakes up Peru's politics	LIMA (Reuters) - Peruâ€™s President Pedro Pablo Kuczynski could end up the surprise wir	politicsNews	December 23, 2017
27	Trump signs tax, government spending bills into law	WASHINGTON (Reuters) - U.S. President Donald Trump signed Republicansâ€™ massive $	politicsNews	December 22, 2017

Fake.csv

A1 | title

	A	B	C	D
1	title	text	subject	date
2	Donald Trump Sends Out Embarrassing New Yearâ€™s Eve	Donald Trump just couldn t wish all America	News	December 31, 2017
3	Drunk Bragging Trump Staffer Started Russian Collusion Inv	House Intelligence Committee Chairman De	News	December 31, 2017
4	Sheriff David Clarke Becomes An Internet Joke For Threaten	On Friday, it was revealed that former Milw	News	December 30, 2017
5	Trump Is So Obsessed He Even Has Obamaâ€™s Name Cod	On Christmas day, Donald Trump announce	News	December 29, 2017
6	Pope Francis Just Called Out Donald Trump During His Chris	Pope Francis used his annual Christmas Day	News	December 25, 2017
7	Racist Alabama Cops Brutalize Black Boy While He Is In Han	The number of cases of cops brutalizing and	News	December 25, 2017
8	Fresh Off The Golf Course, Trump Lashes Out At FBI Deputy	Donald Trump spent a good portion of his d	News	December 23, 2017
9	Trump Said Some INSANELY Racist Stuff Inside The Oval Off	In the wake of yet another court decision th	News	December 23, 2017
10	Former CIA Director Slams Trump Over UN Bullying, Openly	Many people have raised the alarm regardir	News	December 22, 2017
11	WATCH: Brand-New Pro-Trump Ad Features So Much A** K	Just when you might have thought we d get	News	December 21, 2017
12	Papa Johnâ€™s Founder Retires, Figures Out Racism Is Bad	A centerpiece of Donald Trump s campaign,	News	December 21, 2017
13	WATCH: Paul Ryan Just Told Us He Doesnâ€™t Care About	Republicans are working overtime trying to	News	December 21, 2017
14	Bad News For Trump â€" Mitch McConnell Says No To Rep	Republicans have had seven years to come	News	December 21, 2017
15	WATCH: Lindsey Graham Trashes Media For Portraying Trun	The media has been talking all day about Tr	News	December 20, 2017
16	Heiress To Disney Empire Knows GOP Scammed Us â€" SHR	Abigail Disney is an heiress with brass ovarie	News	December 20, 2017
17	Tone Deaf Trump: Congrats Rep. Scalise On Losing Weight /	Donald Trump just signed the GOP tax scam	News	December 20, 2017
18	The Internet Brutally Mocks Disneyâ€™s New Trump Robot	A new animatronic figure in the Hall of Presi	News	December 19, 2017
19	Mueller Spokesman Just F-cked Up Donald Trumpâ€™s Chri	Trump supporters and the so-called preside	News	December 17, 2017
20	SNL Hilariously Mocks Accused Child Molester Roy Moore F	Right now, the whole world is looking at the	News	December 17, 2017
21	Republican Senator Gets Dragged For Going After Robert M	Senate Majority Whip John Cornyn (R-TX) th	News	December 16, 2017
22	In A Heartless Rebuke To Victims, Trump Invites NRA To Xm	It almost seems like Donald Trump is trolling	News	December 16, 2017
23	KY GOP State Rep. Commits Suicide Over Allegations He Mc	In this #METOO moment, many powerful m	News	December 13, 2017
24	Meghan McCain Tweets The Most AMAZING Response To D	As a Democrat won a Senate seat in deep-r	News	December 12, 2017
25	CNN CALLS IT: A Democrat Will Represent Alabama In The S	Alabama is a notoriously deep red state. It s	News	December 12, 2017
26	White House: It Wasnâ€™t Sexist For Trump To Slut-Shame	A backlash ensued after Donald Trump laun	News	December 12, 2017
27	Despicable Trump Suggests Female Senator Would â€˜Do A	Donald Trump is afraid of strong, powerful	News	December 12, 2017

Fake

In this section, we load the dataset from Kaggle.

The datasets are downloaded from kaggle and loaded.

The pd.read_csv() function reads data from CSV files and stores it in Pandas DataFrames. The fake_news and real_news DataFrames will contain the fake and real news data, respectively.

STEP 3: DATA PREPROCESSING:

AI_Phase4.ipynb

```
# Combine the real and fake news datasets
fake_news['label'] = 1
real_news['label'] = 0
data = pd.concat([fake_news, real_news])

# Shuffle the data
data = data.sample(frac=1).reset_index(drop=True)
```

In this section, we combine the fake and real news datasets into one DataFrame called data. We add a 'label' column, where '1' indicates fake news and '0' indicates real news. The data is shuffled to ensure randomness, and the index is reset.

AI_Phase4.ipynb

```
# Text preprocessing
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

Here, we download the list of English stopwords from NLTK, which are common words that don't carry significant meaning in text data.

AI_Phase4.ipynb

```python
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    # Join the tokens back into a string
    text = ' '.join(tokens)
    return text

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

This part defines a function preprocess_text to apply text preprocessing to the 'text' column in the DataFrame. The steps are as follows:

- Convert the text to lowercase to make it uniform.
- Tokenize the text into individual words using NLTK.
- Remove common English stopwords from the tokenized words.
- Join the tokenized words back into a cleaned text.
-

```
[8] print(data['text'])

0        claiming least racist person , donald trump li...
1        dunkin donuts american global donut company co...
2        london ( reuters ) - britain made substantive ...
3        washington ( reuters ) - republican party ’ tw...
4        anyone else wondering cop-hating , racist , be...
                               ...
44893    30 years , donald trump built great negotiator...
44894    washington ( reuters ) - united states monday ...
44895    sarah palin marked return national stage part ...
44896    seoul ( reuters ) - south korean president moo...
44897    samarkand , uzbekistan ( reuters ) - senior of...
Name: text, Length: 44898, dtype: object
```

STEP 4: FEATURE EXTRACTION:

AI_Phase4.ipynb

```
# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(data['text'])
y = data['label']
```

In this section, we use TF-IDF vectorization to convert the preprocessed text data into numerical features. Let's explain this step by step:

- TfidfVectorizer is initialized with max_features set to 5000. This specifies that we want to consider the top 5000 most important terms in the dataset.
- tfidf_vectorizer.fit_transform(data['text']) computes the TF-IDF scores for each term in the text data, and X is a sparse matrix that represents the transformed data.
- y contains the labels for the corresponding data.

STEP 5: MODEL TRAINING AND EVALUATION:

AI_Phase4.ipynb

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

In this part, the data is split into training and testing sets. Here's what each line does:

- train_test_split(X, y, test_size=0.2, random_state=42) splits the feature matrix X and the labels y into training and testing sets.
- The test_size parameter specifies the percentage of data to use for testing (20% in this case), and random_state ensures reproducibility.

AI_Phase4.ipynb

```
# Train a classification model (Naive Bayes)
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

Here, we create a Multinomial Naive Bayes classifier **clf** and train it using the training data. Naive Bayes is a simple yet effective classification algorithm, commonly used in text classification tasks.

AI_Phase4.ipynb

```
# Make predictions
y_pred = clf.predict(X_test)
```

The model is used to make predictions on the testing set, and the predictions are stored in the y_pred variable.

AI_Phase4.ipynb

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
```

Here, we evaluate the model's performance.

- accuracy_score calculates the accuracy of the model's predictions.
- confusion_matrix generates a confusion matrix, which shows the number of true positives, true negatives, false positives, and false negatives.
- classification_report provides a comprehensive report including precision, recall, F1-score, and support for each class.
- print("Accuracy:", accuracy), print("Confusion Matrix:\n", confusion), and print("Classification Report:\n", classification_rep) display the evaluation results to the console.

AI_Phase4.ipynb

```
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.9359688195991092
Confusion Matrix:
 [[3983  301]
 [ 274 4422]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.93      4284
           1       0.94      0.94      0.94      4696

    accuracy                           0.94      8980
   macro avg       0.94      0.94      0.94      8980
weighted avg       0.94      0.94      0.94      8980
```

EXAMPLE PROGRAM USING LOGISTIC REGRESSION AND NEURAL NETWORKS:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score
from sklearn.linear_model import LogisticRegression
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, LSTM, Dense

# Load the "Fake.csv" dataset
fake_data = pd.read_csv("C:\\Users\\Bylee\\Downloads\\Fake.csv\\Fake.csv")

# Load the "True.csv" dataset
true_data = pd.read_csv("C:\\Users\\Bylee\\Downloads\\True.csv\\True.csv")
```

```
# Add labels to distinguish between fake and true news
fake_data['label'] = 0 # 0 for fake news
true_data['label'] = 1  # 1 for true news

# Combine the datasets
combined_data = pd.concat([fake_data, true_data], ignore_index=True)

# Data Preprocessing
combined_data['text'] = combined_data['title'] + " " + combined_data['text']

# Feature Extraction (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
tfidf_matrix = tfidf_vectorizer.fit_transform(combined_data['text'])

# Model Selection
X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix,
combined_data['label'], test_size=0.2, random_state=42)

# Logistic Regression Model
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)

# Model Training (Neural Network)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(combined_data['text'])
X_train_nn = tokenizer.texts_to_sequences(combined_data['text'])
X_train_nn = pad_sequences(X_train_nn, maxlen=100)

model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
model.add(LSTM(128))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.fit(X_train_nn, combined_data['label'], epochs=5, batch_size=64)

# Evaluation
# For Logistic Regression
y_pred = logistic_regression_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f"Logistic Regression Accuracy: {accuracy}")
print(f"Logistic Regression Precision: {precision}")
print(f"Logistic Regression Recall: {recall}")
print(f"Logistic Regression F1-Score: {f1}")
print(f"Logistic Regression ROC-AUC: {roc_auc}")

# For Neural Network
X_test_nn = tokenizer.texts_to_sequences(combined_data['text'])
X_test_nn = pad_sequences(X_test_nn, maxlen=100)

loss, accuracy = model.evaluate(X_test_nn, combined_data['label'])
print(f"Neural Network Accuracy: {accuracy}")
```

1. Importing Libraries:

AI_Phase4.ipynb

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
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, LSTM, Dense
```

In this part, we import the necessary libraries and modules for the project. Here's what each library/module does:

- pandas is used for data manipulation.
- TfidfVectorizer from sklearn.feature_extraction.text is used for TF-IDF feature extraction.
- train_test_split from sklearn.model_selection splits the data into training and testing sets.
- accuracy_score, precision_score, recall_score, f1_score, and roc_auc_score from sklearn.metrics are used for model evaluation.
- LogisticRegression from sklearn.linear_model is for training a logistic regression model.
- Tokenizer and pad_sequences from tensorflow.keras.preprocessing.text are for text tokenization and padding sequences.
- Sequential, Embedding, LSTM, and Dense from tensorflow.keras.models and tensorflow.keras.layers are for building a neural network model.

2. Data Loading

AI_Phase4.ipynb

```
# Load the dataset from Kaggle
fake_data = pd.read_csv("/Fake.csv")
true_data = pd.read_csv("/True.csv")
```

Here, we load the "Fake.csv" and "True.csv" datasets. The file paths should be replaced with your specific file locations. The pd.read_csv() function reads the data from CSV files into Pandas DataFrames.

3. Data Preprocessing:

AI_Phase4.ipynb

```
fake_data['label'] = 0  # 0 for fake news
true_data['label'] = 1  # 1 for true news
combined_data = pd.concat([fake_data, true_data], ignore_index=True)
combined_data['text'] = combined_data['title'] + " " + combined_data['text']
```

In this section:

- We add labels to distinguish fake (0) and true (1) news.
- The datasets are combined into one DataFrame, combined_data.
- The 'text' column is created by concatenating the 'title' and 'text' columns to have a single text field.

4. Feature Extraction (TF-IDF):

AI_Phase4.ipynb

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
tfidf_matrix = tfidf_vectorizer.fit_transform(combined_data['text'])
```

Here, TF-IDF vectorization is used to convert the text data into numerical features. The steps:

- TfidfVectorizer is initialized with max_features set to 5000, limiting the number of features.

- tfidf_vectorizer.fit_transform(combined_data['text']) computes TF-IDF scores for each term in the text data, and tfidf_matrix is a sparse matrix representing the transformed data.

5. Model Selection

AI_Phase4.ipynb

```
X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix,
combined_data['label'], test_size=0.2, random_state=42)
```

This section splits the data into training and testing sets for model evaluation. Here's a breakdown:

- train_test_split(tfidf_matrix, combined_data['label'], test_size=0.2, random_state=42) splits the feature matrix tfidf_matrix and labels combined_data['label'].
- test_size specifies the percentage of data used for testing (20%).
- random_state ensures reproducible results.

6. Model Training (Logistic Regression and Neural Network)

Logistic Regression Model

AI_Phase4.ipynb

```
# Logistic Regression Model
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)
```

Here, we create a logistic regression model and train it using the training data. Logistic regression is a linear classification model.

Neural Network Model

AI_Phase4.ipynb

```python
# Model Training (Neural Network)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(combined_data['text'])
X_train_nn = tokenizer.texts_to_sequences(combined_data['text'])
X_train_nn = pad_sequences(X_train_nn, maxlen=100)

model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
model.add(LSTM(128))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=
['accuracy'])

model.fit(X_train_nn, combined_data['label'], epochs=5, batch_size=64)
```

For the neural network:

- We use the Tokenizer to tokenize the text data, and pad_sequences to ensure sequences have a consistent length.
- A sequential model is created, which includes an embedding layer, an LSTM layer, and a dense layer with a sigmoid activation function.
- The model is compiled with loss, optimizer, and evaluation metrics.
- Finally, it's trained with the tokenized and padded sequences.

7. Evaluation

For Logistic Regression

AI_Phase4.ipynb

```
y_pred = logistic_regression_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
```

In this part, we evaluate the logistic regression model and calculate key metrics, including accuracy, precision, recall, F1-score, and ROC-AUC score.

```
print(f"Logistic Regression Accuracy: {accuracy}")
print(f"Logistic Regression Precision: {precision}")
print(f"Logistic Regression Recall: {recall}")
print(f"Logistic Regression F1-Score: {f1}")
print(f"Logistic Regression ROC-AUC: {roc_auc}")
```

```
Logistic Regression Accuracy: 0.9979286193847656
Logistic Regression Precision: 0.9866008462623413
Logistic Regression Recall: 0.9882269837532376
Logistic Regression F1-Score: 0.9874132455005294
Logistic Regression ROC-AUC: 0.9880919410631812
```

For Neural Network

AI_Phase4.ipynb

```python
X_test_nn = tokenizer.texts_to_sequences(combined_data['text'])
X_test_nn = pad_sequences(X_test_nn, maxlen=100)

loss, accuracy = model.evaluate(X_test_nn, combined_data['label'])
```

Here, we evaluate the neural network model and calculate accuracy. The model's loss is also calculated during evaluation.

```
[30] X_test_nn = tokenizer.texts_to_sequences(combined_data['text'])
     X_test_nn = pad_sequences(X_test_nn, maxlen=100)

     loss, accuracy = model.evaluate(X_test_nn, combined_data['label'])

     1404/1404 [==============================] - 96s 68ms/step - loss: 0.0064 - accuracy: 0.9979

     print(f"Neural Network Accuracy: {accuracy}")

     Neural Network Accuracy: 0.9979286193847656
```

Neural Network Accuracy: 0.9979286193847656

The results for both models are printed to the console.

OUTPUT:

```
Epoch 1/5
  1/702 [..............................] - ETA: 43:52 - loss: 0.6930 - accuracy: 0.5000
  2/702 [..............................] - ETA: 14:11 - loss: 0.6907 - accuracy: 0.5625
  3/702 [..............................] - ETA: 9:13 - loss: 0.6899 - accuracy: 0.5365
  4/702 [..............................] - ETA: 7:11 - loss: 0.6887 - accuracy: 0.5234
  5/702 [..............................] - ETA: 6:13 - loss: 0.6866 - accuracy: 0.5531
  6/702 [..............................] - ETA: 5:26 - loss: 0.6835 - accuracy: 0.5677
  7/702 [..............................] - ETA: 4:58 - loss: 0.6814 - accuracy: 0.5871
  8/702 [..............................] - ETA: 5:19 - loss: 0.6807 - accuracy: 0.5840
  9/702 [..............................] - ETA: 5:29 - loss: 0.6774 - accuracy: 0.6042
 10/702 [..............................] - ETA: 5:13 - loss: 0.6739 - accuracy: 0.6281
 11/702 [..............................] - ETA: 5:06 - loss: 0.6711 - accuracy: 0.6491
 12/702 [..............................] - ETA: 5:04 - loss: 0.6653 - accuracy: 0.6719
 13/702 [..............................] - ETA: 4:49 - loss: 0.6595 - accuracy: 0.6815
 14/702 [..............................] - ETA: 4:59 - loss: 0.6553 - accuracy: 0.6786
 15/702 [..............................] - ETA: 4:49 - loss: 0.6468 - accuracy: 0.6875
 16/702 [..............................] - ETA: 4:43 - loss: 0.6392 - accuracy: 0.6953
 17/702 [..............................] - ETA: 4:38 - loss: 0.6317 - accuracy: 0.7050
 18/702 [..............................] - ETA: 4:36 - loss: 0.6230 - accuracy: 0.7144
 19/702 [..............................] - ETA: 4:33 - loss: 0.6128 - accuracy: 0.7253
 20/702 [..............................] - ETA: 4:27 - loss: 0.6033 - accuracy: 0.7289
 21/702 [..............................] - ETA: 4:23 - loss: 0.5881 - accuracy: 0.7374
 22/702 [..............................] - ETA: 4:18 - loss: 0.5747 - accuracy: 0.7450
 23/702 [..............................] - ETA: 4:14 - loss: 0.5639 - accuracy: 0.7473
 24/702 [>.............................] - ETA: 4:11 - loss: 0.5506 - accuracy: 0.7546
 25/702 [>.............................] - ETA: 4:10 - loss: 0.5381 - accuracy: 0.7613
 26/702 [>.............................] - ETA: 4:09 - loss: 0.5282 - accuracy: 0.7686
 27/702 [>.............................] - ETA: 4:06 - loss: 0.5170 - accuracy: 0.7755
 28/702 [>.............................] - ETA: 4:05 - loss: 0.5069 - accuracy: 0.7801
 29/702 [>.............................] - ETA: 4:07 - loss: 0.4970 - accuracy: 0.7850
 30/702 [>.............................] - ETA: 4:06 - loss: 0.4898 - accuracy: 0.7891
 31/702 [>.............................] - ETA: 4:05 - loss: 0.4817 - accuracy: 0.7918
 32/702 [>.............................] - ETA: 4:03 - loss: 0.4753 - accuracy: 0.7939
 33/702 [>.............................] - ETA: 4:03 - loss: 0.4689 - accuracy: 0.7969
 34/702 [>.............................] - ETA: 4:02 - loss: 0.4632 - accuracy: 0.7996
 35/702 [>.............................] - ETA: 4:03 - loss: 0.4541 - accuracy: 0.8045
 36/702 [>.............................] - ETA: 4:02 - loss: 0.4469 - accuracy: 0.8073
 37/702 [>.............................] - ETA: 4:03 - loss: 0.4424 - accuracy: 0.8095
 38/702 [>.............................] - ETA: 4:04 - loss: 0.4362 - accuracy: 0.8129
 39/702 [>.............................] - ETA: 4:04 - loss: 0.4277 - accuracy: 0.8165
 40/702 [>.............................] - ETA: 4:03 - loss: 0.4205 - accuracy: 0.8195
 41/702 [>.............................] - ETA: 4:05 - loss: 0.4148 - accuracy: 0.8224
 42/702 [>.............................] - ETA: 4:05 - loss: 0.4095 - accuracy: 0.8248
 43/702 [>.............................] - ETA: 4:06 - loss: 0.4030 - accuracy: 0.8281
 44/702 [>.............................] - ETA: 4:06 - loss: 0.3968 - accuracy: 0.8317
 45/702 [>.............................] - ETA: 4:07 - loss: 0.3900 - accuracy: 0.8351
 46/702 [>.............................] - ETA: 4:07 - loss: 0.3840 - accuracy: 0.8380
```

```
006
 93/702 [==>...........................] - ETA: 6:23 - loss: 0.2426 - accuracy: 0.9014
 94/702 [===>..........................] - ETA: 6:28 - loss: 0.2408 - accuracy: 0.9023
 95/702 [===>..........................] - ETA: 6:33 - loss: 0.2395 - accuracy: 0.9028
 96/702 [===>..........................] - ETA: 6:37 - loss: 0.2379 - accuracy: 0.9038
 97/702 [===>..........................] - ETA: 6:42 - loss: 0.2362 - accuracy: 0.9048
 98/702 [===>..........................] - ETA: 6:47 - loss: 0.2343 - accuracy: 0.9056
 99/702 [===>..........................] - ETA: 6:56 - loss: 0.2338 - accuracy: 0.9058
100/702 [===>..........................] - ETA: 7:05 - loss: 0.2326 - accuracy: 0.9062
101/702 [===>..........................] - ETA: 7:18 - loss: 0.2311 - accuracy: 0.9070
102/702 [===>..........................] - ETA: 7:26 - loss: 0.2296 - accuracy: 0.9076
103/702 [===>..........................] - ETA: 7:33 - loss: 0.2278 - accuracy: 0.9084
104/702 [===>..........................] - ETA: 7:40 - loss: 0.2261 - accuracy: 0.9093
105/702 [===>..........................] - ETA: 7:48 - loss: 0.2253 - accuracy: 0.9095
106/702 [===>..........................] - ETA: 7:55 - loss: 0.2239 - accuracy: 0.9101
107/702 [===>..........................] - ETA: 8:03 - loss: 0.2224 - accuracy: 0.9108
108/702 [===>..........................] - ETA: 8:11 - loss: 0.2209 - accuracy: 0.9115
109/702 [===>..........................] - ETA: 8:18 - loss: 0.2204 - accuracy: 0.9120
110/702 [===>..........................] - ETA: 8:27 - loss: 0.2190 - accuracy: 0.9126
111/702 [===>..........................] - ETA: 8:34 - loss: 0.2178 - accuracy: 0.9131
112/702 [===>..........................] - ETA: 8:41 - loss: 0.2166 - accuracy: 0.9138
113/702 [===>..........................] - ETA: 8:48 - loss: 0.2153 - accuracy: 0.9145
114/702 [===>..........................] - ETA: 8:55 - loss: 0.2149 - accuracy: 0.9150
115/702 [===>..........................] - ETA: 9:05 - loss: 0.2140 - accuracy: 0.9155
116/702 [===>..........................] - ETA: 9:14 - loss: 0.2124 - accuracy: 0.9162
117/702 [====>.........................] - ETA: 9:21 - loss: 0.2112 - accuracy: 0.9167
118/702 [====>.........................] - ETA: 9:28 - loss: 0.2100 - accuracy: 0.9172
119/702 [====>.........................] - ETA: 9:35 - loss: 0.2096 - accuracy: 0.9174
120/702 [====>.........................] - ETA: 9:42 - loss: 0.2085 - accuracy: 0.9180
121/702 [====>.........................] - ETA: 9:50 - loss: 0.2070 - accuracy: 0.9186
122/702 [====>.........................] - ETA: 10:01 - loss: 0.2069 - accuracy: 0.9185
123/702 [====>.........................] - ETA: 10:10 - loss: 0.2058 - accuracy: 0.9190
124/702 [====>.........................] - ETA: 10:20 - loss: 0.2050 - accuracy: 0.9192
125/702 [====>.........................] - ETA: 10:28 - loss: 0.2042 - accuracy: 0.9196
126/702 [====>.........................] - ETA: 10:40 - loss: 0.2034 - accuracy: 0.9198
127/702 [====>.........................] - ETA: 10:52 - loss: 0.2032 - accuracy: 0.9198
128/702 [====>.........................] - ETA: 11:02 - loss: 0.2019 - accuracy: 0.9203
129/702 [====>.........................] - ETA: 11:12 - loss: 0.2013 - accuracy: 0.9205
130/702 [====>.........................] - ETA: 11:23 - loss: 0.2004 - accuracy: 0.9210
131/702 [====>.........................] - ETA: 11:36 - loss: 0.2000 - accuracy: 0.9213
132/702 [====>.........................] - ETA: 11:50 - loss: 0.1988 - accuracy: 0.9218
133/702 [====>.........................] - ETA: 12:01 - loss: 0.1975 - accuracy: 0.9221
134/702 [====>.........................] - ETA: 12:10 - loss: 0.1963 - accuracy: 0.9226
135/702 [====>.........................] - ETA: 12:19 - loss: 0.1956 - accuracy: 0.9230
136/702 [====>.........................] - ETA: 12:28 - loss: 0.1947 - accuracy: 0.9235
137/702 [====>.........................] - ETA: 12:37 - loss: 0.1935 - accuracy: 0.9239
138/702 [====>.........................] - ETA: 12:47 - loss: 0.1929 - accuracy: 0.9244
139/702 [====>.........................] - ETA: 12:57 - loss: 0.1916 - accuracy: 0.9249
140/702 [====>.........................] - ETA: 13:10 - loss: 0.1906 - accuracy: 0.9254
141/702 [=====>........................] - ETA: 13:18 - loss: 0.1893 - accuracy: 0.9260
142/702 [=====>........................] - E
```

CONCLUSION:

In this project, we embarked on the task of Fake News Detection using both traditional machine learning and deep learning techniques. We initially loaded and combined two datasets, 'Fake.csv' and 'True.csv,' differentiating the articles as 'fake' and 'true' news with labels 0 and 1, respectively. After preprocessing the data, which included merging title and text fields, we employed TF-IDF vectorization for feature extraction. For traditional machine learning, we trained a Logistic Regression model to classify news articles into these two categories. Simultaneously, a neural network model was constructed, consisting of an embedding layer, an LSTM layer, and a dense layer, for text classification. Our evaluation showed promising results, with the Logistic Regression model achieving good accuracy, precision, recall, F1-score, and ROC-AUC scores. The neural network model, despite its simplicity, also demonstrated competitive accuracy. Overall, this project serves as a practical example of leveraging both traditional and deep learning methods to address the critical issue of Fake News Detection