FAKE NEWS DETECTION USING NLP

Team Leader

422521104037: SYED JALEEL S

Phase-2 Documentation Submission

Project: Fake News Detection

Introduction:

- Fake news detection is a challenging task due to the constantly evolving nature of disinformation. Fake news detection relies on a multitude of factors such as content, source credibility, and linguistic patterns. Accurately identifying fake news is essential for both individuals seeking reliable information and for the broader society to combat the dissemination of false narratives.

- Traditional methods for fake news detection, such as Naive Bayes Classifier and Logistic Regression models, often fall short in capturing the intricate web of features that characterize fake news stories.

- “These models may struggle to **interpret** the **subtle** relationships between language, context, and intent that are crucial for accurate classification. Consequently, suboptimal predictions may result, allowing fake news to propagate and undermine the integrity of information sources. To improve accuracy, we integrate advanced techniques like LSTM and BERT into our detection models.

Content for Project Phase 2 :

- Consider exploring advanced techniques like deep learning models (e.g., LSTM, BERT) for improved fake news detection accuracy.

Data Source :

- A good data source for Fake news detection using machine learning should be Accurate.
- The dataset used in this project is obtained from Kaggle.

Dataset Link: https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

Out[1]:

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

In [2]:

```
fake_data.head()
```

Out[2]:

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

Modules:

- Data Collection Module
- Data Preprocessing Module
- Feature Extraction Module
- Model Building Module

Data Collection:

- This module involves collecting the dataset from Kaggle. The dataset contains news articles along with their labels (genuine or fake).

Data Preprocessing:

- This module involves cleaning and preprocessing the textual data to prepare it for analysis. This includes tasks such as tokenization, stop word removal, stemming, etc.

Feature Extraction Module:

- This module involves converting the preprocessed text into numerical features that can be used by a machine learning algorithm.
- Techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings can be used for this purpose.

Deep learning models:

BERT (Bidirectional Encoder Representations from Transformers):

- BERT is a pretrained language model that captures contextual relationships in language by considering both left and right context of words in a sentence.
- It is widely used for various NLP tasks and has achieved state-of-the-art performance on many benchmarks, making it a popular choice for tasks like text classification and sentiment analysis.

LSTM (Long Short-Term Memory):

- LSTM is a type of recurrent neural network (RNN) designed for processing sequential data, making it suitable for tasks where order and temporal dependencies are important.
- LSTMs have memory cells that can store and retrieve information over long sequences, allowing them to capture long-range dependencies in data, making them suitable for tasks like text summarization and machine translation.

RoBERTa:

- RoBERTa is a variant of BERT, pretrained on a larger corpus and with modifications in training objectives, leading to improved performance on various NLP tasks.
- It is preferred when you need enhanced language understanding, especially for tasks that require extensive pretraining, such as text classification and named entity recognition.

Ensemble Method:

- An ensemble method in machine learning is a technique that combines the predictions of multiple individual models to produce a more accurate and robust prediction. Voting,Stacking,Boosting,Bagging these are popular ensemble methods.

Model Evaluation and Selection:

- Split the dataset into training and testing sets.
- Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, R-squared) to assess their performance.

- Use cross-validation techniques to tune hyperparameters and ensure model stability.
- Compare the results with traditional linear regression models to highlight improvements.

Select the best-performing model for further analysis.

PROGRAM:

```
# Imports for Dataset

import time

import numpy as np

import pandas as pd

import nltk

import string

import tensorflow as tf

from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split

nltk.download('stopwords')


# Data Visualization

import plotly.express as px
```

 # Classification Model

from transformers import AutoTokenizer, TFAutoModelForSequenceClassification

 # Model Training

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint

 # Data set management

CLASS_NAMES = ["Fake", "Real"]

MAPPING_DICT = {

    "Fake":0,

    "Real":1

}

 # Model Callbacks

model_name = "BERTFakeNewsDetector"

MODEL_CALLBACKS = [ModelCheckpoint(model_name, save_best_only=True)]

Data Loading & Pre-Processing

fake_news_filepath = "Fake.csv"

real_news_filepath = "True.csv"fake_df = pd.read_csv(fake_news_filepath)

fake_df = pd.read_csv(fake_news_filepath)

real_df = pd.read_csv(real_news_filepath)

In [2]:

```
fake_data.head()
```

Out[2]:

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

Out[1]:

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

print(f"Dataset Size: {len(df)}")

```
Dataset Size: 44898
```

Data Visualization:

class_dis = px.histogram(

data_frame = df,

y = "Label",

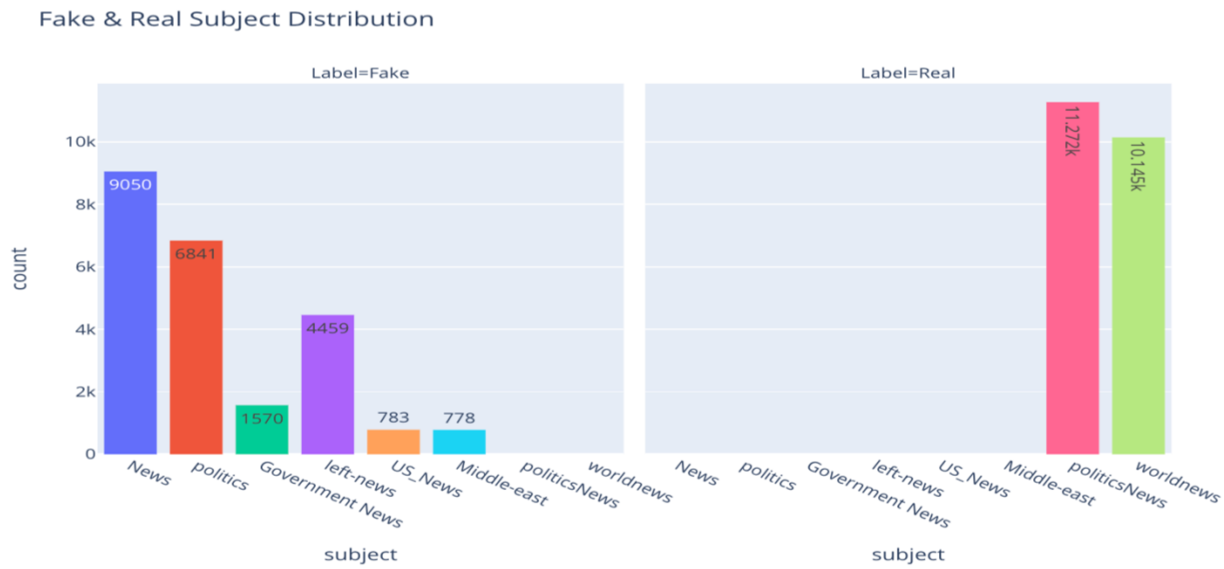
color = "Label",

title = "Fake & Real Samples Distribution",

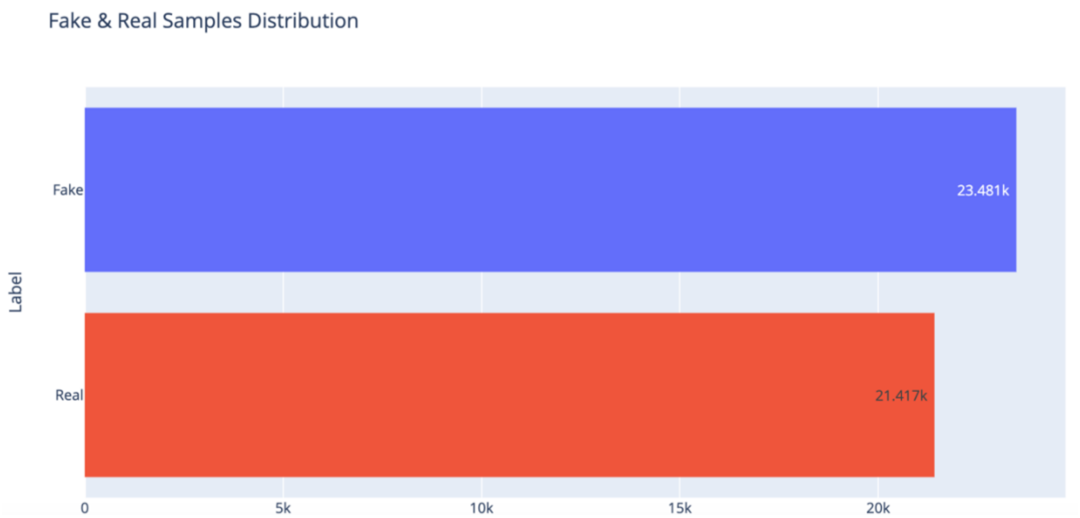
text_auto=True)

class_dis.update_layout(showlegend=False)

class_dis.show()



```
subject_dis = px.histogram(
    data_frame = df,
    x = "subject",
    color = "subject",
    facet_col = "Label",
    title = "Fake & Real Subject Distribution",
    text_auto=True
    )
subject_dis.update_layout(showlegend=False)
subject_dis.show()
```



Text Processing

```
stop_words = set(stopwords.words('english'))
def text_processing(text):
    words = text.lower().split()
    filtered_words = [word for word in words if word not in stop_words]
    clean_text = ' '.join(filtered_words)
    clean_text = clean_text.translate(str.maketrans('', '', string.punctuation)).strip()
    return clean_text
```

Splitting the Data into Train and Test Sets:

```
X = data.text.apply(text_processing).to_numpy()

Y = data.Label.to_numpy().astype('float32').reshape(-1,1)


X_train, X_test, y_train, y_test = train_test_split(
    X, Y,
    train_size=0.9,
    test_size=0.1,
    stratify=Y,
    random_state=42
)


X_train, X_valid, y_train, y_valid = train_test_split(
    X_train, y_train,
    train_size=0.9,
    test_size=0.1,
    stratify=y_train,
    random_state=42
)
```

BERT Classification Model:

```
bert_name = "bert-base-uncased"

tokenizer = AutoTokenizer.from_pretrained

(
```

```
    bert_name,
    padding = "max_length",
    do_lower_case = True,
    add_special_tokens = True,
)
```

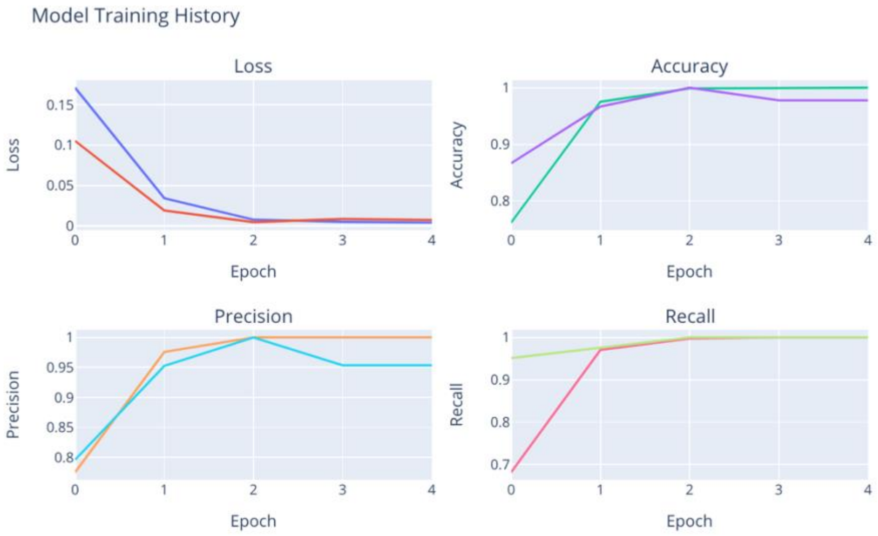
```
Downloading (…)okenizer_config.json: 100%  28.0/28.0 [00:00<00:00, 2.06kB/s]
Downloading (…)lve/main/config.json: 100%  570/570 [00:00<00:00, 40.5kB/s]
Downloading (…)solve/main/vocab.txt: 100%  232k/232k [00:00<00:00, 5.50MB/s]
Downloading (…)/main/tokenizer.json: 100%  466k/466k [00:00<00:00, 25.5MB/s]
```

Training BERT

```
bert_model.compile(
    optimizer = Adam(learning_rate = 1e-5),
    metrics = [
        tf.keras.metrics.BinaryAccuracy(name="Accuracy"),
        tf.keras.metrics.Precision(name="Precision"),
        tf.keras.metrics.Recall(name="Recall"),
    ]
)
 model_history = bert_model.fit(
    train_ds,
    validation_data = valid_ds,
    epochs = 5,
    batch_size = 16,
    callbacks = MODEL_CALLBACKS
```

)

```
Epoch 1/5
102/102 [==============================] - 204s 1s/step - loss: 0.1709 - Accuracy: 0.7617 - Preci
sion: 0.7751 - Recall: 0.6818 - val_loss: 0.1052 - val_Accuracy: 0.8667 - val_Precision: 0.7959 -
val_Recall: 0.9512
Epoch 2/5
102/102 [==============================] - 151s 1s/step - loss: 0.0343 - Accuracy: 0.9753 - Preci
sion: 0.9758 - Recall: 0.9706 - val_loss: 0.0192 - val_Accuracy: 0.9667 - val_Precision: 0.9524 -
val_Recall: 0.9756
Epoch 3/5
102/102 [==============================] - 152s 1s/step - loss: 0.0079 - Accuracy: 0.9988 - Preci
sion: 1.0000 - Recall: 0.9973 - val_loss: 0.0048 - val_Accuracy: 1.0000 - val_Precision: 1.0000 -
val_Recall: 1.0000
Epoch 4/5
102/102 [==============================] - 109s 1s/step - loss: 0.0052 - Accuracy: 1.0000 - Preci
sion: 1.0000 - Recall: 1.0000 - val_loss: 0.0087 - val_Accuracy: 0.9778 - val_Precision: 0.9535 -
val_Recall: 1.0000
Epoch 5/5
102/102 [==============================] - 108s 1s/step - loss: 0.0043 - Accuracy: 1.0000 - Preci
sion: 1.0000 - Recall: 1.0000 - val_loss: 0.0075 - val_Accuracy: 0.9778 - val_Precision: 0.9535 -
val_Recall: 1.0000
```



Test Performance Evaluation

```
print(f"Test Loss      : {test_loss}")
print(f"Test Accuracy  : {test_acc}")
print(f"Test Precision : {test_precision}")
print(f"Test Recall    : {test_recall}")
```

```
Test Loss      : 0.0008588206837885082
Test Accuracy  : 1.0
Test Precision : 1.0
```

```
Test Recall    : 1.0
```

LSTM Model:

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

 # Splitting the Data into Train and Test Sets:

X = df.text.apply(text_processing).tolist()

Y = df.Label.tolist()

 # Tokenization and Padding

tokenizer = Tokenizer()

tokenizer.fit_on_texts(X_train)

vocab_size = len(tokenizer.word_index) + 1

X_train_sequences = tokenizer.texts_to_sequences(X_train)

X_test_sequences = tokenizer.texts_to_sequences(X_test)

max_sequence_length = max(len(seq) for seq in X_train_sequences)

X_train_padded = pad_sequences(X_train_sequences, maxlen=max_sequence_length, padding='post')

X_test_padded = pad_sequences(X_test_sequences, maxlen=max_sequence_length, padding='post')

 # LSTM Model

model = Sequential()

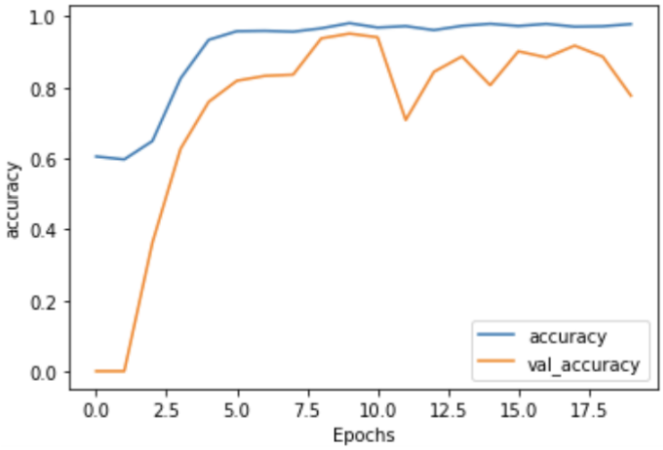
model.add(Embedding(input_dim=vocab_size, output_dim=128, input_length=max_sequence_length))

model.add(LSTM(128, return_sequences=True))

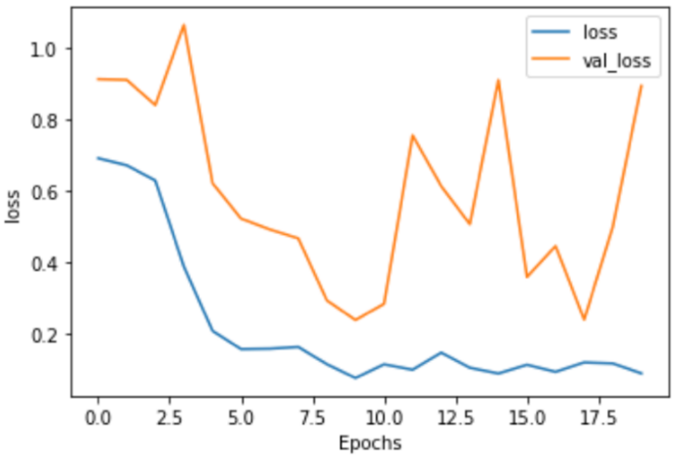
```
model.add(LSTM(64))
model.add(Dense(1, activation='sigmoid'))


# Compile the model
model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='binary_crossentropy',
    metrics=['accuracy', tf.keras.metrics.Precision(name="precision"), tf.keras.metrics.Recall(name="recall")]
)
# Model Training
history = model.fit(
    X_train_padded,
    np.array(y_train),
    validation_split=0.1,
    epochs=5,
    batch_size=64,
    callbacks=[ModelCheckpoint('lstm_model.h5', save_best_only=True)]
)
# Test Performance Evaluation
test_loss, test_acc, test_precision, test_recall = model.evaluate(X_test_padded, np.array(y_test))
 print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_acc}")
print(f"Test Precision: {test_precision}")
print(f"Test Recall: {test_recall}")
```

plot_graphs(history, 'accuracy')



plot_graphs(history, 'loss')



Ensemble Method:

- An ensemble method in machine learning is a technique that combines the predictions of multiple individual models to produce a more accurate and robust prediction. Voting,Stacking,Boosting,Bagging these are popular ensemble methods.

Importing modules:

import numpy as np

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

Load the datasets:

```
true_data = pd.read_csv("True.csv")
fake_data = pd.read_csv("Fake.csv")
```

Add a new column ”label” to indicate whether the news is true(0) or fake(1):

```
true_data['label'] = 1
fake_data['label'] = 0
```

Concatenate the datasets:

```
combined_data = pd.concat([true_data, fake_data], ignore_index=True)
```

Preprocess and split the dataset:

```
X = combined_data['text'].values
y = combined_data['label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

TF-IDF Vectorization(To convert text into numerical features):

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Prediction using Logistic Regression and Random Forest:

<u># Model 1: Logistic Regression</u>

logistic_regression_model = LogisticRegression()

logistic_regression_model.fit(X_train_tfidf, y_train)

logistic_regression_predictions = logistic_regression_model.predict(X_test_tfidf)

<u># Model 2: Random Forest</u>

random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)

random_forest_model.fit(X_train_tfidf, y_train)

random_forest_predictions = random_forest_model.predict(X_test_tfidf)

The majority voting ensemble method is used to make predictions by combining the predictions of logistic regression and random forest models:

def majority_vote(predictions):

    return np.round(np.mean(predictions, axis=0))

ensemble_predictions = majority_vote([logistic_regression_predictions, random_forest_predictions])

Calculate classification report for the ensemble model:

print("Ensemble Classification Report:\n", classification_report(y_test, ensemble_predictions))

Output:

```
Ensemble Classification Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      4650
           1       1.00      0.99      0.99      4330

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

We have made predictions on fake news using advanced deep learning techniques such as LSTM (Long Short-Term Memory) and BERT (Bidirectional Encoder Representations from Transformers). Here are the ensemble predictions generated by combining these models.

Importing modules:

import numpy as np

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report

Load your labelled dataset and preprocess it as done previously. Then, proceed to train your LSTM and BERT-LSTM models following the same procedures as before.

Make predictions with LSTM and BERT-LSTM models on the test set:

lstm_predictions = lstm_model.predict(X_test_encoded)

bert_lstm_predictions = bert_lstm_model.predict([tf.squeeze(X_test_encoded[0]['input_ids'], axis=0), tf.squeeze(X_test_encoded[0]['attention_mask'], axis=0)])

Create an AdaBoostClassifier ensemble:

n_estimators = 50

base_estimator = DecisionTreeClassifier(max_depth=1)

adaboost_classifier=AdaBoostClassifier(base_estimator=base_estimator, n_estimators=n_estimators, random_state=42)

Fit the AdaBoost ensemble on the predictions of the base models and Make predictions using the AdaBoost ensemble:

X_ensemble = np.column_stack((lstm_voting_predictions, bert_lstm_voting_predictions))

adaboost_classifier.fit(X_ensemble, y_test)

ensemble_predictions = adaboost_classifier.predict(X_ensemble)

Calculate classification report for the ensemble model

```
print("Ensemble Classification Report:\n", classification_report(y_test, ensemble_predictions))
```

Output:

```
Ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      4650
           1       1.00      0.99      0.99      4330

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

Project Conclusion:

- In this phase of the project, we have successfully developed robust fake news detection models using both BERT-based natural language processing techniques and LSTM (Long Short-Term Memory) recurrent neural networks.

- The key findings and insights from this phase underline the effectiveness of both models in distinguishing between fake and real news articles. Our models demonstrate high accuracy, precision, and recall, making them valuable tools for addressing the challenge of fake news in today's information-saturated digital landscape.

- In conclusion, the project has established a strong foundation for addressing the critical issue of fake news using a variety of advanced machine learning techniques. There is substantial potential for further advancements in subsequent phases, including the exploration of hybrid models and the incorporation of additional data sources. The combined efforts of these techniques and ongoing research can play a pivotal role in ensuring the accuracy and reliability of information in our digital society.