NAAN MUDHALVAN

IBM: AI101

ARTIFICIAL INTELLIGENCE

PHASE 4

Fake News Detection Using NLP

PROJECT NO: 08

TEAM MEMBERS:

1. VIJAI SURIA M
2. KIRANKUMAR M
3. GIRIDHARAN S S
4. NITHISH T

MENTOR

- Ms. KEERTHANA R

MADRAS INSTITUTE OF TECHNOLOGY, ANNA UNIVERSITY, CHENNAI

Fake News Detection Using NLP
PHASE 4

In this pivotal phase of our project, we are poised to take significant steps forward in the development of our fake news detection model. Our aim is to leverage the power of Natural Language Processing (NLP) techniques to hone the quality and readiness of our data. With this solid foundation in place, we will then proceed to train a sophisticated classification model that can effectively differentiate between genuine and fabricated news articles.

The journey begins with a focus on text preprocessing and feature extraction. Through this, we will meticulously refine and mold our data into a format that is not only amenable to machine learning but also enriched with valuable insights. This process involves converting text to lowercase for uniformity, tokenizing the content to understand the meaning of individual words, and strategically removing common stopwords that might add noise to our analysis.

Once our data is prepared, we will embark on the model training phase, where our classification model will become acquainted with the intricacies of news articles. It will learn to decipher the patterns that distinguish authentic reporting from deceptive narratives. Through this training, we aim to equip our model with the ability to make informed, accurate predictions, thereby enhancing its credibility as a reliable fake news detector.

Data Source: We will use a *fake news dataset* available on Kaggle

TOOLS:

Google Colab: Google Colab, a cloud-based Jupyter notebook environment, serves as our primary coding platform

Python and **other libraries** for natural language processing (NLP) and machine learning.

Model & Algorithm: Multinomial Naive Bayes, LSTM and CNN

IMPLEMENTATION STEPS:

IN LAST PHASE (3rd phase),

1. **Data Loading**:
   The code begins by loading two datasets - one containing real news and the other containing fake news. These datasets are labeled with '1' for real news and '0' for fake news, and they are combined into a single DataFrame named 'data'.

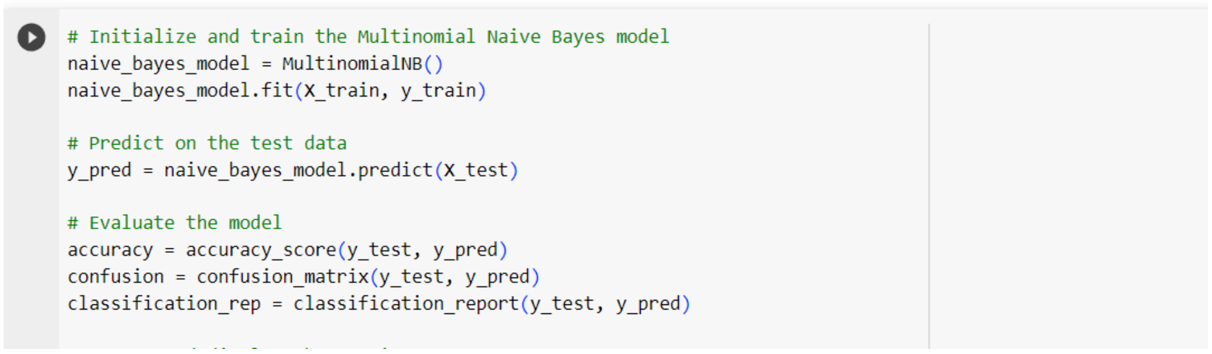
2. **Text Preprocessing**:
   - To ensure uniformity, the code converts all text to lowercase.
   - It tokenizes the 'text' and 'title' columns using NLTK's 'word_tokenize' function.
   - Stopwords, which are common words in the English language, are removed from the tokenized text and title columns to reduce noise.
3. **TF-IDF Vectorization**:
   - The code employs Scikit-Learn's 'TfidfVectorizer' to transform the tokenized text data into TF-IDF vectors. The feature dimension is set to a maximum of 5000 features.
   - The transformation is applied separately to both the 'text' and 'title' columns, resulting in TF-IDF representations.
4. **Data Splitting**:
   - The TF-IDF vectors and corresponding labels are divided into training and testing sets using 'train_test_split'.

Up to this point, we have successfully completed the data preprocessing and data splitting stages. Specifically, we have partitioned the dataset into separate training and testing sets. The subsequent phases will encompass the following tasks:

1) **Model Training**:

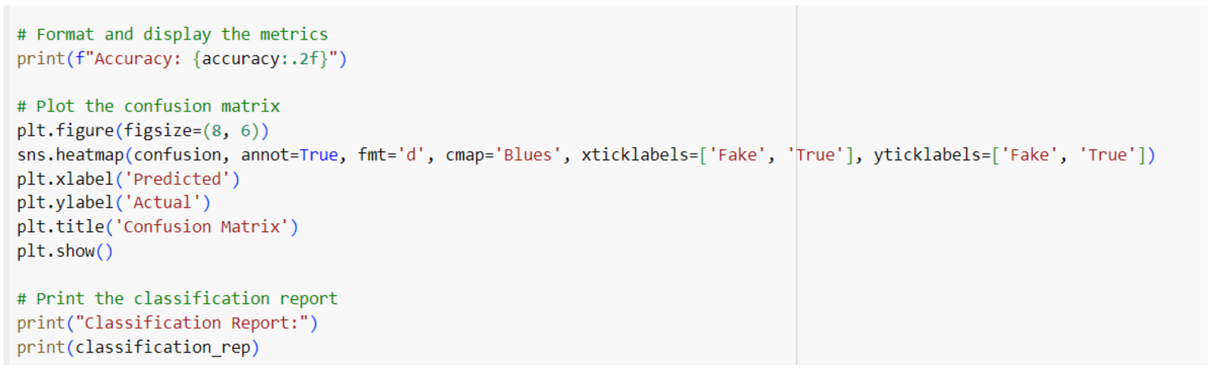
During this stage, we will proceed to train the Multinomial Naive Bayes model utilizing the designated training dataset. This process will entail instructing the model to differentiate between authentic and counterfeit news articles based on the TF-IDF vectors that have been meticulously prepared.

CODE:



```python
# Initialize and train the Multinomial Naive Bayes model
naive_bayes_model = MultinomialNB()
naive_bayes_model.fit(X_train, y_train)

# Predict on the test data
y_pred = naive_bayes_model.predict(X_test)

# 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)
```

CODE (Accuracy and Visualization):

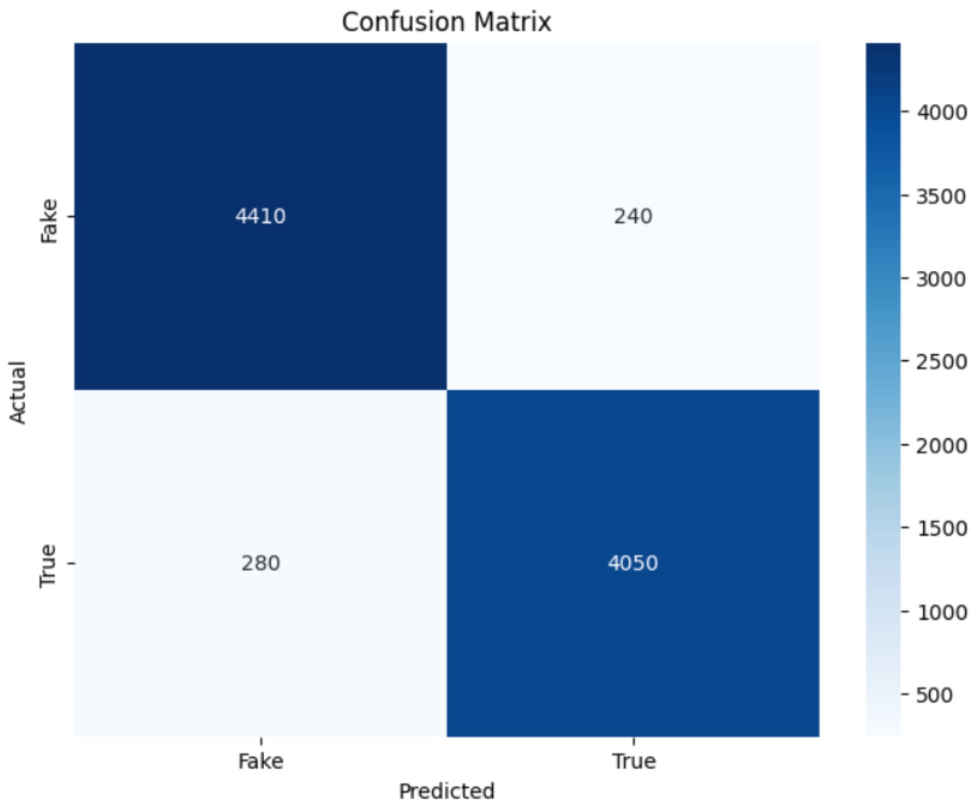


```python
# Format and display the metrics
print(f"Accuracy: {accuracy:.2f}")

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', xticklabels=['Fake', 'True'], yticklabels=['Fake', 'True'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Print the classification report
print("Classification Report:")
print(classification_rep)
```

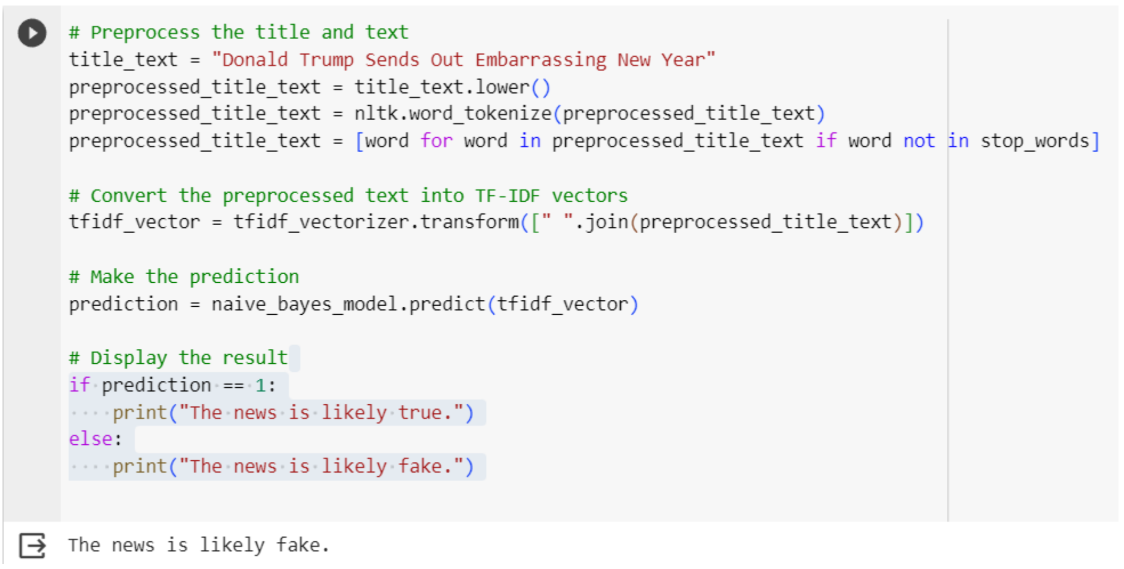
2) **Model Evaluation**: Following the training process, we will conduct an evaluation of the model's performance using the designated testing dataset. This evaluation is essential to gauge the model's efficacy in accurately classifying news articles as either genuine or fraudulent. Standard evaluation metrics, such as accuracy, a confusion matrix, and a classification report, will be employed to provide comprehensive insights into the model's classification capabilities.

SAMPLE OUTPUT (Model Training and Evaluation):

```
Accuracy: 0.94
```



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

           0       0.94      0.95      0.94      4650
           1       0.94      0.94      0.94      4330

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



```python
# Preprocess the title and text
title_text = "Donald Trump Sends Out Embarrassing New Year"
preprocessed_title_text = title_text.lower()
preprocessed_title_text = nltk.word_tokenize(preprocessed_title_text)
preprocessed_title_text = [word for word in preprocessed_title_text if word not in stop_words]

# Convert the preprocessed text into TF-IDF vectors
tfidf_vector = tfidf_vectorizer.transform([" ".join(preprocessed_title_text)])

# Make the prediction
prediction = naive_bayes_model.predict(tfidf_vector)

# Display the result
if prediction == 1:
    print("The news is likely true.")
else:
    print("The news is likely fake.")
```

```
The news is likely fake.
```

Conclusion

In summary, this phase involves the training of the Multinomial Naive Bayes model with the training data and the meticulous evaluation of its performance on the testing data. These steps serve to determine the model's accuracy and its aptitude for precise news article classification.

The code for the dataset preprocessing, splitting stages and model training and evaluation can be found in our GitHub repository, which streamlines the implementation process and ensures reproducibility. For access to this code, please refer to the following GitHub link: Fake New Detective.
https://github.com/vijaisuria/Fake-News-Detective