Preprocessing Summary

In the preprocessing step, we focused on preparing the data for training a multi-class fake news detection model using two datasets: the Fake News Corpus and the Kaggle Fake and Real News Dataset.

1. Dataset Merging

- Fake News Corpus:
- Columns: id, domain, type, url, content, scraped_at, inserted_at, updated_at, title, authors, keywords, meta_keywords, meta_description, tags, summary, source.
- Focused columns: title, content, authors, keywords, source.
- Categorization:
- opensources → Fake
- nytimes → Real
- webhose → Unverified/Biased
- Kaggle Fake and Real News Dataset:
- Columns: title, text, category (Fake/Real).
- Used the title and text columns, labeled them as Fake or Real.

We merged both datasets and added the category label:

- Fake → Merged Fake News Corpus and Kaggle Fake dataset.
- Real → Merged Kaggle Real dataset.
- Unverified → Articles from the webhose category in the Fake News Corpus.

2. Variables and Their Types

Here are the variables in the dataset, their types, and their purpose:

Variable	Description	Type
title	Title of the news article	String
content	Main body of the article	String
author	Author of the article	String (or Unknown)
category	Label for the news article (Fake, Real, Unverified)	Integer (0/1/2)
keywords	Keywords associated with the article	String (comma-separated)
article_length	Length of the article (word count)	Integer
num_keywords	Number of keywords (split by comma)	Integer
fake_news_score	A score representing the author's credibility based on past articles	Float

3. Handling Missing Data

- Missing Text Fields (title, content):

Missing values in the title or content columns were filled with "Unknown" to ensure no data is lost and can still be used for model training.

```
final_data.fillna("Unknown", inplace=True)
```

- Missing Author Data:

Missing author values were assigned as "Unknown" to maintain consistency without dropping rows, ensuring more data is used.

4. Feature Engineering

We created the following new features to improve the model's performance:

- Article Length: Calculated the word count of the article's content.

```
final_data["article_length"] = final_data["content"].apply(lambda x: len(str(x).split()))
```

- Number of Keywords: Counted the number of keywords (split by commas).

```
final_data["num_keywords"] = final_data["keywords"].apply(lambda x: len(str(x).split(',')) if x != "Unknown" else 0)
```

5. Categorization and Labeling

We categorized news articles into three classes:

- Fake: Articles from unreliable sources (e.g., opensources.co).
- Real: Articles from trusted sources (e.g., nytimes).
- Unverified/Biased: Articles from aggregators or unverified sources (e.g., webhose).

This resulted in a multi-class classification problem where the model will predict Fake, Real, or Unverified/Biased news.

6. Final Dataset Structure

The final dataset (processed_news.csv) was structured as follows:

Variable	Description
title	The title of the news article
content	The body/content of the article
author	The author of the article (or "Unknown")
category	The label: Fake (0), Real (1), Unverified (2)
article_length	Length of the article in words
num_keywords	Number of keywords associated with the article
fake_news_score	The credibility score of the author based on previous articles

Key Insights and Next Steps

- Data Cleansing: We ensured no data loss by filling missing values properly.
- Feature Engineering: By creating new features (article length, number of keywords), we aimed to improve the model's performance.
- Categorization: We turned the Fake News Corpus and Kaggle datasets into a unified format with Fake, Real, and Unverified/Biased labels.

Next Steps

- Train our model on this processed dataset using either a traditional machine learning model (like Random Forest) or an advanced model like BERT.
- Evaluate the model's performance using accuracy, precision, and recall.