Building a fake news detection model involves several steps, including loading and preprocessing the dataset, as well as model selection, training, and evaluation. In this response, I'll focus on the initial steps of loading and preprocessing the dataset. For this example, I'll assume you have a labeled dataset with two classes: "fake" and "real" news articles. You can adapt these steps to your specific dataset and requirements.

1. **Import Necessary Libraries**:

First, import the libraries you'll need, such as Python, NumPy, pandas, and scikit-learn (for machine learning):

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
```

2. **Load and Explore the Dataset**:

Load your dataset into a DataFrame. Ensure that it contains at least two columns: one for the text of the news articles and one for the labels (e.g., "fake" or "real").

```python
data = pd.read_csv("your_dataset.csv")
```

3. **Data Preprocessing**:

Preprocess the text data to prepare it for model training. Common preprocessing steps include:

- **Text Cleaning**: Remove any special characters, HTML tags, and irrelevant formatting.

```python
```

```
import re

def clean_text(text):
    # Remove special characters and digits
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text
data['text'] = data['text'].apply(clean_text)
```

- **Tokenization**: Split the text into words or tokens.

```python
from nltk.tokenize import word_tokenize

data['tokens'] = data['text'].apply(word_tokenize)
```

- **Text Vectorization**: Convert the text into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).

```python
vectorizer = TfidfVectorizer(max_features=5000)  # Adjust the number of features as needed
X = vectorizer.fit_transform(data['text'])
```

- **Label Encoding**: Encode the labels (e.g., "fake" and "real") into numerical values.

```python
from sklearn.preprocessing import LabelEncoder
```

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

4. **Train-Test Split**:

Split the dataset into training and testing sets to evaluate the model's performance.

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

Now, you have a preprocessed dataset ready for training your fake news detection model. The next steps involve selecting a machine learning or deep learning model, training it, and evaluating its performance using appropriate metrics. Additionally, you can consider using techniques like natural language processing (NLP) models such as BERT or other advanced approaches for improved fake news detection.