```
#Import essential libraries
import numpy as np
import pandas as pd
import re
import string
import zipfile
import os
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,f1_score


# Extract dataset from ZIP file
zip_path = "/content/archive (4).zip"
extract_path = "/mnt/data/fake_news_dataset"
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)



#Load datasets
fake_news_path = os.path.join(extract_path, "Fake.csv")
true_news_path = os.path.join(extract_path, "True.csv")

data_fake = pd.read_csv(fake_news_path)
data_true = pd.read_csv(true_news_path)


#Add labels to datasets
data_fake["label"] = 1
data_true["label"] = 0


#Combine datasets
data = pd.concat([data_fake, data_true], ignore_index=True)
data = data[["text","label"]]


# Define manual stopwords
manual_stopwords = set(["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yo



# Text preprocessing function
def simple_preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    text = ' '.join([word for word in text.split() if word not in manual_stopwords])  # Remove stopwords
    return text


# Apply preprocessing
data["clean_text"] = data["text"].apply(simple_preprocess_text)




# Convert text into numerical features using TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(data["clean_text"])
y = data["label"]


# Split dataset 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)

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

```
▾ MultinomialNB
MultinomialNB()
```

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


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(" The Model Accuracy is:", accuracy)
print("The Classification Report is:\n", classification_rep)
print("The F1 score is:",f1)
```

```
 The Model Accuracy is: 0.9423162583518931
The Classification Report is:
               precision    recall  f1-score   support

           0       0.93      0.94      0.94      4247
           1       0.95      0.94      0.95      4733

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

The F1 score is: 0.9450339558573854
```

```
# Function to predict new news articles
def predict_news(news_text):
    processed_text = simple_preprocess_text(news_text)  # Preprocess input text
    vectorized_text = vectorizer.transform([processed_text])  # Convert to TF-IDF
    prediction = model.predict(vectorized_text)  # Predict using trained model
    return "Fake News" if prediction == 1 else "Real News"

# Example test case
test_news = "Breaking: Scientists discover a new way to generate unlimited energy!"
print("Prediction:", predict_news(test_news))
```

```
Prediction: Fake News
```