```
# Data handling
import pandas as pd
import numpy as np

# Text preprocessing
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Feature extraction
from sklearn.feature_extraction.text import TfidfVectorizer

# Model building
from sklearn.naive_bayes import MultinomialNB

# Data splitting
from sklearn.model_selection import train_test_split

# Evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("/content/news.csv", encoding='latin-1')

df = df[['label', 'text']]
df.columns = ['label', 'text']

print(df.head())
print("Dataset Size:", df.shape)
print("\nClass Distribution:\n", df['label'].value_counts())
```

```
  label                                               text
0  FAKE  Daniel Greenfield, a Shillman Journalism Fello...
1  FAKE  Google Pinterest Digg Linkedin Reddit Stumbleu...
2  REAL  U.S. Secretary of State John F. Kerry said Mon...
3  FAKE  â Kaydee King (@KaydeeKing) November 9, 2016...
4  REAL  It's primary day in New York and front-runners...
Dataset Size: (6335, 2)

Class Distribution:
 label
REAL    3171
FAKE    3164
Name: count, dtype: int64
```

```
nltk.download('stopwords')
nltk.download('wordnet')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))

    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_w

    return " ".join(words)

df['clean_text'] = df['text'].apply(preprocess)

print(df[['text', 'clean_text']].head())
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
                                                text  \
0  Daniel Greenfield, a Shillman Journalism Fello...
1  Google Pinterest Digg Linkedin Reddit Stumbleu...
2  U.S. Secretary of State John F. Kerry said Mon...
3  â Kaydee King (@KaydeeKing) November 9, 2016...
4  It's primary day in New York and front-runners...

                                          clean_text
0  daniel greenfield shillman journalism fellow f...
1  google pinterest digg linkedin reddit stumbleu...
2  u secretary state john f kerry said monday sto...
3  â kaydee king kaydeeking november lesson ton...
4  primary day new york frontrunners hillary clin...
```

```
vectorizer = TfidfVectorizer(
    max_features=10000,
    ngram_range=(1,2),     # Unigrams + Bigrams
    min_df=2
)

X = vectorizer.fit_transform(df['clean_text'])
y = df['label'].map({'FAKE': 0, 'REAL': 1}) # Ensure correct label mapping

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

```
print("Feature Matrix Shape:", X.shape)
print("Training size:", X_train.shape)
print("Testing size:", X_test.shape)
print("Sample Features (first 20):", vectorizer.get_feature_names_out()[:20])
```

```
Feature Matrix Shape: (6335, 10000)
Training size: (5068, 10000)
Testing size: (1267, 10000)
Sample Features (first 20): ['aaron' 'abandon' 'abandoned' 'abandoning' 'abc' 'a
 'abdulazeez' 'abdullah' 'abedin' 'abedinâ' 'ability' 'able' 'able get'
 'aboard' 'aborigine' 'abortion' 'abortion right' 'aboutâ' 'abraham']
```

```
from sklearn.linear_model import LogisticRegression

model_lr = LogisticRegression(solver='liblinear', max_iter=1000, random_state=42

model_lr.fit(X_train, y_train)

print(model_lr.get_params())
```

```
{'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercep
```

```
y_pred_lr = model_lr.predict(X_test)

accuracy_lr = accuracy_score(y_test, y_pred_lr)
precision_lr = precision_score(y_test, y_pred_lr)
recall_lr = recall_score(y_test, y_pred_lr)
f1_lr = f1_score(y_test, y_pred_lr)

print("Logistic Regression Model Performance:")
print("Accuracy:", accuracy_lr)
print("Precision:", precision_lr)
print("Recall:", recall_lr)
print("F1-Score:", f1_lr)

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_lr))
```

```
Logistic Regression Model Performance:
Accuracy: 0.9187056037884768
Precision: 0.9417637271214643
Recall: 0.8927444794952681
F1-Score: 0.9165991902834008

Classification Report:

              precision    recall  f1-score   support

           0       0.90      0.94      0.92       633
           1       0.94      0.89      0.92       634
```

```
              accuracy                           0.92      1267
             macro avg       0.92      0.92      0.92      1267
          weighted avg       0.92      0.92      0.92      1267
```

```
cm_lr = confusion_matrix(y_test, y_pred_lr) # Calculate the confusion matrix for

plt.figure(figsize=(6,5)) # Create a new figure with a specified size for the he
sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Blues', # Generate a heatmap of th
            xticklabels=['FAKE','REAL'], # Label the x-axis ticks (predicted cla
            yticklabels=['FAKE','REAL']) # Label the y-axis ticks (actual classe

plt.xlabel("Predicted") # Set the label for the x-axis
plt.ylabel("Actual") # Set the label for the y-axis
plt.title("Confusion Matrix for Logistic Regression") # Set the title of the plo
plt.show() # Display the plot
```