```
# For data loading and handling
import pandas as pd
import numpy as np

# For text preprocessing
import re #Removes unwanted patterns (punctuation, numbers).
import string #Provides punctuation symbols.
import nltk  #Used for text preprocessing.
from nltk.corpus import stopwords  #Removes common words like is, the, and.
from nltk.stem import PorterStemmer  #Reduces words to root form

# For feature extraction
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# For train-test split
from sklearn.model_selection import train_test_split

# For model building
from sklearn.naive_bayes import MultinomialNB

# For evaluation
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Download stopwords (only once) - ensures necessary data for NLP is available
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
# Load dataset from the specified CSV file
df = pd.read_csv('/content/news.csv')

# Display the first 5 rows of the DataFrame to inspect the data structure
print("First 5 samples:")
print(df.head())

# Print the shape of the dataset (number of rows, number of columns)
print("\nDataset Shape:", df.shape)

# Display the column names to understand available features
print("\nColumns:", df.columns)

# Assuming:
# text column = 'text' (This is the content of the news article)
# label column = 'label' (This indicates if the news is 'FAKE' or 'REAL')
# (Change column names if different based on your dataset)

# Print the distribution of classes in the 'label' column to check for imbalance
print("\nClass Distribution:")
print(df['label'].value_counts())
```

```
First 5 samples:
   Unnamed: 0                                              title  \
0        8476                       You Can Smell Hillary’s Fear   
1       10294  Watch The Exact Moment Paul Ryan Committed Pol...   
2        3608        Kerry to go to Paris in gesture of sympathy   
3       10142  Bernie supporters on Twitter erupt in anger ag...   
4         875   The Battle of New York: Why This Primary Matters   

                                                text label  
0  Daniel Greenfield, a Shillman Journalism Fello...  FAKE  
1  Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE  
2  U.S. Secretary of State John F. Kerry said Mon...  REAL  
3  — Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE  
4  It's primary day in New York and front-runners...  REAL  

Dataset Shape: (6335, 4)

Columns: Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')

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

```
# ==============================
# STEP 4: Text Preprocessing
# ==============================
# Initialize stop words for English language
stop_words = set(stopwords.words('english'))
# Initialize Porter Stemmer for reducing words to their root form
stemmer = PorterStemmer()

# Define a function to preprocess text
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove punctuation from the text using a regular expression
    text = re.sub(f"[{string.punctuation}]", "", text)

    # Remove numbers from the text
    text = re.sub(r'\d+', '', text)

    # Tokenize the text into individual words
    words = text.split()

    # Remove stopwords and apply stemming to each word
    words = [stemmer.stem(word) for word in words if word not in stop_words]

    # Join the processed words back into a single string
    return " ".join(words)

# Apply the preprocessing function to the 'text' column and store in a new 'clean_text' column
df['clean_text'] = df['text'].apply(preprocess_text)

# Display a sample of the original and cleaned text to verify preprocessing
print("\nSample Cleaned Text:")
print(df[['text', 'clean_text']].head())
```

```

Sample Cleaned Text:
                                                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  — Kaydee King (@KaydeeKing) November 9, 2016 T...
4  It's primary day in New York and front-runners...

                                          clean_text
0  daniel greenfield shillman journal fellow free...
1  googl pinterest digg linkedin reddit stumbleup...
2  us secretari state john f kerri said monday st...
3  — kayde king kaydeek novemb lesson tonight dem...
4  primari day new york frontrunn hillari clinton...
```

```
# Initialize TF-IDF Vectorizer with a maximum of 5000 features
vectorizer = TfidfVectorizer(max_features=5000)

# Fit the vectorizer on the 'clean_text' and transform it into a feature matrix X
X = vectorizer.fit_transform(df['clean_text'])
# Assign the 'label' column to y, which will be our target variable
y = df['label']

# Print the shape of the feature matrix (number of samples, number of features)
print("Feature Matrix Shape:", X.shape)

# Display the first 20 feature names extracted by the vectorizer
print("\nSample Feature Names:")
print(vectorizer.get_feature_names_out()[:20])
```

```
Feature Matrix Shape: (6335, 5000)

Sample Feature Names:
['abandon' 'abc' 'abdullah' 'abedin' 'abil' 'abl' 'aboard' 'abolish'
 'aborigin' 'abort' 'about' 'abraham' 'abroad' 'absenc' 'absent' 'absente'
 'absolut' 'absorb' 'absurd' 'abu']
```

```
# Split the data into training and testing sets
# X: features, y: target labels
# test_size=0.2: 20% of data for testing, 80% for training
# random_state=42: ensures reproducibility of the split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

# Print the shape of the training feature set
print("Training size:", X_train.shape)
# Print the shape of the testing feature set
```

```
print("Testing size:", X_test.shape)
```

```
Training size: (5068, 5000)
Testing size: (1267, 5000)
```

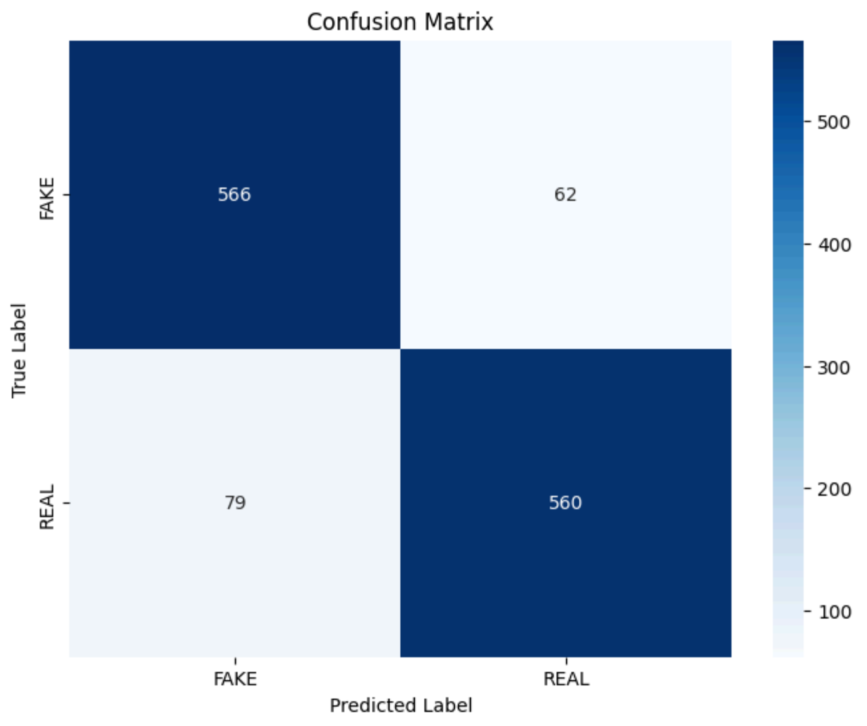
```
# Initialize the Multinomial Naive Bayes model
model = MultinomialNB()

# Train the model using the training features (X_train) and training labels (y_train)
model.fit(X_train, y_train)

# Confirm that the model training is complete
print("Model trained successfully!")

# Display the parameters of the trained model
print("\nModel Parameters:")
print(model.get_params())
```

```
Model trained successfully!

Model Parameters:
{'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True}
```

```
# Make predictions on the test set using the trained model
y_pred = model.predict(X_test)

# Calculate and print the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate and print the classification report, which includes precision, recall, f1-score
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Generate and print the confusion matrix to evaluate classification performance
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8887134964483031

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.88      0.90      0.89       628
        REAL       0.90      0.88      0.89       639

    accuracy                           0.89      1267
   macro avg       0.89      0.89      0.89      1267
weighted avg       0.89      0.89      0.89      1267


Confusion Matrix:
[[566  62]
 [ 79 560]]
```

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

# Get the confusion matrix from the last step (calculated in the previous cell)
cm = confusion_matrix(y_test, y_pred)

# Define class labels for the confusion matrix for better readability
class_labels = ['FAKE', 'REAL']

# Create a figure with a specific size for the heatmap
plt.figure(figsize=(8, 6))
# Generate the heatmap using seaborn, with annotations, integer format, and 'Blues' colormap
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
# Set the title of the plot
plt.title('Confusion Matrix')
# Set the label for the x-axis (Predicted Label)
plt.xlabel('Predicted Label')
# Set the label for the y-axis (True Label)
plt.ylabel('True Label')
# Display the plot
plt.show()
```


Confusion Matrix
True Label
FAKE
REAL
566
62
79
560
FAKE
REAL
Predicted Label
500
400
300
200
100