```
import pandas as pd
import numpy as np

import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_rep

# Download NLTK data if not already present
try:
    stopwords.words('english')
    WordNetLemmatizer()
except LookupError:
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

```
df = pd.read_csv("news.csv")
display(df.head())
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary’s Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE

```
# Feature Extraction (TF-IDF Vectorization)
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # Limiting to 5000 features for demonstration
X = tfidf_vectorizer.fit_transform(df['processed_text'])
y = df['label'] # Assuming 'label' is the target variable

# Split the data 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)

print(f"Shape of X_train: {X_train.shape}")
print(f"Shape of X_test: {X_test.shape}")
print(f"Shape of y_train: {y_train.shape}")
print(f"Shape of y_test: {y_test.shape}")
```

```
Shape of X_train: (5068, 5000)
Shape of X_test: (1267, 5000)
Shape of y_train: (5068,)
Shape of y_test: (1267,)
```

```
# Initialize WordNetLemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Check if text is a string, otherwise return empty string
    if not isinstance(text, str):
        return ""

    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = re.sub(r'[^a-z\s]', '', text) # Keep only alphabets and spaces

    # Tokenize and remove stopwords, then lemmatize
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]

    return ' '.join(words)

# Apply preprocessing to the 'text' column (assuming 'text' column exists)
# First, check if the 'text' column exists in the DataFrame
if 'text' in df.columns:
    df['processed_text'] = df['text'].apply(preprocess_text)
    display(df[['text', 'processed_text']].head())
else:
    print("Error: 'text' column not found in the DataFrame.")
```

	text	processed_text
0	Daniel Greenfield, a Shillman Journalism Fello...	daniel greenfield shillman journalism fellow f...
1	Google Pinterest Digg Linkedin Reddit Stumbleu...	google pinterest digg linkedin reddit stumbleu...
2	U.S. Secretary of State John F. Kerry said Mon...	u secretary state john f kerry said monday sto...
3	— Kaydee King (@KaydeeKing) November 9, 2016 T...	kaydee king kaydeeking november lesson tonight...
4	It's primary day in New York and front-runners...	primary day new york frontrunners hillary clin...

```
# Display sample feature names
feature_names = tfidf_vectorizer.get_feature_names_out()
print("Sample Feature Names (first 20):")
print(feature_names[:20])
print(f"Total number of features: {len(feature_names)}")
```

```
Sample Feature Names (first 20):
['abandon' 'abandoned' 'abc' 'abdullah' 'abedin' 'ability' 'able'
 'abortion' 'abroad' 'absence' 'absent' 'absentee' 'absolute' 'absolutely'
 'absurd' 'abuse' 'academic' 'academy' 'accept' 'acceptable']
Total number of features: 5000
```

```
from sklearn.naive_bayes import MultinomialNB

# Initialize and train the Multinomial Naive Bayes model
mnb_model = MultinomialNB()
mnb_model.fit(X_train, y_train)

# Display model parameters (if any, or just a confirmation of training)
print("Multinomial Naive Bayes model trained successfully.")
print("Model parameters:")
print(mnb_model.get_params())
```

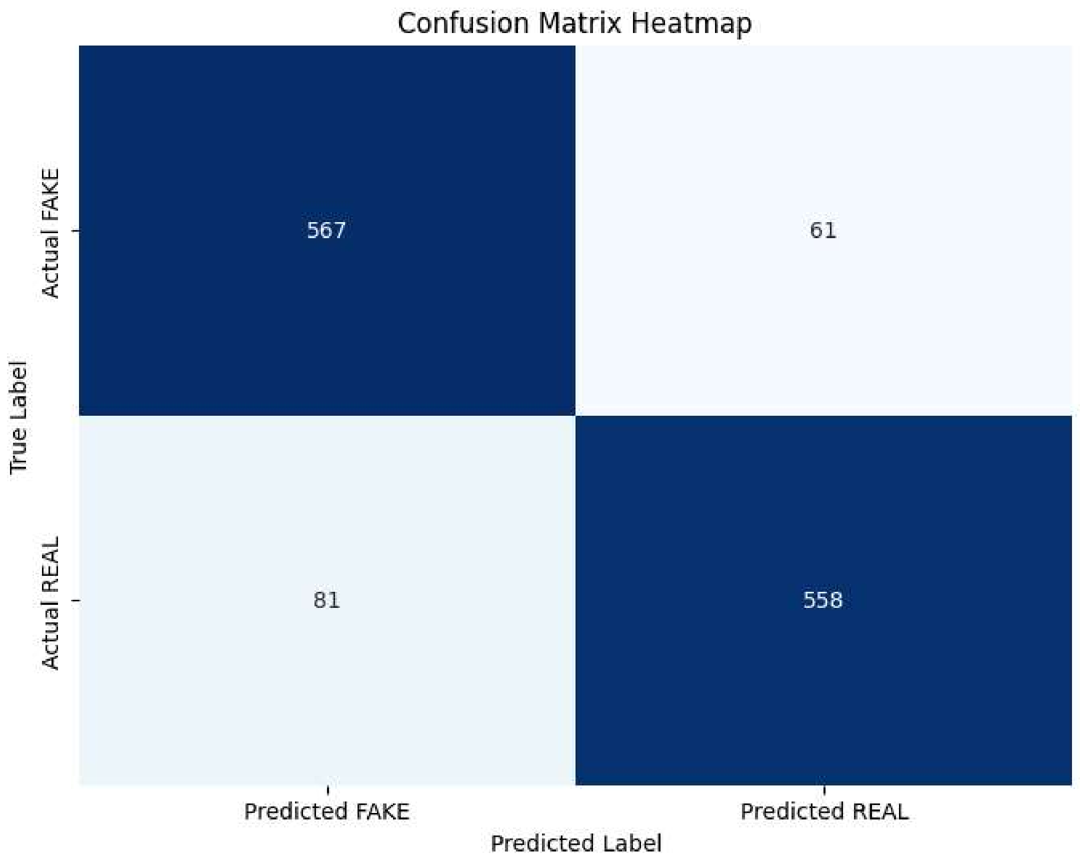
```
Multinomial Naive Bayes model trained successfully.
Model parameters:
{'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True}
```

```
# Make predictions on the test set
y_pred = mnb_model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)

# Display the metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)

# Optionally, display a classification report for more detailed metrics per class
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8879
Precision: 0.8883
Recall: 0.8879
F1-Score: 0.8879

Confusion Matrix:
[[567  61]
 [ 81 558]]

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.88      0.90      0.89       628
        REAL       0.90      0.87      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
```

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

```
# Plotting the Confusion Matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Predicted FAKE', 'Predicted REAL'],
            yticklabels=['Actual FAKE', 'Actual REAL'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```


Confusion Matrix Heatmap
True Label
Actual FAKE
Actual REAL
Predicted FAKE
Predicted REAL
Predicted Label