Import Required Libraries

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

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

# Download stopwords (run once)
nltk.download('stopwords')

# Feature extraction
from sklearn.feature_extraction.text import TfidfVectorizer

# Model building
from sklearn.naive_bayes import MultinomialNB

# Train-test split
from sklearn.model_selection import train_test_split

# Model evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

# Visualization (optional)
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Load and Explore Dataset

Gemini

```
# Load dataset
df = pd.read_csv('/content/news.csv')

# Display first 5 rows
print("First 5 samples:")
display(df.head())

# Print dataset size
print("Dataset Shape:", df.shape)

# Check column names
print("Columns:", df.columns)

# Check class distribution
print("\nClass Distribution:")
print(df['label'].value_counts())
```

```
First 5 samples:
```

	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
4	875	The Battle of New York: Why This Primary Matters	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
```

Text Preprocessing

```
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))

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

    # Remove stopwords
    words = text.split()
    words = [word for word in words if word not in stop_words]

    return " ".join(words)

# Apply preprocessing
df['clean_text'] = df['text'].apply(preprocess_text)

print("Preprocessing Completed.")
display(df[['text', 'clean_text']].head())
```

```
Preprocessing Completed.
```

	text	clean_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...	us secretary state john f kerry said monday st...
3	— Kaydee King (@KaydeeKing) November 9, 2016 T...	— kaydee king kaydeeking november lesson tonig...
4	It's primary day in New York and front-runners...	primary day new york frontrunners hillary clin...

Feature Extraction (TF-IDF)

```
vectorizer = TfidfVectorizer(ngram_range=(1,2), max_features=10000)

X = vectorizer.fit_transform(df['clean_text'])

y = df['label']

print("Feature Matrix Shape:", X.shape)

# Display some feature names
print("Sample Feature Names:")
print(vectorizer.get_feature_names_out()[:20])
```

```
Feature Matrix Shape: (6335, 10000)
Sample Feature Names:
['aaron' 'abandon' 'abandoned' 'abandoning' 'abc' 'abc news' 'abcs'
 'abdeslam' 'abdulazeez' 'abdullah' 'abedin' 'ability' 'able' 'able get'
 'aboard' 'abortion' 'abortion rights' 'abortions' 'about' 'abraham']
```

Train-Test Split

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

print("Training Data Shape:", X_train.shape)
print("Testing Data Shape:", X_test.shape)
```

```
Training Data Shape: (5068, 10000)
Testing Data Shape: (1267, 10000)
```

Train Naive Bayes Model

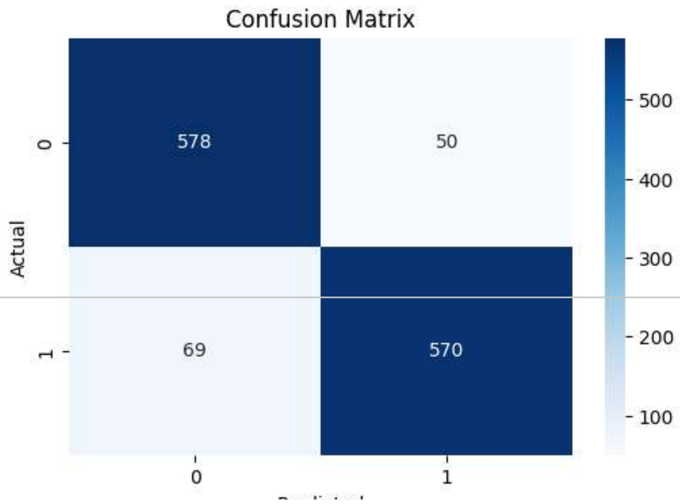
```
model = MultinomialNB()

# Train model
model.fit(X_train, y_train)

print("Model Training Completed.")
print("Model Parameters:")
print(model)
```

```
Model Training Completed.
Model Parameters:
MultinomialNB()
```

Model Evaluation

```
y_pred = model.predict(X_test)

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')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.9060773480662984
Precision: 0.9064672589198135
Recall: 0.9060773480662984
F1-Score: 0.9060684544575558

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.89      0.92      0.91       628
        REAL       0.92      0.89      0.91       639

    accuracy                           0.91      1267
   macro avg       0.91      0.91      0.91      1267
weighted avg       0.91      0.91      0.91      1267
```


Confusion Matrix
Actual