

```

# pandas → used to load and handle dataset in table format
import pandas as pd

# numpy → used for numerical operations (optional but commonly used)
import numpy as np

# string → contains list of punctuation symbols to remove from text
import string

# TfidfVectorizer → converts text into numerical feature vectors
from sklearn.feature_extraction.text import TfidfVectorizer

# train_test_split → used to divide dataset into training and testing sets
from sklearn.model_selection import train_test_split

# MultinomialNB → Naive Bayes model suitable for text classification
from sklearn.naive_bayes import MultinomialNB

# Metrics → used to evaluate model performance
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confus

```

```

# Load your dataset file
data = pd.read_csv("news.csv")

# Display first 5 rows to understand structure
print(data.head())

# Check dataset size
print("Dataset Shape:", data.shape)

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

      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')

```

```

# Reload the dataset to ensure 'title' is present and consistent
import pandas as pd
data = pd.read_csv("news.csv")

# We need 'title' and 'text' as input and 'label' as output
data = data[['title', 'text', 'label']]

# Remove rows where 'label' is NaN
data = data.dropna(subset=['label'])

```

```
# Check class distribution (FAKE vs REAL)
print(data['label'].value_counts())
```

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

```
# Function to clean text data
def preprocess_text(text):

    # Convert text to lowercase → makes "News" and "news" same
    text = text.lower()

    # Remove punctuation → removes symbols like ! ? . ,
    text = "".join([char for char in text if char not in string.punctuation])

    return text

# Fill any NaN values in the 'text' column with an empty string
data['text'] = data['text'].fillna('')
data['text'] = data['text'].str.lower()
data['text'] = data['text'].str.replace('[^a-zA-Z ]', '', regex=True)

# Apply preprocessing to all text data
data['text'] = data['text'].apply(preprocess_text)
```

```
# Create TF-IDF vectorizer
# stop_words='english' removes common words like "is", "the"
# Create TF-IDF vectorizer with better settings for NLP classification
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Improved TF-IDF settings
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_df=0.5,
    min_df=2,
    ngram_range=(1,2),    # bigrams help a lot
    sublinear_tf=True
)
```

```
# Convert text into numeric feature matrix
X = vectorizer.fit_transform(data['text'])
```

```
# Target labels
y = data['label']
```

```
# Print feature matrix shape
print("Feature Matrix Shape:", X.shape)
```

```
# Display some feature words
print("Sample Features:", vectorizer.get_feature_names_out()[:20])
```

```
Feature Matrix Shape: (6335, 312062)
Sample Features: ['aa' 'aa superluminal' 'aaa' 'aab' 'aadmi' 'aadmi party' 'aaiball'
 'aaiball exclude' 'aam' 'aam aadmi' 'aamaq' 'aamaq news' 'aap'
 'aap supremo' 'aaron' 'aaron burr' 'aaron david' 'aaron davis'
 'aaron dykes' 'aaron klein']
```

```
# Split dataset into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y  # ★ VERY IMPORTANT
)

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

```
Training Data Size: (5068, 312062)
Testing Data Size: (1267, 312062)
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(
    max_iter=2000,
    C=5,
    solver='liblinear'
)

model.fit(X_train, y_train)
```

```
▼ LogisticRegression ⓘ ?  
LogisticRegression(C=5, max_iter=2000, solver='liblinear')
```

```
# Create Multinomial Naive Bayes model

from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB(alpha=0.1)  # tuning parameter
model.fit(X_train, y_train)
```

```
# Train model using training data

print("Model Training Completed")
```

```
Model Training Completed

# Import required evaluation metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, classification_report
```

```
from sklearn.linear_model import LogisticRegression

# -----
# STEP 1: Train the model
# -----


# Logistic Regression is best for text classification
model = LogisticRegression(max_iter=1000)

# Train model using training data
model.fit(X_train, y_train)

# -----
# STEP 2: Make Predictions on Test Data
# -----


# Predict labels for unseen test data

y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test, y_pred)*100)

# -----
# STEP 3: Calculate Evaluation Metrics
# -----


# Accuracy → Overall correctness of model

# Precision → Out of predicted FAKE, how many are actually FAKE
precision = precision_score(y_test, y_pred, pos_label='FAKE')

# Recall → Out of actual FAKE news, how many were detected
recall = recall_score(y_test, y_pred, pos_label='FAKE')

# F1 Score → Balance between precision and recall
f1 = f1_score(y_test, y_pred, pos_label='FAKE')

# -----
# STEP 4: Print Results
# -----


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

# -----
# STEP 5: Confusion Matrix
# -----


print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# -----
# STEP 6: Detailed Classification Report
# -----
```

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

```
Accuracy: 92.73875295974744
Precision: 0.9092284417549168
Recall: 0.9494470774091627
F1 Score: 0.9289026275115919
```

```
Confusion Matrix:
```

```
[[601 32]
 [ 60 574]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
FAKE	0.91	0.95	0.93	633
REAL	0.95	0.91	0.93	634
accuracy			0.93	1267
macro avg	0.93	0.93	0.93	1267
weighted avg	0.93	0.93	0.93	1267

```
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['FAKE','REAL'],
            yticklabels=['FAKE','REAL'])

plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



