```python
import pandas as pd
import numpy as np

from datasets import load_dataset

# Load IMDB dataset
dataset = load_dataset("imdb")

# Convert training data to Pandas DataFrame
train_df = pd.DataFrame(dataset["train"])
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"d7e9fa593d5d46fb903651a2c6d9a407","version_major":2,"vers
ion_minor":0}

{"model_id":"59a240a7dfec4d2394118d24a1a126ea","version_major":2,"vers
ion_minor":0}

{"model_id":"3c4fe2d038fe47ad9b9bf30d50bf586a","version_major":2,"vers
ion_minor":0}

{"model_id":"fc793fccd9db49f093cd82cb6a3b8831","version_major":2,"vers
ion_minor":0}

{"model_id":"76af4ed7005a4752a1bb310880815714","version_major":2,"vers
ion_minor":0}

{"model_id":"5adeb59aacec4eb8ade07d9758c20d7f","version_major":2,"vers
ion_minor":0}

{"model_id":"d9142465ea1b46e6bfe64c7d7780415a","version_major":2,"vers
ion_minor":0}
```


```python
# First 5 rows
print(train_df.head())

# Info about dataset
print("\n--- INFO ---")
print(train_df.info())

# Check missing values
print("\n--- NULL VALUES ---")
```

```python
print(train_df.isnull().sum())

# Check balance of labels
print("\n--- LABEL COUNTS ---")
print(train_df["label"].value_counts())

# Add text length column
train_df["text_length"] = train_df["text"].apply(len)
print("\n--- TEXT LENGTH STATS ---")
print(train_df["text_length"].describe())
```

```
                                                text  label
0  I rented I AM CURIOUS-YELLOW from my video sto...      0
1  "I Am Curious: Yellow" is a risible and preten...      0
2  If only to avoid making this type of film in t...      0
3  This film was probably inspired by Godard's Ma...      0
4  Oh, brother...after hearing about this ridicul...      0

--- INFO ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    25000 non-null  object
 1   label   25000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 390.8+ KB
None

--- NULL VALUES ---
text     0
label    0
dtype: int64

--- LABEL COUNTS ---
label
0    12500
1    12500
Name: count, dtype: int64

--- TEXT LENGTH STATS ---
count    25000.00000
mean      1325.06964
std       1003.13367
min         52.00000
25%        702.00000
50%        979.00000
75%       1614.00000
```

```
max      13704.00000
Name: text_length, dtype: float64
```

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

# Download necessary resources
nltk.download("stopwords")
nltk.download("punkt")
nltk.download("punkt_tab")
nltk.download("wordnet")

# Load stopwords & lemmatizer
stop_words = set(stopwords.words("english"))
# Keep negations for sentiment analysis
for keep in ["not", "no", "nor", "n't"]:
    if keep in stop_words:
        stop_words.remove(keep)

lemmatizer = WordNetLemmatizer()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
def clean_text(text):
    # 1. Lowercase
    text = text.lower()

    # 2. Remove HTML tags
    text = re.sub(r"<.*?>", "", text)

    # 3. Remove punctuation & numbers
    text = re.sub(r"[^a-z\s]", "", text)

    # 4. Tokenize
    words = nltk.word_tokenize(text)

    # 5. Remove stopwords & lemmatize
    words = [lemmatizer.lemmatize(w) for w in words if w not in
stop_words]

    return " ".join(words)

# Apply cleaning
```

```
train_df["clean_text"] = train_df["text"].apply(clean_text)

# Show before & after for first review
print("Before:\n", train_df["text"].iloc[0])
print("\nAfter:\n", train_df["clean_text"].iloc[0])
```

```
Before:
 I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded it when it was first released in 1967. I also heard that at first it was seized by U.S. customs if it ever tried to enter this country, therefore being a fan of films considered "controversial" I really had to see this for myself.<br /><br />The plot is centered around a young Swedish drama student named Lena who wants to learn everything she can about life. In particular she wants to focus her attentions to making some sort of documentary on what the average Swede thought about certain political issues such as the Vietnam War and race issues in the United States. In between asking politicians and ordinary denizens of Stockholm about their opinions on politics, she has sex with her drama teacher, classmates, and married men.<br /><br />What kills me about I AM CURIOUS-YELLOW is that 40 years ago, this was considered pornographic. Really, the sex and nudity scenes are few and far between, even then it's not shot like some cheaply made porno. While my countrymen mind find it shocking, in reality sex and nudity are a major staple in Swedish cinema. Even Ingmar Bergman, arguably their answer to good old boy John Ford, had sex scenes in his films.<br /><br />I do commend the filmmakers for the fact that any sex shown in the film is shown for artistic purposes rather than just to shock people and make money to be shown in pornographic theaters in America. I AM CURIOUS-YELLOW is a good film for anyone wanting to study the meat and potatoes (no pun intended) of Swedish cinema. But really, this film doesn't have much of a plot.

After:
 rented curiousyellow video store controversy surrounded first released also heard first seized u custom ever tried enter country therefore fan film considered controversial really see myselfthe plot centered around young swedish drama student named lena want learn everything life particular want focus attention making sort documentary average swede thought certain political issue vietnam war race issue united state asking politician ordinary denizen stockholm opinion politics sex drama teacher classmate married menwhat kill curiousyellow year ago considered pornographic really sex nudity scene far even not shot like cheaply made porno countryman mind find shocking reality sex nudity major staple swedish cinema even ingmar bergman arguably answer good old boy john ford sex scene filmsi commend filmmaker fact sex shown film shown artistic purpose rather shock people make money shown pornographic theater america curiousyellow good film anyone wanting study meat potato no pun intended swedish cinema really film doesnt much plot
```

```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# 1. Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    train_df["clean_text"], train_df["label"],
    test_size=0.2,
    random_state=42
)

print("Train size:", len(X_train))
print("Test size:", len(X_test))

# 2. TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000)  # limit vocab to top
5000 words
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

print("Vectorized shape (train):", X_train_vec.shape)
print("Vectorized shape (test):", X_test_vec.shape)
```

```
Train size: 20000
Test size: 5000
Vectorized shape (train): (20000, 5000)
Vectorized shape (test): (5000, 5000)
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# 1. Initialize model
model = LogisticRegression(max_iter=1000)

# 2. Train the model on training data
model.fit(X_train_vec, y_train)

# 3. Make predictions on test data
y_pred = model.predict(X_test_vec)

# 4. Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

```
Accuracy: 0.8802

Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.87      0.88      2515
           1       0.87      0.89      0.88      2485
```

```
    accuracy                           0.88      5000
   macro avg       0.88      0.88      0.88      5000
weighted avg       0.88      0.88      0.88      5000
```

```python
# Test with your own custom reviews
sample_reviews = [
    "I hated this movie, waste of time!",   # Positive
    # "The film was too long and very boring.",           # Negative
    # "An average movie, not too bad but not great either." # Neutral-
like
]

# Clean the reviews (using the same function as before)
sample_cleaned = [clean_text(r) for r in sample_reviews]

# Vectorize with the SAME vectorizer used earlier
sample_vec = vectorizer.transform(sample_cleaned)

# Predict
predictions = model.predict(sample_vec)

# Show results
for review, pred in zip(sample_reviews, predictions):
    sentiment = "Positive 😊" if pred == 1 else "Negative 😞"
    print(f"Review: {review}\nPredicted Sentiment: {sentiment}\n")
```

```
Review: I hated this movie, waste of time!
Predicted Sentiment: Negative 😞
```

```python
def predict_sentiment(review):
    # Step 1: Clean the input review
    cleaned = clean_text(review)

    # Step 2: Convert it into vector form using the same TF-IDF
vectorizer
    vec = vectorizer.transform([cleaned])

    # Step 3: Predict using the trained model
    pred = model.predict(vec)[0]

    # Step 4: Return the result
    if pred == 1:
        return "Positive 😊"
    else:
        return "Negative 😞"

# □ Test the function
print(predict_sentiment("I loved the acting, such a brilliant
```

```
movie!"))
print(predict_sentiment("Worst movie ever, I wasted my time."))
print(predict_sentiment("hey this movie was great"))
```

```
Positive 😊
Negative 😞
Positive 😊
```

```
import gradio as gr

def predict_sentiment(review):
    # Clean the text
    clean = clean_text(review)

    # Transform text into features
    features = vectorizer.transform([clean])

    # Predict class (0 = Negative, 1 = Positive)
    prediction = model.predict(features)[0]

    # Predict probability
    prob = model.predict_proba(features)[0]  # gives [prob_negative,
prob_positive]

    # Select probability of predicted class
    confidence = prob[prediction] * 100

    # Final readable output
    sentiment = "Positive 😊" if prediction == 1 else "Negative 😞"
    result = f"{sentiment} (Confidence: {confidence:.2f}%)"

    return result

# Create UI
ui = gr.Interface(
    fn=predict_sentiment,
    inputs=gr.Textbox(lines=5, placeholder="Type a movie review
here..."),
    outputs="text",
    title="□ Movie Review Sentiment Analyzer",
    description="Enter a movie review and see if it's Positive or
Negative with confidence score!"
)

ui.launch()
```

```
It looks like you are running Gradio on a hosted Jupyter notebook,
which requires `share=True`. Automatically setting `share=True` (you
can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set
```

```
debug=True in launch()
* Running on public URL: https://becb919aa002439a8a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)

<IPython.core.display.HTML object>
```