```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```python
import os

folder_path = '/content/drive/MyDrive/senti/sentiment-analysis-project/data/raw'
print("Files in folder:", os.listdir(folder_path))
```

```
Files in folder: ['IMDB Dataset.csv']
```

```python
import pandas as pd
data_path = '/content/drive/MyDrive/senti/sentiment-analysis-project/data/raw/IMDB Dataset.csv'  # update this
df = pd.read_csv(data_path)
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Next steps: Generate code with df View recommended plots New interactive sheet

```python
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```python
import pandas as pd
import nltk
nltk.download('stopwords')

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

# Reload the cleaned dataset
data_path = '/content/drive/MyDrive/senti/sentiment-analysis-project/data/raw/IMDB Dataset.csv'
df = pd.read_csv(data_path)

# Text cleaning functions (same as before)
def clean_text(text):
    text = text.lower()
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

def preprocess_text(text):
    text = clean_text(text)
    tokens = text.split()
    stop_words = set(stopwords.words('english'))
    tokens = [t for t in tokens if t not in stop_words and len(t) > 2]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(t) for t in tokens]
    return ' '.join(tokens)

# Apply preprocessing again in this notebook
df['processed_review'] = df['review'].apply(preprocess_text)
```

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

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Features and labels
X = df['processed_review']
y = df['sentiment'].map({'positive': 1, 'negative': 0})  # convert to 0 and 1

# Step 2: TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_vec = vectorizer.fit_transform(X)

# Step 3: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)

# Step 4: Train Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 5: Evaluate
y_pred = model.predict(X_test)
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
✅ Accuracy: 0.8861

 Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.87      0.88      4961
           1       0.88      0.90      0.89      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

```
import joblib

# Save the trained model
joblib.dump(model, 'model.pkl')

# Save the fitted vectorizer
joblib.dump(vectorizer, 'vectorizer.pkl')
```

```
['vectorizer.pkl']
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)  # Set max_iter to avoid convergence warning
model.fit(X_train, y_train)

print("✅ Model training complete!")
```

```
✅ Model training complete!
```

```
X = df['processed_review']
y = df['sentiment'].map({'positive': 1, 'negative': 0})
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Make predictions
y_pred = model.predict(X_test)

# Print accuracy
print("✅ Accuracy:", accuracy_score(y_test, y_pred))

# Print classification report
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred))
```

```
✅ Accuracy: 0.8861

📊 Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.89      0.87      0.88      4961
           1       0.88      0.90      0.89      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

```
import joblib
```

```
model_path = '/content/drive/MyDrive/sentiment-analysis-project/models/sentiment_model.pkl'
vectorizer_path = '/content/drive/MyDrive/sentiment-analysis-project/models/tfidf_vectorizer.pkl'
```

```
# Save the trained Logistic Regression model
joblib.dump(model, model_path)

# Save the TF-IDF vectorizer
joblib.dump(vectorizer, vectorizer_path)

print("✅ Model and vectorizer saved to Google Drive!")
```

```
✅ Model and vectorizer saved to Google Drive!
```

```
def predict_sentiment(review_text):
    # Preprocess the text (same way as training)
    text = preprocess_text(review_text)

    # Transform with saved TF-IDF vectorizer
    text_vector = vectorizer.transform([text])

    # Predict with trained model
    prediction = model.predict(text_vector)[0]

    # Show result
    return "Positive 😊" if prediction == 1 else "Negative 😞"

# 🔍 Try your own review
review = "The movie was surprisingly good with amazing acting!"
print("Sentiment Prediction:", predict_sentiment(review))
```

```
Sentiment Prediction: Positive 😊
```

```
sample = ["I absolutely loved the movie. The acting was fantastic!"]
sample_clean = [preprocess_text(sample[0])]
sample_vec = vectorizer.transform(sample_clean)
predicted = model.predict(sample_vec)
print("Predicted Sentiment:", "Positive" if predicted[0] == 1 else "Negative")
```

```
Predicted Sentiment: Positive
```

```
import ipywidgets as widgets
from IPython.display import display

text_input = widgets.Textarea(
    value='The movie was amazing!',
    placeholder='Type your own movie review here...',
    description='Your Review:',
    layout=widgets.Layout(width='100%', height='100px')
)

button = widgets.Button(description='Predict Sentiment')
output = widgets.Output()

def on_button_click(b):
    output.clear_output()
    with output:
        prediction = predict_sentiment(text_input.value)
        print("Predicted Sentiment:", prediction)

button.on_click(on_button_click)

display(text_input, button, output)
```

Your Review: The movie was amazing!

Predict Sentiment

```
Predicted Sentiment: Positive 😊
```

```
import joblib

# Save the model and vectorizer
joblib.dump(model, 'model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')
```

```
['vectorizer.pkl']
```