```
# Install necessary libraries (only need to run this once)
!pip install nltk scikit-learn
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
```

```
# Import all required libraries
import pandas as pd
import numpy as np
import re
import string
import nltk
from google.colab import files
```

```
# Import specific NLP and ML modules
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
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
```

```
# Download required NLTK data packages (only need to run this once)
# 'punkt' is for tokenization
# 'stopwords' is for the list of common stop words
# 'wordnet' is the lexical database for lemmatization
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[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 punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
# --- Load the Dataset ---
print("Please upload the 'IMDB Dataset.csv' file.")
uploaded = files.upload()
```

```
Please upload the 'IMDB Dataset.csv' file.
Choose Files IMDB Dataset.csv
IMDB Dataset.csv(text/csv) - 66262310 bytes, last modified: 11/13/2025 - 100% done
Saving IMDB Dataset.csv to IMDB Dataset.csv
```

```
# Get the filename and load it into a pandas DataFrame
filename = list(uploaded.keys())[0]
df = pd.read_csv(filename)
```

```
print("\nDataset loaded successfully!")
print("Shape of the dataset:", df.shape)
print("\nFirst 5 rows:")
print(df.head())
```

```

Dataset loaded successfully!
Shape of the dataset: (50000, 2)

First 5 rows:
                                              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
```

```
# Initialize our NLP tools
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
def preprocess_text(text):
    """
    This function defines our NLP pipeline to clean and prepare the text.
    """
    # NLP Task 1: Noise Removal (HTML Tags and Punctuation)
    # We remove elements that don't carry sentiment meaning.
    text = re.sub(r'<.*?>', ' ', text) # Remove HTML tags
    text = text.translate(str.maketrans('', '', string.punctuation)) # Remove punctuation

    # NLP Task 2: Normalization (Lowercase Conversion)
    # We convert all text to lowercase to treat words like "Good" and "good"
    # as the same word.
    text = text.lower()

    # NLP Task 3: Tokenization
    # We break down the continuous string of text into a list of individual
    # words, or "tokens". This is a foundational step for any further analysis.
    # Example: "this was great" -> ["this", "was", "great"]
    tokens = word_tokenize(text)

    # NLP Task 4: Stop Word Removal & Lemmatization
    # We process the list of tokens to further refine our vocabulary.
    cleaned_tokens = []
    for word in tokens:
        # Stop Word Removal: We remove extremely common words (e.g., 'the', 'a',
        # 'is', 'in') that add little value to sentiment analysis.
        if word not in stop_words and word.isalpha():
            # Lemmatization: We reduce each word to its base or dictionary form
            # (its "lemma"). This helps group related words.
            # Example: 'running', 'ran', 'runs' all become 'run'.
            lemmatized_word = lemmatizer.lemmatize(word)
            cleaned_tokens.append(lemmatized_word)

    # Join tokens back into a single, clean string
    return " ".join(cleaned_tokens)

print("\nStarting the NLP preprocessing pipeline...")
# Apply the NLP pipeline to the 'review' column.
# This may take a few minutes for 50,000 reviews.
df['cleaned_review'] = df['review'].apply(preprocess_text)

print("NLP preprocessing complete!")
print("\nExample of original vs. cleaned review:")
print("Original:", df['review'][0])
print("Cleaned:", df['cleaned_review'][0])
```

```

Starting the NLP preprocessing pipeline...
NLP preprocessing complete!

Example of original vs. cleaned review:
Original: One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, a
Cleaned: one reviewer mentioned watching oz episode youll hooked right exactly happened first thing struck oz brutality unfl
```

```
print("\nApplying TF-IDF vectorization to convert text into numerical features...")

# Initialize the TF-IDF Vectorizer
# We'll limit features to the top 5000 most significant words to keep it efficient.
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# Learn the vocabulary from our cleaned reviews and transform the text into a
# numerical matrix (X).
X = tfidf_vectorizer.fit_transform(df['cleaned_review'])

# Prepare our target variable (y).
# Convert the sentiment labels to numbers (0 for negative, 1 for positive).
df['sentiment_label'] = df['sentiment'].map({'positive': 1, 'negative': 0})
y = df['sentiment_label']

print("TF-IDF vectorization complete.")
print("Shape of the feature matrix (X):", X.shape)
print("This means we have 50,000 reviews, each represented by 5,000 numerical features.")
```

```

Applying TF-IDF vectorization to convert text into numerical features...
TF-IDF vectorization complete.
Shape of the feature matrix (X): (50000, 5000)
```

```
This means we have 50,000 reviews, each represented by 5,000 numerical features.
```

```
print("\nSplitting data and training the Logistic Regression model...")

# Split the data into 80% for training and 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Initialize and train the model on the numerical data generated by our NLP process
model = LogisticRegression(solver='liblinear', random_state=42)
model.fit(X_train, y_train)

# Make predictions on the unseen test data
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])

print("Model training complete.")
print(f"Model Accuracy: {accuracy:.4f}")
print("\nClassification Report:\n", report)
```

```
Splitting data and training the Logistic Regression model...
Model training complete.
Model Accuracy: 0.8880

Classification Report:
               precision    recall  f1-score   support

    Negative       0.90      0.88      0.89      5000
    Positive       0.88      0.90      0.89      5000

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

```
print("\nPreparing final dataset for export...")

# Add review length as a potentially interesting metric for the dashboard
df['review_length'] = df['review'].apply(len)

# Use our trained model to predict the sentiment for ALL reviews
# This allows us to compare the model's prediction with the original label.
df['predicted_sentiment_label'] = model.predict(X)
df['predicted_sentiment'] = df['predicted_sentiment_label'].map({1: 'positive', 0: 'negative'})

# Create a final export DataFrame with selected columns
export_df = df[['review', 'sentiment', 'predicted_sentiment', 'review_length']]

# Save the final DataFrame to a CSV file
output_filename = 'sentiment_analysis_results_for_power_bi.csv'
export_df.to_csv(output_filename, index=False)

print(f"\nSuccessfully created '{output_filename}'.")
print("This file is ready to be imported into Power BI.")

# Automatically download the file from Colab to your computer
files.download(output_filename)
```

```
Preparing final dataset for export...

Successfully created 'sentiment_analysis_results_for_power_bi.csv'.
This file is ready to be imported into Power BI.
```