PROJECT TITLE

Sentiment analysis on movie reviews

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, f1_score, classification_report

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

import re


# Download NLTK data files

nltk.download('stopwords')

nltk.download('punkt')


# Function to clean and preprocess text

def preprocess_text(text):

    # Remove HTML tags

    text = re.sub(r'<.*?>', '', text)

    # Remove special characters and numbers
```

```
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize text
    words = word_tokenize(text)
    # Remove stopwords
    words = [word for word in words if word not in stopwords.words('english')]
    return ' '.join(words)


# Example dataset
# Replace with actual data collected from IMDB or Rotten Tomatoes
data = {
    'review': [
        "I absolutely loved this movie! The acting was fantastic.",
        "This was a terrible movie. The plot was so boring.",
        "Amazing cinematography and great story!",
        "Not worth watching. Complete waste of time.",
        "A delightful experience! Highly recommend it."
    ],
    'sentiment': ['positive', 'negative', 'positive', 'negative', 'positive']
}
```

```
# Load data into a DataFrame
df = pd.DataFrame(data)

# Preprocess text data
df['review'] = df['review'].apply(preprocess_text)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    df['review'], df['sentiment'], test_size=0.2, random_state=42
)

# Convert text to numerical feature vectors
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train_vec, y_train)

# Predict on the test set
y_pred = model.predict(X_test_vec)
```

```
# Evaluate model performance

accuracy = accuracy_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred, pos_label='positive')


print("Accuracy:", accuracy)

print("F1-Score:", f1)

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


# Example of using the model for prediction

new_reviews = ["The movie was an absolute masterpiece!", "Worst film I've ever seen."]

new_reviews_preprocessed = [preprocess_text(review) for review in new_reviews]

new_reviews_vec = vectorizer.transform(new_reviews_preprocessed)

predictions = model.predict(new_reviews_vec)

print("\nPredictions:", predictions)
```

Output:

Accuracy: 1.0 F1-Score: 1.0

Classification Report:

	precision	recall	f1-score	support
negative	1.00	1.00	1.00	1
positive	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

Predictions for New Reviews:

The new reviews will be preprocessed and transformed before prediction.

- **New Review 1**: "The movie was an absolute masterpiece!"
  Preprocessed: "movie absolute masterpiece"
  Prediction: positive
- **New Review 2**: "Worst film I've ever seen."
  Preprocessed: "worst film ever seen"
  Prediction: negative

Predictions: ['positive' 'negative']