```
# Install required libraries
!pip install -q pandas scikit-learn matplotlib seaborn nltk

# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from nltk.corpus import stopwords
import re
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Download NLTK stopwords
nltk.download('stopwords', quiet=True) # Added quiet=True to suppress output
stop_words = set(stopwords.words('english'))



# Show the first few rows
print(df.head())

# Print the column names to identify the text column
print(df.columns) # Added this line to help the user identify the correct text column

# Check for missing values
print(df.isnull().sum())

# Basic text cleaning function
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # remove URLs
    text = re.sub(r'\@w+|\#','', text)  # remove mentions and hashtags
    text = re.sub(r'[^A-Za-z\s]', '', text)  # remove punctuation and numbers
    # Ensure stop_words is defined when this function is called. It is defined globally above.
    text = ' '.join(word for word in text.split() if word not in stop_words)
    return text

# Apply cleaning
# IMPORTANT: Replace '<insert_actual_text_column_name_here>' with the actual column
name from df.columns that contains your text data.
# For example, if your text column is named 'TweetContent', change the line to:
```

```
# df['clean_text'] = df['TweetContent'].apply(clean_text)
# Based on the original code, 'text' is assumed to be the column name. Let's use 'text' as a placeholder, but the user should verify this with df.columns.
# *** YOU NEED TO REPLACE '<insert_actual_text_column_name_here>' BELOW WITH THE CORRECT COLUMN NAME ***
if '<insert_actual_text_column_name_here>' in df.columns:
    df['clean_text'] = df['<insert_actual_text_column_name_here>'].apply(clean_text)
else:
    print("Error: The specified text column was not found. Please inspect df.columns and update the code with the correct column name.")
    # You might want to add a more robust error handling or exit here
    # For now, we will assume 'text' is the column and proceed, but be aware this might fail if it's not.


# Vectorization, Model Training, Prediction, and Evaluation (including Confusion Matrix)
# Check if 'clean_text' column was successfully created before proceeding
if 'clean_text' in df.columns:
    vectorizer = TfidfVectorizer(max_features=5000)
    X = vectorizer.fit_transform(df['clean_text']).toarray()
    y = df['sentiment'] # Assuming 'sentiment' is the target column name

    # Split dataset
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train a classifier
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # Predict
    y_pred = model.predict(X_test)

    # Print classification report
    print(classification_report(y_test, y_pred))

    # Generate the confusion matrix
    conf_matrix = confusion_matrix(y_test, y_pred)

    # Plot the heatmap
    plt.figure(figsize=(6, 5))
    # Ensure xticklabels and yticklabels use the unique values from the actual target variable y
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='coolwarm', xticklabels=np.unique(y), yticklabels=np.unique(y))
    plt.title('Confusion Matrix Heatmap')
```

```
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.tight_layout()
    plt.show()
else:
    print("Skipping vectorization, model training, prediction, and confusion matrix generation as the text column was not found or cleaning failed.")

# Removed the duplicate confusion matrix plotting code here

# Generate a Heatmap of Sentiment Counts
sentiment_counts = df['Sentiment'].value_counts().to_frame().T

plt.figure(figsize=(8, 3))
sns.heatmap(sentiment_counts, annot=True, cmap='coolwarm', fmt='d')
plt.title('Heatmap of Sentiment Categories')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```