```
# Install necessary libraries
!pip install scikit-learn pandas

# Import libraries
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# 1. Load and preprocess the dataset
try:
    # Load dataset from 'train.csv' and handle encoding issues
    df = pd.read_csv('/content/train.csv', encoding='utf-8')
except UnicodeDecodeError:
    # Fallback in case of encoding errors
    df = pd.read_csv('/content/train.csv', encoding='latin1')

# Ensure the dataset has the necessary columns
if 'text' not in df.columns or 'sentiment' not in df.columns:
    raise ValueError("The dataset must contain 'text' and 'sentiment' columns.")

# Drop rows with missing values in 'text' or 'sentiment'
df.dropna(subset=['text', 'sentiment'], inplace=True)

# Split data into features and labels
X = df['text']
y = df['sentiment']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 2. Train a Sentiment Analysis Model
# Convert text into numerical data using CountVectorizer
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train a Naive Bayes Classifier
model = MultinomialNB()
model.fit(X_train_vec, y_train)

# Evaluate the model
y_pred = model.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# 3. Chatbot Simulation
print("\nSentiment Analysis Chatbot is ready! Type 'exit' to end the chat.")

while True:
    # Take user input
    user_input = input("You: ")

    # Exit condition
    if user_input.lower() == 'exit':
        print("Chatbot: Goodbye!")
        break

    # Predict sentiment
    input_vec = vectorizer.transform([user_input])  # Transform input text to match the model
    sentiment = model.predict(input_vec)[0]

    # Respond based on sentiment
    if sentiment.lower() == 'positive':  # Check sentiment case-insensitively
        response = "That sounds great!"
    elif sentiment.lower() == 'negative':
        response = "I'm sorry to hear that."
    else:
        response = "Thanks for sharing!"

    print(f"Chatbot: {response}")
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
```

```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Model Accuracy: 64.72%

Sentiment Analysis Chatbot is ready! Type 'exit' to end the chat.
You: good day!
Chatbot: That sounds great!
You:
```