Source code:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab') # Download the punkt_tab resource
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix

# ----------------------------------
# STEP 2: Load Dataset
# ----------------------------------

# ----------------------------------
# STEP 3: Preprocessing
# ----------------------------------
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words("english"))

def preprocess(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    tokens = word_tokenize(text)
    filtered = [w for w in tokens if w not in stop_words]
    return " ".join(filtered)

df["clean_query"] = df["query"].apply(preprocess)
df.head()

# ----------------------------------
# STEP 4: Exploratory Data Analysis (EDA)
# ----------------------------------
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x="intent", order=df["intent"].value_counts().index)
```

```
plt.xticks(rotation=90)
plt.title("Distribution of Intents")
plt.show()

df["query_length"] = df["query"].apply(lambda x: len(x.split()))
plt.hist(df["query_length"], bins=15, color='skyblue', edgecolor='black')
plt.title("Query Length Distribution")
plt.xlabel("Number of Words")
plt.ylabel("Frequency")
plt.show()

# ----------------------------------
# STEP 5: Feature Encoding and Scaling
# ----------------------------------
# Not needed explicitly here since we use TF-IDF within a pipeline.

# ----------------------------------
# STEP 6: Model Training (Logistic Regression)
# ----------------------------------
X = df["clean_query"]
y = df["intent"]

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

pipeline = Pipeline([
    ("tfidf", TfidfVectorizer()),
    ("clf", LogisticRegression(max_iter=200))
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)

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

# Confusion Matrix
plt.figure(figsize=(12, 8))
cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
sns.heatmap(cm, annot=True, fmt="d", xticklabels=pipeline.classes_,
yticklabels=pipeline.classes_, cmap="Blues")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title("Confusion Matrix")
plt.show()

# ----------------------------------
# STEP 7: Chatbot Demo
# ----------------------------------
```

```
intent_response_map =
df.drop_duplicates(subset=["intent"]).set_index("intent")["response"].to_dict()

def chatbot_response(user_input):
    processed_input = preprocess(user_input)
    predicted_intent = pipeline.predict([processed_input])[0]
    return intent_response_map.get(predicted_intent, "I'm sorry, I didn't understand that.")

# Try it!
while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Bot: Goodbye!")
        break
    print("Bot:", chatbot_response(user_input
```