**1. Upload the Dataset**

from google.colab import files

uploaded = files.upload()

**Step 2: Load the Dataset**

import pandas as pd

import json

# Load the JSON file (adjust name if different)

with open("chatbot\_data.json", "r") as file:

data = json.load(file)

# Convert to DataFrame

df = pd.json\_normalize(data)

df.head()

**Step 3: Data Exploration**

# Overview of the data

df.info()

df.describe()

df.head()

**Step 4: Check for Missing Values and Duplicates**

# Check missing values

print(df.isnull().sum())

# Check duplicates

print(f"Duplicates: {df.duplicated().sum()}")

**Step 5: Visualize a Few Features**

import matplotlib.pyplot as plt

import seaborn as sns

# Example: Visualize distribution of a column (change 'column\_name')

sns.countplot(x='tag', data=df)

plt.title("Distribution of Tags")

plt.xticks(rotation=45)

plt.show()

**Step 6: Identify Target and Features**

# Assuming 'tag' is the label and 'patterns' are features

X = df['patterns']

y = df['tag']

**Step 7: Convert Categorical Columns to Numerical (if needed)**

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

**Step 8: One-Hot Encoding (optional if needed for multiple classes)**

from sklearn.preprocessing import OneHotEncoder

onehot\_encoder = OneHotEncoder(sparse=False)

y\_onehot = onehot\_encoder.fit\_transform(y\_encoded.reshape(-1, 1))

**Step 9: Feature Scaling (only if features are numeric)**

# Skip if working with text data

**Step 10: Train-Test Split**

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

**Step 11: Model Building (Example: Text Classification using TF-IDF + Naive Bayes)**

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import make\_pipeline

model = make\_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(X\_train, y\_train)

**Step 12: Evaluation**

from sklearn.metrics import classification\_report, accuracy\_score

y\_pred = model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

**Step 13: Make Predictions from New Input**

new\_input = ["hello, how can I help you?"]

prediction = model.predict(new\_input)

predicted\_label = label\_encoder.inverse\_transform(prediction)

print("Predicted Tag:", predicted\_label[0])

**Step 14: Convert to DataFrame and Encode (if working with new raw data)**

# For new raw input data

new\_df = pd.DataFrame({'patterns': ["hi, can you help me?"], 'tag': ["greeting"]})

new\_df['tag\_encoded'] = label\_encoder.transform(new\_df['tag'])

new\_df

**Step 15: Predict the Final Grade (custom logic if needed)**

# If you have a grading system based on tag or prediction

# Example:

if predicted\_label[0] == "greeting":

print("Grade: A")

**Step 16: Deployment - Building an Interactive App**

!pip install gradio

import gradio as gr

**Step 17: Create a Prediction Function**

def chatbot\_response(text):

prediction = model.predict([text])

return label\_encoder.inverse\_transform(prediction)[0]

**Step 18: Create the Gradio Interface**

interface = gr.Interface(fn=chatbot\_response, inputs="text", outputs="text", title="Chatbot Tag Predictor")

interface.launch()

**Step 19: Customer Service Chatbot**

intents = {item['tag']: item['responses'][0] for item in data['intents']}

def chatbot\_full\_response(text):

tag = model.predict([text])[0]

tag = label\_encoder.inverse\_transform([tag])[0]

return intents.get(tag, "Sorry, I didn’t understand that.")

interface = gr.Interface(fn=chatbot\_full\_response, inputs="text", outputs="text", title="Customer Service Chatbot")

interface.launch()