import gradio as gr

import pandas as pd

import joblib

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.linear\_model import LogisticRegression

# Load the dataset

df = pd.read\_csv('/content/customer\_support\_dataset.csv')

# Assuming 'Response' is your target column

target\_column = 'Response'

# 1. Separate features and target

X = df.drop(columns=[target\_column])

y = df[target\_column]

# 2. Encode the target variable using LabelEncoder

label\_encoder = LabelEncoder()

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

# 3. One-hot encode categorical features in X

categorical\_features = X.select\_dtypes(include=['object']).columns

X\_encoded = pd.get\_dummies(X, columns=categorical\_features, drop\_first=True)

# 4. Scale numerical features in X\_encoded (if any)

# Check if there are numerical features before scaling

if len(X\_encoded.select\_dtypes(include=['number']).columns) > 0:

    numerical\_features = X\_encoded.select\_dtypes(include=['number']).columns

    scaler = StandardScaler()

    X\_encoded[numerical\_features] = scaler.fit\_transform(X\_encoded[numerical\_features])

# 5. Train-test split using the encoded data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X\_encoded, y\_encoded, test\_size=0.2, random\_state=42)

# 6. Model building and training

model = LogisticRegression()

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

# Save model components using joblib

joblib.dump(model, "g3\_model.pkl")

joblib.dump(scaler, "scaler.pkl")  # Assuming you have a scaler

joblib.dump(X\_encoded.columns, "model\_columns.pkl")

def predict\_g3(query):

    # Create a dictionary from inputs

    student = {"Query": query}  # Adjust to your actual input features

    # Convert to DataFrame and preprocess

    df = pd.DataFrame([student])

    df\_encoded = pd.get\_dummies(df, drop\_first=True)  # One-hot encoding

    # Align with training data columns

    df\_aligned = df\_encoded.reindex(columns=columns, fill\_value=0)

    df\_scaled = scaler.transform(df\_aligned)  # Scaling

    # Predict and return

    prediction = model.predict(df\_scaled)[0]

    # Inverse transform to get original label

    predicted\_response = label\_encoder.inverse\_transform([prediction])[0]

    return predicted\_response

# Load model, scaler, and training column names

model = joblib.load("g3\_model.pkl")

try:

    scaler = joblib.load("scaler.pkl")

except FileNotFoundError:

    scaler = StandardScaler()  # Create a new scaler if file not found

columns = joblib.load("model\_columns.pkl")

# Gradio Interface

iface = gr.Interface(

    fn=predict\_g3,

    inputs=gr.Textbox(lines=2, placeholder="Enter your customer query here..."),

    outputs="text",

    title="Customer Support Response Predictor",

    description="Enter a customer query to get a predicted response.",

)

iface.launch()