ï»¿# -\*- coding: utf-8 -\*-

"""Copy of sustainable smart city assistant using IBM granite LLM

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1GFjDV2SHpCe\_hrZeGMZZlcJhWDQ4SYnA

"""

!pip install gradio transformers torch scikit-learn pandas numpy

!pip install accelerate

import gradio as gr

from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from io import StringIO

from gradio.themes.base import Base

from gradio.themes.utils import colors, fonts

import torch

# GPU Check (Optional Debug Info)

print("âœ… Model loading... GPU available:", torch.cuda.is\_available())

# Custom theme

custom\_theme = Base(

primary\_hue=colors.green,

font=fonts.GoogleFont("Poppins")

)

# Load IBM Granite model

model\_name = "ibm-granite/granite-3.3-2b-instruct"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(

model\_name,

device\_map="auto",

torch\_dtype=torch.float16 # Faster inference on GPU

)

llm = pipeline("text-generation", model=model, tokenizer=tokenizer)

# Module 1: Policy Summarization

def policy\_summarizer\_v2(text, file):

if file is not None:

content = file.read().decode("utf-8")

elif text.strip():

content = text.strip()

else:

return "âš  Please upload a file or paste some text."

prompt = f"Summarize the following city policy in simple terms:\n{content}\nSummary:"

result = llm(prompt, max\_new\_tokens=100)[0]["generated\_text"]

return result.replace(prompt, "").strip()

# Module 2: Citizen Feedback

def citizen\_feedback(issue):

return f"âœ… Thank you! Your issue '{issue}' has been logged and categorized appropriately."

# Module 3: KPI Forecasting

def kpi\_forecasting(csv\_file):

df = pd.read\_csv(csv\_file.name)

X = df.iloc[:, 0].values.reshape(-1, 1)

y = df.iloc[:, 1].values

model = LinearRegression().fit(X, y)

next\_year = [[X[-1][0] + 1]]

prediction = model.predict(next\_year)[0]

return f"ðŸ“ˆ Predicted KPI for {next\_year[0][0]}: {round(prediction, 2)}"

# Module 4: Eco Tips Generator

def eco\_tips(keyword):

prompt = f"Give 3 actionable eco-friendly tips related to: {keyword}"

result = llm(prompt, max\_new\_tokens=100)[0]["generated\_text"]

return result.replace(prompt, "").strip()

# Module 5: Anomaly Detection

def detect\_anomaly(csv\_file):

df = pd.read\_csv(csv\_file.name)

if 'value' not in df.columns:

return "âš  CSV must contain a 'value' column."

mean = df["value"].mean()

std = df["value"].std()

anomalies = df[np.abs(df["value"] - mean) > 2 \* std]

if anomalies.empty:

return "âœ… No significant anomalies detected."

return "âš  Anomalies found:\n" + anomalies.to\_string(index=False)

# Module 6: Chat Assistant

def chat\_assistant(question):

prompt = f"Answer this smart city sustainability question:\n\nQ: {question}\nA:"

result = llm(prompt, max\_new\_tokens=100, temperature=0.7)[0]["generated\_text"]

return result.replace(prompt, "").strip()

# Module 7: Carbon Footprint Calculator

def carbon\_footprint(distance\_km, electricity\_kwh, waste\_kg):

"""

Simple estimation of carbon emissions:

- Transport: ~0.12 kg CO2 per km (average car)

- Electricity: ~0.82 kg CO2 per kWh (average grid)

- Waste: ~0.5 kg CO2 per kg of waste

"""

transport\_emission = distance\_km \* 0.12

electricity\_emission = electricity\_kwh \* 0.82

waste\_emission = waste\_kg \* 0.5

total = transport\_emission + electricity\_emission + waste\_emission

return (f"ðŸŒ Estimated Carbon Footprint:\n"

f"- Transport: {transport\_emission:.2f} kg COâ‚‚\n"

f"- Electricity: {electricity\_emission:.2f} kg COâ‚‚\n"

f"- Waste: {waste\_emission:.2f} kg COâ‚‚\n"

f"\*Total: {total:.2f} kg COâ‚‚ per day\*")

# Gradio App UI

with gr.Blocks(theme=custom\_theme) as app:

gr.Markdown("## ðŸŒ† Sustainable Smart City Assistant")

gr.Markdown("Built with IBM Granite LLM ðŸ§  to empower urban planning, feedback, sustainability, and innovation.")

with gr.Tabs():

with gr.Tab("ðŸ“ Policy Summarization"):

with gr.Column():

gr.Markdown("Upload a .txt file or paste policy text to generate a summary.")

with gr.Row():

policy\_file = gr.File(label="Upload .txt File", file\_types=[".txt"])

policy\_text = gr.Textbox(label="Or paste policy text", lines=10)

policy\_output = gr.Textbox(label="Summary", lines=5)

summarize\_btn = gr.Button("Summarize")

summarize\_btn.click(policy\_summarizer\_v2, inputs=[policy\_text, policy\_file], outputs=policy\_output)

with gr.Tab("ðŸ“£ Citizen Feedback"):

feedback\_input = gr.Textbox(lines=3, label="Describe the Issue")

feedback\_output = gr.Textbox(label="Acknowledgement")

feedback\_btn = gr.Button("Submit Feedback")

feedback\_btn.click(citizen\_feedback, inputs=feedback\_input, outputs=feedback\_output)

with gr.Tab("ðŸ“Š KPI Forecasting"):

kpi\_input = gr.File(label="Upload KPI CSV")

kpi\_output = gr.Textbox(label="Forecast Result")

kpi\_btn = gr.Button("Forecast KPI")

kpi\_btn.click(kpi\_forecasting, inputs=kpi\_input, outputs=kpi\_output)

with gr.Tab("ðŸŒ± Eco Tips Generator"):

tip\_input = gr.Textbox(label="Keyword (e.g. Plastic, Solar)")

tip\_output = gr.Textbox(label="Generated Tips")

tip\_btn = gr.Button("Get Eco Tips")

tip\_btn.click(eco\_tips, inputs=tip\_input, outputs=tip\_output)

with gr.Tab("ðŸš¨ Anomaly Detection"):

anomaly\_input = gr.File(label="Upload CSV with 'value' column")

anomaly\_output = gr.Textbox(label="Anomaly Results")

anomaly\_btn = gr.Button("Detect Anomalies")

anomaly\_btn.click(detect\_anomaly, inputs=anomaly\_input, outputs=anomaly\_output)

with gr.Tab("ðŸ’¬ Chat Assistant"):

chat\_input = gr.Textbox(label="Ask your question")

chat\_output = gr.Textbox(label="Assistant Response")

chat\_btn = gr.Button("Ask")

chat\_btn.click(chat\_assistant, inputs=chat\_input, outputs=chat\_output)

with gr.Tab("ðŸŒ Carbon Footprint Calculator"):

gr.Markdown("Estimate your daily carbon emissions from travel, electricity, and waste.")

with gr.Row():

distance\_input = gr.Number(label="Distance Traveled (km per day)", value=10)

electricity\_input = gr.Number(label="Electricity Used (kWh per day)", value=5)

waste\_input = gr.Number(label="Waste Produced (kg per day)", value=2)

footprint\_output = gr.Textbox(label="Estimated Footprint")

footprint\_btn = gr.Button("Calculate Footprint")

footprint\_btn.click(carbon\_footprint,

inputs=[distance\_input, electricity\_input, waste\_input],

outputs=footprint\_output)

app.launch(share=True)