import streamlit as st

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.cluster import KMeans

# -----------------------------

# Generate synthetic dataset

# -----------------------------

np.random.seed(42)

n = 1000

distance = np.random.randint(1, 200, n)              # km per month

transport = np.random.choice(["Car", "Bus", "Train", "Bike"], n)

electricity = np.random.randint(50, 500, n)          # kWh per month

diet = np.random.choice(["Veg", "Non-veg"], n)

waste = np.random.randint(5, 50, n)                  # kg per month

transport\_factors = {"Car": 0.21, "Bus": 0.08, "Train": 0.05, "Bike": 0.02}

diet\_factors = {"Veg": 100, "Non-veg": 200}

footprint = (

    distance \* [transport\_factors[t] for t in transport] +

    electricity \* 0.85 +

    [diet\_factors[d] for d in diet] +

    waste \* 2.5

)

df = pd.DataFrame({

    "distance": distance,

    "transport": transport,

    "electricity": electricity,

    "diet": diet,

    "waste": waste,

    "footprint": footprint

})

# Encode categorical variables

le\_transport = LabelEncoder()

df["transport\_enc"] = le\_transport.fit\_transform(df["transport"])

le\_diet = LabelEncoder()

df["diet\_enc"] = le\_diet.fit\_transform(df["diet"])

X = df[["distance", "transport\_enc", "electricity", "diet\_enc", "waste"]]

y = df["footprint"]

# -----------------------------

# Train models

# -----------------------------

# Regression model

reg\_model = LinearRegression()

reg\_model.fit(X, y)

# Classification model

df["category"] = pd.cut(df["footprint"],

                        bins=[0, 300, 600, np.inf],

                        labels=["Low", "Medium", "High"])

clf = RandomForestClassifier(random\_state=42)

clf.fit(X, df["category"])

# Clustering model

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10)

df["cluster"] = kmeans.fit\_predict(X\_scaled)

# -----------------------------

# Streamlit UI

# -----------------------------

st.title("🌍 Carbon Footprint Calculator")

st.markdown("Enter your lifestyle details and let AI/ML models predict your \*\*carbon footprint\*\* 🚀")

# User inputs

distance\_in = st.slider("🚗 Distance travelled per month (km)", 1, 500, 50)

transport\_in = st.selectbox("🚌 Transport mode", ["Car", "Bus", "Train", "Bike"])

electricity\_in = st.slider("💡 Electricity usage (kWh/month)", 50, 1000, 200)

diet\_in = st.radio("🍽️ Diet type", ["Veg", "Non-veg"])

waste\_in = st.slider("🗑️ Waste generated (kg/month)", 1, 100, 20)

# Encode inputs

transport\_enc = le\_transport.transform([transport\_in])[0]

diet\_enc = le\_diet.transform([diet\_in])[0]

user\_data = [[distance\_in, transport\_enc, electricity\_in, diet\_enc, waste\_in]]

# Predictions

if st.button("🔮 Predict My Footprint"):

    footprint\_pred = reg\_model.predict(user\_data)[0]

    category\_pred = clf.predict(user\_data)[0]

    cluster\_pred = kmeans.predict(scaler.transform(user\_data))[0]

    st.subheader("📊 Results")

    st.write(f"\*\*Estimated Carbon Footprint:\*\* {footprint\_pred:.2f} kg CO₂/month")

    st.write(f"\*\*Category:\*\* {category\_pred}")

    st.write(f"\*\*Cluster (lifestyle group):\*\* {cluster\_pred}")

    # Simple suggestions

    st.subheader("🌱 Suggestions to Reduce Footprint")

    if transport\_in == "Car" and distance\_in > 100:

        st.write("- Try using public transport or carpooling to reduce emissions.")

    if electricity\_in > 400:

        st.write("- Consider switching to energy-efficient appliances or reducing electricity use.")

    if diet\_in == "Non-veg":

        st.write("- Reducing meat consumption can significantly lower your footprint.")

    if waste\_in > 30:

        st.write("- Recycle or compost to reduce waste-related emissions.")