import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

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

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

import time

# 1. Simulate Water Level Sensor Data

def simulate\_sensor\_data(n=500):

np.random.seed(42)

rainfall = np.random.randint(0, 200, n) # mm

drain\_flow = np.random.randint(50, 500, n) # L/s

elevation = np.random.choice([1, 2, 3], size=n) # relative elevation

blockage = np.random.choice([0, 1], size=n, p=[0.8, 0.2]) # 0 = no blockage

# Label: 1 = Flood risk, 0 = No risk

flood\_risk = ((rainfall > 120) & (drain\_flow < 150) & (blockage == 1)).astype(int)

return pd.DataFrame({

'Rainfall\_mm': rainfall,

'DrainFlow\_Lps': drain\_flow,

'Elevation': elevation,

'Blockage': blockage,

'FloodRisk': flood\_risk

})

# 2. Train AI Model

def train\_model(df):

X = df[['Rainfall\_mm', 'DrainFlow\_Lps', 'Elevation', 'Blockage']]

y = df['FloodRisk']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

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

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

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

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

return model

# 3. Real-time Simulation and Prediction

def simulate\_real\_time(model, steps=10, delay=1):

print("\nStarting real-time simulation...\n")

for i in range(steps):

rainfall = np.random.randint(0, 200)

drain\_flow = np.random.randint(50, 500)

elevation = np.random.choice([1, 2, 3])

blockage = np.random.choice([0, 1], p=[0.8, 0.2])

features = [[rainfall, drain\_flow, elevation, blockage]]

prediction = model.predict(features)[0]

print(f"Time {i+1}s | Rainfall: {rainfall} mm | Drain Flow: {drain\_flow} L/s | "

f"Blockage: {blockage} | Flood Risk: {'YES' if prediction else 'NO'}")

time.sleep(delay)

# 4. Run All

if \_\_name\_\_ == "\_\_main\_\_":

df = simulate\_sensor\_data()

model = train\_model(df)

simulate\_real\_time(model, steps=10, delay=0.5)