```
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)
```