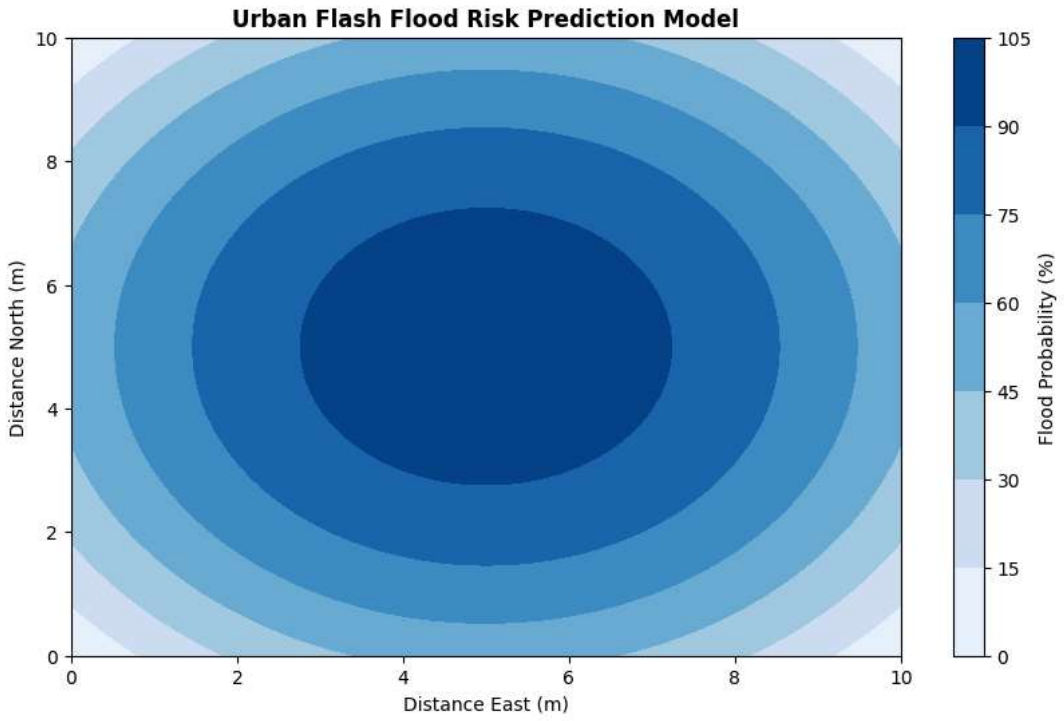
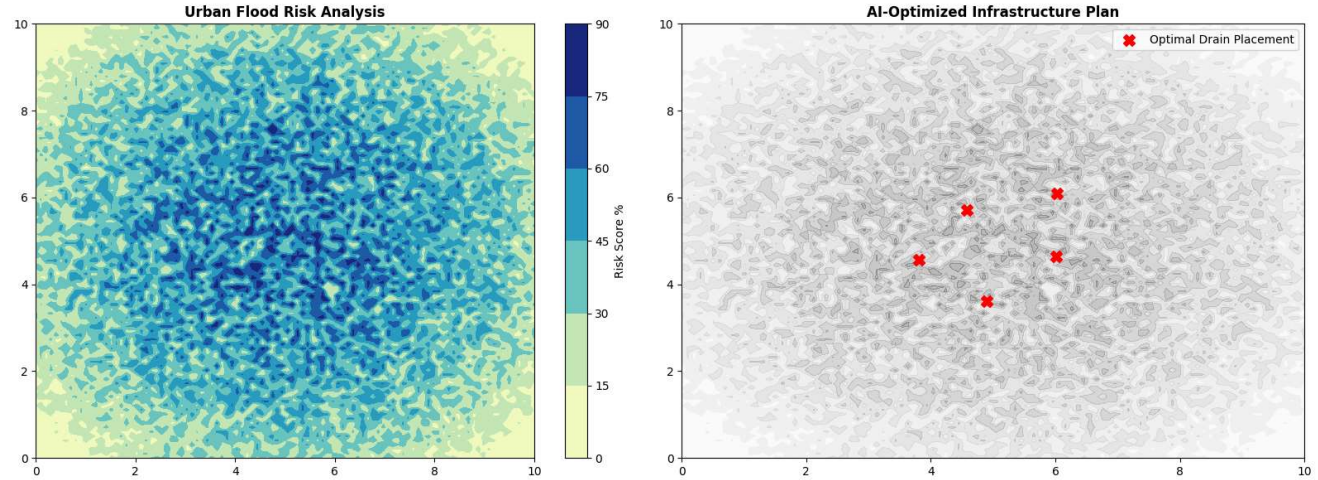
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# 1. Simulate Urban Terrain (Elevation Data)
# Imagine this is a 100x100 meter area of a city
grid_size = 100
x, y = np.meshgrid(np.linspace(0, 10, grid_size), np.linspace(0, 10, grid_size))
# Create a 'valley' shape where water would naturally collect
elevation = (x - 5)**2 + (y - 5)**2

# 2. Add 'Rainfall' and calculate Flood Risk
# Risk increases as elevation decreases
flood_risk = 100 - (elevation * 2)
flood_risk = np.clip(flood_risk, 0, 100) # Keep risk between 0-100%

# 3. Professional Visualization
plt.figure(figsize=(10, 6))
plt.contourf(x, y, flood_risk, cmap='Blues')
plt.colorbar(label='Flood Probability (%)')
plt.title('Urban Flash Flood Risk Prediction Model', fontweight='bold')
plt.xlabel('Distance East (m)')
plt.ylabel('Distance North (m)')
plt.show()

print("Project Started: Model successfully identified low-lying 'High Risk' zones.")
```




```
Project Started: Model successfully identified low-lying 'High Risk' zones.
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# --- STEP 1: TERRAIN & SATELLITE (NDVI) SIMULATION ---
grid_size = 100
x, y = np.meshgrid(np.linspace(0, 10, grid_size), np.linspace(0, 10, grid_size))

# Simulate elevation (Lower values = valleys/drains)
elevation = (x - 5)**2 + (y - 5)**2

# Simulate Satellite NDVI (Greenery vs Concrete)
# Lower NDVI (0.1) = Concrete/High Flood Risk | Higher (0.8) = Park/Low Risk
ndvi = np.random.uniform(0.1, 0.8, (grid_size, grid_size))

# --- STEP 2: CALCULATE MULTI-LAYER FLOOD RISK ---
# Logic: Risk is high where elevation is LOW and NDVI is LOW (Concrete)
```

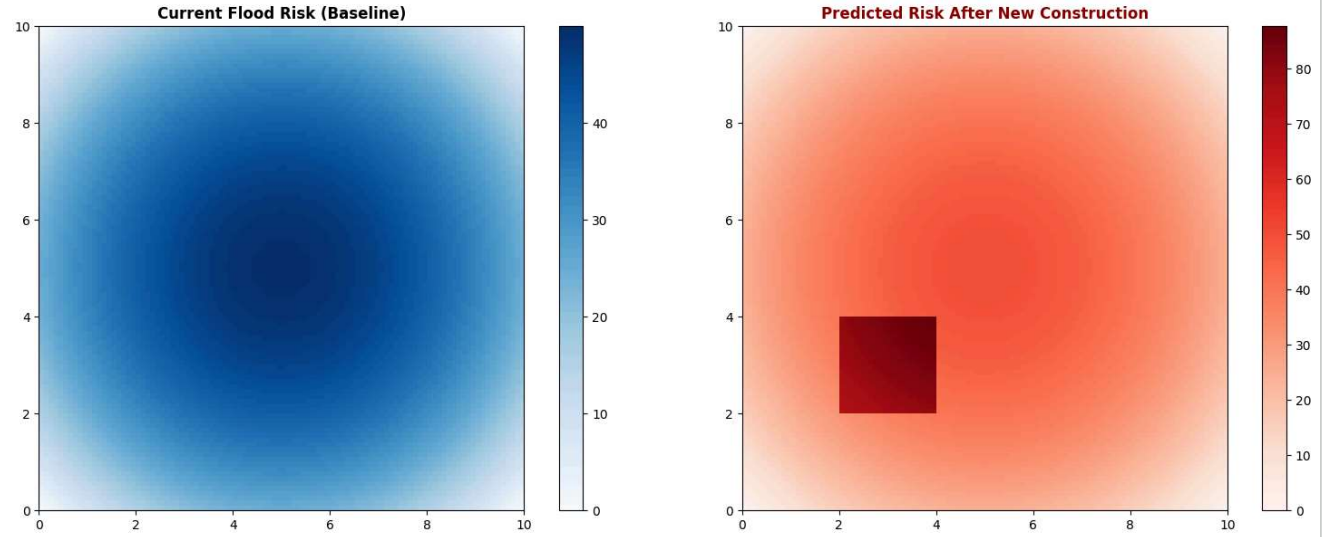
```
flood_risk = (100 - elevation*2) * (1 - ndvi)
flood_risk = np.clip(flood_risk, 0, 100)

# --- STEP 3: AI LOGIC (OPTIMAL DRAINAGE PLACEMENT) ---
# We find the coordinates of the "Highest Risk" points
high_risk_coords = np.argwhere(flood_risk > 80)

# Use K-Means to find 5 "Optimal Centers" for new drainage nodes
kmeans = KMeans(n_clusters=5, n_init=10).fit(high_risk_coords)
drainage_spots = kmeans.cluster_centers_

# --- STEP 4: DASHBOARD VISUALIZATION ---
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# Map 1: Flood Risk Heatmap
im1 = ax[0].contourf(x, y, flood_risk, cmap='YlGnBu')
ax[0].set_title('Urban Flood Risk Analysis', fontweight='bold')
plt.colorbar(im1, ax=ax[0], label='Risk Score %')

# Map 2: AI-Suggested Drainage Solutions
ax[1].contourf(x, y, flood_risk, alpha=0.3, cmap='Greys')
ax[1].scatter(drainage_spots[:, 1]/10, drainage_spots[:, 0]/10,
            color='red', s=100, marker='X', label='Optimal Drain Placement')
ax[1].set_title('AI-Optimized Infrastructure Plan', fontweight='bold')
ax[1].legend()

plt.tight_layout()
plt.show()

print(f"AI Analysis Complete: Identified {len(drainage_spots)} critical drainage nodes.")
```




```
AI Analysis Complete: Identified 5 critical drainage nodes.
```

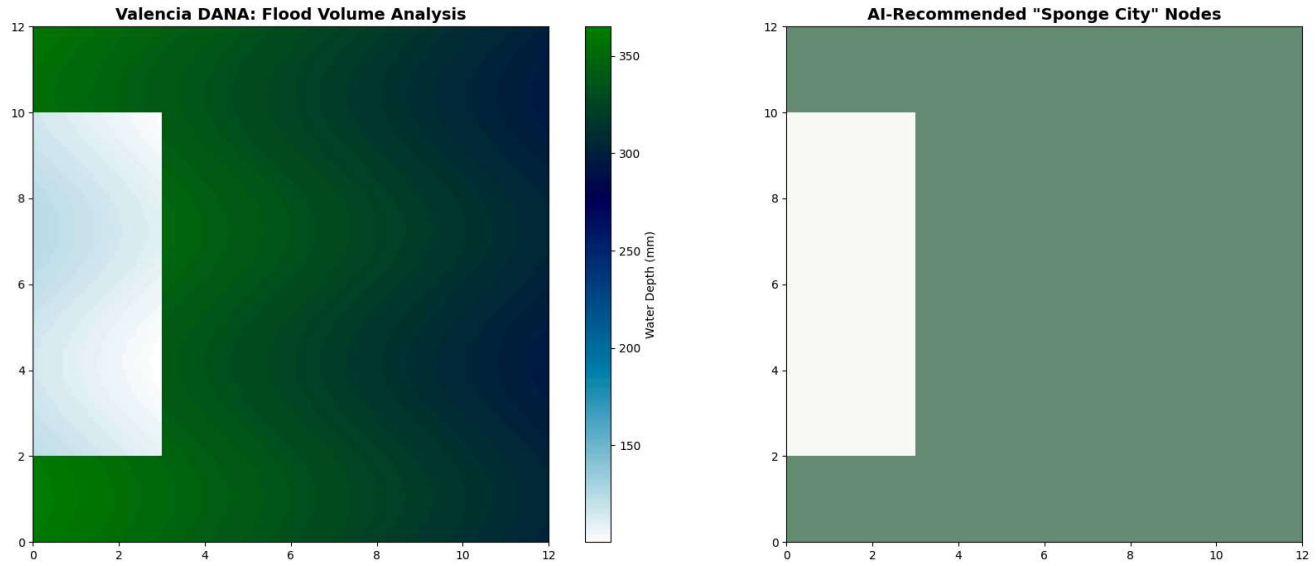
```
import numpy as np
import matplotlib.pyplot as plt

# 1. Setup the City Grid
grid_size = 100
x, y = np.meshgrid(np.linspace(0, 10, grid_size), np.linspace(0, 10, grid_size))
elevation = (x - 5)**2 + (y - 5)**2

# 2. Baseline Scenario (Standard City)
baseline_risk = (100 - elevation * 2) * 0.5

# 3. NEW DEVELOPMENT Scenario (Adding a large Concrete Plaza)
# We 'pave over' a square in the city (from x=2 to 4 and y=6 to 8)
new_development_impact = np.zeros((grid_size, grid_size))
new_development_impact[60:80, 20:40] = 40 # Increased runoff score

# 4. Final Result (Combined Effect)
future_risk = baseline_risk + new_development_impact

# --- Visualization ---
```

```
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# Map A: Before Development
im1 = ax[0].imshow(baseline_risk, cmap='Blues', extent=[0,10,0,10])
ax[0].set_title('Current Flood Risk (Baseline)', fontweight='bold')
plt.colorbar(im1, ax=ax[0])

# Map B: After Development (The 'Warning' Map)
im2 = ax[1].imshow(future_risk, cmap='Reds', extent=[0,10,0,10])
ax[1].set_title('Predicted Risk After New Construction', fontweight='bold', color='darkred')
plt.colorbar(im2, ax=ax[1])

plt.tight_layout()
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt

# 1. Create a simplified grid of Valencia's coastal topography
grid_size = 120
x, y = np.meshgrid(np.linspace(0, 12, grid_size), np.linspace(0, 12, grid_size))

# Elevation: High mountains to the West (left), Low coast to the East (right)
valencia_topo = (x * 0.5) + (np.sin(y) * 0.5)

# 2. Simulate Extreme Rainfall (DANA Event)
# 400mm of rain falling in a short window
rainfall_intensity = 400

# 3. Model Concrete vs. Green Space
# 80% of the city center is concrete (0% absorption)
absorption_rate = np.full((grid_size, grid_size), 0.1)
# Add a 'Green Belt' simulation (higher absorption)
absorption_rate[20:100, 0:30] = 0.7

# 4. Calculate "Runoff Volume" (Flood Intensity)
flood_volume = rainfall_intensity * (1 - absorption_rate) - (valencia_topo * 10)
flood_volume = np.clip(flood_volume, 0, None)

# --- THE PROFESSIONAL DASHBOARD ---
fig, ax = plt.subplots(1, 2, figsize=(18, 7))

# Map 1: The Disaster (Flood Intensity)
im1 = ax[0].imshow(flood_volume, cmap='ocean_r', extent=[0,12,0,12])
ax[0].set_title('Valencia DANA: Flood Volume Analysis', fontweight='bold', fontsize=14)
plt.colorbar(im1, ax=ax[0], label='Water Depth (mm)')

# Map 2: The Solution (Proposed Sponge City Zones)
# We highlight areas where 'Sponge City' tech would reduce risk by >50%
```

```
sponge_priority = np.where(flood_volume > 200, 1, 0)
ax[1].imshow(sponge_priority, cmap='Greens', alpha=0.6, extent=[0,12,0,12])
ax[1].set_title('AI-Recommended "Sponge City" Nodes', fontweight='bold', fontsize=14)

plt.tight_layout()
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt

# 1. SETUP DATA (Simulating Valencia Urban Logic)
grid_size = 100
# Elevation (West to East slope toward the Mediterranean)
x, y = np.meshgrid(np.linspace(0, 10, grid_size), np.linspace(0, 10, grid_size))
elevation = (x * 0.5)

# Land Use: 0=Urban(Concrete), 1=Green(Parks), 2=Industrial
land_use = np.random.choice([0, 1, 2], size=(grid_size, grid_size), p=[0.6, 0.2, 0.2])
absorption = np.vectorize(lambda x: [0.1, 0.8, 0.2][x])(land_use)

# Flood Calculation (High Rainfall - Topography)
flood_depth = 400 * (1 - absorption) - (elevation * 20)
flood_depth = np.clip(flood_depth, 0, None)

# --- THE 4-PART DASHBOARD ---
fig, ax = plt.subplots(2, 2, figsize=(18, 14))

# 1. THE MAP (Spatial Intelligence)
im = ax[0, 0].imshow(flood_depth, cmap='ocean_r')
ax[0, 0].set_title('Spatial Flood Depth (Valencia City)', fontweight='bold')
plt.colorbar(im, ax=ax[0, 0], label='Water Depth (mm)')

# 2. THE HISTOGRAM (Statistical Distribution)
ax[0, 1].hist(flood_depth.flatten(), bins=25, color='#1f77b4', edgecolor='white')
ax[0, 1].set_title('Frequency of Water Depth Levels', fontweight='bold')
ax[0, 1].set_xlabel('Depth (mm)')
ax[0, 1].set_ylabel('City Grid Cells')

# 3. THE PIE CHART (Urban Composition)
labels = ['Concrete (High Risk)', 'Parks (Sponge)', 'Industrial']
sizes = [np.sum(land_use == 0), np.sum(land_use == 1), np.sum(land_use == 2)]
ax[1, 0].pie(sizes, labels=labels, autopct='%1.1f%%', colors=['#555555', '#2ca02c', '#ff7f0e'], startangle=140)
ax[1, 0].set_title('Urban Surface Absorption Ratio', fontweight='bold')

# 4. THE BAR CHART (Impact by Sector)
impacts = [np.mean(flood_depth[land_use == i]) for i in range(3)]
ax[1, 1].bar(labels, impacts, color=['#d62728', '#2ca02c', '#ff7f0e'])
ax[1, 1].set_title('Average Flood Impact by Sector', fontweight='bold')
ax[1, 1].set_ylabel('Avg Depth (mm)')
```

```
ax[1, 1].set_ylabel('Avg Depth (mm)')

plt.tight_layout()
plt.show()
```