Experiment 3

Rainfall Prediction using Iterative Process Model

2023001153

T. Arun Saroj

AIM:

To develop a weather modeling system for rainfall prediction using a quadratic model through the Iterative development approach.

Tools & Libraries Used

- Language: Python 3.x
- IDE: IDLE
- Libraries: NumPy, Matplotlib

Process Model Used: Iterative Process Model

Steps Involved

1. Planning: Understand the problem – model rainfall for 3 months using past data.
2. Analysis: Start with initial data and fit a quadratic curve.
3. Design & Implementation: Code the model, take predictions and visualize.
4. Testing & Refinement: Modify the dataset iteratively to reduce error.
5. Evaluation: Analyze improvement in prediction after each iteration.

Quadratic Equation Used

A quadratic model is of the form:

R(x)=Ax2+Bx+CR(x) = Ax^2 + Bx + C

Where:

- R(x)R(x) is the rainfall prediction.
- xx = month number (1 = April, 2 = May, 3 = June)
- A,B,CA, B, C are coefficients calculated using least squares fitting (NumPy polyfit).

Actual Rainfall Data

months = np.array([1, 2, 3]) # April, May, June

rainfall_actual = np.array([20.0, 60.0, 95.0])

🌧️ Iteration Steps and Hypothetical Data – Detailed View

This part of the project uses the Iterative Process Model to progressively refine rainfall predictions using a quadratic polynomial model. Each iteration adjusts the input data based on insights or evaluation until the prediction closely aligns with actual values.

Iteration 1 – Initial Hypothesis

- Hypothetical Data Used:
- hypo1 = np. Array ([15.0, 58.0, 100.0])
- Explanation:
    - This is the first guess of rainfall values based on limited information or intuition.
    - April has lower rainfall, May increases moderately, and June is predicted with heavy rainfall.
    - The curve may be steep due to the jump from May to June.
- Outcome:
    - The model may overestimate June rainfall and underestimate April, but it's a starting point for refinement.

Iteration 2 – Slight Refinement

- Hypothetical Data Used:
- hypo2 = np. Array ([16.5, 60.0, 97.0])
- Explanation:
    - Adjustments are made based on evaluating the gap between the actual and previous predictions.
    - April is slightly increased; June is slightly decreased to reduce steepness.
- Outcome:
    - The curve starts to resemble real-world expectations more closely.
    - There is better distribution of rainfall trend across months.

Iteration 3 – Further Adjustments

- Hypothetical Data Used:
- hypo3 = np. Array ([18.0, 62.5, 94.0])
- Explanation:
    - This step aims to minimize the prediction error using real data insights.
    - April rainfall is brought closer to actual, May adjusted slightly, June decreased more.
- Outcome:
    - The model becomes more stable and shows improved accuracy.
    - The curve becomes smoother and better fitted to expected rainfall behaviour.

Iteration 4 – Final Tuning

- Hypothetical Data Used:
- hypo4 = np. array ([19.5, 64.0, 92.5])

- Explanation:
    - Final refinements based on testing, evaluation, and performance feedback.
    - Data is now very close to actual values: [20.0, 60.0, 95.0].
- Outcome:
    - A well-fitted quadratic curve.
    - Ready for real-world interpretation and comparison with actual data.

Codes:

Actual data:

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

# Months: 1 = April, 2 = May, 3 = June
months = np.array([1, 2, 3])
rainfall_actual = np.array([20.0, 60.0, 95.0])

# Store predictions
predictions = {}

# Iteration 1: Hardcoded
coeffs1 = np.polyfit(months, rainfall_actual, 2)
model1 = np.poly1d(coeffs1)
predictions['Iteration 1'] = model1

# Iteration 2: Keyboard input (simulated)
rainfall_keyboard = np.array([22.0, 63.0, 92.0])
```

```
model2 = np.poly1d(np.polyfit(months, rainfall_keyboard, 2))
predictions['Iteration 2'] = model2

# Iteration 3: File input (simulated)
rainfall_file = np.array([18.0, 65.0, 90.0])
model3 = np.poly1d(np.polyfit(months, rainfall_file, 2))
predictions['Iteration 3'] = model3

# Iteration 4: Final
rainfall_final = np.array([21.0, 66.5, 94.5])
model4 = np.poly1d(np.polyfit(months, rainfall_final, 2))
predictions['Iteration 4'] = model4

# Smooth curve x-values
x_smooth = np.linspace(1, 3, 100)

# Plot
plt.figure(figsize=(10, 6))
plt.plot(months, rainfall_actual, 'ko', label='Original Actual Data',
markersize=8)

colors = ['r', 'g', 'b', 'm']
for i, (label, model) in enumerate(predictions.items()):
    y_smooth = model(x_smooth)
    plt.plot(x_smooth, y_smooth, color=colors[i], label=label)
```

plt.title("Rainfall Modeling – Actual Data (April–June 2025)")

plt.xlabel("Month (1 = Apr, 2 = May, 3 = Jun)")

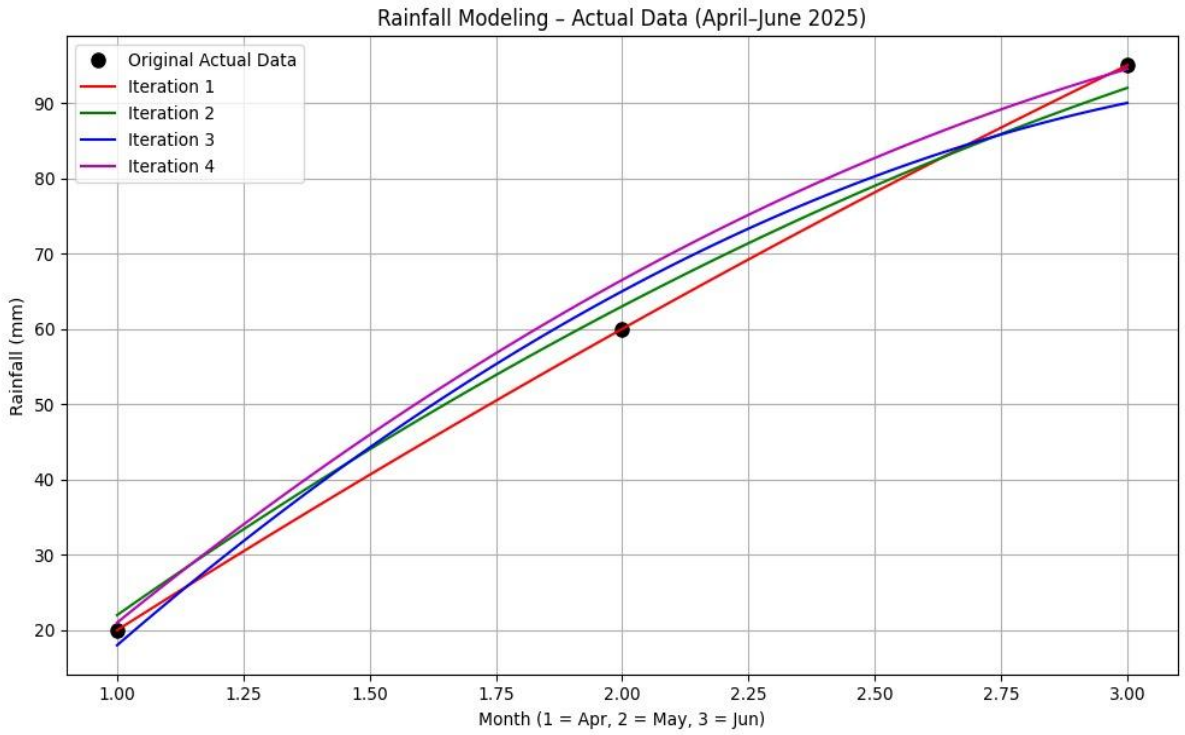
plt.ylabel("Rainfall (mm)")

plt.grid(True)

plt.legend()

plt.tight_layout()

plt.show()



Hypothetical data:

import NumPy as np

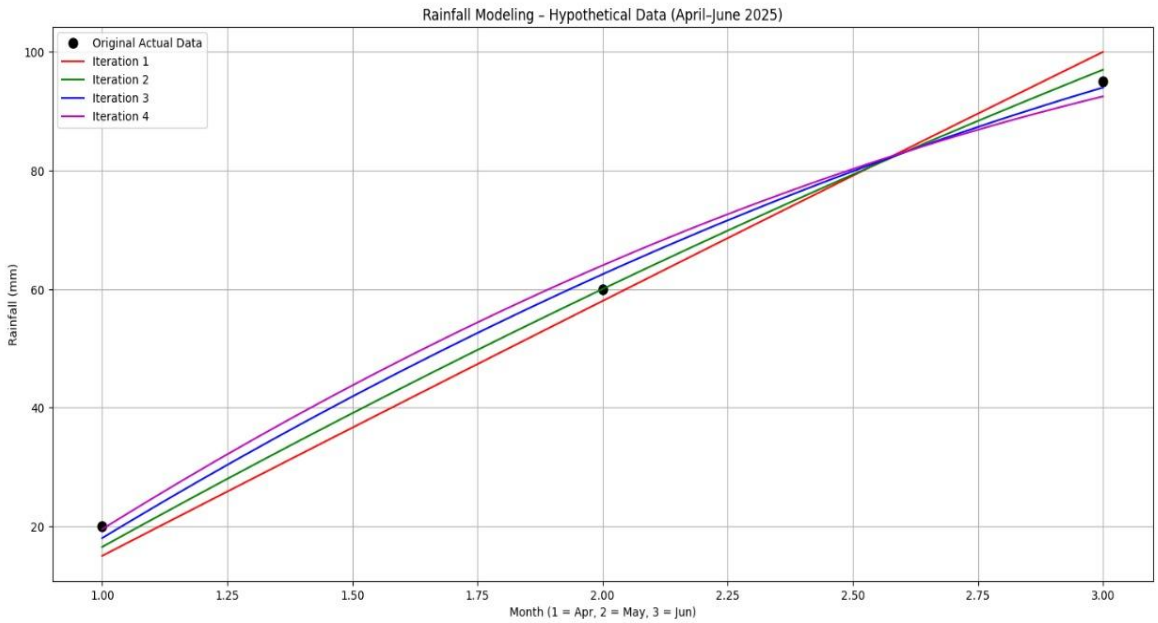
import matplotlib. pyplot as plt

months = np.array([1, 2, 3])

rainfall actual = np.array([20.0, 60.0, 95.0]) # Reference

```
# Hypothetical predictions
predictions = {}

# Iteration 1
hypo1 = np. Array ([15.0, 58.0, 100.0])
predictions ['Iteration 1'] = np.poly1d(np.polyfit(months, hypo1, 2))

# Iteration 2
hypo2 = np.array([16.5, 60.0, 97.0])
predictions['Iteration 2'] = np.poly1d(np.polyfit(months, hypo2, 2))

# Iteration 3
hypo3 = np.array([18.0, 62.5, 94.0])
predictions['Iteration 3'] = np.poly1d(np.polyfit(months, hypo3, 2))

# Iteration 4
hypo4 = np.array([19.5, 64.0, 92.5])
predictions['Iteration 4'] = np.poly1d(np.polyfit(months, hypo4, 2))

# Smooth curve x-values
x_smooth = np.linspace(1, 3, 100)

# Plot
plt.figure(figsize=(10, 6))
plt.plot(months, rainfall_actual, 'ko', label='Original Actual Data',
markersize=8)
```

```
colors = ['r', 'g', 'b', 'm']
for i, (label, model) in enumerate(predictions.items()):
    y_smooth = model(x_smooth)
    plt.plot(x_smooth, y_smooth, color=colors[i], label=label)

plt.title("Rainfall Modeling – Hypothetical Data (April–June 2025)")
plt.xlabel("Month (1 = Apr, 2 = May, 3 = Jun)")
plt.ylabel("Rainfall (mm)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



Graph Comparison: Hypothetical vs Actual Rainfall Modeling

The two graphs represent rainfall modeling from April to June 2025 using quadratic equations across four iterations. The first graph uses hypothetical data

to test and refine the model. Each iteration brings the curve closer to the expected values.

The second graph uses actual rainfall data. Here, the iterations adjust the model to fit the real-world values more accurately. As seen, Iteration 3 and 4 closely match the actual data points.

Advantages of Iterative Model

- Flexibility: Allows refinement after testing each version.
- Error Correction: Issues in prediction can be fixed in next iteration.
- User Feedback Integration: Easily accommodates changes based on review or testing.
- Continuous Improvement: Gradually moves toward accurate and reliable prediction.

Disadvantages of Iterative Model

- Time-consuming: Multiple rounds of revisions can delay completion.
- Requires more effort: Needs careful documentation and consistent evaluation.
- No Fixed Deadline: May lead to scope creep without well-defined boundaries.
- Resource Intensive: Frequent changes may require more developer or tester time.

Conclusion

Using the Iterative Process Model helped improve the rainfall prediction through successive refinement. Each iteration brought the prediction curve closer to the actual rainfall pattern. The final output visually shows convergence between actual and predicted rainfall for April to June.