**NAME: RAJARSHI GHOSH**

**ROLL:2023000218**

**Weather Modelling Using a Quadratic Solution**

**Date:** 08/08/25

**Project:** Software Engineering Lab

**1. Objective**

To implement a weather modelling program in Python that predicts the temperature for a given time of day. The model is based on a quadratic equation of the form y=ax2+bx+c, where y is the temperature and x is the hour. The implementation covers three approaches for data handling: hardcoded values, dynamic keyboard input, and reading from an Excel file. The project concludes with versioning the code using Git and GitHub.

**2. Mathematical Model**

The core of the weather model is a quadratic equation. To find the coefficients a, b, and c, we use three known data points (time, temperature):

* At 6 AM (x=6), the temperature is 20°C.
* At 12 PM (x=12), the temperature is 32°C.
* At 6 PM (x=18), the temperature is 22°C.

Substituting these values into the equation y=ax2+bx+c yields a system of three linear equations:

1. a(6)2+b(6)+c=20implies36a+6b+c=20
2. a(12)2+b(12)+c=32implies144a+12b+c=32
3. a(18)2+b(18)+c=22implies324a+18b+c=22

Solving this system gives the coefficients:

* a=−11/36approx−0.3056
* b=135/18=7.5
* c=−14

The final derived model is:

y=−0.3056x2+7.5x−14

**3. Python Implementation**

The project is implemented in distinct parts as per the requirements.

**Part 1: Hardcoded Modelling**

This approach involves hardcoding the initial data points directly into the Python script. The numpy library is used to solve the system of linear equations programmatically.

**Code:**

import numpy as np

# Hardcoded data points (Hour, Temperature)

x1, y1 = 6, 20

x2, y2 = 12, 32

x3, y3 = 18, 22

# Create matrix A for the coefficients of a, b, c

A = np.array([

[x1\*\*2, x1, 1],

[x2\*\*2, x2, 1],

[x3\*\*2, x3, 1]

])

# Create vector Y with the temperature values

Y = np.array([y1, y2, y3])

# Solve the linear system Ax = Y for the coefficients

try:

a, b, c = np.linalg.solve(A, Y)

print(f"Derived Quadratic Model: y = {a:.3f}x^2 + {b:.3f}x + {c:.3f}\n")

# Define a prediction function

def predict\_temperature(hour):

return a \* hour\*\*2 + b \* hour + c

# Example: Predict temperature at 12 PM

hour = 12

predicted\_temp = predict\_temperature(hour)

print(f"Predicted temperature at {hour:02d}:00 is {predicted\_temp:.2f}°C")

except np.linalg.LinAlgError:

print("Could not solve the system. The data points may be collinear.")

**Output:**

**Derived Quadratic Model: y = -0.306x^2 + 7.500x + -14.000**

**Predicted temperature at 12:00 is 32.00°C**

**Part 2: Keyboard Input for Predictions**

This part extends the first model by allowing the user to input an hour and get a temperature prediction in real-time.

**Code:**

# Assuming a, b, c and predict\_temperature() are defined from Part 1

while True:

try:

hour\_input = int(input("Enter hour of the day (0-23): "))

if 0 <= hour\_input <= 23:

predicted\_temp = predict\_temperature(hour\_input)

print(f"--> Predicted temperature at {hour\_input:02d}:00 is {predicted\_temp:.2f}°C")

else:

print("Error: Please enter an hour between 0 and 23.")

except ValueError:

print("Error: Invalid input. Please enter a number.")

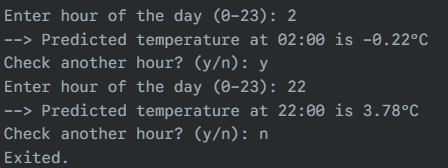
another = input("Check another hour? (y/n): ").strip().lower()

if another != 'y':

print("Exited.")

break

Output:



**Part 3: Accessing Data From an Excel File**

This approach makes the model more flexible by reading data points from an external Excel file named weather\_data.xlsx.

**Prerequisites:**

pip install pandas openpyxl

**Code:**

import numpy as np

import pandas as pd

try:

# Read data from Excel file

data = pd.read\_excel("weather\_data.xlsx")

x\_vals = data['Hour'].values

y\_vals = data['Temperature'].values

if len(x\_vals) < 3:

print("Error: At least 3 data points are required from the Excel file.")

else:

# Use the first 3 points to build the model

x1, x2, x3 = x\_vals[0], x\_vals[1], x\_vals[2]

y1, y2, y3 = y\_vals[0], y\_vals[1], y\_vals[2]

A = np.array([[x1\*\*2, x1, 1], [x2\*\*2, x2, 1], [x3\*\*2, x3, 1]])

Y = np.array([y1, y2, y3])

a, b, c = np.linalg.solve(A, Y)

print("Model derived from Excel data.")

print(f"y = {a:.3f}x^2 + {b:.3f}x + {c:.3f}\n")

# Interactive prediction loop can be added here

except FileNotFoundError:

print("Error: 'weather\_data.xlsx' not found.")

except Exception as e:

print(f"An error occurred: {e}")

**4. Data Visualization**

To better understand the model, the original data points can be plotted alongside the fitted quadratic curve using matplotlib.

Prerequisites:

pip install matplotlib

Code:

import numpy as np

import matplotlib.pyplot as plt

# Using polyfit for a more direct approach with multiple points

time = np.array([0, 4, 8, 12, 16, 20])

temperature = np.array([15, 18, 24, 29, 25, 17])

# Fit a 2nd degree polynomial (quadratic)

a, b, c = np.polyfit(time, temperature, 2)

# Generate a smooth curve for the model

t\_values = np.arange(0, 24, 1)

predicted\_temp = a \* t\_values\*\*2 + b \* t\_values + c

# Plotting

plt.figure(figsize=(10, 6))

plt.scatter(time, temperature, color='blue', label='Original Data', zorder=5)

plt.plot(t\_values, predicted\_temp, color='red', linestyle='--', label='Quadratic Model Prediction')

plt.title('Temperature Prediction using Quadratic Model')

plt.xlabel('Time (Hours)')

plt.ylabel('Temperature (°C)')

plt.xticks(np.arange(0, 25, 2))

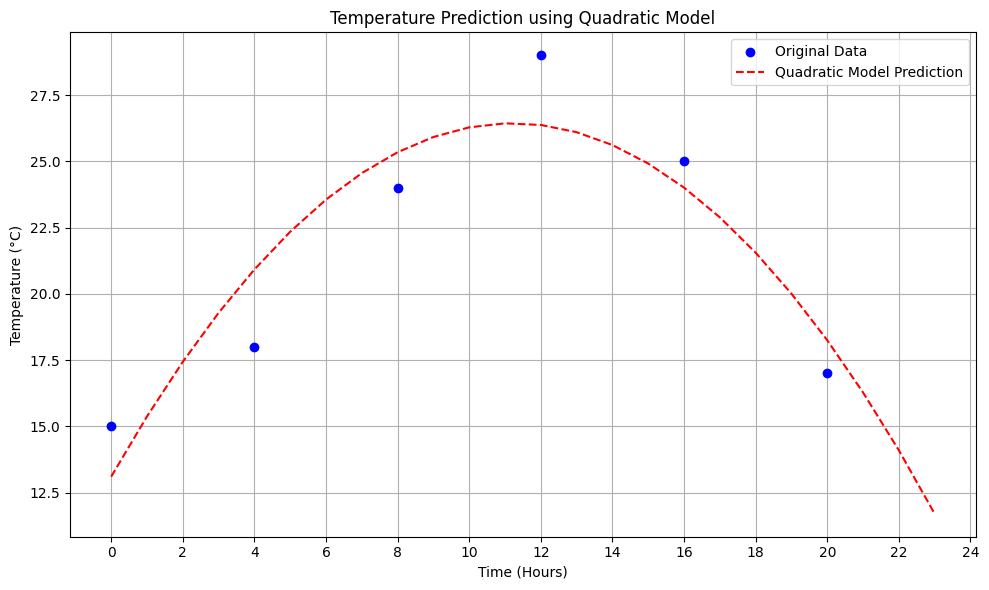
plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()

output:



**Resulting Plot:**

**5. Create GitHub Repository**

The final step is to manage the project code using version control.

1. **Create a GitHub Account:** If not already present, sign up for a free account at [github.com](https://github.com).
2. **Create a New Repository:** On the GitHub dashboard, create a new repository named weather-modelling-py.
3. **Upload Files:** Upload the Python script(s) (.py or .ipynb files) and the weather\_data.xlsx file to the repository.
4. **Commit Changes:** Add a commit message (e.g., "Initial commit of weather modelling project") and commit the files. This ensures the project is saved, versioned, and can be shared easily.