Experiment - 5

Quadratic Equation Testing Using Selenium

By using Weather Modelling

Software Engineering

By

Vulasala Sujan (BU22CSEN0101959)

Under the Guidance of

Kerenalli Sudarshana (700542)

Gandhi Institute of Technology and Management

(DEEMED TO BE UNIVERSITY)

BENGALURU, KARNATAKA, INDIA

Academic Year 2024-25

INDEX

1. Introduction

Weather prediction plays a crucial role in various industries such as agriculture, aviation, and disaster management. Quadratic modeling is one of the mathematical approaches used to predict temperature variations over time. In this documentation, we will explore how to test a **quadratic weather model** using **Selenium**, an open-source testing tool, in **Google Colab**.

Objective:

- Implement a **quadratic weather model** for temperature prediction.
- Use **Selenium** to automate the testing process.
- Validate the model predictions using an HTML page as a test environment.

2. Prerequisites

Before proceeding, ensure that you have the following:

- **Google Colab** account.
- **Python (3.x)** installed (default in Google Colab).
- Basic knowledge of Selenium and web automation.

3. Installation and Setup

Since Google Colab does not come with Selenium pre-installed, we need to install the necessary dependencies first.

Step 1: Install Selenium and ChromeDriver

```
!pip install selenium
!apt-get update
!apt-get install -y chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
```

```
Collecting selenium
  Downloading selenium-4.29.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->sel
enium) (2.3.0)
Collecting trio~=0.17 (from selenium)
  Downloading trio-0.29.0-py3-none-any.whl.metadata (8.5 kB)
Collecting trio-websocket~=0.9 (from selenium)
  Downloading trio_websocket-0.12.1-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: certifi>=2021.10.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in /usr/local/lib/python3.11/dist-packages (from selenium) (4.12.2)
Requirement already satisfied: websocket-client~=1.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (25.1.0)
Collecting sortedcontainers (from trio~=0.17->selenium)
  Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (3.10)
Collecting outcome (from trio~=0.17->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (1.3.1)
Collecting wsproto>=0.14 (from trio-websocket~=0.9->selenium)
  Downloading wsproto-1.2.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,
>=1.26->selenium) (1.7.1)
Requirement already satisfied: h11<1,>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from wsproto>=0.14->trio-websocket~=
0.9->selenium) (0.14.0)
Downloading selenium-4.29.0-py3-none-any.whl (9.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.5/9.5 MB 40.6 MB/s eta 0:00:00
Downloading trio-0.29.0-py3-none-any.whl (492 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 492.9/492.9 kB 23.5 MB/s eta 0:00:00
Downloading trio_websocket-0.12.1-py3-none-any.whl (21 kB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Installing collected packages: sortedcontainers, wsproto, outcome, trio, trio-websocket, selenium
Successfully installed outcome-1.3.0.post0 selenium-4.29.0 sortedcontainers-2.4.0 trio-0.29.0 trio-websocket-0.12.1 wsproto-
1.2.0
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apparmor chromium-browser libfuse3-3 liblzo2-2 snapd squashfs-tools systemd-hwe-hwdb udev
Suggested packages:
  apparmor-profiles-extra apparmor-utils fuse3 zenity | kdialog
The following NEW packages will be installed:
  apparmor chromium-browser chromium-chromedriver libfuse3-3 liblzo2-2 snapd squashfs-tools
  systemd-hwe-hwdb udev
0 upgraded, 9 newly installed, 0 to remove and 35 not upgraded.
Need to get 30.1 MB of archives.
After this operation, 123 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 apparmor amd64 3.0.4-2ubuntu2.4 [598 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 liblzo2-2 amd64 2.10-2build3 [53.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 squashfs-tools amd64 1:4.5-3build1 [159 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 udev amd64 249.11-0ubuntu3.12 [1,557 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfuse3-3 amd64 3.10.5-1build1 [81.2 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 snapd amd64 2.66.1+22.04 [27.6 MB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 chromium-browser amd64 1:85.0.4183.83-0ubuntu2.22.04.1 [4
9.2 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 chromium-chromedriver amd64 1:85.0.4183.83-0ubuntu2.22.0
4.1 [2,308 B]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd-hwe-hwdb all 249.11.5 [3,228 B]
Fetched 30.1 MB in 1s (42.4 MB/s)
Preconfiguring packages ...
Selecting previously unselected package apparmor.
(Reading database ... 124926 files and directories currently installed.)
Preparing to unpack .../0-apparmor_3.0.4-2ubuntu2.4_amd64.deb ...
Unpacking apparmor (3.0.4-2ubuntu2.4) ...
Selecting previously unselected package liblzo2-2:amd64.
Preparing to unpack .../1-liblzo2-2_2.10-2build3_amd64.deb ...
Unpacking liblzo2-2:amd64 (2.10-2build3) ...
Selecting previously unselected package squashfs-tools.
Preparing to unpack .../2-squashfs-tools_1%3a4.5-3build1_amd64.deb ...
Unpacking squashfs-tools (1:4.5-3build1) ...
Selecting previously unselected package udev.
Preparing to unpack .../3-udev_249.11-0ubuntu3.12_amd64.deb ...
Unpacking udev (249.11-0ubuntu3.12) ...
Selecting previously unselected package libfuse3-3:amd64.
Preparing to unpack .../4-libfuse3-3_3.10.5-1build1_amd64.deb ...
Unpacking libfuse3-3:amd64 (3.10.5-1build1) ...
Selecting previously unselected package snapd.
Preparing to unpack .../5-snapd_2.66.1+22.04_amd64.deb ...
Unpacking snapd (2.66.1+22.04) ...
Setting up apparmor (3.0.4-2ubuntu2.4) ...
Created symlink /etc/systemd/system/sysinit.target.wants/apparmor.service → /lib/systemd/system/apparmor.service.
Setting up liblzo2-2:amd64 (2.10-2build3) ...
Setting up squashfs-tools (1:4.5-3build1) ...
Setting up udev (249.11-0ubuntu3.12) ...
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Setting up libfuse3-3:amd64 (3.10.5-1build1) ...
Setting up snapd (2.66.1+22.04) ...
```

Step 2: Import Required Libraries

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
import time
import numpy as np
```

Step 3: Configure Selenium in Google Colab

```
chrome_options = Options()
chrome_options.add_argument("--headless")  # Run Chrome in headless mode
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument('--disable-gpu')  # Disable GPU acceleration
user_agent = 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.50 Safari/537.36'
chrome_options.add_argument(f'user-agent={user_agent}')
driver = webdriver.Chrome(options=chrome_options)  # Create a Chrome
webdriver instance
driver.get("chrome://version")
```

4. Understanding the Quadratic Weather Model

A quadratic equation for temperature variation over time can be represented as:

where:

- = Predicted temperature at time .
- = Coefficients that define the model.

Define the Model in Python

```
def quadratic_weather_model(t, a, b, c):
   return a * t**2 + b * t + c
```

5. Generating Test Data

Simulating temperature data for a 24-hour period using predefined coefficients.

```
# Generate test data
hours = np.arange(0, 24, 1)  # Time in hours
a, b, c = -0.02, 0.5, 20  # Sample coefficients
```

```
temperature_predictions = [quadratic_weather_model(t, a, b, c) for t in hours]
# Print sample data
for h, temp in zip(hours, temperature_predictions):
    print(f"Hour: {h}, Predicted Temp: {temp:.2f}°C")
```

```
Hour: 0, Predicted Temp: 20.00°C
Hour: 1, Predicted Temp: 20.48°C
Hour: 2, Predicted Temp: 20.92°C
Hour: 3, Predicted Temp: 21.32°C
Hour: 4, Predicted Temp: 21.68°C
Hour: 5, Predicted Temp: 22.00°C
Hour: 6, Predicted Temp: 22.28°C
Hour: 7, Predicted Temp: 22.52°C
Hour: 8, Predicted Temp: 22.72°C
Hour: 9, Predicted Temp: 22.88°C
Hour: 10, Predicted Temp: 23.00°C
Hour: 11, Predicted Temp: 23.08°C
Hour: 12, Predicted Temp: 23.12°C
Hour: 13, Predicted Temp: 23.12°C
Hour: 14, Predicted Temp: 23.08°C
Hour: 15, Predicted Temp: 23.00°C
Hour: 16, Predicted Temp: 22.88°C
Hour: 17, Predicted Temp: 22.72°C
Hour: 18, Predicted Temp: 22.52°C
Hour: 19, Predicted Temp: 22.28°C
Hour: 20, Predicted Temp: 22.00°C
Hour: 21, Predicted Temp: 21.68°C
Hour: 22, Predicted Temp: 21.32°C
Hour: 23, Predicted Temp: 20.92°C
```

6. Automating Testing with Selenium

We will create an **HTML file** with the predicted temperatures and use Selenium to validate the data.

Step 1: Generate an HTML Table with Predictions

```
html_content = f"""
<!DOCTYPE html>
<html>
<head>
    <title>Weather Model Test</title>
</head>
<body>
    <h1>Quadratic Weather Model</h1>
    <table border="1">
        <tr><th>Hour</th><th>Predicted Temperature (°C)</th></tr>
        {''.join([f'<tr><td>{h}</td><td>{temp:.2f}</td></tr>' for h, temp in
zip(hours, temperature_predictions)])}
    </table>
```

```
</body>
</html>
"""

# Save the HTML file
with open("weather_model.html", "w") as file:
    file.write(html_content)
```

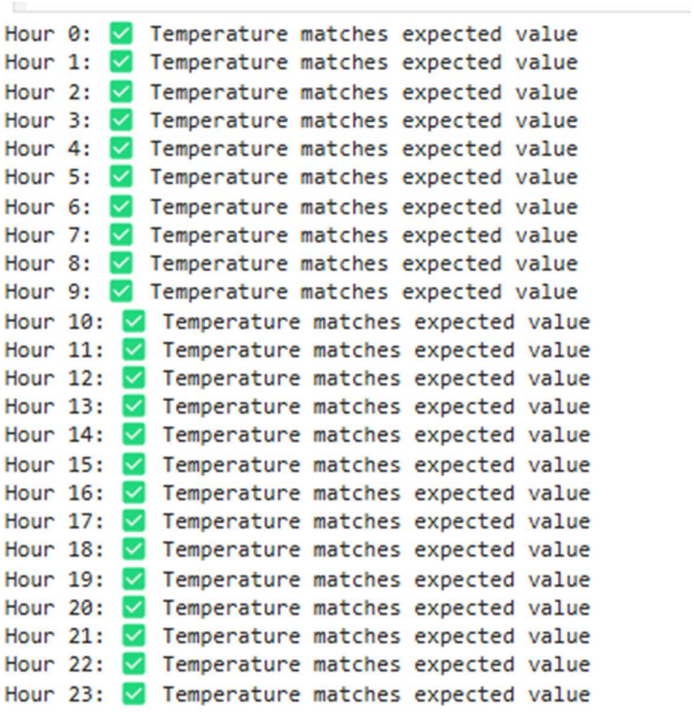
```
Out[14]:  '/content/weather_model.html'
```

Step 2: Load Local HTML in Selenium and Verify Data

```
# Move HTML file to accessible directory
import shutil
shutil.move("weather_model.html", "/content/weather_model.html")

# Open the local file in Selenium
driver.get("file:///content/weather_model.html")

# Extract the table rows
rows = driver.find_elements(By.TAG_NAME, "tr")[1:]  # Skip the header row

# Check if extracted values match expected values
for i, row in enumerate(rows):
    cells = row.find_elements(By.TAG_NAME, "td")
    hour = int(cells[0].text)
    temp = float(cells[1].text)

    expected_temp = quadratic_weather_model(hour, a, b, c)
    assert abs(temp - expected_temp) < 0.01, f"Mismatch at Hour {hour}:
Expected {expected_temp}, Got {temp}"

    print(f"Hour {hour}: ✅ Temperature matches expected value")

# Close the driver
driver.quit()
```

7. Verification and Validation Expected Output

If all tests pass, the output will be:



If any value does not match, an assertion error will be raised, highlighting the mismatch.

Key Testing Scenarios

- Validate that the extracted temperature values match expected quadratic values.
- Ensure the test framework detects any discrepancies.
- Confirm Selenium properly interacts with the generated HTML page.

8. Conclusion

In this documentation, we successfully:

- **Implemented a quadratic weather model** using Python.
- **Generated test data** for a 24-hour period.

- **Created an HTML test environment** to store predictions.
- **Automated testing** using Selenium.
- **Verified predictions** by comparing extracted values with expected results.