PRE-MACHINE LEARNING AND DEEP LEARNING EXAMINATION

Capstone Project: AI-Powered Website Traffic Monitoring Tool (CLI + Poisson Distribution)

Project: Building an AI-Driven Website Traffic Anomaly Detection CLI Tool Using Python

Project Overview

In this project, you will build a **Python-based Command Line (CLI) application** that continuously monitors a given website (for example, https://codehouse.cloud), tracks its response time, and automatically detects **abnormal performance deviations** using the **Poisson distribution**.

Your program will:

- Periodically ping the website,
- Log response times and status codes,
- Use statistical AI logic to detect anomalies,
- Display colored alerts directly in the terminal.

Learning Objectives

By completing this project, you will learn to:

1. Implement **Poisson distribution** for anomaly detection.
2. Use Python's **requests**, **NumPy**, and (**SciPy)** libraries for monitoring.
3. Build a **CLI dashboard** that updates in real-time.
4. Log and analyze data using **CSV** and **pandas**.
5. Automate real-time alerts for website performance monitoring.

Project Tasks

Task 1: Setup and Libraries

Install the required libraries:

pip install requests numpy scipy pandas rich

rich is used for colored CLI output.

Task 2: Create the Project File

Create a file named traffic_monitor.py.

Task 3: Collect Website Data

Write a function to send periodic requests and record response times.

```
import requests
import pandas as pd
import numpy as np
import time
from datetime import datetime
from scipy.stats import poisson
from rich.console import Console

console = Console()

def fetch_website_response(url):
    """Ping website and return response time."""
    try:
        start = time.time()
        response = requests.get(url, timeout=5)
        end = time.time()
        response_time = end - start
        status = response.status_code
        return response_time, status
```

except requests.RequestException:

return 0, 0

Task 4: Analyze Using Poisson Distribution

Add a function that detects anomalies using the Poisson probability model.

```
def analyze_poisson_anomaly(response_times, latest_time):
    if len(response_times) < 5:
        return "CodeAI is Gathering data...", "info"

    lam = np.mean(response_times)
    prob = 1 - poisson.cdf(latest_time, mu=lam)

    if prob < 0.05:
        return f"🚨 ALERT: Response {latest_time:.2f}s deviates from normal (λ={lam:.2f})", "error"
    else:
        return f"✅ Normal: Latest {latest_time:.2f}s (λ={lam:.2f})", "success"
```

Task 5: Main Monitoring Loop

Create a loop that runs continuously, logging each ping to a CSV file and showing CLI updates.

```
def monitor_website(url, interval=10, log_file="traffic_logs.csv"):
    console.print(f"🌐 Starting AI Traffic Monitor for {url}\n", style="bold cyan")

    # Load or create CSV
    try:
        df = pd.read_csv(log_file)
    except FileNotFoundError:
```

```
    df = pd.DataFrame(columns=["timestamp", "response_time", "status"])
    df.to_csv(log_file, index=False)

  while True:
    response_time, status = fetch_website_response(url)
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    new_row = pd.DataFrame({"timestamp": [timestamp], "response_time": [response_time],
"status": [status]})
    df = pd.concat([df, new_row], ignore_index=True)
    df.to_csv(log_file, index=False)

    message, level = analyze_poisson_anomaly(df["response_time"].tail(30), response_time)

    if level == "success":
      console.print(message, style="green")
    elif level == "error":
      console.print(message, style="bold red")
    else:
      console.print(message, style="yellow")

    time.sleep(interval)
```

Task 6: Run the Tool

Finally, add a CLI entry point:

```
if __name__ == "__main__":
```

website_url = input("Enter website URL to monitor (e.g., https://codehouse.cloud): ")

interval = int(input("Enter check interval (in seconds, e.g., 10): "))

monitor_website(website_url, interval)

Example Output

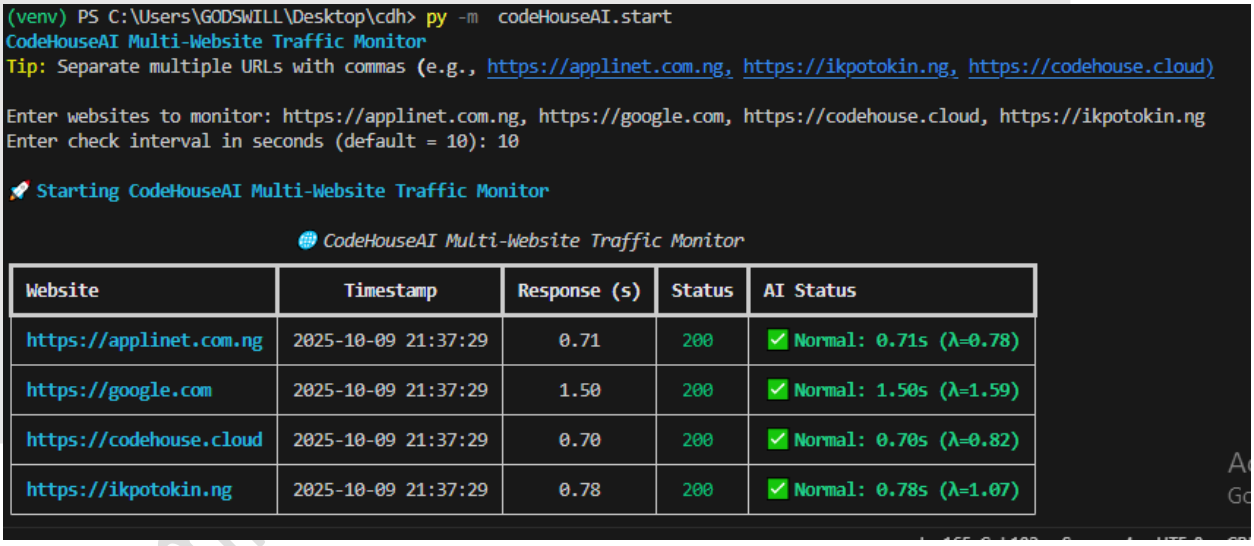
🌐 Starting AI Traffic Monitor for https://codehouse.cloud

✅ Normal: Latest 0.32s (λ=0.45)

✅ Normal: Latest 0.41s (λ=0.44)

🚨 ALERT: Response 2.18s deviates from normal (λ=0.45)

✅ Normal: Latest 0.37s (λ=0.46)



Every result is logged in a traffic_logs.csv file like:

timestamp	response_time	status
2025-10-09 14:10:00	0.32	200
2025-10-09 14:10:10	0.41	200

timestamp	response_time	status
2025-10-09 14:10:20	2.18	504

Challenge Extension Ideas

- Visualize results using **Matplotlib and seaborns** (compulsory).
- Monitor **multiple websites** at once (multithreading).
- Use **colored ASCII charts** to display trends in the CLI.
- Integrate **AI prediction** to forecast expected λ values dynamically.

Deliverables

1. The complete Python script traffic_monitor.py.
2. A CSV file showing recorded logs.
3. A short video or screenshot of the CLI tool running.

Assessment Criteria

Criteria	Description	Marks
Statistical Accuracy	Correct use of Poisson distribution	20
Code Functionality	Proper logging and alerting	25
CLI Interface Design	Clear, colored, and readable display	15
Automation	Continuous monitoring loop works	20
Innovation	Add-ons like charts, email alerts	20

Expected Outcome

A **Python CLI AI Monitoring Tool** that:

- Tracks website health in real time,
- Logs every event to CSV, and
- Raises instant alerts when response behavior deviates from the expected **Poisson-based normal** pattern.