KAIROS PROJECT PROGRESS

Data Analysis and Visualization

TEAM

6688030	Sarach Islam
6688055	Takka Leeheng
6688059	Athip Madnoth
6688124	Nithit Teeraworawit
6688173	Passakorn Piboonmahachotikul

OUR OBJECTIVE

To optimize the process of analysis and visualization of data and create a better system for such task.

RECORD EXAMPLE OF OUR DATASET

This is raw data we will process and there are 20 column records to manage

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
anxiety_le	self_esteer	mental_he	depression	headache	blood_pres	sleep_qual	breathing_	noise_level	living_con	safety	basic_neec	academic_	study_load	teacher_st	future_care	social_sup	peer_press	extracurric	bullying	stress_level
14	20	0	11	2	1	2	4	2	3	3	2	3	2	3	3	2	3	3	2	1
15	8	1	15	5	3	1	4	3	1	2	2	1	4	1	5	1	4	5	5	2
12	18	1	14	2	1	2	2	2	2	3	2	2	3	3	2	2	3	2	2	1
16	12	1	15	4	3	1	3	4	2	2	2	2	4	1	4	1	4	4	5	2
16	28	0	7	2	3	5	1	3	2	4	3	4	3	1	2	1	5	0	5	1
20	13	1	21	3	3	1	4	3	2	2	1	2	5	2	5	1	4	4	5	2
4	26	0	6	1	2	4	1	1	4	4	4	5	1	4	1	3	2	2	1	0
17	3	1	22	4	3	1	5	3	1	1	1	1	3	2	4	1	4	4	5	2
13	22	1	12	3	1	2	4	3	3	3	3	3	3	2	3	3	3	2	2	1
6	8	0	27	4	3	1	2	0	5	2	2	2	2	1	5	1	5	3	4	1
17	12	1	25	4	3	1	3	4	2	1	1	1	3	1	4	1	4	4	5	2
17	15	1	22	3	3	1	5	5	2	1	1	1	3	1	4	1	5	5	4	2
5	28	0	8	1	2	4	2	2	3	5	5	5	2	4	1	3	1	1	1	0
9	23	1	24	4	3	1	0	1	2	4	3	1	2	3	3	0	1	0	1	2
2	28	0	3	1	2	4	2	1	3	4	4	4	2	5	1	3	1	2	1	0
11	21	0	14	3	1	2	4	2	2	2	2	3	3	3	3	2	3	2	2	1
6	28	0	1	1	2	4	2	1	4	5	4	5	1	5	1	3	2	2	1	0

THE PROGRESS SO FAR....

We’ve separated our work into 3 files

01_data_cleaning.ipynb ⏳

02_feature_importance.ipynb

03_exploratory&statistical_analysis.ipynb

THE OVERVIEW OF OUR CODE

1ST FILE

This file does 3 things

Memory Management

Files I/O

Process Scheduling

OS CONCEPT WE APPLY TO THE FIRST FILE

1. Simulate Process Scheduling

Break tasks into separate multiprocessing jobs

2.Optimize File I/O

test and compare file reading speeds to simulate OS-level file handling.

3.Improve Memory Management

Track how much memory each step uses — simulate what an OS does.

MEMORY MANAGEMENT

We use psutil to monitor the memory and CPU usage.

```python
# ✅ Utility: Memory and CPU usage logger
def log_memory_cpu(tag, output_queue):
    process = psutil.Process(os.getpid())
    mem_mb = process.memory_info().rss / 1024 ** 2
    cpu_percent = process.cpu_percent(interval=0.1)
    output_queue.put(f"[{tag}] Memory: {mem_mb:.2f} MB | CPU: {cpu_percent:.2f}%")
```

```python
# ✅ Resource monitoring (3 seconds)
def monitor_resources(output_queue, duration=3):
    process = psutil.Process(os.getpid())
    for i in range(duration):
        mem = process.memory_info().rss / 1024 ** 2
        cpu = process.cpu_percent(interval=1)
        output_queue.put(f"[Monitor {i+1}] Memory: {mem:.2f} MB | CPU: {cpu:.2f}%")
```

FILES I/O

Instead of downloading our file with panda, we use mmap instead. We also track the time it takes.

```python
# ✅ Process 1: Load CSV with pandas and monitor resources
def load_data(output_queue, timing_queue):
    try:
        monitor_thread = threading.Thread(target=monitor_resources, args=(output_queue, 3))
        monitor_thread.start()

        log_memory_cpu("Before reading CSV", output_queue)
        start = time.time()

        if not os.path.exists("raw_data.csv"):
            output_queue.put("[ERROR] raw_data.csv not found.")
            return

        df = pd.read_csv("raw_data.csv")

        elapsed = time.time() - start
        log_memory_cpu("After reading CSV", output_queue)
        output_queue.put(f"[Pandas] CSV loaded in {elapsed:.4f} seconds")
        timing_queue.put(("pandas", elapsed))

        monitor_thread.join()

    except Exception as e:
        output_queue.put(f"[ERROR] Pandas read failed: {str(e)}")
```

Load CSV file with panda

```python
# ✅ Process 2: mmap reading (full file)
def mmap_read(output_queue, timing_queue):
    try:
        start = time.time()
        if not os.path.exists("raw_data.csv"):
            output_queue.put("[ERROR] raw_data.csv not found.")
            return

        with open("raw_data.csv", 'r') as f:
            with mmap.mmap(f.fileno(), length=0, access=mmap.ACCESS_READ) as mm:
                content = mm.read()  # ✅ Read entire file content
                text = content.decode(errors='ignore')  # decode bytes to string

                output_queue.put("[mmap] Full file read successfully.")
                output_queue.put(f"[mmap] Total bytes read: {len(content)}")

        elapsed = time.time() - start
        output_queue.put(f"[mmap] Read completed in {elapsed:.4f} seconds")
        timing_queue.put(("mmap", elapsed))

    except Exception as e:
        output_queue.put(f"[ERROR] mmap read failed: {str(e)}")
```

Load CSV file with mmap

PROCESS SCHEDULING

Lastly, while running these processes, we use multiprocessing to schedule each process.

```python
# ✅ Main execution
if __name__ == '__main__':
    output_queue1 = multiprocessing.Queue()
    output_queue2 = multiprocessing.Queue()
    timing_queue = multiprocessing.Queue()

    print("\n[PROCESS SCHEDULING] Starting subprocesses...")

    p1 = multiprocessing.Process(target=load_data, args=(output_queue1, timing_queue))
    p2 = multiprocessing.Process(target=mmap_read, args=(output_queue2, timing_queue))
    p1.start()
    p2.start()
    p1.join()
    p2.join()

    print("\n--- Pandas CSV Load Outputs ---")
    while not output_queue1.empty():
        print(output_queue1.get())

    print("\n--- mmap File Read Outputs ---")
    while not output_queue2.empty():
        print(output_queue2.get())

    # Collect timings for plotting
    timings = []
    while not timing_queue.empty():
        timings.append(timing_queue.get())

    # ✅ Plot the comparison
    plot_results(timings)

    print("\n[DONE] All subprocesses completed with visualization.")
```

PROCESS SCHEDULING

The result of the run, including the memory usage and the time used for each process

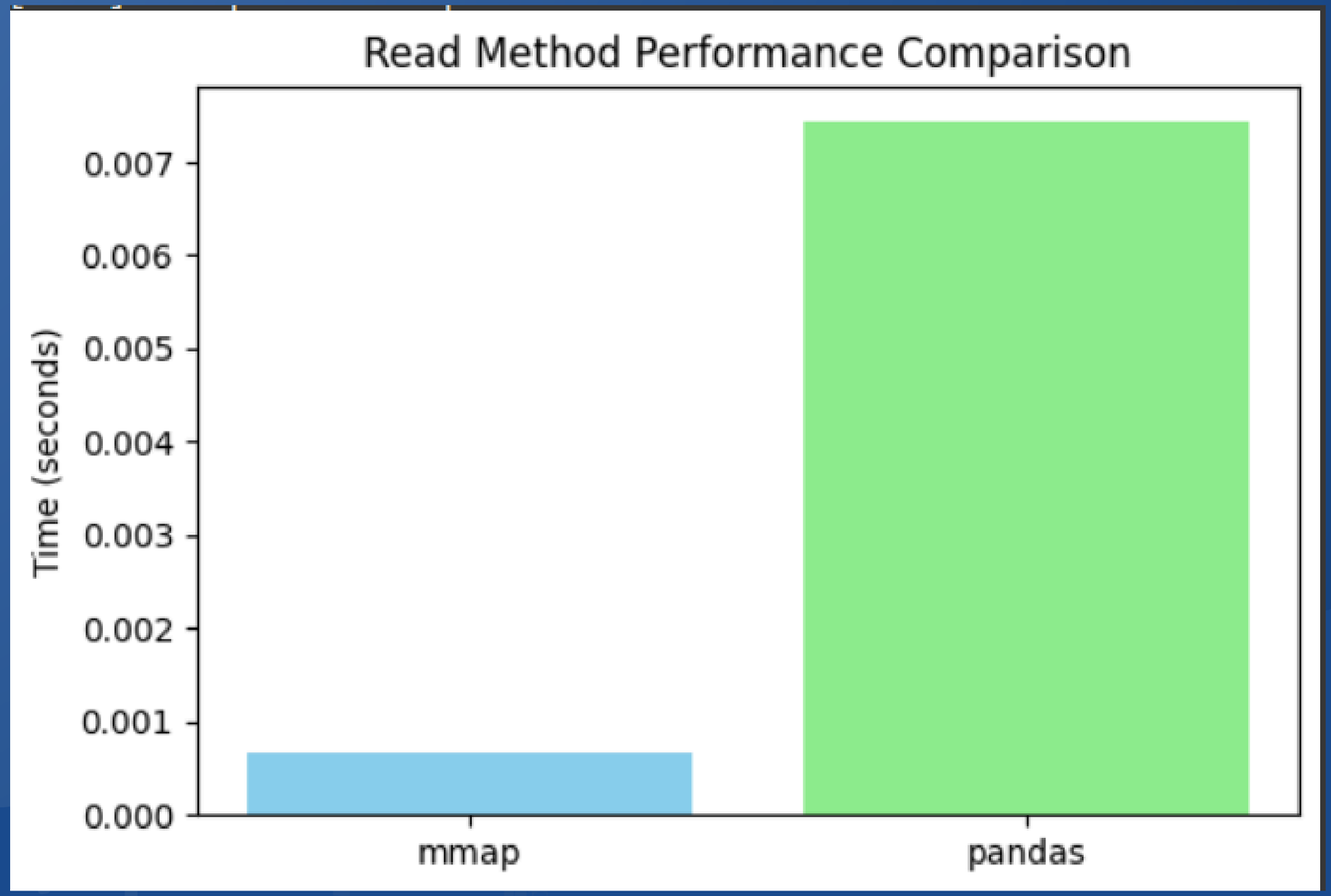
```
[PROCESS SCHEDULING] Starting subprocesses...

--- Pandas CSV Load Outputs ---
[Before reading CSV] Memory: 116.70 MB | CPU: 0.00%
[After reading CSV] Memory: 122.23 MB | CPU: 0.00%
[Pandas] CSV loaded in 0.0074 seconds
[Monitor 1] Memory: 116.70 MB | CPU: 0.00%
[Monitor 2] Memory: 122.23 MB | CPU: 0.00%
[Monitor 3] Memory: 122.23 MB | CPU: 0.00%

--- mmap File Read Outputs ---
[mmap] Full file read successfully.
[mmap] Total bytes read: 48717
[mmap] Read completed in 0.0007 seconds
[ 📊 ] Plot saved as performance_comparison.png
```

RESULT BENCHMARK

Performance different between these two methods.



STAY TUNED!!!