

**RETRIEVING, COMPARING AND PROFILING 2 MACHINE  
LEARNING ALGORITHM SYSTEM RESOURCE USAGE DATA.**

**BY  
ABRAHAM OLU**

## Contents

1.0 INTRODUCTION.....	2
2.0 IMPLEMENTATION .....	3
ALGORITHM 1 .....	3
ALGORITHM 2 .....	4
ALGORITHMS RUNNING SIMULTANEOUSLY .....	5
3.0 CONCLUSION ON RESULT.....	7
4.0 REFERENCES .....	9

## 1.0 INTRODUCTION

A multi-process computer system resources such as; CPU, memory space, I/O use, are shared amongst a number of processes being executed simultaneously on the system. This resource management between tasks is one of the crucial functions of the Operating System running on such a machine(multi-process system).

In this project, the system resource allocation to two running processes is profiled and extracted. These system resources include;

- CPU usage during run-time; such CPU times in user mode, kernel/system mode, interrupt(hardware or software) signal servicing, etc.
- CPU utilization; the percentage utilization of the processor by the process(s).
- Memory usage; main memory allocation and usage by the process(s).
- RSS, non-swapped allocated size to the process in main memory
- VMS, available virtual memory allocated to the process.
- Hard drive usage; the i/o count or read-write bytes to the external memory by the process.
- Number of page faults; OS paging when virtual memory saturation occurs.

These resource usages are collated for both process(algorithms) running individually on the machine and then simultaneously using multi-threads.

The algorithms selected for this project are the simpler of Regression fit algorithms in Machine Learning(ML). Intended to show a comparison between a continuous ML algorithm, Linear Regression and a categorical algorithm, Logistic Regression. ML algorithm is chosen due to suspected high number of read and write bytes, processor and memory usage, etc. in fitting a model with satisfactory accuracy on a given dataset. Intend to investigate as well, if there are differences in resource usage, and if substantial, between these two types of ML algorithm.

The dataset as parameters used in both algorithms is that of passengers onboard the ship, RMS Titanic. We have selected 10 features and randomly selected 891 data instances out of the publicly available dataset. This dataset is cleaned and prepared for model fitting outside the scope of this project, so as to focus only on the algorithms execution system resource use as much as possible. The Linear Regression algorithm is used to predict the continuous variable -age- of passengers onboard the Titanic, while Logistic Regression algorithm is used to predict survival of passengers.

Machine configuration of the machine on which the algorithms were run and system resource monitored is detailed below;

- OS: Microsoft Windows 10 Home
- Machine architecture: intel x64 architecture based machine
- Processor: Intel i7-8750H, 6 Cores, 12 LPs
- Main memory/Available: (16 - 4 GB) dual channel(A partial) 20.0/19.8 GB
- Virtual memory: 22.8 GB
- Page File: 3.0 GB

## 2.0 IMPLEMENTATION

This project implementation has been structured in a form representative of restricted dependent modules callable by the user-accessible code. The restricted module `ProgramClass()` contains both the system resource monitoring code and both algorithms being compared. The user-accessible component, `Main`, contains simple calls and presents the results of the program execution. As pointed out on this machine, there are 12 logical processors available for computation.

### ALGORITHM 1

The Logistic Regression algorithm was executed alone as program 1 with a reported execution time of 0.25 sec. The system resource use reported indicated an OS generated 15 sub-threads for this process in this particular execution, execution time in user-space = 0.33 sec, time in kernel space = 0.37 sec. Reported CPU utilization percentage < 0%, the system was idle for greater than 98%. Resident set size(RSS) in the main memory = 83.43 MB, available virtual memory size(VMS) to program 1 = 436.89 MB. Other system resources reported are as shown in figure 1 below.

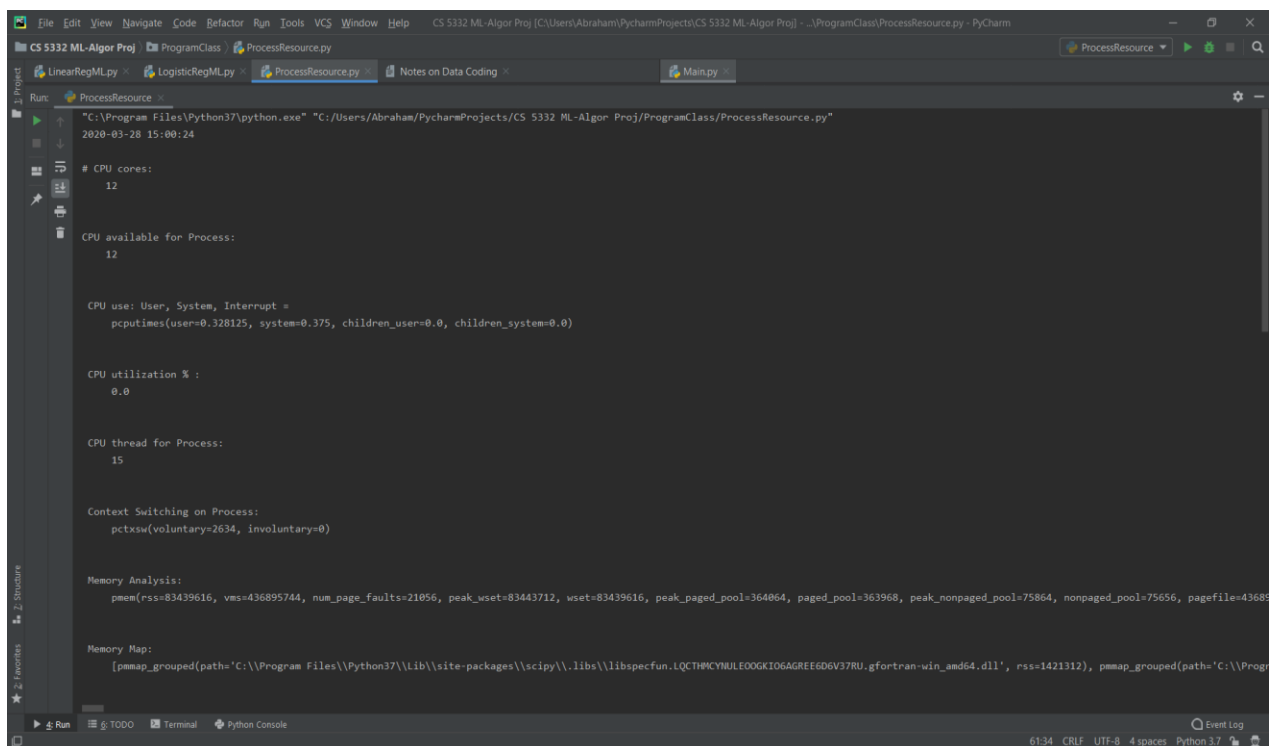


Figure 1 Logistic Regression Algorithm System Resource Use

The reported page fault = 21.05 KB, paged pool, peak memory usage(peak\_wset) approximately equal memory usage(wset) = 83.43 MB, showing there isn't much of memory use increase in program execution. RSS percentage on total memory = 0.3% and the memory map of the program execution and other resource usage reported is shown in figure 1 above.

The data input for the ML algorithm, Titanic dataset is 80 KB in size with 11 features and 899 members. The system disk i/o bytes reported are, read\_byte = 14.7 MB, write\_byte = 26.68 KB. Other secondary memory, hard disk, usages such as the i/o count of the read and write operations calls is shown in figure 2 below.

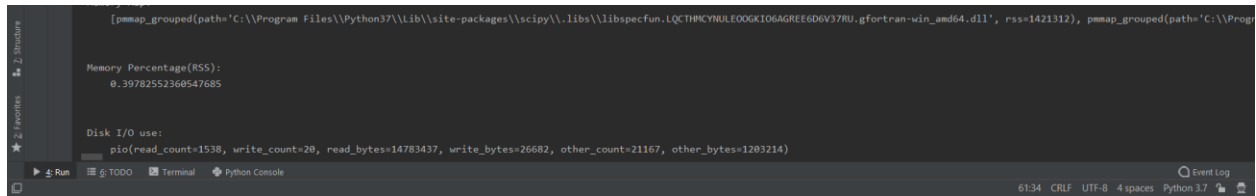


Figure 2 Logistic Regression Disk i/o Usage

Finally, the process profile is reported in figure 3 below. 139 function calls were made, with 138 primitive calls as expected, the 1 recursive call obtained is from a python built-in module len() method used in the program.

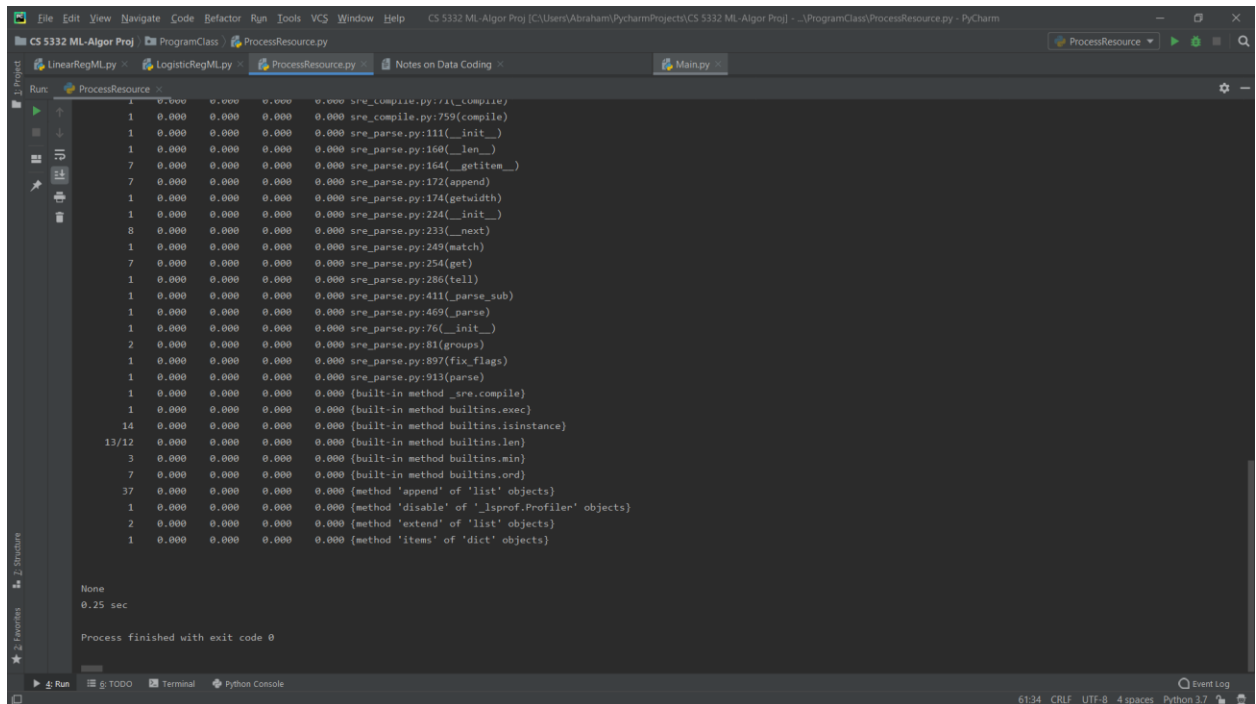


Figure 3 Logistic Regression Algorithm cProfile Result

## ALGORITHM 2

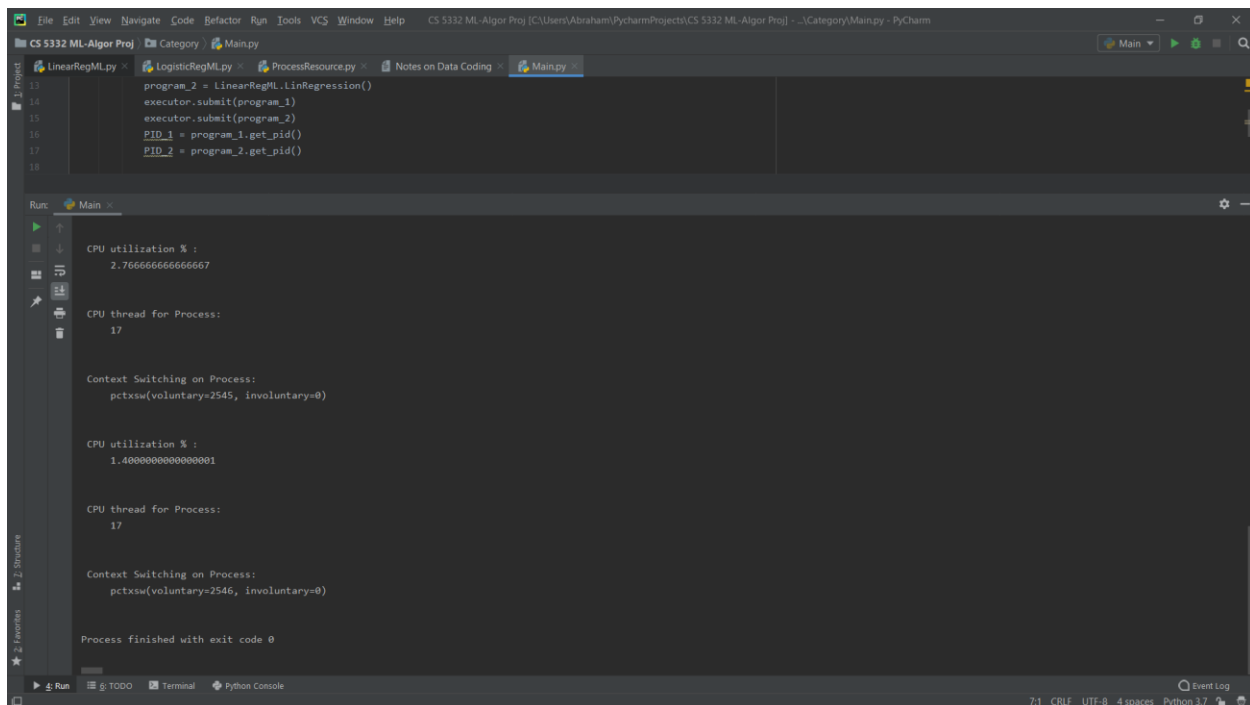
The Linear Regression algorithm is executed similarly to algorithm 1 detailed above. Reported CPU utilization < 0%, reported OS context switching = 1550, execution time in user-space = 0.45 sec, execution in the system space = 0.31 sec.

Main memory usage analysis reported, RSS = 83.72 MB, the reported pagefile or VMS available to the second algorithm = 437.24 MB. Peak\_wset  $\approx$  wset = 83.78 MB, paged\_pool = 363.96 KB and non\_paged\_pool = 75.65 KB. RSS percentage of main memory = 0.3%, and reported page faults = 21.06 MB.

Secondary memory usage analysis reported, on the same dataset, the Titanic Passengers dataset, used in the first algorithm, the expected read\_count and read\_byte reported is similar to the first algorithm. The profile of the second algorithm reports 139 procedure calls and the execution time of the program is 0.23 sec.

## ALGORITHMS RUNNING SIMULTANEOUSLY

Finally, both algorithms are executed simultaneously, using multi-threading with the `ThreadPoolExecutor` python module. The reported system resource usage varies significantly under this condition compared to when both algorithms ran separately. The process CPU time reported in the user space = 0.5 sec and execution time in kernel space = 0.31 sec, the process threads increased to 17, and OS context switching = 2546, approximately double of the single processes. CPU utilization for program 1 = 2.76% and utilization for program 2 = 1.40% as shown in figure 4 below.



```
program_2 = LinearRegML.LinearRegression()
executor.submit(program_1)
executor.submit(program_2)
PID_1 = program_1.get_pid()
PID_2 = program_2.get_pid()
```

Run: Main

CPU utilization % :  
2.766666666666667

CPU thread for Process:  
17

Context Switching on Process:  
pctxsw(voluntary=2545, involuntary=0)

CPU utilization % :  
1.4000000000000001

CPU thread for Process:  
17

Context Switching on Process:  
pctxsw(voluntary=2546, involuntary=0)

Process finished with exit code 0

Figure 4 System Resource Usage Under Simultaneous Execution

In main memory analysis, the reported resource allocated to algorithm 1(program 1) are as follows; RSS = 83.15 MB, VMS = 436.79 MB. The disk read\_count = 1544 and write\_count = 21, as shown in figure 5. The RSS percentage  $\approx 0.4\%$ .

For the second algorithm the system resource usage reported is as follows; RSS = 84.59 MB, VMS = 438.07 MB, peak\_wset  $\approx$  wset = 84.6 MB. The disk i/o process count, for the read\_count = 1544, for the write\_count = 23, the read\_byte = 14.7 MB and other reported system resource use is shown in figure 6.

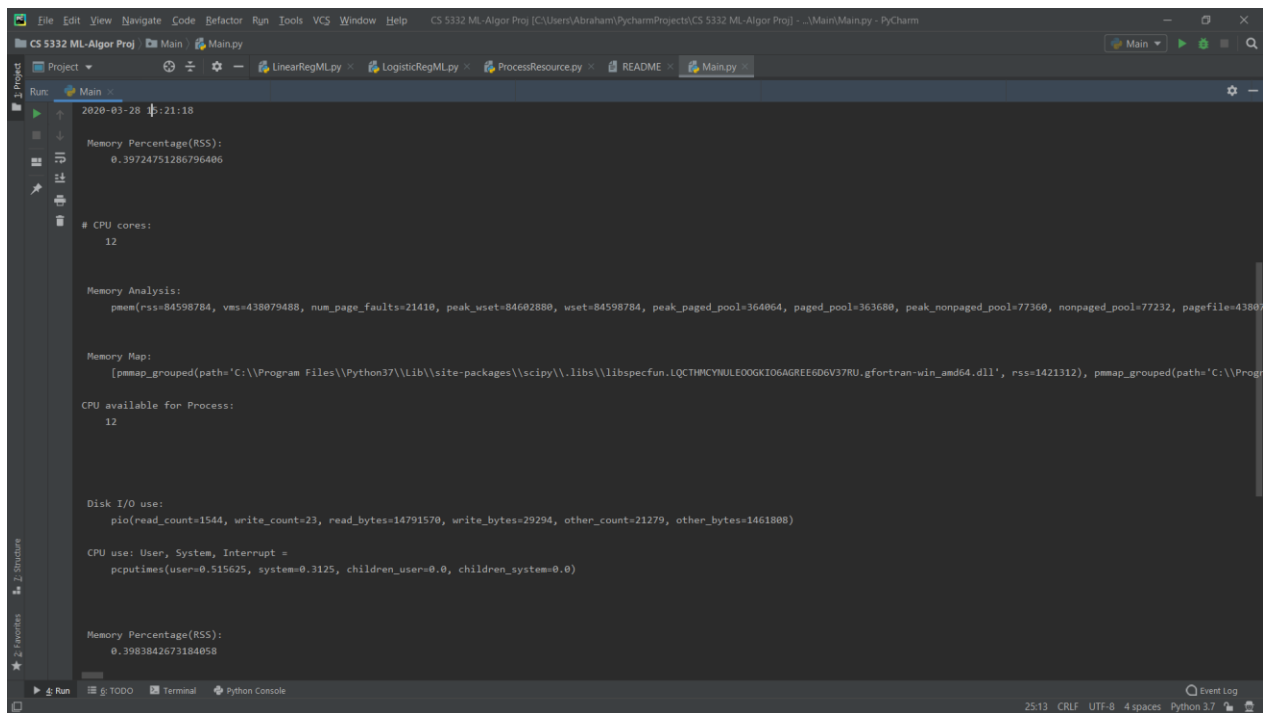


Figure 5 Logistic Regression Algorithm Resource Usage under multi-thread Conditions

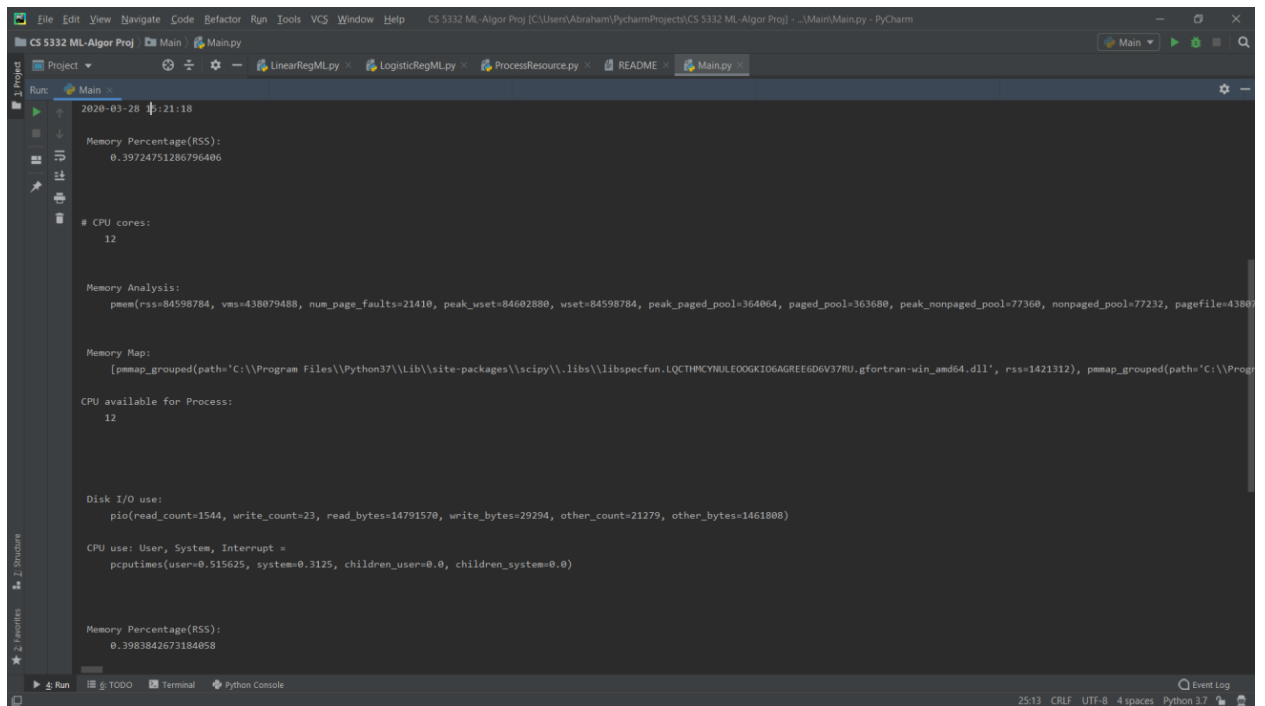


Figure 6 Linear Regression Algorithm Resource Usage under multi-thread Conditions

The RSS main memory percentage reported for the Linear Regression algorithm  $\approx 0.4\%$  likewise.

### 3.0 CONCLUSION ON RESULT

Executing both algorithms simultaneously, the memory usage under this condition is observed to closely mirror that observed in separate execution, with a discrepancy of  $\pm 1 MB$ . However, some other system resources show a marked difference, for example, the disk i/o counter increased from read\_count = 1538, write\_count = 20 in separate executions of the algorithms to 1544 and 23 as seen in figures 5 and 6 above. This additional read/write operations could be associated with possible increase in context switching by the operating system.

The CPU usage also showed a noticeable difference; CPU time in the user space increased to 0.5 sec in concurrent execution from around 0.3 sec in separate execution, while it stayed relatively the same in the system space. Also, 2 additional threads were used in the concurrent execution from the 15 threads used in separate. Importantly, the CPU utilization increased from less than 0% in the separate executions of both algorithms to 2.76% for the Logistic Regression algorithm and 1.40% in the Linear Regression algorithm under concurrent executions.

Furthermore, a separate increased dataset was finally used as input to the second algorithm, Logistic Regression. It was trained on all 891 instances used in the first scenario and tested to predict the same variable on a new 417 instance of the dataset. The process profiling and system resource usage reported stayed relatively constant, however, a discrepancy was noticed when the program was executed after the training dataset had been loaded into memory from a previous execution, in this case, the context switching reported was 1150 and the CPU time in the system space  $\approx 0.3$  sec. On the other hand, on executing after a complete power down, reported context switching = 2622 and CPU times were user = 0.48 sec, system = 0.25 sec as shown in figure below. The data once loaded into main memory is able to serve other processes that call the dataset rather than fetching from disk, this explains the possible reduction of context switching by the system. CPU utilization  $< 0\%$  as expected on running a non-recursive algorithm on a 6 core 12 LPs system.

Finally, checking for energy consumption due to the execution of both algorithms using GreenCode, the algorithms used similar energy (Joules) in executing. GreenCode API reported that the first algorithm used  $\approx 15.32$  J of energy, of which 5.76 J was due to direct execution of the program. On the second program,  $\approx 14.53$  J of energy was used and  $\approx 5.53$  J of energy was used directly due to the execution of the algorithm.



The image shows the PyCharm Run window for a project named 'CS 5332 ML-Algor Proj'. The run configuration is 'ProcessResource.py'. The output displays various system metrics:

```
CPU use: User, System, Interrupt =
pcputimes(user=0.484375, system=0.25, children_user=0.0, children_system=0.0)

CPU utilization % :
0.0

CPU thread for Process:
15

Context Switching on Process:
pctxsw(voluntary=2622, involuntary=0)

Memory Analysis:
pmem(rss=83623936, vms=437080064, num_page_faults=21063, peak_wset=83628032, wset=83623936, peak_paged_pool=368160, paged_pool=368064, peak_nonpaged_pool=75864, nonpaged_pool=75656, pagefile=437080064)

Memory Map:
[pmmmap_grouped(path='C:\\Program Files\\Python37\\Lib\\site-packages\\scipy\\libs\\libspecfun.LQTHWICYMULEOOGKIT06AGREED06V37RU.gfortran-win_amd64.dll', rss=1421312), pmmmap_grouped(path='C:\\Program Files\\Python37\\Lib\\site-packages\\scipy\\libs\\libspecfun.LQTHWICYMULEOOGKIT06AGREED06V37RU.gfortran-win_amd64.dll', rss=1421312)]

Memory Percentage(RSS):
0.398692539711746

Disk I/O use:
pio(read_count=1538, write_count=20, read_bytes=14783401, write_bytes=26679, other_count=21167, other_bytes=1203670)

139 function calls (138 primitive calls) in 0.000 seconds
```

The bottom status bar shows '41/122 CRLF UTF-8 4 sources Python 3.7'.

Figure 7 Logistic Regression Algorithm Resource Usage with Extended Data Input

## 4.0 REFERENCES

1.[https://nbviewer.jupyter.org/github/chaupmcs/show\\_cpu\\_mem\\_info/blob/master/CPU\\_MEM\\_info.ipynb?flush\\_cache=true](https://nbviewer.jupyter.org/github/chaupmcs/show_cpu_mem_info/blob/master/CPU_MEM_info.ipynb?flush_cache=true)

2.<https://www.kaggle.com/jamesleslie/titanic-cleaned-data/data>