An Empirical Study on the Energy Usage and Performance of Pandas and Polars Data Analysis Python Libraries

Felix Nahrstedt*
f.a.nahrstedt@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Mehdi Karmouche*
m.karmouche@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Karolina Bargieł*
k.a.bargiel@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Pouyeh Banijamali*
p.banijamali@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Apoorva Nalini Pradeep Kumar*
a.nalini.pradeep.kumar@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Ivano Malavolta
i.malavolta@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

ABSTRACT

Context. Python's growing popularity in data analysis and the contemporary emphasis on energy-efficient software tools necessitate an investigation into the energy implications of data operations, particularly in resource-intensive domains like data science.
Goal. We aim to assess the energy usage of Pandas, a widely-used Python data manipulation library, and Polars, a Rust-based library known for its performance. The study aims to provide insights for data scientists by identifying scenarios where one library outperforms the other in terms of energy usage, while exploring the possible correlations between energy and performance metrics.
Method. We performed four separate experiment blocks including 8 Data Analysis Tasks (DATs) from an official TPCH Benchmark done by Polars and 6 Synthetic DATs. Both DATs groups are run with small and large dataframes and for both libraries.
Results. Polars is more energy-efficient than Pandas when manipulating large dataframes. For small dataframes, the TPCH Benchmarking DATs does not show significant differences, while for the Synthetic DATs, Polars performs significantly better. We identified strong positive correlations between energy usage and execution time, as well as memory usage for Pandas, while Polars did not show significant memory usage correlations for the majority of runs. There is a significantly negative correlation between energy usage and CPU usage for Pandas.
Conclusions. We recommend using Polars for energy-efficient and fast data analysis, emphasizing the importance of CPU core utilization in library selection.

ACM Reference Format:
Felix Nahrstedt, Mehdi Karmouche, Karolina Bargieł, Pouyeh Banijamali, Apoorva Nalini Pradeep Kumar, and Ivano Malavolta. 2024. An Empirical Study on the Energy Usage and Performance of Pandas and Polars Data Analysis Python Libraries. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3661167.3661203

*These authors contributed equally to the study

1 INTRODUCTION

In this era of increased environmental awareness and concerns about energy consumption, it is imperative to assess the energy efficiency of software tools. The design choices that can be made on the software level, could prolong the life of the hardware, as well as reduce the costs for maintenance of the system. This is particularly important in the context of data science, as the data operations are resource-heavy and can take up a lot of computational power. Thus, our research endeavours to serve as a foundational exploration, providing valuable insights for developers seeking to make informed decisions regarding library selection for their projects.
One of the most popular choices of data libraries for data analysis and manipulation is **Pandas**[1]. Pandas is a Python library used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data. Pandas allows for the analysis of large data and make conclusions based on statistical theories. It simplifies data loading from diverse formats, such as CSV, Excel, and DBMS, by streamlining the process of importing data into Python. Pandas also assists in data cleaning by managing missing values, duplicates, and format transformations. It empowers data exploration with tools for calculating statistics, data visualization, and generating summary reports. Pandas supports data transformation operations such as pivoting, melting, and merging, and enables data analysis through grouping, aggregation, and statistical calculations.
Polars[2] is a Rust-based data manipulation library that provides data processing capabilities similar to Pandas but with a focus on performance. Polars follows a relatively similar syntax to Pandas. Listing 1 shows a sample of some basic data manipulation tasks done by Pandas and Polars.

[1]https://pandas.pydata.org/
[2]https://www.pola.rs/

Listing 1: Examples of Python code using Pandas and Polars

```
# Load data from a CSV file into a DataFrame
#Pandas
pd_df = pd.read_csv('data.csv')
#Polars
pl_df = pl.read_csv('data.csv')
# Filter rows where the 'column_name' is greater than 50
#Pandas
 pd_filtered_df = pd_df[pd_df['column_name'] > 50]
#Polars
 pl_filtered_df = pl_df. filter (pl_df['column_name'] > 50)
# Sort the DataFrame by 'column_name'
#Pandas
pd_sorted_df = pd_df. sort_values (by='column_name')
#Polars
 pl_sorted_df = pl_df. sort ('column_name')
```

Polars can be integrated with Python projects through its Python bindings. Polars also integrates with common Python libraries like numPy and PyArrow. Leveraging the power of Rust, known for its performance and memory-safety[14], Polars excels in managing large datasets. Its core revolves around dataframes, akin to Pandas, which offer a structured way to work with tabular data consisting of rows and columns. Polars empowers users with data manipulation capabilities, encompassing tasks such as data cleaning, filtering, transformation, and aggregation. It also supports operations like selecting, merging, and joining data frames. Polars stands out for its support of lazy evaluation, a key feature where operations on dataframes build a computation plan rather than executing immediately. This approach improves the efficiency of complex data operations. Polars also harnesses the power of multi-core processors to execute data operations concurrently, speeding up data processing tasks significantly. This parallel processing capability is particularly advantageous when dealing with large datasets and complex computations. This parallelism also allows data professionals to take full advantage of modern hardware, making an attractive choice for data processing tasks that demand scalability and speed.

The **goal** of this study is to compare and evaluate the Pandas and Polars data analysis Python libraries in terms of their energy and performance. Polars maintainers published a benchmark against several other solutions claiming it has better performance when running SQL queries, especially compared to Pandas[3]. There also exist other benchmarks comparing the performance of Pandas and Polars using different queries which report similar results[4]. However, none of these benchmarks is systematic or peer-reviewed. We look at whether the improved performance that has been reported by Polars, translates into energy usage. Our contribution specifically studies the energy impact of using Pandas and Polars in different scenarios like dataset size and operations performed.

The **main contributions** of this study are: (i) an empirical assessment of energy and performance of two of the most used Python libraries for data analysis (Pandas and Polars), (ii) a discussion of the main implications of our results for both professionals and researchers, and (iii) the full replication package of the study.

[3]https://www.pola.rs/benchmarks.html

[4]https://www.datacamp.com/tutorial/high-performance-data-manipulation-in-python-pandas2-vs-polars

Study Replicability. The replication package of this study contains all the raw data, code, and instructions needed to replicate/reproduce our experiment: https://github.com/S2-group/ease-2024-pandas-polars-energy-perf-rep-pkg.

2 RELATED WORK

Shanbhag and Chimalakonda [10] examined a wide range of dataframe processing tasks, including input/output operations, handling missing data, row/column operations, and statistical aggregation, in their study to determine the energy consumption of different dataframe processing libraries, including Pandas, Vaex, and Dask. Their research shows that the selection of a library has a significant impact on the energy used for particular procedures, with some libraries using up to 202 times less energy than others. This study highlights the possibility for reducing energy use during the data-focused stages of the ML pipeline, highlighting the need for additional research and advancement in this promising area.
In the field of ML and data processing, many researchers focused their work on improving performance through parallelism. For instance, Widanage et al. [13] introduced Cylon, a high-performance, open-source distributed data processing library that excelled in harnessing parallelism. Cylon seamlessly integrated with established Big Data and ML frameworks, boasting a flexible C++ core and a streamlined data structure. It stood by its unique architecture that used Bulk Synchronous Parallel (BSP) in which each worker or process held a partition of the data that logically belonged to the table.
An important choice concerning energy efficiency is the used programming language, which the work by Pereira et al. inspects by using various programming paradigms to compare 27 popular programming languages [7]. Also, the complex relationship between energy consumption and execution time has been discussed, highlighting that while there can be a correlation between the two, it is not always proportional, and the choice of the most energy-efficient language depends on the specific use case - which is why the benchmarking results for Polars on performance optimization cannot be directly translated for energy efficiency - which will be discussed further in this paper [2,5]. Another paper concerning energy efficiency had been presented by Georgiou et al. [2], where they explore the environmental and financial costs associated with using different deep learning (DL) frameworks. The study compares the energy consumption and runtime performance of six DL models across PyTorch and TensorFlow, using a well-known NVIDIA benchmark. The study reveals statistically significant differences in the cost incurred by the two frameworks in most cases, with TensorFlow generally outperforming PyTorch in terms of energy and runtime efficiency during the training phase, while PyTorch tends to have better performance in the inference phase. The findings suggest that the choice of DL framework can have a substantial impact on the environmental footprint and computational costs of DL applications, highlighting the need for developers to consider these factors in the selection of DL frameworks. As it is not only important what library is used but also how efficiently developers use it, another study explores energy-efficient coding in Python with

[5]https://medium.com/cuenex/pandas-2-0-vs-polars-the-ultimate-battle-a378eb75d6d1

differences observed in operations such as list manipulation, data access, string formatting or sorting [8]. In this study, Reya et al. are highlighting the advantages of efficient data structures, dynamic programming, and the energy efficiency of NumPy over Pandas, while emphasizing better guidance for data-related programming practices to contribute to a more sustainable computing industry.

3 STUDY DESIGN

3.1 Goal and Research Questions

The **primary goal** of this experiment is to analyse whether there exists a difference between the performance of Polars and Pandas and their energy usage. In this study, we focus specifically on Pandas ad Polars since (i) the former is one of the most widely used Python data analysis libraries [3] and (ii) the latter has grown to one of the fastest open-source query engines[6]. In previous benchmarking efforts, Polars exhibited better performance than Pandas, owing to its internal features such as its ability for lazy evaluation, support for multi-threading, caching, and Rust-based implementation. This is in contrast to Pandas, which employs eager evaluation, single-threaded processing and Python-based implementation. Even though the two libraries offer similar capabilities to the developer, they are internally different, and thus there is a need to evaluate their energy usage in performing data manipulation and analysis operations. The results of our experiment are relevant for developers and data scientists performing data manipulation and analysis tasks with Python. The choice of libraries to perform data analysis and manipulation can differ based on the requirements of the project in terms of energy, memory usage, execution time, and CPU utilization, to name a few.

Our **research questions** focus on Data Analysis Tasks (DATs). A DAT is defined as a set of processes conducted to perform operations on dataframes. The main research question of this study are:

[RQ1]: What is the difference between Pandas and Polars in terms of energy usage when performing different data analysis tasks (DATs)? – RQ1 focuses on the energy usage of the two libraries when performing different data manipulation and analysis like I/O operations, handling missing values, etc., to name a few, and gives further guidance to developers on the instances where using one over the other is beneficial.

[RQ2]: How does the energy usage of Pandas and Polars correlate with their performance when performing different data analysis tasks (DATs)?

[RQ2.1]: How does the energy usage of Pandas and Polars correlate with *memory usage* when performing different DATs?

[RQ2.2]: How does the energy usage of Pandas and Polars correlate with *execution time* when performing different DATs?

[RQ2.3]: How does the energy usage of Pandas and Polars correlate with *CPU usage* when performing different DATs?

RQ2 is about the performance correlation to energy usage. As mentioned above the Polars perform better in many cases in terms of performance, and this part examines whether this translates to energy usage in terms of Joules. The comparison would take place across multiple functions and datasets that would allow for precise measurement of both performance and energy usage as observable variables. The RQ2 is further split into three subquestions that tackle the problem from three performance units memory usage (percentage), execution time (seconds), and CPU usage (utilization in percentage).

3.2 Subjects Selection

For this experiment, the population we are considering includes all the existing DATs used by data scientists. From this population, we decided to select a sample including 8 TPCH Benchmarking DATs and 6 Self Written DATs to examine their energy usage and performance in both Pandas and Polars. For this purpose, we are going to divide the experiment into four blocks of subjects. The first two blocks of subjects each includes a set of 8 DATs, addressing small and large dataframe size respectively, taken from a benchmark already done by Polars maintainers[3], which compares Polars-0.18.3 against several other solutions including Pandas-2.0.1. The TPCH benchmark performs DATs on different tables that follow two groups of sizes "small" and "large". The sizes are mentioned in the Table 1, the small dataframe size being 6 million and the large dataframe size being 60 million.

Table 1: Subjects of the experiment

Subject	Source	Data Frame Size
TPCH Benchmark DATs	Official Polars Benchmarks	6 million rows
TPCH Benchmark DATs	Official Polars Benchmarks	60 million rows
Synthetic DATs	As in [10]	2 million rows
Synthetic DATs	As in [10]	15 million rows

Table 2 depicts the details of the operations of each DAT that is part of the TPCH Benchmarking suite. Further details on the TPCH Benchmarking DATs can be found in the replication package of this study. According to the results from this benchmark, Polars performs better than Pandas in terms of speed for the examined DATs. Therefore, we aim to evaluate and compare the energy usage of the same operations. This allows us to evaluate if the better performance in terms of execution time reported by Polars translates into more efficient energy usage. For the third and the fourth block of subjects, we have another set of 8 DAT each, that we write ourselves, taking inspiration from [10]. In fact, we use real-world data to create our synthetic dataframes with the particular size we desire from the US census dataset; in other words, we augment the existing real-world data to meet our size requirements. As such, the small synthetic dataframe has 2 million rows, while the large one has 15 million rows. We focus on the most popular data analysis operations commonly used by data scientists. As mentioned in [10], these operations include tasks such as processing of files, removing columns, combining dataframes, eliminating duplicates, merging dataframes, selecting subsets, sampling, and performing aggregation operations.

Table 2: The TPCH DATs used in this experiment

Task	Description
Q1	Group by, aggregation function for statistics, sort values
Q2	Merge tables, group by and sort
Q3	Merge tables, group by, sum, and sort
Q4	Group by, count, and sort values
Q5	Merge tables, group by, sum and sort Group by, sort values
Q6	Sum, Group by, sort values
Q7	Group, merge, concatenate, group by and sort by, sort values
Q8	Merge tables, group by and sort

[6]The popularity of Polars is growing significantly, its GitHub repository has been forked 1.5k and starred 25.7k times as of April 2024 – https://github.com/pola-rs/polars

The decision to further divide the two blocks by dataframe size is made due to the fact that grey literature[7]claims that Polars can handle larger data frames better than pandas before running into out-of-memory errors. The study by Shanbhag and Chimalakonda investigates and compares energy usage of three popular dataframe processing libraries, namely Pandas, Vaex, and Dask for some of the current most popular DATs [10]. After preparing the samples we can start measuring and comparing energy usage and the performance metrics including execution time, memory usage and CPU utilization of the libraries. Additionally, we only look at DATs that exist in both libraries with similar functionalities so we can only focus on the comparison of the energy usage and the relevant performance metrics correlation. Table 1 summarizes our selected subjects.

Table 3: The synthetic DATs used in this experiment

Data task name	Description
FileTypes (FT)	Read csv file and create new json and parquet
InputOutput (I/O)	Read and write json, parquet and csv files
MissingData (MD)	Check for missing data, drop missing data, fill None with default value
StatisticalAggregation-MinMaxUnique (SAMMU)	Calculate maximum, minimum, and unique values
StatisticalAggregation-SumMean (SASM)	Calculate count, sum and mean values
ViewData (VD)	Displaying head, tail, dataframe type, generate descriptive statistics and shape of data frame

The Synthetic DATs are further categorized into 6 different categories that target most commonly used data analysis tasks. The details of this categorisation are explained in the Table 3.

3.3 Experimental Variables

To answer RQ1 and RQ2, we consider the library used to perform DATs as the **independent variable**. The two treatments for our independent variable are the two libraries of interest: Pandas and Polars. For our RQ1, the **dependent variable** is the *energy usage* in Joules in performing DATs. On the other hand, since our RQ2 correlates the energy usage and different performance measures of the two libraries in performing DATs, the dependent variables for RQ2 are: (i) *energy usage*, (ii) the average *memory usage* in performing DATs in percentage, (iii) *execution time* in seconds, and (iv) *CPU usage* in performing DATs in percentage. As stated in Section 3.2, we have four different blocks of subjects: the DATs from the official Polars Benchmark [3] and the synthetic DATs inspired from the study by Shriram Shanbhag and Sridhar Chimalakonda, each with two different dataframe sizes [10]. We have chosen the *DATs source* and *dataframe size* as our **blocking factors**.

3.4 Experimental Hypotheses

The null hypothesis for RQ1 states that the mean energy usage $\overline{x}$ of Polars and Pandas is equal, while the alternative hypothesis states that their mean energy usage differs.

$$H_0^1 : \overline{x}_{\text{Polars}} = \overline{x}_{\text{Pandas}} \text{ (measured in Joules)} \tag{1}$$

$$H_A^1 : \overline{x}_{\text{Polars}} \neq \overline{x}_{\text{Pandas}} \text{ (measured in Joules)} \tag{2}$$

For RQ2, we formulate the following set of hypotheses that tackle our dependent variables of Memory usage, CPU usage, and Execution Time. The set of null hypotheses states that there is no significant correlation between performance units (memory, CPU usage, and execution time respectively) and energy usage. Thus, the sample correlation coefficient between energy usage and respective performance units equals zero for Pandas and Polars.

$$H_0^2 : \forall r_{L/x+e} = 0, \quad L \in \{\text{Pandas, Polars}\}, \quad x \in \{\text{m, c, t}\} \tag{3}$$

$$H_A^2 : \exists r_{L/x+e} \neq 0, \quad L \in \{\text{Pandas, Polars}\}, \quad x \in \{\text{m, c, t}\} \tag{4}$$

where, r = Sample correlation, L = Python Library (Polars or Pandas), m = Memory usage in percentage, c = CPU usage in percentage, t = Execution time in seconds, e = Energy usage in Joules.

3.5 Experiment Plan

After identifying our subjects and variables, and eliciting our hypotheses, we proceed to design our experiment. Since we have four different blocks in our experiment as described in the above sections, each block has a one-factor-two-treatments design type in which the factor is the Python library and the two treatments are Polars and Pandas. Additionally, for our synthetic DATs block, we are using the data from the US Census Demographic Dataset available on Kaggle[8]. In addition, we conduct a full factorial design to assess all combinations possible within each block. Lastly, we have 8 DATs for the first and second blocks of the experiment each and 6 DATs for the third and fourth blocks, which results in 28 DATs. This results in generating all the raw data related to our experiment such as execution time, CPU usage, memory usage, and energy usage, before conducting further analysis.

We provide an overview of the experiment execution plan in Table 4. Each run in the execution plan is a combination of treatment (namely the Pandas or Polars library), the selected subject (one DAT from either TPCH Benchmark or synthetic), and the dataframe size. Every run (comprising treatment, DAT, and dataframe size combinations) is conducted 10 times in a random order to reduce the likelihood of statistical errors due to fluctuations in the collected measures. For instance, a single run involves executing one of the DATs (for instance the File Types (FT) task) on one of the blocks of experiment (for instance synthetic DATs) with one of the treatments (for instance Pandas) and a dataframe size of 2 million. With a repetition value of 10 taken into account, a total of 560 runs are conducted in our experiment.

Table 4: Experiment execution plan

Treatment	Subject	DAT name	Dataframe size
Pandas	TPCH Benchmark DATs	{Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8}	6 million rows
Polars	TPCH Benchmark DATs	{Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8}	6 million rows
Pandas	TPCH Benchmark DATs	{Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8}	60 million rows
Polars	TPCH Benchmark DATs	{Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8}	60 million rows
Pandas	Synthetic DATs	{FT,I/O,MD,SAMMU,SASM,VD}	2 million rows
Polars	Synthetic DATs	{FT,I/O,MD,SAMMU,SASM,VD}	2 million rows
Pandas	Synthetic DATs	{FT,I/O,MD,SAMMU,SASM,VD}	15 million rows
Polars	Synthetic DATs	{FT,I/O,MD,SAMMU,SASM,VD}	15 million rows

[7]https://medium.com/cuenex/pandas-2-0-vs-polars-the-ultimate-battle-a378eb75d6d1

[8]https://www.kaggle.com/datasets/muonneutrino/us-census-demographic-data

3.6 Data Analysis

Data Exploration. Initially, we analyze the collected data using descriptive statistics, including tools like violin and density plots to gain insights into energy usage and performance metrics. For hypothesis 2, scatter plots are used to assess correlation visually.

Checks of assumptions. The data is first assessed for normality via Q-Q plots. Then, we apply the Shapiro-Wilk test for the same purpose, with a significance level set at 0.05. In case of non-normally distributed data, we attempt data transformation. Specifically, the Square root, logarithmic, and exponential data transformations are used. For both the t-test as well as for the Wilcoxon Rank-Sum test, other assumptions need to be checked. Concerning the paired t-test, it is crucial to note that the data is both continuous and independent. This independence arises from the utilization of distinct libraries. This characteristic also holds true for the Wilcoxon Rank-Sum test.

Hypothesis testing and effect size estimation. If the data conforms to a normal distribution for H^1, paired t-tests are employed to assess whether the mean of the differences of energy usage between the two samples is statistically equivalent to zero. For assessing the correlation (H^2) between the outcome variables between the two treatments, if both samples are normally distributed, the Pearson correlation will be used to test for a significant correlation. In the event of a significant disparity between the two treatments (H^1) as indicated by the tests, Cohen's d is applied to gauge the effect size. In case of non-normally-distributed data, for H^1 we employ the non-parametric Wilcoxon Rank-Sum test to evaluate potential differences between the samples. If one or both dependent variables for the sub-research questions of H^2 are not normally distributed, the Spearman's rank correlation is used, which can handle non-normal data. If a significant difference is detected between the treatments (H^1), we turn to Cliff's delta to quantify the effect size, shedding light on the magnitude of the observed effects.

4 EXPERIMENT EXECUTION

We run our experiment on a single Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-84-generic x86 64) Server with the following hardware specification: CPU: 2 x (2.1Ghz x 8 cores) - Intel Xeon, RAM: 384Gb, Hard Drive: 36Tb. The experiment is orchestrated via Experiment Runner, a generic framework to automatically execute measurement-based experiments [5]. We use Python3 across all runs and the following versions of the targeted libraries: Pandas-2.1.1 and Polars-0.19.7.

We measure energy in each run of the experiment via PowerJoular [6]. This tool has been a reliable tool previously used in research efforts by other research groups across many other similar experiences - especially in the IOT Domain [1, 9, 12], as it provides a high level of precision. PowerJoular monitors the power consumption of CPU and GPU for PCs. We access the first module of software that provides monitoring for PC/servers: the Intel RAPL through Linux Power. The measures provided by PowerJoular allow us to capture the energy usage in Joules, as well as the CPU utilization. As a supporting tool, we also use PS[9], a Linux command line that allows us to capture the memory usage in percentage using the command line. For accessing more precise information about working with time in Python queries we use Python's time module. Finally, we use R and Rstudio for analysing the collected data.

[9]https://www.man7.org/linux/man-pages/man1/ps.1.html

5 RESULTS

5.1 Data Exploration and Normality Checks

Following the run of the TPCH Benchmark DATs and Synthetic DATs, a initial examination of the data has been conducted, which will be explained thoroughly here. In each trial, four types of data have been collected: the amount of energy used (measured in Joules), memory usage (in percentages), CPU usage (in percentages), and the time it took for the execution (in seconds). It is important to note that the experiments were carried out on a CPU with 16 cores, so the highest possible CPU usage percentage is 1600. Tables 5 and 6 present the mean, median, and standard deviation values for our experiments for each combination of library type and data frame size. Just by looking at the data, it can already be observed that the mean energy usage values for Polars are lower compared to that of Pandas for almost all subject blocks. For large dataframe sizes especially, the average execution time is significantly faster for Polars.

As visible in the density plots for the Synthetic DATs (see Figure 1), the libraries behave differently with respect to the used energy while following a rather bimodal distribution (both for Small and Large data frame size). This distribution visible across both libraries is the result of the DATs variability, as some tasks always take less energy, while others have higher energy usage. For the TPCH Benchmark DATs, visual assessment is more ambiguous, as the distributions overlap while also being multimodal. For both dataframes it is worth mentioning, that visually the probability for Polars is more clustered within certain energy usage intervals while Pandas tends to use small and large amounts of energy more evenly distributed. When comparing the standard deviation of Pandas and Polars, it can be observed that Pandas has a value 3 times higher for the large TPCH dataframe and almost 5 times higher for the Synthetic DATs large dataframe. While Pandas for the TPCH Benchmark DATs dataframes tends to have a negative skew according to the boxplots, Polars is less skewed, but more in a positive direction. For the Synthetic DATs dataframe, less skew is observable while keeping similar tendencies as in the TPCH Benchmark DATs dataframe. While the multimodality is clearly visible within the violin plots, it is noteworthy that large data for Polars within the TPCH Benchmark DATs dataframe has a peculiarly distinct spike of lower energy usage.

5.2 Checks of assumptions

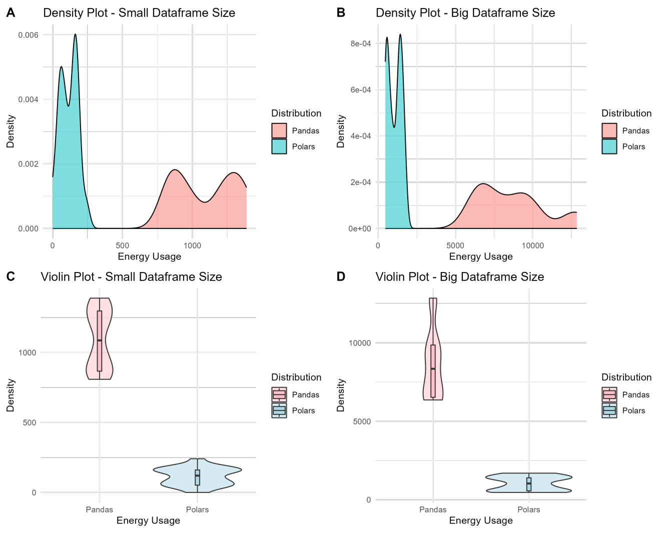
Upon visual inspection and by computing skewness, we determined that employing square root transformation for moderately positively skewed data, logarithmic transformations for strongly positively skewed data, and exponentiation with factors of 2 and 3 for moderately and strongly negatively skewed data, respectively, did not yield any noteworthy impact in the pursuit of achieving normality in the data except for the small dataframe when used with the Pandas library (an improvement from 0.02 to 0.08). This has been tested with the Shapiro-Wilks Test. The results for normality testing show that, even though rejecting the null hypothesis as none of our data shows p-values obtained higher than the predetermined significance level of 0.05 without applying transformation, some data is leaning stronger towards normality than other. This is true for the small TPCH Benchmark DATs (p-value of 0.02) and

Table 5: Measures of Central Tendency and Variability for Synthetic DATs

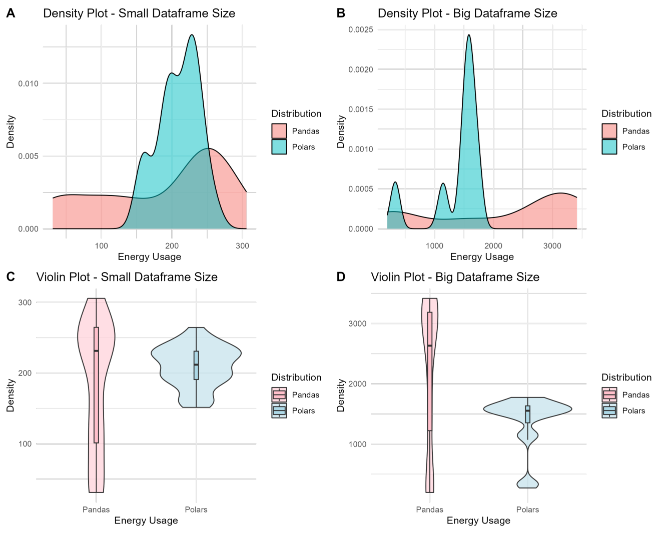
Synthetic DATs	Dataframe size	Energy usage			Memory usage			Execution time			CPU usage		
		Mean	*Median*	*SD*	*Mean*	*Median*	*SD*	*Mean*	*Median*	*SD*	*Mean*	*Median*	*SD*
Polars	Small	115.00	120.20	63.24	0.29	0.27	0.11	8.11	8.10	3.57	1539.50	1668.40	800.49
Polars	Large	1061.60	1042.40	438.16	1.44	1.44	0.57	64.11	61.96	29.66	1505.90	1680.20	755.04
Pandas	Small	1092.90	1086.50	209.30	0.18	0.1530	0.05	85.77	85.27	16.51	123.50	122.10	3.68
Pandas	Large	8658.00	8346.00	2146.60	1.68	1.496	0.349	676.8	647.10	178.20	104.10	104.00	0.79

Table 6: Measures of Central Tendency and Variability for TPCH Benchmark DATs

TPCH Benchmark DATs	Dataframe size	Energy usage			Memory usage			Execution time			CPU usage		
		Mean	*Median*	*SD*	*Mean*	*Median*	*SD*	*Mean*	*Median*	*SD*	*Mean*	*Median*	*SD*
Polars	Small	209.50	211.80	28.843	0.37	0.37	0.0177	15.26	15.95	1.648	556.40	528.60	68.52
Polars	Large	1382.10	1553.00	434.67	3.14	3.45	0.976	81.84	92.03	23.739	1147.00	1205.20	282.18
Pandas	Small	184.50	231.36	89.74	0.39	0.40	0.053	15.80	20.00	6.993	358.40	281.20	135.25
Pandas	Large	2179.80	2632.10	1240.48	3.42	3.89	1.22	169.69	217.31	96.14	282.60	221.50	124.29



(a) Synthetic DATs



(b) TPCH Benchmark DATs

Figure 1: Energy consumption of Pandas and Polars

Synthetic DATs data (0.003) data when used with the Polars library (which is also visible slightly within the density plots in figure 1). Additionally, we also conducted outliers analysis that determined there are not any significant outliers for our energy usage. We have repeated the above reasoning with performance variables – CPU usage, memory usage, and execution time. However, since we do not explore the means of these values but just the correlation with energy usage, it is the lack of normality identified within our main factor that determined the choice for hypothesis testing.

5.3 Hypothesis testing and effect size

5.3.1 RQ1 – Energy Usage (H^1). To address the first hypothesis by looking at the two different benchmarks with either Synthetic DATs or TPCH Benchmark DATs as well as the different dataframe sizes, several result can be discussed. This is an extension of the visual results that have already been discussed for energy usage of the different experiment settings in the previous section. Therefore, the applied Wilcoxon rank-sum test, a non-parametric test, is specifically chosen because it does not rely on assumptions of normality and is well-suited for comparing two independent groups, which in this case are Pandas and Polars. This test assesses whether there is a statistically significant difference in the distribution of energy usage between the two libraries, which is the dependent variable of interest for H^1 . Additionally, by using non-parametric tests, we ensure the validity of the hypothesis testing process, even when dealing with data that does not conform to normality assumptions and may exhibit skewness and multimodality.

Table 7: Hypothesis 1 Results Table

Dataframe size	Subject block	W-Value	P-Value	Cliff's Delta	Lower CI	Upper CI	Delta
Small	TPCH Benchmark DATs	2562	0.64	0.05	-0.17	0.26	Negligible
Small	Synthetic DATs	3600	2.20×10^{-16}	1	1	1	Large
Large	TPCH Benchmark DATs	4399	4.31×10^{-5}	0.37	0.17	0.55	Medium
Large	Synthetic DATs	3600	2.20×10^{-16}	1	1	1	Large

Table 7 shows the results that have been obtained by the Wilcoxon rank-sum test as well as effect size calculated with Cliff's Delta. Furthermore, lower and upper confidence intervals have been described and will be analysed as described by Macbeth et al. [4]. For the *Small TPCH Benchmark DATs*, the Wilcoxon rank sum test resulted in a W-value of 2562 and a p-value of 0.6422, suggesting that there is no significant difference in energy usage between the Pandas and Polars library in the "Small" dataframe size with the "TPCH Benchmark DATs" dataframe, which means that the null hypothesis for this case cannot be rejected. Regarding the *Small Synthetic DAT*, the test yielded a W-value of 3600 and an

Table 8: Spearman test results for Synthetic DATs

Synthetic DATs	Dataframe size	Energy usage vs Memory usage			Energy usage vs Execution time			Energy usage vs CPU usage		
		Coefficent	*P-value*	*Interpretation*	*Coefficent*	*P-value*	*Interpretation*	*Coefficent*	*P-value*	*Interpretation*
Polars	Small	0.21	0.10	Non significant	0.77	0.00	Significant strong positive	-0.12	0.36	Non significan
Polars	Large	0.05	0.71	Non significant	0.80	0.00	Significant strong positive	-0.12	0.36	Non significan
Pandas	Small	0.56	0.00	Significant positive	0.97	0.00	Significant strong positive	-0.73	0.00	Significant strong negative
Pandas	Large	0.58	0.00	Significant positive	0.99	0.00	Significant strong positive	-0.12	0.36	Non significan

Table 9: Spearman test results for TPCH Benchmark DATs

TPCH DATs	Dataframe size	Energy usage vs Memory usage			Energy usage vs Execution time			Energy usage vs CPU usage		
		Coefficent	*P-value*	*Interpretation*	*Coefficent*	*P-value*	*Interpretation*	*Coefficent*	*P-value*	*Interpretation*
Polars	Small	0.13	0.27	Non significant	0.58	0.00	Significant strong positve	-0.21	0.80	Non significan
Polars	Large	0.71	0.00	Significant positve	0.90	0.00	Significant strong positve	-0.17	0.14	Non significan
Pandas	Small	0.45	0.00	Significant positve	0.88	0.00	Significant strong positve	-0.88	0.00	Significant strong negative
Pandas	Large	0.89	0.00	Significant positive	0.98	0.00	Significant strong positve	-0.88	0.36	Non significan

extremely low p-value (2.2×10^{-16}), indicating a highly significant difference in energy usage between Pandas and Polars datasets, which results in rejecting the null hypothesis. The Cliff's Delta is 1, showing a high and significant difference, and the confidence intervals (0.9996 to 1) are very narrow, indicating high precision in the estimated effect size. Within the *TPCH Benchmark DATs* large dataframe, the test resulted in a W-value of 4399 and a small p-value (4.313×10^{-5}), indicating a statistically significant difference in energy usage between the two dataframe size, which means rejecting the null hypothesis. Cliff's Delta of 0.3747 suggests a moderate effect size, and the confidence interval (0.1742 to 0.5454) shows that there is a range within which this effect size is likely to fall. Similar to the *small* subject, the *large* subject with the *Synthetic DATs* dataframe shows highly significant results and an extremely low p-value, indicating a significant difference in energy usage and the rejection of the null hypothesis. The Cliff's Delta is 1, indicating a substantial difference, and the confidence intervals are very narrow, showing high precision in the estimated effect size.

5.3.2 RQ2 – Correlations (H^2). We begin our analysis by looking at scatter plots of energy usage versus the various performance factors discussed in hypothesis H^2 (see figure 2). In the TPCH Benchmark DATs experiment block, we can already observe that the majority of the scatter plots do not show a strong positive or negative correlation. The exception to the above is the execution time correlation with energy usage which shows that positive correlation for all combinations of dataframe sizes and libraries. A noteworthy point is that there seems to be a negative correlation between Pandas energy usage versus CPU usage. It is also visible that for Polars large dataframe size there is a positive correlation visible for all the performance factors within TPCH block. In the tables below we can observe in data analysis for the values. In the Synthetic DATs experiment, also the same trends can be observed. The majority of the scatter plots in figure 2 do not show a positive or a negative correlation. The exception again is for execution time which shows an even less strong positive correlation compared to TPCH Benchmark DATs. There is also a strong positive correlation visible for memory and energy correlation of Pandas large dataframe.

As shown in Table 8, the results of the Spearman test has been presented. We present the two values for each of the dataframe size and library pair, namely the P-value of correlation and the correlation coefficient. An additional column with the interpretation of the p-value has been added, to mark the significance level of the results. The results should be interpreted as follows. In the cases where the correlation coefficient is high and the p-value is 0.00, we would conclude that there is a strong correlation and that it is statistically significant. This is true for the correlation between energy usage and execution time. For the memory usage in Pandas library, we concluded that even though there is a weak correlation between the two variables, the correlation is statistically significant and cannot be ignored. For energy usage vs. memory usage in Polars library we concluded that there is no correlation between the two variables. Finally, for the energy usage vs. cpu usage , we concluded a statistically significant negative correlation for Pandas. For Polars however we we cannot conclude the statistical significance of this correlation. About the TPCH block, as shown in Table 9, The yielded results remain similar to the ones for synthetic DATs. The correlation between, energy usage and execution time remains strong and statistically significant. For the memory usage in Pandas library, we concluded that even though there is a weak correlation between the two variables for small dataframe, the correlation is statistically significant and cannot be ignored. For large dataset, however, the correlation remains strong and statically significant. For energy usage vs. memory usage in the Polars library for small dataframe, we concluded that there is no correlation between the two variables. However, in this experiment we observed a strong (0.71) and statistically significant correlation for a large dataframe between energy usage and memory usage. For the energy usage vs. cpu usage correlation, we concluded a statistically significant negative correlation for Pandas. For Polars however we we cannot conclude the statistical significance of this correlation.

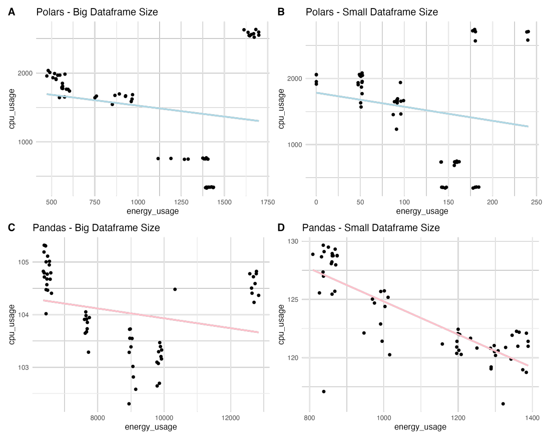
Table 10: Hypothesis results for the TPCH block

	Energy Vs Memory	Energy vs CPU	Energy vs Execution time
Polars Small	$H_0 : \forall r_{Polars/m+e} = 0$	$H_0 : \forall r_{Polars/c+e} = 0$	$H_0 : \forall r_{Polars/t+e} = 0$
Polars Large	$H_0 : \forall r_{Polars/m+e} = 0$	$H_0 : \forall r_{Polars/c+e} = 0$	$H_0 : \forall r_{Polars/t+e} = 0$
Pandas Small	$H_0 : \forall r_{Pandas/m+e} = 0$	$H_0 : \forall r_{Pandas/c+e} = 0$	$H_0 : \forall r_{Pandas/t+e} = 0$
Pandas Large	$H_0 : \forall r_{Pandas/m+e} = 0$	$H_0 : \forall r_{Pandas/c+e} = 0$	$H_0 : \forall r_{Pandas/t+e} = 0$

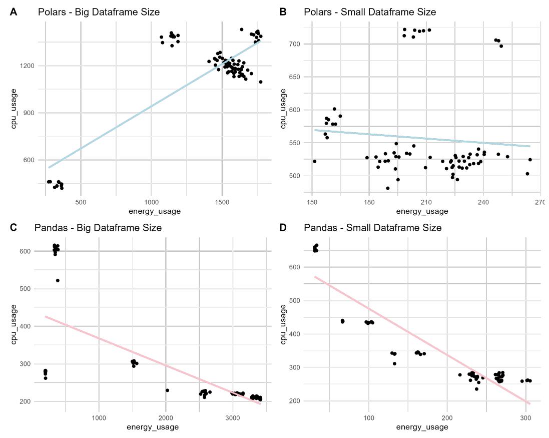
Table 11: Hypothesis results for the DAT block

	Energy Vs Memory	Energy vs CPU	Energy vs Execution time
Polars Small	$H_0 : \forall r_{Polars/m+e} = 0$	$H_0 : \forall r_{Polars/c+e} = 0$	$H_0 : \forall r_{Polars/t+e} = 0$
Polars Large	$H_0 : \forall r_{Polars/m+e} = 0$	$H_0 : \forall r_{Polars/c+e} = 0$	$H_0 : \forall r_{Polars/t+e} = 0$
Pandas Small	$H_0 : \forall r_{Pandas/m+e} = 0$	$H_0 : \forall r_{Pandas/c+e} = 0$	$H_0 : \forall r_{Pandas/t+e} = 0$
Pandas Large	$H_0 : \forall r_{Pandas/m+e} = 0$	$H_0 : \forall r_{Pandas/c+e} = 0$	$H_0 : \forall r_{Pandas/t+e} = 0$

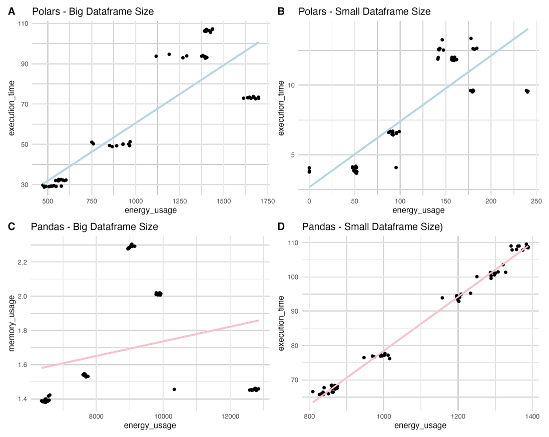
The results regarding the hypothesis, for respective dependent variables are presented in Tables 10 and 11. Out of the 12 hypotheses examined in the TPCH Benchmark DATs experiment, 9 are



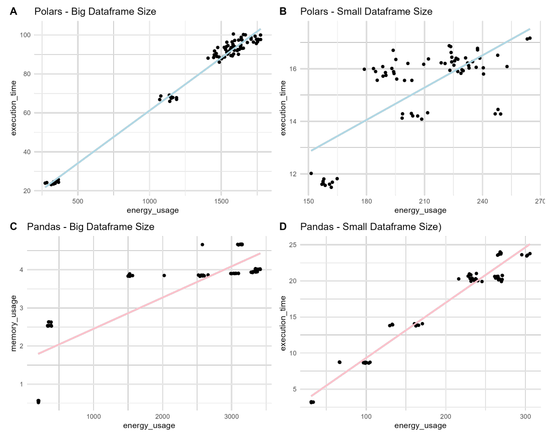
(a) Synthetic Dats - CPU & Energy



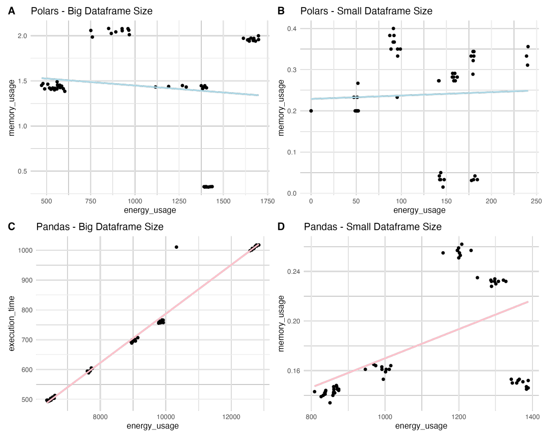
(b) TPCH Benchmark DATs - CPU & Energy



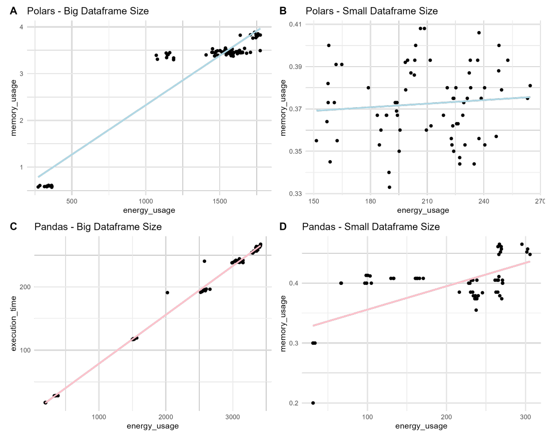
(c) Synthetic Dats - Execution Time & Energy



(d) TPCH Benchmark DATs - Execution Time & Energy



(e) Synthetic Dats - Memory & Energy



(f) TPCH Benchmark - Memory & Energy

Figure 2: Scatter plots for the different benchmarks for Pandas (pink color) and Polars (blue color)

rejected based on their respective p-values, as indicated by a white background in the table. For the Synthetic DATs we have rejected 8 out of the 12 hypotheses variations. The null hypotheses that we failed to reject are highlighted in green. It is important to emphasize that we treat both the query origin and dataframe size as blocking factors. Consequently, we assess each hypothesis independently within these distinct pairings.

In summary, we observed a strong positive and statistically significant correlation between energy usage and execution time, regardless of the library type, data frame size, or the origin of queries. Also, there is a positive significant correlation between energy usage and memory usage for Pandas across both data type sizes and origin of queries. We do not have evidence about the correlation between energy usage and memory usage for Polars in 3 out of 4 cases. Lastly, we observed a statistically significant negative correlation between energy usage and CPU usage.

5.4 Further Observations

Apart from the statistical analyses described above, we have drawn several other noteworthy observations that can serve as valuable directions for future research. The following section provides a qualitative, non-statistical perspective on the outcomes. In the Synthetic DATs segment of the experiment, where we designed six distinct queries for various data analysis tasks, a significant performance discrepancy emerged between Pandas and Polars. Notably, Pandas exhibited noticeably lower performance, primarily attributed to Input/Output (I/O) operations with data. This contrast becomes more evident when comparing it with the TPCH Benchmark DATs, where execution times were lower, likely due to the limited DATs of this nature. Additionally, in the Synthetic DATs segment, we conducted I/O operations on multiple file types (csv, parquet, json), while TPCH primarily utilized parquet files. The inherently smaller size of parquet files and their faster retrieval likely contribute to the improved TPCH block performance.

Moreover, we observed that there is a negative correlation between energy Usage and CPU usage for Pandas, and no substantial one for Polars across both origin of DATs. This discrepancy can be attributed to modern multicore systems that employ effective CPU utilization overload mitigation mechanisms. This might be also a result of Polars employing all the available CPU cores for employing operations, whereas Pandas just utilizes a single core by default.

Conversely, a correlation between energy usage and memory usage was detected, especially in favor of the Polars library. The correlation coefficient was consistently smaller in most of our trials for Polars compared to Pandas. This suggests that Polars may employ more efficient memory management. This not only results in lower memory usage but also translates to reduced energy consumption. Furthermore, our results indicate that Polars tends to offer superior energy efficiency even when memory usage is identical.

6 DISCUSSION

6.1 Summary and Reflections on RQ1

For RQ1, we aimed to explore the difference between Pandas and Polars in terms of energy usage when handling various DATs. Across both experiment blocks, TPCH Benchmark DATs and Synthetic DATs in large dataframe size, we find a statistically significant difference in their energy usage. Remarkably, **Polars emerges as the more energy-efficient choice when compared to Pandas for large dataframe sizes** (15 million and 60 million rows). In the case of small dataframe sizes, different results were obtained between TPCH Benchmark DATs and Synthetic DATs. On the one hand, we obtained statistically significant evidence that Polars performs better for small dataframe sizes in Synthetic DAT experiment. On the other hand, for the TPCH Benchmark DATs for small dataframe size, we could not find a statistically significant difference in the mean energy usage between Pandas and Polars. Additionally, although we did not compare our DATs of different nature and application directly, we observed a better performance and energy efficiency for Polars especially for large dataframes and for the following task types: Input/output operations, handling missing data, data manipulation operations,statistical aggregation operations. The potential extension of this analysis can include investigation of more complex DATs such as AI-related DATs.

6.2 Summary and Reflections on RQ2

For RQ2 and its associated sub-research questions, we look into the correlation between energy usage and the various performance metrics, including CPU Usage, Memory Usage, and Execution Time. Our investigations reveal relevant insights into the correlation between energy usage and performance.

Energy Usage and Execution Time: Our analysis demonstrates **a significant positive correlation between Energy usage and Execution time** for both TPCH Benchmark DATs and Synthetic DATs experiment blocks, irrespective of dataframe size. This correlation not only validates the benchmarking conducted by Polars but also underscores the real-world relevance of energy usage in relation to task completion times.

Energy Usage and Memory Usage: When exploring the correlation between Energy and Memory usage, Pandas exhibited a statistically significant positive correlation across all experiment blocks. In contrast, for Polars we did not reject the null hypothesis in three out of four experiment blocks, suggesting that the correlation was not statistically significant. Intriguingly, we observed a significant correlation for Polars, specifically within the TPCH experiment block for large dataframe sizes. This variance emphasizes the **importance of context and further research in understanding the memory usage of data analysis Python libraries**.

Energy Usage and CPU Usage: The analysis reveals a statistically significant negative correlation between energy usage and CPU usage for Pandas in all experiment blocks. In contrast, Polars does not display a significant correlation for all experiment blocks. This observation is particularly relevant given the inherent differences in CPU core utilization between the two libraries. Pandas by default utilizes a single CPU core, whereas Polars is designed to leverage all available CPU cores. Our experiments were conducted on a Linux Server with a substantial CPU configuration (2 x 2.1GHz x 8 cores). The lack of correlation in Polars can be attributed to its effective utilization of all CPU cores.

6.3 Main Implications

We encourage data scientists to use Polars as it is more energy efficient than Pandas when they perform DATs on large dataframe sizes like the ones in our experiment. This is a direct suggestion from our results of RQ1 that Polars is more energy efficient than Pandas for large dataframe size (15 million and 60 million rows) in TPCH and Synthetic DATs block of the experiment. During the conduction of the experiment, we noticed that the Polars library is still new and lacks some of the functionalities that Pandas already offers. Our experiment also validates the claim of Polars maintainers that **Polars has a much shorter execution**

time than their competition, especially Pandas. Thus, we suggest the usage of Polars for performing fast, energy-efficient DATs. This research also lays a solid foundation for future investigations into optimizing and fine-tuning these libraries for improved energy usage and performance. Firstly, **researchers are advised to delve deeper into the specific mechanisms by which Polars effectively utilizes multiple CPU cores, potentially identifying opportunities for more efficient parallel processing in terms of energy usage**. This can be a potential pointer for Pandas developers to implement within the library, as up till now it is only possible with additional features from the Dask library. **Investigating the potential impact of different hardware configurations on energy usage, such as CPU architectures and memory types**, could provide valuable insights for optimizing these libraries for a variety of computing environments. Additionally, conducting a more extensive comparison with other Python data analysis libraries and frameworks could provide a comprehensive understanding of where Polars and Pandas stand in terms of energy usage and performance. This would involve benchmarking against alternative tools like Dask and Modin.

7 THREATS TO VALIDITY

Internal Validity. During our experiment, we made sure to execute all the runs in a randomized order and in an isolated environment; in other words, the server was fully dedicated to the experiment and all the processes run by the OS were the same for all runs. Thus, the effect of time on our experiment is rather negligible. In addition, concerning the synthetic DATs part of the experiment, we established our experiment in a way to repeat the execution of each type of DAT exactly 10 times in order to have a balanced dataset.
External Validity. For our experiment, we used a server that runs on a Linux OS, which is commonly used in data center servers [11]. However, the external validity of the experiment can be compromised due to the limitations of just one setup. The data analysis operations are conducted in various environments and operating systems therefore more heterogeneous environment could have been used. Moreover, we have used the TPCH Benchmark DATs as a decision support benchmark to compare Polars and Pandas as a means to verify the statements given by the Polars community. Moreover, the Synthetic DATs have been defined based on previous research [10] in order to have two blocks in our experiment for a more diverse task representation.
Construct Validity. To avoid inadequate pre-operational explication of constructs, we defined our constructs through the GQM. Moreover, we have stated the goal of the experiment by eliciting all the RQs and explaining them while clarifying how one library can be evaluated and judged as better than the other. Furthermore, we aimed to mitigate the mono-method bias in our experiment by introducing the dataframe size and looking at how the libraries behave when the dataframe size is either small or large.
Conclusion Validity. While statistically analyzing the results of our experiment, we executed the right statistical tests in order to avoid any possible violated assumption, bias, or error. In order to increase the statistical power of our experiment, we aimed to increase the number of data points, in other words, increasing the number of times that our DATs are executed. Additionally, a replication package was established as a means to allow others to reproduce our work and verify our methodology and results.

8 CONCLUSIONS

This study empirically assessed the energy usage and performance of two of the most popular Python libraries for data analysis. In terms of energy usage, our findings show that Polars tends to be more efficient than Pandas while dealing with large dataframes (15 million and 60 million rows). Putting it more precisely, for this type of scenarios, on average Polars consumed (i) about 8 times less energy than Pandas in the synthetic DATs block and (ii) about 63% of the energy needed by Pandas for the TPCH Benchmarking DATs block. Our experiment revealed that energy usage has a strong positive correlation with execution time, regardless of the library, dataframe size, or the origin of the query. There was also a positive correlation between energy usage and memory usage for Pandas across different dataframe sizes and query origins, which was not observed in most cases for Polars. Interestingly, the correlation between energy usage and CPU usage tended to be non-existent for Polars, likely because Polars uses all the available cores of the CPU by default. This observation suggests a promising avenue for future analysis and optimization.

ACKNOWLEDGMENTS

Special thanks to Giuseppe Procaccianti for providing the initial inspiration for this study. This study is partially supported by (i) the European COST programme under action CA19135 (CERCIRAS), (ii) the Individual Travel Grant of the Amsterdam University Fund, and (iii) the Dutch Prize for ICT Research.

REFERENCES

[1] Promise R. Agbedanu, Richard Musabe, James Rwigema, Ignace Gatare, and Yanis Pavlidis. 2022. IPCA-SAMKNN: A Novel Network IDS for Resource Constrained Devices. In *International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*. 540–545.

[2] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. 2022. Green ai: Do deep learning frameworks have different costs?. In *Proceedings of the 44th International Conference on Software Engineering*. 1082–1094.

[3] Pramod Gupta and Anupam Bagchi. 2024. *Introduction to Pandas*. Springer Nature Switzerland, Cham, 161–196.

[4] Guillermo Macbeth, Eugenia Razumiejczyk, and Rubén Daniel Ledesma. 2010. Cliff´s Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica* 10, 2 (jun 2010), 545–555.

[5] Ivano Malavolta, Eoin Martino Grua, Cheng-Yu Lam, Randy De Vries, Franky Tan, Eric Zielinski, Michael Peters, and Luuk Kaandorp. 2020. A framework for the automatic execution of measurement-based experiments on android devices. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 61–66.

[6] Adel Noureddine. 2022. PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools. In *18th International Conference on Intelligent Environments*. Biarritz, France.

[7] Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2021. Ranking programming languages by energy efficiency. *Science of Computer Programming* 205 (may 2021), 102609.

[8] Nurzihan Fatema Reya, Abtahi Ahmed, Tashfia Zaman, and Md. Motaharul Islam. 2023. GreenPy: Evaluating Application-Level Energy Efficiency in Python for Green Computing. *Annals of Emerging Technologies in Computing* (2023).

[9] Michael P Rooney and Suzanne J Matthews. 2023. Evaluating FFT performance of the C and Rust Languages on Raspberry Pi platforms. In *2023 57th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 1–6.

[10] Shriram Shanbhag and Sridhar Chimalakonda. 2023. An Exploratory Study on Energy Consumption of Dataframe Processing Libraries. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 284–295.

[11] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. 2016. United States data center energy usage report.

[12] Linus Wagner, Maximilian Mayer, Andrea Marino, Alireza Soldani Nezhad, Hugo Zwaan, and Ivano Malavolta. 2023. On the Energy Consumption and Performance of WebAssembly Binaries across Programming Languages and Runtimes in IoT. In *Proceedings of EASE 2023* (Oulu, Finland) *(EASE '23)*. ACM, New York, NY, USA, 72–82.

[13] Chathura Widanage, Niranda Perera, Vibhatha Abeykoon, Supun Kamburugamuve, Thejaka Amila Kanewala, Hasara Maithree, Pulasthi Wickramasinghe, Ahmet Uyar, Gurhan Gunduz, and Geoffrey Fox. 2020. High Performance Data Engineering Everywhere. In *2020 IEEE International Conference on Smart Data Services (SMDS)*. 122–132.

[14] Hui Xu, Zhuangbin Chen, Mingshen Sun, Yangfan Zhou, and Michael R. Lyu. 2021. Memory-Safety Challenge Considered Solved? An In-Depth Study with All Rust CVEs. *ACM Transactions on Software Engineering and Methodology* 31, 1 (sep 2021), 1–25.