An Exploratory Study on Energy Consumption of Dataframe Processing Libraries

Shriram Shanbhag
RISHA Lab
Department of Computer Science & Engineering
Indian Institute of Technology Tirupati, India
cs20s503@iittp.ac.in

Sridhar Chimalakonda
RISHA Lab
Department of Computer Science & Engineering
Indian Institute of Technology Tirupati, India
ch@iittp.ac.in

2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR) | 979-8-3503-1184-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/MSR59073.2023.00048

***Abstract*—The energy consumption of machine learning applications and their impact on the environment has recently gained attention as a research area, focusing on the model creation and training/inference phases. The data-oriented stages of the machine learning pipeline, which involve pre-processing, cleaning, and exploratory analysis, are critical components. However, energy consumption during these stages has received limited attention. Dataframe processing libraries play a significant role in these stages, and optimizing their energy consumption is important for reducing environmental impact and operational costs. Therefore, as a first step towards studying their energy efficiency, we investigate and compare the energy consumption of three popular dataframe processing libraries, namely *Pandas*, *Vaex*, and *Dask*. We perform experiments across 21 dataframe processing operations within four categories, utilizing three distinct datasets. Our results indicate that no single library is the most energy-efficient for all tasks, and the choice of a library can have a significant impact on energy consumption based on the types and frequencies of operations performed. The findings of this study suggest the potential for optimization of the energy consumption of data-oriented stages in the machine learning pipeline and warrant further research in this area.**

***Index Terms*—dataframe, data preprocessing, energy efficiency, machine learning pipeline**

I. Introduction

Machine Learning (ML) is a branch of artificial intelligence that focuses on using data and algorithms to make the systems learn about a particular task. The problems addressed by ML applications include but are not limited to computer vision [1], speech recognition [2], natural language processing [3], machine translation [4] and software engineering [5] among others. The core component in the ML algorithm is data, which plays a critical role in ML as it is required to train the models to make predictions [6]. In recent years, there has been a drastic improvement in the performance of ML models trained using vast amounts of data driven by new architectures such as neural networks [7] and the availability of GPU-based robust hardware infrastructure [1], [8]. However, there is also a critical and growing concern about the energy consumption of these models. A study from 2019 by Strubell et al. [9] revealed that a single neural network model with 626,155 parameters has a carbon footprint equivalent to the lifetime carbon footprint of 5 cars. In recent times, many organizations have been developing ML models that are being trained on billions of parameters [10]. Thus, the energy consumption to train the newer ML models is increasing exponentially.

In general, a machine learning workflow [11] comprises "data-oriented" stages such as data pre-processing, cleaning, and exploratory data analysis to handle the data before the training of an ML model. The use of relational databases for these stages comes with a lot of limitations. Often, the data collected is not well-structured, making it challenging to work with database queries [12]. Writing complex queries would require a strong familiarity with the schema, which is not easy in the case of wide tables with many columns [13]. To overcome some of these limitations, "dataframes" have been introduced as an alternative. Dataframe is a data structure that organizes data into rows and columns like a spreadsheet. Over the years, dataframes have become widely popular as they provide an intuitive way of storing and working with data [12], [14]. They are convenient for use in REPL-style imperative interfaces and data science notebooks [15]. Several libraries [16]–[18] have been developed to enable the handling and manipulation of dataframes. These libraries provide a convenient interface to work with dataframes and are widely used for exploratory analysis, data ingest, data preparation, and feature engineering. Dataframes offer several features, including implicit ordering on both rows and columns, by treating them symmetrically. The dataframe manipulation libraries offer data analysis modalities that include relational operators like a *filter, join, transpose* and spreadsheet-like operators such as *pivot*.

The dataframe manipulation libraries are used in the data-oriented stages of the machine learning workflow. *Pandas*[1], one of the most popular dataframe processing library, has been downloaded over 300 million times[2] with its GitHub repository starred over 33k times as of April 2022[3]. ML models have become very data intensive in recent years. As the data gets larger, it cannot fit in the system RAM leading to the need for more efficient, out-of-core dataframe processing methods. This has prompted the need for the development of faster and scalable dataframe processing methods, i.e., the

[1]https://pandas.pydata.org/
[2]https://pypistats.org/packages/pandas
[3]https://github.com/pandas-dev/pandas

2574-3864/23/$31.00 ©2023 IEEE
DOI 10.1109/MSR59073.2023.00048

libraries such as *Dask*[4], *Vaex*[5], *RAPIDS*[6], and many more. These libraries enable scaling either through parallelization or through the use of efficient computation techniques and memory usage. While the emphasis in these libraries remains focused on achieving faster processing and scalability, the energy consumption aspect of these libraries is not discussed.

In recent years, the energy consumed by software systems has gained significant attention from researchers. Several works have observed a link between the design of software and its energy consumption [19], [20]. The lack of knowledge about the energy implications of poor design choices has also been observed by Pinto et al. [21]. In the domain of machine learning induced software systems, the computation and energy demands have been increasing exponentially, which would require a more significant energy production. This has the potential to cause a significant portion of the carbon emissions [22]. Owing to this, many researchers have suggested improving current ML models. There has been a focus on measuring the energy consumption of ML models [23], [24], reducing the energy requirements of training [25], [26], and inferencing [24]. However, there has been limited research on the energy efficiency of the data-oriented stages in the machine learning pipeline. Although Verdecchia et al. [27] have analysed the impact of reduction in size of the dataset and the number of features on energy consumption of training, no study has focused on energy consumption of pre-processing activities of the data-oriented stages. Dataframe processing libraries are crucial in the data-oriented stages for pre-processing tasks. Despite their importance, the energy consumption of these libraries has not been studied in the literature. To fill this gap, as a first step, we perform an exploratory study on the energy consumption of dataframe libraries.

We perform an empirical evaluation of the energy consumption of three popular libraries, namely *Pandas*, *Vaex* and *Dask* for four categories of tasks. Although *Dask* is not a library specifically meant for dataframes, it does contain a dataframe module that can perform operations on dataframes efficiently using computation graphs [17]. The four categories of tasks include *"input-output operations", "handling missing data", "row/column operations"*, and *"statistical aggregation"*. They are elaborated further in Section III.

The research questions addressed in the study are as follows.

- **RQ1: What is the comparative analysis of energy consumption among *Pandas*, *Vaex* and *Dask* for different dataframe processing tasks?** - Here, we wish to investigate the energy consumption of different dataframe processing libraries for a few basic operations. We would also like to explore energy-consumption differences among these libraries for each task.
- **RQ2: Is the most energy-efficient dataframe processing library for a given task also the fastest?** - We aim to investigate whether the library that is most energy-efficient for a particular task is also the fastest in terms of data processing speed, as this is an important consideration for users.

The replication package for the study that includes scripts, results, and summary reports is available at *https://github.com/rishalab/dataframe_libraries_energy_comparison*

In order to investigate *RQ1*, we performed experiments on three datasets from the UCI Machine Learning Repository[7], using the tasks specified in Table I. We used Intel's RAPL interface [28] to measure the energy consumption of each task across ten trials. Furthermore, to address *RQ2*, we recorded the time required to complete each task. To achieve this, we employed the *Pyjoules*[8] library, which allows for the simultaneous measurement of both energy consumption based on Intel's RAPL and the time duration.

The rest of the paper is organized as follows. We provide a brief description of the libraries used in our study in Section II. The tasks addressed in the study are discussed in Section III. The datasets used in the study are presented in Section IV. The experimental settings and the procedure followed are described in Section V. The results of the experiments are presented in Section VI. Section VII discusses the potential threats that may impact the validity of our study. The discussion, future directions, related work, and conclusion are presented in Section VIII, Section IX, Section X, and Section XI, respectively.

II. Dataframe Processing Libraries

The data used in the ML model is structured in a tabular form called dataframes. Dataframe is a two-dimensional labeled data structure with columns of potentially different types. Dataframes provide a standard structure to work with data irrespective of their domains, such as finance, statistics, and social sciences, among many others. Libraries like *Pandas* and *Vaex* provide a rich set of functionalities to manipulate the dataframes, enabling easier data cleaning and preparation. The dataframe libraries considered in this study are open-source Python libraries.

Pandas [16] is an open-source Python library for data manipulation developed in the year 2008. It provides data structures and operations for the manipulation of tabular data. It was originally developed by *Wes McKinney* for working with financial data. In addition to data manipulation, it can also be used for data analysis and visualization. *Vaex* [18], on the other hand, is a Python library for working with lazy out-of-core dataframes on vast amounts of tabular data. *Vaex* claims to be an alternative to *Pandas* that has a more efficient and faster implementation of data manipulation functions using memory mapping, a zero memory copy policy, and lazy computations.

Dask is an open-source library for parallel computing in Python. *Dask* provides support for dataframe processing through a module with an interface similar to *Pandas*. The

[4]https://docs.dask.org/en/stable/dataframe.html
[5]https://vaex.io
[6]https://rapids.ai/
[7]https://archive.ics.uci.edu/ml/index.php
[8]https://pypi.org/project/pyJoules/

module implements a large dataframe out of many *Pandas* dataframes. It uses a threaded scheduler that enables efficient computation on partitioned datasets, thereby minimizing the amount of data held in memory while parallelizing the tasks using computation graphs [17].

III. Dataframe Processing Tasks

This section describes all dataframe processing tasks considered for experiments in this study. Dataframe processing libraries can be used to perform several operations on dataframes, including read/write operations, visualization, and relational and statistical operations. In order to compare the energy consumption of these libraries while working with the dataframes, we used different libraries to perform the same set of operations on three different datasets. We address a few frequently performed tasks, i.e., input/output operations, relational operations, and statistical operations. We picked some of the commonly performed operations on dataframes as tasks for our evaluation. A brief description of each of these tasks along with the task ID and the category is given in Table I.

A. Input/Output Operations

Dataframes can be read and stored in several formats, including CSV, HDF5, Parquet, JSON, and many more. The choice of a format is decided based on the type of data being stored and the type of operations being performed on the data. As an example, storing dataframes in a format like Parquet can be beneficial when used in the case of big data systems like Hadoop or Spark. However, unlike CSV, this storage format is not human-readable.

Since our work focuses on the energy, we compare the energy consumption of input and output operations on different file formats used to represent the dataframes. Specifically, we compare the energy consumption values of reading and writing the dataframes in three popular formats, namely CSV, JSON, and HDF5, using *Pandas*, *Vaex*, and *Dask* libraries.

- CSV stands for *comma-separated values*. CSVs store data as plain text where each row is a line of columns separated by a comma. It is widely used because it is human-readable and can be read from and written to by most data software.
- JSON stands for *JavaScript Object Notation* is another human-readable format that uses text to store data using attribute-value pairs and arrays. Like CSV, JSON can also be readily displayed and edited in simple editors. It also has the advantage of being a preferred format for transmitting data in web applications.
- HDF5 stands for *Hierarchical Data Format 5*, suited for storing and organizing large amounts of heterogeneous data. The data is stored as an internal file-like structure which provides the ability to access different parts of the data randomly. HDF5 offers efficient storage and faster access when dealing with massive data.

B. Handling missing data

Missing data is a prevalent problem in real-world datasets. They are represented using unique value markers such as NaN (Not a Number). It is either caused by lack of collection of existing data or because of the non-existence of the data. They are considered undesirable as they generally interfere with the operations and calculations being performed on rows/columns. This makes handling missing data a frequently performed task in data pre-processing. The functions considered for handling missing data are described in Table I.

C. Row/Column Operations

Dataframes represent the data in the form of a table consisting of rows and columns. Dataframes support relational operations like “join”, “filter”, “groupby”, and so on. They also support reshape and transpose operations on the tables. These operations make it easier to merge data obtained from multiple sources in multiple dataframes and order them in the desired way to enable better exploratory analysis. They also enable the users to apply functions on rows/columns that allow modification and filtering of the data. The missing data handling functions compared in our study are described in Table I.

D. Statistical Aggregation Operations

Statistical aggregation functions enable the users to perform statistical analysis of their data. They can be used for operations like aggregating multiple values under a row or column using a single summary value, computing the correlation and covariance between pairs of rows/columns, finding unique occurrences, count of rows/columns, and so on. These functions are frequently used in exploratory data analysis to get the summary or description of the dataset. The statistical aggregation functions compared in our study are described in Table I.

IV. Datasets

In order to compare the energy consumption of the dataframe processing libraries, we have to perform the same set of operations on the same datasets. Toward this end, we relied on standard datasets from the UCI Machine Learning repository [29]. The energy consumption of the libraries may also vary based on the size of the dataframe they process on. A library that may be more energy-efficient on a smaller dataframe may become less energy-efficient as the dataframes get bigger. In order to account for these variations, we performed our experiments on three different datasets of varying sizes from the UCI repository shown in Table II. It must be noted that the datasets were solely available in CSV format. As their utilization in our experiments necessitated input/output operations in JSON and HDF5 formats, we performed pre-conversion of the CSV files and stored them in both JSON and HDF5 formats. The datasets used are briefly described below.

TABLE I
DATAFRAME PROCESSING TASKS ADDRESSED IN THE STUDY.

Task ID	Task Name	Description	Category
csvin	Read CSV	Read a comma-separated values (csv) file into dataframe	I/O operation
jsonin	Read JSON	Read JSON file into a Dataframe	I/O operation
hdfin	Read HDF5	Read an heirarchical data format 5 (hdf5) file into dataframe	I/O operation
csvout	Write to CSV	Write dataframe into a comma-separated values (csv) file	I/O operation
jsonout	Write to JSON	Write dataframe into a JSON file	I/O operation
hdfout	Write to HDF5	Write dataframe into a heirarchical data format 5 (hdf5) file	I/O operation
isna	Detect missing values	Detect missing values in a given series object	Handling missing data
dropna	Drop missing values	Drop rows/columns containing missing values	Handling missing data
fillna	Fill missing values	Fill missing values with a default	Handling missing data
replace	Replace values	Replace a value with a specified value	Handling missing data
drop	Drop rows/columns	Drop specified labels from rows or columns	Row/Column operations
groupby	Groupby Operation	Group rows that have the same values together	Row/Column operations
concat	Concatenation	Concatenate dataframes along a particular axis	Row/Column operations
sort	Sort values	Sort the values along an axis	Row/Column operations
merge	Merge dataframes	Merge dataframes with a database-join style	Row/Column operations
count	Count cells	Count the number of non-NA cells along an axis	Statistical Aggregation
sum	Sum values	Sum values along an axis	Statistical Aggregation
mean	Average values	Average values along an axis	Statistical Aggregation
min	Minimum value	Find the minimum of the values along an axis	Statistical Aggregation
max	Maximum value	Find the maximum of the values along an axis	Statistical Aggregation
unique	Unique values	Find the set of unique values along the axis	Statistical Aggregation

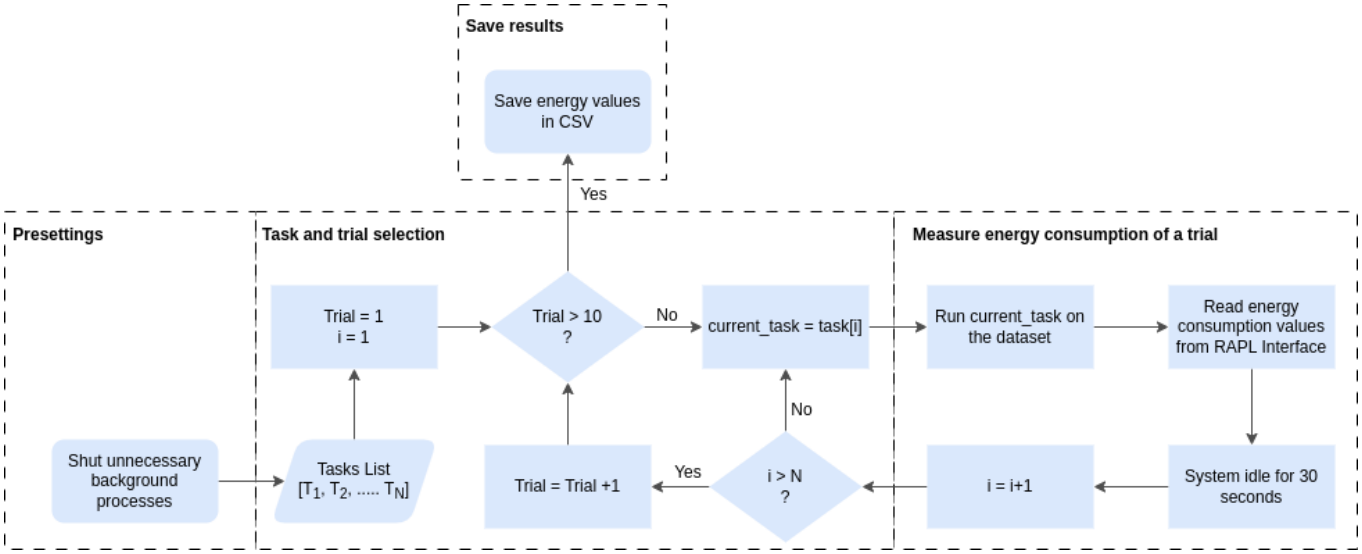

Fig. 1. Flowchart describing the experimental procedure followed in the study.

TABLE II
DATASETS USED IN THE STUDY

Dataset Name	Number of Data Points	Number of Attributes
Adult (D1)	48842	14
Drug Review (D2)	215063	6
US Census Data 1990 (D3)	2458285	68

A. Adult Dataset

The dataset[9] was extracted based on the 1994 census database. The task associated with the dataset is the classification task to determine whether a person makes over 50K a year or not. The dataset contains 48842 data points with 14 attributes. We would refer to this dataset as *D1* in the rest of the paper.

B. Drug Review Dataset

This dataset[10] contains patient reviews on specific drugs along with related health conditions. It also contains a 10-star patient rating of the drug. The dataset has 215063 data points with six attributes. We would refer to this dataset as *D2* in the rest of the paper.

[9]https://archive.ics.uci.edu/ml/datasets/adult

[10]https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29

C. US Census Data (1990)

The dataset[11] contains one percent sample of the selection of individual records from the PUMS (Public Use Microdata Samples) of the complete 1990 census sample. The dataset contains 2458285 data points with 68 attributes. We would refer to this dataset as *D3* in the rest of the paper.

V. Experimental Procedure

A. System and Dataframe Library Information

The experiments were conducted on a system equipped with an Intel Xeon Gold 6226R processor, which featured a clock speed of 2.9GHz, 16 logical cores, and 128GB of main memory. We used Python 3.8.10 as the programming language for all our experiments. The system had Ubuntu 20.04 installed as the operating system. We used *Pandas* version 1.4.4, *Vaex* version 4.13.0, and *Dask* version 2022.9.1 for our study.

B. Intel RAPL and PyJoules

To measure the energy consumption values, we used Intel's Running Average Power Limit (RAPL) interface. RAPL provides an estimation of the global energy consumption of the processor [28]. Several studies related to measuring the energy consumption of software systems have relied on the RAPL interface [30]–[32]. The tool provides energy consumption values of various power domains such as:

- *Package (PKG)*: This domain provides the energy consumption values of all the cores, integrated graphics, and uncore components such as last-level caches and memory controllers.
- *Core*: The energy consumption values of all the CPU cores are provided by this domain.
- *Uncore*: This domain provides the energy consumption values of all the caches, integrated graphics, and memory controllers.
- *DRAM*: The energy consumption values corresponding to the random access memory attached to the memory controller are provided by this domain.

Since the PKG values are inclusive of core and uncore values, we only record the PKG and DRAM energy consumption values in our experiments.

In order to record values given by the RAPL interface by running Python code, we use the *Pyjoules* library. It is a toolkit that measures the energy consumption of a host machine during the execution of Python code. It monitors the energy consumption of specific devices on the host machine, such as the Intel CPU socket package, RAM (for Intel server architectures), Intel integrated GPU (for client architectures), and NVIDIA GPU.

The experimental procedure using a single library is represented as a flowchart in Fig. 1. At the start of the experiment, we shut down all the processes that were not required to run the OS. This was done to ensure minimal variation in the readings due to interference from the processes running in the background.

[11]https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)

Power consumption measurement in the system can be influenced by noise. So one of the methods used to minimize the effect of the noise on the measurements is to run the same experiment multiple times and take the mean value of the measurements for all the experiments as followed in a study by Georgiou et al. [33]. Based on this, we ran the experiment for each operation ten times and recorded the mean value of the energy measurements as shown in Fig. 1. After running each task on a dataset, we let the system remain idle for 30 seconds using the command "sleep". This was done in order to avoid the power tail states [34] influencing the reading and allow the system to reach a stable state again before the start of another task. At the end of each task, the RAPL readings were stored in an excel sheet for further analysis. The information included the time required to run each task and the components' energy consumption values.

The experiment was performed using all three libraries considered for the study. The scripts used for the study, the detailed energy consumption values of the tasks, and their summarized version are available in the replication package.

VI. Results

A. What is the comparative analysis of energy consumption among Pandas, Vaex and Dask for different dataframe processing tasks?

This section discusses the results of the experiments measuring the energy consumption of 3 different libraries on various dataframe operations. The energy consumption measurements for input operations are shown in Table III and output operations in Table IV. The energy consumption of operations on handling missing data is shown in Table V. The energy consumption values of the row/column and statistical operations are shown in Table VI and Table VII, respectively. We use the abbreviation *PKG* for the energy consumption values of all core and non-core components of the processor. The RAPL interface also provides the energy consumption of the core and non-core components separately. Since those are covered under *PKG*, we do not report them separately. We use *DRAM* for energy consumption of the main memory. The RAPL interface provides the energy consumption values in micro-joules (1 micro-joule = 10^{-6} Joules). However, we present them in the results by converting them into Joules. The values are rounded off to one or two decimals for better readability. The energy consumption values in all the tables are mean values from 10 trials of each operation performed as described in Section V. We present only the mean values in the paper. However, the summary reports in the replication package also contain standard deviation and median energy consumption values for each task.

*1) **Input/Output Operations**:* The energy consumption of the dataframe input operation in CSV, JSON, and HDF5 formats for the datasets *D1*, *D2* and *D3* are shown in Table III. *Vaex* consumes the highest amount of energy while Dask consumes the lowest for inputs in CSV format. While the energy consumption of *Pandas* and *Vaex* increase with the size of the dataset for CSV inputs, the energy consumption

TABLE III
MEAN VALUES FOR ENERGY CONSUMPTION (IN JOULES) FOR DATAFRAME INPUT IN DIFFERENT FORMATS. FOR EACH LIBRARY, THE COLUMNS P AND D REPRESENT PKG AND DRAM ENERGY CONSUMPTION VALUES RESPECTIVELY.

	Adult Dataset						**Drug Dataset**						**US Census Dataset**					
	Pandas		Vaex		Dask		Pandas		Vaex		Dask		Pandas		Vaex		Dask	
File	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D
CSV	3.23	0.43	5.73	0.69	0.57	0.08	81.16	7.78	93.67	8.83	0.73	0.10	542.67	48.34	549.79	50.55	0.96	0.13
JSON	20.8	2.29	27.3	2.97	20.9	2.34	950.6	78.6	965.8	79.7	944.9	78.4	7168.3	716.7	8305.3	789.3	9291.7	868.6
HDF5	1.39	0.27	1.31	0.19	1.58	0.22	42.87	4.65	0.96	0.13	1.61	0.23	210.72	21.92	2.52	0.34	1.67	0.23

TABLE IV
MEAN VALUES FOR ENERGY CONSUMPTION (IN JOULES) FOR DATAFRAME OUTPUT IN DIFFERENT FORMATS. FOR EACH LIBRARY, THE COLUMNS P AND D REPRESENT PKG AND DRAM ENERGY CONSUMPTION VALUES RESPECTIVELY.

	Adult Dataset						**Drug Dataset**						**US Census Dataset**					
	Pandas		Vaex		Dask		Pandas		Vaex		Dask		Pandas		Vaex		Dask	
File	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D
CSV	9.38	1.1	13.1	1.50	28.2	2.8	185.6	15.2	178.9	15.3	969.67	91.48	1598.9	128.0	1549.7	133.7	2600.4	359.7
JSON	4.36	0.6	9.32	1.21	24.8	2.6	50.36	5.09	97.09	7.91	905.22	88.20	1519.6	141.3	2220.8	207.0	2636.7	248.5
HDF5	1.89	0.3	20.8	2.56	30.7	3.1	27.69	3.02	30.78	3.86	2555.7	238.3	179.81	19.15	157.53	15.64	342.01	34.43

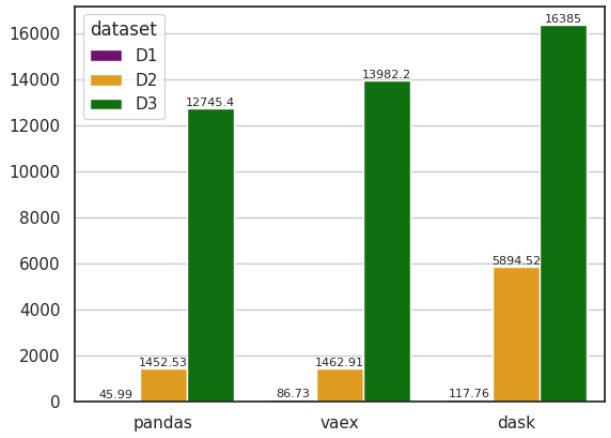

Fig. 2. Cumulative energy consumption (in Joules) of I/O operations of the libraries for both datasets.

of Dask is comparable in all datasets. In the case of HDF5 inputs, we observe that *Pandas* is the most energy-expensive library. *Vaex* consumes the least amount of energy for HDF5 inputs in *D1* and *D2* while *Dask* is the least expensive for *D3*.

Across all libraries, JSON is the most energy-expensive format for data inputs. We also observe that while *Vaex* consumes the highest energy for JSON inputs on *D1* and *D2*, *Dask* consumes the highest energy on *D3*. The energy consumption of JSON inputs increases with input size across the libraries. We can also observe that the energy consumption of CSV and HDF5 inputs for *Dask* is not affected much by an increase in the dataset size when compared with *Pandas* and *Vaex*. The reason for this could be because of the lazy execution using computation graphs where it only loads information about the header, and the datatypes [17].

The energy consumption of the dataframe output operation in CSV, JSON, and HDF5 formats for the datasets *D1*, *D2* and *D3* are shown in Table IV. We observe that *Dask* is the most expensive library in terms of energy consumption across all formats and datasets. *Pandas* consumes the least amount of energy for outputs in JSON format. For outputs in CSV format, *Vaex* is the least energy-expensive library for datasets *D2* and *D3*, while *Pandas* is the least expensive for D1. For output in HDF5 format, Pandas consumes the lowest energy for the datasets *D1* and *D2* while *Vaex* consumes the least energy for *D3*.

In terms of data format, HDF5 is the least expensive for output operation when working with *Pandas* and *Vaex*. The energy consumption of all libraries across formats increases with the size of the output. An exception for this is the case of HDF5 output in *Dask*, where the energy consumed for *D2* is higher than that of *D3*. We suspect that the reason for this anomaly could be the presence of large text-based reviews in *D2*. The cumulative energy consumption of input and output operations of the libraries for all datasets is shown in Fig. 2.

The following are some of the key highlights from the observation of the results of the experiment on input/output operations for the datasets considered.

1) *Vaex* and *Dask* are the most expensive libraries for CSV and HDF5 inputs, respectively.
2) *Dask* is the most energy-efficient library for CSV inputs whose energy consumption is consistent across datasets.
3) HDF5 is the most energy-efficient format for input operations, while JSON is the most energy-intensive format.
4) *Dask* is the most energy-intensive library for output operations.
5) *Pandas* is the most energy-efficient library for JSON outputs.

*2) **Handling Missing Data**:* The energy consumption values of the missing data handling functions are shown in Table V. *Pandas* consumes the highest energy for *dropna*, *fillna* and *replace* operations across all datasets. Dask is the least

TABLE V
MEAN VALUES FOR ENERGY CONSUMPTION (IN JOULES) FOR MISSING DATA HANDLING OPERATIONS. FOR EACH LIBRARY, COLUMNS P AND D REPRESENT PKG AND DRAM ENERGY CONSUMPTION VALUES, RESPECTIVELY.

	Adult Dataset						**Drug Dataset**						**US Census Dataset**					
	Pandas		Vaex		Dask		Pandas		Vaex		Dask		Pandas		Vaex		Dask	
Operation	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D
isna	0.28	0.04	0.16	0.02	0.18	0.03	1.86	0.27	0.17	0.02	0.19	0.03	0.17	0.02	0.22	0.03	0.21	0.03
dropna	2.58	0.35	0.97	0.14	0.54	0.07	7.87	0.98	0.76	0.11	0.49	0.06	55.14	5.98	3.70	0.50	0.91	0.13
fillna	2.57	0.34	1.44	0.2	0.53	0.08	7.29	0.90	1.14	0.16	0.48	0.07	51.46	5.58	6.85	0.85	0.93	0.12
replace	0.29	0.02	0.16	0.02	0.17	0.03	1.09	0.17	0.11	0.02	0.17	0.03	0.46	0.08	0.14	0.02	0.18	0.03

TABLE VI
MEAN VALUES FOR ENERGY CONSUMPTION (IN JOULES) FOR ROW/COLUMN OPERATIONS. FOR EACH LIBRARY, COLUMNS P AND D REPRESENT PKG AND DRAM ENERGY CONSUMPTION VALUES, RESPECTIVELY.

	Adult Dataset						**Drug Dataset**						**US Census Dataset**					
	Pandas		Vaex		Dask		Pandas		Vaex		Dask		Pandas		Vaex		Dask	
Operation	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D
drop	0.31	0.05	0.17	0.03	0.48	0.07	0.89	0.15	0.17	0.03	0.48	0.06	50.24	5.43	0.34	0.05	0.83	0.12
groupby	0.09	0.15	4.00	0.64	0.11	0.01	0.10	0.01	3.35	0.46	0.14	0.01	0.10	0.01	3.66	0.52	0.15	0.02
concat	0.75	0.11	0.22	0.03	1.66	0.24	3.30	0.54	0.17	0.03	2.35	0.34	23.82	2.73	0.49	0.07	3.37	0.45
sort	0.65	0.09	0.46	0.05	0.42	0.06	2.76	0.43	0.55	0.07	0.40	0.06	89.83	9.86	2.68	0.41	345.2	33.8
merge	2.90	0.39	0.58	0.08	2.14	0.29	37.02	3.79	0.39	0.06	1.86	0.26	231.15	24.24	2.38	0.33	9.37	1.08

expensive library in terms of energy consumption for *dropna* and *fillna* operations. While *Vaex* consumes the lowest energy for *replace* operation, it must also be noted that the difference between the energy consumption of *Vaex* and *Dask* for replace operations ranges between a mere 0.3 - 0.8 Joules.

In the case of *Pandas*, the energy consumption increases with the increase in the size of the dataset for *dropna* and *fillna* operations, the *replace* and *isna* operations are not affected to the same degree by the size of the dataset. Across all libraries, we observe that *dropna* and *fillna* are the most energy-expensive operations. The cumulative energy consumption of missing data operations of the libraries for all the datasets is shown in Fig. 3.

The key highlights from the observation of the results of the experiments on missing data handling operations for the datasets considered are:

1) Overall, *Pandas* is the most energy-expensive library for missing data handling operations.
2) *Dask* is the most energy-efficient library for *fillna* and *dropna* operation.
3) *Vaex* is the most energy-efficient library for *replace* operation.

3) ***Row/Column Operations****:* The energy consumption values for the row/column operations are shown in Table VI. *Vaex* consumes the least energy for *drop*, *merge* and *concat* operations across all datasets. *Dask* has the highest energy consumption for *drop* operation on *D1* while *Pandas* consumes the highest on *D2* and *D3*. *Vaex* has the highest energy consumption for *groupby* operation while *Pandas* consumes the lowest across all datasets. For *concat* operation, *Dask* is the most expensive library on *D1* while *Pandas* is the most expensive library on *D2* and *D3*. *Pandas* is the most energy consuming library for *concat* and *merge* operations across datasets.

It is further observed that the variation in energy consumption based on the size of the dataset is the greatest in the case of *Pandas* for *drop*, *concat*, and *merge* operations. *Vaex* has the lowest increase in energy consumption of all operations except *groupby* with the increase in dataset size. For *groupby* operation, *Vaex* exhibits the highest increase in energy consumption based on dataset size.

We also observe a drastic increase in energy consumption for *sort* operation using *Dask* for *D3*. A possible reason for this could be the presence of a higher number of unique values in the sorted column in the case of *D3* along with a substantially larger number of data points.

The following are the key highlights from the results on row/column operations for the datasets considered in the study.

1) *Vaex* is the most energy-efficient library for *drop*, *concat*, and *merge* operations.
2) *Vaex* is the most energy-expensive framework for groupby operation.
3) Overall, *Pandas* exhibits the highest influence of the dataset size on the increase in energy consumption.

4) ***Statistical Aggregation Operations****:* The energy consumption values for the statistical aggregation operations are shown in Table VII. From the table, we observe that *Dask* is the most energy-consuming library for statistical operations across all datasets. *Pandas* exhibits the lowest energy consumption for all operations other than *count* across datasets. *Vaex* is the most energy-efficient library for *count* operation.

The increase in energy consumption with the increase in dataset size is the highest in the case of *Dask*. For all

TABLE VII
MEAN VALUES FOR ENERGY CONSUMPTION (IN JOULES) FOR STATISTICAL AGGREGATION OPERATIONS. FOR EACH LIBRARY, COLUMNS P AND D REPRESENT PKG AND DRAM ENERGY CONSUMPTION VALUES, RESPECTIVELY.

	Adult Dataset						**Drug Dataset**						**US Census Dataset**					
	Pandas		Vaex		Dask		Pandas		Vaex		Dask		Pandas		Vaex		Dask	
Operation	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D	P	D
count	2.54	0.34	0.94	0.13	5.76	0.75	6.99	0.86	1.66	0.24	104.82	10.09	10.19	1.38	2.68	0.38	256.63	24.3
sum	0.09	0.01	1.14	0.16	2.45	0.37	0.11	0.01	2.39	0.33	46.43	4.66	0.23	0.03	3.08	0.44	92.25	9.16
mean	0.10	0.01	1.35	0.19	2.98	0.45	0.11	0.01	2.80	0.39	46.75	4.70	0.37	0.06	3.42	0.49	93.68	9.43
min	0.07	0.01	1.30	0.18	2.47	0.36	0.09	0.01	2.48	0.35	45.75	4.60	0.21	0.03	3.09	0.43	92.11	9.23
max	0.06	0.01	1.27	0.18	2.43	0.36	0.15	0.01	2.15	0.30	45.77	4.60	0.21	0.03	3.01	0.43	92.54	9.27
unique	0.10	0.01	1.66	0.24	2.34	0.34	1.26	0.18	7.39	1.04	47.53	4.77	1.34	0.19	3.39	0.46	91.73	9.23

TABLE VIII
INSTANCES WHERE THE FASTEST PERFORMING LIBRARY WAS NOT THE MOST EFFICIENT ONE. E REPRESENTS THE TOTAL ENERGY CONSUMED (IN JOULES), AND T REPRESENTS THE TIME DURATION OF THE OPERATION (IN MS).

Dataset	**Operation**	**Pandas**		**Vaex**		**Dask**	
		E	T	E	T	E	T
Adult (D1)	groupby	0.10	1.85	4.64	6.42	0.12	1.77
Adult (D1)	sort	0.74	11.58	0.51	6.07	0.49	8.43
Drug Review (D2)	groupby	0.12	2.09	3.81	6.01	0.16	1.99

operations except *count*, the increase in energy consumption with respect to the size of the dataset is the least for *Pandas*. In the case of *count* operation, *Vaex* has the least amount of increase in energy consumption with an increase in the dataset size. It can also be seen that *count* is the most energy-consuming statistical operation for *Pandas* and *Dask*. At the same time, *unique* is the most energy-consuming operation for *Vaex*. The cumulative energy consumption of statistical aggregation operations of the libraries for all datasets is shown in Fig. 4.

The following are the key highlights from the results on statistical aggregation operations for three datasets.

1) *Dask* is the most expensive library for statistical aggregation operations.
2) *Pandas* is the most energy-efficient library for all operations considered other than *count*.
3) *Vaex* is the most energy-efficient library for *count* operation.

B. Is the most energy-efficient dataframe processing library for a given task also the fastest?

This section examines the correlation between the processing speed and energy consumption of the libraries for the tasks evaluated in our study. We computed the average time duration (in seconds) needed to execute each task for every dataset over ten trials, utilizing the values obtained through the use of the *Pyjoules* library. These values are available in the replication package.

In this study, we performed 21 operations on three datasets, making 63 comparisons between the libraries. Out of these 63 comparisons, the library with the fastest processing was also the most energy-efficient in 60 instances. The three exceptions where the most energy-efficient library was not the fastest one are listed in Table VIII. The results indicate that the library with the fastest processing is also the most energy-efficient for input-output operations, missing data handling operations, and statistical operations. However, for row/column operations, *Dask* was the fastest performing library for *D1* and *D2*, while *Pandas* was the most energy-efficient. Additionally, for the sort operation on *D1*, *Vaex* was the fastest library, whereas *Dask* was the most energy-efficient.

Based on the results of our experiment, the most energy-efficient dataframe processing library is also the fastest for a given task barring a few exceptions.

VII. THREATS TO VALIDITY

In this section, we discuss potential systematic errors in our work that may pose threats to the outcome of our study.

A. Internal Validity

One of the main limitations of our study is the potential impact of noise, voltage spikes, daemons, and other background processes on the accuracy of energy measurement. To mitigate this issue, we repeated the same task 10 times and calculated the average of the recorded values. We also ensured that the system was idle for 30 seconds before the start of the next task to minimize the impact of tail states. While RAPL is one of the most reliable tools available for estimating software energy consumption, it only provides overall energy consumption and does not offer detailed measurements at the process level. Hence we took steps to minimize the impact of background processes on our measurements by shutting down non-essential services and daemons, thus reducing the overhead added by the operating system during execution. It is also worth noting that the results may vary depending on the machine and system configuration. The libraries may have different performances on different hardware.

B. External Validity

It should be acknowledged that the generalizability of the results is limited as the experiments were conducted on only three datasets. However, we have taken care to select datasets that vary in size, type of data, and the number of attributes in order to minimize this limitation. It is important to note that the results we obtained were based on the time when the study was conducted. Open-source libraries are often developed by a large number of contributors across the globe. As libraries compete with each other, the implementations become more efficient over time. Therefore, the results of similar experiments conducted in the future may differ due to updates and modifications to the libraries. To account for this, we have provided the versions of the libraries used in our study in Section V. This allows for replication of our results and enable comparison with future studies.

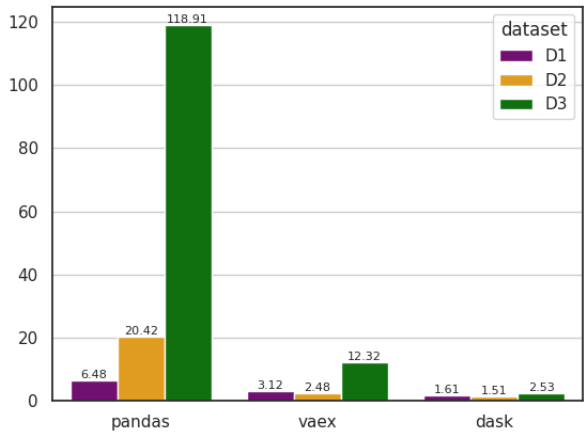

Fig. 3. Cumulative energy consumption (in Joules) of missing data operations of the libraries for both the datasets

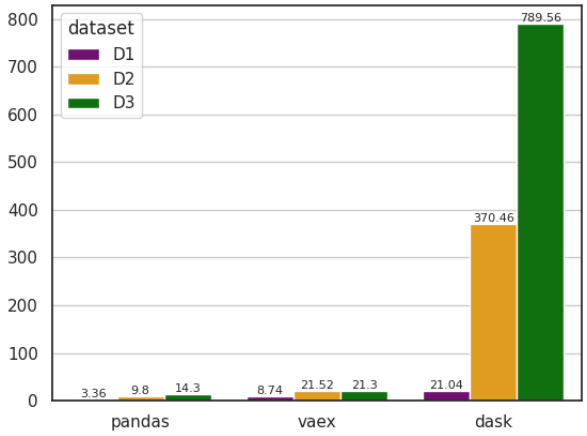

Fig. 4. Cumulative energy consumption (in Joules) of statistical aggregation operations of the libraries for both datasets.

VIII. Discussion

This paper presents an initial examination into the energy efficiency of the data-oriented stages of the machine learning pipeline through a comparative analysis on dataframe processing libraries. Through our research, we aim to draw attention to the importance of energy efficiency in data-oriented stages and encourage further exploration in this area. Our findings suggest that the choice of libraries for a given task could potentially influence the energy consumption of the dataframe processing. We also found that the file format used for storing the data influences the energy consumption during input and output.

The results of our findings can have the following implications for practitioners and researchers.

- ➢ No dataframe processing library is the most energy-efficient for all tasks. The decision of which library to use should be made based on the specific requirements and frequently performed tasks of the user. As an example, *Dask* can be a poor choice in terms of energy when most frequent operations are based on statistical aggregation.
- ➢ The outcomes of this study also highlight the need for further improvements in energy efficiency for certain tasks by the developers of these libraries.
- ➢ The results suggest the need for further research to explore the root causes and factors that contribute to the energy inefficiencies of libraries for specific tasks, as well as potential solutions to mitigate these inefficiencies.

The study by Verdecchia et al. [27] has revealed the potential for reduction in energy consumption of training through data modifications such as the dataset size and the number of features. However, data pre-processing remains an essential step in the pipeline despite these modifications. Our results suggest the possiblity of additional reductions in energy consumption within the data-oriented stages of the pipeline through the selection an appropriate library.

It is important to acknowledge that this study has evaluated the energy consumption of various libraries for a specific set of tasks, yet alternative methods or functions within a library may also be capable of achieving the same results. For instance, many missing data handling operations could potentially be completed using the *apply* function in *Pandas*. The energy consumption of these alternate methods may vary and they may be more energy-efficient or energy-intensive. However, this study does not investigate such alternatives.

The experiments conducted did not incorporate GPU acceleration as the comparison of energy consumption across various libraries for a given task requires a standardized hardware setup. While *Vaex* and *Dask* do have GPU support, the absence of native GPU support in *Pandas* precluded the utilization of GPUs in the experiments.

Additionally, we have observed that in some cases the energy consumption values for a task on a larger dataset are lower than the values for a smaller dataset. To give a few examples, the energy consumption of Dask for *output* in HDF5 format is greater on *D2* compared to *D3*. In case of missing data operations, the energy consumption on *D1* is greater than

that on *D2* for *Vaex* and *Dask* in tasks that involve NaN values. In the case of statistical operations, we observe that energy consumption value of *unique* operation for the *Vaex* library in the case of *D2* is higher than that of *D3*. In the case of row/column operations, we observe similar inconsistency in the case of *merge* using *Vaex* and *Dask* libraries on *D1* and *D2*, and the *concat* operation using *Vaex* on *D1* and *D2*. Although these results may look counter-intuitive, we conjecture that it could be influenced by factors such as the number of attributes, the type of data being processed, or the internal workings of these libraries for the given task. Further investigation of these factors could provide more insights on such behaviors.

IX. Future Directions

In this section, we propose a few potential future directions based on the findings of our study.

Our investigation evaluates the energy consumption of diverse dataframe processing libraries for 21 tasks distributed across four categories. The findings have the potential to assist developers in selecting the most efficient library for a specific task. However, the frequency of performing each task in a data pipeline is not consistent. It is important to take frequency into account because determining the most appropriate library for a particular case depends on the type of operations that occur most frequently. Since we lacked adequate information in the literature to allocate weights to the tasks, our study did not emphasize assigning weights to operations based on their frequencies in a typical pipeline. A subsequent study that focuses on acquiring data from real-world projects to determine task weights would increase the practical significance of our results.

Our experiments have revealed instances where the energy consumption on a smaller dataset was higher than that on a larger dataset for the same task performed using the same library as discussed in Section VIII. Although the results obtained from our study do not provide a clear explanation for these anomalies, various factors, such as the number of attributes, the type of values, and the frequency of certain values (e.g., NaN), may impact energy consumption. Investigating the influence of these features on energy consumption during data pre-processing could offer valuable insights and represents an interesting avenue for future research.

The present study focuses on comparing the energy consumption of various dataframe processing libraries. However, a limitation of our study is that the results may become outdated as future versions of these libraries may produce different outcomes due to updates. The libraries used in the study utilize different techniques for data processing. For instance, *Vaex* uses zero memory copy policy and *Dask* uses of computation graphs for parallelization of tasks [17]. Thus, a more in-depth future study that compares the energy consumption of these techniques employed by the libraries internally could potentially provide more enduring insights.

X. Related Work

Energy consumption is a crucial concern in software, as it has not only an environmental impact but also financial implications for organizations. In recent years, the energy consumption of software has caught the attention of researchers. Given the availability of multiple alternatives to perform the same task, comparative energy analysis with the intention of helping developers make energy-efficient choices has been a popular theme among researchers.

In the field of Java programming, there have been several studies that have explored the comparison of different libraries, APIs, and platforms [31], [32], [35], [36]. Rouvoy et al. [35] performed an empirical investigation of differences in energy consumption of read/write methods of some of the famous Java libraries. They found some methods to be more efficient than others. Hasan et al. [36] studied the energy profiles of various Java collection classes and found that the choice of an inefficient collection can cost as much as 300% more energy. Ournani et al. [31] compared the energy consumption of 27 Java I/O methods for different file sizes and found that the energy consumption varies across APIs, with some of them consuming about 30% less energy than the others. Kumar et al. [32] analyzed the energy consumption of Java command line options and found Oracle JDK to be more energy-efficient than Open JDK. They also found that UseG1GC and Xint were the most and the least energy-efficient command line options, respectively [32].

Given the large number of programming languages, a comparative study of their energy consumption has also been explored. Pereira et al. [30] conducted an investigation into the energy efficiency of 27 different programming languages, comparing their energy consumption in relation to speed and memory utilization. This study provided insight into the energy consumption of various programming languages and the potential impact of language choice on energy efficiency. Other areas of study that have explored the energy consumption of multiple alternatives include the examination of energy consumption in design patterns. Maleki et al. [37] found that the use of certain patterns, such as the decorator and overloading patterns, can negatively impact the energy efficiency of software. Similarly, in the field of mobile app development, Chowdhury et al. [38] conducted a comparative study that analyzed the energy consumption of two versions of the HTTP protocol and found that HTTP/2 was more energy-efficient than HTTP/1.1 in most scenarios. Cruz et al. [39] analyzed the energy consumption of eight UI automation frameworks and found that certain frameworks can increase energy consumption by over 2000 percent.

Energy efficiency is of crucial importance in server systems and cloud applications, as it has significant economic and ecological implications. Researchers have sought to guide developers in making energy-efficient choices among the various options available. As such, energy efficiency has been directed towards this domain [40], [41]. Singh et al. [40] conducted an investigation of the energy costs associated with running

various Java APIs on servers to accomplish a set of tasks and found that developers can reduce energy costs by selecting energy-efficient APIs. Khomh et al. [41] explored the energy consumption of 6 cloud patterns and found that they can be energy-efficient in certain cases.

In the domain of machine learning, several researchers have been working towards improving energy efficiency. These model-centric approaches include model compression techniques like quantization [23], [42], pruning [43], distillation [24], use of efficient hardware accelerators [44]–[46], and use of efficient computation methods in the hardware [25], [26]. The software engineering approach includes the catalog of energy-efficient practices to be followed during the development of deep learning applications [47].

Comparative studies on energy consumption aimed at helping practitioners pick the most energy efficient option have also been performed in the area of machine learning. Georgiou et al. [33] performed a comparative study on two popular deep learning frameworks, namely Pytorch and Tensorflow. They compared the energy consumption of the frameworks during the training and inferencing process for comparable levels of accuracy. They found that Tensorflow achieves significantly better energy efficiency in training phase where as Pytorch performs better during inferencing phase. McIntosh et al. [48] conducted an empirical study to investigate the energy efficiency of various algorithms used for training machine learning models on Android devices. The study results indicate that j48, smo, and mlp algorithms are more energy-efficient, accurate, and correlate with the complexity of the algorithms. Furthermore, the study identifies several factors that can significantly impact the energy consumption of machine learning on Android devices, including the size of the dataset and the number of fields. Verdecchia et al. [27] compared the energy consumption of different machine learning models for the same task of detecting spam messages and found that choosing certain models over others can lead to reduction of energy consumption by over 94%.

Nonetheless, the comparative energy consumption of various libraries used to accomplish pre-processing tasks remains unexplored. This study represents an initial step towards addressing this gap.

XI. Conclusion

In this study, we have conducted an exploratory analysis of the energy consumption of dataframe processing libraries. The focus of this work is to initiate the examination of energy efficiency in the data-oriented stages of the machine learning pipeline, which encompasses tasks such as data preprocessing, cleaning, and exploratory data analysis.

Our study presents an exploratory analysis of the energy consumption of dataframe processing libraries in the context of data-oriented stages of the machine learning pipeline. We specifically focused on three popular libraries, *Pandas*, *Vaex*, and *Dask*, and evaluated their energy efficiency in performing various dataframe processing tasks, including input-output operations, missing data handling, row/column operations, and statistical aggregation operations. The energy consumption was measured using the Intel RAPL interface through the *Pyjoules* library. We used three datasets from the UCI repository, namely the *Adult* Dataset, *Drug Review* dataset, and *US Census (1990)* dataset, to run the tasks in our experiment. The results show that the choice of library and the format of the data storage can influence the energy consumption of the data pre-processing and data cleaning stages. To optimize the energy consumption of dataframe processing, the selection of the library should consider several factors, including the frequency of operations, data format, and dataset size.

In addition to comparing the energy consumption of various dataframe processing libraries, this study aims to draw the attention of the research community towards the importance of energy efficiency in the stages of the machine learning pipeline other than model training and inferencing. A more thorough investigation of the energy consumption of the data-oriented stages, including data pre-processing, data cleaning, and exploratory analysis, could potentially lead to the identification of energy-efficient practices in these areas.

Although this study provides only an initial exploration of comparative energy consumption of dataframe libraries, we plan to extend it in several ways. Specifically, we intend to conduct experiments on multiple systems with different configurations to obtain more generalized results, as well as repeat the experiments on larger datasets to explore trends with respect to the energy consumption values for different tasks. Further research in this area is necessary, and we have outlined some of potential future directions based on the results obtained in our study.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[3] R. Cai, B. Xu, X. Yang, Z. Zhang, Z. Li, and Z. Liang, "An encoder-decoder framework translating natural language to database queries," *arXiv preprint arXiv:1711.06061*, 2017.

[4] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[5] S.-R. Lee, M.-J. Heo, C.-G. Lee, M. Kim, and G. Jeong, "Applying deep learning based automatic bug triager to industrial projects," in *Proceedings of the 2017 11th Joint Meeting on foundations of software engineering*, 2017, pp. 926–931.

[6] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[9] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[11] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 291–300.

[12] Y. Wu, "Is a dataframe just a table?" in *10th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[13] S. Gathani, P. Lim, and L. Battle, "Debugging database queries: A survey of tools, techniques, and users," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–16.

[14] D. Petersohn, "Dataframe systems: Theory, architecture, and implementation," Ph.D. dissertation, Ph. D. Dissertation. EECS Department, University of California, Berkeley . . . , 2021.

[15] F. Perez and B. E. Granger, "Project jupyter: Computational narratives as the engine of collaborative data science," *Retrieved September*, vol. 11, no. 207, p. 108, 2015.

[16] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[17] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proceedings of the 14th python in science conference*, vol. 130. Citeseer, 2015, p. 136.

[18] M. A. Breddels and J. Veljanoski, "Vaex: big data exploration in the era of gaia," *Astronomy & Astrophysics*, vol. 618, p. A13, 2018.

[19] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 12–21.

[20] L. Cruz and R. Abreu, "Catalog of energy patterns for mobile applications," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2209–2235, 2019.

[21] G. Pinto and F. Castor, "Energy efficiency: a new concern for application software developers," *Communications of the ACM*, vol. 60, no. 12, pp. 68–75, 2017.

[22] O. Edenhofer, *Climate change 2014: mitigation of climate change*. Cambridge University Press, 2015, vol. 3.

[23] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5900–5904.

[24] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[25] X. Jiao, V. Akhlaghi, Y. Jiang, and R. K. Gupta, "Energy-efficient neural networks using approximate computation reuse," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1223–1228.

[26] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 2, pp. 1–23, 2018.

[27] R. Verdecchia, L. Cruz, J. Sallou, M. Lin, J. Wickenden, and E. Hotellier, "Data-centric green ai an exploratory empirical study," in *2022 International Conference on ICT for Sustainability (ICT4S)*. IEEE, 2022, pp. 35–45.

[28] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: Memory power estimation and capping," in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. IEEE, 2010, pp. 189–194.

[29] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[30] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, "Energy efficiency across programming languages: how do energy, time, and memory relate?" in *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, 2017, pp. 256–267.

[31] Z. Ournani, R. Rouvoy, P. Rust, and J. Penhoat, "Comparing the energy consumption of java i/o libraries and methods," in *37th International Conference on Software Maintenance and Evolution (ICSME)*, 2021.

[32] M. Kumar and W. Shi, "Energy consumption analysis of java command-line options," in *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2019, pp. 1–8.

[33] S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, and Y. Zou, "Green ai: Do deep learning frameworks have different costs?" ACM: Association for Computing Machinery, 2022.

[34] J. Bornholt, T. Mytkowicz, and K. S. McKinley, "The model is not enough: Understanding energy consumption in mobile devices," in *2012 IEEE Hot Chips 24 Symposium (HCS)*. IEEE, 2012, pp. 1–3.

[35] R. Rouvoy, P. Rust, and J. Penhoat, "Comparing the energy consumption of java i/o libraries and methods," in *ICSME 2021-37th International Conference on Software Maintenance and Evolution*, 2021.

[36] S. Hasan, Z. King, M. Hafiz, M. Sayagh, B. Adams, and A. Hindle, "Energy profiles of java collections classes," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 225–236.

[37] S. Maleki, C. Fu, A. Banotra, and Z. Zong, "Understanding the impact of object oriented programming and design patterns on energy efficiency," in *2017 Eighth International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2017, pp. 1–6.

[38] S. A. Chowdhury, V. Sapra, and A. Hindle, "Client-side energy efficiency of http/2 for web and mobile app developers," in *2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 529–540.

[39] L. Cruz and R. Abreu, "On the energy footprint of mobile testing frameworks," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2260–2271, 2019.

[40] J. Singh, K. Naik, and V. Mahinthan, "Impact of developer choices on energy consumption of software on servers," *Procedia Computer Science*, vol. 62, pp. 385–394, 2015.

[41] F. Khomh and S. A. Abtahizadeh, "Understanding the impact of cloud patterns on performance and energy consumption," *Journal of Systems and Software*, vol. 141, pp. 151–170, 2018.

[42] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst, "Energy-efficient convnets through approximate computing," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–8.

[43] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5687–5695.

[44] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[45] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 688–698.

[46] J. H. Ko, B. Mudassar, T. Na, and S. Mukhopadhyay, "Design of an energy-efficient accelerator for training of convolutional neural networks using frequency-domain computation," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[47] S. Shanbhag, S. Chimalakonda, V. S. Sharma, and V. Kaulgud, "Towards a catalog of energy patterns in deep learning development," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, 2022, pp. 150–159.

[48] A. McIntosh, S. Hassan, and A. Hindle, "What can android mobile app developers do about the energy consumption of machine learning?" *Empirical Software Engineering*, vol. 24, no. 2, pp. 562–601, 2019.