UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Kadri-Ketter Kont

Benchmarking Energy and Performance of Standard Machine Learning Libraries: An Empirical Study

Bachelor's Thesis (9 ECTS)

Supervisor:
Hina Anwar, PhD

Tartu 2025

Benchmarking Energy and Performance of Standard Machine Learning Libraries: An Empirical Study

Abstract:

The impact of programming language and library selection on the energy consumption of machine learning tasks has not been thoroughly explored. This thesis compared three widely used programming languages in machine learning and their libraries, focusing on energy efficiency, runtime performance, and model accuracy. Using a classification dataset, five machine learning algorithms were implemented using standard libraries in each language. The study examined how the choice of language and library impacts energy consumption, execution time, and predictive performance, exploring trade-offs between these metrics. Statistical analysis was conducted to validate the results and compare the algorithms' energy usage and performance across different implementations. The results provided insights into how the choice of standard libraries impacts the energy efficiency of machine learning algorithms.

Keywords: Energy efficiency, programming languages, machine learning, ML libraries, green software, green AI

CERCS: P174 Informatics, systems theory, P176 Artificial intelligence

Standardsete masinõppe teekide energiatõhususe ja sooritusvõime võrdlus

Lühikokkuvõte:

Programmeerimiskeele ja teegi valiku mõju masinõppeülesannete energiakulule ei ole seni põhjalikult uuritud. Lõputöö eesmärk oli võrrelda kolme masinõppes levinud programmeerimiskeelt ja nende teeke, keskendudes energiatõhususele, käitusajale ja mudeli täpsusele. Klassifitseerimisülesande andmestiku põhjal rakendati standardteekide abil igas keeles viis masinõppealgoritmi. Uurimuses analüüsiti, kuidas keele ja teegi valik mõjutab energiatarbimist, käitusaega ja täpsust, ning uuriti nende näitajate vahelisi kompromisse. Tulemuste kinnitamiseks viidi läbi statistiline analüüs, mille abil võrreldi algoritmide energiakasutust ja jõudlust erinevate implementatsioonide puhul. Töö tulemusena tehti ülevaade, kuidas standardteekide valik mõjutab masinõppe algoritmide energiatõhusust.

Võtmesõnad: Energiatõhusus, programmeerimiskeeled, masinõpe, ML teegid, roheline tarkvara, roheline AI

CERCS: P175 Informaatika, süsteemiteooria, P176 Tehisintellekt

Table of Contents

1. Introduction

Technology is deeply embedded in daily life from when individuals wake up to when they go to bed. Unlocking a phone uses facial recognition powered by machine learning (ML) for secure access [1]. While driving, Google Maps analyses real-time data with ML to predict traffic patterns and optimise routes [2]. At the bank, advanced ML algorithms detect potential fraud during transactions [3]. Later, Netflix uses deep learning to recommend personalised content based on preferences and viewing history [4]. Artificial intelligence (AI) and ML quietly power much of modern life, enhancing everyday convenience.

Optimising AI and ML for lower energy consumption has become essential as global energy efficiency becomes a priority. With the rise of cloud computing, the growth of AI and ML has led to significant environmental impacts. In fact, between 2020 and 2022, emissions from cloud and content data servers grew by 46%, and electricity consumption rose by 63% [5]. With AI models and ML applications now being deployed on a massive scale, their energy consumption is an urgent concern.

While numerous studies have explored the trade-offs between energy consumption and factors such as algorithm efficiency, model complexity, dataset configurations, frameworks, and batch size optimisation [6], [7], [8], [9], [10], there has been little investigation into whether and how the choice of programming language and libraries influences energy consumption in ML tasks. Furthermore, how do these choices impact other key performance metrics, such as execution time and model accuracy?

This research aims to fill that gap by analysing the impact of library selection on the energy consumption of machine learning tasks. Specifically, the analysis will compare implementations of five different algorithms across three programming languages, each using standard libraries. These programs will be compiled and executed with state-of-the-art compilers, interpreters, and libraries for each language. The performance of these implementations will be evaluated based on three key metrics: execution time, model accuracy, and energy consumption. The findings will provide insights into how different libraries trade off regarding energy usage, execution speed, and model accuracy, offering a deeper understanding of the interplay between these critical factors.

This thesis was written with the assistance of OpenAI's ChatGPT [11] and Grammarly AI [12] for language refinement.

2. Related Work

The study of energy efficiency in programming languages within machine learning contexts is a relatively underexplored area. While energy profiling and comparisons of programming languages have been widely researched, most studies focus on general-purpose tasks rather than the specific demands of ML algorithms. Conversely, research on energy efficiency in ML typically emphasises how parameters such as model architecture, hyperparameters, or hardware choices affect consumption, with less attention given to the role of the programming language. To bridge this gap, it is essential to explore two distinct areas of related work:

- Studies investigating the general energy efficiency of programming languages.
- Research examining energy consumption within machine learning and artificial intelligence contexts.

By addressing these separately, a more precise understanding can be gained of how these fields intersect and highlight the unique contribution of evaluating programming languages in ML-specific energy profiling.

2.1 Energy Consumption of Programming Languages

Numerous studies have been carried out to understand the energy efficiency of programming languages, progressively building upon one another to broaden knowledge in this area.

Couto et al. [13] described a study on the energy efficiency of ten different programming languages using a set of computing problems from the Computer Languages Benchmarks Game (CLBG) project[1]. The CLBG project provides a collection of programs designed to compare programming languages by performance regarding runtime and memory consumption. Couto et al. analysed the energy consumption and execution time of the programs from the CLBG repository. Energy measurements were obtained using Intel's Running Average Power Limit (RAPL) tool[2]. The results showed that the compiled languages were more energy efficient and faster than the interpreted ones. The C programming language came up as the quickest and most energy-efficient. The study also revealed that energy consumption is not always correlated with execution time, as some slower languages consumed less energy.

[1] https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

[2] https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html

Building on this work, Pereira et al. [14] expanded the scope by analysing 27 programming languages sourced from the CLBG repository. As before, Intel's RAPL tool was used to measure energy consumption. Memory usage data was also collected using the time command available in Unix-based systems. Unlike earlier studies focused solely on CPU energy consumption and execution time, this research incorporated peak memory consumption, allowing for the exploration of the relationship between memory usage and energy efficiency. The study further confirmed that a faster language is not always the most energy-efficient. Additionally, Pereira et al. introduced a ranking system that assessed languages based on various combinations of metrics—Energy vs Time, Energy vs Peak Memory, Time vs Peak Memory, and Energy vs Time vs Peak Memory—providing new insights into language performance across multiple dimensions and highlighting the trade-offs involved in selecting a language for energy efficiency.

Pereira et al. [15] extended this research further. The methodology and tools, including the CLBG repository and Intel's RAPL tool, remained consistent with prior studies, but the authors introduced two new extensions. First, instead of focusing solely on peak memory usage, they analysed total memory usage, representing the cumulative amount consumed throughout the application's lifecycle. Second, they presented another extensive study using the Rosetta Code repository to validate earlier rankings. Unlike CLBG, which is performance-focused, Rosetta Code[3] emphasises clarity and pedagogy by providing alternative solutions to programming problems designed to assist programmers in understanding the syntactic and semantic aspects of languages outside their expertise. The comparison between the CLBG and Rosetta Code results revealed many similarities. However, differences in ranking were analysed, and Pereira et al. could quickly localise and explain such occurrences based on improper or inefficient use of the programming language. This dual-repository approach validated prior findings and highlighted how different benchmarks influence programming language performance rankings.

Most recently, this line of research was extended by Gordillo et al. [16], who explored whether using hardware-based measurements compared to software-based estimation approaches influences the ranking of programming languages based on energy consumption. Using a hardware-based measurement approach, the researchers could monitor the energy consumption of specific Device Under Test (DUT) hardware components, such as the HDD, graphics card, and processor. The study concluded that the choice of hardware-based or software-based

[3] https://rosettacode.org/wiki/Rosetta_Code

measurement approaches did not significantly alter the rankings of programming languages, suggesting that both methods are reliable for this type of analysis. Furthermore, Gordillo et al. demonstrated the value of analysing the energy consumption of specific hardware components. They created separate rankings for the HDD, graphics card, and processor, offering new perspectives on how programming languages interact with different hardware elements. This study provided actionable insights for researchers and practitioners, emphasising the importance of hardware-aware energy profiling in making informed decisions about programming language selection.

Previous studies primarily relied on benchmark repositories like CLBG and Rosetta Code, which feature generic algorithms. It has also been observed that differences between performance-oriented and idiomatic, human-written code are minimal, suggesting that evaluations using realistic, less optimised code are valid and reflective of real-world programming practices [15]. Additionally, findings show that hardware-based and software-based energy measurement approaches produce comparable results, making software-based estimates suitable for ranking programming languages [16]. The previous studies establish a transparent methodology for comparing and benchmarking the energy efficiency of programming languages. However, these benchmarks do not fully represent the computational demands of machine learning tasks. This study addresses this limitation by focusing on ML-specific algorithms to better align with the requirements of AI workloads.

2.2 Energy Consumption in ML and AI

Energy efficiency in AI and ML has become an important research area, driven by the goal of developing "Green AI" systems that reduce environmental impact without compromising performance. Various studies have explored the trade-offs between energy consumption and different aspects of AI and ML systems, including algorithm efficiency, model complexity, dataset configurations, frameworks, and batch size optimisation [6], [7], [8], [9], [10].

Many studies focus on comparing the energy efficiency of various ML algorithms and neural network architectures. For example, Yarally et al. [7] explored the energy consumption of neural networks with different structures, such as DenseLinearNN (gradually reducing neurons), DensePolyNN (halving neurons per layer), and SimpleCNN (a mix of convolutional and linear layers).

Furthermore, Yarally et al. [6] later investigated the balance between energy consumption and response time during inference for computer vision tasks. Their study compared five state-of-

the-art neural networks, examining metrics such as energy usage per image and response time under varying batch sizes. The research aimed to identify how batching impacts energy efficiency while maintaining acceptable response times, offering a practical perspective on real-time AI systems.

The other two studies closely align with the objectives of this research but focus on different aspects of energy efficiency in machine learning. Verdecchia et al. [8] explore how algorithms, dataset sizes, and feature counts influence the energy consumption of AI systems. They analyse trade-offs between energy efficiency and algorithm performance, examining whether optimising datasets can reduce energy usage without compromising accuracy. Similarly, Santos et al. [10] compare six machine learning models' energy efficiency and performance, including traditional algorithms and neural networks. They assess accuracy and energy consumption during data preparation, training, and testing, using statistical methods to highlight models that achieve the best balance between energy efficiency and accuracy.

While all these studies provide valuable insights into energy-efficient AI and ML, they primarily focus on factors such as algorithm design, neural network architecture, dataset optimisation, and batch processing. These approaches have significantly contributed to understanding how to balance energy consumption with performance. However, they overlook another important factor: the role of programming language and library selection in determining energy efficiency. This study aims to bridge this gap by investigating how programming language and library choices affect the energy efficiency of machine learning algorithms, offering a fresh perspective on achieving greener AI systems.

3. Methodology

This study aims to evaluate and compare the performance of machine learning algorithms implemented using standard libraries across multiple programming languages. Specifically, the focus will be on three key metrics: energy consumption, runtime performance, and accuracy. By analysing these metrics, the aim is to gain insights into potential trade-offs and ascertain whether the choice of programming language influences the efficiency and effectiveness of machine learning models when using popular libraries. To achieve this, the following research questions will be addressed:

RQ1: Do machine learning algorithms, when implemented using standard libraries, exhibit differences in energy consumption, runtime performance, and accuracy across programming languages?

RQ2: What trade-offs exist between energy consumption, runtime performance, and accuracy when implementing machine learning algorithms using standard libraries across different programming languages?

To provide clarity and structure in our investigation, hypotheses have been formulated. The need for these hypotheses stems from a desire to establish a basis for comparison and to empirically test our assumptions about the performance characteristics of machine learning algorithms across different languages. For **RQ1**, the hypotheses are defined as follows:

H0 (Null Hypothesis): There is no significant difference in energy consumption, runtime performance, and accuracy across programming languages when machine learning algorithms are implemented using standard libraries.

H1 (Alternative Hypothesis): There is a significant difference in energy consumption, runtime performance, and accuracy across programming languages when machine learning algorithms are implemented using standard libraries.

Programming languages differ in key areas such as memory management, execution models, and I/O handling. These factors can influence the efficiency and performance of machine learning algorithms. For instance, compiled languages may exhibit faster runtime, while interpreted languages might be more flexible but less efficient. As such, it is reasonable to expect differences in energy consumption, runtime, and accuracy across languages.

For **RQ2**, the hypotheses are as follows:

H0 (Null Hypothesis): There are no trade-offs between energy consumption, runtime performance, and accuracy when implementing machine learning algorithms using standard libraries across programming languages.

H1 (Alternative Hypothesis): There are trade-offs between energy consumption, runtime performance, and accuracy when implementing machine learning algorithms using standard libraries across programming languages.

The relationship among these performance metrics indicates that optimising one aspect, such as runtime, could lead to compromises in another, like energy consumption or accuracy. Different programming languages and libraries may prioritise these trade-offs differently based on their inherent design and execution models, suggesting that such trade-offs are likely to exist.

In this context, this chapter presents the methodology employed in this study. It details the selection of programming languages and libraries, the chosen algorithms and their implementations, the dataset description, experimental setup, and measurement processes for energy consumption, runtime, and accuracy. Additionally, data preprocessing and statistical analysis techniques will be covered. Collectively, these sections ensure a systematic approach to addressing the research questions and rigorously testing the hypotheses.

3.1 Programming Languages and Libraries

This study analyses the energy efficiency, runtime performance and accuracy of machine learning tasks implemented in three widely used programming languages: Python, Julia, and C++. These languages were chosen due to their strong significance in the machine learning field [17], [18], [19].

To ensure a fair comparison, machine learning classification algorithms will be implemented in each language using relevant libraries and frameworks, allowing for a comprehensive evaluation of their energy consumption. Specifically, scikit-learn [20] and pandas [21] are used for Python**,** MLJ.jl [22] for Julia and mlpack [23] for C++ to implement the algorithms and perform necessary data preprocessing.

Table 1. Machine learning libraries and frameworks overview

Library/ Framework	Programming Language	Algorithm Coverage	Reason for Selection
Scikit-learn	Python	Logistic Regression, Naïve Bayes, SVM, Decision Tree, Random Forest	A widely-used library in Python, providing a broad range of algorithms and tools for classification tasks.
MLJ.jl	Julia	Logistic Regression, Naïve Bayes, Decision Tree, Random Forest	A flexible machine learning framework for Julia, known for its ease of integration and wide support for various models.
mlpack	C++	Logistic Regression, Naïve Bayes, SVM, Decision Tree, Random Forest	A simple and modular C++ library, offering easy-to-use implementations of common machine learning algorithms.

The libraries and frameworks selected for this study are summarised in Table 1, which outlines the relevant tools and the classification algorithms covered. These libraries were chosen for their relevance to their respective programming languages and because they provide all the required machine learning algorithms for the study.

3.2 Selected Machine Learning Algorithms and Implementations

The chosen algorithms represent a variety of classification techniques, including logistic regression, naïve Bayes, support vector machine (SVM), decision trees, and random forests. These algorithms were selected for their suitability in classification tasks and availability in the chosen libraries and frameworks, as shown in Table 1, which details the algorithm coverage for each library.

Given the absence of a dedicated repository for benchmarking machine learning tasks across different programming languages, the algorithms were manually implemented from scratch based on insights drawn from the documentation of various libraries, including Scikit-learn (Python), MLJ.jl (Julia), and mlpack (C++), among others [20], [21], [22], [23], [24], [25], [26]. In total, 15 implementations were developed, covering five algorithms across three languages. All implementations followed a consistent structure to ensure comparability across languages. The general workflow is outlined in the pseudocode in Figure 1. All source code is available in a public GitHub repository[4].

[4] https://github.com/kkkont/bsc-thesis

Algorithm 0: General Machine Learning Training and Evaluation Pipeline

Output: Model accuracy and the random seed used
Import necessary libraries (e.g., sklearn, mlpack, MLJ etc.)
Preprocess(Data);
    Identify and scale numerical features;
Main Procedure;
    Generate a random seed for reproducibility;
    Load dataset from a fixed file path;
    Call the preprocessing function to process the dataset;
    Split data into input features X and target labels y;
    Divide X and y into training and testing sets;
    Initialize a machine learning model from a library, using the random seed for reproducibility;
    Train the model on the training set;
    Make predictions on the test set;
    Calculate model accuracy;
    Output the accuracy and the random seed used;

Figure 1. Pseudocode for ML algorithm implementations

With the implementations completed, the next step is to ensure consistency in the evaluation process. To achieve this, a standardised dataset will be used across all algorithms to evaluate their energy efficiency, runtime performance, and accuracy. The following section details the dataset chosen for this study and the preprocessing steps applied to it.

3.3 Dataset Description

The same dataset will be employed across all algorithms to evaluate energy efficiency consistently. The chosen dataset, the Credit Card Fraud Detection Dataset 2023 from Kaggle[5], consists of over 550,000 instances and 30 features. This dataset encompasses credit card transactions made by European cardholders in 2023 and aims to classify transactions as either fraudulent (1) or legitimate (0), making it suitable for classification tasks. As the dataset already consists of numerical and standardised values, minimal preprocessing was required, allowing more focus on implementing and evaluating the algorithms. For each algorithm, the dataset was preprocessed with an 80:20 train-test split.

3.4 Experimental Procedure

The measurements were all conducted on a desktop with the following specifications: Linux Ubuntu 22.04.5 LTS operating system With 16GB of RAM, a 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz.

[5] https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023/data

Following the scientific guide by Luis Cruz [27], the system was configured in a minimal-interference state to ensure accurate measurements. This involved closing all applications, turning off non-essential services (e.g., WiFi, Bluetooth, VPN), turning off internet access, unplugging the power cable, and setting screen brightness to the lowest level. Additionally, the battery level was kept consistent across all measurements to reduce variability from background activity.

3.5 Measurements and Data Collection

To obtain accurate energy consumption measurements and execution time for each algorithm in every programming language, performance metrics were recorded using Perf[6], a Linux-based performance monitoring tool. Perf provides accurate system-level statistics, including energy usage in Joules (eliminating the need for conversions) and elapsed time in seconds. System-level measurement was preferred over hardware-level monitoring due to its practicality and portability, as it does not require specialised equipment. A previous study by Gordillo [16] found no significant differences between hardware and software-based energy rankings, indicating that the choice of measurement approach does not significantly impact the results. This makes Perf a reliable tool for the analysis.

Each machine learning algorithm was executed using `perf` commands to track energy consumption (in Joules) and execution time (in seconds), ensuring consistency across all programming languages [28].

The following command was used to collect energy measurements:

```
perf stat -e power/energy-cores/,power/energy-gpu/,power/energy-pkg/,power/energy-psys/ <command>
```

where `<command>` corresponds to running a specific algorithm, such as:

- **Python:** `python3 logistic_regression.py`
- **Julia:** `julia logistic_regression.jl`
- **C++:** `./logistic_regression`

[6] https://perfwiki.github.io/main/

Following the scientific guide by Luis Cruz [27], each machine learning algorithm was executed and measured 30 times to collect energy consumption and execution time samples. This approach helps minimise the impact of cold starts and cache effects, ensuring more consistent measurements. To prevent overheating, which could influence energy readings, a cooldown period of one minute was observed between each measurement.

Before measuring the energy consumption and execution time of each machine learning algorithm, baseline system measurements were also recorded. Using the same tool, Perf, and maintaining consistent system conditions, 30 measurements were taken over 60 seconds. The average energy consumption from these measurements was then calculated and converted to idle power (in watts). After collecting this baseline data, it was subtracted from subsequent measurements to obtain net values.

3.6 Data Preprocessing and Statistical Analysis

The measurement data was organised into separate files based on the algorithm name (e.g., decision_tree.csv, svm.csv), all of which can be found in the thesis GitHub repository[7]. Each file contains 90 measurements, with 30 measurements for each implementation. The dataset consists of the following columns, as detailed in Table 2.

Table 2. Measurements dataset structure

Field	Description
`experiment_no`	A unique identifier for each experimental run.
`language`	The programming language used for the experiment (e.g., Python, C++, Julia).
`energy_cores`	Energy consumed by CPU cores during the experiment (in Joules).
`energy_gpu`	Energy consumed by the GPU during the experiment (in Joules).
`energy_pkg`	Total energy consumed by the CPU package (in Joules)
`energy_psys`	Platform subsystem energy, including memory and peripheral usage (in Joules).
`total_energy`	Sum of all measured energy sources (cores, GPU, package, psys) in Joules.
`elapsed_time`	Time taken to complete the task (in seconds).
`accuracy`	Model accuracy achieved during the experiment (as a decimal fraction).
`random_seed`	Random seed used for the model to ensure reproducibility.
`net_energy`	Energy consumed by the experiment after subtracting the system's baseline (idle) energy consumption (in Joules).

[7] https://github.com/kkkont/bsc-thesis

To ensure data reliability, a Shapiro-Wilk test[8], a normality test, was initially performed on the measurement data (e.g., on net_energy, elapsed_time, and accuracy) of individual programs to identify and remove outliers. After removing the outliers, the Shapiro-Wilk test was re-applied to the entire dataset for each machine learning algorithm (e.g., decision tree across all languages) to assess the normality of each metric across all languages.

Based on the normality assessment, statistical tests were selected accordingly. When the normality assumption was not met, the Kruskal-Wallis test[9] was used, a non-parametric method for comparing the median values of three or more independent groups (in this case, programming languages). In cases where the normality assumption held, a one-way ANOVA[10] was applied, as it is more suitable for normally distributed data and serves a similar purpose as the Kruskal-Wallis test.

For all algorithms, these tests were used to evaluate whether statistically significant differences existed in energy consumption, runtime and accuracy across different programming language implementations. Such an analysis is essential for understanding how the choice of programming language, or the library used, affects model performance, energy consumption, and efficiency.

To measure the magnitude of observed differences, effect size metrics were calculated using Epsilon squared (ε^2) for non-parametric tests and Eta squared (η^2) for ANOVA.

To assess the trade-offs between energy consumption, runtime, and accuracy, the Spearman correlation coefficient[11] was employed. This coefficient is a statistical measure of the strength of a monotonic relationship between paired data. Specifically, three pairs of variables were examined: energy consumption & elapsed time, energy consumption & accuracy, and elapsed time & accuracy, to explore how these metrics are interrelated across different libraries.

[8] https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test
[9] https://en.wikipedia.org/wiki/Kruskal%E2%80%93Wallis_test
[10] https://en.wikipedia.org/wiki/One-way_analysis_of_variance
[11] https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

4. Results

In this section, an analysis and discussion of the results of the study are presented. While the primary focus is on understanding the energy efficiency of various library-based implementations of machine learning algorithms, it is equally important to explore the relationships between energy consumption, execution time, and accuracy. Furthermore, this section addresses the two research questions outlined in the Methodology, each discussed in its dedicated subsection.

4.1 Results of RQ1: *Do machine learning algorithms, when implemented using standard libraries, exhibit differences in energy consumption, runtime performance, and accuracy across programming languages?*

To answer **RQ1**, energy consumption, runtime performance and accuracy data were collected as described in Section 3.5. The Shapiro-Wilk test was initially applied to determine whether our data followed a normal distribution. The results revealed that for each algorithm implementation, the Shapiro-Wilk p-values for energy consumption, elapsed time, and accuracy were all very small (close to zero), indicating that these variables do not follow a normal distribution across most selected algorithm implementations. However, for the logistic regression algorithm, the accuracy p-value was 0.227, exceeding 0.05, suggesting that accuracy may exhibit a normal distribution for this algorithm across the three programming languages.

Since most data did not satisfy the normality assumption, the Kruskal-Wallis test was employed—a non-parametric method for comparing the median values of three or more independent groups (in this case, programming languages). The Kruskal-Wallis p-values for energy consumption, runtime, and accuracy were consistently very low (typically < 0.05, often ranging from 1e-16 to 1e-12). These results indicate statistically significant differences between the programming languages across all three performance metrics. An exception was observed in the logistic regression algorithm, where accuracy followed a normal distribution; here, a one-way ANOVA was applied, yielding a low p-value (0.00013) and signifying significant accuracy differences.

Following the Kruskal-Wallis and ANOVA tests, post-hoc analysis was performed using Epsilon squared (ε^2) and Eta squared (η^2) as measures of effect size to quantify further the magnitude of differences observed between programming languages across different algorithms and performance metrics. Across all algorithms, ε^2 values for energy consumption

and runtime were consistently high (typically > 0.86), indicating a large effect size and substantial differences between languages. For accuracy, while the effect sizes varied more, ranging from moderate (e.g., $\varepsilon^2 = 0.6416$ for SVM) to high (e.g., $\varepsilon^2 = 0.8564$ for Decision Tree), they still consistently pointed to meaningful differences. In the specific case of logistic regression accuracy, where data was normally distributed and ANOVA was applied, the η^2 value of 0.1938 also indicates a moderate to large effect size, suggesting that the choice of programming language had a notable influence on model accuracy. These post-hoc metrics confirm that the statistical significance observed in the initial tests is statistically valid and practically meaningful regarding performance impact.

Figure 2 provides a closer look at the energy consumption patterns of library-based algorithm implementations across different programming languages. In most cases, Python using the scikit-learn library demonstrated the most efficient energy performance. An exception to this trend was the SVM algorithm, where C++ with the mlpack library recorded the lowest energy usage. On the other hand, Julia, utilising the MLJ framework, generally showed the highest energy consumption across the board, except for the decision tree algorithm, where C++ had the highest energy usage.

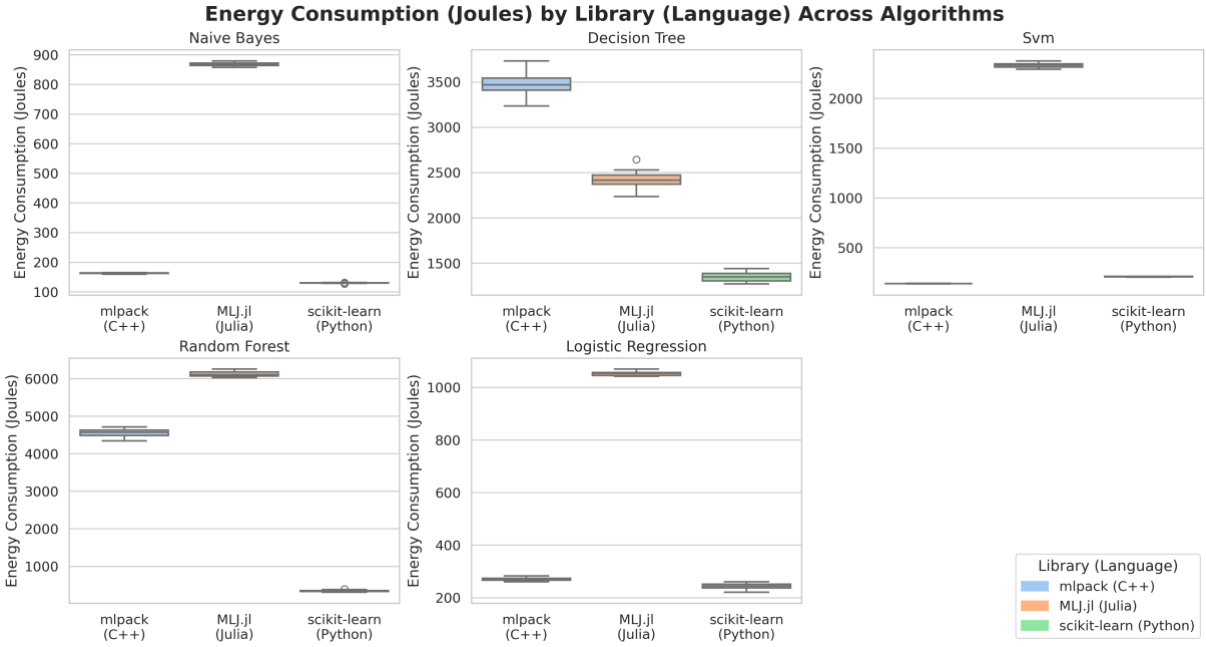

Figure 2. Energy consumption of algorithms across different programming languages and libraries

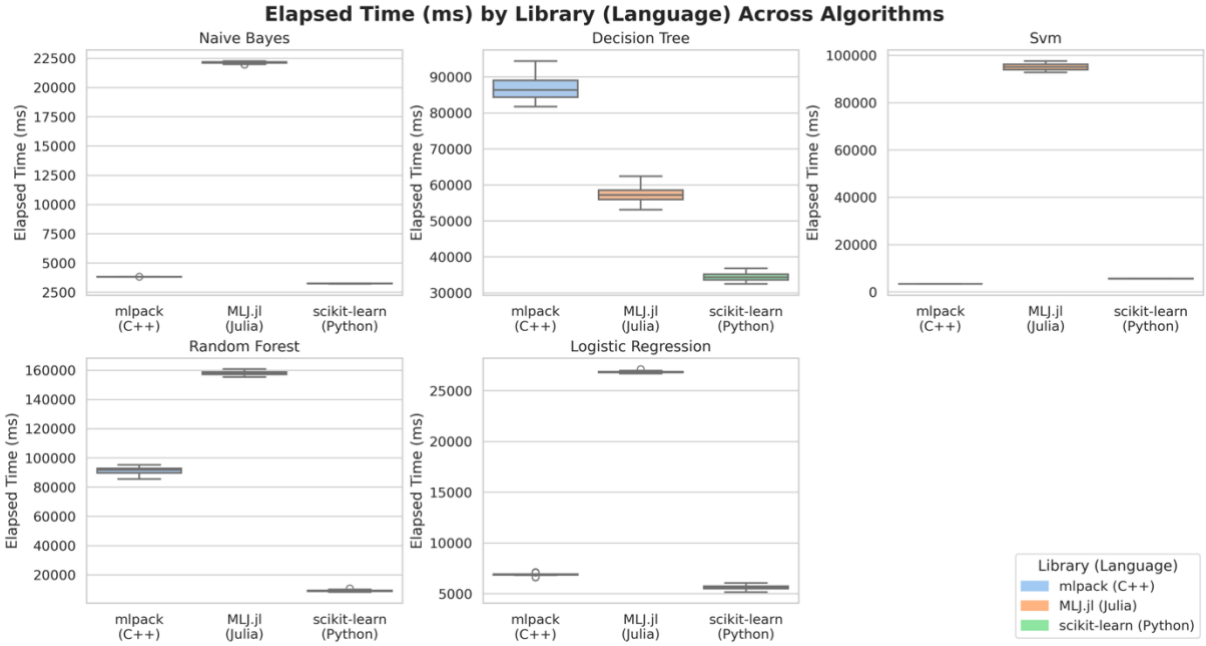

Figure 3. Elapsed time of algorithms across different programming languages and libraries

Building on these findings, Figure 3 presents the variations in elapsed time across the same set of languages and libraries. The results closely mirror those observed for energy consumption, highlighting the correlation between execution time and energy usage. Once again, Python with the scikit-learn library generally achieved the best performance in terms of speed, except for the SVM algorithm, where C++ and mlpack proved to be the fastest. Similarly, Julia with the MLJ framework tended to have the longest execution times in most cases, except for the decision tree algorithm, where C++ exhibited the slowest performance.

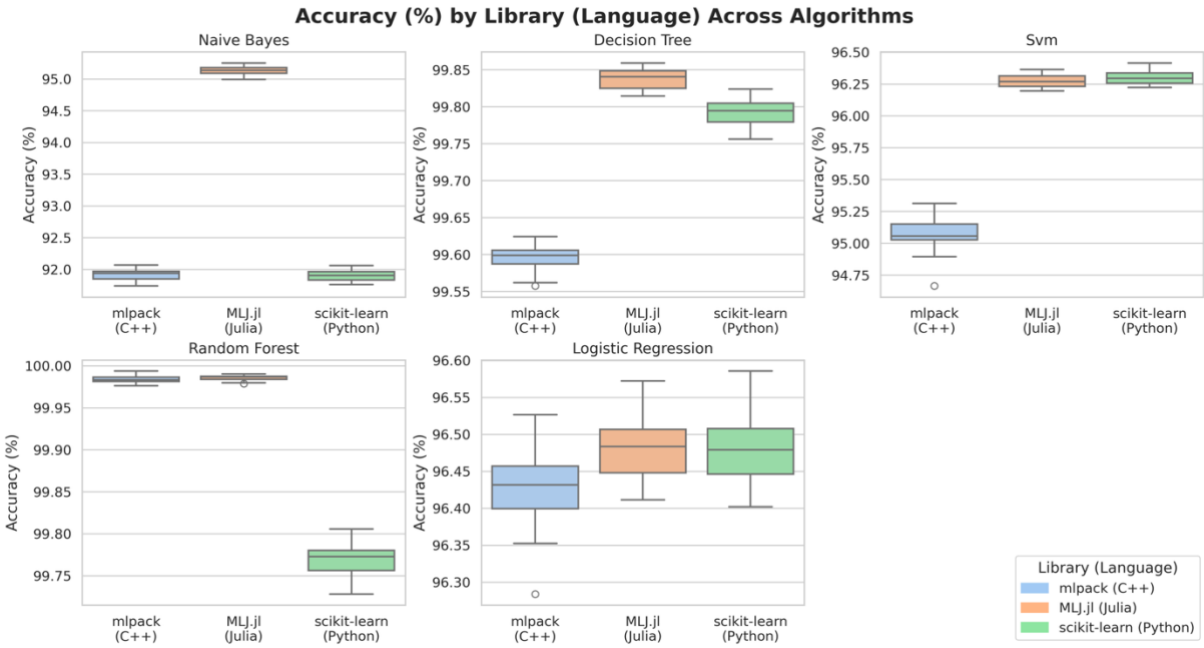

Figure 4. Accuracy of algorithms across different programming languages and libraries.

Turning to accuracy, Figure 4 reveals how the choice of language and library influences algorithmic performance. Using the MLJ framework, Julia achieved the highest accuracy in decision tree, naïve Bayes, and random forest, placing second, just behind Python with scikit-learn, in SVM and logistic regression. Python and scikit-learn led in SVM and logistic regression, but ranked second in decision tree and third in random forest and naïve Bayes. C++ with the mlpack library generally performed the worst in accuracy for decision tree, SVM, and logistic regression, although it placed second in random forest and naïve Bayes. As previously mentioned, the accuracy values for logistic regression displayed signs of normality based on the Shapiro-Wilk test. This trend is also visually evident in Figure 4, where the three languages/libraries exhibit the most significant variability in accuracy for Logistic Regression.

Summarising the findings from Figures Figure 2, Figure 3 and Figure 4, Table 3 presents the average measurement values to rank each algorithm's programming languages (and their associated libraries/frameworks) based on energy consumption. The rankings reflect performance from most to least efficient in terms of energy use. Arrows $\Downarrow_2$ show how many places the ranking would change if sorted by the third column (accuracy).

Table 3. Energy, runtime, and accuracy average values benchmarked for decision tree, random forest, SVM, logistic regression and naïve Bayes.

Decision Tree			
	Energy (Joules)	Time (ms)	Accuracy (%)
scikit-learn (Python) $\Downarrow_1$	1348,28	34460	99,79
MLJ.jl (Julia) $\Uparrow_1$	2420,26	57175	99,84
mlpack (C++)	3479,42	86722	99,6
Random Forest			
	Energy (Joules)	Time (ms)	Accuracy (%)
scikit-learn (Python) $\Downarrow_2$	345,88	9093	99,77
mlpack (C++)	4555,74	91260	99,98
MLJ.jl (Julia) $\Uparrow_2$	6123,13	157979	99,99
SVM			
	Energy (Joules)	Time (ms)	Accuracy (%)
mlpack (C++) $\Downarrow_2$	140,1	3449	95,07
scikit-learn (Python) $\Uparrow_1$	209,53	5640	96,3
MLJ.jl (Julia) $\Uparrow_1$	2331,82	97178	96,27
Logistic Regression			
	Energy (Joules)	Time (ms)	Accuracy (%)
scikit-learn (Python)	243,61	5610	96,48
mlpack (C++) $\Downarrow_1$	271,33	6900	96,43
MLJ.jl (Julia) $\Uparrow_1$	1052,51	26860	96,48
Naive Bayes			
	Energy (Joules)	Time (ms)	Accuracy (%)
scikit-learn (Python) $\Downarrow_2$	130,01	3249	91,9
mlpack (C++)	163,07	3827	91,92
MLJ.jl (Julia) $\Uparrow_2$	868,22	22130	95,14

Based on the statistical analysis, the alternative hypothesis **H1** was supported, confirming significant differences in energy consumption, runtime performance, and accuracy. As a result, the answer to **RQ1** is affirmative: library-based implementations of machine learning algorithms do vary in energy efficiency, execution time, and accuracy across C++, Python, and Julia. The Kruskal-Wallis and ANOVA tests, along with the post-hoc analysis, revealed statistically significant differences in performance metrics for algorithms including support vector machine (SVM), decision tree, logistic regression, naïve Bayes, and random forest. This confirms that the choice of programming language significantly impacts energy consumption, runtime, and accuracy.

4.2 Results of RQ2: *What trade-offs exist between energy consumption, runtime performance, and accuracy when implementing machine learning algorithms using standard libraries across different programming languages?*

To answer **RQ2** and assess the relationships between energy consumption, runtime, and accuracy, the Spearman correlation coefficient was used. This coefficient is a statistical measure of the strength of a monotonic relationship between paired data. Specifically, three pairs of variables were examined: energy consumption & elapsed time, energy consumption & accuracy, and elapsed time & accuracy, to evaluate how these metrics are interrelated across different programming languages. Since the Spearman method produces a coefficient between -1 and 1, indicating a negative or positive correlation, the classification method proposed by Rea and Parker [29] was adopted, as also applied in previous studies such as those by Pereira et al. [15]. The corresponding classification ranges for interpreting the correlation values are presented in Table 4.

Table 4. Strength of association based on Spearman's ρ as interpreted by Rea and Parker

Range	Strength Interpretation
$0.00 < \rho \leq 0.10$	Negligible
$0.10 < \rho \leq 0.20$	Weak
$0.20 < \rho \leq 0.40$	Moderate
$0.40 < \rho \leq 0.60$	Relatively strong
$0.60 < \rho \leq 0.80$	Strong
$0.80 \leq \rho \leq 1.00$	Very strong

Table 5, Table 6, Table 7, Table 8, and Table 9 present the results obtained for the algorithms across the three programming languages considered. The corresponding Spearman ρ values and their respective nominal classifications are provided for each pair of metrics within each language. Data analysis from these tables shows a consistent correlation between energy consumption and execution time. Statistical evidence indicates that a decrease in an algorithm's execution time is highly likely to reduce energy consumption. This direct relationship is expected, as it follows from the physical formula Energy (J) = Time (s) × Power (W).

Table 5. Spearman correlations between energy, time, and accuracy for naïve Bayes

Algorithm – Naive Bayes

	Energy & Time		Energy & Accuracy		Time & Accuracy	
	Spearman ρ	Correlation	Spearman ρ	Correlation	Spearman ρ	Correlation
mlpack (C++)	0.537	Relatively strong	0.175	Weak	-0.030	Negligible
MLJ.jl (Julia)	0.512	Relatively strong	-0.065	Negligible	-0.190	Weak
scikit-learn (Python)	0.654	Strong	0.001	Negligible	0.203	Moderate

Table 6. Spearman correlations between energy, time, and accuracy for logistic regression

Algorithm – Logistic Regression

	Energy & Time		Energy & Accuracy		Time & Accuracy	
	Spearman ρ	Correlation	Spearman ρ	Correlation	Spearman ρ	Correlation
mlpack (C++)	0.463	Relatively strong	0.131	Weak	0.012	Negligible
MLJ.jl (Julia)	0.363	Moderate	0.149	Weak	-0.158	Weak
scikit-learn (Python)	0.988	Very strong	0.005	Negligible	0.017	Negligible

Table 7. Spearman correlations between energy, time, and accuracy for random forest

Algorithm – Random Forest

	Energy & Time		Energy & Accuracy		Time & Accuracy	
	Spearman ρ	Correlation	Spearman ρ	Correlation	Spearman ρ	Correlation
mlpack (C++)	0.943	Very strong	0.085	Negligible	0.091	Negligible
MLJ.jl (Julia)	0.759	Strong	-0.082	Negligible	0.135	Negligible
scikit-learn (Python)	0.985	Very strong	0.051	Negligible	-0.169	Negligible

Table 8. Spearman correlations between energy, time, and accuracy for decision tree

Algorithm – Decision Tree

	Energy & Time		Energy & Accuracy		Time & Accuracy	
	Spearman ρ	Correlation	Spearman ρ	Correlation	Spearman ρ	Correlation
mlpack (C++)	0.965	Very strong	-0.060	Negligible	0.003	Negligible
MLJ.jl (Julia)	0.920	Very strong	0.047	Negligible	0.019	Negligible
scikit-learn (Python)	0.905	Very strong	-0.061	Negligible	-0.146	Negligible

Table 9. Spearman correlations between energy, time, and accuracy for SVM

Algorithm – SVM

	Energy & Time		Energy & Accuracy		Time & Accuracy	
	Spearman ρ	Correlation	Spearman ρ	Correlation	Spearman ρ	Correlation
mlpack (C++)	0.491	Relatively strong	-0.404	Relatively strong	0.-290	Moderate
MLJ.jl (Julia)	0.720	Strong	-0.082	Negligible	0.135	Weak
scikit-learn (Python)	0.776	Strong	-0.214	Moderate	-0.169	Weak

Additionally, the results demonstrate that, for most algorithms and languages, there is little to no correlation between energy consumption and algorithm accuracy (Table 5, Table 6, Table 7, Table 8, Table 9). An exception is observed for the SVM algorithm implemented in C++ (Table 9), where a relatively strong negative correlation between energy consumption and

accuracy is identified, indicating that higher energy consumption tends to be associated with lower accuracy, and lower energy consumption with higher accuracy.

Furthermore, the results indicate that, across all selected algorithms and languages, the correlation between execution time and accuracy is generally negligible to weak (Table 5, Table 6, Table 7, Table 8, Table 9). Exceptions include the SVM algorithm in C++ (Table 9), where a moderate negative correlation is present, suggesting that longer execution times are associated with lower accuracy, and shorter execution times with higher accuracy. Another exception is noted for the Python implementation of the naïve Bayes algorithm (Table 5), where a moderate positive correlation between execution time and accuracy is observed, meaning that, in this case, longer execution times are associated with improved accuracy.

Based on this statistical analysis, the alternative hypothesis **H1** is accepted, confirming that trade-offs exist between energy consumption, runtime performance and accuracy when implementing machine learning algorithms using standard libraries across programming languages. Therefore, in response to **RQ2**, it can be concluded that a consistent and statistically supported relationship exists between energy consumption and execution time: algorithms with reduced execution time also tend to exhibit lower energy consumption. However, the relationship between energy consumption and accuracy, as well as between execution time and accuracy, is generally weak or negligible for most algorithms and languages. This indicates that improving runtime performance and reducing energy consumption do not necessarily compromise model accuracy.

5. Threats to Validity

This study aimed to measure and compare the machine learning algorithms implemented using standard libraries across different programming languages. The analysis focused on three key metrics: energy consumption, runtime performance, and accuracy. This approach aims to provide deeper insight into how various languages and their associated libraries differ regarding energy efficiency and overall performance. In this subsection, potential threats to the validity of the study are discussed and categorised into four commonly recognised types [30]: conclusion validity, internal validity, construct validity, and external validity.

Conclusion validity. While statistical techniques such as the Kruskal-Wallis test, ANOVA, and Spearman correlation were appropriately applied, the limited number of algorithms and the use of a single dataset may constrain the generalisability of the findings and reduce the strength of causal inferences. However, the study's primary goal is not to generalise across all machine learning scenarios, but to examine whether consistent, statistically significant differences emerge in energy consumption, runtime, and accuracy when using different programming languages with standard libraries under controlled conditions. Repeated measurements, a large real-world dataset, and effect size analysis help ensure the reliability and validity of the observed trends.

Internal validity. This category concerns factors that may unintentionally influence the results of the study. A key threat to internal validity arises from potential variability in the system environment during measurement. For instance, differences in interpreter versions, background processes, or temporary system states could introduce noise into energy, runtime, and accuracy readings. Each algorithm implementation was executed under identical conditions and measured 30 times to mitigate these effects. Outliers were removed, and normality was assessed to ensure data consistency. These steps helped minimise the influence of unpredictable system behaviour. While some degree of system-level interference is inevitable, the consistency of the results across repeated trials supports the reliability of the measurements. It suggests that any remaining variability had minimal impact on the overall findings.

Construct validity. This study involved 15 implementations of five machine learning algorithms across three programming languages, each using a widely adopted standard library—scikit-learn for Python, MLJ.jl for Julia, and mlpack for C++. All implementations were developed by the author, following the official documentation of each library to ensure correct and idiomatic usage. While minor syntactic and structural differences between

languages were unavoidable, the core logic and structure of the code invoking the machine learning models were kept as consistent as possible across all implementations. This uniformity, combined with the use of a common dataset and evaluation procedure, ensures that the observed differences in performance metrics stem primarily from the libraries and languages themselves rather than from inconsistencies in implementation. As a result, the comparisons can be considered fair and meaningful within the scope of the study.

External validity. This category concerns the generalizability of the results beyond the specific conditions of the study. The implementations relied on each library's most recent stable versions at the time of experimentation. While future updates may improve performance or introduce more efficient models, such changes are unlikely to significantly alter the relative performance trends—unless a major architectural overhaul occurs. Additionally, results could vary slightly across different hardware configurations or operating systems. However, the experimental methodology was designed for ease of replication. The thesis public repository[12] provides all essential resources, including source code, language versions, compiler details, and library versions. This level of transparency supports both reproducibility and generalizability, enabling future researchers to replicate or extend the study under varied conditions.

[12] https://github.com/kkkont/bsc-thesis

6. Conclusion

This study examined the impact of programming language choice on machine learning algorithms' energy efficiency, runtime performance, and accuracy when implemented using standard libraries. Through a comprehensive experimental setup involving Python, Julia, and C++—each paired with popular ML libraries—and a consistent dataset, the study aimed to determine whether meaningful differences and trade-offs exist across implementations.

The findings confirm that programming language and associated libraries significantly influence all three performance metrics. Statistical tests revealed consistent and substantial differences in energy consumption, execution time, and model accuracy between languages. Notably, Python with scikit-learn generally offered the best energy efficiency and runtime balance. At the same time, Julia with MLJ often produced the most accurate results, albeit at higher energy and time costs. C++, although efficient in specific scenarios such as SVM, C++ tended to underperform in accuracy.

Additionally, the analysis of trade-offs revealed a strong and predictable relationship between energy consumption and runtime—faster implementations tend to consume less energy. In contrast, the relationship between these metrics (runtime and energy) and model accuracy was generally weak or inconsistent. This suggests that it is often possible to optimise models for computational efficiency (i.e., faster runtime and lower energy use) without significantly compromising their predictive performance (i.e., accuracy).

These insights are valuable for researchers and practitioners developing more sustainable AI systems. They also highlight the importance of considering the full stack—language, libraries, and implementation strategy—when designing ML workflows. While previous studies have often reinforced the preconception that Python is neither the fastest nor the most energy-efficient language for general tasks (especially when compared with C++), this study reveals that for machine learning tasks, Python can indeed be quite 'green' in terms of both energy efficiency and runtime performance. Future work could expand this analysis to other domains, include more diverse algorithms and languages, or explore parallel and GPU-based implementations for deeper insights into green AI.

References

[1] ‘About Face ID advanced technology’, Apple Support. Accessed: Jan. 16, 2025. [Online]. Available: https://support.apple.com/en-us/102381

[2] ‘Google Maps 101: How AI helps predict traffic and determine routes’, Google. Accessed: Jan. 16, 2025. [Online]. Available: https://blog.google/products/maps/google-maps-101-how-ai-helps-predict-traffic-and-determine-routes/

[3] Skillfloor, ‘Machine Learning in Finance: Predictive Analytics and Risk Assessment’, Medium. Accessed: Jan. 16, 2025. [Online]. Available: https://skillfloor.medium.com/machine-learning-in-finance-predictive-analytics-and-risk-assessment-701a8d60f12

[4] Z. Hong, ‘Personalized Recommendations: How Netflix and Amazon Use Deep Learning to Enhance User Experience’, Medium. Accessed: Jan. 16, 2025. [Online]. Available: https://medium.com/@zhonghong9998/personalized-recommendations-how-netflix-and-amazon-use-deep-learning-to-enhance-user-experience-e7bd6fcd18ff

[5] ITU and World Bank, *Measuring the Emissions and Energy Footprint of the ICT Sector*. World Bank and International Telecommunication Union, 2024.

[6] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, and A. Van Deursen, ‘Batching for Green AI - An Exploratory Study on Inference’, in *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Durres, Albania: IEEE, Sep. 2023, p. 112–119. doi: 10.1109/SEAA60479.2023.00026.

[7] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, and A. Van Deursen, ‘Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI’, in *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, Melbourne, Australia: IEEE, May 2023, p. 25–36. doi: 10.1109/CAIN58948.2023.00012.

[8] 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)*, Plovdiv, Bulgaria: IEEE, Jun. 2022, p. 35–45. doi: 10.1109/ICT4S55073.2022.00015.

[9] N. Alizadeh and F. Castor, ‘Green AI: A Preliminary Empirical Study on Energy Consumption in DL Models Across Different Runtime Infrastructures’, in *2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, Apr. 2024, p. 134–139. Accessed: Jan. 15, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10556302

[10] S. O. S. Santos, A. Skiarski, D. García-Núñez, V. Lazzarini, R. De Andrade Moral, E. Galvan, A. L. C. Ottoni, and E. Nepomuceno, ‘Green Machine Learning: Analysing the Energy Efficiency of Machine Learning Models’, in *2024 35th Irish Signals and Systems Conference (ISSC)*, Belfast, United Kingdom: IEEE, Jun. 2024, p. 1–6. doi: 10.1109/ISSC61953.2024.10603302.

[11] OpenAI (2022). ChatGPT (GPT-4o): https://chatgpt.com

[12] Grammarly (Version 1.117.0): https://www.grammarly.com/

[13] M. Couto, R. Pereira, F. Ribeiro, R. Rua, and J. Saraiva, ‘Towards a Green Ranking for Programming Languages’, in *Proceedings of the 21st Brazilian Symposium on Programming Languages*, Fortaleza CE Brazil: ACM, Sep. 2017, p. 1–8. doi: 10.1145/3125374.3125382.

[14] 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, Vancouver BC Canada: ACM, Oct. 2017, p. 256–267. doi: 10.1145/3136014.3136031.
[15] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, 'Ranking programming languages by energy efficiency', *Sci. Comput. Program.*, vol. 205, p. 102609, May 2021, doi: 10.1016/j.scico.2021.102609.
[16] A. Gordillo, C. Calero, Á. Moraga, F. Garcia, J. P. Fernandes, R. Abreu, and J. Saraiva, 'Programming languages ranking based on energy measurements', *Softw. Qual. J.*, vol. 32, no. 4, p. 1539–1580, Dec. 2024, doi: 10.1007/s11219-024-09690-4.
[17] K. Kolodiazhnyi, *Hands-On Machine Learning with C++: Build, Train, and Deploy End-To-end Machine Learning and Deep Learning Pipelines*, 2nd ed. Birmingham: Packt Publishing, Limited, 2024.
[18] K. Gao, G. Mei, F. Piccialli, S. Cuomo, J. Tu, and Z. Huo, 'Julia language in machine learning: Algorithms, applications, and open issues', *Comput. Sci. Rev.*, vol. 37, p. 100254, Aug. 2020, doi: 10.1016/j.cosrev.2020.100254.
[19] A. Zollanvari, *Machine Learning with Python: Theory and Implementation*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-33342-2.
[20] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *J. Mach. Learn. Res.*, vol. 12, no. 85, p. 2825–2830, 2011.
[21] The pandas development team, *pandas-dev/pandas: Pandas*. (Sep. 20, 2024). Zenodo. doi: 10.5281/ZENODO.3509134.
[22] A. Blaom, F. Kiraly, T. Lienart, Y. Simillides, D. Arenas, and S. Vollmer, 'MLJ: A Julia package for composable machine learning', *J. Open Source Softw.*, vol. 5, no. 55, p. 2704, Nov. 2020, doi: 10.21105/joss.02704.
[23] R. R. Curtin *et al.*, 'mlpack 4: a fast, header-only C++ machine learninglibrary', *J. Open Source Softw.*, vol. 8, no. 82, p. 5026, Feb. 2023, doi: 10.21105/joss.05026.
[24] C. Sanderson and R. Curtin, 'Armadillo: An Efficient Framework for Numerical Linear Algebra'.
[25] M. Bouchet-Valat and B. Kamiński, '**DataFrames.jl** : Flexible and Fast Tabular Data in *Julia*', *J. Stat. Softw.*, vol. 107, no. 4, 2023, doi: 10.18637/jss.v107.i04.
[26] C. Sanderson and R. Curtin, 'Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation', *Math. Comput. Appl.*, vol. 24, no. 3, p. 70, Jul. 2019, doi: 10.3390/mca24030070.
[27] L. Cruz, 'Green Software Engineering Done Right: a Scientific Guide to Set Up Energy Efficiency Experiments', 2023, *figshare*. doi: 10.6084/M9.FIGSHARE.22067846.V1.
[28] L. Cruz, 'Tools to Measure Software Energy Consumption from your Computer', 2022, *figshare*. doi: 10.6084/M9.FIGSHARE.19145549.V1.
[29] L. M. Rea and R. A. Parker, *Designing and conducting survey research: a comprehensive guide*, 4. ed. San Francisco, Calif: Jossey-Bass, 2014.
[30] T. D. Cook and D. T. Campbell, *Quasi-experimentation: design & analysis issues for field settings*. Boston: Houghton Mifflin, 1979.

License

I, **Kadri-Ketter Kont**,

1. grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

   Benchmarking Energy and Performance of Standard Machine Learning Libraries: An Empirical Study,

   supervised by **Hina Anwar, PhD**;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kadri-Ketter Kont
15/05/2025