Towards Sustainable Machine Learning: Analyzing Energy-Efficient Algorithmic Strategies for Environmental Sensor Data

Berkay Cetkin [1], Lejla Begic Fazlic [1], Achim Guldner [1], Stefan Naumann [1], and Guido Dartmann [1]

Abstract:

This study evaluates the energy efficiency of machine learning (ML) classification models across 49 test setups, each representing different conditions derived from a set of scenarios. Utilizing internet of things (IoT) technology with an ESP8266 microcontroller, we collected and analyzed environmental data including temperature, humidity, and CO_2 levels from a simulated room environment. We measured energy consumption for data preprocessing, model training, and testing, alongside energy efficiency metrics that consider output, processing time, and F1 score. The study also performed correlation analyses to explore the relationship between energy consumption and performance metrics. Furthermore, it assessed the trade-offs between accuracy and energy efficiency by comparing an ensemble model to its constituent algorithms. The measurements, conducted according to the Green Software Measurement Model (GSMM), provide essential insights into selecting energy-efficient algorithms for a broad spectrum of IoT applications.

Keywords: algorithmic optimization, energy efficiency, internet of things, machine learning

1 Introduction

In the rapidly expanding fields of internet of things (IoT) and machine learning (ML), the deployment of smart devices in diverse environments is transforming data collection and processing. However, as the number of these devices increases, their total energy consumption also rises. In light of global efforts to reduce energy consumption and carbon emissions, this trend raises critical questions about the sustainability of IoT solutions, especially if they are combined with ML approaches for data analysis. This study addresses this pressing issue by exploring the energy efficiency of ML algorithms within IoT frameworks. The need for sustainable practices in algorithm development and deployment is highlighted, emphasizing the need to optimize energy use without compromising the performance of IoT systems.

The objective of this research is to conduct a systematic analysis and comparison of the energy efficiency of six ML algorithms, along with an ensemble model (EM) derived from

1 Trier University of Applied Science, Institute for Sofware Systems (ISS), Birkenfeld, Germany,
b.cetkin@umwelt-campus.de, https://orcid.org/0009-0003-6531-8626;
l.begic@umwelt-campus.de, https://orcid.org/0000-0002-9869-0219;
a.guldner@umwelt-campus.de, https://orcid.org/0000-0002-7532-4523;
s.naumann@umwelt-campus.de, https://orcid.org/0009-0000-6542-2229;
g.dartmann@umwelt-campus.de, https://orcid.org/0000-0002-6786-6664

these algorithms. This EM operates based on a majority voting mechanism, utilizing the outputs of the individual algorithms. The specific algorithms analyzed are from the Python ML library scikit-learn [Pe11] and include the Gradient Boosting Classifier (GBC), Hist Gradient Boosting Classifier (HGBC), Bagging Classifier (BC), Extra Trees Classifier (ETC), Decision Tree Classifier (DTC), and Random Forest Classifier (RFC). The EM and each algorithm are evaluated across seven distinct scenarios using environmental sensor data. In each scenario, the type and dimension of features are varied, resulting in a total of 49 unique model-scenario combinations. All measurements in the study were conducted according to the procedures and criteria outlined in the Green Software Measurement Model (GSMM) [Gu24]. To comprehensively assess energy consumption, the study considered the total energy used, measured in watt-hours, across all scenarios for preprocessing, training, and testing of each algorithm. Additionally, the research introduced three key metrics to measure energy efficiency. These metrics aid in understanding the trade-offs between energy consumption and algorithmic performance, and help to identify the scenarios in which algorithms operate most efficiently.

2 Related Work

The concept of eco-friendly and sustainable software, introduced by [Na11], continues to influence technology development towards environmental consciousness. Rapid technological advancements necessitate frequent hardware updates, which significantly impact the environment due to emissions from manufacturing processes. Recent studies [Da20; Gó20; Te23a; Te23b] emphasize the development of energy-efficient ML models, especially critical for power-sensitive applications like IoT devices. Research [Is23] has measured the energy usage of various ML classifiers and their carbon footprints based on regional electricity generation emissions. A study by [Ve22] shows that data-centric modifications can significantly improve AI systems' energy efficiency. [GLG17] highlight the importance of evaluating energy consumption when assessing data mining algorithms, demonstrating potential energy savings through optimization. Comparative research in industrial settings assesses ML models' environmental impacts, considering training duration, CO_2 emissions, and energy consumption [Hu23]. The methodology for measuring AI-based methods has been detailed in works by [GKN21] and further elaborated by authors in [Gu24]. [Ke18] developed a causal model to assess the indirect impact of software products on natural resources. This research enhances our ability to choose environmentally efficient AI solutions and contributes to sustainable technology deployment.

3 Data and Methods

3.1 Data Generation with a Ventilation Demonstrator

We developed a demonstrator designed to simulate a room equipped with two windows, aimed at capturing ventilation-related data through a sensor array as time series information.

At the core of this setup is the IoT Octopus[2], conceptualized by the IoT Workshop—a collaboration between the IoT expert group[3] and the Environmental Campus Birkenfeld in Germany—which primarily features an ESP8266[4] microcontroller and a Bosch BME680[5] gas sensor. The latter allows for the assessment of relative humidity, atmospheric pressure, ambient temperature, and volatile organic compounds (VOCs). To enhance the precision of the collected data, we also attached a superior SCD30[6] sensor to the Octopus, enabling more accurate measurements of CO_2 levels, humidity, and temperature. Given the demonstrator's objective to simulate a room with two windows and ascertain the current ventilation status—thereby predicting the condition of the windows and whether ventilation is occurring, to provide pertinent ventilation recommendations—two reed switches[7] were affixed to the Octopus, one for each window. These electromechanical switches, activated by a magnetic field, serve to determine the status of each window, which can be closed, open, or tilted. To collect data, we positioned a vial of carbonated water within the demonstrator to simulate people breathing in the room, thereby attempting to mimic the environmental conditions of an enclosed space. The data is subsequently transmitted to a Raspberry Pi[8] using MQTT[9], a lightweight messaging protocol tailored for small sensors and mobile devices. On the Raspberry Pi, Node-RED[10], a flow-based development tool for visual programming, is utilized to manage data flows and facilitate storage in a database. The chosen database for this setup is InfluxDB[11], a time-series database optimized for high write and query loads, with Grafana[12] being employed for data visualization purposes. Additionally, a graphical user interface (GUI) was established using Node-RED, not only to provide a more simplified visualization of the data but also to control the Octopus. The complete assembly of the demonstrator is depicted in Fig. 1.

3.2 Data Preprocessing

The entire process of data preprocessing is depicted in Fig. 2. In the illustration, $X_{P,1:N}$ represents the collected sensor values, while $y_{1:N}$ denotes the labeled values. Each entry in $y_{1:N}$ can assume a value from 0 to 8, corresponding to two windows each exhibiting three distinct states (yielding 3^2 combinations). The data encoding scheme ensures that a $y_{1:N}$ value of 0 indicates both demonstrator windows are closed, with any other value indicating that ventilation is currently occurring.

---

2 `https://www.tindie.com/products/FabLab/iot-octopus-badge-for-iot-evaluation` [2024-05-08]
3 `http://www.iot-werkstatt.de` [2024-05-08]
4 `https://www.espressif.com/en/products/socs/esp8266` [2024-05-08]
5 `https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme680/` [2024-05-08]
6 `https://sensirion.com/products/catalog/SCD30/` [2024-05-08]
7 `https://www.sparkfun.com/products/13247` [2024-05-08]
8 `https://www.raspberrypi.com` [2024-05-08]
9 `https://mqtt.org/` [2024-05-08]
10 `https://nodered.org` [2024-05-08]
11 `https://www.influxdata.com` [2024-05-08]
12 `https://grafana.com` [2024-05-08]

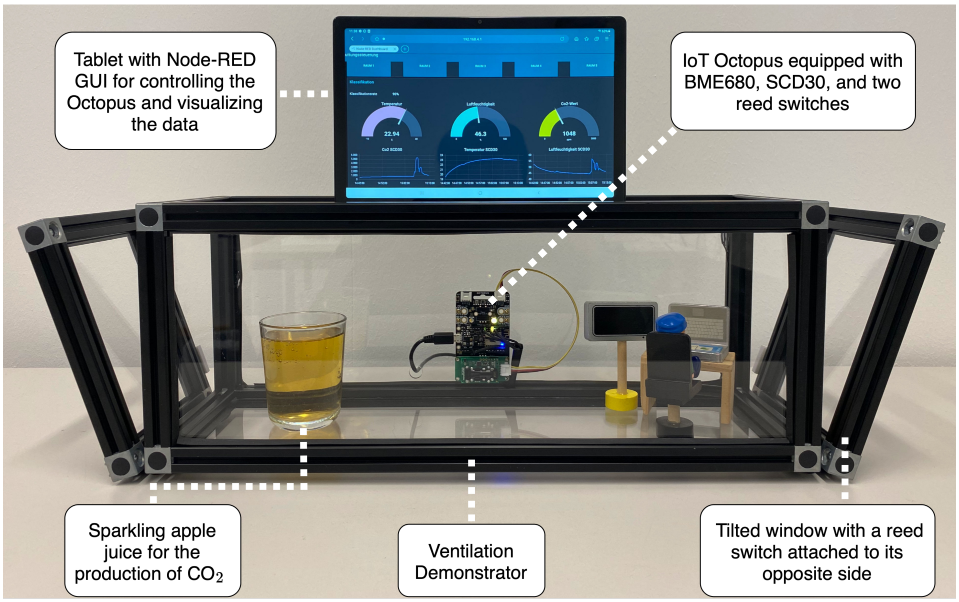

Fig. 1: Ventilation demonstrator

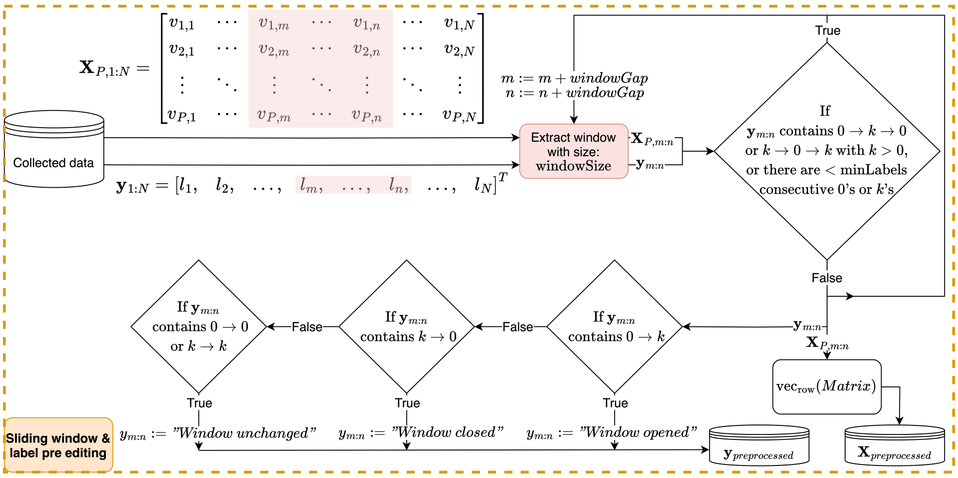

Fig. 2: Preprocessing methodology

The data extraction follows a sliding window technique, where the $windowSize$, represented as a red box in the figure, is set to 12 for this study. The sensor data was recorded at a sampling frequency of 0.2 Hz, meaning a sensor value was captured every 5 seconds by the Octopus device. Consequently, a $windowSize$ of 12 encapsulates data from the past minute, with larger sizes typically enhancing accuracy but increasing reliance on longer data periods. The $windowGap$ is set to 1, shifting the window by one column after each extraction step.

Post-extraction, certain windows are discarded based on specific conditions: if the associated labels for a window change from 0 to k and back to 0, or from k to 0 and back to k (with $k > 0$, indicating an open window), or if fewer than $minLabels$ (set to 4 in this study) of consecutive 0's or k's occur, then the window is excluded. This criterion helps eliminate windows that only briefly exhibit a ventilation state, enhancing the robustness of the data.

The remaining windows undergo label adjustment: if labels within a window change from 0 to k, the window is relabeled as *"Window opened"*. If labels change from k to 0, it is marked *"Window closed"*. If there is no change (either 0 to 0 or k to k), it is labeled *"Window unchanged"*. Thus, each extracted window is simplified to one of three possible labels. Finally, the extracted windows are vectorized by row, meaning the matrices under consideration ($\mathbf{X}_{P,m:n}$) are concatenated such that they form a single vector, with all rows of the original matrix combined into a single row.

3.3 Prepared Scenarios

The seven distinct scenarios we are referring to, are characterized by various combinations of sensor readings: CO_2, humidity, and temperature. The specific sensor readings employed in each scenario are detailed in Tab. 1, where an 'X' denotes their inclusion. All scenarios, which are applied to each of the ML algorithms mentioned, are characterized by the following sequence in the pipeline: initially, the data preprocessing phase occurs, followed by the training and testing phases. These training and testing phases are conducted 100 times, with the *random_state* parameter of each considered algorithm being set to the value of the respective iteration. This approach is adopted to enable a more precise performance measurement concerning the F1 score metric, as the presence of a *random_state* parameter means that a single iteration would not be robust enough. Additionally, each scenario utilized an 80% to 20% train/test split ratio. Throughout the iterations, different subsets of training and test data were extracted from the total dataset, which consists of approximately 2800 windows, as detailed in Fig. 2.

Tab. 1: Scenario Configurations—Representation of Sensor Readings Considered for Each Scenario

Readings	**Scen. 1**	**Scen. 2**	**Scen. 3**	**Scen. 4**	**Scen. 5**	**Scen. 6**	**Scen. 7**
CO_2	X			X	X		X
Humidity		X		X		X	X
Temperature			X		X	X	X

4 Measurement Results and Discussion

The resource measurements follow the methodology and guidelines set forth in the GSMM [Gu24]. Our measurements were conducted on a computer system equipped with an Intel Core i5-650 CPU. The system included 4 GB of memory, distributed across two 2 GB RAM modules. The storage configuration consisted of a 500 GB hard disk drive (HDD) for ample storage space and a 250 GB solid-state drive (SSD) for fast data access. The analysis is structured around the 49 model-scenario combinations mentioned in the introduction, considering a broad range of metrics for a balanced evaluation. All results from our measurements are presented in the Tables 2 and 3. Additional details, including the relevant code and datasets, are available in our GitLab repository at `https://gitlab.rlp.net/rgdsai/gc-em`.

Tab. 2: Combined Performance Metrics for Measurements

Performance Metrics - HGBC							
Metric	**Scen.1**	**Scen.2**	**Scen.3**	**Scen.4**	**Scen.5**	**Scen.6**	**Scen.7**
Mean Power [W]	86.92	**86.53**	**89.99**	87.20	89.83	87.91	87.76
Preprocessing Time [s]	0.57	0.57	**0.56**	0.67	0.67	0.68	**0.80**
Train Time [s]	0.91	**0.84**	1.04	0.90	1.06	0.99	**1.09**
Test Time [s]	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Total Energy [Wh]	**0.00**	**0.00**	**0.03**	**0.00**	**0.03**	**0.00**	**0.03**
F1 Score	0.79	0.74	**0.43**	**0.87**	0.81	0.73	0.86
Performance Metrics - BC							
Mean Power [W]	**76.65**	**69.23**	76.40	73.87	73.90	75.99	75.48
Preprocessing Time [s]	**0.57**	0.58	**0.57**	0.68	0.68	0.68	**0.78**
Train Time [s]	0.19	0.18	**0.13**	0.41	0.36	0.36	**0.62**
Test Time [s]	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
Total Energy [Wh]	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F1 Score	**0.77**	0.62	**0.48**	0.76	0.69	0.62	0.70
Performance Metrics - DTC							
Mean Power [W]	**58.48**	61.33	62.13	58.99	63.86	66.89	**74.86**
Preprocessing Time [s]	0.57	0.57	0.57	**0.18**	0.67	0.67	**0.78**
Train Time [s]	0.03	0.03	**0.02**	**0.13**	0.06	0.06	0.10
Test Time [s]	**0.00**	**0.00**	**0.00**	**0.03**	**0.00**	**0.00**	**0.00**
Total Energy [Wh]	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F1 Score	0.76	0.64	**0.46**	**0.77**	0.71	0.64	0.73
Performance Metrics - RFC							
Mean Power [W]	76.61	**76.25**	77.43	78.22	77.25	78.06	**79.74**
Preprocessing Time [s]	**0.57**	0.58	**0.57**	0.68	0.66	0.68	**0.78**
Train Time [s]	0.61	0.61	**0.45**	0.84	0.68	0.73	**1.11**
Test Time [s]	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Total Energy [Wh]	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.03**	**0.00**
F1 Score	**0.82**	0.61	**0.50**	**0.82**	0.79	0.66	0.77

The F1 performance of the ML models under review is illustrated in Fig. 3. It is apparent that nearly all models achieve their peak accuracy in Scenario 4, where CO_2 and humidity are measured in combination. Conversely, scenarios involving temperature, particularly Scenario 3, which exclusively considers temperature, exhibit the lowest accuracy. This outcome was expected, as temperature plays a less critical role in the data generation of the

Tab. 3: Combined Performance Metrics for Measurements (cont.)

			Performance Metrics - GBC				
Metric	**Scen.1**	**Scen.2**	**Scen.3**	**Scen.4**	**Scen.5**	**Scen.6**	**Scen.7**
Mean Power [W]	82.81	83.40	**82.51**	**84.13**	83.83	84.08	83.07
Preprocessing Time [s]	0.57	0.57	**0.56**	0.68	0.68	0.68	**0.79**
Train Time [s]	3.18	3.33	**2.30**	5.90	4.83	4.88	**8.49**
Test Time [s]	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
Total Energy [Wh]	0.07	0.08	**0.05**	0.14	0.11	0.11	**0.20**
F1 Score	0.75	0.65	**0.47**	**0.77**	0.70	0.64	0.71
			Performance Metrics - ETC				
Mean Power [W]	71.09	70.52	69.65	**69.25**	69.51	73.11	**80.80**
Preprocessing Time [s]	**0.56**	0.57	0.57	0.67	0.69	0.67	**0.78**
Train Time [s]	**0.22**	**0.22**	**0.27**	0.23	0.24	0.24	**0.27**
Test Time [s]	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Total Energy [Wh]	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F1 Score	0.91	0.76	**0.52**	**0.97**	0.90	0.81	0.95
			Performance Metrics - EM				
Mean Power [W]	85.03	**85.08**	84.98	84.58	**84.47**	84.81	84.55
Preprocessing Time [s]	1.10	**0.56**	1.06	1.18	**1.19**	**1.19**	0.77
Train Time [s]	2.77	5.21	**2.54**	2.82	2.80	2.88	**10.66**
Test Time [s]	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**	**0.09**
Total Energy [Wh]	0.10	0.12	**0.08**	0.10	0.10	0.10	**0.27**
F1 Score	**0.83**	0.66	**0.50**	**0.83**	0.79	0.69	0.79

ventilation demonstrator compared to real-world rooms with actual windows opening to the outside, where significant temperature differences between the interior and exterior can occur. Notably, even though Scenario 4 typically offers the best scores, switching to Scenario 1, which only measures CO_2, can be done with minimal loss in accuracy. This finding is crucial, as Scenario 1 is more energy-efficient and faster in execution times due to its lower feature dimensions compared to Scenario 4. Such insights underscore the importance of a precise analysis of the features under consideration to maximize optimization potential. To further analyze energy efficiency, we introduce two metrics: Time Based Energy Efficiency (TBEE) and Performance Based Energy Efficiency (PBEE), which are defined in Eq. (1).

$$\eta_{TBEE} = \frac{\text{Total Energy [Wh]}}{\text{Total Time [s]}} \qquad \eta_{PBEE} = \frac{\text{Total Energy [Wh]}}{\text{F1 Score}} \tag{1}$$

The associated plots displaying results derived from these metrics are presented in Fig. 4. As expected, the EM consumes the most energy and time since it combines various algorithms. The HGBC model shows the most fluctuations across the scenarios in both charts, demonstrating inconsistent performance. Conversely, the ETC model proves to be the most efficient among the evaluated models, consistently achieving the highest accuracy and, when combined with energy efficiency, the best overall results. To determine which algorithms were more or less energy-efficient than the EM across the scenarios, we

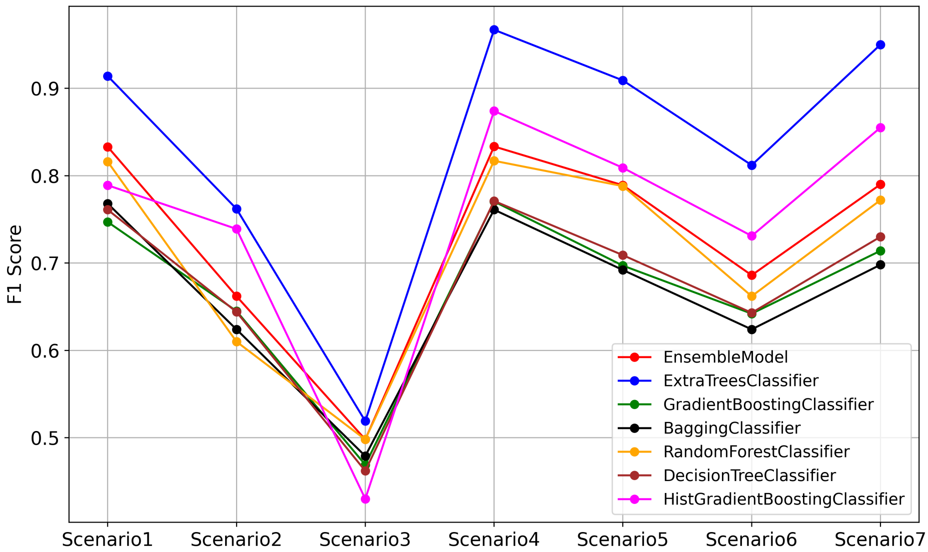

Fig. 3: F1 score results across different scenarios

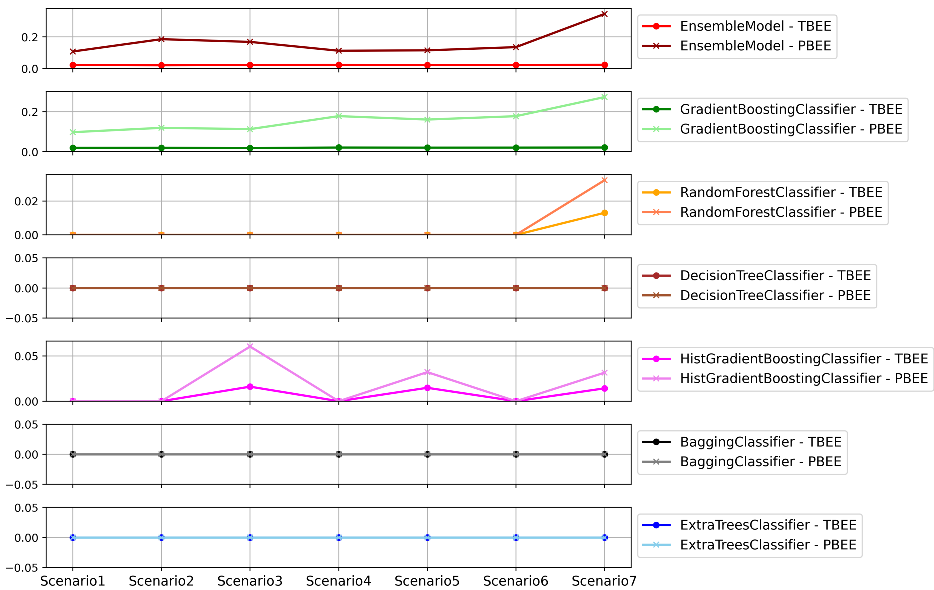

Fig. 4: TBEE and PBEE results across different scenarios

performed additional analysis using our proposed Relative Performance (RP) metric, as defined in Eq. (2).

$$\eta_{RP} = \frac{\eta^{Clf}_{PBEE} - \eta^{EM}_{PBEE}}{\eta^{EM}_{PBEE}} \tag{2}$$

The results are displayed in Fig. 5. If an algorithm shows a positive percentage difference from EM, it is less energy-efficient; the greater the positive value, the lesser its efficiency. Conversely, a negative percentage indicates higher energy efficiency compared to EM, with larger negative values denoting significant improvements.

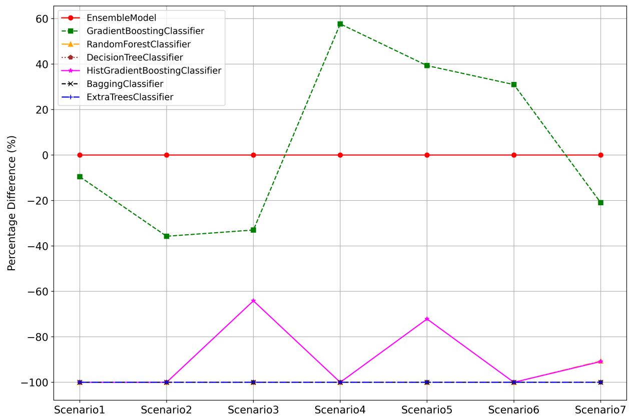

Fig. 5: Relative performance to EM

5 Conclusion

The findings of this study shed light on the complex interplay between model performance and environmental impact in ML. By exploring the role of feature selection, model simplicity, and the integration of environmental sensors in IoT settings, we uncover valuable insights that can guide practitioners towards more sustainable and efficient practices. The results challenge the notion that increased complexity always leads to better outcomes, instead highlighting the potential of streamlined models to achieve comparable accuracy while minimizing energy consumption. As we continue to push the boundaries of ML, this research serves as a reminder to approach model development with a holistic perspective, considering not only performance but also the environmental footprint of our choices.

Acknowledgements

This work was funded by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety, and Consumer Protection project “KIRA” under Grant 67KI32013B, and the German Federal Ministry for Economic Affairs and Climate Action project “EASY” under Grant 01MD22002D.

References

[Da20] Daghero, F.; Burrello, A.; Jahier Pagliari, D.; Benini, L.; Macii, E.; Poncino, M.: Energy-Efficient Adaptive Machine Learning on IoT End-Nodes With Class-Dependent Confidence. In. Pp. 1–4, 2020, DOI: 10.1109/ICECS49266.2020.9294863.

[GKN21] Guldner, A.; Kreten, S.; Naumann, S.: Exploration and systematic assessment of the resource efficiency of Machine Learning, INFORMATIK 2021, 2021, DOI: 10.18420/informatik2021-023.

[GLG17] Garcia-Martin, E.; Lavesson, N.; Grahn: Identification of energy hotspots: A case study of the very fast decision tree. International Conference: Green, Pervasive, and Cloud Computing, pp. 267–281, 2017.

[Gó20] Gómez-Carmona, O.; Casado-Mansilla, D.; Kraemer, F. A.; López-de-Ipiña, D.; García-Zubia, J.: Exploring the computational cost of machine learning at the edge for human-centric Internet of Things. Future Generation Computer Systems 112, pp. 670–683, 2020, ISSN: 0167-739X, DOI: https://doi.org/10.1016/j.future.2020.06.013.

[Gu24] Guldner, A. et al.: Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green Software Measurement Model (GSMM). Future Generation Computer Systems 155, pp. 402–418, 2024, ISSN: 0167-739X, DOI: 10.1016/j.future.2024.01.033.

[Hu23] Huertas-García, Á.; Martí-González, C.; Maezo, R. G.; Rey, A. E.: A Comparative Study of Machine Learning Algorithms for Anomaly Detection in Industrial Environments: Performance and Environmental Impact, 2023, DOI: https://doi.org/10.48550/arXiv.2307.00361, arXiv: 2307.00361 [cs.LG].

[Is23] Islam, M. S.; Zisad, S. N.; Kor, A.-L.; Hasan, M. H.: Sustainability of Machine Learning Models: An Energy Consumption Centric Evaluation. In: 2023 International Conference on Electrical, Computer and Communication Engineering (ECCE). Pp. 1–6, 2023, DOI: 10.1109/ECCE57851.2023.10101532.

[Ke18] Kern, E. et al.: Sustainable software products—Towards assessment criteria for resource and energy efficiency. Future Generation Computer Systems 86, pp. 199–210, 2018, ISSN: 0167-739X, DOI: https://doi.org/10.1016/j.future.2018.02.044.

[Na11] Naumann, S.; Dick, M.; Kern, E.; Johann, T.: The greensoft model: A reference model for green and sustainable software and its engineering. Sustainable Computing: Informatics and Systems 1 (4), pp. 294–304, 2011.

[Pe11] Pedregosa, F. et al.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, https://github.com/scikit-learn/scikit-learn [2024-05-08], pp. 2825–2830, 2011.

[Te23a] Tekin, N.; Acar, A.; Aris, A.; Uluagac, A. S.; Gungor, V. C.: Energy consumption of on-device machine learning models for IoT intrusion detection. Internet of Things 21, p. 100670, 2023.

[Te23b] Tekin, N.; Aris, A.; Acar, A.; Uluagac, S.; Gungor, V. C.: A review of on-device machine learning for IoT: An energy perspective. Ad Hoc Networks, p. 103348, 2023.

[Ve22] Verdecchia, R. et al.: Data-Centric Green AI An Exploratory Empirical Study. In: 2022 International Conference on ICT for Sustainability (ICT4S). IEEE Computer Society, Los Alamitos, CA, USA, pp. 35–45, 2022, DOI: 10.1109/ICT4S55073.2022.00015.