HowSUS: A Sustainability Scoring Framework for Open-Source Libraries

Seyidali Bulut, Johan van den Berg, Michal Kuchar, Artin Sanaye

March 2025

1 Introduction

In 2024 the Nutri-Score was introduced in the Netherlands [15]. Food producers can voluntarily put this on their packaging to help the consumer make the healthier choice. This idea of influencing consumer behaviour based on a label can be extended to software libraries and their sustainability. The need for guiding users to more sustainable libraries is becoming more and more evident. In 2018 a study found that if no improvements were made, the share of global emissions from the ICT industry would increase to 14% in 2040 [5]. This was so even before the recent explosive growth of AI-related computing. The energy consumption of generating one image using GenAI is the same as that of fully charging your phone [10]. The increasing need for attention to sustainable software prompted our research into applying Nutri-Scores to software libraries.

Previous research has been conducted into measuring the sustainability of software and how to improve such systems in Noman et al. [12]. Additionally, proposals exist for a sustainability label for software, such as in Deneckère and Rubio [7] and energy labelling for specific tasks such as machine learning models [8]. However, a generalisation of these proposals to all types of software libraries is missing. We aim to close this research gap by proposing a generally applicable Nutri-Score for software libraries.

Section 2 will discuss the current literature on Nutri-Scores and sustainability metrics. In section 3 the research methodology will be outlined. The results will be discussed in section 4. Finally the limitations, extensions and conclusions will be discussed in section 6, 7 and 8.

2 Related Work

This section will provide an overview of the current literature on Nutri-Score and software sustainability metrics.

2.1 Nutri-Score

In this section we look into the details of Nutri-Score and explain what it resembles.

Nutri-Score is a nutritional label that is designed to provide easy understanding to customers about the nutritional quality of food products. Nutri-Score is developed by coordination of several European countries and it ranges from A (green) to E (red), with A indicating the healthiest options and E indicating the least healthy [9].

2.1.1 Scoring System

The Nutri-Score system assigns scores to food products based on their nutritional content, considering both negative and positive factors. The scoring is determined by subtracting the total positive points from the total negative points, which are calculated per 100 grams of the product.

Negative points are assigned for nutrients that should be limited in the diet:

- Energy content: High energy density per 100g or 100ml
- Saturated Fat: High levels of saturated fatty acids.
- Total Sugar: High sugar content.
- Salt Level: High sodium content.

Positive points are awarded for nutrients that are beneficial:

- Fruits, Vegetables, Legumes, and Nuts: High content of these promotes a better score
- Fiber Content: High fiber levels contribute positively.
- Protein Content: High protein levels are also beneficial.

The final score ranges from -15 to +40, with lower scores indicating better nutritional quality. Products are classified into five categories: A (best) to E (worst), based on their score[15].

2.1.2 Different Categories and Reliability

The Nutri-Score system, while designed to provide a universal scoring method for various food products, faces challenges in its application across different categories. This can lead to inconsistencies and potential misinterpretations when comparing products from various categories.

For example, beverages, particularly those with a high sugar content, tend to score poorly due to their sugar levels. However, some sugary drinks can also contain significant amounts of fruit, which could make them healthier. For example, water will always score better than a sugary drink, even if the latter contains beneficial ingredients, such as fruit.

Furthermore, the Nutri-Score calculates scores based on 100g of a product, which can be misleading when comparing different types of food. For example, 100g of nuts is a reasonable serving size, but 100g of bread is less typical, as people usually consume more bread in a single serving. This discrepancy means that the Nutri-Score might not accurately reflect the nutritional impact of a typical serving size of different foods. This is why the Nutri-Score system assigns scores within individual categories only. This approach ensures fairer comparisons within a category, but also limits direct comparability across categories.

2.2 Software Sustainability Measurement

In this section, we explore relevant sustainable software literature, and analyze how the ideas contained within can be applied in developing a sustainability scoring system for open-source libraries.

In a recent review, Adadoga et al. [1] identified a range of practices, techniques and tools used for sustainable software engineering. The authors point out a number of practices which, when applied during the development and deployment of a software product, can be conducive to the product's sustainability:

During the design of a product, the choice of modular and reusable software architecture patterns (e.g. microservices) can improve sustainability by making future maintenance easier, and optimizing energy consumption with containerization. During development, algorithms and data structures can be optimized to reduce energy usage, and validated with energy-aware testing. During deployment, optimizing data center operations can lead to reduced energy usage and carbon emissions.

Bengtsson et al. [6] zoom in closer on the architecture design stage of software development, proposing an analysis method called architecture-level modifiability analysis (ALMA), with the goal of reducing the cost of evolving a system over time. This is related to the idea of reusable software architecture patterns in [1].

Work in this field also extends beyond technical aspects of sustainability. Nauman et al. [11], Becker et al. [4] and Noman et al. [12] recognize multiple domains or dimensions of sustainability. **Social**, i.e. impact on one's sense of belonging, perception and treatment, **individual**, i.e. impact on one's healthy, privacy and safety, **economic**, i.e. impact on value, supply chains and customer relationships, and **technical** and **environmental**, i.e. resource usage, maintainability and other aspects already mentioned in this section.

While some of this literature focuses on software sustainability through more abstract concepts like modifiability or supportability, Albertao et al. [2], [3] focus on more concrete measurements related to these concepts. For example, the authors of [3] suggest that modifiability be measured through e.g. *instability*, defined as $I = Ce/(Ca + Ce)$, with Ca the number of afferent couplings, and Ce the number of efferent couplings within a software package. Methods for labling sustainability have also been proposed for specific categories of software like machine learning models [8]. While these offer a quantitative perspective on software sustainability, value can be added by integrating multiple such metrics into one, novel, holistic score that can be

applied to any library.

Broadly speaking, the reviewed literature describes introducing sustainability considerations into the development and deployment process of (mostly commercial) software products and services. While this empowers organizations to take control of the sustainability of their products, it also means that it is mostly in the hands of the organizations themselves to scrutinize the sustainability of their own products, which could lead to conflicts of interest. Open-source libraries, however, can be developed through decentralized contributions from developers across the world. This means we need a novel ability to evaluate sustainability post-hoc. An evaluator that had no involvement in the development process should be able to assign a library a sustainability score by using it and analyzing its code.

At the same time, the score should be based on metrics which when measured post-hoc, represent as many domains of sustainability as possible. As evidenced by the literature described in this section, this includes not only technical sustainability and environmental sustainability, but also individual, social and economic sustainability.

Notably, such a post-hoc score could also enable some external scrutiny of commercial software developed by organizations. Some commercial products depend on open-source libraries, and developers from such organizations sometimes make contributions to open-source libraries to extend the functionality of their products which rely on those libraries [14].

With these requirements in mind, we describe the design of our scoring system in the following section.

3 Methodology

In this section, we explain how we adapt the concept of Nutri-Score to create the HowSUS score - a score of sustainability of software libraries. We select a range of metrics that intend to measure how well the given library adheres to various sustainability domains, and demonstrate a framework for computing a score using these measurements. We also emphasize that the specific metrics and calculation parameters we present are not inherent to the framework, but rather an example use case. For clarity, we highlight alternative metrics and use cases of our framework to emphasize its flexibility and generality.

3.1 Libraries and Categorization

The same way Nutri-Score (outlined in Section 2.1.2) does intra-category rankings, we design HowSUS in a similar way. This is because different categories of libraries are suited for different tasks - for example, a string parsing library might consume significantly less energy than a scientific computing library, but that does not necessarily mean it is more sustainable. Instead we aim to rank libraries by sustainability within their category, so that the user can select the most sustainable option for their use case.

To prove our concept, we opt for Python libraries, and categorize them as follows:

Table 1: Library Categories and Their Members

Category	**Libraries**
Matrix Multiplication	Sympy, NumPy, PyTorch, TensorFlow, Pandas
CSV Processing	Pandas, PyArrow, NumPy
Sorting	Pytorch, TensorFlow, NumPy, Pandas
Machine Learning	Scikit-Learn, TensorFlow, PyTorch, XGBoost, LightGBM
Network Libraries	requests, urllib3, aiohttp, HTTPX
String Processing	Built-in `re`, regex, pyparsing, NLTK, spaCy

3.2 Library Test Descriptions

We created tests to measure how libraries perform and how sustainable they are. The tests were based on the main tasks each type of library is designed for. For each type, we assessed key performance metrics such as response time, memory usage, and energy consumption (estimated from execution time). To make sure the averages of the test are reliable, each test has been run 30 times.
Matrix Multiplication: Two randomly generated 1000x1000 matrices had to be multiplied by the library function.

CSV Processing: The CSV readers had to read a 100.000 line CSV file into memory and write it back to disk in another location.
Sorting: This task involved sorting a dataset of 10 million random integers ranging from 1 to 100 million.
Machine Learning: We trained a logistic regression model on a dataset comprising 10,000 samples and 20 features, 15 of which were informative and 5 redundant. Additionally, we standardized the number of epochs/iterations across all libraries to ensure a fair comparison
Network Libraries: To assess network libraries, we performed download tests involving a 100MB file from various endpoints. Both synchronous and asynchronous tasks were evaluated by measuring total download time and average download speed. This approach provides insights into network throughput and overall efficiency.
String Processing: For string processing libraries, we evaluated performance on tasks such as tokenization and pattern matching. The tests used different text processing approaches using Python's built-in and other modules. A large text sample was replicated multiple times to ensure statistical significance in measuring execution time and memory usage.

3.3 General Performance Metrics

This section describes the metrics we selected to evaluate technical and environmental sustainability.

Memory Usage, measured in megabytes (MB), is an important metric, as lower memory consumption signifies more efficient resource utilization, reducing strain on hardware and minimizing power usage. **Energy Consumption**, measured in joules (J), during operation is another key measure, where reduced energy usage directly translates to a smaller carbon footprint and supports sustainability. **Execution Time**, measured in seconds, is crucial as faster execution reduces energy use, which is especially important when processing large amounts of data quickly.

These metrics are chosen because they directly indicate the resource efficiency of libraries in executing their domain-specific tasks. For instance, in string manipulation, shorter execution times not only speed up text parsing but also reduce CPU loads, enhancing sustainability in large-scale applications.

We implemented these metrics to compute a Nutri-Score, offering a qualitative grade (ranging from "E" to "A"). This score integrates weighted averages to provide an assessment of a library's performance and long-term sustainability.

3.4 General Code Metrics

This section describes the metrics we selected to evaluate other dimensions of sustainability, i.e. economic, social and individual.

Instability and Abstractness Taken directly from Albertao et al. [3], instability is defined as:

$$I = \frac{Ce}{Ce + Ca}$$

with Ca the number of afferent couplings and Ce the number of efferent couplings. Abstractness is defined as:

$$A = \frac{\text{Number of abstract classes}}{\text{Total number of classes}}$$

We chose to include these because both quantify the maintainability and reusability of a package, representing an aspect of sustainable design and engineering in the total score. Lower instability might make maintenance easier because changes in one part of the package are less likely to break other parts. Higher abstractness might make adding features easier since abstract classes serve as valuable general building blocks.

These were measured using a custom Python program, given as part of the replication package (see Section 9).

Number of resolved issues in the past d days These metrics are measured from the Github repositorities of the libraries (see Section 9). We measure the number of approved merge requests in the past d days, and the number of closed issues in the past d days. For our sample calculation, we took the measurements with $d = 30$, but we intentionally parametrize this for adaptation to other use cases. These metrics were included to measure participation and sense of community, representing the social dimension. Higher numbers of resolved issues

indicate a healthier, more active open-source community, indicating that the library is received well and has a positive impact on the community.

Number of known security vulnerabilities This metric was measured manually, by counting the number of low-severity, medium-severity, high-severity and critical vulnerabilities reported for the given pip packge on `security.snyk.io`. Let n_l, n_m, n_h and n_c be the counts of low, medium, high and critical vulnerabilities respectively. We calculate the overall vulnerability score as

$$w_l n_l + w_m n_m + w_h n_h + n_c$$

with $\sum w = 1$. For our example calculations, we chose $w_l = 0.2$, $w_m = 0.25$ and $w_h = 0.5$, i.e. 2 high-severity vulnerabilities count for one critical and so on. Again, the formula itself is intentionally parametrized.

We chose to include this metric to represent individual and economic sustainability. Security vulnerabilities in commonly used libraries could invite attacks on software that uses them, potentially affecting the well-being of the organization responsible for the software, and disrupting the lives of people involved.

3.5 Library-specific Metrics

It is also possible that the sustainability of a category of libraries can be measured in some additional way. Therefore, our method allows for the possibility to include some library-specific metrics to measure and include in the final score.

Firstly, for the category of CSV file readers and writers we also included disk accesses. This means measuring read and write operations on the disk of the task. This was included as more operations means more disk wear. If a library is thus not efficiently implemented in this respect and performs much more read and write operations than another, it will get a penalty for this in the final score.

Secondly, for the machine learning task, we also measured the average accuracy across multiple runs for each library. Accuracy is a crucial metric in evaluating machine learning models, as it reflects their performance and indicates how much additional training may be required. A model that completes the task in a shorter amount of time does not necessarily perform better if its accuracy is lower. Lower accuracy can imply that the model needs further training, which in turn increases both the overall training time and energy consumption.

Metric	**Dimension**
Energy consumption	technical, environmental
Memory usage	technical, environmental
Execution time	technical, environmental
Disk accesses	technical, environmental
Accuracy	technical, environmental
Instability	technical, economic
Abstractness	technical, economic
Resolved issues	technical, social
Security vulnerabilities	technical, economic, individual

Table 2: An overview of how the metrics represent the domains of sustainability.

3.6 Score Calculation

Firstly, we conducted an analysis of the calculation of Nutri-Scores for food [13]. It was found that a product could obtain certain plus and minus points for a score in a specific category. These scores were then added up and the final point score would then lie between -15 and 40. Finally, a label is assigned based on some pre-set thresholds.

A big difference between our use-case and that of the Nutri-Score is the amount of products within a category. If you go to an average supermarket and look in the aisle containing for example cookies, there are many choices, typically over 100. This is not the case with software libraries, if there exist 10 libraries for a specific task, then this is considered many. The smaller sample size is a motivation for some of our design choices. Like in the original Nutri-Score the calculation allows for the inclusion of library-specific metrics such as accuracy in machine learning tasks and disk accesses in CSV readers.

We decided that the HowSUS score would lie between -1 and +2, this lower value is due to the lower amount of scores compared to the original calculation. The calculations are done per category of libraries, possibly taking into account library-specific metrics. The average per metric per library (such as average memory usage for numpy) out of 30 experiments is calculated. This is done for every library in a category, per metric (such as average memory usage) the results are scaled such that the

lowest is -1 and the highest is +2, or vice versa depending on whether lower is better. To do so, the minimum and maximum average score for that metric are identified. The scaling is done using the formula below for every average score x:

$$score = \frac{3 * (x - minimumscore)}{(maximumscore - minimumscore)} - 1$$

The scaled metrics are then added up for each library and divided by the sum of the weighting of every category to obtain a final score. This calculation is given in the formula below, with the scores being per library and the weights remaining constant:

$$finalscore = \frac{\sum (score * weight)}{\sum (weight)}$$

The weights in our calculation are uniform (e.g. 1 for all metrics), as this study should mostly be seen as a proof of concept. The interval between -1 and +2 was divided into 5 equal intervals corresponding to label 'E' to 'A'. The interval in which the final score falls in is also the label the library gets assigned.

4 Results

Table 3: HowSUS scores of Libraries by Category

Category	Library	HowSUS score
CSV Processing	PyArrow	A
	Pandas	C
	NumPy	E
Matrix Multiplication	PyTorch	B
	NumPy	C
	TensorFlow	C
	Pandas	C
	Sympy	D
Sorting	Pytorch	B
	Pandas	C
	Numpy	C
	TensorFlow	E
Machine Learning	LightGBM	B
	Pytorch	B
	XGBoost	B
	Scikit-Learn	C
	TensorFlow	D
Network Libraries	aiohttp	A
	httpx-async	E
	httpx-sync	E
	urllib3	B
String Processing	NLTK tokenization	A
	pyparsing	A
	re (built-in)	A
	regex	A
	spaCy tokenization	E

The results are presented in Table 3, where each category is displayed along with the corresponding library and its Nutri-Score label. Some categories include libraries with varying labels, while others consist of libraries mostly with the same label.

5 Discussion

The resulting labels highlight differences between libraries within a given category. However, comparing identical labels across different categories may not indicate the same level of underlying metrics (energy consumption, time spent etc.). However, when evaluated within their own category the labels reflect differences in sustainability levels. The labels created can help inform users about the sustainability of different libraries, allowing them to make more intentional choices rather than simply selecting the most commonly used or easily accessible option.

Although the differences in sustainability between libraries may be small, we believe that consistently choosing the library with a better Nutri-

Score, especially on a larger scale, can have a positive impact. It also signals to major companies that users value sustainability. As awareness grows, investment in more sustainable programming tools is likely to increase, leading to the development of more sustainable products and technologies.

During our calculations, we gained a better understanding of why it is difficult to compare libraries from different categories - much like comparing foods between categories is challenging. This is because the tasks performed by different libraries can vary significantly, requiring different levels of computational power and specialised metrics such as disk accesses or accuracy for certain categories.

This introduced an additional layer of complexity to our formula. While incorporating new metrics for specific tasks can be useful, it is important that they do not compromise the overarching principle of the Nutri-Score, providing a comparable measure across all categories. In other words, a newly introduced metric should not independently define a category, as this would undermine the broader goal of ensuring a unified scoring system.

Overall, this process helped us gain a better understanding of how sustainability labels work for Python libraries and why direct comparisons between different task categories are often inaccurate, similar to the classic food-based Nutri-Score. It also highlighted the difficulties of developing a universal labelling system that ensures comparability across all categories.

6 Limitations

An important point to evaluate is how well our framework is able to represent each domain of sustainability. It is clear from Table 2 that the technical and environmental domains are the most represented. However, this is also not too surprising. Even though these should be considered only examples of what one could measure, we found it considerably easier to measure technical and environmental metrics.

Many technical metrics exist on a level of abstraction that makes them very quantitative in nature and more readily available. On top of that, metrics that measure technical sustainability tend to correlate with environmental sustainability. For example, energy consumption and memory usage of course reflect resource consumption and environmental sustainability, but they also affect scalability which is a matter of technical sustainability.

While reviewing literature, we did encounter some quantitative metrics that reflect non-technical and non-environmental sustainability, but they did not align well with our perspective of independently evaluating open-source libraries. For example, Albertao et al. [3] suggested measuring supportability by dividing the number of user questions that required assistance by the number of minutes the software was used in a given session. This could reflect individual and economic sustainability by reflecting how active the support staff needs to be, but open-source libraries might not have dedicated support staff, and data about session duration could be very difficult to obtain.

Additionally, effectively measuring social sustainability seems to be even more challenging, because it exists on a higher-order, abstract level. Measuring how much a library creates a sense of participation in a safe, trustworthy community might involve sentiment analysis on social media, or analysis of news articles about software that uses the given library. The catch is that both of these examples could be research problems of their own.

A technical level, a limitation of the way our score is calculated is that we take the average of metrics, which makes it sensitive to outliers. In Table 3, in the string processing category, there are 4 libraries with an A score and one library with an E score. There might be meaningful variance between the 4 libraries with the A score, but because spaCy drastically stands out with its runtime, the range stretches so far that the other 4 end up in the same bucket.

7 Extending the Framework

With the proof of concept laid out in this study, the framework can be extended by adding additional metrics that span broader sustainability categories. For instance, while our current design already includes technical, environmental and some community based metrics. In the future, social and economic metrics, like community involvement, engagement of users, economic resources, could provide a broader picture of a library's sustainability,

although some of these other metrics would involve more complex data collection and analysis.

For instance, some of the metrics called for by Albertao et al.[3] include more sustainable attributes such as supportability and maintainability. While these metrics show valuable options, they are harder to measure, because they rely on qualitative data or subjective evaluations.

Regarding the scoring system's sensitivity to outliers, one recommendation is to apply a logarithmic transformation to the raw metric values before scaling. This method reduces the disproportionate influence of extreme values. For example, in the string processing category, some libraries have exceptionally short processing times that can misrepresent the overall score. A log transformation would provide a more uniform and accurate comparison of performance across metrics.

While the current framework focuses on Python libraries hosted on GitHub, it could be extended to include libraries from other programming languages and platforms. Similarly, if a library spans multiple functional categories, it should be assessed separately. The final sustainability score should then be calculated as a total or weighted average of the individual scores from each category. This approach ensures a fair and transparent assessment across categories, avoiding any disadvantage to libraries that span multiple categories due to reliance on a generalized score.

Therefore, while the framework provides a solid foundation, these additions would lead to a more comprehensive sustainability assessment that stays relevant to software sustainability and evolves in step with changing trends in the field over time.

8 Conclusion

The purpose of this report was to present a framework for a sustainability scoring system for software libraries. To do so, the inspiration for our idea, the Nutri-Score, was analysed and used as a basis for our own proposal. The proposed framework, which was named 'HowSUS', was subsequently tested by applying it to python libraries for different tasks.

In line with the Nutri-Score our proposed method is aimed at intra-category comparison of software libraries. We have identified metrics such as memory consumption, execution time and energy consumption to be general measures of performance regardless of the category of library. Additionally, we identified general code measures, including instability, abstractness and the number of recently resolved issues and security vulnerabilities. Our framework allows for the inclusion of category specific metrics, such as amount of disk accesses for IO applications and accuracy for machine learning libraries.

After identification of the metrics relevant for a sustainability score of the library we applied our framework to different categories of python libraries. A final score was calculated for each library based on the metrics obtained from the measurements. Based on the final score a HowSUS label is assigned ranging from 'E' for the worst to 'A' for the best sustainability score.

In this report we have proposed and implemented a generally applicable framework HowSUS for scoring software libraries on sustainability. We have demonstrated the framework using python libraries and emphasised that expansion to any type of software library along with additional metrics can be made possible. We recommend wide adoption of our framework to ensure users make the most sustainable choices possible.

9 Replication Package

The replication package can be found at `https://github.com/the-chef0/howsus`.

References

[1] Akoh Atadoga et al. "Tools, techniques, and trends in sustainable software engineering: A critical review of current practices and future directions". In: *World Journal of Advanced Engineering Technology and Sciences* 11.1 (Feb. 28, 2024), pp. 231–239. ISSN: 25828266. DOI: 10.30574/wjaets.2024.11.1.0051. URL: https://wjaets.com/content/tools-techniques-and-trends-sustainable-software-engineering-critical-review-current (visited on 03/25/2025).

[2] Felipe Albertao. *Sustainable Software Engineering — PDF — Usability — Sustainability*. URL: https://www.scribd.com/document/5507536/Sustainable-Software-Engineering (visited on 03/28/2025).

[3] Felipe Albertao et al. "Measuring the Sustainability Performance of Software Projects". In: *2010 IEEE 7th International Conference on E-Business Engineering*. 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE). Shanghai, China: IEEE, Nov. 2010, pp. 369–373. ISBN: 978-1-4244-8386-0. DOI: 10.1109/ICEBE.2010.26. URL: http://ieeexplore.ieee.org/document/5704342/ (visited on 03/28/2025).

[4] Christoph Becker et al. "Requirements: The Key to Sustainability". In: *IEEE Software* 33.1 (Jan. 2016), pp. 56–65. ISSN: 0740-7459. DOI: 10.1109/MS.2015.158. URL: http://ieeexplore.ieee.org/document/7325195/ (visited on 03/28/2025).

[5] Lotfi Belkhir and Ahmed Elmeligi. "Assessing ICT global emissions footprint Trends to 2040 recommendations". In: *Journal of Cleaner Production* 177 (2018), pp. 448–463. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2017.12.239. URL: https://www.sciencedirect.com/science/article/pii/S095965261733233X.

[6] PerOlof Bengtsson et al. "Architecture-level modifiability analysis (ALMA)". In: *Journal of Systems and Software* 69.1 (Jan. 2004), pp. 129–147. ISSN: 01641212. DOI: 10.1016/S0164-1212(03)00080-3. URL: https://linkinghub.elsevier.com/retrieve/pii/S0164121203000803 (visited on 03/28/2025).

[7] Rébecca Deneckère and Gregoria Rubio. "EcoSoft: Proposition of an Eco-Label for Software Sustainability". In: *Advanced Information Systems Engineering Workshops*. Ed. by Sophie Dupuy-Chessa and Henderik A. Proper. Cham: Springer International Publishing, 2020, pp. 121–132. ISBN: 978-3-030-49165-9.

[8] Pau Duran et al. *GAISSALabel: A tool for energy labeling of ML models*. 2024. arXiv: 2401.17150 [cs.SE]. URL: https://arxiv.org/abs/2401.17150.

[9] FoodChain ID. *Food Labeling Regulations and Nutri-Score*. en-US. Feb. 2025. URL: https://www.foodchainid.com/resources/the-nutri-score-towards-a-unified-food-labeling-system-in-eu/.

[10] Sasha Luccioni, Yacine Jernite, and Emma Strubell. "Power Hungry Processing: Watts Driving the Cost of AI Deployment?" In: *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '24. ACM, June 2024, pp. 85–99. DOI: 10.1145/3630106.3658542. URL: http://dx.doi.org/10.1145/3630106.3658542.

[11] Stefan Naumann et al. "The GREENSOFT Model: A reference model for green and sustainable software and its engineering". In: *Sustainable Computing: Informatics and Systems* 1.4 (Dec. 2011), pp. 294–304. ISSN: 22105379. DOI: 10.1016/j.suscom.2011.06.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S2210537911000473 (visited on 03/28/2025).

[12] Hira Noman et al. "Towards sustainable software systems: A software sustainability analysis framework". In: *Information and Software Technology* 169 (2024), p. 107411. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2024.107411. URL: https://www.sciencedirect.com/science/article/pii/S0950584924000168.

[13] Scientific Committee of the Nutri-Score. *Update of the Nutri-Score algorithm*. Publisher Name Scientific Committee of the Nutri-Score, June 2022.

[14] *PyTorch — NVIDIA NGC*. URL: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch (visited on 04/02/2025).

[15] RIVM. *Nutri-Score*. https://www.rivm.nl/voedsel-en-voeding/nutri-score. [Accessed 28-03-2025]. Jan. 2025.