Project Report: IBM x ESILV Hackathon

Team 1: Héloïse Roméo, Natalia Gérard, Alexis Denneulin, Valentin Rech

Paul Ranc, Valentin Templé

Table des matières

Technical approach

Our project is based on the idea that every interaction with a Large Language Model (LLM) consumes computational resources, which directly translate into energy usage and, ultimately, CO_2 emissions. To estimate this impact in a transparent and understandable way, we designed a workflow composed of four main components:

1. Query Processing

For the prompt preprocessing stage, the key idea is to only keep information that is available with a simple user prompt. Since our goal is to estimate the cost of an LLM request in advance, we cannot rely on features that depend on the generated answer or on the runtime behaviour itself. Originally, each measurement file was named using a combination of the model and the hardware on which it was executed, for example: llama3_8b_laptop1 or gemma_2b_server. To make these factors usable as structured input, we separated them into two distinct categorical variables: the model's name and the hardware type. We then encoded these with one-hot vectors (e.g., model_llama3_8b, model_codellama_70b, hardware_laptop2, hardware_server, etc.), where exactly one value in each group is set to 1 and the others to 0. This explicit encoding makes it easy for the cost estimator model to learn differences in computational cost across model families and machine configurations.

Next, we focus on the textual features extracted directly from the user's prompt. These include basic descriptive counts, such as word_count and sentence_count, but also more fine-grained linguistic indicators like noun_count, verb_count, adj_count, and adverb_count, obtained through part-of-speech tagging. We also track letter_count as a measure of raw character length, and question_marks as a simple proxy for interrogative or analytical prompt style. These features are chosen because they correlate with the number of tokens the model will likely have to process, which in turn affects computation time and cost.

In total, we started with 89 features, many of which were tied to post-execution observations (e.g., actual latency, generated token count, memory usage, response length). Since these values are unknown at prediction time, we exclude them entirely. We only retain the features we can guarantee to know at the moment the user submits their request: the characteristics of the prompt itself, and the selected model and hardware.

This ensures the cost estimation remains realistic and applicable in real-world usage, where we must make a prediction before the model actually runs.

2. Consumption prediction

The model was trained on the Watsonx platform, which allowed us to combine heterogeneous feature types and leverage scalable MLOps capabilities for

experimentation and monitoring. Our feature set included detailed linguistic metrics extracted from the input prompt—such as word count, sentence count, average word length, unique word count, punctuation density, and the proportion of long or monosyllabic words—to characterize the textual complexity of each request.

Watsonx also enabled us to integrate task descriptors (e.g., alpaca-style prompting or code-feedback scenarios) as structured categorical features. In addition, we included model-related indicators reflecting which foundation model was used during inference (such as LLaMA, Gemma, or CodeLlama in various parameter sizes), as well as hardware-specific features describing the compute environment (different laptops, a workstation, or a server). Thanks to Watsonx's feature engineering and data lineage capabilities, these inputs were cleanly tracked, versioned, and fed into the training pipeline to predict power consumption in watts.

After obtaining the energy predictions, we applied a conversion based on standard energy-to-CO_2 emission factors to estimate the carbon footprint associated with each inference. This allowed us not only to model energy usage, but also to quantify and compare the environmental impact of different prompts, models, and hardware configurations, providing actionable insights for optimizing sustainability in AI workloads.

Regarding model performance, we obtained an R^2 score of 0.56 and an RMSE of approximately 7×10^−5. While these values indicate that the model does not capture all the variance in energy consumption, they are reasonable given that we rely solely on high-level, easily extractable information such as prompt structure, task type, model identity, and hardware category. Since we do not include fine-grained runtime metrics—like token-by-token inference traces, GPU utilization curves, or temperature readings—the model's ability to explain more than half of the variance is acceptable and demonstrates that even basic descriptive features can provide meaningful insight into energy and CO_2 consumption patterns.

3. CO_2 Emissions Conversion

Contextualizing Energy Consumption Through Tangible Comparisons

One of the major challenges in raising awareness about the environmental impact of LLM queries lies in the abstraction of energy data. Indeed, expressing consumption in kilowatt-hours (kWh) remains difficult for the average user to grasp. To address this issue, we chose to translate these metrics into concrete everyday examples.

Our interface presents relatable equivalence: the consumption of an LED bulb over a given duration, or the energy required to charge a smartphone. For reference, charging a smartphone completely (from 0% to 100%) consumes approximately 5 Wh. These

comparisons allow users to realize that with each query sent to a language model, they are consuming something tangible and measurable, making the environmental impact immediately perceptible.

From Energy Consumption to Carbon Emissions

Beyond simple electrical consumption, it is essential to consider that this energy must be produced, and that this production generates variable CO_2 emissions depending on the source used. The energy mix, whether it involves wind, nuclear, solar, or fossil energy, significantly influences the final carbon footprint of an LLM query.

This is why we integrated a conversion into CO_2e equivalent (carbon dioxide equivalent), expressed in kilograms. To make this metric even more concrete, we also provide an equivalence in terms of tree absorption: how long would it take for a tree to absorb the CO_2 emitted by this query?

For this calculation, we used the following baseline: a mature tree absorbs approximately 20 to 25 kg of CO_2 per year. Taking 20 kg of CO_2/year as our reference, this corresponds to approximately 20,000 g / 365 days ≈ 55 g of CO_2 per day. This representation allows us to materialize the ecological impact in a temporal and natural dimension, providing users with an intuitive understanding of how their queries translate into real-world environmental consequences.

Beyond Inference: The Training Footprint

While our solution primarily focuses on the inference phase, that is, the direct use of the model by the user, it should be noted that the environmental impact of LLMs is not limited to this dimension alone. The initial training of models represents an extremely energy-intensive phase, sometimes several orders of magnitude greater than inference itself.

Although our project focuses on estimating consumption per query, we acknowledge the importance of this systemic dimension. A complete approach to the carbon footprint of LLMs should ideally integrate this training phase, distributed across the model's entire lifecycle. This more holistic perspective would enable a truly comprehensive assessment of the environmental impact of generative artificial intelligence.

Aware that these figures need to be nuanced and contextualized, we made the choice to systematically cite the scientific sources and methodologies that enabled us to establish these conversions and equivalences. This transparency ensures the credibility of our approach and allows users to deepen their understanding of the underlying calculations.

4. Impact Visualization

To ensure that the cost estimation system is not only technically robust but also usable and educational, we designed a comprehensive visual web application that allows users to explore the dataset, understand model behaviors, and directly estimate the

environmental and computational cost of their own prompts. The interface is structured to be intuitive even for non-technical users, while still offering depth for power users, researchers, and enterprise teams.

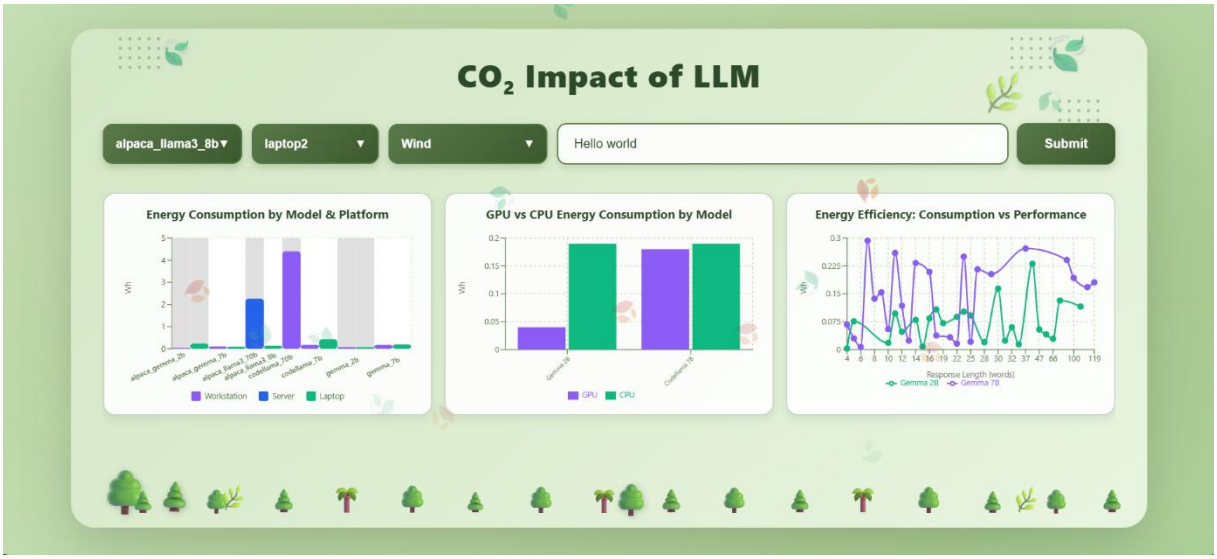
Global Statistics & Insights Dashboard

The first part of the web application is a data exploration dashboard presenting the global characteristics of our dataset and the performance profiles of different LLMs across hardware configurations. The objective is to provide transparency, interpretability, and context.

This dashboard includes multiple interactive visualization components:

- Model Performance Heatmaps: showing how compute time and energy usage scale with model size.
- Cost Distribution Graphs: comparing inference cost per token across models.
- Hardware Efficiency Charts: visualizing energy consumption differences between laptops, workstations, and servers.
- Token Length vs. Latency Scatter Plots: illustrating the relationship between prompt complexity and compute requirements.
- Energy Mix Comparison View: demonstrating how emissions vary depending on the electricity source (renewable-heavy grid vs. standard grid vs. datacenter power contract).

This data transparency layer is crucial to encourage informed decision making, model selection awareness, and sustainable AI practices.



Prompt Evaluation & Cost Feedback Interface

The second major section of the web app is a prompt input interface, designed to feel familiar—similar to interacting with any modern LLM or chatbot. The goal is simplicity: the user should only need to write their prompt and make a few selections.

The user provides:

1. The prompt text (as they would normally send to an LLM)
2. The model choice (e.g., Gemma 2B, LLaMA 3 8B, CodeLLaMA 70B, etc.)
3. The hardware environment (e.g., laptop, workstation, server)
4. The energy source profile (e.g., default grid, renewable mix, datacenter low-carbon plan)

Once submitted, instead of waiting for a generated response, the system immediately performs a cost estimation and displays:

- Estimated CO_2 emissions (in grams)
- Estimated compute energy required (in joules or watt-hours)
- Relative monetary cost equivalent

By presenting emissions in relatable units, the web app bridges the gap between abstract environmental data and real-world meaning.
This encourages users to adopt more resource-efficient prompting strategies and to choose smaller models when appropriate.

Challenges faced

1. Difficulty getting started with Watsonx*

We found the Watsonx platform quite challenging to use at first. It wasn't very intuitive: for example, understanding how to structure projects and navigate the interface took some time.
We also struggled to add multiple team members to the same project, which slowed down collaboration during the early stages.

2. Uncertainty about how to train the model

Another major challenge was deciding how to train our model.
We had access to the inputs, outputs, the model itself, and the CO_2 consumption data, and our goal was to build a platform where a user could ask a question and immediately see how much energy that query would consume.
However, we couldn't train the model on the output side, since our interface wouldn't actually generate the outputs. On the other hand, training only on the inputs didn't accurately reflect how much computation or CO_2 the model would produce.
As a result, our training data didn't perfectly capture the relationship between inputs and actual energy usage, which affected the model's accuracy.

Future Development

There are several directions to enhance the accuracy, usability, and strategic value of the cost estimation tool:

1. Real-Time Hardware Energy Monitoring

Currently, the system relies on manual user input hardware performance values. A future improvement would be to integrate real-time monitoring of machine-level energy consumption (e.g., GPU wattage, CPU power draw, fan speed, thermal load). This would allow the cost estimation to reflect not only computational effort but also energy impact, enabling more precise carbon footprint estimation and sustainability-aware decision making.

2. Comparative Model Benchmarking

We plan to expand the tool to systematically compare different models side-by-side for a given prompt. Instead of only estimating the cost of selected model, the system would present trade-offs across several models (e.g., accuracy vs. speed vs. cost). This would help users quickly decide whether a smaller model is "good enough" for their task or whether a larger one is worth the additional compute.

3. Enterprise Sustainability Reporting

For organizations, the tool could generate monthly or quarterly reports that summarize compute usage, costs, per-team energy footprint, and sustainability KPIs. These reports would help companies track their environmental impact and optimize resource allocation in a transparent, measurable way.

4. Support for More Models

We intend to expand compatibility to cover a broader range of open-source and proprietary LLMs, including future model releases. This will require continuously updating the one-hot model encodings and ensuring we maintain benchmark consistency across hardware tiers.

5. More Accurate Prompt Understanding via Embeddings

Future versions could integrate semantic embeddings of the prompt, capturing nuance beyond word counts and POS features. Combined with richer feature engineering and more training time (beyond the initial constraint of ~10–15 minutes), this would significantly improve cost prediction accuracy, especially for complex or domain-specific prompts.

Sources:

- U.S. Department of Energy – Energy Saver (link)
- Solar Technologies – “How much electricity does a light bulb use?” (link)
- IEEE Spectrum (link)
- Next Business Energy (link)