Measuring and Modeling CO_2 Emissions in Machine Learning Processes

Ivo Hrib
Jožef Stefan Institute
Ljubljana, Slovenia
ivo.hrib@gmail.com

Oleksandra Topal
Jožef Stefan Institute
Ljubljana, Slovenia
Oleksandra.Topal@ijs.si

Jan Šturm
Jožef Stefan Institute
Ljubljana, Slovenia
jan.sturm@ijs.si

Maja Škrjanc
Jožef Stefan Institute
Ljubljana, Slovenia
maja.skrjanc@ijs.si

Abstract

With the rapid expansion of the computing industry, efficient energy utilization and reduction of CO_2 emissions are critically important. This research develops analytical tools to predict CO_2 emissions from various machine learning processes. We present a novel methodology for data acquisition and analysis of CO_2 emissions during model training and testing. Our results demonstrate the environmental impact of different algorithms and provide insights into optimizing energy consumption in artificial intelligence applications.

Keywords

CO_2 Emissions, Machine Learning, Energy Consumption, Environmental Impact, AI Model Optimization, Green AI, Sustainable Computing, Carbon Footprint

1 Introduction

The global computing industry significantly contributes to CO_2 emissions, with data centers accounting for 2.5 to 3.7 percent of global greenhouse gas emissions [1]. These emissions exceed those of the aviation industry due to continuous operations and heavy reliance on fossil fuels [11]. Given the growing demand for artificial intelligence (AI) applications, there is an urgent need for CO_2-conscious solutions.

This research aims to develop tools for predicting CO_2 emissions associated with machine learning processes, thus enabling the reduction of the environmental impact of AI models. In collaboration with Eviden (Spain) and under the FAME EU project, we have developed a CO_2 emissions analysis system using tools like CodeCarbon [2] and eco2AI [3].

1.1 Research Goals

The primary goal of this research is to develop a service that predicts CO_2 emissions and power consumption of different machine learning models during both training and evaluation phases with emphasis on hyperparameter dependency. The CO_2 emissions are measured in kilograms per second ($\frac{kg}{s}$) , while the power consumption is measured in kilowatt-hours (kWh).

While existing services, such as CodeCarbon [2] or eco2AI [3], provide real-time measurement of emissions, they do not offer insights into a model's emissions before its construction or use. The service we aim to provide addresses this gap by offering an estimation of emissions and power consumption for different models before they are selected for specific use cases. This forward-looking approach allows for more informed decisions when choosing models, potentially reducing their environmental footprint.

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2024.sikdd.23

2 Related Work

The environmental impact of machine learning models has been a growing concern in recent years. Several studies have focused on quantifying and reducing the carbon footprint of artificial intelligence (AI) processes. For instance, [12] highlighted the energy consumption of training large neural models and suggested methods for minimizing emissions. Similarly, tools like CodeCarbon [2] and eco2AI [3] have emerged to measure real-time CO_2 emissions from computational tasks. However, these tools often lack predictive capabilities for assessing emissions before model selection, as pointed out by. Our work builds on these existing methodologies, concretely on the work of eco2AI[3], by providing a forward-looking approach that estimates emissions during the model selection phase, thus complementing real-time monitoring tools. This is achieved through heavy dependency on eco2AI[3] measuring systems for data collection, later used for modeling based on the collected data and registered hyperparameters.

2.1 Research Gap and Contribution

Despite the growing availability of tools like CodeCarbon [2] and eco2AI [3], a significant gap remains in the preemptive evaluation of environmental impact during the machine learning (ML) model selection phase. The mentioned tools are valuable for post hoc analyses but do not assist ML practitioners in making **informed decisions upfront**—before model development—on the environmental footprint of different model architectures or hyperparameters.

This gap is crucial, as the model selection phase often involves trial-and-error across multiple models and configurations, potentially leading to unnecessary resource consumption. Without predictive capabilities, practitioners have limited insight into which models will have the lowest environmental impact before engaging in resource-intensive training.

Our research aims to fill this gap by introducing a **predictive service** that estimates the environmental footprint of different ML models before they are trained or used. This service leverages the data collected from existing tools like eco2AI [3], incorporating key features such as hyperparameters, and model architecturre into predictive models. By doing so, we enable developers to make **more sustainable choices** at the model selection stage, reducing carbon emissions from the start of the ML lifecycle.

The table 1 below presents a feature matrix comparing our proposed service with current tools, showing how our approach addresses unmet needs:

3 Methodology

Due to the lack of suitable data on CO_2 emissions of machine learning models, we began by developing an infrastructure for data collection. This infrastructure is composed of the following steps:

- **Dataset Generation:** Creating synthetic datasets using random data generation methods.
- **Data Preprocessing:** Cleaning and preparing the data for analysis.
- **CO_2 Emission Measurement:** Recording CO_2 emissions during both training and testing phases using different machine learning algorithms.
- **Feature Extraction:** Extracting relevant features such as project ID, experiment details, epoch duration, power consumption, and hardware configurations.
- **Adding Hyperparameters to Final Dataset:** Documenting hyperparameters used in each experiment to assess their impact on emissions.
- **Containerization:** Utilizing Docker for containerization to ensure reproducibility and scalability of the experiments.
- **Data Storage:** Storing all datasets, features, and emission records systematically in a database for further analysis.
- **Modeling:** Developing and training machine learning models to predict CO_2 emissions and power consumption.

The software implementation uses Python, with dependencies including pandas [7], scikit-learn [10], matplotlib [5], eco2AI [3], TensorFlow [**abadi2016tensorflow**], Keras [**chollet2015keras**], and Docker for containerization [**merkel2014docker**].

3.1 Dataset Generation

In this step, we created a synthetic dataset by generating random data points using tools like `sklearn.datasets.make_regression` or `make_classification`. The primary objective here is not to reflect real-world data scenarios but to produce a controlled environment where the focus is on measuring CO_2 emissions and power consumption during model training and evaluation. Datasets generated vary in size from ranges of 250 to 15000 samples and 5 to 2000 features. In classification cases additionally the number of classes ranges from 2 to 50. These parameter ranges were selected to mitigate the risk of computational overload, ensuring that the experiments remain feasible within the available computational resources while maintaining the integrity of the analysis.

3.2 Data Preprocessing

Before analysis, the dataset must be cleaned and prepared. This includes handling missing values, normalizing or standardizing data, encoding categorical variables, and splitting the data into training and testing sets. Proper preprocessing ensures that the data is in the optimal format for the models to learn from and minimizes biases that may affect model performance and emission measurements.

3.3 CO_2 Emission Measurement

We measure CO_2 emissions produced during both the training and testing phases of the machine learning models. This involves using tools like eco2AI [3] to track energy consumption and convert it into equivalent CO_2 emissions. The measurements are taken for various models, such as Decision Trees, Random Forests, Logistic Regression, and Neural Networks, to assess their environmental impact under different computational loads.

3.4 Feature Extraction

To gain deeper insights, we extract various features that could impact CO_2 emissions and energy consumption. These features include project identifiers, detailed descriptions of each experiment, the duration of each training epoch, power consumption metrics, hardware configurations (such as the type of CPU/GPU used), and hyperparameters. The project identifiers refer to unique alphanumeric codes assigned to each machine learning experiment upon execution. These identifiers help differentiate between various model configurations and experimental setups. They are generated and stored automatically by our system during the dataset generation process to ensure traceability and reproducibility of the experiments.

3.5 Adding Hyperparameters to Final Dataset

We document the hyperparameters used in each machine learning experiment, such as learning rates, batch sizes, and the number of layers in neural networks. This allows us to evaluate how these hyperparameters influence CO_2 emissions and energy consumption.

3.6 Containerization

To ensure reproducibility and scalability of our experiments, we employ Docker for containerization. This approach encapsulates the code, dependencies, and environment settings, allowing the experiments to be easily replicated and deployed across different platforms.

3.7 Data Storage

All datasets, extracted features, hyperparameter configurations, and CO_2 emission records are systematically stored in a database. This central repository facilitates efficient querying, retrieval, and analysis of data to support ongoing and future research.

3.8 Modeling

In this step, we develop and train machine learning models to predict CO_2 emissions and power consumption based on various features, such as the type of algorithm used, hardware configuration, and model parameters. This modeling allows us to estimate emissions for different machine learning workflows before their actual deployment. The models help identify the most efficient algorithms and configurations, thus guiding the selection of environmentally friendly AI solutions.

The general pipeline for the previously mentioned steps can be seen below (see Figure 1).

A more thorough view of the workings of this can be seen as shown below for running a single measurement (see Figure 2).

4 Model Architecture

In this section, we explain the architecture of the model used for predicting CO_2 emissions and power consumption based on

Tool/Technology	Platform Compatibility	Model Coverage	Metric Granularity	Carbon Metrics	Energy Metrics	Additional Features	Real-time measurement	Forward-looking Prediction
CodeCarbon	Cloud, On-Premise	All ML models	Per training session	CO_2 emissions (kg)	Energy consumption (kWh)	Dashboard Visualization	Yes	No
eco2AI	Cloud, On-Premise	All ML models	Per training session	CO_2 emissions (kg)	Energy consumption (kWh)	Not RAPL based	Yes	No
Proposed Service	On-Premise	Specific models (mentioned bellow)	Per model, per selection phase	CO_2 emissions $\frac{kg}{s}$	Energy consumption (kWh)	Predictive modeling	No	Yes

Table 1: Feature comparison of existing tools and the proposed service

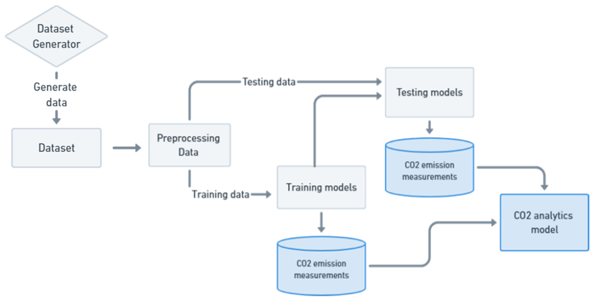


Figure 1: General Measurement Pipeline

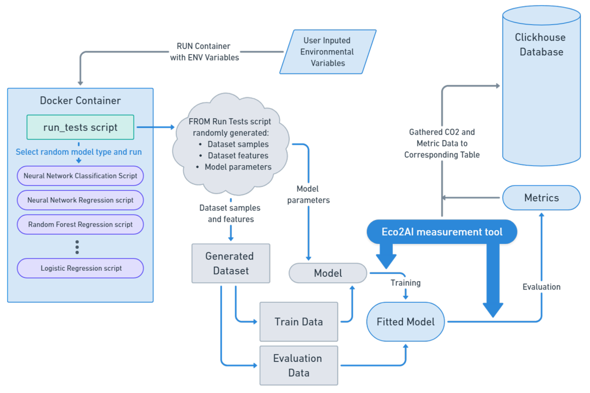

RUN Container with ENV Variables
User Inputed Environmental Variables
Clickhouse Database
Docker Container
run_tests script
Select random model type and run
Neural Network Classification Script
Neural Network Regression script
Random Forest Regression script
Logistic Regression script
FROM Run Tests script randomly generated:
• Dataset samples
• Dataset features
• Model parameters
Model parameters
Dataset samples and features
Gathered CO2 and Metric Data to Corresponding Table
Metrics
Eco2AI measurement tool
Generated Dataset
Model
Training
Evaluation
Train Data
Fitted Model
Evaluation Data

Figure 2: Single Model Measurement Pipeline

various features such as CPU type, GPU type, region, and other experiment-specific details. The model implementation is encapsulated within a Python class named `MultiModel`, which is responsible for managing the entire process from data preprocessing to training and prediction.

The model employs two separate neural networks for predicting CO_2 emissions and power consumption. The architecture for each neural network is as follows:

- **Input Layer:** Receives the scaled and encoded features.
- **Hidden Layers:** Consist of multiple Dense layers with ReLU activation functions. The CO_2 emissions model includes three hidden layers with 128, 64, and 128 neurons, respectively, while the power consumption model has three hidden layers with 64, 64, and 128 neurons.
- **Output Layer:** A single neuron that outputs the predicted value for either CO_2 emissions or power consumption.

4.1 Model Training

The model is compiled using the Adam optimizer [6] and the Mean Squared Error (MSE) loss function. Seeing as we were unable to gather adequate real-time environmental data of factors that may influence our predictions (e.g. Distribution of energy sources, real time CO_2 per kWh), our model relies on static yearly averages of these values[8] [9] . Our model uses the aforementioned features for the purpose of regression with the goal of predicting power consumption and CO_2 emissions gathered by previously mentioned random tests. Each model is trained for 25 epochs using the preprocessed data. After training, the models, along with their respective scalers and encoders, are saved to disk for later use.

4.2 Prediction

Once trained, the model can predict CO_2 emissions and power consumption for new data points by loading the appropriate model, scaler, and one-hot encoder. The input data is preprocessed in the same manner as during training, and the predictions are obtained by applying the trained models.

This modular approach allows for easy extension to additional models or data sources and provides a scalable solution for analyzing the environmental impact of machine learning processes.

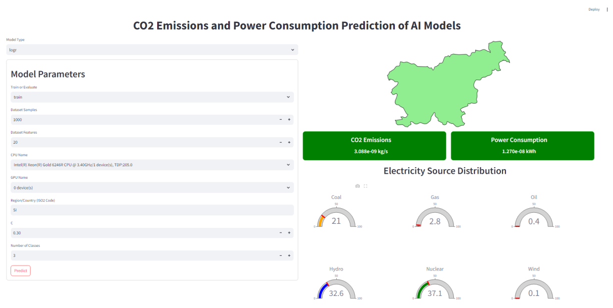
5 Web Application Interface for CO_2 Emissions and Power Consumption Prediction

In addition to the backend model developed for predicting CO_2 emissions and power consumption of various AI models, a web application was created to provide a user-friendly interface for real-time predictions. The web app, as shown in Figure 3, allows users to select different machine learning models and configure parameters to estimate the associated environmental impacts.

5.1 Key Features of the Web Application

The web application interface is designed with simplicity and functionality in mind. It includes several key components:

- **Model Selection:** Users can choose the type of machine learning model they are interested in evaluating (e.g., Logistic Regression (abbr. LogR), Decision Tree Classifier (abbr. DTC), Decision Tree Regression (abbr. DTR), Neural


CO2 Emissions and Power Consumption Prediction of AI Models

Model Type: logr

Model Parameters

Train or Evaluate: train

Dataset Samples: 1000

Dataset Features: 20

CPU Name: Intel(R) Xeon(R) Gold 6246R CPU @ 3.40GHz/1 device(s), TDP:205.0

GPU Name: 0 device(s)

Region/Country (ISO2 Code): SI

C: 0.30

Number of Classes: 3

CO2 Emissions: 3.088e-09 kg/s

Power Consumption: 1.270e-08 kWh

Electricity Source Distribution

Figure 3: Web App Interface

Network Classifier (abbr. NNC), Neural Network Regression (abbr. NNR), Linear Regression (abbr. LinR), Random Forest Classifier (abbr. RFC) and Random Forest Regression (abbr. RFR)). The dropdown menu in the upper-left corner of the interface provides a list of available models.

- **Model Parameters Configuration:** A section labeled "Model Parameters" allows users to specify various inputs:
  - **Train or Evaluate:** Users can choose whether to estimate emissions for the training or evaluation phase of the model.
  - **Dataset Samples and Features:** Input fields are provided for users to define the size of the dataset in terms of the number of samples and features.
  - **CPU and GPU Specifications:** The app allows the selection of the CPU and GPU type, reflecting different hardware configurations, such as "Intel(R) Xeon(R) Gold 6246R CPU @ 3.40GHz/1 device(s), TDP:205.0" or "AMD Ryzen 7 4800H with Radeon Graphics/1 device(s), TDP:45.0".
  - **Region/Country Selection:** A dropdown to select the geographic location where the model is being executed, which influences the CO_2 emissions based on local energy sources.
- **Real-Time Predictions:** Once all parameters are configured, the application dynamically calculates and displays:
  - **CO_2 Emissions:** The predicted emissions are shown in kilograms per second (kg/s).
  - **Power Consumption:** The power consumption is provided in kilowatt-hours (kWh).
- **Electricity Source Distribution:** A graphical representation is provided for the distribution of electricity sources, such as coal, gas, and oil, in the selected region. This information is crucial for understanding the environmental impact of power consumption based on the local energy mix.

5.2 User Experience and Accessibility

The web application is developed with accessibility in mind, ensuring that users, regardless of technical background, can interact with the model's predictive capabilities. By offering a clear and intuitive interface, it aims to make the process of estimating CO_2 emissions and power consumption transparent and straightforward.

Figure 3 illustrates the application's main screen, where the model type, parameters, and results are all visible at a glance. This real-time feedback loop allows users to make informed decisions based on the predicted environmental impact.

6 Results

6.1 Model Error

To evaluate the performance and accuracy of the models, we conducted a 10-fold cross-validation to estimate the errors in predicting CO_2 emissions and power consumption. The results are presented in Table 2. The errors for both CO_2 emissions and power consumption were computed for both training and evaluation phases of each model type.

Note: In this context, "Train." refers not to the error on the training set, but rather to the error made by our model in predicting the CO_2 emissions / Power Consumption during the training phase of the listed model. Similarly, "Eval." refers not to the error on the evaluation set, but rather to the error made by our model in predicting the CO_2 emissions / Power Consumption when the listed model makes predictions. This distinction is crucial to understanding the results accurately.

Table 2: Model Scaled Error Estimates from 10-Fold Cross-Validation

Model	Phase	CO_2 Error	Power Error
DTC	Eval.	0.0036	0.0043
DTC	Train.	0.0631	0.0649
DTR	Eval.	0.0032	0.0034
DTR	Train.	0.0133	0.0517
RFC	Eval.	0.0094	0.0098
RFC	Train.	0.3242	0.3582
RFR	Eval.	0.0087	0.0081
RFR	Train.	0.2565	0.2779
LogR	Eval.	0.0063	0.0057
LogR	Train.	0.0055	0.0043
LinR	Eval.	0.0099	0.0105
LinR	Train.	0.0104	0.0095
NNC	Eval.	0.0018	0.0030
NNC	Train.	0.1083	0.1216
NNR	Eval.	0.0045	0.0112
NNR	Train.	0.1051	0.1008

Based on the results obtained through the 10-fold cross-validation, it is evident that the model performance varies significantly across different algorithms and phases. One notable observation is that the errors in predicting CO_2 emissions and power consumption are relatively higher during the training phases, particularly for more complex models like Neural Networks and Random Forests [4].

This discrepancy in model performance can be attributed to the sparsity of the data collected during the measurement phase. The limited data points lead to substantial gaps in the attribute space covered by the models, resulting in erratic behavior when predicting outside these ranges. Consequently, the models show diminished accuracy and reliability when confronted with input configurations that fall beyond the scope of the original data.

Future research should focus on enhancing the robustness of these models by expanding the dataset to include a broader range of scenarios and conditions. This would help mitigate the effects of sparsity and improve the model's generalizability, ensuring more reliable predictions across diverse settings.

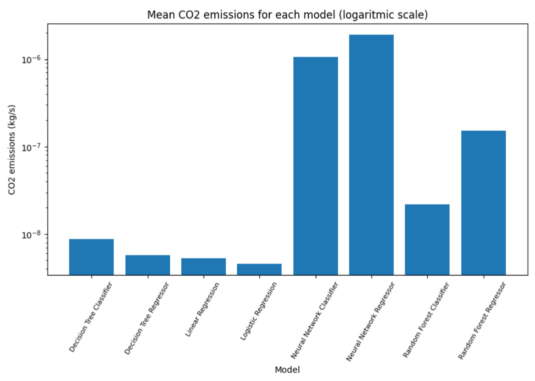


Figure 4: Logarithmically scaled mean emissions across different models

6.2 CO_2 Emission Analysis Across Different Models

Figure 4 provides a comparative analysis of the mean CO_2 emissions generated by different machine learning models during their operation, represented on a logarithmic scale to accommodate the wide range of emission values.

The chart highlights significant variations in CO_2 emissions among models, with the Neural Network Classifier and Neural Network Regressor exhibiting the highest emissions by a considerable margin. This is expected due to the intensive computational requirements and numerous parameters these models necessitate, resulting in elevated power consumption and consequently higher CO_2 output.

In contrast, simpler models like Logistic Regression, Linear Regression, and Decision Tree models show substantially lower CO_2 emissions, reflecting their reduced computational complexity and lower resource demand.

Interestingly, the Random Forest models, particularly the Regressor, present moderate emissions, illustrating that even ensemble methods, which typically involve training multiple decision trees, can maintain reasonable emission levels depending on their configuration.

This analysis underscores the importance of model selection not only for performance but also for minimizing environmental impact, particularly when scaling up operations or deploying in resource-constrained settings.

7 Discussion

The results highlight the significant environmental impact of training complex AI models, particularly neural networks. The variability in emissions suggests that optimizing model hyperparameters and selecting appropriate hardware configurations can reduce CO_2 output.

Future research should focus on model improvement for better and more accurate prediction, expanding the range of algorithms studied, as well as intensive data collection to accommodate gaps in training data.

8 Limitations

This study presents several limitations, particularly regarding the data, model evaluation, and hardware configurations, which must be considered when interpreting the results.

8.1 Training Duration and Model Learning

The models were trained for a fixed number of epochs (e.g., 10 or 20), prioritizing computational cost over learning performance. The focus was on estimating CO_2 emissions rather than model accuracy or convergence, meaning the models may not have fully captured patterns in the data. As such, the reported emissions reflect standardized training durations (with an upper limit for computational efficiency), not optimized learning outcomes.

8.2 Lack of Meaningful Learning Objective

The use of randomly generated data limits the evaluation of model learning. Since the data lacked inherent structure, the models' ability to learn was not assessed. Instead, the models were primarily evaluated on their resource consumption during training, reducing the focus on generalization or predictive power.

8.3 Hardware and Software Considerations

The experiments were conducted on specific hardware (e.g., GPU/CPU configurations), and variations in hardware were not examined. Different hardware setups, especially energy-efficient systems, could significantly impact CO_2 emissions and energy consumption. Therefore, the findings may not generalize across all hardware environments. However, we would like to point out that this was due to lack of infrastructure for broader experimentation.

9 Future Work

Future research should incorporate real-world datasets, optimize hyperparameters, and evaluate diverse hardware configurations to extend these findings to broader machine learning scenarios. The exploration of more complex architectures and learning objectives will provide a deeper understanding of the trade-offs between performance and environmental impact.

10 Conclusion

Our study presents a methodology for monitoring and analyzing CO_2 emissions during machine learning processes. The findings demonstrate that different machine learning models exhibit significant variability in their energy consumption and CO_2 emissions, with complex models like neural networks having a higher environmental impact. By providing predictive insights into these emissions, our approach enables more informed decision-making during model selection, thus contributing to the broader goal of reducing the carbon footprint of AI applications.

Future work will focus on expanding the dataset to include more diverse models and configurations. Additionally, we plan to integrate real-time monitoring tools to compare predictions with actual emissions, further refining our predictive capabilities. Moreover, optimizing model hyperparameters and exploring alternative, more sustainable hardware configurations will be key areas of investigation for minimizing the environmental impact of machine learning workflows.

Acknowledgements

This work was supported by the FAME project, funded by the European Union's Horizon 2023 Research and Innovation Programme under grant agreement nº 101092639.

References

[1] Climatiq. 2023. Climatiq: emissions intelligence platform. Provides data on the carbon emissions of various activities, including computing. (2023). https://www.climatiq.io/.

[2] CodeCarbon Development Team. 2023. Codecarbon: an open source tool for tracking the carbon emissions of machine learning experiments. An open-source tool for tracking and reducing carbon emissions in machine learning models. (2023). https://github.com/mlco2/codecarbon.

[3] Eco2AI Development Team. 2023. Eco2ai: real-time co2 emission tracking for machine learning. A tool for real-time tracking of CO2 emissions during machine learning processes. (2023). https://github.com/sb-ai-lab/Eco2AI.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. A comprehensive resource on machine learning algorithms, including neural networks. MIT Press. http://www.deeplearningbook.org.

[5] John D Hunter. 2007. Matplotlib: a 2d graphics environment. *Computing in science & engineering*, 9, 3, 90–95.

[6] Diederik P Kingma and Jimmy Ba. 2014. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[7] Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*. Vol. 445, 51–56.

[8] OurWorldInData. [n. d.] (). https://ourworldindata.org/grapher/carbon-intensity-electricity?tab=table.

[9] OurWorldInData. [n. d.] (). https://ourworldindata.org/electricity-mix.

[10] Fabian Pedregosa et al. 2011. Scikit-learn: machine learning in python. *Journal of machine learning research*, 12, 2825–2830.

[11] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *arXiv preprint arXiv:1907.10597*. 12 pages. DOI: 10.48550/arXiv.1907.10597.

[12] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 3645–3650.