EcoML: A tool to track and predict the carbon footprint of machine learning tasks

Stefan Igescu, Giovanni Monea, Vincenzo Pecorella

***Abstract*—In Machine Learning field, researchers, engineers and users are usually interested in getting finer predictions and, to achieve this result, training complex models for days or weeks is allowed and even encouraged, even if results were only slightly better.**
However, this could result in very small increases in quality at the cost of huge amounts of energy. As in recent years climate change consequences and environmental issues are getting dramatically worse, the authors think ML researchers and engineers need to be aware of the cost of their models.
We therefore propose and describe our framework with two purposes. The first one is to make the measurement of carbon footprint of computations as easy and accurate as possible, so that one can estimate and share the carbon footprint of ML models with minimum effort. The second aim is to help ML engineers choose the greenest ML model and avoid costly techniques if their goal could be nonetheless reached without them.

I. Introduction

Nowadays, environmental and climate concerns are spreading across the different fields of science and technology. Although clean energy efforts are focused on industries like transports and agriculture, the negative impact of ICT sector cannot be ignored anymore[1].

In particular, in the case of Artificial Intelligence, a relentless desire for higher accuracy and for larger datasets can cause huge emissions for very little return in the quality of the algorithms. In this spirit, the authors decided to contribute to increasing this awareness in Machine Learning field.

For this work, we built on an existing tool, Cumulator[2]. Cumulator is a tool designed to quantify and report the carbon footprint (or, equivalently, consumption) of machine learning computations and communication. Even if we expanded an existing tool, this is a very new and fresh field, with not many developments and solutions. Therefore, we designed and built from scratch a Machine Learning tool with the aim of leading to a more green choice of the Machine Learning methods used.

II. Work Done summarized

The first part of our work aims to improve the existing features of Cumulator. In particular, we improved the estimation of the consumption by adding the possibility to take into account the position of the user and the hardware used for the computations. Then, we added a function so that the user does not have to start and stop Cumulator by hand, in order to make its use as comfortable as possible. In addition, we integrated a continuous integration system to it (namely, Cirrus CI), in order to ease the approval of further improvements to Cumulator. Finally, we added new types of measures of consumption to give a better idea of the cost of computations in an everyday-life context.

Finally, we describe a new ambitious feature we integrated in Cumulator: a tool to lead the user to a green choice of which Machine Learning algorithm to use. Precisely, given a dataset, our tool will predict the F1 score and the consumption of different classification methods on this dataset, so that the user can choose the model that can reach an optimal trade-off between utility and consumption.

In order to realize this tool, we required what we call a meta-dataset, a dataset where rows represent datasets themselves, along with information about their training. As the authors are not aware of the existence of such a dataset, we designed a completely automated system to populate this meta-dataset: all the datasets are gathered from OpenML[3], an online machine learning platform for sharing and organizing data, machine learning algorithms and experiments. Then, all the datasets will be trained with an AutoML tool. This will let us have an arbitrarily big meta-dataset, without the weight of training every single dataset by ourselves. Beyond the scope of this work, we also believe this pipeline can be easily adopted for further ML tasks that require the training of a similar meta-dataset.

III. Cumulator Software Improvements

In order to make Cumulator[2] more reliable in estimating consumption and to add functionalities, we have included the following features:

A. Cirrus CI integration

We integrated Cirrus CI[4] in the GitHub repository of Cumulator to automatically perform unit and integration tests after every software push. We then developed tests to verify the correct functioning of the tool.

B. Pratical metrics of consumption

To concretely let the user understand the order of magnitude of the energy consumed, we included in the tool different measuring metrics such as the number of meters driven by an average passenger vehicle or the number of smartphones that could be charged with the amount of energy consumed by the user. These will be displayed together with the real consumption as an output of the tool at the end of the computation.

C. Auto tracking

In the original version, Cumulator[2] had to be manually started and stopped to measure consumption in the required period. We also added the possibility to do it automatically just calling a Cumulator's method. The goal was to be able to track the carbon emissions of a generic ML task using external libraries such as SKLearn[5].

D. Improved Estimation

We then further improved the accuracy of the carbon footprint taking into account the type of hardware used, and so its precise consumption, and the location where the computation is performed (check the appendix for more information about this).

In order to estimate the current *Hardware Thermal Design Power*, we have employed the CPU/GPU Specs Database from TechPowerUp[6]: it contains the *TDP* of the most common CPU/GPU present on the market and we used the python libraries *CPUInfo*[7] and *GPUutils*[8] to find the current hardware used.

In addition, we implemented an ip-to-position resolution with *geocoder* library to obtain the current position of the user to have a more accurate measure of carbon footprint.

Both these improved metrics are automatically computed when the tool is started and they can be optionally deselected if the user doesn't want to install additional libraries or has privacy concerns on them. In that case, the tool is going to use the standard values to compute the carbon footprint of the requested task.

IV. Cumulator's Prediction Functionality

Our final scope was to integrate in Cumulator[2] a new prediction functionality. The prediction task was to return, given a dataset in input, the best ML algorithm to train a model on the input dataset that can reach an optimal trade-off between prediction F1 score and carbon footprint.

After some more experiments (described in the appendix) we concluded that it could be possible to predict the F1 score and the carbon footprint of a specific machine learning algorithm given only the dataset on which it is to be trained. To accomplish this task, we needed a meta-dataset containing in each the characteristics of a generic dataset, a ML algorithm, the F1 score and the consumption obtained training that algorithm on that dataset. Since an in-depth analysis of scientific literature's papers on the topic did not lead to concrete results, we decided to build our own meta-dataset. Following our assumptions, given K generic datasets and N machine learning algorithms, our meta-dataset will be composed of $|\text{MD}| = N \cdot K$ rows. A general idea of the procedure is described in the first part of Figure 1.

A. Meta-Dataset definition and construction

In order to build the meta-dataset we faced three main challenges:

- The necessity of collecting K datasets.
- The definition of which features will compose the meta-dataset and will be than able to self-describe the dataset they represent.
- The necessity to run N algorithms on a significant number K of datasets and register for each combination the F1 score and the carbon footprint.

To collect a large number of dataset and to manage the huge amount of data generated, we based our work on OpenML collection of datasets[3]. In particular we used OpenML python APIs[9] to retrieve the datasets and carry out computations over them. We decide to focus our experiments only on datasets intended for classification purposes, with a number of features ≤ 1000, a number of instances ≤ 100000 and a number of classes of the target attribute from 2 to 20. With those constraints, we fetched from OpenML around 1000 datasets to work with.

To define which features of the meta-dataset we should use, we carried out further tests on the common datasets highlighted above in order to determine the best measures and characteristics to capture the fundamental traits of them that could become the features of the meta-dataset. We also included measures computed directly by OpenML on its datasets to clearly identify the final features of the meta-dataset.

Features of the Meta-Datasets	
F1	time
MajorityClassSize	MaxNominalAttDistinctValues
MinorityClassSize	NumberOfNumericFeatures
NumberOfClasses	NumberOfSymbolicFeatures
NumberOfFeatures	TDP
NumberOfInstances	Country
NumberOfInstances WithMissingValues	Max_correlation
NumberOfMissingValues	Algorithm

Table I
Features of the meta-dataset: A complete description of them can be found in the appendix

In order to construct the $N \cdot K$ rows of the meta-dataset we used MLJar python library[10]: this AutoML library gave us the possibility to train automatically the required models over the K datasets. The built-in pre-processing and hyper-parameters fitting allowed us to make the accuracy obtained and its relative consumption realistic. We further decided to use $N = 4$ ML algorithms as follow: Decision Tree, Logistic Regression, Neural Network and Random Forest.

Thanks to MLJar we achieved F1 score results comparable to those that could be obtained manually, thus resulting in a plausible estimate of the F1 score and Carbon Footprint results but using a fully automated procedure.

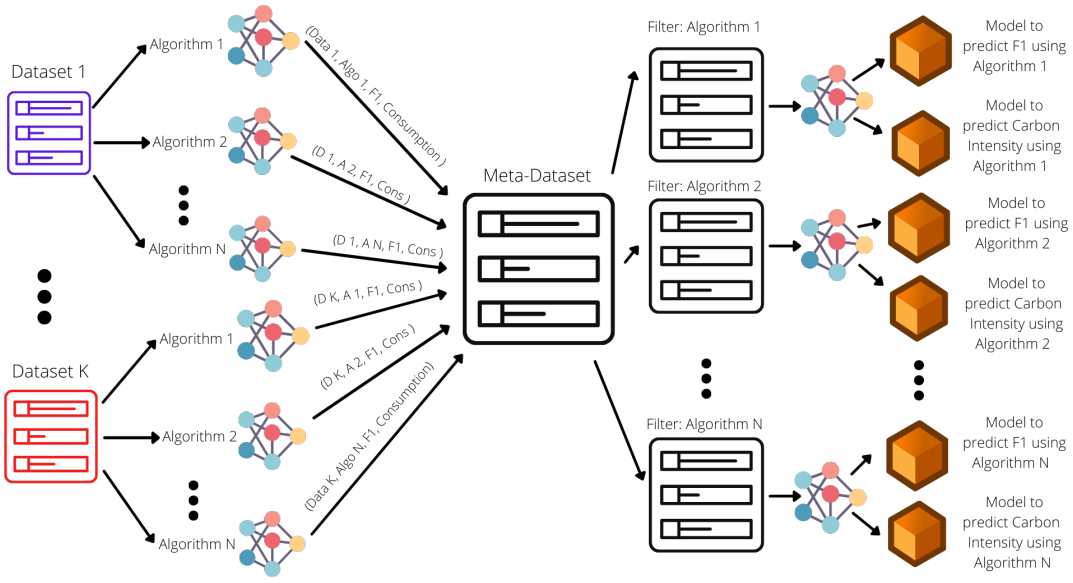


Figure 1. Pipeline of the project

B. Predicting F1 and Carbon footprint over the Meta-Dataset

Once obtained the meta-dataset, we split it in $N = 4$ subsets, each containing the results of a particular machine learning algorithms from the ones mentioned above. As showed in Figure 1, we then tried to build 8 models to predict respectively the F1 score and the carbon footprint achieved training each on input dataset, one of the $N = 4$ algorithms specified above. Using our 8 models and given a dataset from the user, the final goal was then to plot for each of the used algorithm, the predicted F1 score and carbon footprint obtained, to let then the user choose the algorithm with the best trade-off needed for his scope.

V. Prediction Results

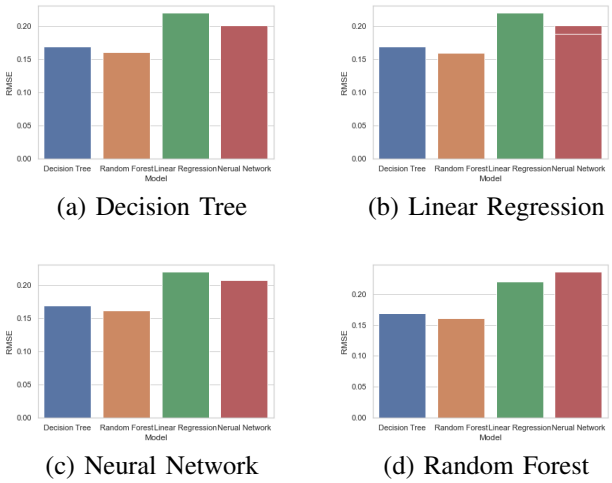
In all cases, we used the $RMSE$ to measure how good our models performed, and we split our dataset into train and test sets with a test size of 33% and a train size of 67%.

We recall how our objective is to obtain a model for predicting the F1-score and the consumption (i.e the carbon footprint) of a given algorithm on a given dataset. Therefore at the end, we will obtain 8 different models: 4 for predicting F1-score respectively for the algorithms Decision Tree, Linear Regression, Neural Network and Random forest, and 4 for predicting consumption on the same algorithms.

A. F1-Score

To be able to make the best prediction of the F1-score, we decided to use different algorithms in parallel and to pick the one giving us the best results. Therefore we trained Linear Regression, Random Forest, Decision Tree and Neural Network models for each subset before mentioned. In all cases, when possible, we performed cross-validation in order to fine-tune the respective hyper-parameters. In the end, we picked the model performing the best.

In addition, to increase our capacity in prediction, we studied the distribution of our features and observed how the majority of them has a heavy-tailed distribution. Therefore we decided to apply a logarithmic scaling to them. Besides, we selected the $k = 7$ most relevant features having the highest mutual information value with respect to the target vector.



(a) Decision Tree (b) Linear Regression

(c) Neural Network (d) Random Forest

Figure 2. $RMSE$ of different models trained on the 4 subsets: Decision Tree, Linear Regression, Neural Network, Random Forest

In Figure 2 we show our results: each plot represent the results of the different models we trained in parallel on one among the $N = 4$ subsets of the meta-dataset.

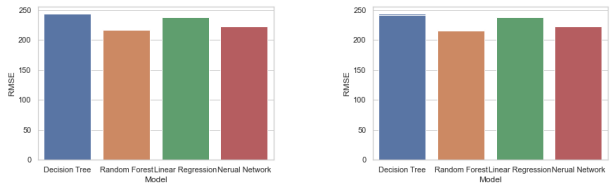
We can see how the best model turned out to be the Random Forest in all cases, with $RMSE \approx 0.16$.

As our target, the F1-score, has values in the interval $[0, 1]$ the $RMSE$ can be considered acceptable.

Since in all cases the Random Forest algorithm performed the best, we decided to use it as our final model for predicting the F1-score in all the subsets.

B. Consumption

The approach for creating the models for predicting the consumption of a given data set is the same as before. Therefore we directly report the results we obtained with our models after pre-processing and fine-tuning in Figure 3. Here the $RMSE$ takes values in the interval $[200, 250]$, and again it is acceptable since our target vector has values in the interval $[200, 4000]$.



(a) Decision Tree (b) Linear Regression

(c) Neural Network (d) Random Forest

Figure 3. $RMSE$ of different models trained on the 4 subsets: Decision Tree, Linear Regression, Neural Network, Random Forest

It is important to remark how in addition to the previous preprocessing we had to remove also some outliers from the target vector to have acceptable results. We proceeded by using the definition of outlier:

$$[Q_1 - kI, Q_2 + kI]$$

where I represents the target vector and Q_1, Q_2 are the lower and upper quartiles respectively. We used a value of $k = 0.2$.

In the end, as shown in Figure 3, also in the prediction of the consumption the best model resulted to be the Random Forest in all the subsets.

VI. Discussion and Further Improvements

By looking at Figure 2 and Figure 3, we can observe how the results of our first attempt of trying to predict the F1 score of a given model on a given dataset turned out to be promising.
However, there are different ways of improving them.

First of all, a larger quantity of data would help in this task. Even if we managed to build a consistent meta-dataset thanks to the OpenML API, this can't be considered enough for such a task.

Therefore we propose the idea of gathering information from the users' usage of Cumulator: each time someone would use it, we could record the properties of the dataset, the model the user trained and the relative score and append this information to our meta-dataset.

A second important consideration must be done about the features we choose to have in our meta-dataset. In our current implementations, we have 11 features, however, this may not be enough to describe in-depth our datasets in order to be able to predict correctly.
Thus, further research about extra features should be done.

In conclusion, the next step, which would make this project more complete, concerns expanding also to regression models. Indeed, in our work, we focused on predicting the F1-score and the consumption of only classification models, whereas the regression ones have been ignored for simplicity.

VII. Summary

In conclusion, we can see how the path for improving the Cumulator tool involved several steps. We started by enhancing it from a software point of view. Subsequently, we tried to improve Cumulator by implementing a feature based on a Machine Learning model to give the user of Cumulator the choice of selecting the best-suited method for their needs and with the least possible carbon footprint. For this, we designed and built an arbitrarily big meta-dataset with an automated pipeline.

Overall, we saw how it is possible to try to measure and understand the impact of Machine Learning in terms of CO2 emissions and we are confident that with time Cumulator may make people more aware of the consequences of intense machine learning training and modelling.

In addition, we hope that the idea of giving the possibility to the user to select their algorithm based on the prediction of the F1-score and the consumption will lead to a more green choice of the methods used, therefore to a reduction of CO2 emissions.

VIII. APPENDIX

A. Technical details on carbon footprint and Cumulator

As showed in the original Cumulator paper[2], time can be assumed as a good estimator of the energy efficiency, Thermal Design Power (TDP) of the processor as a good estimator of the GPU's power consumption. Therefore, the carbon footprint of computations was calculated as

$$\text{Carbon footprint} = Time(s) \cdot \frac{\text{H\_TDP}(0.250kW) \cdot \text{CI}(447gCO2eq/kWh)}{3600s/h}$$

Where H_TDP is the *Hardware Thermal Design Power* of a specific CPU/GPU and CI is the *Carbon intensity* of the energy mix that is being used. Both these values were fixed in the original version of Cumulator, independently of the specific CPU/GPU used for the task and the location where the computation is run.

As reported above, the carbon footprint depends on the *Carbon intensity*. The latter is defined as the amount of greenhouse gas emissions in grams carbon dioxide equivalent per unit of energy produced (gCO2eq/kWh) and strongly depends on the type of energy source used to provide the electricity grid at the location of the computing machine or data centre. The constant value (447gCO2eq/kWh) used in the original version of Cumulator corresponded to the average carbon intensity value in the EU in 2014. This value, however, is very variable, both in time and in the country in which the machine is located. Therefore, we decide to improve the accuracy of this measure. For this purpose, we collected the dataset on Energy by Our World in Data[11] that contains the *Carbon intensity* for most of the countries and we implemented an ip-to-position resolution with *geocoder* library to obtain the current position of the user.

B. Experiments of section IV

During our experiments on some of the most common datasets used in academic publications, we noticed that both F1 score and carbon footprint are strongly dependent on the characteristics of the observed dataset, as reported in the table below.

Using an heavier algorithm in terms of power consumption does not always result in better performance. Moreover, with some particular datasets such as Credit Card Fraud[13], more powerful algorithms can also perform worse. In this case it was due to the small support of one of the target classes that led to an extremely low recall (0.04) over that class.

Dataset	Algorithm	F1 Score	Carbon Footprint (mgCO2eq)
Titanic (target = 'Survived')	Logistic Regression	0,75	0,13445
	Random Forest	0,75	3,14490
	Neural Network	0,76	13,79248
Credit Card Fraud (target='Class')	Logistic Regression	0,70	76,16450
	Random Forest	0,86	1880,92903
	Neural Network	0,076	1119, 52090

Table II
Common ML algorithms trained over Titanic[12] and Credit Card Fraud[13] Datasets. Soft preprocessing and model tuning performed. Measurements performed in Switzerland with a 2,5 GHz Quad-Core Intel Core i7 CPU.

C. Features of the meta-dataset

We provide a more precise description of the features used in the meta-dataset. Also, we note that we do not store the consumption itself. What we store is the time spent for training, the country where the training took place and the TDP of the device used for the training. With these three variables, we can finally compute the consumption, as shown in section III-D. In this case, we also included the dataset ID, the unique identifier of the dataset on OpenML[3]:

- did: Unique identifier of the dataset
- F1: F1-score of the model
- time: Time spent for training the model
- MajorityClassSize: Size of the largest class
- MinorityClassSize: Size of the smallest class
- MaxNominalAttDistinctValues: Maximum number of distinct values among nominal type features
- NumberOfNumericFeatures: Number of numerical features in the dataset
- NumberOfClasses: Number of different classes of the task
- NumberOfSymbolicFeatures: Number of symbolic features in the dataset
- NumberOfFeatures: Number of features of each sample
- NumberOfInstances: Number of rows in the dataset
- TDP: Thermal Design Power of the processor used for training the model
- Country: Country where the model was trained
- NumberOfInstancesWithMissingValues: Number of rows containing missing values
- NumberOfMissingValues: Number of missing values in the dataset
- Max_correlation: Maximum linear correlation between numerical features and the class label
- Algorithm: ML algorithm trained

REFERENCES

[1] (2021) It's not easy being green. [Online]. Available: https://www.thelancet.com/journals/landig/article/PIIS2589-7500(21)00257-0/fulltext

[2] M. J. Tristan Trebaol, Mary-Anne Hartley and H. S. Ghadikolaei, "A tool to quantify and report the carbon footprint of machine learning computations and communication in academia and healthcare," *Infoscience EPFL: record 278189*, 2020.

[3] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013. [Online]. Available: http://doi.acm.org/10.1145/2641190.2641198

[4] (2021) Cirrus ci website. [Online]. Available: https://cirrus-ci.org

[5] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[6] (2021) Techpowerup website. [Online]. Available: https://www.techpowerup.com/cpu-specs/?sort=name

[7] (2021) py-cpuinfo 8.0.0 library. [Online]. Available: https://pypi.org/project/py-cpuinfo/

[8] (2021) Gputil 1.4.0 library. [Online]. Available: https://pypi.org/project/GPUtil/

[9] A. K. e. a. Matthias Feurer, Jan N. van Rijn, "Openml-python: an extensible python api for openml," *arXiv*, vol. 1911.02490, 2019. [Online]. Available: https://arxiv.org/pdf/1911.02490.pdf

[10] A. Płońska and P. Płoński, "Mljar: State-of-the-art automated machine learning framework for tabular data. version 0.10.3," Łapy, Poland, 2021. [Online]. Available: https://github.com/mljar/mljar-supervised

[11] H. Ritchie and M. Roser, "Energy," *Our World in Data*, 2020. [Online]. Available: https://ourworldindata.org/energy

[12] (2021) Titanic dataset on kaggle. [Online]. Available: https://www.kaggle.com/c/titanic

[13] (2021) Credit card fraud dataset on kaggle. [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud