

# The global warming of the Web

**Measuring the environmental footprint of HTTP requests (EcoIndex)**

[christophe.cerin@univ-grenoble-alpes.fr](mailto:christophe.cerin@univ-grenoble-alpes.fr)

March 27, 2023 – GreenDays

# 1. Problem statement

# Focus on HTTP requests (not really the Web)

Can you quantify the evolution of the environmental footprint of HTTP requests (increase or decrease)?

You need:

- A metric: URL  $\rightarrow$  score;
- A dataset of URLs.



Programme national nutrition santé (PNNS),  
présidé par [Serge Hercberg](#),

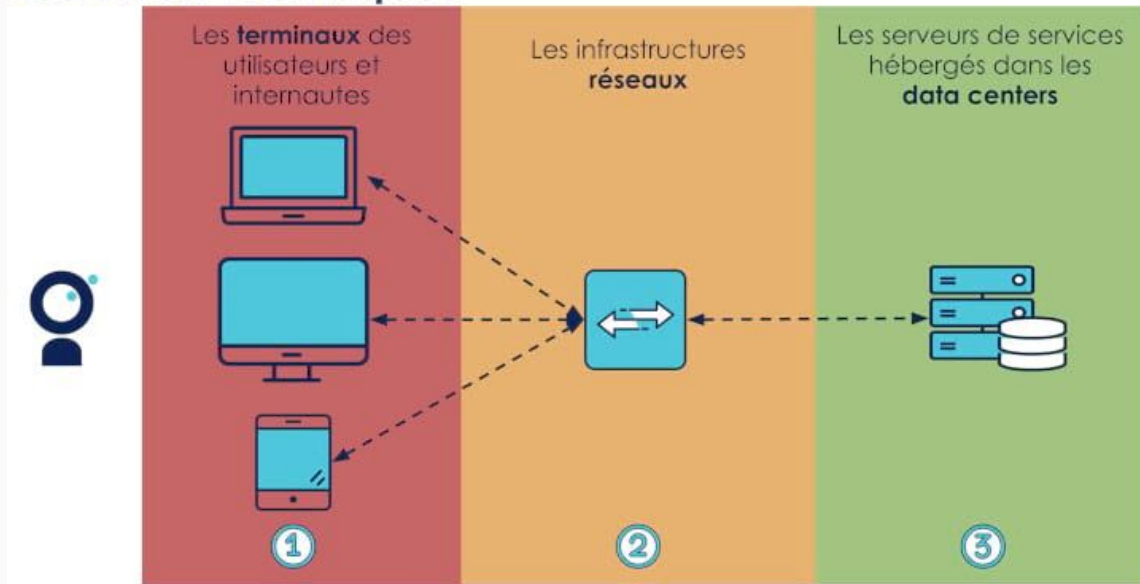
In the EcoIndex, it was chosen to represent:

An endpoint  $\Rightarrow$  by the number of DOM elements;

The network  $\Rightarrow$  by the "weight" of the page (KB returned) ;

The servers  $\Rightarrow$  by the number of http/https requests in the returned page.

## Les 3 tiers techniques



# Fundamental equation

$$Ecoindex = 100 - \frac{5 * [3 * Fd(Taille_{DOM}) + 1 * Fs(Poids_{Page}) + 2 * Fr(Nb_{Requêtes})]}{6}$$

Key points:



1. Normalization;
2. Weights 3, 2, and 1 based on macro studies;
3. Fd, Fs, and Fr – return ranks (from 0 to 20) in quantiles vectors. Check with <https://github.com/cnumr/GreenIT-Analysis/blob/master/script/ecoIndex.js>

# EcoIndex is also...

1. A set of best practices;
2. A score from A to G;
3. A score for gas emission;
4. A score for water consumption.

```
function getEcoIndexGrade(ecoIndex)
{
  if (ecoIndex > 80) return "A";
  if (ecoIndex > 70) return "B";
  if (ecoIndex > 55) return "C";
  if (ecoIndex > 40) return "D";
  if (ecoIndex > 25) return "E";
  if (ecoIndex > 10) return "F";
  return "G";
}

function computeGreenhouseGasesEmissionfromEcoIndex(ecoIndex)
{
  return (2 + 2 * (50 - ecoIndex) / 100).toFixed(2);
}

function computeWaterConsumptionfromEcoIndex(ecoIndex)
{
  return (3 + 3 * (50 - ecoIndex) / 100).toFixed(2);
}
```

## 2. Criticisms and new approaches

# Limitations inherent to the 3-tier model

BAD – EcoIndex does not consider:

- environmental impact of the computer making the query or of a user browsing;
- environmental impact of servers in the classical sense of life cycle assessments (LCA), nor of the different network types of equipment.

GOOD –

- discussion of models and their attributes that would meaningfully characterize the environmental impact of digital at the HTTP requests level;
- loading, creation, and display of the page in the browser are not simulated.



# Limitations inherent to the calculation itself

## QUESTIONABLE –

- Weights (3,2,1) are static: they do not consider variations over time or country, and these are limits for the model that we can describe as temporal and geographical;
- Stability over time of quantiles? A priori, websites are regularly reviewed to adopt, over time, the best eco-design practices, and therefore there is no reason for quantiles to be seen as static quantities;
- Websites are dynamic;
- The A-G scores correspond to the EcolIndex ranges of 100-81 for A and 10-0 for G. How were these different bounds determined? Do they fit the quantiles for the HTTParchive EcolIndex measures? This is still being determined.

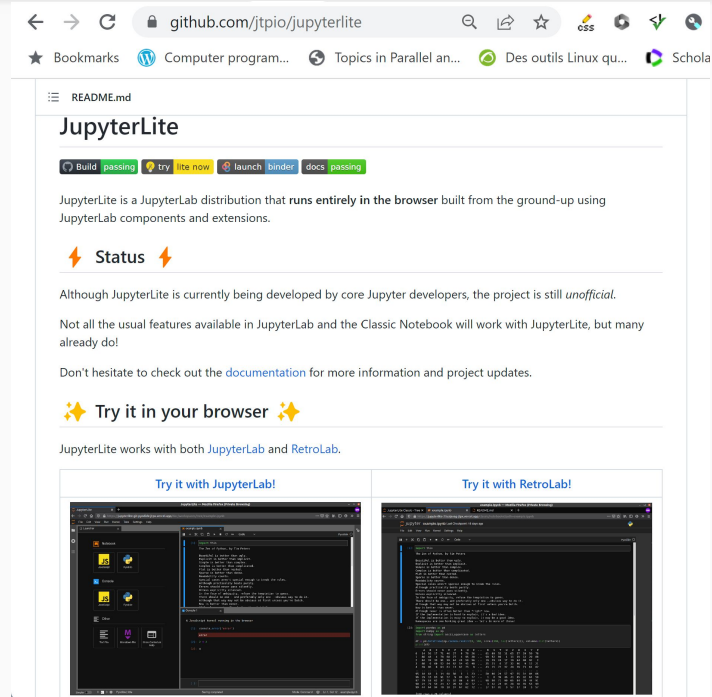
# Limitations inherent in the attributes that make sense

How to introduce new attributes into the model to capture:

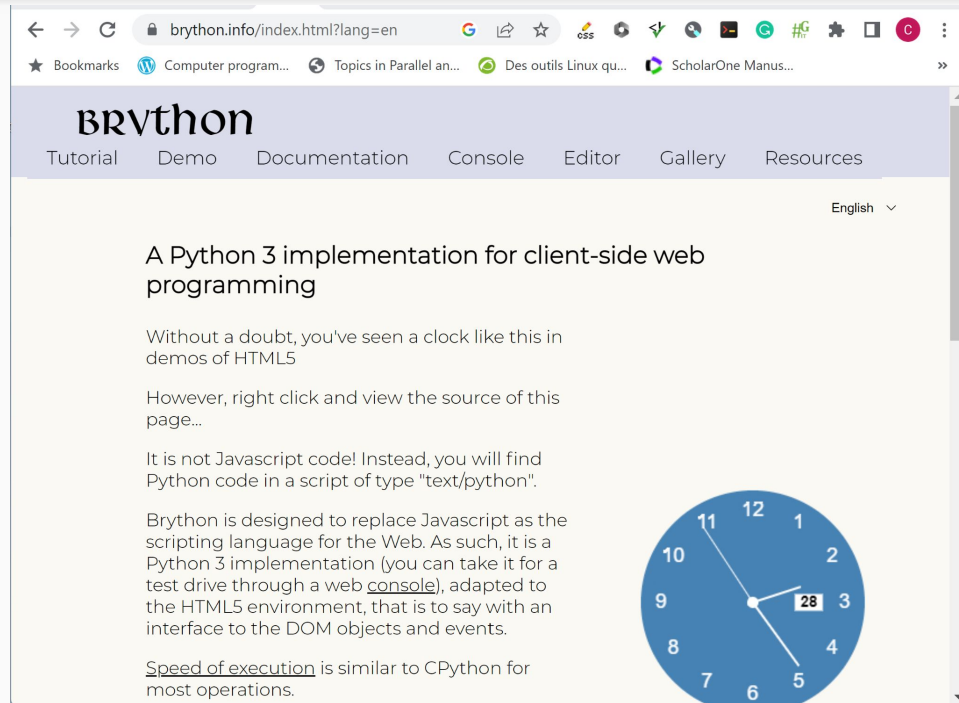
- Energy mix;
- HTTP request goes through a 4/5G network or fiber;
- aggregate the CO<sub>2</sub> impact of the operator;
- ...

A potential of 10, 100, ... attributes, all related to “energy”. How to deal with a large number of attributes?

# Limits inherent to the potential use of other forms of interaction models



The screenshot shows the GitHub repository for JupyterLite. The page title is "JupyterLite". Below the title, there are buttons for "Build", "try", "launch", and "docs". The "try" button is highlighted with a yellow background. The text below the buttons states: "JupyterLite is a JupyterLab distribution that runs entirely in the browser built from the ground-up using JupyterLab components and extensions." There is a section titled "Status" with a lightning bolt icon. The text below it says: "Although JupyterLite is currently being developed by core Jupyter developers, the project is still *unofficial*. Not all the usual features available in JupyterLab and the Classic Notebook will work with JupyterLite, but many already do! Don't hesitate to check out the [documentation](#) for more information and project updates." There is a section titled "Try it in your browser" with a lightning bolt icon. The text below it says: "JupyterLite works with both [JupyterLab](#) and [RetroLab](#)." At the bottom, there are two side-by-side screenshots. The left one is titled "Try it with JupyterLab!" and shows a JupyterLab interface. The right one is titled "Try it with RetroLab!" and shows a RetroLab interface.



The screenshot shows the Brython website. The page title is "Brython". Below the title, there are links for "Tutorial", "Demo", "Documentation", "Console", "Editor", "Gallery", and "Resources". The "Demo" link is highlighted. The text below the links states: "A Python 3 implementation for client-side web programming". There is a section titled "Without a doubt, you've seen a clock like this in demos of HTML5". The text below it says: "However, right click and view the source of this page...". There is a section titled "It is not Javascript code! Instead, you will find Python code in a script of type 'text/python'". The text below it says: "Brython is designed to replace Javascript as the scripting language for the Web. As such, it is a Python 3 implementation (you can take it for a test drive through a [web console](#)), adapted to the HTML5 environment, that is to say with an interface to the DOM objects and events." There is a section titled "Speed of execution" with the text: "is similar to CPython for most operations." At the bottom right, there is a blue clock face showing the time 12:28.

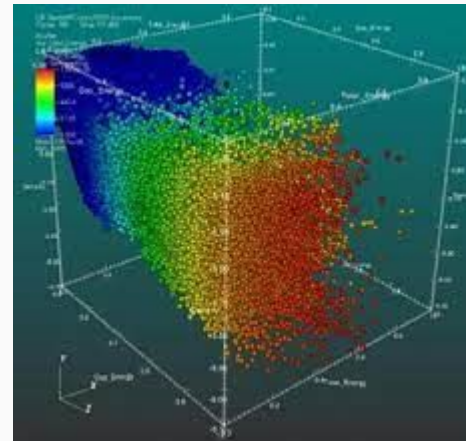
# Towards new or extended definitions

**Bypass the limits:** (some new directions classified into two directions)

1. Forget weights and quantiles;
2. Consider, for a URL,  $N \gg 1$  attributes related to direct or indirect energy consumption:
  - a. cluster URL (EcoIndex relates to the neighbors): relative measure;
  - b. place/order URLs on a line: absolute measure.

**Why?**

- Observe, classify, and query URLs to better understand their similarities. Assume that  $N \gg 1$  attributes fully characterize the "System."



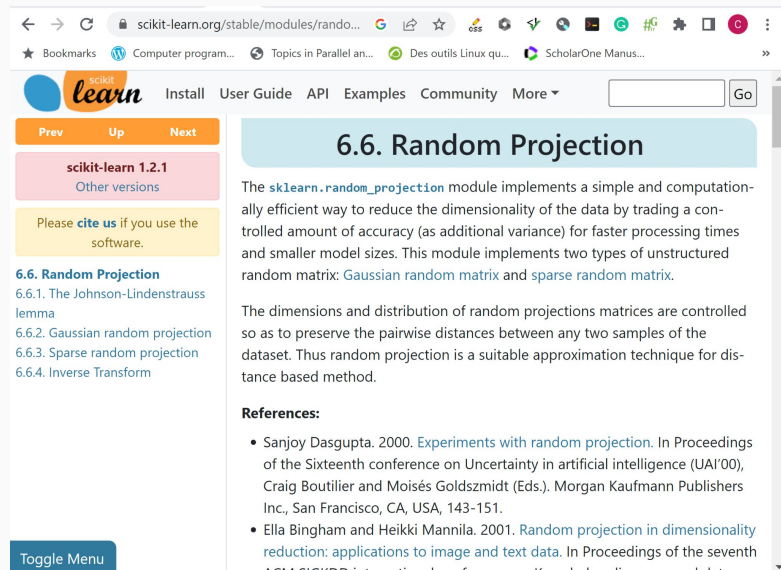
# ML techniques help to deal with the explosion of attributes

**Random Projection:** a lemma states that if points in a vector space are of sufficiently high dimension, then they may be projected into a suitable lower-dimensional space in a way which approximately preserves the distances between the points.

**Make a projection** on the line and between 0 and 100 to bypass  $F_d$  and  $F_s$  and  $F_r$

The “new EcolIndex” is the projection!

**Note:** We still reduce to a single value.



The screenshot shows the scikit-learn documentation page for the Random Projection module. The page title is "6.6. Random Projection". The text describes the `sklearn.random_projection` module, which implements a simple and computationally efficient way to reduce the dimensionality of the data by trading a controlled amount of accuracy (as additional variance) for faster processing times and smaller model sizes. It implements two types of unstructured random matrix: Gaussian random matrix and sparse random matrix. The dimensions and distribution of random projections matrices are controlled so as to preserve the pairwise distances between any two samples of the dataset. Thus random projection is a suitable approximation technique for distance based method. The page also includes a "References" section with two entries: Sanjoy Dasgupta, 2000. Experiments with random projection. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (UAI'00), Craig Boutilier and Moisés Goldszmidt (Eds.), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 143-151. and Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'01), San Jose, California, USA, 265-273.

6.6. Random Projection

The `sklearn.random_projection` module implements a simple and computationally efficient way to reduce the dimensionality of the data by trading a controlled amount of accuracy (as additional variance) for faster processing times and smaller model sizes. This module implements two types of unstructured random matrix: Gaussian random matrix and sparse random matrix.

The dimensions and distribution of random projections matrices are controlled so as to preserve the pairwise distances between any two samples of the dataset. Thus random projection is a suitable approximation technique for distance based method.

**References:**

- Sanjoy Dasgupta, 2000. Experiments with random projection. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (UAI'00), Craig Boutilier and Moisés Goldszmidt (Eds.), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 143-151.
- Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'01), San Jose, California, USA, 265-273.

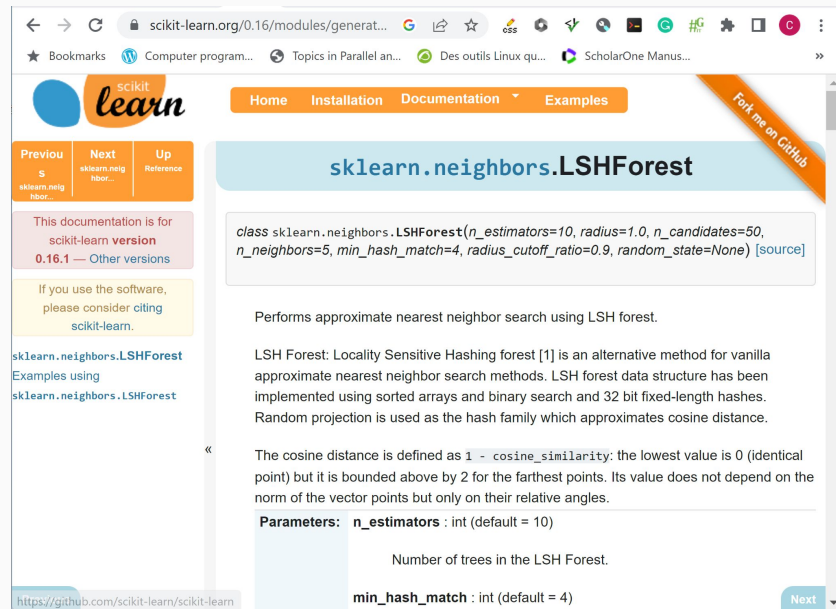
# ML techniques help to deal with the explosion of attributes

**Locality-sensitive hashing (LSH)** is an algorithmic technique that hashes similar input items into the same "buckets" with high probability.

The “new EcolIndex” is the rank of the bucket!

One of the main applications of LSH is to provide a method for efficient approximate **nearest neighbor search** algorithms.

**Note:** We still reduce to a single value.



The screenshot shows the scikit-learn documentation page for `sklearn.neighbors.LSHForest`. The page includes a navigation bar with links for Home, Installation, Documentation, and Examples. A sidebar on the left contains links for Previous, Next, and Up versions, as well as a note about citing the software. The main content area displays the class name `sklearn.neighbors.LSHForest` and its signature: `class sklearn.neighbors.LSHForest(n_estimators=10, radius=1.0, n_candidates=50, n_neighbors=5, min_hash_match=4, radius_cutoff_ratio=0.9, random_state=None)`. Below the signature, the text describes the class as performing approximate nearest neighbor search using LSH forest. It also provides a definition of cosine distance and lists the parameters: `n_estimators` (number of trees) and `min_hash_match` (number of matching hashes).

scikit-learn

Home Installation Documentation Examples

sklearn.neighbors.LSHForest

```
class sklearn.neighbors.LSHForest(n_estimators=10, radius=1.0, n_candidates=50,
n_neighbors=5, min_hash_match=4, radius_cutoff_ratio=0.9, random_state=None) [source]
```

Performs approximate nearest neighbor search using LSH forest.

LSH Forest: Locality Sensitive Hashing forest [1] is an alternative method for vanilla approximate nearest neighbor search methods. LSH forest data structure has been implemented using sorted arrays and binary search and 32 bit fixed-length hashes. Random projection is used as the hash family which approximates cosine distance.

The cosine distance is defined as  $1 - \text{cosine\_similarity}$ : the lowest value is 0 (identical point) but it is bounded above by 2 for the farthest points. Its value does not depend on the norm of the vector points but only on their relative angles.

**Parameters:** `n_estimators` : int (default = 10)

Number of trees in the LSH Forest.

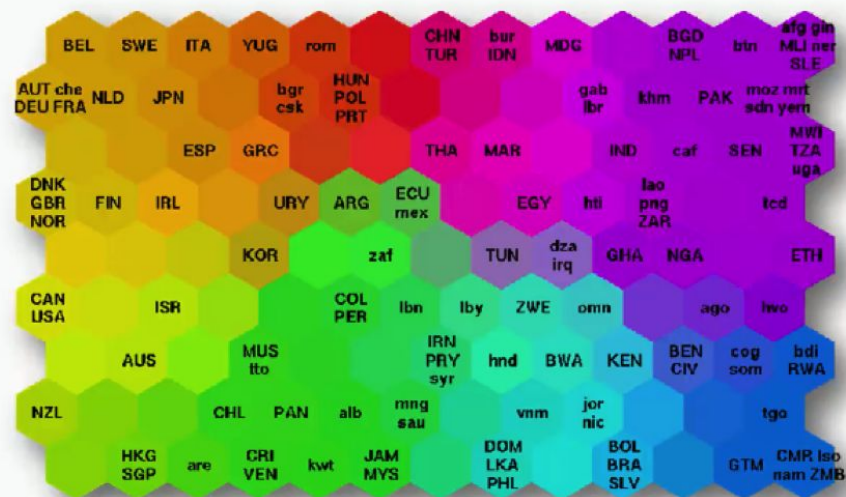
`min_hash_match` : int (default = 4)

# ML techniques help to deal with the explosion of attributes

A **self-organizing map (SOM)** is an **unsupervised machine learning** technique used to produce a **low-dimensional** (typically two-dimensional) representation of a higher dimensional data set while preserving the **topological structure** of the data.

A **set** of URLs corresponds to a SOM.

**Good:** position of one URL in the map (color) gives my neighbors - clustering (for free)



**Note:** We get a map not a single value.

# Self Organizing Map (Open Data exemple)

=== ARCEP data

2022\_QoS\_Metropole\_data\_habitations.  
csv ===

Column names of ARCEP data:

lieu  
situation  
date  
heure  
opérateur  
Profil  
rsrp  
latitude  
longitude  
protocole  
url  
file\_name

=== ENEDIS data

consommation-electrique-par-secteur-dact  
ivite-commune.csv ===

Column names of ENEDIS (consommation):

Année  
Code Commune  
Nom Commune  
....  
CODE GRAND SECTEUR  
CODE SECTEUR NAF2  
Nb sites  
Conso totale (MWh)  
Conso moyenne (MWh)  
.....

=== ENEDIS data

production-electrique-par-filiere-a-la-maille-co  
mmune.csv ===

Column names of ENEDIS data (production):

Année  
Nom commune  
Code commune  
Energie produite annuelle Photovoltaïque  
Enedis (MWh)  
Energie produite annuelle Eolien Enedis  
(MWh)  
Energie produite annuelle Hydraulique  
Enedis (MWh)  
Energie produite annuelle Bio Energie  
Enedis (MWh)  
Energie produite annuelle Cogénération  
Enedis (MWh)  
Energie produite annuelle Autres filières



# Self Organizing Map (exemple for 2021)

**ORANGE** ; 46.47044 ; -1.02718 ; 85121 ; FONTENAY-LE-COMTE ; <https://www.allocine.fr/> ; 1593 ; 29 ; 1921444 ; 40.78 ; 2.18 ; 3.28 ; 1.0 ; 2.9 ; 3651.1647229005457 ; 6.799189428120197 ; 151.0991898335209 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

**BOUYGUES TELECOM** ; 42.69109 ; 2.92053 ; 66028 ; PERPIGNAN ; ; 1.0 ; 4.25 ; 27254.2314006874 ; 5.425887199021979 ; 437.42442519542254 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

**FREE** ; 50.40982 ; 2.96885 ; 62427 ; LENS ; ; nan ; 10 ; 54767.62509277385 ; 4.428171498445493 ; 167.0849727571493 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

**ORANGE** ; 42.71066 ; 2.88409 ; 66136 ; PERPIGNAN ; <https://www.instagram.com/accounts/login/> ; 178 ; 95 ; 29041638 ; 55.95 ; 1.88 ; 2.82 ; 1.0 ; 3.46 ; 249164.689899692 ; 3.4576912601780707 ; 2704.200723216868 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

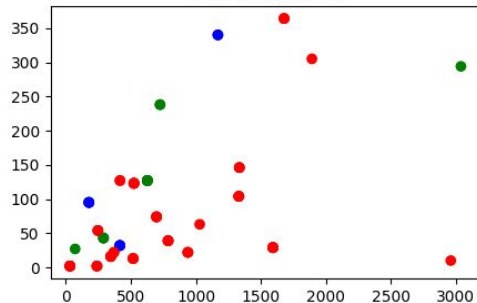
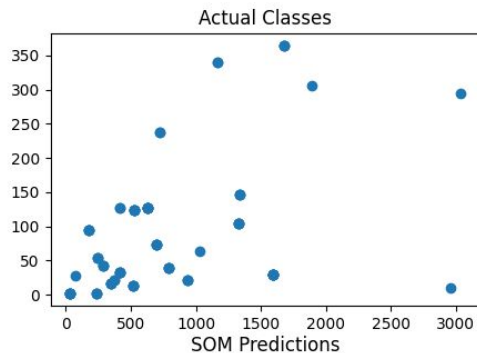
**FREE** ; 42.71066 ; 2.88409 ; 66136 ; PERPIGNAN ; <https://www.jeuxvideo.com/> ; 2961 ; 10 ; 974937 ; 43.90 ; 2.12 ; 3.18 ; 1.0 ; 5.25 ; 249164.689899692 ; 3.4576912601780707 ; 2704.200723216868 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

**ORANGE** ; 47.28475 ; -1.55131 ; 44201 ; NANTES ; [https://fr.m.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil\\_principal](https://fr.m.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal) ; 1677 ; 364 ; 355418 ; 21.56 ; 2.57 ; 3.85 ; 1.0 ; 2.1 ; 22694.704171610498 ; 7.1254958152623225 ; 115.771 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

**FREE** ; 50.43486 ; 3.05734 ; 62497 ; LENS ; <https://www.allocine.fr/> ; 1593 ; 29 ; 1921751 ; 40.78 ; 2.18 ; 3.28 ; 1.0 ; 2.68 ; 13576.048222272546 ; 4.43082513781741 ; 0.0 ; 0.0 ; 0.0 ; 429.376 ; 0.0 ; 0.0

**SFR** ; 46.15188 ; -1.14069 ; 17300 ; LA ROCHELLE ; <https://www.service-public.fr/> ; 524 ; 123 ; 1115732 ; 47.31 ; 2.05 ; 3.08 ; 1.0 ; 3.3 ; 174586.3200622886 ; 3.279724978627303 ; 4028.48056147116 ; 0.0 ; 0.0 ; 0.0 ; 0.0 ; 0.0

# Self Organizing Map (example for 2021)

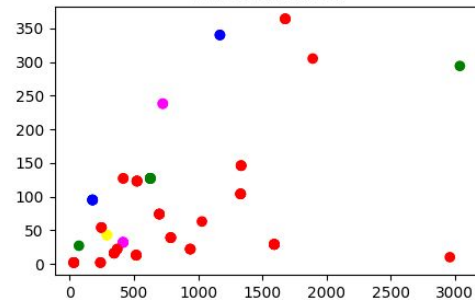
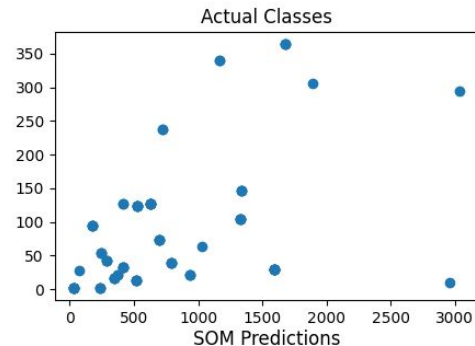


3 clusters

5 clusters

*Interpretation seems difficult!*

**Scientific question:** how such map can help in defining a 'weather situation'?



# 3. Conclusion

# Future directions

- The EcolIndex: a good starting point understanding of the phenomena and issues related to the role of digital technology in global warming;
- We need to understand better the relationships between the different high-level models of 3-tier architecture, server or client, and the field analysis of the life cycle of a digital product or equipment (LCA) or the field of "best practices";
- Introducing many attributes in environmental impact models: **immerse the problem in a machine learning formulation  $\Leftrightarrow$  weather of web requests** ;
- Context of the "client-side web programming" vision.

# Contribute:

## <https://github.com/christophe-cerin/Ecoindex-Revisited>

The screenshot shows the GitHub repository page for `christophe-cerin/Ecoindex-Revisited`. The repository is public and has 1 contributor. The README.md file is displayed, showing the project's introduction and development goals. The introduction states that the project revisits the calculation method of the Ecoindex metric, which is used to evaluate the absolute environmental performance of a given URL. The development goals include following the evolution of web requests from a carbon footprint point of view and computing the score based on new hypotheses.

### EcoIndex-Revisited

#### Introduction

In this project, we revisit the calculation method of the Ecoindex metric. This metric has been proposed to evaluate its absolute environmental performance from a given URL using a score out of 100 (higher is better). Our motivation comes from the fact that the calculation is based on both prior quantile calculations and weightings. We propose keeping only the weighting mechanism corresponding to documented and regularly available figures of the proportional breakdown of ICT's carbon footprint.

This way, we will be able to follow, from year to year, the evolution of web requests from a carbon footprint point of view. For a URL, our new calculation method takes as parameters three weights and the three typical values of the Ecoindex (DOM size, number of HTTP/HTTPS requests, KB transferred) and returns an environmental performance score.

We develop several ways to compute the score based on our new hypothesis, either using learning techniques (Locality Sensitive Hashing, K Nearest Neighbor) or matrix computation constitutes the project's first contribution. The second contribution corresponds to an experimental

The screenshot shows the GitHub repository page for `christophe-cerin/Ecoindex-Revisited`. The repository is public and has 1 branch and 0 tags. The file list shows the following files and their commit history:

File	Commit Message	Commit Hash	Time Ago	Commits
LICENCE	Create LICENCE	9f78666	16 hours ago	28
README.md	Update README.md		3 months ago	
analysis_mj.ipynb	Add files via upload		2 days ago	
analysis_mj.pdf	Add files via upload		16 hours ago	
collinearity.py	Add files via upload		16 hours ago	
lsh.py	Add files via upload		3 months ago	
random_projection.py	Add files via upload		3 months ago	
requirements.txt	Add files via upload		3 months ago	
test_ecoindex.py	Add files via upload		2 days ago	
url_4ecoindex_dataset.csv	Add files via upload		3 months ago	

The README.md file is also visible, showing the project's introduction and development goals. The introduction states that the project revisits the calculation method of the Ecoindex metric, which is used to evaluate the absolute environmental performance of a given URL. The development goals include following the evolution of web requests from a carbon footprint point of view and computing the score based on new hypotheses.

### EcoIndex-Revisited

Thank you...  
and mind your  
HTTP requests!

*Special thanks:* Mathilde, Laurent,  
and Denis

christophe.cerin@univ-paris13.fr  
<http://lipn.univ-paris13.fr/~cerin/>

