|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | |  |  | |
|  | |  |  | |
|  | | PROJECT |  | |
| **EMISSION TRACKER** | | | | |
|  | | | | |
| CLASS | EXPLORATIVE POJECT | | | |
| CONFIDENTIALITY LEVEL | TEAM | | | |
| STATUS | FIRST DRAFT | | | |
|  |  | | |  |
|  |  | | | Last update |
| AUTHOR | MG | | |  |
| REVIEWER |  | | |  |
| APPROVED BY |  | | |  |

ABSTRACT

This explorative project focuses on the assessment and dissemination of Carbon balance for what concerning SISAL INNOVATION LAB processes.

The amount of produced carbon dioxide is measured for a set of chosen projects. Data are reported by an interactive dashboard.

NDA :

Table of contents

[1 Rationale 3](#_Toc153545692)

[2 CodeCarbon 3](#_Toc153545693)

[2.1 Overview 3](#_Toc153545694)

[2.2 Implementation 3](#_Toc153545695)

[2.3 Output 4](#_Toc153545696)

[3 Dashboard 6](#_Toc153545697)

[3.1 Overview 6](#_Toc153545698)

[3.2 Main page components 6](#_Toc153545699)

[3.3 Aerial view 8](#_Toc153545700)

[Code requirements 9](#_Toc153545701)

[Figure 1 Example of single project time line 7](#_Toc153545702)

[Figure 2 Example of multiple projects timeline 7](#_Toc153545703)

[Figure 3 Example of project data 8](#_Toc153545704)

[Figure 4 Thematic mapping 8](#_Toc153545705)

[Figure 5 Animation 9](#_Toc153545706)

# Rationale

The idea behind this explorative project is to assess the impact on carbon balance of our use of the infrastructure to run models.

To cope with the task, we leveraged on the library CodeCarbon[[1]](#footnote-1) which is a lightweight software package that seamlessly integrates into our Python codebase. The package estimates the amount of carbon dioxide (CO2) produced by the computing resources used to execute the code.

The collected data are then used to populate a dashboard, designed to ease the findings interpretation and to provide a comprehensive look at the actual status of our computing impact in terms of carbon balance.

# CodeCarbon

## Overview

CodeCarbon uses CO₂-equivalents [CO₂eq], which is a standardized measure used to express the global warming potential of various greenhouse gases: the amount of CO₂ that would have the equivalent global warming impact. For computing, which emits CO₂ via the electricity it is consuming, carbon emissions are measured in kilograms of CO₂-equivalent per kilowatt-hour.

Carbon dioxide (CO₂) emissions are the product of two main factors :

* : Carbon Intensity of the electricity consumed for computation: quantified as g of CO₂ emitted per kilowatt-hour of electricity.
* : Energy Consumed by the computational infrastructure: quantified as kilowatt-hours.

Carbon dioxide emissions (CO₂eq) can then be calculated as

Carbon Intensity of the consumed electricity is calculated as a weighted average of the emissions from the different energy sources that are used to generate electricity, including fossil fuels and renewables (Table 1). CodeCarbon links the fossil fuels: coal, petroleum, and natural gas to specific carbon intensities. Renewable sources are taken into account by knowing the actual mix of energy sources that likely the computing facility uses based on its location.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Energy Source | Coal | Oil | Nat. Gas | Geoth.. | Hydro | Nuclear | Solar | Wind |
| Carbon Intensity  (kg/MWh) | 995 | 816 | 743 | 38 | 26 | 29 | 48 | 26 |

Table 1 Carbon intensity vs energy source

More detail about the library and installation instructions are available in the library website

## Implementation

Implementation of CodeCarbon is straightforward by using decorators as in the snippet below

from codecarbon import track\_emissions

from sklearn.model\_selection import train\_test\_split

@track\_emissions(project\_name='XGBoost model', output\_file='yearly\_log.csv')

def split\_data(X, y):

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

    return X\_train, X\_test, y\_train, y\_test

Snippet 1 : example of Emission Tracker implementation

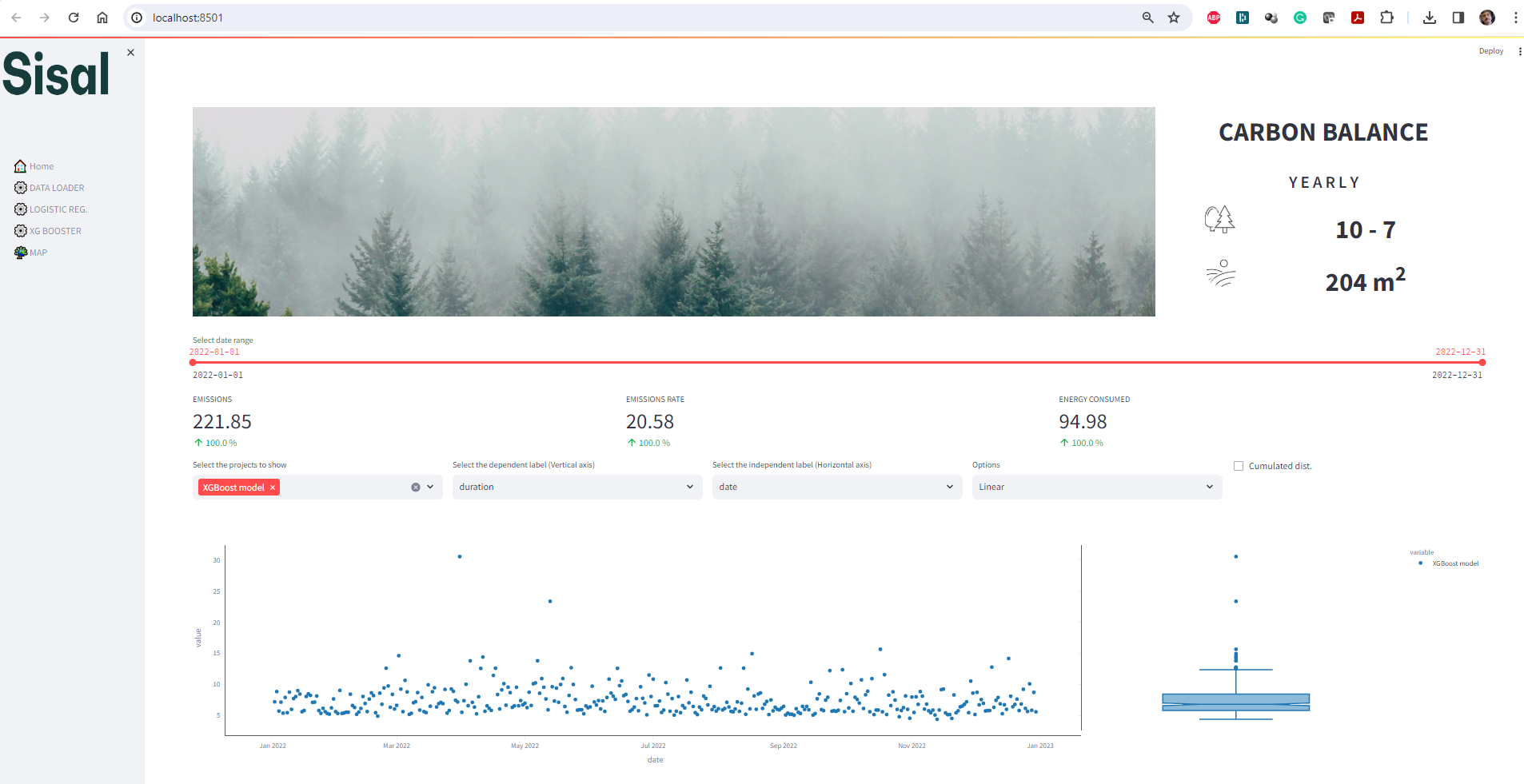
## Output

CodeCarbon output is a standard, self-explanatory, textual file that can be easily assimilated by Pandas. CSV headers are shown in Table 2

| **Field** | **Description** |
| --- | --- |
| timestamp | Time of the experiment in %Y-%m-%dT%H:%M:%S format |
| project\_name | Name of the project, defaults to codecarbon |
| run-id | id of the run |
| duration | Duration of the compute, in seconds |
| emissions | Emissions as CO₂-equivalents [CO₂eq], in kg |
| emissions\_rate | emissions divided per duration, in Kg/s |
| cpu\_power | CPU power (W) |
| gpu\_power | GPU power (W) |
| ram\_power | RAM power (W) |
| cpu\_energy | Energy used per CPU (kWh) |
| gpu\_energy | Energy used per GPU (kWh) |
| ram\_energy | Energy used per RAM (kWh) |
| energy\_consumed | sum of cpu\_energy, gpu\_energy and ram\_energy (kWh) |
| country\_name | Name of the country where the infrastructure is hosted |
| country\_iso\_code | 3-letter alphabet ISO Code of the respective country |
| region | Province/State/City where the compute infrastructure is hosted |
| on\_cloud | Y if the infrastructure is on cloud, N in case of private infrastructure |
| cloud\_provider | One of the 3 major cloud providers, aws/azure/gcp |
| cloud\_region | Geographical Region for respective cloud provider,  examples us-east-2 for aws, brazilsouth for azure, asia-east1 for gcp |
| os | os on the device  example Windows-10-10.0.19044-SP0 |
| python\_version | example 3.8.10 |
| cpu\_count: | number of CPU |
| cpu\_model | example Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz |
| gpu\_count | number of GPU |
| gpu\_model | example 1 x NVIDIA GeForce GTX 1080 Ti |
| longitude | Longitude, with reduced precision to a range of 11.1 km / 123 km².  This is done for privacy protection. |
| latitude | Latitude, with reduced precision to a range of 11.1 km / 123 km².  This is done for privacy protection. |
| ram\_total\_size | total RAM available (Go) |
| Tracking\_mode: | machine or process``(default to ``machine) |

Table 2 Logged data fields

# Dashboard



## Overview

The dashboard is created with the aim of ease CodeCarbon output and provide synthetic information about the carbon balance.

The sidebar also provides details about each monitored project and an aerial view of “*TreeFarming*” project effectiveness

The *TreeFarming* project is a mock-up of a possible socially oriented initiative that could involve schools, agencies, and/or businesses actually planting the trees needed to offset the emitted greenhouse gas estimated by this project.

## Main page components

The main page is designed to offer the user some level of interactivity.

A dual tail time slider allows controlling the data range the user wants to inspect. Four dropboxes allows the selection of which project has to be displayed and configure the chart axis. The rightmost dropbox is devoted to select the chart type.

The chart below adapts to the user’s selction.

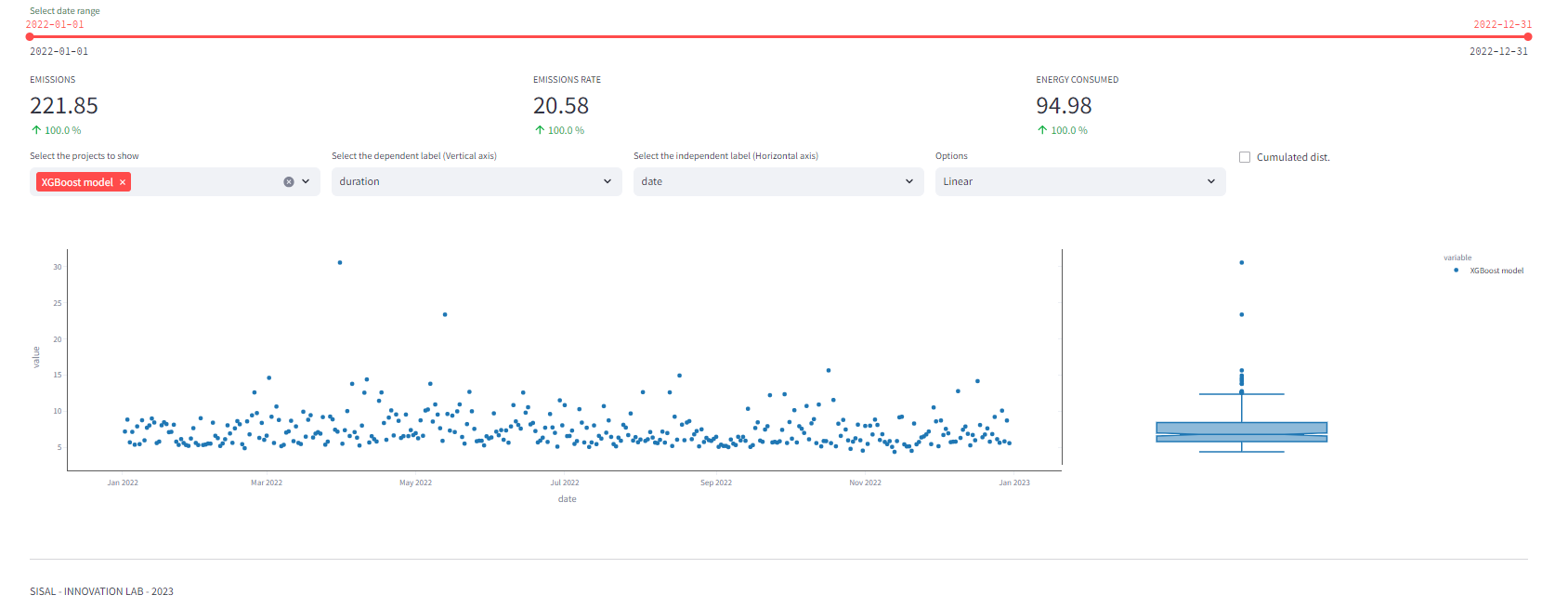


Figure 1 Example of single project time line

Figure 1 and Figure 2 show two different possibilities of a timeline chart.

On the right side of each chart, a box plot is shown and it can be used to infer synthetically the relationships between the different project (Figure 2).

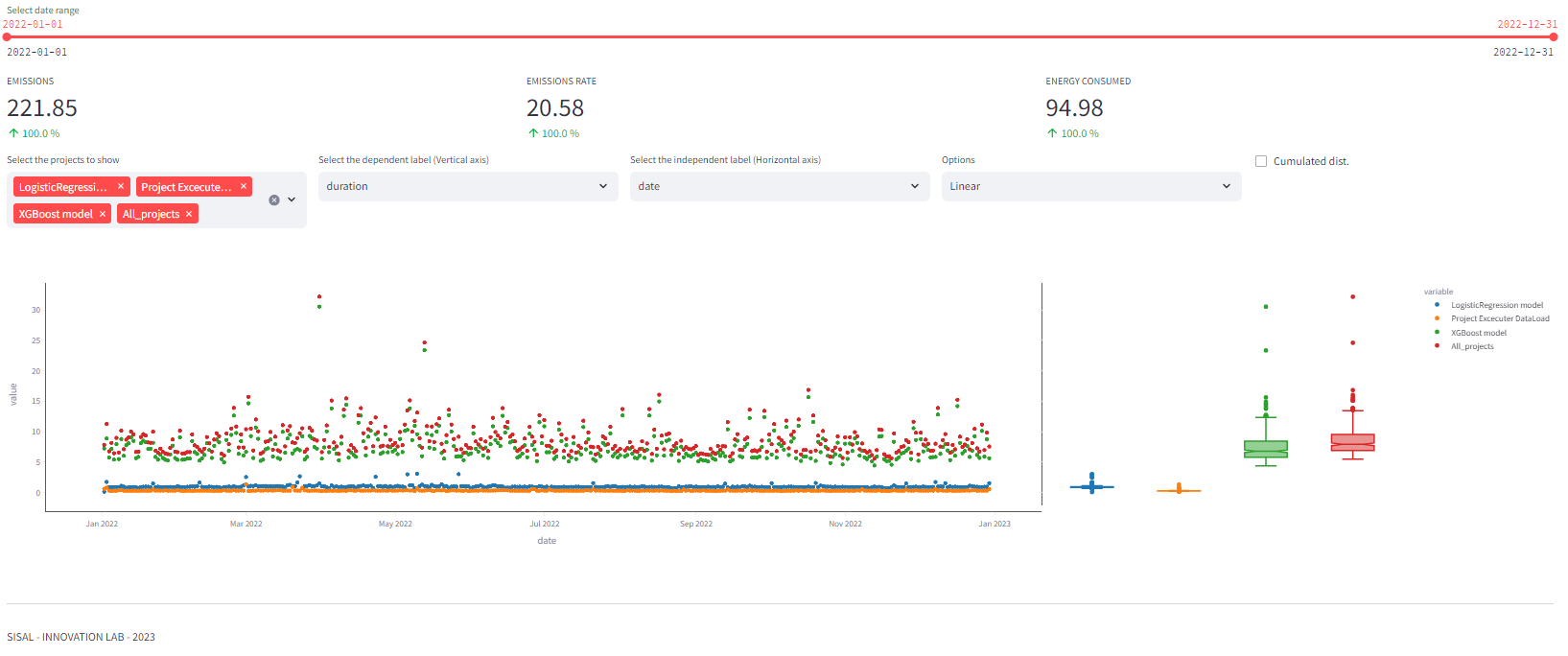


Figure 2 Example of multiple projects timeline

|  |  |
| --- | --- |
| Each project data can be accessed by the listed pages in the sidebar  as in the picture aside. LMB click on the project name pops the corresponding as in the example of Figure 3  From the project page the user has the ability to inspect the actual data and also download local copy.  A fully customizable chart utility is also available.  The chart can be manipulated using the embedded tool popping up when the mouse hoovers on the chart’s area. |  |

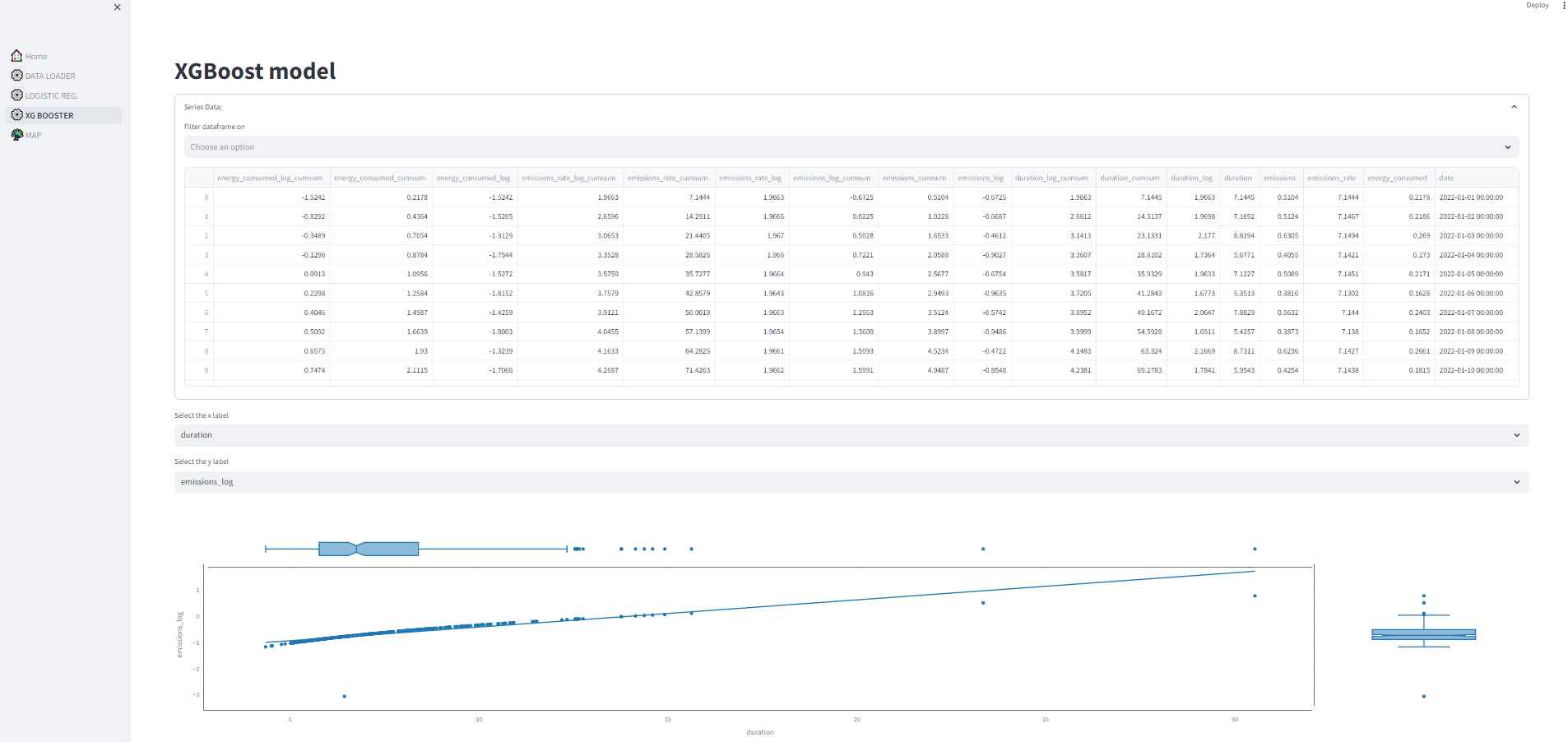


Figure 3 Example of project data

## Aerial view

The aerial view is a mock-up for a potential social initiative targeting schools, agencies, volunteer organizations or companies that join an actual planting trees programme.

The idea is to funding those entities and get feedback about the actual tree farming activity with the intention of disseminate the efficiency assessment

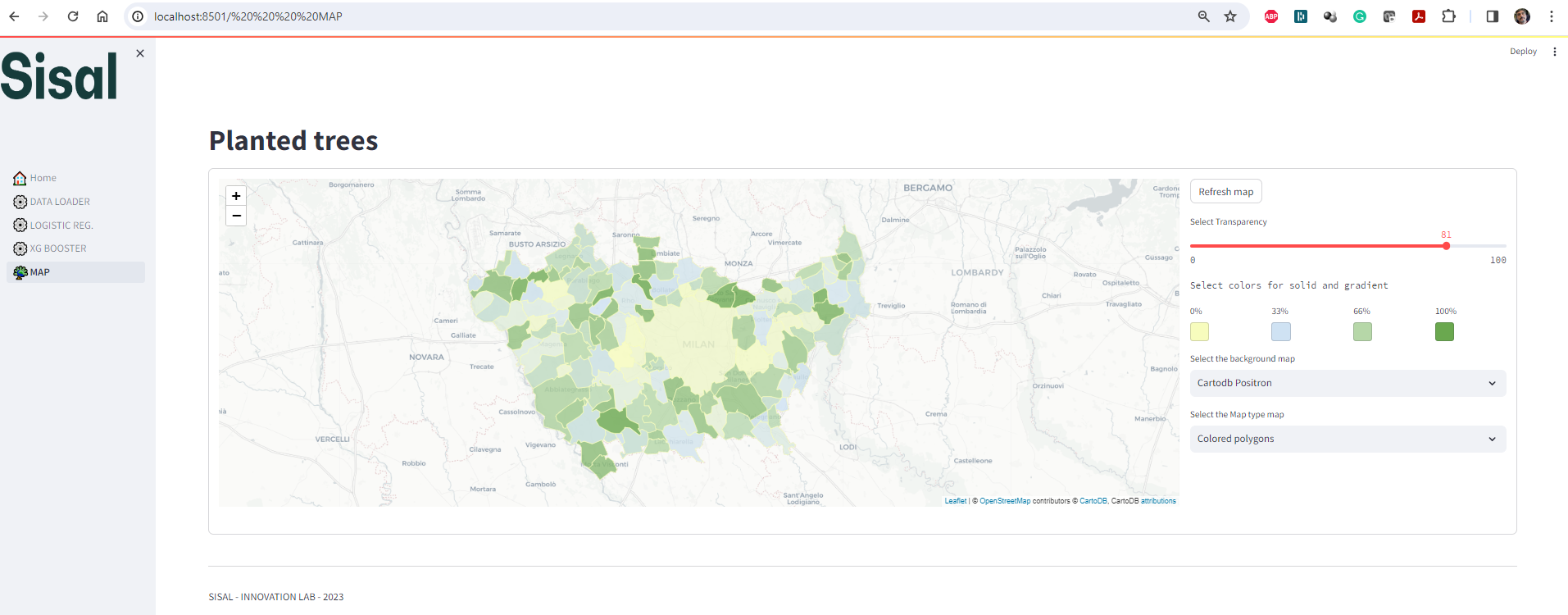


Figure 4 Thematic mapping

The map (Figure 4) is encapsulated in form: colours and features can be selected by the controls on the right panel. After altering any feature the user is requested to manually refresh the map.

The coloured polygons reflect the actual efficiency of the tree-farming activity as percentage of completion. Deep green means the task for the given municipality is completed.

The colours’ palette is a four nodes linear gradient with breakpoints at 33% and 66% &. The colours corresponding the break points are fully customizable.

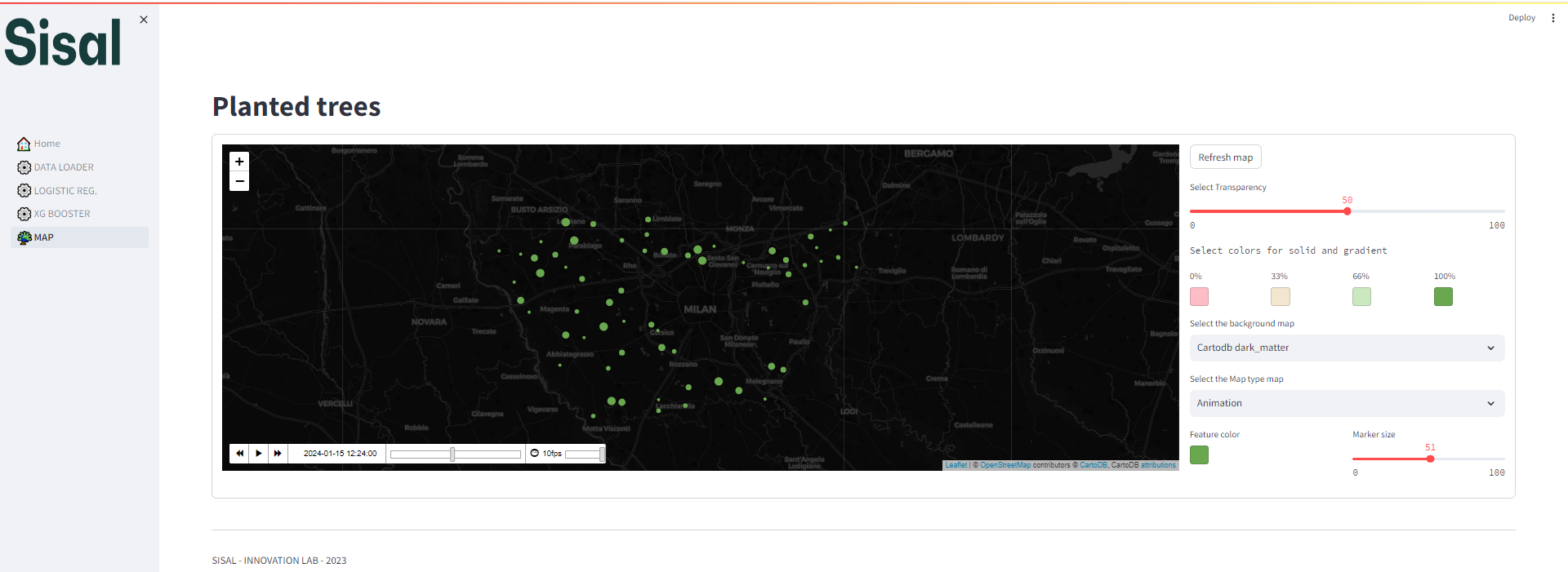


Figure 5 Animation

Figure 5 show the animation of same data used to create the thematic map of Figure 4.

The dashboard, when the animation chart is selected, add other controls to the right-side panel, such as item’s colour and scale. The actual item’s size is proportional to the farming activity level of completion for each district.

Animation is also controllable using the dedicated toolbox located at the bottom left corner of the map.

Code requirements

The dashboard is created using Streamlit (https://streamlit.io/). Streamlit is a Python library to ease the creation of webapps. Details and installation instructions are available at <https://docs.streamlit.io/library/get-started>

Here follows the list of required libraries.

# data preparation

import yaml, datetime

import base64

import pandas as pd

import numpy as np

#streamlit

import streamlit as st

from st\_clickable\_images import clickable\_images

from streamlit\_extras.switch\_page\_button import switch\_page

from streamlit\_extras.dataframe\_explorer import dataframe\_explorer

#charting

import plotly.figure\_factory as ff

import plotly.express as px

1. https://codecarbon.io/ [↑](#footnote-ref-1)