TP 2 – Measuring Carbon Emissions in Data pipelines using CodeCarbon

Case Study: Books & Reviews Data Processing

---

1. Learning Objectives

At the end of this practical session, you will be able to:

1. Measure the carbon emissions of Python data pipelines using CodeCarbon.
2. Compare Pandas vs. PySpark (local mode, but you can use Google Collab) in terms of runtime, energy, and CO_2 footprint.
3. Understand trade-offs between computation efficiency and sustainability.
4. Propose improvements for greener data engineering workflows.

---

2. Context

You are a **Data Engineer** at a company analyzing millions of book reviews.
You need to:

- Merge metadata from **books.csv** with reviews from **reviews.csv**.
- Perform data cleaning, aggregation, and analysis.
- Measure and compare emissions between:
  - a **Pandas-based pipeline** (single machine)
  - a **Spark-based pipeline** (distributed execution, local mode, you can use Google Colab)

Your manager wants not only accuracy and performance, but also **sustainability metrics**.

---

3. Datasets Overview

books.csv

Column	Description
Title	Book title
Description	Short summary
Authors	List of authors
Publisher	Publishing company
PublishedDate	Date of publication
Categories	Book categories
RatingsCount	Number of ratings

reviews.csv

Column	Description
Id	Book ISBN or unique ID
Title	Title of the book
Price	Price (optional)
User_id	Reviewer ID
profileName	Reviewer name
review/score	Rating (1–5)
review/time	Timestamp
review/summary	Short summary
review/text	Full review content

---

4. Experimental Design

Goal

Compare the **CO_2 emissions** and **execution times** for equivalent data pipelines written in **Pandas** and **PySpark**.

Pipeline steps (identical for both frameworks)

1. **Load data** from CSV files.
2. **Clean data:**
   - Handle missing values.
   - Normalize authors and categories.
3. **Join datasets** on title.
4. **Compute metrics:**
   - Average rating per author.
   - Number of reviews per publisher.
   - Top 10 most-reviewed categories.
5. **Text processing:**
   - Compute average review length.
   - Count most frequent keywords.
6. **Save final results** to CSV.

---

5. Tasks

Task 1 — Pandas baseline

1. Implement the full data pipeline using Pandas.
2. Wrap the entire script with **CodeCarbon's EmissionsTracker**.
3. Log:
    - Duration (seconds)
    - CPU usage
    - CO_2 emitted (kg)
    - Energy consumed (kWh)

Deliverable: `emissions_pandas.csv`

Task 2 — Spark (local mode)

1. Reproduce the same logic using PySpark DataFrames.
2. Use the same transformations and join keys.
3. Measure emissions using CodeCarbon again.

Deliverable: `emissions_spark.csv`

Task 3 — Comparison and Analysis

Create a summary table:

Step	Framework	Duration (s)	Energy (kWh)	CO_2 (kg)	Memory (MB)

Then:

- Visualize emissions and duration per framework (bar chart).

- Compute efficiency ratio:
  Efficiency=Data processed (rows)CO$_2$ emitted (g)\text{Efficiency} = \frac{\text{Data processed (rows)}}{\text{CO$_2$ emitted (g)}}Efficiency=CO$_2$ emitted (g)Data processed (rows)
- Discuss:
  - Which framework is faster?
  - Which one emits less CO$_2$?
  - Does parallelism always mean greener computation?

---

6. Task 4 — Eco-Design Experiment

You must choose **one optimization strategy** and re-measure emissions:

- Reduce dataset size (sample 10% / 50%).
- Filter out unused columns.
- Cache intermediate results (Spark).
- Vectorize text cleaning functions (Pandas).

Compare before/after emissions and explain results.

---

7. Deliverables

Each team must submit:

- `tp_codecarbon_books.ipynb`
- `emissions_pandas.csv`
- `emissions_spark.csv`

- One figure comparing emissions and runtime for each framework

- 10-line reflection : On comparing the usage of Pandas & PySpark.