Sistemi e Architetture per Big Data

1° Progetto - Electricity Maps

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Giuseppe Marseglia, matricola n. 0350066

Introduzione

Electricity Maps

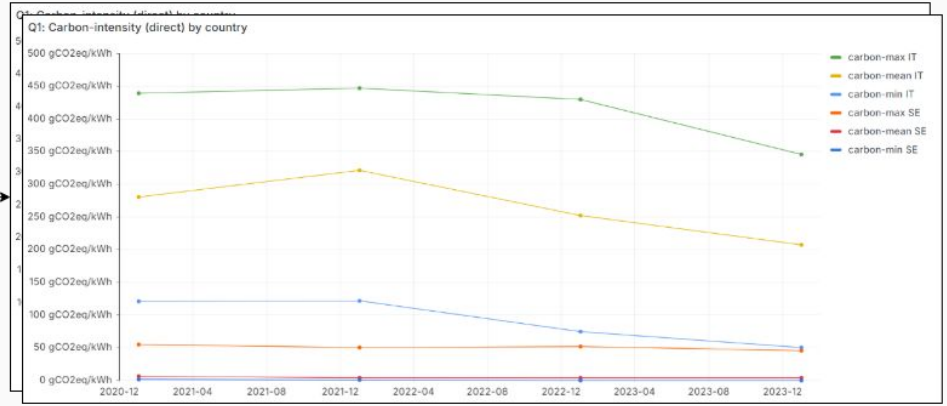
```
Datetime (UTC),Country,Zone name,Zone id,Carbon intensity gCO₂eq/kWh ...
2021-01-01 00:00:00,Italy,Italy,IT,317.69,401.21,38.61,36.51,,false,
2021-01-01 01:00:00,Italy,Italy,IT,320.21,403.67,38.89,37.38,,false,
2021-01-01 02:00:00,Italy,Italy,IT,314.39,397.0,40.11,37.97,,false,
2021-01-01 03:00:00,Italy,Italy,IT,311.25,393.58,40.54,37.95,,false,
```

Q_1

```
date,country,carbon-mean,carbon-min,carbon-max,cfe-mean,cfe-min,cfe-max
2021,IT,280.08424543378885,121.24,439.06,46.30593150684933,15.41,77.02
2022,IT,321.617976027397,121.38,447.33,41.24412671232864,13.93,77.44
2023,IT,251.81946461187323,74.44,429.93,51.596057077625616,20.39,85.02
2024,IT,207.29918943533622,50.18,345.65,57.431828324225826,20.9,90.26
...
```

Q_2

```
date,carbon-intensity,cfe
2024_05,158.24088709677372,68.98973118279565
2024_04,170.67088888888884,66.25395833333332
2024_06,171.9787916666666,65.48779166666664
2024_03,192.85387096774187,60.91955645161283
2023_05,203.49448924731166,59.87700268817201
```

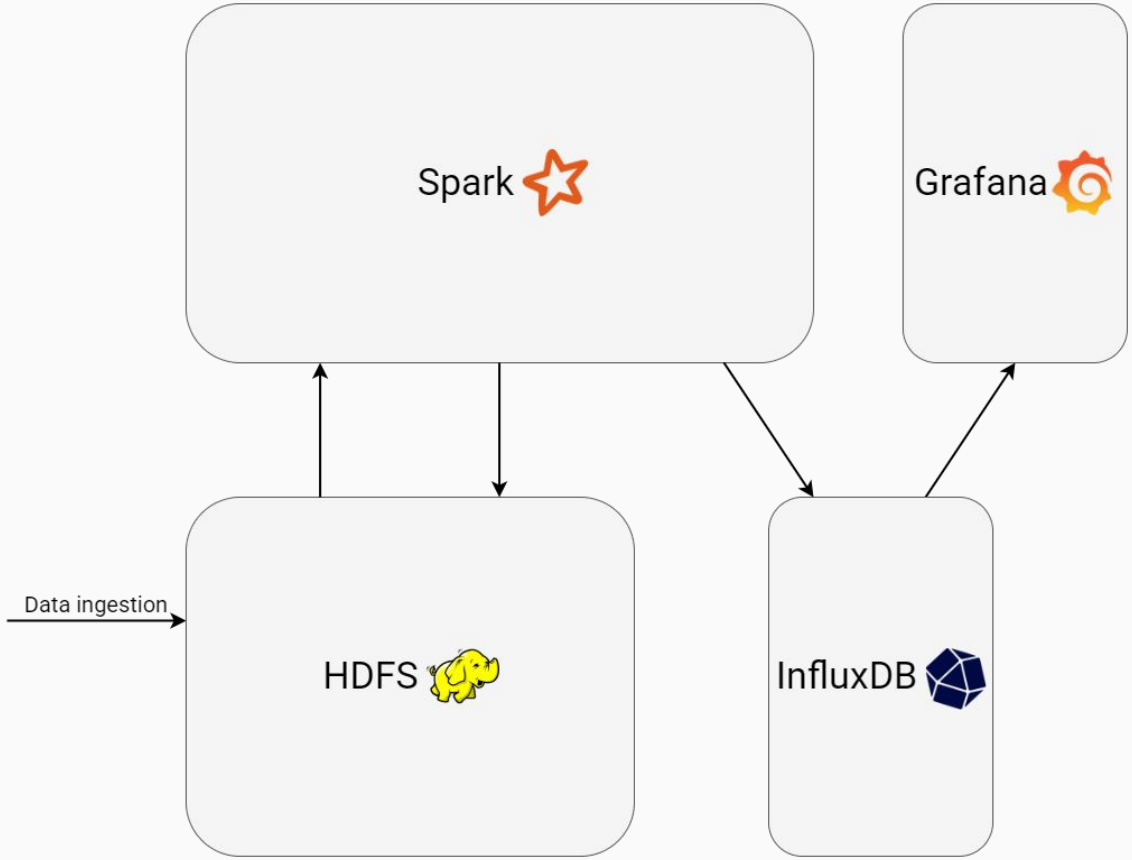


Architettura

HDFS e InfluxDB per **storage**.
Spark per **processamento batch**.
Grafana per la **visualizzazione** dei dati.

Le frecce rappresentano flussi di dati.

Data ingestion automatica, ma senza pre-processamento e framework specifico.



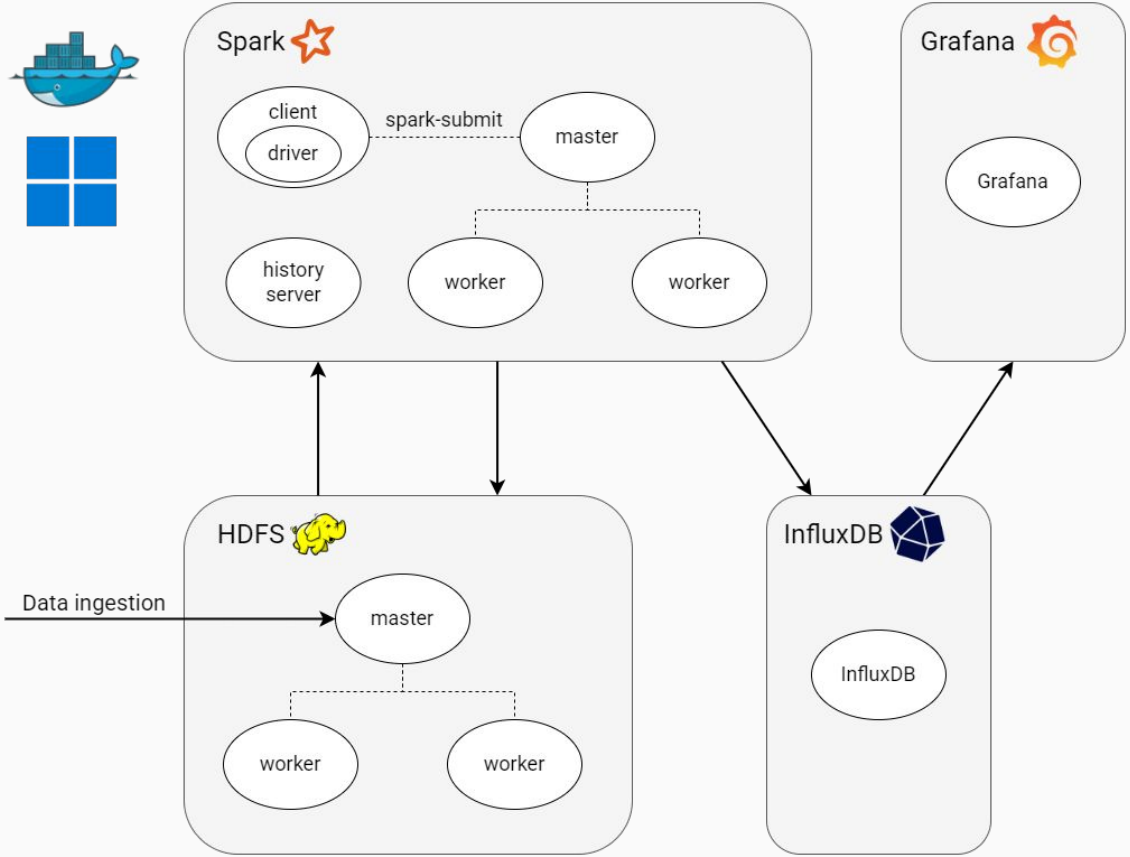
Deployment

Deployment tramite **Docker Compose**, su un nodo singolo con Windows 11 e Windows Linux Subsystem (**WSL**).

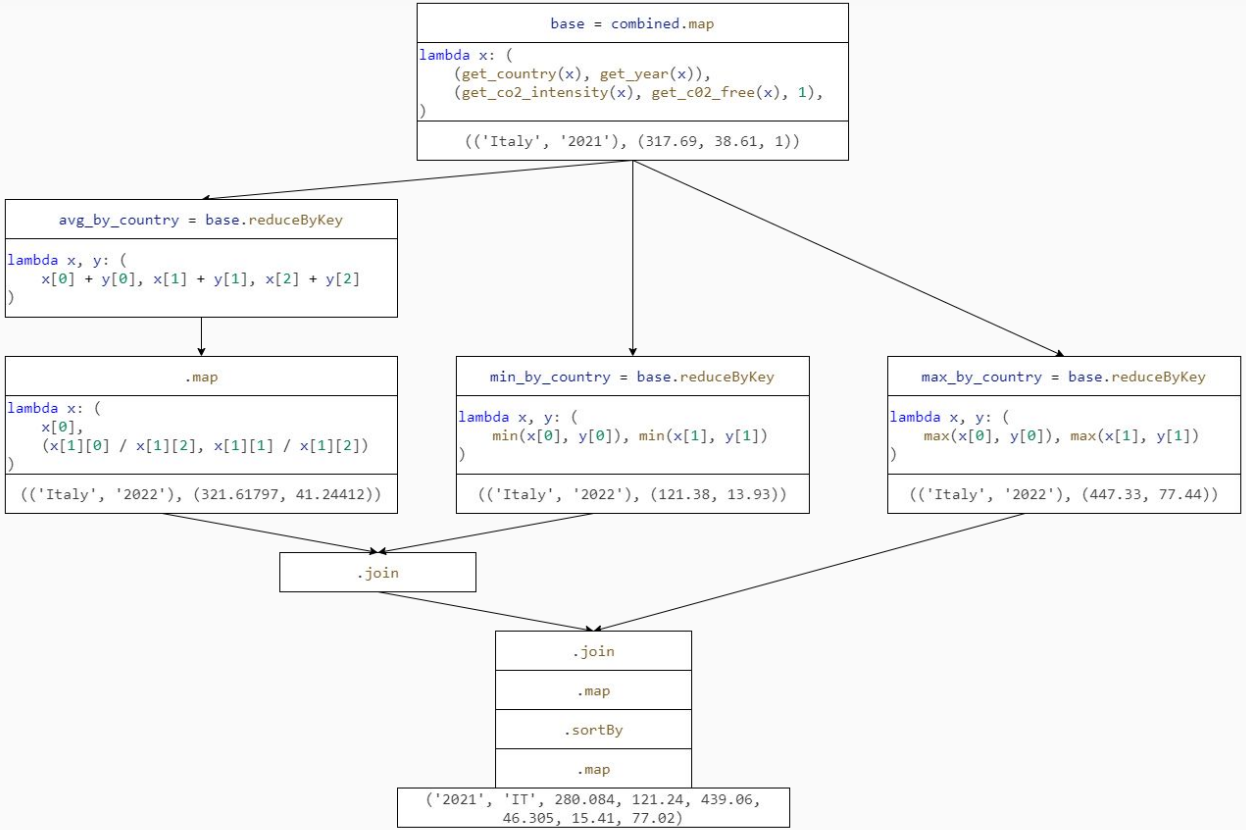
HDFS e Spark in modalità **cluster**. InfluxDB e Grafana in modalità **standalone**.

Data ingestion è eseguita dal master di HDFS che comunica con il file system locale tramite Bind Mounts.

Il client è un container di Spark.



Confronto Query 1



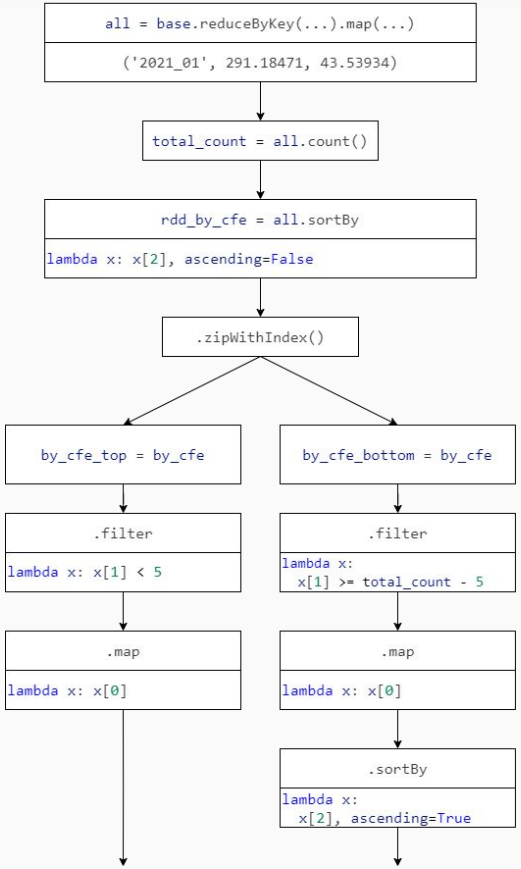

```
result = base.groupBy("Country", "Year").agg

F.avg("CO2_intensity_direct").alias(...),
F.min("CO2_intensity_direct").alias(...),
F.max("CO2_intensity_direct").alias(...),
F.avg("Carbon_free_energy_percent").alias(...),
F.min("Carbon_free_energy_percent").alias(...),
F.max("Carbon_free_energy_percent").alias(...),
```

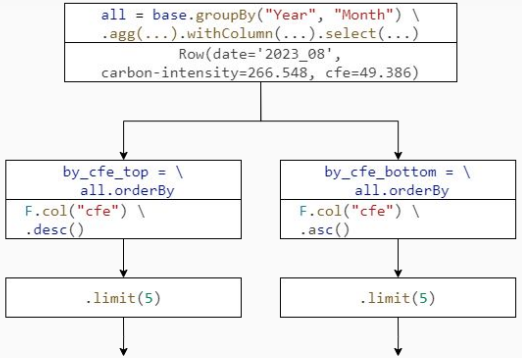
```
Row(date='2021', country='IT',
carbon-mean=280.08424, carbon-min=121.24,
carbon-max=439.06, cfe-mean=46.30593,
cfe-min=15.41, cfe-max=77.02)
```

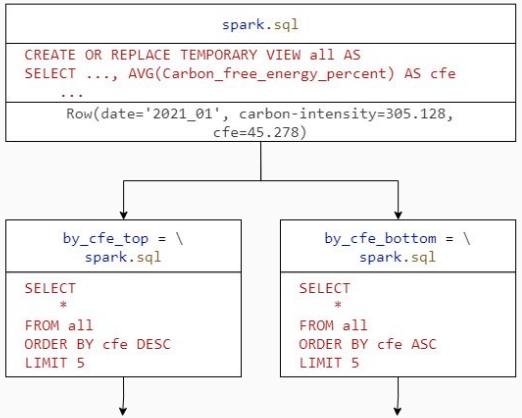
```
result = spark.sql

SELECT
    YEAR(Datetime) AS year, Country AS country,
    AVG(CO2_intensity_direct) AS `carbon-mean`,
    MIN(CO2_intensity_direct) AS `carbon-min`,
    MAX(CO2_intensity_direct) AS `carbon-max`,
    AVG(Carbon_free_energy_percent) AS `cfe-mean`,
    MIN(Carbon_free_energy_percent) AS `cfe-min`,
    MAX(Carbon_free_energy_percent) AS `cfe-max`
FROM carbon_data
GROUP BY country, year
ORDER BY country, year
```

```
Row(year=2021, country='Italy',
carbon-mean=280.084, carbon-min=121.24,
carbon-max=439.06, cfe-mean=46.305,
cfe-min=15.41, cfe-max=77.02)
```

Confronto Query 2







Introduzione

Valutare l'impatto sul tempo di processamento di:
1) Le **differenti API**: RDD, DataFrame e SparkSQL.
2) I **differenti formati**: CSV, Avro e Parquet.
3) **L'uso del caching**.
Per ogni esperimento è stato condotto anche uno studio dell'impatto della **variazione della grandezza del dataset***.
Questo porta il totale a **6 esperimenti condotti**.

*: Il dataset più piccolo conta 35064 entry per Italia e Svezia, mentre quello più grande conta 210384 entry per Italia, 140256 entry per Svezia. Il dataset per la Q1 aumenta di 5 volte, mentre quello per la Q2 aumenta di 6 volte.

Le misurazioni sono:
- **Raccolte dentro il driver**, utilizzando le funzioni built-in di Python.
- Calcolati da prima dell'inizio della query fino a **dopo una `.collect()`**.
- Salvati e analizzati direttamente in InfluxDB tramite **query in Flux**.

Esperimento 1: API

Setup

- Formato CSV.
- Caching, dove possibile.

'Baseline' è un'implementazione in Python nativo, senza nessun framework aggiuntivo.

Considerazioni

- La baseline è notevolmente più veloce delle implementazioni in Spark, ma anche la più sensibile alla grandezza del dataset.
- L'implementazione con RDD è la più veloce tra quelle di Spark.
- Le implementazioni di Spark scalano meglio.

Query	API	Dataset	Run	Media in s	StdDev
1	baseline	country	10	0.772	0.6
1	baseline	region	10	3.294	0.812
1	rdd	country	10	7.262	0.568
1	rdd	region	10	7.654	0.295
1	sql	country	10	12.607	0.471
1	sql	region	10	14.394	0.818
1	df	country	10	13.535	0.795
1	df	region	10	15.408	0.888
2	baseline	country	10	0.303	0.034
2	baseline	region	10	1.523	0.064
2	rdd	country	10	7.348	0.132
2	rdd	region	10	7.797	0.475
2	sql	country	10	13.372	0.431
2	sql	region	10	15.709	0.65
2	df	country	10	17.448	0.549
2	df	region	10	19.603	0.805

TABLE II: Risultati dell'esperimento 1.b

Esperimento 2: Formato

Setup

- No caching.
- In 2.a: Dataset per paese.
- In 2.b: SparkSQL.

Considerazioni

- Avro e Parquet hanno sempre performance migliori di CSV.
- Avro e Parquet scalano meglio di CSV.
- Avro e Parquet hanno un impatto positivo anche per lo storage, ad esempio per Q2:
  - Dataset piccolo, da 2.4 MB in CSV a 996 KB per Avro e 674 KB per Parquet.
  - Dataset grande, da 19 MB in CSV a 6.1 MB per Avro e 2.6 MB per Parquet.

Query	API	**Formato**	Run	Media in s	StdDev
1	df	avro	10	9.855	0.409
1	df	parquet	10	10.977	0.396
1	df	csv	10	13.948	0.765
1	sql	avro	10	9	0.181
1	sql	parquet	10	9.708	0.437
1	sql	csv	10	12.607	0.471
2	df	avro	10	11.54	1.091
2	df	parquet	10	11.614	0.69
2	df	csv	10	13.713	0.843
2	sql	parquet	10	11.565	0.848
2	sql	avro	10	11.875	0.441
2	sql	csv	10	13.372	0.431

TABLE III: Risultati dell'esperimento 2.a

2	csv	country	10	13.372	0.431
2	csv	region	10	15.709	0.65
2	avro	country	10	11.875	0.441
2	avro	region	10	12.186	0.596
2	parquet	country	10	11.565	0.848
2	parquet	region	10	12.626	0.965

TABLE IV: Risultati dell'esperimento 2.b

Esperimento 3: Cache

Setup

- Formato CSV.
- In 3.b: RDD.

Considerazioni

- L'effetto del caching dipende dalla specifica implementazione della query.
- Quando l'effetto è positivo, il caching può aiutare a ridurre il degrado delle performance.

Query	API	Caching	Run	Media in s	StdDev
1	rdd	TRUE	10	7.262	0.568
1	rdd	FALSE	10	7.398	0.421
2	df	FALSE	10	13.713	0.843
2	df	TRUE	10	17.448	0.549
2	rdd	TRUE	10	7.348	0.132
2	rdd	FALSE	10	7.497	0.2

TABLE V: Risultati dell'esperimento 3.a

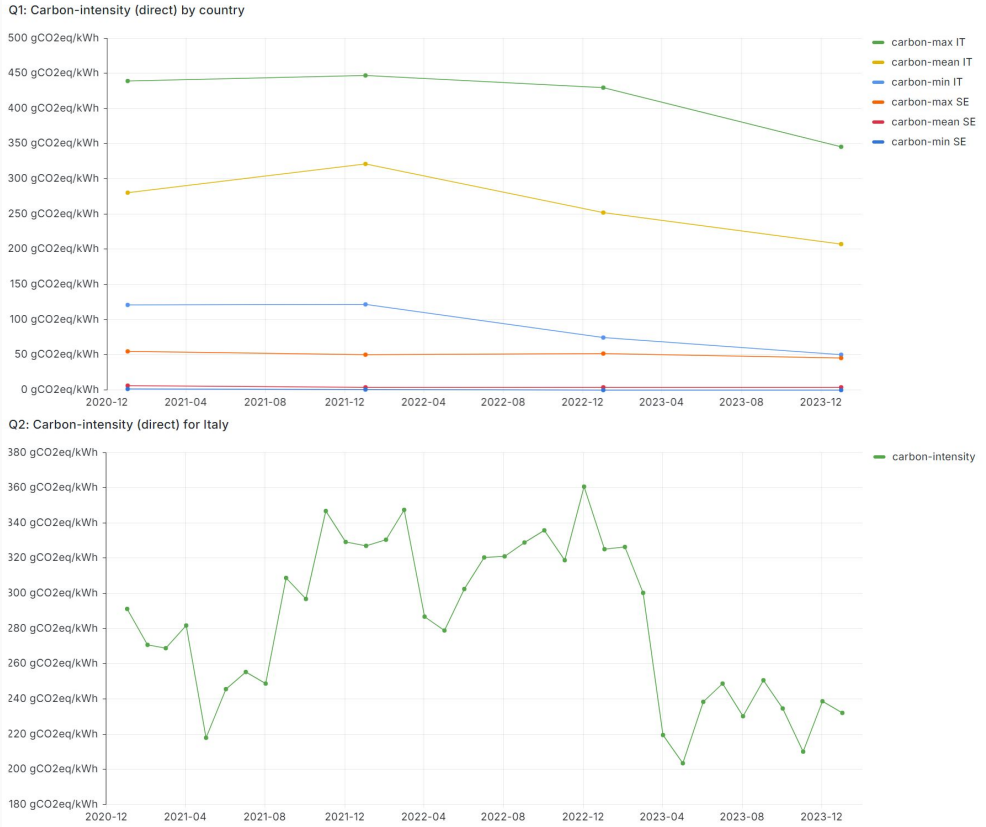
Query	Caching	Dataset	Run	Media in s	StdDev
1	FALSE	country	10	7.398	0.421
1	FALSE	region	10	8.12	0.253
1	TRUE	country	10	7.262	0.568
1	TRUE	region	10	7.654	0.295
2	FALSE	country	10	7.497	0.2
2	FALSE	region	10	7.844	0.4
2	TRUE	country	10	7.348	0.132
2	TRUE	region	10	7.797	0.475

TABLE VI: Risultati dell'esperimento 3.b

CSV e grafici


```
date,country,carbon-mean,carbon-min,
carbon-max,cfe-mean,cfe-min,cfe-max

2021,IT,280.084,121.24,439.06,46.305,
15.41,77.02
2022,IT,321.617,121.38,447.33,41.244,
13.93,77.44
2023,IT,251.819,74.44,429.93,51.596,2
0.39,85.02
...
```

```
date,    carbon-intensity,cfe
2022_12,360.51999,        35.83831
2022_03,347.35907,        35.82221
2021_11,346.72851,        33.07668
2022_10,335.78474,        39.16716
2022_02,330.48989,        38.98059
```



Links

Viene riportato il link al repository di GitHub con l'implementazione del progetto e la relazione completa.

- Link al repository su GitHub

Grazie per l'attenzione