# CARTE-Enbridge Bootcamp

Lab 4-2

# Understanding the Carbon Cost of Machine Learning

With the rise of Large Language Models, there has been a growing discussion about the climate impact of using deep learning. In this lab, we are going to explore the carbon cost of training a model. We will use the codecarbon library to measure the carbon footprint of a few different machine learning methods.

In [1]:
```
!pip install -U -q codecarbon pint transformers datasets torch "accelerate>=
```

In [2]:
```
# Check if we are running with a GPU
import torch
if torch.cuda.is_available():
    print('GPU available')
else:
    raise Exception('GPU not available - select Runtime -> Change runtime ty
```

```
GPU available
```

In [3]:
```
!codecarbon init
```

```
Welcome to CodeCarbon, here is your experiment id:
7da01208-8255-4211-92aa-4cb84f2ff963 (from ./.codecarbon.config)
```

CodeCarbon is a Python library that allows you to measure the carbon footprint of your code. It works by measuring the power consumption of your machine and estimating the carbon emissions associated with that power consumption. It generates a detailed report that includes the carbon footprint of your code, helping us to understand and compare the impact of different models.

Let's start by using CodeCarbon to investigate the impact of training a simple linear regression model. We will use the California Housing dataset from scikit-learn. This dataset contains information about housing prices in California in the 1990s. We will use the median income of the residents to predict the median house value.

In [4]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from pint import UnitRegistry
```

```
import pandas as pd
from codecarbon import EmissionsTracker
ureg = UnitRegistry()
```

In [5]:
```
# Load the data
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

Before we go any further, let's take a look at the data we're working with. It's always important to understand what we're predicting.

This housing dataset tasks us with predicting the median house value in a given area. The dataset contains 8 features:

- MedInc: median income in block
- HouseAge: median house age in block
- AveRooms: average number of rooms
- AveBedrms: average number of bedrooms
- Population: block population
- AveOccup: average house occupancy
- Latitude: house block latitude
- Longitude: house block longitude

The house values are measured in hundreds of thousands of dollars.

We will use `Mean Absolute Error` as our evaluation metric. This metric is easy to interpret, as it is in the same units as the target variable. It is also robust to outliers, which is important in this dataset.

In [6]:
```
x_train.head()
```

Out[6]:

| | MedInc | HouseAge | AveRooms | AveBedrms | Population | AveOccup | Lat |
|---|---|---|---|---|---|---|---|
| **14196** | 3.2596 | 33.0 | 5.017657 | 1.006421 | 2300.0 | 3.691814 | |
| **8267** | 3.8125 | 49.0 | 4.473545 | 1.041005 | 1314.0 | 1.738095 | |
| **17445** | 4.1563 | 4.0 | 5.645833 | 0.985119 | 915.0 | 2.723214 | |
| **14265** | 1.9425 | 36.0 | 4.002817 | 1.033803 | 1418.0 | 3.994366 | |
| **2271** | 3.5542 | 43.0 | 6.268421 | 1.134211 | 874.0 | 2.300000 | |

In [7]:
```
y_train[:5]
```

Out[7]:
```
array([1.03 , 3.821, 1.726, 0.934, 0.965])
```

Using CodeCarbon's EmissionsTracker is easy. When we want to record the cost of a specific training run, we simply wrap the training code in a with statement.

Let's train a linear regression model and see how much carbon it emits.

In [8]:
```
def report_emissions(emissions_tracker: EmissionsTracker):
    energy = emissions_tracker.final_emissions_data.energy_consumed
    energy = energy * ureg.kilowatt_hour
    carbon = emissions_tracker.final_emissions_data.emissions
    carbon = carbon * ureg.kilogram
    print(f'Carbon emitted:      {carbon.to_compact():~.2f}')
    print(f'Energy consumed:     {energy.to_compact():~.2f}')
```

In [9]:
```
%%capture

model = LinearRegression()

# Wrap the training code in a with statement
with EmissionsTracker(project_name="Linear Regression") as tracker:
    model.fit(x_train, y_train)

with EmissionsTracker(project_name="Linear Regression Prediction") as predic
    y_hat = model.predict(x_test)
```

We are going to also record the carbon cost of making a prediction with each of these models. This is for later, when we look at LLMs.

In [10]:
```
report_emissions(tracker)
print(f'Mean absolute error: {mean_absolute_error(y_test, y_hat):.2f}')
```

```
Carbon emitted:      4.07 µg
Energy consumed:     103.09 µWh
Mean absolute error: 0.53
```

Unsurprisingly, training a simple linear regression model has a very small carbon footprint. Let's see what happens when we train a more complex model. Let's train a large Random Forest model.

In [11]:
```
%%capture

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, max_depth=None, n_jobs=-1) #
with EmissionsTracker(project_name="Random Forest") as tracker:
    model.fit(x_train, y_train)

with EmissionsTracker(project_name="Random Forest Prediction") as predict:
    y_hat = model.predict(x_test)
```

In [12]:
```
report_emissions(tracker)
print(f'Mean absolute error: {mean_absolute_error(y_test, y_hat):.2f}')
```

```
Carbon emitted:      605.80 µg
Energy consumed:     15.34 mWh
Mean absolute error: 0.33
```

Our model has improved significantly, but at an increased carbon cost.

Something important to note is that we are currently running these experiments in Google Colab. The location of the computing resources powering our code has a significant impact on the carbon footprint of our code. We can check what region the code is running in, using CodeCarbon:

In [13]:
```
print(f'Region:         {tracker.final_emissions_data.region}')
print(f'Country:        {tracker.final_emissions_data.country_name}')
emissions_rate = tracker.final_emissions_data.emissions_rate * ureg.kilogram
print(f'Emissions rate: {emissions_rate.to_compact():~.2f}')
```

```
Region:         ontario
Country:        Canada
Emissions rate: 470.27 ng / Wh
```

The region you see here is variable, but it's likely to be in the US. In one test, we received the following results:

```
Region:         oregon
Country:        United States
Emissions rate: 1.82 µg / Wh
```

Unsurprisingly, if you run this code in Ontario, where we have a much higher proportion of renewable energy, the emissions rate is much lower:

```
Region:         ontario
Country:        Canada
Emissions rate: 312.58 ng / Wh
```

Another important factor to consider is the efficiency of the hardware that we're using. CodeCarbon reports the power consumption of the computing resources we are working on:

In [14]:
```
print(f'CPU Power: {tracker.final_emissions_data.cpu_power * ureg.watt:~.2f}')
print(f'RAM Power:  {tracker.final_emissions_data.ram_power * ureg.watt:~.2f}')
```

```
CPU Power: 32.50 W
RAM Power:  5.73 W
```

**Your turn**

Before we move on, let's try one more experiment. Choose a machine learning model in Scikit-Learn and train it on the California Housing dataset. Use CodeCarbon to measure the carbon footprint of your model. How does it compare to the models we've already trained? **Hint: If you aren't sure what model to use, try the Extra Random Trees model.**

In [15]:
```
# Train Extra Random Trees model:

from sklearn.ensemble import ExtraTreesRegressor
```

```
model = ExtraTreesRegressor(n_estimators=100, max_depth=None, n_jobs=-1) # U

with EmissionsTracker(project_name="Extra Random Trees") as tracker:
    model.fit(x_train, y_train)

with EmissionsTracker(project_name="Extra Random Trees Prediction") as predi
    y_hat = model.predict(x_test)

report_emissions(tracker)
print(f'Mean absolute error: {mean_absolute_error(y_test, y_hat):.2f}')
```

```
[codecarbon INFO @ 16:38:54] [setup] RAM Tracking...
[codecarbon INFO @ 16:38:54] [setup] GPU Tracking...
[codecarbon INFO @ 16:38:54] Tracking Nvidia GPU via pynvml
[codecarbon INFO @ 16:38:54] [setup] CPU Tracking...
[codecarbon WARNING @ 16:38:54] No CPU tracking mode found. Falling back on
CPU constant mode.
[codecarbon INFO @ 16:38:55] CPU Model on constant consumption mode: 11th Ge
n Intel(R) Core(TM) i5-11400 @ 2.60GHz
[codecarbon INFO @ 16:38:55] >>> Tracker's metadata:
[codecarbon INFO @ 16:38:55]   Platform system: Linux-6.2.0-36-generic-x86_6
4-with-glibc2.35
[codecarbon INFO @ 16:38:55]   Python version: 3.11.6
[codecarbon INFO @ 16:38:55]   CodeCarbon version: 2.3.1
[codecarbon INFO @ 16:38:55]   Available RAM : 15.283 GB
[codecarbon INFO @ 16:38:55]   CPU count: 12
[codecarbon INFO @ 16:38:55]   CPU model: 11th Gen Intel(R) Core(TM) i5-1140
0 @ 2.60GHz
[codecarbon INFO @ 16:38:55]   GPU count: 1
[codecarbon INFO @ 16:38:55]   GPU model: 1 x NVIDIA GeForce RTX 3060
[codecarbon INFO @ 16:38:59] Energy consumed for RAM : 0.000001 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:38:59] Energy consumed for all GPUs : 0.000001 kWh. To
tal GPU Power : 7.19155372907921 W
[codecarbon INFO @ 16:38:59] Energy consumed for all CPUs : 0.000004 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:38:59] 0.000005 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:38:59] [setup] RAM Tracking...
[codecarbon INFO @ 16:38:59] [setup] GPU Tracking...
[codecarbon INFO @ 16:38:59] Tracking Nvidia GPU via pynvml
[codecarbon INFO @ 16:38:59] [setup] CPU Tracking...
[codecarbon WARNING @ 16:38:59] No CPU tracking mode found. Falling back on
CPU constant mode.
[codecarbon INFO @ 16:39:00] CPU Model on constant consumption mode: 11th Ge
n Intel(R) Core(TM) i5-11400 @ 2.60GHz
[codecarbon INFO @ 16:39:00] >>> Tracker's metadata:
[codecarbon INFO @ 16:39:00]   Platform system: Linux-6.2.0-36-generic-x86_6
4-with-glibc2.35
[codecarbon INFO @ 16:39:00]   Python version: 3.11.6
[codecarbon INFO @ 16:39:00]   CodeCarbon version: 2.3.1
[codecarbon INFO @ 16:39:00]   Available RAM : 15.283 GB
[codecarbon INFO @ 16:39:00]   CPU count: 12
[codecarbon INFO @ 16:39:00]   CPU model: 11th Gen Intel(R) Core(TM) i5-1140
0 @ 2.60GHz
[codecarbon INFO @ 16:39:00]   GPU count: 1
[codecarbon INFO @ 16:39:00]   GPU model: 1 x NVIDIA GeForce RTX 3060
[codecarbon INFO @ 16:39:03] Energy consumed for RAM : 0.000000 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:39:03] Energy consumed for all GPUs : 0.000000 kWh. To
tal GPU Power : 0.0 W
[codecarbon INFO @ 16:39:03] Energy consumed for all CPUs : 0.000000 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:39:03] 0.000000 kWh of electricity used since the begi
nning.
```

```
Carbon emitted:      202.47 µg
Energy consumed:     5.13 mWh
Mean absolute error: 0.33
```

# Understanding the Carbon Cost of Large Language Models

Now that we have a better understanding of how CodeCarbon works, let's use it to investigate the carbon cost of using a large language model.

We are going to use HuggingFace to train a version of GPT-2 for a single epoch (i.e. one pass through the training data). We will then use the model to generate some text, and measure the carbon cost of the training and prediction steps.

While the training part of this code is quite simple, getting the data ready requires a little bit of effort. We are going to rush through it here, as it isn't the focus of this lab, but the code is commented in case you're interested.

In [16]:
```
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, Train

dataset = load_dataset("wikitext", "wikitext-2-raw-v1") # Load the raw datas
tokenizer = AutoTokenizer.from_pretrained("gpt2") # Load the tokenizer - con

# To speed up, let's use half the data
dataset['train'] = dataset['train'].select(range(0, len(dataset['train']), 2

tokenized_datasets = dataset.map(
    lambda x: tokenizer(x["text"]), # Tells this function how to use the tok
    batched=True, # Apply to groups of examples
    num_proc=4, # Use 4 cores
    remove_columns=["text"] # Remove the text column, as we don't need it an
)

block_size = 256 # The maximum number of tokens in a single input

# The main data processing function that will concatenate all texts from our
def group_texts(examples):
    # Concatenate all texts.
    concatenated_examples = {k: sum(examples[k], []) for k in examples.keys(
    total_length = len(concatenated_examples[list(examples.keys())[0]])
    total_length = (total_length // block_size) * block_size
    # Split by chunks of max_len.
    result = {
        k: [t[i : i + block_size] for i in range(0, total_length, block_size
        for k, t in concatenated_examples.items()
    }
    result["labels"] = result["input_ids"].copy()
    return result

# Finally, apply the function above to our data
lm_datasets = tokenized_datasets.map(
```

```
    group_texts,
    batched=True,
    num_proc=4,
)
```

```
/home/alex/mambaforge/envs/enbridge_pytorch/lib/python3.11/site-packages/datasets/table.py:1421: FutureWarning: promote has been superseded by mode='default'.
  table = cls._concat_blocks(blocks, axis=0)
```

Whew! With that out of the way, we can set up the actual training and measure its carbon cost.

In [17]:
```
model = AutoModelForCausalLM.from_pretrained("gpt2")

training_args = TrainingArguments(
    output_dir="./gpt2-wikitext2",
    overwrite_output_dir=True,
    num_train_epochs=1, # Train for one epoch
    per_device_train_batch_size=4,
    save_steps=10_000,
    save_total_limit=1,
    prediction_loss_only=True,
    logging_steps=1,
    logging_first_step=True,
    learning_rate=2e-5,
    weight_decay=0.01,
    report_to=None,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=lm_datasets["train"],
    eval_dataset=lm_datasets["validation"],
    data_collator=None,
)
```

```
[codecarbon INFO @ 16:39:06] [setup] RAM Tracking...
[codecarbon INFO @ 16:39:06] [setup] GPU Tracking...
[codecarbon INFO @ 16:39:06] Tracking Nvidia GPU via pynvml
[codecarbon INFO @ 16:39:06] [setup] CPU Tracking...
[codecarbon WARNING @ 16:39:06] No CPU tracking mode found. Falling back on
CPU constant mode.
[codecarbon INFO @ 16:39:07] CPU Model on constant consumption mode: 11th Ge
n Intel(R) Core(TM) i5-11400 @ 2.60GHz
[codecarbon INFO @ 16:39:07] >>> Tracker's metadata:
[codecarbon INFO @ 16:39:07]   Platform system: Linux-6.2.0-36-generic-x86_6
4-with-glibc2.35
[codecarbon INFO @ 16:39:07]   Python version: 3.11.6
[codecarbon INFO @ 16:39:07]   CodeCarbon version: 2.3.1
[codecarbon INFO @ 16:39:07]   Available RAM : 15.283 GB
[codecarbon INFO @ 16:39:07]   CPU count: 12
[codecarbon INFO @ 16:39:07]   CPU model: 11th Gen Intel(R) Core(TM) i5-1140
0 @ 2.60GHz
[codecarbon INFO @ 16:39:07]   GPU count: 1
[codecarbon INFO @ 16:39:07]   GPU model: 1 x NVIDIA GeForce RTX 3060
```

In [18]:
```
with EmissionsTracker(project_name="GPT-2 Training") as tracker:
  trainer.train()
```

```
[codecarbon INFO @ 16:39:10] [setup] RAM Tracking...
[codecarbon INFO @ 16:39:10] [setup] GPU Tracking...
[codecarbon INFO @ 16:39:11] Tracking Nvidia GPU via pynvml
[codecarbon INFO @ 16:39:11] [setup] CPU Tracking...
[codecarbon WARNING @ 16:39:11] No CPU tracking mode found. Falling back on
CPU constant mode.
[codecarbon INFO @ 16:39:12] CPU Model on constant consumption mode: 11th Ge
n Intel(R) Core(TM) i5-11400 @ 2.60GHz
[codecarbon INFO @ 16:39:12] >>> Tracker's metadata:
[codecarbon INFO @ 16:39:12]   Platform system: Linux-6.2.0-36-generic-x86_6
4-with-glibc2.35
[codecarbon INFO @ 16:39:12]   Python version: 3.11.6
[codecarbon INFO @ 16:39:12]   CodeCarbon version: 2.3.1
[codecarbon INFO @ 16:39:12]   Available RAM : 15.283 GB
[codecarbon INFO @ 16:39:12]   CPU count: 12
[codecarbon INFO @ 16:39:12]   CPU model: 11th Gen Intel(R) Core(TM) i5-1140
0 @ 2.60GHz
[codecarbon INFO @ 16:39:12]   GPU count: 1
[codecarbon INFO @ 16:39:12]   GPU model: 1 x NVIDIA GeForce RTX 3060
```

[ 2/1169 : < :, Epoch 0.00/1]

| Step | Training Loss |
|---|---|

```
[codecarbon INFO @ 16:39:30] Energy consumed for RAM : 0.000024 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:39:30] Energy consumed for all GPUs : 0.000569 kWh. To
tal GPU Power : 136.46114814490255 W
[codecarbon INFO @ 16:39:30] Energy consumed for all CPUs : 0.000135 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:39:30] 0.000728 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:39:30] Energy consumed for RAM : 0.000024 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:39:30] Energy consumed for all GPUs : 0.000572 kWh. To
tal GPU Power : 137.19117396879653 W
[codecarbon INFO @ 16:39:30] Energy consumed for all CPUs : 0.000135 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:39:30] 0.000731 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:39:45] Energy consumed for RAM : 0.000048 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:39:45] Energy consumed for all GPUs : 0.001157 kWh. To
tal GPU Power : 141.23462647096108 W
[codecarbon INFO @ 16:39:45] Energy consumed for all CPUs : 0.000271 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:39:45] 0.001476 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:39:45] Energy consumed for RAM : 0.000048 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:39:45] Energy consumed for all GPUs : 0.001160 kWh. To
tal GPU Power : 141.1532496750833 W
[codecarbon INFO @ 16:39:45] Energy consumed for all CPUs : 0.000271 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:39:45] 0.001479 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:00] Energy consumed for RAM : 0.000072 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:00] Energy consumed for all GPUs : 0.001749 kWh. To
tal GPU Power : 142.02245049690114 W
[codecarbon INFO @ 16:40:00] Energy consumed for all CPUs : 0.000406 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:00] 0.002227 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:00] Energy consumed for RAM : 0.000072 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:00] Energy consumed for all GPUs : 0.001753 kWh. To
tal GPU Power : 142.2748275642665 W
[codecarbon INFO @ 16:40:00] Energy consumed for all CPUs : 0.000406 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:00] 0.002231 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:15] Energy consumed for RAM : 0.000096 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:15] Energy consumed for all GPUs : 0.002339 kWh. To
tal GPU Power : 141.66305160684715 W
[codecarbon INFO @ 16:40:15] Energy consumed for all CPUs : 0.000542 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:15] 0.002976 kWh of electricity used since the begi
nning.
```

```
[codecarbon INFO @ 16:40:15] Energy consumed for RAM : 0.000096 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:15] Energy consumed for all GPUs : 0.002343 kWh. To
tal GPU Power : 141.5766209958817 W
[codecarbon INFO @ 16:40:15] Energy consumed for all CPUs : 0.000542 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:15] 0.002980 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:30] Energy consumed for RAM : 0.000119 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:30] Energy consumed for all GPUs : 0.002928 kWh. To
tal GPU Power : 141.41479561016007 W
[codecarbon INFO @ 16:40:30] Energy consumed for all CPUs : 0.000677 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:30] 0.003725 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:30] Energy consumed for RAM : 0.000119 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:30] Energy consumed for all GPUs : 0.002931 kWh. To
tal GPU Power : 141.34201182963176 W
[codecarbon INFO @ 16:40:30] Energy consumed for all CPUs : 0.000677 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:30] 0.003728 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:45] Energy consumed for RAM : 0.000143 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:45] Energy consumed for all GPUs : 0.003523 kWh. To
tal GPU Power : 142.90533593269254 W
[codecarbon INFO @ 16:40:45] Energy consumed for all CPUs : 0.000813 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:45] 0.004479 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:40:45] Energy consumed for RAM : 0.000143 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:40:45] Energy consumed for all GPUs : 0.003527 kWh. To
tal GPU Power : 143.07682835445496 W
[codecarbon INFO @ 16:40:45] Energy consumed for all CPUs : 0.000813 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:40:45] 0.004483 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:00] Energy consumed for RAM : 0.000167 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:00] Energy consumed for all GPUs : 0.004114 kWh. To
tal GPU Power : 141.66704879856314 W
[codecarbon INFO @ 16:41:00] Energy consumed for all CPUs : 0.000948 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:00] 0.005229 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:00] Energy consumed for RAM : 0.000167 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:00] Energy consumed for all GPUs : 0.004117 kWh. To
tal GPU Power : 141.5662071933855 W
[codecarbon INFO @ 16:41:00] Energy consumed for all CPUs : 0.000948 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:00] 0.005232 kWh of electricity used since the begi
nning.
```

```
[codecarbon INFO @ 16:41:15] Energy consumed for RAM : 0.000191 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:15] Energy consumed for all GPUs : 0.004705 kWh. To
tal GPU Power : 141.84677534081302 W
[codecarbon INFO @ 16:41:15] Energy consumed for all CPUs : 0.001083 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:15] 0.005979 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:15] Energy consumed for RAM : 0.000191 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:15] Energy consumed for all GPUs : 0.004709 kWh. To
tal GPU Power : 141.95679684072198 W
[codecarbon INFO @ 16:41:15] Energy consumed for all CPUs : 0.001083 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:15] 0.005983 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:30] Energy consumed for RAM : 0.000215 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:30] Energy consumed for all GPUs : 0.005302 kWh. To
tal GPU Power : 143.46713917244085 W
[codecarbon INFO @ 16:41:30] Energy consumed for all CPUs : 0.001219 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:30] 0.006736 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:30] Energy consumed for RAM : 0.000215 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:30] Energy consumed for all GPUs : 0.005306 kWh. To
tal GPU Power : 143.28362638255527 W
[codecarbon INFO @ 16:41:30] Energy consumed for all CPUs : 0.001219 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:30] 0.006739 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:45] Energy consumed for RAM : 0.000239 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:45] Energy consumed for all GPUs : 0.005895 kWh. To
tal GPU Power : 142.29216139542706 W
[codecarbon INFO @ 16:41:45] Energy consumed for all CPUs : 0.001354 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:45] 0.007488 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:41:45] Energy consumed for RAM : 0.000239 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:41:45] Energy consumed for all GPUs : 0.005898 kWh. To
tal GPU Power : 142.2653223888415 W
[codecarbon INFO @ 16:41:45] Energy consumed for all CPUs : 0.001354 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:41:45] 0.007491 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:00] Energy consumed for RAM : 0.000263 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:00] Energy consumed for all GPUs : 0.006488 kWh. To
tal GPU Power : 142.4007315296677 W
[codecarbon INFO @ 16:42:00] Energy consumed for all CPUs : 0.001490 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:00] 0.008241 kWh of electricity used since the begi
nning.
```

```
[codecarbon INFO @ 16:42:00] Energy consumed for RAM : 0.000263 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:00] Energy consumed for all GPUs : 0.006491 kWh. To
tal GPU Power : 142.36106395896675 W
[codecarbon INFO @ 16:42:00] Energy consumed for all CPUs : 0.001490 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:00] 0.008244 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:15] Energy consumed for RAM : 0.000286 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:15] Energy consumed for all GPUs : 0.007082 kWh. To
tal GPU Power : 142.5533413328669 W
[codecarbon INFO @ 16:42:15] Energy consumed for all CPUs : 0.001625 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:15] 0.008994 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:15] Energy consumed for RAM : 0.000287 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:15] Energy consumed for all GPUs : 0.007086 kWh. To
tal GPU Power : 142.74110960965731 W
[codecarbon INFO @ 16:42:15] Energy consumed for all CPUs : 0.001625 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:15] 0.008998 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:30] Energy consumed for RAM : 0.000310 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:30] Energy consumed for all GPUs : 0.007669 kWh. To
tal GPU Power : 140.8192555729788 W
[codecarbon INFO @ 16:42:30] Energy consumed for all CPUs : 0.001760 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:30] 0.009740 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:30] Energy consumed for RAM : 0.000310 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:30] Energy consumed for all GPUs : 0.007672 kWh. To
tal GPU Power : 140.7314450188939 W
[codecarbon INFO @ 16:42:30] Energy consumed for all CPUs : 0.001760 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:30] 0.009743 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:45] Energy consumed for RAM : 0.000334 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:45] Energy consumed for all GPUs : 0.008271 kWh. To
tal GPU Power : 144.5884500855775 W
[codecarbon INFO @ 16:42:45] Energy consumed for all CPUs : 0.001896 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:45] 0.010501 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:42:45] Energy consumed for RAM : 0.000334 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:42:45] Energy consumed for all GPUs : 0.008274 kWh. To
tal GPU Power : 144.52196829652803 W
[codecarbon INFO @ 16:42:45] Energy consumed for all CPUs : 0.001896 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:42:45] 0.010505 kWh of electricity used since the begi
nning.
```

```
[codecarbon INFO @ 16:43:00] Energy consumed for RAM : 0.000358 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:43:00] Energy consumed for all GPUs : 0.008859 kWh. To
tal GPU Power : 141.0162844395877 W
[codecarbon INFO @ 16:43:00] Energy consumed for all CPUs : 0.002031 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:43:00] 0.011248 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:43:00] Energy consumed for RAM : 0.000358 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:43:00] Energy consumed for all GPUs : 0.008863 kWh. To
tal GPU Power : 141.19147033847003 W
[codecarbon INFO @ 16:43:00] Energy consumed for all CPUs : 0.002031 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:43:00] 0.011252 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:43:07] Energy consumed for RAM : 0.000370 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:43:07] Energy consumed for all GPUs : 0.009152 kWh. To
tal GPU Power : 143.11373352570294 W
[codecarbon INFO @ 16:43:07] Energy consumed for all CPUs : 0.002097 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:43:07] 0.011619 kWh of electricity used since the begi
nning.
[codecarbon INFO @ 16:43:07] Energy consumed for RAM : 0.000370 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:43:07] Energy consumed for all GPUs : 0.009152 kWh. To
tal GPU Power : 143.08069456747666 W
[codecarbon INFO @ 16:43:07] Energy consumed for all CPUs : 0.002098 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:43:07] 0.011620 kWh of electricity used since the begi
nning.
```

In [19]:
```
report_emissions(tracker)
```

```
Carbon emitted:      458.96 mg
Energy consumed:     11.62 Wh
```

**Your turn**

Now that we've trained our model, let's use it to generate some text. Use the `generate` method on the `trainer` object to generate some text. You can use the documentation to help you. Use CodeCarbon to measure the carbon footprint of generating the text. How does it compare to the carbon footprint of training the model? We can also look at the results of all our experiments thus far by running the following command:

`pd.read_csv('codecarbon.csv')`

In [20]:
```
with EmissionsTracker(project_name="GPT-2 Prediction") as predict:
  tokens = tokenizer.encode("Hello, my name is", return_tensors='pt').to('cu
  output = model.generate(tokens, do_sample=True, max_length=50, top_k=50, t
report_emissions(predict)
print(tokenizer.decode(output[0]))
```

```
[codecarbon INFO @ 16:43:07] [setup] RAM Tracking...
[codecarbon INFO @ 16:43:07] [setup] GPU Tracking...
[codecarbon INFO @ 16:43:07] Tracking Nvidia GPU via pynvml
[codecarbon INFO @ 16:43:07] [setup] CPU Tracking...
[codecarbon WARNING @ 16:43:07] No CPU tracking mode found. Falling back on
CPU constant mode.
[codecarbon INFO @ 16:43:09] CPU Model on constant consumption mode: 11th Ge
n Intel(R) Core(TM) i5-11400 @ 2.60GHz
[codecarbon INFO @ 16:43:09] >>> Tracker's metadata:
[codecarbon INFO @ 16:43:09]   Platform system: Linux-6.2.0-36-generic-x86_6
4-with-glibc2.35
[codecarbon INFO @ 16:43:09]   Python version: 3.11.6
[codecarbon INFO @ 16:43:09]   CodeCarbon version: 2.3.1
[codecarbon INFO @ 16:43:09]   Available RAM : 15.283 GB
[codecarbon INFO @ 16:43:09]   CPU count: 12
[codecarbon INFO @ 16:43:09]   CPU model: 11th Gen Intel(R) Core(TM) i5-1140
0 @ 2.60GHz
[codecarbon INFO @ 16:43:09]   GPU count: 1
[codecarbon INFO @ 16:43:09]   GPU model: 1 x NVIDIA GeForce RTX 3060
The attention mask and the pad token id were not set. As a consequence, you
may observe unexpected behavior. Please pass your input's `attention_mask` t
o obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[codecarbon INFO @ 16:43:12] Energy consumed for RAM : 0.000001 kWh. RAM Pow
er : 5.7310380935668945 W
[codecarbon INFO @ 16:43:12] Energy consumed for all GPUs : 0.000005 kWh. To
tal GPU Power : 37.467016822370674 W
[codecarbon INFO @ 16:43:12] Energy consumed for all CPUs : 0.000004 kWh. To
tal CPU Power : 32.5 W
[codecarbon INFO @ 16:43:12] 0.000010 kWh of electricity used since the begi
nning.
Carbon emitted:      385.13 µg
Energy consumed:     9.75 mWh
Hello, my name is Hilda.
 = = = = Koopas = = =
 In the movie, O'Donnell ( Bill Mullen " Hilda ", and Barry Johnson ) narrat
es O'Donnell and the K
```

In [21]:
```
df = pd.read_csv('emissions.csv')
df
```

Out[21]:

| | timestamp | project_name | run_id | duration | emissions | emission |
|---|---|---|---|---|---|---|
| **0** | 2023-11-09T16:30:51 | Linear Regression | 1b5b8cc1-f631-4d82-96be-900600d1fb03 | 0.102026 | 4.966841e-08 | 4.8682 |
| **1** | 2023-11-09T16:30:56 | Linear Regression Prediction | 5ea4c281-4ce7-4b7c-9efc-5574ace3e7f2 | 0.011489 | 4.711100e-09 | 4.1003 |
| **2** | 2023-11-09T16:31:02 | Random Forest | 4f297ca6-cb5a-400d-9c0e-514f3f7eb116 | 1.490961 | 6.943096e-07 | 4.6567 |
| **3** | 2023-11-09T16:31:06 | Random Forest Prediction | 90659181-1c8e-4cc5-aae8-49056b065497 | 0.027983 | 1.163435e-08 | 4.1576 |
| **4** | 2023-11-09T16:31:35 | Extra Random Trees | ad6bee1a-473a-4dac-b43d-5ed0513ab5a6 | 0.425336 | 1.867338e-07 | 4.3902 |
| **5** | 2023-11-09T16:31:40 | Extra Random Trees Prediction | b5e2ac6f-cf99-4b51-abce-4c72521b22b7 | 0.038878 | 1.888297e-08 | 4.8569 |
| **6** | 2023-11-09T16:36:04 | GPT-2 Training | ff6edc35-701f-486c-a1e3-1f51dc82359d | 232.182654 | 4.583554e-04 | 1.9741 |
| **7** | 2023-11-09T16:36:19 | GPT-2 Prediction | c9ca7b40-3b5b-44dd-8af6-524d9a0d0d2d | 0.003132 | 1.030058e-09 | 3.2889 |
| **8** | 2023-11-09T16:36:33 | GPT-2 Prediction | c94227e9-57d1-439f-9b6b-5bc044582fa7 | 0.003734 | 1.088585e-09 | 2.9154 |
| **9** | 2023-11-09T16:36:41 | GPT-2 Prediction | a8dee499-a4f8-4658-85a4-d950eb87878f | 0.003347 | 1.070240e-09 | 3.1976 |
| **10** | 2023-11-09T16:37:05 | GPT-2 Prediction | ce604510-01d5-4ff0- | 0.003324 | 1.029192e-09 | 3.0964 |

| | timestamp | project_name | run_id | duration | emissions | emission |
|---|---|---|---|---|---|---|
| | | | b4c2-44694345026b | | | |
| **11** | 2023-11-09T16:37:33 | GPT-2 Prediction | 16140f06-8f15-4347-b07c-eed775eaa5e1 | 0.005158 | 1.808987e-09 | 3.5068 |
| **12** | 2023-11-09T16:37:44 | GPT-2 Prediction | d96ed251-7fe1-456d-9021-26f1fefe06c3 | 0.016630 | 6.947820e-09 | 4.1779 |
| **13** | 2023-11-09T16:38:05 | GPT-2 Prediction | 093cc5ba-988a-4172-9adb-96bb3c4428ed | 1.066819 | 8.070748e-07 | 7.5652 |
| **14** | 2023-11-09T16:38:40 | Linear Regression | 67dd9fb8-2646-4cc5-8ba3-fcc0621995c8 | 0.010108 | 4.071640e-09 | 4.0282 |
| **15** | 2023-11-09T16:38:44 | Linear Regression Prediction | c8af6b3b-34af-4684-85ee-a30633644b66 | 0.002886 | 1.119355e-09 | 3.8791 |
| **16** | 2023-11-09T16:38:50 | Random Forest | a620b3ce-9f7f-4f3c-869c-9978c563e8e7 | 1.288191 | 6.058023e-07 | 4.7027 |
| **17** | 2023-11-09T16:38:54 | Random Forest Prediction | 6e49c03f-b4fd-4f46-ab20-5f2ceac656ee | 0.026212 | 1.089720e-08 | 4.1573 |
| **18** | 2023-11-09T16:38:59 | Extra Random Trees | 4a01889c-9cbf-4f96-8414-eca1e5e4bec4 | 0.407088 | 2.024677e-07 | 4.9735 |
| **19** | 2023-11-09T16:39:03 | Extra Random Trees Prediction | f3fe2372-d8a3-4061-b61b-151d063b2307 | 0.038591 | 1.592424e-08 | 4.1263 |
| **20** | 2023-11-09T16:43:07 | GPT-2 Training | 89eba29f-4887-4b7b-9522-69ee474f615b | 232.401682 | 4.589567e-04 | 1.9748 |

| | timestamp | project_name | run_id | duration | emissions | emission |
|---|---|---|---|---|---|---|
| **21** | 2023-11-09T16:43:12 | GPT-2 Prediction | eaea4258-44c7-45a4-8287-e9f593ca8dd6 | 0.464987 | 3.851254e-07 | 8.2824 |

22 rows × 31 columns

For reference, one estimate of the carbon cost to train GPT-4 is around 12,500 metric tons of CO2. This is based on the assumption that the model is trained in California, using about 25,000 NVIDIA A100 GPUs. This is the equivalent of the anual emissions of 2,700 cars. This is a lot of carbon, but it's important to remember that this is a one-time cost. Once the model is trained, it can be used by many people, with a much lower carbon cost per user.

# Conclusion

In this lab, we have explored the carbon cost of training a machine learning model. We have seen that the cost of training a large language model is substantially higher than a traditional machine learning algorithm. There are a number of ways that we can try to reduce carbon emissions in machine learning:

- **Use more efficient hardware**: The hardware we use to train our models has a significant impact on the carbon footprint of our code. Using more efficient hardware, such as GPUs, can reduce the carbon footprint of our code.
- **Use more efficient algorithms**: Some algorithms are more efficient than others. For example, linear regression is much more efficient than a large language model.
- **Run code in regions with renewable energy**: The location of the computing resources powering our code has a significant impact on the carbon footprint of our code. Running our code in regions with a high proportion of renewable energy can reduce the carbon footprint of our code.
- **Train less often**: Training a model has a much higher carbon cost than using it. If we are careful about how often we train our models, we can spread the carbon cost over a longer period of time.
- **Use smaller models**: Large language models are very powerful, but they also have a high carbon cost. If we can use a smaller model, we can reduce the carbon footprint of our code.