AGILE 2022 JULY 18-22
NASHVILLE

Environmental Impact of Digital Products

Archana Joshi
Head – IT Strategy, LTI – Larsen & Toubro Infotech

Note: The views expressed in the presentation are solely of the presenter and do not represent those of the company /clients she is associated with

20+ yrs in IT
Developer - Scrum Master - Transformation Agent - Tech Incubator

GOODS CARRIER
MH01CR6257

CONCERT À NOT
MARDI 7 AOÛT
BACH - MON
COUPERIN

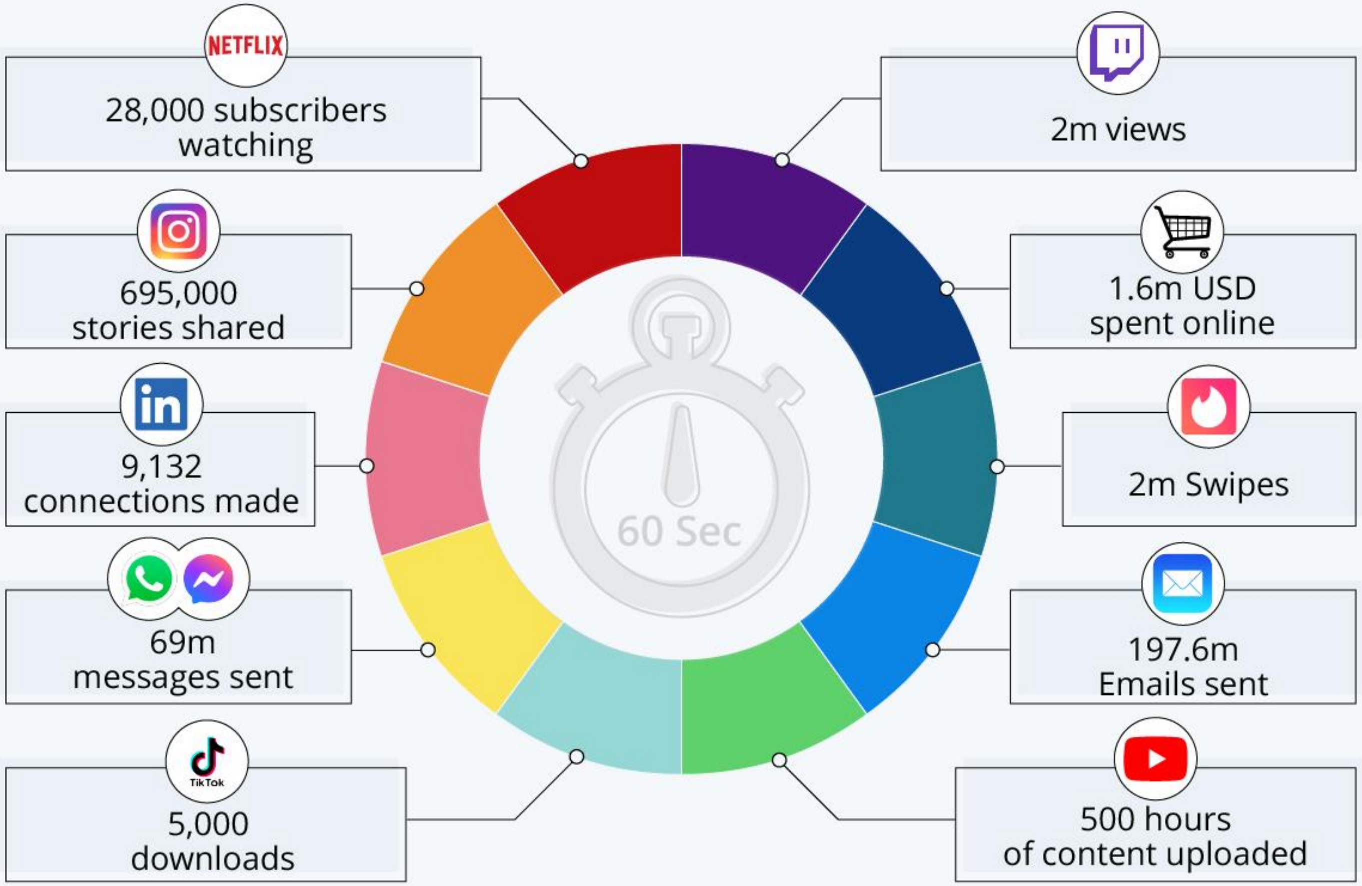
Tele medicine

E-payments

Education

A Minute on the Internet in 2021

Estimated amount of data created
on the internet in one minute



28,000 subscribers watching

695,000 stories shared

9,132 connections made

69m messages sent

5,000 downloads

2m views

1.6m USD spent online

2m Swipes

197.6m Emails sent

500 hours of content uploaded

Source: Lori Lewis via AllAccess

statista

Digital Jury

Digital Clothes

To meet this change

enterprises need to invest in supporting the digital paradigm

202x truly a “TechDecade”

Global spending

on digital transformation will reach **$6.8 trillion** by 2023 equivalent to GDP of 2 continents

- IDC

While we are doing all this

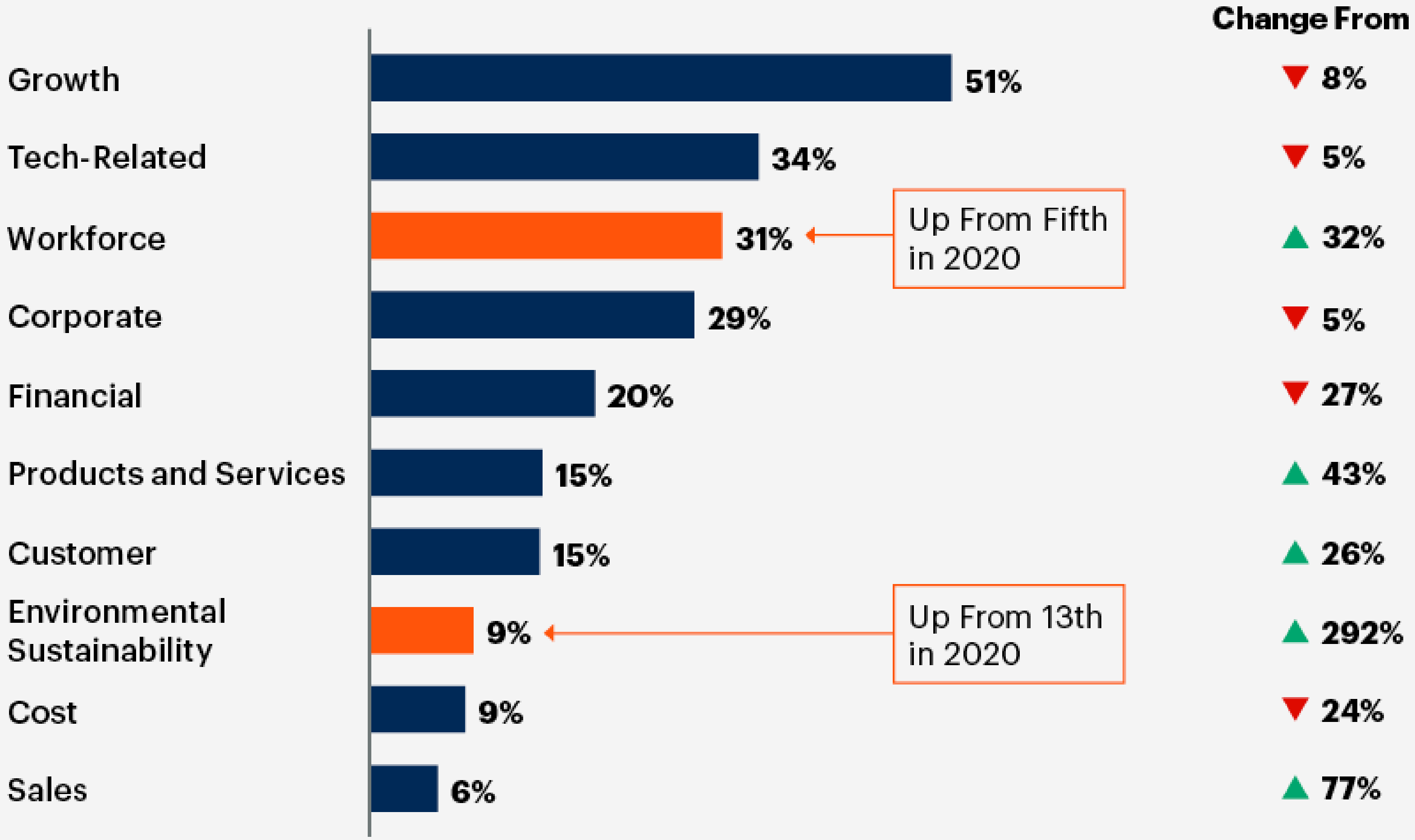
are we doing it **responsibly?**

The Paris Agreement is a legally binding international treaty on climate change.

It was adopted by 196 Parties in Paris, on 12 December 2015 and entered into force on 4 November 2016.

CEOs' Top 10 Strategic Business Priority Areas for 2022-2023

Summary Top Three Mentions, Coded Responses

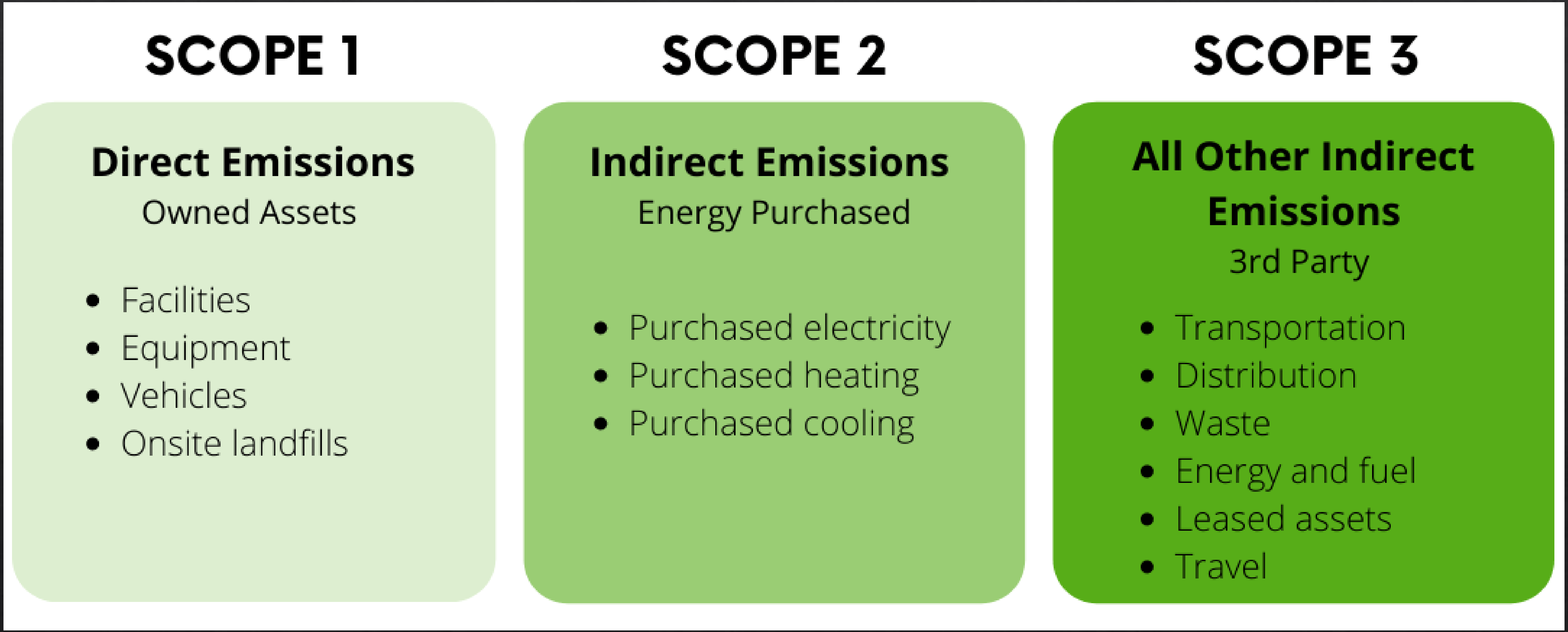


Up From Fifth in 2020

Up From 13th in 2020

Source: Gartner

Gartner



SCOPE 1

Direct Emissions

Owned Assets

- Facilities
- Equipment
- Vehicles
- Onsite landfills

SCOPE 2

Indirect Emissions

Energy Purchased

- Purchased electricity
- Purchased heating
- Purchased cooling

SCOPE 3

All Other Indirect Emissions

3rd Party

- Transportation
- Distribution
- Waste
- Energy and fuel
- Leased assets
- Travel

Hardware Emissions

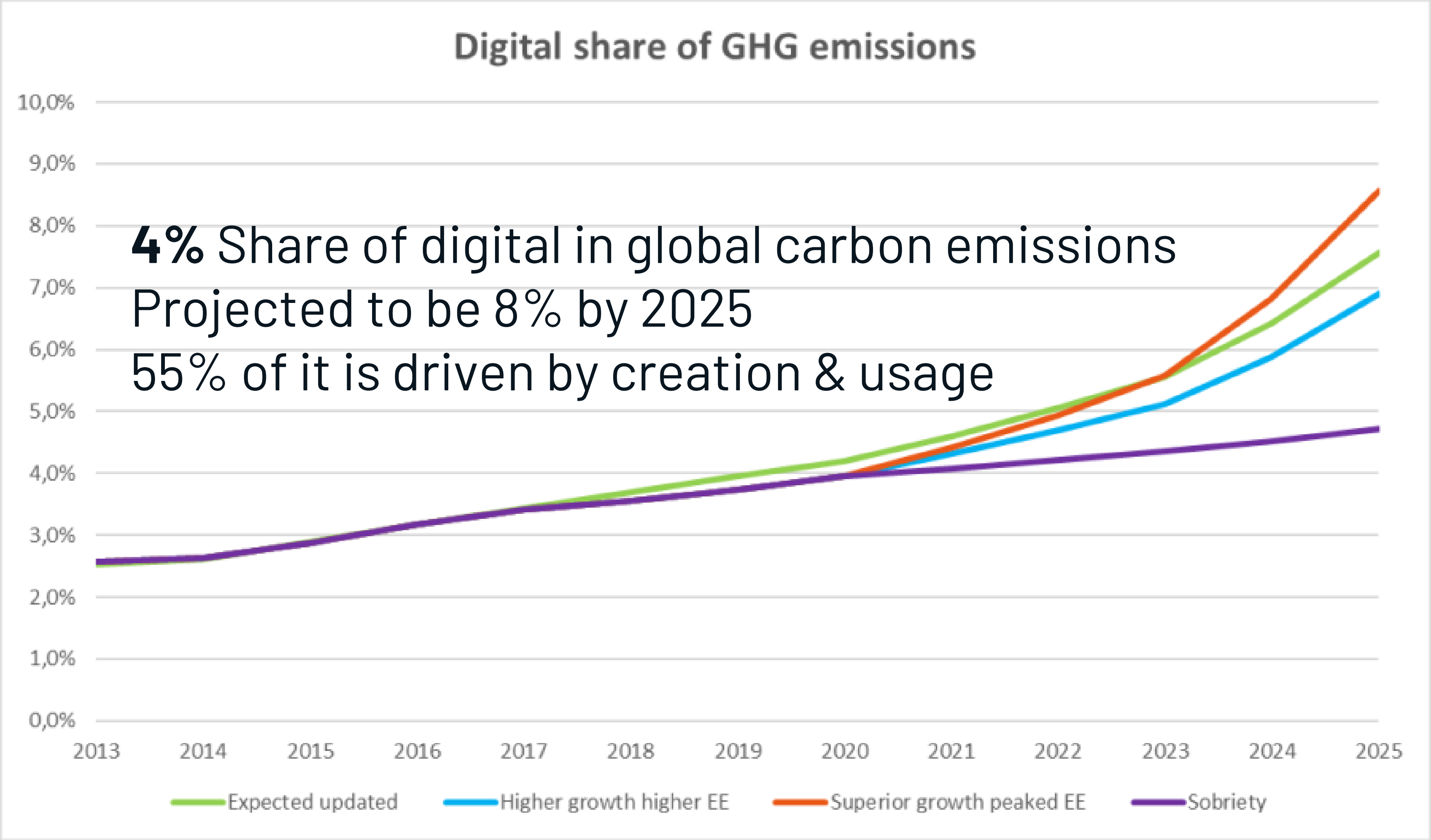
- These are the emissions due to the servers, data storage, sensors and devices

Software package Emissions

- These are the emissions due to how software is developed and used

Is it true?
The way software is designed affects the carbon emissions?

Let's See an Example



4% Share of digital in global carbon emissions

Projected to be 8% by 2025

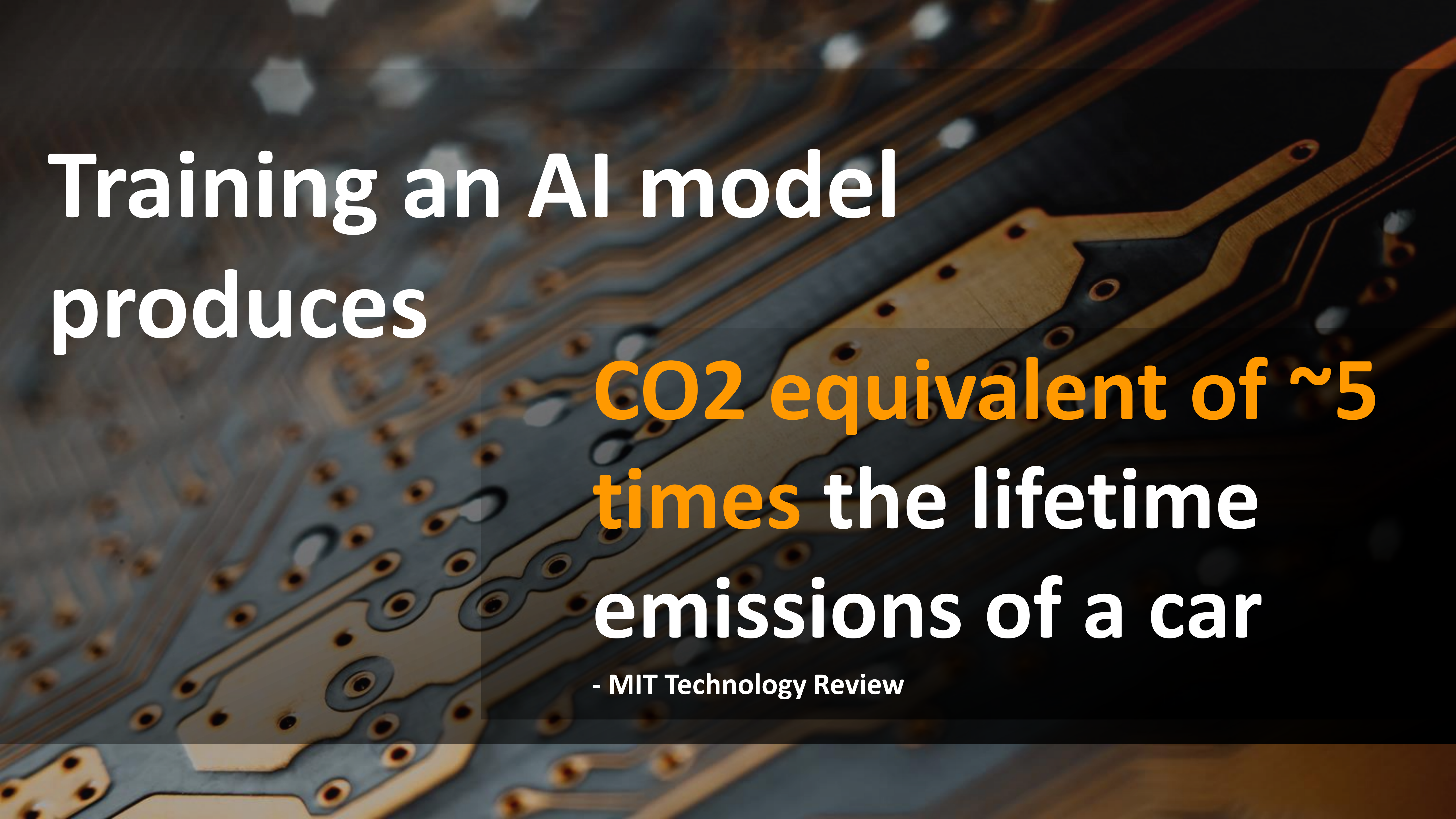
55% of it is driven by creation & usage

Figure 3: Evolution 2013-2025 of the share of digital technology in GHG emissions. The share of digital technology in GHG emissions.
[Source: [Lean ICT Materials] Forecast Model. Produced by The Shift Project from data published by (Andrae & Edler, 2015)

A program written in Python consumes

75.88 times more energy than a program written in C

- University Research in Portugal



Training an AI model produces

CO2 equivalent of ~5 times the lifetime emissions of a car

- MIT Technology Review

Bitcoin’s annual electricity consumption is more than three times that of New Zealand

- Energetics.au

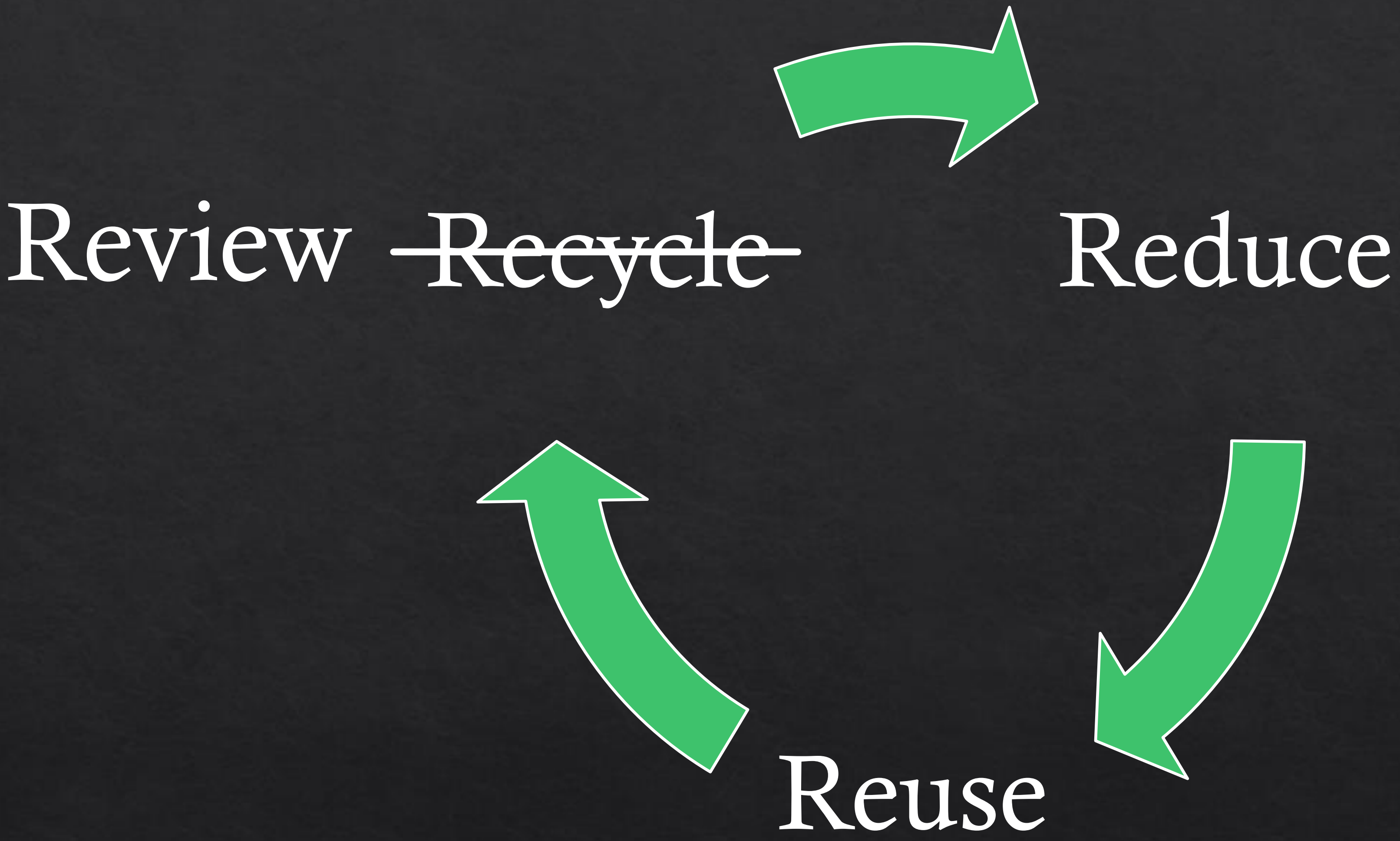
How many trees to be planted per year to offset effects of data generated in US?

Carbon offsetting is not enough

There is a pressing need to

relook at software engineering practices

to curb the overall carbon footprint.



Review ~~Recycle~~

Reduce

Reuse

Start by asking these 5 questions

1. Do we really need these features?

2. Can we remove the low priority ones?

3. Is similar feature available else where?

4. Can I share the developed feature with others?

5. Do we really need the software “Always ON”?

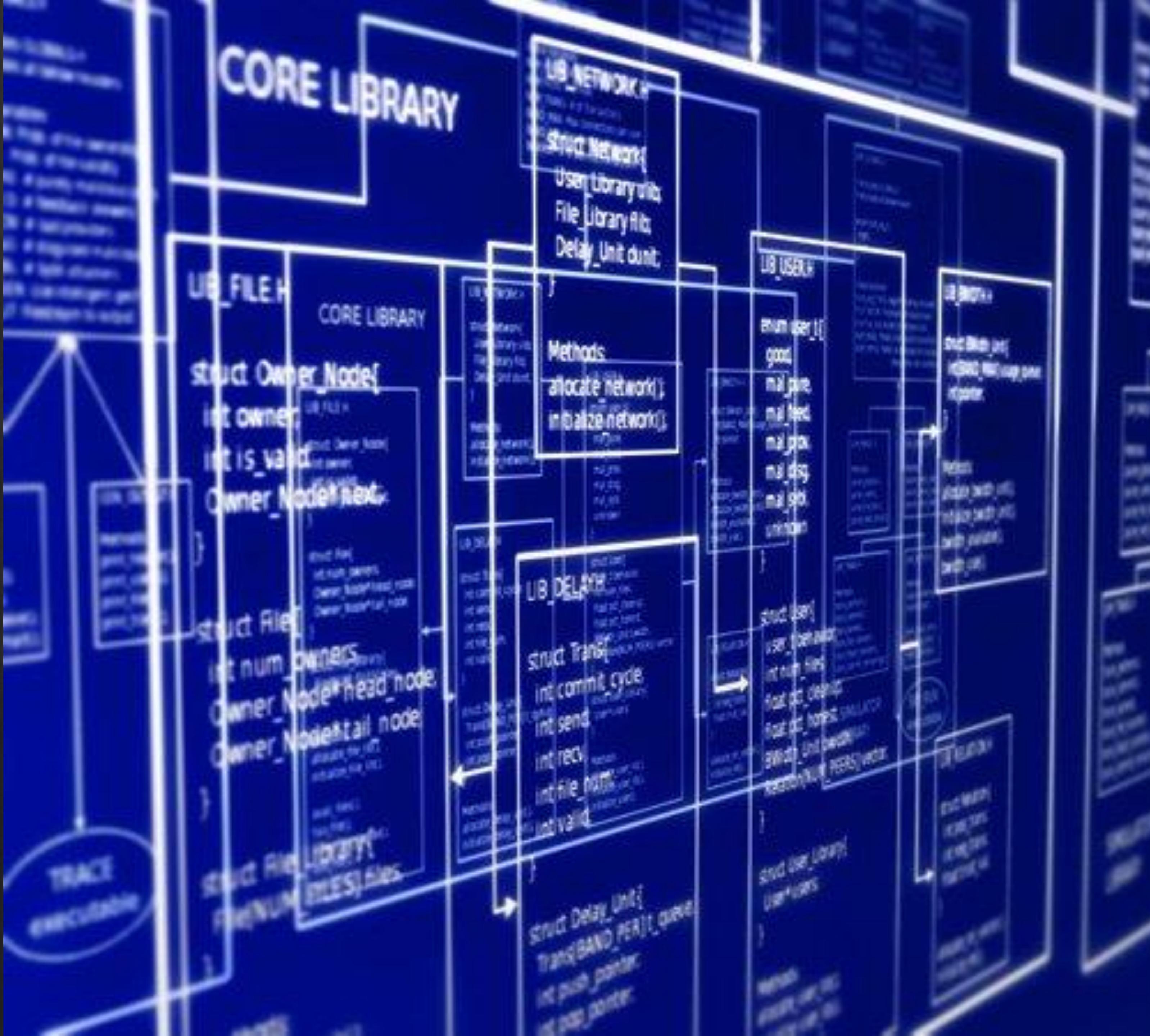
And then follow Green Software practices

1. Move to Cloud

Opting for a cloud server which is based 100% on green energy instead of an on-premise server that is powered by non-green energy can reduce carbon emission by up to 70%

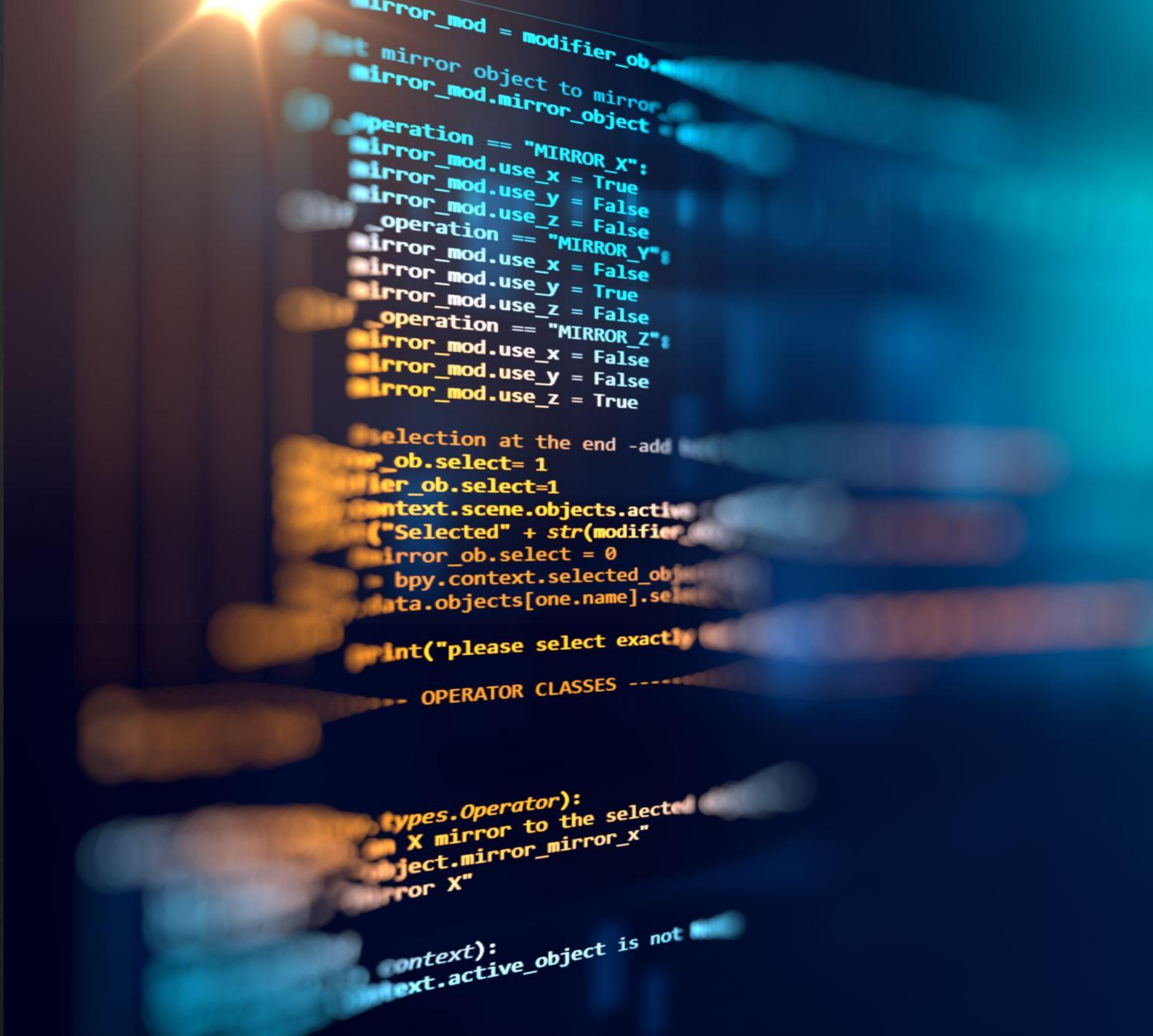
2. Change in Architecture

Moving towards event-based computing, serverless computing, containerization can reduce the response time, increase efficiency and reduce the carbon footprint



3. Coding/ Language Preferences

Code restructuring, reducing the lines of code, making the right language choices can reduce the CPU runtime and result in reduction of energy consumption



```
irror_mod = modifier_ob
et mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
_operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
_operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

selection at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.activ
("Selected" + str(modifier
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

 OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
ext.active_object is not
```

4. Database design

Delete unnecessary data while using data life-cycle policies. Prevent data movement across network boundaries as much as possible.

5. Carbon neutral solutions

When buying 3rd party software packages, check for their emissions data and choose carbon neutral solutions

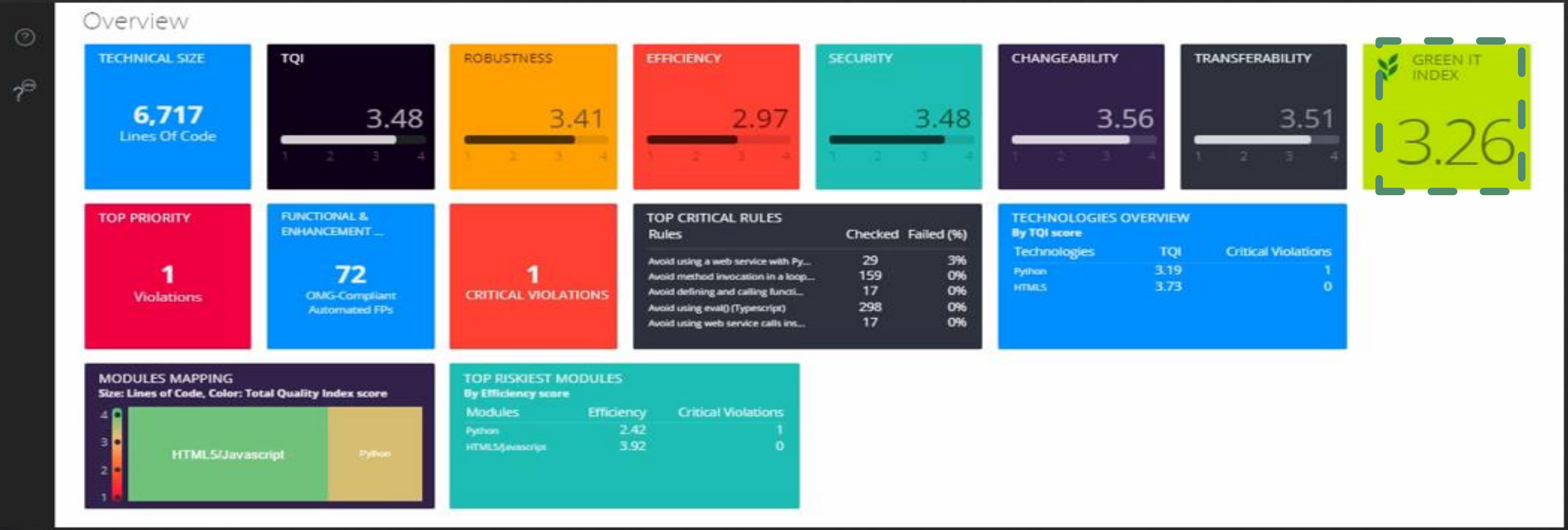
IOTA

CARDANO

6. Smart AI

Smart AI should focus on exploring possibilities to shift a company's AI training and processing to solutions that are greener

How do I measure?



Overview

TECHNICAL SIZE

6,717

Lines Of Code

TQI

3.48

ROBUSTNESS

3.41

EFFICIENCY

2.97

SECURITY

3.48

CHANGEABILITY

3.56

TRANSFERABILITY

3.51

GREEN IT INDEX

3.26

TOP PRIORITY

1

Violations

FUNCTIONAL & ENHANCEMENT ...

72

OMG-Compliant Automated FPs

1

CRITICAL VIOLATIONS

TOP CRITICAL RULES

Rules	Checked	Failed (%)
Avoid using a web service with Py...	29	3%
Avoid method invocation in a loop...	159	0%
Avoid defining and calling functi...	17	0%
Avoid using eval() (Typescript)	298	0%
Avoid using web service calls ins...	17	0%

TECHNOLOGIES OVERVIEW

By TQI score

Technologies	TQI	Critical Violations
Python	3.19	1
HTML5	3.73	0

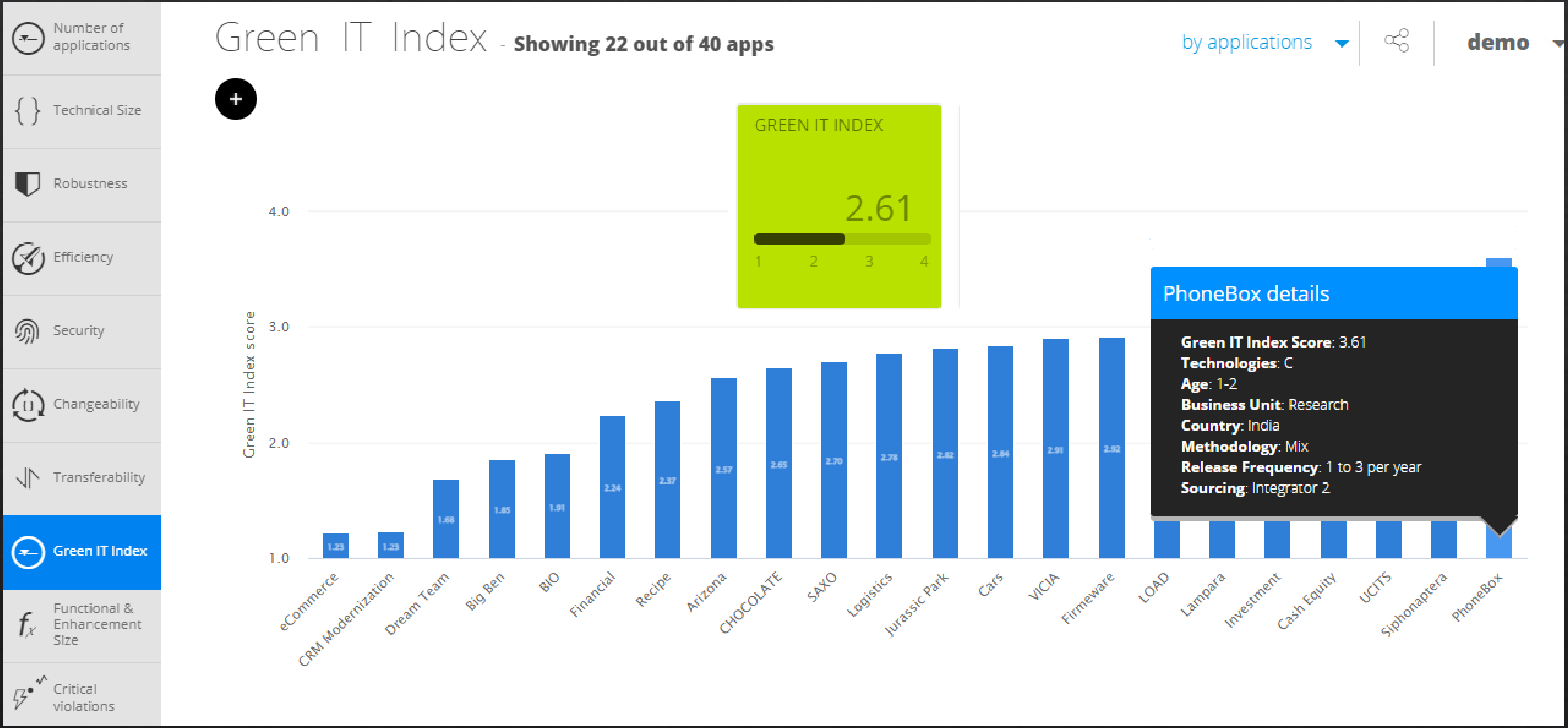
MODULES MAPPING

Size: Lines of Code, Color: Total Quality Index score

TOP RISKIEST MODULES

By Efficiency score

Modules	Efficiency	Critical Violations
Python	2.42	1
HTML5/Javascript	3.92	0



Green IT Index - Showing 22 out of 40 apps

GREEN IT INDEX 2.61

PhoneBox details

Green IT Index Score: 3.61
Technologies: C
Age: 1-2
Business Unit: Research
Country: India
Methodology: Mix
Release Frequency: 1 to 3 per year
Sourcing: Integrator 2

Underlying Technology criterion to assess the application’s Green IT Index

Complexity – SQL Queries

Programming Practices – Error and Exception Handling | Unexpected behaviors

Efficiency– Network, Data and Disk space management

Performance– SQL and Data Handling performance

Secure Coding– Weak security features, Time and State

Efficiency– Call in loops

What do enterprises need to do to build the Green Software culture?

Some links to get you started

1. Principles of Green Software Engineering: https://principles.green/

2. Tool that helps in calculating the Green Debt of your software: https://www.castsoftware.com/green-it

3. Measure and reduce website's CO2 emissions: https://greenframe.io/

4. Cloud Carbon Emissions Measurement and Analysis Tool: https://www.cloudcarbonfootprint.org/

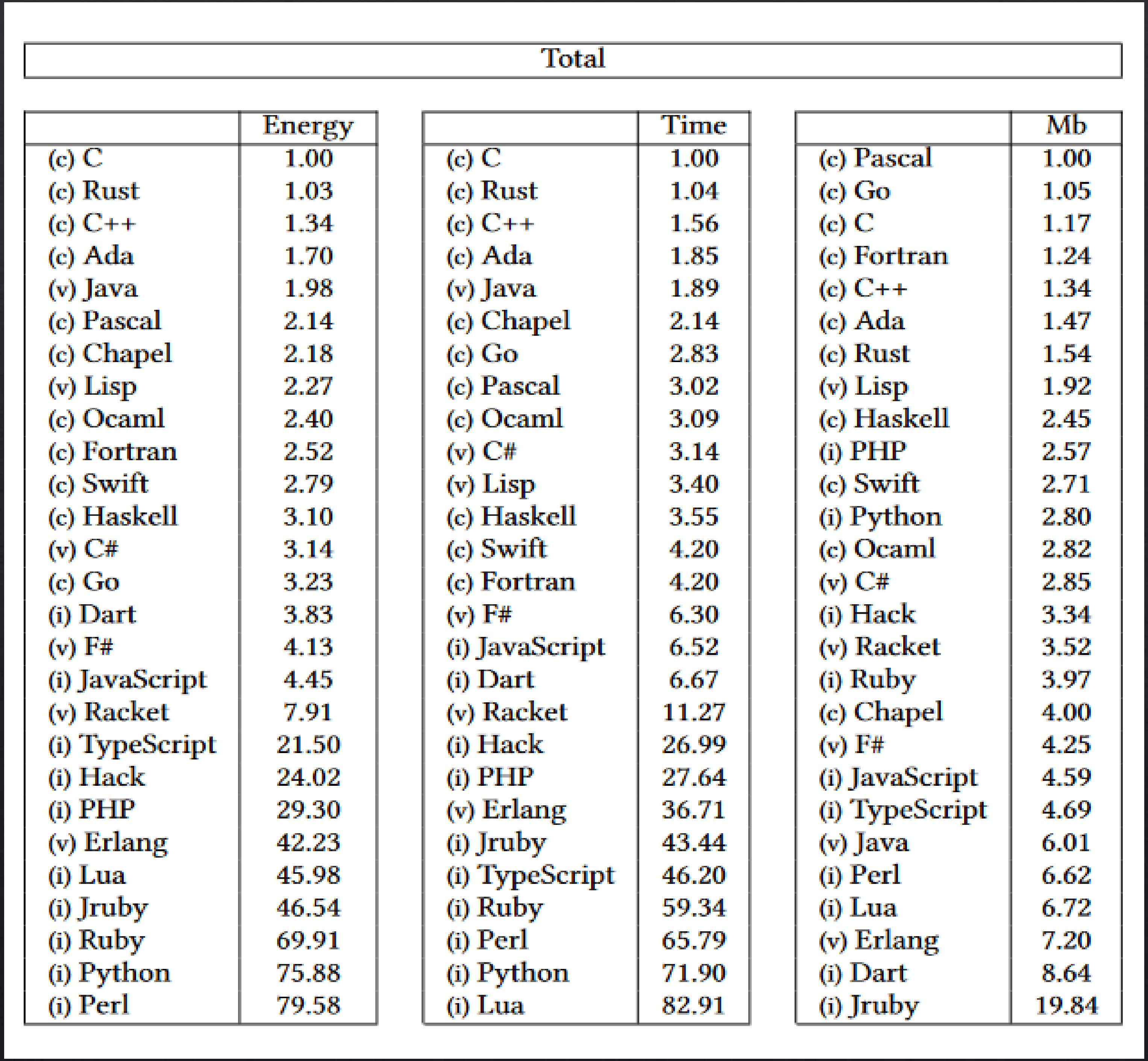
5. Machine Learning Emissions Calculator: https://mlco2.github.io/impact/#compute

6. Not for profit think tank that works on areas of climate change: https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project_2019.pdf

Be Digital
Be Responsible

Thank You
Archana Joshi

https://www.linkedin.com/in/arcjoshi/



Total

	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84