16 AUG 2024, MARCO DI FRANCESCO

MSc Thesis Presentation

Efficient Online Learning in Resource-Constrained Automation Environments

Why this project?
Robotic & Edge Computing

Robots tasks: Welding or Package sorting

Robots require:	High accuracy	Detect issues
	↓	↓
Machine Learning:	Calibration	Anomaly detection

ML models deployed in Edge Device: Small hardware close to the source

Provides:

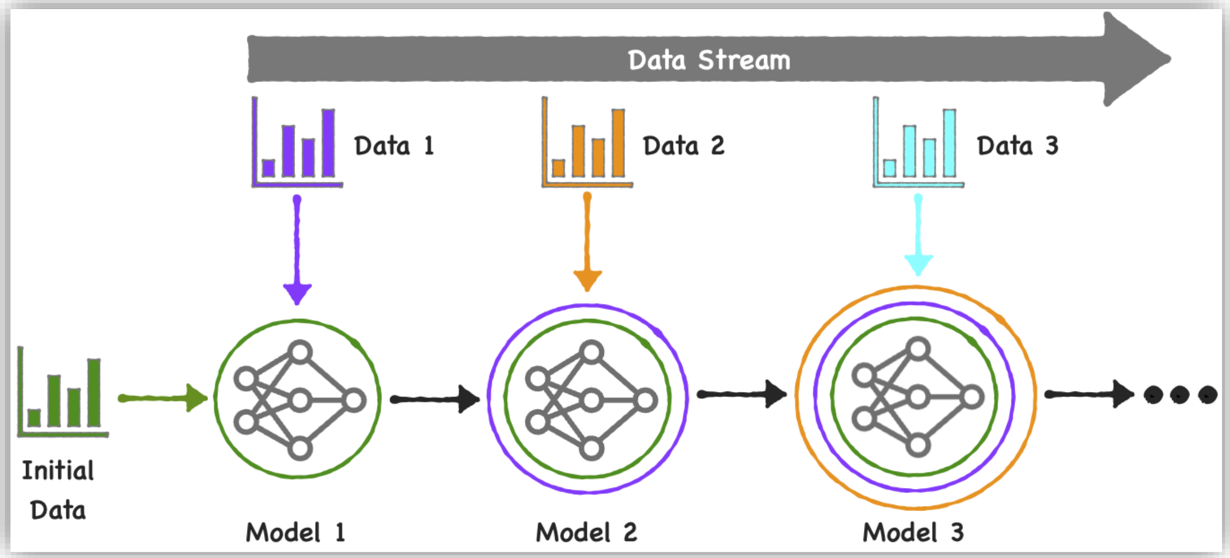
- Small latency
- Privacy: cannot transmit data covered by IP rights

Why this project?
Online Machine Learning

Degradation → Robots changes behaviour

Model needs constant updates

Domain: Online Machine Learning



Why not Reinforcement Learning?

- Computationally expensive

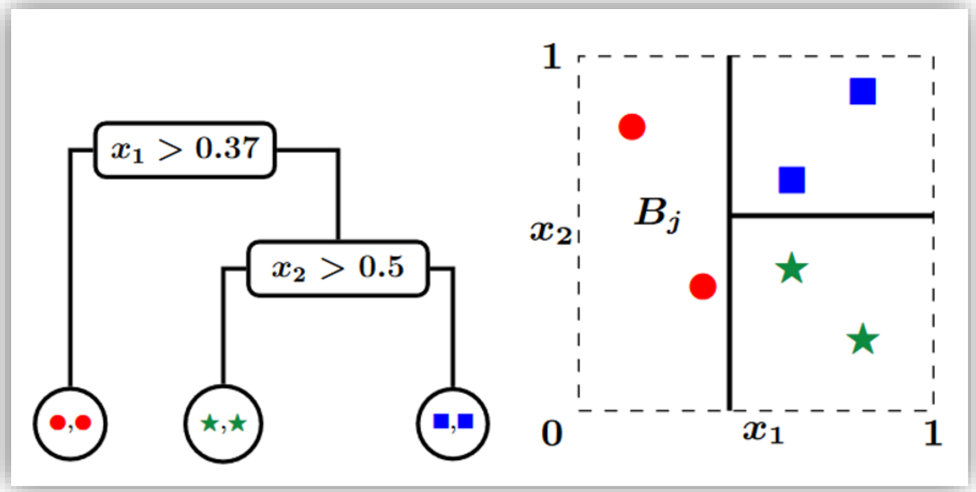
ML Models

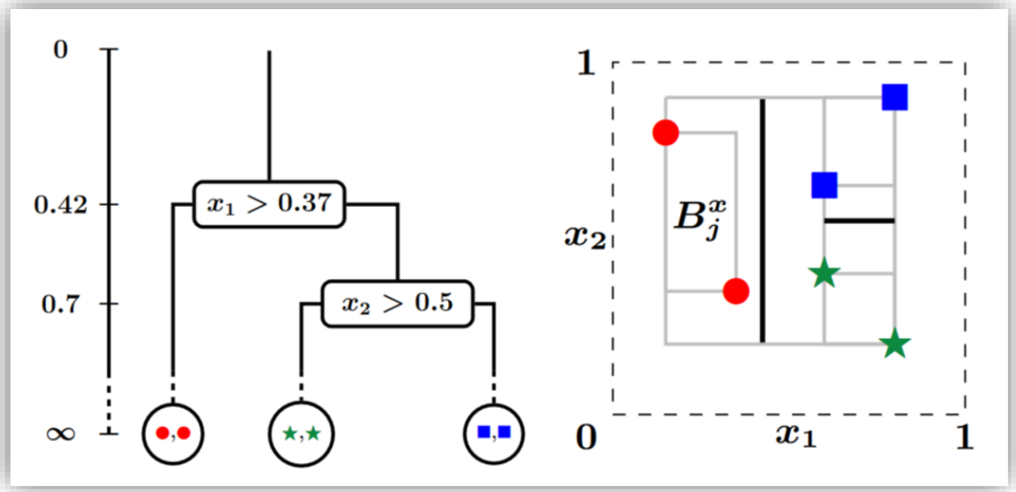
Decision Trees:

- Fast execution in CPU

Mondrian Forests:

- Adapts to changing environment
- Algorithm is very fast: 1 order of magnitude faster than other state-of-the-art decision tree models

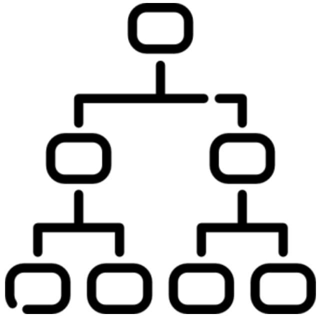




Project Outline

The 3 phases

Model Development



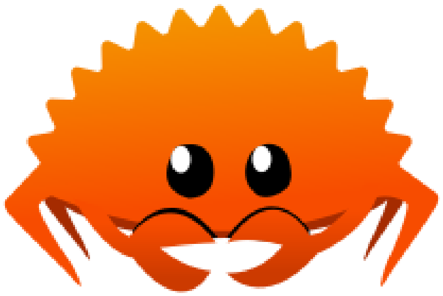
Optimization

Robotic Application

[Images] Flaticon.com – Freepik, Freepik, Smalllikeart

Model Development

Software



Current models: Python[1,2,3] Java[4,5]

- Additional software in the controller — Python Interpreter / JVM
- No granular control of the memory

Compressed model: Onnx / TensorFlow Lite

- Not suitable for incremental learning

Implementation in Rust:

- Performance
- Memory safety
- LightRiver library

LIGHTRIVER

[1, Paper] **River** - Montiel, J., et al. (2021). River: Machine learning for streaming data in Python online-ml/river (github.com)
[2, Paper] **OneLearn** - Wang, Y., et al. Onelearn: A Unified and Distributed Machine Learning Platform with High Performance. onelearn/onelearn (github.com)
[3, Repository] **Vowpal Wabbit** - Vowpal Wabbit (vowpalwabbit.org)
[4, Paper] **MOA** - Bifet, A., et. al (2018). Machine Learning for Data Streams: With Practical Examples in MOA. *The MIT Press*. https://doi.org/10.7551/mitpress/10654.001.0001
[3, Repository] **CapyMOA** - adaptive-machine-learning/CapyMOA (github.com)

Model Development
Python vs Rust

Model: Mondrian Forests

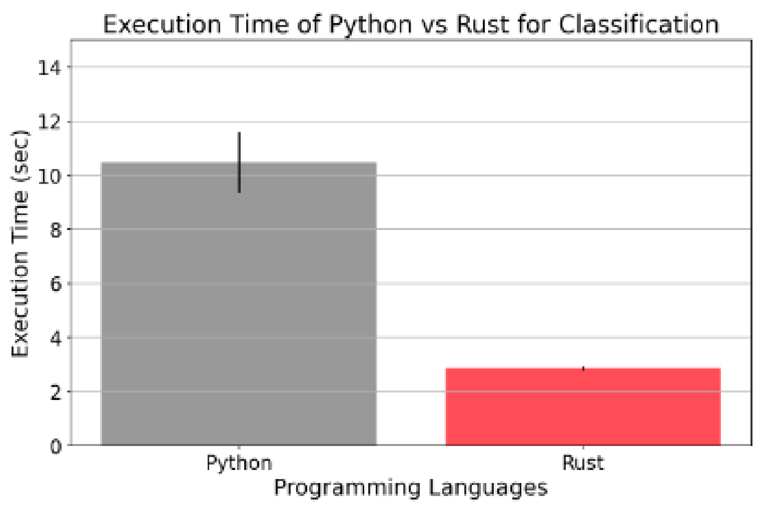
Dataset: 100.000 samples, 2 features

Classification:

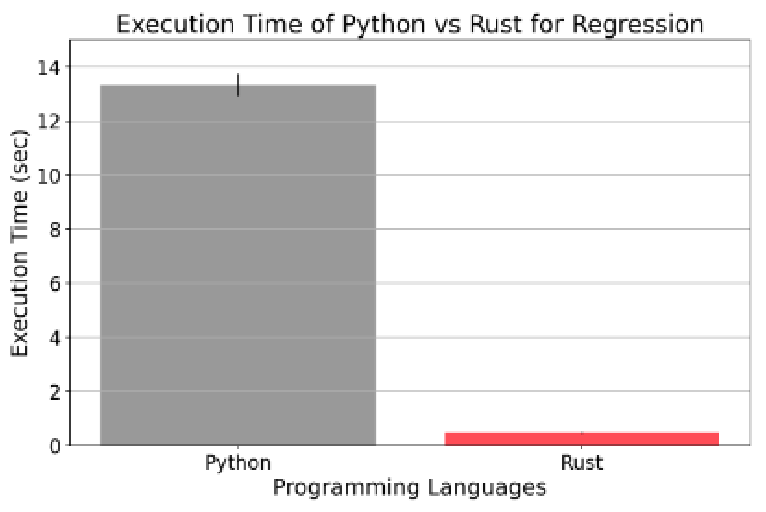
- Rust 3 times faster than Python
- 35 000 samples/sec

Regression:

- Rust 28 times faster than Python
- 213 000 samples/sec



(b) Classification

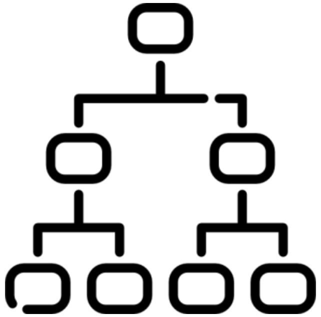


(a) Regression

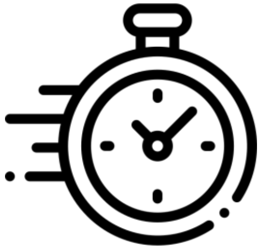
Project Outline

The 3 phases

Model Development



Optimization



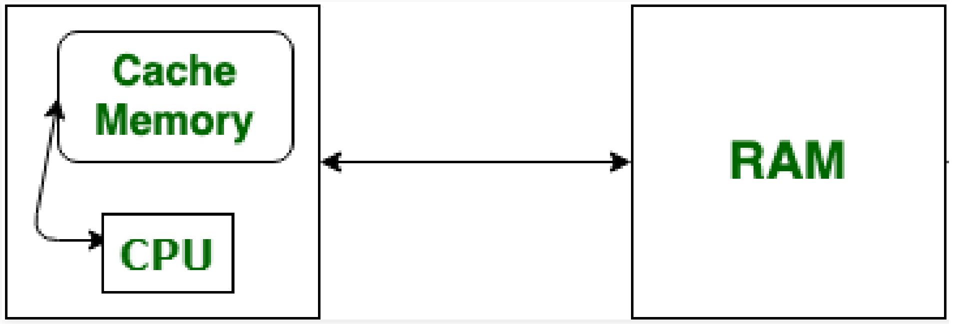
Robotic Application

🖼️ [Images] Flaticon.com – Freepik, Freepik, Smalllikeart

Optimization
Why memory layout matters?

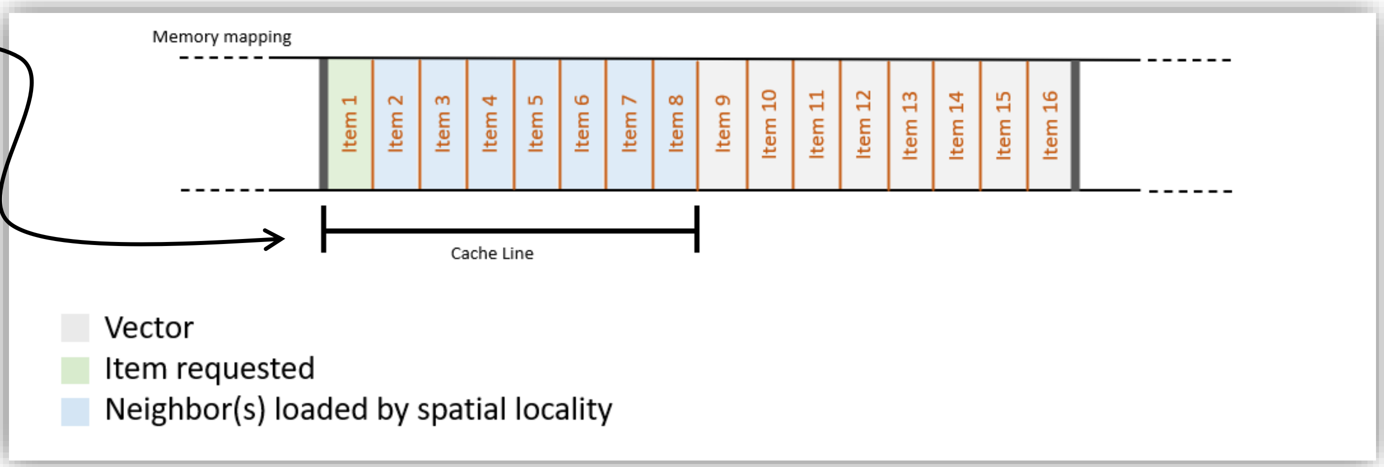
Robot controller: limited amount of compute available, additional hardware is costly

→ Optimizations can be applied



CPU cache is ~100 times faster than DDR memory[1]

Load RAM to Cache: organized in blocks of ~2 KB



[1, Website] performance - Approximate cost to access various caches and main memory?
[2, Specs] Dataset: 20 features, for classification 5 labels. Cache line of 4KB.

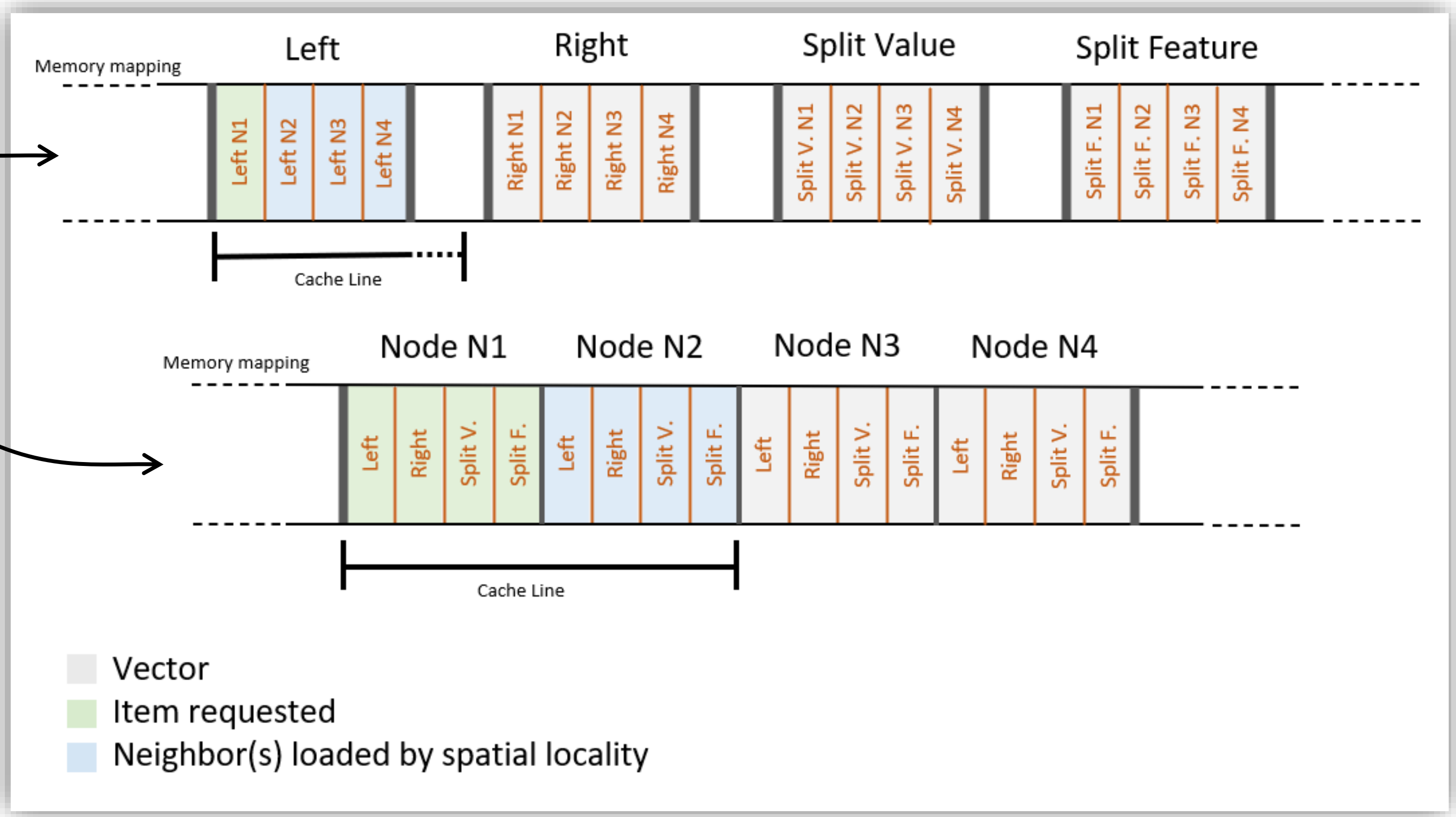
Optimization
New memory layout matters?

Representation of a decision tree in memory:

- Vector
- Struct

One cache line[1]:

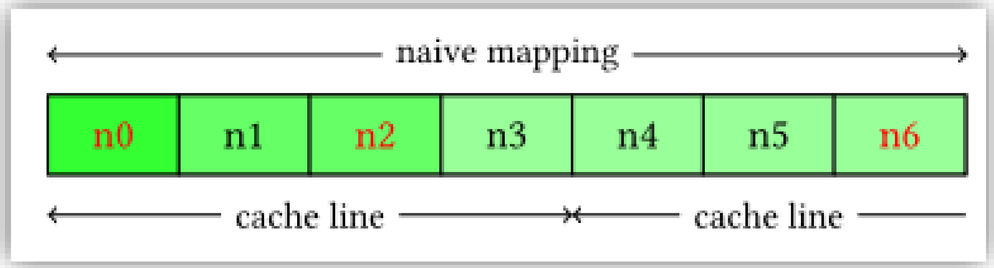
- Classification: 7 nodes
- Regression: 29 nodes

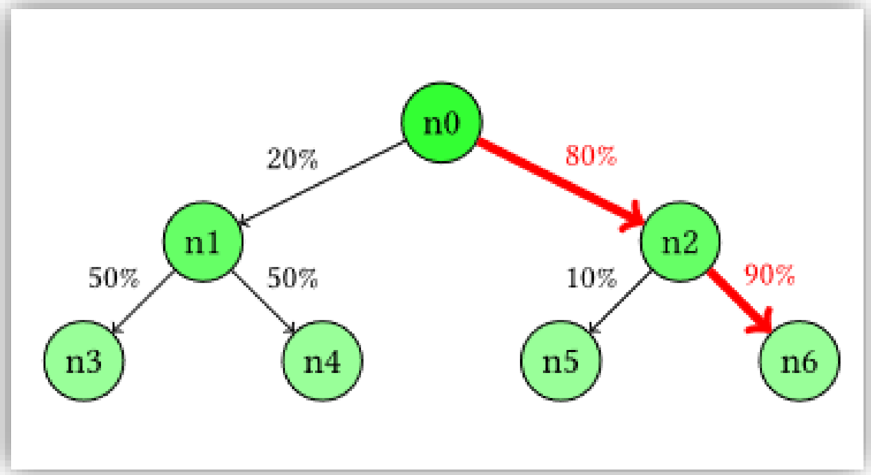


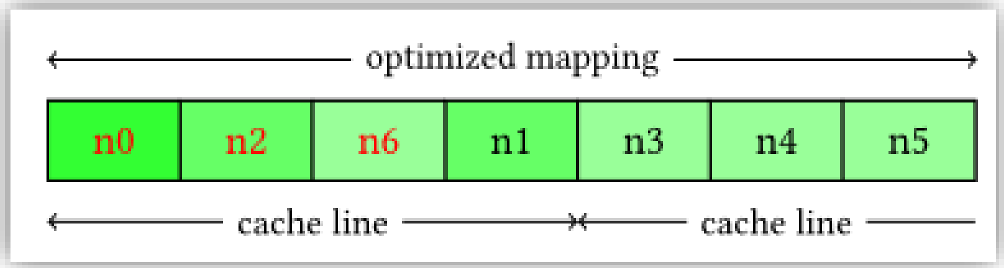
[1, Website] performance - Approximate cost to access various caches and main memory?
[2, Specs] Dataset: 20 features, for classification 5 labels. Cache line of 4KB.

Optimization
Applying optimizations

Decision Tree: Sorting nodes by probability







Related works: Only Offline
→ Our contribution: Implement Online

[*, Picture] Very simplified version of the algorithm. The algorithm is more complex than this.

[1, Picture, Paper] Chen, K.-H., *et al.* (2022). Efficient Realization of Decision Trees for Real-Time Inference. *ACM Transactions on Embedded Computing Systems*, *21*(6), 1–26. https://doi.org/10.1145/3508019

Optimization
Results

Dataset: 500.000 samples

Time for each iteration
“without optimizations” vs “with optimizations”

↓ No sort

↓ Sort every 1 000 iterations
(optimization time is removed)

Performance is chunked every 25 000 iterations

Result: each iteration is in median

18% faster for Regression

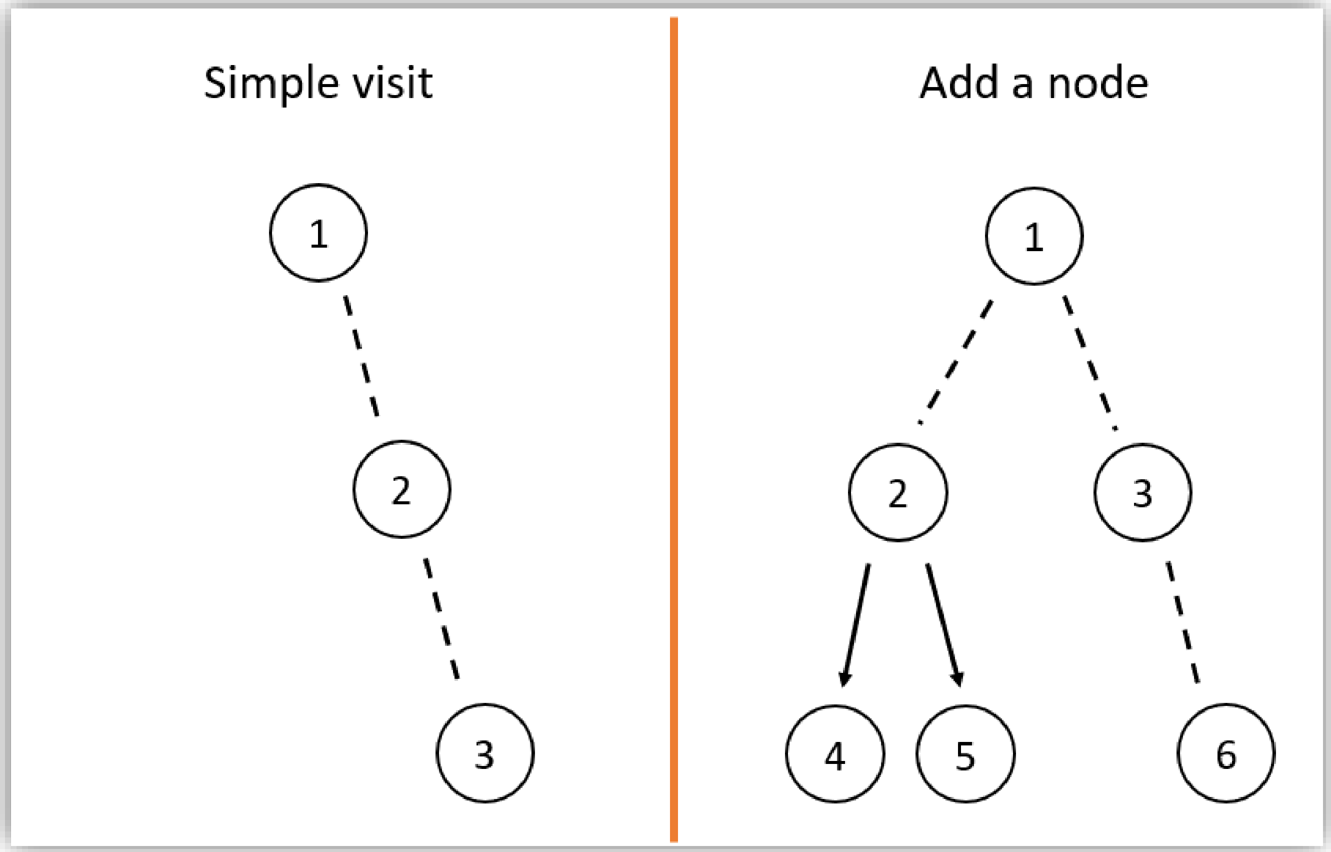
8% for Classification

Optimization

Node access patter

Mondrian Forest behavior:

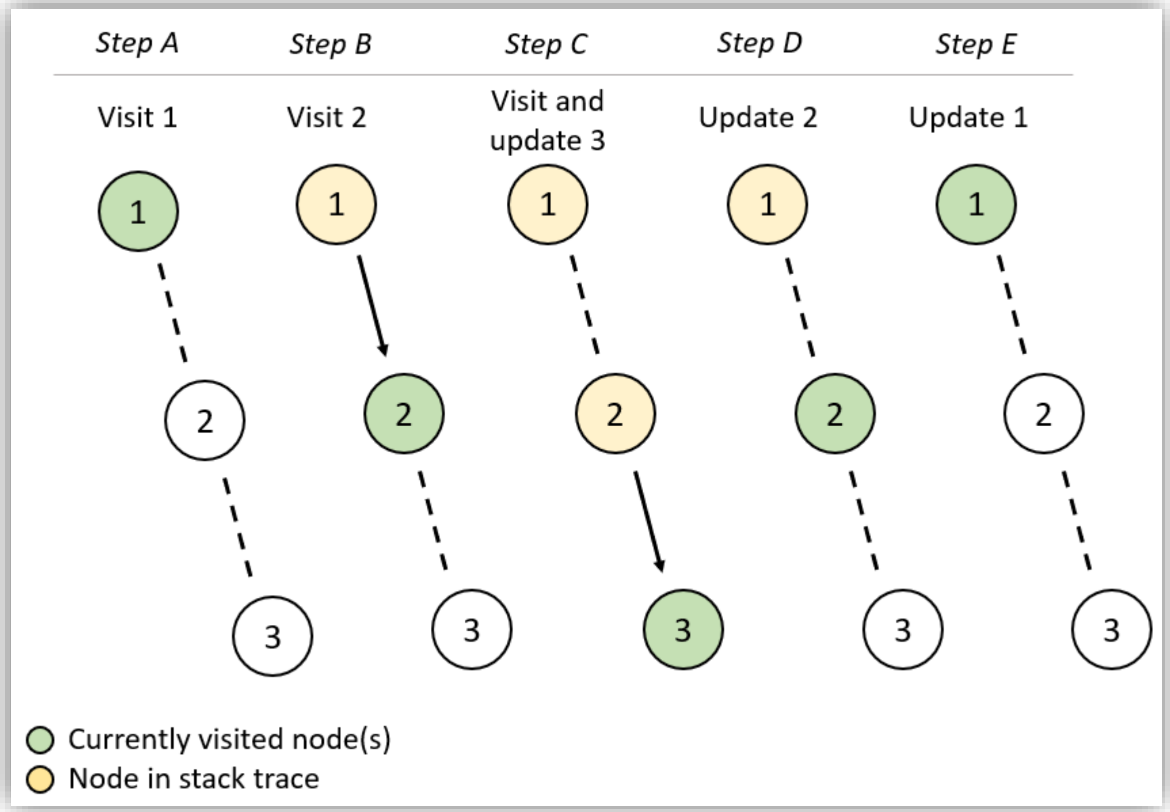
- Simple Visit - Inference
- Add a node - Train



Optimization
Node access patter

Case: Simple visit

Spatial locality works when doing a simple visit root to leaf.

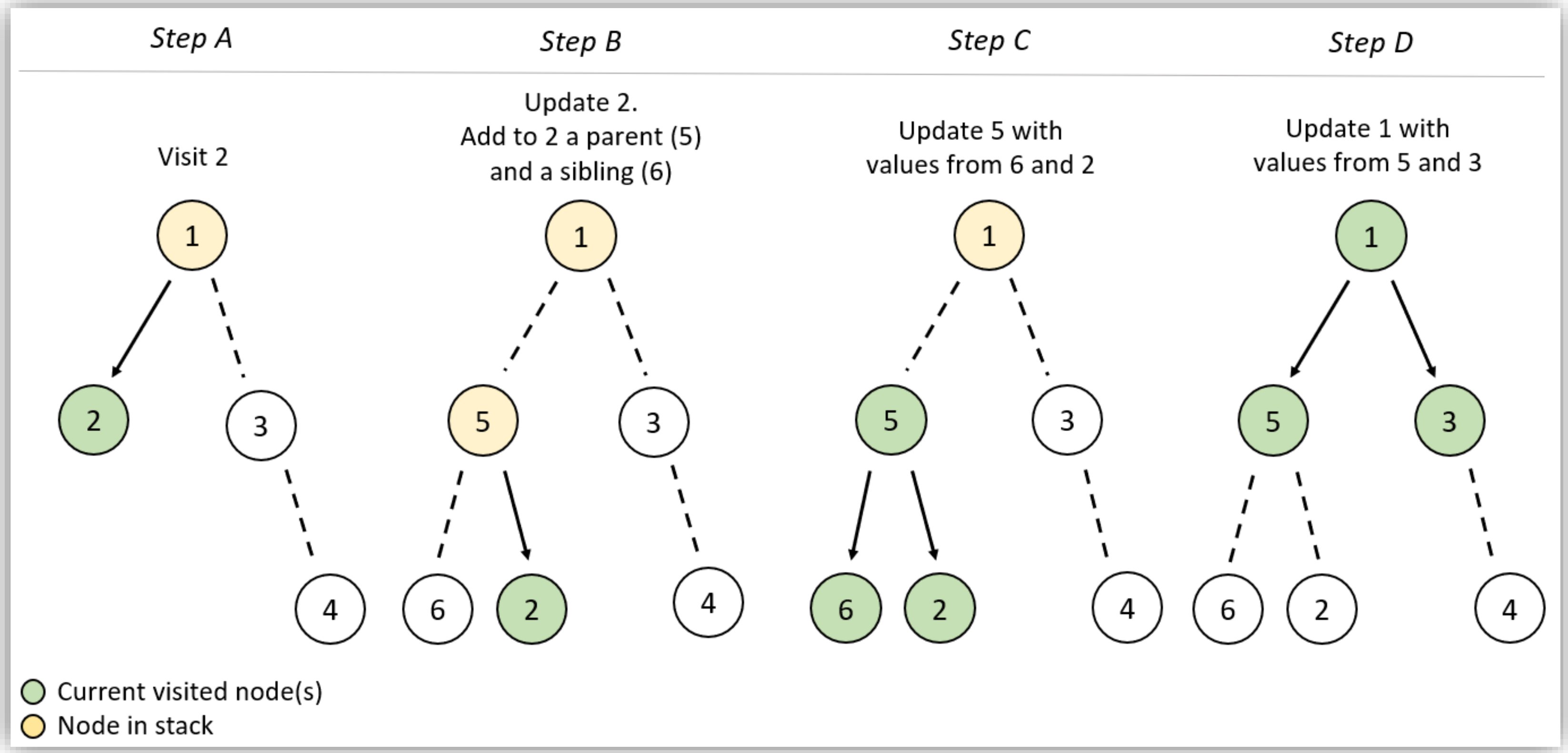


Node access patter

Node access patter

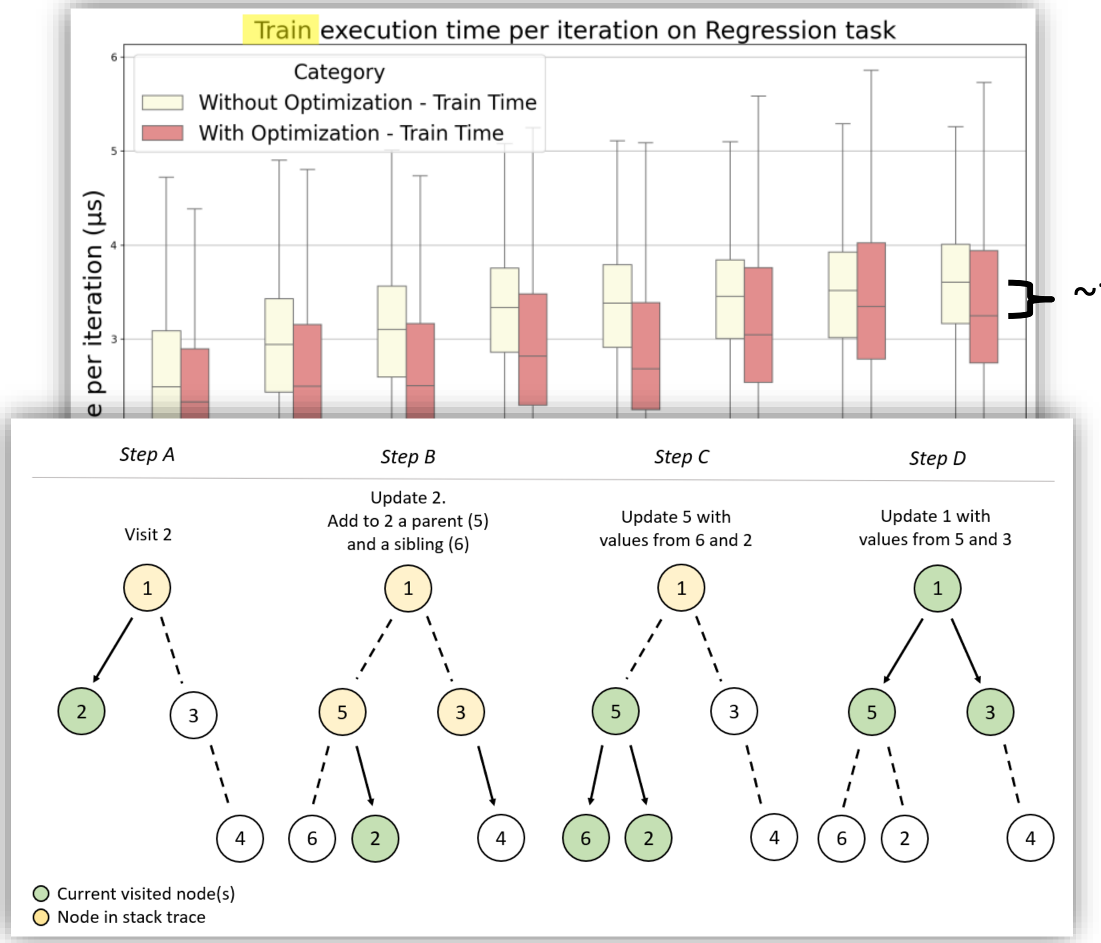
Case: Add a node

Spatial locality never used in this case.

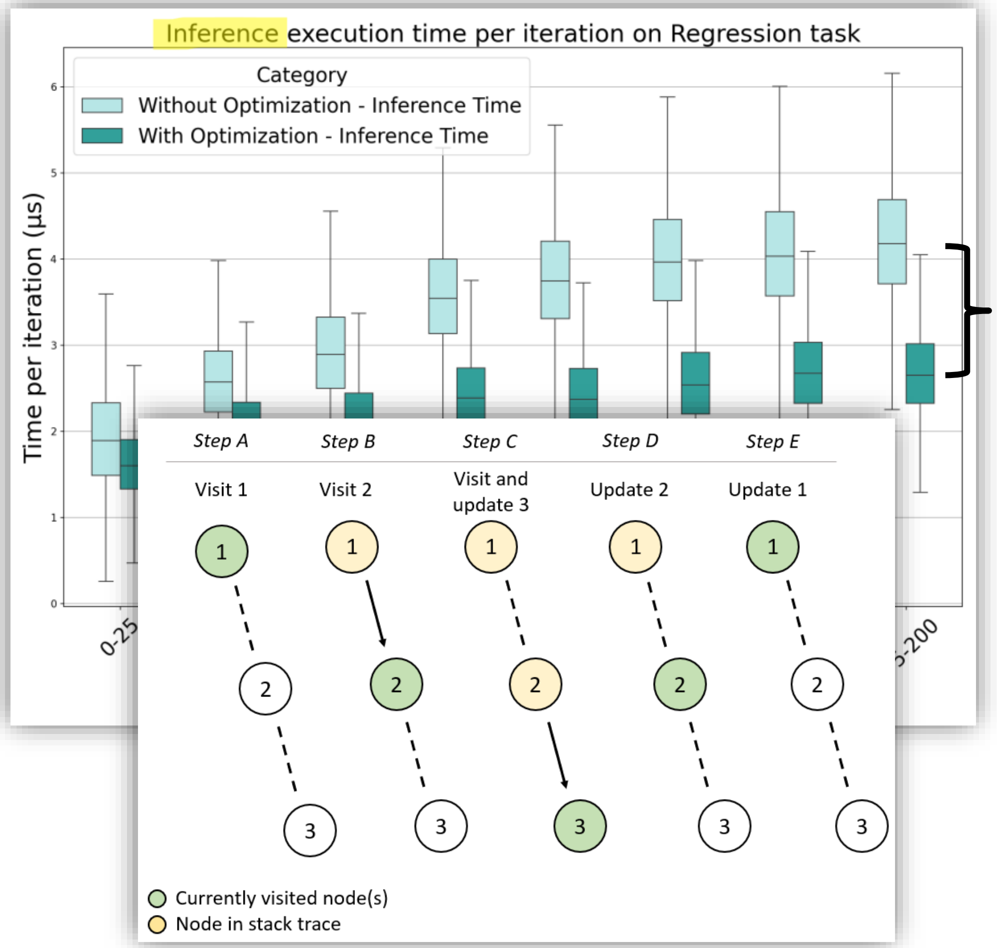


Why are optimizations working?

Breaking down the execution: Train vs Inference



~7%



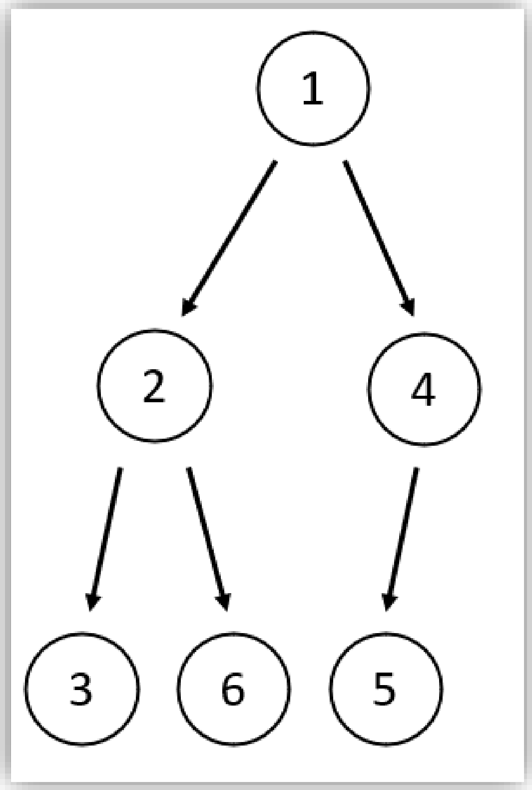
~34%

Are nodes accessed sequentially?

Are we decreasing the number of non-sequential accesses?

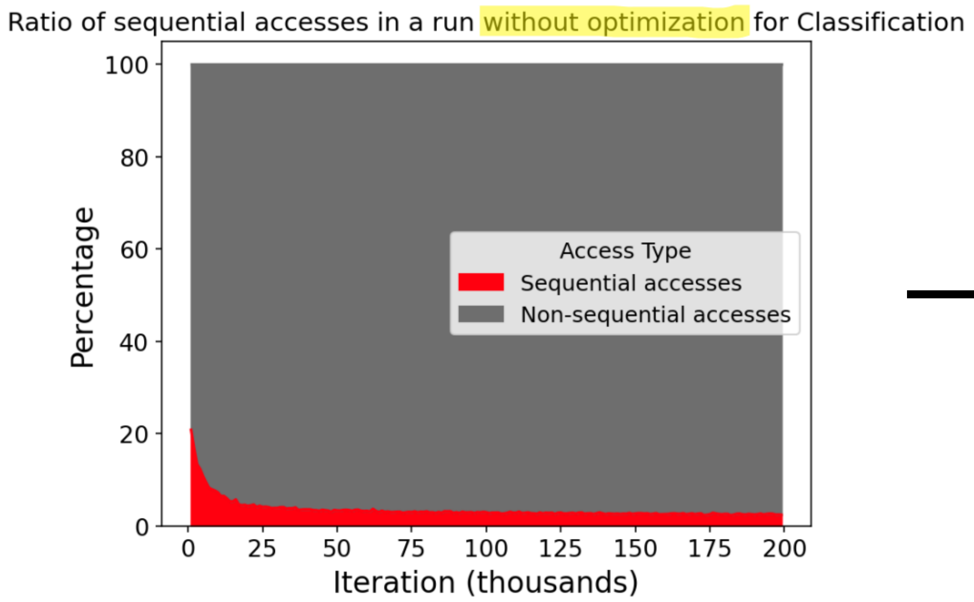
e.g., sequence [1 → 2 → 3] → [1 → 2 → 6] → [1 → 4 → 5]

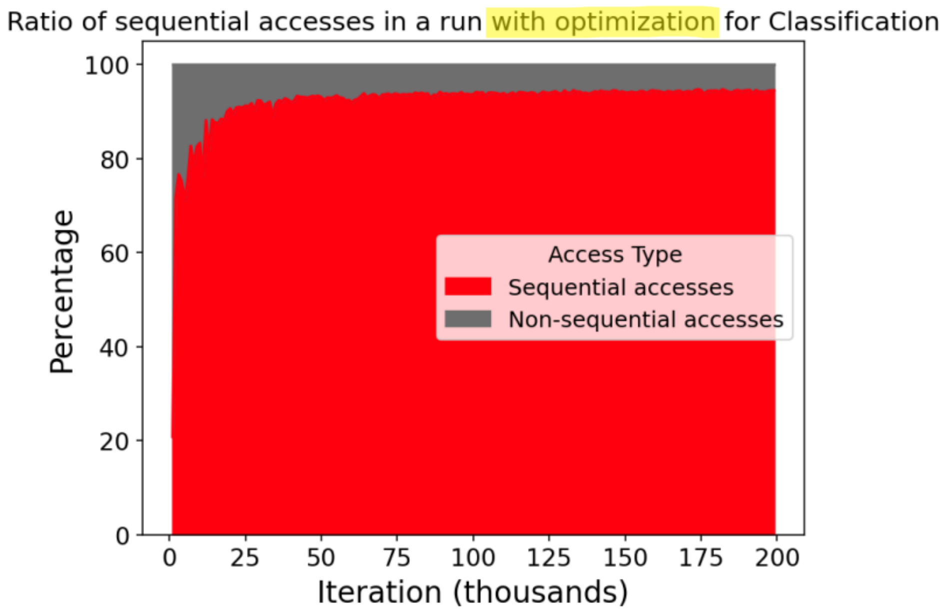
- 4 nodes "sequentially access"
- 5 nodes "non-sequential access"



Are nodes accessed sequentially?

Optimizing every 1000 iterations.





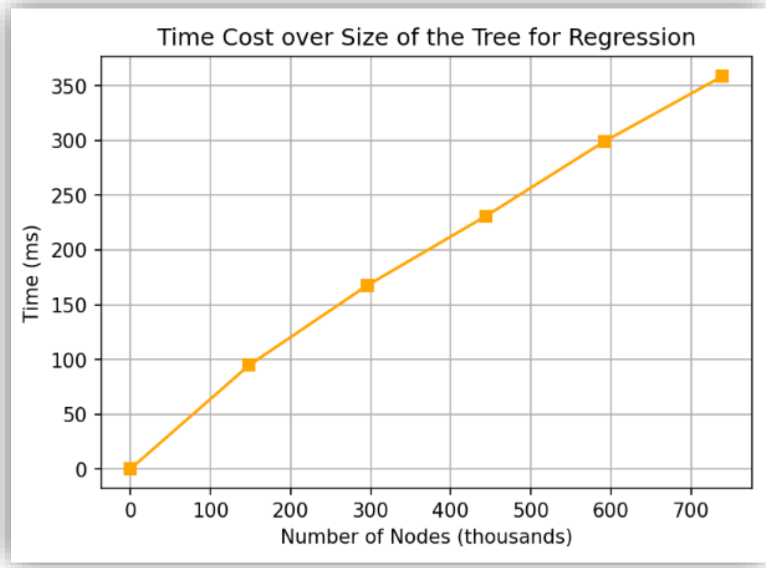
Optimization
Sorting Cost vs. Gain

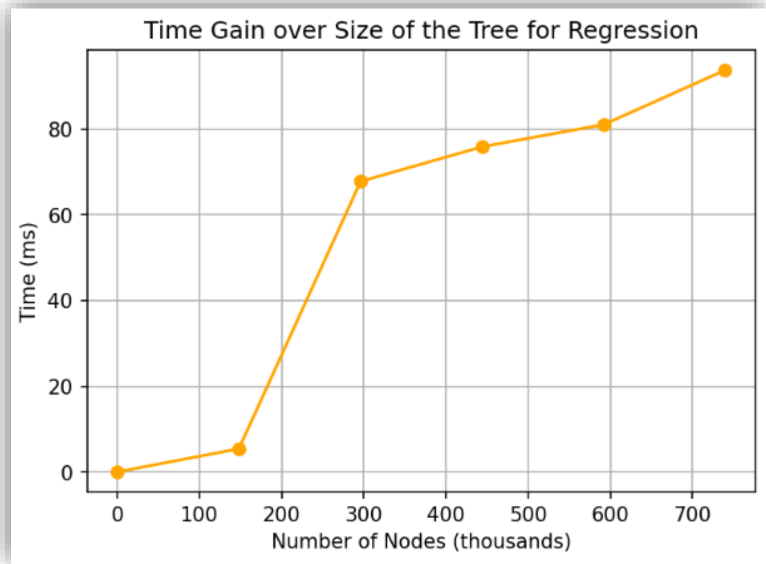
Dataset: 500.000 samples

Sort every 100.000 sample

Cost: 0.36 seconds

Gain: 0.09 seconds





Model add nodes: 74% of the times

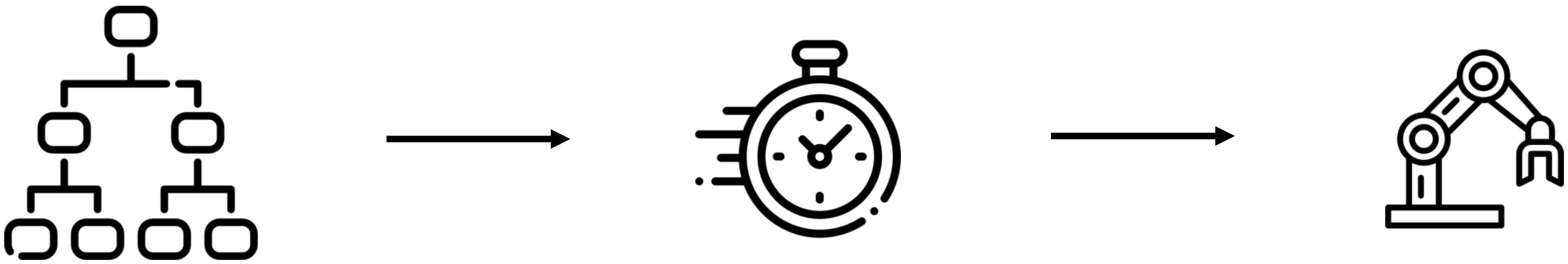
Assuming linearity in Cost and Gain:

- We have a gain when adding nodes less often than 16% of the times

Project Outline

The 3 phases

Model Development | Optimization | Robotic Application



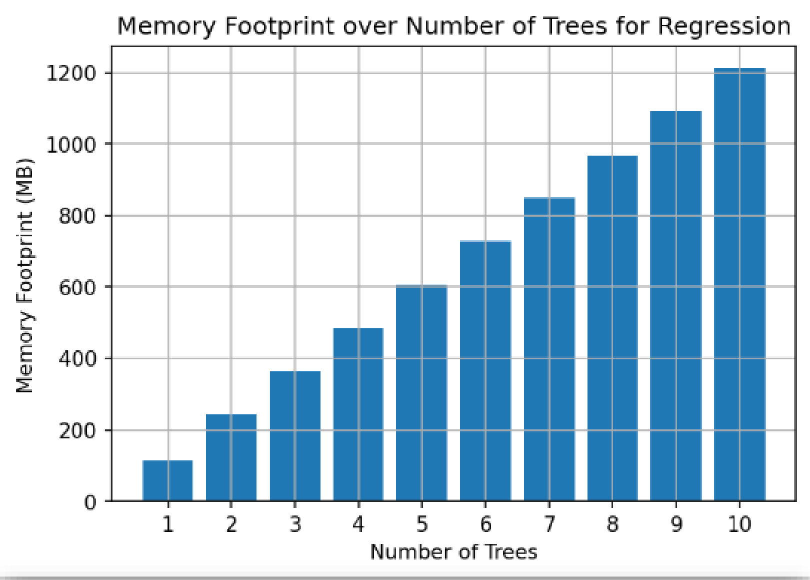
[Images] Flaticon.com – Freepik, Freepik, Smalllikeart

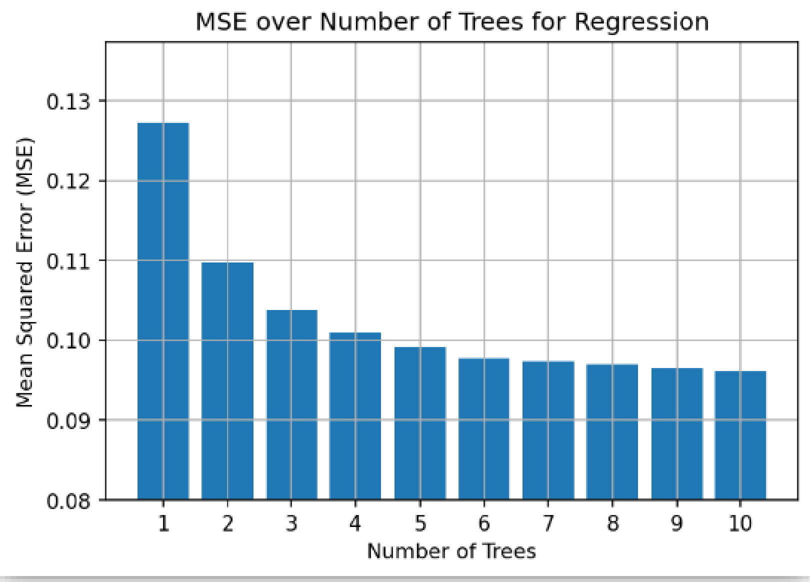
Robotic Application

Raising number of trees of the forest

- Memory footprint: increases linearly (121 ± 3 MB)
- Accuracy / MSE: have diminishing returns

Depending on the memory availability -> Choose necessary size of forest
e.g. 500 MB available => 4 trees





Future works

Problem: When we apply the optimizations, the robot freezes

- Apply it in background
- Sort in iteration idle time

Model optimization:

- Floating Point 32 -> 16
- Model Limitation: Limit number of nodes

Problem: When the robot turns off, we lose the progress

- Export/Load Weights

Acknowledgements

UNIVERSITY OF TWENTE.

Kuan Chen
Supervisor

River

Saulo Martiello Mastelini
Code review

ABB

Jordis Herrmann
Supervisor

Mikael Norrlöf
Advisor

Jonathan Styrud
Advisor

Adil Zouitine
Code review

Max Halford
Advisor

ABB