*IWST22 — International Workshop on Smalltalk Technologies Novi Sad, Serbia*

# How Fast is AI in Pharo? Benchmarking Linear Regression


**Oleksandr ZAITSEV**[1,2], **Sebastian JORDAN MONTAÑO**[2], Stéphane DUCASSE [2]

[1]Arolla, Paris

[2]Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL

oleksandr.zaitsev@arolla.fr

# Training machine learning models


Fast

*Scikit-learn*
*(Python)*

Why?

Very slow!

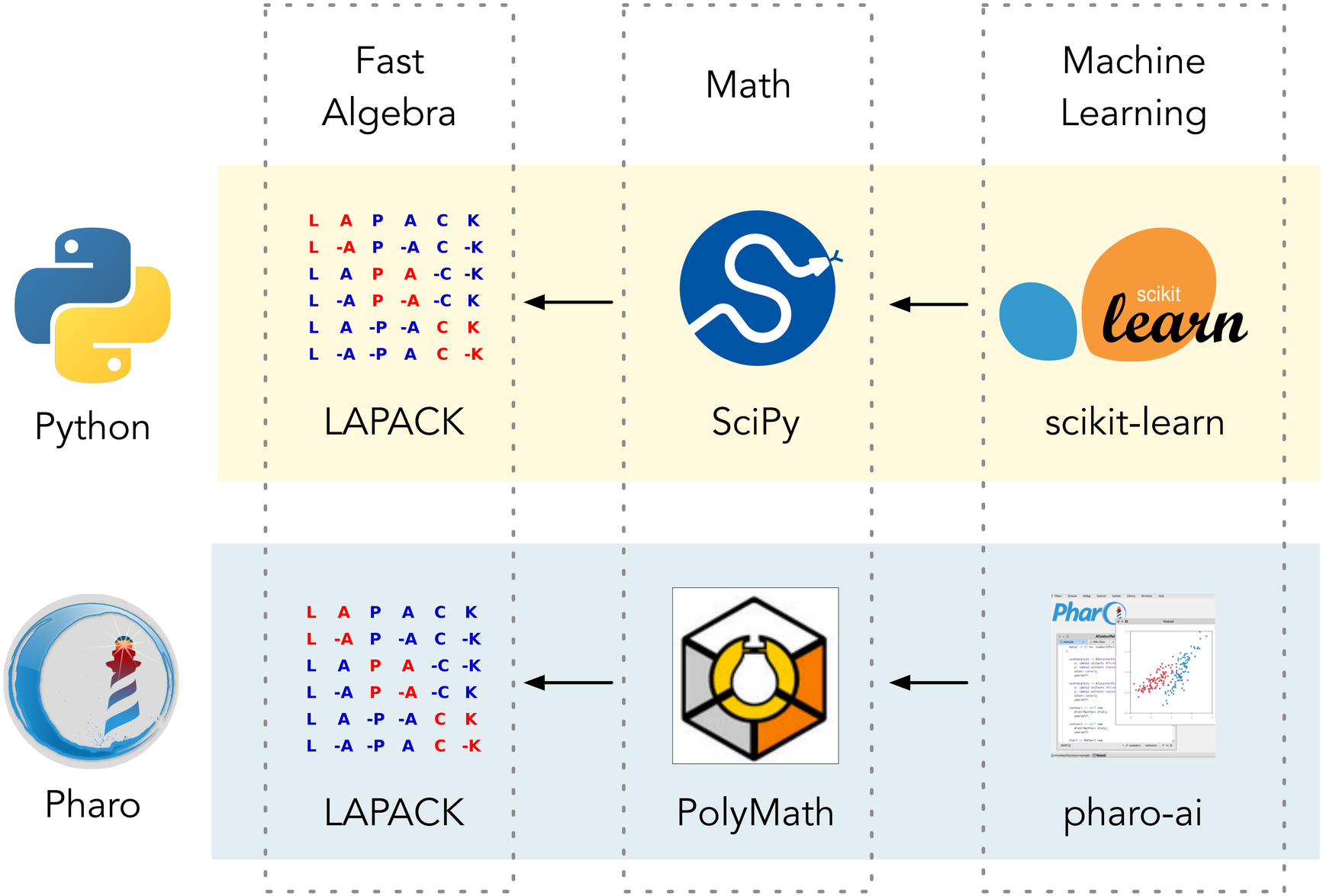
*pharo-ai*

# Training machine learning models


L A P A C K
L -A P -A C -K
L A P A -C -K
L -A P -A -C K
L A -P -A C K
L -A -P A C -K

Can we do the same in Pharo?


We Can Do It!
POST FEB. 15 TO FEB. 28
WAR PRODUCTION CO-ORDINATING COMMITTEE

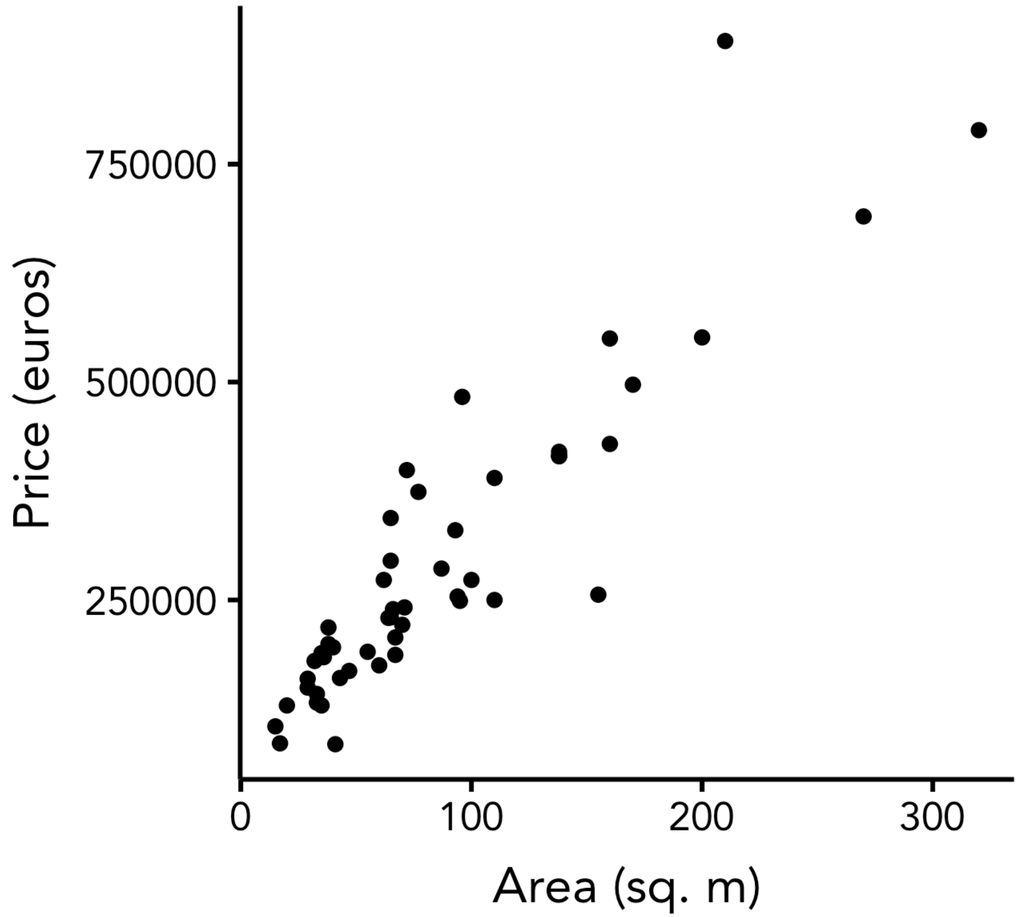

Fast Algebra
Math
Machine Learning
L A P A C K
L -A P -A C -K
L A P A -C -K
L -A P -A -C K
L A -P -A C K
L -A -P A C -K
Python
LAPACK
SciPy
scikit-learn
Pharo
LAPACK
PolyMath
pharo-ai

We want to provide a proof of concept that math & AI algorithms in Pharo can be boosted by delegating time-consuming operations to highly-optimised external libraries.

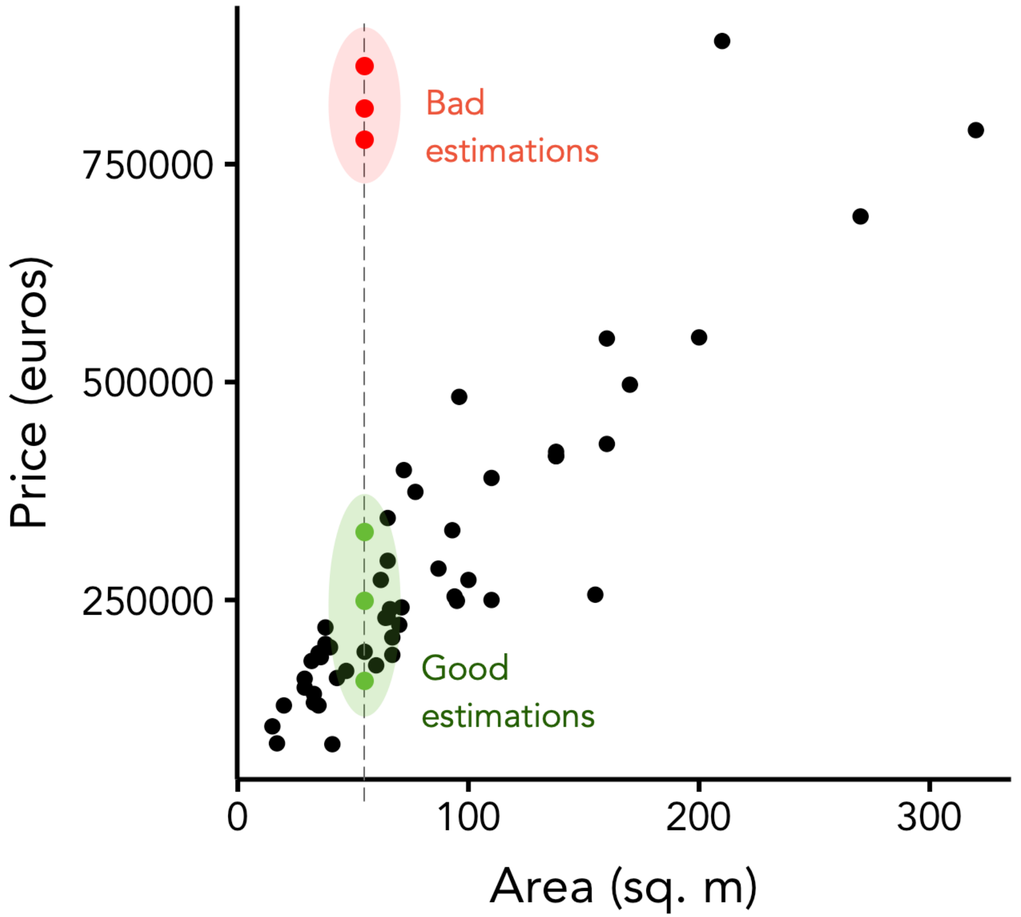
To do that, we build a prototype implementation of linear regression based on the DGELSD routine of LAPACK


# Linear regression and how it is implemented

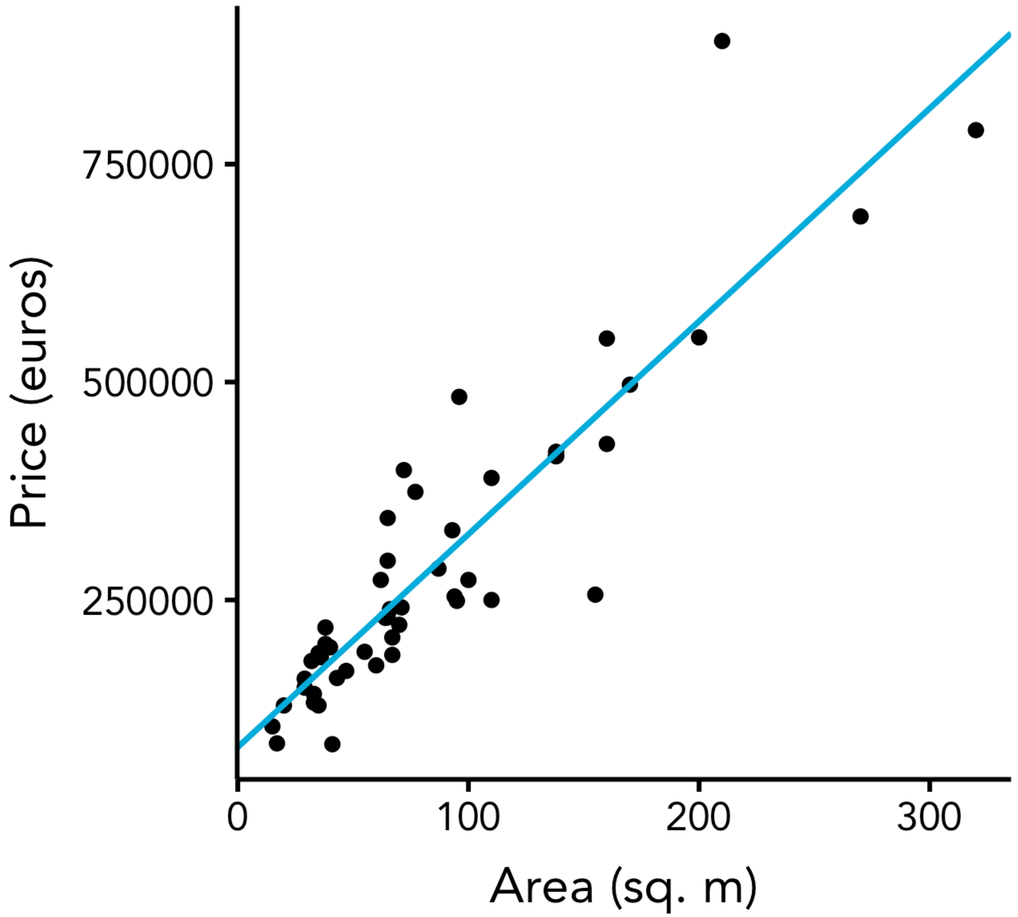
# Linear Regression Problem



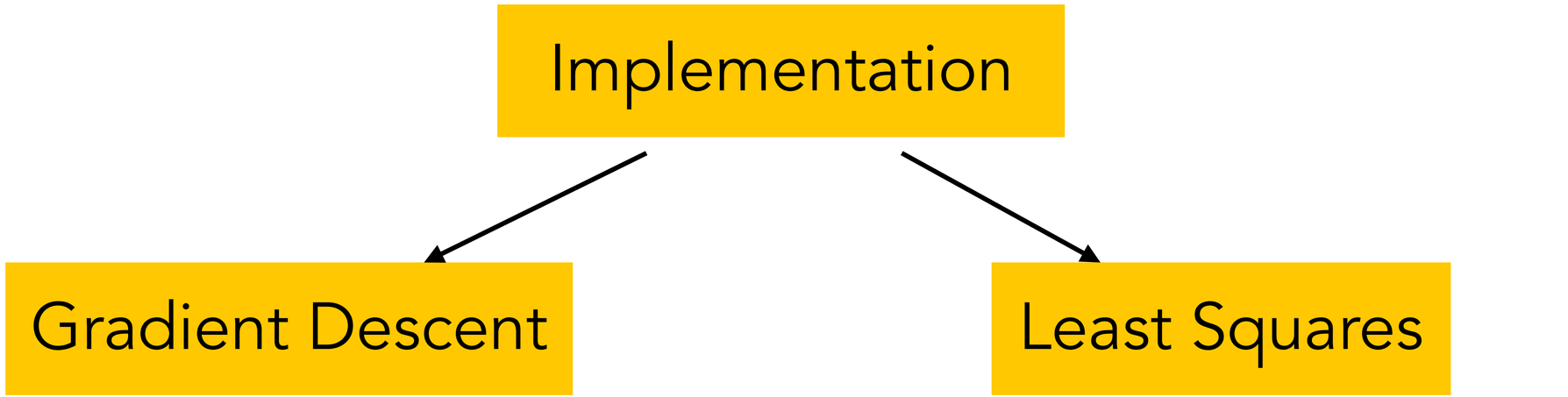
# Linear Regression Problem



# Linear Regression Problem



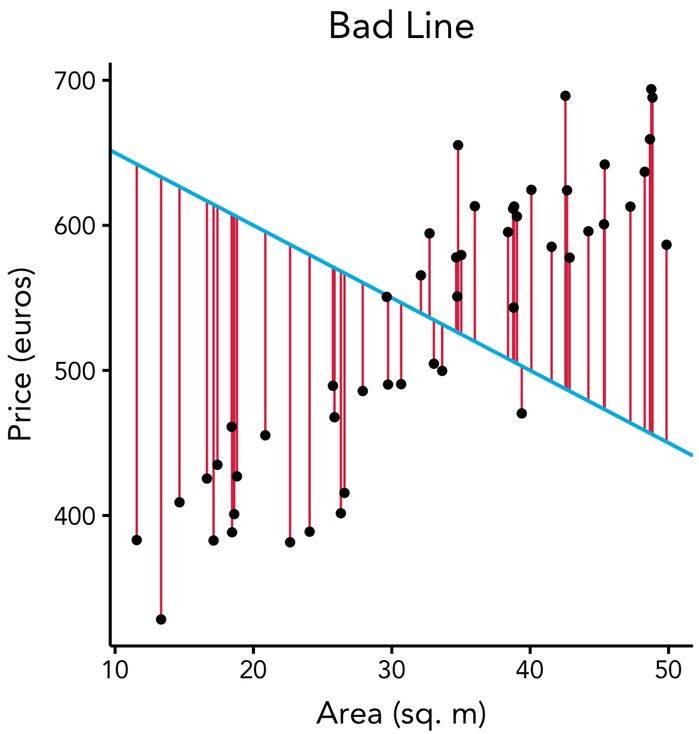
# Linear Regression Implementations

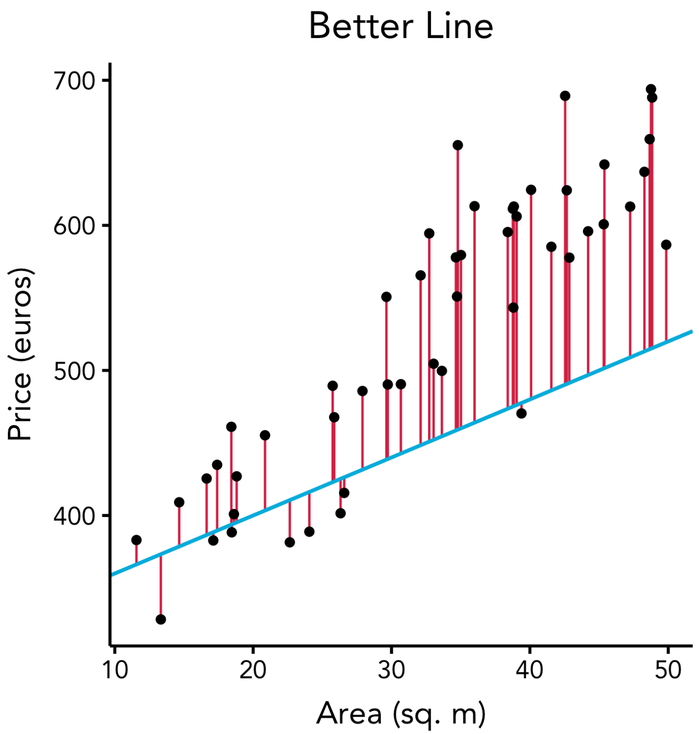


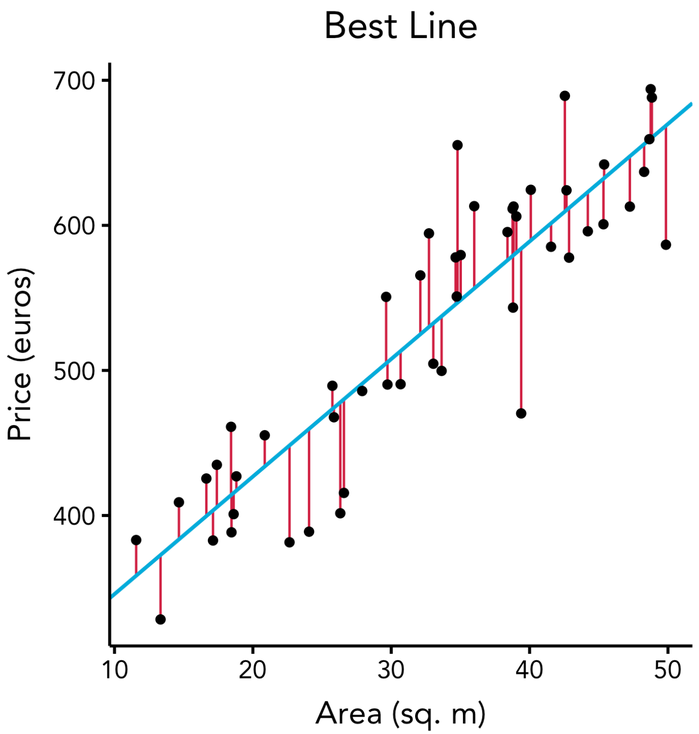

- Iterative
- Approximate

- One iteration
- Exact solution

# **Implementation 1.** Gradient Descent



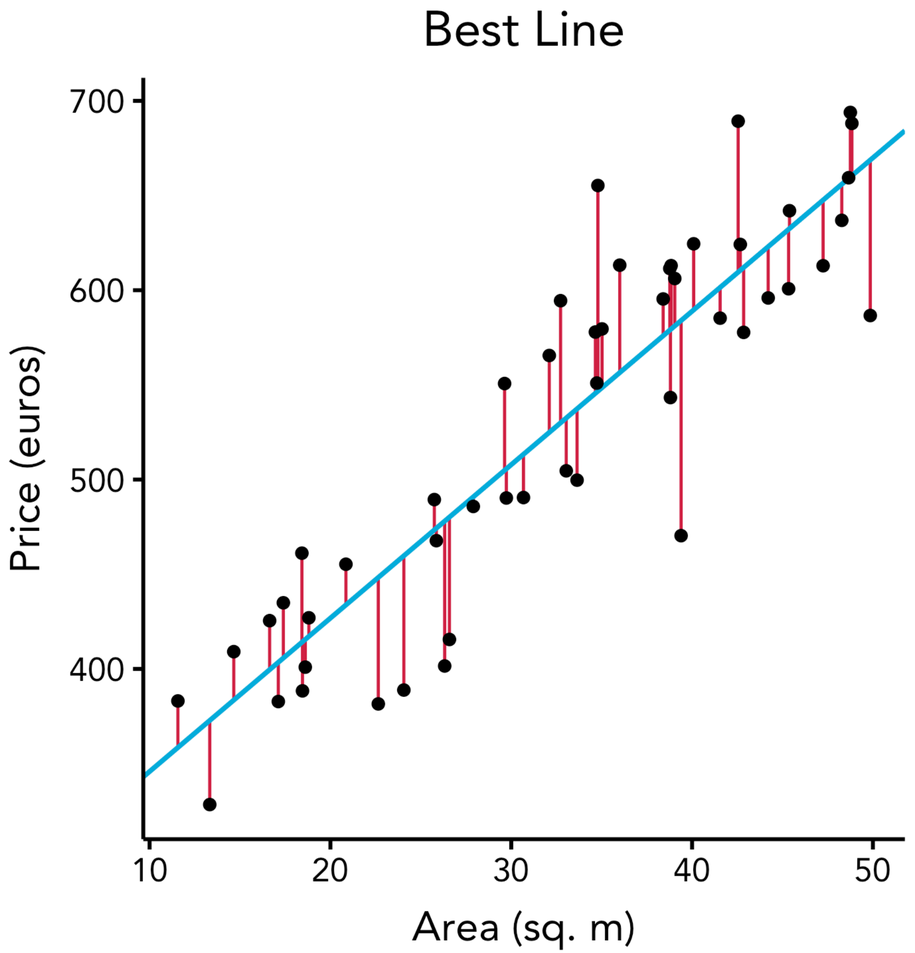
# **Implementation 1.** Gradient Descent

Need to minimise the errors

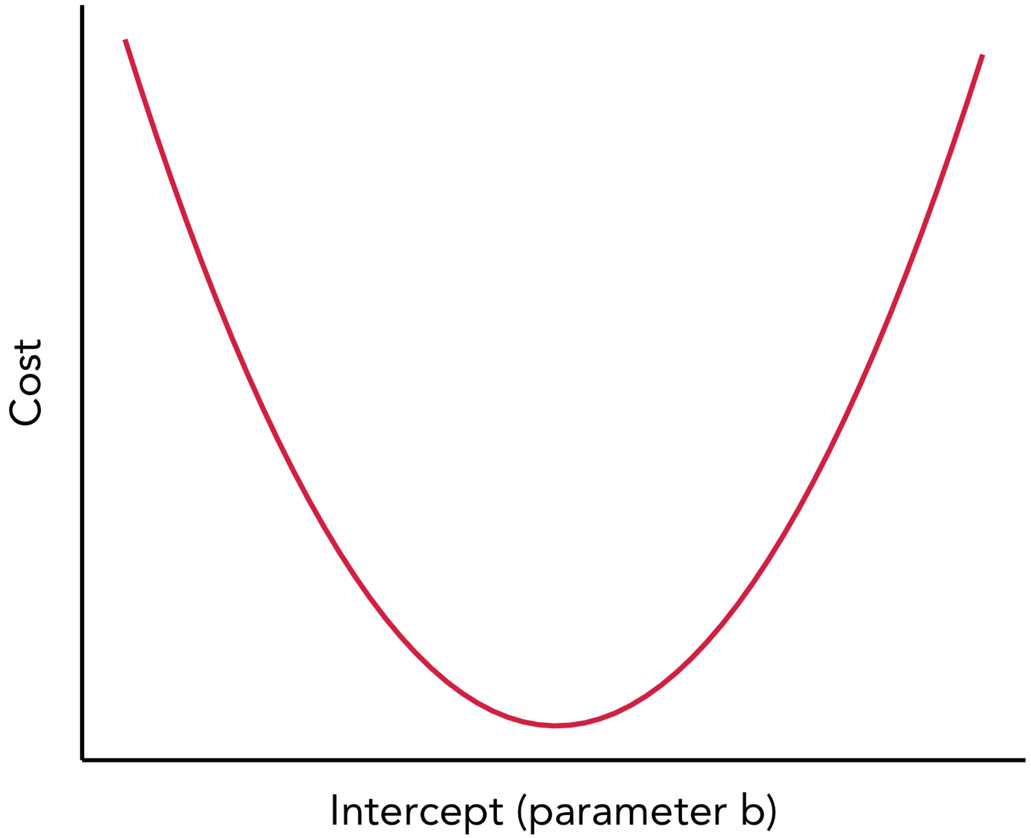
$$e_i = \hat{y}_i - y_i$$

Cost function
- mean squared errors

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} e_i^2$$

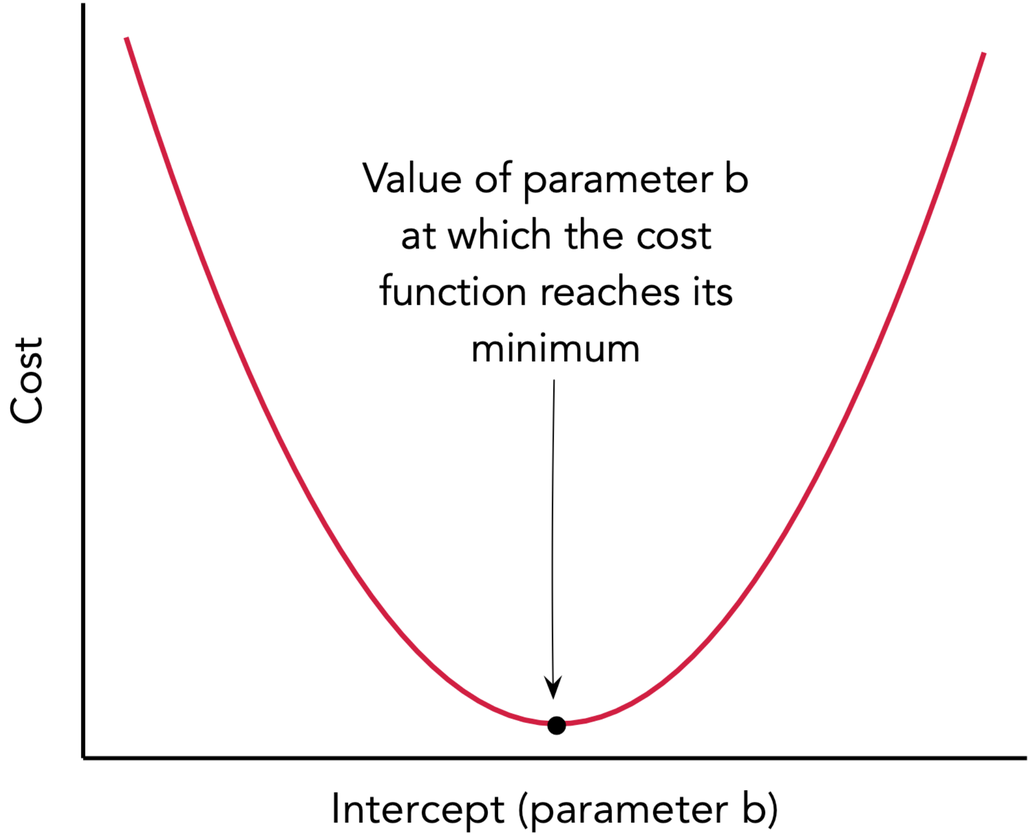


# **Implementation 1.** Gradient Descent



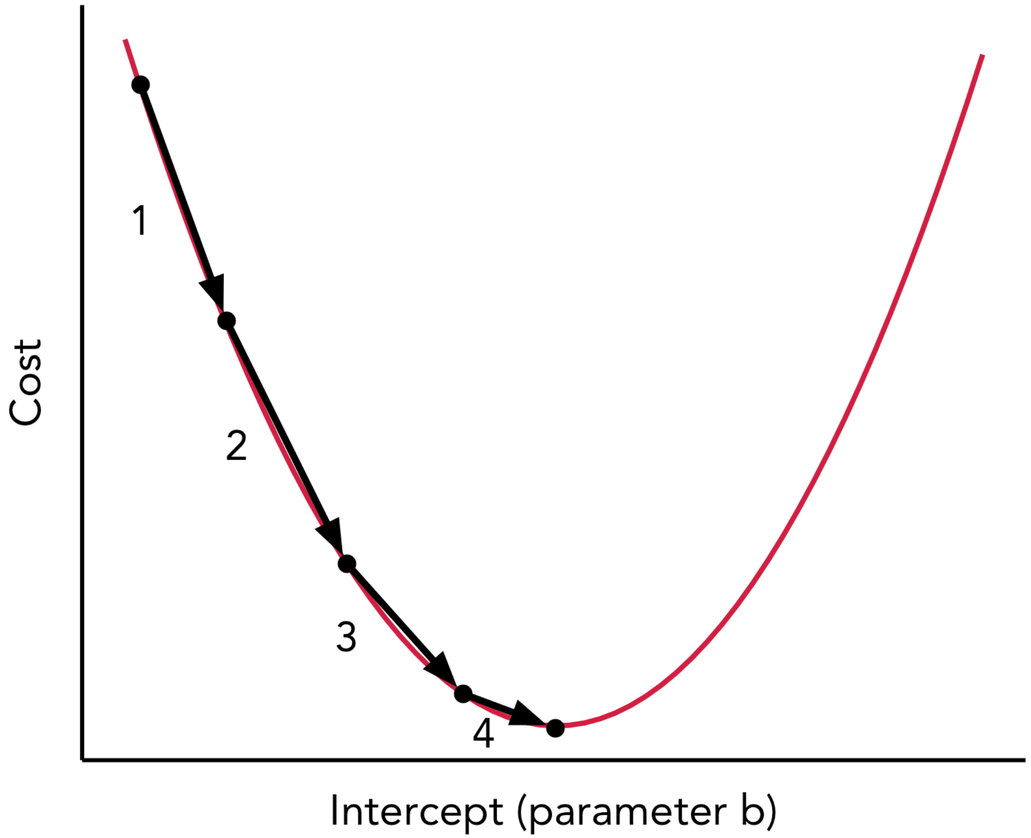

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

# **Implementation 1.** Gradient Descent




$$J(w) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

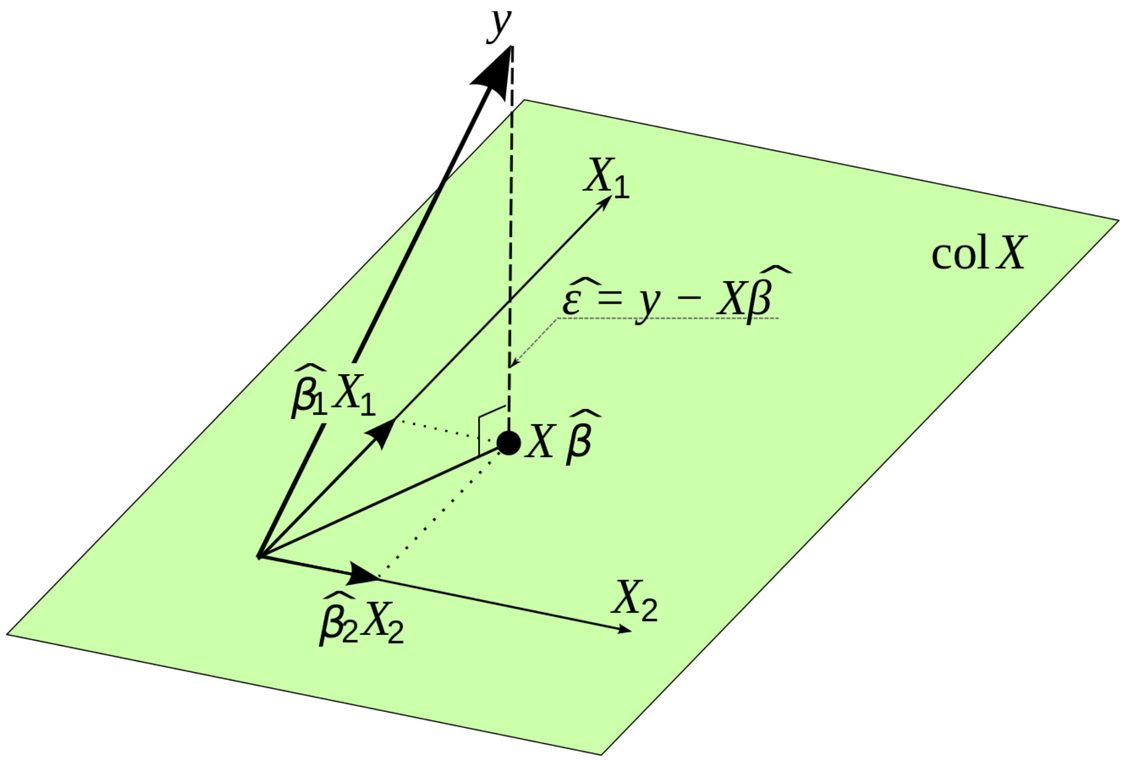
# **Implementation 1.** Gradient Descent




$$J(w) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

$$\theta^{(new)} = \theta^{(old)} - \alpha\frac{\partial}{\partial\theta_i}J(\theta)$$

# **Implementation 2.** Least Squares



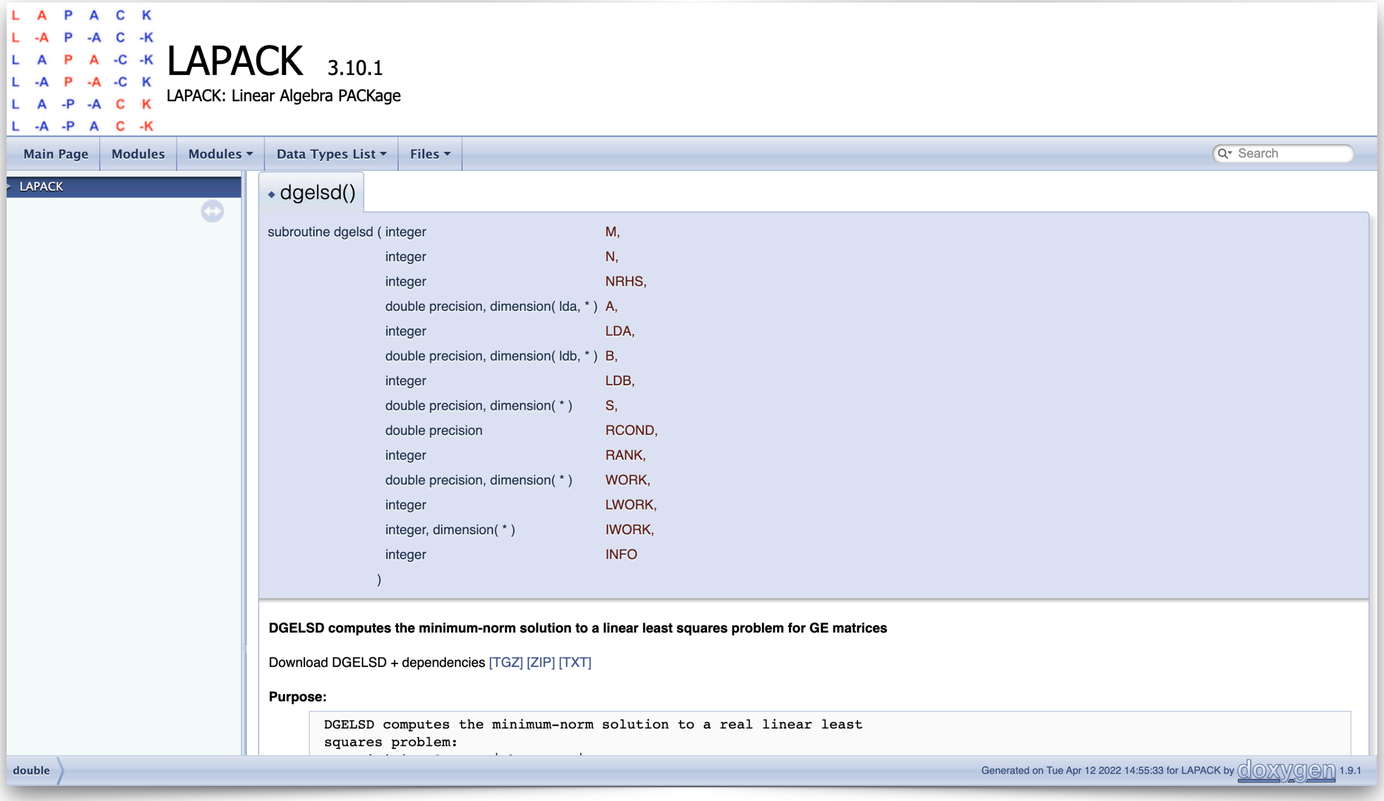

$$\hat{\theta} = argmin\,||y - X\theta||_2$$

Orthogonal projection

# **Implementation 2.** Least Squares

Implemented as
DGELSD routine
In LAPACK

We can call it
through FFI



Demo will come in the end

# Wait for it!


# Experiment

# Datasets

| Name | Columns | Rows | Size |
|---|---|---|---|
| Small | 200,000 | 20 | 82 Mb |
| Medium | 1,000,000 | 20 | 411 Mb |
| Large | 5,000,000 | 20 | 2.06 Gb |

# Research Questions

- **RQ.1** - **Measuring LAPACK speedup.**
  How much time improvement can we achieve by calling LAPACK from Pharo?

- **RQ.2** - **Comparing to scikit-learn.**
  How does Pharo & LAPACK implementation compare to the one provided by scikit-learn?

- **RQ.3** - **Comparing pure Pharo with Python.**
  How does pure Pharo implementation of linear regression compare to equivalent pure Python implementation?

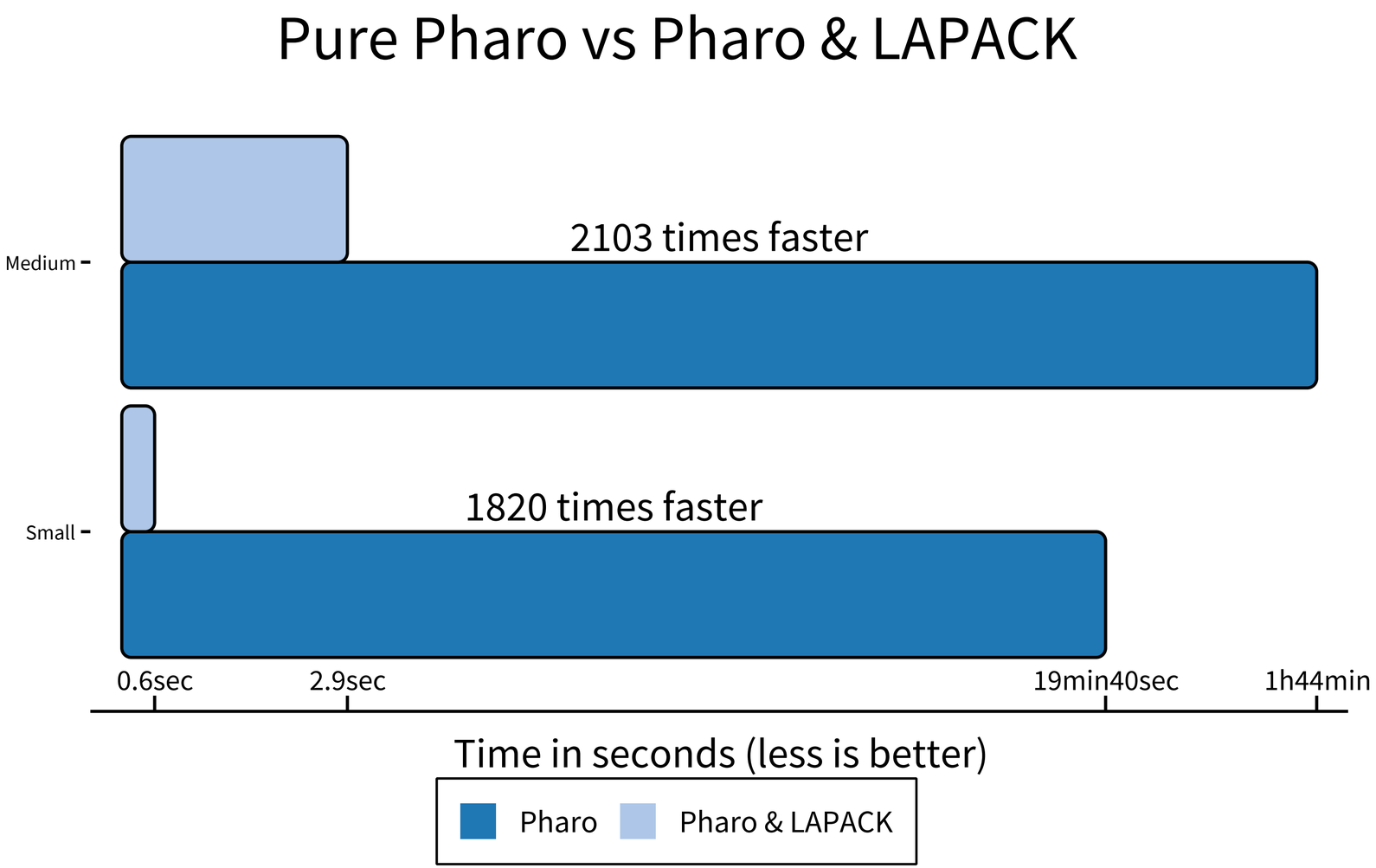
# RQ.1 - Measuring LAPACK speedup

**Table 2**
The speedup achieved by calling LAPACK routines fom Pharo.

| Dataset | Pharo | Pharo & LAPACK | Diff |
|---|---|---|---|
| Small | 00:19:39.090 | 00:00:00.648 | 1820× |
| Medium | 01:43:59.000 | 00:00:02.967 | 2103× |
| Large | ∞ | 00:00:15.676 | — |

# RQ.1 - Measuring LAPACK speedup

Help

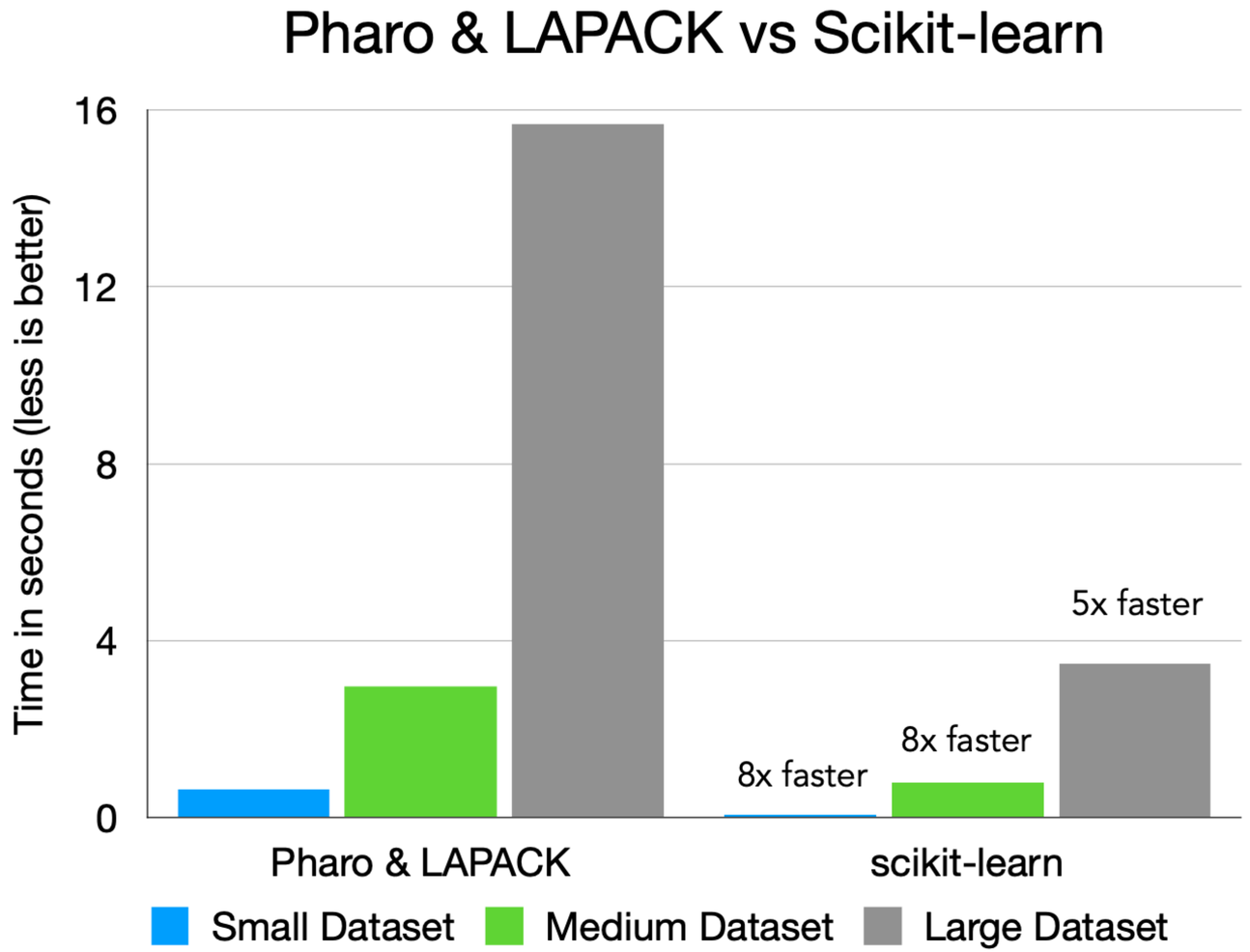


# RQ.2 - Comparing to scikit-learn.

**Table 3**
Comparison with scikit-learn (both implementations use LAPACK).

| Dataset | Pharo & LAPACK | scikit-learn | Diff |
|---|---|---|---|
| Small | 00:00:00.648 | 00:00:00.079 | 8× |
| Medium | 00:00:02.967 | 00:00:00.790 | 8× |
| Large | 00:00:15.676 | 00:00:03.499 | 5× |

# RQ.2 - Comparing to scikit-learn.

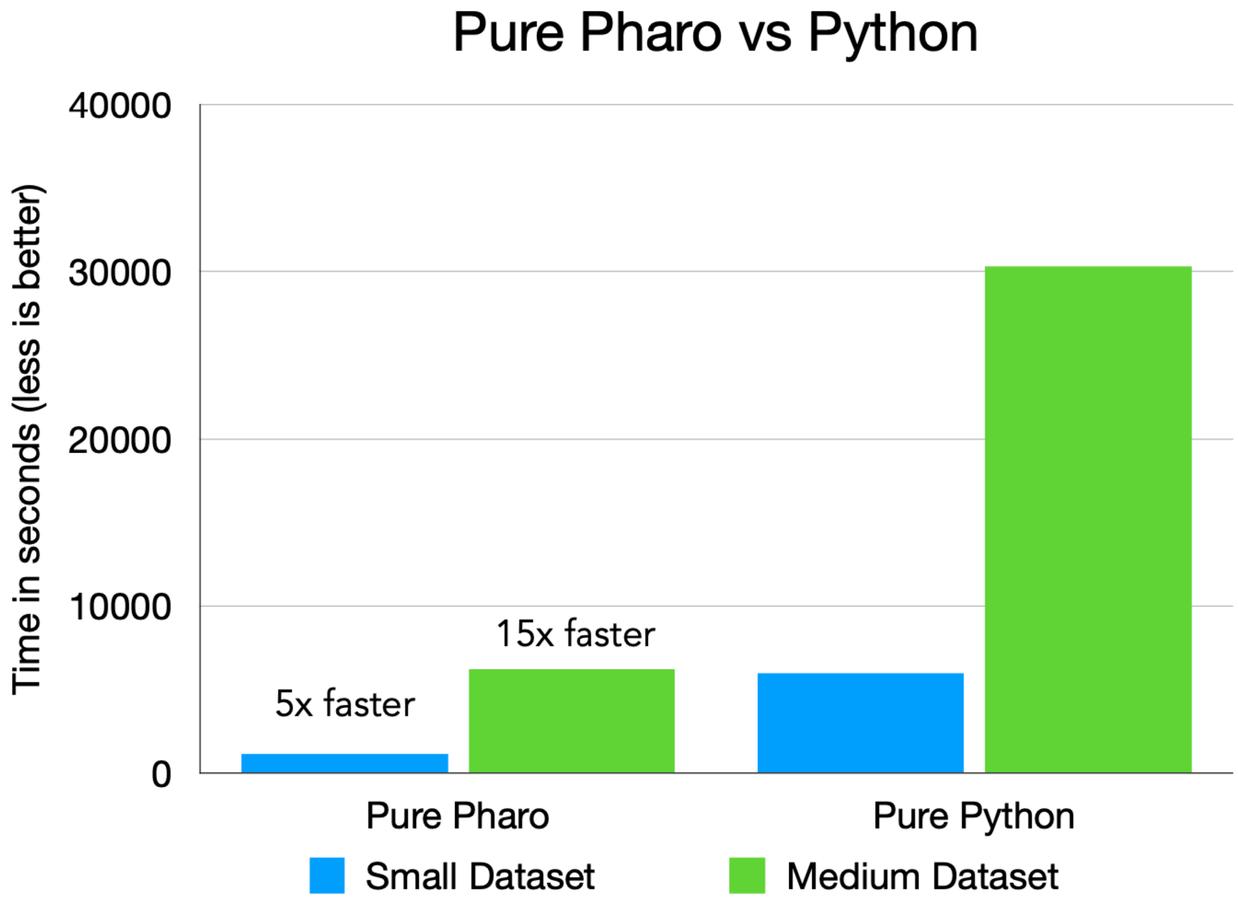


# RQ.3 - Comparing pure Pharo with Python

**Table 4**
Comparison between pure Pharo and pure Python

| Dataset | Pharo | Python | Diff |
|---|---|---|---|
| Small | 00:19:39.090 | 01:39:54.275 | 5× |
| Medium | 01:43:59.000 | 08:25:00.000 | 15× |
| Large | ∞ | ∞ | — |

# RQ.3 - Comparing pure Pharo with Python



# Summary

- We propose a prototype implementation of Linear Regression based on LAPACK
- We show that LAPACK & Pharo is up to 2103 times faster than pure Pharo
- We also show that scikit-learn is 8-5 times faster than our prototype, depending on the size of the data.
- Finally, we demonstrate that pure Pharo is up to 15 times faster than the equivalent implementation in pure Python
- Those findings can lay the foundation for the future work in building fast numerical libraries for Pharo and further using them in higher-level libraries such as pharo-ai.