

IWST22 — International Workshop on Smalltalk Technologies Novi Sad, Serbia

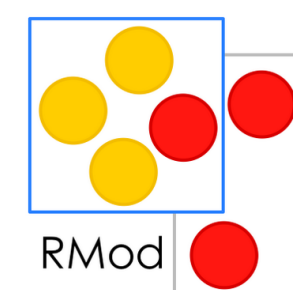
How Fast is AI in Pharo? Benchmarking Linear Regression

Oleksandr ZAITSEV^{1,2}, Sebastian JORDAN MONTAÑO², Stéphane DUCASSE²

¹Arolla, Paris

²Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRISStAL

oleksandr.zaitsev@arolla.fr



Training machine learning models



Scikit-learn
(Python)

Fast

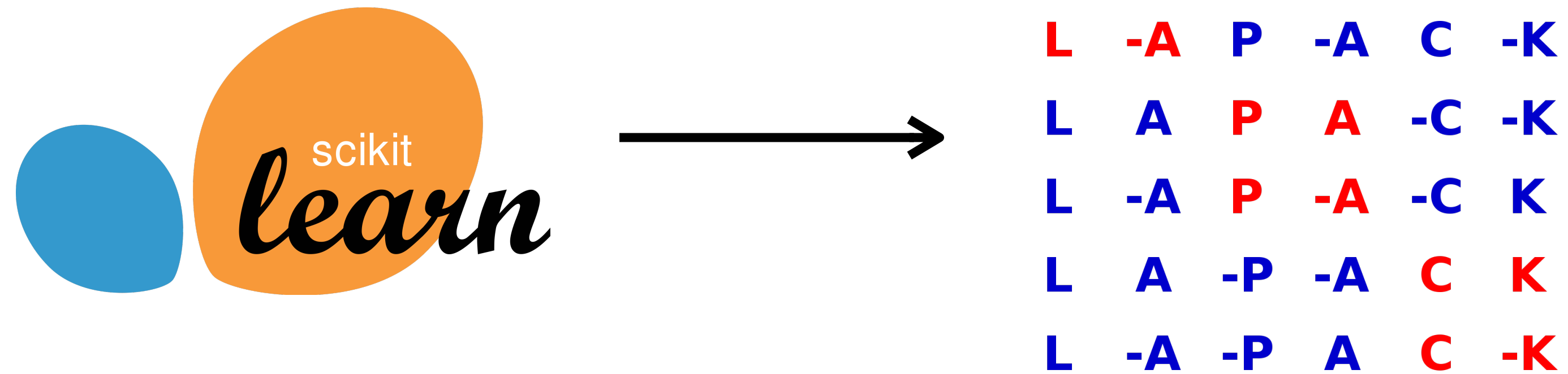


pharo-ai

Very slow!

Why?

Training machine learning models



Can we do the same in Pharo?

We Can Do It!

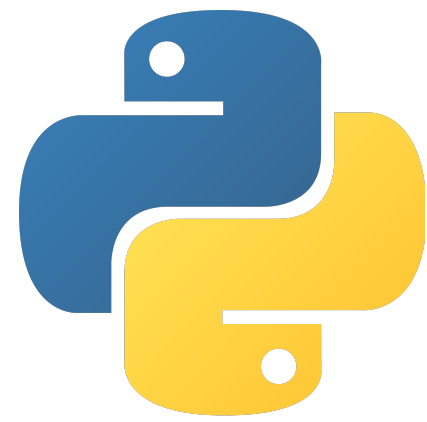


J. Howard Miller '43

POST FEB. 15 TO FEB. 28



WAR PRODUCTION CO-ORDINATING COMMITTEE



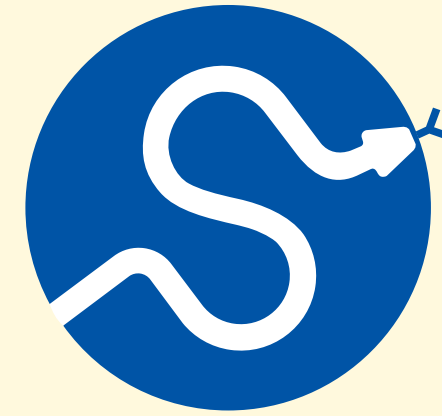
Python

Fast Algebra

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

LAPACK

Math



SciPy

Machine Learning



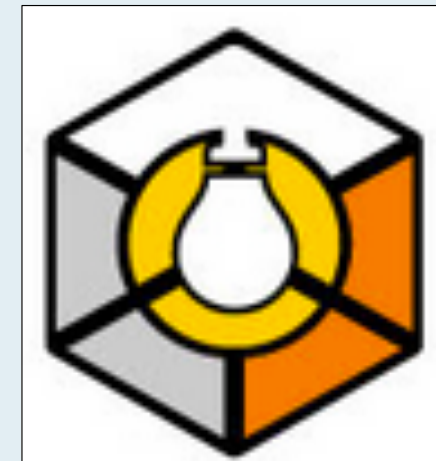
scikit-learn



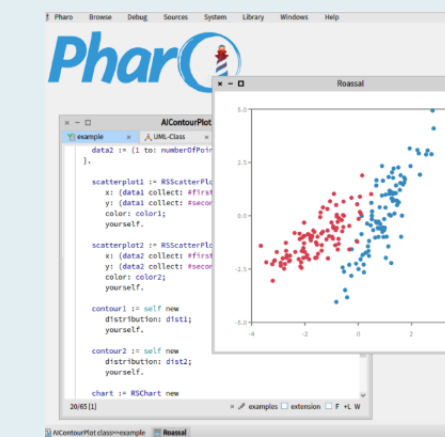
Pharo

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

LAPACK



PolyMath



pharo-ai

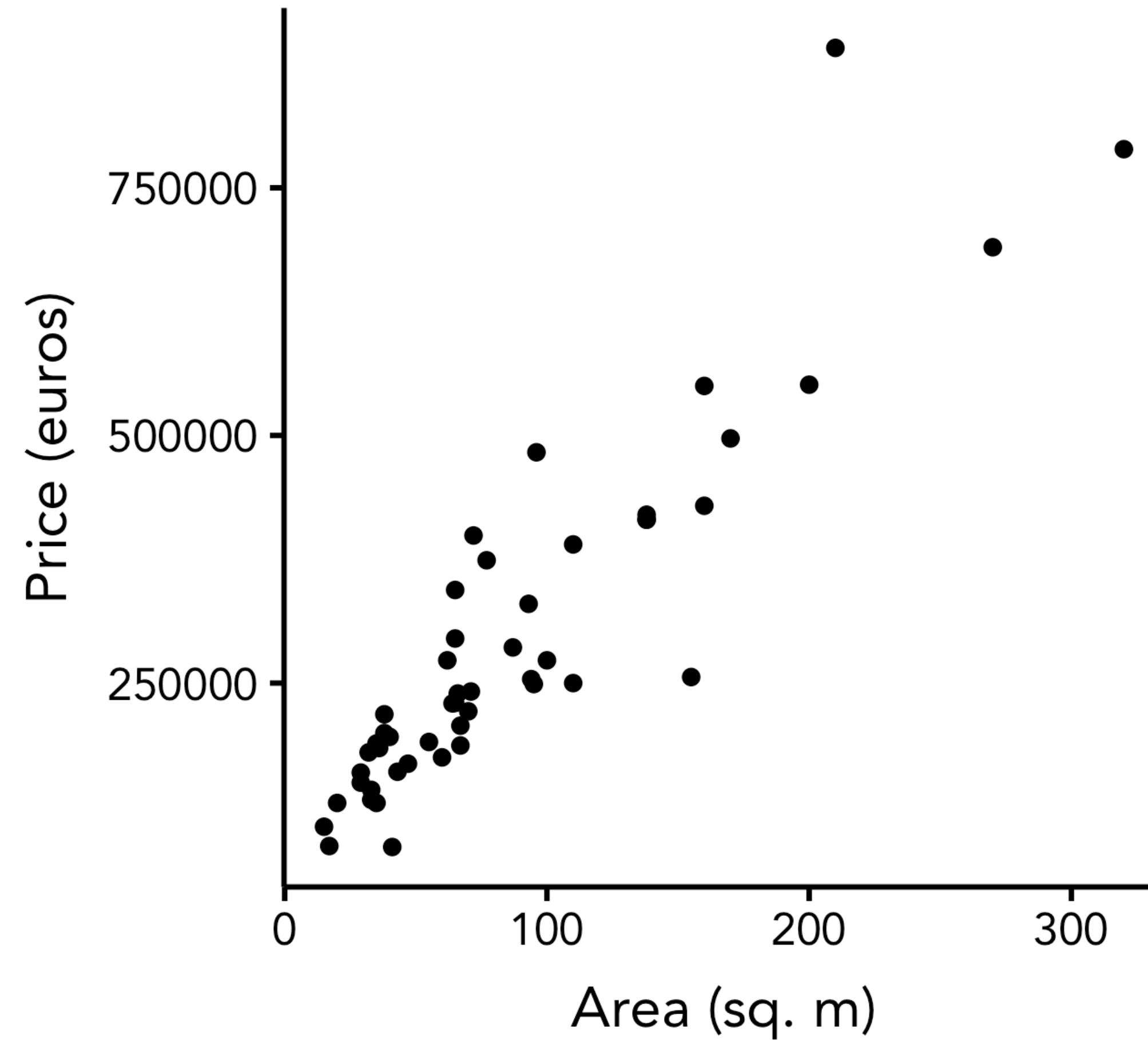
We want to provide a proof of concept that math & AI algorithms in Pharo can be boosted by delegating time-consuming operations to highly-optimised external libraries.

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

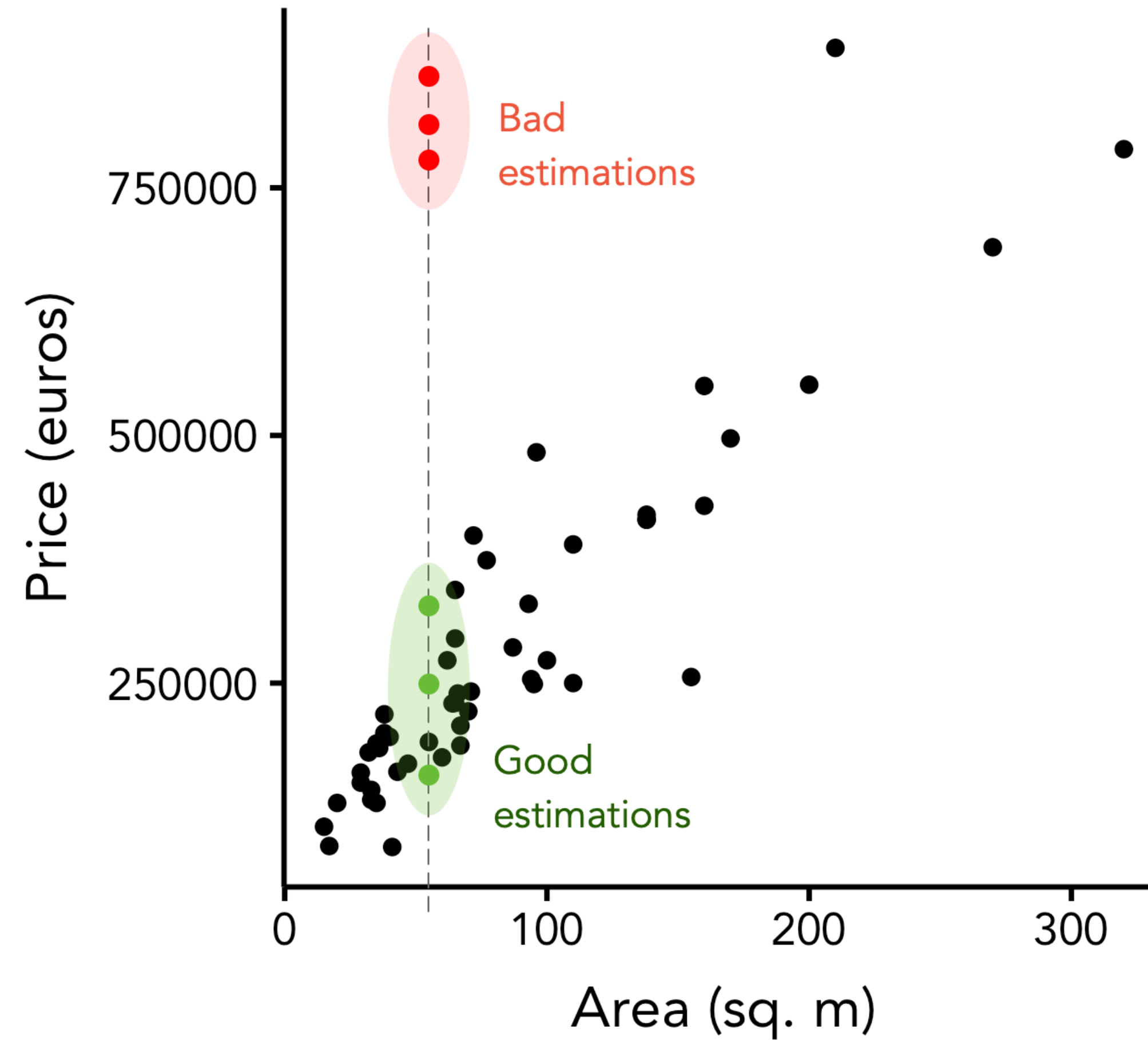


Linear regression and how it is implemented

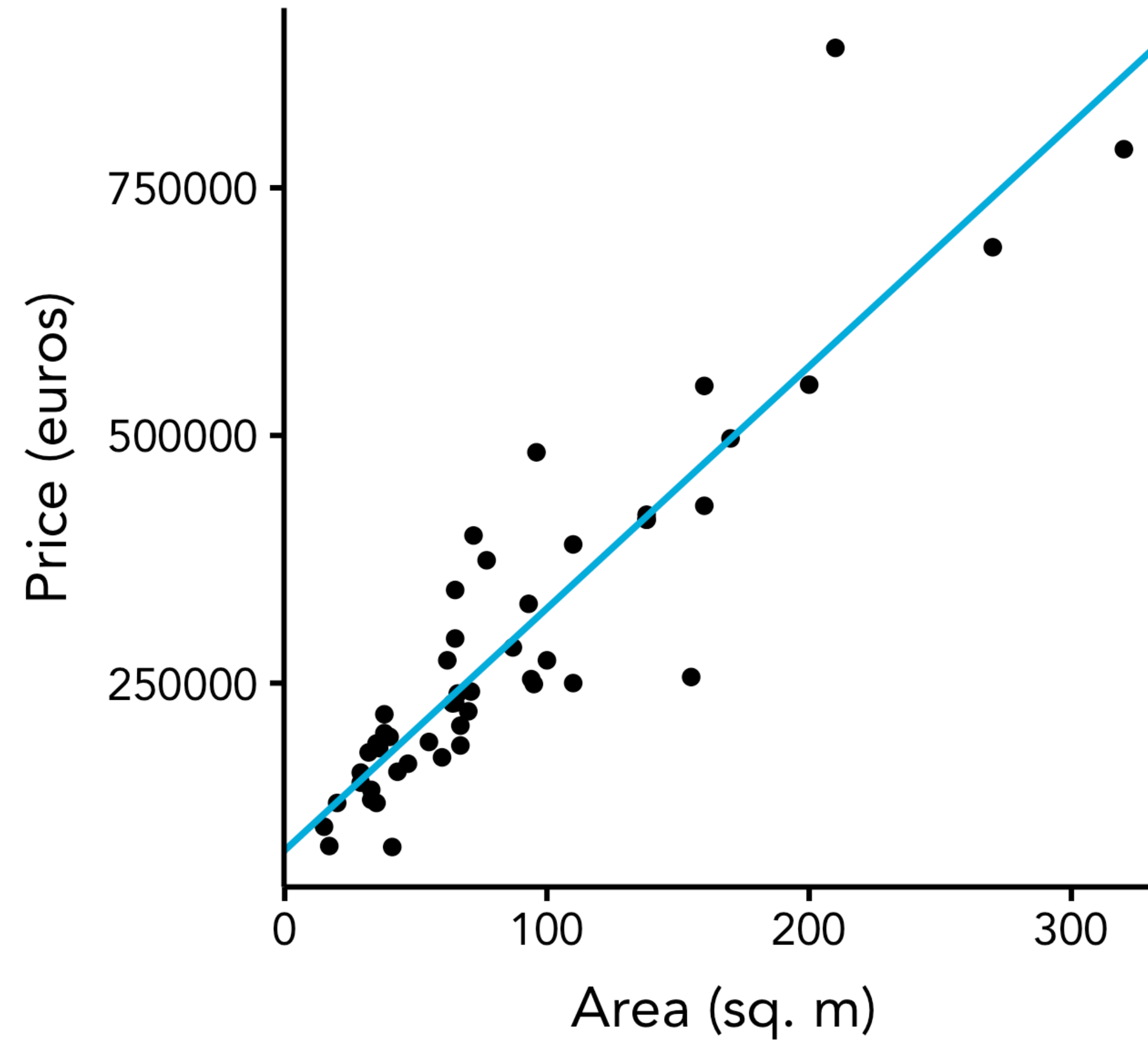
Linear Regression Problem



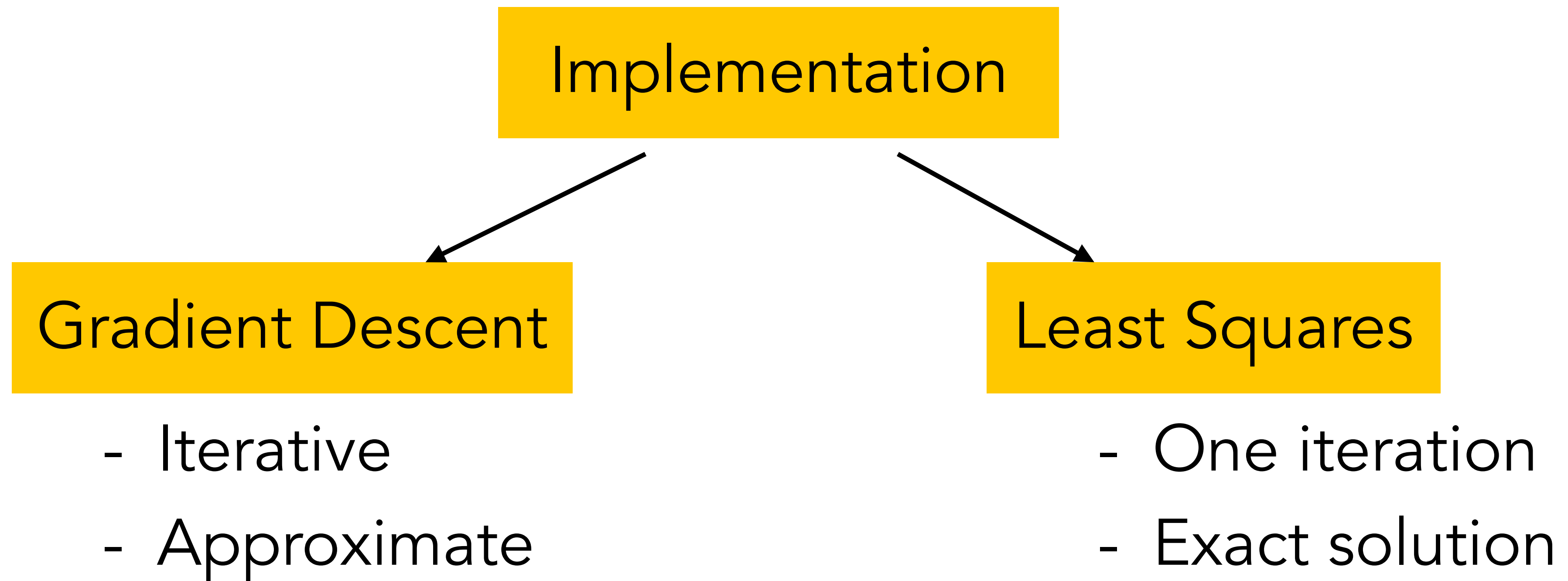
Linear Regression Problem



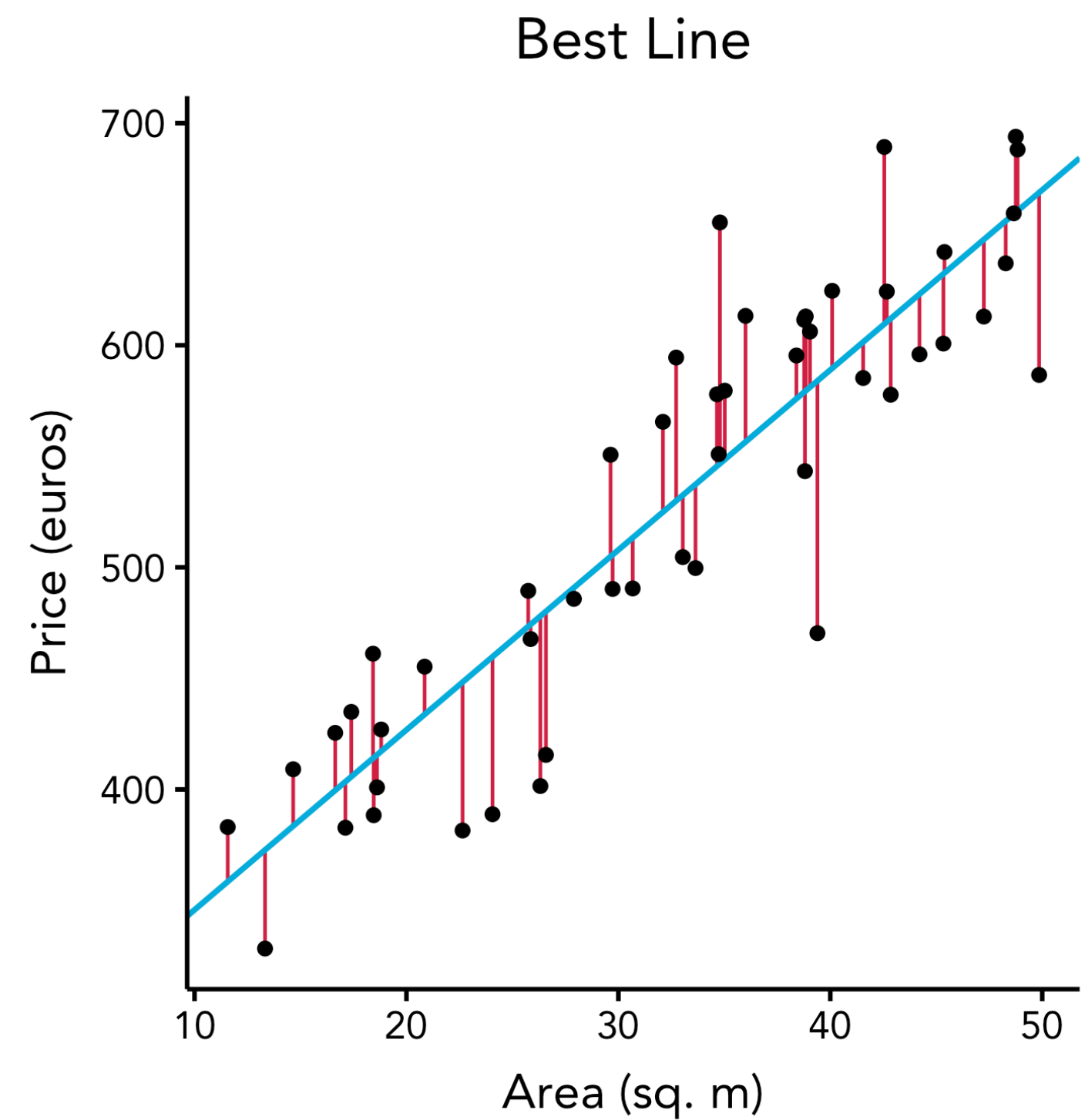
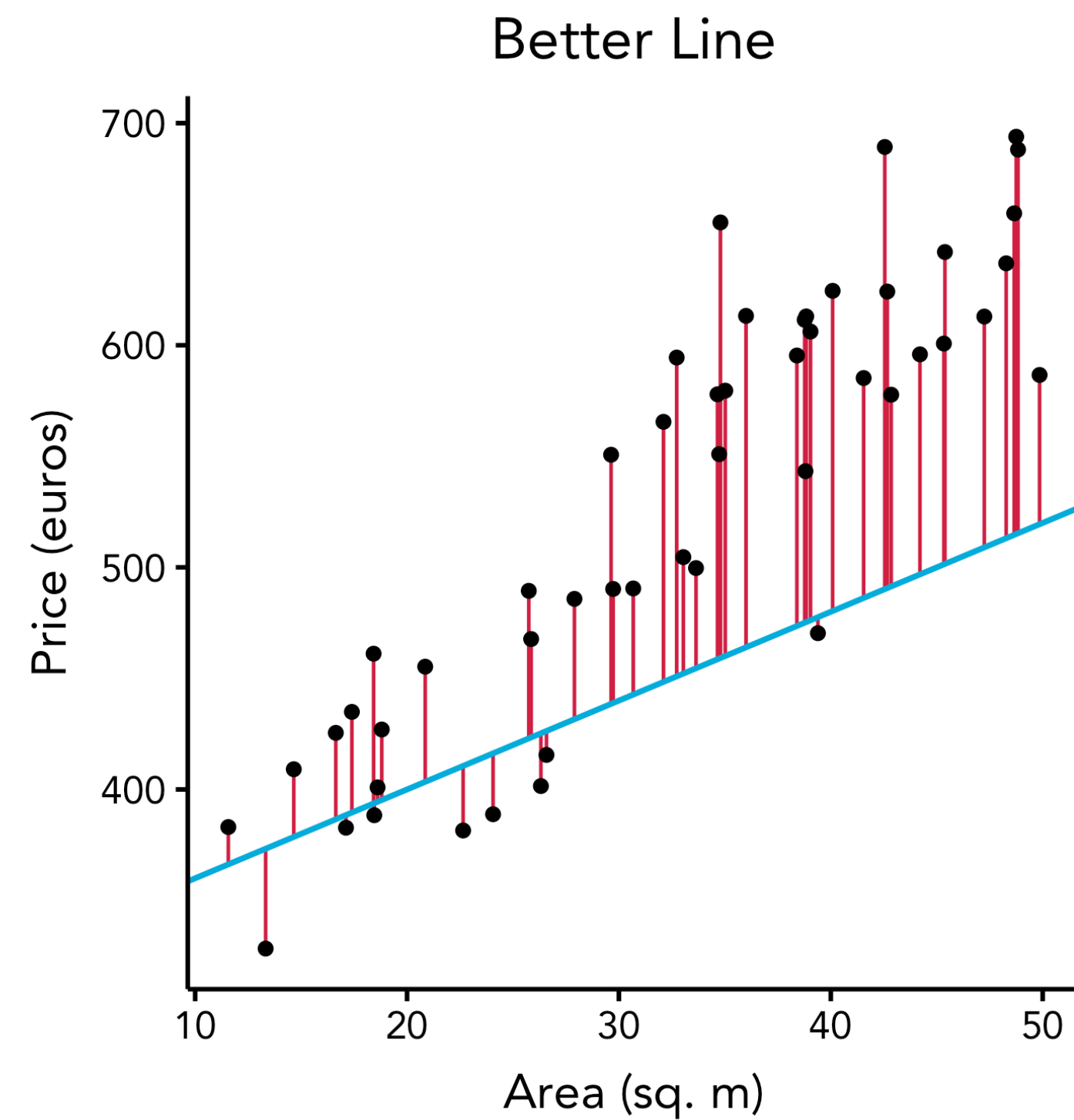
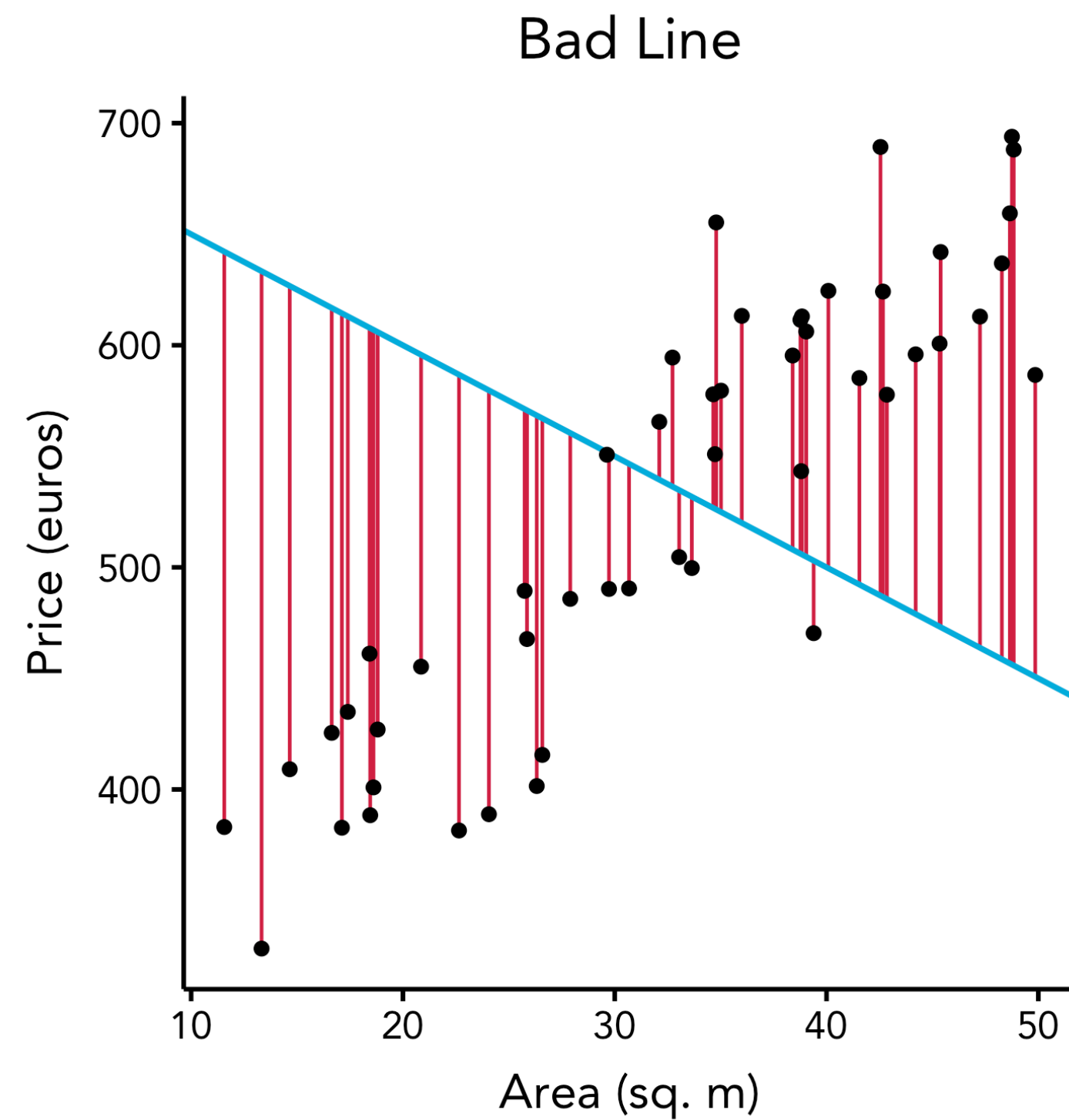
Linear Regression Problem



Linear Regression Implementations



Implementation 1. Gradient Descent



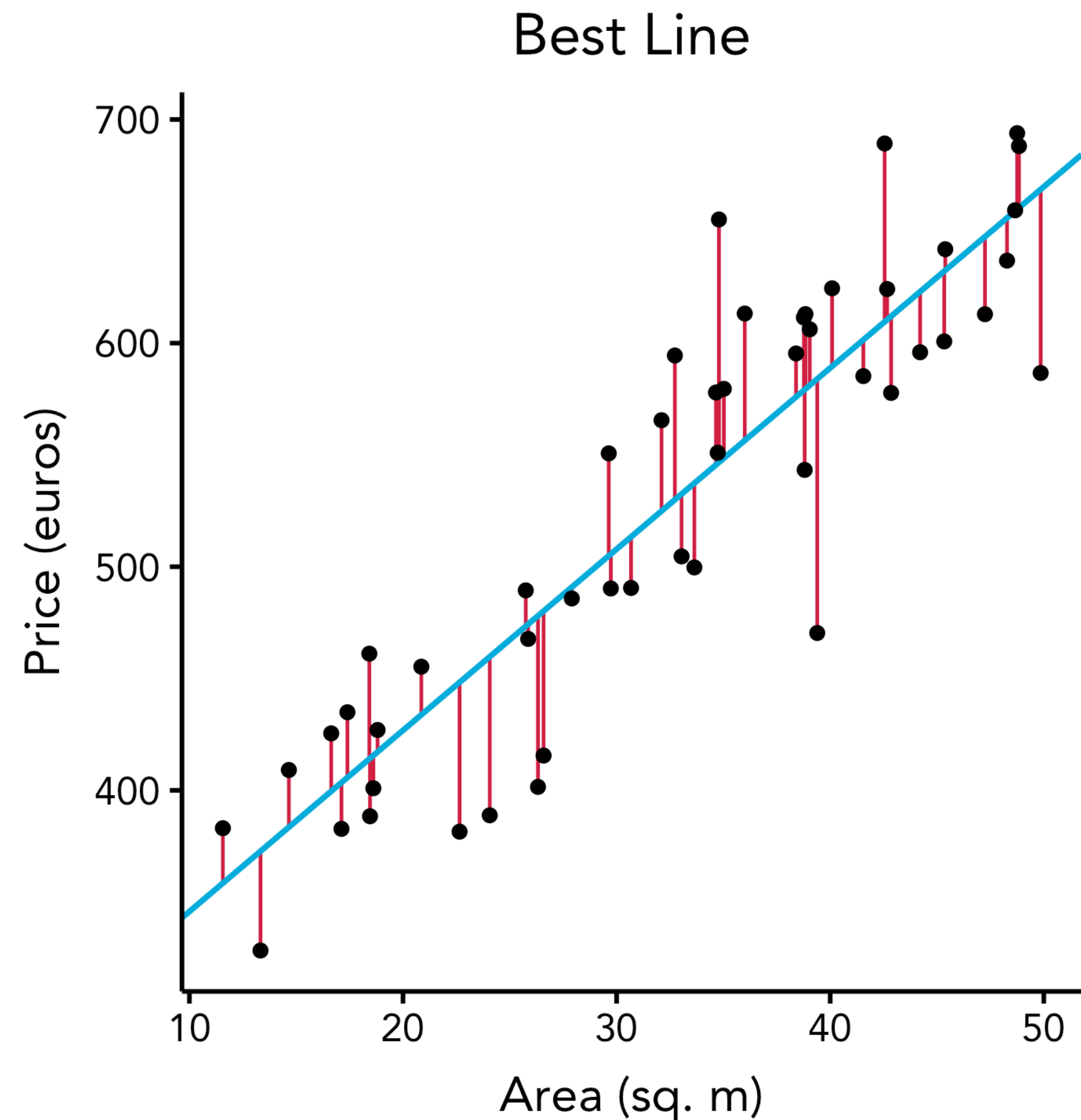
Implementation 1. Gradient Descent

Need to minimise the errors

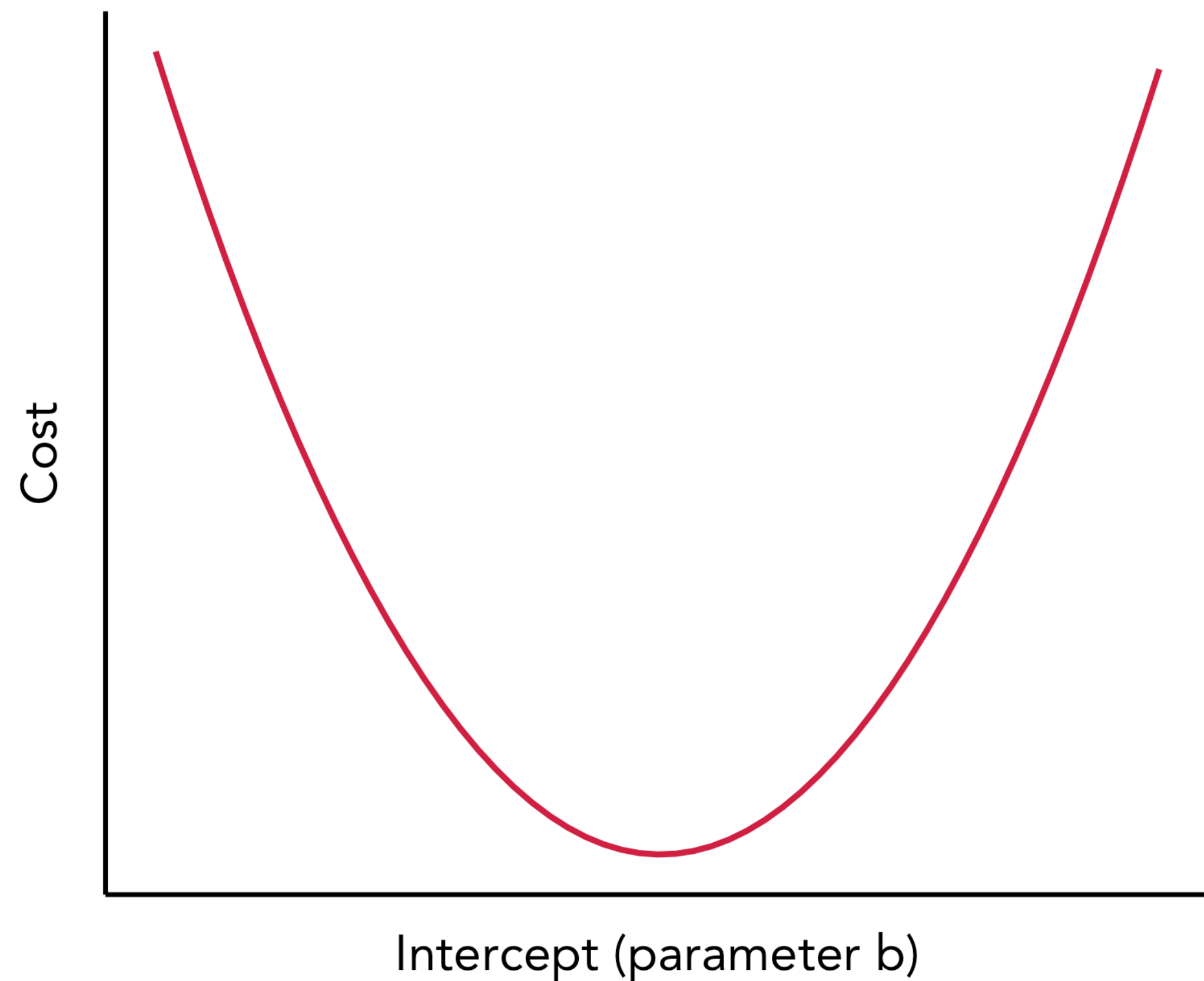
$$e_i = \hat{y}_i - y_i$$

Cost function
- mean squared errors

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

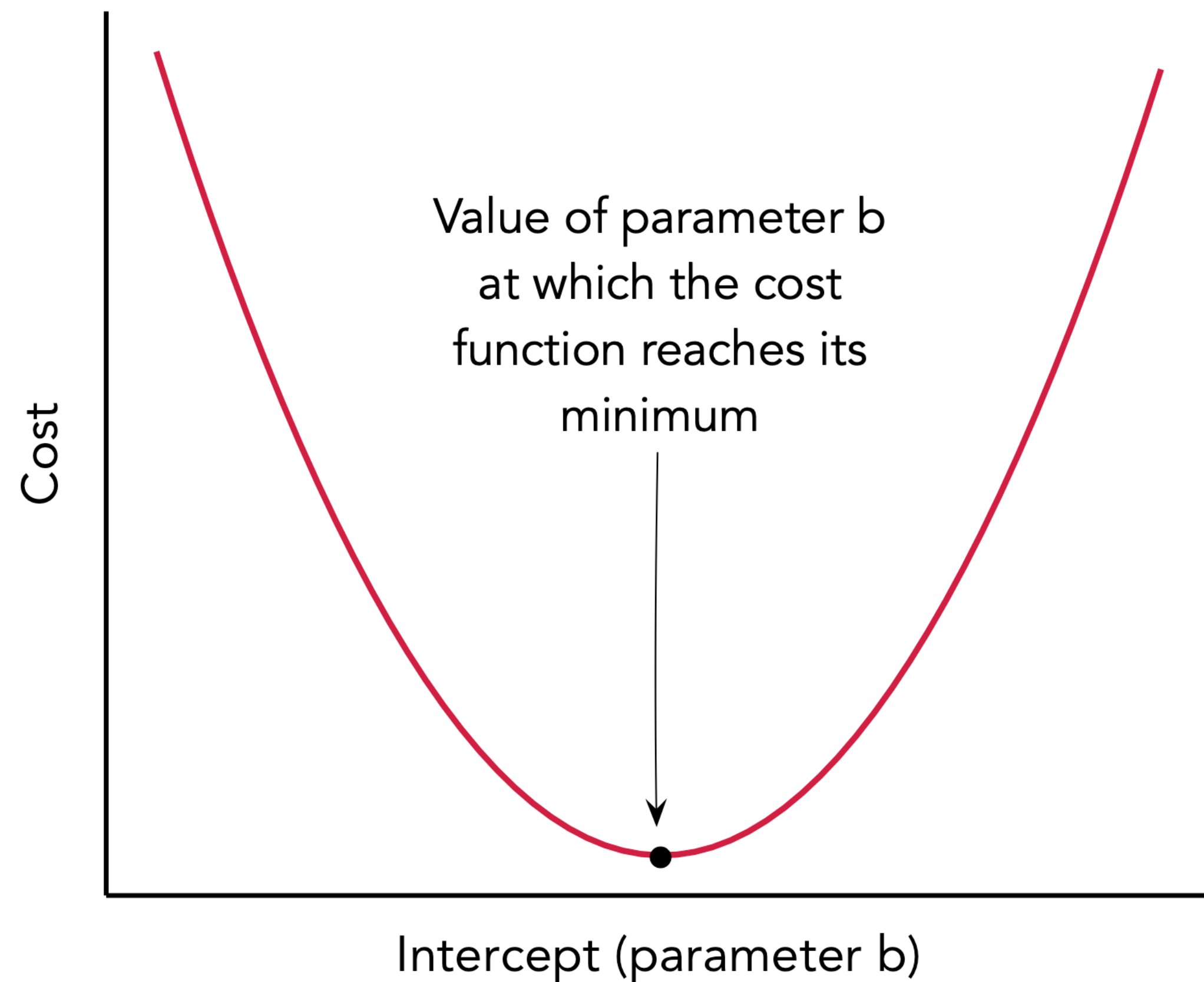


Implementation 1. Gradient Descent



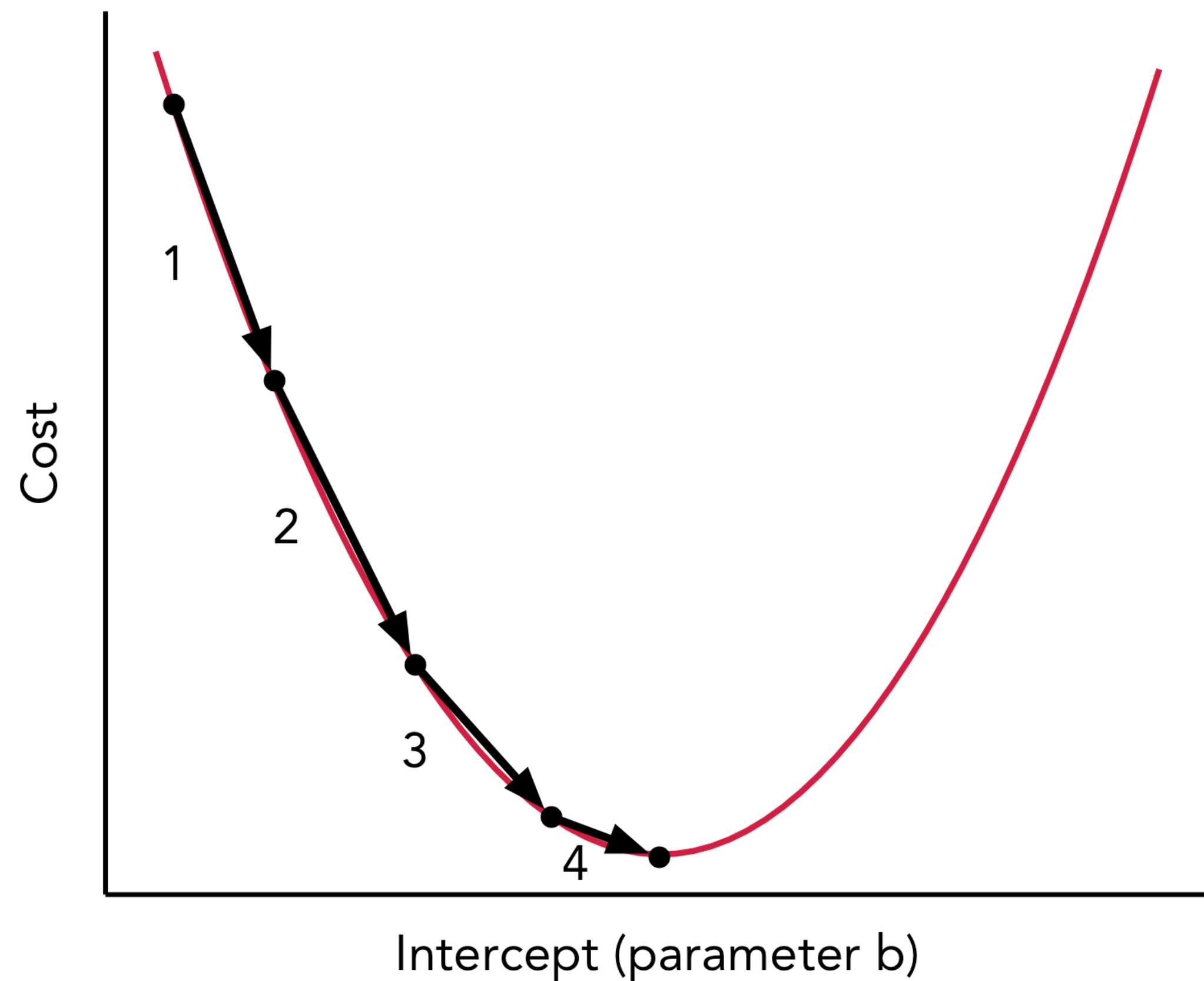
$$J(w) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2$$

Implementation 1. Gradient Descent



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

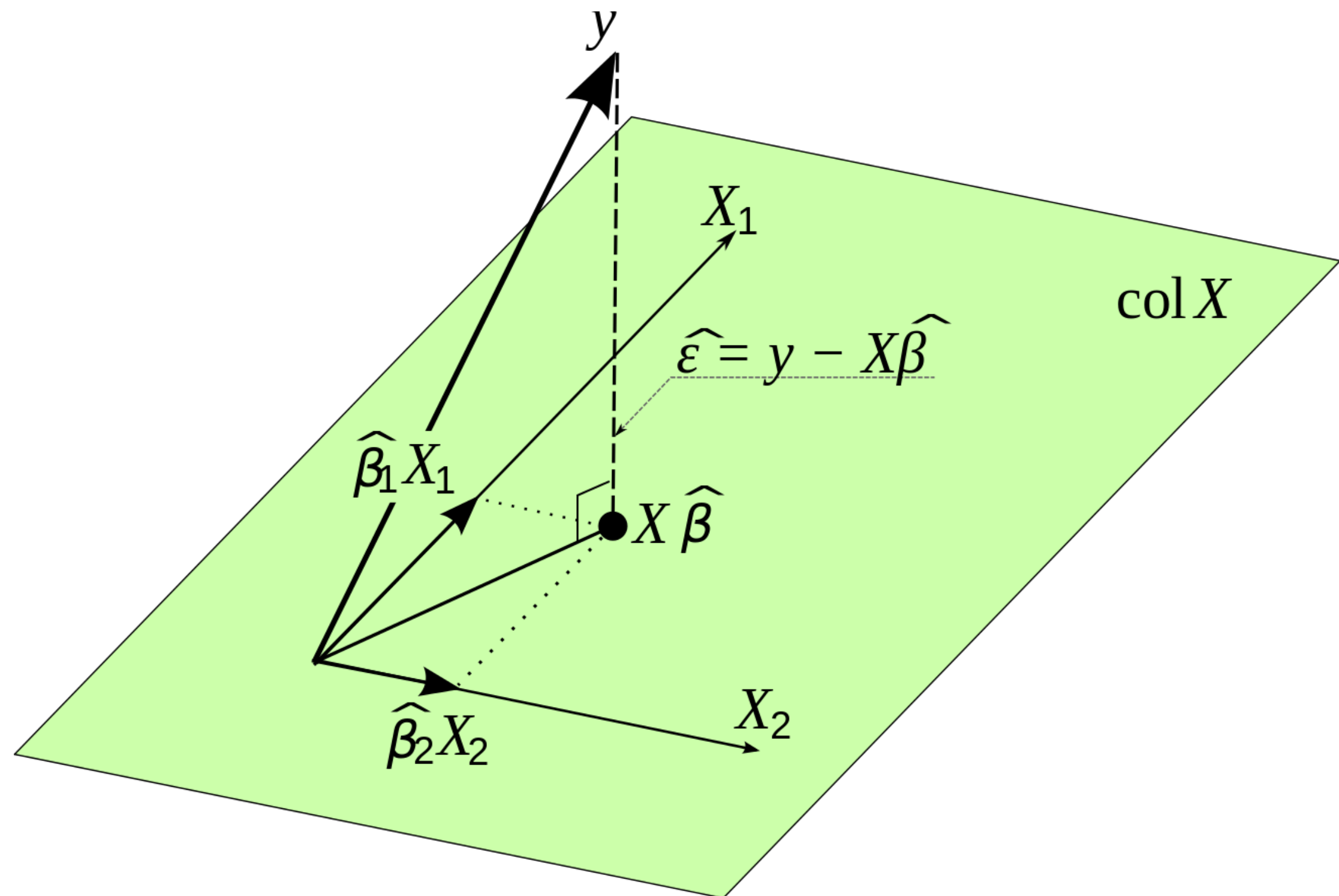
Implementation 1. Gradient Descent



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

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

Implementation 2. Least Squares



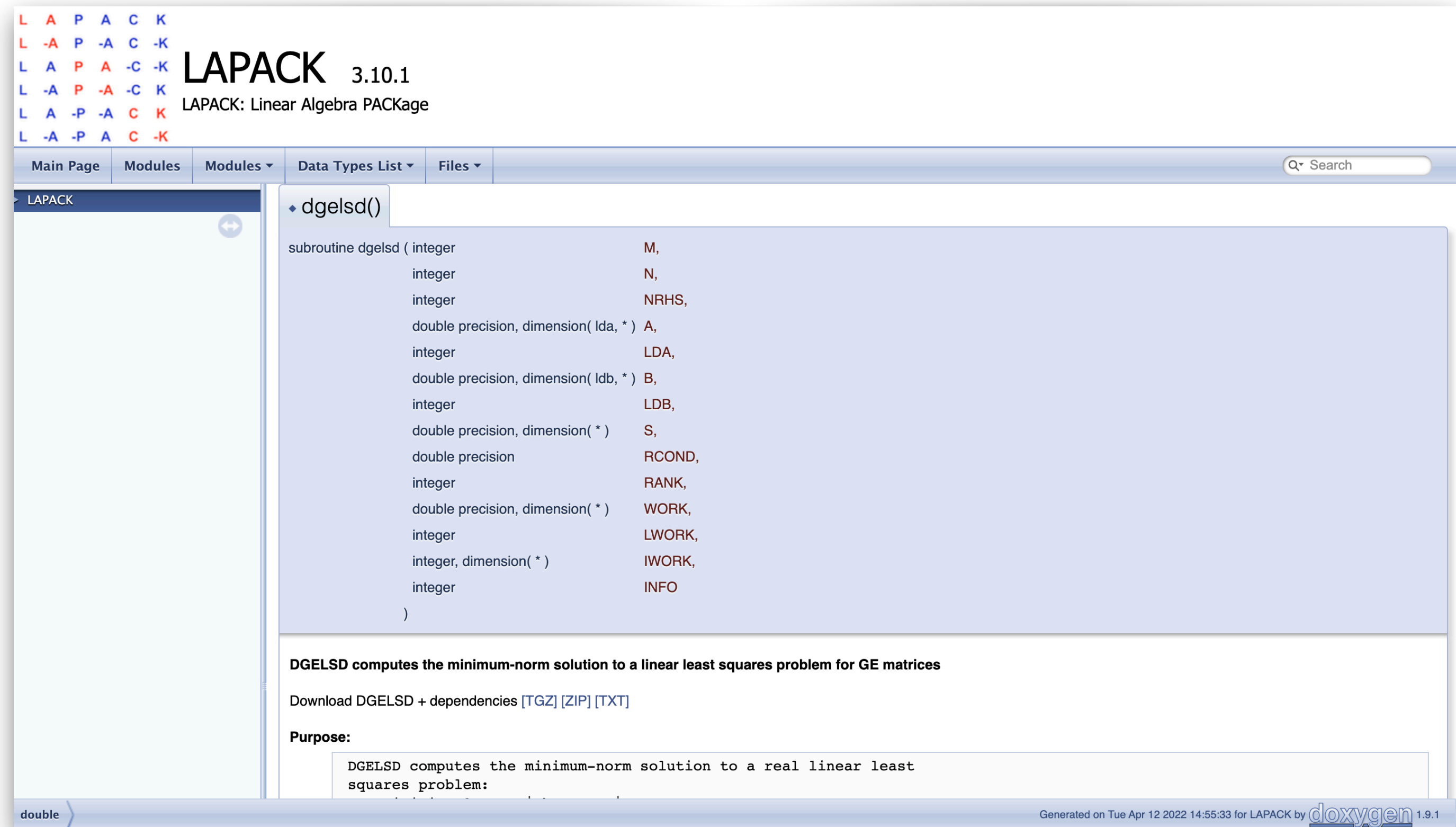
$$\hat{\theta} = \operatorname{argmin} \|y - X\theta\|_2$$

Orthogonal projection

Implementation 2. Least Squares

Implemented as
DGELSD routine
In LAPACK

We can call it
through FFI



The screenshot shows the LAPACK 3.10.1 documentation page for the `dgelsd()` routine. The page header includes the LAPACK logo (L A P A C K) and the version number 3.10.1. The main content area displays the routine signature and its purpose. The routine signature is: `subroutine dgelsd (integer M, integer N, integer NRHS, double precision, dimension(lda, *) A, integer LDA, double precision, dimension(ldb, *) B, integer LDB, double precision, dimension(*) S, double precision RCOND, integer RANK, double precision, dimension(*) WORK, integer LWORK, integer, dimension(*) IWORK, integer INFO)`. Below the signature, the text states: "DGELSD computes the minimum-norm solution to a linear least squares problem for GE matrices". There are links for "Download DGELSD + dependencies [TGZ] [ZIP] [TXT]". The "Purpose:" section contains a text box with the text: "DGELSD computes the minimum-norm solution to a real linear least squares problem:". The footer of the page indicates it was generated on Tue Apr 12 2022 14:55:33 for LAPACK by doxygen 1.9.1.

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

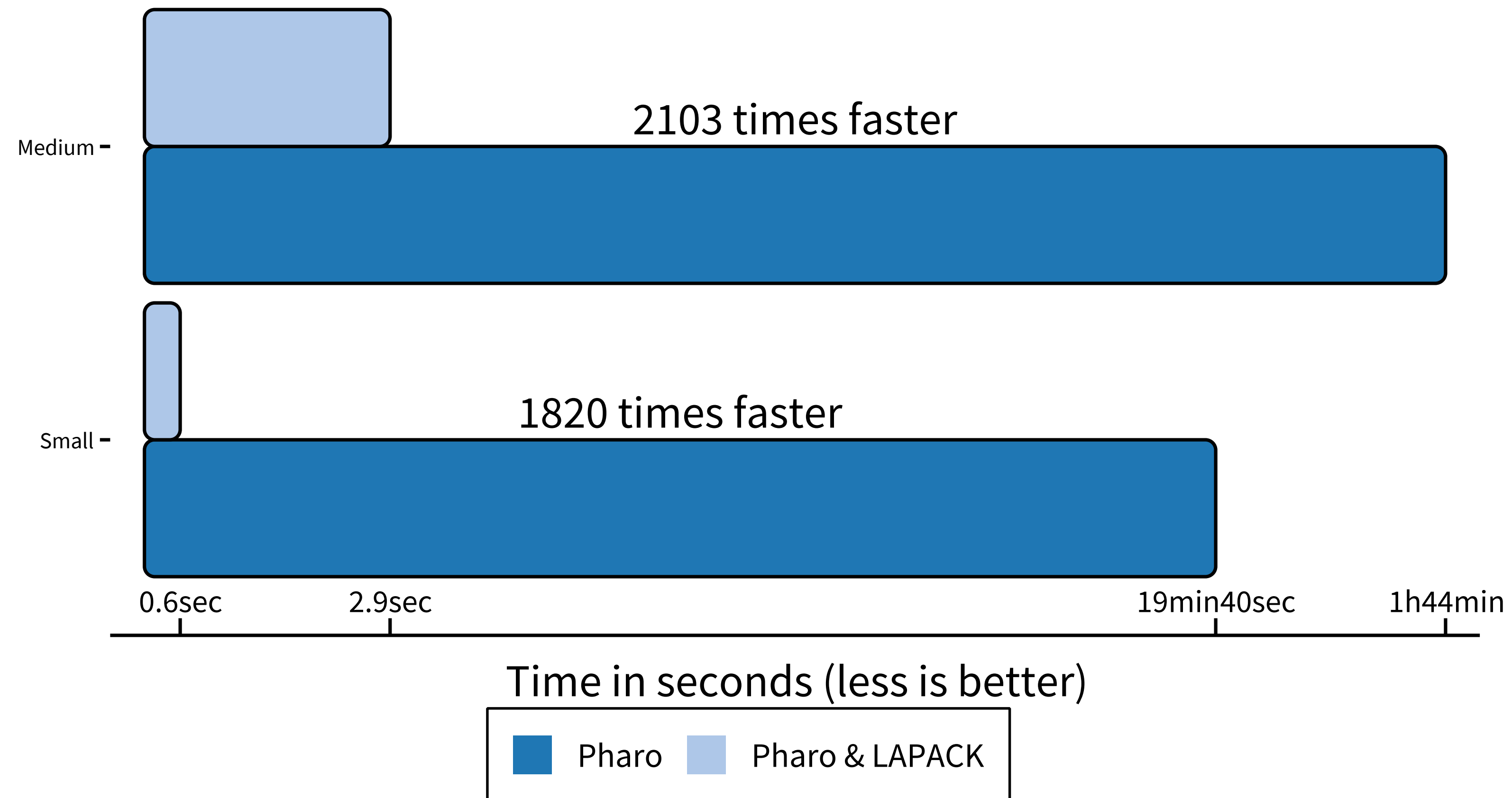
The speedup achieved by calling LAPACK routines from Pharo.

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

Pure Pharo vs Pharo & LAPACK



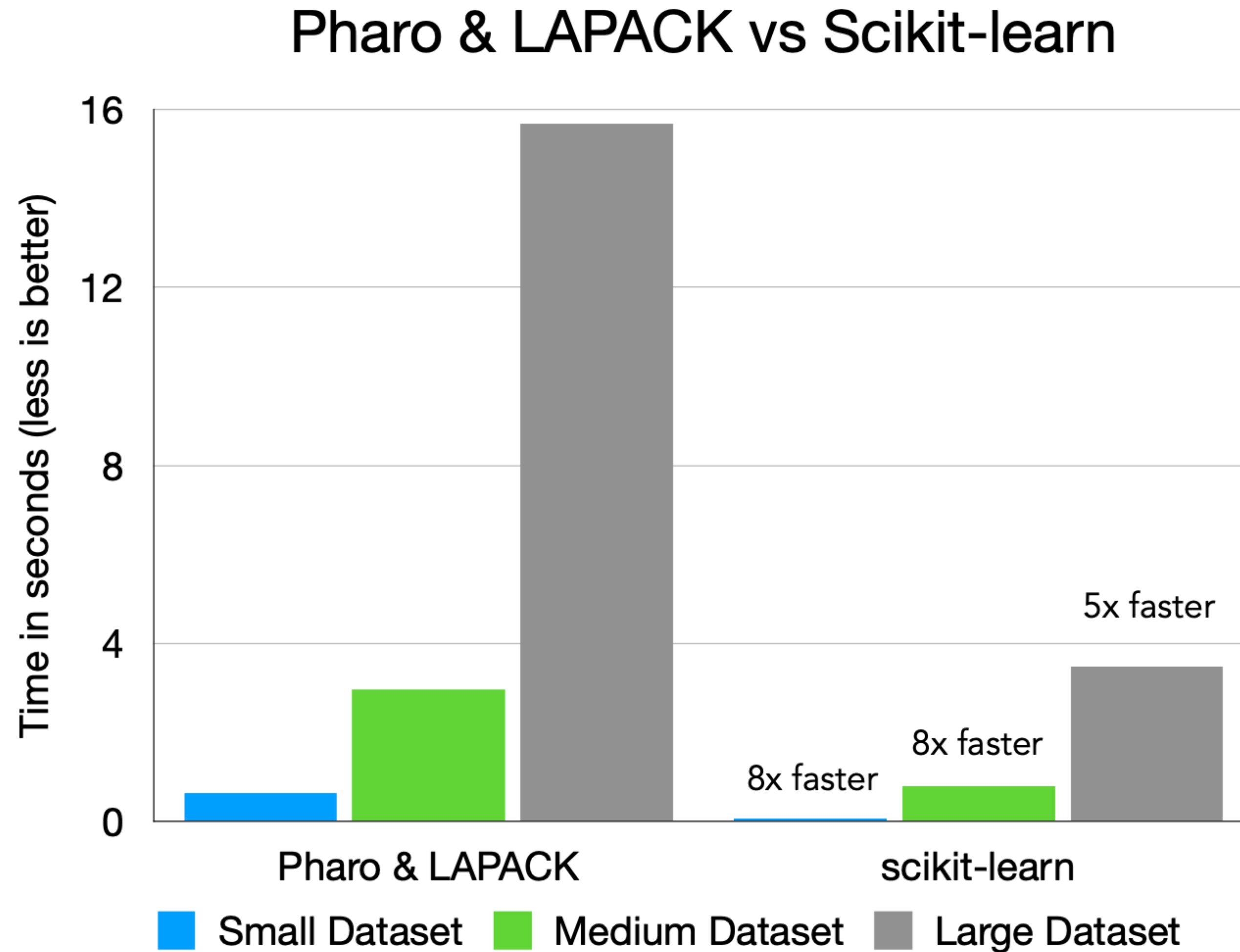
RQ.2 - Comparing to scikit-learn.

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.



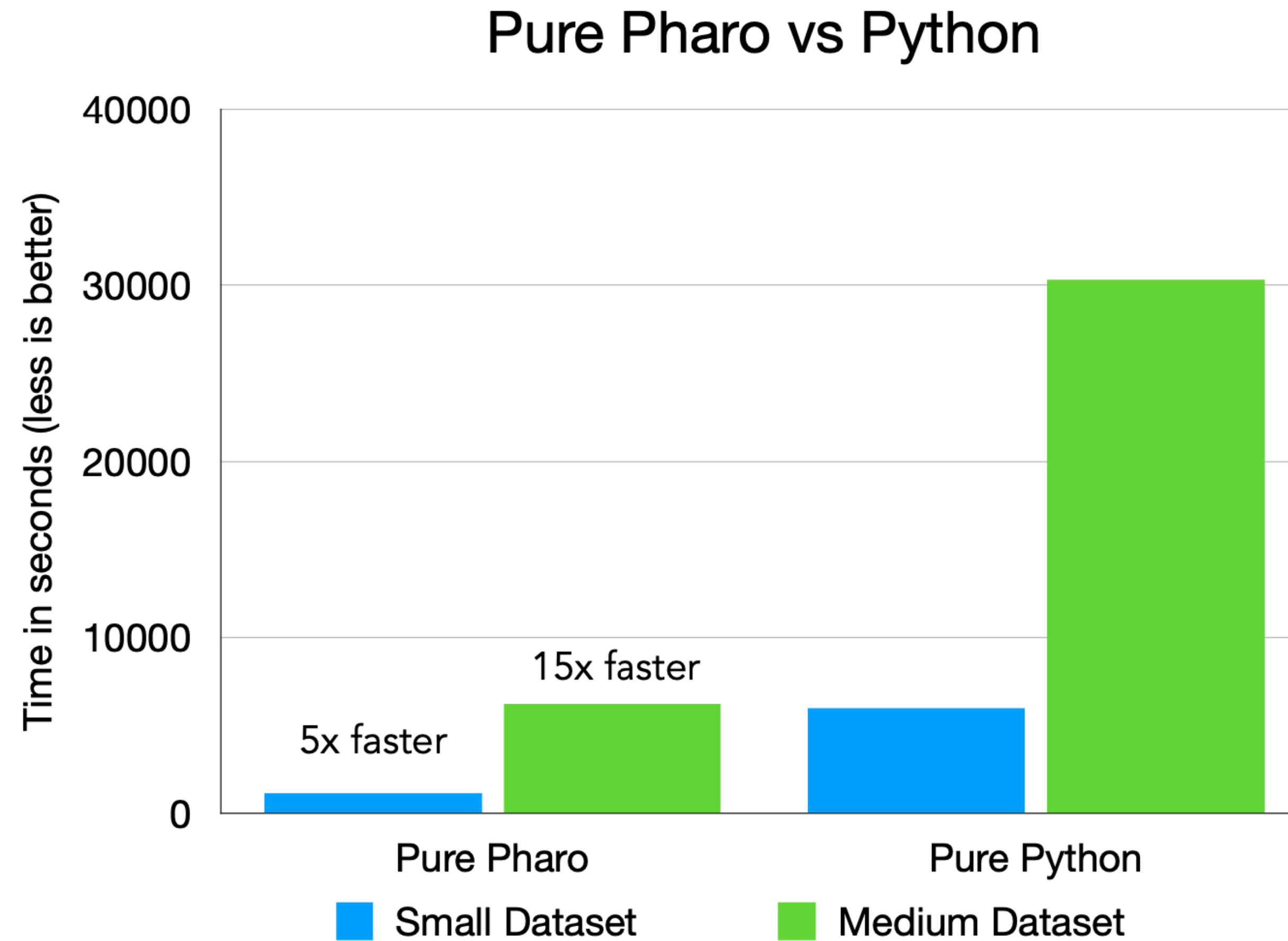
RQ.3 - Comparing pure Pharo with Python

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.