Linear Regression

By Dr. Nzamba Bignoumba

$$y = -ax + b$$

Average training duration: 4 hours 00 minute

Outline

Machine learning overview	→	15 min
Linear regression: theory	→	45 min
Linear regression: use case	→	01.30 h
Model deployment	→	01.30 h

Machine learning overview

AI

what can we do with it?

Natural Language Processing
Computer Vision
Signal Processing
⋮

Machine learning overview

Content summary of one or more documents
Translation from one language to another
Code generation
⋮

OpenAI-ChatGPT
DeepSeek
GitHub Copilot
Cursor | Codex

Visual content generation
Medical image classification
Agricultural Image Classification
Object detection
⋮

Midjourney
Canva
Aidoc
IA Agri
AgriHyphen AI

Natural Language Processing
Computer Vision
Signal Processing
⋮

Weather forecast
Disease and mortality forecasts/predictions
Stock market forecasts
Electricity consumption forecasts
Anomaly detection (cybersecurity)
⋮

AWS SageMaker – DeepAR
Nixtla-TimeGPT
Meta-Prophet
Zindi Africa
Amini

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,L. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
N. Bignoumba, N. Mellouli, S. B. Yahia, A new efficient alignment-driven neural network for mortality prediction from irregular multivariate time series data, Expert Systems with Applications 238 (2024) 122148.

Machine learning overview

AI

what can we do with it?

Natural Language Processing Computer Vision Signal Processing
⋮

With what type of algorithms?

Machine Learning

Deep Learning

Machine learning overview

K-NN

Logistic Regression

K-Means

Support Vector Machines

Decision Trees

Gradient Boosting Machines

Random Forest

Linear Regression

Principal Component Analysis

Machine Learning

...

Deep Learning

Generative Adversarial Neural Network

Varational Autoencoder

Feed Forward Neural Network

State Space Model

Word Embeddings

Autoencoder

Graph Neural Network

Neural Ordinary Differential Equations

Diffusion Model

Recurrent Neural Network

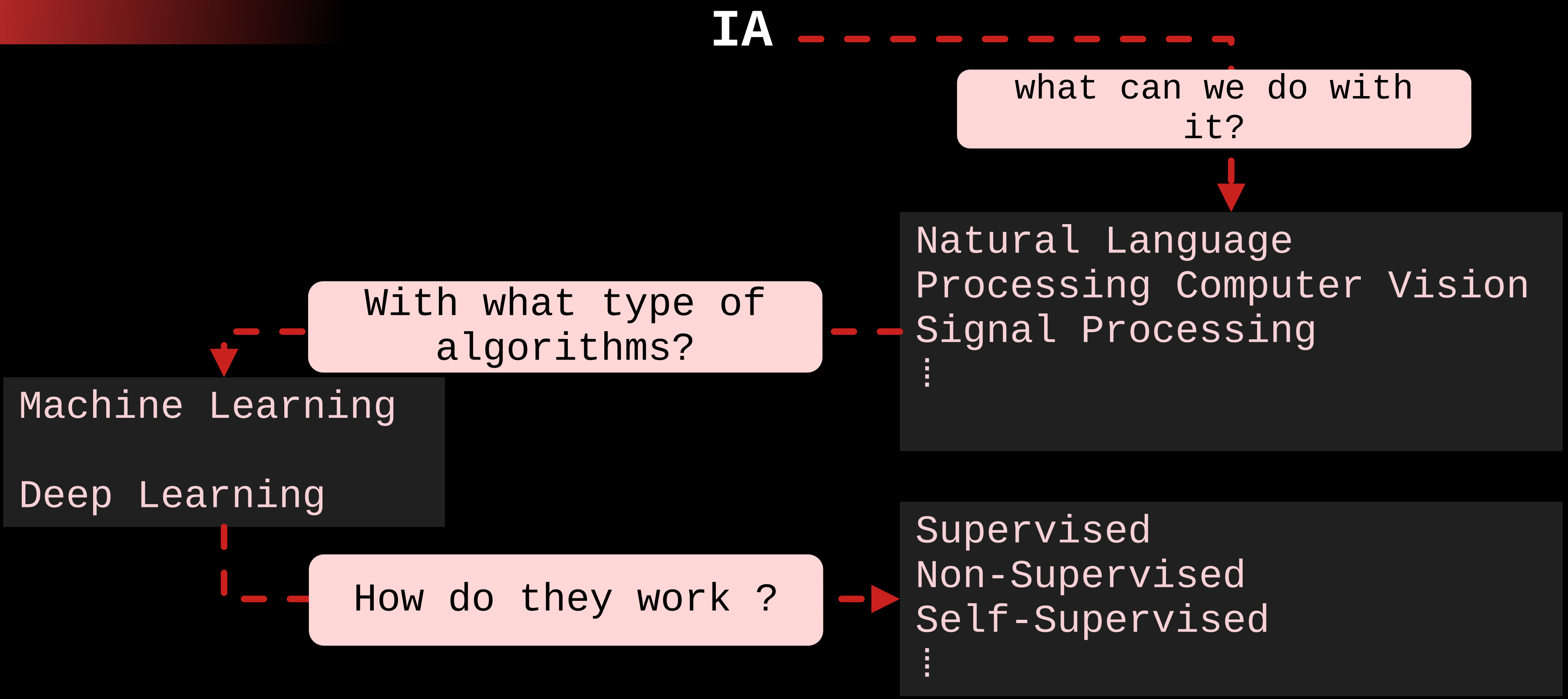
Neural Radiance Field

Transformer

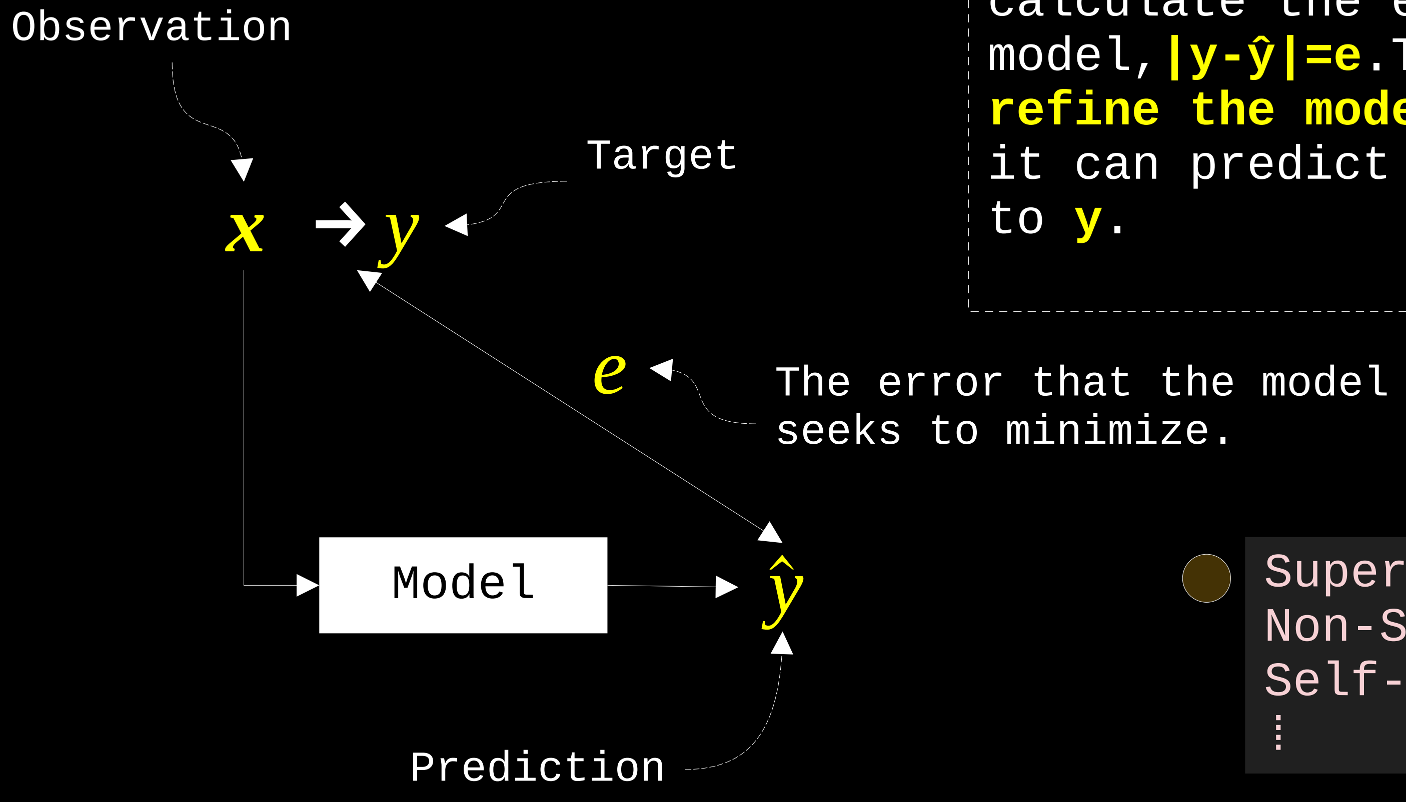
Normalizing Flows

...

Machine learning overview



Machine learning overview

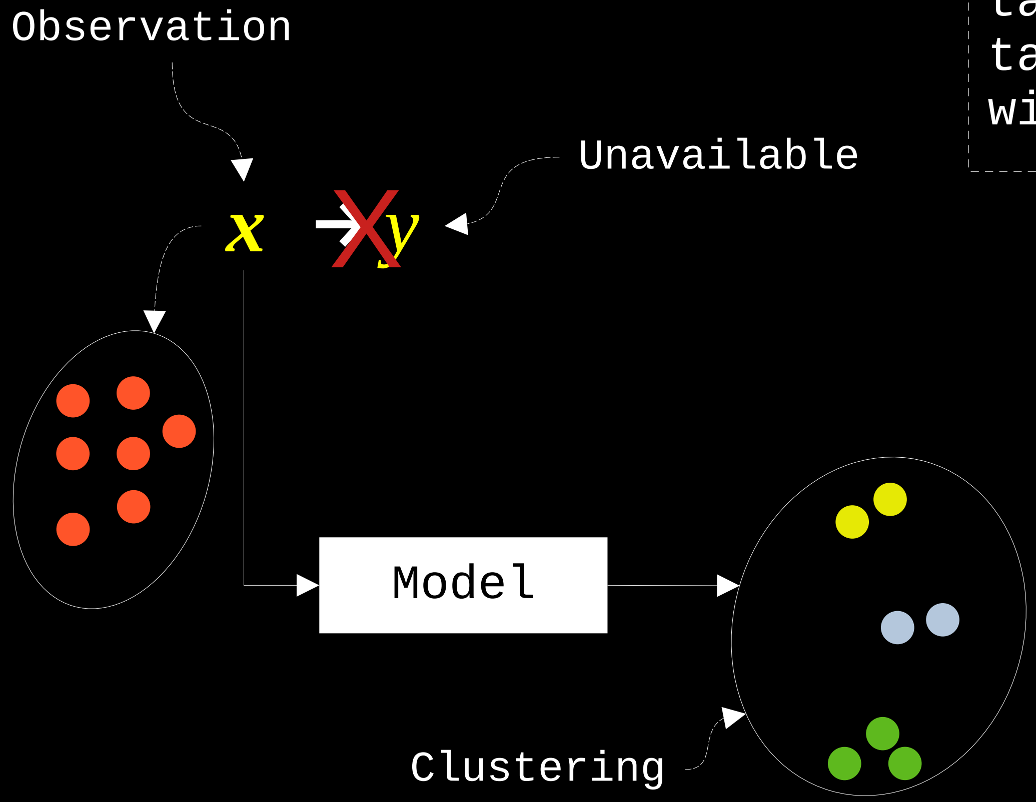


Let’s have the dataset **Xs** and its corresponding target **ys**. The model will use **X** to predict **ŷ**, **modèle(X)=ŷ**. Then, **ŷ** will be compared to **y** to calculate the error **e** made by the model, **|y-ŷ|=e**. This error serves to **refine the model parameters** so that it can predict a value **ŷ** very close to **y**.

Supervised
Non-Supervised
Self-Supervised
⋮

Machine learning overview

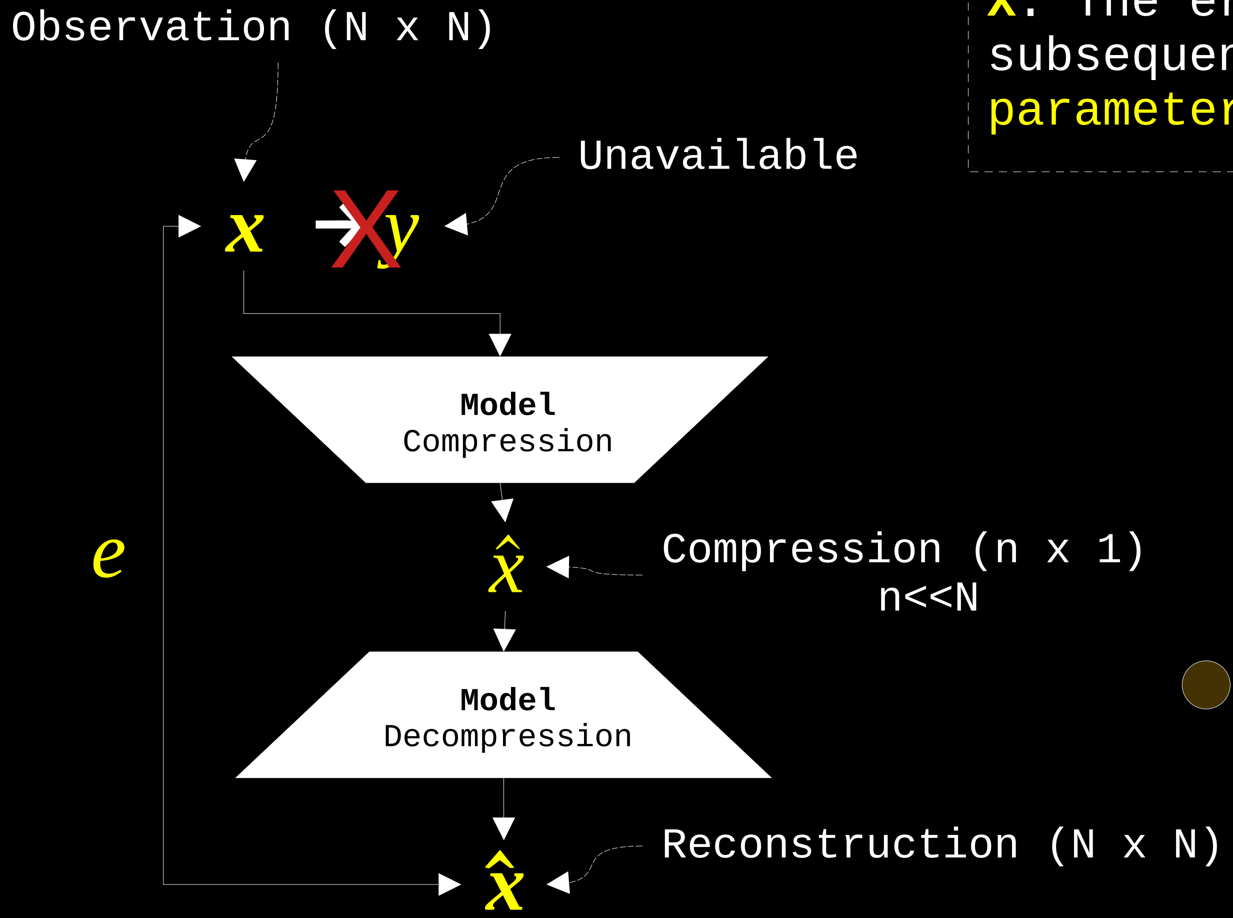
We only have **Xs** data. No matching **y** targets are available. The model will leverage data **similarities** and **co-occurrences** to perform the assigned task. For example, the clustering task, which consists of grouping data with similar patterns.



- Supervised
- Non-Supervised
- Self-Supervised
- ⋮

Machine learning overview

We only have the training dataset **Xs**. The model takes an observation **X** as input, **model(X)=X̂**, and compares its prediction **X̂** to this same observation **X**. The error **|X-X̂|=e**, will subsequently be used to fit the model parameters.



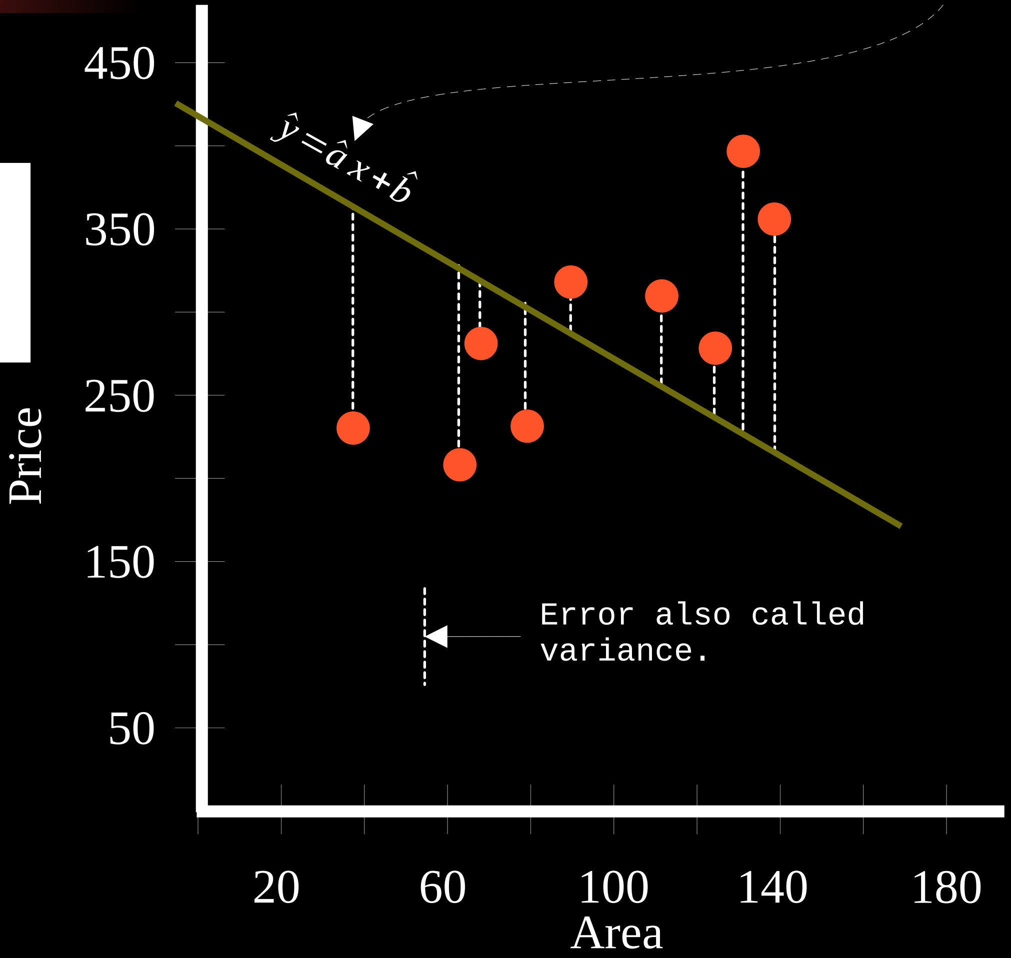
- Supervised
- Non-Supervised
- Self-Supervised
- ⋮

Theory

Model→ Linear Regression= $\hat{y}=\hat{a}x+\hat{b}$

$\hat{a}<0, \hat{b}\simeq 425$

Price chart (in XAF) of house locations based on their surface area (in m²).



The training consists of finding the values **â** and **b̂** that minimize the error (**E**) as much as possible.

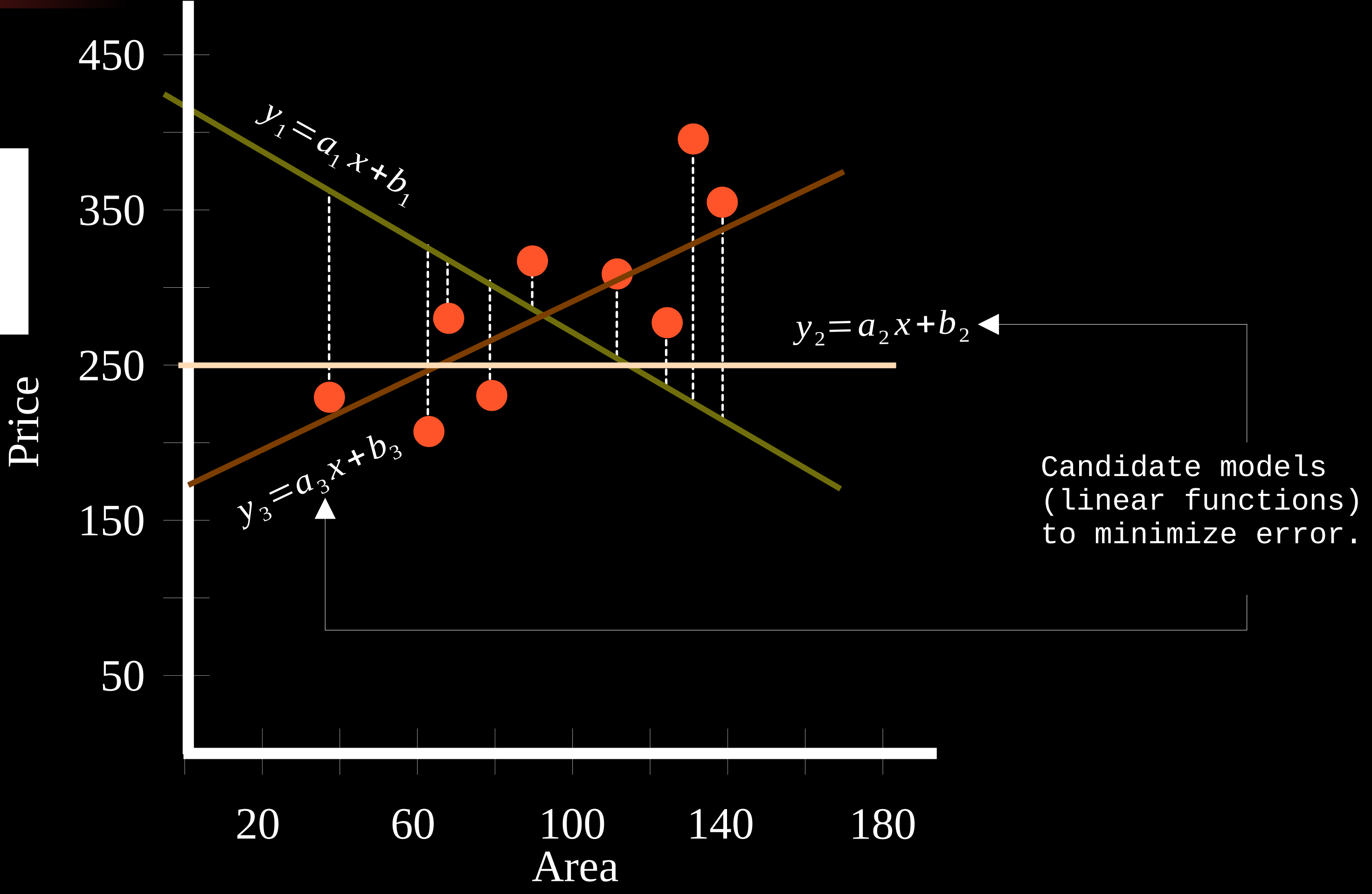
$$E = \; + \; + \; + \cdots$$

$$E=\left(e_1+e_2+\ldots+e_9\right)/I$$

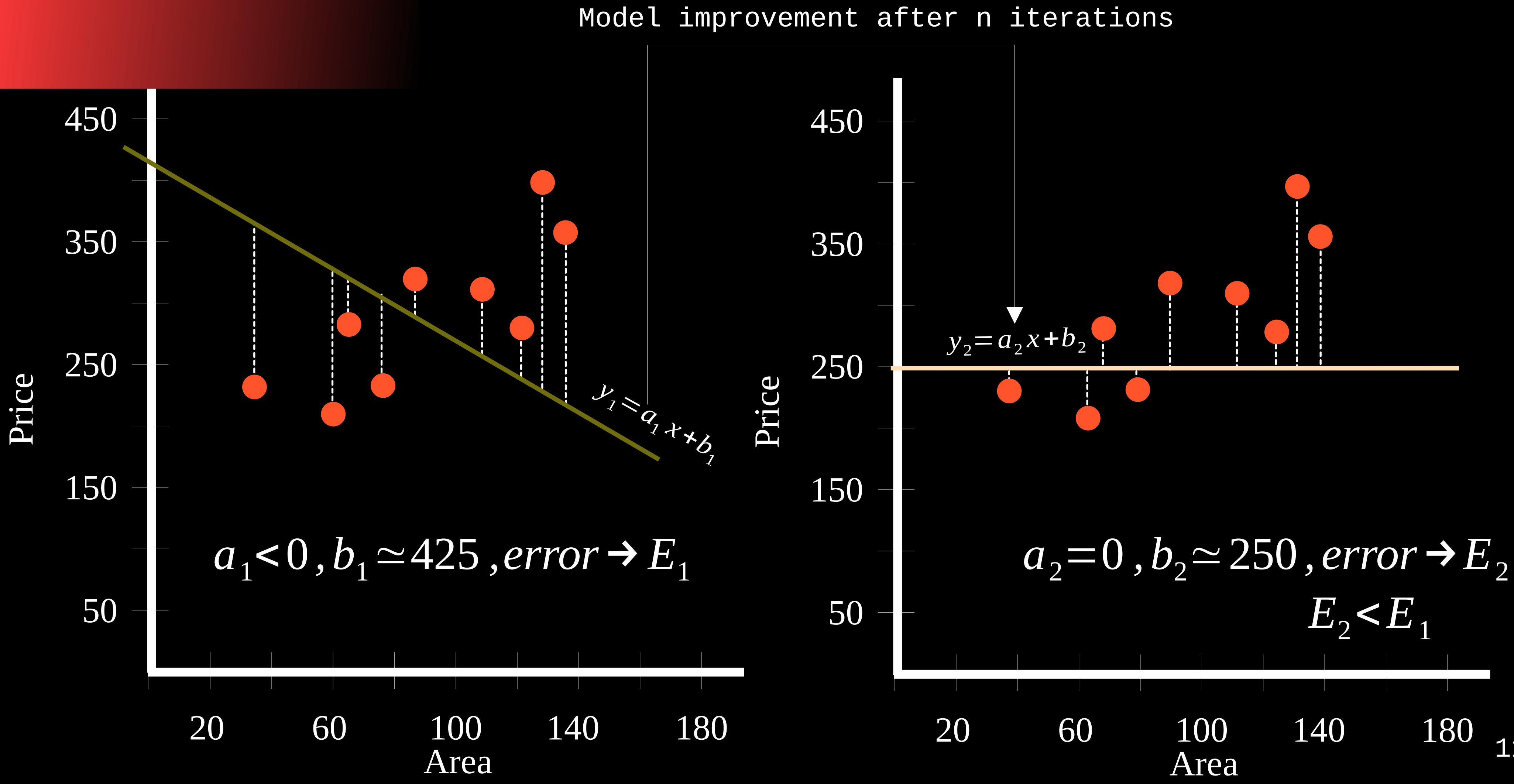
I=9, number of data points.

Theory

Price chart (in XAF) of house locations based on their surface area (in m²).

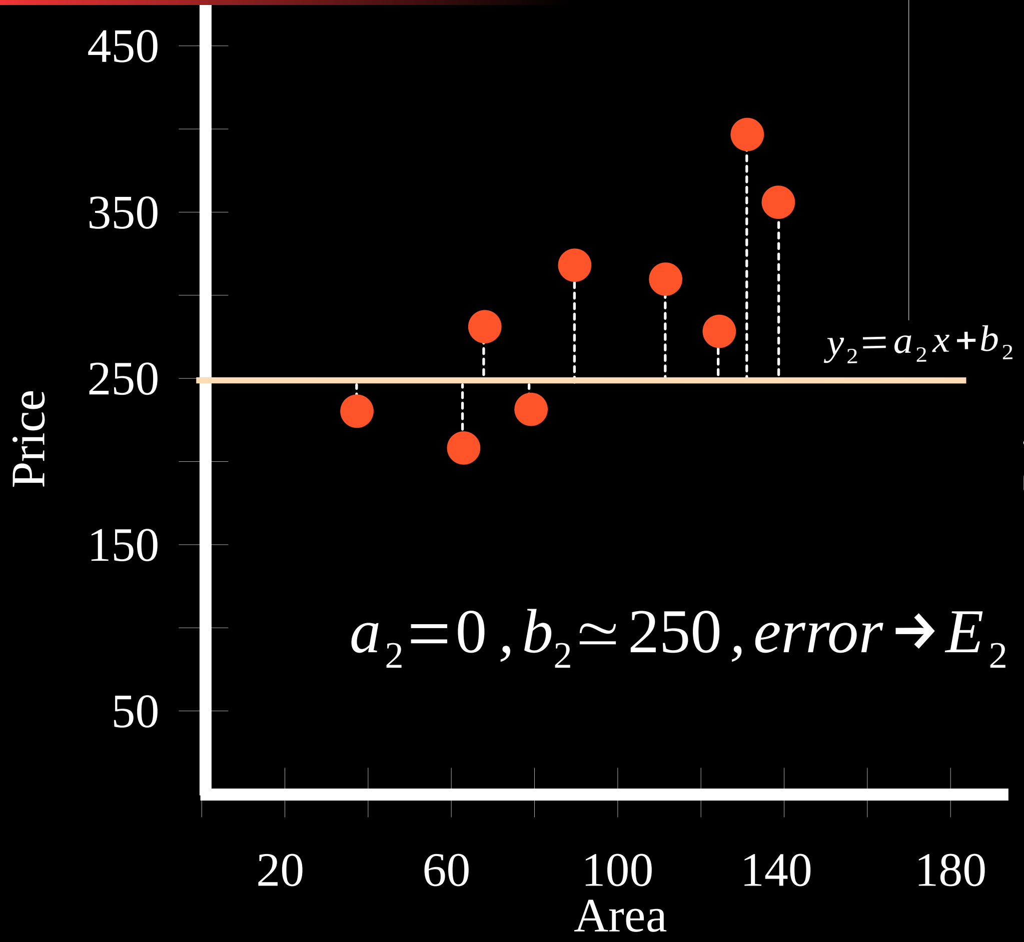


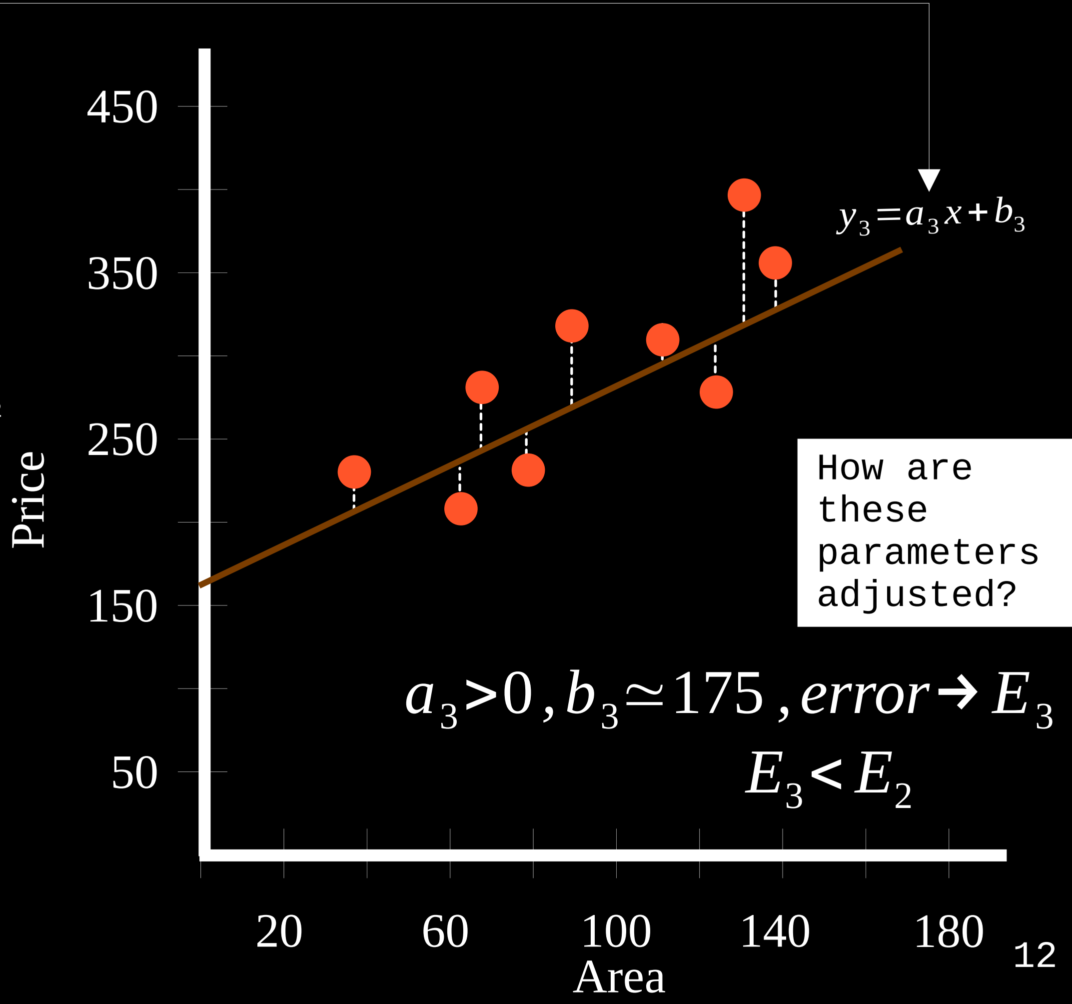
Theory


Model improvement after n iterations
450
350
250
150
50
Price
20
60
100
140
180
Area
$y_1 = a_1 x + b_1$
$a_1 < 0, b_1 \simeq 425, error \rightarrow E_1$
450
350
250
150
50
Price
20
60
100
140
180
Area
$y_2 = a_2 x + b_2$
$a_2 = 0, b_2 \simeq 250, error \rightarrow E_2$
$E_2 < E_1$

Theory

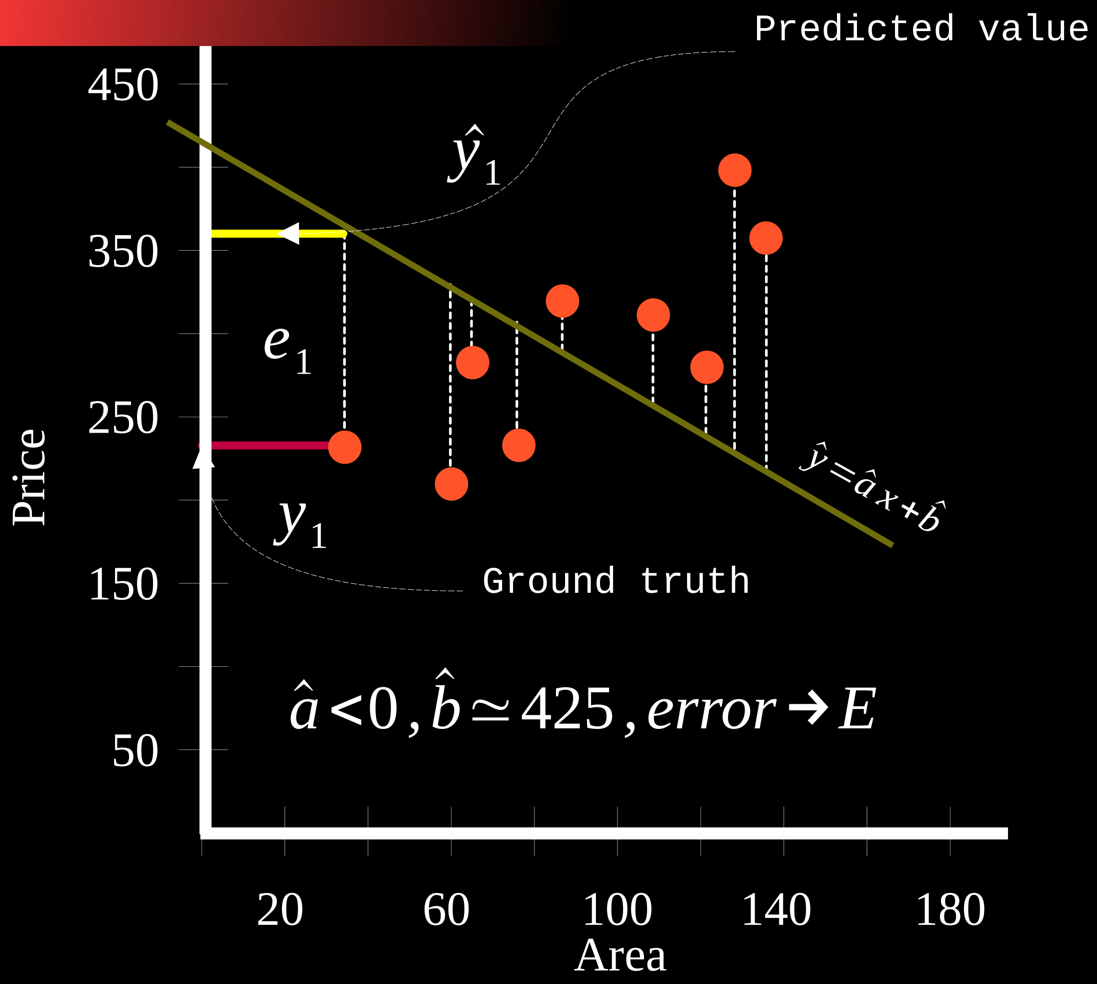
Model improvement after n+k (k>0) iterations





How are these parameters adjusted?

Theory



$$e_1=(y_1-\hat{y}_1)^2=[y_1-(\hat{a}x_1+\hat{b})]^2$$

$$E=(e_1+e_2+\ldots+e_9)/I=\frac{1}{I}\sum_{i=1}^{I}e_i$$

E is called **Mean Squared Error (MSE)** and I=9, is the number of data points.

$$E=\frac{1}{I}\sum_{i=1}^{I}[y_i-(\hat{a}x_i+\hat{b})]^2$$

How to minimize error E?
By finding the optimal values of â and ̂b that minimize this error.
Let’s differentiate E to find the optimal parameters.

Theory

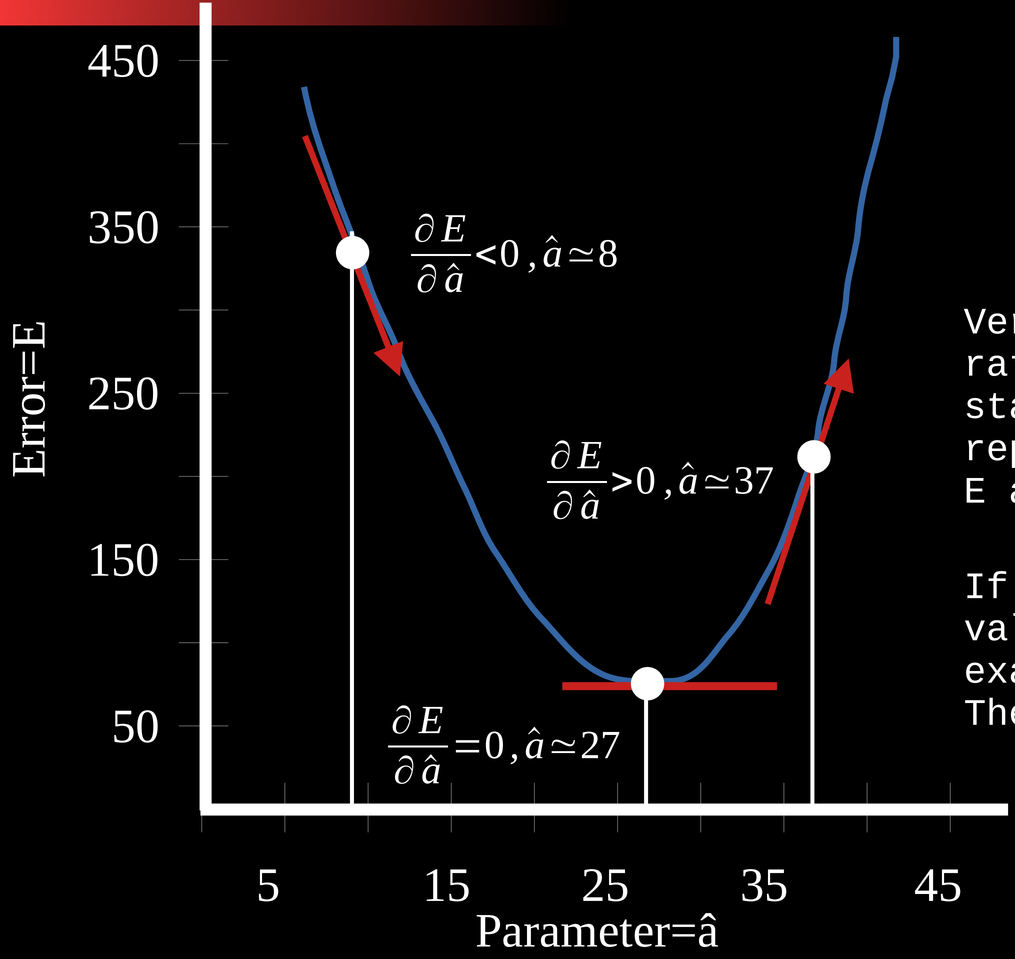
$$E = \frac{1}{I}\sum_{i=1}^{I}\left[y_i - \left(\hat{a}x_i + \hat{b}\right)\right]^2$$

$$\left[\left(u-v\right)^2\right]' = 2\left(u-v\right)\left(u'-v'\right), u = y_i \text{ and } v = \left(\hat{a}x_i + \hat{b}\right)$$

$$\frac{\partial E}{\partial \hat{a}} = \frac{1}{I}\sum_{i=1}^{I} 2\left[y_i - \left(\hat{a}x_i + \hat{b}\right)\right]\left(-x_i\right) = \frac{-2}{I}\sum_{i=1}^{I}\left[y_i - \left(\hat{a}x_i + \hat{b}\right)\right]\left(x_i\right)$$

$$\frac{\partial E}{\partial \hat{b}} = \frac{1}{I}\sum_{i=1}^{I} 2\left[y_i - \left(\hat{a}x_i + \hat{b}\right)\right]\left(-1\right) = \frac{-2}{I}\sum_{i=1}^{I}\left[y_i - \left(\hat{a}x_i + \hat{b}\right)\right]$$

Reminder on the derivatives of functions



Suppose we want to calculate the derivative of E at the point â=x, formally we have:

$$E'(\hat{a}=x) = \frac{\partial E}{\partial \hat{a}}$$

Verbally, the above function means: what is the rate of change of E (decrease, increase, stagnation), when we are at point â. This rate is represented by a vector tangent to the curve of E at â.

If we want to minimize E, we need to find the value â that leads to this minimization In our example, this value is approximately equal to 27. The same process is carried out for b̂.

Let’s calculate the derivative of E as a function of â and b̂.

Linear algebra reminder

Addition of vectors

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} a_1+b_1 \\ a_2+b_2 \\ \vdots \\ a_k+b_k \end{bmatrix} \text{ Exemple } \rightarrow \begin{bmatrix} 2.5 \\ 18.0 \\ \vdots \\ 7.5 \end{bmatrix} + \begin{bmatrix} 3.9 \\ 8.1 \\ \vdots \\ 4.2 \end{bmatrix} = \begin{bmatrix} 6.4 \\ 26.1 \\ \vdots \\ 11.7 \end{bmatrix}$$

Product of a scalar with a vector

$$X \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} Xa_1 \\ Xa_2 \\ \vdots \\ Xa_k \end{bmatrix} \text{ Exemple } \rightarrow 2 \begin{bmatrix} 3.9 \\ 8.1 \\ \vdots \\ 4.2 \end{bmatrix} = \begin{bmatrix} 7.8 \\ 16.2 \\ \vdots \\ 8.4 \end{bmatrix}$$

Linear algebra reminder

To multiply two matrices, the number of columns in the first matrix must be equal to the number of rows in the second.

Product of matrices

2 x 2 2 x 3 2 x 3

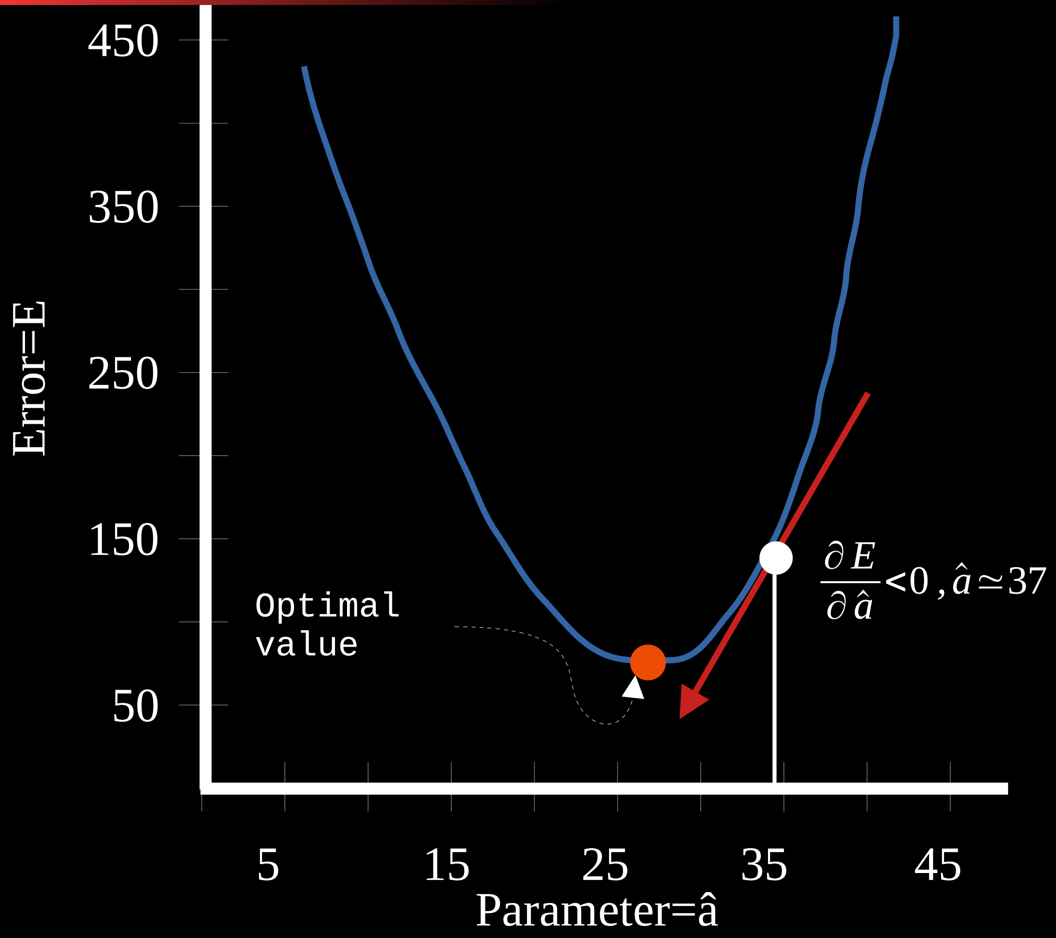
$$\begin{bmatrix} a_1^1 & a_2^1 \\ a_1^2 & a_2^2 \end{bmatrix} \begin{bmatrix} b_1^1 & b_2^1 & b_3^1 \\ b_1^2 & b_2^2 & b_3^2 \end{bmatrix} = \begin{bmatrix} a_1^1 b_1^1 + a_2^1 b_1^2 & a_1^1 b_2^1 + a_2^1 b_2^2 & a_1^1 b_3^1 + a_2^1 b_3^2 \\ a_1^1 b_1^1 + a_2^1 b_1^2 & a_1^1 b_2^1 + a_2^1 b_2^2 & a_1^1 b_3^1 + a_2^1 b_3^2 \end{bmatrix}$$

Exemple

$$\begin{bmatrix} 2{,}1 & 4{,}3 \\ 1{,}7 & 7{,}0 \end{bmatrix} \begin{bmatrix} 2{,}5 & 10{,}0 & 13{,}2 \\ 5{,}4 & 4{,}4 & 6{,}5 \end{bmatrix} = \begin{bmatrix} 2{,}1*2{,}5+4{,}3*5{,}4 & 2{,}1*10{,}0+4{,}3*4{,}4 & 2{,}1*13{,}2+4{,}3*6{,}5 \\ 1{,}7*2{,}5+7{,}0*5{,}4 & 1{,}7*10{,}0+7{,}0*4{,}4 & 1{,}7*13{,}2+7{,}0*6{,}5 \end{bmatrix}$$

$$\begin{bmatrix} 2{,}1 & 4{,}3 \\ 1{,}7 & 7{,}0 \end{bmatrix} \begin{bmatrix} 2{,}5 & 10{,}0 & 13{,}2 \\ 5{,}4 & 4{,}4 & 6{,}5 \end{bmatrix} = \begin{bmatrix} 28{,}47 & 39{,}92 & 55{,}67 \\ 42{,}05 & 47{,}8 & 67{,}94 \end{bmatrix}$$

Theory



The new value of â will be:

$$\hat{a} = \hat{a} - \ni \frac{\partial E}{\partial \hat{a}}$$

$\ni > 0$ Is the **learning rate**. It is often set at 0.001.

This adjustment process called **gradient descent** is repeated n times until the optimal value of â (as well as b̂)that minimizes E is found.

What happens if we have several features?

Theory

$$\hat{y}=\hat{a}_1 x_1+\hat{a}_2 x_2+\ldots+\hat{a}_k+x_k+\hat{b}=[x_1, x_2,\ldots,x_k]\begin{bmatrix}\hat{a}_1\\ \hat{a}_2\\ \vdots\\ \hat{a}_k\end{bmatrix}+\hat{b}$$

$$\hat{y}=X\theta+\hat{b} \mid X=[x_1, x_2,\ldots,x_k], \theta=\begin{bmatrix}\hat{a}_1\\ \hat{a}_2\\ \vdots\\ \hat{a}_k\end{bmatrix}$$

$$E=\frac{1}{I}\sum_{i=1}^{I}[y_i-(X_i\theta+\hat{b})]^2$$

$$X=[x_1, x_2,\ldots,x_k] \rightarrow X^T=\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_k\end{bmatrix}$$

$$\nabla_\theta E(\theta)=\frac{1}{I}\sum_{i=1}^{I}2[y_i-(X_i\theta+\hat{b})](-X_i)^T=\frac{-2}{I}\sum_{i=1}^{I}[y_i-(X_i\theta+\hat{b})](X_i)^T$$

Theory

$$X=[x_1,x_2,\ldots,x_k]\rightarrow X^T=\begin{bmatrix}x_1\\x_2\\\vdots\\x_k\end{bmatrix}$$

$$\alpha_i=[y_i-(X_i\theta+\hat{b})]$$

$$\nabla_\theta E(\theta)=\frac{-2}{I}\sum_{i=1}^{I}[y_i-(X_i\theta+\hat{b})](X_i)^T=\frac{-2}{I}\sum_{i=1}^{I}\alpha_i\begin{bmatrix}x_1^i\\x_2^i\\\vdots\\x_k^i\end{bmatrix}$$

$$\theta=\theta-\ni\nabla_\theta E(\theta)=\theta-\ni\frac{-2}{I}\sum_{i=1}^{I}\alpha_i\begin{bmatrix}x_1\\x_2\\\vdots\\x_k\end{bmatrix}=\begin{bmatrix}\hat{a}_1\\\hat{a}_2\\\vdots\\\hat{a}_k\end{bmatrix}-\ni\frac{-2}{I}\sum_{i=1}^{I}\alpha_i\begin{bmatrix}x_1^i\\x_2^i\\\vdots\\x_k^i\end{bmatrix}$$

Use Case

Cloud computing with Google Colab

Environment setting with Bash Script

Programming language Python

Rental price prediction based on factors such as size, number of rooms, location, and others.

IDEs : Visual Studio & Jupyter Notebook.

Model versioning

Tools used to implement the model

A Brief Introduction to Python

- *Python is a programming language coded in C;*
- *Imperative and interpreted language;*
- *Most used language in data science;*
- *Includes various libraries, such as Pandas for structured data manipulation and NumPy for mathematical calculations, among others.*

https://www.datacamp.com/fr/blog/top-programming-languages-for-data-scientists-in-2022

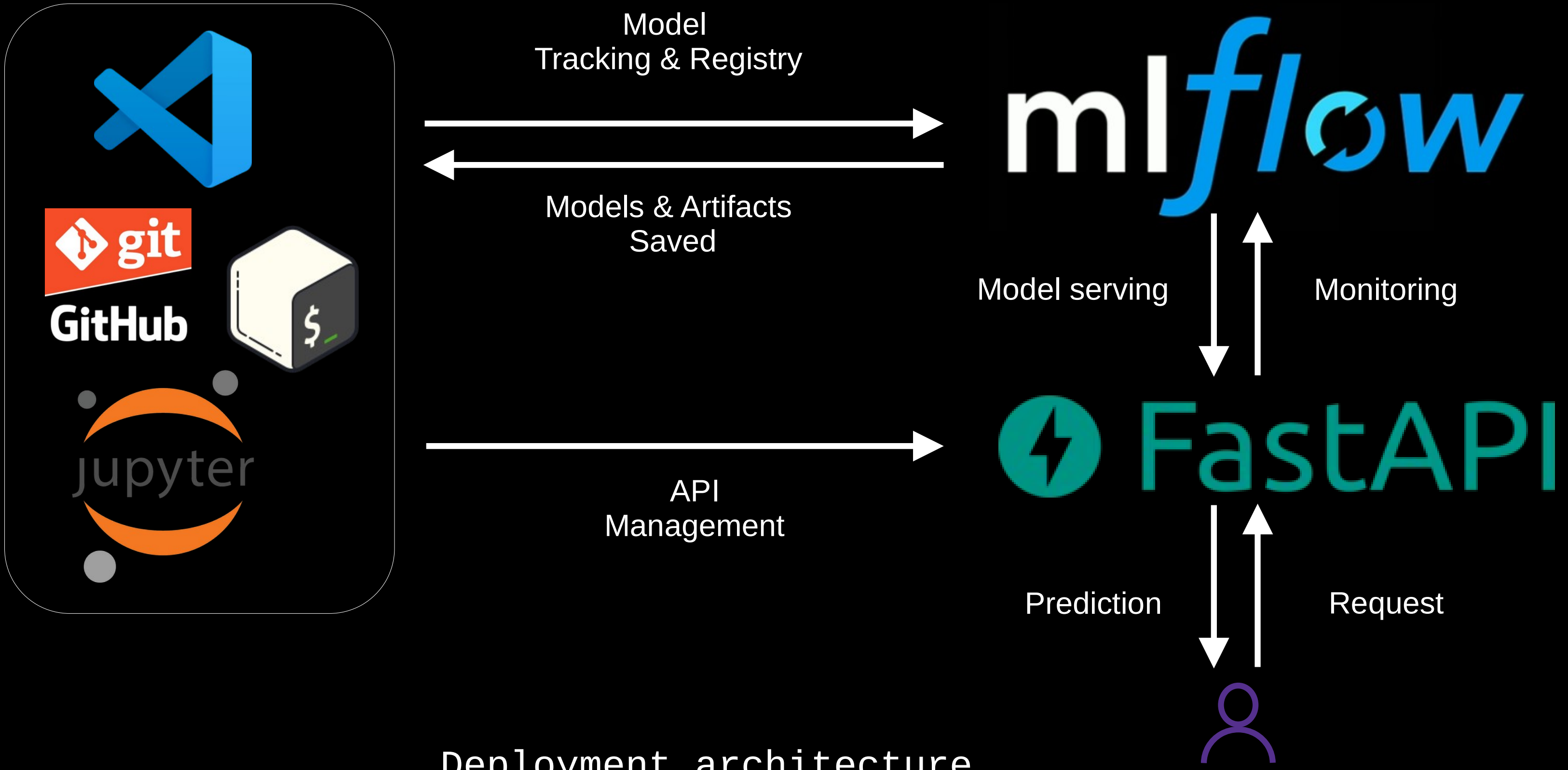
Use Case

```python
observations=['pieds_carres','nombre_chambres','nombre_etages','distance_centre_ville','annee_construction','taille_garage','score_localisation','avec_piscine']
# observations=['pieds_carres','nombre_etages','distance_centre_ville','annee_construction']


X=data[observations].values#Observations

print('Example des observations/features avant la standardization.')
print(X[0])

scaler = StandardScaler()
X_ = scaler.fit_transform(X)
print('\n')
print('Example des observations/features après la standardization.')
print(X_[0])

y=data[['prix']].values#Target (cible)
#2% des données (soit 100 échantillons/samples) seront utilisées pour le test.
#random_state permet  de reproduire la distribution des données d'entrainement et de test.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1234)

X_train=scaler.fit_transform(X_train)#Standardiser les données d'entrainement.
```

Let’s code

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mlflow
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from datetime import time
from sklearn.preprocessing import StandardScaler
import random
```

Model deployment



Deployment architecture