Demystifying Artificial Intelligence Sorcery

(Part 2: Machine Learning)[a]

Abdelbacet Mhamdi
abdelbacet.mhamdi@bizerte.r-iset.tn

Dr.-Ing. in Electrical Engineering
Senior Lecturer at ISET Bizerte

[a]Available @ https://github.com/a-mhamdi/jlai/

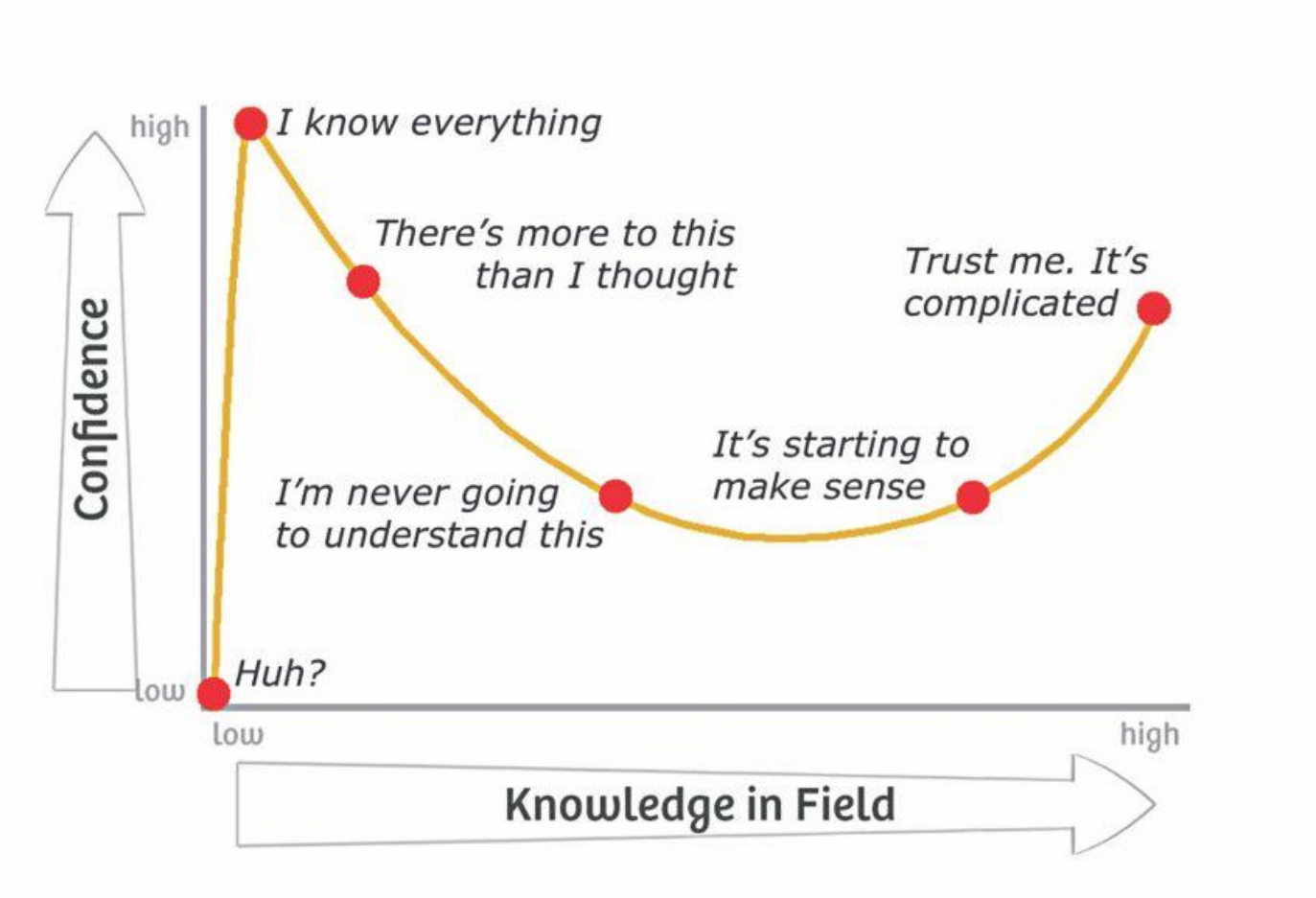
Disclaimer

This document features some materials gathered from multiple online sources.

Please note no copyright infringement is intended, and I do not own nor claim to own any of the original materials. They are used for educational purposes only.

I have included links solely as a convenience to the reader. Some links within these slides may lead to other websites, including those operated and maintained by third parties. The presence of such a link does not imply a responsibility for the linked site or an endorsement of the linked site, its operator, or its contents.

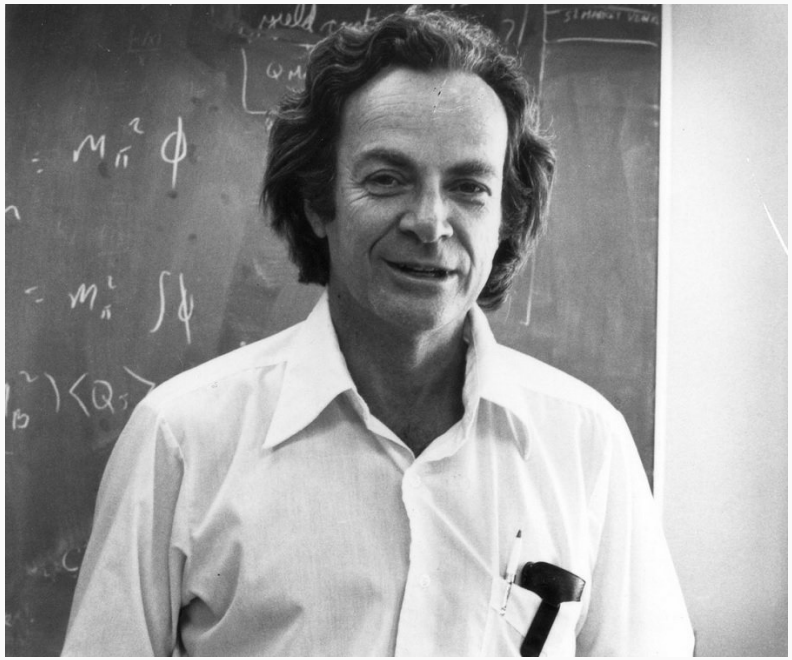
Dunning-Kruger Effect



Kruger, J. and Dunning, D. (1999) *Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments.* **J Pers Soc Psychol.** 77(6) pp. 1121–1134.
doi 10.1037/0022-3514.77.6.1121

"Knowledge isn't free. You have to pay attention."

Richard P. Feynman

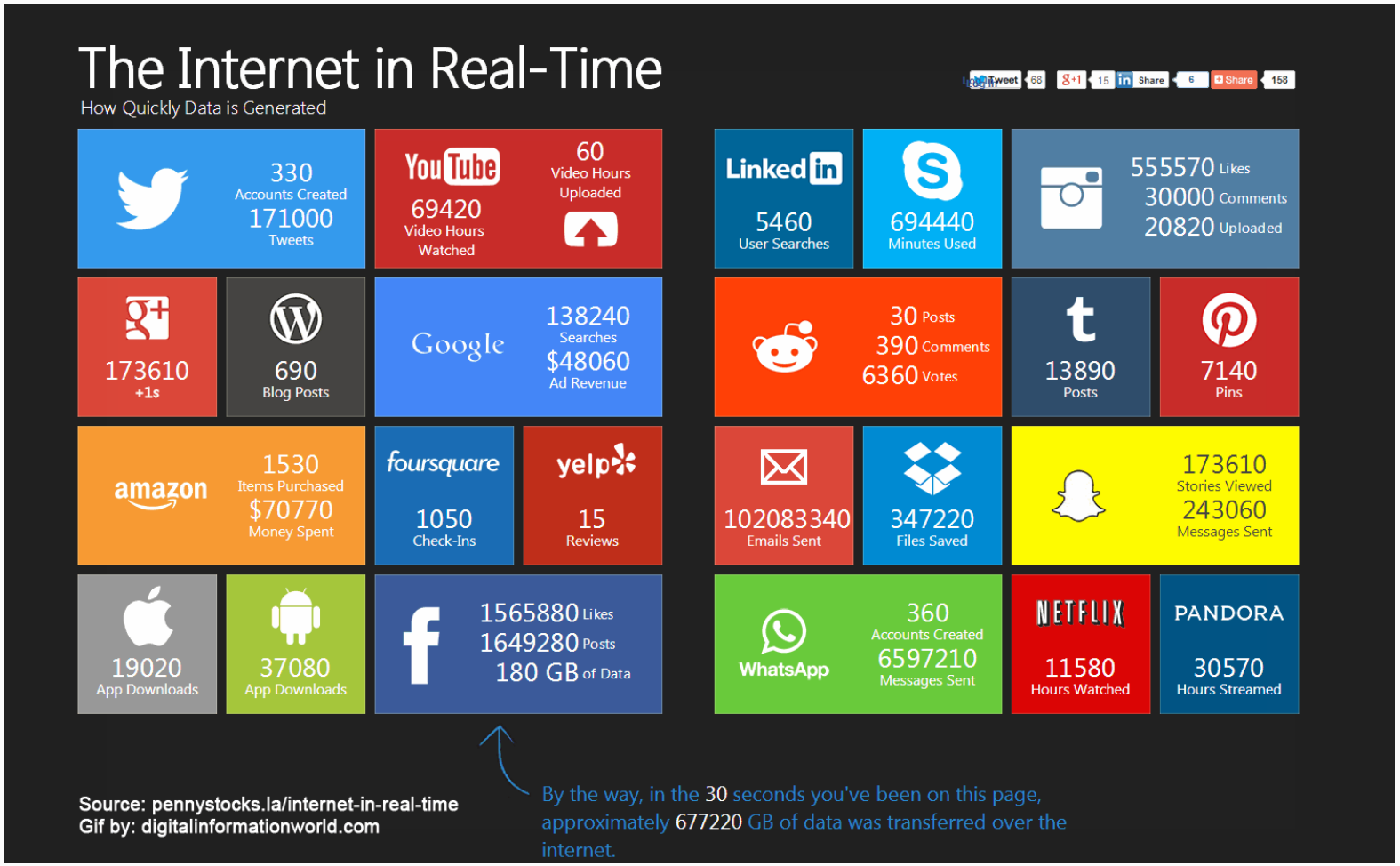


Roadmap

1. An overview
2. Supervised Learning
3. Unsupervised Learning
4. Complementary Lab. Project
5. ML Landscape through Quizzes

An overview

Global Data Traffic



Update on the internet in real time is available here.

Literature Review (1/3)

"The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience."

Mitchell, T. (1997) *Machine Learning*. **McGraw-Hill International Editions. McGraw-Hill.**

Literature Review (2/3)

"Machine learning (ML) is a scientific discipline that concerns developing learning capabilities in computer systems. Machine learning is one of central areas of Artificial Intelligence (AI). It is an interdisciplinary area that combines results from statistics, logic, robotics, computer science, computational intelligence, pattern recognition, data mining, cognitive science, and more."

Wojtusiak, J. (2012) Machine learning. **In *Encyclopedia of the Sciences of Learning*, pages 2082–2083. Springer US.**

Literature Review (3/3)

"Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment. They are considered the working horse in the new era of the so-called big data. Techniques based on machine learning have been applied successfully in diverse fields ranging from pattern recognition, computer vision, spacecraft engineering, finance, entertainment, and computational biology to biomedical and medical applications. [...] The ability of machine learning algorithms to learn from current context and generalize into unseen tasks would allow improvements in both the safety and efficacy of radiotherapy practice leading to better outcomes."

El Naqa, I. and Murphy, M. J. (2015) *What Is Machine Learning?*, pages 3–11. **Springer International Publishing.**

Debrief

Arthur Samuel (1959)
Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)
Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

Task #1
Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $\mathcal{T}$ in this setting?

1. Classifying emails as spam or not spam;
2. Watching you label emails as spam or not spam;
3. The number (or fraction) of emails correctly classified as spam/not spam;
4. None of the above-this not a machine learning problem.

Debrief

Arthur Samuel (1959)
Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998)
Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

Task #1
Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $\mathcal{T}$ in this setting?

1. Classifying emails as spam or not spam;
2. Watching you label emails as spam or not spam;
3. The number (or fraction) of emails correctly classified as spam/not spam;
4. None of the above-this not a machine learning problem.

Debrief

Arthur Samuel (1959)
Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

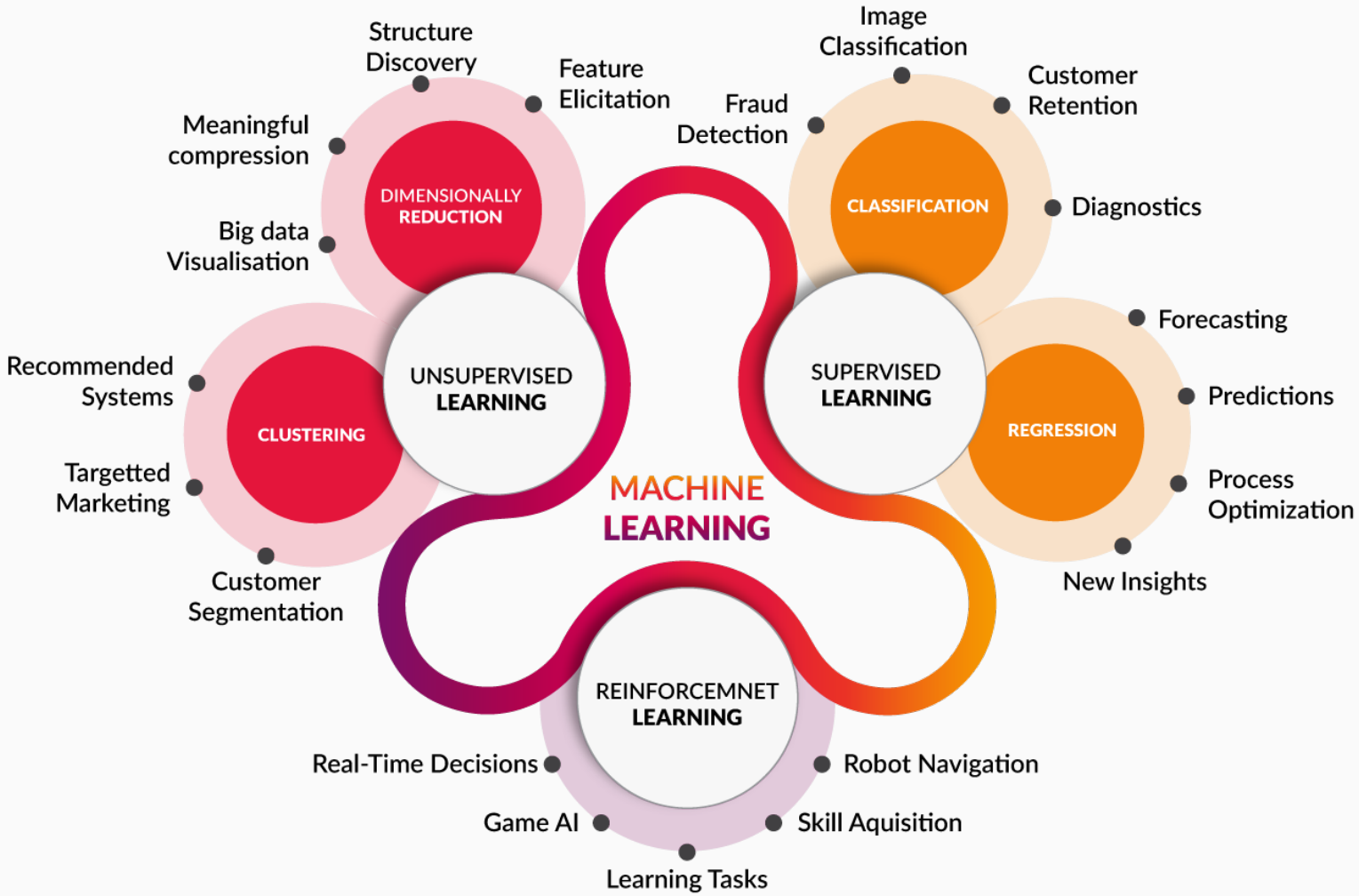
Tom Mitchell (1998)
Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

Task #1
Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task $\mathcal{T}$ in this setting?

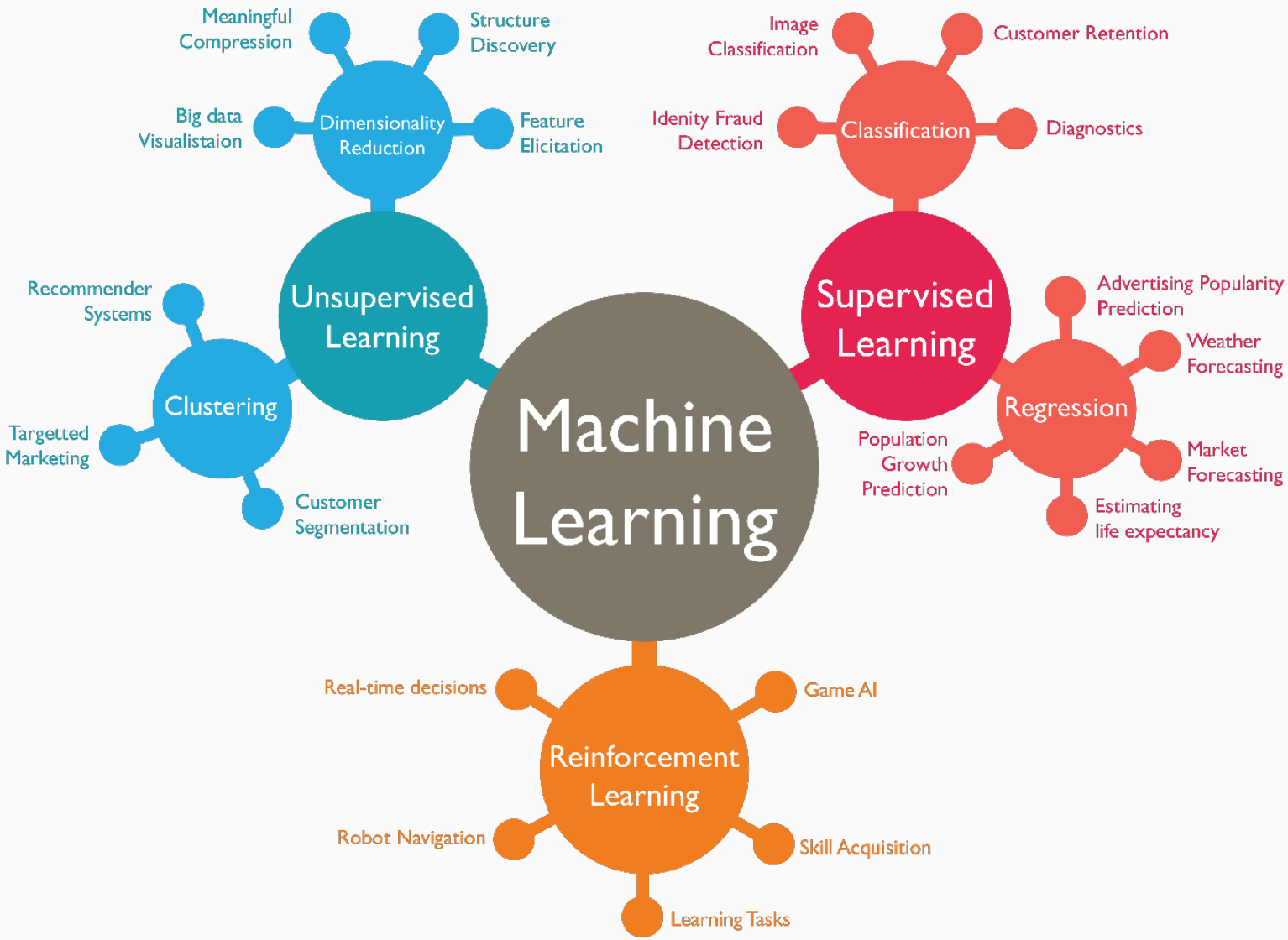
1. Classifying emails as spam or not spam;
2. Watching you label emails as spam or not spam;
3. The number (or fraction) of emails correctly classified as spam/not spam;
4. None of the above-this not a machine learning problem.

Overall Methodology

1. Define the problem;
2. Gather dataset;
3. Choose measure of success;
4. Decide evaluation protocol;
5. Prepare the data;
6. Develop a model;
7. Iterate models.

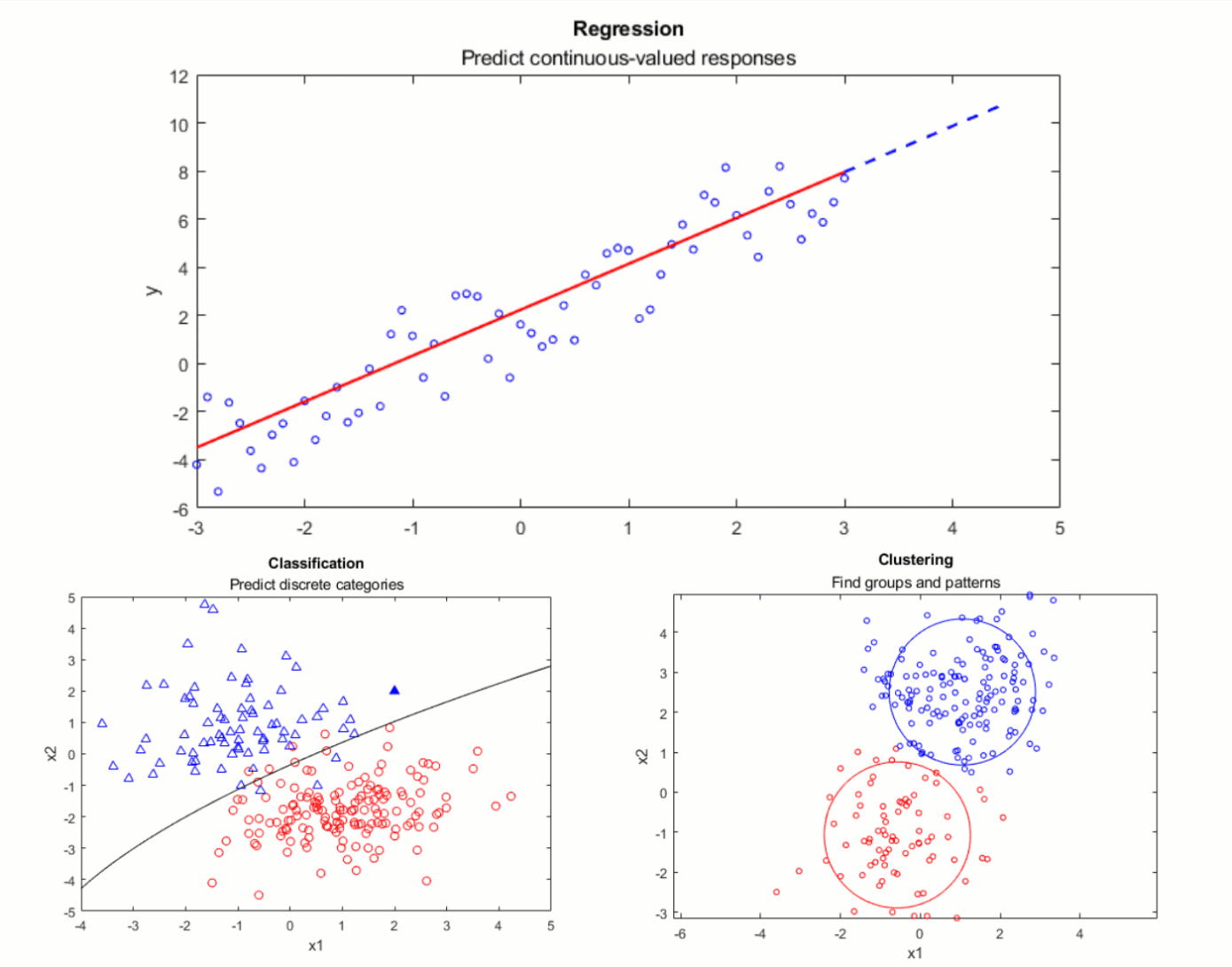


https://www.cognub.com/index.php/cognitive-platform/



https://vitalflux.com/great-mind-maps-for-learning-machine-learning/

Regression | Classification | Clustering



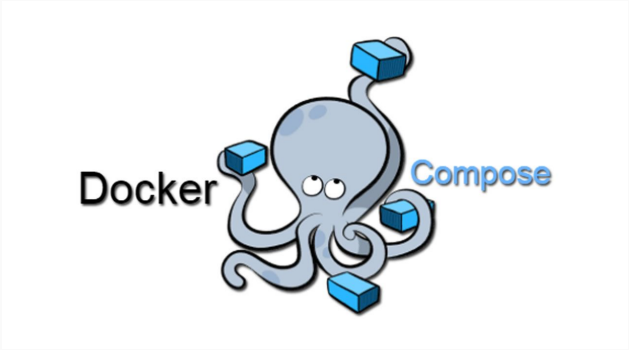
https://github.com/MathWorks-Teaching-Resources/Machine-Learning-for-Regression


REMINDER

Programming Language

julialang.org/

Development Environments

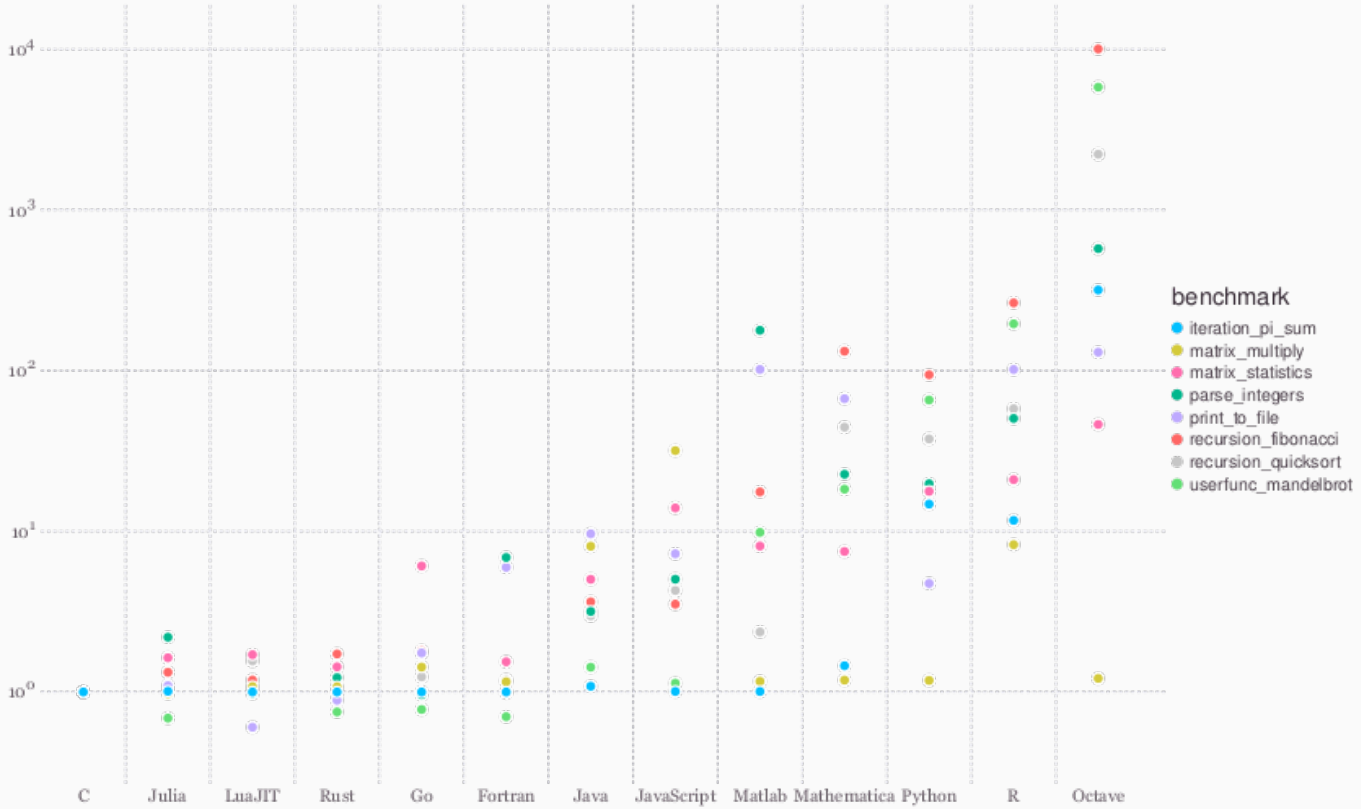


- $ docker compose up
- $ docker compose down

Julia in a Nutshell

- **Fast:** native code for multiple platforms via LLVM;
- **Dynamic:** good support for interactive use *(like a a scripting language)*;
- **Reproducible:** environment recreation across platforms, with pre-built binaries;
- **Composable:** multiple dispatch as a paradigm *(oop & functional programming)*;
- **General:** asynchronous I/O, metaprogramming, debugging, logging; profiling, pkg, ...
- **Open Source:** GitHub repository at https://github.com/JuliaLang/julia.

Julia Micro-Benchmarks (1/2)



https://julialang.org/benchmarks

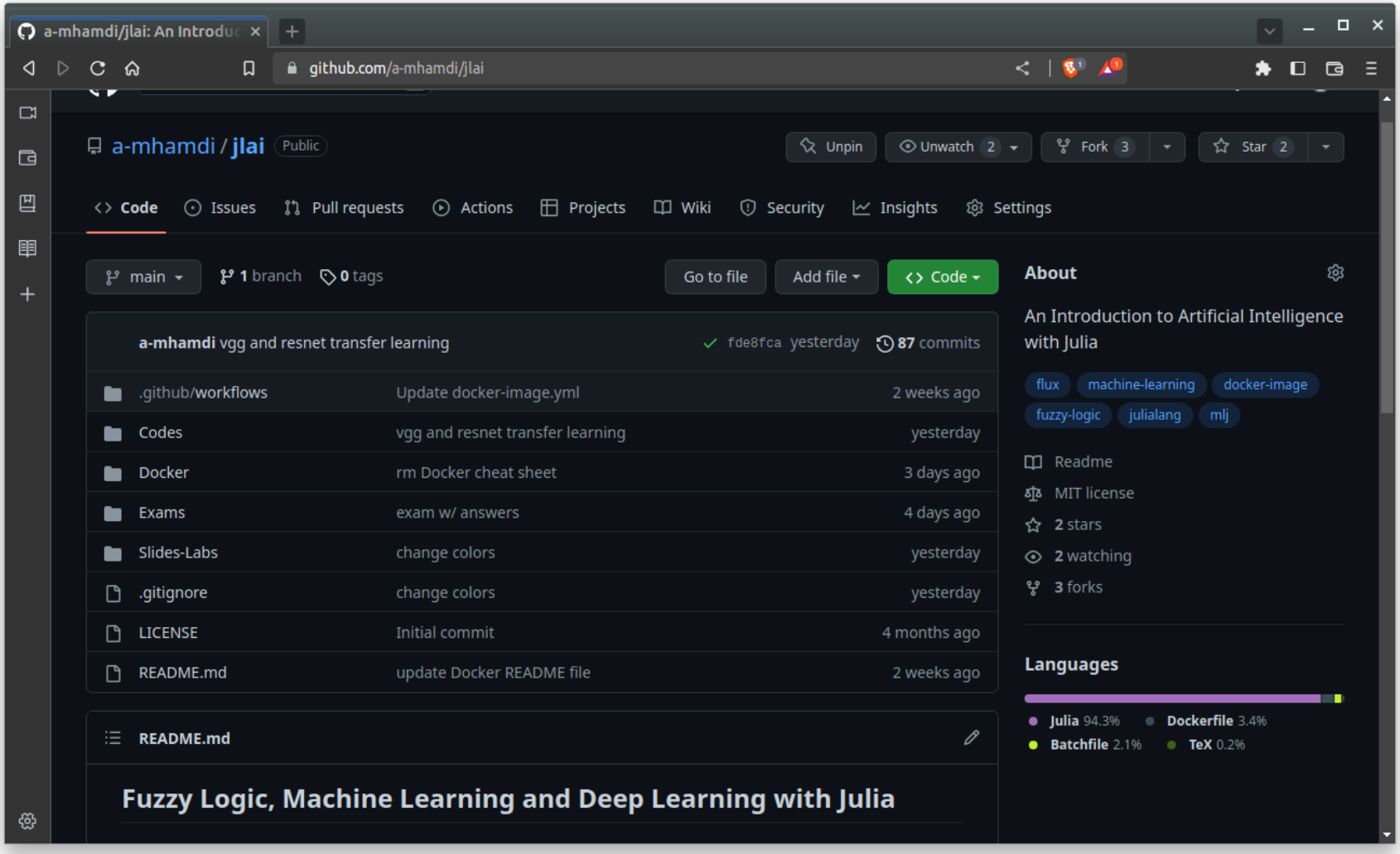
Julia Micro-Benchmarks (2/2)

Geometric Means[1] of Micro-Benchmarks by Language

1	C	1.0
2	Julia	1.17006
3	LuaJIT	1.02931
4	Rust	1.0999
5	Go	1.49917
6	Fortran	1.67022
7	Java	3.46773
8	JavaScript	4.79602
9	Matlab	9.57235
10	Mathematica	14.6387
11	Python	16.9262
12	R	48.5796
13	Octave	338.704

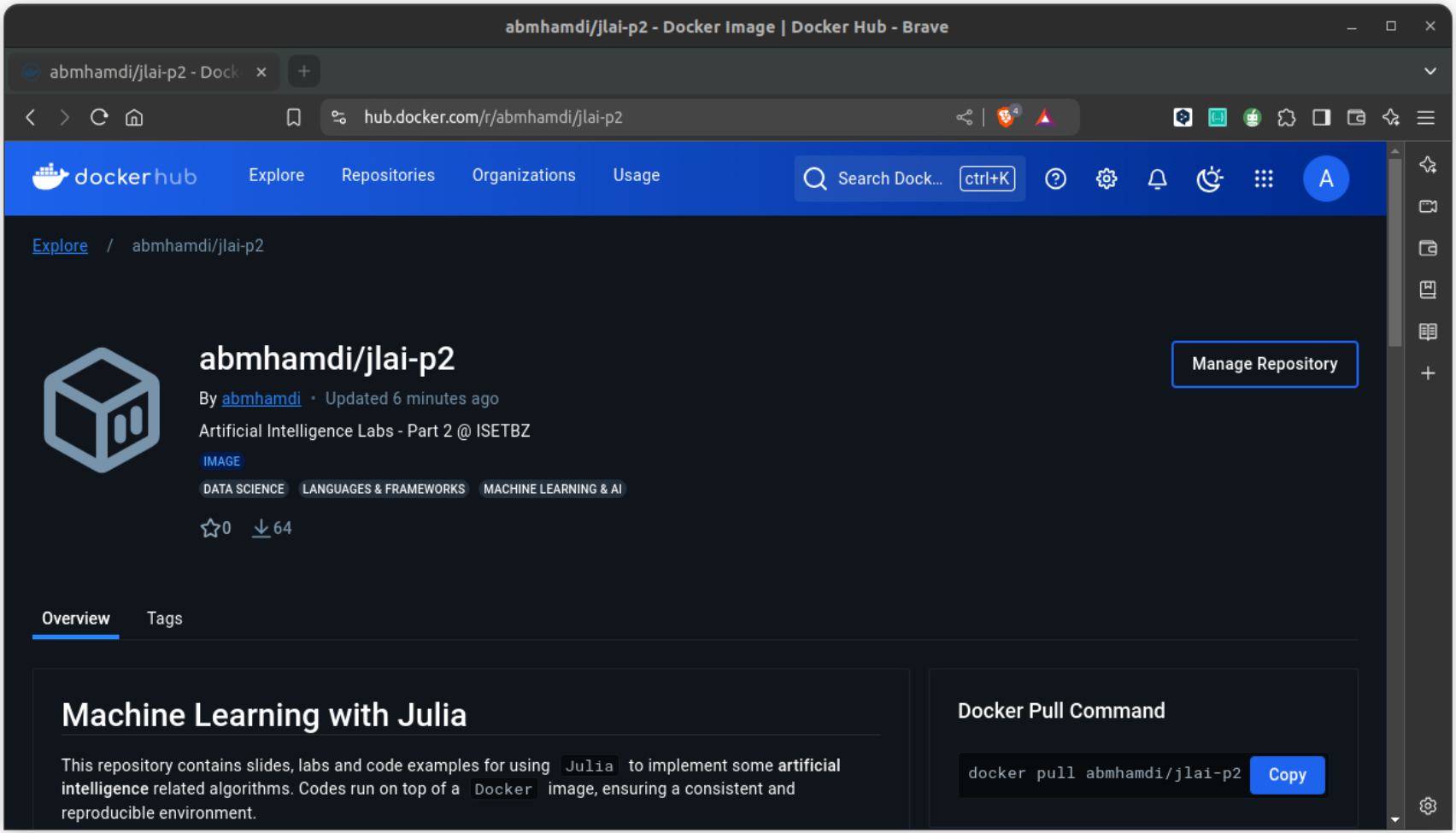
[1] Measure of central tendency expressed as $(x_1 \times x_2 \times \cdots \times x_n)^{1/n}$

Source Control Management (SCM)



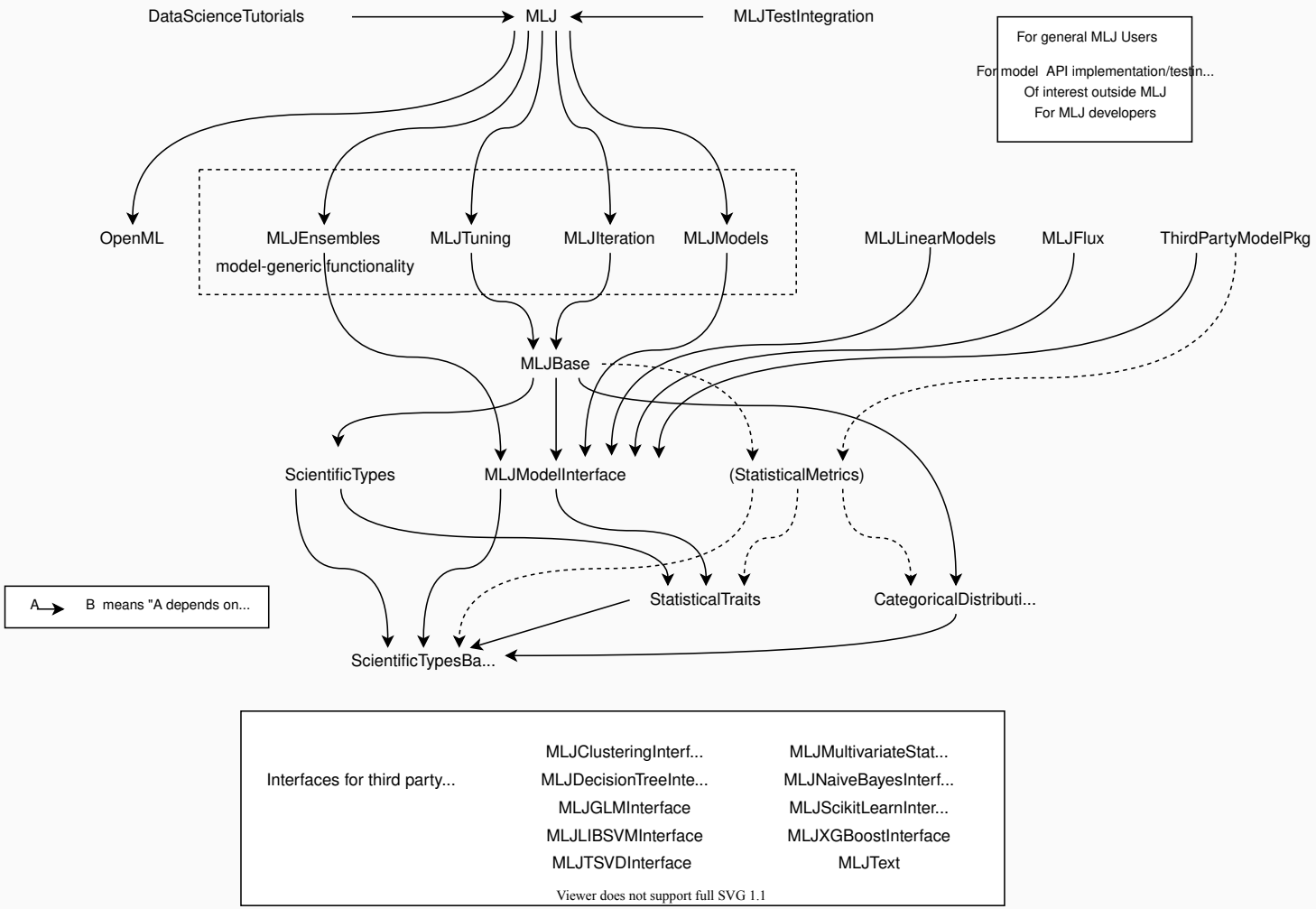
https://github.com/a-mhamdi/jlai

Continuous Integration (CI)



https://hub.docker.com/r/abmhamdi/jlai-p2

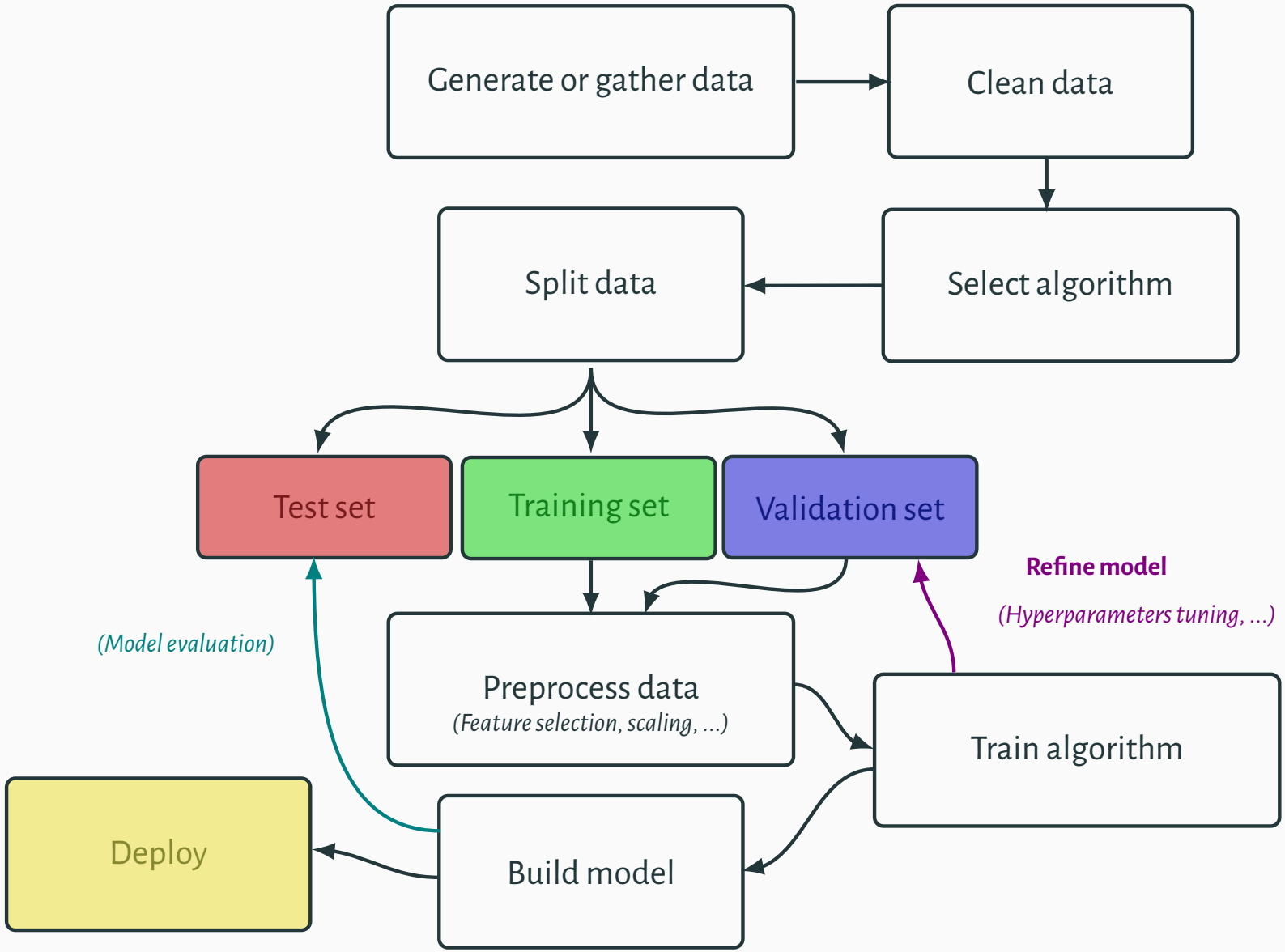
A Machine Learning Framework for Julia



https://docs.juliahub.com/MLJ/

Supervised Learning

Workflow in Machine Learning



Data Preprocessing

How?

Cleaning Identifying and correcting or removing inaccuracies and inconsistencies in the data.

Transformation Converting data from one format or structure to another.

Normalization Scaling the data so that it fits within a specific range. This is often done to make the data more amenable to certain operations or algorithms.

Data Preprocessing

Why?

- Raw data is often messy and may need to be cleaned and formatted before it can be used for machine learning.
  (This may involve removing missing or invalid data, handling outliers, and encoding categorical variables.)
- Normalizing the data can help to scale the features so that they are on the same scale.
  (This can be important for algorithms that use distance measures, as features on different scales can dominate the distance measure.)
- Preprocessing techniques such as feature selection and feature extraction can help to reduce the dimensionality of the data.
  (This may improve the performance of the model and reduce the risk of overfitting.)
- Preprocessing techniques such as feature selection can help to identify the most important features in the data.
  (This can make the model more interpretable and easier to understand.)

Data Preprocessing

Feature Scaling

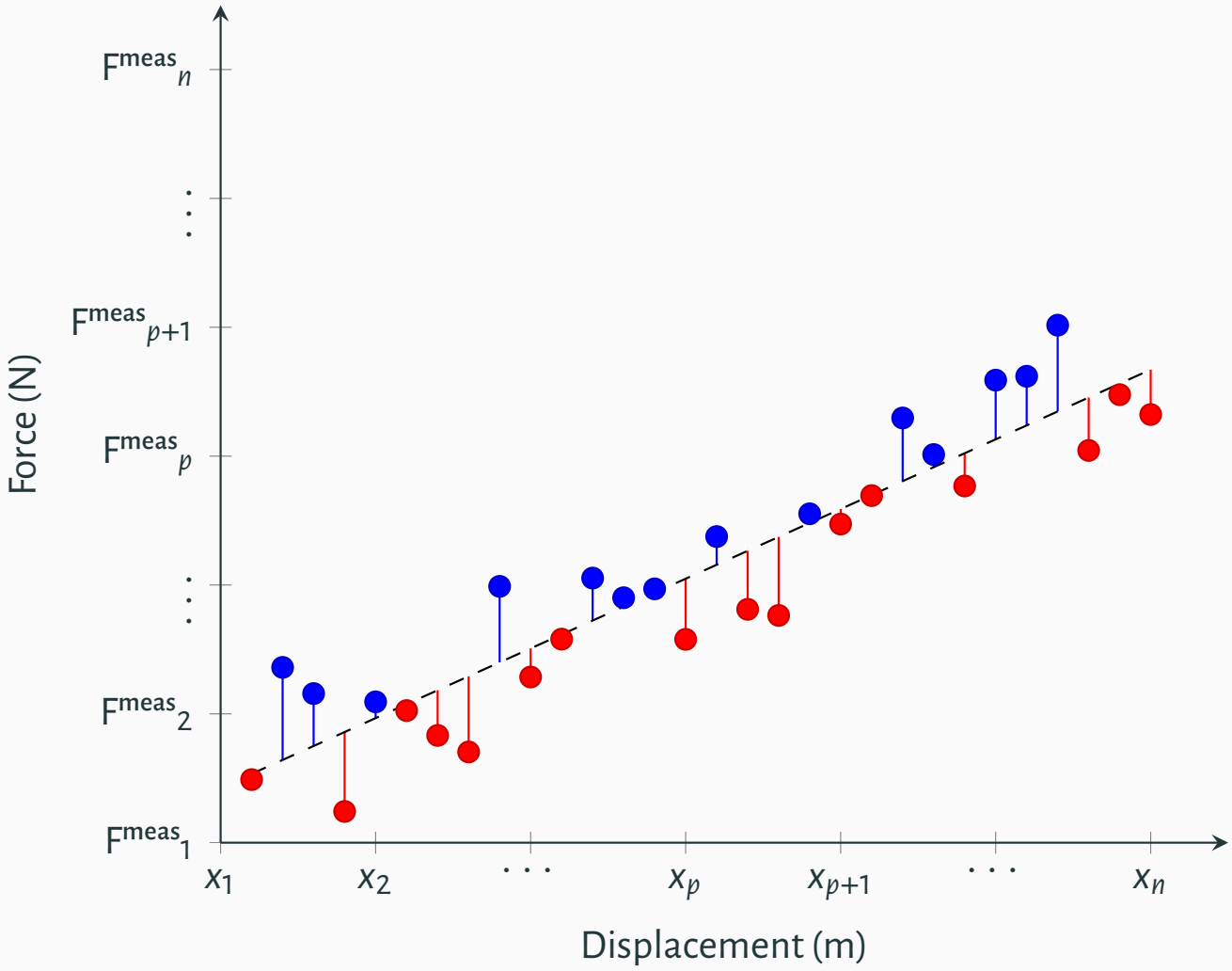
Normalization	**Standardization** *(Standardizer)*
$X \triangleq \frac{X - \text{minimum}(X)}{\text{maximum}(X) - \text{minimum}(X)}$	$X \triangleq \frac{X - \mu}{\sigma}$
▲ No assumption on data distribution	▲ More recommended when following normal distribution

Data Preprocessing Template

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *ml-workflows.jl*

⟶ *Jupyter* ⟶ *ml-workflows.ipynb*


F^{meas}_n
F^{meas}_{p+1}
F^{meas}_p
F^{meas}_2
F^{meas}_1
Force (N)
x_1
x_2
x_p
x_{p+1}
x_n
Displacement (m)

Consider the example of a spring. Our main goal is to determine the stiffness k of this spring, given some experimental data. The mathematical model *(Hooke's law)*:

$$F = kx \tag{1}$$

Restoring force is proportional to displacement.

Table 1: Measurements of couple $(x_i,\ \mathrm{F}^{\mathrm{meas}}{}_i)$

x_i	x_1	$\cdots$	x_p	$\cdots$	x_n
$\mathrm{F}^{\mathrm{meas}}{}_i$	$\mathrm{F}^{\mathrm{meas}}{}_1$	$\cdots$	$\mathrm{F}^{\mathrm{meas}}{}_p$	$\cdots$	$\mathrm{F}^{\mathrm{meas}}{}_n$

$$\begin{aligned} \mathrm{F}^{\mathrm{meas}}{}_i &= \mathrm{F}_i + \varepsilon_i \\ &= kx_i + \varepsilon_i, \end{aligned} \tag{2}$$

where F_i denotes the unknown real value of the force applied to the spring. In order to estimate the stiffness value k, we can consider the quadratic criterion:

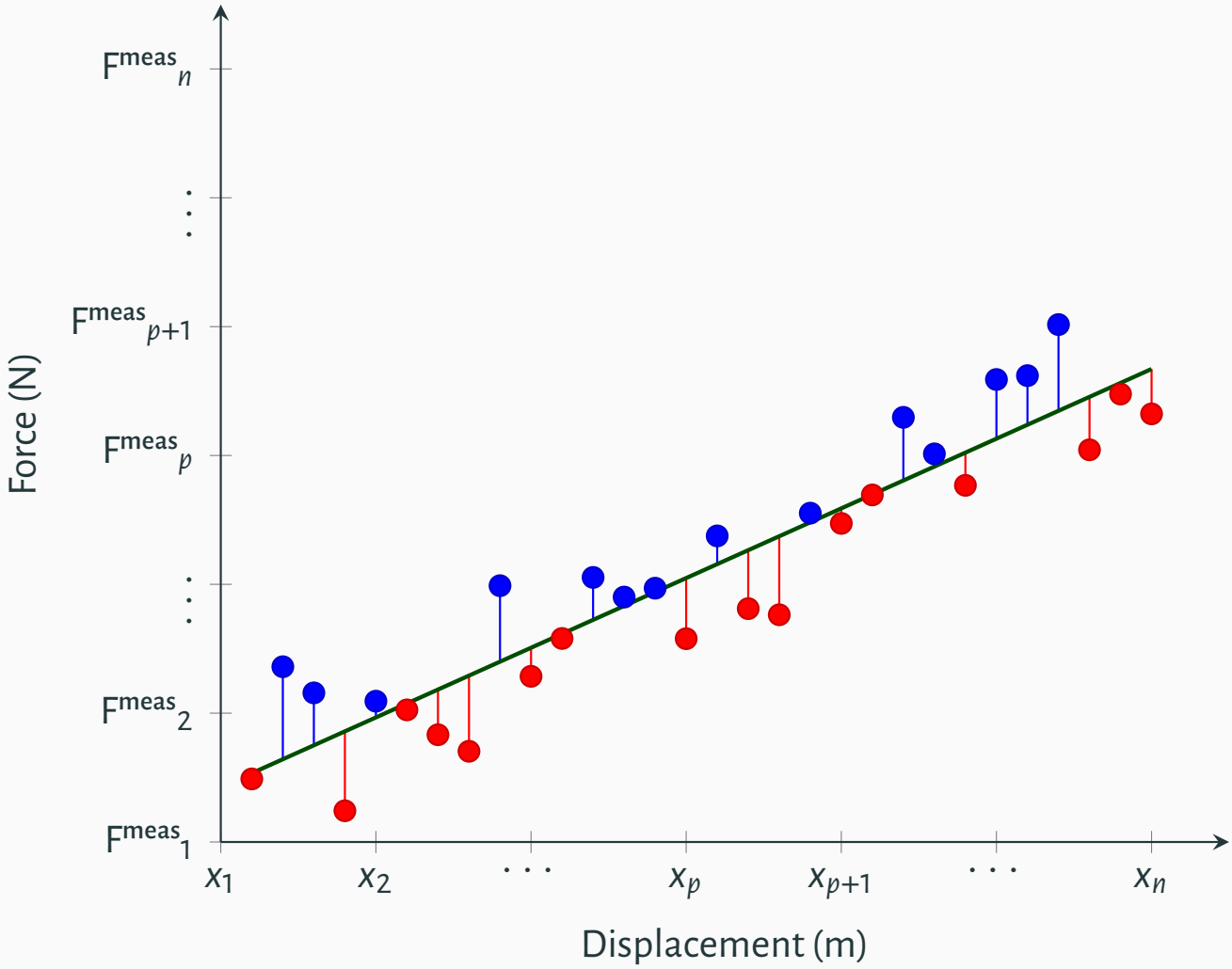
$$\begin{aligned} \mathcal{J} &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \sum_{i=1}^{n} \left(\mathrm{F}^{\mathrm{meas}}{}_i - kx_i\right)^2 \end{aligned}$$

$$\frac{\partial \mathcal{J}}{\partial k} = 0 \tag{3}$$

$$2 \sum_{i=1}^{n} \left({F^{meas}}_i - k x_i \right) \sum_{i=1}^{n} \frac{\partial \left({F^{meas}}_i - k x_i \right)}{\partial k} = 0$$

$$\sum_{i=1}^{n} \left({F^{meas}}_i - k x_i \right) \sum_{i=1}^{n} x_i = 0$$

$$\sum_{i=1}^{n} {F^{meas}}_i \, x_i = k \sum_{i=1}^{n} x_i^2 \iff \boxed{\hat{k} = \frac{\sum_{i=1}^{n} {F^{meas}}_i \, x_i}{\sum_{i=1}^{n} x_i^2}}$$


F^{meas}_n
F^{meas}_{p+1}
F^{meas}_p
F^{meas}_2
F^{meas}_1
Force (N)
x_1
x_2
x_p
x_{p+1}
x_n
Displacement (m)

Simple Linear Regression

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *simple-regression.jl*

⟶ *Jupyter* ⟶ *simple-regression.ipynb*

This example consists on determining the unknown couple $(y_0,\ v_0)$ of a mobile solid. We assume that the trajectory is linear. The mathematical model that relates the position y to time t is given by this equation:

$$y = y_0 + v_0 t \tag{4}$$

Table 2: Measurements of position y

t_k	t_1	$\cdots$	t_p	$\cdots$	t_n
$y^{meas}{}_k$	$y^{meas}{}_1$	$\cdots$	$y^{meas}{}_p$	$\cdots$	$y^{meas}{}_n$

$$\begin{aligned} y^{meas}{}_k &= y_k + \varepsilon_k \\ &= y_0 + v_0 t_k + \varepsilon_k, \end{aligned} \tag{5}$$

where y_k denotes the unknown real value of the position y at time point t_k.

In order to estimate the values taken by the couple $\begin{bmatrix} y_0, & v_0 \end{bmatrix}^T$, we consider the quadratic criterion again, as follows:

$$\begin{aligned} \mathcal{J} &= \sum_{k=1}^{n} \varepsilon_k^2 \\ &= \varepsilon^T \times \varepsilon \end{aligned}$$

The vector ε is set by $\varepsilon_k, \ \forall k \geq 1$:

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & \cdots & \varepsilon_n \end{bmatrix}^T$$

$$\frac{\partial \mathcal{J}}{\partial \begin{bmatrix} y_0 \\ \\ v_0 \end{bmatrix}} = 0 \tag{6}$$

Multiple Linear Regression

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *multivariable-regression.jl*

⟶ *Jupyter* ⟶ *multivariable-regression.ipynb*

Consider the following multivariable equation:

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m \tag{7}$$

For a particular single measurement, eq. (7) can be updated as

$$y_k = \theta_1 x_{(1,k)} + \theta_2 x_{(2,k)} + \cdots + \theta_m x_{(m,k)} + \varepsilon_k$$

We denote hereafter by θ the vector $\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}$. The function y_k becomes:

$$y_k = \underbrace{\left[x_{(1,k)},\, x_{(2,k)},\, \cdots,\, x_{(m,k)}\right]}_{x_k^T} \theta + \varepsilon_k$$

We assume that we have n measurements for y. Then we can transform the previous equation into

$$y = H\theta + \varepsilon,$$

where $y^T = [y_1,\, y_2,\, \cdots,\, y_n]$, $X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$, and $\varepsilon^T = [\varepsilon_1,\, \varepsilon_2,\, \cdots,\, \varepsilon_n]$.

We can consider the mean squared error or quadratic criterion in order to compute the approximated value of θ:

$$\begin{aligned} \mathcal{J} &= \sum_{k=1}^{n} \varepsilon_k^2 \\ &= \varepsilon^T \varepsilon \end{aligned}$$

The best well estimated value of $\hat{\theta}$ corresponds to the absolute minimum of $\mathcal{J}$. This leads to calculate the gradient of $\mathcal{J}$ with respect to θ:

$$\frac{\partial \mathcal{J}}{\partial \theta} = \frac{\partial \left(\varepsilon^T \varepsilon\right)}{\partial \theta}$$

$$\frac{\partial \left(\varepsilon^T \varepsilon\right)}{\partial \theta} = 2 \left(\frac{\partial \varepsilon}{\partial \theta}\right)^T \varepsilon$$

Recall that $\varepsilon = y - X\theta$, the term $\frac{\partial \varepsilon}{\partial \theta}$ hence becomes:

$$\frac{\partial \varepsilon}{\partial \theta} = -X$$

$$\begin{aligned}\frac{\partial J}{\partial \theta} &= 2(-X)^T (y - X\theta) \\ &= 0\end{aligned}$$

The vector $\hat{\theta}$ is given by $\boxed{\hat{\theta} = \left(X^T X\right)^{-1} X^T y}$

$X^T X$ **is not invertible (singular/degenerate)**

▼ Redundant Features

Some features are linearly dependent, i.e, $\exists$ some $x_p \propto$ some x_l, e.g., x_p in feet and x_l in m.

▼ Too many features

Fewer observations compared to the number of features, i.e, $m \geq n$.

- Delete some features
- Add extra observations
- Use regularization: $\underbrace{\lambda \sum_{i=2}^{m} |\theta_i|}_{\text{Lasso}} \quad \underbrace{\frac{1}{2}\lambda \sum_{i=2}^{m} \theta_i^2}_{\text{Ridge}} \quad \underbrace{r\lambda \sum_{i=2}^{m} |\theta_i| + \frac{(1-r)}{2}\lambda \sum_{i=2}^{m} \theta_i^2}_{\text{Elastic net}}$

$$\hat{\theta} = \left(X^T X + \lambda \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & 1 & \ddots & & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}_{(m,m)} \right)^{-1} X^T y$$

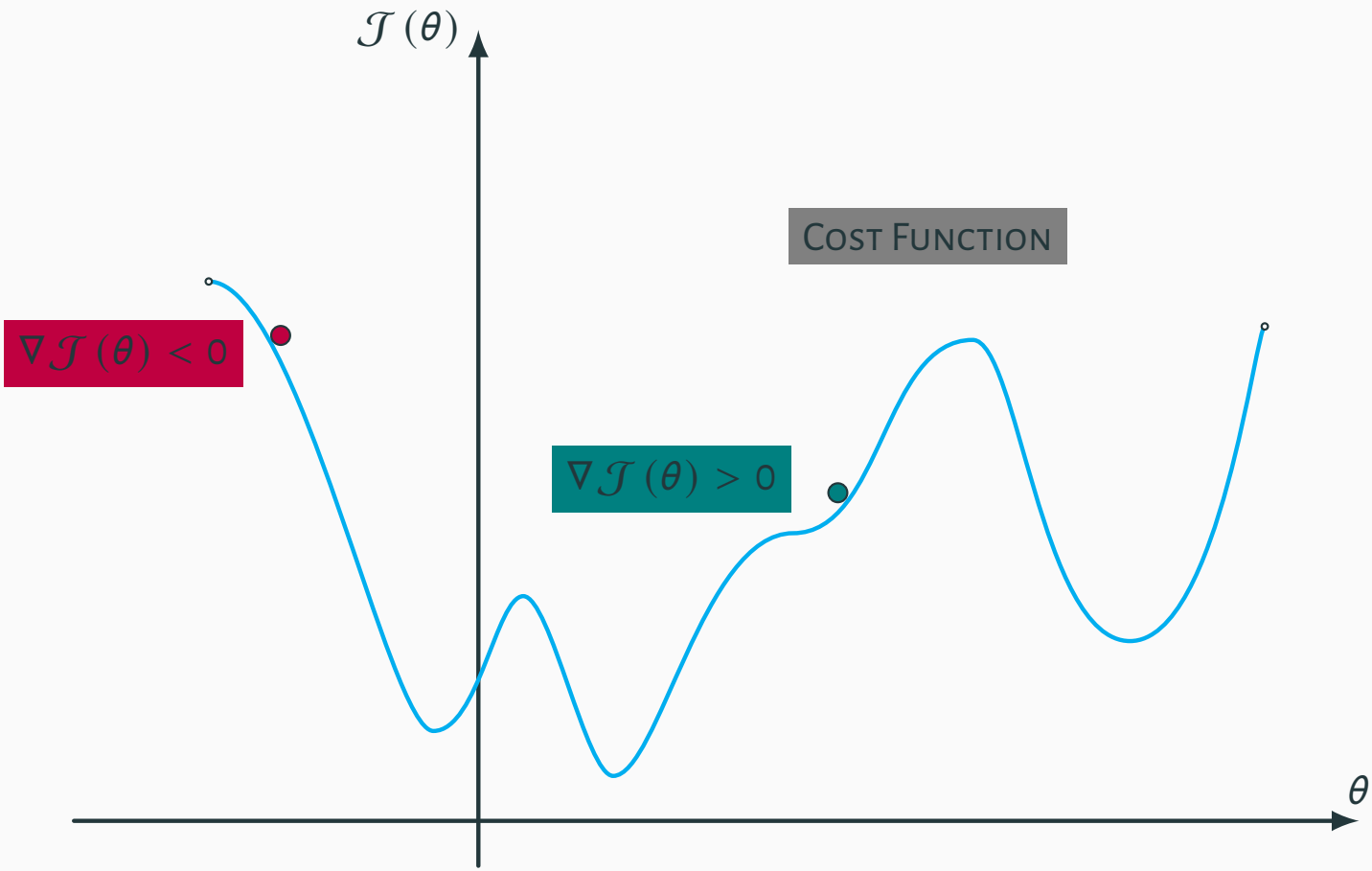
Polynomial Regression

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *polynomial-regression.jl*

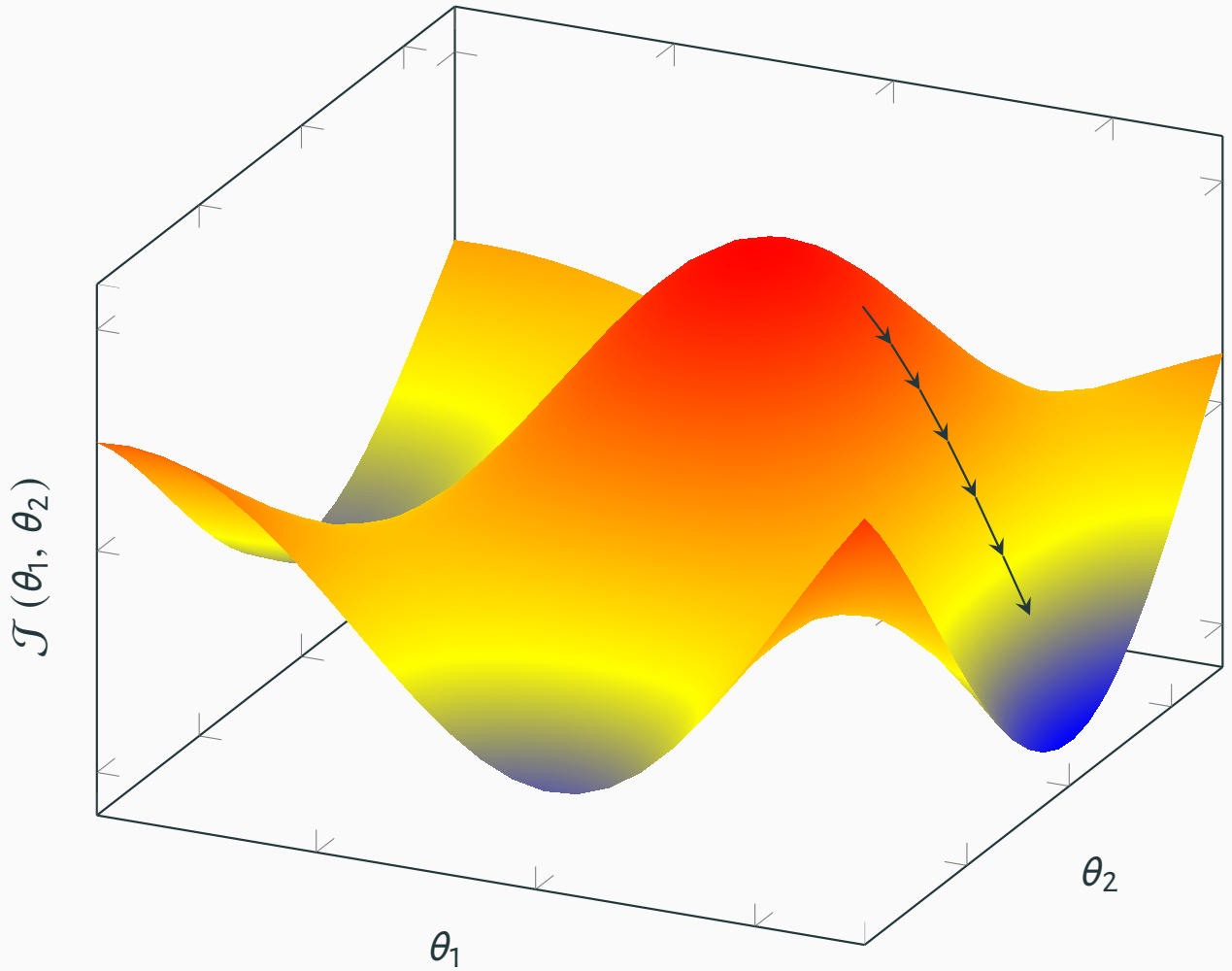
⟶ *Jupyter* ⟶ *polynomial-regression.ipynb*

Gradient Descent



$$\underbrace{\theta}_{\text{New value}} \triangleq \underbrace{\theta}_{\text{Old value}} - \overbrace{\alpha}^{\text{Weighting factor}} \underbrace{\nabla\mathcal{J}(\theta)}_{\text{Gradient term is steepest ascent}}$$

Gradient Descent



① Start with some random values of θ_1 and θ_2

② Keep changing θ_1 and θ_2 to reduce $\mathcal{J}(\theta_1, \theta_2)$ until we hopefully end up at minimum

Gradient Descent

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Recall that

$$\mathcal{J} = \frac{1}{2n} \sum_{k=1}^{n} \left(y_k - h_\theta(x_k)\right)^2 \quad \Longrightarrow \quad \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_\theta(x_k)\right) x_{(i,k)}$$

$$\theta_1 \triangleq \theta_1 + \alpha \frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_\theta(x_k)\right) x_{(1,k)} \qquad \theta_2 \triangleq \theta_2 + \alpha \frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_\theta(x_k)\right) x_{(2,k)}$$

$$\vdots$$

$$\theta_m \triangleq \theta_m + \alpha \frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_\theta(x_k)\right) x_{(m,k)}$$

Gradient Descent

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Recall that with $\mathbf{L}_2$ regularization term

$$\mathcal{J} = \frac{1}{2n}\sum_{k=1}^{n}\left(y_k - h_\theta(x_k)\right)^2 + \frac{\lambda}{2n}\sum_{i=2}^{m}\theta_i^2 \quad \Longrightarrow \quad \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta(x_k)\right)x_{(i,k)} + \frac{\lambda}{n}\theta_i \text{ iff } i \neq 1$$

$$\theta_1 \triangleq \left(1 - \alpha\cancel{\frac{\lambda}{n}}\right)\theta_1 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(1,k)}$$

$$\theta_2 \triangleq \left(1 - \alpha\frac{\lambda}{n}\right)\theta_2 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(2,k)}$$

$$\vdots$$

$$\theta_m \triangleq \left(1 - \alpha\frac{\lambda}{n}\right)\theta_m + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(m,k)}$$

Task #2

The yield y of a chemical process is a random variable whose value is considered to be a linear function of the temperature x. The following data of corresponding values of x and y is found:

Temperature in °C (x)	0	25	50	75	100
Yield in grams (y)	14	38	54	76	95

The linear regression model $y = \theta_1 + \theta_2 x$ is used. Determine the values of θ_0, θ_1.

1. Using normal equation,
2. Using gradient descent for 5 iterations, given the following initial settings:

$$\alpha = 0.01 \qquad \text{and} \qquad \theta = \begin{bmatrix} 1 \\ \\ 0.5 \end{bmatrix}$$

① Normal Equation

$$y = \begin{bmatrix} 14 \\ 38 \\ 54 \\ 76 \\ 95 \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} 1 & 0 \\ 1 & 25 \\ 1 & 50 \\ 1 & 75 \\ 1 & 100 \end{bmatrix} \implies \hat{\theta} = \begin{bmatrix} \hat{\theta}_1 \\ \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} 15.4 \\ \\ 0.8 \end{bmatrix}$$

② Stochastic Gradient Descent

k	1	2	3	4	5
y	14	38	54	76	95
$h_\theta(x_k)$	1	13.63	330.999	−9894.410	734688.376
$\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 \\ \\ \hat{\theta}_2 \end{bmatrix}$	$\begin{bmatrix} 1.13 \\ \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 1.374 \\ \\ 6.592 \end{bmatrix}$	$\begin{bmatrix} -1.396 \\ \\ -131.907 \end{bmatrix}$	$\begin{bmatrix} 98.308 \\ \\ 7345.901 \end{bmatrix}$	$\begin{bmatrix} -7247.626 \\ \\ -727247.475 \end{bmatrix}$

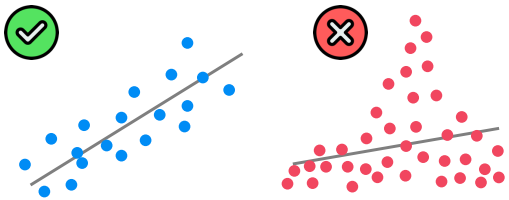
GD IN ACTION

julia

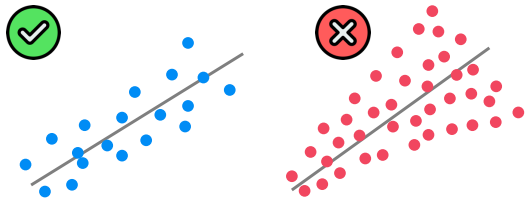
```
X = [1 0; 1 25; 1 50; 1 75; 1 100] # Features
y = [14, 38, 54, 76, 95] # Target

alpha, n, theta = 0.01, 5, [1; .5]
J = []
for k in 1:5
    h_th = X[k, :]' * theta
    println("h_th is $(h_th)")
    cost = (y[k] - h_th)^2
    push!(J, cost);
    theta += alpha * (y[k] - h_th) * X[k, :]
    println("theta is $(theta)")
end
```

Assumptions of Linear Regression

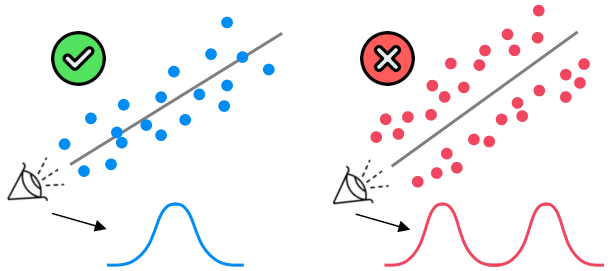
1. Linearity
(Linear relationship between Y and each X)



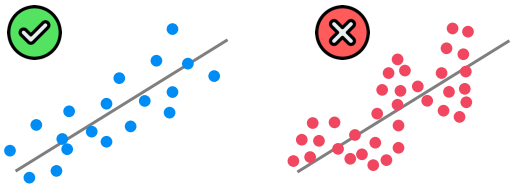
2. Homoscedasticity
(Equal variance)



3. Multivariate Normality
(Normality of error distribution)



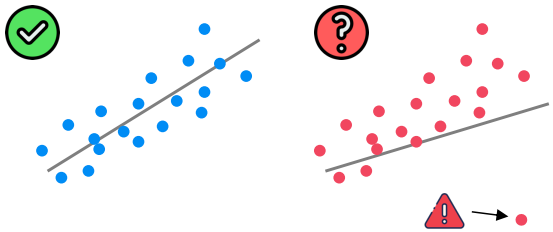
4. Independence
(of observations. Includes “no autocorrelation”)



5. Lack of Multicollinearity
(Predictors are not correlated with each other)

$X_1 \not\sim X_2$ $X_1 \sim X_2$

6. The Outlier Check
(This is not an assumption, but an “extra”)



▶ Source

Evaluation metrics (1/2)

Mean Absolute Error (MAE) measures the average difference of absolute values between predicted and actual targets.

$$\text{MAE} = \frac{1}{n}\sum_{k=1}^{n}|y_k - \hat{y}_k|$$

👍 *A lower **MAE** indicates a better fit of the model to the data.*

Root Mean Squared Error (RMSE) measures the difference between predicted and actual values.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}$$

👍 *A lower **RMSE** indicates a better fit of the model to the data.*

Evaluation metrics (2/2)

R-squared is a statistical measure that quantifies the proportion of the variance in the dependent variable that is explained by the independent variables in the model.

$$\mathcal{R}^2 = 1 - \frac{SS_{\text{residuals}}}{SS_{\text{total}}} = 1 - \frac{\sum_{k=1}^{n} (y_k - \hat{y}_k)^2}{\sum_{k=1}^{n} (y_k - \bar{y})^2}$$

👍 1 *indicates that the model explains* ***ALL*** *the variance in the dependent variable*

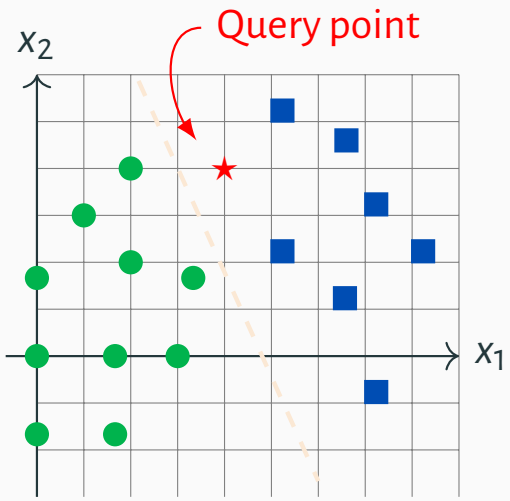
👎 0 *indicates that the model explains* ***NONE*** *of the variance in the dependent variable*

Adjusted R-squared is a modified version of R-squared that accounts for the number of independent variables in the model.

$$\text{Adjusted } \mathcal{R}^2 = 1 - \left(1 - \mathcal{R}^2\right) \frac{n - 1}{n - m - 1}$$

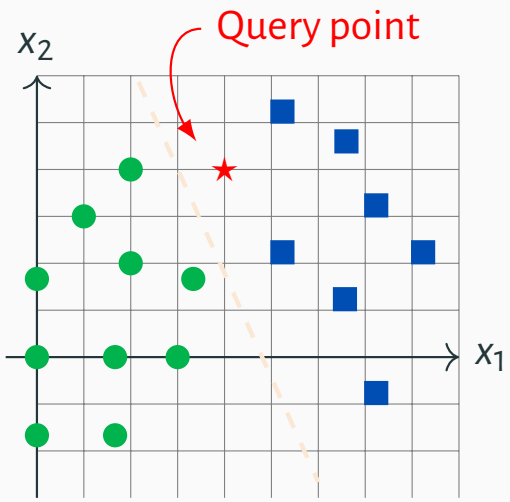
Introduction

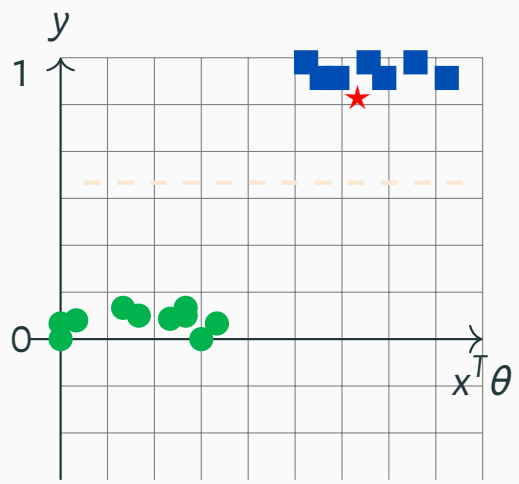
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes* y, so that it can accurately predict the class of new, unseen query points.



Introduction

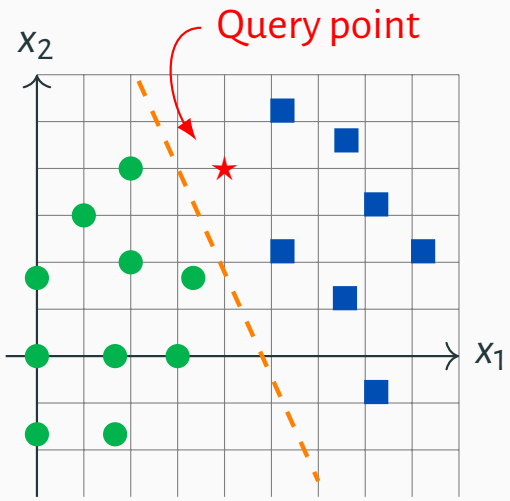
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes* y, so that it can accurately predict the class of new, unseen query points.

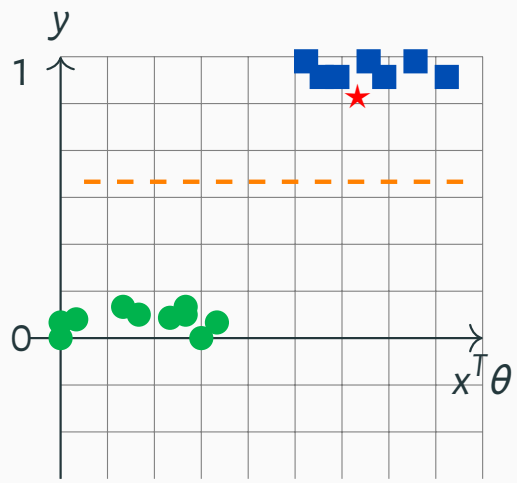




Introduction

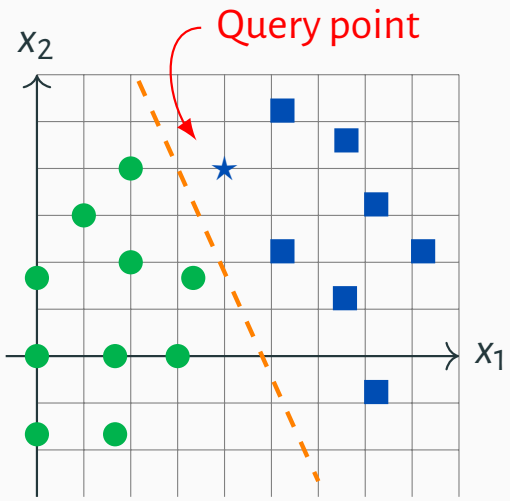
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes* y, so that it can accurately predict the class of new, unseen query points.

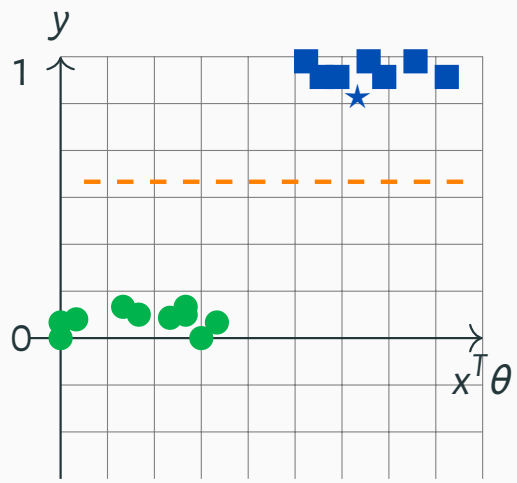




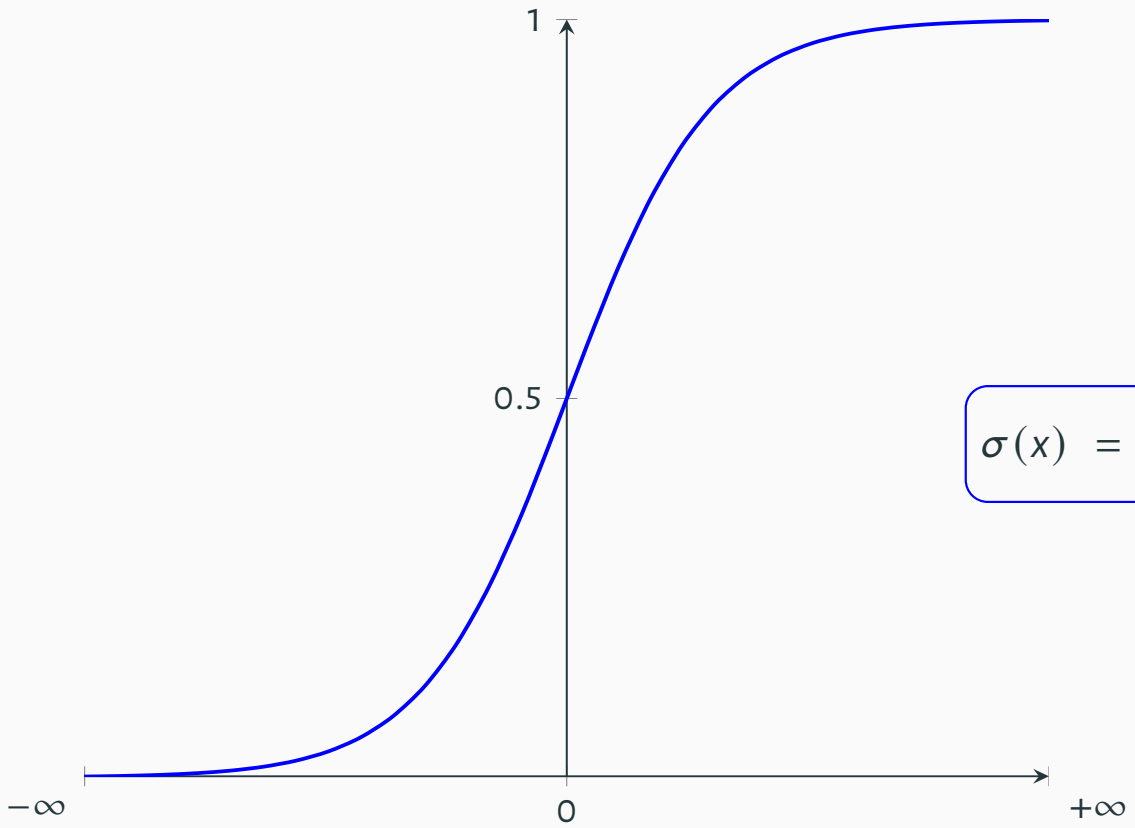
Introduction

Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes* y, so that it can accurately predict the class of new, unseen query points.





Logistic or S-shaped function σ



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- σ squashes range of distance from $]-\infty, +\infty[$ to $[0, 1]$
- σ is differentiable and easy to compute: $\dot{\sigma} = \sigma \times (1 - \sigma)$

Decision boundary

$$y = \sigma\left(\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m\right)$$

$$y = \frac{1}{1 + \mathrm{e}^{-x^T\theta}}$$

Hypothesis

$$h_\theta(x) = P(y = 1|x;\theta) = \frac{1}{1 + \mathrm{e}^{-x^T\theta}}$$

For some given x_k

$$h_\theta(x_k) = P(y = 1|x_k;\theta) = \frac{1}{1 + \mathrm{e}^{-x_k^T\theta}}$$

Cost function

$$\mathcal{J} = \begin{cases} -\ln\left(h_\theta(x)\right) & \text{if} \quad y = 1 \\ -\ln\left(1 - h_\theta(x)\right) & \text{if} \quad y = 0 \end{cases}$$

$$\boxed{\mathcal{J} = -y\ln\left(h_\theta(x)\right) - (1 - y)\ln\left(1 - h_\theta(x)\right)}$$

Gradient Descent

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Generalizing $\mathcal{J}$ yields:

$$\mathcal{J} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k \ln\left(h_\theta(x_k)\right) + (1-y_k)\ln\left(1-h_\theta(x_k)\right)\right)$$

$$\Longrightarrow \quad \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta(x_k)\right)x_{(i,k)}$$

$$\theta_1 \triangleq \theta_1 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(1,k)} \qquad \theta_2 \triangleq \theta_2 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(2,k)}$$

$$\vdots$$

$$\theta_m \triangleq \theta_m + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(m,k)}$$

Gradient Descent

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{Learning Rate}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Generalizing $\mathcal{J}$ with $\mathbf{L_2}$ regularization term yields:

$$\mathcal{J} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k \ln\left(h_\theta(x_k)\right) + (1-y_k)\ln\left(1-h_\theta(x_k)\right)\right) + \frac{\lambda}{2n}\sum_{i=2}^{m}\theta_i^2$$

$$\Longrightarrow \quad \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta(x_k)\right)x_{(i,k)} + \frac{\lambda}{n}\theta_i \quad \text{iff } i \neq 1$$

$$\theta_1 \triangleq \left(1 - \alpha\cancel{\frac{\lambda}{n}}\right)\theta_1 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(1,k)}$$

$$\theta_2 \triangleq \left(1 - \alpha\frac{\lambda}{n}\right)\theta_2 + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(2,k)}$$

$$\vdots$$

$$\theta_m \triangleq \left(1 - \alpha\frac{\lambda}{n}\right)\theta_m + \alpha\frac{1}{n}\sum_{k=1}^{n}\left(y_k - h_\theta\left(x_k\right)\right)x_{(m,k)}$$

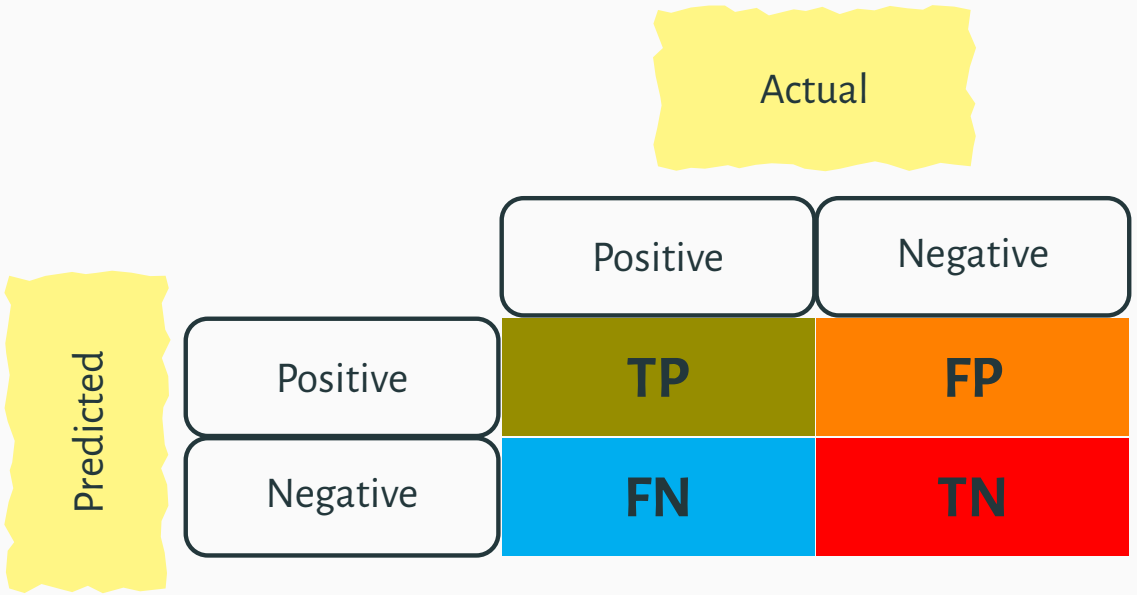
Logistic Regression

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *logistic-regression.jl*

⟶ *Jupyter* ⟶ *logistic-regression.ipynb*

Confusion Matrix



$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$f1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

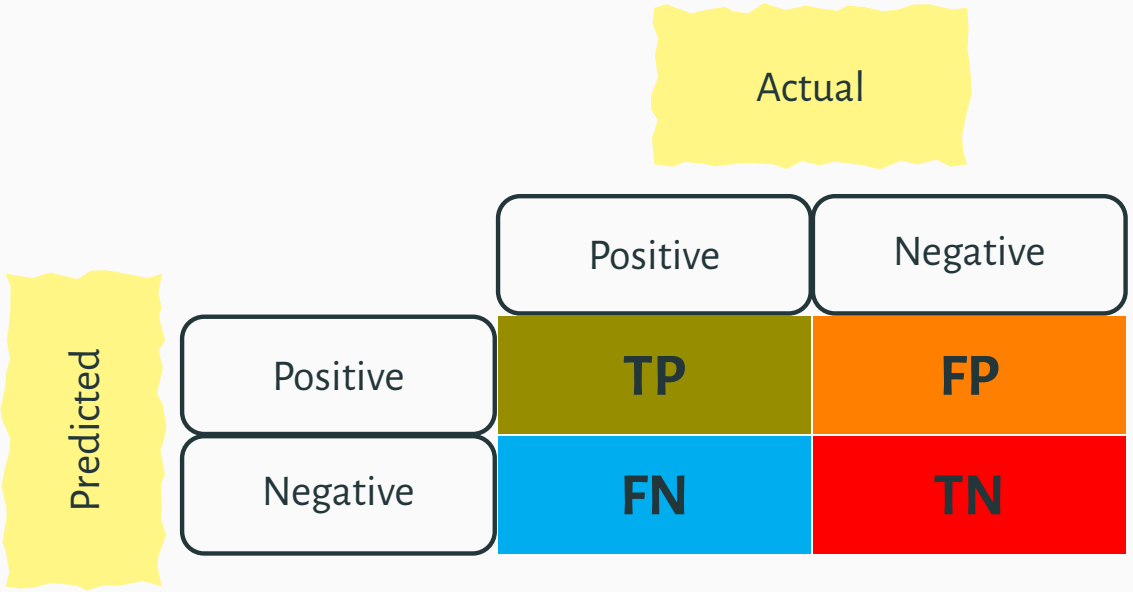
183	141
13	663

$Accuracy = 0.846$
$Precision = 0.565$
$Recall = 0.934$
$f1 - score = 0.704$

320	20
43	538

$Accuracy = 0.932$
$Precision = 0.941$
$Recall = 0.882$
$f1 - score = 0.910$

Confusion Matrix



$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$f1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

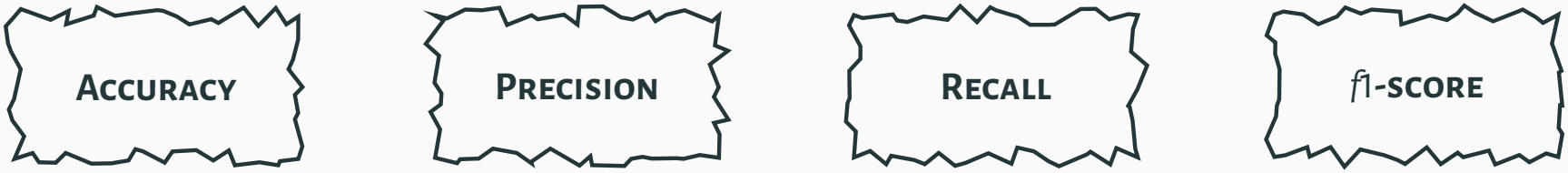
183	141
13	663

$Accuracy = 0.846$
$Precision = 0.565$
$Recall = 0.934$
$f1 - score = 0.704$

320	20
43	538

$Accuracy = 0.932$
$Precision = 0.941$
$Recall = 0.882$
$f1 - score = 0.910$

Evaluation metrics



Accuracy denotes the ratio of how many we got right over all cases:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision designates how many positives do we get right over all positive predictions:

$$Precision = \frac{TP}{TP + FP}$$

Recall is the ratio of how many positives we got right over all actual positive cases:

$$Recall = \frac{TP}{TP + FN}$$

$f1 - score$ denotes the Harmonic Mean of *Precision* & *Recall*:

$$f1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Evaluation Metrics

Follow up

$$f_\beta - \text{score} = \frac{1+\beta^2}{\frac{1}{Precision} + \frac{\beta^2}{Recall}}$$

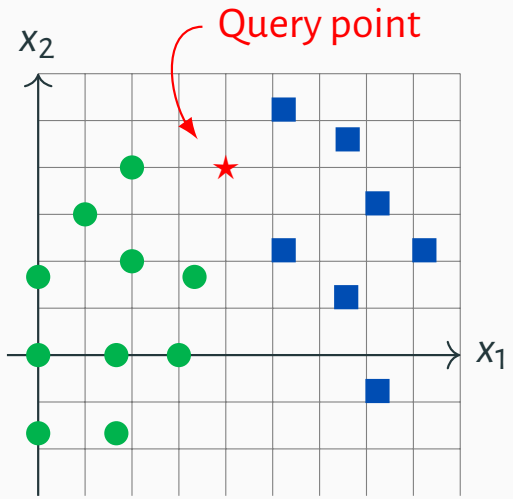
Case #1: Prioritize *Precision* over *Recall*, *e.g.*, $\beta = 0.5$

- Mail spam detection
- Predicting appropriate day to launch a satellite

Case #2: Prioritize *Recall* over *Precision*, *e.g.*, $\beta = 2$

- Detection of life threatening diseases like cancer
- Fraud detection

k-Nearest Neighbors (1/6)



Minkowski distance

$$d(x; y) = \left(\sum_{i=1}^{n} |y_i - x_i|^p \right)^{1/p}$$

Manhattan distance (p=1)

$$d(x; y) = \sum_{i=1}^{n} |y_i - x_i|$$

Euclidean distance (p=2)

$$d(x; y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

k-Nearest Neighbors (2/6)

- Evelyn Fix and Joseph Hodges, 1951
- Thomas Cover, 1966

Algorithm 1 Summary Construction

1: **procedure** How does k-NN work? (Finding Nearest Neighbors)
Input: A query point;
Output: Assign a class label to that point.
2: Define how many neighbors will be checked to classify the specific query point;
3: Compute the distance $d(x;\ y)$ of the query point to other data points;
4: Count the number of the data points in each category;
5: Assign the query point to the class with most frequent neighbors.
6: **end procedure**

k-Nearest Neighbors (3/6)

Task #3
Let be the following coordinate points:

$$A(1,\ 6);\ B(2,\ 6);\ C(3,\ 1);\ D(4,\ 2);\ E(6,\ 0);\ F(7,\ 5);\ G(7,\ 3);\ H(10,\ 3);\ I(-4,\ -1)$$

Using the Euclidean distance, what are the two closest neighbors of point $P(5,\ 5)$?

$$d(A;\ P) = \sqrt{17} \approx 4.12 \qquad d(B;\ P) = \sqrt{10} \approx 3.16 \qquad d(C;\ P) = \sqrt{20} \approx 4.47$$

$$d(D;\ P) = \sqrt{10} \approx 3.16 \qquad d(E;\ P) = \sqrt{26} \approx 5.1 \qquad d(F;\ P) = \sqrt{4} = 2$$

$$d(G;\ P) = \sqrt{8} \approx 2.83 \qquad d(H;\ P) = \sqrt{29} \approx 5.38 \qquad d(I;\ P) = \sqrt{117} \approx 10.82$$

```
function dds(a, b) # `a` and `b` are coordinates of some point
    d_squared = (a-5)^2+(b-5)^2
    (d_squared, sqrt(d_squared))
end

dds(1, 6) # Point `A`
dds(2, 6) # Point `B`
```

k-Nearest Neighbors (4/6)

Task #4[2]

We try to predict the color of a fruit according to its width (w) and height (h). The following training data is available:

Fruit	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
w	2	5	2	6	1	4	2	6
h	6	6	5	5	2	2	1	1
Color	Red	Yellow	Orange	Purple	Red	Blue	Violet	Green

The goal here is to study the influence of neighbors on the color property of a fruit. Let U be the new fruit of width $w = 1$ and height $h = 4$

1. What is its color if we consider 1 neighbor?
2. What is its color if we consider 3 neighbors?
3. Rather than majority voting, we would like to consider the vote of neighbors weighted by the distance. Each neighbor votes according to a weight inversely proportional to the square of its distance: $\frac{1}{d^2}$. We take 3 neighbors, what is the color of U? Compare your results to those in question 2.

k-Nearest Neighbors (5/6)

$d(U;\ F_1) = \sqrt{5} \approx 2.24$ $d(U;\ F_2) = \sqrt{20} \approx 4.47$ $d(U;\ F_3) = \sqrt{2} \approx 1.41$

$d(U;\ F_4) = \sqrt{26} \approx 5.1$ $d(U;\ F_5) = \sqrt{4} = 2$ $d(U;\ F_6) = \sqrt{13} \approx 3.6$

$d(U;\ F_7) = \sqrt{10} \approx 3.16$ $d(U;\ F_8) = \sqrt{34} \approx 5.83$

1. Color of U is Orange because $d(U;\ F_3)$ is the smallest.
2. Color of U is Red: F_1 and F_5 *(+2 to Red class)*, F_3 *(+1 to Orange class)*
3. Color of U is Orange

$$\mathcal{S}(\text{Red}) = \frac{1}{d^2(U;\ F_1)} + \frac{1}{d^2(U;\ F_5)} = 0.45 \qquad \mathcal{S}(\text{Orange}) = \frac{1}{d^2(U;\ F_3)} = 0.5$$

k-Nearest Neighbors (6/6)

```
function dds(w, h) # `w` and `h` are width and height of some fruit
    d_squared = (w-1)^2+(h-4)^2
    (d_squared, sqrt(d_squared))
end

dds(2, 6) # Fruit `F_1`
dds(5, 6) # Fruit `F_2`
```

[2] From Prof. Winston's book

k-NN

julia

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *knn.jl*

Pluto.jl

⟶ *Jupyter* ⟶ *knn.ipynb*

Rule of Thumb to Choose k

k **is even** if the number of classes is odd

k **is odd** if the number of classes is even

k is an important hyperparameter that can affect the performance of the model.

1. Larger values of k will result in a smoother decision boundary, which can lead to a more generalized model.
2. Smaller values of k will result in a more complex decision boundary, which can lead to a model that is more prone to overfitting.
3. The optimal value of k may depend on the specific dataset and the characteristics of the data.

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are supervised learning algorithms for classification and regression. They identify the optimal hyperplane to separate data into distinct classes.

Hyperplane A decision boundary separating classes in feature space.

Support Vectors Closest data points to the hyperplane, determining its position and orientation.

Margin Distance between the hyperplane and the nearest data points. SVM maximizes this margin.

For a dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$:

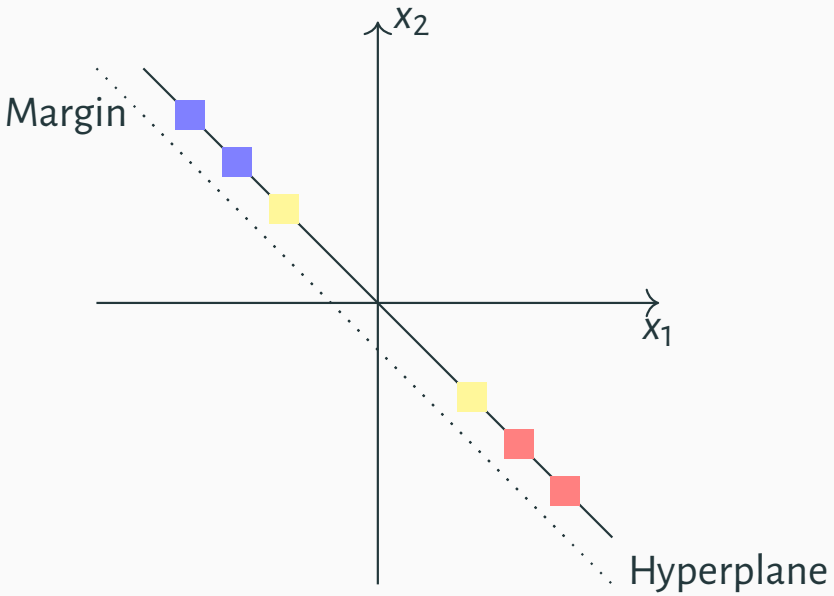
- Find a hyperplane $w^T x + b = 0$ that maximizes the margin.
- Constrained optimization:

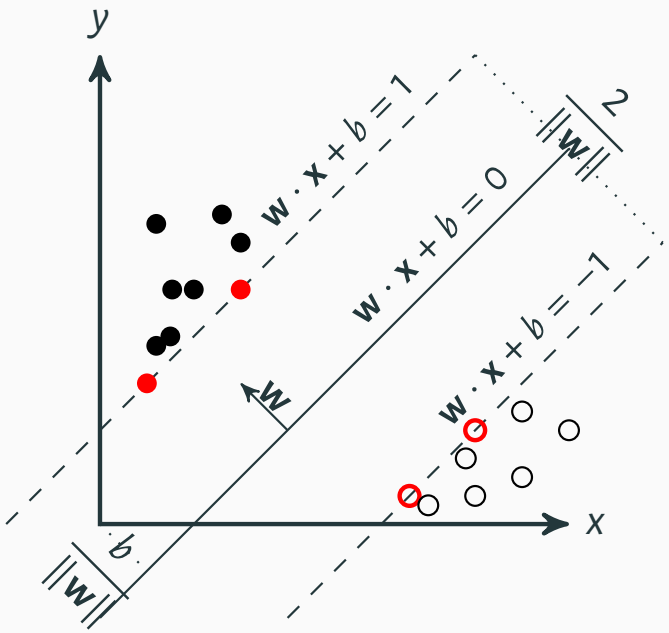
$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i$$

Types of SVMs

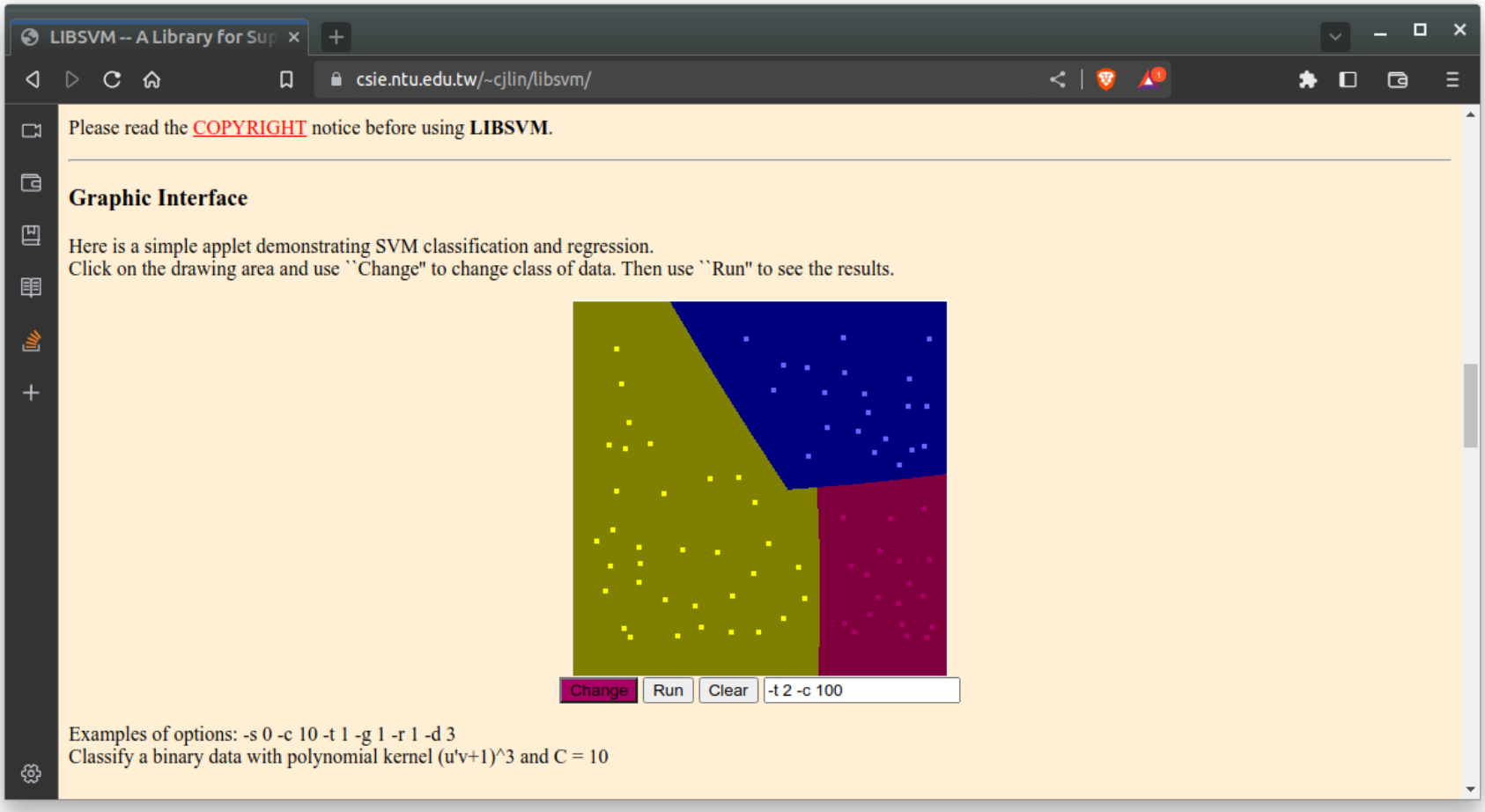
- **Linear SVM:** For linearly separable data.
- **Kernel SVM:** Maps data to higher dimensions for non-linear separation.

Visualization of SVM (1/2)





Visualization of SVM (2/2)



https://www.csie.ntu.edu.tw/~cjlin/libsvm/

SVM: Loss Function

Hinge Loss Function: The hinge loss is used in SVMs to penalize misclassifications and maximize the margin:

$$L(y, f(x)) = \max(0, 1 - yf(x)),$$

where:

- $y \in \{-1, +1\}$ is the true label,
- $f(x) = w^T x + b$ is the predicted score.

The loss is zero if $yf(x) \geq 1$ (correct classification with margin), and increases linearly otherwise.

Why SVMs are Powerful

- **Robustness:** Focuses on support vectors, making it less sensitive to outliers.
- **High-Dimensional Data:** Performs well even with a large number of features.
- **Kernel Trick:** Enables non-linear classification by mapping data to higher dimensions.

Task #5

You are given a dataset with two classes: **Class A** ($y = +1$) and **Class B** ($y = -1$). The features (x_1, x_2) and corresponding labels are:

Data Point	1	2	3	4	5	6
x_1	2	3	1	6	7	8
x_2	2	4	1	2	3	2
y	+1	+1	+1	-1	-1	-1

1. Plot the data points on a 2D plane.
2. Determine the optimal hyperplane separating the two classes using SVM.
3. Calculate the margin width if the weight vector is $\mathbf{w} = [1, 1]^T$.
4. Identify the support vectors.

Hint!

- The margin width is given by:

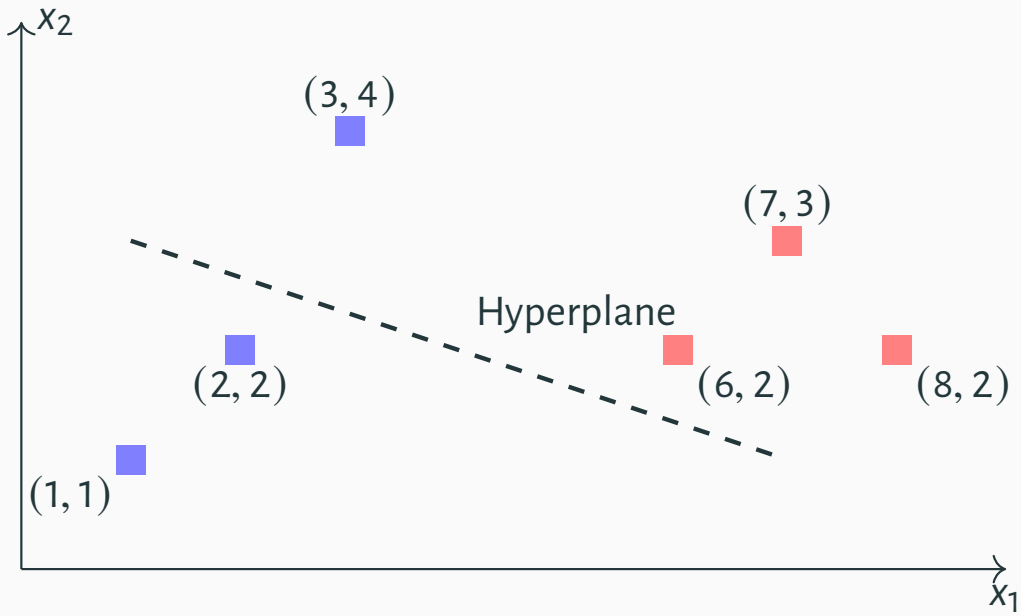
$$\text{Margin Width} = \frac{2}{\|\mathbf{w}\|}$$

- Support vectors are the data points closest to the hyperplane.

Solution (1/2)

Q. #1: Plot the Data Points

The data points are visualized on the plane (x_1, x_2):



Solution (2/2)

Q. #2: Optimal Hyperplane

The separating hyperplane is given by:

$$w_1 x_1 + w_2 x_2 + b = 0 \quad \text{where} \quad \mathbf{w} = [1, 1]^T,\ b = -4$$

Thus, the equation becomes:

$$x_1 + x_2 = 4$$

Q. #3: Margin Width

The margin width is calculated as:

$$\text{Margin Width} = \frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{1^2 + 1^2}} = \sqrt{2}$$

Q. #4: Support Vectors

The support vectors are the points closest to the hyperplane:

$$(1, 1) \text{ from Class A}, \quad (7, 3) \text{ from Class B}.$$

SVM for Classification

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *svc.jl*

⟶ *Jupyter* ⟶ *svc.ipynb*

Naive Bayes: Introduction

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem. It is commonly used for classification tasks, such as spam detection and text classification.

Bayes' Theorem:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

where:

- $P(Y|X)$: Posterior probability of class Y given features X.
- $P(X|Y)$: Likelihood of features X given class Y.
- $P(Y)$: Prior probability of class Y.
- $P(X)$: Marginal probability of features X.

Naive Bayes: Key Assumptions and Types

Key Assumptions

Independence: Features are assumed to be conditionally independent given the class.

Equal Importance: All features contribute equally to the prediction.

Types

Gaussian Naive Bayes: Assumes features follow a normal distribution.

Multinomial Naive Bayes: Used for discrete data (e.g., word counts).

Bernoulli Naive Bayes: Used for binary features.

- ▲ Simple and fast to train.
- ▲ Works well with high-dimensional data (e.g., text).
- ▲ Requires less training data compared to other algorithms.

- ▼ Assumes feature independence, which is rarely true in real-world data.
- ▼ Struggles with zero probabilities (requires smoothing techniques).
- ▼ Less accurate for complex relationships between features.

Exercise: Naive Bayes for Spam Detection (1/5)

Features: Words in an email (e.g., “free”, “offer”, “money”).

Classes: “Spam” or “Not Spam”.

Steps:

1. Calculate $P(\text{Spam})$ and $P(\text{Not Spam})$ from the training data.
2. Calculate $P(\text{Word}|\text{Spam})$ and $P(\text{Word}|\text{Not Spam})$ for each word.
3. Use Bayes’ Theorem to predict the class of a new email.

Exercise: Naive Bayes for Spam Detection (2/5)

Task #6
You are given a dataset of emails with the following word frequencies and labels:

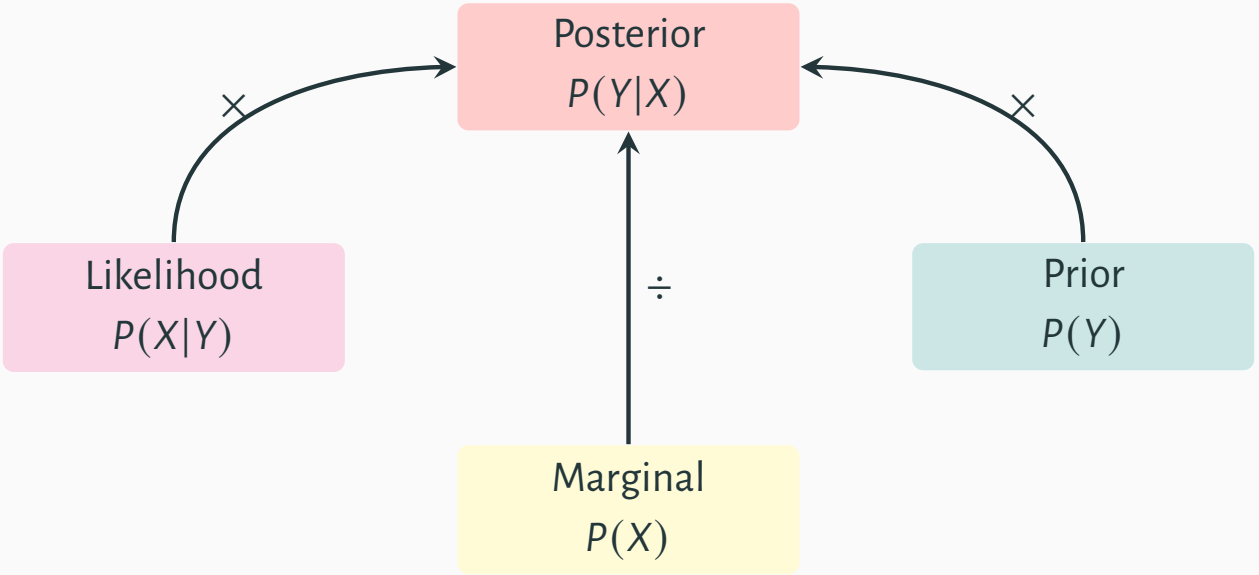
Email	"free"	"money"	Label
1	1	0	Spam
2	1	1	Spam
3	0	0	Not Spam
4	0	1	Not Spam

1. Calculate the prior probabilities $P(\text{Spam})$ and $P(\text{Not Spam})$.
2. Calculate the likelihoods $P(\text{"free"}|\text{Spam})$, $P(\text{"money"}|\text{Spam})$, $P(\text{"free"}|\text{Not Spam})$, and $P(\text{"money"}|\text{Not Spam})$.
3. Use Naive Bayes to classify a new email with the words "free" and "money".

Hint!

- Use Laplace smoothing (add +1 smoothing) to handle zero probabilities.

Exercise: Naive Bayes for Spam Detection (3/5)



Step 1: Calculate Prior Probabilities

- $P(\text{Spam}) = \dfrac{\text{Number of Spam Emails}}{\text{Total Emails}} = \dfrac{2}{4} = 0.5$
- $P(\text{Not Spam}) = \dfrac{\text{Number of Not Spam Emails}}{\text{Total Emails}} = \dfrac{2}{4} = 0.5$

Exercise: Naive Bayes for Spam Detection (4/5)

Step 2: Calculate Likelihoods with Laplace Smoothing

- $P(\text{“free”}|\text{Spam}) = \dfrac{\text{Count of “free” in Spam} + 1}{\text{Total Words in Spam} + \text{Unique Words}} = \dfrac{1+1}{2+2} = \dfrac{2}{4} = 0.5$
- $P(\text{“money”}|\text{Spam}) = \dfrac{\text{Count of “money” in Spam} + 1}{\text{Total Words in Spam} + \text{Unique Words}} = \dfrac{1+1}{2+2} = \dfrac{2}{4} = 0.5$
- $P(\text{“free”}|\text{Not Spam}) = \dfrac{\text{Count of “free” in Not Spam} + 1}{\text{Total Words in Not Spam} + \text{Unique Words}} = \dfrac{0+1}{2+2} = \dfrac{1}{4} = 0.25$
- $P(\text{“money”}|\text{Not Spam}) = \dfrac{\text{Count of “money” in Not Spam} + 1}{\text{Total Words in Not Spam} + \text{Unique Words}} = \dfrac{1+1}{2+2} = \dfrac{2}{4} = 0.5$

Exercise: Naive Bayes for Spam Detection (5/5)

Step 3: Classify a New Email

- For the email with words "free" and "money":

$$P(\text{Spam}|\text{"free"}, \text{"money"}) \propto P(\text{"free"}|\text{Spam}) \cdot P(\text{"money"}|\text{Spam}) \cdot P(\text{Spam})$$
$$= 0.5 \cdot 0.5 \cdot 0.5 = 0.125$$

$$P(\text{Not Spam}|\text{"free"}, \text{"money"}) \propto P(\text{"free"}|\text{Not Spam}) \cdot P(\text{"money"}|\text{Not Spam}) \cdot P(\text{Not Spam})$$
$$= 0.25 \cdot 0.5 \cdot 0.5 = 0.0625$$

- Since $0.125 > 0.0625$, the email is classified as **Spam**.

Decision Trees: Definition and Key Concepts (1/2)

A decision tree is a supervised learning algorithm used for classification and regression. It partitions the feature space into regions by applying a series of decision rules.

Components

Root Node: Represents the entire dataset.

Decision Nodes: Split the dataset based on a feature and a threshold.

Leaf Nodes: Represent the final output (class label or regression value).

Decision Trees: Definition and Key Concepts (2/2)

Loss Functions for Classification

- **Gini Index:** Measures impurity in a node.

$$G = 1 - \sum_{i=1}^{k} p_i^2$$

- **Entropy:** Measures uncertainty in a node.

$$H = -\sum_{i=1}^{k} p_i \log(p_i)$$

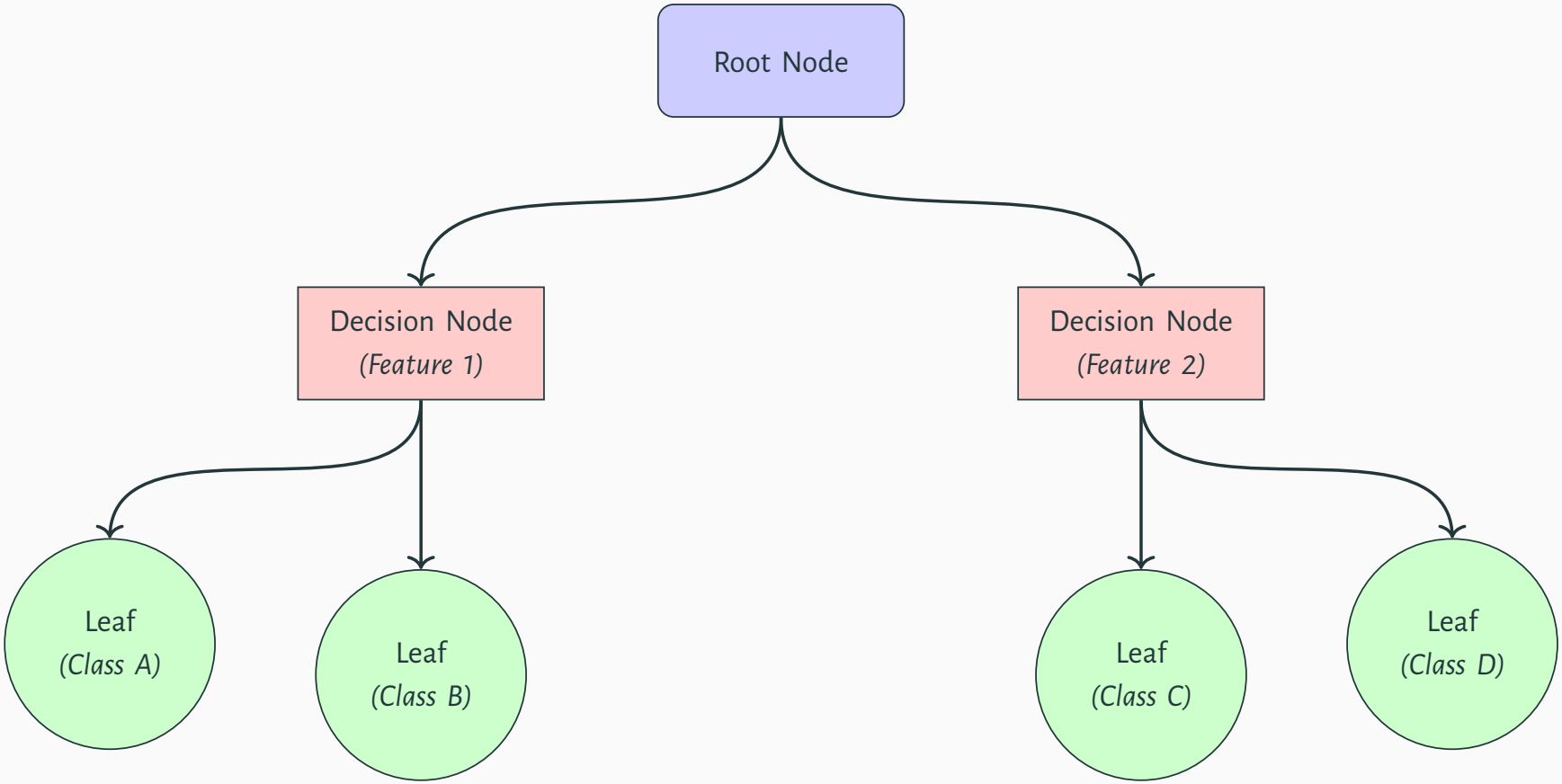
Key Concepts

Splitting Criteria: Choose the feature and threshold that maximize information gain or minimize impurity.

Pruning: Remove unnecessary branches to prevent overfitting.

Feature Importance: Rank features based on their contribution to the model.

Decision Tree Visualization (1/1)



Random Forests: Definition and Ensemble Approach (1/2)

A random forest is an ensemble method that combines multiple decision trees to improve classification accuracy. It uses bagging (bootstrap aggregation) and random feature selection.

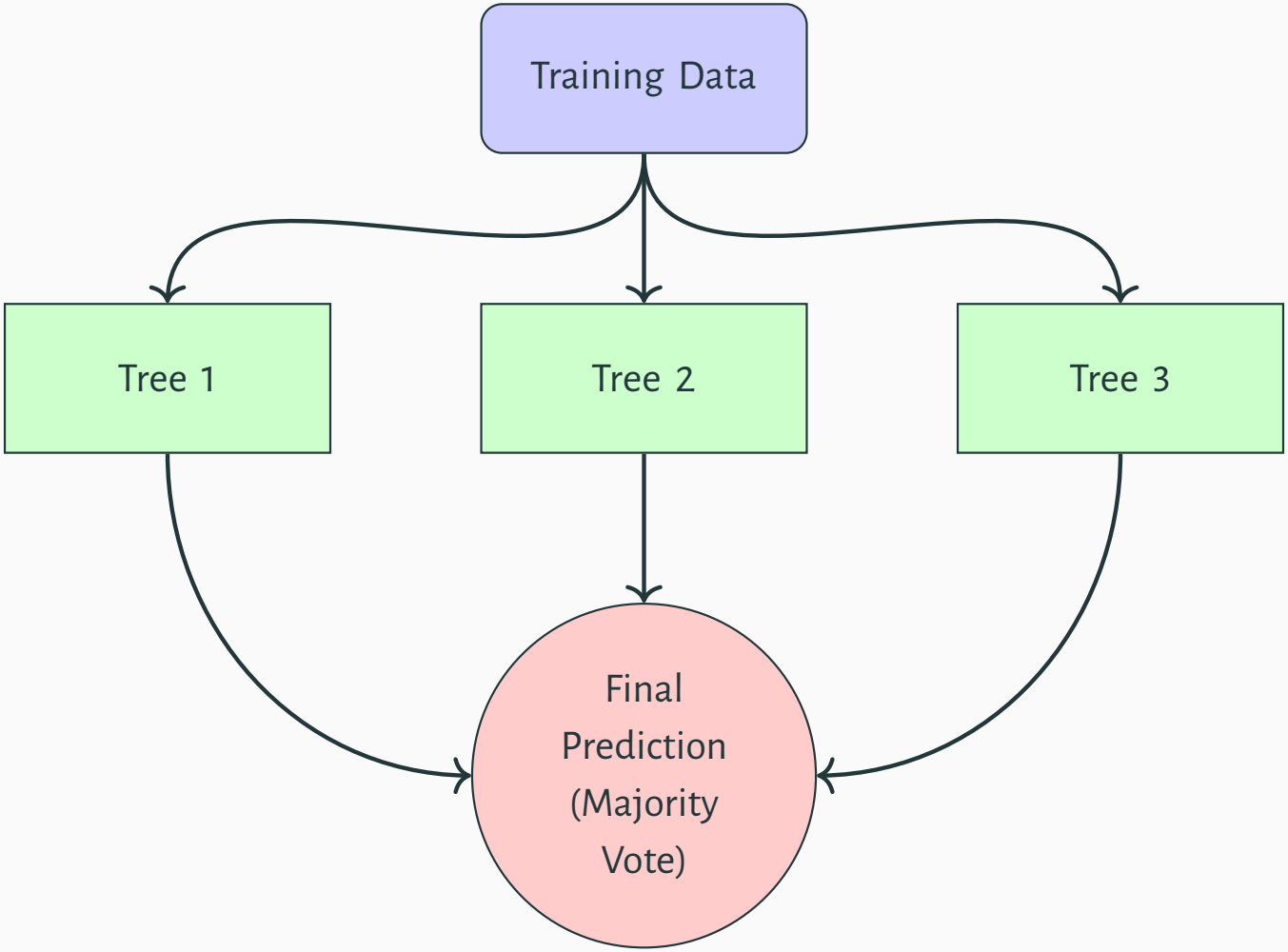
Key Features

- **Diversity:** Trees are trained on different subsets of data and features.
- **Voting Mechanism:** For classification, the final output is the mode of the tree predictions.

Loss Function

- Combines the loss functions of individual trees, e.g., Gini Index or Entropy, during splits.

Random Forests: Definition and Ensemble Approach (2/2)



Loss Functions: Gini Index vs Entropy

Gini Index

- Measures the impurity of a node.
- Lower values indicate purer nodes.

$$G = 1 - \sum_{i=1}^{k} p_i^2$$

Entropy

- Measures the uncertainty of a node.
- Lower values indicate more certainty.

$$H = -\sum_{i=1}^{k} p_i \log(p_i)$$

Comparison

- Both are used for splitting criteria in decision trees.
- Gini Index is faster to compute; Entropy provides more precise splits in some cases.

Numerical Example (1/5)

Task #7

A company wants to classify customers into two categories: **Potential Buyers (P)** and **Non-Buyers (N)** based on their purchasing history.

1. Construct a **decision tree** using the following sample data (assume a depth of 2):

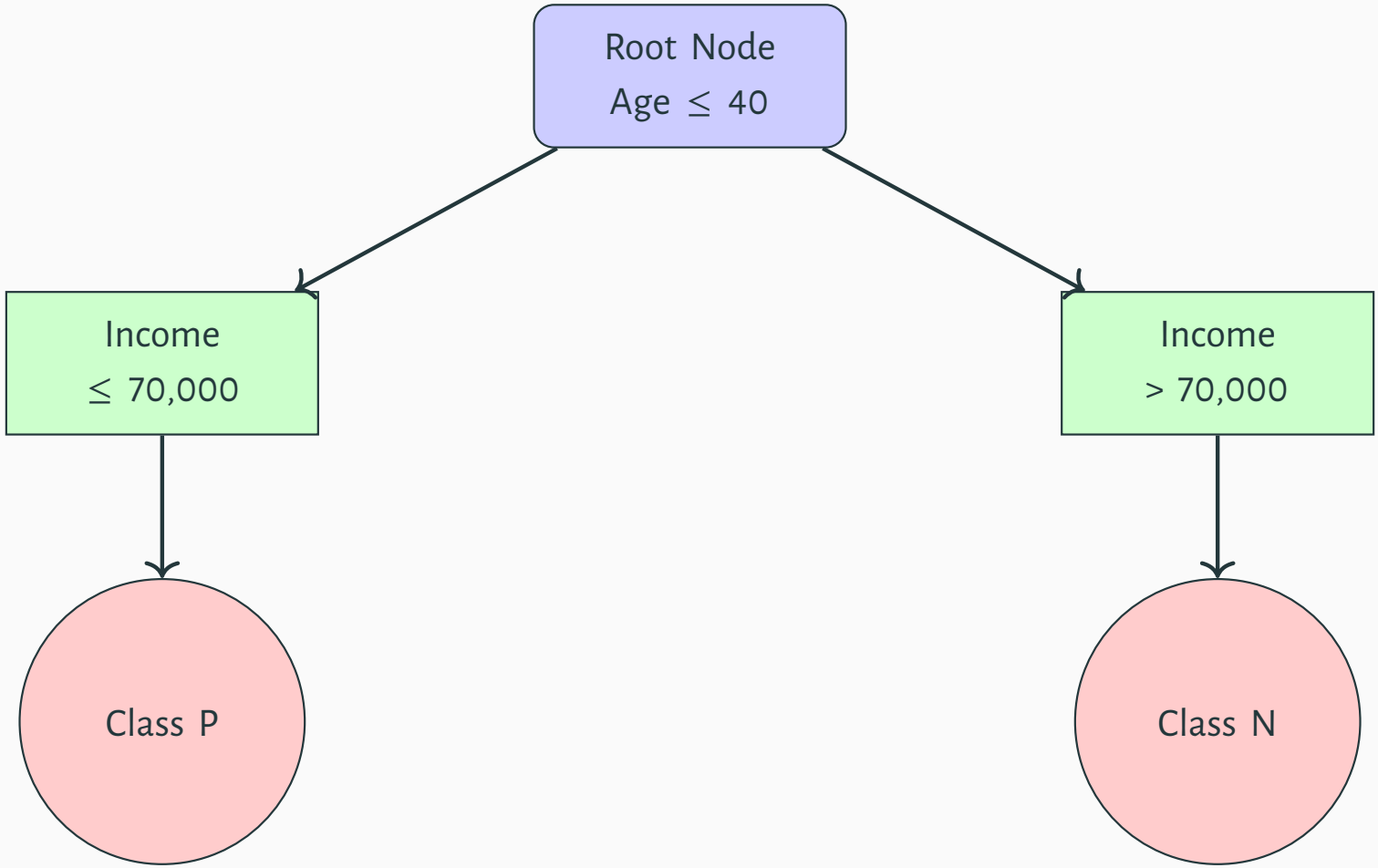
Age	Income	Previous Purchases	Class
25	50.000	*Yes*	*P*
45	80.000	*No*	*N*
30	60.000	*Yes*	*P*
50	90.000	*No*	*N*

   Use **Gini Index** to determine the splits.

2. Explain how a **random forest** could improve the classification accuracy compared to a single decision tree.
3. Identify potential advantages and disadvantages of using random forests for this dataset.

Numerical Example (2/5)

Q. #1: Construct a Decision Tree Using Gini Index



Numerical Example (3/5)

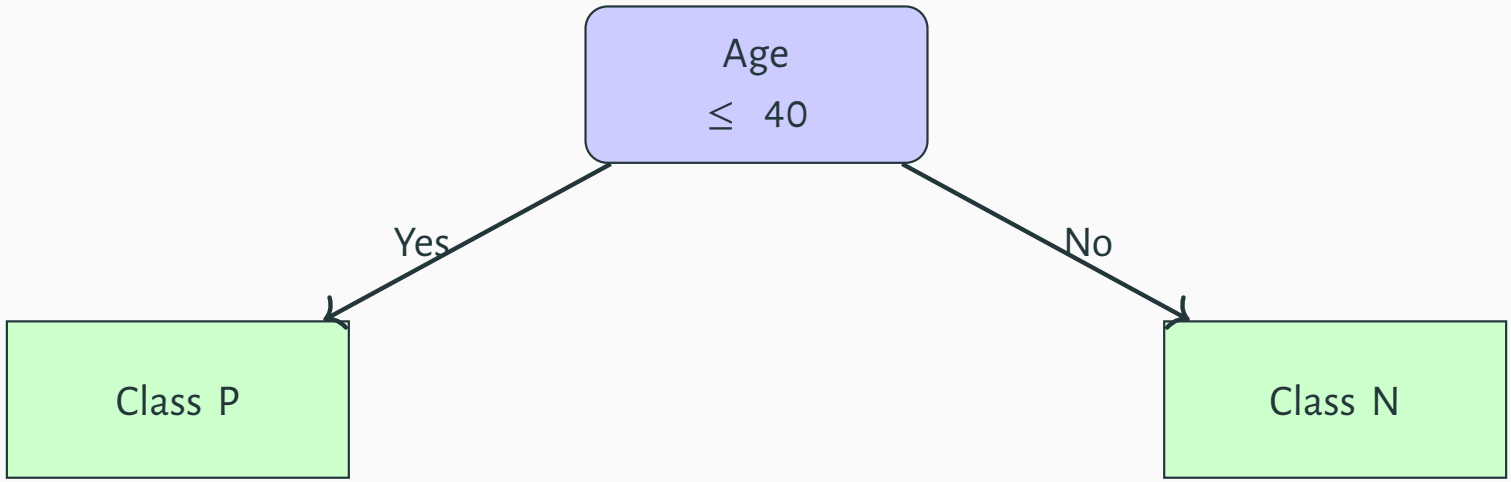
Step 1: Calculate Gini Index for Splits

- **Root Node:** All 4 samples (2 P, 2 N) Gini $= 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$
- **Split by Age $\leq$ 40:**
  - Left (Age $\leq$ 40): 2 samples (2 P, 0 N), Gini $= 1 - 1^2 - 0^2 = 0$
  - Right (Age $>$ 40): 2 samples (0 P, 2 N), Gini $= 1 - 0^2 - 1^2 = 0$

Weighted Gini $= \frac{2}{4}(0) + \frac{2}{4}(0) = 0$

Numerical Example (4/5)

Step 2: Optimal Split The best split is on **Age** $\leq$ 40, as it results in pure nodes.



Numerical Example (5/5)

Q. #2: Random Forest

- Trains multiple decision trees on bootstrapped subsets of the data.
- Randomly selects features at each split to reduce correlation between trees.
- Final prediction is determined by majority voting.

Q. #3: Advantages and Disadvantages of Random Forests

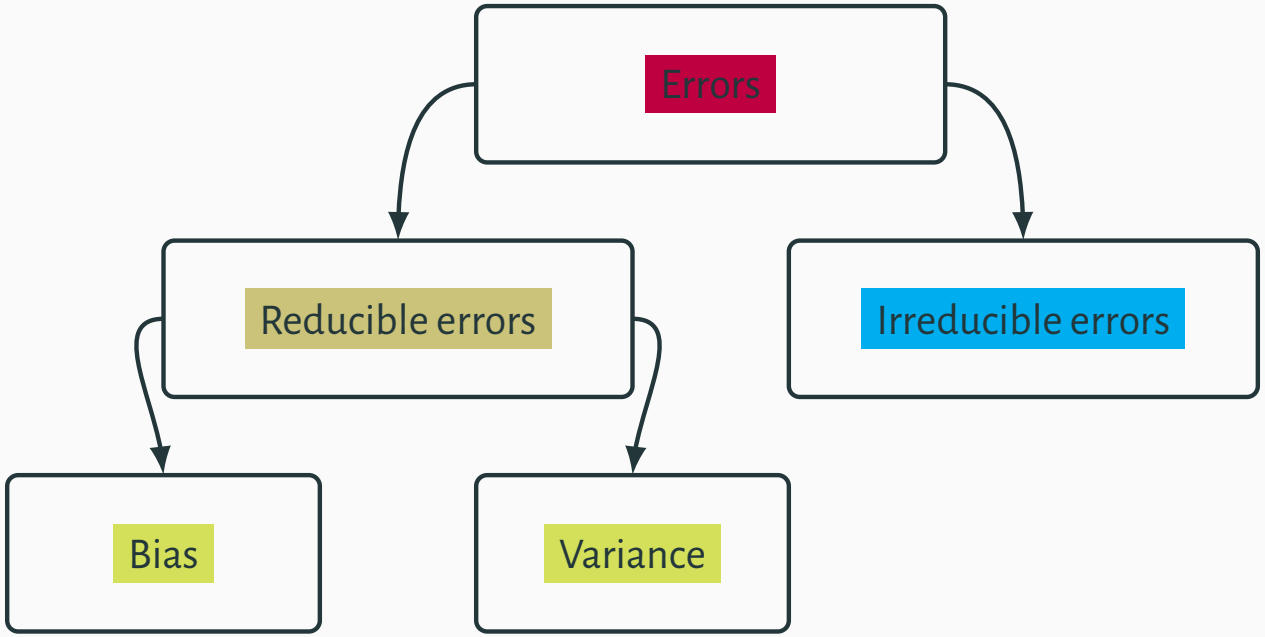
- ▲ Reduces overfitting compared to a single decision tree.
- ▲ Handles high-dimensional datasets effectively.
- ▼ Less interpretable than a single decision tree.
- ▼ Computationally more expensive.

Summary

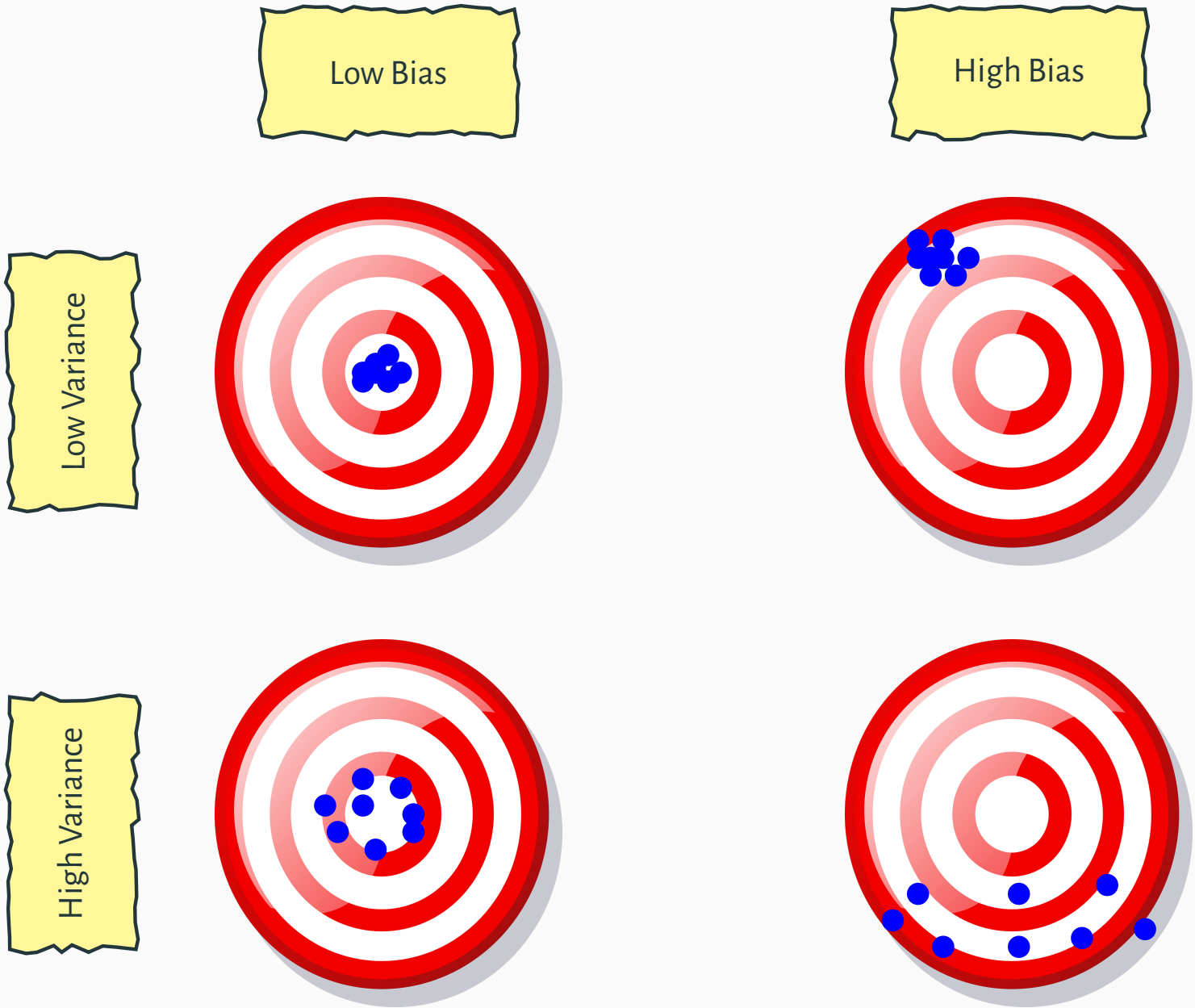
Method		Pros		Cons
Logistic Regression	▲	Probabilistic	▼	Almost linearly separable data
k-NN	▲	Fast and efficient	▼	# of neighbors k
			▼	Detecting outliers[3]
SVM	▲	Memory efficient	▼	Kernel's choice
	▲	Versatile	▼	Large datasets
	▲	Noise and outliers	▼	Overlapping classes
	▲	High dimension	▼	Interpretability
Naive Bayes	▲	Simplicity and efficiency	▼	Independence between features
	▲	High dimension	▼	$\exists$ of irrelevant features
Decision Tree	▲	Interpretability	▼	Overfitting
	▲	Numerical and categorical data	▼	Unstable
	▲	Robust to outliers	▼	Continuous variables
	▲	High accuracy	▼	# of input features
Random Forest	▲	Less prone to overfitting	▼	Computation
	▲	High dimension	▼	Interpretability

[3] Points that differ significantly from the rest of the data points.

Errors in ML


Errors
Reducible errors
Irreducible errors
Bias
Variance

Bias-variance tradeoff



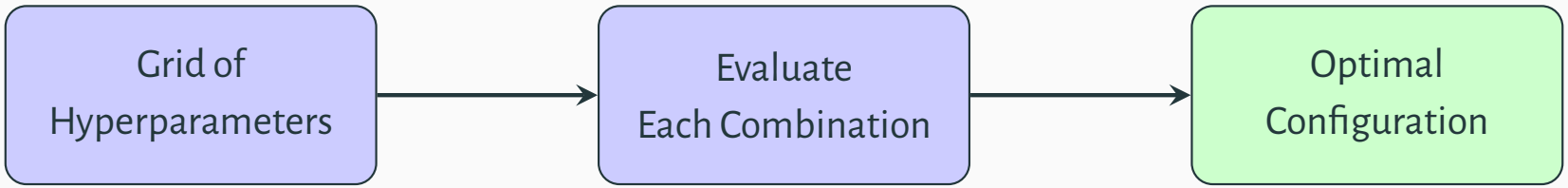
Grid Search: Optimizing Hyperparameters

Grid Search is a systematic approach to finding the best hyperparameter combination for a machine learning model by evaluating all possible configurations.

Key Concepts

- Defines a **grid** of hyperparameter values.
- Evaluates all combinations using a performance metric (e.g., accuracy, precision).
- Identifies the **optimal configuration** for the model.

Process



Challenges

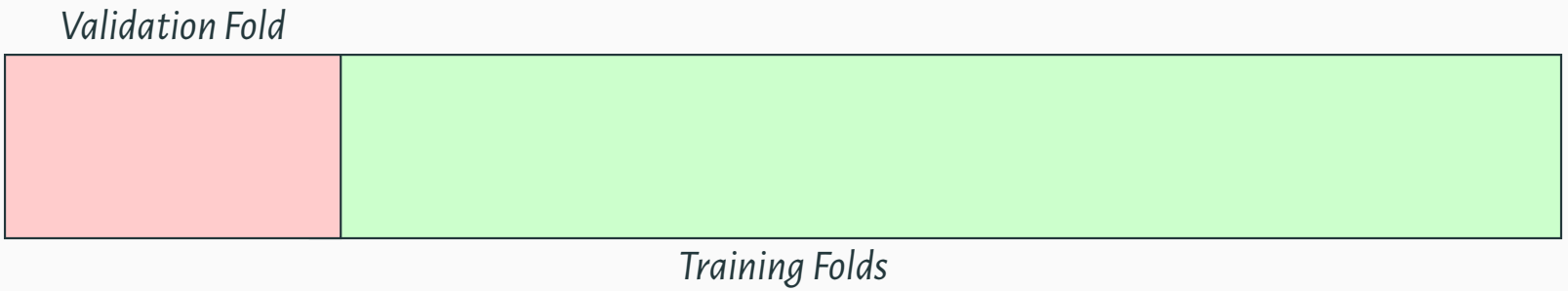
Computationally Expensive: Large grids require significant resources.

Alternative: Randomized Search samples combinations to reduce computation.

Cross-Validation: Ensuring Robustness

Cross-validation is a resampling technique used to evaluate model performance by dividing the dataset into multiple subsets (folds) and validating the model on each subset.

1. Split the dataset into k subsets (folds).
2. Train the model on $k - 1$ folds and validate on the remaining fold.
3. Repeat k times, using a different fold for validation each time.
4. Compute the average performance metric across all folds.



Benefits

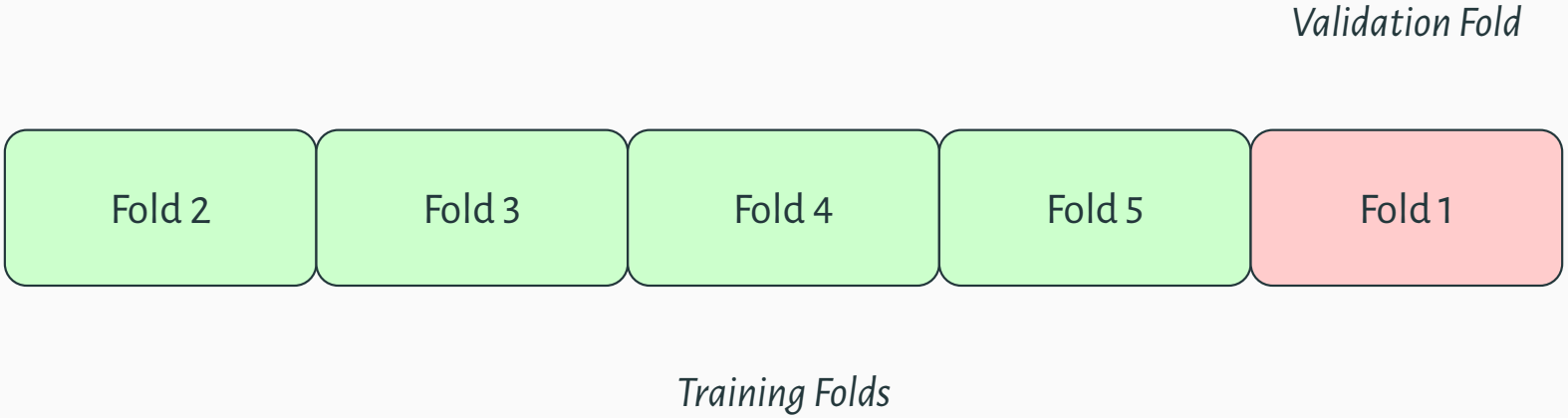
- Reduces overfitting by validating on multiple subsets.
- Provides a reliable estimate of model performance.

K-Folds Cross-Validation: Details

K-Folds Cross-Validation divides the dataset into k equal parts (folds) and performs the training-validation cycle k times, ensuring robust model evaluation.

1. Shuffle the dataset randomly.
2. Split into k equal parts (folds).
3. For each fold:
   - Use the current fold as validation data.
   - Use the remaining folds as training data.
4. Average the performance metric across all folds.

Example: 5-Fold Cross-Validation

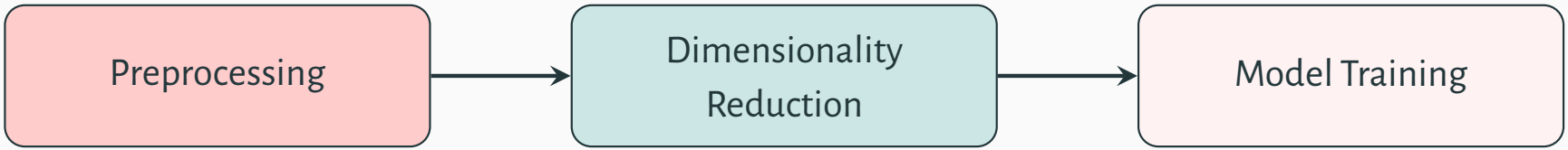


Pipeline Mechanism: Automating Workflow

A pipeline automates the sequential application of preprocessing steps and model training, ensuring a consistent and efficient workflow.

Steps in a Pipeline

1. Preprocess data (e.g., scaling, normalization).
2. Apply dimensionality reduction or feature selection.
3. Train a machine learning model.



Advantages

- Ensures consistency across training and testing data.
- Simplifies hyperparameter optimization.

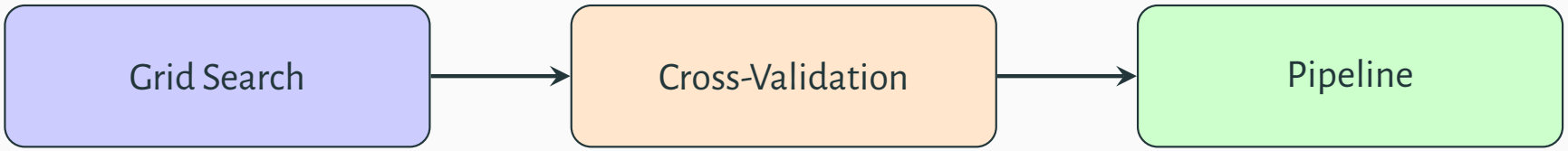
Integrating Grid Search, Cross-Validation, and Pipelines

Overview

Grid Search: Finds the best hyperparameters.

Cross-Validation: Ensures robust performance evaluation.

Pipeline: Automates preprocessing and training.



End-to-End Workflow

1. Define preprocessing steps in a pipeline.
2. Use Grid Search to optimize hyperparameters.
3. Validate results using K-Folds Cross-Validation.

Unsupervised Learning

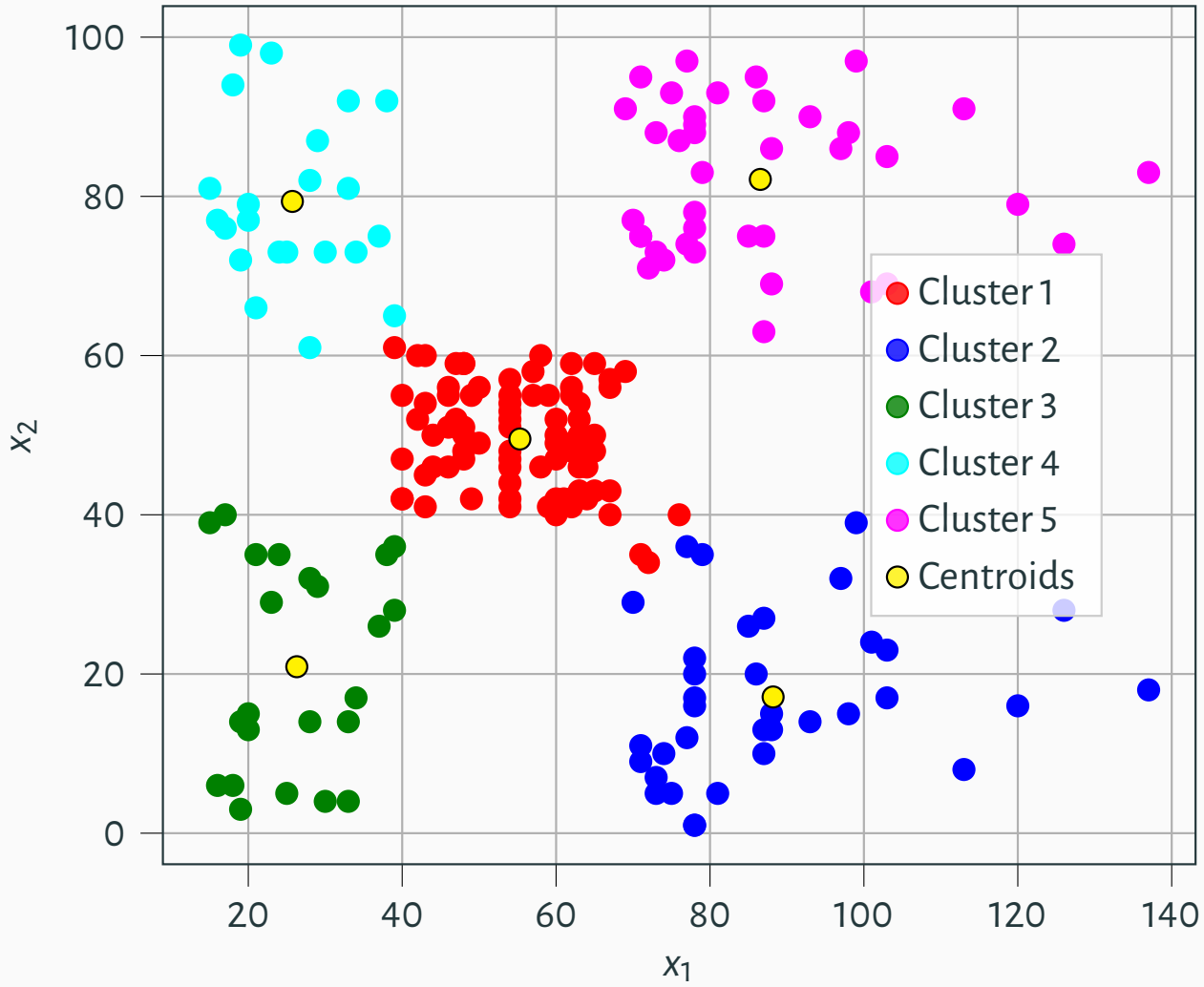
K-Means Clustering (1/3)

The algorithm *K*-**Means** allows to display regularities or patterns in unlabeled data.

- The term 'means' refers to averaging the data when computing each centroid;
- A centroid is the arithmetic mean of all the data points belonging to a particular cluster.

This technique identifies a certain number of centroids within a data set. The algorithm then allocates every data point to the nearest cluster as it attempts to keep the clusters as small as possible. At the same time, *K*-Means attempts to keep the other clusters as different as possible.

K-Means Clustering (2/3)



K-Means Clustering (3/3)

Algorithm 2 Summary Construction

procedure How does *K*-Means work? (Discovering similarities)
Input: Unlabeled data sets;
Output: Grouping into clusters.
    Define how many clusters will be used to group the data sets;
    Initialize all the coordinates of the k cluster centers
    repeat
        Assign each point to its nearest cluster;
        Update the centroids coordinates;
    until No changes to the centers of the clusters
    Assign new cases to one of the clusters
end procedure

Task #8[4]

Of the following examples, which would you address using an unsupervised learning algorithms? *(Check all that apply.)*

1. Given email labeled as spam/not spam, learn a spam filter
2. Given a set of news articles found on the web, group them into set of articles about the same story
3. Given a database of customer data, automatically discover market segments and group customers into different market segments
4. Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

[4] From 'Machine Learning' course on 'Coursera'

Task #8[4]

Of the following examples, which would you address using an unsupervised learning algorithms? *(Check all that apply.)*

1. Given email labeled as spam/not spam, learn a spam filter
2. Given a set of news articles found on the web, group them into set of articles about the same story
3. Given a database of customer data, automatically discover market segments and group customers into different market segments
4. Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

[4] From 'Machine Learning' course on 'Coursera'

Task #9[5]

Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10);\ \beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

[5]Credit: Shokoufeh Mirzaei, PhD

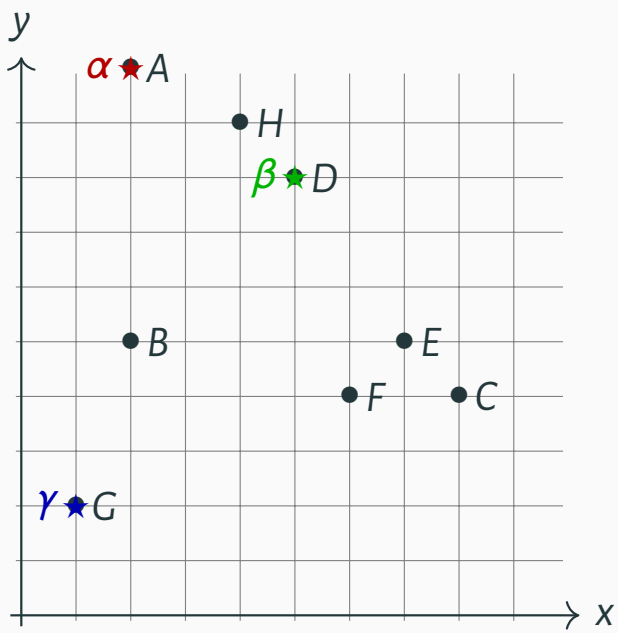
Task #9[5]

Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

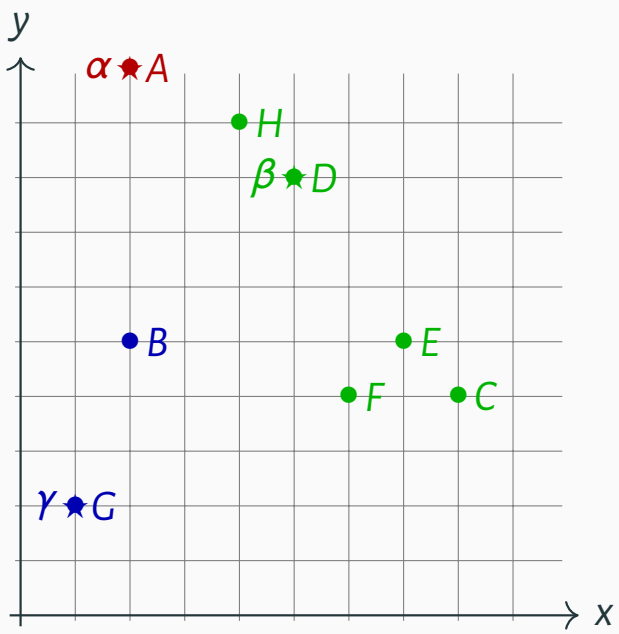
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(2,\ 10)$	$\beta(5,\ 8)$	$\gamma(1,\ 2)$	#
$A(2,\ 10)$	0	5	9	1
$B(2,\ 5)$	5	6	4	3
$C(8,\ 4)$	12	7	9	2
$D(5,\ 8)$	5	0	10	2
$E(7,\ 5)$	10	5	9	2
$F(6,\ 4)$	10	5	7	2
$G(1,\ 2)$	9	10	0	3
$H(4,\ 9)$	3	2	10	2



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

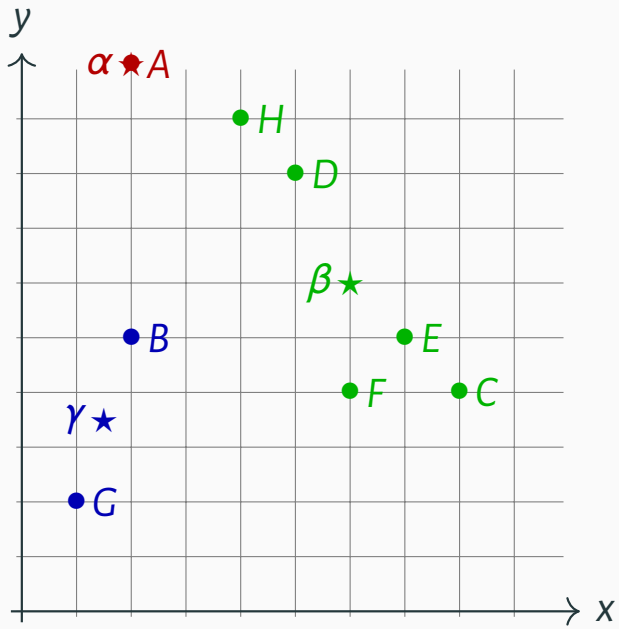
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(2,\ 10)$	$\beta(5,\ 8)$	$\gamma(1,\ 2)$	#
$A(2,\ 10)$	0	5	9	1
$B(2,\ 5)$	5	6	4	3
$C(8,\ 4)$	12	7	9	2
$D(5,\ 8)$	5	0	10	2
$E(7,\ 5)$	10	5	9	2
$F(6,\ 4)$	10	5	7	2
$G(1,\ 2)$	9	10	0	3
$H(4,\ 9)$	3	2	10	2
	$\alpha(2,\ 10)$	$\beta(6,\ 6)$	$\gamma(1.5,\ 3.5)$	



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

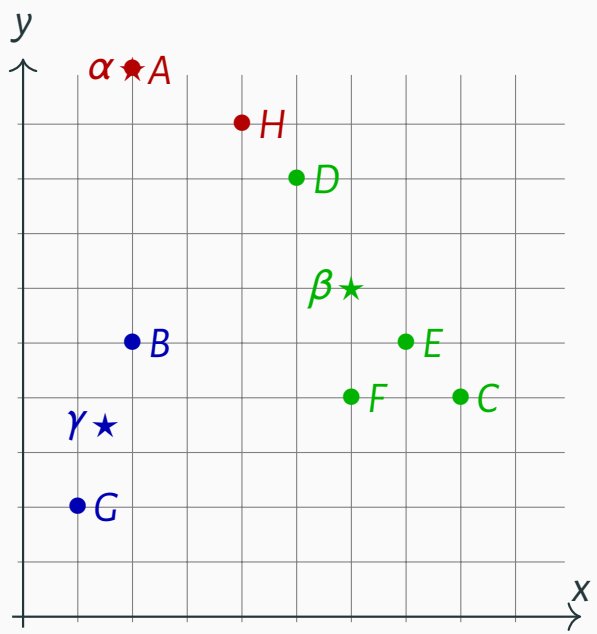
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(2,\ 10)$	$\beta(6,\ 6)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	0	8	7	1
$B(2,\ 5)$	5	5	2	3
$C(8,\ 4)$	12	4	7	2
$D(5,\ 8)$	5	3	8	2
$E(7,\ 5)$	10	2	7	2
$F(6,\ 4)$	10	2	5	2
$G(1,\ 2)$	9	9	2	3
$H(4,\ 9)$	3	5	8	1



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

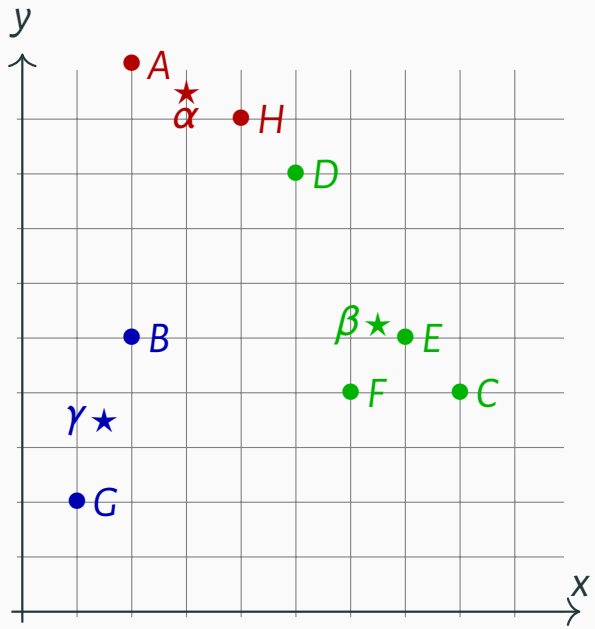
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(2,\ 10)$	$\beta(6,\ 6)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	0	8	7	1
$B(2,\ 5)$	5	5	2	3
$C(8,\ 4)$	12	4	7	2
$D(5,\ 8)$	5	3	8	2
$E(7,\ 5)$	10	2	7	2
$F(6,\ 4)$	10	2	5	2
$G(1,\ 2)$	9	9	2	3
$H(4,\ 9)$	3	5	8	1
	$\alpha(3,\ 9.5)$	$\beta(6.5,\ 5.25)$	$\gamma(1.5,\ 3.5)$	



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

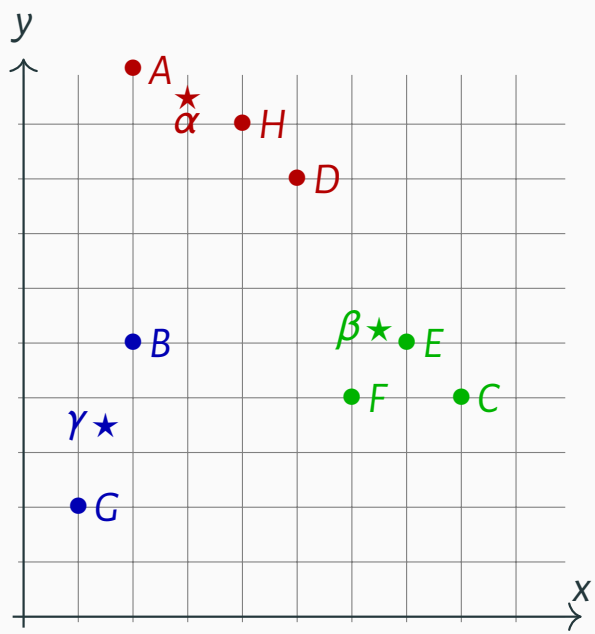
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(3,\ 9.5)$	$\beta(6.5,\ 5.25)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	1.5	9.25	7	1
$B(2,\ 5)$	5.5	4.75	2	3
$C(8,\ 4)$	10.5	2.75	7	2
$D(5,\ 8)$	3.5	4.25	8	1
$E(7,\ 5)$	8.5	0.75	7	2
$F(6,\ 4)$	8.5	1.75	5	2
$G(1,\ 2)$	9.5	8.75	2	3
$H(4,\ 9)$	1.5	6.25	8	1



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

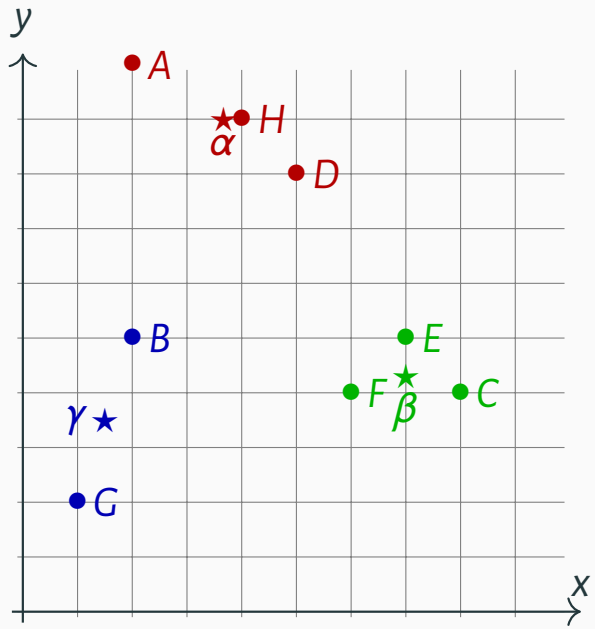
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(3,\ 9.5)$	$\beta(6.5,\ 5.25)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	1.5	9.25	7	1
$B(2,\ 5)$	5.5	4.75	2	3
$C(8,\ 4)$	10.5	2.75	7	2
$D(5,\ 8)$	3.5	4.25	8	1
$E(7,\ 5)$	8.5	0.75	7	2
$F(6,\ 4)$	8.5	1.75	5	2
$G(1,\ 2)$	9.5	8.75	2	3
$H(4,\ 9)$	1.5	6.25	8	1
	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

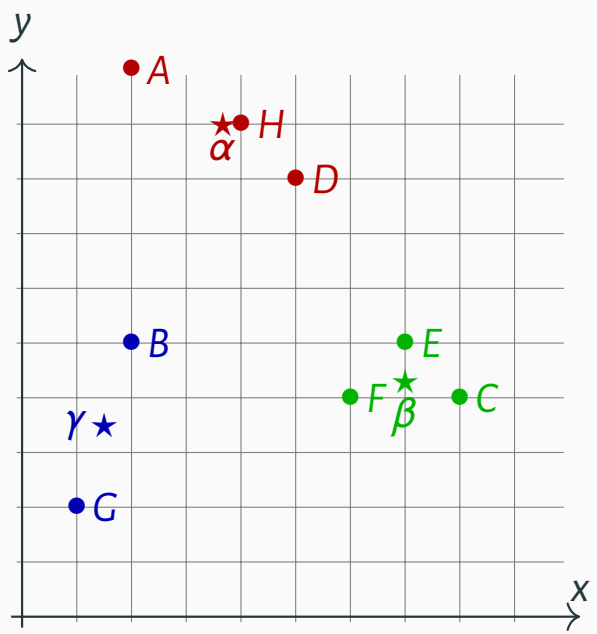
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	2.67	10.7	7	1
$B(2,\ 5)$	5.67	5.7	2	3
$C(8,\ 4)$	9.33	1.3	7	2
$D(5,\ 8)$	2.33	5.7	8	1
$E(7,\ 5)$	7.33	0.7	7	2
$F(6,\ 4)$	7.33	1.3	5	2
$G(1,\ 2)$	9.67	8.3	2	3
$H(4,\ 9)$	0.33	7.7	8	1



[5]Credit: Shokoufeh Mirzaei, PhD

Task #9[5]

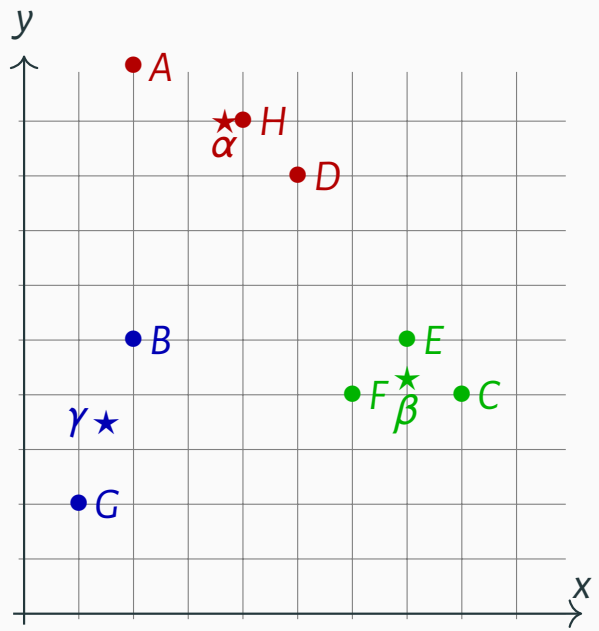
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- Initial cluster centers are: $\alpha(2,\ 10)$; $\beta(5,\ 8)$ and $\gamma(1,\ 2)$
- The distance between two points: $M(x_m,\ y_m)$ and $N(x_n,\ y_n)$ is defined as

$$d(M;\ N) \ = \ |x_m - x_n| + |y_m - y_n|$$

Point	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	#
$A(2,\ 10)$	2.67	10.7	7	1
$B(2,\ 5)$	5.67	5.7	2	3
$C(8,\ 4)$	9.33	1.3	7	2
$D(5,\ 8)$	2.33	5.7	8	1
$E(7,\ 5)$	7.33	0.7	7	2
$F(6,\ 4)$	7.33	1.3	5	2
$G(1,\ 2)$	9.67	8.3	2	3
$H(4,\ 9)$	0.33	7.7	8	1
	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	



[5]Credit: Shokoufeh Mirzaei, PhD

K-Means

The code is available @ github.com/a-mhamdi/jlai ⟶ *Codes* ⟶ *Julia* ⟶ *Part-2*

⟶ *Pluto* ⟶ *kmeans.jl*

⟶ *Jupyter* ⟶ *kmeans.ipynb*

DBSCAN: Overview

DBSCAN is a density-based clustering algorithm that groups together points closely packed in space while marking points in sparse regions as outliers.

Epsilon (ε): Maximum distance between two points to consider them as neighbors.

MinPts: Minimum number of points required to form a dense region.

Core Point: A point with at least `MinPts` neighbors within ε-distance.

Border Point: A point within ε-distance of a core point but with fewer than `MinPts` neighbors.

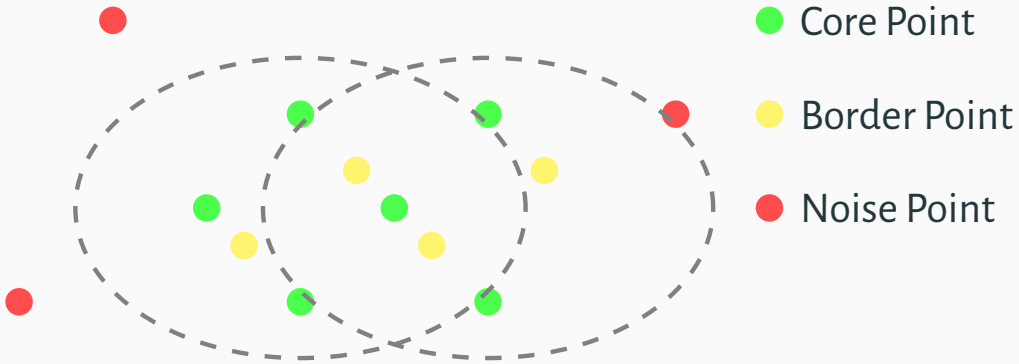
Noise Point: A point that is neither a core point nor a border point.

Steps of DBSCAN

1. Select an unvisited point and check its ε-neighborhood.
2. Mark it as a core, border, or noise point based on `MinPts`.
3. Expand clusters iteratively by connecting core points.
4. Continue until all points are visited.

DBSCAN: Visual Representation

- Points grouped based on density.
- Noise points are labeled as outliers.



- ▲ Detects clusters of arbitrary shapes.
- ▲ Robust to noise and outliers.

- ▼ Sensitive to the choice of ε and `MinPts`.
- ▼ Struggles with clusters of varying densities.
- ▼ Performance degrades with high-dimensional data.

DBSCAN: Exercise with Solution (1/3)

Task #10
Consider the following 2D dataset of points:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1.5 & 1.5 \\ 2 & 2 \\ 8 & 8 \\ 8.5 & 8.5 \\ 9 & 9 \\ 25 & 25 \end{bmatrix}$$

Apply the DBSCAN algorithm with $\varepsilon = 1.5$ and MinPts $= 3$. Identify the core, border, and noise points, and group the data into clusters.

DBSCAN: Exercise with Solution (2/3)

1. **Step #1: Find Core Points.**
   - A point is a core point if there are at least `MinPts` points within ε distance from it.
   - For $\varepsilon = 1.5$ and MinPts $= 3$, we calculate the neighbors for each point:
     - $(1, 1)$: Neighbors = $\{(1.5, 1.5), (2, 2)\}$ → Not a core point (only 2 neighbors).
     - $(1.5, 1.5)$: Neighbors = $\{(1, 1), (2, 2)\}$ → Not a core point (only 2 neighbors).
     - $(2, 2)$: Neighbors = $\{(1, 1), (1.5, 1.5)\}$ → Not a core point (only 2 neighbors).
     - $(8, 8)$: Neighbors = $\{(8.5, 8.5), (9, 9)\}$ → Not a core point (only 2 neighbors).
     - $(8.5, 8.5)$: Neighbors = $\{(8, 8), (9, 9)\}$ → Not a core point (only 2 neighbors).
     - $(9, 9)$: Neighbors = $\{(8, 8), (8.5, 8.5)\}$ → Not a core point (only 2 neighbors).
     - $(25, 25)$: No neighbors → Noise point.

DBSCAN: Exercise with Solution (3/3)

2. **Step #2: Identify Clusters.**
   - Since there are no core points, DBSCAN cannot form any clusters.
3. **Step #3: Classify Points as Core, Border, or Noise.**
   - Core Points: None (no point has at least 3 neighbors within ε distance).
   - Border Points: $(1, 1)$, $(1.5, 1.5)$, $(2, 2)$, $(8, 8)$, $(8.5, 8.5)$, $(9, 9)$ (each has 2 neighbors within ε distance).
   - Noise Point: $(25, 25)$.

Final Answer

- Clusters: None (no core points to form clusters).
- Noise: $\{(25, 25)\}$.

Principal Component Analysis (PCA): Overview

PCA is a dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional space by identifying the directions of maximum variance.

- Reduce the number of features while retaining most of the dataset's variability.
- Identify patterns in data by capturing principal components.
- Remove redundant or irrelevant information.

How PCA Works

1. Standardize the dataset (mean = 0, variance = 1).
2. Compute the covariance matrix.
3. Calculate eigenvalues and eigenvectors of the covariance matrix.
4. Project the data onto the eigenvectors with the largest eigenvalues (principal components).

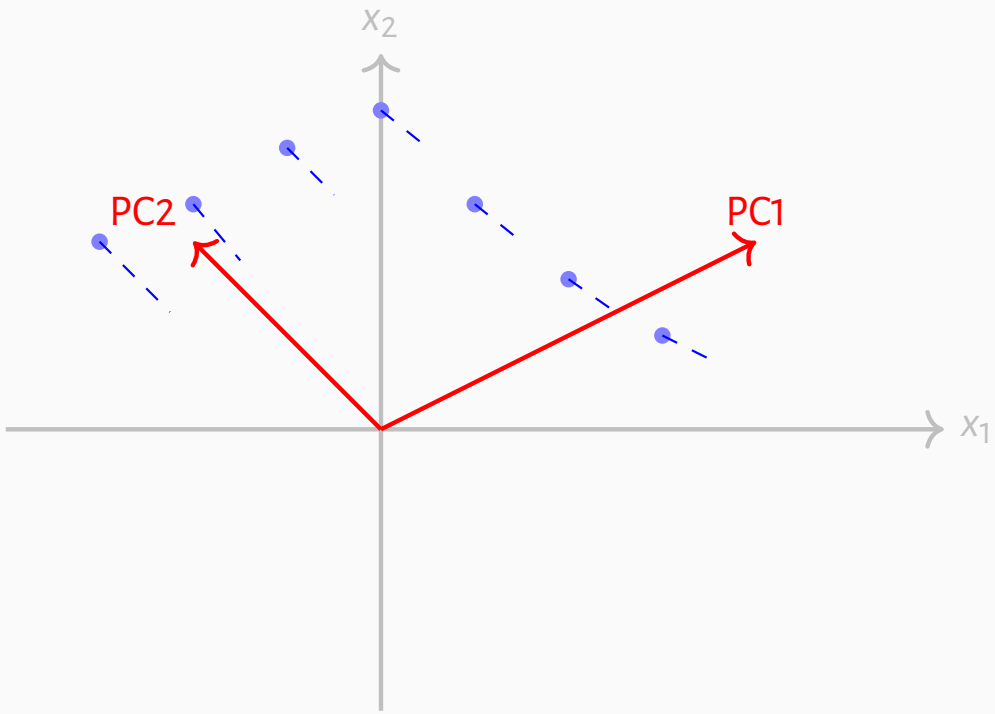
Applications

- Image compression.
- Data visualization in 2D or 3D.
- Noise reduction.

Visualizing PCA

Understanding PCA Through a 2D Example:

- Dataset with two features (x_1, x_2).
- PCA identifies the principal axes of variance (principal components).
- Data is projected onto these axes.



① PC1: Captures the most variance in the data.

② PC2: Captures the remaining variance, orthogonal to PC1.

PCA: Numeric Example (1/4)

Step-by-Step PCA Example

$$\mathbf{X} = \begin{bmatrix} 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \\ 1.9 & 2.2 \\ 3.1 & 3.0 \\ 2.3 & 2.7 \\ 2.0 & 1.6 \\ 1.0 & 1.1 \\ 1.5 & 1.6 \\ 1.1 & 0.9 \end{bmatrix}$$

PCA: Numeric Example (2/4)

Step-by-Step PCA Example

Step #1: Mean Normalization

$$\text{Mean} = \begin{bmatrix} \mu_{x_1} \\ \mu_{x_2} \end{bmatrix} = \begin{bmatrix} 1.81 \\ 1.91 \end{bmatrix}, \quad \mathbf{X}_{\text{normalized}} = \mathbf{X} - \text{Mean}$$

Step #2: Covariance Matrix

$$\mathbf{C} = \frac{1}{n-1} \mathbf{X}_{\text{normalized}}^{\top} \mathbf{X}_{\text{normalized}} = \begin{bmatrix} 0.6165 & 0.6152 \\ 0.6152 & 0.7165 \end{bmatrix}$$

Step #3: Eigenvalues and Eigenvectors

$$\text{Eigenvalues: } \lambda_1 = 1.284, \quad \lambda_2 = 0.049$$

$$\text{Eigenvectors: } \mathbf{v}_1 = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$$

PCA: Numeric Example (3/4)

Step-by-Step PCA Example

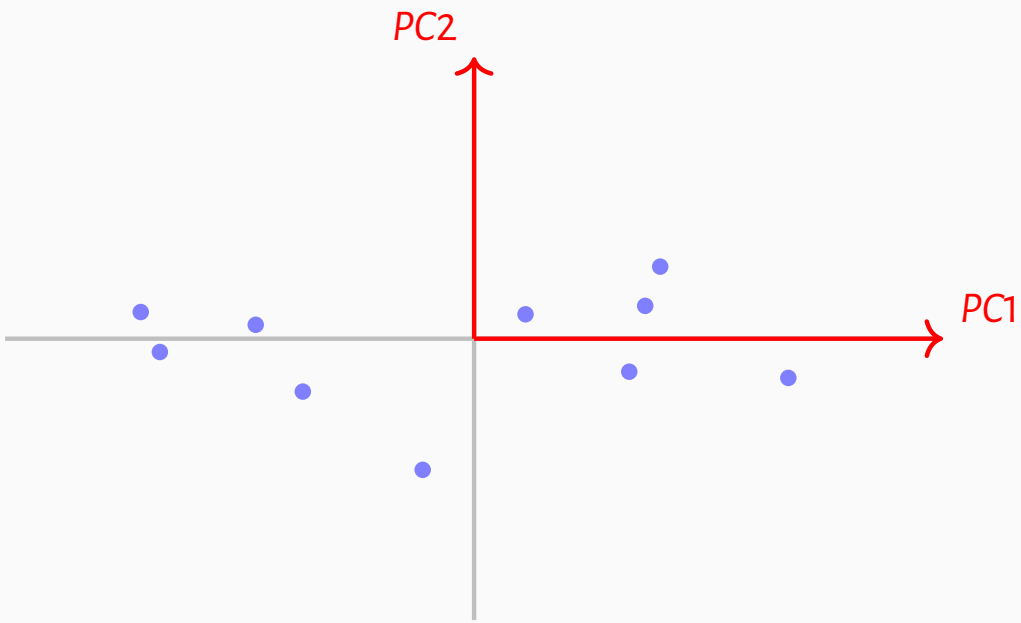
Step #4: Project Data onto Principal Components

$$\mathbf{Z} = \mathbf{X}_{\text{normalized}}\mathbf{V}, \quad \mathbf{Z} = \begin{bmatrix} 0.827 & -0.176 \\ -1.777 & 0.142 \\ 0.992 & 0.384 \\ 0.274 & 0.130 \\ 1.675 & -0.209 \\ 0.912 & 0.175 \\ -0.274 & -0.699 \\ -1.675 & -0.071 \\ -0.913 & -0.282 \\ -1.164 & 0.074 \end{bmatrix}$$

PCA: Numeric Example (4/4)

Step-by-Step PCA Example

Step #5: Visualization of Principal Components:



Key Observations

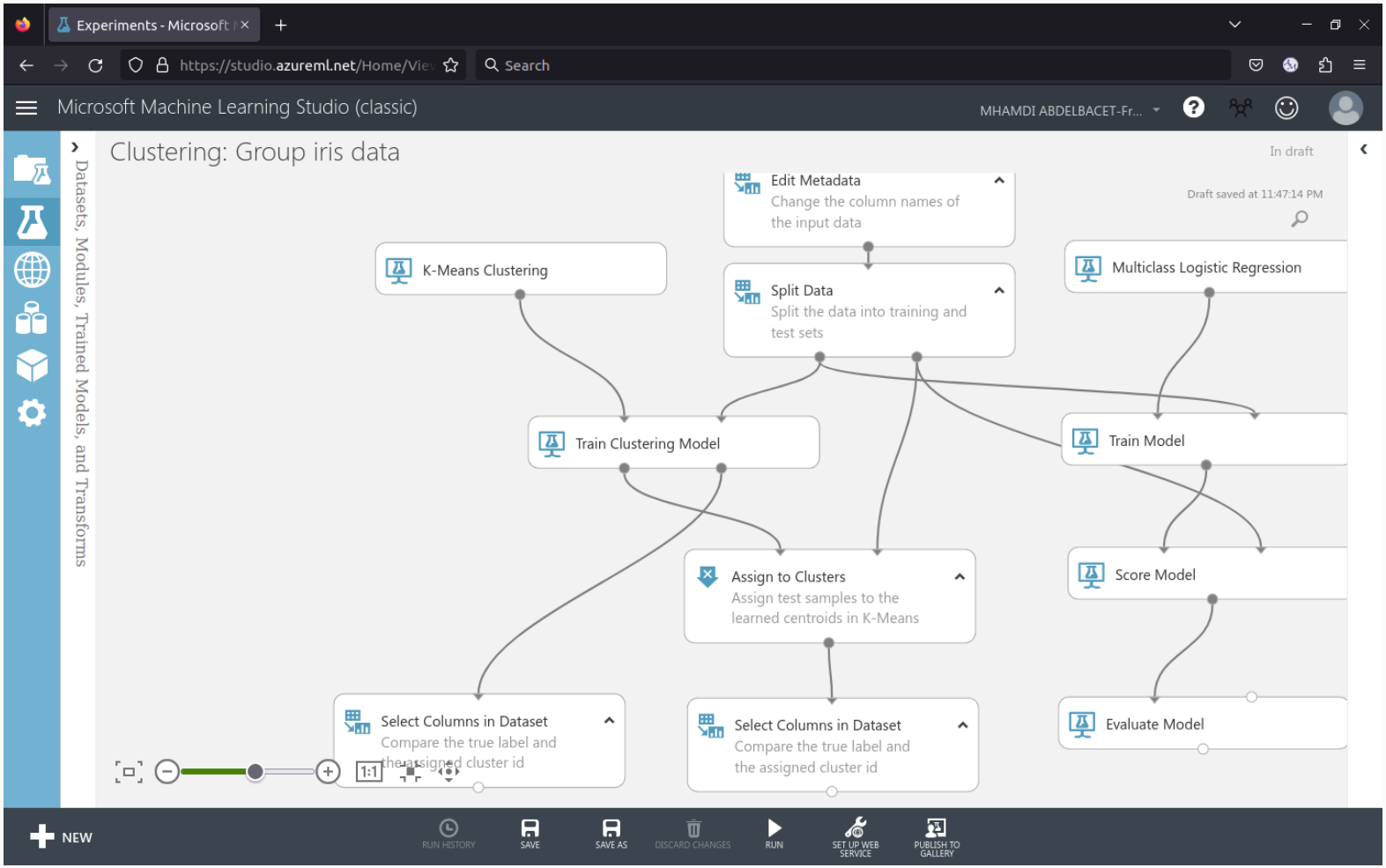
- Most variance is along *PC*1 (horizontal axis).
- Data projected onto *PC*2 (vertical axis) shows minimal variance.

Complementary Lab. Project

On the day of assignment, you will be informed about the **dataset to consider**, **specific features to keep**, and **name of machine learning model to build**. You will be asked to:

① conduct the experiment successfully *(pipeline, featurization, split, etc.)*;

② deploy a fully functional web service app that meets the given specifications.

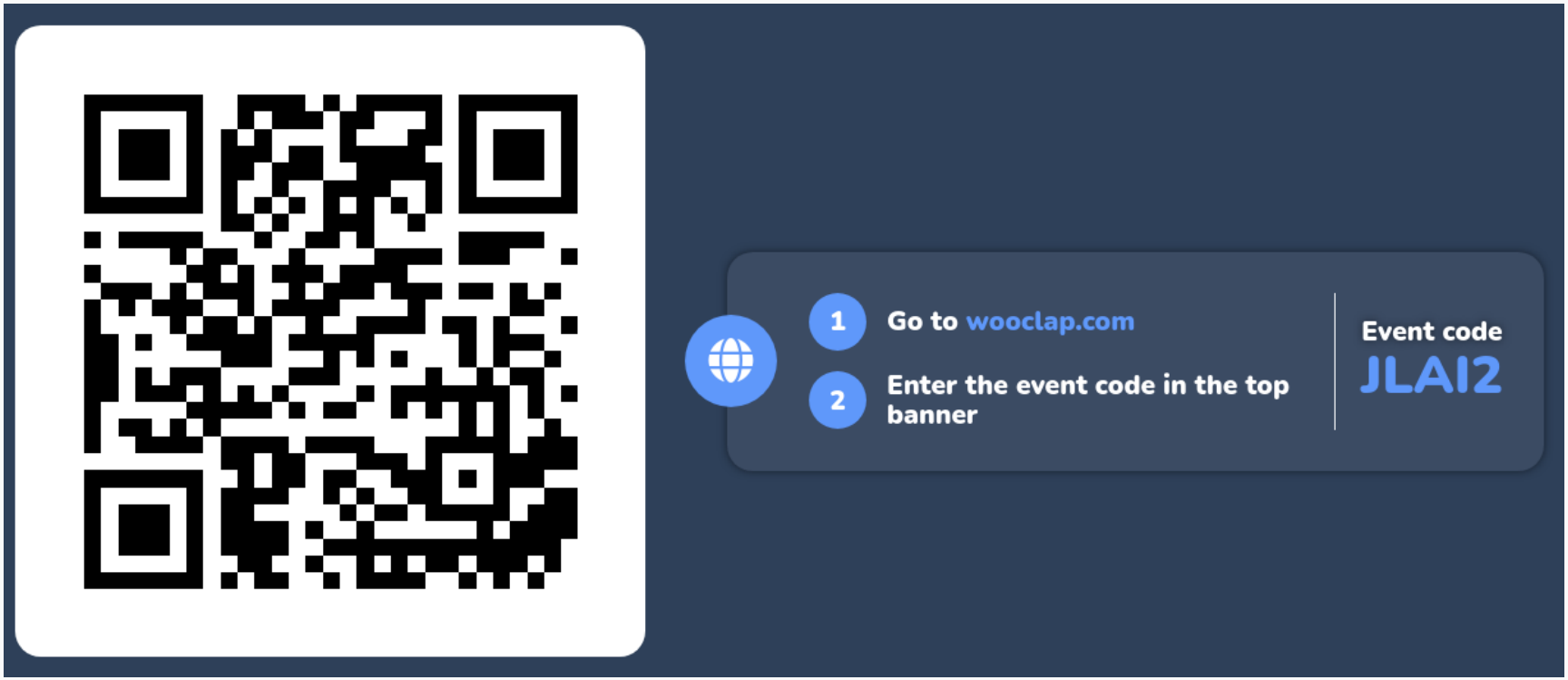


https://studio.azureml.net/

DEMO!

ML Landscape through Quizzes

Knowledge Check



https://app.wooclap.com/JLAI2

References

[Bur19] A. Burkov. ***The Hundred-Page Machine Learning Book.*** Andriy Burkov, Jan. 1, 2019. 160 pp.

[Bur20] A. Burkov. ***Machine Learning Engineering.*** True Positive Inc., Sept. 8, 2020. 310 pp.

[DFO20] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. ***Mathematics for Machine Learning.*** Cambridge University Pr., Apr. 1, 2020. 398 pp.

[ENM15] I. El Naqa and M. J. Murphy. **"What Is Machine Learning?"** In: *Machine Learning in Radiation Oncology: Theory and Applications*. Ed. by I. El Naqa, R. Li, and M. J. Murphy. Cham: Springer International Publishing, 2015, pp. 3–11. DOI: 10.1007/978-3-319-18305-3_1.

[Fla12] P. Flach. **"References".** In: *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, Sept. 2012, pp. 367–382. DOI: 10.1017/CBO9780511973000.017.

[GBC16] I. Goodfellow, J. Bengio, and A. Courville. ***Deep Learning.*** MIT Press Ltd, Nov. 18, 2016. 800 pp.

[Gé19] A. Géron. ***Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.*** O'Reilly Media, Oct. 15, 2019. 819 pp.

[HYU21] T. J. Hui (York University. ***Machine Learning Fundamentals.*** Cambridge University Press, Nov. 25, 2021. 420 pp.

Further Reading (2/2)

[Jia22] H. Jiang. ***Machine Learning Fundamentals.*** Cambridge University Pr., Jan. 31, 2022.

[JPM21] L. M. John Paul Mueller. ***Machine Learning For Dummies.*** Wiley John + Sons, Apr. 8, 2021. 464 pp.

[Mit97] T. Mitchell. ***Machine Learning.*** McGraw-Hill International Editions. McGraw-Hill, 1997.

[Pra18] M. L. de Prado. ***Advances in Financial Machine Learning.*** John Wiley & Sons Inc, May 4, 2018. 400 pp.

[SG16] A. C. M. Sarah Guido. ***Introduction to Machine Learning with Python.*** O'Reilly Media, July 31, 2016.

[Woj12] J. Wojtusiak. **"Machine Learning".** In: *Encyclopedia of the Sciences of Learning*. Springer US, 2012, pp. 2082–2083. DOI: 10.1007/978-1-4419-1428-6_1927.