Demystifying Artificial Intelligence Sorcery

(Part 2: Machine Learning)[a]

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

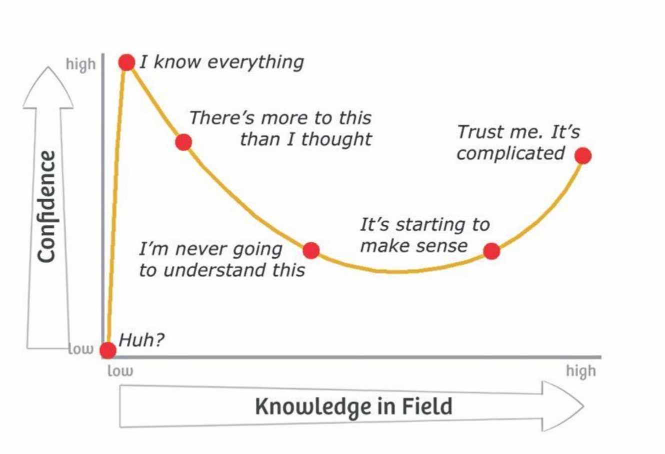
Dr.-Ing. in Electrical Engineering
Senior Lecturer at ISET Bizerte

[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.

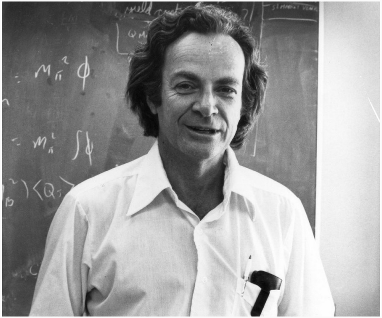
Dunning-Kruger Effect



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."

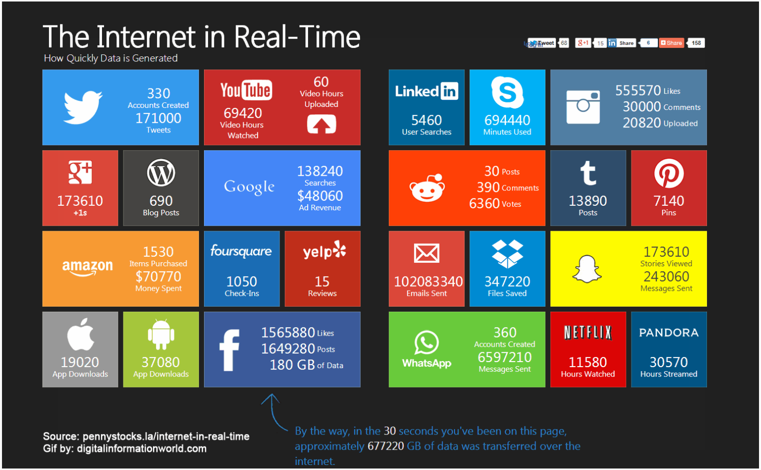
Richard P. Feynman



Roadmap

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.

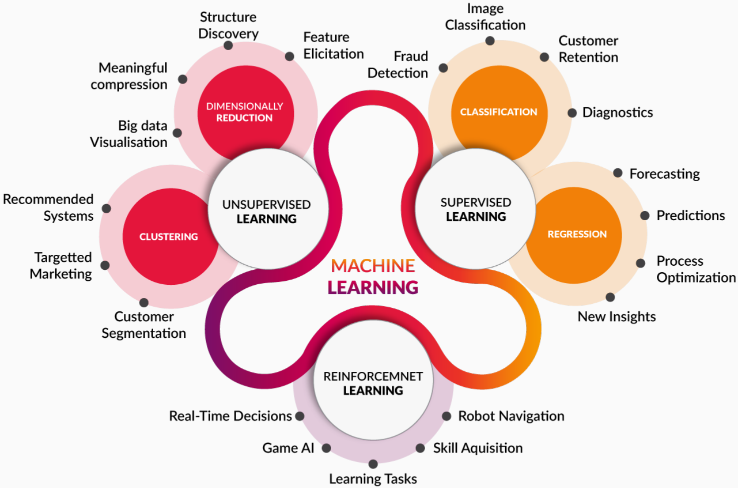
Debrief

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

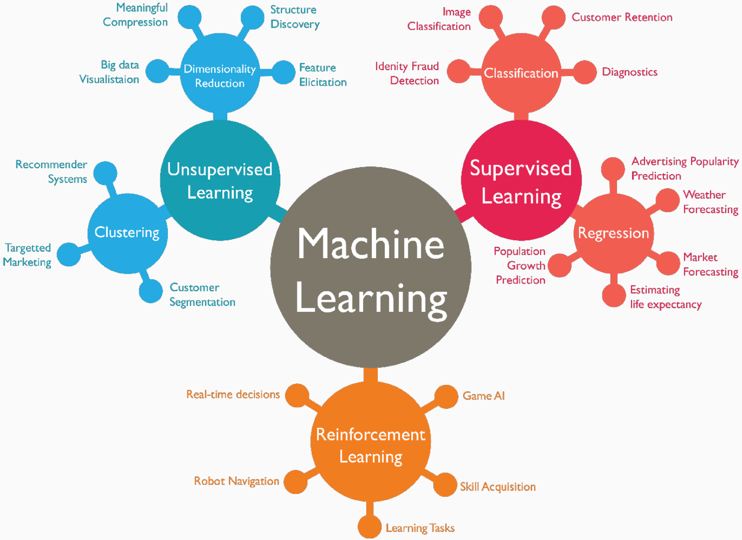
Tom Mitchell (1998)
Well-posed Learning Problem: A computer is said to learn from experience $\mathcal{E}$ with respect to some task $\mathcal{T}$ and some performance measure $\mathcal{P}$, if its performance on $\mathcal{T}$, as measured by $\mathcal{P}$, improves with experience $\mathcal{E}$.

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.

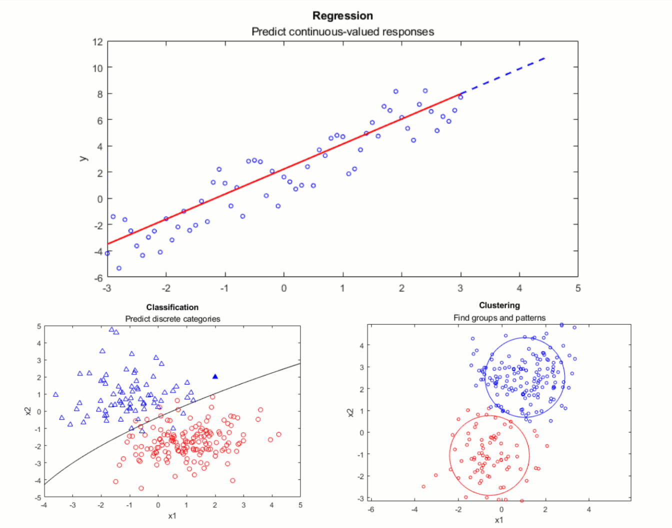


https://www.cognub.com/index.php/cognitive-platform/



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/

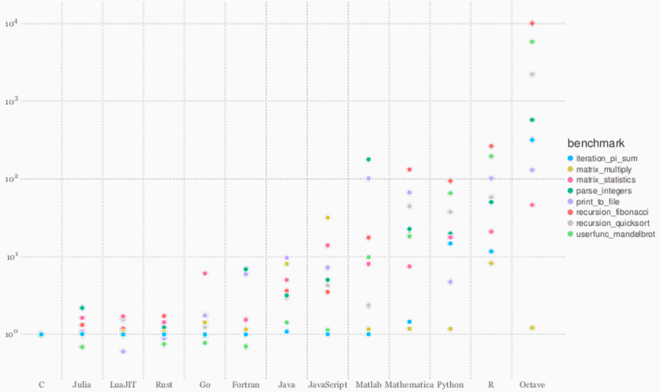
Development Environments

- ▲ $ 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

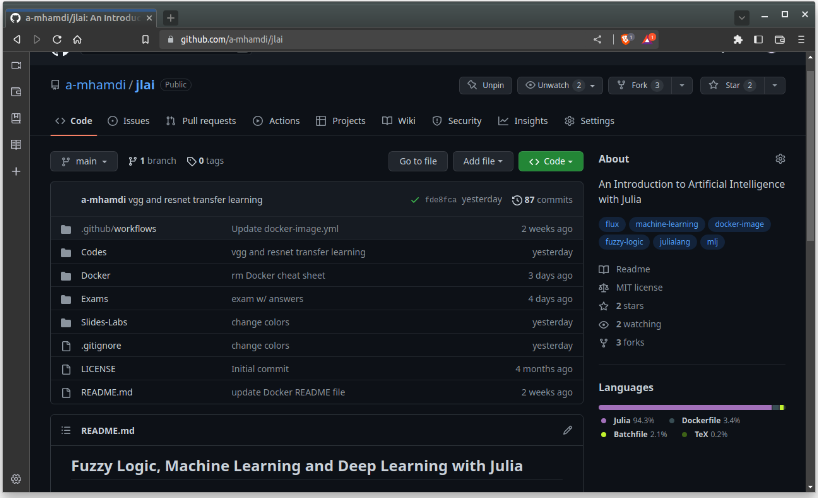
Julia Micro-Benchmarks (2/2)

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

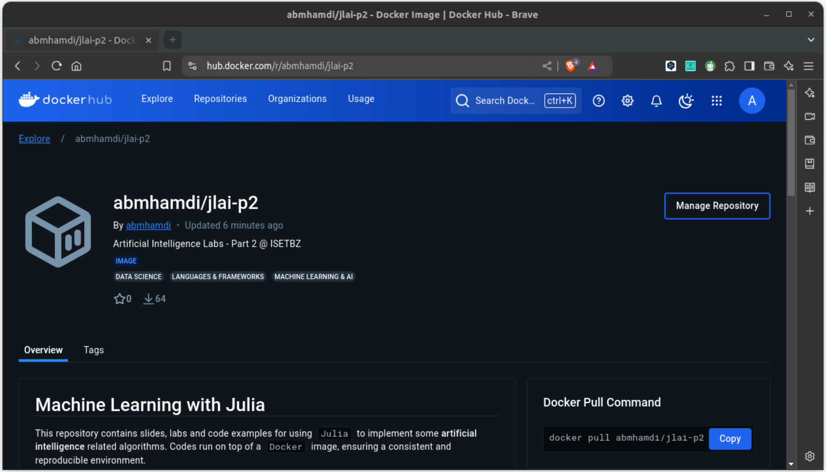
[1]Measure of central tendency expressed as $(x_1 \times x_2 \times \cdots \times x_n)^{1/n}$

Source Control Management (SCM)



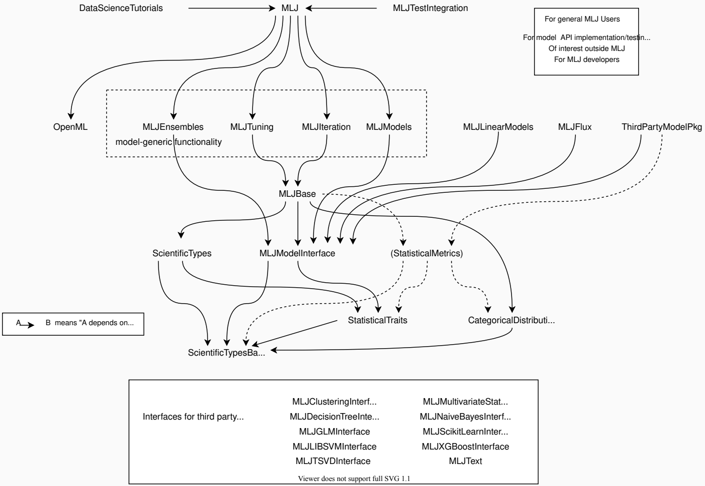
https://github.com/a-mhamdi/jlai

Docker Image



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

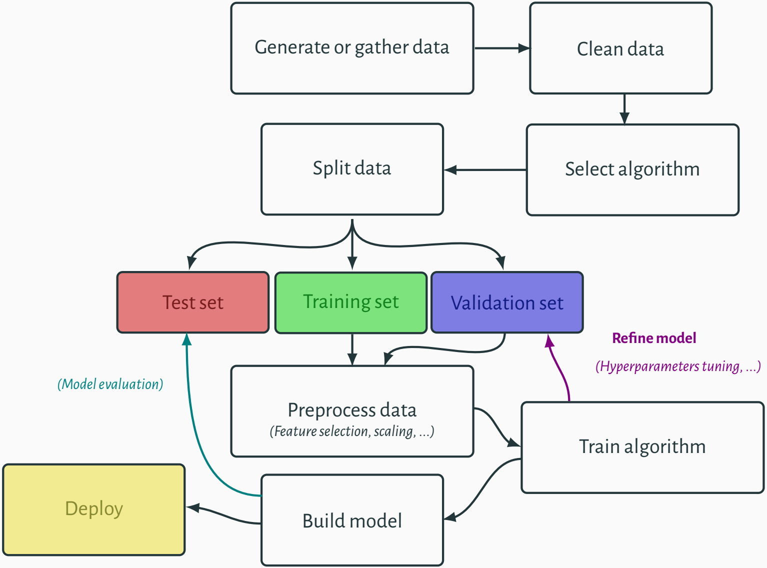
A Machine Learning Framework for Julia



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

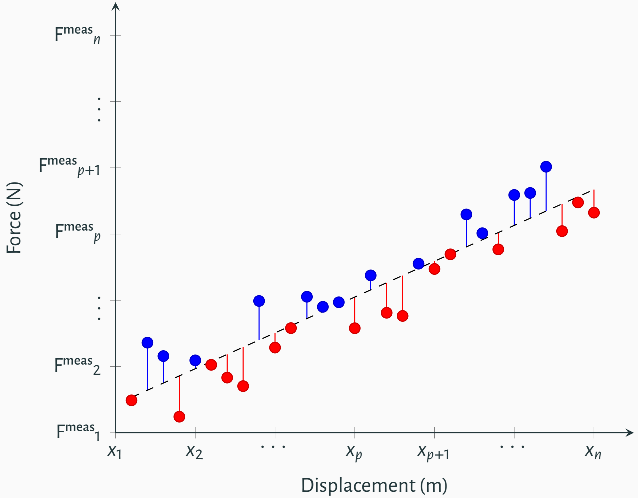
Normalization	**Standardization** *(Standardizer)*
$X \triangleq \dfrac{X - \text{minimum}(X)}{\text{maximum}(X) - \text{minimum}(X)}$	$X \triangleq \dfrac{X - \mu}{\sigma}$
▲ No assumption on data distribution	▲ More recommended when following normal distribution

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 $(x_i,\ \mathrm{F^{meas}}_i)$

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

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

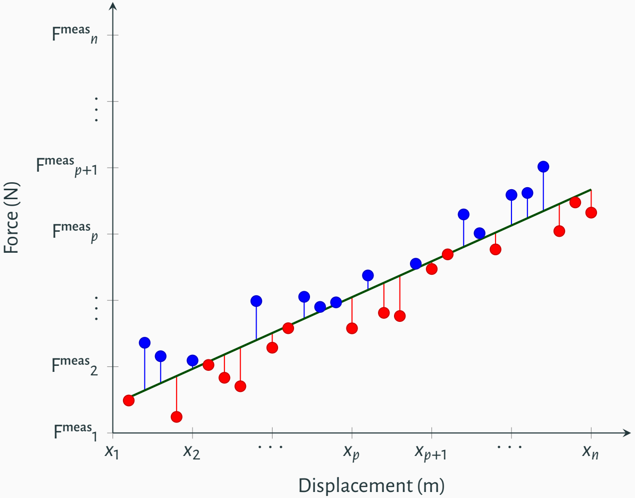
where F_i denotes the unknown real value of the force applied to the spring. In order to estimate the stiffness value k, we can consider the quadratic criterion:

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

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

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

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



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

PRACTICAL IMPLEMENTATION

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:

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

Table 2: Measurements of position y

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

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

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:

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

The vector $\boldsymbol{\varepsilon}$ is set by $\varepsilon_k, \ \forall k \geq 1$:

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

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

Practical Implementation

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:

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

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

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

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

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

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

$$\mathbf{y} \quad = \quad \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y}^T = [y_1,\ y_2,\ \cdots\ ,\ y_n]$, $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix}$, and $\boldsymbol{\varepsilon}^T = [\varepsilon_1,\ \varepsilon_2,\ \cdots\ ,\ \varepsilon_n]$.

We can consider the mean squared error or quadratic criterion in order to compute the approximated value of $\boldsymbol{\theta}$:

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

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

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

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

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

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

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

The vector $\hat{\boldsymbol{\theta}}$ is given by

$$\boxed{\hat{\boldsymbol{\theta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}}$$

$\mathbf{X}^T\mathbf{X}$ 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: $\underbrace{\boxed{\lambda \sum_{i=2}^{m} |\theta_i|}}_{\text{LASSO}} \underbrace{\boxed{\frac{1}{2}\lambda \sum_{i=2}^{m} \theta_i^2}}_{\text{RIDGE}} \underbrace{\boxed{r\lambda \sum_{i=2}^{m} |\theta_i| + \frac{(1-r)}{2}\lambda \sum_{i=2}^{m} \theta_i^2}}_{\text{ELASTIC NET}}$

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

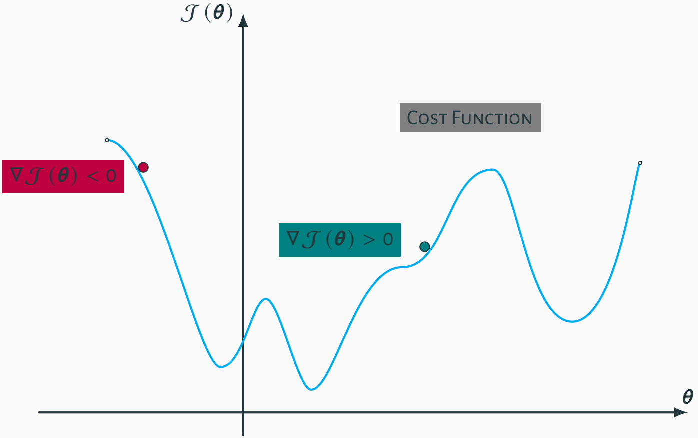
Practical Implementation

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

$\longrightarrow$ *Pluto* $\longrightarrow$ *polynomial-regression.jl*

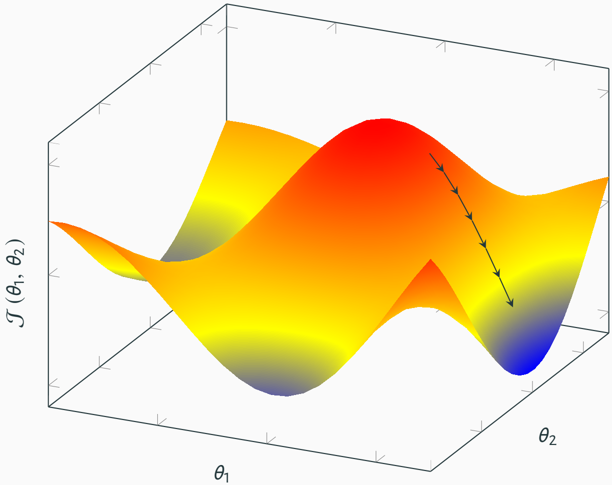
$\longrightarrow$ *Jupyter* $\longrightarrow$ *polynomial-regression.ipynb*

Gradient Descent



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

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

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

Recall that

$$\mathcal{J} = \frac{1}{2n} \sum_{k=1}^{n} \left(y_k - h_{\boldsymbol{\theta}}(\mathbf{x}_k)\right)^2 \implies \frac{\partial \mathcal{J}}{\partial \theta_i} = -\frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_{\boldsymbol{\theta}}(\mathbf{x}_k)\right) x_{(i,k)}$$

$$\boxed{\theta_1 \triangleq \theta_1 + \alpha \frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_{\boldsymbol{\theta}}\left(\mathbf{x}_k\right)\right) x_{(1,k)}} \quad \boxed{\theta_2 \triangleq \theta_2 + \alpha \frac{1}{n} \sum_{k=1}^{n} \left(y_k - h_{\boldsymbol{\theta}}\left(\mathbf{x}_k\right)\right) x_{(2,k)}}$$

$$\vdots$$

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

Gradient Descent

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

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

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

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

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

$$\vdots$$

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

Hands-on Example (1/3)

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 $\theta_0,\ \theta_1$.

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

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

Hands-on Example (2/3)

① **Normal Equation**

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

② **Stochastic Gradient Descent**

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

Hands-on Example (3/3)

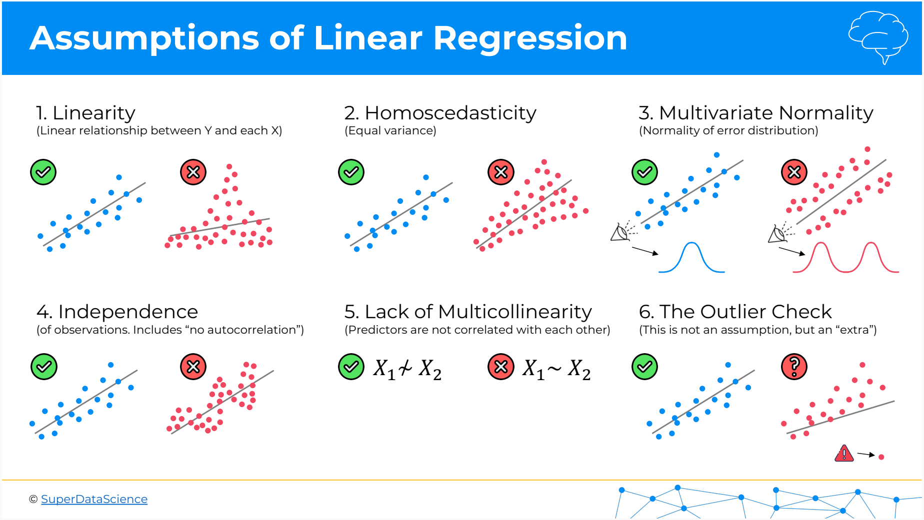
```julia
X = [1 0; 1 25; 1 50; 1 75; 1 100] # Features
y = [14, 38, 54, 76, 95] # Target

α, n, θ = 0.01, 5, [1; .5]

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

julia

https://github.com/a-mhamdi/journey-into-ML/blob/main/Julia/gradient-descent.jl


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)
$X_1 \nsim X_2$
$X_1 \sim X_2$
6. The Outlier Check
(This is not an assumption, but an “extra”)
© SuperDataScience

Source

Evaluation metrics (1/2)

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

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

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

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

Mean Absolute Percentage Error (MAPE) is a measure of the prediction quality. It is equivalent to doing weighted **MAE**.

$$\textbf{MAPE} = \frac{1}{n}\sum_{k=1}^{n}\left|\frac{y_k - \hat{y}_k}{y_k}\right| \times 100\%$$

A lower error 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.

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

Hands-on Example (1/3)

Task #3
We consider a univariate regression problem with only two predictors $m = 2$. Compute the error metrics for the given data.

y	1	1	-2	5	-3.5	1
$\hat{y}$	0.9	0.85	-2.2	4.8	-3.3	1.2

Hands-on Example (2/3)

MAE: $\frac{1}{6}\sum_{k=1}^{6}|y_k - \hat{y}_k| \approx 0.175$

RMSE: $\sqrt{\frac{1}{6}\sum_{k=1}^{6}(y_k - \hat{y}_k)^2} \approx 0.179$

MAPE: $\frac{1}{6}\sum_{k=1}^{6}\left|\frac{y_k - \hat{y}_k}{y_k}\right| \times 100\% \approx 10.786$

$\mathcal{R}^2$**:** $1 - \frac{\sum_{k=1}^{6}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{6}(y_k - \bar{y})^2} \approx 0.996$

Adjusted $\mathcal{R}^2$**:** $1 - \left(1 - \mathcal{R}^2\right)\frac{6-1}{6-2-1} \approx 0.992$

Hands-on Example (3/3)

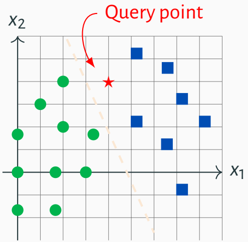
```
y = [1 1 -2 5 -3.5 1]
ŷ = [.9 .85 -2.2 4.8 -3.3 1.2]

mae = 1/6 * sum( abs.(y .- ŷ) ) # 0.17500000000000007
rmse = √(1/6 * sum( (y .- ŷ).^2 )) # 0.1791182105017057
mape = 1/6 * sum( abs.((y .- ŷ)./y) ) * 100 # 10.785714285714288

y  = 1/6 * sum(y); # 0.41666666666666663 (y\bar = ...)
# using Statistics; y  = mean(y) (y\bar = mean(y))
r2 = 1 - sum( (y .- ŷ).^2 )/sum( (y .- y ).^2 ) # 0.9955448408871745
adj_r2 = 1 - (1-r2) * (6-1)/(6-2-1) # 0.9925747348119576
```

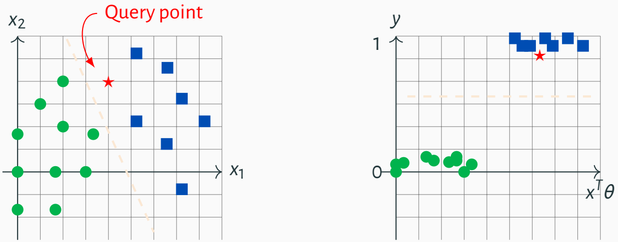
Introduction

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.



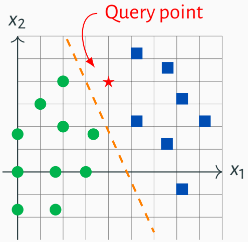
Introduction

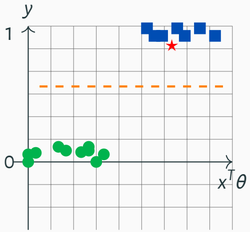
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes*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



Introduction

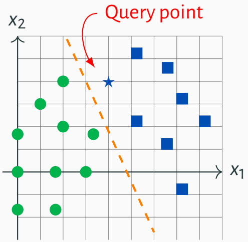
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features* x and the corresponding *output classes*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

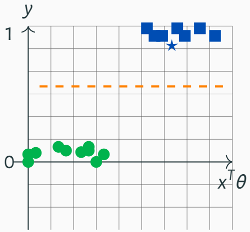




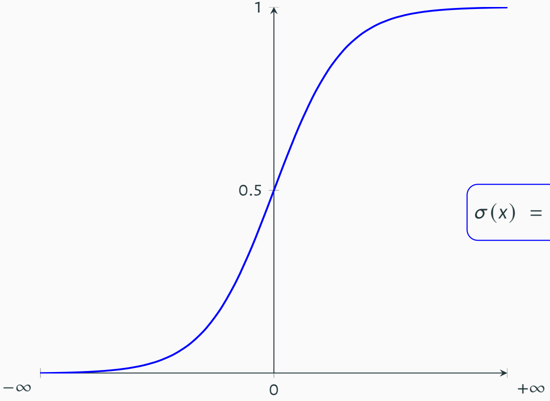
Introduction

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



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

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

Decision boundary

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

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

Hypothesis

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

For some given x_k

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

Cost function

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

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

Gradient Descent

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

Generalizing $\mathcal{J}$ yields:

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

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

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

$$\vdots$$

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

Gradient Descent

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

Generalizing $\mathcal{J}$ with $\mathbf{L}_2$ regularization term yields:

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

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

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

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

$$\vdots$$

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

Practical Implementation

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

⟶ *Pluto* ⟶ *logistic-regression.jl*

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

Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	**TP**	**FP**
	Negative	**FN**	**TN**

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

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

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

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

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

Predicted \ Actual	Positive	Negative
Positive	**TP**	**FP**
Negative	**FN**	**TN**

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

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

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

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

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 | Precision | Recall | $f1$-score

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

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

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

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

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

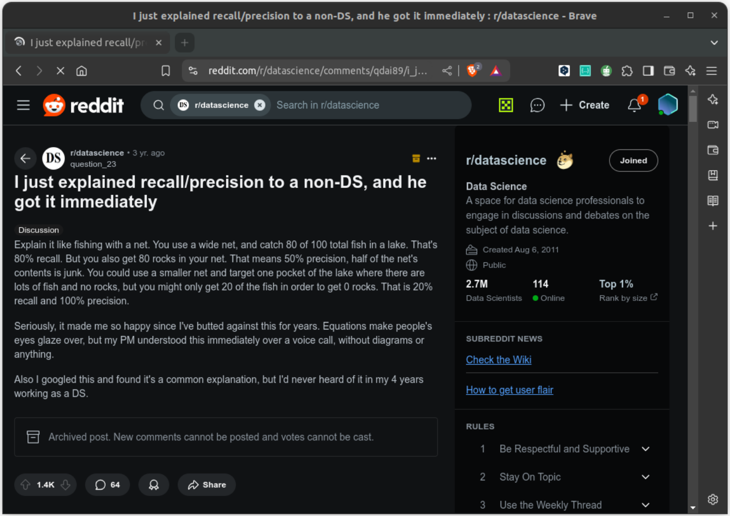
$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$f1 - \text{score}$ denotes the Harmonic Mean of *Precision* & *Recall*:

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

Evaluation metrics

An Analogy



▶ Reddit

Evaluation Metrics

Follow up

f_β-SCORE

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

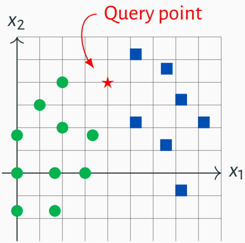
Case #1: Prioritize *Precision* over *Recall*, *e.g.*, $\beta = 0.5$

- 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 (1/2)



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 (2/2)

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

Algorithm 1 Summary Construction

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

Hands-on Example (1/2)

Task #4
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 $P(5,\ 5)$?

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

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

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

Hands-on Example (2/2)

```julia
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`
```

Hands-on Example (1/3)

Task #5

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.

Hands-on Example (2/3)

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

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

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

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

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

Hands-on Example (3/3)

```julia
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`
```

Practical Implementation

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

⟶ *Pluto* ⟶ *knn.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 $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$, where $\mathbf{x}_k \in \mathbb{R}^d$ and $y_k \in \{-1, +1\}$:

- Find a hyperplane $\mathbf{x}^\top \mathbf{w} + b = 0$ that maximizes the margin.
- Constrained optimization:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_k(\mathbf{x}_k^\top \mathbf{w} + b) \geq 1 \quad \forall k$$

Types of SVMs

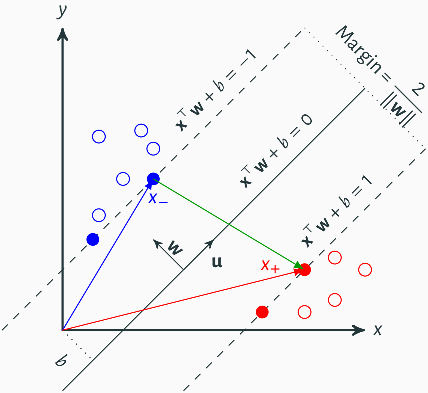
- **Linear SVM:** For linearly separable data.
- **Kernel SVM:** Maps data to higher dimensions for non-linear separation.

VISUALIZATION OF SVM

u is a direction vector of Δ.

$$\Delta : \mathbf{x}^\top \mathbf{w} + b = 0 \implies \mathbf{w} \perp \mathbf{u}$$

$$\begin{aligned}
\text{Margin} &= \frac{|(\mathbf{x}_+ - \mathbf{x}_-) \cdot \mathbf{w}|}{\|\mathbf{w}\|} \\
&= \frac{\left|\mathbf{x}_+^\top \mathbf{w} - \mathbf{x}_-^\top \mathbf{w}\right|}{\|\mathbf{w}\|} \\
&= \frac{|(1-b) - (-1-b)|}{\|\mathbf{w}\|} \\
&= \frac{2}{\|\mathbf{w}\|}
\end{aligned}$$



SVM: Loss Function

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

$$L(y, h(\mathbf{x})) = \max(0, 1 - yh(\mathbf{x})),$$

where:

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

The loss is zero if $yh(\mathbf{x}) \geq 1$ (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.

Solving the Lagrangian Problem in SVM (1/3)

Primal Problem *(Hard-Margin SVM)*

The objective is to maximize the margin between two classes. The primal optimization problem is:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_k\left(\mathbf{x}_k^\top \mathbf{w} + b\right) \geq 1, \quad \forall k$$

Construct the Lagrangian

Introduce Lagrange multipliers $\alpha_k \geq 0$ for each constraint and form the Lagrangian:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{n} \alpha_k \left(y_k\left(\mathbf{x}_k^\top \mathbf{w} + b\right) - 1\right)$$

Take Derivatives and Set to Zero

Minimize $\mathcal{L}$ with respect to $\mathbf{w}$ and b:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{k=1}^{n} \alpha_k y_k \mathbf{x}_k = 0 \implies \mathbf{w} = \sum_{k=1}^{n} \alpha_k y_k \mathbf{x}_k$$

Solving the Lagrangian Problem in SVM (2/3)

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{k=1}^{n} \alpha_k y_k = 0 \implies \sum_{k=1}^{n} \alpha_k y_k = 0$$

Substitute Back into the Lagrangian

Plugging **w** and the constraint into $\mathcal{L}$, we obtain the **dual problem**:

$$\mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{n} \alpha_k \alpha_l y_k y_l \mathbf{x}_k^\top \mathbf{x}_l$$

Solve the Dual Problem

Maximize $\mathcal{L}_D(\boldsymbol{\alpha})$ subject to:

$$\alpha_k \geq 0 \quad \text{and} \quad \sum_{k=1}^{n} \alpha_k y_k = 0$$

Solving the Lagrangian Problem in SVM (3/3)

Obtain the Solution

- The optimal $\mathbf{w}^\star$ is a linear combination of support vectors:

$$\mathbf{w}^\star = \sum_{k \in SV} \alpha_k y_k \mathbf{x}_k$$

- The bias $b^\star$ is computed using any support vector $\mathbf{x}_k$:

$$b^\star = y_k - \mathbf{w}^{\star T} \mathbf{x}_k$$

Final Decision Function

For a new input $\mathbf{x}$, the prediction is:

$$f(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^{\star T} \mathbf{x} + b^\star\right)$$

Hands-on Example (1/9)

Task #6

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. Identify the support vectors.
3. Determine the optimal hyperplane separating the two classes using SVM.
4. Calculate the margin width if the weight vector is $\mathbf{w} = [-1/2, 1/4]^\top$.

Hint!

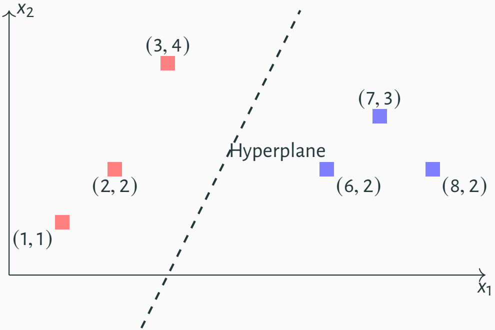
Support vectors are the data points closest to the hyperplane. The margin width is given by:

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

Hands-on Example (2/9)

Q. #1: Plot the Data Points

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



Hands-on Example (3/9)

Q. #2: Support Vectors

The support vectors are the points closest to the hyperplane:

$$\boxed{(2,2) \text{ from Class A}, \quad (3,4) \text{ from Class A}, \quad (6,2) \text{ from Class B.}}$$

Q. #3: Optimal Hyperplane

$$\begin{aligned}
\mathcal{L}_D(\boldsymbol{\alpha}) &= \sum_{k=1}^{6} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{6} \alpha_k \alpha_l y_k y_l \mathbf{x}_k^\top \mathbf{x}_l \\
&= \alpha_1 + \alpha_2 + \alpha_4 \\
&\quad - \frac{1}{2} \left(\alpha_1 \left[8\alpha_1 + 14\alpha_2 - 16\alpha_4 \right] + \alpha_2 \left[14\alpha_1 + 25\alpha_2 - 26\alpha_4 \right] - \alpha_4 \left[16\alpha_1 + 26\alpha_2 - 40\alpha_4 \right] \right)
\end{aligned}$$

Given

$$\sum_{k=1}^{6} \alpha_k y_k = 0 \therefore \alpha_1 + \alpha_2 - \alpha_4 = 0$$

$$\mathcal{L}_D(\boldsymbol{\alpha}) = 2\alpha_1 + 2\alpha_2 - \frac{1}{2} \left(16\alpha_1^2 + 13\alpha_2^2 + 24\alpha_1\alpha_2 \right)$$

Hands-on Example (4/9)

$$\frac{\partial \mathcal{L}_D(\boldsymbol{\alpha})}{\partial \alpha_1} = 0 \quad \therefore \quad 16\alpha_1 + 12\alpha_2 = 2$$

$$\frac{\partial \mathcal{L}_D(\boldsymbol{\alpha})}{\partial \alpha_1} = 0 \quad \therefore \quad 12\alpha_1 + 13\alpha_2 = 2$$

The resulting linear system is:

$$\begin{bmatrix} 16 & 12 \\ 12 & 13 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$\boldsymbol{\alpha}$ is:

$$\boldsymbol{\alpha}^{\mathsf{T}} = \begin{bmatrix} 1/32 & 1/8 & 0 & 5/32 & 0 & 0 \end{bmatrix}$$

- The optimal $\mathbf{w}^\star$ is a linear combination of support vectors:

$$\mathbf{w}^\star = 1/32 \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1/8 \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 5/32 \begin{bmatrix} 6 \\ 2 \end{bmatrix} = \begin{bmatrix} -1/2 \\ 1/4 \end{bmatrix}$$

Hands-on Example (5/9)

- The bias $b^\star$ is computed using any support vector $\mathbf{x}_k$:

$$b^\star = y_k - \mathbf{x}_k^T \mathbf{w}^\star = 3/2$$

The separating hyperplane is given by:

$$x_1 w_1 + x_2 w_2 + b = 0 \quad \text{where} \quad \mathbf{w} = \begin{bmatrix} -1/2 \\ 1/4 \end{bmatrix}^\top, \; b = 3/2$$

Thus, the equation becomes:

$$-2x_1 + x_2 + 6 = 0$$

Q. #4: Margin Width

The margin width is calculated as:

$$\text{Margin Width} = \frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{(1/2)^2 + (1/4)^2}} = \frac{8}{\sqrt{5}}$$

Hands-on Example (6/9)

```
using DataFrames, MLJ
using Plots

df = DataFrame(
    x1 = [2, 3, 1, 6, 7, 8],
    x2 = [2, 4, 1, 2, 3, 2],
    y = [1, 1, 1, -1, -1, -1]
)

scatter(df.x1, df.x2, group=df.y)

schema(df)
#=
┌───────┬──────────┬───────┐
│ names │ scitypes │ types │
├───────┼──────────┼───────┤
│ x1    │ Count    │ Int64 │
│ x2    │ Count    │ Int64 │
│ y     │ Count    │ Int64 │
└───────┴──────────┴───────┘
```

Hands-on Example (7/9)

```
=#

coerce!(df, :x1 => Continuous, :x2 => Continuous, :y => OrderedFactor)
scitype(df.y) <: AbstractVector{<:OrderedFactor} # true
schema(df)
#=
┌───────┬──────────────────┬──────────────────────────────────┐
│ names │ scitypes         │ types                            │
├───────┼──────────────────┼──────────────────────────────────┤
│ x1    │ Continuous       │ Float64                          │
│ x2    │ Continuous       │ Float64                          │
│ y     │ OrderedFactor{2} │ CategoricalValue{Int64, UInt32} │
└───────┴──────────────────┴──────────────────────────────────┘
=#

X = select(df, [:x1, :x2])
y = df.y

"[LIBSVM](https://www.csie.ntu.edu.tw/~cjlin/libsvm/)"
LinearSVC = @load LinearSVC pkg=LIBSVM
```

Hands-on Example (8/9)

```
# doc("LinearSVC", pkg="LIBSVM")

svc = LinearSVC()
#=
LinearSVC(
  solver = LIBSVM.Linearsolver.L2R_L2LOSS_SVC_DUAL,
  tolerance = Inf,
  cost = 1.0,
  bias = -1.0)
=#

mach_svc = machine(svc, X, y) |> fit!
fitted_params(mach_svc)
#=
(libsvm_model = LIBLINEAR.LinearModel{UInt32}(1, 2, 2, [-0.486189670915792, 0.
 ↪8956996119492998], Int32[1, 2], UInt32[0x00000002, 0x00000001], -1.0, 0.0),
 encoding = Dict{UInt32, CategoricalArrays.CategoricalValue{String, UInt32}}
 ↪(0x00000002 => "1", 0x00000001 => "-1"),)
=#

```

Hands-on Example (9/9)

```julia
using LIBSVM
Data = Matrix(X)'
mdl = svmtrain(D, y, kernel=Kernel.Linear)
mdl.SVs
#=
LIBSVM.SupportVectors{CategoricalArrays.CategoricalVector{Int64, UInt32, Int64, CategoricalArrays.CategoricalValue{Int64, UInt32}, Union{}}, Matrix{Float64}}(3, Int32[2, 1], CategoricalArrays.CategoricalValue{Int64, UInt32}[1, 1, -1], [2.0 3.0 6.0; 2.0 4.0 2.0], Int32[1, 2, 4], LIBSVM.SVMNode[LIBSVM.SVMNode(1, 2.0), LIBSVM.SVMNode(1, 3.0), LIBSVM.SVMNode(1, 6.0)])
=#
```

Hard Margin vs. Soft Margin SVM

- **Data Assumption**

  Hard: Strict linear separability
  Soft: Tolerates noise/overlap

- **Primal Form**

  Hard: $\min \frac{1}{2}\|w\|^2 \qquad \text{s.t. } y_k\left(\mathbf{x}_k^\top \mathbf{w} + b\right) \geq 1$
  Soft: $\min \frac{1}{2}\|w\|^2 + C\sum \xi_k \qquad \text{s.t. } y_k\left(\mathbf{x}_k^\top \mathbf{w} + b\right) \geq 1 - \xi_k$

- **Key Features**

  Hard: No slack variables (ξ), $C \longrightarrow \infty$
  Soft: Slack variables ($\xi_i \geq 0$), finite C (trade-off)

- **Dual Form**

  Hard: $0 \leq \alpha_i$
  Soft: $0 \leq \alpha_i \leq C$
  Both: $\sum \alpha_i y_i = 0$

- **Practicality**

  Hard: Rarely used (overfits)
  Soft: Robust to noise, works with kernels

Kernel Trick in SVMs

Dual Form with Kernel:

$$\mathcal{L}_D(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{n} \alpha_k \alpha_l y_k y_l \underbrace{\mathbf{x}_k^\top \mathbf{x}_l}_{\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l)}$$

$$\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l) = \Phi^\top(\mathbf{x}_k)\,\Phi(\mathbf{x}_l)$$

Kernel Trick: Replace inner products with kernel functions - no need to compute $\Phi(\mathbf{x})$ explicitly.

Common Kernels:

- **Linear**: $\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l) = \mathbf{x}_k^\top \mathbf{x}_l$
- **Polynomial**: $\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l) = \left(\gamma \mathbf{x}_k^\top \mathbf{x}_l + r\right)^d$
- **RBF (Gaussian)**: $\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\gamma \left\|\mathbf{x}_k - \mathbf{x}_l\right\|^2\right)$
- **Sigmoid**: $\mathcal{K}(\mathbf{x}_k, \mathbf{x}_l) = \tanh\left(\gamma \mathbf{x}_k^\top \mathbf{x}_l + r\right)$

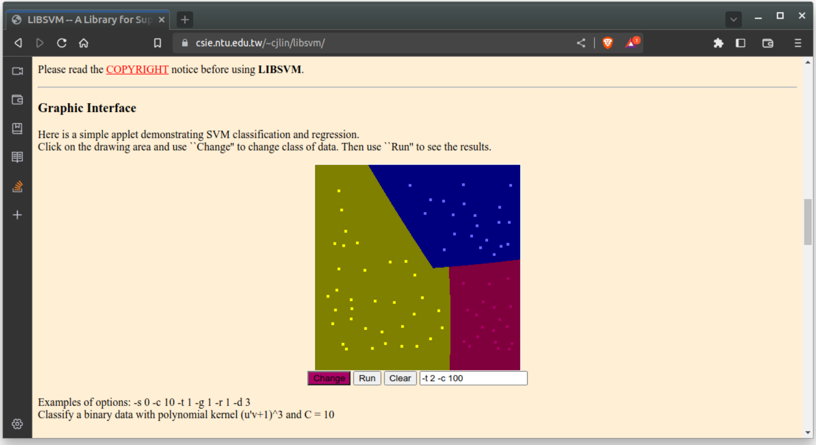
Parameters

d: Polynomial degree (#non-linear terms)

γ: Kernel scale (RBF/poly)

C: Regularization (controls margin vs. errors)

LIBSVM LIBRARY



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

Practical Implementation

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

⟶ *Pluto* ⟶ *svc.jl*

⟶ *Jupyter* ⟶ *svc.ipynb*

Naive Bayes

COVID-19 Testing

Task #7
Imagine we have the following information:

- 1% of the population has COVID-19 (prior probability)
- A COVID test has 95% sensitivity (true positive rate) - if you have COVID, there's a 95% chance the test will be positive
- The test has 90% specificity (true negative rate) - if you don't have COVID, there's a 90% chance the test will be negative

Now, if someone receives a positive test result, what's the probability they actually have COVID-19?

Naive Bayes

COVID-19 Testing

Task #7

Imagine we have the following information:

- 1% of the population has COVID-19 (prior probability)
- A COVID test has 95% sensitivity (true positive rate) - if you have COVID, there's a 95% chance the test will be positive
- The test has 90% specificity (true negative rate) - if you don't have COVID, there's a 90% chance the test will be negative

Now, if someone receives a positive test result, what's the probability they actually have COVID-19?

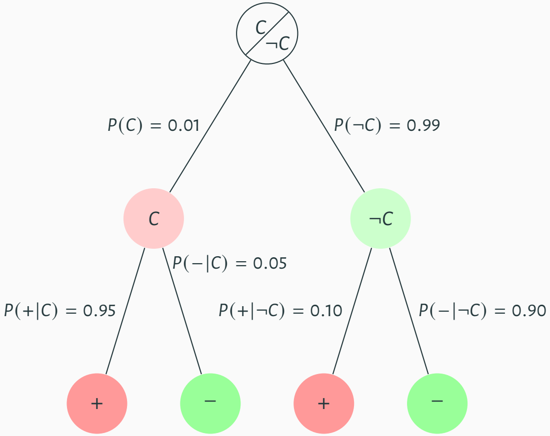
$$P(+) = P(C \cap +) + P(\neg C \cap +) = 0.0095 + 0.099 = 0.1085$$

$$P(C|+) = \frac{P(C \cap +)}{P(+)} = \frac{0.0095}{0.1085} \approx 0.088 \ \mathbf{(8.8\%)}$$

$\therefore$ Despite a positive test, there's only an 8.8% chance the person actually has COVID-19.

Naive Bayes

COVID-19 Testing



$$P(C \cap +) = 0.01 \times 0.95 = 0.0095$$
$$P(C \cap -) = 0.01 \times 0.05 = 0.0005$$

$$P(\neg C \cap +) = 0.99 \times 0.10 = 0.099$$
$$P(\neg C \cap -) = 0.99 \times 0.90 = 0.891$$

Naive Bayes (1/2)

Definition and Key Concept

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.

$$\hat{y} = \arg\max_{y \in Y} P(y|X) = \arg\max_{y \in Y} P(X|y) \cdot P(y)$$

Naive Bayes (2/2)

Definition and Key Concept

Naive Bayes for Spam Detection

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

Classes: “Spam” (S) or “Not Spam” $(\neg S)$.

Steps:

① Calculate $P(S)$ and $P(\neg S)$ from the training data.

② Calculate $P(\text{Word}|S)$ and $P(\text{Word}|\neg S)$ for each word.

③ Use Bayes’ Theorem to predict the class of a new email.

Hands-on Example (1/7)

Task #8
You are given a dataset of emails with the following word frequencies and labels:

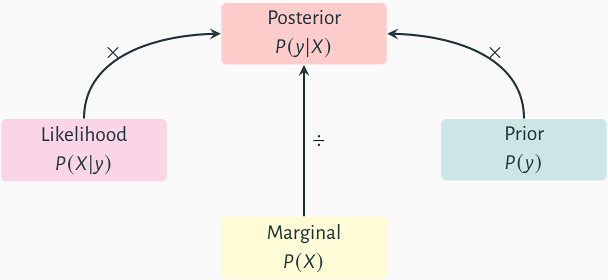
Email	"free"	"money"	Label
1	1	0	S
2	0	1	S
3	0	0	$\neg S$
4	0	1	$\neg S$

1. Calculate the prior probabilities $P(S)$ and $P(\neg S)$.
2. Calculate the likelihoods $P(\text{"free"}|S)$, $P(\text{"money"}|S)$, $P(\text{"free"}|\neg S)$, and $P(\text{"money"}|\neg S)$.
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.

Hands-on Example (2/7)



Step 1: Calculate Prior Probabilities

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

Hands-on Example (3/7)

Step 2: Calculate Likelihoods with Laplace Smoothing

$$P\left(w_i|c\right) = \frac{\text{count}\left(w_i, c\right) + k}{\sum_{w \in V} \text{count}(w, c) + k|V|}$$

k: Smoothing parameter ($k \geq 0$)

$|V|$: Vocabulary size

count $\left(w_i, c\right)$: Frequency of word w_i in class c

- $P(\text{“free”}|S) = \frac{\text{Count of “free” in Spam} + 1}{\text{Count of Words in Spam} + |\text{Vocabulary}|} = \frac{1+1}{2+2} = \frac{2}{4} = 0.5$
- $P(\text{“money”}|S) = \frac{\text{Count of “money” in Spam} + 1}{\text{Count of Words in Spam} + |\text{Vocabulary}|} = \frac{1+1}{2+2} = \frac{2}{4} = 0.5$
- $P(\text{“free”}|\neg S) = \frac{\text{Count of “free” in Not Spam} + 1}{\text{Count of Words in} \neg \text{Spam} + |\text{Vocabulary}|} = \frac{0+1}{1+2} = \frac{1}{3} \approx 0.333$
- $P(\text{“money”}|\neg S) = \frac{\text{Count of “money” in Not Spam} + 1}{\text{Count of Words in} \neg \text{Spam} + |\text{Vocabulary}|} = \frac{1+1}{1+2} = \frac{2}{3} \approx 0.666$

Hands-on Example (4/7)

Step 3: Classify a New Email

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

$$P(S|\text{"free" and "money"}) \propto P(\text{"free"}|S) \cdot P(\text{"money"}|S) \cdot P(S)$$
$$= 1/2 \cdot 1/2 \cdot 1/2 = 1/6$$

$$P(\neg S|\text{"free" and "money"}) \propto P(\text{"free"}|\neg S) \cdot P(\text{"money"}|\neg S) \cdot P(\neg S)$$
$$= 1/3 \cdot 2/3 \cdot 1/2 = 1/9$$

- Since $1/6 > 1/9$, the email is classified as **Spam**.

Hands-on Example (5/7)

```
using DataFrames, MLJ

df = DataFrame(
    free = [1, 0, 0, 0],
    money = [0, 1, 0, 1],
    label = ["S", "S", "NS", "NS"]
    )
schema(df)
#=
┌───────┬──────────┬────────┐
│ names │ scitypes │ types  │
├───────┼──────────┼────────┤
│ free  │ Count    │ Int64  │
│ money │ Count    │ Int64  │
│ label │ Textual  │ String │
└───────┴──────────┴────────┘
=#

coerce!(df, :label => OrderedFactor)
scitype(df.label) <: AbstractVector{<:OrderedFactor} # true
```

Hands-on Example (6/7)

```
schema(df)
#=
┌───────┬─────────────────┬──────────────────────────────────┐
│ names │ scitypes        │ types                            │
├───────┼─────────────────┼──────────────────────────────────┤
│ free  │ Count           │ Int64                            │
│ money │ Count           │ Int64                            │
│ label │ OrderedFactor{2} │ CategoricalValue{String, UInt32} │
└───────┴─────────────────┴──────────────────────────────────┘
=#

X = select(df, [:free, :money])
y = df.label

MultinomialNBClassifier = @load MultinomialNBClassifier pkg=NaiveBayes
mnb = MultinomialNBClassifier()
#=
MultinomialNBClassifier(
  alpha = 1)
=#
```

Hands-on Example (7/7)

```julia

mach_nb = machine(mnb, X, y) |> fit!
predict(mach_nb, DataFrame(free = 1, money = 1))
#=
1-element CategoricalDistributions.UnivariateFiniteVector{Multiclass{2}, String, UInt32, Float64}:
 UnivariateFinite{Multiclass{2}}(NS=>0.471, S=>0.529)
=#
```

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.

Practical Implementation

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

⟶ *Pluto* ⟶ *naive-bayes.jl*

⟶ *Jupyter* ⟶ *naive-bayes.ipynb*

Decision Tree: Definition and Key Concepts

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).

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.

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

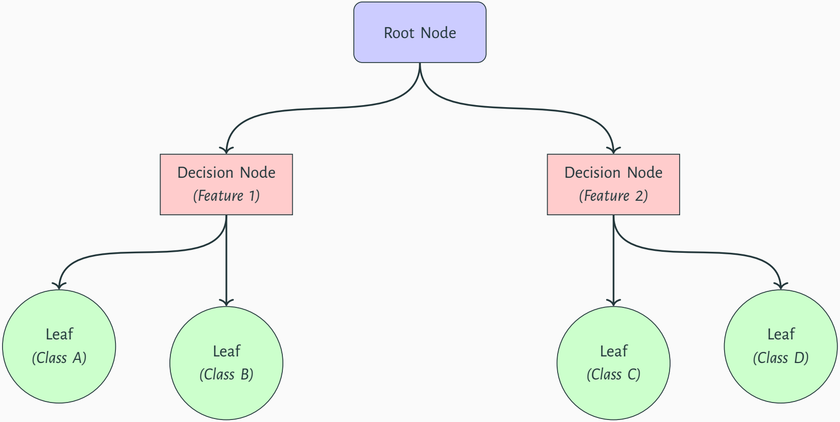
- Measures the uncertainty of a node.
- Lower values indicate more certainty.

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

Comparison

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

Decision Tree Visualization



Hands-on Example (1/13)

Task #9
Considering the following restaurant wait time dataset, we need to predict if a reservation will be honored (Yes/No). Construct a **decision tree** using Gini Index:

Party Size	Day of Week	Special Occasion?	Reservation Honored?
2	Weekday	No	Yes
4	Weekend	Yes	No
6	Weekend	Yes	No
2	Weekend	No	Yes
4	Weekday	No	Yes
8	Weekend	Yes	No
2	Weekday	Yes	Yes
4	Weekend	No	No
6	Weekday	No	Yes
2	Weekend	Yes	No

Hands-on Example (2/13)

Step 1: Root Node Gini Impurity

$$G_{\text{Root}} = 1 - \left(\left(\frac{5}{10}\right)^2 + \left(\frac{5}{10}\right)^2\right) = 1 - (0.25 + 0.25) = 0.5$$

Step 2: Feature Split Analysis

Split 1: "Party Size"

$$G_{ps=2} = 1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right) = 0.375$$

$$G_{ps=4} = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right) = 0.444$$

$$G_{ps=6} = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) = 0.5$$

$$G_{ps=8} = 1 - \left(\left(\frac{0}{1}\right)^2 + \left(\frac{1}{1}\right)^2\right) = 0$$

$$G_{\text{split}} = \frac{4}{10} \times 0.375 + \frac{3}{10} \times 0.444 + \frac{2}{10} \times 0.5 + \frac{1}{10} \times 0 = 0.383$$

Hands-on Example (3/13)

Split 2: "DOTW"'

$$
\begin{aligned}
G_{\text{weekday}} &= 1 - \left(\left(\frac{4}{4}\right)^2 + \left(\frac{0}{4}\right)^2\right) &= 0 \\
G_{\text{weekend}} &= 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) &= 0.278 \\
G_{\text{split}} &= \frac{4}{10} \times 0 + \frac{6}{10} \times 0.278 &= 0.167
\end{aligned}
$$

Split 3: "Special Occasion?"

$$
\begin{aligned}
G_{\text{special}} &= 1 - \left(\left(\frac{1}{5}\right)^2 + \left(\frac{4}{5}\right)^2\right) &= 0.32 \\
G_{\text{no special}} &= 1 - \left(\left(\frac{4}{5}\right)^2 + \left(\frac{1}{5}\right)^2\right) &\approx 0.32 \\
G_{\text{split}} &= \frac{5}{10} \times 0.32 + \frac{5}{10} \times 0.32 &\approx 0.32
\end{aligned}
$$

Hands-on Example (4/13)

Step 3: Optimal Split Selection

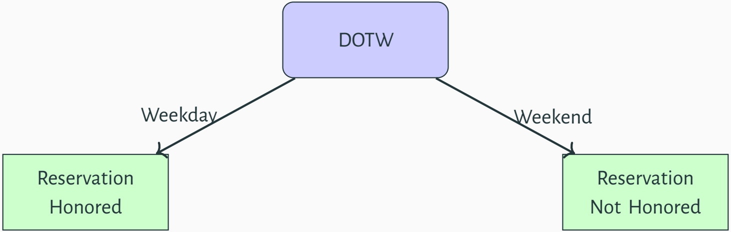
- **DOTW** has the lowest Gini impurity ($G = 0.167$)
- **Party Size?** ($G = 0.383$) and **Special Occasion?** ($G \approx 0.32$) are less optimal

Resulting Decision Tree

1st split: "Weekend?"

Weekday: Predict "Reservation Honored" (4/4 samples) *(pure leaf node)*
Weekend: Predict "Reservation Not Honored" (5/6 samples)

Subsequent splits: Refine with "Party Size" or "Special Occasion?"



Hands-on Example (5/13)

Party Size	SO?	RH?
2	No	Yes
4	No	Yes
2	Yes	Yes
6	No	Yes

Weekday Node Gini Impurity

$$G_{\text{Weekday}} = 1 - \left(\left(\frac{4}{4}\right)^2 + \left(\frac{0}{4}\right)^2\right) = 0$$

Party Size	SO?	RH?
4	Yes	No
6	Yes	No
2	No	Yes
8	Yes	No
4	No	No
2	Yes	No

Weekend Node Gini Impurity

$$G_{\text{Weekend}} = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) = 0.278$$

Hands-on Example (6/13)

"Party Size?" Split

$$
\begin{aligned}
G_{\text{ps}=2} &= 1-\left(\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right) &&= 0.5\\
G_{\text{ps}=4} &= 1-\left(\left(\frac{0}{2}\right)^2+\left(\frac{2}{2}\right)^2\right) &&= 0\\
G_{\text{ps}=6} &= 1-\left(\left(\frac{0}{1}\right)^2+\left(\frac{1}{1}\right)^2\right) &&= 0\\
G_{\text{ps}=8} &= 1-\left(\left(\frac{0}{1}\right)^2+\left(\frac{1}{1}\right)^2\right) &&= 0\\
G_{\text{split}} &= \frac{2}{6}\times 0.5+\frac{2}{6}\times 0+\frac{1}{6}\times 0+\frac{1}{6}\times 0 &&= 0.167
\end{aligned}
$$

Hands-on Example (7/13)

"Special Occasion?" Split

$$
\begin{aligned}
G_{\text{special}} &= 1 - \left(\left(\frac{0}{4}\right)^2 + \left(\frac{4}{4}\right)^2\right) &&= 0 \\
G_{\text{no special}} &= 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) &&\approx 0.5 \\
G_{\text{split}} &= \frac{4}{6} \times 0 + \frac{2}{6} \times 0.5 &&\approx 0.167
\end{aligned}
$$

Hands-on Example (8/13)

Optimal Split Selection

Both "Party Size" and "Special Occasion?" have the same Gini Index (0.1667) for the "Weekend" branch.

In this case, we can choose either one. Let's use "Special Occasion?" as our second split.

Resulting Decision Tree

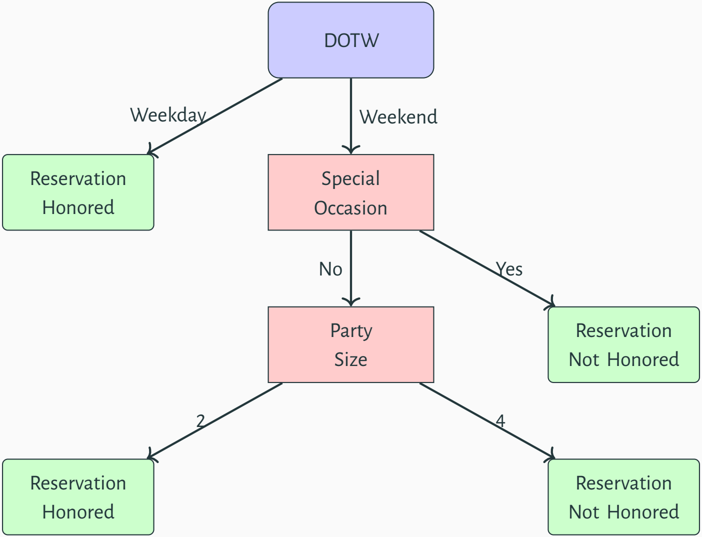
2nd split: Special Occasion?

- **Yes:** Predict "Reservation Not Honored" (4 samples)
- **No:** Refine with "Party Size" (2 samples)

3rd split: Party Size *(pure leaf)*

- **2:** Predict "Reservation Honored" (1/2 sample)
- **4:** Predict "Reservation Not Honored" (1/2 sample)

Hands-on Example (9/13)



Hands-on Example (10/13)

```julia
using DataFrames, MLJ

df = DataFrame(
    PS = [2, 4, 6, 2, 4, 8, 2, 4, 6, 2],
    DOTW = ["Weekday", "Weekend", "Weekend", "Weekend", "Weekday", "Weekend",
    ↪"Weekday", "Weekend", "Weekday", "Weekend"],
    SO = ["No", "Yes", "Yes", "No", "No", "Yes", "Yes", "No", "No", "Yes"],
    RH = ["Yes", "No", "No", "Yes", "Yes", "No", "Yes", "No", "Yes", "No"]
)
schema(df)
#=
┌───────┬──────────┬────────┐
│ names │ scitypes │ types  │
├───────┼──────────┼────────┤
│ PS    │ Count    │ Int64  │
│ DOTW  │ Textual  │ String │
│ SO    │ Textual  │ String │
│ RH    │ Textual  │ String │
└───────┴──────────┴────────┘
=#
```

Hands-on Example (11/13)

```julia

coerce!(df,
    :PS => Continuous,
    :DOTW => OrderedFactor,
    :SO => OrderedFactor,
    :RH => OrderedFactor
)
schema(df)
#=
┌───────┬──────────────────┬──────────────────────────────────┐
│ names │ scitypes         │ types                            │
├───────┼──────────────────┼──────────────────────────────────┤
│ PS    │ Continuous       │ Float64                          │
│ DOTW  │ OrderedFactor{2} │ CategoricalValue{String, UInt32} │
│ SO    │ OrderedFactor{2} │ CategoricalValue{String, UInt32} │
│ RH    │ OrderedFactor{2} │ CategoricalValue{String, UInt32} │
└───────┴──────────────────┴──────────────────────────────────┘
=#

X = select(df, Not(:RH))
```

Hands-on Example (12/13)

```julia
y = df.RH

ohe = OneHotEncoder(drop_last=true)
mach_ohe = machine(ohe, X) |> fit!
W = MLJ.transform(mach_ohe, X)
first(W, 2)
#=
2×3 DataFrame
 Row │ PS       DOTW__Weekday  SO__No
     │ Float64  Float64        Float64
─────┼────────────────────────────────
   1 │     2.0            1.0      1.0
   2 │     4.0            0.0      0.0
=#

DecisionTreeClassifier = @load DecisionTreeClassifier pkg=DecisionTree
dt = DecisionTreeClassifier() # an instance of `DecisionTreeClassifier`

mach_tree = machine(dt, W, y)
fit!(mach_tree, verbosity=2)
```

Hands-on Example (13/13)

```
mach_tree.report[:fit].print_tree(mach_tree.fitresult[1])
#=
Feature 2: "DOTW__Weekday" < 0.5 ?
├─ Feature 3: "SO__No" < 0.5 ?
    ├─ 1 : 4/4
    └─ Feature 1: "PS" < 3.0 ?
        ├─ 2 : 1/1
        └─ 1 : 1/1
└─ 2 : 4/4
=#
```

Practical Implementation

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

⟶ *Pluto* ⟶ *decision-tree-*.jl*

⟶ *Jupyter* ⟶ *decision-tree-*.ipynb*

Ensemble Approach: Random Forest (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.

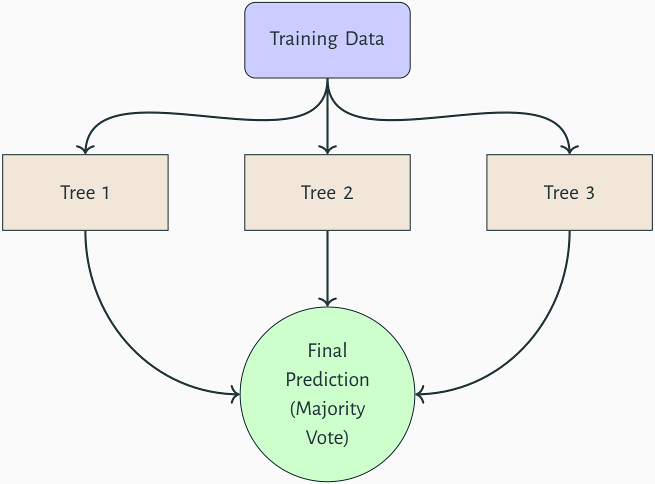
Key Features

- **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.

Ensemble Approach: Random Forest (2/2)



Practical Implementation

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

⟶ *Pluto* ⟶ *random-forest-*.jl*

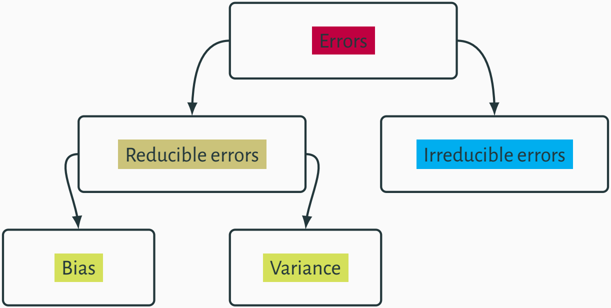
⟶ *Jupyter* ⟶ *random-forest-*.ipynb*

Summary

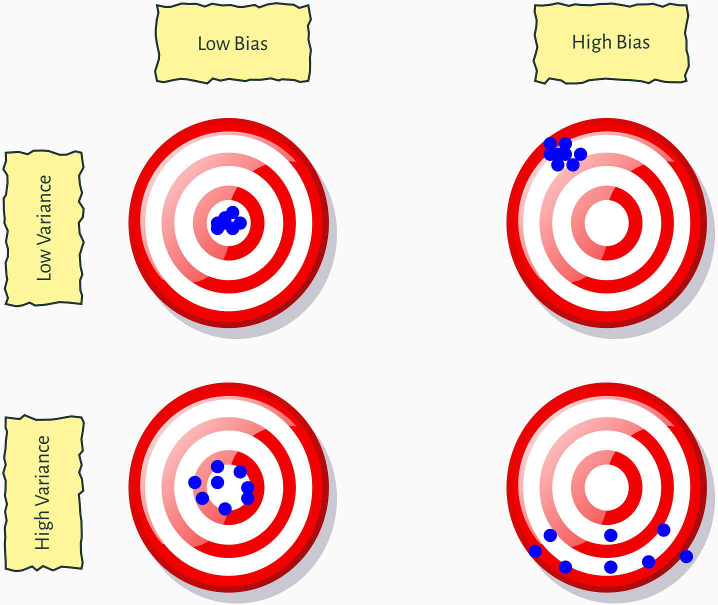
Method		Pros		Cons
Logistic Regression	▲	Probabilistic	▼	Almost linearly separable data
k-NN	▲	Fast and efficient	▼	# of neighbors *k*
			▼	Detecting outliers[2]
SVM	▲	Memory efficient	▼	Kernel's choice
	▲	Versatile	▼	Large datasets
	▲	Noise and outliers	▼	Overlapping classes
	▲	High dimension	▼	Interpretability
Naive Bayes	▲	Simplicity and efficiency	▼	Independence between features
	▲	High dimension	▼	∃ 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

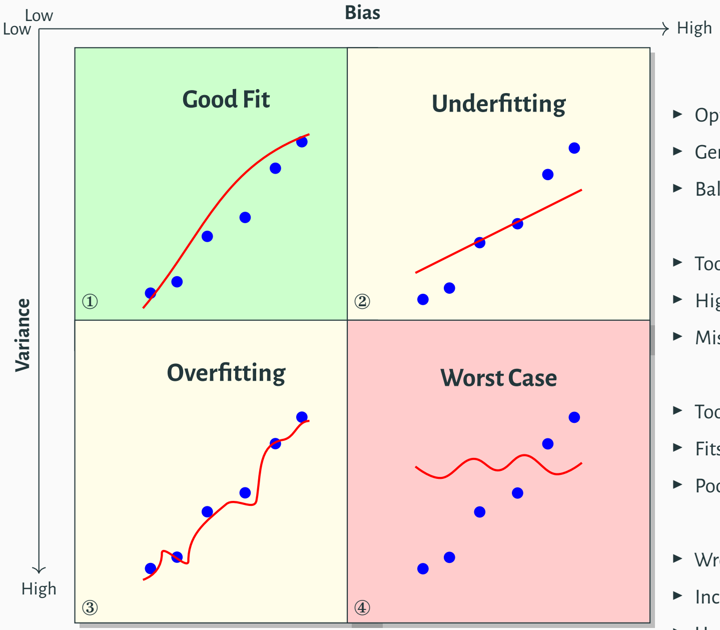
[2] Points that differ significantly from the rest of the data points.

Errors in ML


Errors
Reducible errors
Irreducible errors
Bias
Variance

Bias-Variance Tradeoff





①

- Optimal complexity
- Generalizes well
- Balanced tradeoff

②

- Too simple
- High training error
- Misses patterns

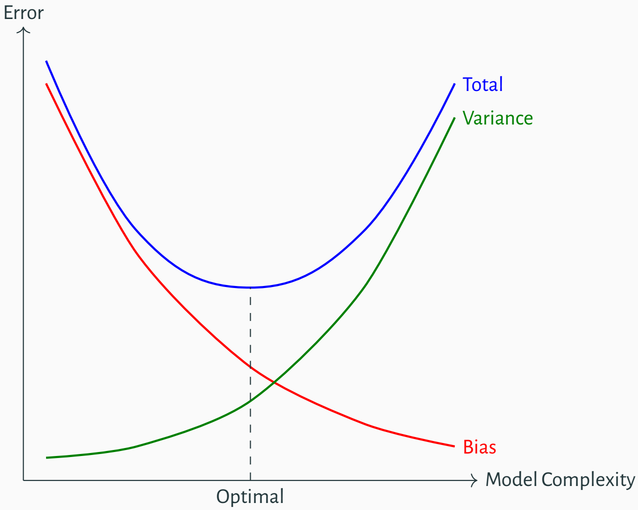
③

- Too complex
- Fits noise
- Poor generalization

④

- Wrong assumptions
- Inconsistent predictions
- Unreliable model

Error Curve vs. Model Complexity



Bias-Variance Tradeoff: Understanding & Resolving

Underfitting & High Bias	Overfitting & High Variance
Model is too simple to capture data patterns	Model memorizes noise/outliers in training data
► High training error ► Poor performance on both seen/unseen data	► Low training error but high test error ► Excellent on training data, poor on new data
► Oversimplified model architecture ► Insufficient features ► Excessive regularization	► Overly complex model ► Too many features ► Insufficient training data

Fix Underfitting (High Bias)	Fix Overfitting (High Variance)
► Increase model complexity ► Add relevant features ► Reduce regularization ► Train longer with better optimization	► Get more training data ► Use regularization (L1/L2) ► Implement early stopping ► Use cross-validation

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

Grid of Hyperparameters → Evaluate Each Combination → Optimal Configuration

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.

Validation Fold *Training Folds*

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Example: 5-Fold Cross-Validation

Benefits

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

Practical Implementation

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

⟶ *Pluto* ⟶ *hp.jl*

Pluto.jl

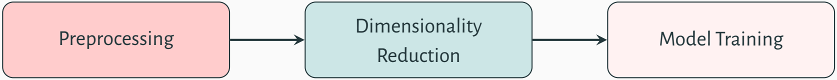
⟶ *Jupyter* ⟶ *hp.ipynb*

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 Pipelines, Grid Search, and Cross-Validation

Overview

Pipeline: Automates preprocessing and training.

Grid Search: Finds the best hyperparameters.

Cross-Validation: Ensures robust performance evaluation.

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.

Overall Methodology (1/2)

1. **Problem Definition**
   - Understand the business or research objective.
   - Define success metrics (e.g., accuracy, F1-score, ROI).
2. **Data Collection**
   - Gather raw data from databases, APIs, or files.
   - Ensure data quality, relevance, and representativeness.
3. **Data Preprocessing & Exploratory Analysis (EDA)**
   - Handle missing values, duplicates, and outliers.
   - Perform statistical and visual analysis.
4. **Feature Engineering & Selection**
   - Transform/create meaningful features (e.g., scaling, encoding).
   - Select optimal features (e.g., using PCA, feature importance).
5. **Model Selection & Training**
   - Split data into training, validation, and test sets.
   - Train multiple algorithms (e.g., regression, neural networks).
   - Apply cross-validation techniques.

Overall Methodology (2/2)

6. **Model Evaluation**
   - Test on unseen data (validation/test sets).
   - Compare performance metrics (e.g., precision, RMSE, AUC-ROC).
7. **Hyperparameter Tuning & Optimization**
   - Fine-tune models using GridSearch, RandomSearch, or Bayesian methods.
8. **Deployment & Monitoring**
   - Deploy the model (e.g., as an API, cloud service, or embedded system).
   - Continuously monitor performance and data drift.
9. **Communication & Maintenance**
   - Document methodology, results, and limitations.
   - Schedule retraining and updates as needed.

Unsupervised Learning

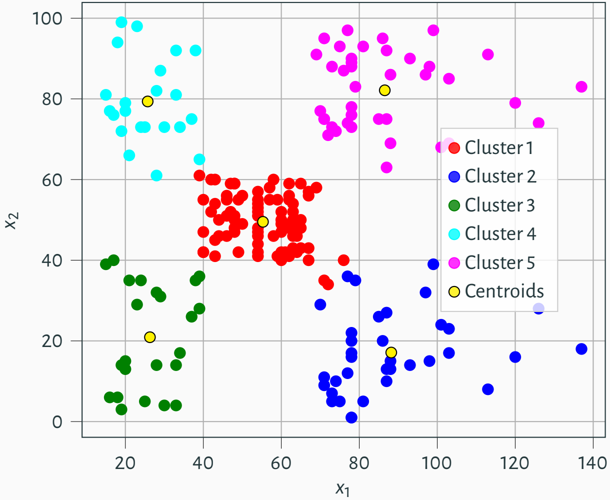
K-Means Clustering (1/3)

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 #10[3]
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.

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

Task #10[3]

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.

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

Task #11[4]
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

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

[4] Credit: Shokoufeh Mirzaei, PhD

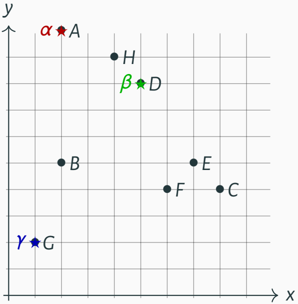
Task #11[4]

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

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



[4]Credit: Shokoufeh Mirzaei, PhD

Task #11[4]

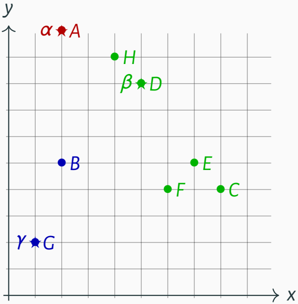
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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



[4]Credit: Shokoufeh Mirzaei, PhD

Task #11[4]

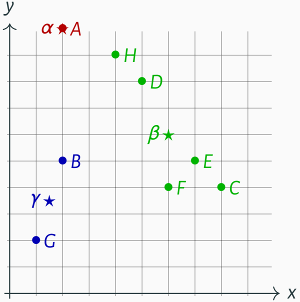
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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
	$\alpha(2,\ 10)$	$\beta(6,\ 6)$	$\gamma(1.5,\ 3.5)$	



[4]Credit: Shokoufeh Mirzaei, PhD

Task #11[4]

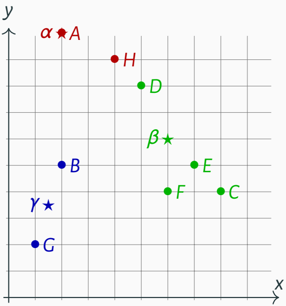
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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



[4]Credit: Shokoufeh Mirzaei, PhD

Task #11[4]
Use K-Means algorithm to cluster the following eight points intro three clusters:

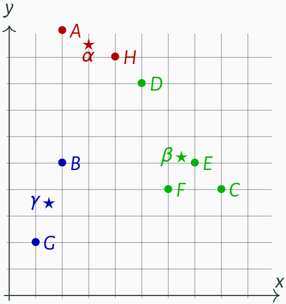
$$A(2,\ 10);\ B(2,\ 5);\ C(8,\ 4);\ D(5,\ 8);\ E(7,\ 5);\ F(6,\ 4);\ G(1,\ 2) \text{ and } H(4,\ 9).$$

- 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

$\alpha(3,\ 9.5)$ $\beta(6.5,\ 5.25)$ $\gamma(1.5,\ 3.5)$



[4]Credit: Shokoufeh Mirzaei, PhD

Task #11[4]

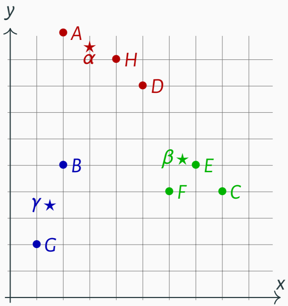
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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



[4]Credit: Shokoufeh Mirzaei, PhD

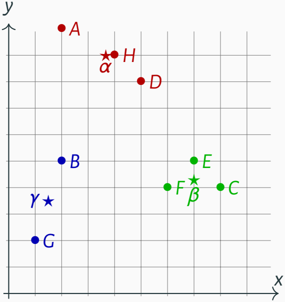
Task #11[4]
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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
	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	



[4]Credit: Shokoufeh Mirzaei, PhD

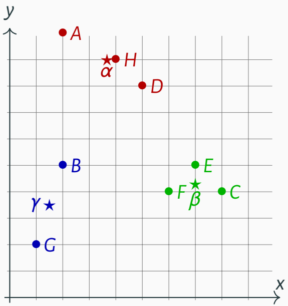
Task #11[4]
Use K-Means algorithm to cluster the following eight points intro three clusters:

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

- 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



[4]Credit: Shokoufeh Mirzaei, PhD

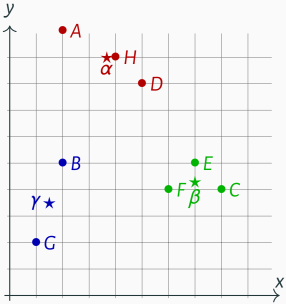
Task #11[4]
Use *K*-Means algorithm to cluster the following eight points intro three clusters:

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

- 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
	$\alpha(3.67,\ 9)$	$\beta(7,\ 4.3)$	$\gamma(1.5,\ 3.5)$	



[4]Credit: Shokoufeh Mirzaei, PhD

Practical Implementation

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

⟶ *Pluto* ⟶ *kmeans.jl*

⟶ *Jupyter* ⟶ *kmeans.ipynb*

DBSCAN: Overview

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) 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.

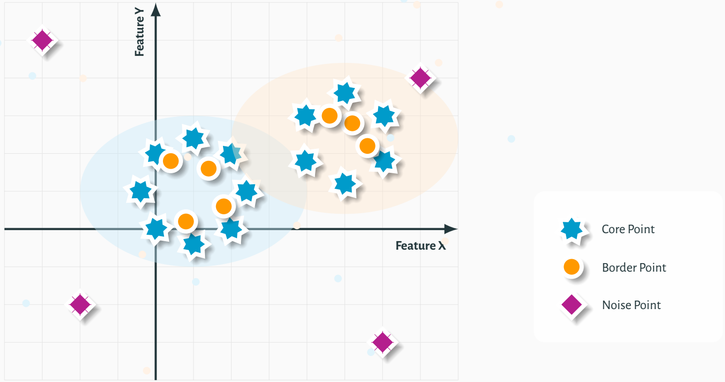
Border Point: A point within ϵ-distance of a core point but with fewer than `MinPts` neighbors.

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



- ▲ 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.

Hands-on Example (1/3)

Task #12
Consider the following synthetic 2D dataset with 5 points:

X	Y
1.0	1.0
1.1	1.1
4.0	4.0
4.1	4.1
10.0	10.0

```
    Y
    |
    |                              • (10, 10)
    |
    |            • (4.1, 4.1)
    |           • (4.0, 4.0)
    |
    |   • (1.1, 1.1)
    |  • (1.0, 1.0)
 0  +-----+-----+-----+-----+-----+---> X
    0     2     4     6     8    10
```

Apply DBSCAN with $\epsilon = 1.5$ and `MinPts` $= 2$.

Hands-on Example (2/3)

Compute Pairwise Distances

	$(1, 1)$	$(1.1, 1.1)$	$(4, 4)$	$(4.1, 4.1)$	$(10, 10)$
$(1, 1)$	0.00	0.141	4.243	4.384	12.738
$(1.1, 1.1)$	0.141	0.00	4.101	4.243	12.586
$(4, 4)$	4.243	4.101	0.00	0.141	8.485
$(4.1, 4.1)$	4.384	4.243	0.141	0.00	8.344
$(10, 10)$	12.738	12.586	8.485	8.344	0.00

Identify Core Points

A point is a **core point** if $\geq$ `MinPts` neighbors exist within ϵ:

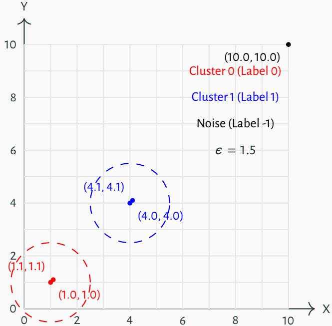
- **(1.0, 1.0)**: Neighbors = $\{(1.1, 1.1)\} \longrightarrow 1$ (<2) → *Border*
- **(1.1, 1.1)**: Neighbors = $\{(1.0, 1.0)\} \longrightarrow 1$ (<2) → *Border*
- **(4.0, 4.0)**: Neighbors = $\{(4.1, 4.1)\} \longrightarrow 1$ (<2) → *Border*
- **(4.1, 4.1)**: Neighbors = $\{(4.0, 4.0)\} \longrightarrow 1$ (<2) → *Border*
- **(10.0, 10.0)**: No neighbors within ϵ → *Noise*

Hands-on Example (3/3)

Cluster Assignment

- **Cluster 1**: {(1.0,1.0), (1.1,1.1)} (mutually reachable via ϵ)
- **Cluster 2**: {(4.0,4.0), (4.1,4.1)} (mutually reachable via ϵ)
- **Noise**: (10.0,10.0) (no nearby points)

Point (X, Y)	Label	Type
(1.0, 1.0)	0	Border
(1.1, 1.1)	0	Border
(4.0, 4.0)	1	Border
(4.1, 4.1)	1	Border
(10.0, 10.0)	-1	Noise



Practical Implementation

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

⟶ *Pluto* ⟶ *dbscan.jl*

⟶ *Jupyter* ⟶ *dbscan.ipynb*

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).

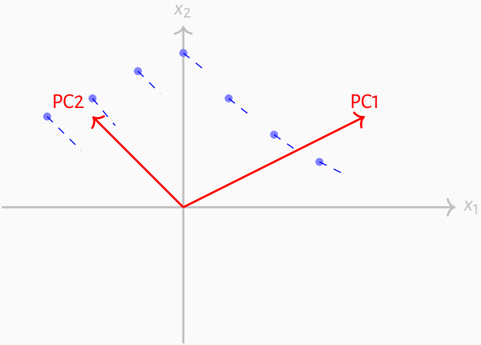
Applications

- 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.

Hands-on Example (1/8)

Step-by-Step PCA Example

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

Hands-on Example (2/8)

Step-by-Step PCA Example

Mean Normalization

$$\bar{X} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} 1.81 \\ 1.91 \end{bmatrix}, \quad \mathbf{X}_{\text{normalized}} = \mathbf{X} - \bar{X}$$

Covariance Matrix

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

Eigenvalues and Eigenvectors

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

$$\text{Eigenvectors: } \mathbf{v}_1 \approx \begin{bmatrix} 0.678 \\ 0.735 \end{bmatrix}, \quad \mathbf{v}_2 \approx \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$$

Hands-on Example (3/8)

Step-by-Step PCA Example

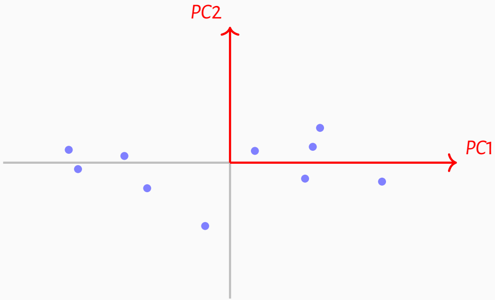
Project Data onto Principal Components

$$\mathbf{Z} = \mathbf{X}_{\text{normalized}}\mathbf{V}, \quad \therefore \quad \mathbf{Z} \approx \begin{bmatrix} 0.827 & -0.175 \\ -1.777 & 0.142 \\ 0.992 & 0.384 \\ 0.274 & 0.130 \\ 1.675 & -0.209 \\ 0.912 & 0.175 \\ -0.099 & -0.35 \\ -1.144 & 0.046 \\ -0.438 & 0.018 \\ -1.224 & -0.163 \end{bmatrix}$$

Hands-on Example (4/8)

Step-by-Step PCA Example

Visualization of Principal Components



Key Observations

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

Hands-on Example (5/8)

Step-by-Step PCA Example

```
X = [
     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 1.1; 1.5 1.6; 1.1 0.9
]

X  = 1/10 * sum(X, dims=1) # X\bar = ...
#=
1×2 Matrix{Float64}:
 1.81  1.91
=#

Xnorm = X .- X  # X .- X\bar
#=
10×2 Matrix{Float64}:
  0.69   0.49
 -1.31  -1.21
  0.39   0.99
  0.09   0.29
  1.29   1.09
```

Hands-on Example (6/8)

Step-by-Step PCA Example

```
  0.49   0.79
  0.19  -0.31
 -0.81  -0.81
 -0.31  -0.31
 -0.71  -1.01
=#

C = 1/(9)*(Xnorm'*Xnorm) # C = 1/(n-1) * ...
#=
2×2 Matrix{Float64}:
 0.616556  0.615444
 0.615444  0.716556
=#

using LinearAlgebra
λ, V = eigen(C)
#=
Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
```

Hands-on Example (7/8)

Step-by-Step PCA Example

```
2-element Vector{Float64}:
 0.04908339893832736
 1.2840277121727837
vectors:
2×2 Matrix{Float64}:
 -0.735179  0.677873
  0.677873  0.735179
=#

Z = Xnorm * V
#=
10×2 Matrix{Float64}:
 -0.175115    0.82797
  0.142857   -1.77758
  0.384375    0.992197
  0.130417    0.27421
 -0.209498    1.6758
  0.175282    0.912949
 -0.349825   -0.0991094
```

Hands-on Example (8/8)

Step-by-Step PCA Example

```
   0.0464173  -1.14457
   0.0177646  -0.438046
  -0.162675   -1.22382
=#
```

Practical Implementation

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

⟶ *Pluto* ⟶ *pca.jl*

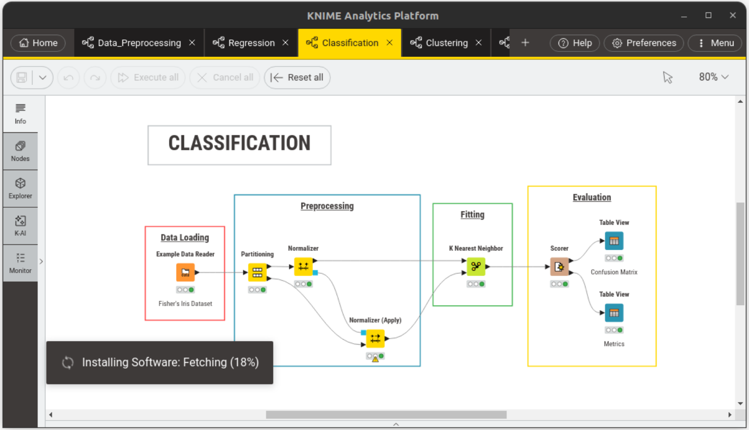
⟶ *Jupyter* ⟶ *pca.ipynb*

Complementary Lab. Project

On the day of assignment, you will be informed about the **dataset to use** *(source: database)*, **specific features to retain**, and the **machine learning model to implement**. You will be asked to:

① ETL*(Extract, Transform, Load)* data from a database;

② Build and evaluate the **ML** model *(pipeline, featurization, split, etc.)*.



ML Landscape through Quizzes

Knowledge Check

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code

JLAI2

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.

[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.