

# Demystifying Artificial Intelligence Sorcery

(Part 2: Machine Learning)<sup>a</sup>

---

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

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

---

<sup>a</sup>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.

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

## **An overview**

---

# GLOBAL DATA TRAFFIC



Update on the internet in real time is available [here](#).

## LITERATURE REVIEW (1/3)



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

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

## LITERATURE REVIEW (2/3)

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

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

## LITERATURE REVIEW (3/3)

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

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



# DEBRIEF

**Arthur Samuel (1959)**

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

**Tom Mitchell (1998)**

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

**Task #1**

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

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

## DEBRIEF

### Arthur Samuel (1959)

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

### Tom Mitchell (1998)

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

### Task #1

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

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

## DEBRIEF

### Arthur Samuel (1959)

Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

### Tom Mitchell (1998)

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

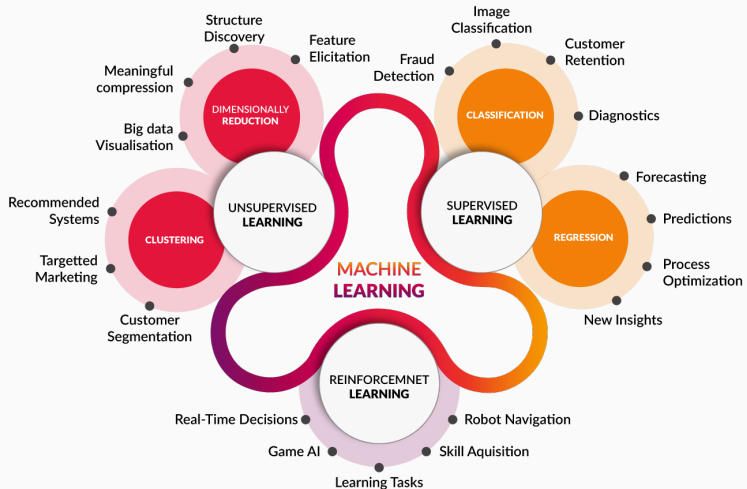
### Task #1

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

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

## OVERALL METHODOLOGY

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



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



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

# REGRESSION | CLASSIFICATION | CLUSTERING



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



**REMINDER**



# PROGRAMMING LANGUAGE



[julialang.org/](https://julialang.org/)

A screenshot of a terminal window titled "~julia/julia". The prompt is "+ julia". The output shows a stylized "julia" logo made of colored circles and lines, followed by the text: "Documentation: https://docs.julialang.org", "Type '?' for help, ']' for Pkg help.", and "Version 1.6.3 (2021-09-23)". Below this, the user enters "julia> println(\"Hello, World!\")" and the output is "Hello, World!". The prompt "julia>" is shown again at the bottom.

```
+ julia

Documentation: https://docs.julialang.org
Type '?' for help, ']' for Pkg help.
Version 1.6.3 (2021-09-23)

julia> println("Hello, World!")
Hello, World!

julia>
```

## DEVELOPMENT ENVIRONMENTS



**Pluto.jl**



▲ \$ docker compose up

▼ \$ docker compose down



# JULIA IN A NUTSHELL

- ▲ Fast
- ▲ Dynamic
- ▲ Reproducible
- ▲ Composable
- ▲ General
- ▲ Open Source



# JULIA MICRO-BENCHMARKS (1/2)



<https://julialang.org/benchmarks>



## JULIA MICRO-BENCHMARKS (2/2)

### Geometric Means 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





# SOURCE CONTROL MANAGEMENT (SCM)

The screenshot shows the GitHub repository page for 'a-mhamdi/jlai'. The repository is public and has 2 stars and 3 forks. The main branch is 'main'. The repository contains a README.md file, a LICENSE file, and a .gitignore file. The README.md file is titled 'Fuzzy Logic, Machine Learning and Deep Learning with Julia'. The repository also has a 'CODE' button, an 'Issues' button, a 'Pull requests' button, an 'Actions' button, a 'Projects' button, a 'Wiki' button, a 'Security' button, an 'Insights' button, and a 'Settings' button. The repository is described as 'An Introduction to Artificial Intelligence with Julia' and includes tags for 'flux', 'machine-learning', 'docker-image', 'fuzzy-logic', 'julia', and 'mjl'.

**Repository: a-mhamdi / jlai** (Public)

Buttons: Unpin, Unwatch (2), Fork (3), Star (2)

Navigation: <> Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings

Branches: main (1 branch), tags (0 tags)

Buttons: Go to file, Add file, <> Code

**Files:**

File	Description	Commit	Time
.github/workflows	Update docker-image.yml	fde8fca	yesterday
Codes	vgg and resnet transfer learning		yesterday
Docker	rm Docker cheat sheet		3 days ago
Exams	exam w/ answers		4 days ago
Slides-Labs	change colors		yesterday
.gitignore	change colors		yesterday
LICENSE	Initial commit		4 months ago
README.md	update Docker README file		2 weeks ago

**About**

An Introduction to Artificial Intelligence with Julia

Tags: flux, machine-learning, docker-image, fuzzy-logic, julialang, mjl

Readme, MIT license, 2 stars, 2 watching, 3 forks

**Languages**

Language	Percentage
Julia	94.3%
Dockerfile	3.4%
Batchfile	2.1%
TeX	0.2%

**README.md**

**Fuzzy Logic, Machine Learning and Deep Learning with Julia**

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



# CONTINUOUS INTEGRATION (CI)


The screenshot shows a web browser window displaying the Docker Hub repository for 'abmhamdi/jlai'. The browser's address bar shows 'hub.docker.com/r/abmhamdi/jlai'. The Docker Hub interface includes a search bar, navigation links for 'Explore', 'Repositories', 'Organizations', and 'Help', and a user profile for 'abmhamdi'. The repository page features a blue cube icon, the name 'abmhamdi/jlai' with a star, and a 'Manage Repository' button. It indicates the repository was updated 21 hours ago by 'abmhamdi' and is associated with 'Artificial Intelligence Labs @ ISETBZ'. The 'Overview' tab is selected, showing a description: 'Fuzzy Logic, Machine Learning and Deep Learning with Julia'. It states that the repository contains slides, labs, and code examples for using Julia to implement artificial intelligence algorithms, running on a Docker image for a consistent and reproducible environment. A status bar shows 'jlai-ci' as 'passing', with 'version latest', 'docker pulls 22', and 'docker stars 0'. A 'Docker Pull Command' box displays the command 'docker pull abmhamdi/jlai'. At the bottom, it instructs users to pull the Docker image by running a command.

abmhamdi/jlai - Docker Im x

hub.docker.com/r/abmhamdi/jlai

docker hub Search Docker Hub Explore Repositories Organizations Help Upgrade abmhamdi

Explore abmhamdi/jlai

 **abmhamdi/jlai** ☆

By [abmhamdi](#) • Updated 21 hours ago

Artificial Intelligence Labs @ ISETBZ

Image

Manage Repository

Pulls 22

Overview Tags

**Fuzzy Logic, Machine Learning and Deep Learning with Julia**

This repository contains slides, labs and code examples for using `Julia` to implement some **artificial intelligence** related algorithms. Codes run on top of a `Docker` image, ensuring a consistent and reproducible environment.

`jlai-ci` passing version `latest` docker pulls 22 docker stars 0

To run the code, you will need to first pull the `Docker` image by running the following command:

**Docker Pull Command**

```
docker pull abmhamdi/jl...
```

<https://hub.docker.com/r/abmhamdi/jlai>

# A MACHINE LEARNING FRAMEWORK FOR JULIA



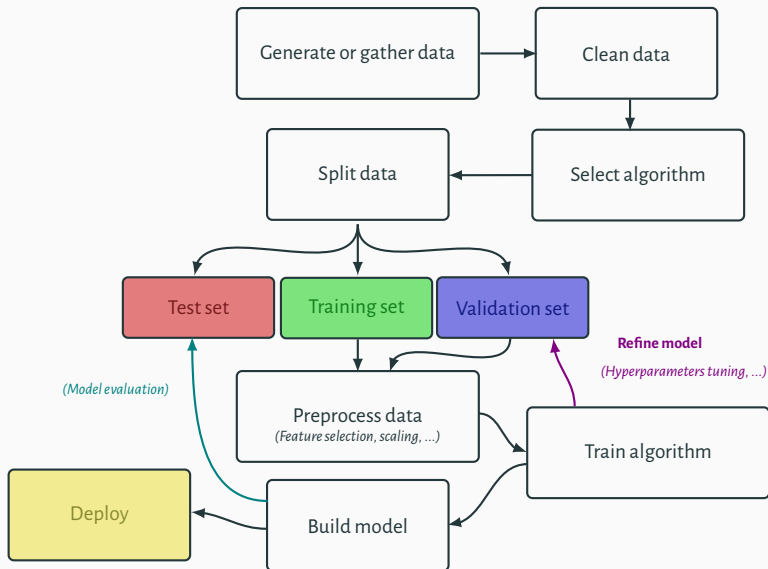
<https://docs.juliahub.com/MLJ/>



## 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

---

$$X \triangleq \frac{X - \text{minimum}(X)}{\text{maximum}(X) - \text{minimum}(X)}$$

▲ No assumption on data distribution

### Standardization (*Standardizer*)

---

$$X \triangleq \frac{X - \mu}{\sigma}$$

▲ More recommended when following normal distribution

---

# DATA PREPROCESSING TEMPLATE

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → ml-workflows.jl





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*):

$$F = kx \quad (1)$$

Restoring force is proportional to displacement.

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

$x_i$	$x_1$	$\dots$	$x_p$	$\dots$	$x_n$
$F^{\text{meas}}_i$	$F^{\text{meas}}_1$	$\dots$	$F^{\text{meas}}_p$	$\dots$	$F^{\text{meas}}_n$

$$\begin{aligned} F^{\text{meas}}_i &= F_i + \varepsilon_i \\ &= kx_i + \varepsilon_i, \end{aligned} \quad (2)$$

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:

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

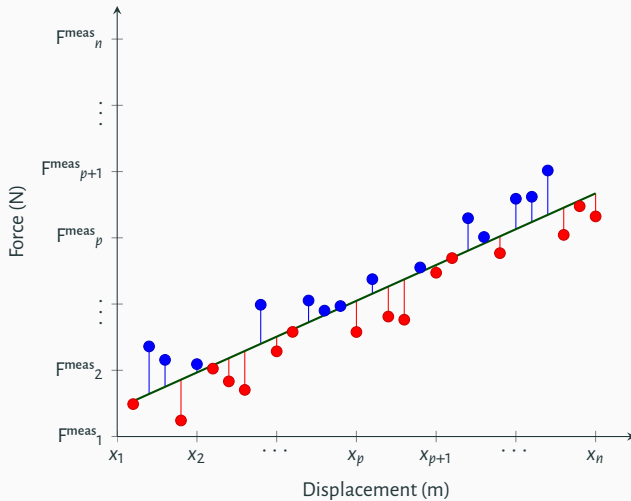


$$\frac{\partial \mathcal{J}}{\partial k} = 0 \quad (3)$$

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

$$\sum_{i=1}^n (F^{\text{meas}}_i - kx_i) \sum_{i=1}^n x_i = 0$$

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



## SIMPLE LINEAR REGRESSION

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → simple-regression-\*.jl



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

$$y = y_0 + v_0 t \quad (4)$$

**Table 2:** Measurements of position  $y$

$t_k$	$t_1$	$\dots$	$t_p$	$\dots$	$t_n$
$y_k^{\text{meas}}$	$y_1^{\text{meas}}$	$\dots$	$y_p^{\text{meas}}$	$\dots$	$y_n^{\text{meas}}$

$$\begin{aligned} y_k^{\text{meas}} &= y_k + \varepsilon_k \\ &= y_0 + v_0 t_k + \varepsilon_k, \end{aligned} \quad (5)$$

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

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

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

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

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

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

## MULTIPLE LINEAR REGRESSION

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>  
→ Codes → Julia → Part-2 → multivariable-regression.jl



Consider the following multivariable equation:

$$y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m \quad (7)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

$$\hat{\theta} = (X^T X)^{-1} X^T y$$



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

#### ▼ Redundant Features

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

#### ▼ Too many features

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

- ▲ Delete some features
- ▲ Add extra observations

▲ Use regularization:

$\lambda \sum_{i=2}^m |\theta_i|$

$\frac{1}{2} \lambda \sum_{i=2}^m \theta_i^2$

$r\lambda \sum_{i=2}^m |\theta_i| + \frac{(1-r)}{2} \lambda \sum_{i=2}^m \theta_i^2$

RIDGE
LASSO
ELASTIC NET

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

# POLYNOMIAL REGRESSION

CODE SNIPPET

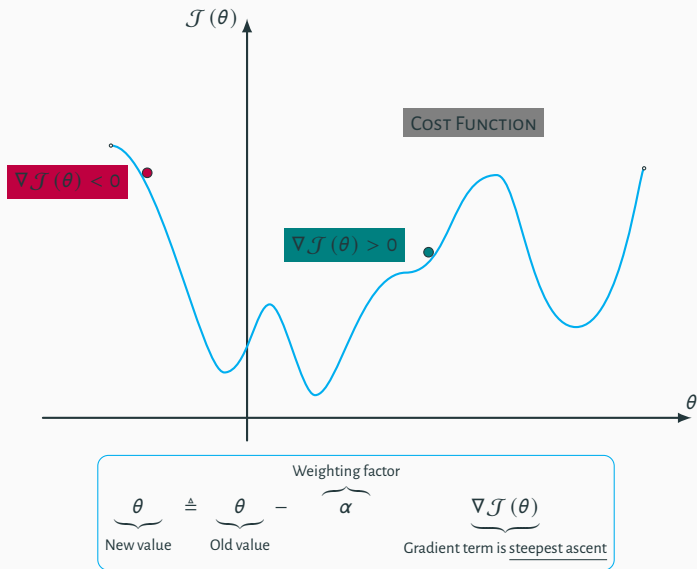


Code is available at <https://github.com/a-mhamdi/jlai/>

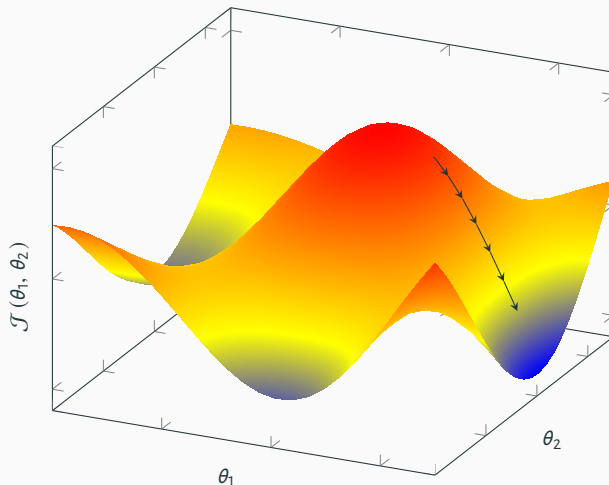
→ Codes → Julia → Part-2 → *polynomial-regression.jl*



## GRADIENT DESCENT



# GRADIENT DESCENT



- ① Start with some random values of  $\theta_1$  and  $\theta_2$
- ② Keep changing  $\theta_1$  and  $\theta_2$  to reduce  $\mathcal{J}(\theta_1, \theta_2)$  until we hopefully end up at minimum

## GRADIENT DESCENT

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{LEARNING RATE}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Recall that

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

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

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

⋮

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

## GRADIENT DESCENT

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{LEARNING RATE}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Recall that with  $L_2$  regularization term

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

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

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

$$\vdots$$

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

**Task #2**

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

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

The linear regression model  $y = \theta_0 + \theta_1 x$  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:

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



## ① Normal Equation

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

## ② Stochastic Gradient Descent

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

## GD IN ACTION

CODE SNIPPET

```

1  X = [1 0; 1 25; 1 50; 1 75; 1 100] # Features
2  y = [14, 38, 54, 76, 95] # Target
3
4  alpha, n, theta = 0.01, 5, [1; .5]
5  J = []
6  for k in 1:5
7      h_th = X[k, :]' * theta
8      println("h_th is $(h_th)")
9      cost = (y[k] - h_th)^2
10     push!(J, cost);
11     theta += alpha * (y[k] - h_th) * X[k, :]
12     println("theta is $(theta)")
13 end

```



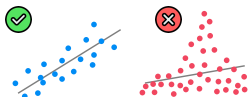
Code is available at <https://github.com/a-mhamdi/jlai>  
 → Codes → Julia → Part-2 → gradient-descent.jl

# Assumptions of Linear Regression



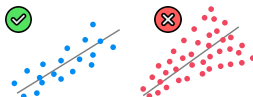
## 1. Linearity

(Linear relationship between Y and each X)



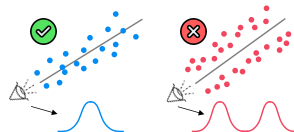
## 2. Homoscedasticity

(Equal variance)



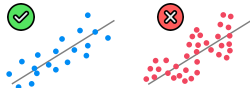
## 3. Multivariate Normality

(Normality of error distribution)



## 4. Independence

(of observations. Includes "no autocorrelation")



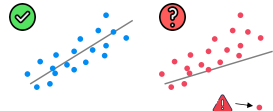
## 5. Lack of Multicollinearity

(Predictors are not correlated with each other)



## 6. The Outlier Check

(This is not an assumption, but an "extra")



## EVALUATION METRICS (1/2)

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

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

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

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

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

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

## EVALUATION METRICS (2/2)

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

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

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

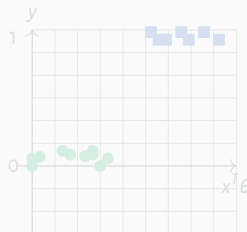
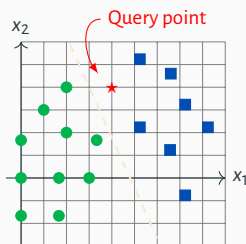
👎 0 indicates that the model explains **NONE** of the variance in the dependent variable

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

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

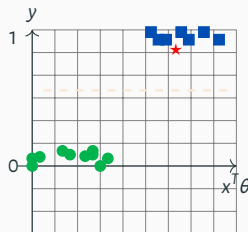
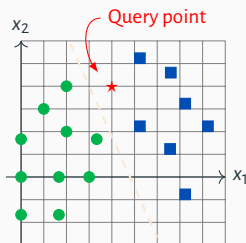
# INTRODUCTION

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



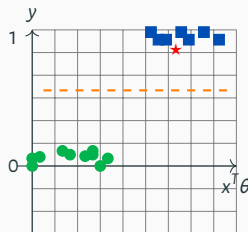
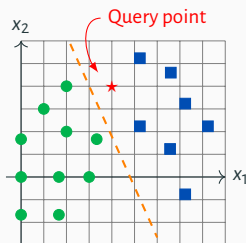
# INTRODUCTION

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



# INTRODUCTION

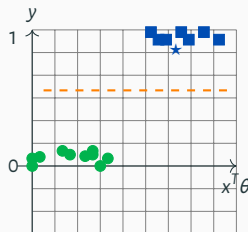
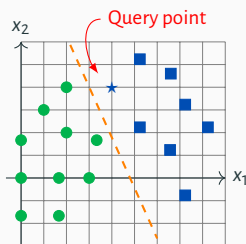
Classification is a type of supervised machine learning algorithm. A model is trained on a set of *labeled data*, where each data point is associated with a known class or category. The goal of the algorithm is to learn the relationship between the *input features*  $x$  and the corresponding *output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



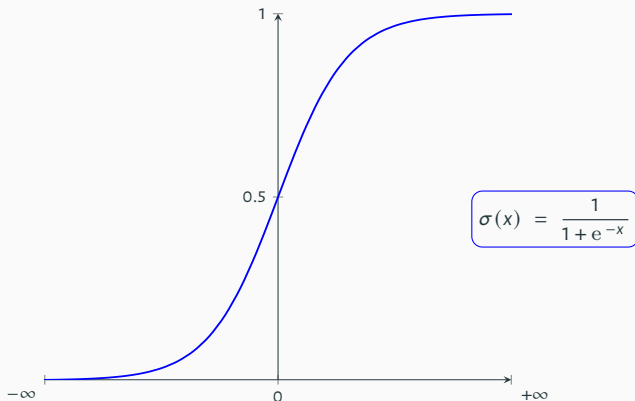


# INTRODUCTION

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



## LOGISTIC OR S-SHAPED FUNCTION $\sigma$



▲  $\sigma$  squashes range of distance from  $]-\infty, +\infty[$  to  $[0, 1]$

▲  $\sigma$  is differentiable and easy to compute:  $\dot{\sigma} = \sigma \times (1 - \sigma)$

## DECISION BOUNDARY

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

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

### Hypothesis

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

For some given  $x_k$

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

### Cost function

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

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

# GRADIENT DESCENT

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{LEARNING RATE}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

Generalizing  $\mathcal{J}$  yields:

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

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

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

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

$\vdots$

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

## GRADIENT DESCENT

$$\theta_i \triangleq \theta_i - \underbrace{\alpha}_{\text{LEARNING RATE}} \frac{\partial \mathcal{J}}{\partial \theta_i}$$

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

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

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

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

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

$$\vdots$$

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

# LOGISTIC REGRESSION

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → logistic-regression.jl



# CONFUSION MATRIX

		Actual	
		Positive	Negative
Predicted	Positive	<b>TP</b>	<b>FP</b>
	Negative	<b>FN</b>	<b>TN</b>

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

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

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

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

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

		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

$$Accuracy = 0.932$$

$$Precision = 0.941$$

$$Recall = 0.882$$

$$f1 - score = 0.910$$



## EVALUATION METRICS

**ACCURACY**

**PRECISION**

**RECALL**

**f1-SCORE**

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

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

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

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

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

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

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

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

## EVALUATION METRICS

### FOLLOW UP



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

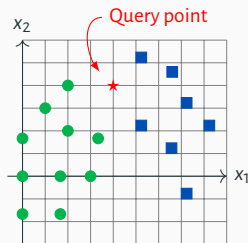
**Case #1:** Prioritize *Precision* over *Recall*, e.g.,  $\beta = 0.5$

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

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

- ▶ Detection of life threatening diseases like cancer
- ▶ Fraud detection

## k-NEAREST NEIGHBORS (1/6)



Minkowski distance

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

Manhattan distance (p=1)

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

Euclidean distance (p=2)

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

## $k$ -NEAREST NEIGHBORS (2/6)

► Evelyn Fix and Joseph Hodges, 1951

► Thomas Cover, 1966

---

### Algorithm 1 Summary Construction

---

1: **procedure** HOW DOES  $k$ -NN WORK? (Finding Nearest Neighbors)

**Input:** A query point;

**Output:** Assign a class label to that point.

2:     Define how many neighbors will be checked to classify the specific query point;

3:     Compute the distance  $d(x; y)$  of the query point to other data points;

4:     Count the number of the data points in each category;

5:     Assign the query point to the class with most frequent neighbors.

6: **end procedure**

---

## k-NEAREST NEIGHBORS (3/6)

### Task #3

Let be the following coordinate points:

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

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

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

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

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

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

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

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

## $k$ -NEAREST NEIGHBORS (4/6)

### Task #4<sup>1</sup>

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

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

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

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

## k-NEAREST NEIGHBORS (5/6)

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

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

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

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

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

$$S(\text{Orange}) = \frac{1}{d^2(U; F_3)} = 0.5$$

## *k*-NEAREST NEIGHBORS (6/6)

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

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

---

<sup>1</sup>From Prof. Winston's book





Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → knn.jl



## RULE OF THUMB TO CHOOSE $k$

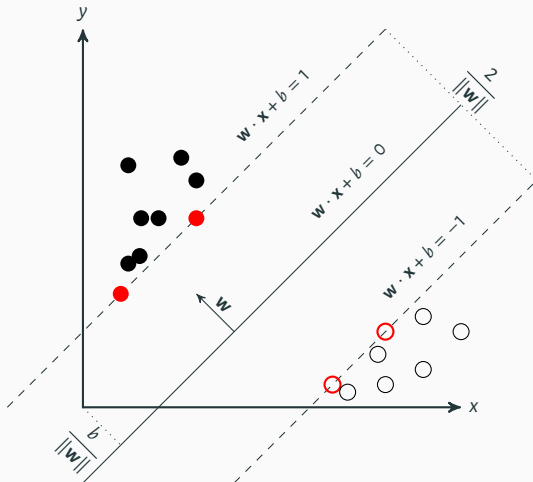
$k$  is **even** if the number of classes is odd

$k$  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 MACHINE (SVM) (1/2)



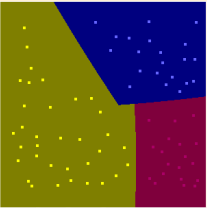
# SUPPORT VECTOR MACHINE (SVM) (2/2)

LIBSVM -- A Library for Support Vector Machines

Please read the [COPYRIGHT](#) notice before using LIBSVM.

### Graphic Interface

Here is a simple applet demonstrating SVM classification and regression.  
Click on the drawing area and use "Change" to change class of data. Then use "Run" to see the results.



Change Run Clear

Examples of options: -s 0 -c 10 -t 1 -g 1 -r 1 -d 3  
Classify a binary data with polynomial kernel  $(u^v+1)^3$  and  $C = 10$

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

# SVM FOR CLASSIFICATION

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → svc.jl

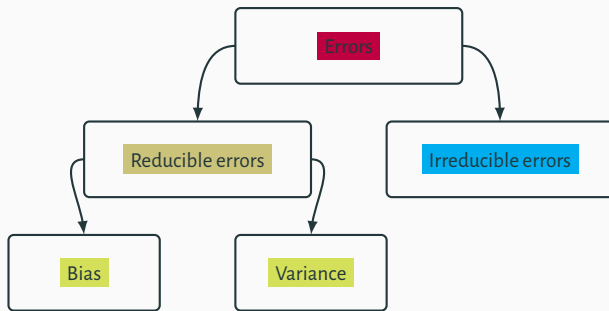


## SUMMARY

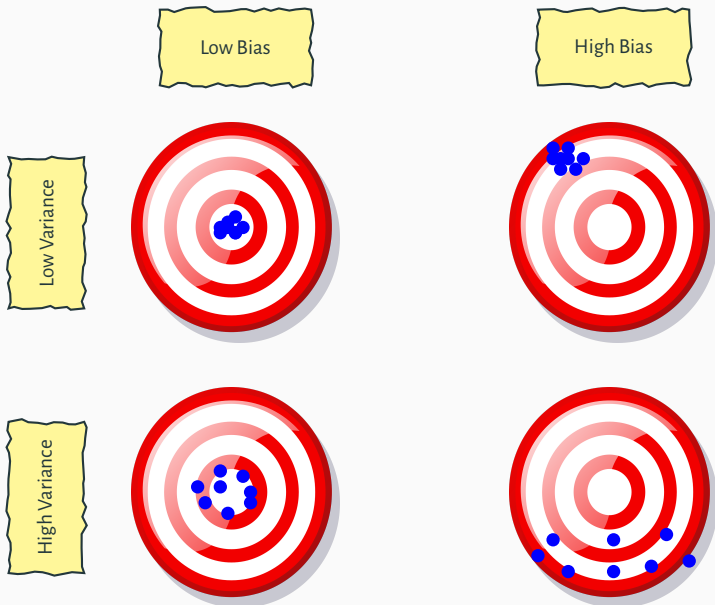
Method		Pros		Cons
<i>Logistic Regression</i>	▲	Probabilistic	▼	Almost linearly separable data
<i>k-NN</i>	▲	Fast and efficient	▼	# of neighbors $k$
			▼	Detecting outliers <sup>2</sup>
<i>SVM</i>	▲	Memory efficient	▼	Kernel's choice
	▲	Versatile	▼	Large datasets
	▲	Noise and outliers	▼	Overlapping classes
	▲	High dimension	▼	Interpretability
<i>Naive Bayes</i>	▲	Simplicity and efficiency	▼	Independence between features
	▲	High dimension	▼	∃ of irrelevant features
<i>Decision Tree</i>	▲	Interpretability	▼	Overfitting
	▲	Numerical and categorical data	▼	Unstable
	▲	Robust to outliers	▼	Continuous variables
	▲	High accuracy	▼	# of input features
<i>Random Forest</i>	▲	Less prone to overfitting	▼	Computation
	▲	High dimension	▼	Interpretability

<sup>2</sup>Points that differ significantly from the rest of the data points.

## ERRORS IN ML



## BIAS-VARIANCE TRADEOFF





## Unsupervised Learning

---

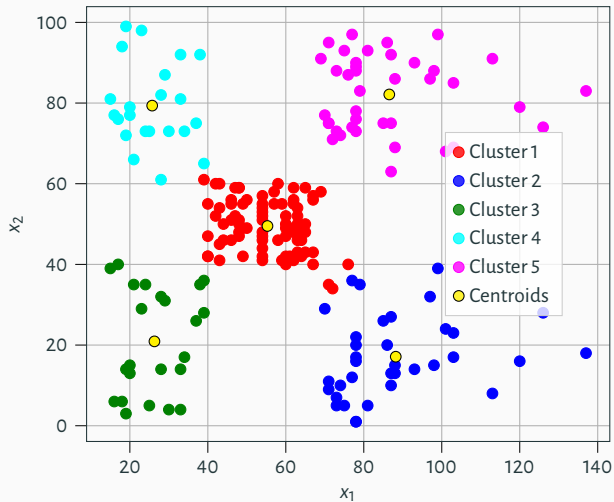
## K-MEANS CLUSTERING (1/3)

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

---

1: **procedure** HOW DOES K-MEANS WORK? (Discovering similarities)

**Input:** Unlabeled data sets;

**Output:** Grouping into clusters.

2:     Define how many clusters will be used to group the data sets;

3:     Initialize all the coordinates of the  $k$  cluster centers

4:     **repeat**

5:         Assign each point to its nearest cluster;

6:         Update the centroids coordinates;

7:     **until** No changes to the centers of the clusters

8:     Assign new cases to one of the clusters

9:     **end procedure**

---

**Task #5<sup>3</sup>**

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.

---

<sup>3</sup>From 'Machine Learning' course on 'Coursera'

**Task #5<sup>3</sup>**

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.

---

<sup>3</sup>From 'Machine Learning' course on 'Coursera'

**Task #6<sup>4</sup>**

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

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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

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

---

<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

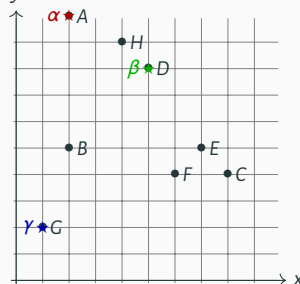
**Task #6<sup>4</sup>**

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

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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

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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD



**Task #6<sup>4</sup>**

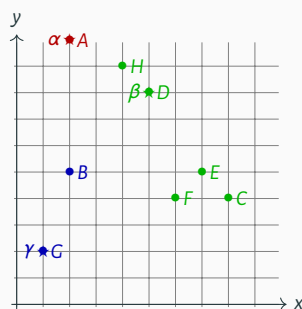
Use K-Means algorithm to cluster the following eight points into three clusters:

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

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

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

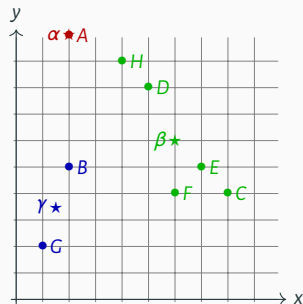
- Initial cluster centers are:  $\alpha(2, 10)$ ;  $\beta(5, 8)$  and  $\gamma(1, 2)$
- The distance between two points:  $M(x_m, y_m)$  and  $N(x_n, y_n)$  is defined as

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

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

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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

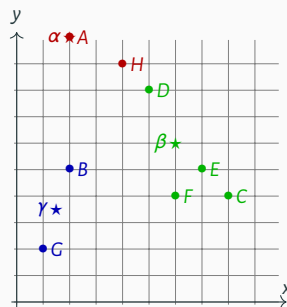
Use K-Means algorithm to cluster the following eight points into three clusters:

A(2, 10); B(2, 5); C(8, 4); D(5, 8); E(7, 5); F(6, 4); G(1, 2) 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	$\alpha(2, 10)$	$\beta(5, 8)$	$\gamma(1.5, 3.5)$	#
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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

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

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

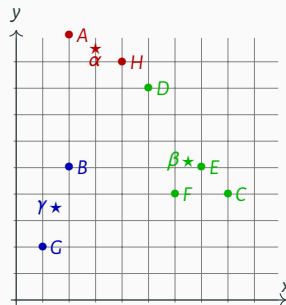
- Initial cluster centers are:  $\alpha(2, 10)$ ;  $\beta(5, 8)$  and  $\gamma(1, 2)$
- The distance between two points:  $M(x_m, y_m)$  and  $N(x_n, y_n)$  is defined as

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

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

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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

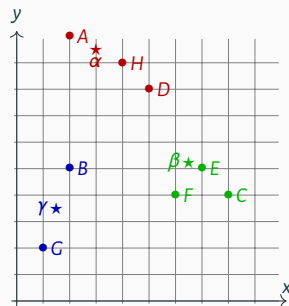
Use K-Means algorithm to cluster the following eight points into three clusters:

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

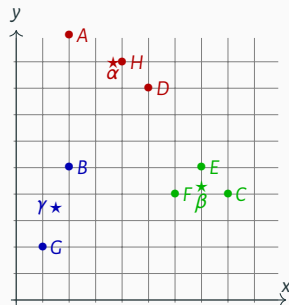
Use K-Means algorithm to cluster the following eight points into three clusters:

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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
<div> <math>\alpha(3.67, 9)</math> <math>\beta(7, 4.3)</math> <math>\gamma(1.5, 3.5)</math> </div>				



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

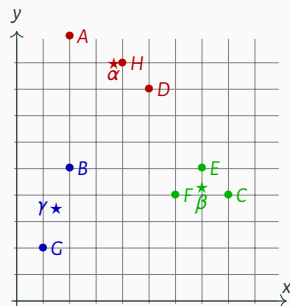
Use K-Means algorithm to cluster the following eight points into three clusters:

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD

**Task #6<sup>4</sup>**

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

$A(2, 10)$ ;  $B(2, 5)$ ;  $C(8, 4)$ ;  $D(5, 8)$ ;  $E(7, 5)$ ;  $F(6, 4)$ ;  $G(1, 2)$  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

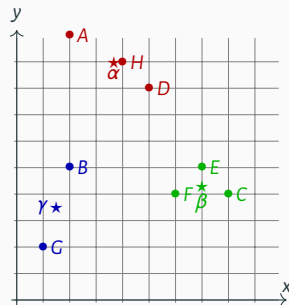
$$d(M; N) = |x_m - x_n| + |y_m - y_n|$$

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

$\alpha(3.67, 9)$

$\beta(7, 4.3)$

$\gamma(1.5, 3.5)$



<sup>4</sup>Credit: Shokoufeh Mirzaei, PhD



# K-MEANS

CODE SNIPPET



Code is available at <https://github.com/a-mhamdi/jlai/>

→ Codes → Julia → Part-2 → *kmeans.jl*

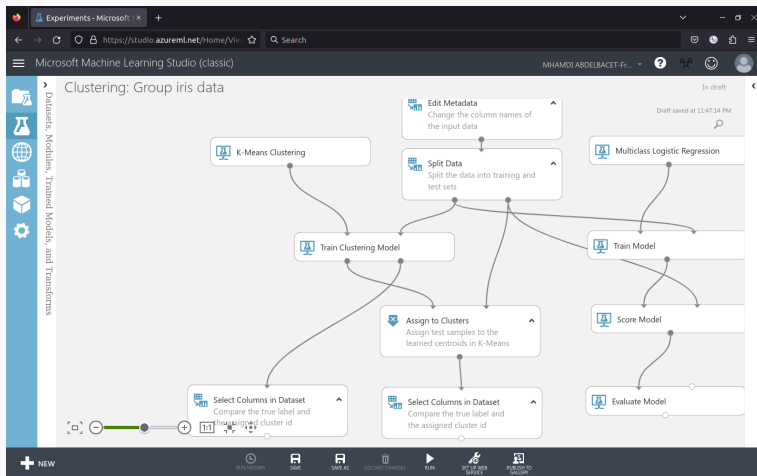


## **Complementary Lab. Project**

---

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

- ① conduct the experiment successfully (*pipeline, featurization, split, etc.*);
- ② deploy a fully functional web service app that meets the given specifications.



<https://studio.azureml.net/>

## **ML Landscape through Quizzes**

---

## KNOWLEDGE CHECK



- 1 Go to [wooclap.com](https://wooclap.com)
- 2 Enter the event code in the top banner

Event code  
**JLAI2**

<https://app.wooclap.com/JLAI2>

## LINK BUNDLE

<https://karpathy.ai/>

<https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>

<http://yann.lecun.com/>

<https://www.ibm.com/downloads/cas/GB8ZMQZ3>

<https://www.hackingnote.com/>

<https://stanford.edu/shervine/teaching/>

<https://machinelearningmastery.com/>

## FURTHER READING (1/2)

### 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/CB09780511973000.017.
- [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.

## FURTHER READING (2/2)

- [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.