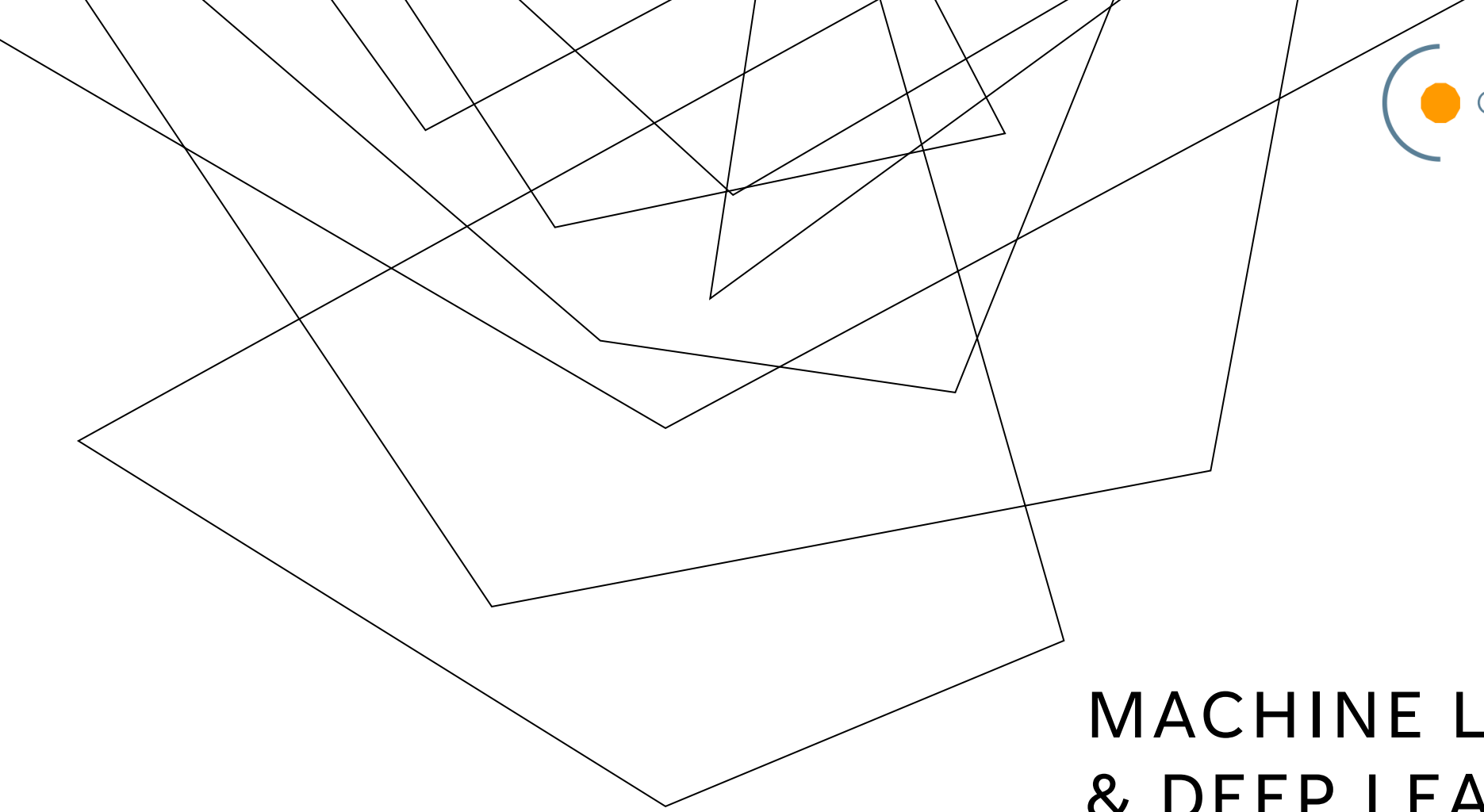


centralelille

ufr3s faculté d'ingénierie et management de la santé

Université de Lille

MACHINE LEARNING & DEEP LEARNING COURSE – MIAS M2

Idriss JAIRI

Idriss.jairi@univ-lille.fr

COURSE PLAN

Session 1: Classical Machine Learning

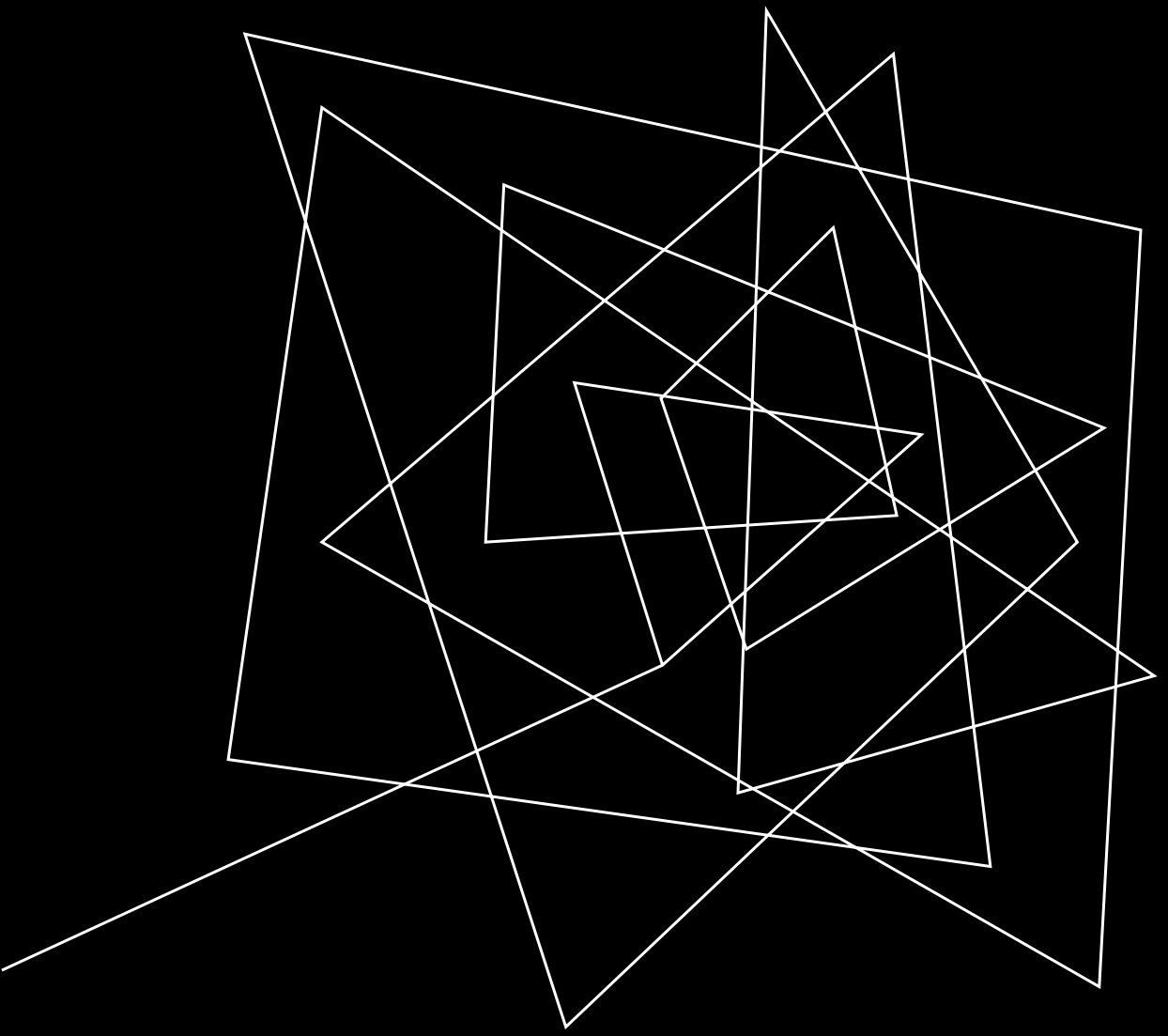
Session 2: Deep Learning

Session 3: Deep Learning for Computer Vision

Session 4: YOLO for Object Detection

Session 5: U-NET for Biomedical Image Segmentation

Session 6: Reinforcement Learning and Granular Computing



SESSION 1: MACHINE LEARNING

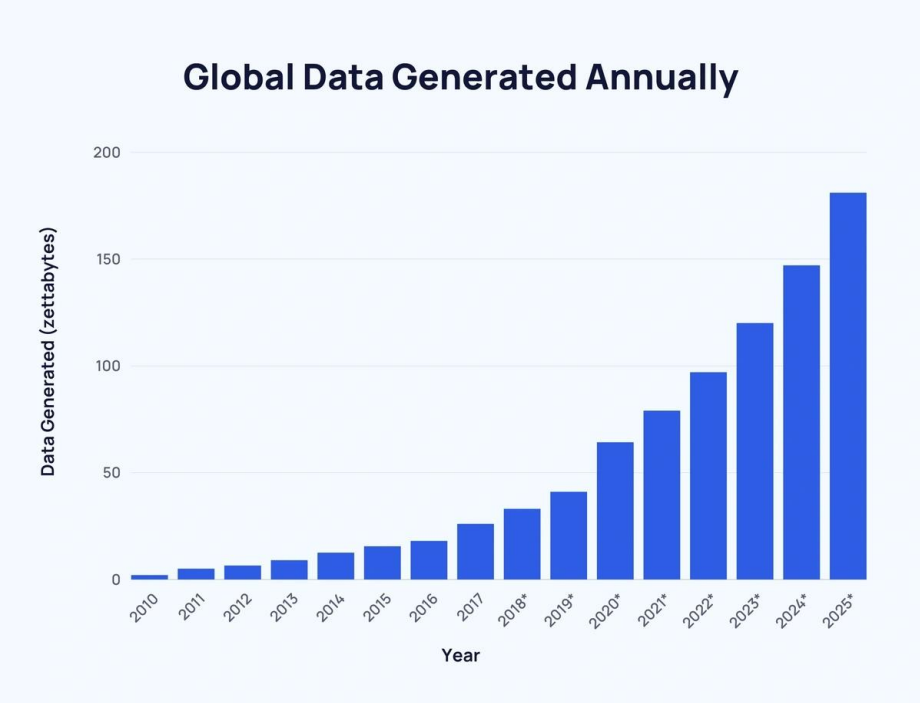
Introduction to AI, Supervised Learning, Unsupervised Learning

Introduction: Why has Artificial Intelligence (AI) become so popular?

Big tech companies invest a lot of effort and money in AI

Scientific research and impressive results

Introduction: Why has Artificial Intelligence (AI) become so popular? <Increased volumes of data>

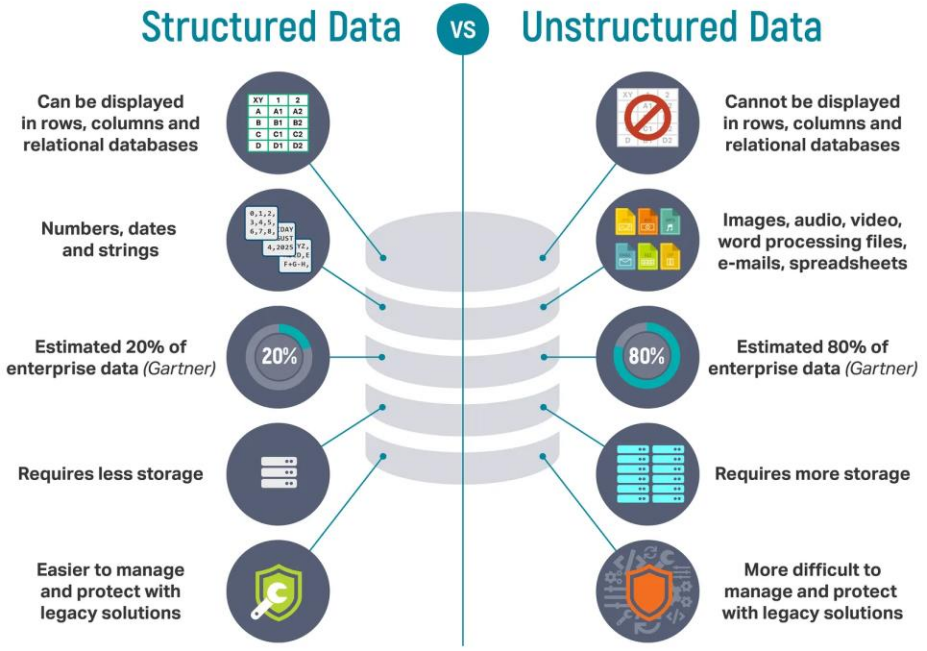


Source: https://explodingtopics.com/blog/data-generated-per-day

Note: 1 Zettabyte = 10^{12} Gigabyte

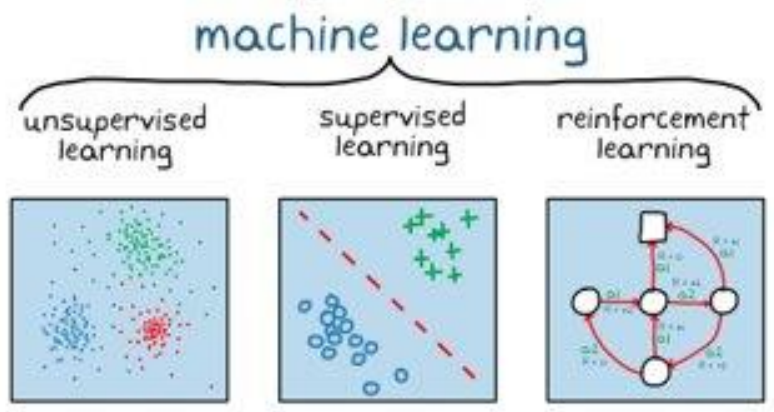
Introduction: Why has Artificial Intelligence (AI) become so popular? <Increased volumes of data>

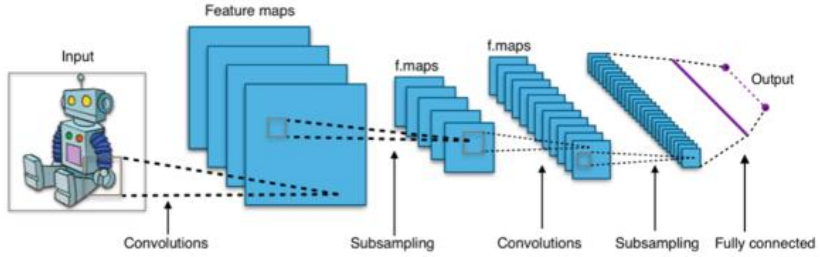
- **Data Availability**: Vast amounts of structured and unstructured data are available for training and testing AI models, which is crucial for their learning and decision-making processes.
- **Data Collection and Storage Technologies**: Innovations in data collection methods, sensors, and storage technologies have made it possible to capture and store vast quantities of data. This is a critical component for training and refining AI algorithms.



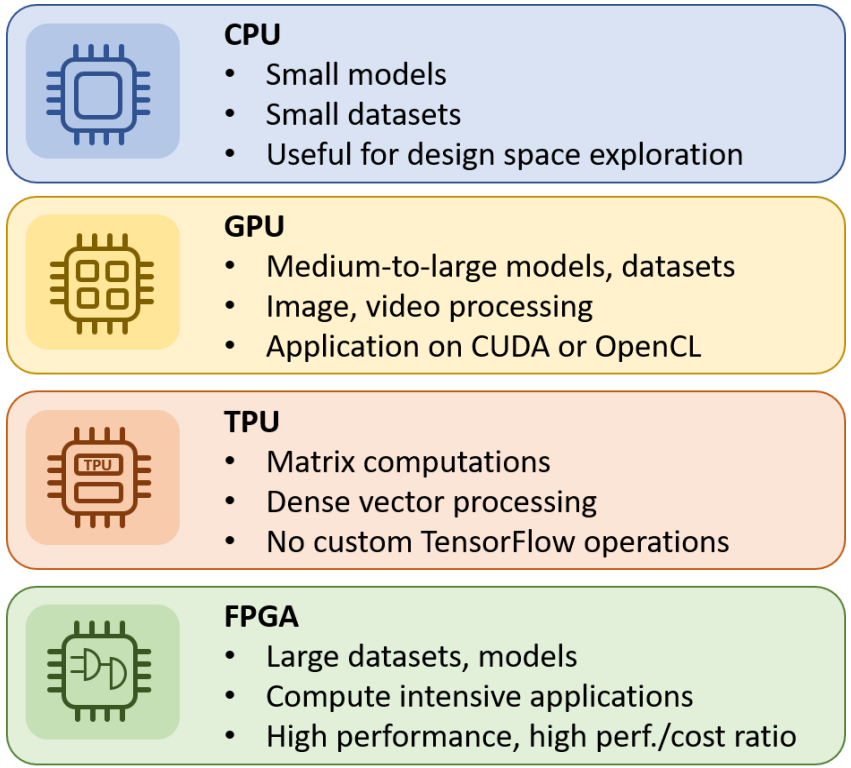
Introduction: Why has Artificial Intelligence (AI) become so popular? <Advanced algorithms>

1. **Machine Learning Algorithms**: Breakthroughs in machine learning algorithms, particularly in areas like deep learning, have significantly improved the performance of AI systems. Deep learning, in particular, has proven to be highly effective in tasks like image recognition, natural language processing, and more.
2. **Transfer Learning and Pretrained Models**: Techniques like transfer learning, where a model trained on a large dataset for one task is adapted for another related task, have accelerated progress in various domains. Pretrained models, which are models that have been trained on large datasets and then fine-tuned for specific tasks, have become a powerful tool for many applications.
3. **Reinforcement Learning and Generative Models**: Advances in reinforcement learning have enabled AI systems to learn through interaction with their environment, making them well-suited for tasks like autonomous control. Generative models, such as Generative Adversarial Networks (GANs), have enabled the creation of realistic synthetic data, which has numerous applications.



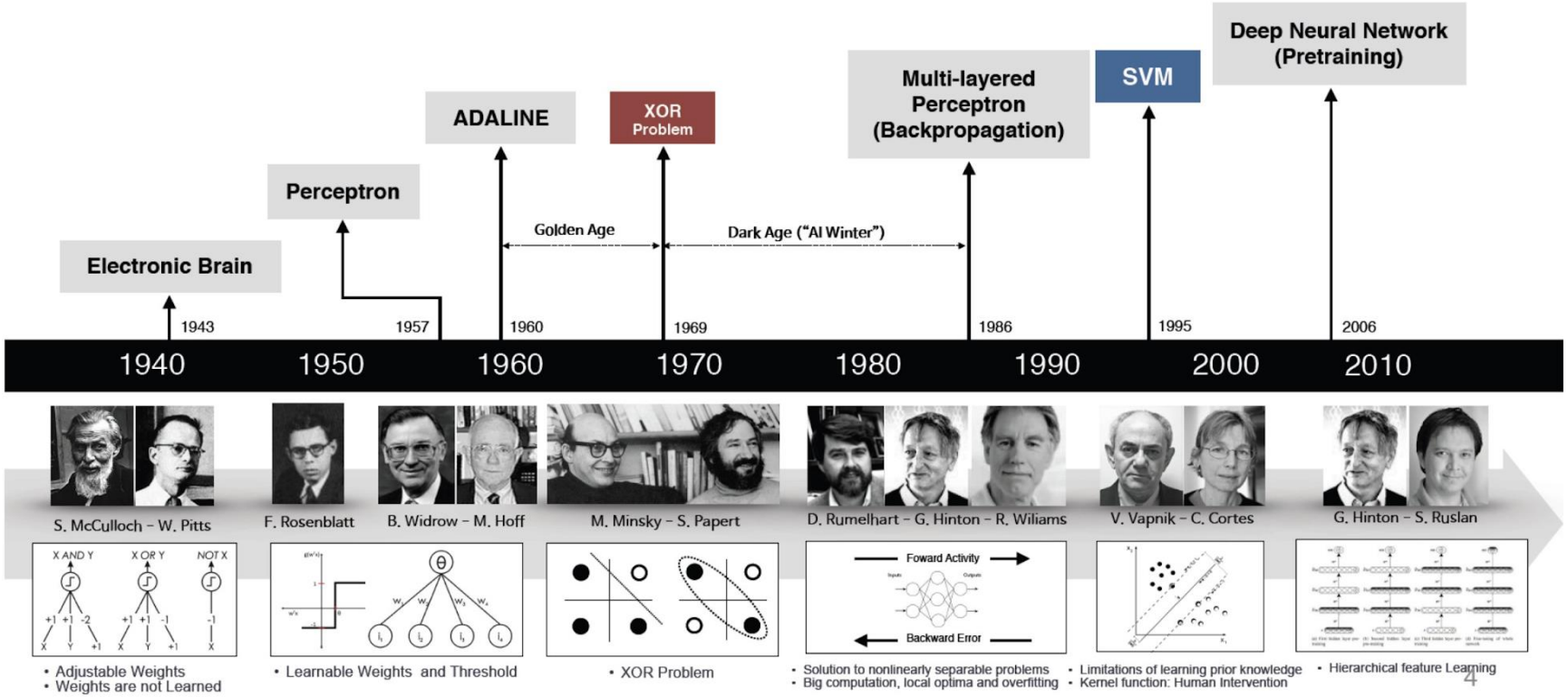


Introduction: Why has Artificial Intelligence (AI) become so popular? <Advancements in Computing Power>



- **CPUs** (Central Processing Units), **GPUs** (Graphics Processing Units), and **TPUs** (Tensor Processing Units) are different types of processors designed for specific types of computational tasks.
- **Specialized Hardware:** The development of specialized hardware like Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) has further accelerated the training and deployment of AI models, especially in deep learning applications.

Introduction: A Brief History of AI



Introduction: A Brief History of AI
<From 1940s to Now>

- **1943: Warren McCulloch and Walter Pitts create a mathematical model of a neural network.**
- **1949**: Donald Hebb proposes a learning rule for neural networks, known as Hebbian learning.
- **1950: Alan Turing introduces the "Turing Test" as a way to evaluate a machine's ability to exhibit intelligent behavior.**
- **1951**: Marvin Minsky and Dean Edmonds build the first neural network computer, SNARC.
- **1956: John McCarthy organized the Dartmouth Conference, which is considered the birth of AI as a field of study. "The term artificial intelligence was first coined by John McCarthy in 1956 "**
- **1967: Frank Rosenblatt develops the perceptron, a simplified model of a biological neuron, which becomes one of the earliest machine learning algorithms. It lays the groundwork for later developments in neural networks.**
- **1960:** John McCarthy develops the programming language LISP, which becomes widely used in AI research.
- **1966:** Shakey the robot, developed at Stanford Research Institute, demonstrates basic problem-solving abilities.
- **1966:** ELIZA is an early natural language processing computer program, the first program that allowed some kind of plausible conversation between humans and machines.

Note: There were two major AI winters (Dark Age of AI) approximately **1974–1980** and **1987–2000**

Introduction: A Brief History of AI <From 1940s to Now>

- **1968:** Terry Winograd develops SHRDLU, a natural language processing system.
- **1970: Expert systems, which are rule-based AI systems, gain popularity. "Expert systems were among the first truly successful forms of artificial intelligence (AI) software"**
- **1972:** The MYCIN system, developed at Stanford, becomes one of the first expert systems used for medical diagnosis.
- **1980:** The first commercial expert system, XCON, is deployed at Digital Equipment Corporation.
- **1986: The concept of backpropagation, a key algorithm for training artificial neural networks, is rediscovered.**
- **1997: IBM's Deep Blue defeats world chess champion Garry Kasparov in a six-game match.**
- **2002:** Roomba, a domestic robot developed by iRobot, is introduced, showcasing advancements in robotics and AI.
- **2010: Deep learning techniques, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), gain prominence.**
- **2011 : IBM's Watson wins the Jeopardy! game show, demonstrating natural language processing and question-answering capabilities.**

Introduction: A Brief History of AI
<From 1940s to Now>

- **2013: Google's DeepMind develops a deep learning algorithm (Q-Learning) that learns to play Atari 2600 video games at a human-level performance. "Playing Atari with Deep Reinforcement Learning".**
- **2016: AlphaGo, a program developed by DeepMind, defeats world Go champion Lee Sedol in a five-game match.**
- 2018: GPT-1 (Generative Pre-trained Transformer) by OpenAI showcases powerful language generation capabilities.
- 2019: OpenAI introduces GPT-2, a large-scale language model
- 2020: OpenAI introduces GPT-3, a large-scale language model.
- **2021: AlphaFold is an artificial intelligence program developed by DeepMind, which performs predictions of protein structure.**
- **2022: Generative Pre-trained Transformer 3.5 (GPT-3.5) is a sub class of GPT-3 Models created by OpenAI in 2022**
- **2023: Generative Pre-trained Transformer 4 (GPT-4) is a multimodal large language model created by OpenAI, and the fourth in its series of GPT foundation models**

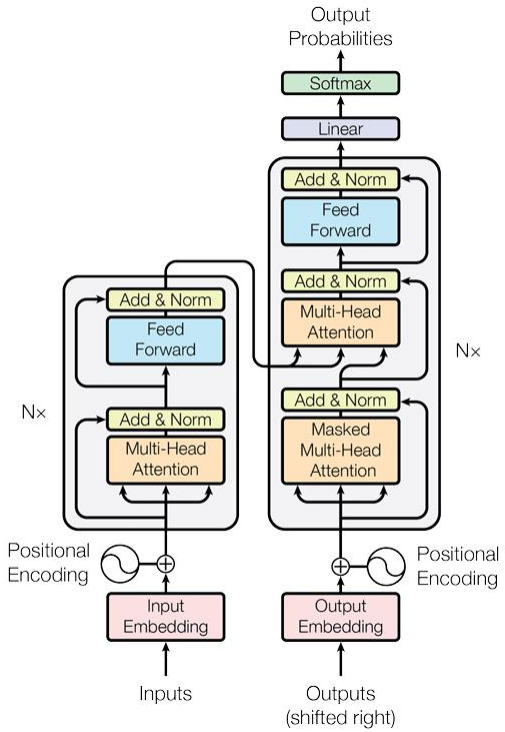
Introduction: A Brief History of AI

1997: Deep Blue IBM chess computer beats Garry Kasparov (Chess Grandmaster)

2011: IBM-Watson Defeats Humans in "Jeopardy!"

2016: Google's AlphaGo (Developed by DeepMind) beats Go master Lee Se-dol.
Full Documentary: AlphaGo – The Movie

Artificial Intelligence: New Trends
<Large Language Models (LLMs)>



Large Language Models (LLMs)

Fig. The Transformer - Model Architecture.
Paper Link: [Attention is All you Need]

Artificial Intelligence: New Trends
<Text to Image Models>

Source: Midjourney.com

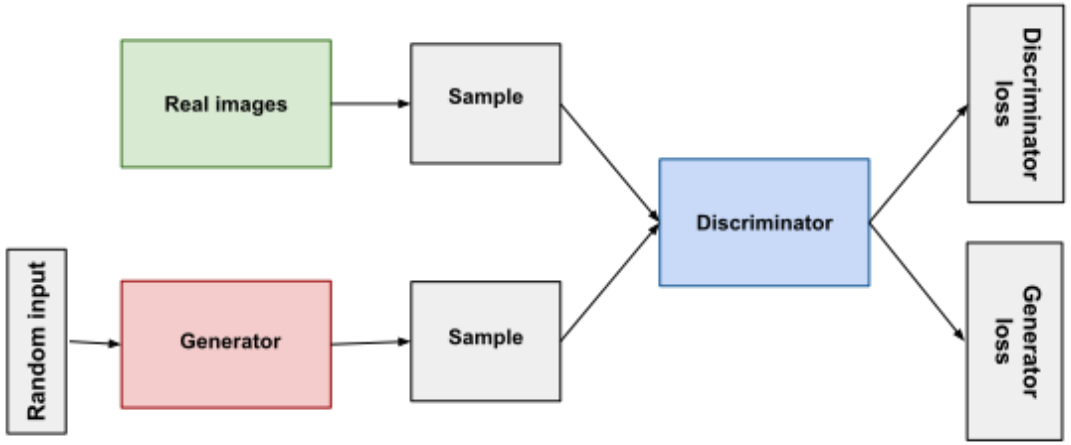
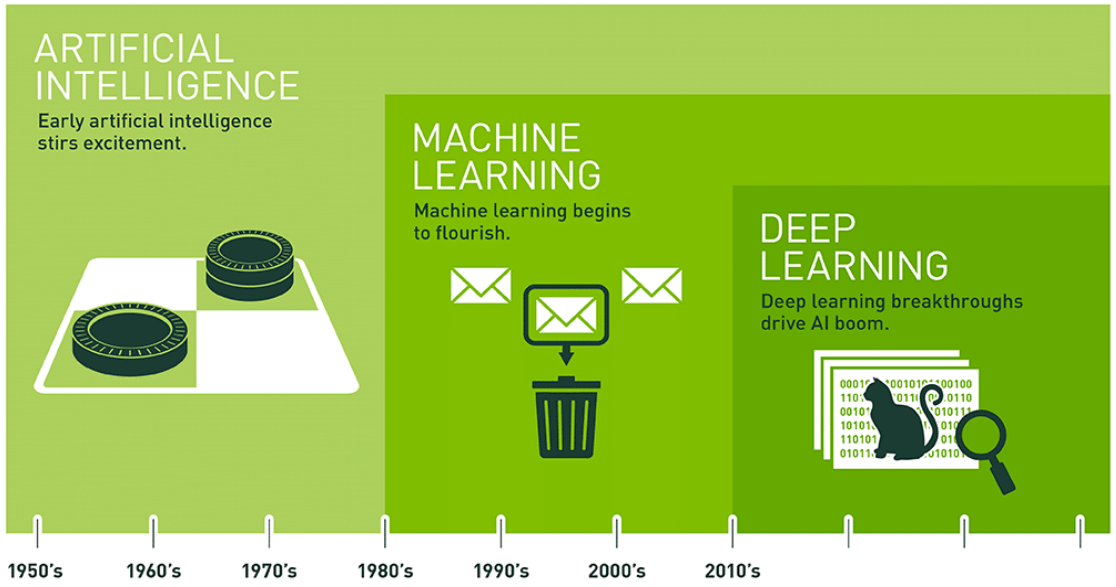


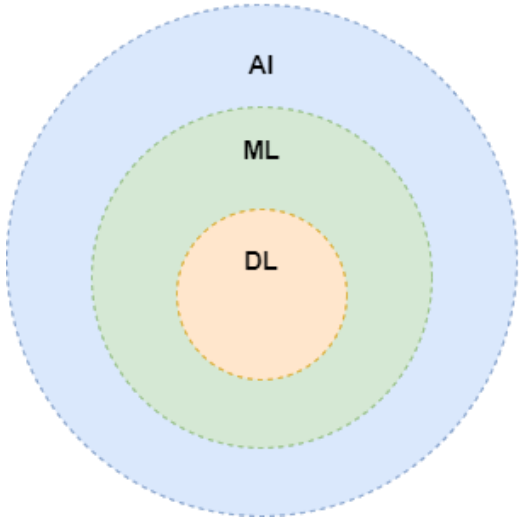
Fig. Overview of Generative Adversarial Networks (GANs)

Source: DALL-E 2

Artificial Intelligence, Machine Learning, and Deep Learning.
<What is the difference?>



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.



Artificial Intelligence

Class of problems we can solve when computers think/act like humans

Machine Learning

The science of getting computers to learn without being explicitly programmed. Data + Algorithms

Deep Learning

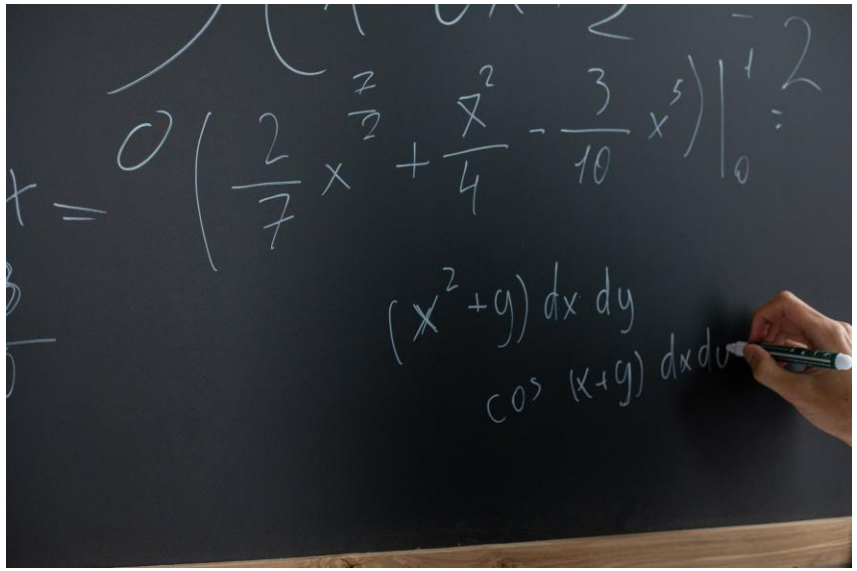
Subset of machine learning that is based on neural networks to mimic the human brain. Generally outperforms classical machine learning on unstructured data

Source: blogs.nvidia.com

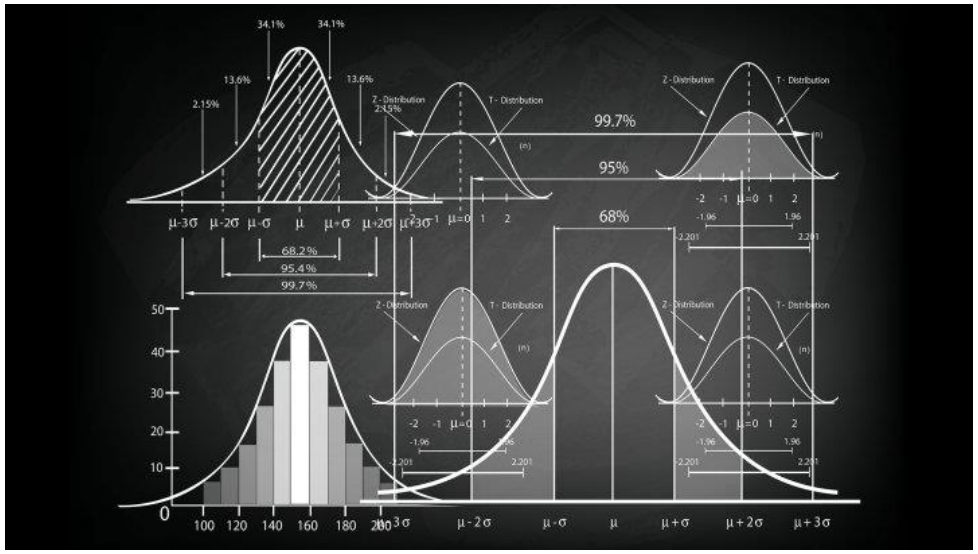
Introduction: The Importance of Mathematics

Mathematics are essential for understanding and working with machine learning algorithms. These include:

1. **Linear Algebra:**
   - Vectors and Matrices: Vectors and matrices are fundamental for representing and manipulating data. They are used extensively in tasks like transformations, regression, and neural networks.
   - Matrix Operations: Operations like addition, multiplication, and inversion of matrices are essential for various machine learning algorithms.
2. **Calculus:**
   - Derivatives and Gradients: Calculus is crucial for optimization algorithms. Understanding derivatives allows for finding the optimal parameters of a model.
   - Integration: Useful in probability theory, which is essential for many machine learning models.

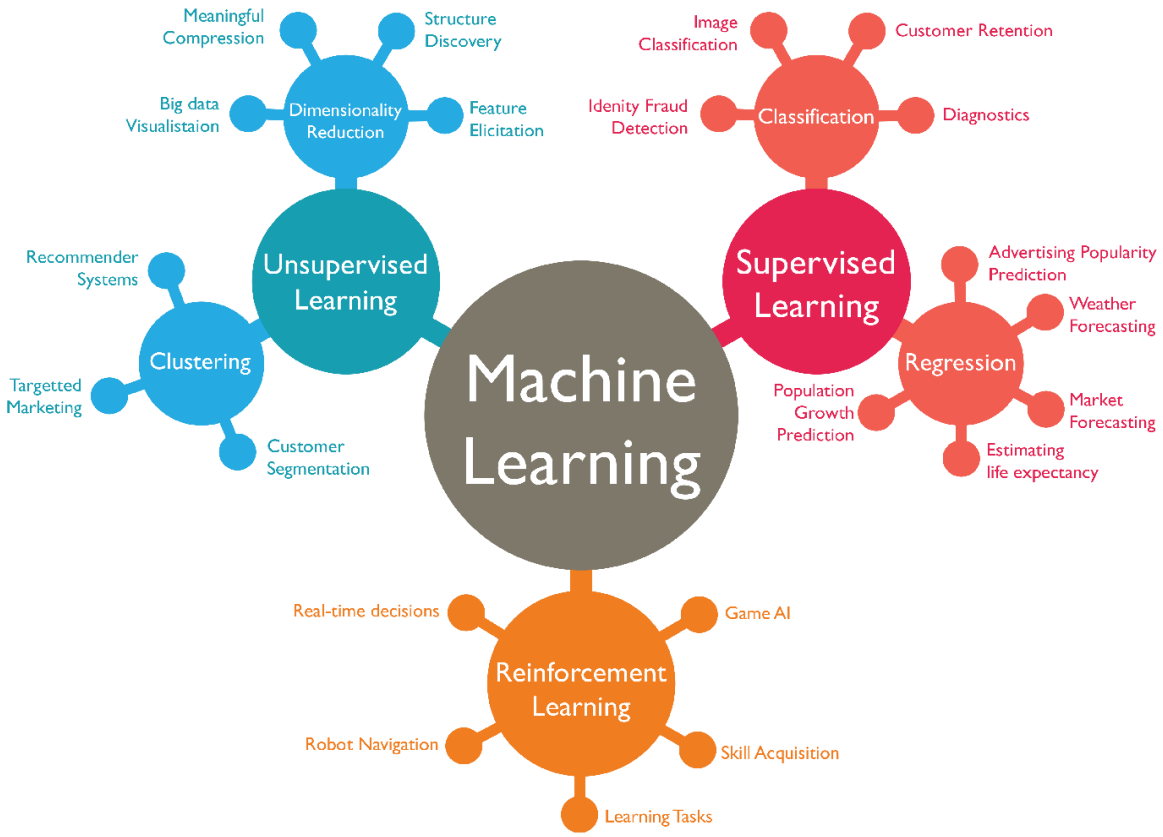


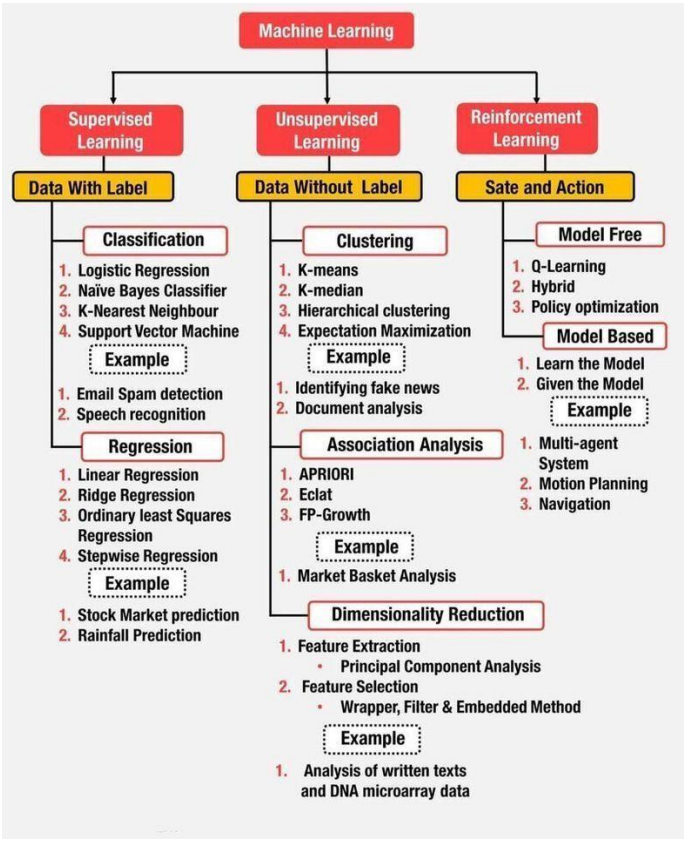
Introduction: The Importance of Mathematics



1. **Probability and Statistics:**
   - Probability Distributions: Understanding different probability distributions is crucial for modeling uncertainty and making predictions.
   - Bayesian Inference: It's used in Bayesian methods, which are important in areas like Bayesian networks and probabilistic programming.
   - Hypothesis Testing and Confidence Intervals: Used for evaluating the significance of results and estimating uncertainties.
2. **Optimization:**
   - Gradient Descent: A widely used optimization algorithm for training machine learning models.
   - Convex Optimization: Important for problems where the objective function is convex, which is common in machine learning.
3. **Information Theory:**
   - Entropy, Mutual Information: These concepts are used in feature selection, dimensionality reduction, and understanding the information content of data.

Machine Learning: Types of Machine Learning

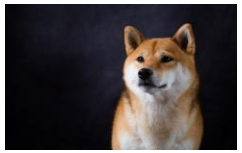




Machine Learning: Types of Machine Learning

Supervised Learning

Input Data: Labeled Data



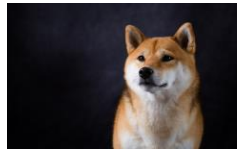
Label: Dog

Label: Cat

Label: Dog

Unsupervised Learning

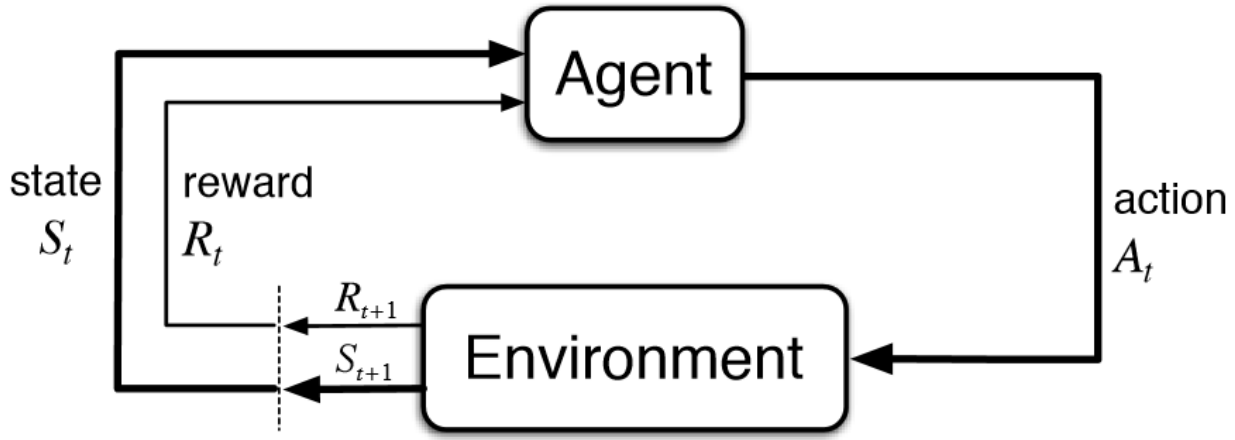
Input Data: Unlabeled Data



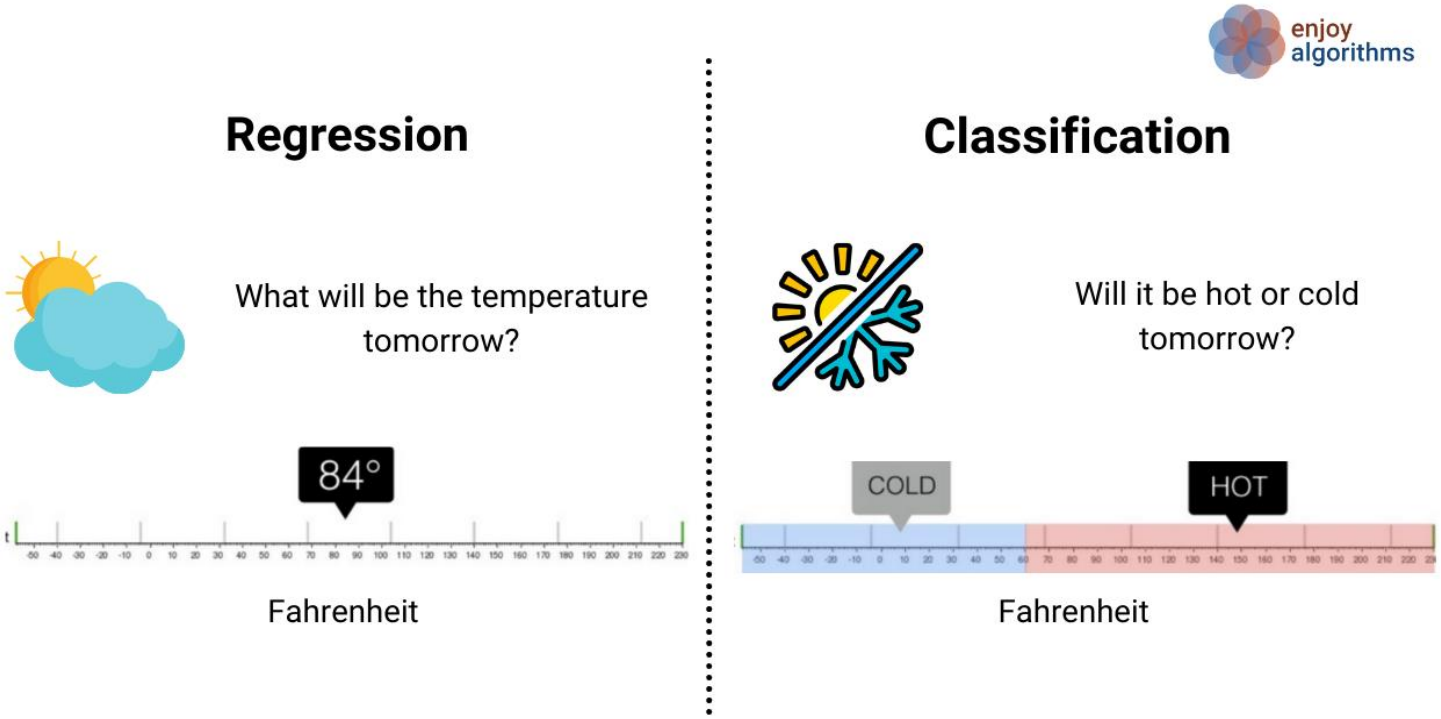
Machine Learning: Types of Machine Learning

Reinforcement Learning

The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent.



Machine Learning: Types of Supervised Learning



Source: enjoyalgorithms.com

Machine Learning: Supervised Learning Algorithms

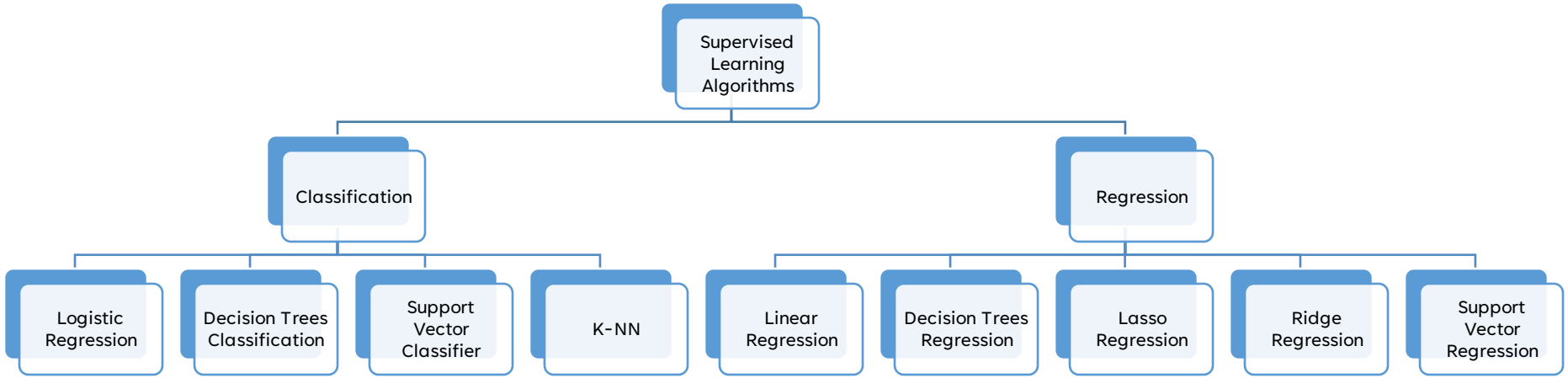
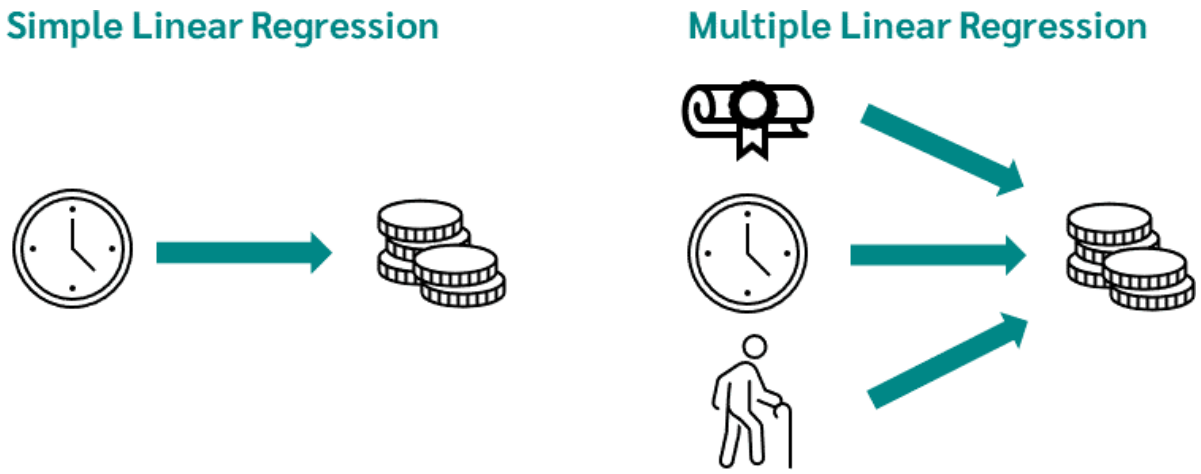


Fig. Some supervised learning algorithms

Supervised Learning Algorithms: Linear Regression

- Linear regression is a fundamental statistical and machine learning technique used to model the relationship between a dependent variable (target) and one or more independent variables (features)
- Depending on whether there are one or more independent variables, a distinction is made between simple and multiple linear regression analysis.
- The output **y** can be calculated from a linear combination of the input variables **X**



Supervised Learning: Linear Regression

$$\hat{y} = b \cdot x + a$$

Estimated dependent variable

Slope

Independent variable

y intercept

Mathematical model for simple linear regression

Simple Linear Regression

$$\hat{y} = b \cdot x + a$$

Multiple Linear Regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

From simple linear regression to multiple linear regression

Supervised Learning: Linear Regression

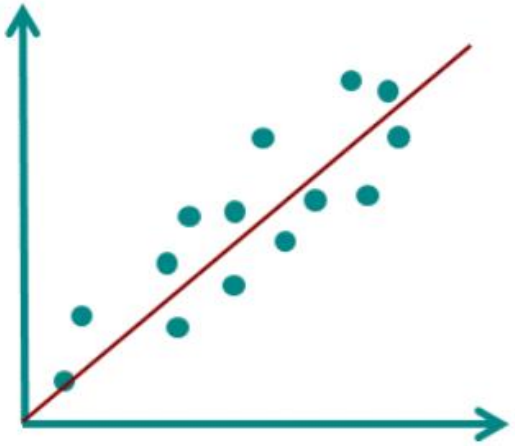
Simple Linear Regression

$$\hat{y} = b \cdot x + a$$

Multiple Linear Regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

Question: How can we find the optimal values for **a** and **b** that fit the data?

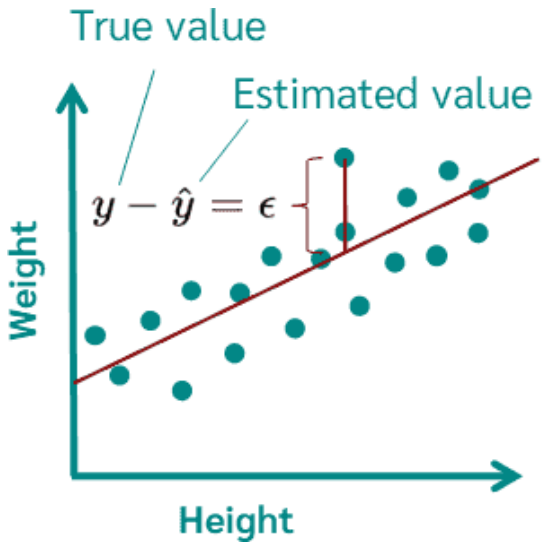


Supervised Learning: Linear Regression
<Ordinary Least Squares and Normal Equations>

Using a mathematical approach called **Least Squares (Ordinary Least Squares)** we can find the values of **a** and **b** that **minimizes** the **error epsilon ε**

$$a, b = \min_{a,b} \sum_{i=1}^{n} \hat{\epsilon}_i^2$$

$$\epsilon = y - \widehat{y} = y - (bX + a)$$



Error epsilon

$$y = b \cdot x + a + \epsilon$$

Example of simple linear regression

Supervised Learning: Linear Regression
<Ordinary Least Squares and Normal Equations>

Normal equations are **equations obtained by setting equal to zero the partial derivatives of the sum of squared errors** (least squares); normal equations allow one to estimate the parameters of a multiple linear regression.

$$LS = \sum_{i=1}^{n} (Y_i - \hat{Y})^2 = \sum_{i=1}^{n} (Y_i - a - bX_i)^2$$

Setting $\frac{\partial LS}{\partial a} = 0$ and $\frac{\partial LS}{\partial b} = 0$

$$a = \overline{Y} - b\overline{X} \qquad b = \frac{\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})}{\sum_{i=1}^{n}(x_i - \overline{X})^2}$$

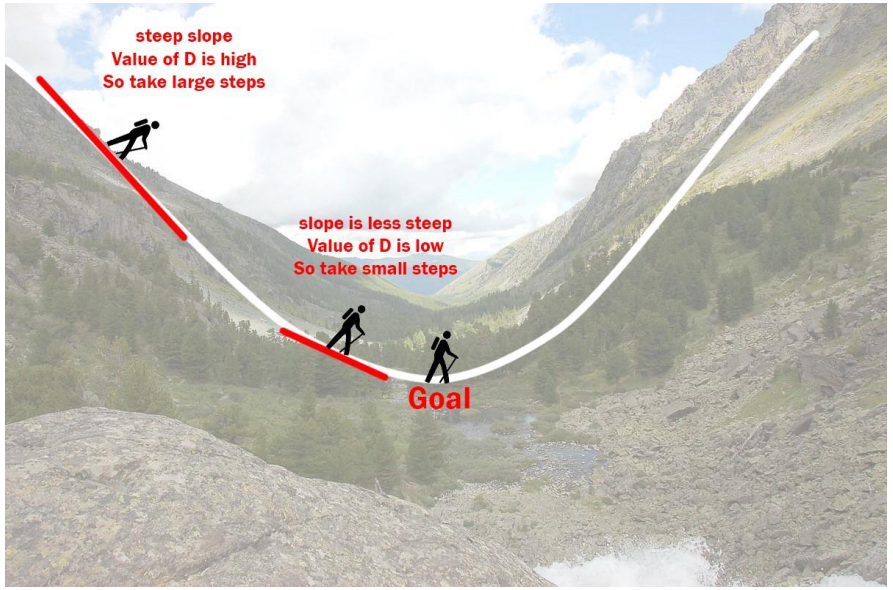
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

If you cannot solve it by yourself, you can check this useful link: http://seismo.berkeley.edu/~kirchner/eps_120/Toolkits/Toolkit_10.pdf

Supervised Learning: Linear Regression
<Gradient Descent>

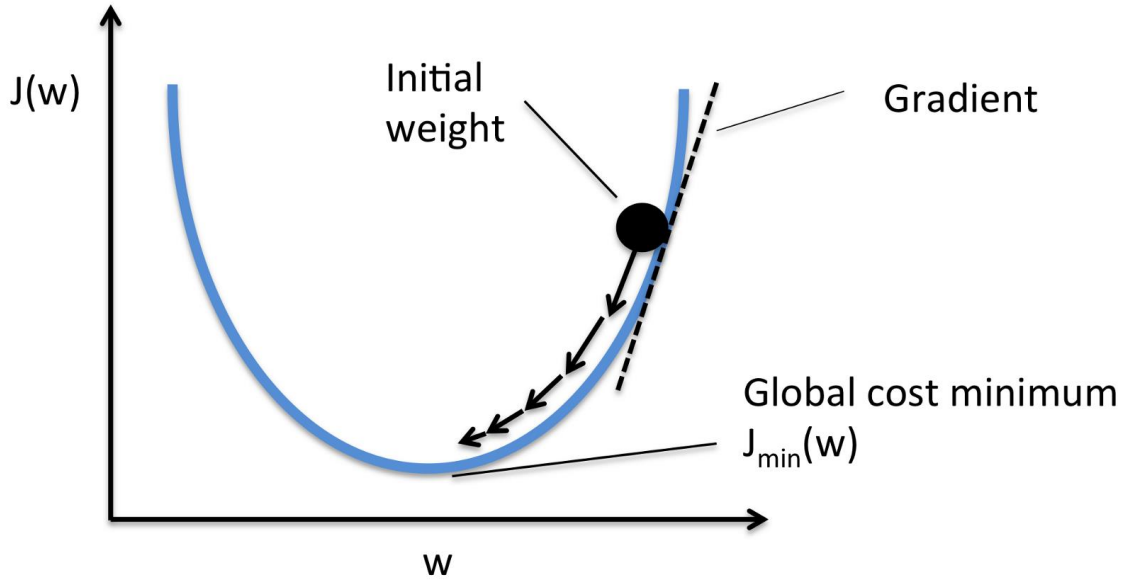
There is another alternative to the normal equations, using an optimization-based approach called **"Gradient Descent"**

- **Definition:** Gradient descent is an iterative optimization algorithm used for finding the minimum of a function. It's widely used in machine learning and deep learning for training models. The basic idea behind gradient descent is to take steps in the direction of steepest decrease in the function, i.e., in the direction of the negative gradient.
- **Objective:** Given a function f(x) that we want to minimize (Loss function/Cost function), gradient descent aims to find the value of x that minimizes f(x).



Gradient Descent: Iterative optimization algorithm

Supervised Learning: Linear Regression
<Gradient Descent>



Gradient Descent: Iterative optimization algorithm

Gradient: The gradient of a function f(x) is a vector that points in the direction of the steepest increase in the function. The negative gradient points in the direction of the steepest decrease.

$$w_i := w_i - \alpha \nabla J(w_i)$$

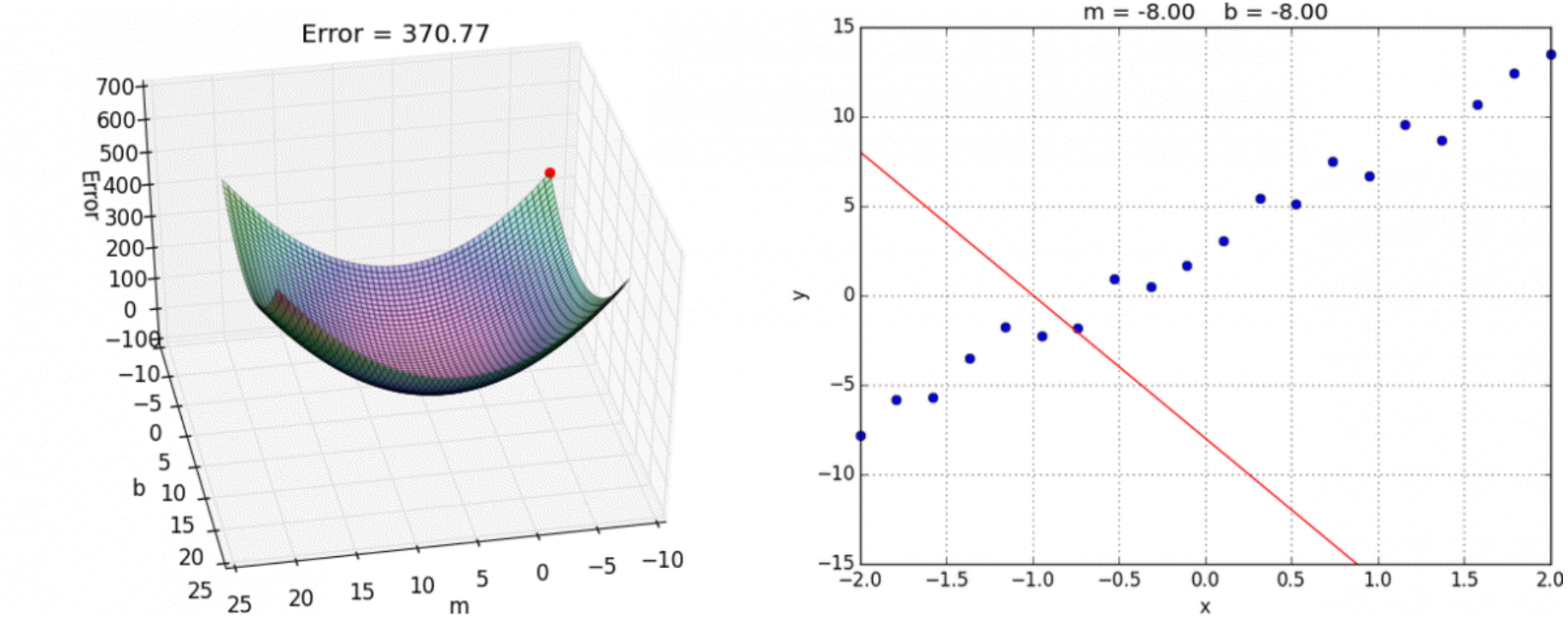
Supervised Learning: Linear Regression
<Gradient Descent>

$$\hat{\alpha}, \hat{\beta} = \min_{\alpha,\beta} \sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \min_{\alpha,\beta} \sum_{i=1}^{n} (Y_i - \beta - \alpha X_i)^2$$

- **Step 1:** Start with a random initialization of the coefficients (α and β) (Set the values to 0 for example)
- **Step 2:** Repeat "K" times this "descending" using the learning rate alpha (but in order to avoid confusion α, we will name it gamma(γ))

$$\beta_k = \beta_{k-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} (\beta_{k-1} + \alpha_{k-1} X_i - Y_i)$$
$$\alpha_k = \alpha_{k-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} (\beta_{k-1} + \alpha_{k-1} X_i - Y_i) \cdot X_i$$

Supervised Learning: Linear Regression
<Gradient Descent>



Source: https://alykhantejani.github.io/a-brief-introduction-to-gradient-descent/

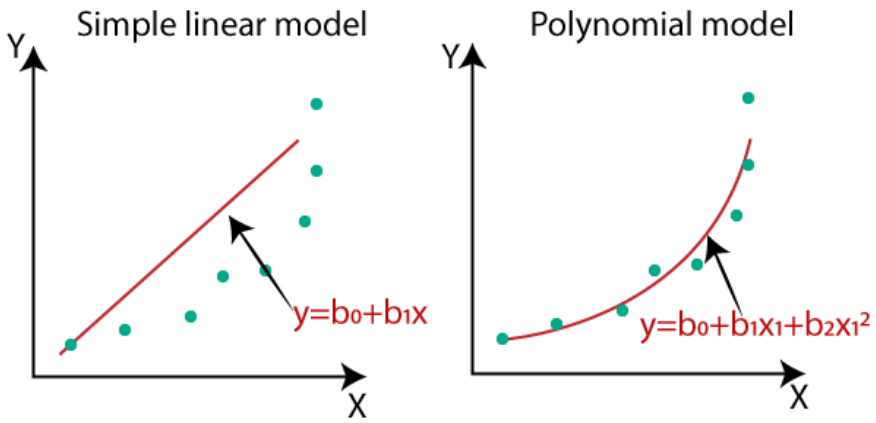
Supervised Learning: Linear Regression
<Gradient Descent>

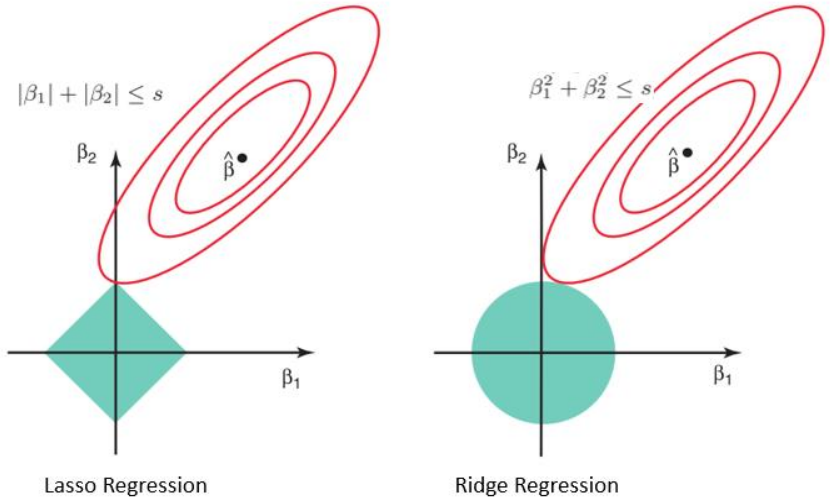
Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

- **Gradient:** The gradient of a function **f(x)** is a vector that points in the direction of the steepest increase in the function. The negative gradient points in the direction of the steepest decrease.

Supervised Learning: Advanced Types of Regression





Polynomial Regression: In which, we describe the relationship between the independent variable x and the dependent variable y using an nth-degree polynomial in x

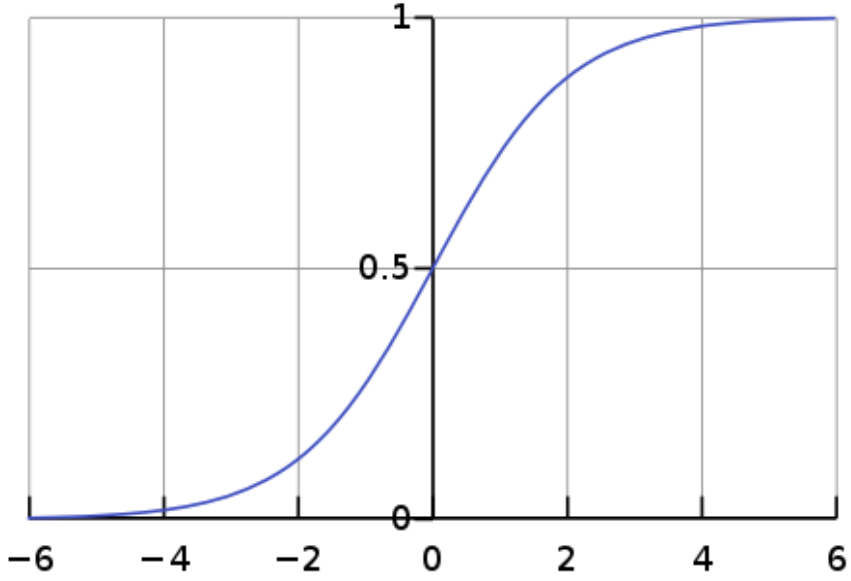
Lasso and **Ridge regression** are both techniques used in **regression analysis** to handle the problem of **overfitting** and to improve the **generalization** of the model. They do this by **adding** a **penalty** term to the standard linear regression **cost function**.

Supervised Learning: Linear Regression
<DEMO>

DEMO: Session 1 – Linear Regression

Supervised Learning Algorithms: Logistic Regression

- **Logistic regression** is a statistical model used for analyzing datasets where there are one or more **independent variables (features)** that can be used to predict the outcome of a categorical **dependent variable (target)**.
- **Binary outcome:** It's particularly useful when the dependent variable is binary, meaning it has only two possible outcomes (e.g., 0 or 1, yes or no, true or false).
    - **Note:** For **multi-class classification** tasks (i.e., more than two classes), logistic regression can be extended using techniques like **one-vs-all** (also known as **one-vs-rest**) or softmax regression.
- **Why there is the "regression" word in a classification algorithm?** Despite its name, logistic regression is primarily used for classification rather than regression. **Logistic regression** gives a continuous value of **P(Y=1)** for a given input **X**, which is later converted to **Y=0** or **Y=1** based on a threshold value.
- **Sigmoid function:** It is the core of logistic regression (also known as the logistic function) to model the relationship between the independent variables (features) and the probability of the outcome. The sigmoid function "**squashes**" the output to be between 0 and 1, which makes it suitable for representing probabilities. It is defined as follows:



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

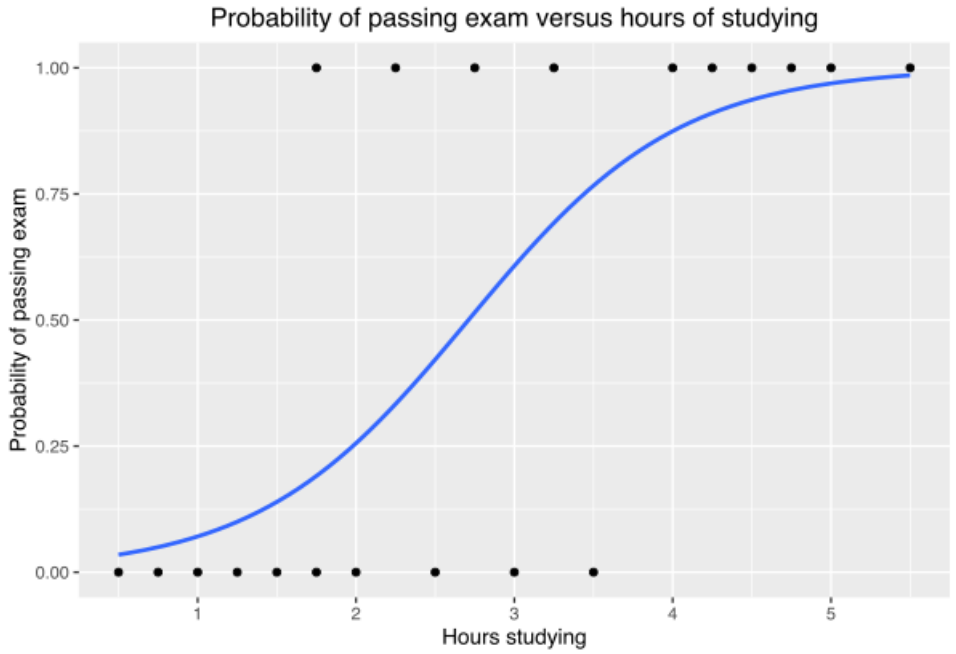
where "x" is the linear combination of the input features and their corresponding coefficients.

Supervised Learning Algorithms: Logistic Regression

Instead of fitting a line to the data (as in Linear Regression), Logistic Regression fits an “S” shaped logistic function called the Sigmoid Function.

$$P(y = 1) = \frac{1}{1 + \exp(-z)}$$

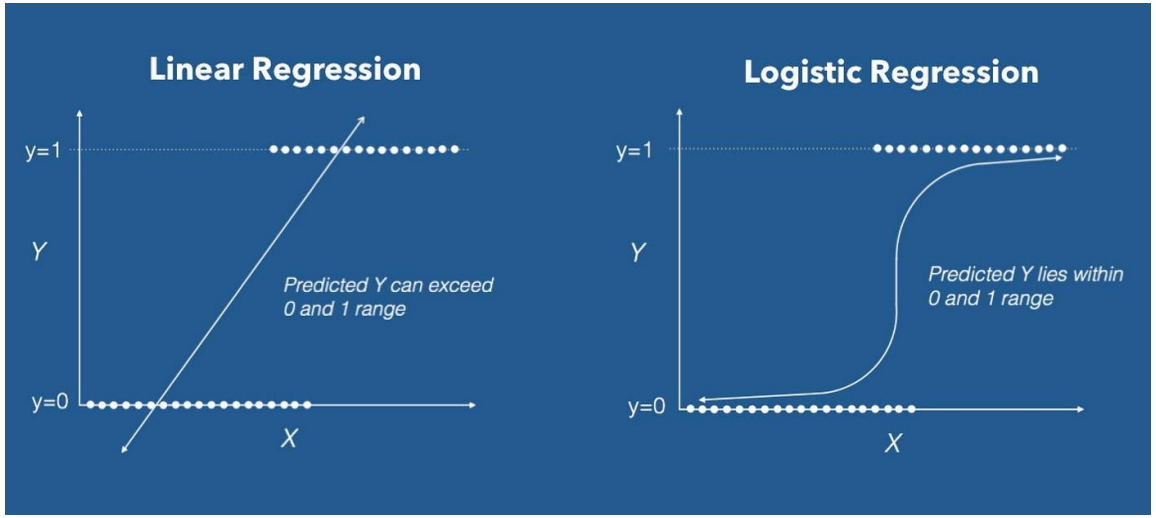
$$P(y = 0) = 1 - P(y = 1)$$



Supervised Learning Algorithms: Logistic Regression

The difference between **logistic regression** and **linear regression**

- One big difference between linear regression and logistic regression is how the line is fit to the data.
- In **linear regression**, we fit the line using least squares, In other words, we find the line that minimizes the sum of the squares of residuals (errors).
- **Logistic regression** does not have the same concept as residual, so it cannot use least squares. Thre is not a closed-form solution to solve the logistic regression
- **Logistic regression** uses something called "Maximum likelihood estimation" or "gradient descent", to estimate the coefficients of the logistic regression.

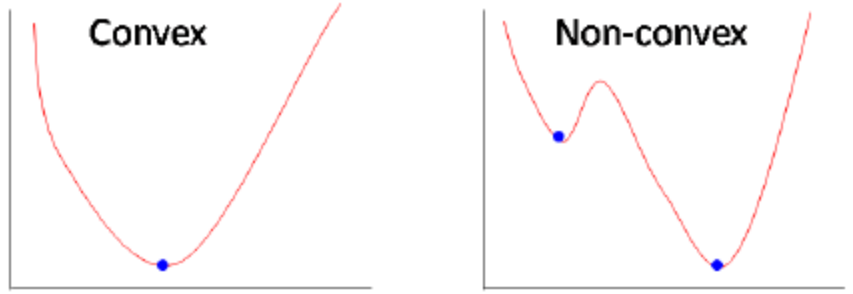


Note: For logistic regression, there is no longer a closed-form solution, due to the nonlinearity of the logistic sigmoid function.

Supervised Learning Algorithms: Logistic Regression
<How to train a logistic regression algorithm?>

Note: As usual, before training any machine learning algorithm we need to define a Loss/Cost Function.

As we have seen before, **linear regression** uses **Least Squared Error** as a loss function that gives a **convex** loss function and then we can complete the optimization by finding its vertex as a global minimum. However, for logistic regression, the hypothesis is changed, the Least Squared Error will result in a **non-convex** loss function with local minimums by calculating with the sigmoid function applied to the model's output.



Supervised Learning Algorithms: Logistic Regression
<How to train a logistic regression algorithm?>

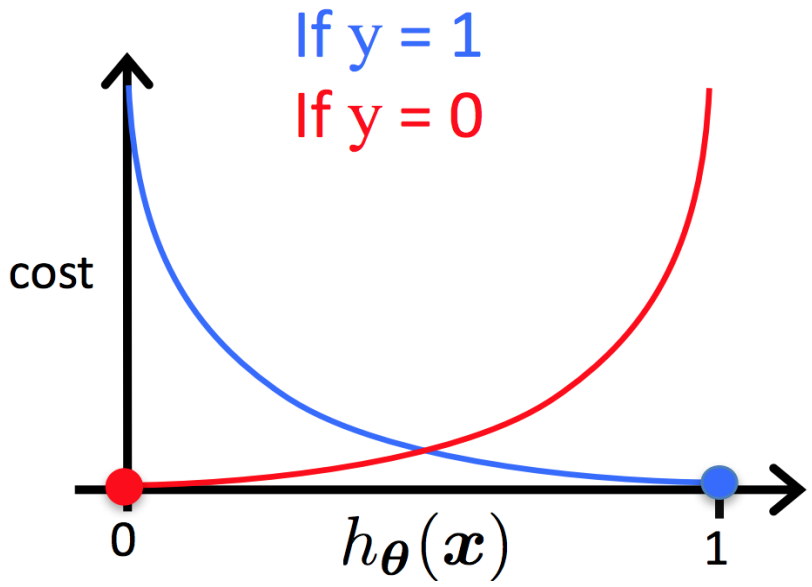
Note: As usual, before training any machine learning algorithm we need to define a Loss/Cost Function.

Due to the problem of non-convex if we apply the same loss function used for linear regression, therefore, the cost function for logistic regression is going to be defined as follows:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Where,

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T X)}$$



Supervised Learning Algorithms: Logistic Regression
<Gradient Descent>

As you can see, If y = 1 and *h(x)* = 1, the cost = 0. But if y = 1 and *h(x)* = 0, we will penalize the learning algorithm by a very large cost, cost = +*inf.*
The two defined functions for cost function can be compressed into a single function as follow:

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

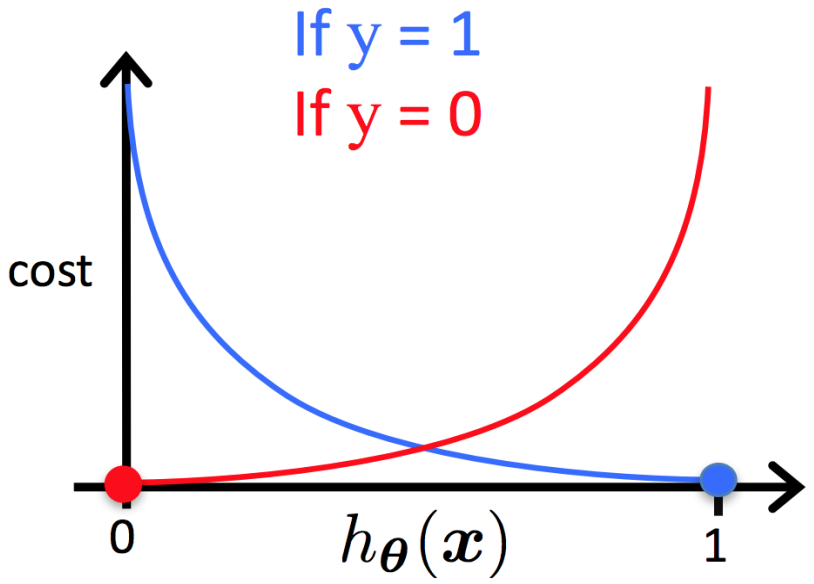
To minimize our cost function *J(θ)*, we are going to use the gradient descent algorithm.

Repeat {

$$\theta_j := \theta_j - \alpha\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

(simultaneously update all θ_j)

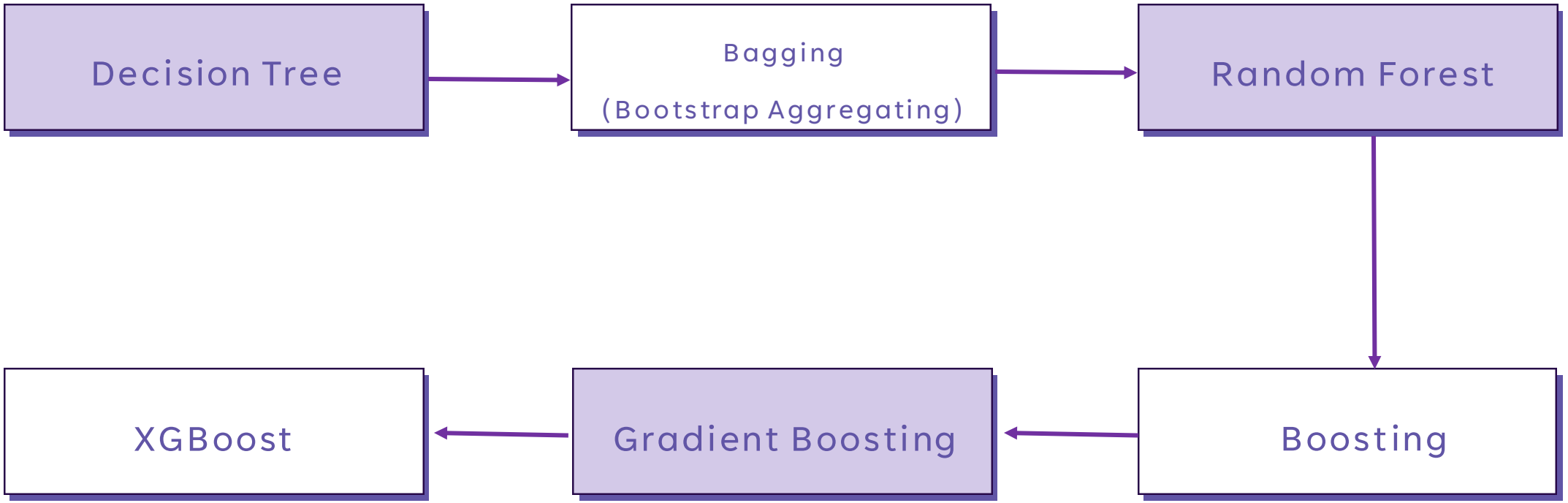
}



Supervised Learning: Logistic Regression <DEMO>

DEMO: Session 1 – Logistic Regression

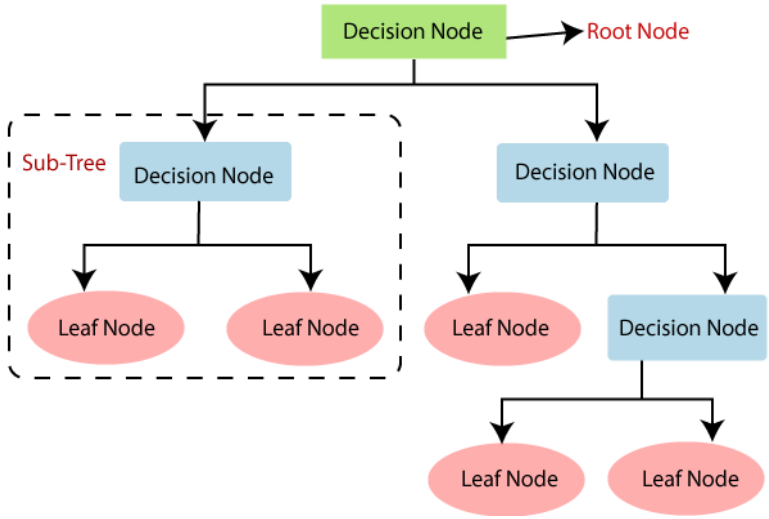
Supervised Learning Algorithms: Tree-Based Algorithms and Ensemble Methods



Supervised Learning Algorithms: Tree-Based Algorithms and Ensemble Methods

These methods can be used for both regression and classification problems.

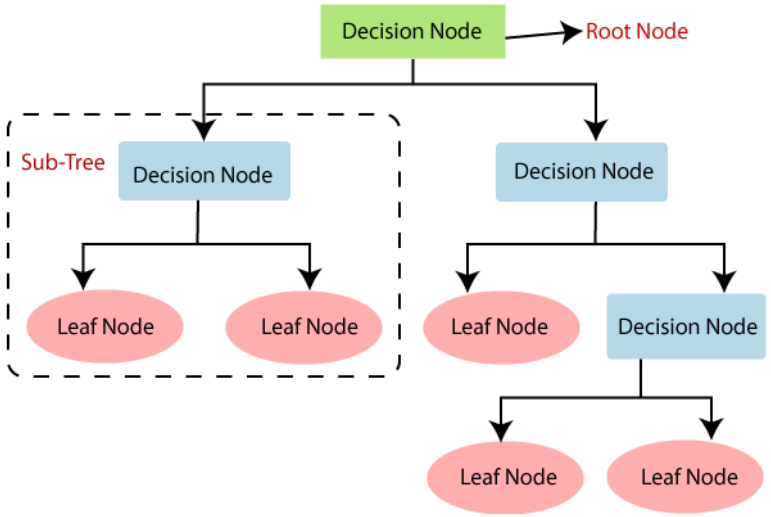
- **CART:** Classification and Regression Trees (CART), commonly known as decision trees, can be represented as **binary/non-binary trees**. They have the advantage to be very interpretable.
- **Bagging:** Bootstrap + Aggregating, is the ensemble technique used by random forest. Bagging chooses a random sample/random subset from the entire data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling.
- **Random Forest:** It is a tree-based technique that uses a high number of decision trees built out of randomly selected sets of features. Contrary to the simple decision tree, it is highly uninterpretable, but its generally good performance makes it a popular algorithm.
- **Boosting:** The idea of boosting methods is to combine several weak learners to form a stronger one. The main ones are gradient boosting (e.g., XGBoost) and Adaptive boosting (AdaBoost)



Supervised Learning Algorithms: Decision Trees

Decision trees (also called Classification And Regression Trees "CART") are a type of machine learning algorithm that makes decisions by splitting data into subsets based on certain features. They are used for both classification (assigning a label to an item) and regression (predicting a numerical value).

1. Structure: Imagine a flowchart where each node represents a decision based on a feature, and the branches represent the possible outcomes or further decisions.
2. How They Work:
    1. Root Node: This is the starting point where the entire dataset is. It's the feature that is most effective at splitting the data.
    2. Internal Nodes: These nodes split the data based on certain conditions (e.g., if a value is greater than a certain threshold).
    3. Leaf Nodes: These are the final nodes that do not split further. They provide the final prediction or classification.



Supervised Learning Algorithms: Decision Trees <Splitting Criteria>

Question: How can we choose the best decision nodes?
Answer: Using splitting criteria, the model chooses the best features to split the data based on some criteria (e.g., Gini impurity for classification, Mean Squared Error for regression).
Splitting Criteria:

- **For classification tasks:** Gini Impurity, Information Gain, Entropy, Gini Gain. (and more)
- **For regression tasks:** Mean squared error, mean absolute error, and variance reduction.

Supervised Learning Algorithms: Decision Trees for Classification

1. **Gini Impurity:**
   - Gini impurity is a measure of how mixed the classes are in a given dataset or subset of data.
   - It ranges from 0 to 0.5, where 0 indicates that a node contains only samples of a single class, and 0.5 indicates that the samples are evenly distributed among the classes.
   - The decision tree algorithm aims to minimize the Gini impurity when making splits.

The formula for Gini impurity for a node is:

$$Gini(t) = 1 - \sum_{i=1}^{c} (p_i)^2$$

2. **Entropy:**
   - Entropy measures the disorder or uncertainty in a set of data.
   - Like Gini impurity, it also ranges from 0 to 1, with 0 indicating perfect order and 1 indicating maximum disorder.
   - Decision trees aim to minimize entropy when making splits.

The formula for entropy for a node is:

$$Entropy(t) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Supervised Learning Algorithms: Decision Trees for Classification

1. **Information Gain:**
   - Information Gain is used to determine the best feature to split the data at each node in a decision tree.
   - It quantifies how much information a feature gives us about the classes.
2. **Gini Gain:**
   - Gini Gain is similar to Information Gain, but it uses Gini impurity as the measure instead of entropy.
   - It's calculated similarly to Information Gain but using Gini impurity instead of entropy.

Information gain is calculated as the difference between the entropy (or Gini impurity) before and after the split:

$$IG(D, A) = H(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} H(D_v)$$

where D is the dataset at the current node, A is the feature being considered for the split, D_v is the subset of D for which feature A has value v, and H represents entropy or Gini impurity.

Supervised Learning Algorithms: Decision Trees for Regression

1. **Mean Squared Error (MSE):**
   - The MSE measures the average squared difference between predicted and actual values. It quantifies the accuracy of a regression model.
   - In the context of decision trees, the MSE is used to evaluate the quality of a split. The split that minimizes the MSE is chosen.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

2. **Mean Absolute Error (MAE):**
   - The MAE is similar to MSE but measures the average absolute difference between predicted and actual values, rather than squared differences.
   - MAE is less sensitive to outliers compared to MSE.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Supervised Learning Algorithms: Decision Trees for Regression

Variance Reduction:

- In regression tasks, the goal is often to minimize the variance of the target variable within each node.
- The variance reduction is used as the splitting criterion, and it measures how much the variance of the target variable is reduced after the split.
- The formula for variance reduction is specific to regression tasks and involves the calculation of variances.
- The formula for variance reduction in the context of regression trees is as follows:
  - Given a node with a dataset **D** containing **n** samples, where $\mathbf{y_i}$ represents the target variable values for each sample, the variance reduction is calculated as:

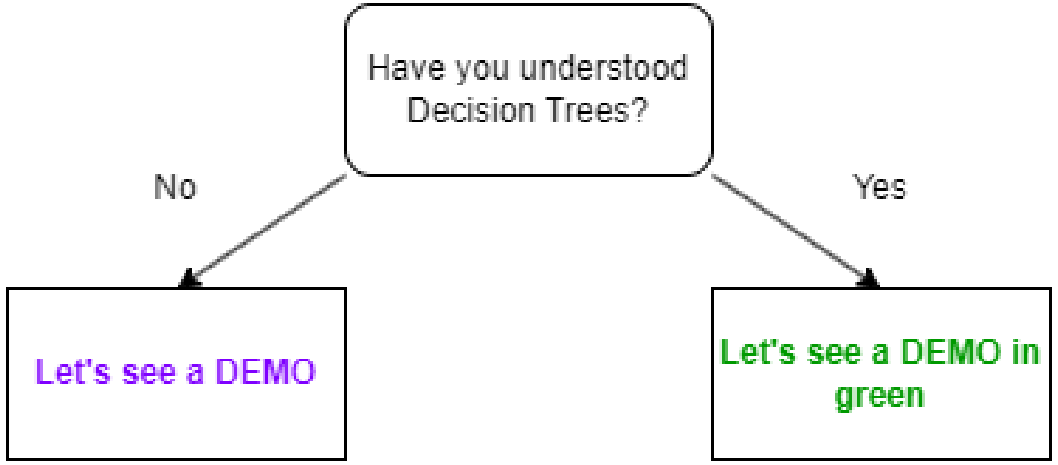
$$\text{Variance Reduction}(D) = \text{Var}(D) - \sum_{i=1}^{m} \frac{|D_i|}{|D|} \times \text{Var}(D_i)$$

$$\text{Var}(D) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2$$

- m is the number of child nodes resulting from the split.
- Di represents the dataset in the i-th child node after the split.
- |Di | is the number of samples in the i-th child node.
- Var(D) is the variance of the target variable in node D
- y-bar is the mean of the target variable values in node D

Supervised Learning Algorithms: Decision Trees <DEMO>

DEMO: Session 1 – Decision Trees



Supervised Learning Algorithms: K-Nearest Neighbors

k-Nearest Neighbors (k-NN) is a **supervised** machine learning algorithm used for both **classification** and **regression** tasks. It's a simple and intuitive algorithm that relies on the idea of finding the "**nearest**" data points in the training set to a given test point and using those neighbors to make a prediction.

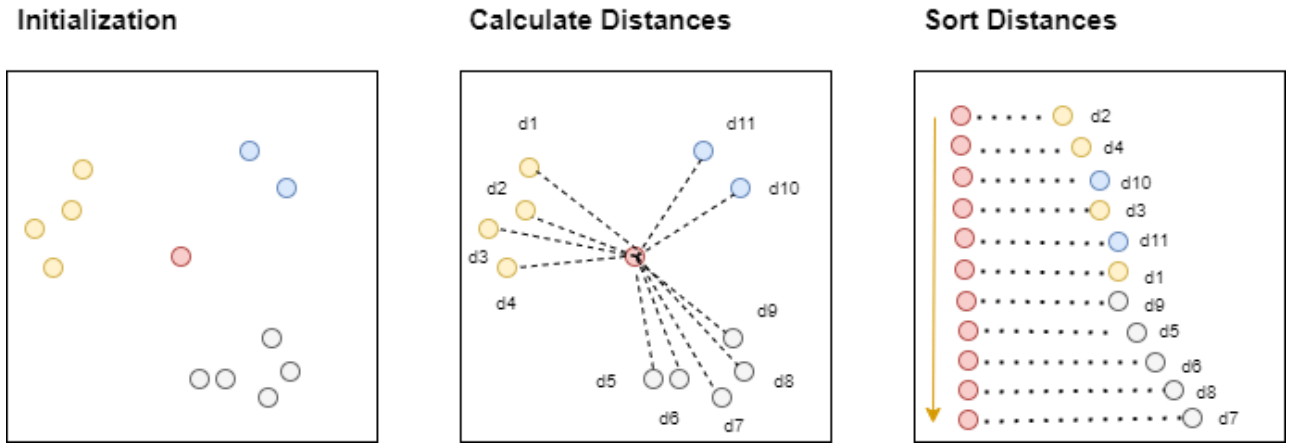


Fig. K-NN Algorithm Steps

Supervised Learning Algorithms: K-Nearest Neighbors

1. **Training Phase:**
   - Store the entire dataset in memory.
   - No explicit training is done, as k-NN is a lazy learner.
2. **Prediction Phase:**
   - When given a new, unseen data point, the algorithm finds the k data points in the training set that are closest (most similar) to the new point.
   - The "closeness" is typically determined by a distance metric like Euclidean distance, Manhattan distance, etc.
3. **Classification:**
   - In the classification task, the algorithm assigns a class label to the new point based on the majority class among its k nearest neighbors.
4. **Regression:**
   - In regression, the algorithm predicts a continuous value for the new point based on the average (or some other measure) of the target values of its k nearest neighbors.

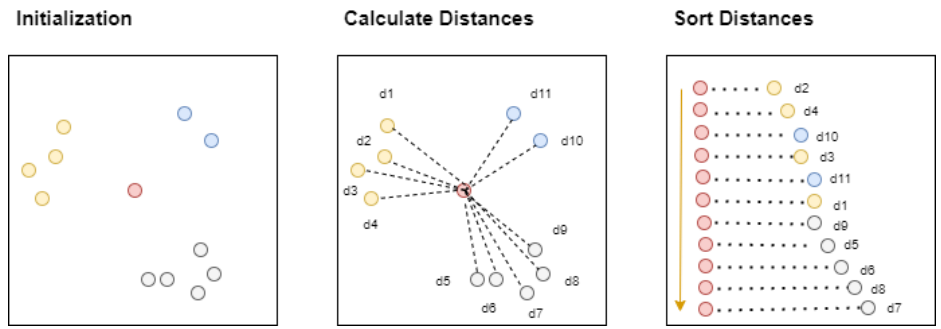


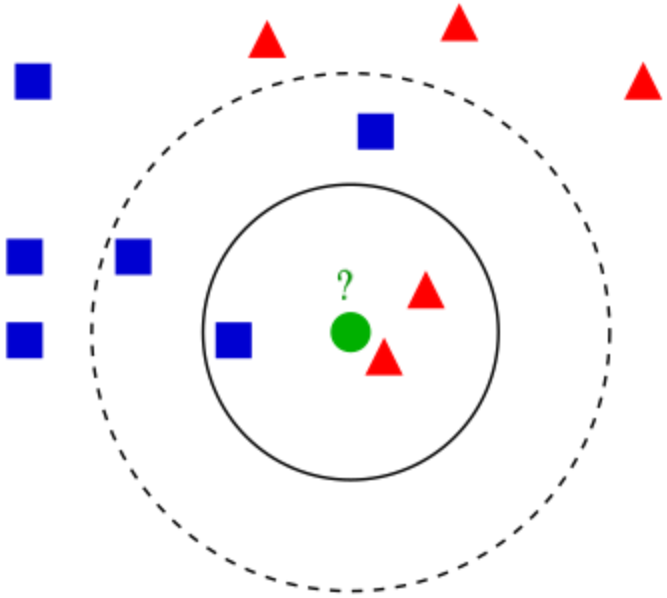
Fig. K-NN Algorithm Steps

Supervised Learning Algorithms: K-Nearest Neighbors

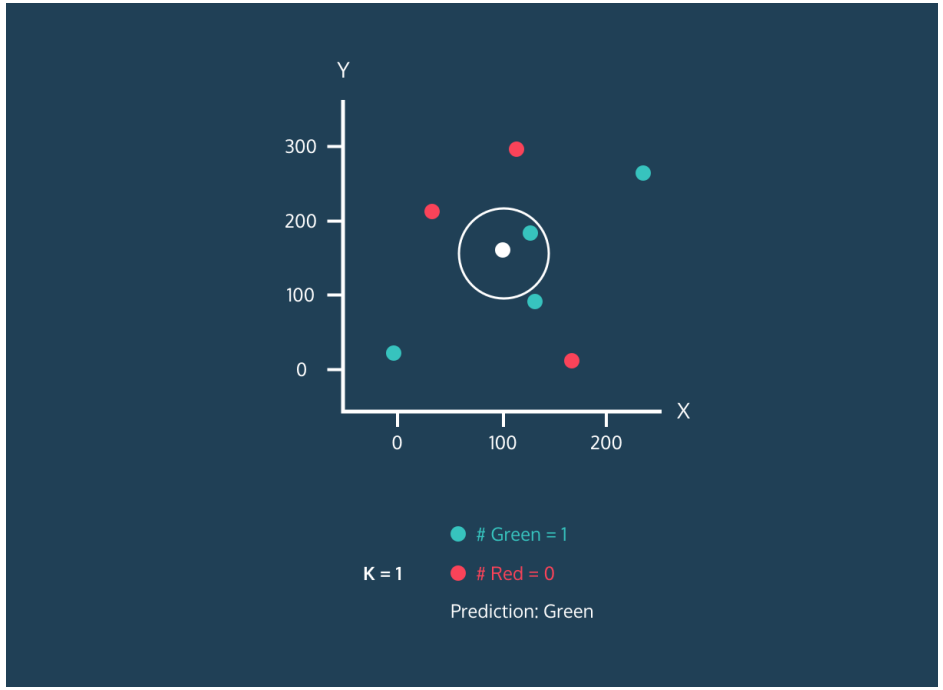
1. **Key Parameters:**
   - k: This is the number of neighbors to consider. It's a hyperparameter that you need to choose. A small k might lead to noise in the prediction, while a large k might smooth out the decision boundaries.
   - Distance Metric: This defines how the "closeness" of data points is calculated. Common options include Euclidean distance, Manhattan distance, Minkowski distance, etc.
2. **Advantages:**
   - Simple to understand and implement.
   - No explicit training phase, making it computationally efficient during training.
   - Can be used for both classification and regression tasks.
3. **Disadvantages:**
   - Can be computationally expensive during prediction, especially with large datasets.
   - Sensitive to the choice of distance metric and value of k.
   - Doesn't learn underlying patterns in the data, which can lead to suboptimal performance in some cases.
4. **Considerations:**
   - Scaling of features is crucial as k-NN is sensitive to the scale of the variables.
   - It's important to choose an appropriate value of k through techniques like cross-validation.

Supervised Learning Algorithms: K-Nearest Neighbors <Example>

Example of *k*-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If *k* = *3* (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If *k* = *5* (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle).



Supervised Learning Algorithms: K-Nearest Neighbors <DEMO>



DEMO: Session 1 - k-Nearest Neighbors (k-NN)

Supervised Learning Algorithms: Commonly Used Regression Algorithms

Regression Algorithms:

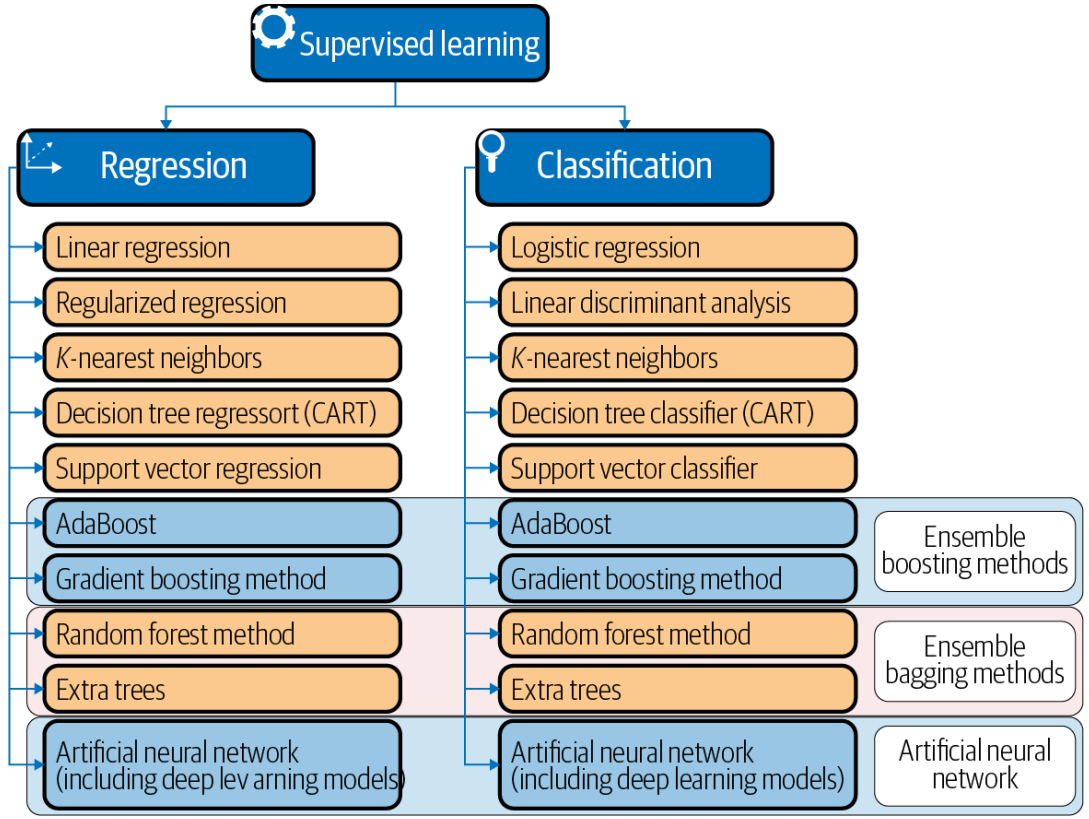
1. **Linear Regression**: Predicts a continuous output variable based on the input features by fitting a linear equation to the observed data.
2. **Ridge Regression (L2 regularization)**: Similar to linear regression but adds a penalty term to the coefficients to prevent overfitting.
3. **Lasso Regression (L1 regularization)**: Similar to ridge regression, but uses the absolute values of coefficients, which can lead to sparsity in the model.
4. **ElasticNet**: A combination of L1 and L2 regularization that balances between Ridge and Lasso regression.
5. **Decision Tree Regression**: Uses a decision tree to predict continuous values.
6. **Random Forest Regression**: Ensemble method that uses multiple decision trees to improve accuracy and control overfitting.
7. **Gradient Boosting Regression**: Builds an additive model in a forward stage-wise manner, optimizing for the residual errors.
8. **Support Vector Regression (SVR)**: Uses support vector machines to perform regression.

Supervised Learning Algorithms: Commonly Used Classification Algorithms

Classification Algorithms:

1. **Logistic Regression**: Used for binary classification problems, models the probability that a given instance belongs to a particular category.
2. **K-Nearest Neighbors (KNN)**: Classifies data points based on the majority class of their k-nearest neighbors.
3. **Support Vector Machines (SVM)**: Finds the hyperplane that best separates classes in a high-dimensional space.
4. **Naive Bayes**: Applies Bayes' theorem with the "naive" assumption that features are independent, commonly used for text classification.
5. **Decision Tree Classification**: Divides the data into subsets based on the value of features to make categorical predictions.
6. **Random Forest Classification**: Ensemble method that uses multiple decision trees for classification.
7. **Gradient Boosting Classification**: Builds an ensemble of weak learners (usually decision trees) in a forward stage-wise manner.
8. **Neural Networks (Deep Learning)**: Multi-layered networks of interconnected nodes used for complex classification tasks.
9. **XGBoost, LightGBM, CatBoost** (Gradient Boosting variations): Highly optimized implementations of gradient boosting algorithms.
10. **Adaboost**: Combines multiple weak learners to create a strong learner.

Supervised Learning Algorithms: Classification and Regression Algorithms



Source: www.oreilly.com

Supervised Learning Algorithms: Parametric vs. Nonparametric

There are two main types of machine learning algorithms: **parametric** and **nonparametric**.

Parametric Algorithms

Parametric algorithms are based on a mathematical model that defines the relationship between inputs and outputs. This makes them more restrictive than nonparametric algorithms, but it also makes them faster and easier to train. Parametric algorithms are most appropriate for problems where the input data is well-defined and predictable.

Examples of parametric algorithms: Linear regression, Logistic regression, and Neural networks.

Nonparametric Algorithms

Nonparametric algorithms are not based on a mathematical model; instead, they learn from the data itself. This makes them more flexible than parametric algorithms but also more computationally expensive. Nonparametric algorithms are most appropriate for problems where the input data is not well-defined or is too complex to be modeled using a parametric algorithm.

Examples of nonparametric algorithms: Decision trees, K-NN.

Supervised Learning: Evaluation Metrics
<Classification Metrics>

Question: When dealing with a classification problem, how can we **evaluate** its **effectiveness** and thus conduct a **comparative** study with other **classifiers**?

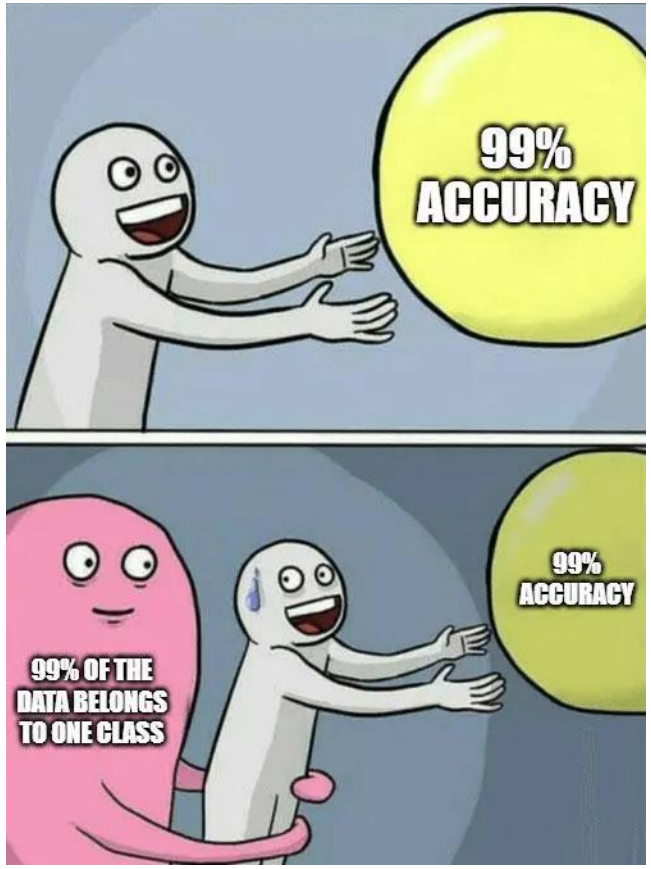
Answer: We calculate the correct classified data points and divide it by the total number of data points in the dataset.

$$accuracy = \frac{Correctly\ classified\ datapoints}{Total\ number\ of\ all\ datapoints}$$

Another Question: Is the accuracy metric enough? Are there potential problems with it?

Short Answer: No, It is not enough and Yes, There is potential problems

Supervised Learning: Evaluation Metrics
<Classification Metrics>



Supervised Learning: Evaluation Metrics <Classification Metrics>

Some potential **problems** with relying only on **accuracy**

Problem	Description	Examples
Imbalanced Datasets	In datasets where one class significantly outweighs the others (class imbalance), a model can achieve high accuracy by simply predicting the majority class most of the time.	In a medical dataset where only 5% of patients have a rare disease, a model that always predicts "no disease" would still achieve 95% accuracy.
Misleading in Rare Events	In applications where detecting rare events is crucial (e.g., fraud detection, disease diagnosis), accuracy can be misleading. A high accuracy score may not reflect the model's ability to correctly identify these critical cases. Accuracy may not reflect the model's ability to detect rare, but important, occurrences.	In fraud detection, a model that misses rare instances of fraud (false negatives) could have high accuracy but would be ineffective.

Supervised Learning: Evaluation Metrics <Classification Metrics>

Question: What is the solution? Are there other alternatives to Accuracy?

Answer: There are other metrics that can be used: Recall/Sensitivity, Precision, Specificity, F-Score, ROC-AUC Curve...etc. **WAIT!!!**

Before explaining those metrics, let's first understand the confusing "**Confusion Matrix**"

Hold on to your laptops and let's see what is the Confusion Matrix

Supervised Learning: Evaluation Metrics
<Classification Metrics>

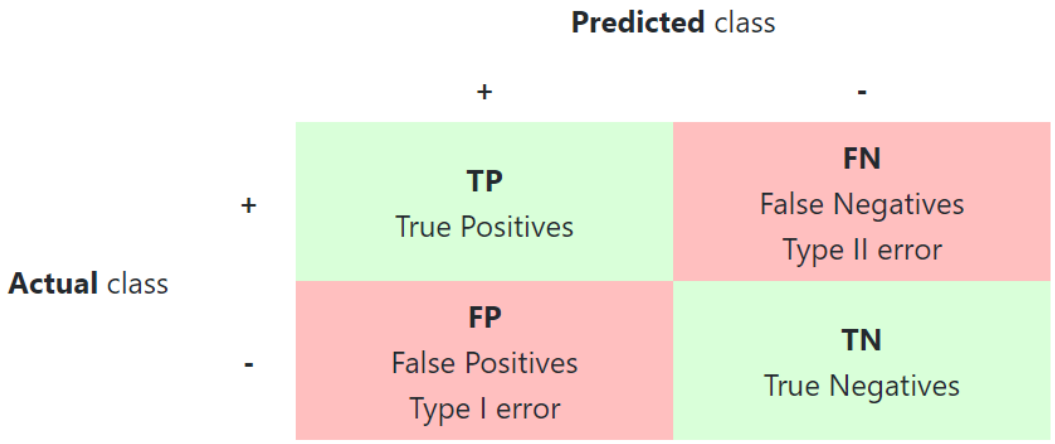


Fig. Confusion Matrix Layout

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes

Tab. Basic classification metrics

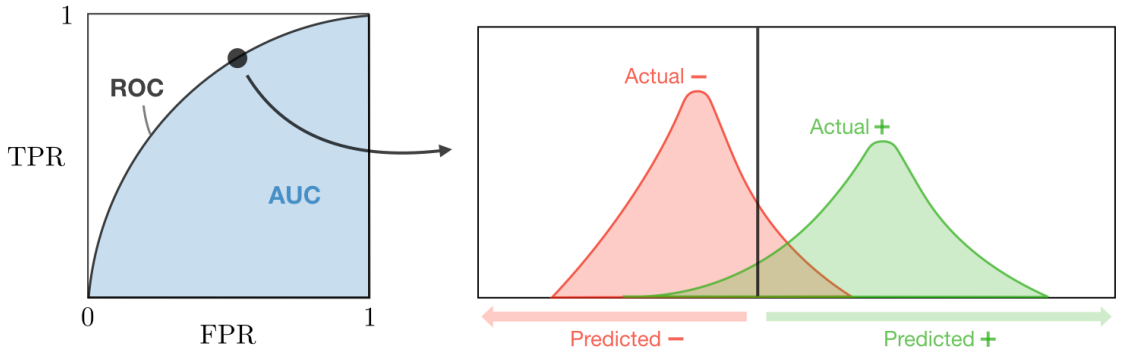
WANT SOME MORE METRICS? LET'S SEE ROC AND AUC CURVES

Supervised Learning: Evaluation Metrics
<Classification Metrics>

ROC: The receiver operating curve, also noted as ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are summed up below:

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	1-specificity

AUC: The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Source: https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Supervised Learning: Evaluation Metrics
<Classification Metrics>

Example Scenario:

Let's say we have a binary classification problem for a medical test that identifies a disease:
Out of 100 actual cases of the disease:
The test correctly identifies 80 as positive (True Positives).
The test incorrectly identifies 20 as negative (False Negatives).
Out of 100 non-cases (healthy individuals):
The test correctly identifies 90 as negative (True Negatives).
The test incorrectly identifies 10 as positive (False Positives).

Find the confusion matrix and compute the accuracy, recall, precision, and specificity!!

Supervised Learning: Evaluation Metrics
<Regression Metrics>

These regression metrics serve different purposes in evaluating the performance of regression models. **MAE**, **MSE**, and **RMSE** focus on the accuracy of predictions, while **R^2 (R-squared)** assesses the overall goodness of fit of the model. Depending on the specific characteristics of the problem, different metrics may be more appropriate for evaluation.

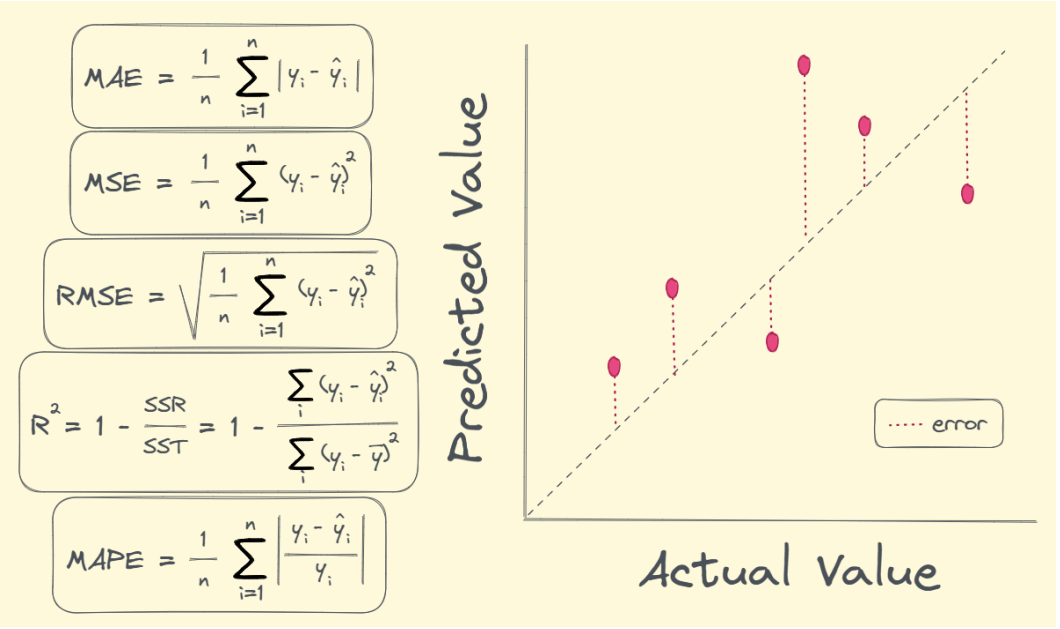


Fig. Regression evaluation metrics

Unsupervised Learning Algorithms

- This is called unsupervised learning because unlike supervised learning, there are no given labels/outputs and the machine itself finds the answers.
- **Motivation:** The goal of unsupervised learning is to find and discover hidden patterns in unlabeled data $\{x^{(1)},...,x^{(m)}\}$.
- **Example:** Clustering similar documents based on content/text.

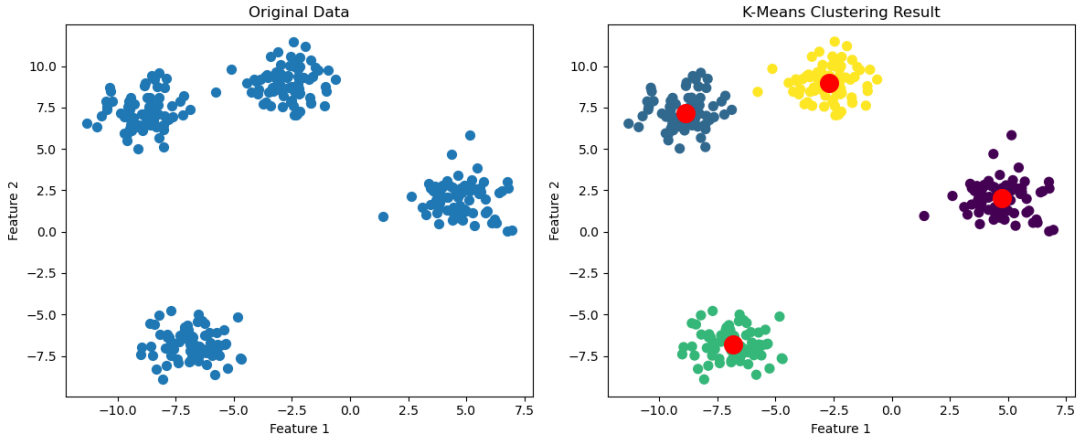


Fig. A simple illustration of the clustering technique

Unsupervised Learning Algorithms

- Types of unsupervised learning algorithms:
  - **Clustering:** Clustering aims to group similar data points together based on their features or attributes. The goal is to identify natural clusters in the data. **Examples:** Customer Segmentation: Grouping customers based on purchasing behavior. Image Segmentation: Dividing an image into regions with similar characteristics.
  - **Dimensionality Reduction:** Dimensionality reduction techniques aim to reduce the number of features or variables in the data while preserving important information.
  - **Association:** Association aims to discover relationships or patterns in data. It identifies sets of items that frequently occur together. **Examples**: Market Basket Analysis: Identifying which products tend to be bought together in a shopping basket. Recommender Systems: Suggesting products or content based on user behavior.

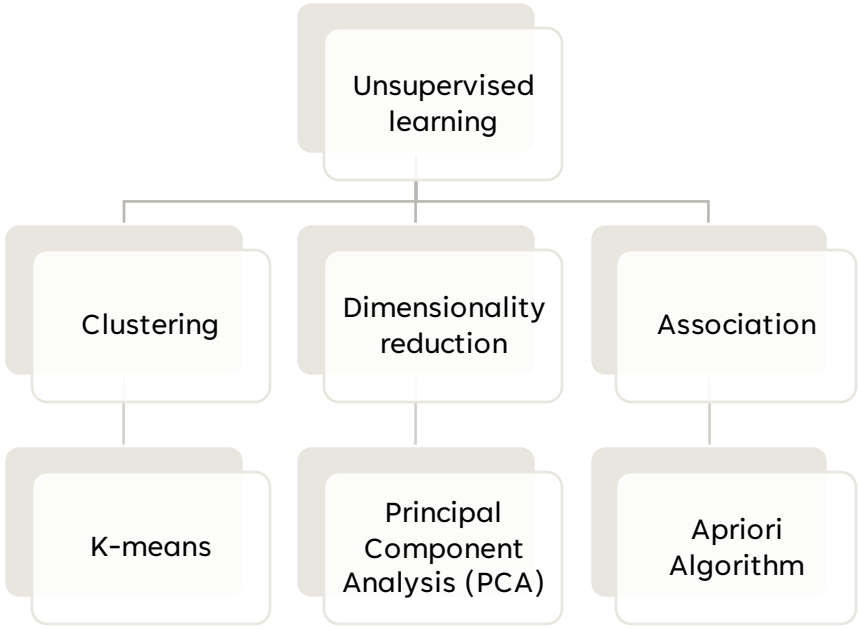


Fig. Some unsupervised learning algorithms

Unsupervised Learning Algorithms: Clustering

- Clustering in machine learning is a technique used to group similar data points together based on their features or attributes. The goal is to identify natural clusters in the data, where data points within a cluster are more similar to each other compared to points in different clusters.
- Clustering is an unsupervised learning technique, which means that the algorithm does not rely on labeled data. It learns from the inherent structure in the input data without any specific guidance.
- As the examples are unlabeled, clustering relies on unsupervised machine learning. If the examples are labeled, then clustering becomes classification.
- Clustering Use Cases:
  - market segmentation
  - social network analysis
  - search result grouping
  - medical imaging
  - image segmentation
  - anomaly detection

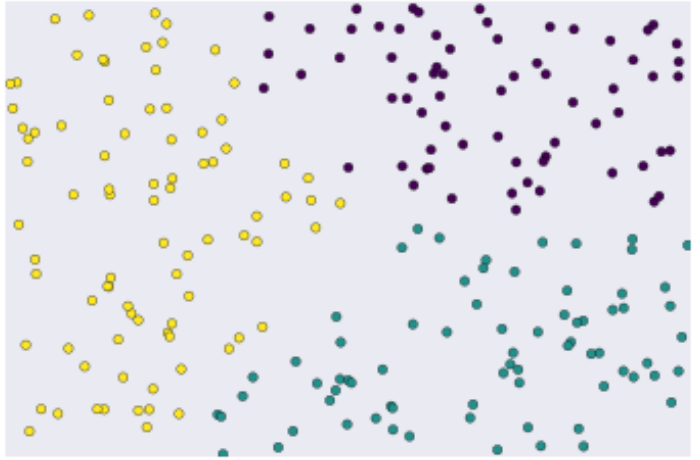
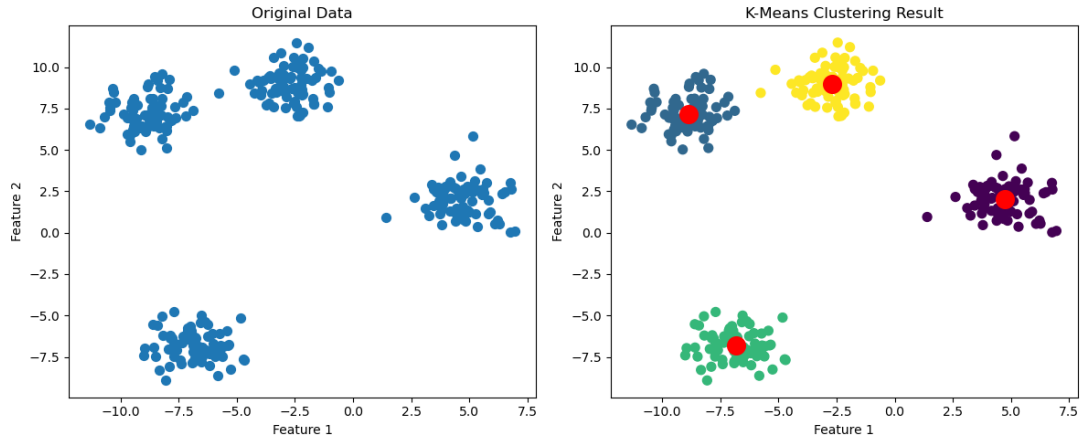


Fig. Unlabeled examples grouped into three clusters.

https://nicola17.github.io/tfjs-tsne-demo/

Unsupervised Learning Algorithms: Clustering using K-Means Algorithm

The K-Means algorithm is a popular clustering algorithm (centroid-based) used in machine learning. It aims to partition a dataset into K distinct, non-overlapping subsets (or clusters) based on the similarity of data points. The "K" in K-Means refers to the number of clusters that the algorithm aims to find.



Unsupervised Learning Algorithms: Clustering using K-Means Algorithm

How does the k-mean algorithm work?

We note $\mathbf{c^{(i)}}$ the cluster of data point **i** and $\boldsymbol{\mu_j}$ the center of cluster **j**.

Algorithm: After randomly initializing the cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$. The k-means algorithm repeats the following step until convergence:

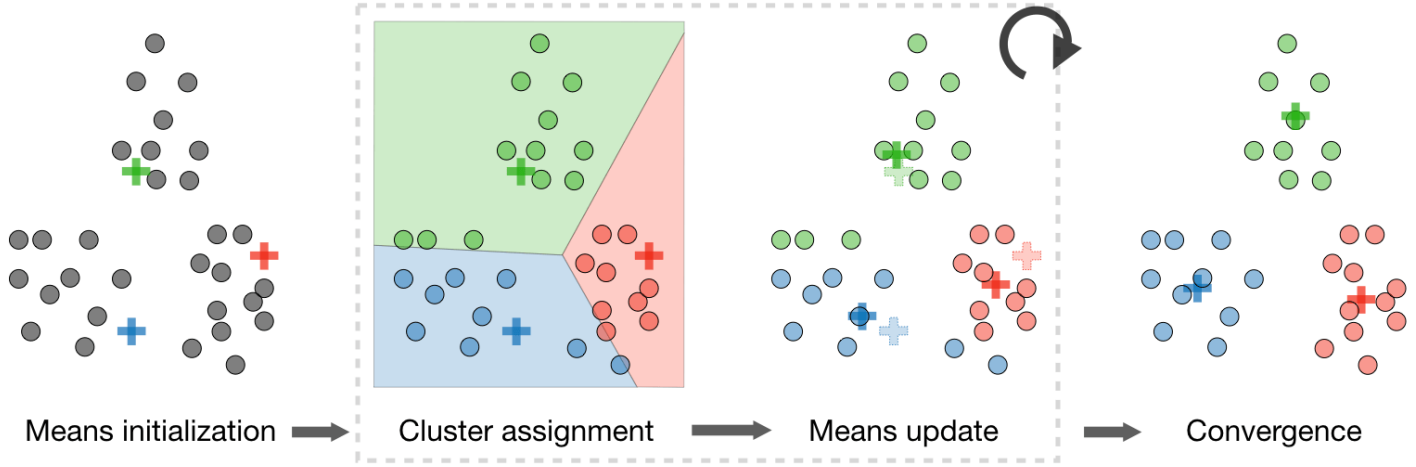
$$c^{(i)} = \arg\min_j ||x^{(i)} - \mu_j||^2$$

and

$$\mu_j = \frac{\sum_{i=1}^{m} 1_{\{c^{(i)}=j\}} x^{(i)}}{\sum_{i=1}^{m} 1_{\{c^{(i)}=j\}}}$$

Unsupervised Learning Algorithms: Clustering using K-Means Algorithm

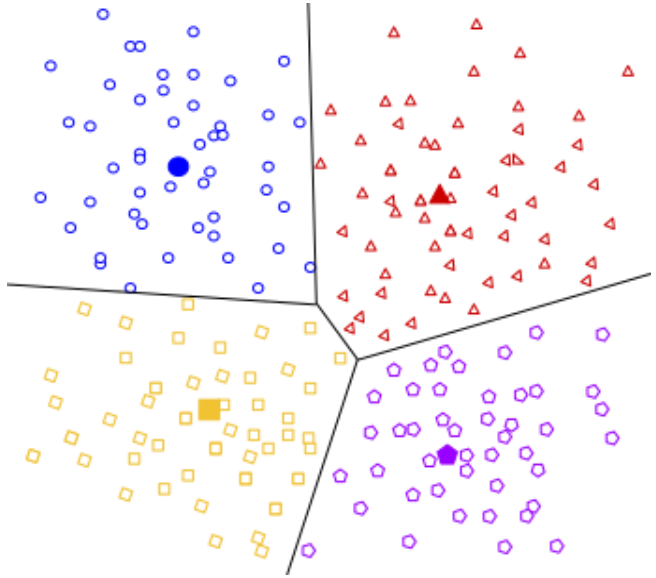
How does the k-mean algorithm work?



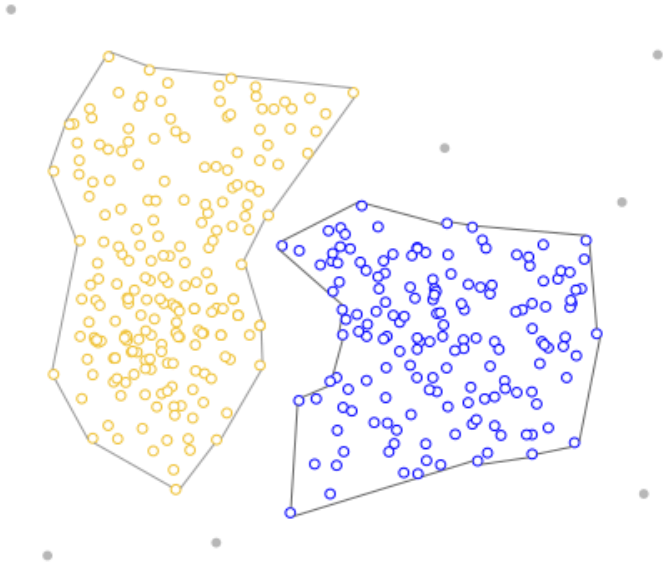
In order to see if the algorithm **converges**, we look at the **distortion function** defined as follows:

$$J(c,\mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

Unsupervised Learning Algorithms: Clustering Algorithms



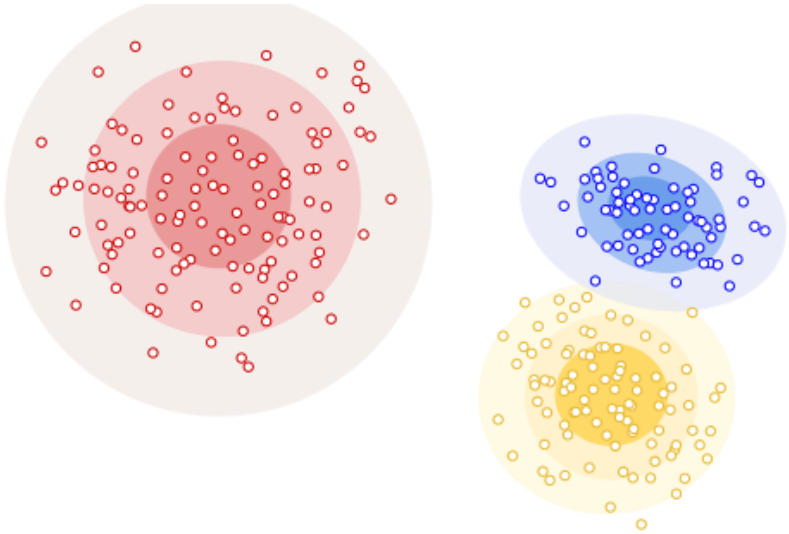
Centroid-based Clustering



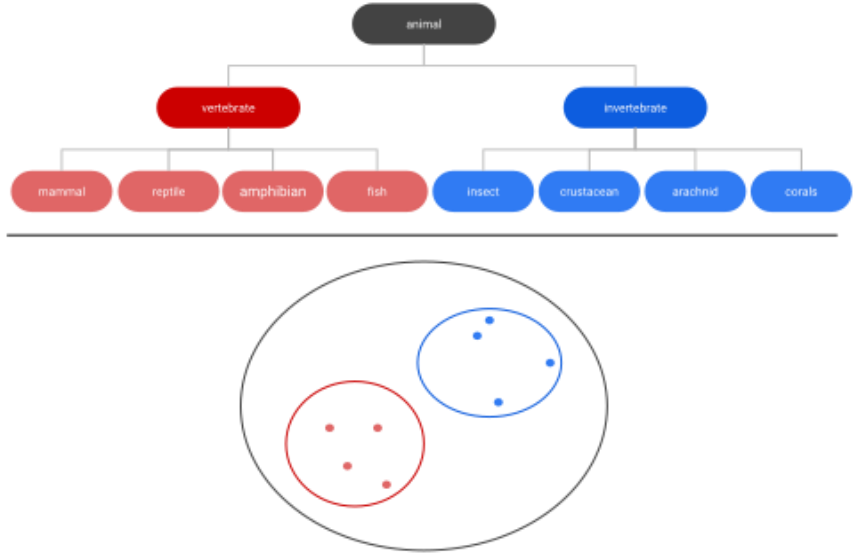
Density-based Clustering

For more details: A Comprehensive Survey of Clustering Algorithms

Unsupervised Learning Algorithms: Clustering Algorithms



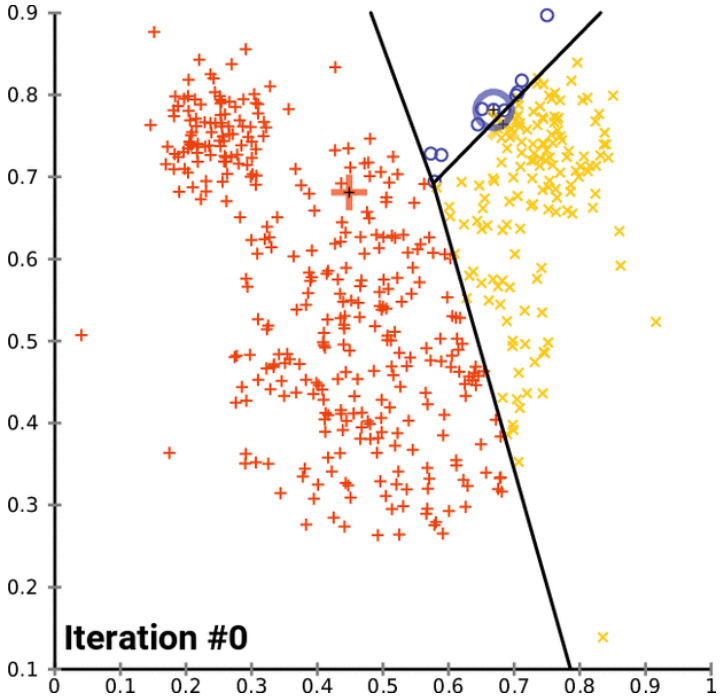
Distribution-based Clustering



Hierarchical Clustering

For more details: A Comprehensive Survey of Clustering Algorithms

Unsupervised Learning Algorithms: Clustering using K-Means – PYTHON DEMO



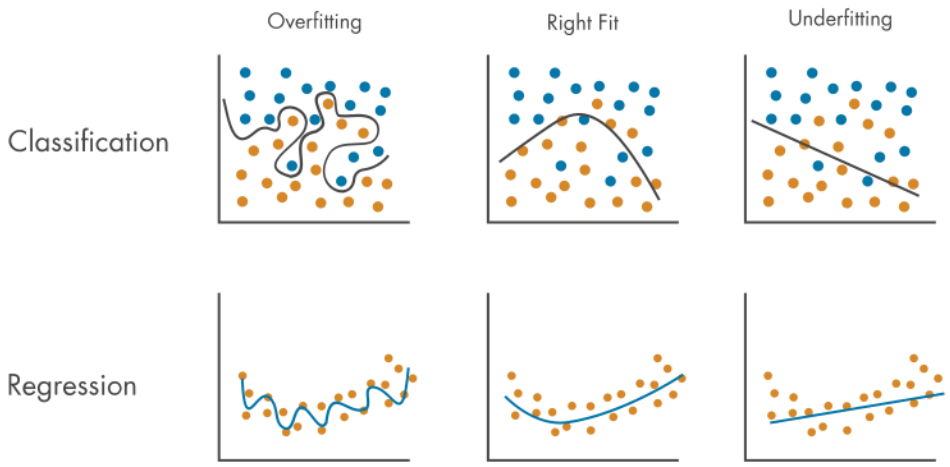
Source: Wikipedia

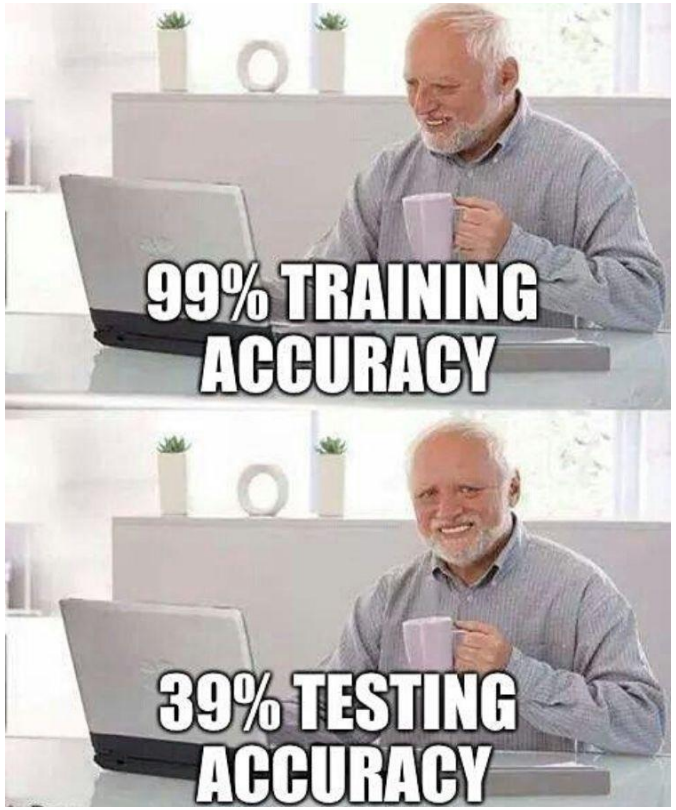
DEMO: Session 1 - Unsupervised Learning Clustering using K-Means Algorithm

Machine Learning: Important Concepts
<Overfitting and Underfitting>

Two **important** concepts:

- **Overfitting** and **underfitting**
- **Bias** and **variance** trade-off

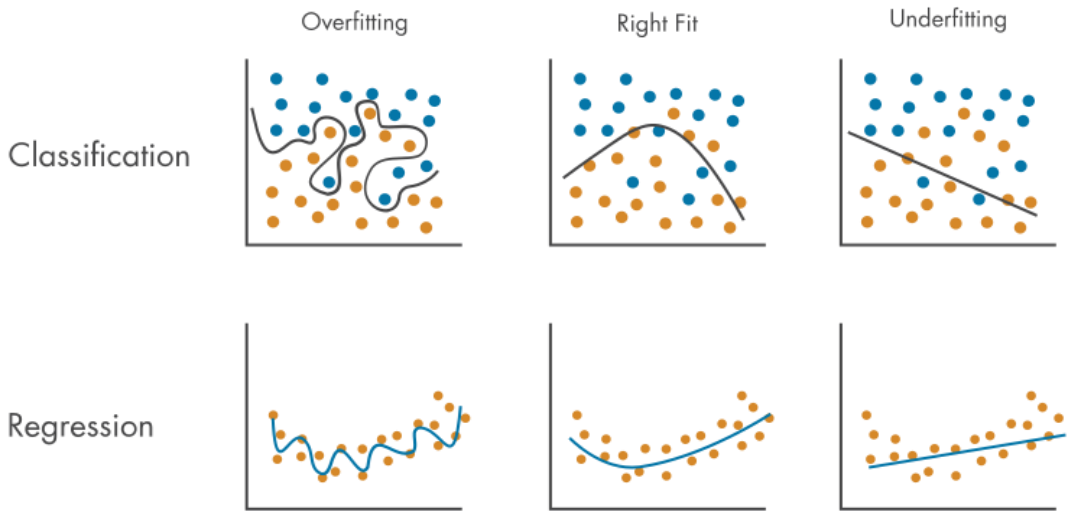




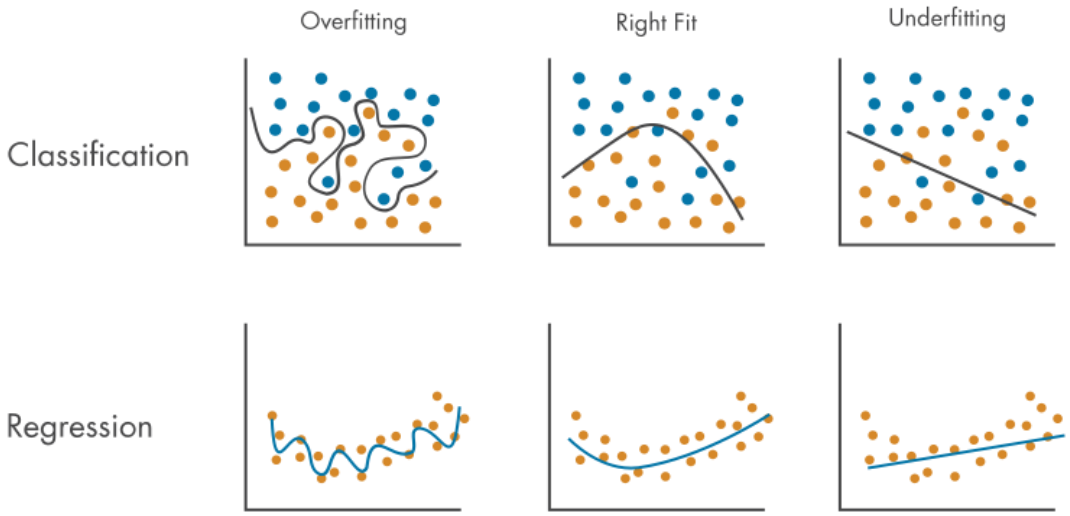
Machine Learning: Important Concepts
<Overfitting and Underfitting>

Underfitting and **overfitting** are two common problems in machine learning that occur when a model is not able to generalize well to new, unseen data.

- **Underfitting:** Underfitting occurs when a model is too simple to capture the underlying trend of the data. It performs poorly on both the training data and unseen data.
  - Causes of underfitting:
    - Using a very simple model (e.g., linear regression for a non-linear problem).
    - Not having enough features in the model.
    - Using too few training samples.

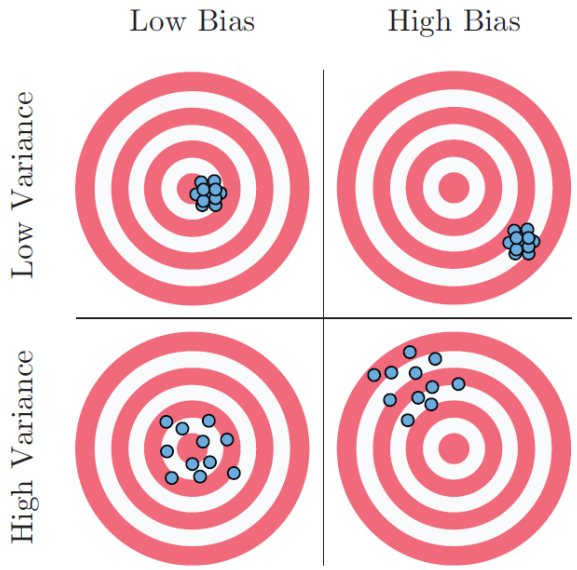


Machine Learning: Important Concepts
<Overfitting and Underfitting>

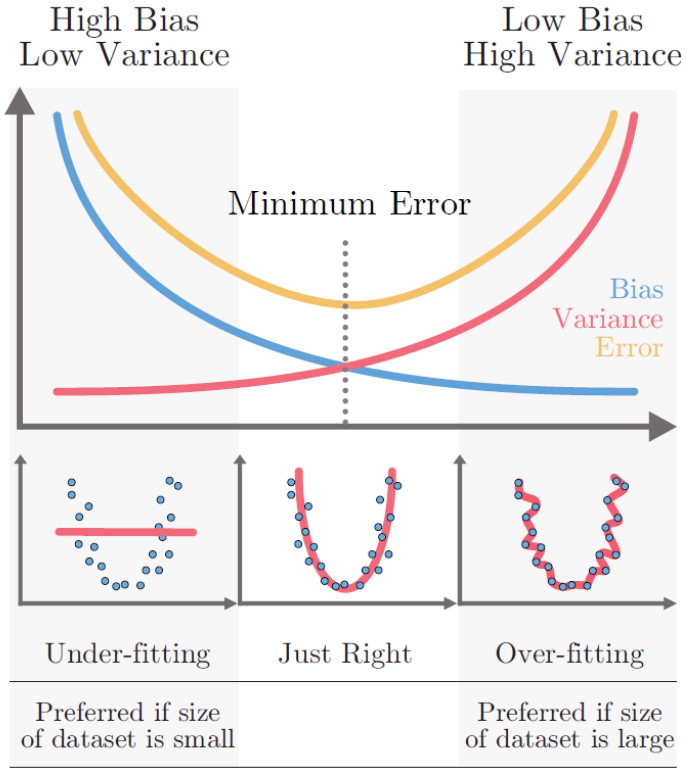


- **Overfitting:** Overfitting happens when a model learns the training data too well, capturing noise or random fluctuations that are not representative of the true underlying pattern. As a result, it performs well on the training data but poorly on unseen data.
  - Causes of underfitting:
    - Using a very complex model (e.g., a high-degree polynomial for a simple problem).
    - Having too many features relative to the number of training samples.

Machine Learning: Important Concepts
<Bias-Variance Tradeoff>



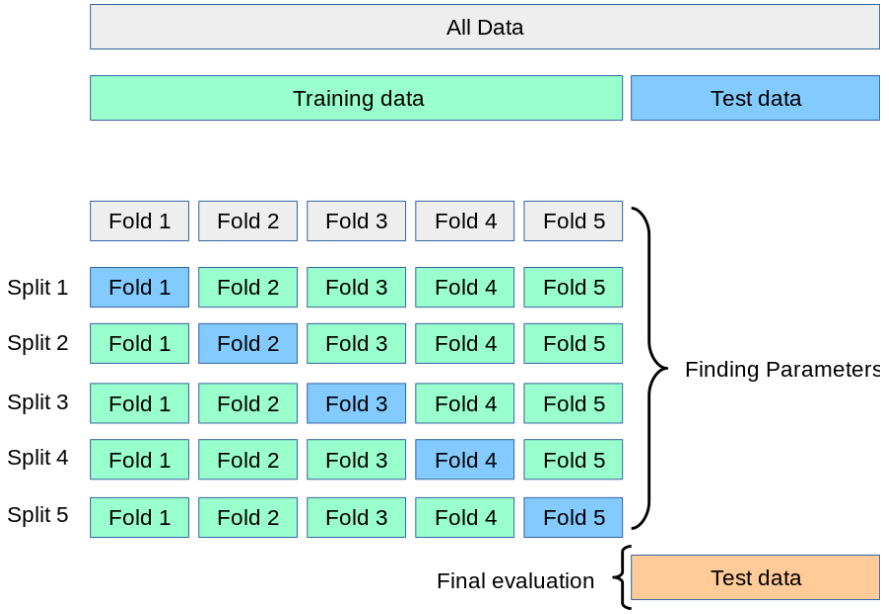
- **What is Bias:**
  - Error between average model prediction and ground truth
  - The bias of the estimated function tells us the capacity of the underlying model to predict the values
- **High Bias:** Overly-simplified model, Underfitting, and High error on both train and test sets
- **What is Variance?**
  - Average variability in the model prediction for the given dataset
  - The variance of the estimated function tells you how much the function can adjust to the change in the dataset
- **High Variance:** Overly-complex model, Overfitting, Low error on train data and high on test, Starts modelling the noise in the input.



Machine Learning: Important Concepts
<K-Fold Cross Validation>

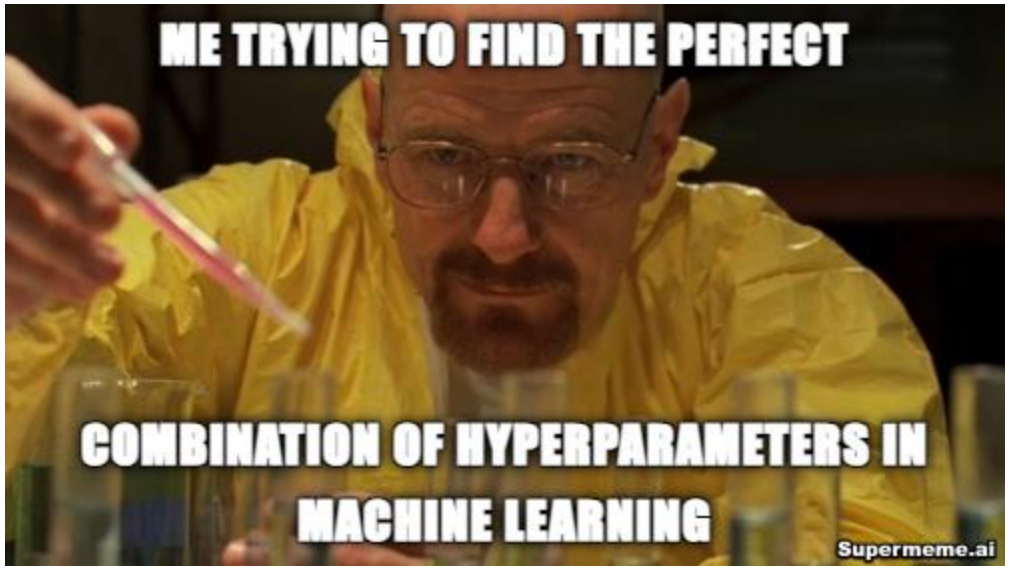
K-fold cross-validation: One of the techniques to evaluate machine learning models on limited data samples and one of the employed techniques to detect overfitting and how well a model will generalize to new, unseen data.

- **Idea:** The k-fold cross-validation (CV) refers to the **k** number of the folds or subset division of the dataset (The model is then trained and evaluated **k** times, using each fold as the testing set once and the remaining **k-1** folds as the training set). It is generally a technique used in machine learning to evaluate the performance of the models on the entire dataset, it ensures that every data point from the dataset has the chances of appearing in both training and testing sets, and thus it is one of the best approaches to evaluate the models' performance.



Machine Learning: Important Concepts
<Hyper-Parameters Optimization/Tuning>

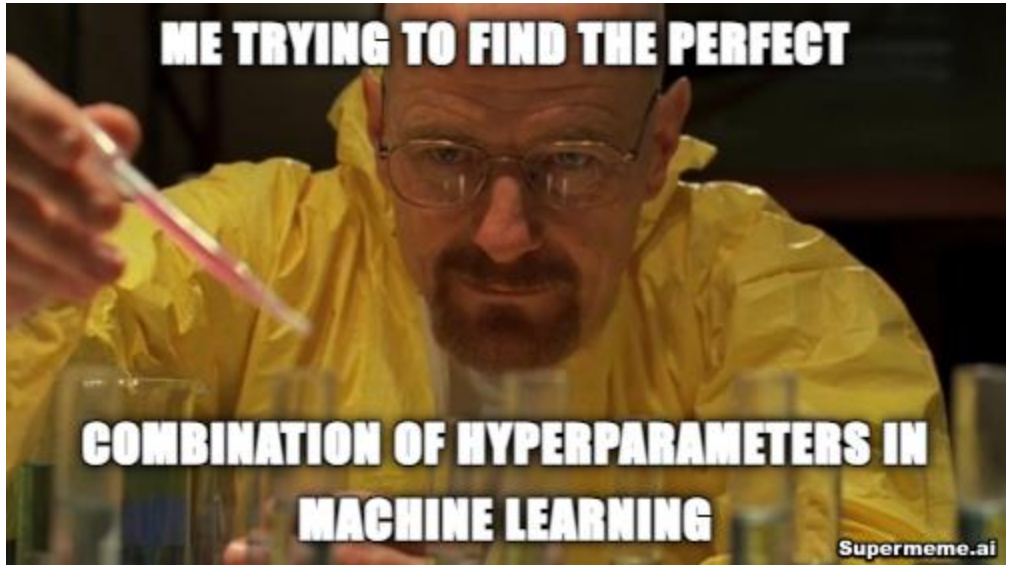
Problem: We have seen so far different machine learning algorithms; linear regression, logistic regression, decision tree, ...etc. In linear regression for example, we talked about something called "learning rate", in decision trees, we talked about the criterion (to measure the quality of a split) like : gini impurity and entropy.
Question: How to choose the best values for these hyperparameters



Machine Learning: Important Concepts
<Hyper-Parameters Optimization/Tuning>

Definition: A hyperparameter is a configuration setting for a machine-learning model that is not learned from the data. Instead, it is set prior to training and remains constant during the training process. These parameters are essential for controlling the learning process, but they are not learned from the data itself.

In contrast, the parameters of a machine learning model are the variables that the model learns from the training data. For example, in a linear regression model, the coefficients for each feature are learned parameters.



Machine Learning: Important Concepts
<Hyper-Parameters Optimization/Tuning>

Hyperparameters of some machine learning algorithms:

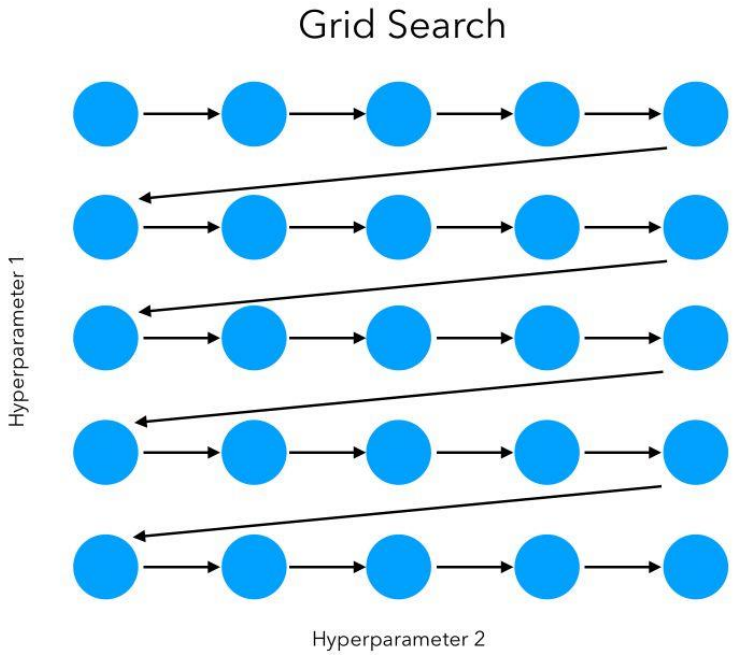
- **Linear regression:** Linear regression is not a complex algorithm; therefore, it does not have hyperparameters except the learning rate (alpha). However, there are some advanced regression algorithms that have more hyperparameters.
- **Logistic Regression:** Solver, penalty, and 'C' regularization strength.
- **Decision Trees:** Criterion, Maximum depth.
- **K-Nearest Neighbour:** Number of Neighbours 'K'.

class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

class `sklearn.linear_model.`**`LogisticRegression`**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Machine Learning: Important Concepts
<Hyper-Parameters Optimization/Tuning>

- **Hyperparameter optimization** is the process of **finding** the **optimal hyperparameters** for a machine-learning model in order to obtain the optimal performance of the model. **Hyperparameters** are parameters not learned from the data but the user sets them before training the model. **Hyperparameter optimization** is a crucial task in the process of building machine learning models because the performance of a machine learning model heavily depends on the chosen values for its **hyperparameters**. Through a systematic exploration of various **hyperparameter** combinations, we can identify the exact values that yield optimal performance for a given task or problem.
- **Several approaches** to hyperparameter optimization/tuning, including **manual tuning**, **grid search**, **random search**, and some other advanced techniques like **Bayesian optimization** and **genetic algorithms.**



Source: https://maelfabien.github.io/machinelearning/Explorium4/#what-is-hyperparameter-optimization

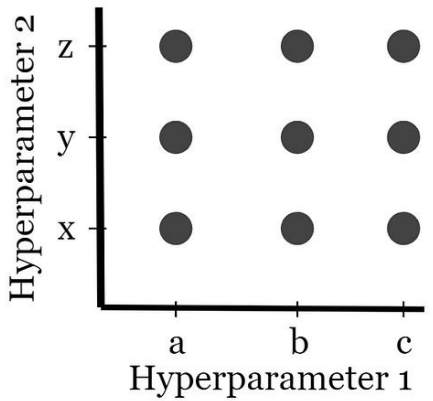
Machine Learning: Important Concepts
<Hyper-Parameters Optimization/Tuning>

- **Grid Search:** In grid search, a set of possible values for each hyperparameter is specified by the user, and the algorithm comprehensively evaluates all possible combinations of hyperparameters.
- **Random Search:** Random search randomly selects hyperparameter values from predefined ranges. It doesn't cover the entire search space like grid search but can be more efficient in practice, especially if some hyperparameters are less influential.
- **Note:** There are some libraries and tools available that can help with hyperparameter optimization/tuning, such as **'GridSearch'** and **'RandomSearch'** in scikit-learn library, **Optuna**, **Hyperopt**, and **Ray Tune**.

Grid Search

Pseudocode

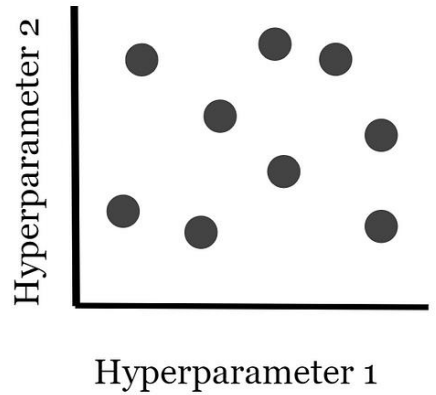
```
Hyperparameter_One = [a, b, c]
Hyperparameter_Two = [x, y, z]
```



Random Search

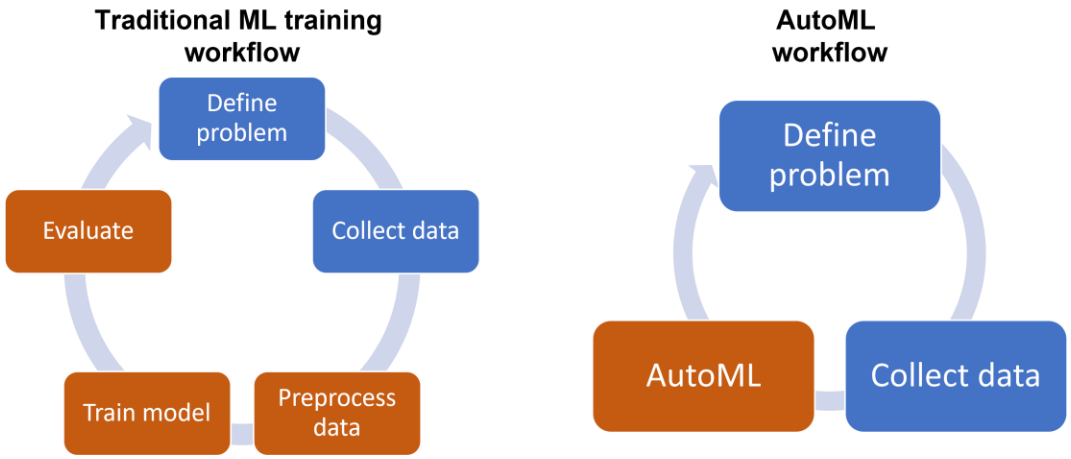
Pseudocode

```
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```



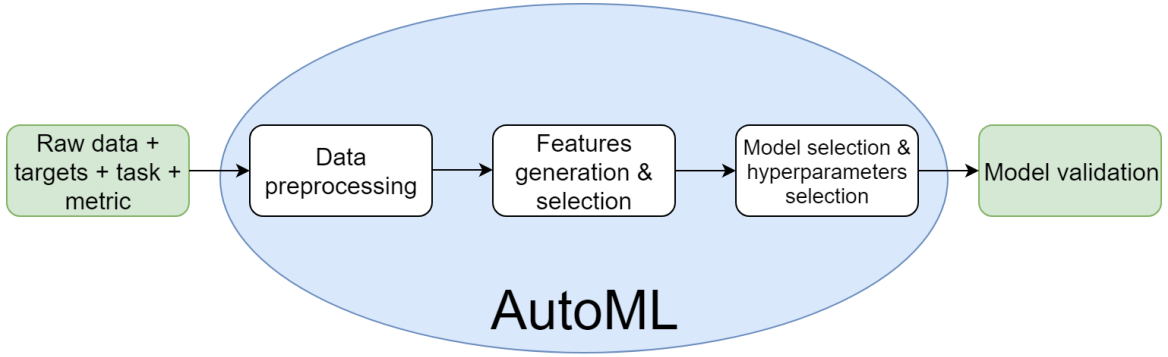
Automated Machine learning (AutoML)

Automated machine learning or simply AutoML, refers to the process of automating the end-to-end process of applying machine learning to real-world problems. It aims to make machine learning more accessible to individuals and organizations with limited expertise in data science and machine learning.

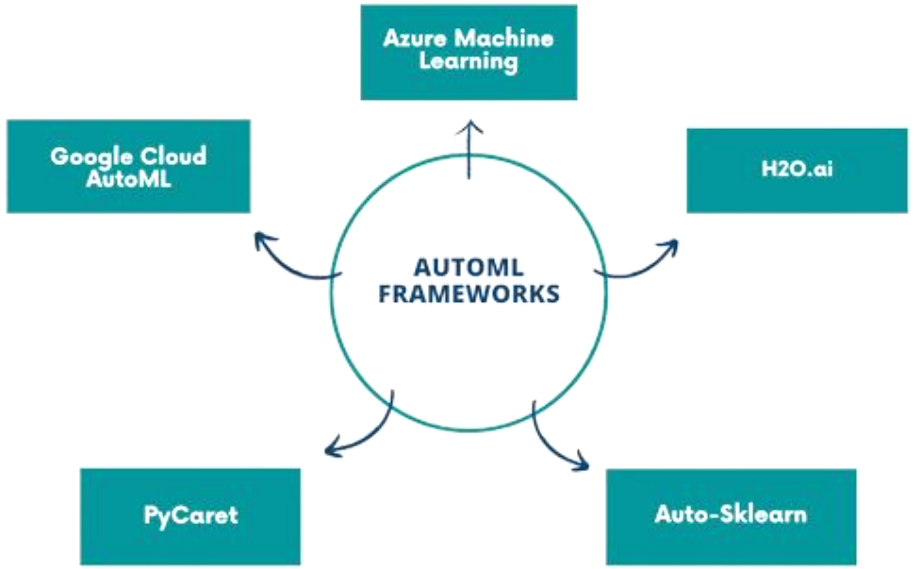


Source: learn.microsoft.com

Automated Machine learning (AutoML)
< Key Aspects >



Automated Machine learning (AutoML)
< Some Frameworks >



Automated Machine learning (AutoML)
<DEMO>

DEMO: Session 1 – Automated Machine Learning AutoML

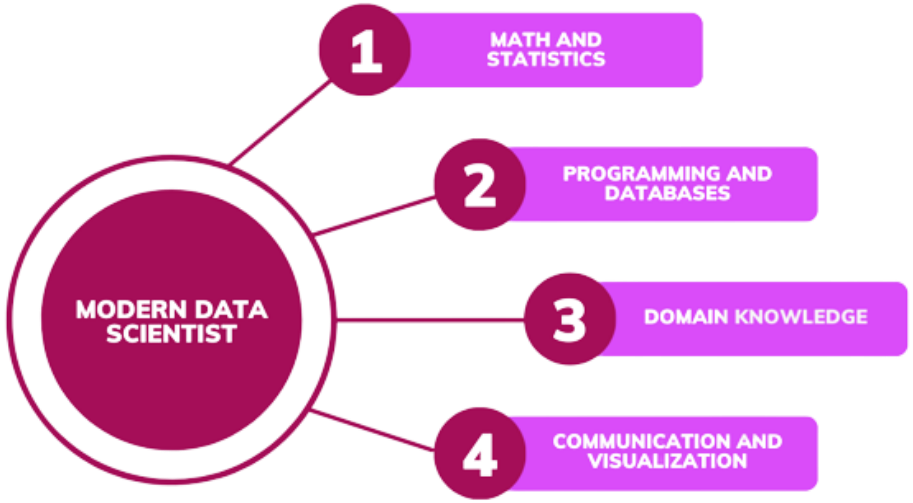
Automated Machine learning (AutoML)
< AutoML vs Data scientists >

Will AutoML replace data scientists and machine learning engineers?

Short Answer: No, but…

Automated Machine learning (AutoML)
< AutoML vs Data scientists >

- **AutoML** will not replace data scientists.
- Data scientists must be specialized and educated much more than before.
- **AutoML** will boost productivity and make the work of a data scientist more interesting and more challenging.
- Domain knowledge will become more important.
- Scripting will become less important.



Classical Machine Learning: TO-DO Projects

Regression: House prices prediction

Classification: Predict survival on the Titanic