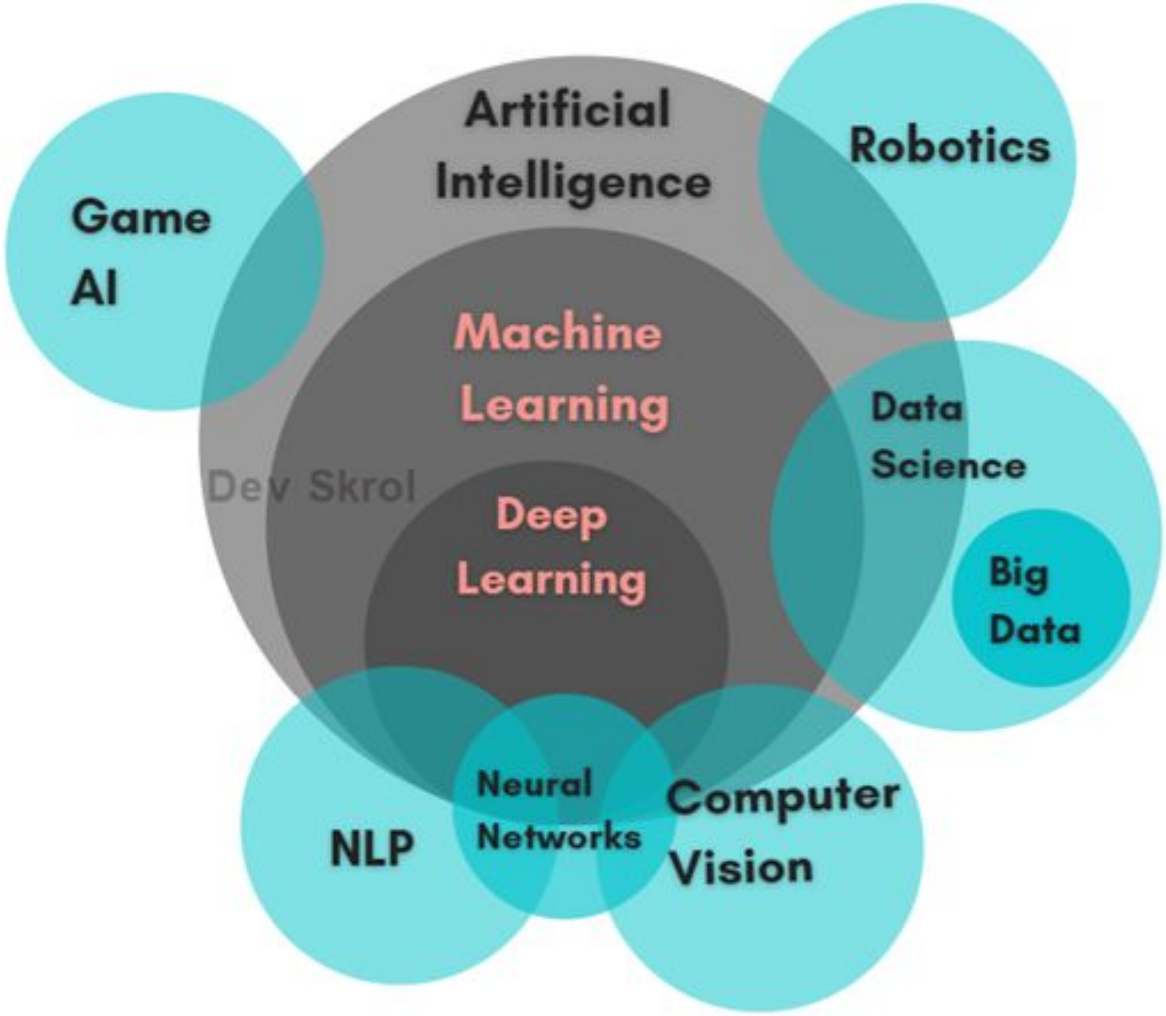
Machine Learning

Course


Artificial Intelligence
Robotics
Game AI
Machine Learning
Data Science
Dev Skrol
Deep Learning
Big Data
Neural Networks
NLP
Computer Vision

Field	Data Type	Simple Description
Machine Learning	Numeric, categorical, textual, images	General algorithms that learn patterns from data and make predictions or group similar data points.
Deep Learning	Complex data (images, text, audio, video)	Advanced techniques using neural networks (like human brain structure) to deeply understand complex patterns in large datasets.
NLP	Textual or speech data	Techniques specialized in understanding, interpreting, and generating human languages.
Computer Vision	Images and videos	Techniques that enable computers to see, recognize, and interpret visual data from the world around us.

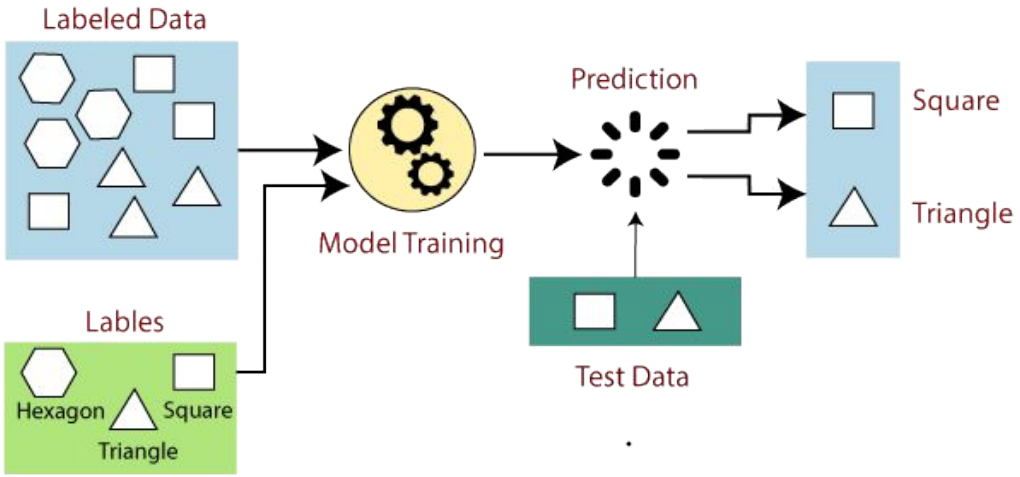
بالتالي، معظم تطبيقات NLP و Computer Vision الحديثة تستخدم Deep Learning، وهي بالتالي جزء من منظومة أكبر هي Machine Learning.

Types of Learning

- **Supervised (inductive) learning**
  - Given: training data + desired outputs (labels)
- **Unsupervised learning**
  - Given: training data (without desired outputs)
- **Semi-supervised learning**
  - Given: training data + a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

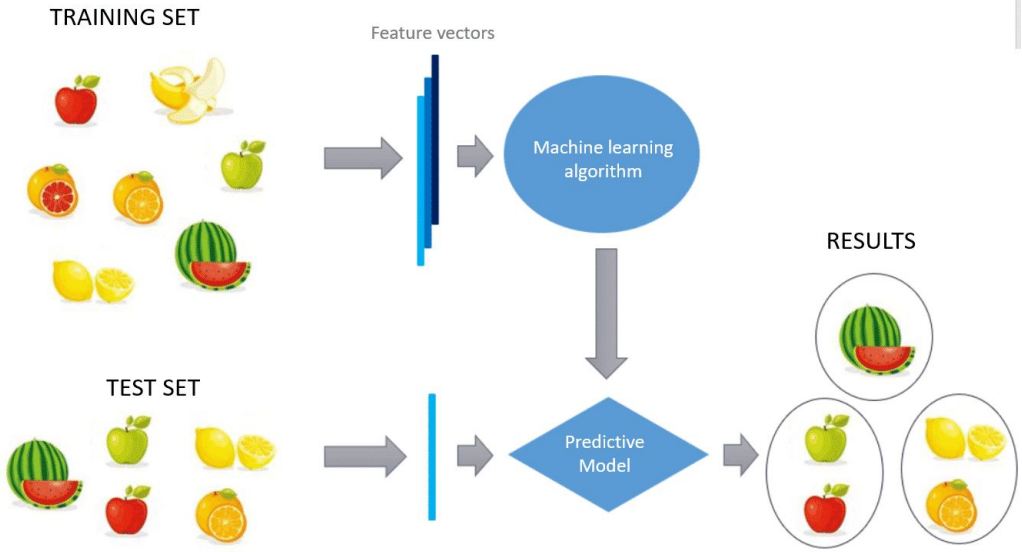
Supervised learning:

- Has the presence of a supervisor as a teacher.
- when we train the machine using data that is labeled.
- Machine is provided with a new set of examples(data) to see how well the algorithm is doing.



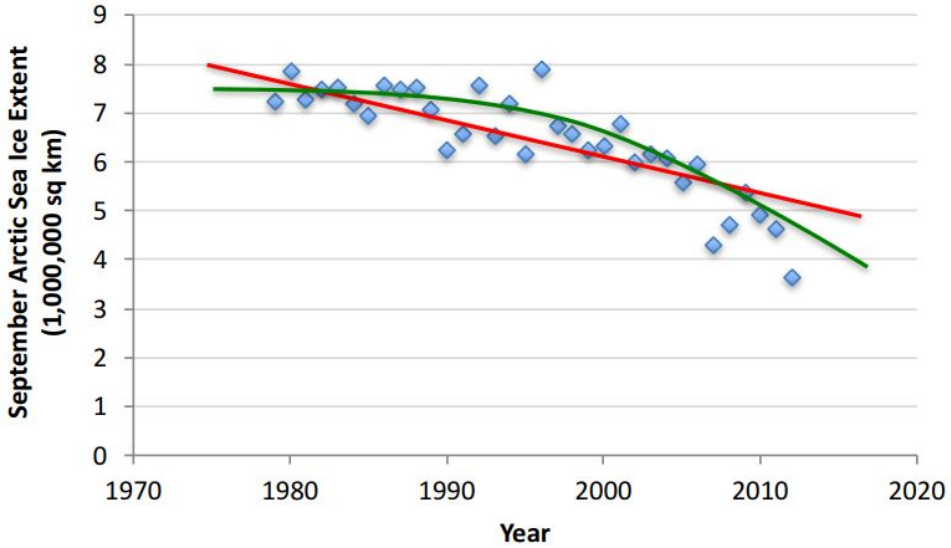
Unsupervised learning :

- Training of a model using information that is not labeled.
- Allowing the algorithm to act on that information without guidance.
- Model is restricted to find the hidden structure in unlabeled data by itself.



Supervised Learning: Regression

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is real-valued == regression



Supervised Learning

enjoy algorithms

Regression

What will be the temperature tomorrow?

Fahrenheit

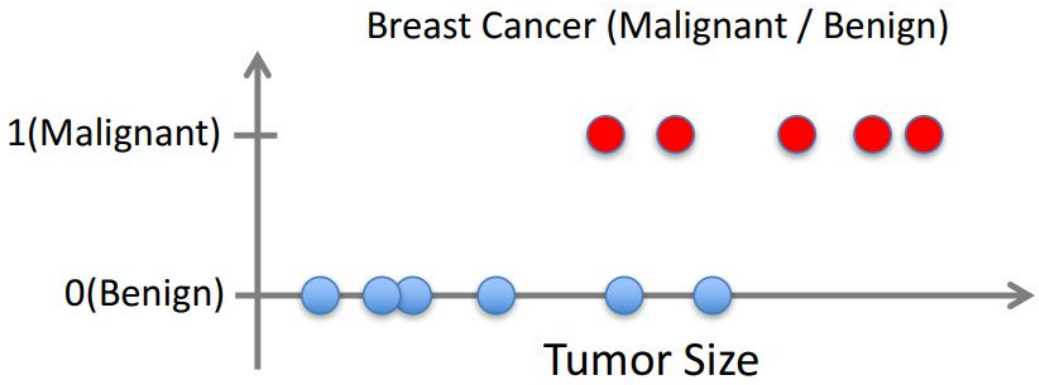
Classification

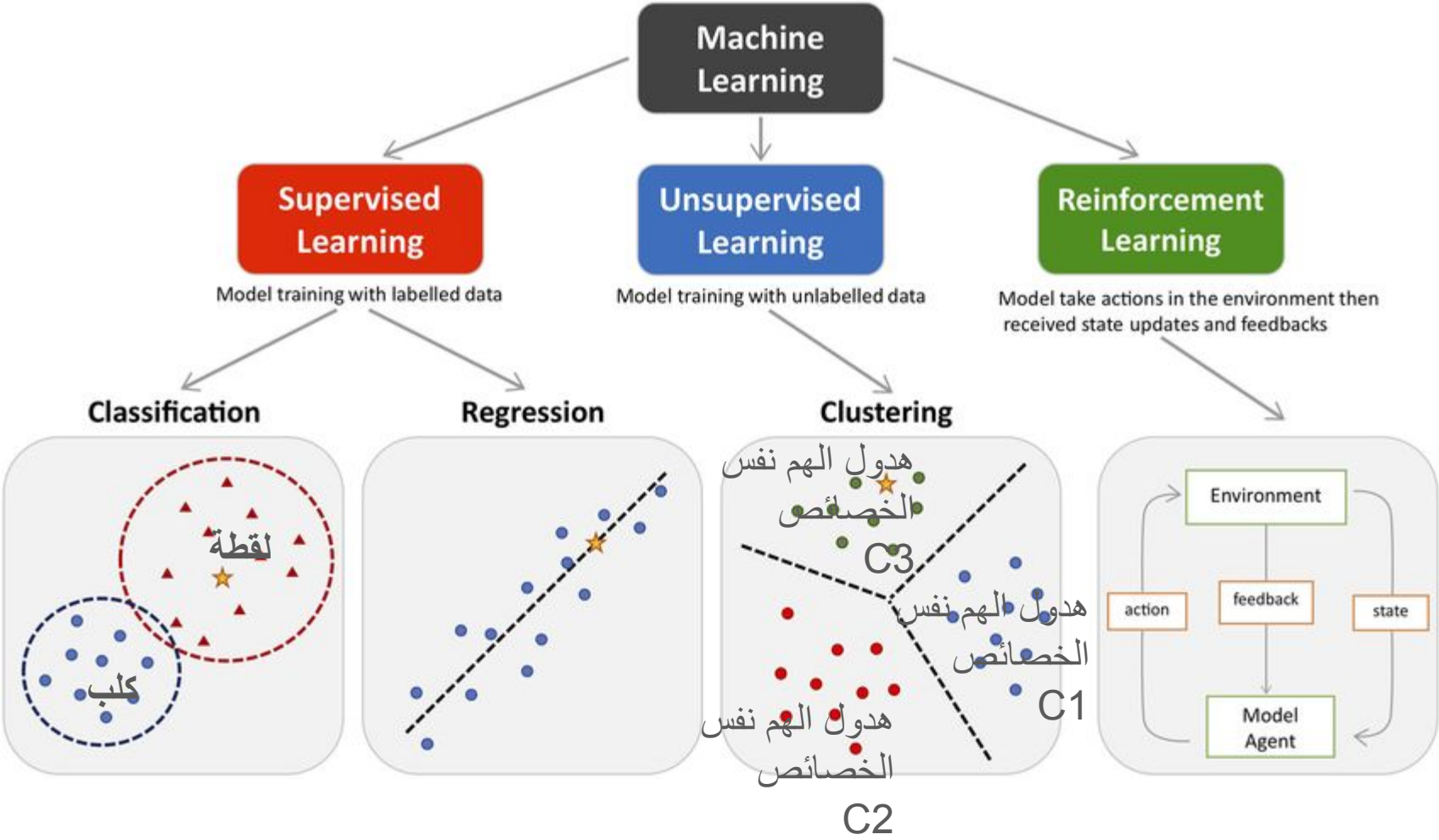
Will it be hot or cold tomorrow?

Fahrenheit

Supervised Learning: Classification

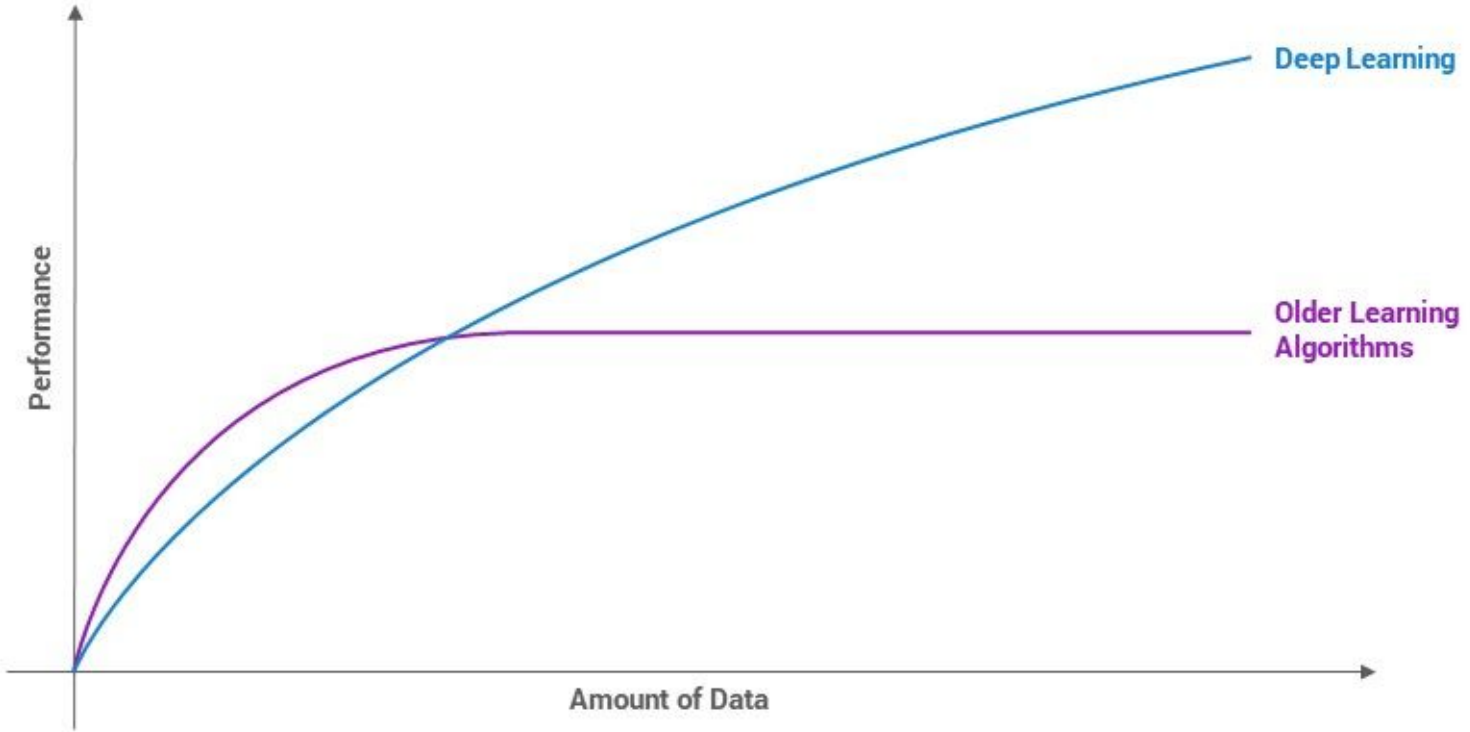
- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification




Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Model training with labelled data
Model training with unlabelled data
Model take actions in the environment then received state updates and feedbacks
Classification
Regression
Clustering
لقطة
كلب
هدول الهم نفس الخصائص
C3
هدول الهم نفس الخصائص
C1
هدول الهم نفس الخصائص
C2
Environment
action
feedback
state
Model Agent

Why is Deep Learning Important?

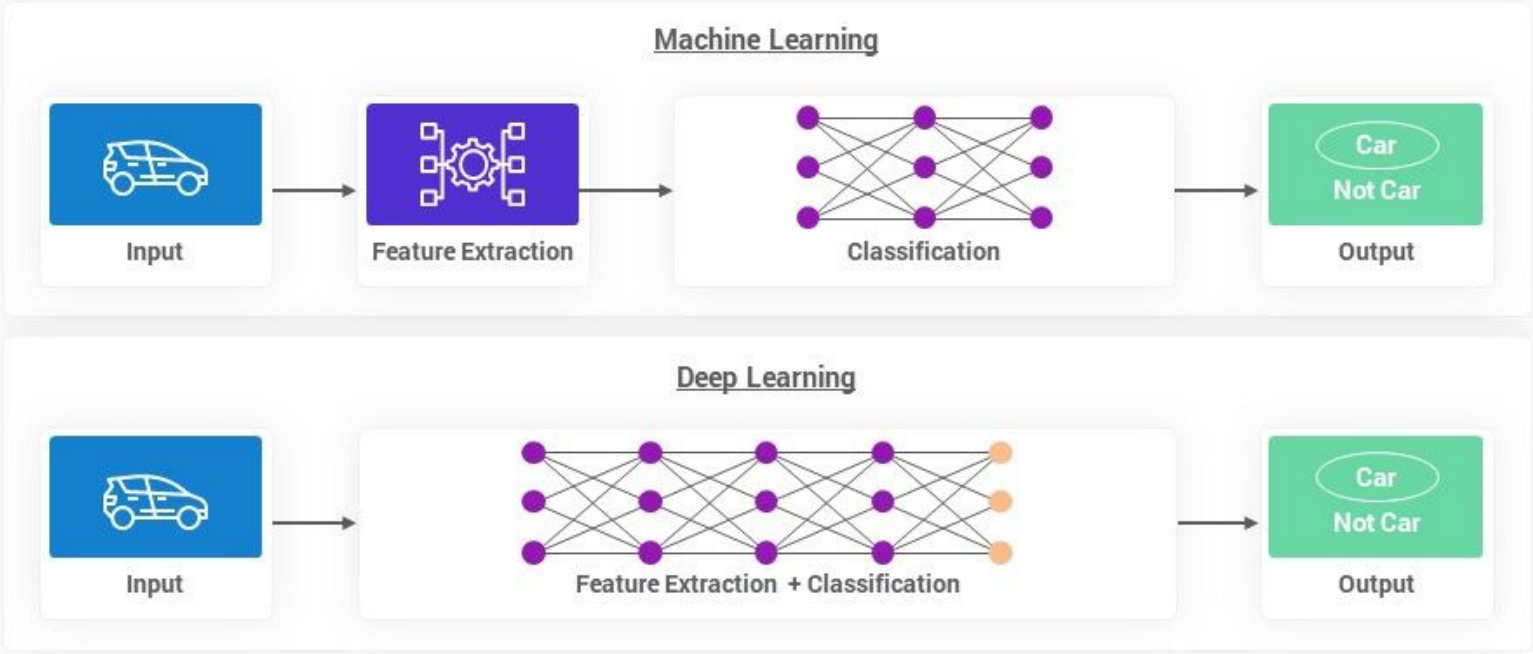
Why is Deep Learning Important?



slidesalad

Machine Learning vs. Deep Learning

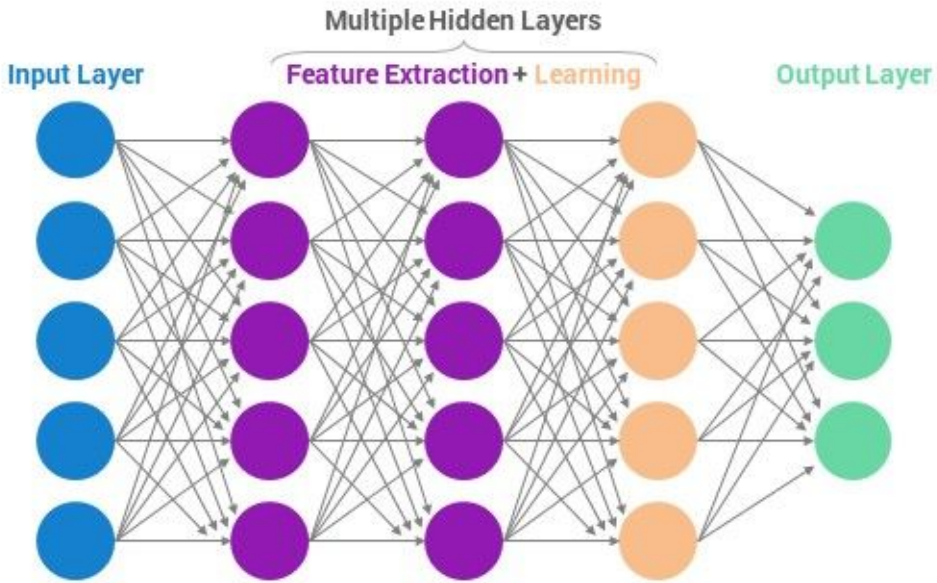
Machine Learning vs. Deep Learning



slidesalad

Deep Neural Network

Deep Neural Network



Input Layer

It contains those units (Artificial Neurons) which receive input from the outside world on which the network will learn, recognize about, or otherwise process.

Output Layer

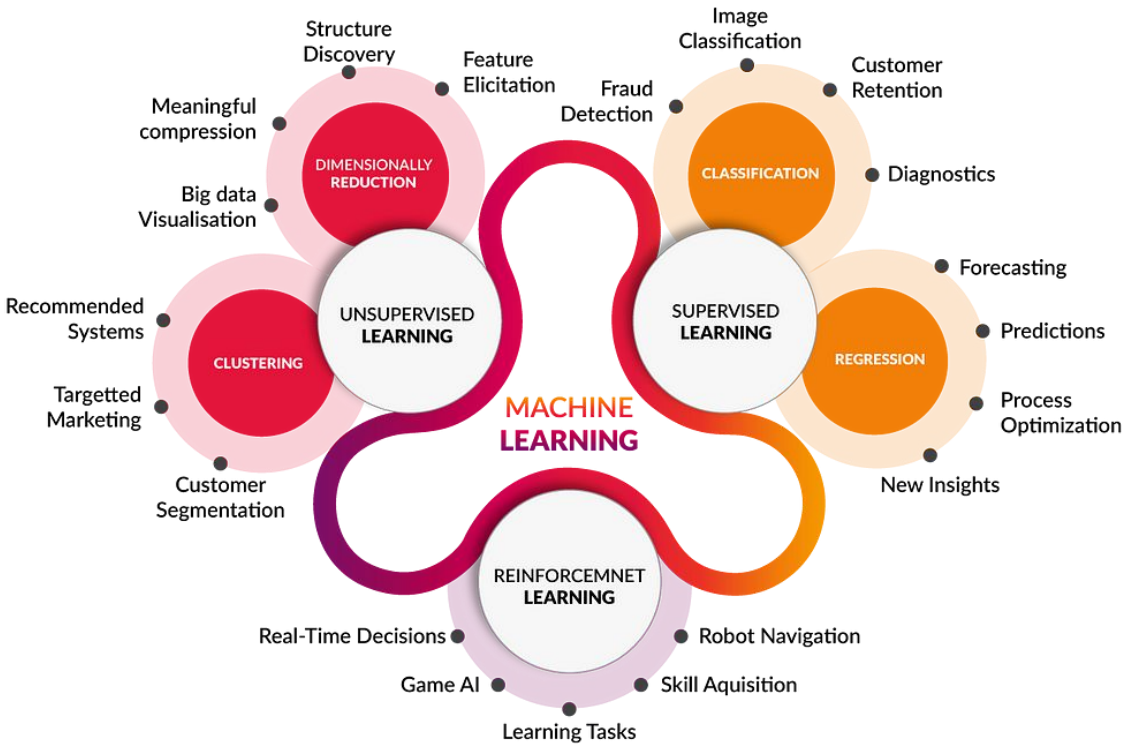
It contains units that respond to the information about how it learn any task.

Hidden Layer

These units are in between input and output layers. The hidden layer's job is to transform the input into something that the output unit can use somehow.

slidesalad

Types of Machine Learning



Comprehensive Framework for Selecting and Understanding Any Machine Learning Algorithm

This **comprehensive framework** is designed to guide ML practitioners—especially beginners—through a **structured, step-by-step approach** to:

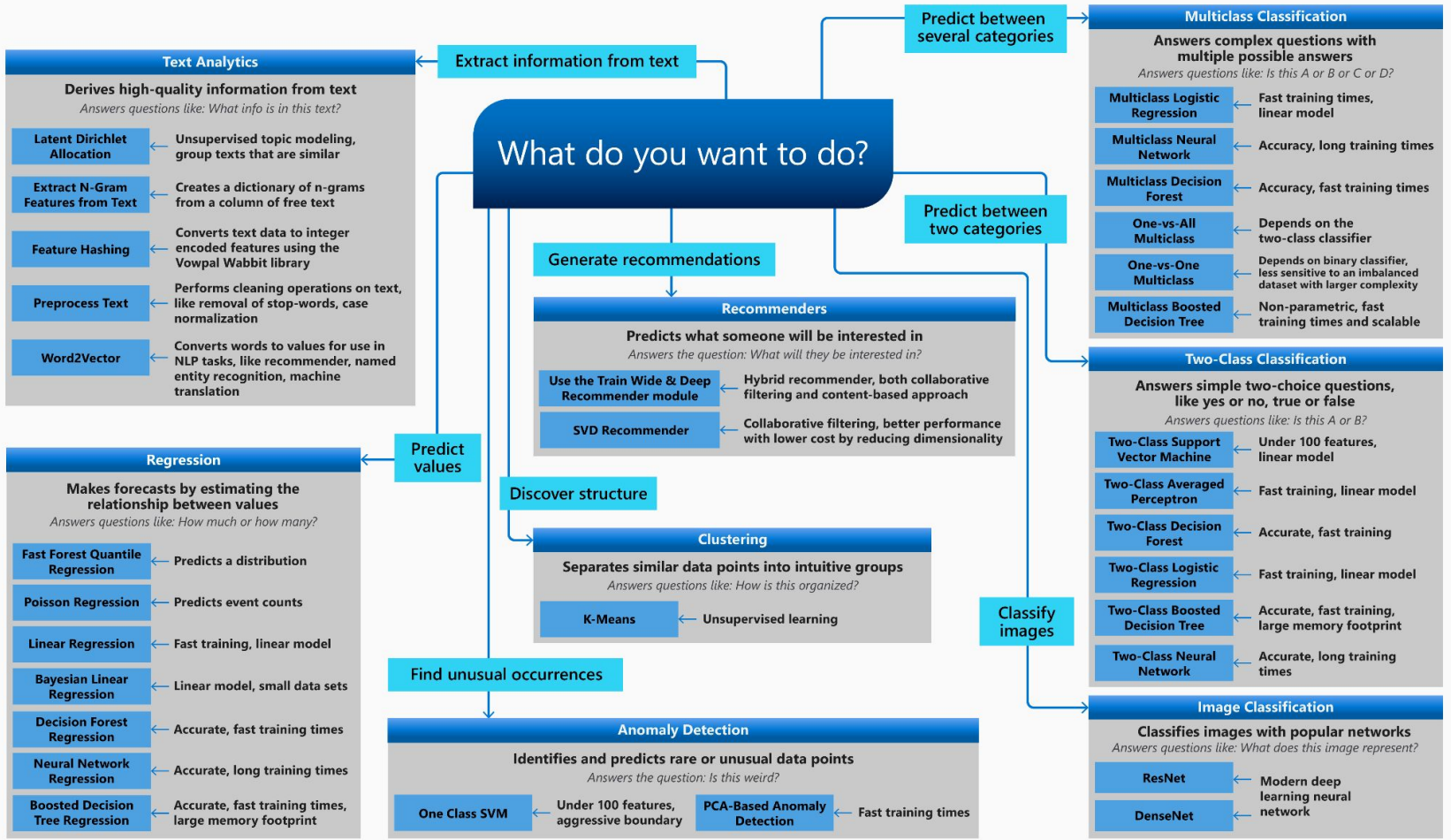
1. **Select the most suitable algorithm** for their specific problem.

2. **Understand the chosen algorithm in depth** to ensure optimal implementation, tuning, and evaluation.

Machine Learning Algorithm Cheat Sheet

This cheat sheet helps you choose the best machine learning algorithm for your predictive analytics solution.
Your decision is driven by both the nature of your data and the goal you want to achieve with your data.

Microsoft Azure



© 2021 Microsoft Corporation. All rights reserved. Share this poster: aka.ms/mlcheatsheet

Regression Analysis

Regression analysis is a powerful tool for uncovering the associations between variables observed in data, but cannot easily indicate causation. It is used in several contexts in business, finance, and economics. For instance, it is used to help investment managers value assets and understand the relationships between factors such as commodity prices and the stocks of businesses dealing in those commodities.

- A regression is a statistical technique that relates a dependent variable to one or more independent (explanatory) variables.
- A regression model is able to show whether changes observed in the dependent variable are associated with changes in one or more of the explanatory variables.
- It does this by essentially fitting a best-fit line and seeing how the data is dispersed around this line.
- Regression helps economists and financial analysts in things ranging from asset valuation to making predictions.
- In order for regression results to be properly interpreted, several assumptions about the data and the model itself must hold

Regression

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables)

$$Y_i = f(X_i, \beta) + e_i$$

Y_i = dependent variable
f = function
X_i = independent variable
β = unknown parameters
e_i = error terms

Coefficients are determined by the regression.

House Price= a + b×**Size**

Dependent Variable **Independent Variables**

HP = 30 + 10*(user size)

Coefficients are determined by the regression.

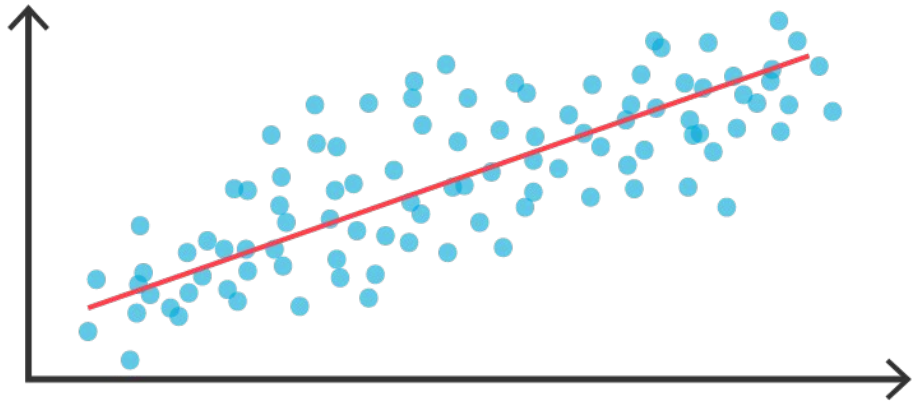
House Price= a + b×**Size** + c ×**No. Of room**

Dependent Variable **Independen t Variables**

Regression types

Simple linear regression:

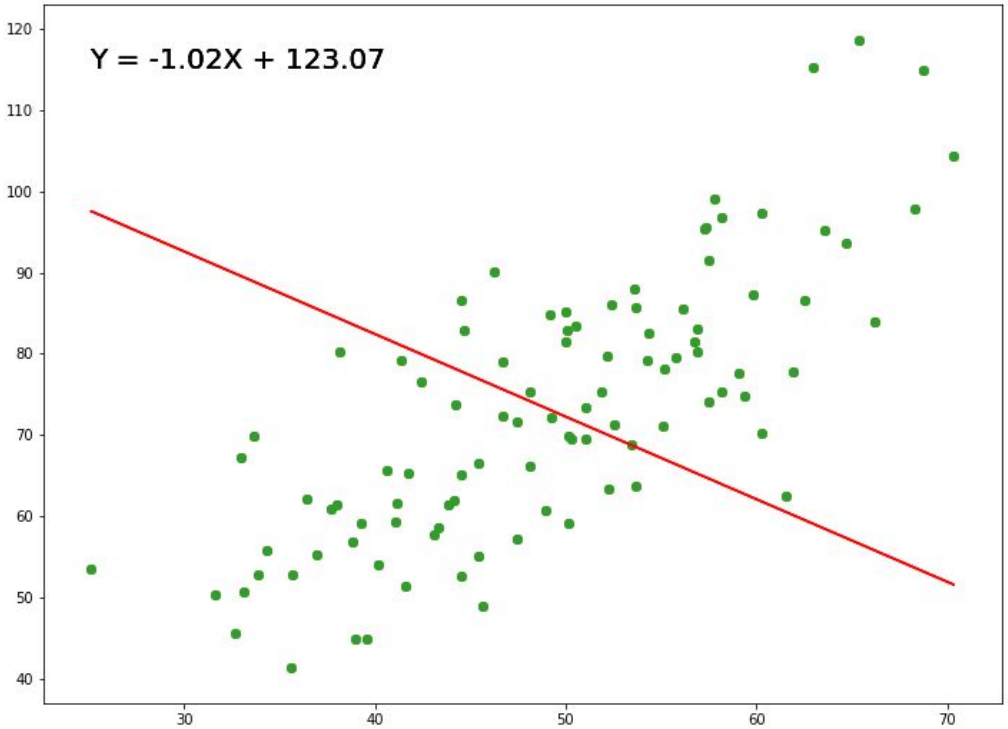
$$Y = a + bX + u$$



Multiple linear regression:

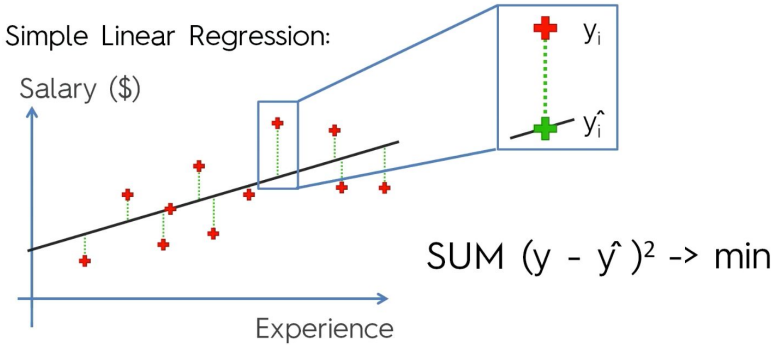
$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + ... + b_tX_t + u$$

Y = 1.3*X + 9
Y = 1.3*(30) + 9



Simple Linear Regression Idea (Ordinary Least Squares OLS)

- The idea of Simple Linear Regression is finding parameters Θ_0 and Θ_1 for which the error term is minimized(optimization problem).
- The model wi*ll minimize the sum of squared residuals* which is (cost/loss) function: we don't want our positive residuals to be cancelled by the negative ones, since they are equally penalizing for our model.
- This procedure is called Ordinary Least Squared error - OLS.



1. The Main Idea:

- We want to find the best-fit line $y = mx + b$ that minimizes the distance to all the data points.
- The goal is to determine the optimal values of m (slope) and b (intercept).

2. Defining the Error:

For each data point:

- The error is the difference between the actual value y_i and the predicted value $\hat{y}_i$ from the line.
- The error for one point: $\text{Error}_i = y_i - \hat{y}_i$.

3. Objective Function (Loss Function):

- To minimize the error, we square it to handle both positive and negative values and sum it up: $\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- This is called the "Sum of Squared Errors"

4. Expanding the Equation:

Substitute $\hat{y}_i = mx_i + b$ into the equation: $\text{SSE} = \sum_{i=1}^{n} (y_i - (mx_i + b))^2$

6. Solving the System of Equations:

Solve these two equations simultaneously to find the optimal m and b.

Y = a0*x + a1

Slope m:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Intercept b:

$$b = \bar{y} - m\bar{x}$$

Where:

- $\bar{x}$: Mean of x-values.
- $\bar{y}$: Mean of y-valu

7. Final Equation:

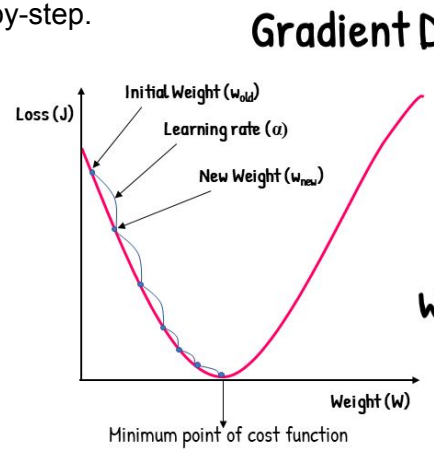
The final regression line is: $\hat{y} = mx + b$ Where m and b are computed to minimize the **Sum of Squared Errors (SSE)**.

$$y = bx + a$$

- x > القوة إلي بترمي فيها الكرة
- b > الميل يعني كل ما تزيد x شو بصير ب y
- a > من وين الخط هيبدأ (يعني من وين هيرمي الكرة اول مرة)
- y > القيمة الي بدك تتنبأ فيها يعني (المسافة عن السلة)

Gradient Descent in Linear Regression

Goal: Minimize the **cost/loss function** by adjusting model parameters step-by-step.



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

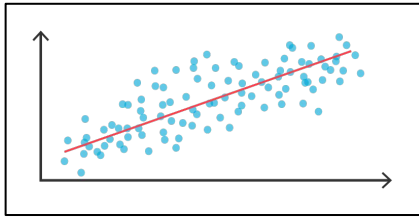
How it works:

1. **Initialize weights** with random values.
2. **Predict** outputs using the current weights.
3. **Calculate error** using a cost/loss function (e.g., MSE).
4. **Find gradient** → the slope of the error with respect to each weight.
5. **Update weights**: move in the opposite direction of the gradient **using the learning rate (α)**:
    - **Too big** → may overshoot the optimal point.
    - **Too small** → very slow convergence.
    - **Example:** walking in a valley at night → big steps might skip the bottom, small steps reach it slowly.
6. **Repeat** until the error stops decreasing or a set number of iterations is reached.

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد الكيلوات المباعة)



Y = X* slope + intercept

Y = X* α + β

(α,β = weight = Coefficients

Parameters إلي الخورازمية هتجيبهم

)

1. OLS – Ordinary Least Squares (بتجيب الأوزان مباشرة بمعادلات جبرية)
2. Gradient Descent (بتجيب الأوزان بالتدريج)

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد

Y = X* slope + intercept

Y = X* α + β

sales = temp* α + β

(α,β = weight = Coefficients

Parameters إلي الخورازمية هتجيبهم

)

OLS (لما يكون عدد ال rows قليل) (بطيئة) (دقتها عالية) (حساسة للقيم الشاذة)

Error = Y - Ypred

sum(y - ypred)^2 >>>>> cost function (min Error)

sum(y - (X* α + β))^2

Derivative w.r.t α to get α value

Derivative w.r.t β to get β value

- صار عندي معادلتين لكلا من β,α بتحسبهم مباشرة من كل قيم x ,y الحقيقين الي موجودين باعمدة الداتا عندي (استخدمنا كل قيم x,y يعني كل rows in csv تبعي)
- فبالتالي هيك عملنا minimize for cost function يعني قللنا الفرق بين القيم المتوقعة والقيم الحقيقية

1. OLS – Ordinary Least Squares (بتجيب الأوزان مباشرة بمعادلات جبرية)

بدنا نعرف سعر الكيلو الي يخلي الزبون يشتري مني بناء على درجة الحرارة

Sales (Y target) > temp (X feature) (simple linear Regression)

بيحاول يرسم خط مستقيم بيقطع اكبر عدد من الداتا (line fit data) (يربط بين درجة الحرارة والعدد الكيلوات المباعة)

5. Iterative Updates:

At each iteration:

1. Compute $\hat{y}_i = mx_i + b$.
2. Calculate the gradients:
   - $\frac{\partial J}{\partial m}$
   - $\frac{\partial J}{\partial b}$
3. Update m and b:

$$m = m - \alpha \frac{\partial J}{\partial m}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

Gradient Descent (لما يكون عدد ال rows كبير) (اسرع) (دقتها اقل مقارنة ب ols
لانها ما بتشوف كل ال rows) (حساسة للقيم الشاذة)

Error = Y - Ypred

sum(y - ypred)^2 >>>>> cost function (min Error)

sum(y - (X* m + b))^2

1) بفترض قيم عشوائية ل b,a

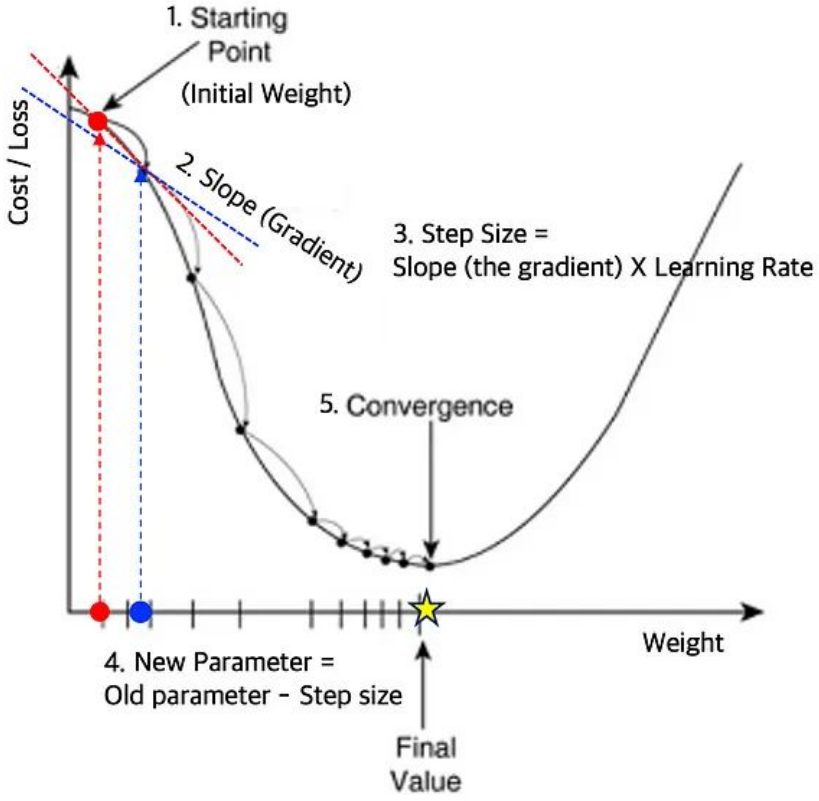
Derivative w.r.t m to get m value

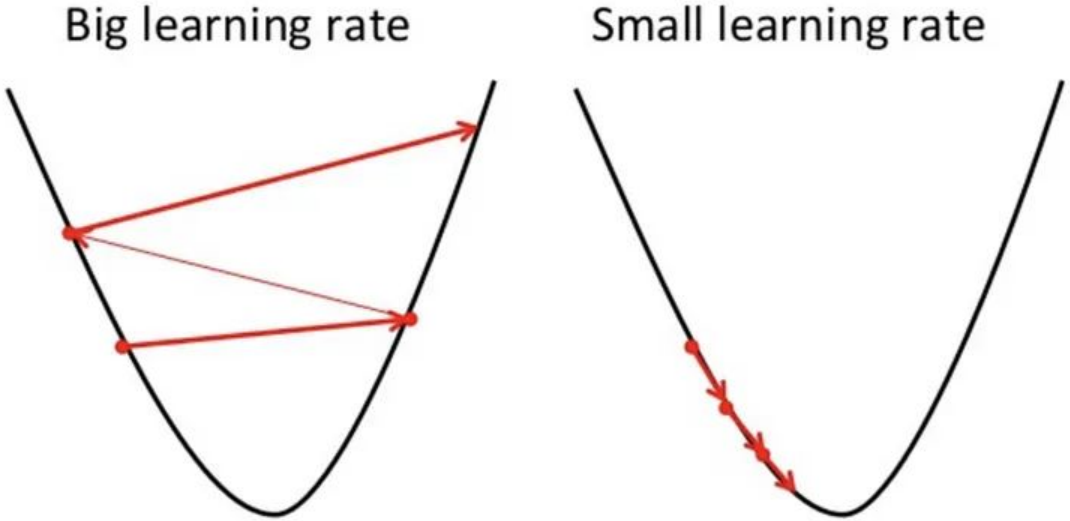
Derivative w.r.t b to get m value

2) α is a learning rate (step size)

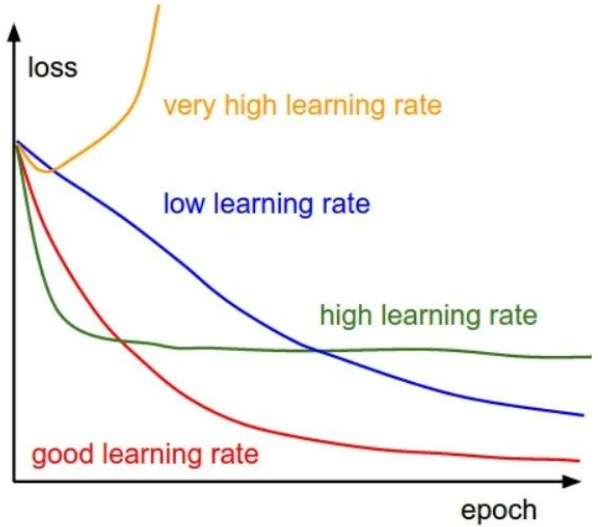
- صار عندي معادلتين لكلا من b,m بتحسبهم مباشرة من كل قيم x ,y الحقيقين الي موجودين باعمدة الداتا عندي (استخدمنا كل قيم x,y يعني كل rows in csv تبعي)
- فبالتالي هيك عملنا minimize for cost function يعني قللنا الفرق بين القيم المتوقعة والقيم الحقيقية

Gradient Descent



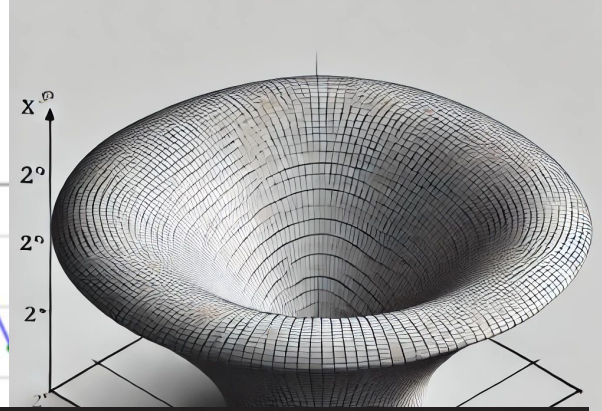


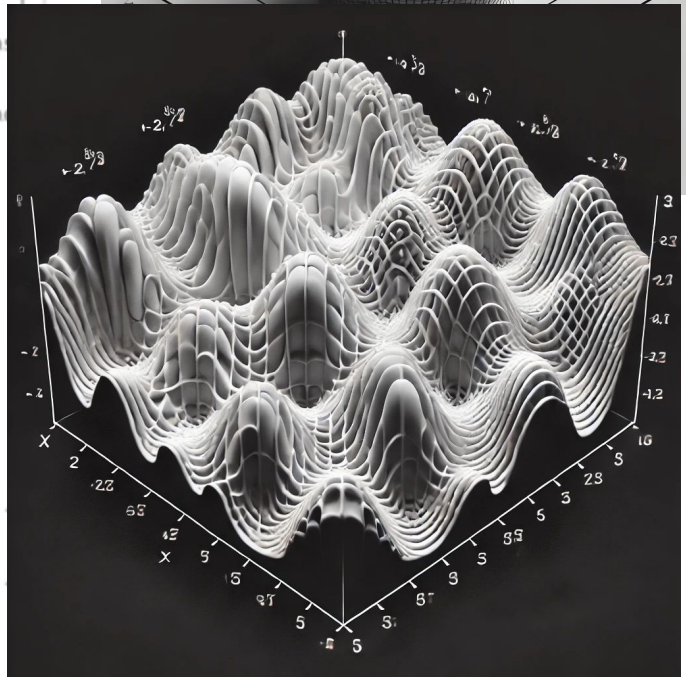
Gradient Descent



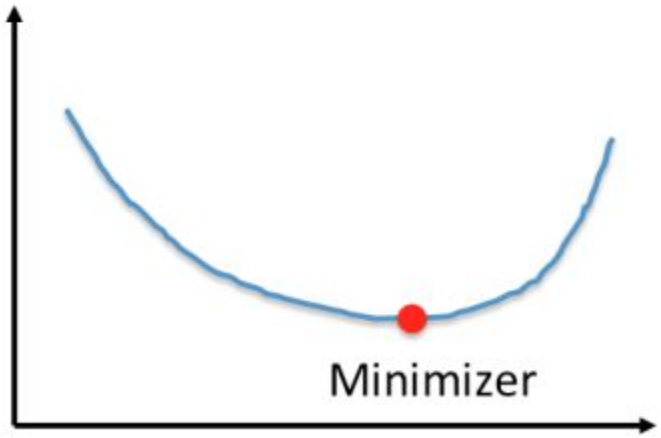
Linear Regression Idea (Gradient Descent)

- Another optimization algorithm used in linear Regression is Gradient Descent.
- It is important, easy to implement, and used in machine learning and deep learning to minimize a cost/loss function .
- Gradient descent algorithms works when error function is :
  - Differentiable: function has a derivative for each point.
  - Convex:
    - For a univariate function, the line segment connecting two function's points lays on or above its curve (it does not cross it).
    - Calculate the second derivative if its value is always bigger than 0 then function is convex.

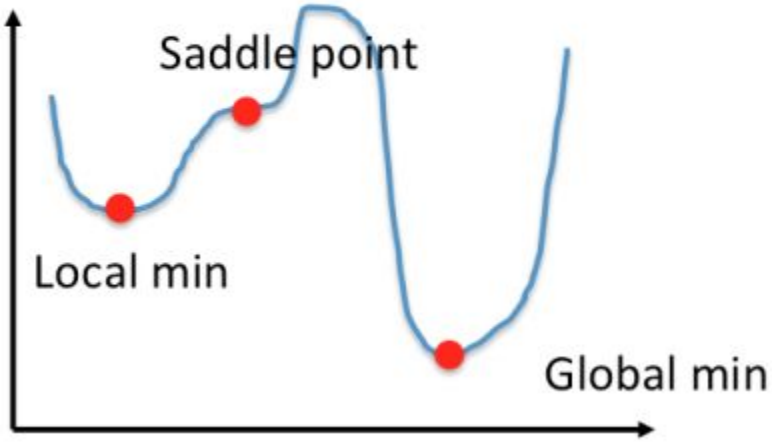




Convex



Non-Convex



Simple Linear Regression (Minimize cost using Gradient Descent)

1. The Main Idea:

- We aim to fit a line $y = mx + b$ that minimizes the error between predicted values $\hat{y}_i$ and actual values y_i.
- Instead of directly solving for m and b, we iteratively find them by minimizing the **cost function** using Gradient Descent.

---

2. Define the Cost Function:

The cost function for linear regression is the Mean Squared Error (MSE):

$$J(m, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- y_i: Actual values.
- $\hat{y}_i = mx_i + b$: Predicted values.
- n: Total number of data points.

3. The Gradient Descent Algorithm:

Gradient Descent updates m and b iteratively to minimize $J(m, b)$.

Update Rules:

$$m = m - \alpha \frac{\partial J}{\partial m}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

Where:

- α: Learning rate (controls step size).
- $\frac{\partial J}{\partial m}$: Partial derivative of J with respect to m.
- $\frac{\partial J}{\partial b}$: Partial derivative of J with respect to b.

4. Compute the Partial Derivatives:

Using the cost function $J(m, b)$, the gradients are:

Gradient w.r.t. m:

$$\frac{\partial J}{\partial m} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - \hat{y}_i)$$

Gradient w.r.t. b:

$$\frac{\partial J}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)$$

5. Iterative Updates:

At each iteration:

1. Compute $\hat{y}_i = mx_i + b$.
2. Calculate the gradients:
   - $\frac{\partial J}{\partial m}$
   - $\frac{\partial J}{\partial b}$
3. Update m and b:

$$m = m - \alpha\frac{\partial J}{\partial m}$$

$$b = b - \alpha\frac{\partial J}{\partial b}$$

6. Convergence:

- Repeat the updates until $J(m, b)$ converges to a minimum (or the changes become very small).
- This gives the optimal values of m and b that minimize the cost.

---

7. Final Result:

After convergence, the best-fit line is:

$$\hat{y} = mx + b$$

Where m and b are the values that minimize the cost function.

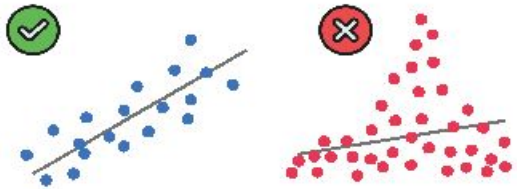
Practical: Linear Regression

- To apply linear regression on a data set we use LinearRegression API offered by sklearn.linear_model .
- Score help us evaluate how our model did (discussed later).
- Sklearn is:
  - Simple and efficient tools for predictive data analysis.
  - Accessible to everybody, and reusable in various contexts.
  - Built on NumPy, SciPy, and matplotlib.
  - Open source, commercially usable.

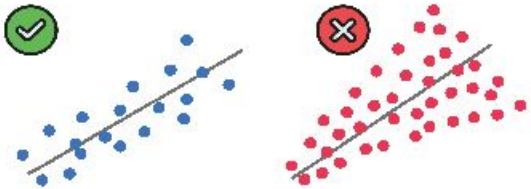
```python
from sklearn.linear_model import LinearRegression

# make model object
model = LinearRegression()

# train
model.fit(x_train, y_train)

# test
model.predict(x_test)

# calculate R2 score on training data
model.score(x_train, y_train)

# calculate R2 score on testing data
model.score(x_test, y_test)

# get model parameters
model.coef_
model.intercept_
```

Assumptions of Linear Regression

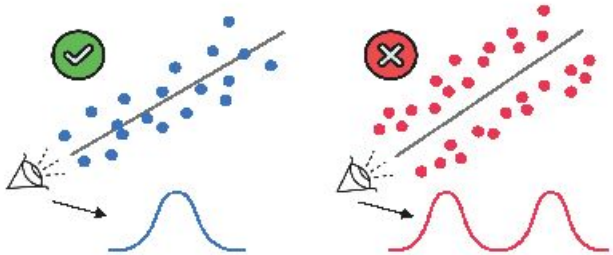
1. Linearity
(Linear relationship between Y and each X)



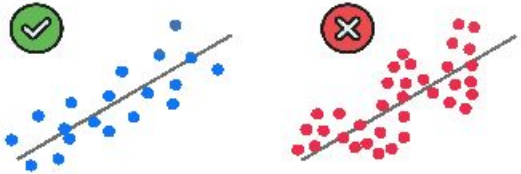
2. Homoscedasticity
(Equal variance)



3. Multivariate Normality
(Normality of error distribution)



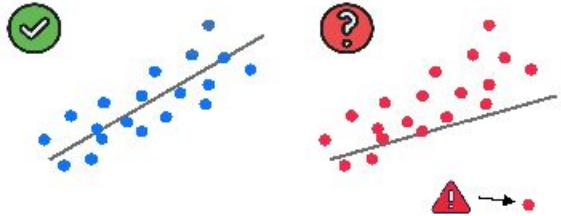
4. Independence
(of observations. Includes "no autocorrelation")



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

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

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

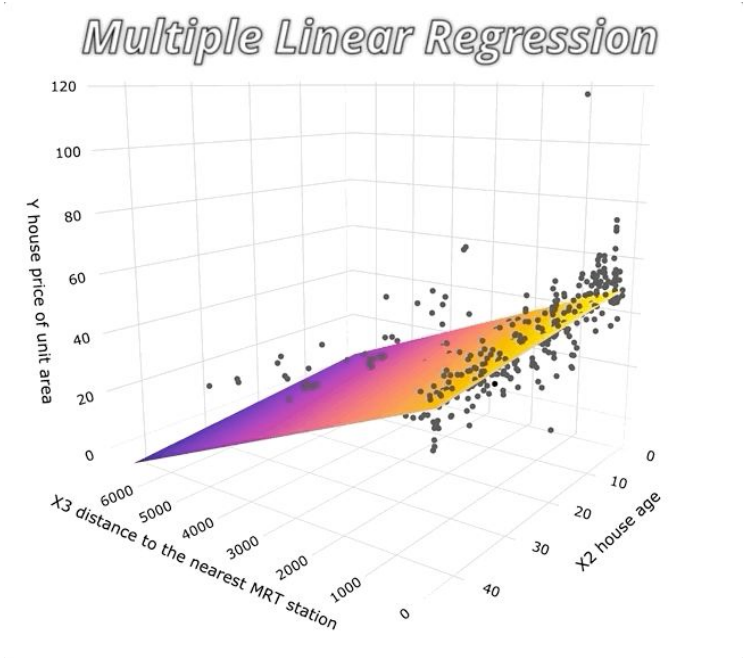


© SuperDataScience

Paint X lite

Multiple Linear Regression:

- It’s a form of linear regression that is used when there are two or more predictors.
- To predict the outcome from multiple input variables.
- The same as simple linear regression optimization is done using OLS or GD.
- E.g.:
  - In simple linear regression model predicts house price from number of rooms.
  - In multiple linear regression model predict price based on size, locations, number of rooms (higher dimension) etc.

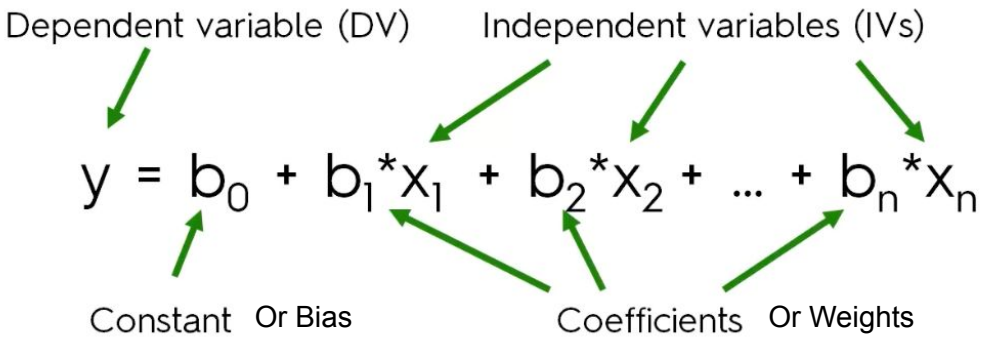


Multiple Linear Regression (Equation)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression



Cost Function: Sum of Squared Errors

- **Objective**: Minimize the cost function to find the best fit line.
- **Cost Function Equation**:
  $J(\beta) = \sum_{i=1}^{m} (Y_i - \hat{Y}_i)^2$
  - m: Number of samples
  - Y_i: Actual value
  - $\hat{Y}_i$: Predicted value

Estimating Parameters Using Least Squares

- **Normal Equation:**

$$\beta = (X^T X)^{-1} X^T Y$$

  - X: Input matrix (including bias term) >>> feature col (area , num_bedroom)
  - Y: Output vector >>> price col
  - β: Coefficient vector

Price = area*a1 + num_bedroom*a2 + b

>>> [a1 , a2 , b]

Example: Predicting House Prices

Area (sqm)		
100		
150		
200		
250		

Solving for β

$$\beta = (X^T X)^{-1} X^T Y$$

Using computation:

$$\beta = \begin{pmatrix} 50000 \\ 1000 \\ 20000 \end{pmatrix}$$

Data Matrices

$$X = \begin{pmatrix} 1 & 100 & 2 \\ 1 & 150 & 3 \\ 1 & 200 & 4 \\ 1 & 250 & 4 \end{pmatrix}$$

$$\begin{pmatrix} 350000 \end{pmatrix} \qquad X^T Y = \begin{pmatrix} 146000000 \\ 2525000 \end{pmatrix}$$

[
b(b
a1 (
a2 (
]

Final Model

Multiple Linear Regression Equation

$$\text{Price} = 50000 + 1000 \times \text{Area} + 20000 \times \text{Bedrooms}$$

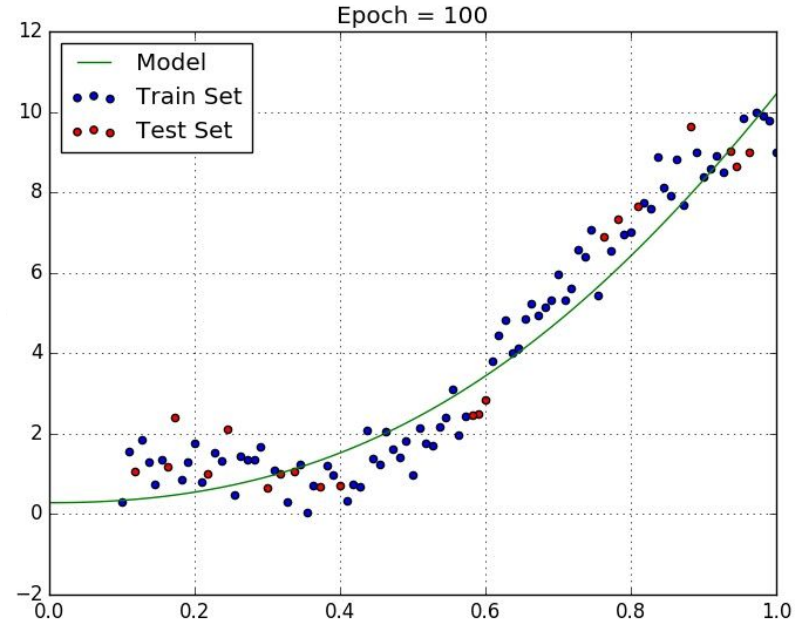
- **Intercept** (β_0): Base price = $50,000
- **Area** (β_1): Each additional square meter adds $1,000 to the price.
- **Bedrooms** (β_2): Each additional bedroom adds $20,000 to the price.

Polynomial Regression:

- Polynomial regression is a special case of linear regression.
- We fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

$$Y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots + \theta_n x^n$$

- We call it linear because we don't look at it from the point of view of the x-variable. We look at coefficients.
- The larger the degree of polynomial (n) the more complex the model.



Note: θ0 is the bias, θ1, θ2, …, θn are the weights in the equation of the polynomial regression, and n is the degree of the polynomial.

Living Area (sq ft)
1200
1500
1800
2100

Living Area (sq ft)	Living Area^2 (sq ft^2)
1200	1,440,000
1500	2,250,000
1800	3,240,000
2100	4,410,000

Original Equation (Simple Linear Regression):

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Equation After Polynomial Expansion (Degree 2):

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$$

We found the following coefficients:

$\beta_0 = 100{,}000$

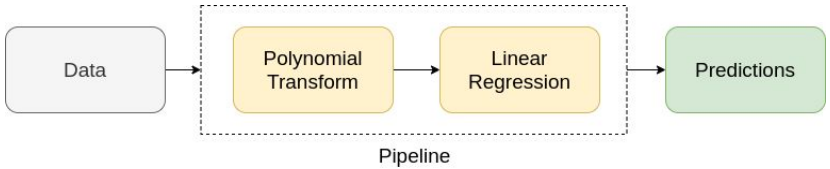
$\beta_1 = 150$

$\beta_2 = 0.02$

$$y = 100{,}000 + 150x + 0.02x^2$$

Polynomial Regression code:

- The implementation of polynomial regression is a two-step process.
  - First, we transform our data into a polynomial using the PolynomialFeatures API from sklearn.
  - Then use linear regression to fit the parameters.

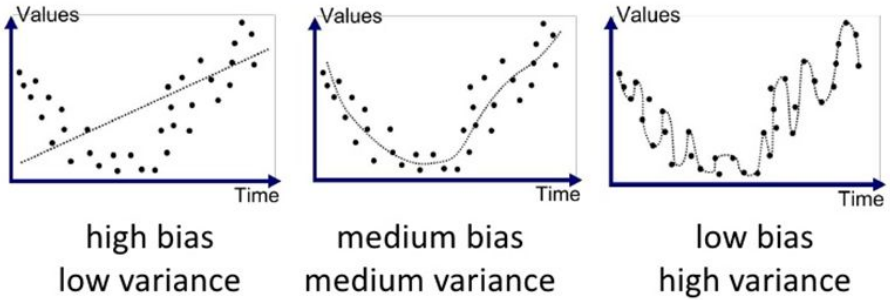



```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# make polynomial features object with degree of 2
poly = PolynomialFeatures(degree=2)
# Creating the new features data
poly_features = poly.fit_transform(x)
# Creating the polynomial regression model
poly_reg_model = LinearRegression()
# fit our data
poly_reg_model.fit(poly_features, y)
# predict labels
poly_reg_model.predict(poly_features)
# Calculate R2
poly_reg_model.score(poly_features, y)
# get model parameters
poly_reg_model.coef_
poly_reg_model.intercept_
```

Underfitting & Overfitting

[4,8,2] [5,5
4.8

- The more model complexity you add the accuracy improves but this can lead to overfitting that means the model memorizes the data too much and not applying generalization.
- So, we should know about the trade-off between Bias and Variance.
- A model with a high bias error underfits data and makes very simplistic assumptions on it.
- A model with a high variance error overfits the data and learns too much from it.
- A good model is where both Bias and Variance errors are balanced



Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low	High	• Need a simpler model • Need more data (more data samples)
Over-fitting		

حافظ مش فاهم

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low Over-fitting	High	• Need a simpler model • Need more data (more data samples) حافظ مش فاهم

High
Under fitting
-
	High	• Need more data
		  • Difficult to learn f(x, z) with only x. Get also z
		  • Get more data samples
		  • Additional features (e.g. x^2, $\frac{1}{x}$)
		• Need a more complex model
		لا حافظ ولا فاهم

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low	High	• Need a simpler model • Need more data (more data samples) حافظ مش فاهم

High	High	• Need more data
		  • Difficult to learn f(x, z) with only x. Get also z
		  • Get more data samples
		  • Additional features (e.g. x^2, x^{-1}) لا حافظ ولا فاهم
		• Need a more complex model

High	Low	Unusual: it could mean that the test data is too similar to the train data. Get more test data.

Over-fitting

Under-fitting

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low	High	• • low bias, high variance حافظ مش فاهم
High	High	• Need more data Difficult to learn f(x, z) with only x. Get also z high bias, potentially high variance too لا حافظ مش فاهم • Need a more complex model x^2
High	Low	Unusual: it could mean that the test data is too similar to the train data. Get more test data.
Low	Low	You' Trade-off between Bias and Variance

Over-fitting

Under-fitting

Underfitting & Overfitting

Techniques to reduce underfitting

- Increase model complexity.
- Increase number of features.
- Performing feature engineering.
- Remove noise from the data.

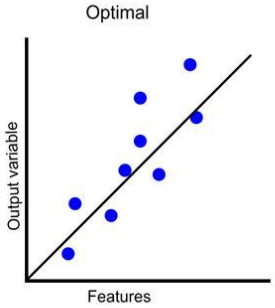
Underfitting & Overfitting

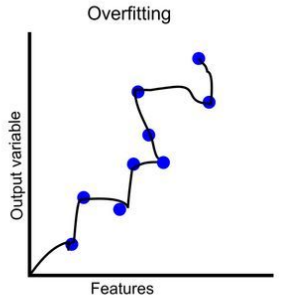
Techniques to reduce overfitting

- Increase training data.
- Remove correlated features.
- Use simpler model.
- **Regularization (add penalty bias).**

Regularization

- **Fine-tuning Machine Learning Models for Optimal Performance**





- **The Challenge:**
  - Overfitting: When a model learns the noise and random fluctuations in the training data, rather than the underlying pattern.
- **The Problem:**
  - Excellent performance on training data, but poor performance on new data.
  - Complex model that is difficult to interpret.
  - Inability to generalize well to real-world data.

What is Regularization?

- Techniques used to prevent or reduce overfitting.
- Works by adding a "penalty" to the model if it becomes too complex.
- Helps to simplify the model and improve its ability to generalize.

Types of Regularization:

1. **L1 Regularization (Lasso):**

   - **Mechanism:** Adds the absolute value of the magnitude of coefficients as a penalty term to the loss function.
   - **Formula:** $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum_{i=1}^{n} |w_i|$
   - **Effect:** Encourages sparsity in the model by driving some coefficients to zero, effectively performing feature selection.

Price = numRoom*a1 + houseSize * a2 + unv*a3
Price = numRoom*a1 + houseSize * a2 + unv*0.5
Price = numRoom*a1 + houseSize * a2 + unv*0

1. Adding L1 Regularization to the Loss Function

Let's start with the original loss function, such as Mean Squared Error (MSE):

$$\text{Loss} = \text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Where:

- y_i: Actual values.
- $\hat{y}_i$: Predicted values from the model.
- N: Number of samples.

When we apply **L1 Regularization (Lasso)**, the loss function becomes:

$$\text{Loss}_{\text{L1}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}|w_j|$$

L1

$$\text{Loss} = \text{Loss}_{\text{original}} + \lambda\sum_{i=1}^{n}|w_i|$$

Where:

- λ: The regularization parameter controlling the strength of L1 Regularization.
- $|w_j|$: The absolute value of the weight w_j.
- p: Number of weights (features).

2. Impact on Gradient Descent

To update the weights during Gradient Descent, we calculate the partial derivatives of the loss function with respect to each weight w_j.

Derivative of the MSE term:

$$\frac{\partial \text{Loss}}{\partial w_j} = -\frac{2}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i)$$

Derivative of the L1 Regularization term:

L1 Regularization involves the absolute value $|w_j|$. Its derivative depends on the sign of w_j:

$$\frac{\partial}{\partial w_j}|w_j| = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ 0 & \text{if } w_j = 0 \end{cases}$$

Combined derivative:

The total derivative of the L1-regularized loss function becomes:

$$\frac{\partial \text{Loss}_{\text{L1}}}{\partial w_j} = -\frac{2}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i) + \lambda \cdot \text{sign}(w_j)$$

Where $\text{sign}(w_j)$ is the sign function:

$$\text{sign}(w_j) = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ 0 & \text{if } w_j = 0 \end{cases}$$

Explanation of sign function:

- **If the weight is positive** ($w_j > 0$): The sign is $+1$, meaning the penalty decreases the weight.
- **If the weight is negative** ($w_j < 0$): The sign is -1, meaning the penalty increases the weight towards zero.
- **If the weight is zero** ($w_j = 0$): The sign is 0, meaning no penalty is applied to this weight.

Simple Example:

- If $w_j = 5$:
  $\text{sign}(w_j) = +1$, meaning the penalty will reduce the weight.
- If $w_j = -3$:
  $\text{sign}(w_j) = -1$, meaning the penalty will increase the weight (move it closer to zero).
- If $w_j = 0$:
  $\text{sign}(w_j) = 0$, meaning the weight remains unchanged.

Types of Regularization:

2. **L2 Regularization (Ridge):**

- **Mechanism**: Adds the squared value of the magnitude of coefficients as a penalty term to the loss function.
- **Formula**: $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum_{i=1}^{n} w_i^2$
- **Effect**: Penalizes large coefficients more heavily than L1 regularization, leading to a model where all features are considered but with smaller weights.

Price = numRoom*a1 + houseSize * a2 + unv*a3
Price = numRoom*a1 + houseSize * a2 + unv*0.5
Price = numRoom*a1 + houseSize * a2 + unv*0.05

The regularization parameter λ

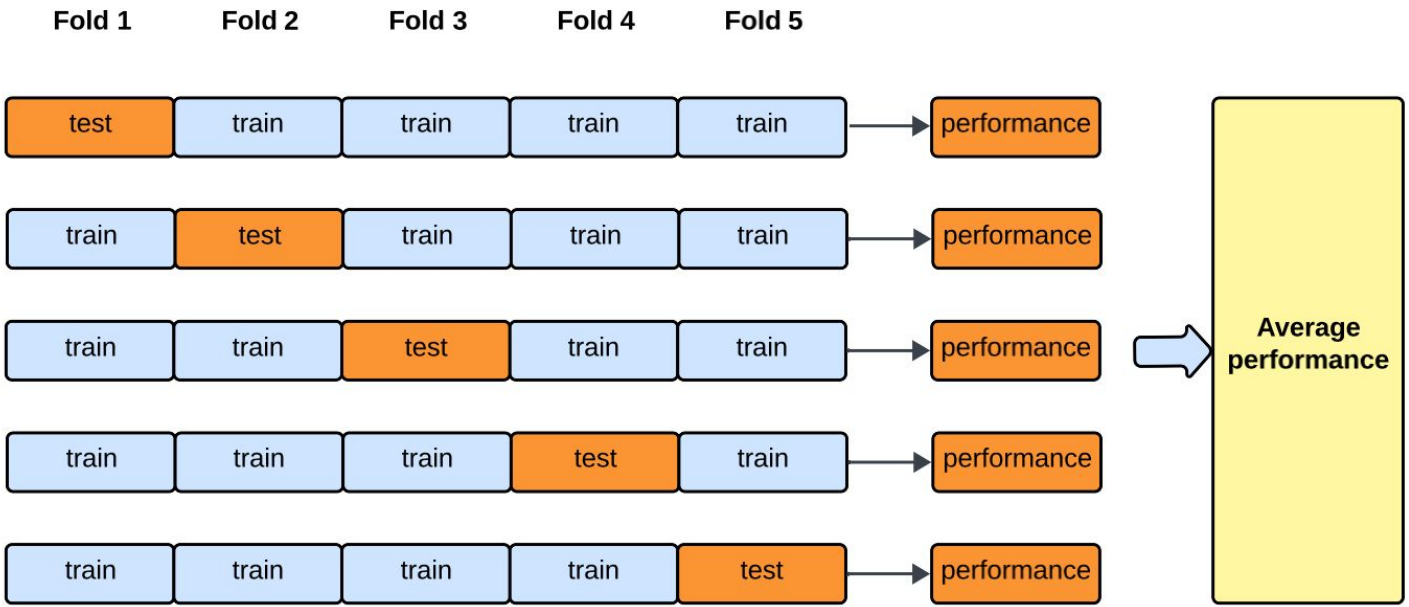
- **λ parameter:** controls the strength of the penalty applied to the coefficients.
- **A larger λ** increases the penalty, leading to more regularization (smaller coefficients),
  - leading to simpler models that **may underfit the data**
- **A smaller λ** reduces the penalty, resulting in less regularization.
  - Allowing the model to fit the training data more closely, which may result in **overfitting**.
- Selecting an appropriate λ is critical for balancing model complexity and generalization.

Methods to Select λ

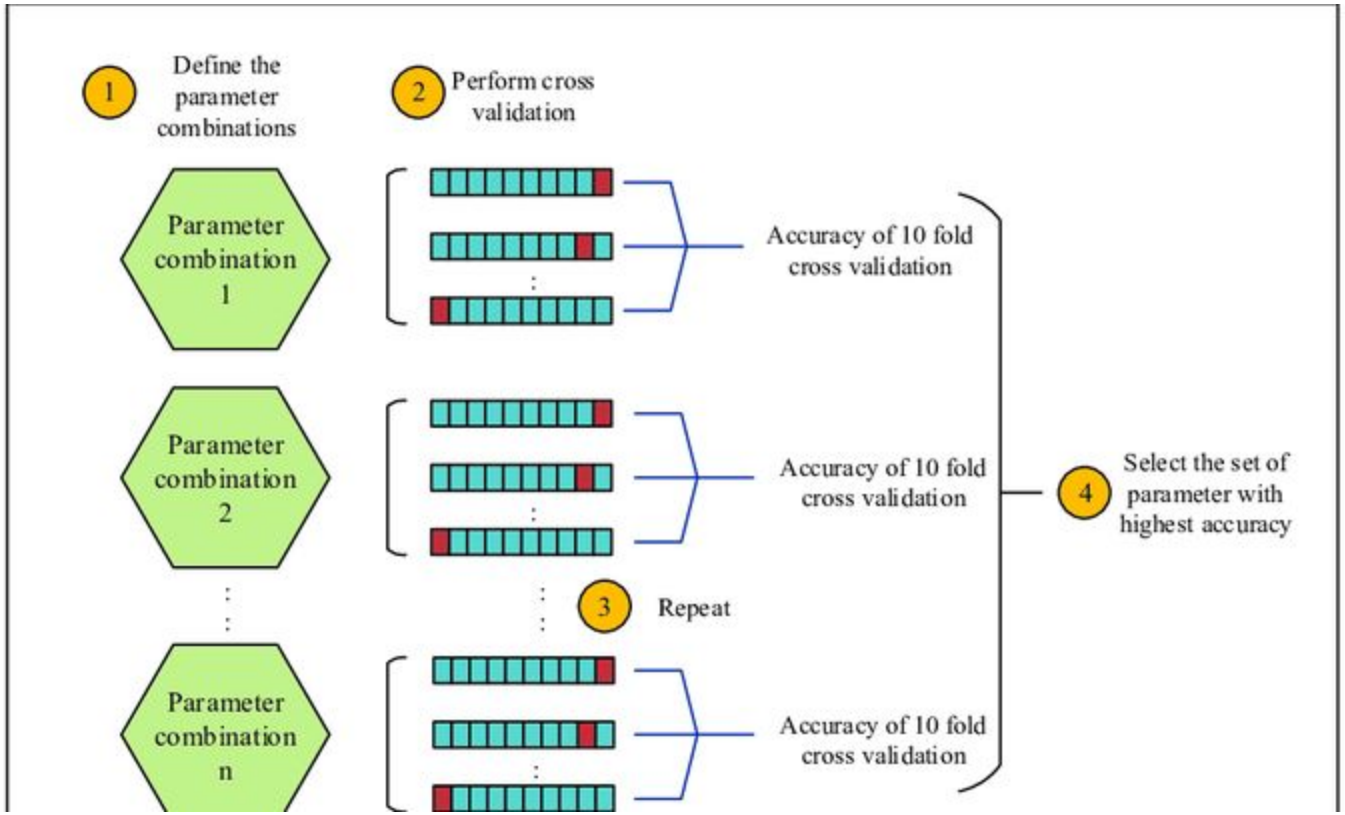
1. **Grid Search**: Test a range of λ values and select the one that performs best on a validation set.
2. **Cross-Validation**: Use k-fold cross-validation to evaluate the performance of different λ values and choose the one with the best cross-validated performance.
3. **Regularization Paths**: Compute the coefficients for a range of λ values and examine the stability and performance of the model coefficients.
4. **Bayesian Optimization**: Use more advanced methods like Bayesian optimization to find the optimal λ by intelligently exploring the hyperparameter space.

K-fold Cross Validation

cv=5 , lam = [0.1,0.01,0.02]



Grid Search CV



Grid Search

1. Definition

- A systematic method to tune hyperparameters.
- We define a **set of candidate values** (a “grid”) and test them one by one.

2. Process

1. Choose a hyperparameter to optimize (e.g., **λ / alpha** in Ridge).
2. Define a grid of possible values, e.g.:
   {0.01, 0.1, 1, 10, 100}
3. For each value, train the model using **k-fold cross-validation**.
4. Compute the average performance (e.g., MSE, R^2, Accuracy).
5. Select the value with the **best score**.

Example (with 3-Fold CV) .3

Average	Fold 3 R^2	Fold 2 R^2	Fold 1 R^2	λ (alpha)
0.803	0.79	0.80	0.82	0.01
0.846	0.83	0.86	0.85	0.1
✅ 0.880	0.89	0.87	0.88	1
0.840	0.82	0.84	0.86	10
0.700	0.72	0.68	0.70	100

Best choice = λ = 1 👉

Evaluating regression algorithms

Evaluating regression algorithms:

- Evaluating machine learning models help us to answer the following questions:
  - How well is my model doing? Is it a useful model?
  - Will training my model on more data improve its performance?
  - Do I need to include more features?
- So, do we see how our model did for every observation on our data set?
  - No, we evaluate our model using a single number (metric).
- The most common metric used for regression is R^2

Importance of Evaluating Regression Models

- Evaluating regression models is crucial to ensure their accuracy and reliability.
- Proper evaluation helps in understanding how well the model fits the data and predicts future outcomes.

على أي أساس بختار مقياس التقييم ؟ الداتا labeled or not

كل ما كانت قيمتها اصغر، أفضل دقة < MSE,MAE,RMES

(R^2) Coefficient of Determination

- Definition: Measures the proportion of the variance in the dependent variable that is predictable from the independent variables

Formula: $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

- R^2 ranges from 0 to 1
- **Example:** R^2= 0.85 means 85% of the variance is explained by the model

(R^2) Coefficient of Determination

- R^2 squared calculates how much regression line is better than a mean line.
- R-Squared is called coefficient of determination where it is commonly between 0 and 1 and as the value approaches 1 more, this gives intuition that the model representation
- is closer to the data.
- So, with help of R squared we have a baseline model to compare a model.

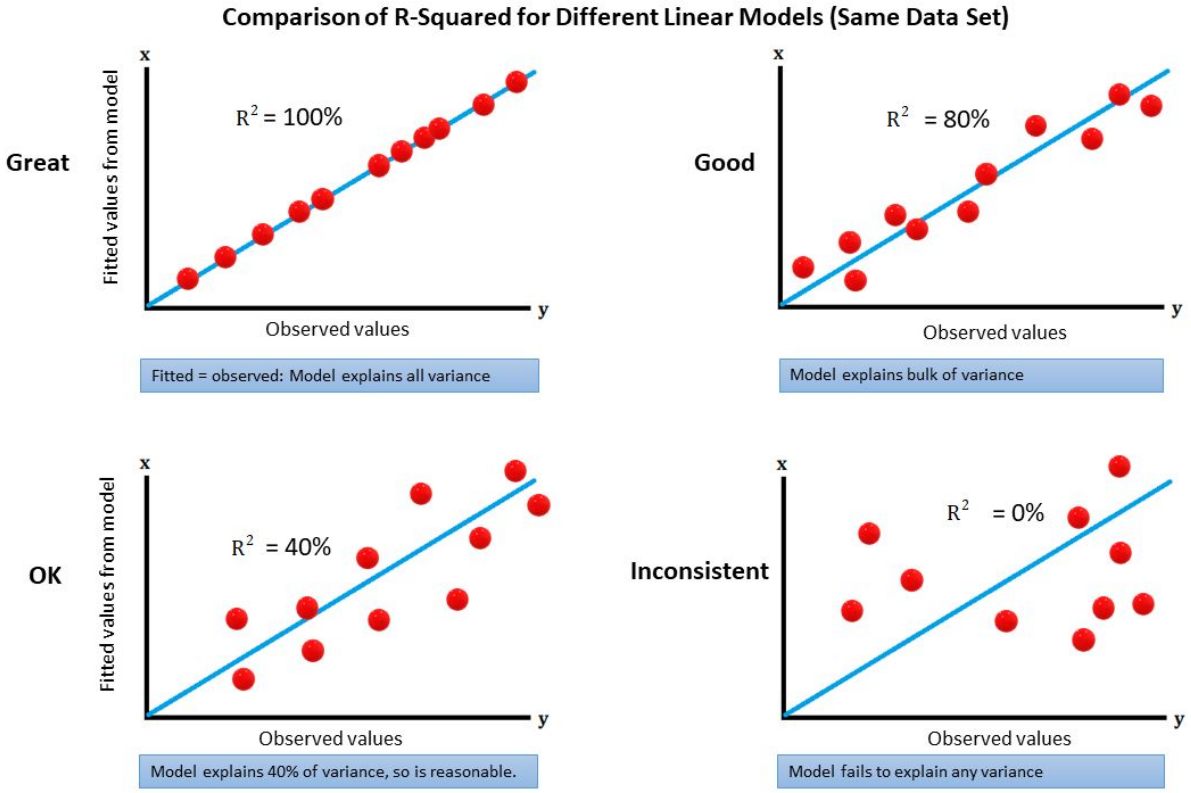
When to Use (R^2)

- Suitable for simple linear regression
- Initial model evaluation
- Quick assessment of model performance

Limitations of

- Can be misleading with multiple predictors
- Does not account for model complexity or overfitting

Evaluating Model Performance (R^2)



adjusted R^2

- Definition: Adjusts the R^2 value based on the number of predictors in the model

Formula: $\text{Adjusted } R^2 = 1 - \left(\frac{1-R^2}{n-p-1}\right)(n-1)$

- adjusted R^2 can decrease if unnecessary predictors are added

P: num of features
n: num of rows

When to Use Adjusted R^2

- Ideal for multiple regression models
- Comparing models with different numbers of predictors
- Feature selection and model refinement

Comparison of R^2 and Adjusted R^2

- R^2 does not penalize for extra predictors
- Adjusted R^2 penalizes for unnecessary predictors
- Adjusted R^2 is more reliable for multiple regression

Example

Data:

- y (house prices in $1000s): [200, 250, 300, 350, 400]
- X (size in square feet): [1500, 1800, 2000, 2300, 2500]

- Predicted values ($\hat{y}$): [210, 260, 290, 340, 390]
- Mean of actual values ($\bar{y}$): 300
- R^2 Calculation: $R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$
- Result: $R^2 = 0.95$ (indicates a very good fit)

- **New Model**: Multiple linear regression with additional predictors:
  - Size (in square feet)
  - Number of bedrooms
  - Age of the house (years)
- Additional data:
  - Bedrooms: [3, 3, 4, 4, 5]
  - Age: [10, 15, 10, 20, 10]
- Predicted values ($\hat{y}$): [205, 255, 305, 355, 405]
- Adjusted R^2 Calculation: Suppose R^2 for the new model is 0.96.
- Adjusted $R^2 = 1 - \left(\frac{1-0.96}{5-3-1}\right)(5-1) = 0.94$

Adjusted **R^2** indicates **whether the new predictors** genuinely improve the model or not.

Evaluating Model Performance (adjusted R^2)

- R-Squared can lead sometimes to misleading results in case of overfitting where the model overfits on data so it gives a value of 1 (means that it kept 100% of the variance of the data) while it is not learning but memorizing.
- Adjusted R2 is better in showing if there is real improvement or not.
- If you add more features to the data, the R2 will either increase or remains the same even if they are useless, however in the adjusted R2, it is more intuitive as if the added feature is useless, it will decrease due to the k value.
- Notes:
  1. k is the number of features in the data while n is the number of samples in the data.
  2. Its value is commonly between 0 and 1 as well and less than or equals R2.

$$R_a^2 = 1 - [(\frac{n-1}{n-k-1}).(1-R^2)]$$

🔍 Comparison of Key Regression Metrics

Metric	Definition	Sensitive to Outliers?	Use Case
MSE (Mean Squared Error)	Average of squared differences between actual and predicted values	✅ Yes	Penalizes large errors more, useful for optimization
RMSE (Root Mean Squared Error)	Square root of MSE	✅ Yes	Interpretable like actual values, popular in evaluation
MAE (Mean Absolute Error)	Average of absolute differences	❌ Less	More robust to outliers, useful for noisy data
R^2 (R-squared)	Proportion of variance explained by the model	❌ No	Measures model fit, easy to interpret
Adjusted R^2	Penalized version of R^2 that accounts for number of features	❌ No	Used when comparing models with different numbers of features

📊 Numerical Example:

Let's say you have two prediction errors:

- A small error: `2`
- A large error (outlier): `20`

✅ MAE Calculation:

$$MAE = \frac{1}{2}\left(|2| + |20|\right) = \frac{22}{2} = 11$$

- The large error **contributes 20**, no more.
- Treated as a large error, but **not exaggerated**.

❌ MSE Calculation:

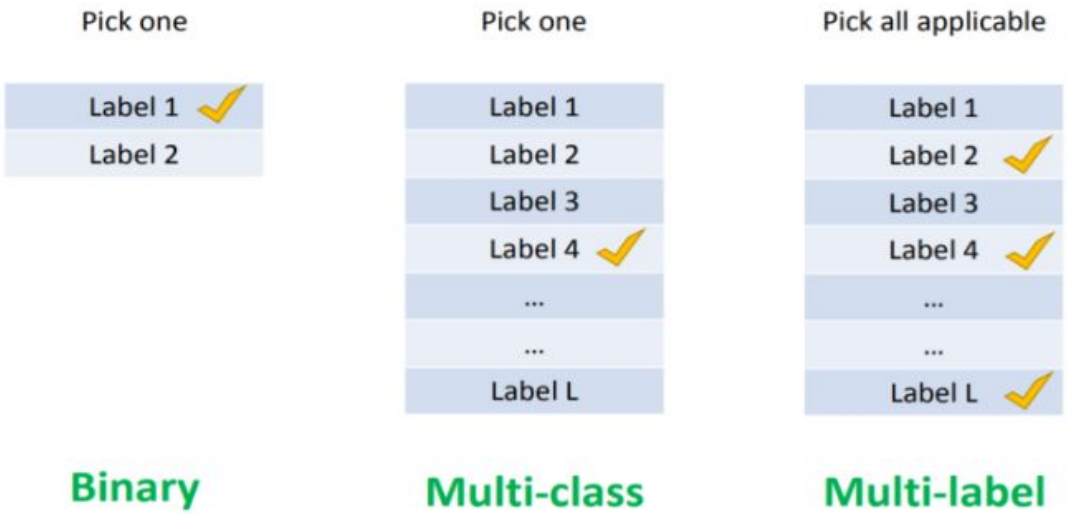
$$MSE = \frac{1}{2}\left(2^2 + 20^2\right) = \frac{4 + 400}{2} = \frac{404}{2} = 202$$

- The large error **contributes 400**!
- One outlier **dominates** the total error.

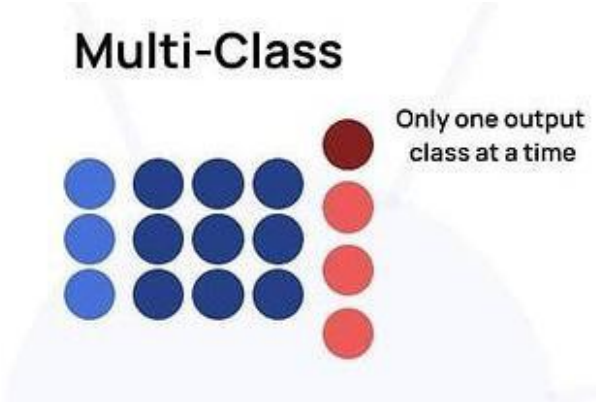
Squaring in MSE **magnifies large errors**, while MAE treats all errors **linearly**.

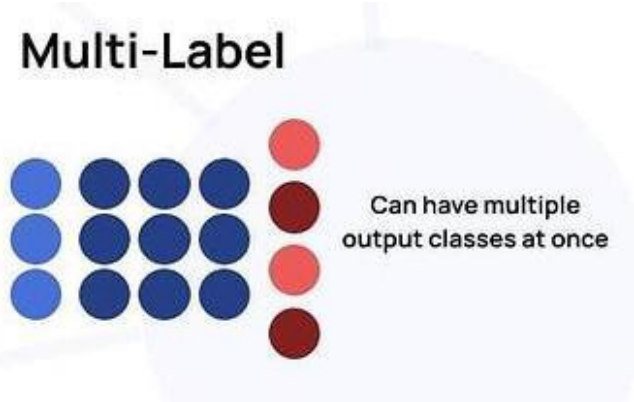
Classification

Classification



Multilabel classification

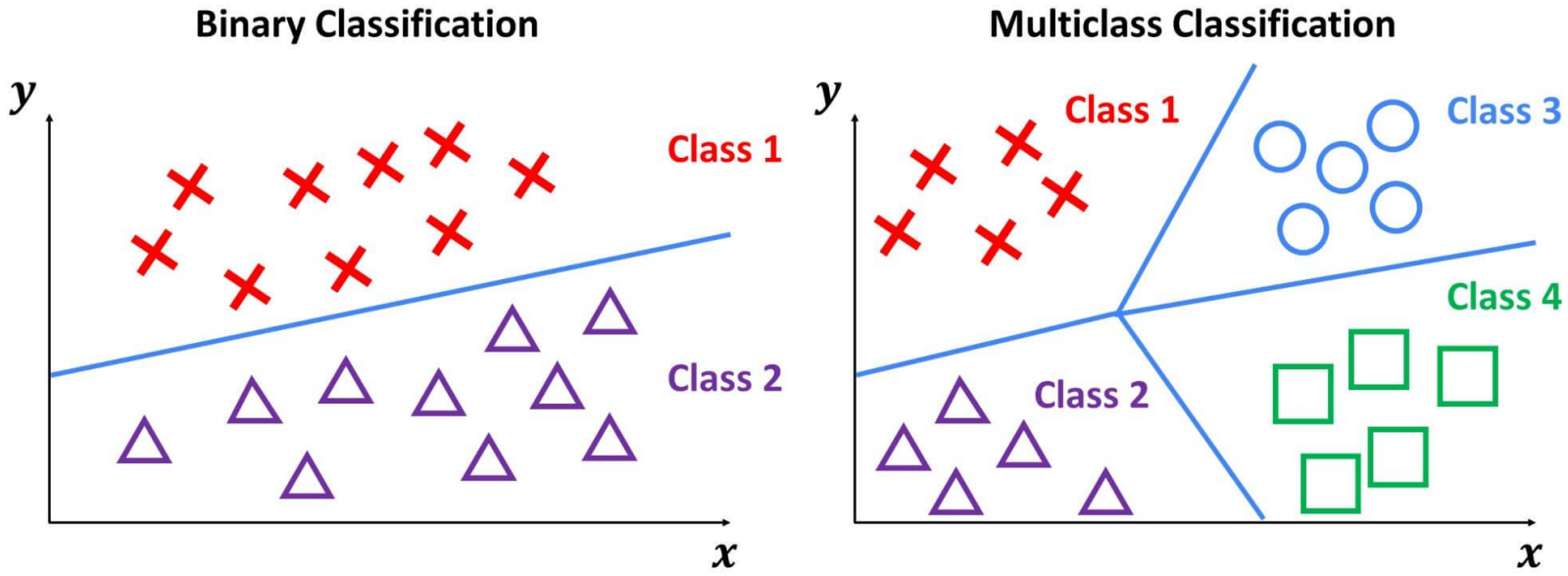




Review	sentiment
Very good quality though	Positive
The design is very odd	Negative
I advise EVERYONE DO NOT BE FOOLED!	Negative
So Far So Good!	Positive
Works great!	positive

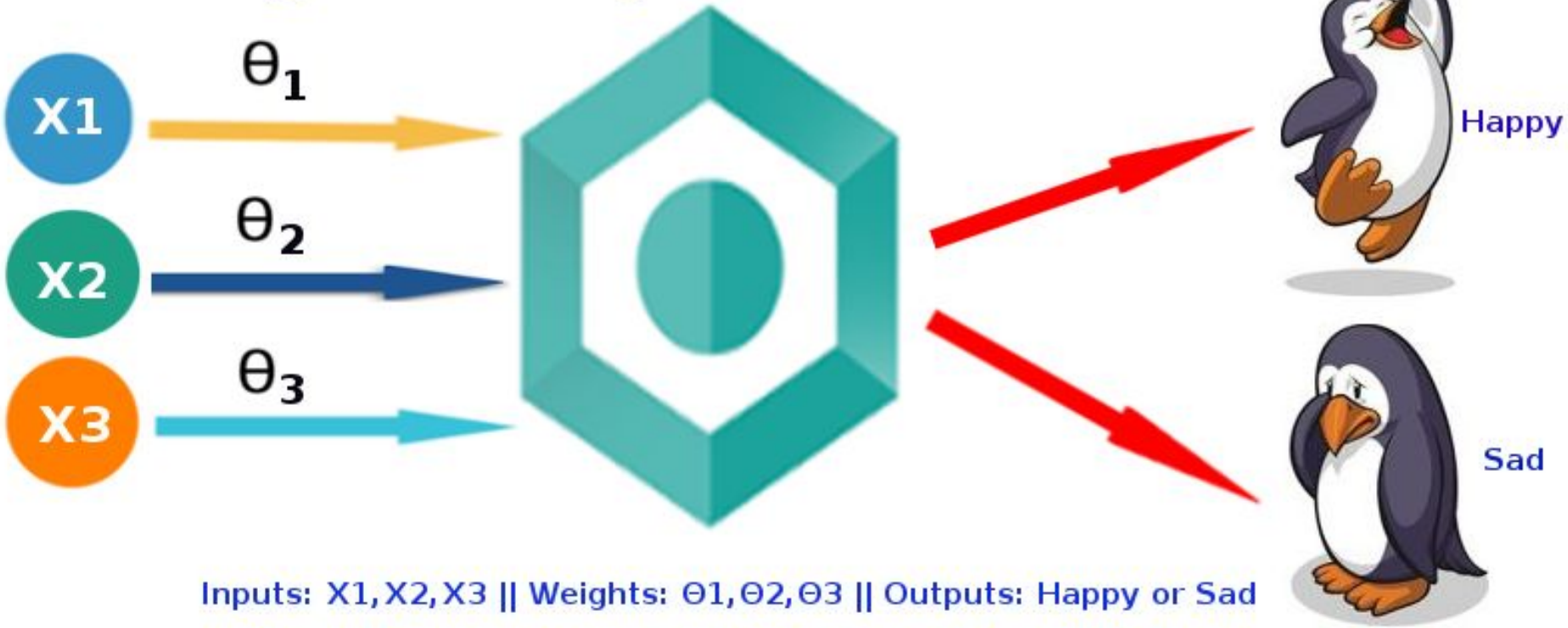
movie	Adventure	Comedy	Fantasy	Crime
Jumanji (1995)	1	0	1	0
Puccini for Beginners (2006)	0	1	0	0
How the Grinch Stole Christmas! (1966)	0	1	1	0

Multi-Class Classification



Logistic Regression

Logistic Regression Model

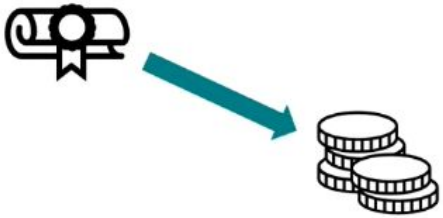


Inputs: X1, X2, X3 || Weights: Θ1, Θ2, Θ3 || Outputs: Happy or Sad

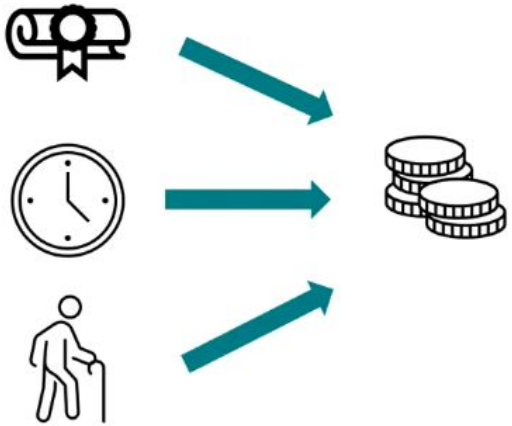
@dataaspirant.com

What is Logistic Regression?

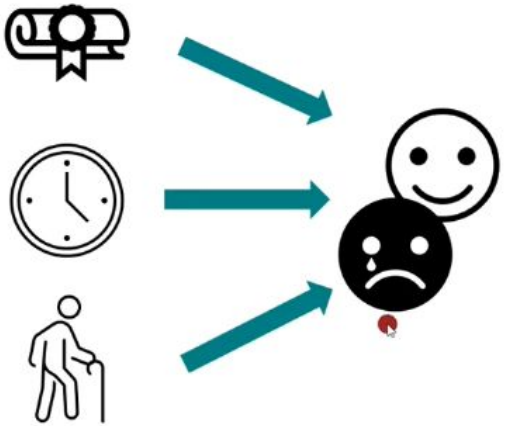
Simple linear regression



Multiple linear regression



Logistic regression



What is Logistic Regression?

Logistic regression is a special case of regression analysis and is calculated when the dependent variable is nominally or ordinally scaled.

Business example:

For an online retailer, you need to predict which product a particular customer is most likely to buy. For this, you receive a data set with past visitors and their purchases from the online retailer.

Medical example:

You want to investigate whether a person is susceptible to a certain disease or not. For this purpose, you receive a data set with diseased and non-diseased persons as well as other medical parameters.

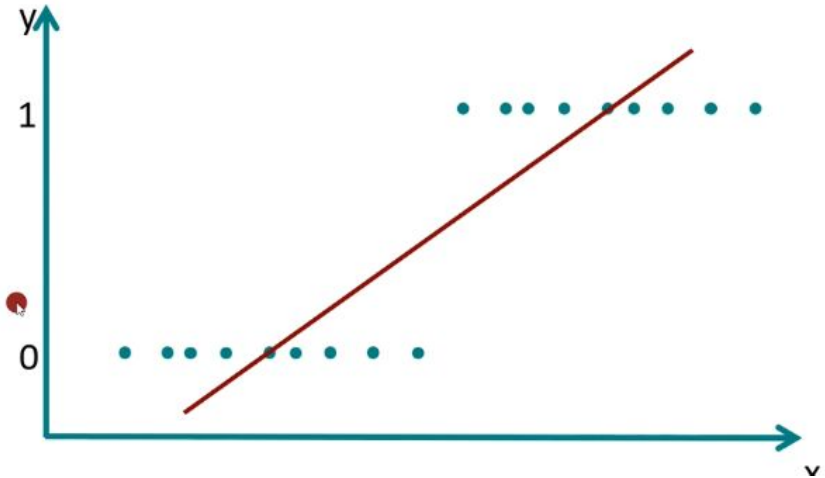
Political example:

Would a person vote for party A if there were elections next weekend?

- https://www.geeksforgeeks.org/getting-started-with-classification/
- https://www.enjoyalgorithms.com/blogs/classification-and-regression-in-machine-learning

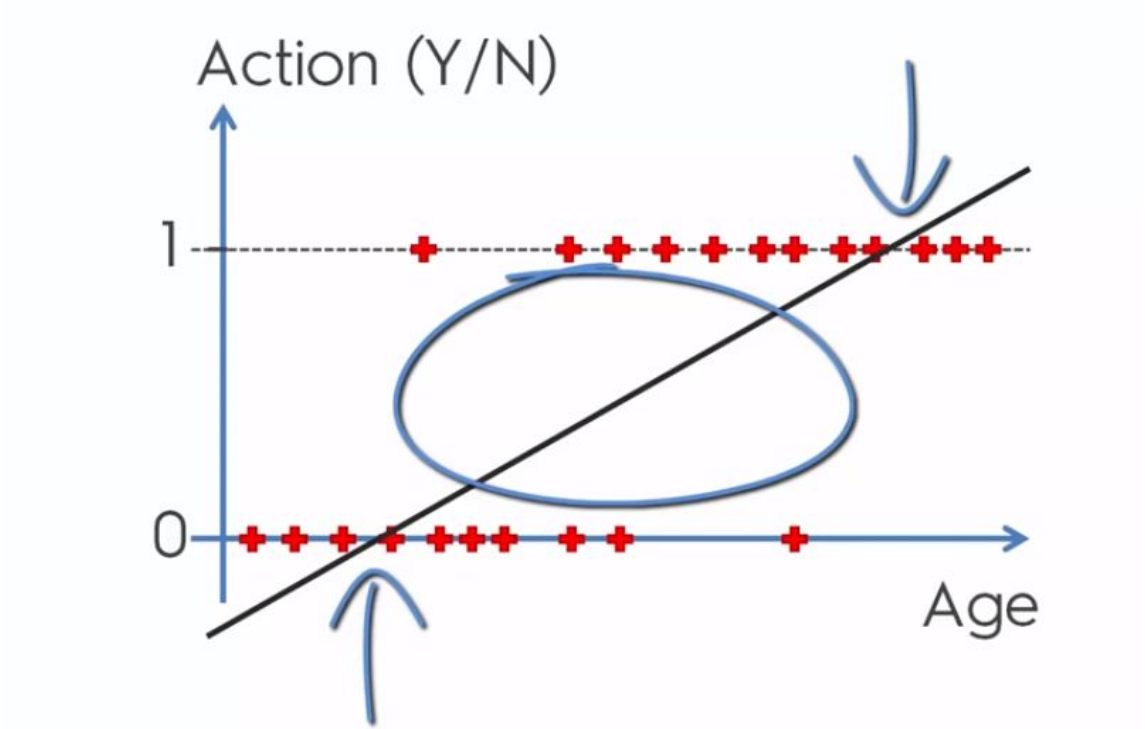
Why not just linear regression?

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

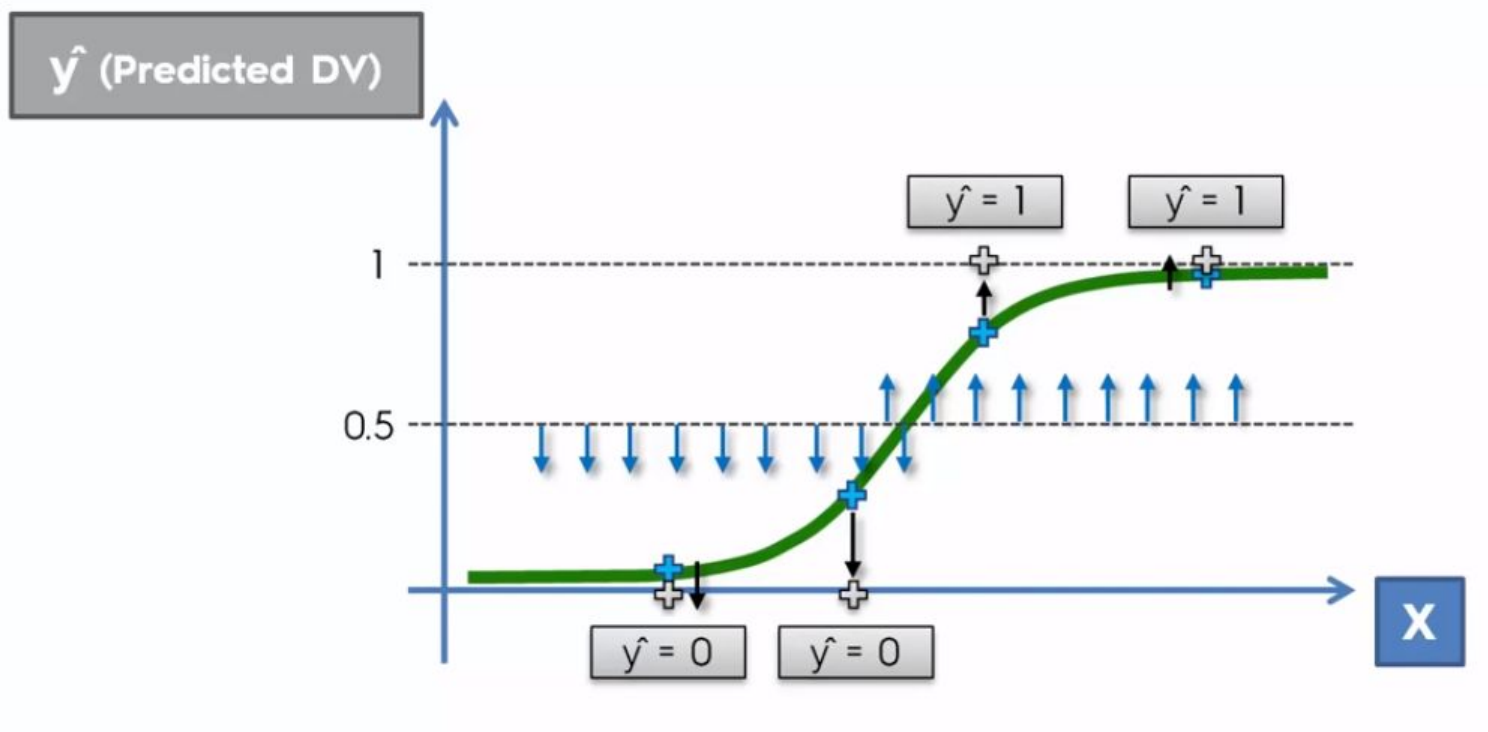


- The graph shows that values between **plus and minus infinity** can now occur.
- The goal of logistic regression is to estimate the **probability of occurrence**, not the value of the variable itself.
- The range of values for the prediction is restricted to the range between 0 and 1.
- To ensure that only values between 0 and 1 are possible, the logistic function f is used.

Logistic Regression

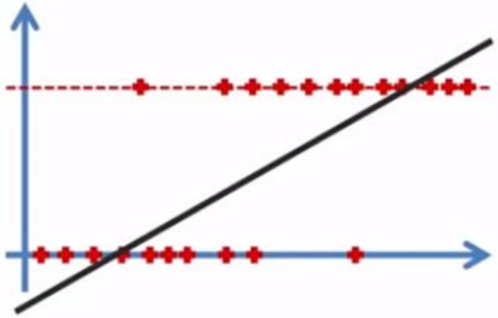


Logistic Regression



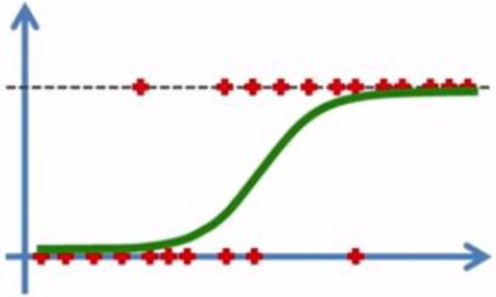
Logistic Regression

$$y = b_0 + b_1 * x$$

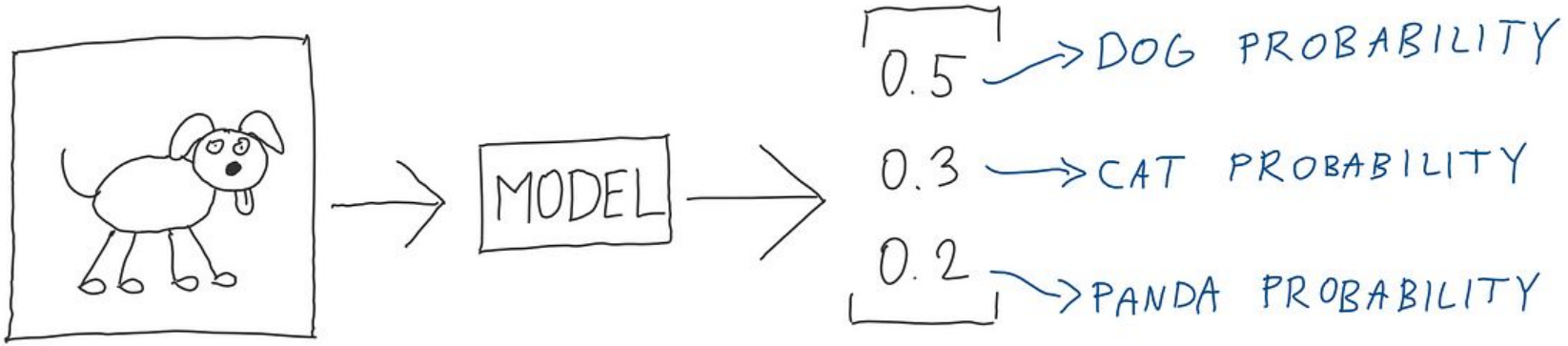


Sigmoid Function

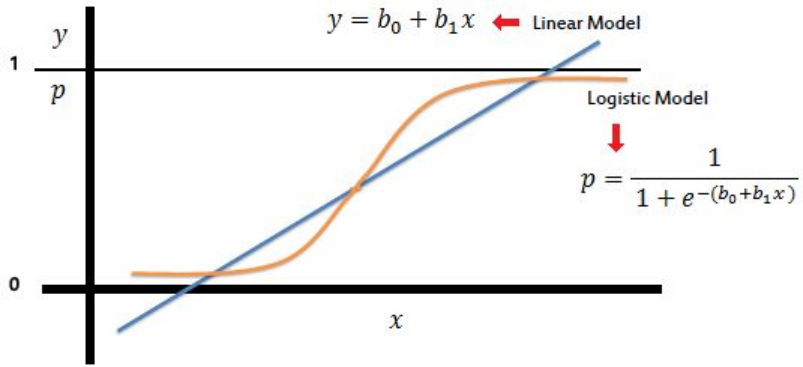
$$p = \frac{1}{1 + e^{-y}}$$



Logistic Regression



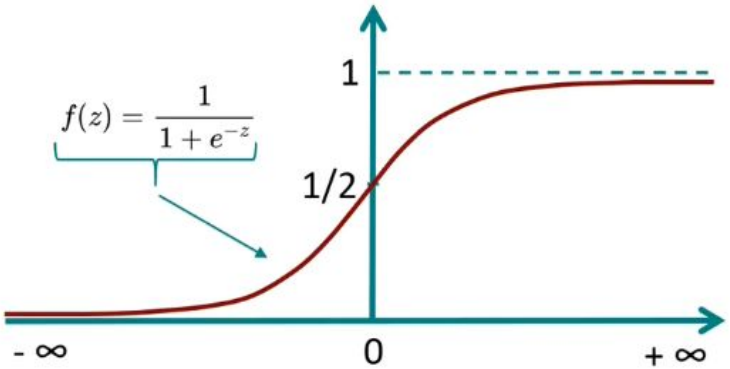
Logistic Regression



A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1

What is Logistic Regression?

The logistic model is based on the logistic function.
The important thing about the logistic function is, that only values **between 0 and 1** are possible.



$$f(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(b_1 \cdot x_1 + \ldots + b_k \cdot x_k + a)}}$$

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

What is Logistic Regression?

$$z = \Theta^T x$$

$$sigmoid(z) = \frac{1}{1 + e^{-(z)}}$$

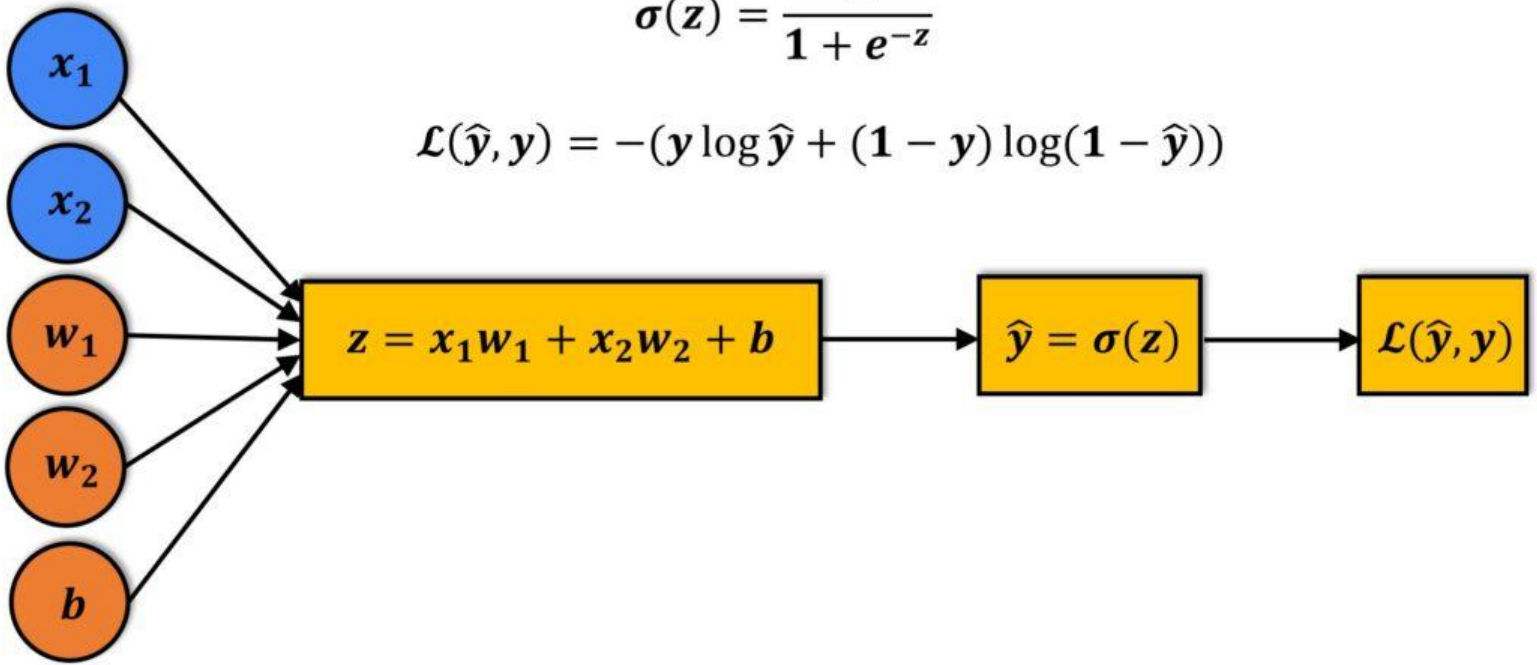
Where:

- Θ = is the weight.

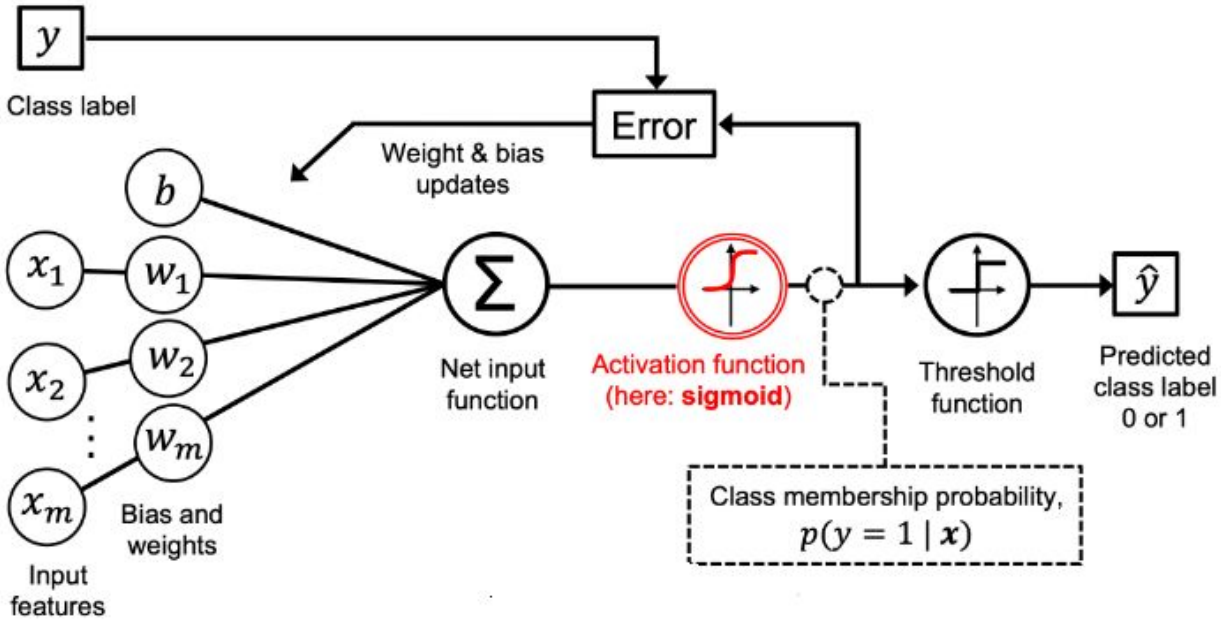
```
def sigmoid(X, weight):
    z = np.dot(X, weight)
    return 1 / (1 + np.exp(-z))
```

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

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$



Logistic Regression



Cross Entropy Loss Function

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Cost function

In binary logistic regression, the cost function (also called the objective or loss function) is used to measure the error or mismatch between the predicted probabilities and the actual binary outcomes in the training data. The goal of logistic regression is to minimize the cost function, which can be accomplished by adjusting the coefficients or weights of the model.

The most commonly used cost function in binary logistic regression defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y$$

Case 1: Actual Class $y_i = 1$

- If the model gives a high probability prediction
  the value $\log(\hat{y}_i)$ will be close to 0 (since $\log(1$
- If the model gives a low probability prediction (
  the value $\log(\hat{y}_i)$ will be large and negative (sir
  very large.

Illustrative Examples

Example 1: Correct Prediction (Low Loss)

- If $y_i = 1$ and $\hat{y}_i = 0.9$:
  $\text{loss} = -\log(0.9) \approx 0.105$ (small loss)
- If $y_i = 0$ and $\hat{y}_i = 0.1$:
  $\text{loss} = -\log(0.9) \approx 0.105$ (small loss)

Example 2: Incorrect Prediction (High Loss)

- If $y_i = 1$ and $\hat{y}_i = 0.1$:
  $\text{loss} = -\log(0.1) \approx 2.302$ (large loss)
- If $y_i = 0$ and $\hat{y}_i = 0.9$:
  $\text{loss} = -\log(0.1) \approx 2.302$ (large loss)

The following is a summary of the gradient descent algorithm for logistic regression:

1. Initialize the coefficients or weights with small random values.
2. Compute the predicted probabilities h0(x^i)for each observation in the training data using the logistic function with the current coefficients or weights.
3. Compute the gradient of the cost function with respect to each coefficient or weight using the predicted probabilities and the actual binary outcomes in the training data.
4. Update the coefficients or weights using the update rule.
5. Repeat steps 2–4 until the cost function is minimized or a stopping criterion is met.

Parametric

- ✅ Fast
- ✅ Simple
- ✅ Less data

- 🚩 Limited complexity
- 🚩 Strong assumptions
- 🚩 Poor fit (if assumptions are not correct)

Linear Regression

Logistic Regression

Nonparametric

- ✅ Flexible
- ✅ Powerful
- ✅ Effective

- 🚩 More data
- 🚩 Computationally expensive
- 🚩 Hard to interpret if models are too complex

K-Nearest Neighbors

Decision Trees

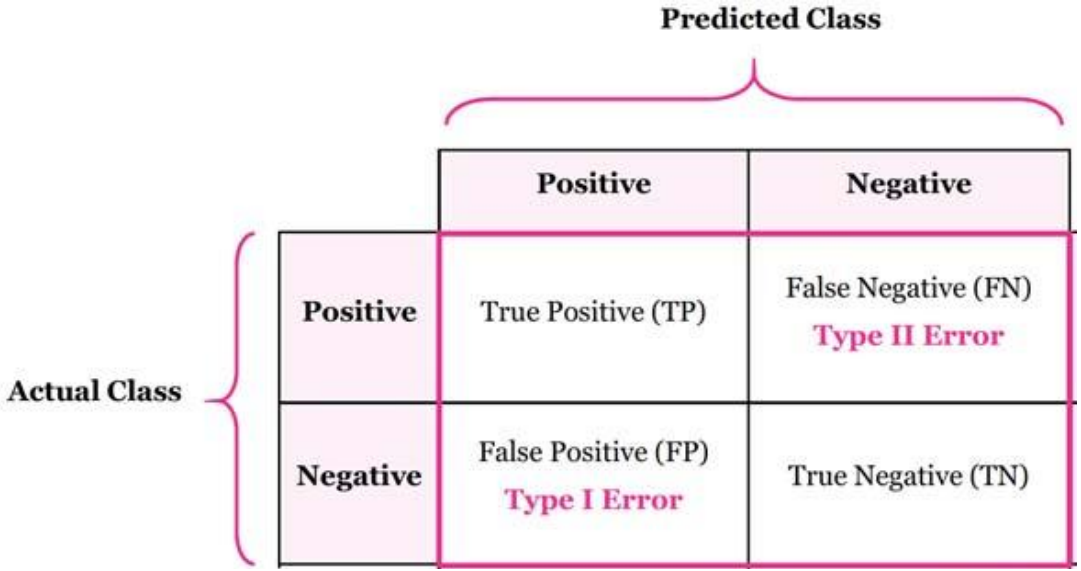
Random Forest

```
From sk.linear_model import LogisticRegression


Model = LogisticRegression()
model.fit(x_train , y_train) # y = 0 or 1
model.predict(X_test) # give u 0 or 1
```

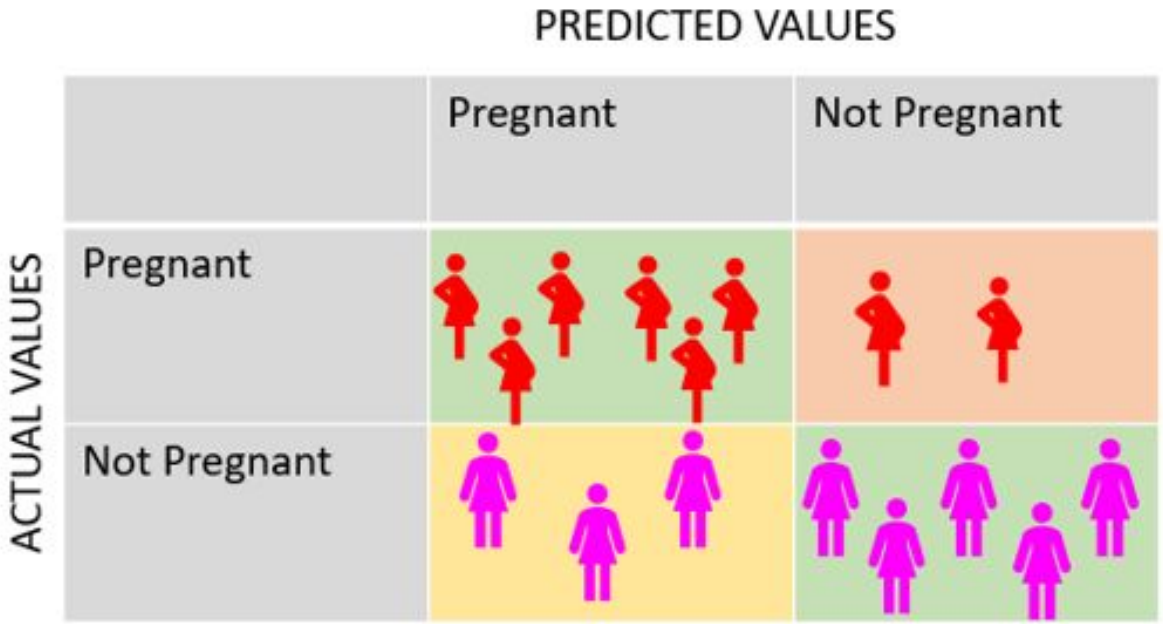
Y	Y^
0	0
1	1
1	0
0	1

Classification Metrics

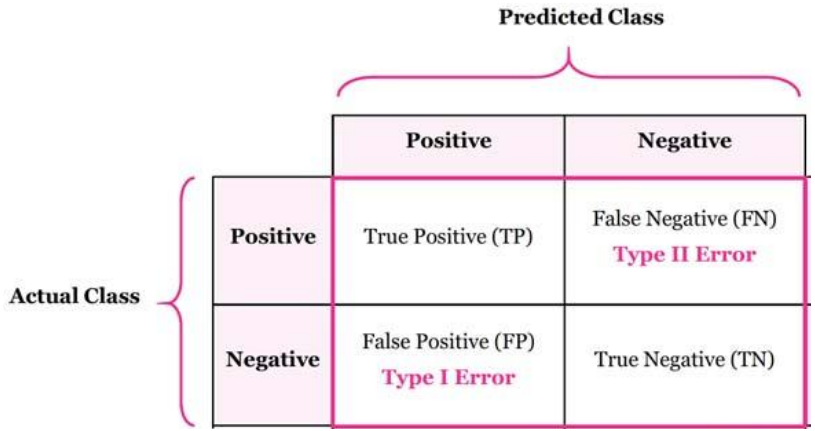
What is Confusion Matrix?



What is Confusion Matrix?

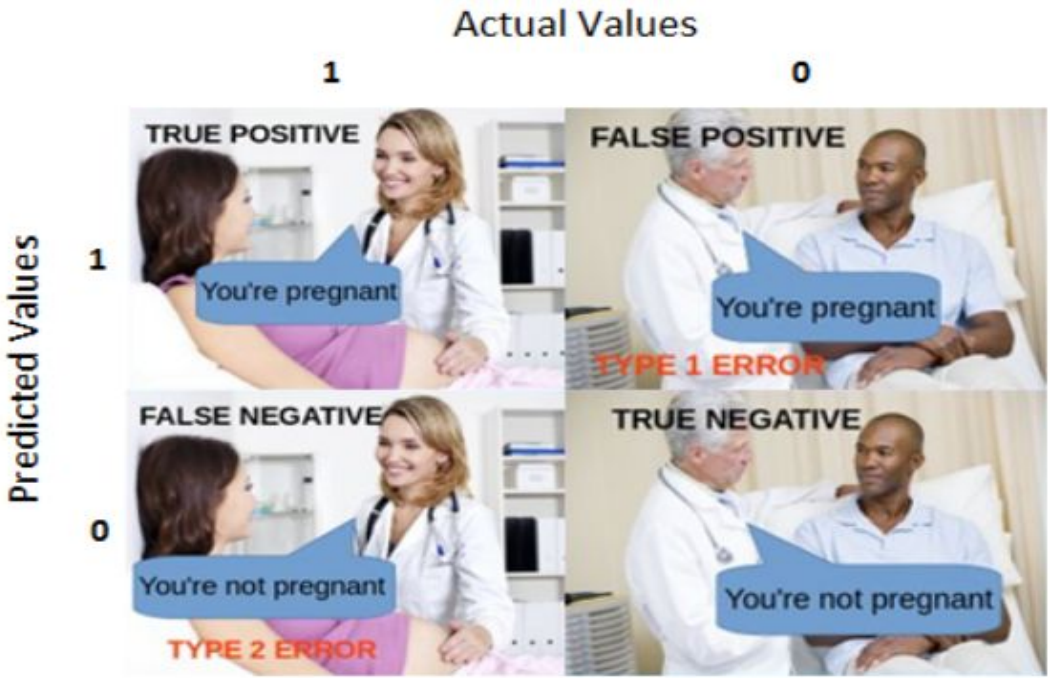


What is Confusion Matrix?



- **True Positive (TP) —** When the model says that the patient has cancer and the patient actually has it
- **False Positive (FP) —** When the model says that the patient has cancer but the patient doesn't have it
- **True Negative (TN)** — When the model says that the patient does not have cancer and the patient actually doesn't have it
- **False Negative (FN)** — When the model says that the patient doesn't have cancer but the patient actually has it. *We don't want this, do we?*

What is Confusion Matrix?

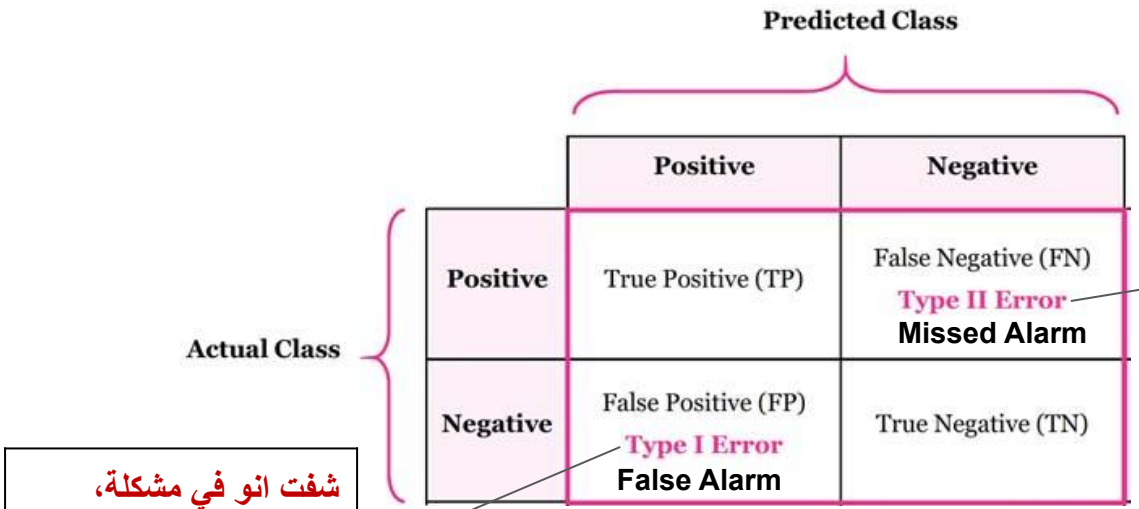


Exercise

- We select 100 people which includes pregnant women, not pregnant women and men with fat belly. Let us assume out of this 100 people 40 are pregnant and the remaining 60 people include not pregnant women and men with fat belly. We now use a machine learning algorithm to predict the outcome.

- Out of 40 pregnant women 30 pregnant women are classified correctly and the remaining 10 pregnant women are classified as not pregnant by the machine learning algorithm.
- On the other hand, out of 60 people in the not pregnant category, 55 are classified as not pregnant and the remaining 5 are classified as pregnant.

Performance evaluation Measures



ما شفت انو في مشكلة،
ولكن كانت موجودة

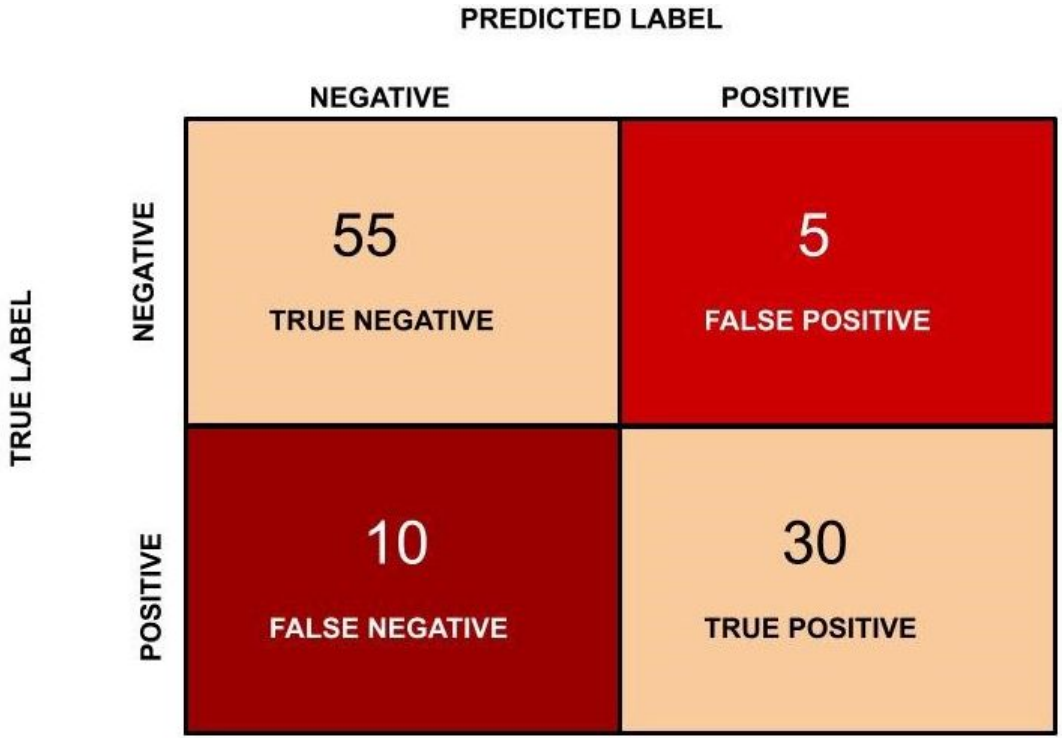
"سمعت صوت غريب بالليل، بس طنّشت وقلت يمكن الريح، وفعليًا كان في لص! "

شفت انو في مشكلة،
وما كانت موجودة

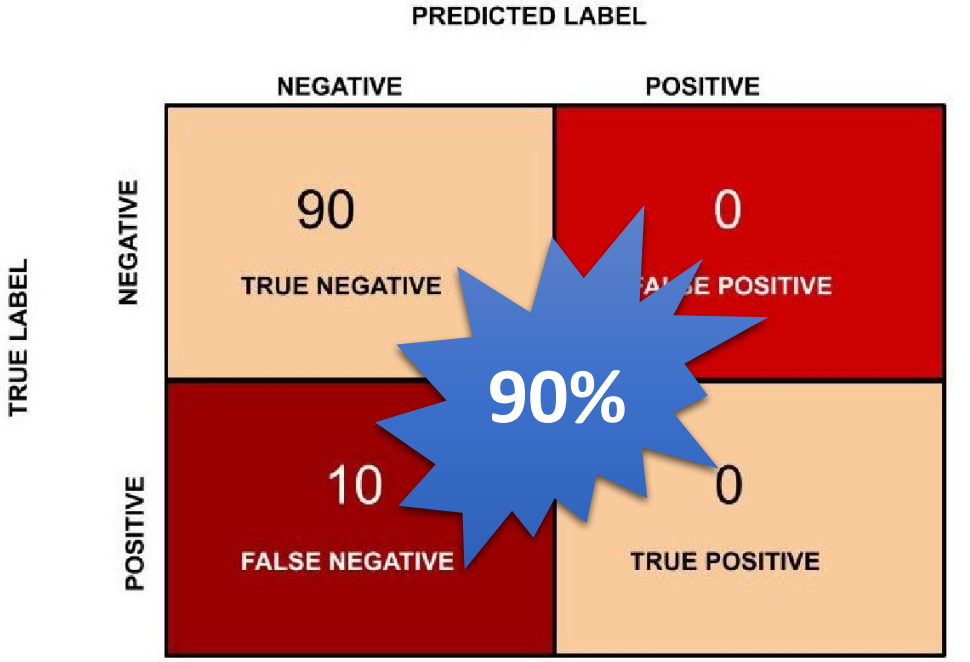
"زي إنك بلغت الشرطة إنه في لص بالبيت، ولما إجوا طلع كلب الجيران."

$$\text{Accuracy} = (TP+TN) / (TP+FP+FN+TN)$$

Example


PREDICTED LABEL
NEGATIVE
POSITIVE
TRUE LABEL
NEGATIVE
POSITIVE
55
TRUE NEGATIVE
5
FALSE POSITIVE
10
FALSE NEGATIVE
30
TRUE POSITIVE

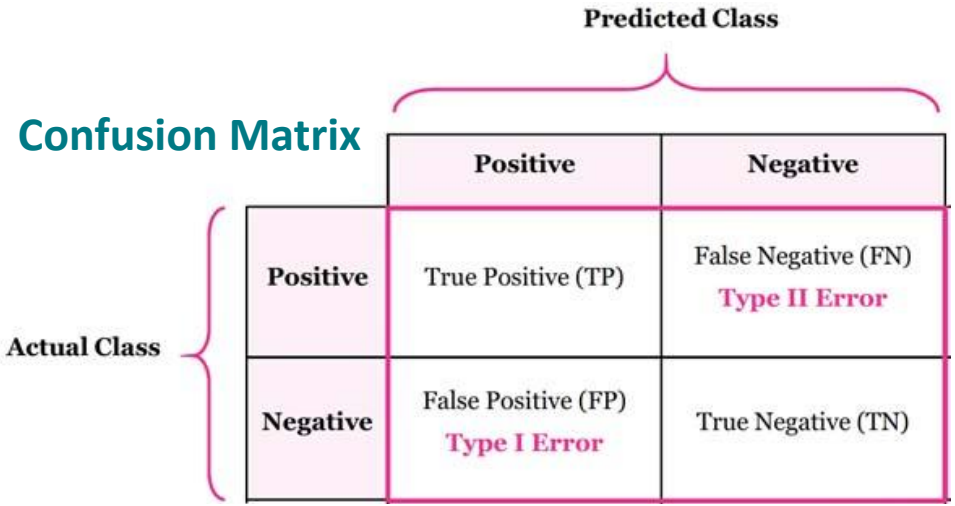
Is accuracy the best measure?



Accuracy = (TP+TN) / (TP+FP+FN+TN)

Performance evaluation Measures

Confusion Matrix



Accuracy = (TP+TN) / (TP+FP+FN+TN)

Precision = TP / (TP+FP)

Recall = TP / TP+FN

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

فعليا عندي 10 نساء (8 حامل، 2 مش حامل)

8 فعلا حامل، وتم التنبؤ انهم حامل >>> TP = 8
2 مش حامل ولكن المودل تنبأ انهم حامل >>> FP = 2

Percision = 8 / (8+2) = 80%

من بين كل النساء الي **المودل** قال عنهم حامل، فعليا 80% منهن حامل و 20% التنبؤ غلط

==========

فعليا عندي 10 نساء حامل

7 تم التنبؤ انهم حامل >>> TP = 7
3 المودل تنبأ انهم مش حامل >>> FN = 3

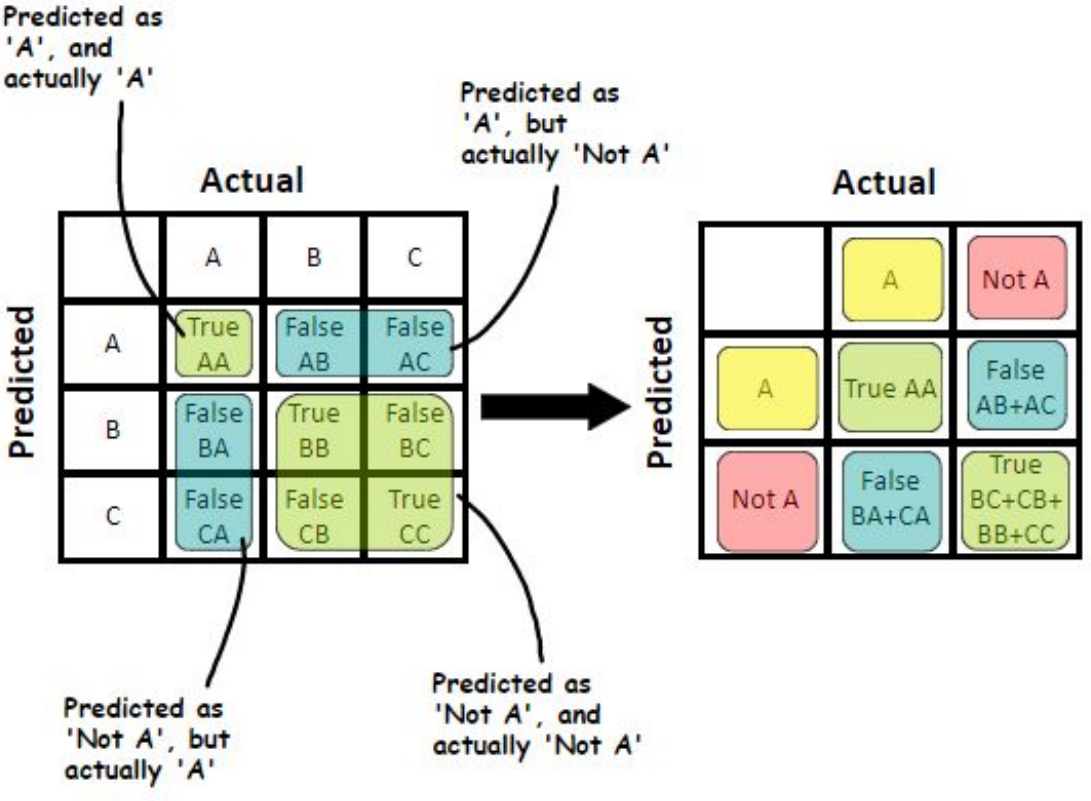
Recall = 7 / 7+3 = 70%

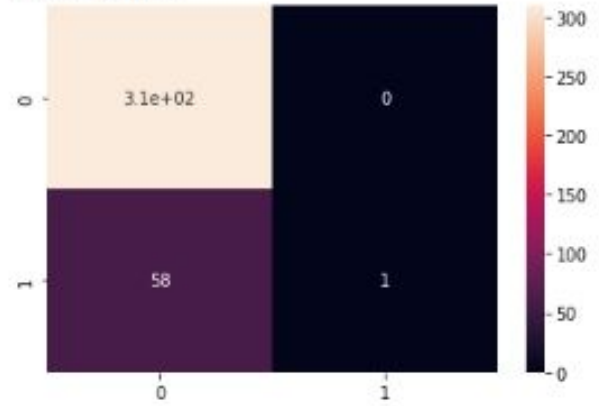
من بين كل الي **فعليا** حامل، المودل قدر يكتشف 70% منهم

==========

المقياس	التعريف	متى منيح نستخدمه؟	المشكلة	Type of Error
Accuracy =(TP+TN) / (TP + TN + FN + FP)	نسبة التوقعات الصحيحة (TP,TN) من كل العينات	لما تكون الداتا عندي Balanced	ممكن تخدعك مع الداتا اللي مش متوازنة (imbalance)	بيحسب الكل ومفيش خطأ محدد
Precision = TP /(TP+FP)	من كل اللي توقعهم المودل صح، كام وحدة منهم صح فعلا؟	**لما يكون مهم أقلل FP:** **لما يكون الFP مُكلف:** ما بدي يحط ايميل مهم بال spam بالغلط	هيتجاهل ال FN	Type 1 error (FP)
Recall (Sensitivity) = TP /(TP+FN)	من كل ال positive الحقيقين، كام وحدة اكتشف المودل؟	**لما يكون مهم اقلل FN** **او لما يكون ال FN مُكلف:** اهم اشي اني اكتشف المريض المصاب بالسرطان وما افوت اي حال	ممكن يرفع ال FP	Type 2 error
F1 Score	المتوسط بين ال **Precision** و **Recall**	لما تكون الداتا عندي unBalanced وبدي مقياس عادل بين الاثنين	-	بيركز على type1 ,type2 مع بعض

MultiClass Classification






```
[ ] accuracy = metrics.accuracy_score(y_test, y_pred)
    print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 84.24%
```

```
from sklearn.metrics import classification_report
print(classification_report(y_true=y_test, y_pred=y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       309
           1       1.00      0.02      0.03        59

    accuracy                           0.84       368
   macro avg       0.92      0.51      0.47       368
weighted avg       0.87      0.84      0.77       368
```

$$\text{Macro Avg} = \frac{\text{Metric (Class 0)} + \text{Metric (Class 1)} + \ldots}{\text{Number of Classes}}$$

$$\text{Weighted Avg} = \frac{\text{Metric (Class 0)} \times \text{Support 0} + \text{Metric (Class 1)} \times \text{Support 1}}{\text{Total Support}}$$

Classifcation metrics

6. ROC Curve and AUC (Area Under the Curve)

- The **Receiver Operating Characteristic (ROC) curve** is a graphical representation that shows the performance of a binary classifier as the discrimination threshold is varied.
  - True Positive Rate (TPR) = Recall = $\frac{TP}{TP+FN}$
  - False Positive Rate (FPR) = $\frac{FP}{FP+TN}$

  The **ROC curve** plots the TPR against the FPR at different threshold levels. The closer the curve is to the top-left corner, the better the model.

Classifcation metrics

6. ROC Curve and AUC (Area Under the Curve)

AUC (Area Under the Curve)

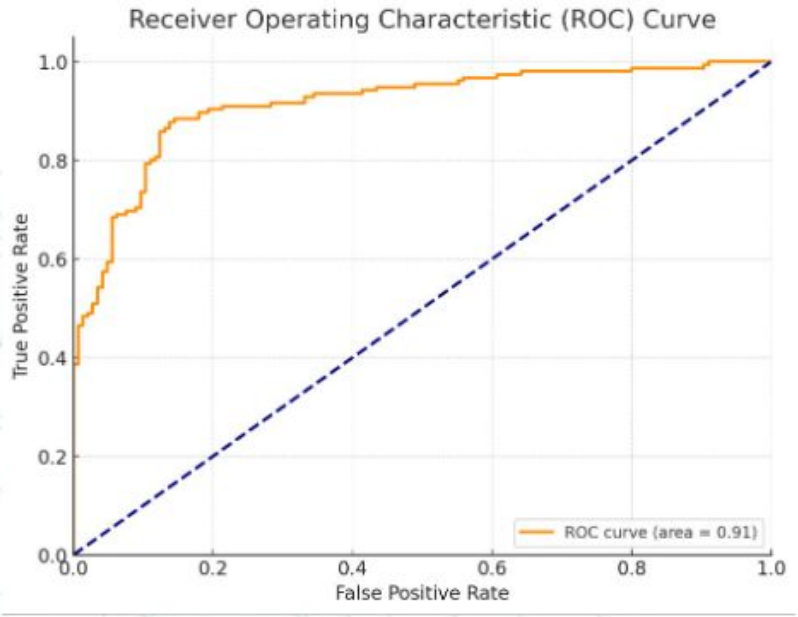
- The **AUC** value represents the degree of separability. Higher AUC means the model is better at distinguishing between positive and negative classes.

  AUC ranges from 0 to 1

  - 1 = Perfect classifier
  - 0.5 = Random guess
  - 0 = Completely wrong classification

Classifcation metrics

AUC (Area Under the Curve)



- ROC Curve is used to evaluate a classification model's performance.
- It plots True Positive Rate (TPR) against False Positive Rate (FPR) at different thresholds.
- The diagonal line represents random guessing.
- The orange line shows the model's performance with an AUC of 0.91, indicating strong performance.
- The closer the curve is to the top left corner, the better the model distinguishes between classes.

Classifcation metrics

6. ROC Curve and AUC (Area Under the Curve)

Key Insight

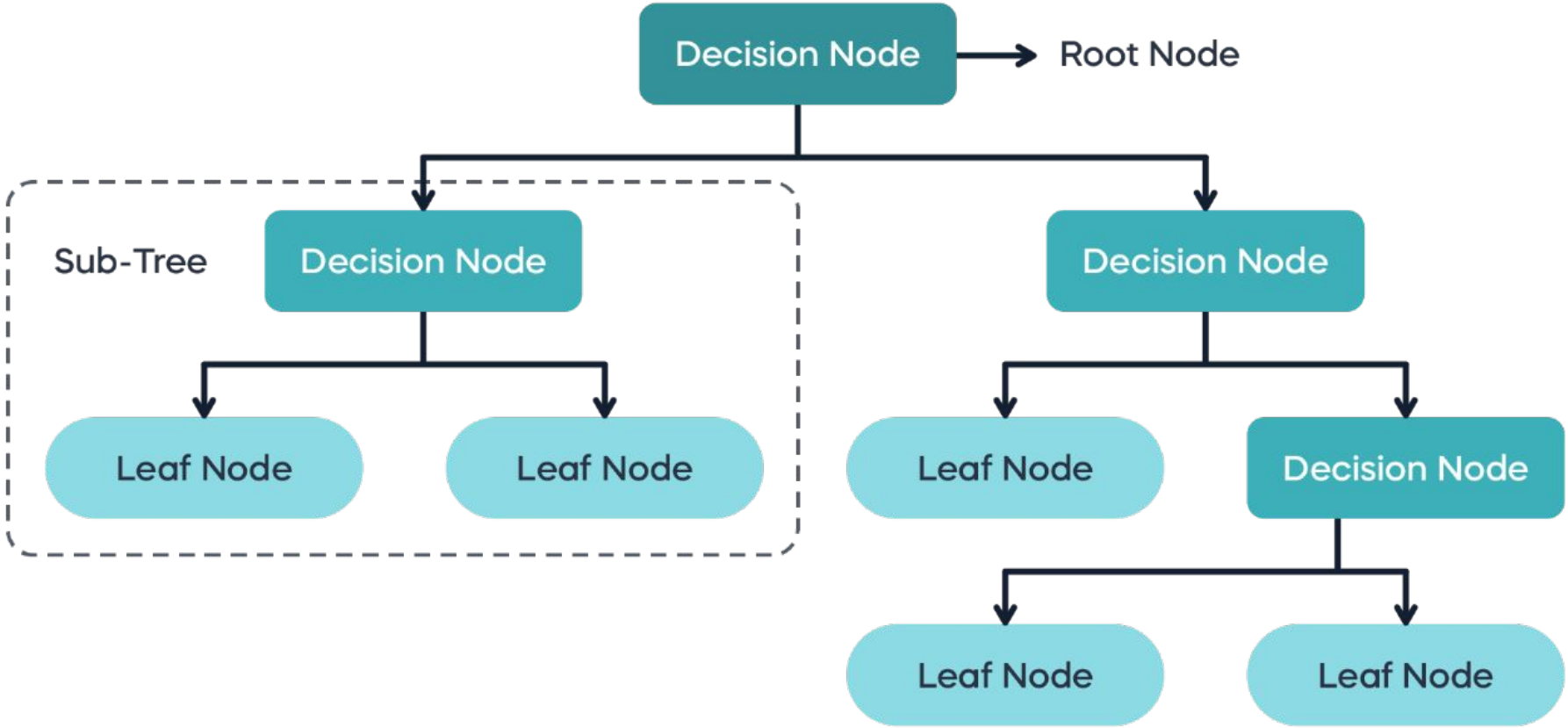
- **ROC-AUC** is ideal when you care about the ranking of predictions rather than the exact predicted class.
- It helps to understand how well the model distinguishes between classes across all thresholds.

Metric	Definition	When to Use?	When is it Useful?	When is it Not Useful?
Confusion Matrix	Table that shows the count of true positives, true negatives, false positives, and false negatives.	When you want detailed insight into the types of errors a model makes.	Useful for in-depth analysis of model behavior and error types.	Not useful for large datasets where analyzing the matrix becomes impractical.
Accuracy	Measures the ratio of correctly predicted instances to the total instances.	When classes are balanced and overall accuracy matters.	Useful for general tasks with balanced data.	Not useful when there's a class imbalance.

Metric	Definition	When to Use?	When is it Useful?	When is it Not Useful?
Precision	Measures the ratio of correctly predicted positive instances to all instances predicted as positive.	When false positives are costly or problematic.	Useful in tasks like medical diagnosis where false positives can cause unnecessary treatments.	Not useful when missing positives is more problematic than false positives.
Recall	Measures the ratio of correctly predicted positive instances to all actual positive instances.	When false negatives are costly or problematic.	Useful in tasks like security, where missing a positive case is critical.	Not useful when false positives are a bigger issue than false negatives.

Metric	Definition	When to Use?	When is it Useful?	When is it Not Useful?
F1 Score	Harmonic mean of Precision and Recall, balancing both.	When you need a balance between Precision and Recall.	Useful in cases where both false positives and false negatives need to be minimized.	Not useful if Precision or Recall is much more important than the other.
ROC/AUC	Measures the model's ability to distinguish between classes at various thresholds.	When you need to evaluate a probabilistic model's performance across thresholds.	Useful in cases with a need to understand model performance across thresholds, like fraud detection.	Not useful when there is a large class imbalance.

Decision Trees


Decision Node
Root Node
Sub-Tree
Decision Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node

CART Algorithm

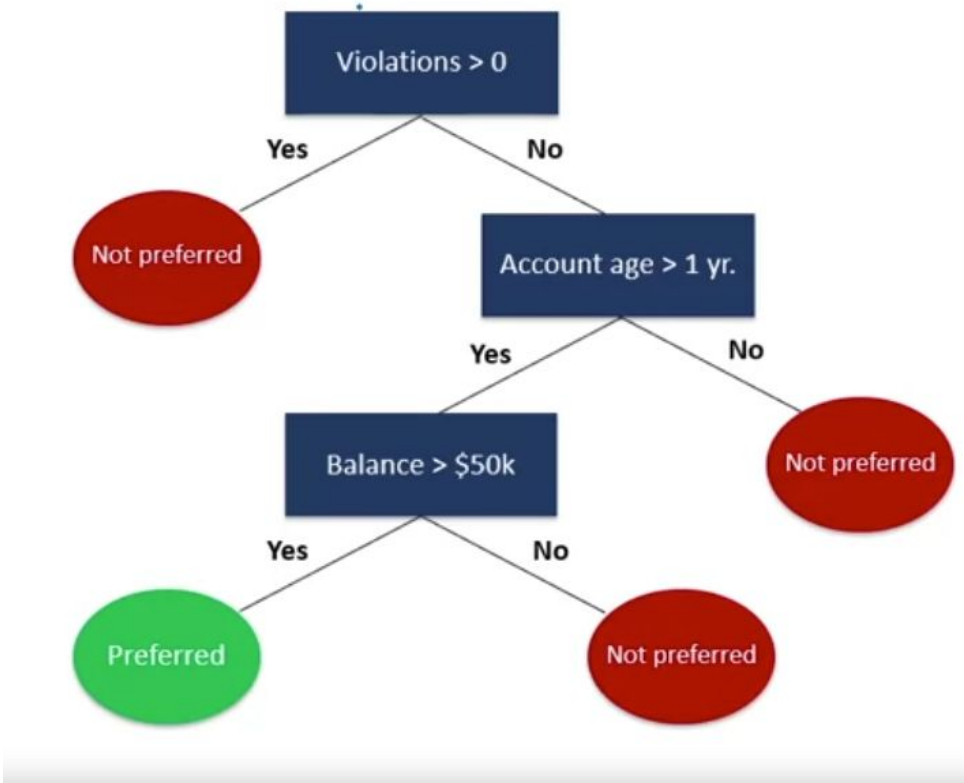
CART Algorithm:

Stands for Classification and Regression Tree.

- **Purpose:** Used in decision trees for both classification and regression tasks.
- **Binary Splits:** At each decision node, the dataset is split into two parts based on a selected feature.

Process:

- Select a feature from the training dataset.
- Split the dataset at each decision node based on the feature.
- Continue splitting recursively using other features.



Binary Splits in CART

Binary Splitting:

- Each node splits into exactly two child nodes.
- Simplifies the tree structure.

Feature Selection:

- Uses the Gini Index to determine the best feature to split on.
- The feature with the lowest Gini Index is selected for splitting.

Gini Index

Purpose: Measures the impurity or purity of a node.

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

- G: Gini Index.
- P_i: Probability of an item being classified into a particular class.
- c: Total number of classes.

Notes:

- **Gini Index = 0**: Node is pure; all samples belong to one class.
- **Higher Gini Index**: Indicates more impurity.

Steps in Calculating the Gini Index

1. **Compute for Each Feature**:
   - Calculate the Gini Index for each feature relative to the label.

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

2. **Weighted Sum**:
   - Combine the Gini Indices of each split using a weighted sum based on the number of samples.

$$\text{Weighted Gini Index} = \sum_{i=1}^{n} \left(\frac{N_i}{N_{\text{total}}} \times Gini_i \right)$$

   - Using the Weighted Gini Index helps in selecting the feature that best partitions the dataset into homogeneous subsets

3. **Select Feature with Lowest Gini Index**
   - This feature becomes the root node or the next decision node.

: Example Scenario: Determine if a bank customer deserves "Preferred" status.

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

- **Data for 'Yes' in Account Age:**
  - Total samples: 5.
  - Preferred 'Yes': 2.
  - Preferred 'No': 3.
- **Gini Index Calculation:**

$$G_{\text{Yes}} = 1 - \left(\left(\frac{2}{5} \right)^2 + \left(\frac{3}{5} \right)^2 \right) = 0.48$$

- **Data for 'No' in Account Age:**
  - Total samples: 3.
  - Preferred 'Yes': 1.
  - Preferred 'No': 2.
- **Gini Index Calculation:**

$$G_{\text{No}} = 1 - \left(\left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 \right) = 0.44$$

Calculating Gini Index for 'Account Age'

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

- **Data for 'Yes' in Account Age:**
  - Total samples: 5.
  - Preferred 'Yes': 2.
  - Preferred 'No': 3.
- **Gini Index Calculation:**

$$G_{\text{Yes}} = 1 - \left(\left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right) = 0.48$$

- **Data for 'No' in Account Age:**
  - Total samples: 3.
  - Preferred 'Yes': 1.
  - Preferred 'No': 2.
- **Gini Index Calculation:**

$$G_{\text{No}} = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right) = 0.44$$

- **Weighted Gini Index:**

$$G_{\text{Account Age}} = \left(\frac{5}{8} \times 0.48\right) + \left(\frac{3}{8} \times 0.44\right) = 0.46$$

- **Interpretation:**
  - This value represents the impurity of the dataset when split by 'Account Age'.

Calculating Gini Index for Violations and Balance

Account age > 1	Violations > 0	Balance > $50,000	Preferred
Yes	No	Yes	No
Yes	Yes	Yes	No
No	No	Yes	No
Yes	No	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
No	No	Yes	Yes
No	Yes	Yes	No

Gini Indices:

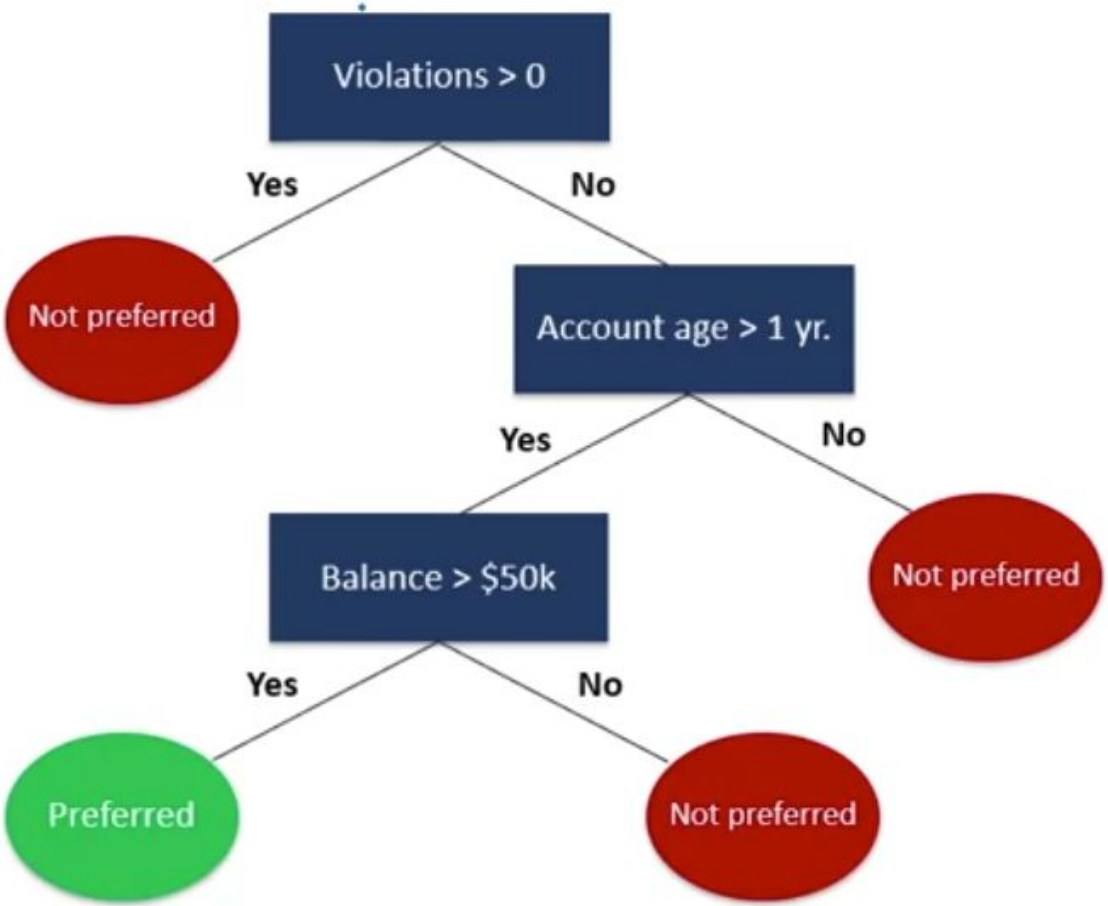
- **Violations**: 0.38
- **Balance**: 0.43
- **Account Age**: 0.46

Comparing Gini Indices of All Features to get decision:

- Since **Violations** has the lowest Gini Index, it **becomes** the **root decision node.**

Building the Decision Tree

- **Root Node**: 'Violations'
- **Split Criteria:**
  - Yes: Customer has violations.
  - No: Customer has no violations.
- **Next Steps**:
  - For each subset resulting from the split on 'Violations', calculate the Gini Index for remaining features.
  - Select the feature with the lowest Gini Index as the next decision node.
- **Process Continues**:
  - Repeat until all nodes are pure (Gini Index = 0) or stopping criteria are met.



Determine the next best feature to split on.

Remaining Features: 'Account Age', 'Balance'

Recalculate Gini Indices for Remaining Features

- Compare Gini Indices:
  - Assume:
    - $G_{\text{Account Age}} = 0.4$
    - $G_{\text{Balance}} = 0.43$
- Select '**Account Age**' as it has a lower Gini Index.

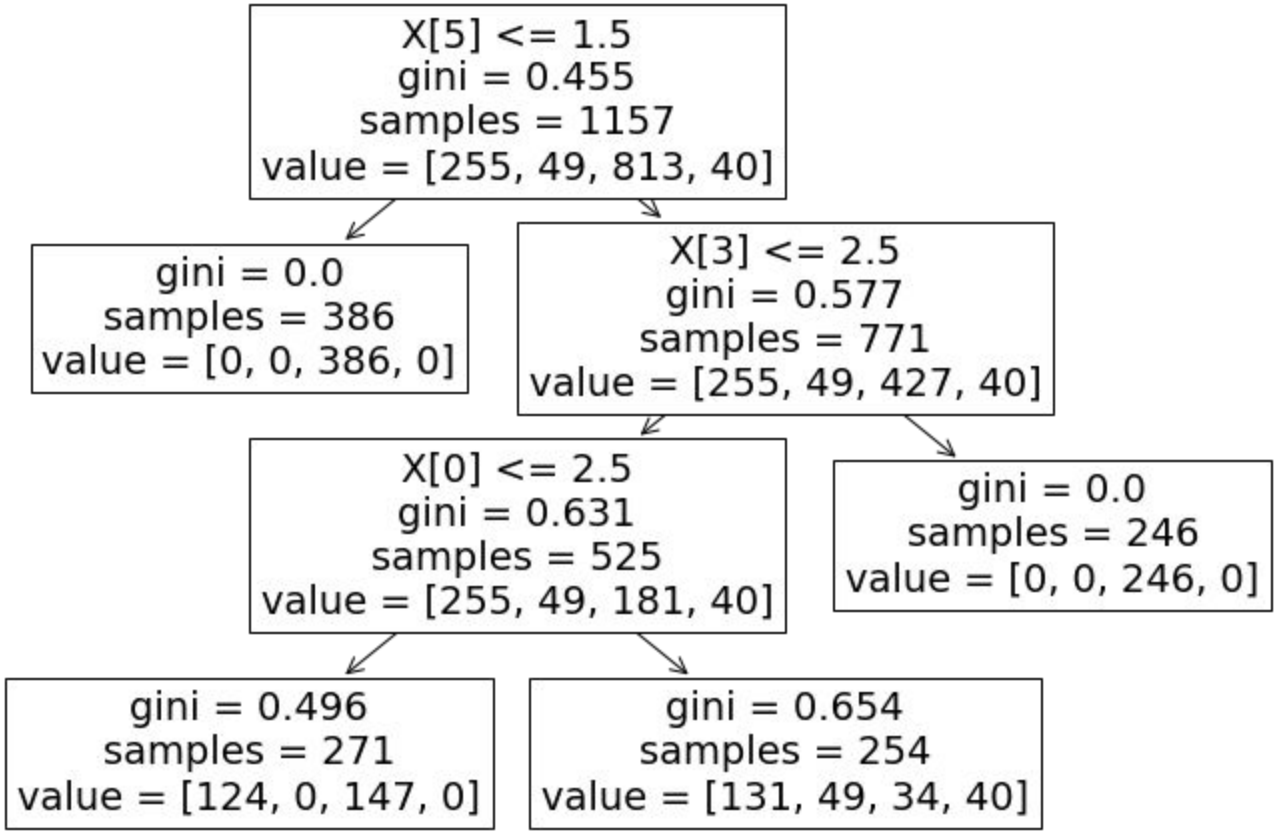

X[5] <= 1.5
gini = 0.455
samples = 1157
value = [255, 49, 813, 40]
gini = 0.0
samples = 386
value = [0, 0, 386, 0]
X[3] <= 2.5
gini = 0.577
samples = 771
value = [255, 49, 427, 40]
X[0] <= 2.5
gini = 0.631
samples = 525
value = [255, 49, 181, 40]
gini = 0.0
samples = 246
value = [0, 0, 246, 0]
gini = 0.496
samples = 271
value = [124, 0, 147, 0]
gini = 0.654
samples = 254
value = [131, 49, 34, 40]

Table 1 Comparisons between different Decision Tree Algorithms

Algorithms	ID3	C4.5	C5.0	CART
Type of data	Categorical	Continuous and Categorical	Continuous and Categorical, dates, times, timestamps	continuous and nominal attributes data
Speed	Low	Faster than ID3	Highest	Average
Pruning	No	Pre-pruning	Pre-pruning	Post pruning
Boosting	Not supported	Not supported	Supported	Supported
Missing Values	Can't deal with	Can't deal with	Can deal with	Can deal with
Formula	Use information entropy and information Gain	Use split info and gain ratio	Same as C4.5	Use Gini diversity index

Hyperparameter Tuning in CART Algorithm

- **What are Hyperparameters?**

  They are parameters that we can modify to influence how the machine learning model is built.

- **Why do we tune Hyperparameters in the CART Algorithm?**

  To control how the nodes are split and how large the tree grows.

  Because CART is prone to overfitting, tuning helps us avoid this issue.

Importance of Hyperparameter Tuning

- **Model Generalization**

  Tuning hyperparameters helps ensure the model can make good predictions on new data.

- **Overfitting**
  - Without tuning, the tree can learn too many details from the training data and may not generalize well.

Key Hyperparameters in Scikit-learn:

- `max_depth`
- `min_samples_split`
- `min_samples_leaf`
- `splitter`

Hyperparameter (`max_depth`)

- **What it does:**
  Controls the maximum depth of the tree.
- **Default behavior:**
  If `max_depth` is not set, the tree will keep splitting until the leaves are pure (Gini Index = 0).
- **Why tune it?**
  To control how large the tree grows and how many splits are allowed.
  Helps prevent the tree from **becoming too complex** (overfitting).
- **How to find the right value?**
  Evaluate model performance with different `max_depth` values and choose the best one.

Hyperparameters 2 & 3 - `min_samples_split` & `min_samples_leaf`

- **`min_samples_split`:**
  The minimum number of samples required to split a node.
  Default value: 2.
- **`min_samples_leaf`:**
  The minimum number of samples that a leaf node must have.
  Default value: 1.
- **Why tune them?**
  Increasing these values can reduce the likelihood of overfitting.
  However, be careful to avoid underfitting (the model becomes too simple).
- **Tips:**
  Try different values and observe their effect on model performance.
  Check if the model starts to lose predictive power.

Hyperparameter 4 - `splitter`

What it does:

Determines the strategy used to split nodes.

Possible values:

- `"best"`: Uses the Gini Index to find the best split.
- `"random"`: Chooses the feature randomly for splitting.

Pros & Cons:

- `"best"`:
  - Provides the best split in terms of purity.
  - Can be slower because it calculates the Gini Index for all options.
- `"random"`:
  - Faster because it doesn’t calculate for all options.
  - Helps avoid overfitting but may not give the best possible split.

```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(
    max_depth=5,
    min_samples_split=10,
    min_samples_leaf=5,
    max_features='sqrt',
    criterion='gini'
)
```

- Don’t go deeper than **5 levels** (max depth).
- Only **split** if you have at least **10 samples**.
- Each **leaf node** must have at least **5 samples**.
- Try √ **(square root) of the number of features** at each split.
- **Evaluate the splits** using the **Gini Index**.

Avoiding Overfitting with Hyperparameters

- **How does overfitting occur?**
  - When the model learns too many details and noise from the training data.
- **Role of Hyperparameters:**
  - Control the complexity of the tree and the number of splits.
  - Balance between model complexity and its ability to generalize.

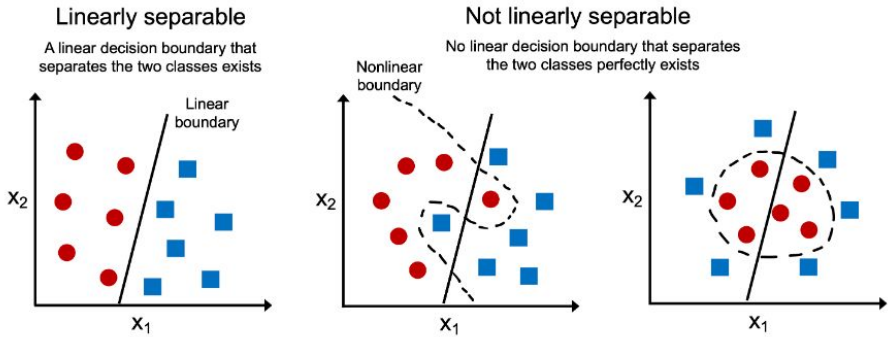
Strategies:

- Set `max_depth` to limit tree depth.
- Increase `min_samples_split` and `min_samples_leaf` carefully.
- Experiment with the `"random" splitter` in certain cases.

Assumptions For Logistic Regression (LR) and Decision Tree (DT)

- **Logistic Regression (LR)**
  - Assumes a **linear relationship** between features and the log-odds of the outcome.
  - Assumes independence between features (no multicollinearity).
  - Works best if data is relatively clean and well-distributed.
  - Outliers sensitivity
- **Decision Tree (DT)**
  - **Non-parametric** → makes **no assumption** about linearity or distribution.
  - Splits data into rules based on thresholds.
  - Very prone to **overfitting** if the tree grows too deep.
  - Outliers sensitivity

Logistic Regression = constrained model (needs assumptions).
Decision Tree = flexible model (fewer assumptions).



Decision Tree – Problems & Solutions

Main Problems	Practical Solutions
Overfitting – tree memorizes training data instead of generalizing	Use **Pruning** (reduce unnecessary branches)
Instability / High Variance – small data changes lead to big model changes	Use **Ensembles** (Random Forest, Gradient Boosting)
High Complexity – deep trees are hard to interpret	Set limits (max depth, min samples per leaf)

Pruning

Pruning

- Pruning is a technique that reduces the size of a decision tree by removing branches of the tree which provide little predictive power
- It is a **regularization** method that reduces the complexity of the final model, thus reducing overfitting
  - Decision trees are prone to overfitting!

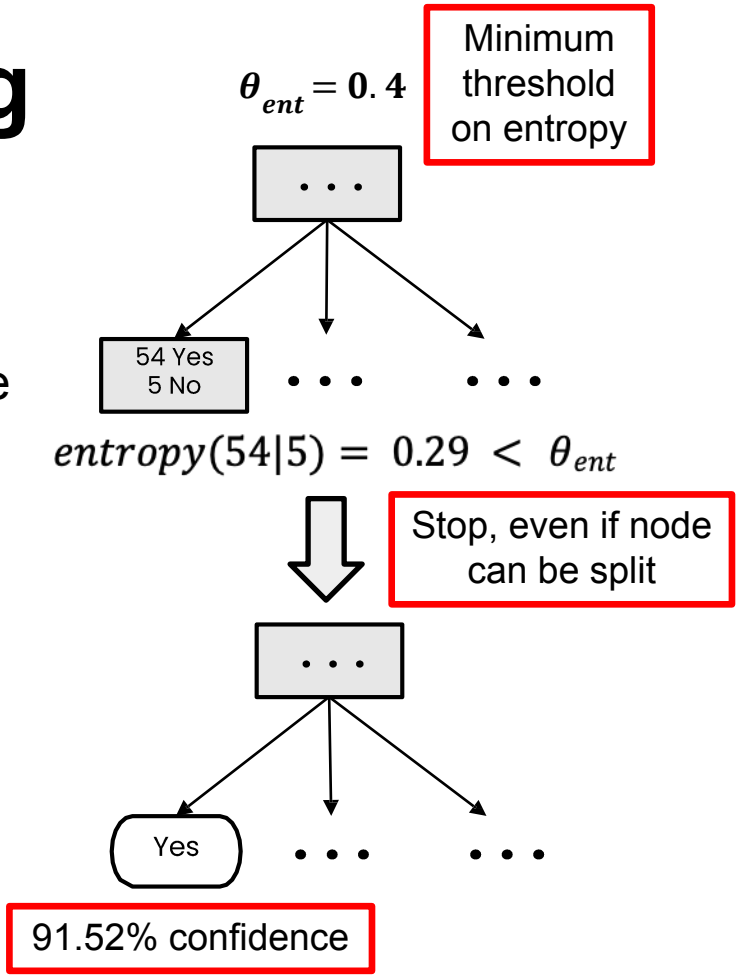
- Pruning methods:
  - Pre-pruning: Stop the tree building algorithm before it fully classifies the data
  - Post-pruning: Build the complete tree, then replace some non-leaf nodes with leaf nodes if this improves validation error

Pre-pruning

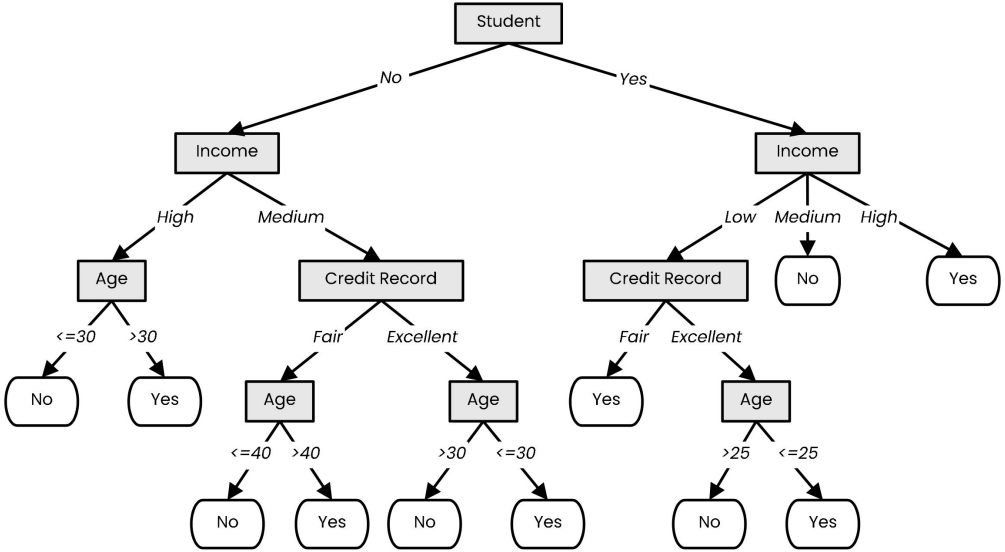
- Pre-pruning implies early stopping:
  - If some condition is met, the current node will not be split, even if it is not 100% pure
  - It will become a leaf node with the label of the majority class in the current set

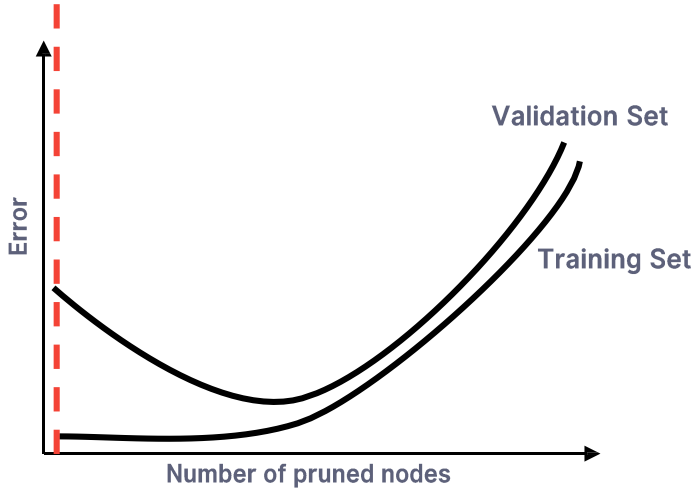
(the class distribution could be used as prediction confidence)

- Common stopping criteria include setting a threshold on:
  - Entropy (or Gini Impurity) of the current set
  - Number of samples in the current set
  - Gain of the best-splitting attribute
  - Depth of the tree



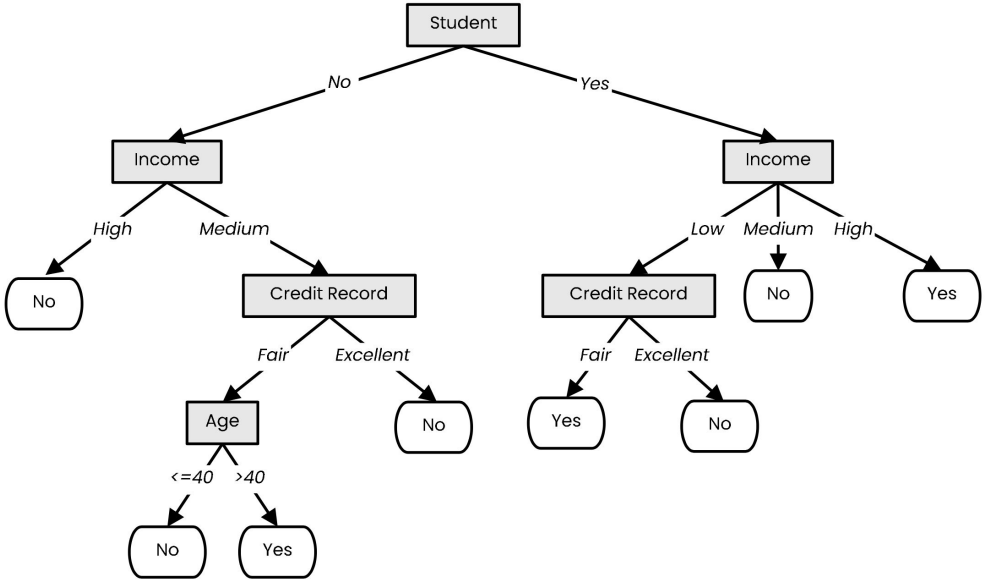
Post-pruning

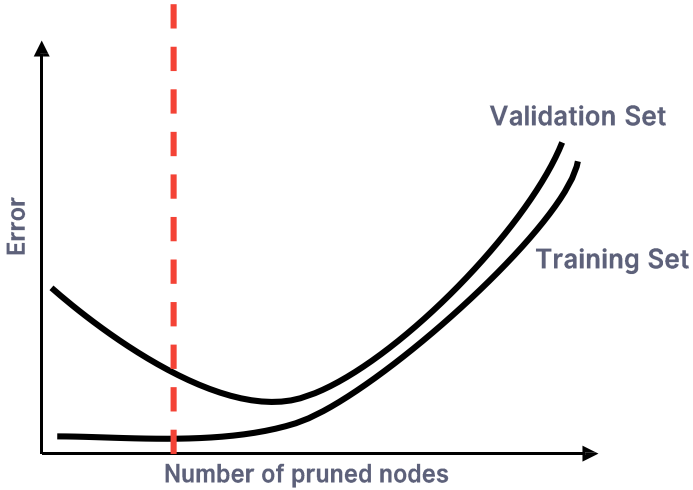




- Prune nodes in a bottom-up manner, if it decreases validation error

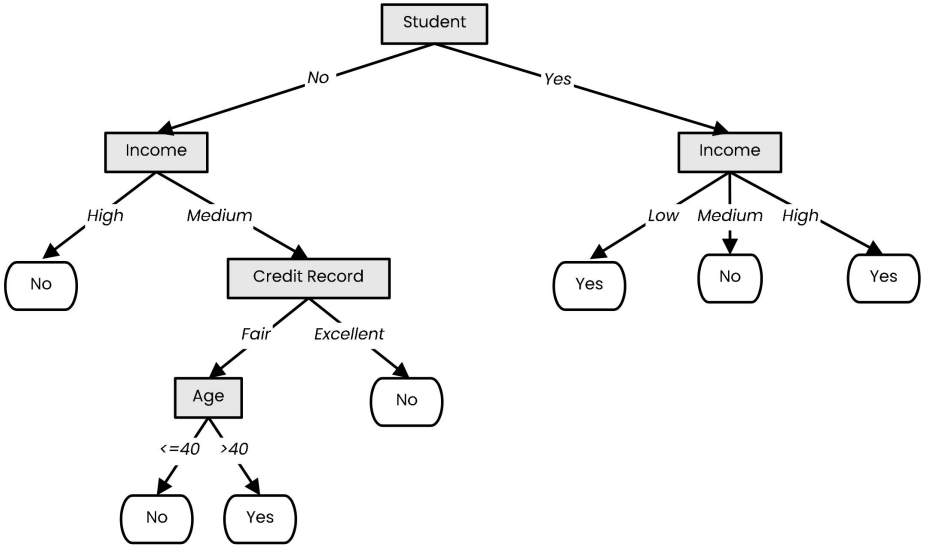
Post-pruning



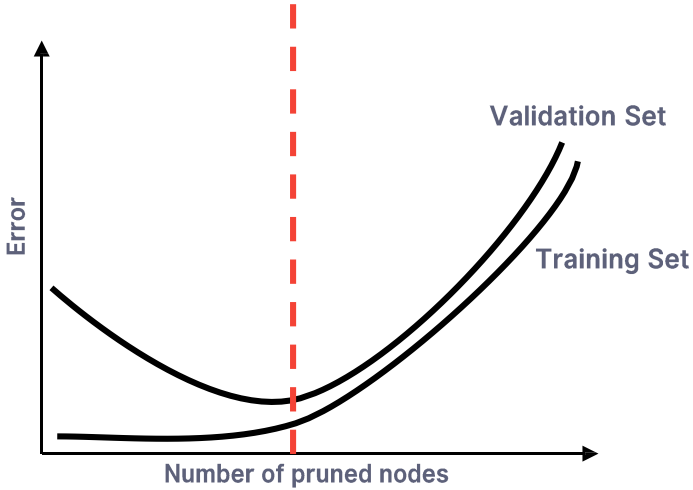


- Prune nodes in a bottom-up manner, if it decreases validation error

Post-pruning



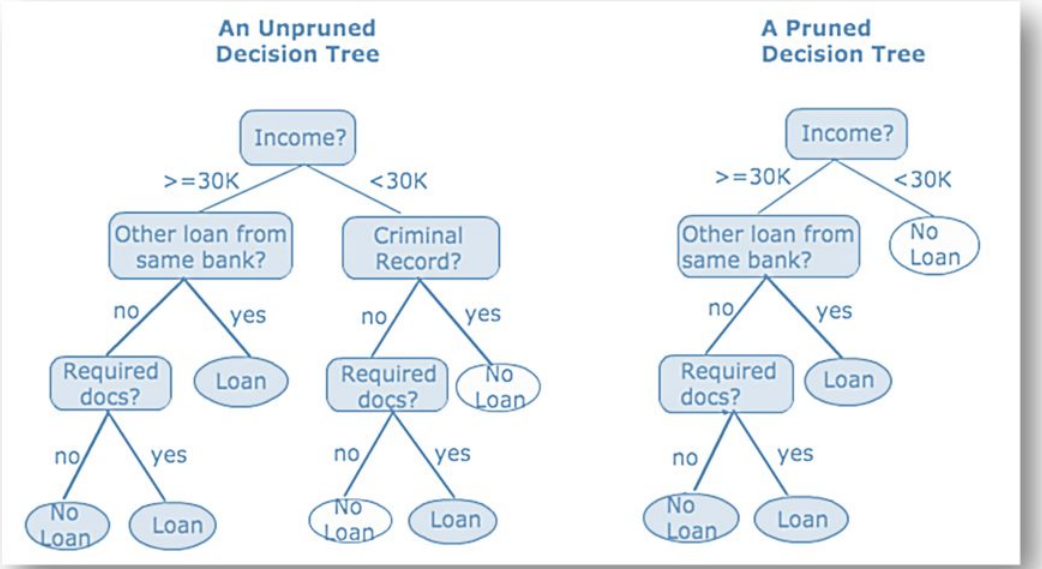
- Prune nodes in a bottom-up manner, if it decreases validation error



Self study

Pruning decision trees

1. **Reduced Error Pruning**
2. **Cost Complexity Pruning**
3. **Minimum Error Pruning**



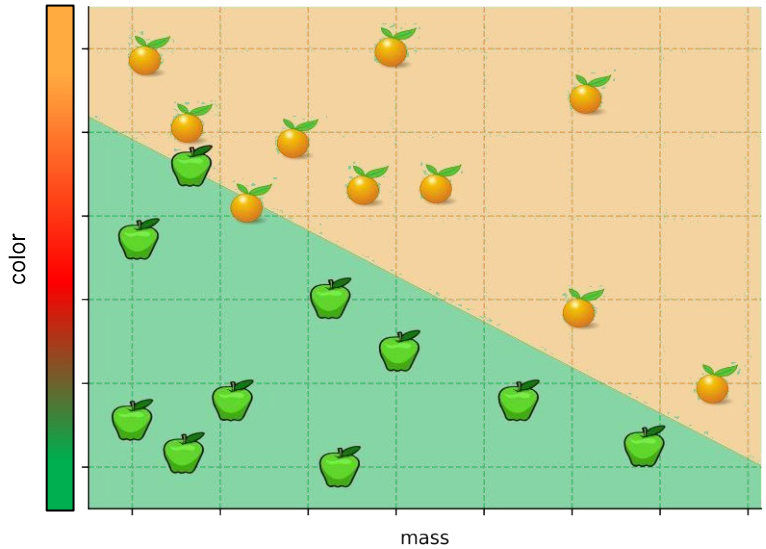
The process of IG-based pruning requires us to identify "twigs", nodes whose **children** are all **leaves**.

"Pruning" a twig removes all of the leaves which are the children of the twig, and makes the **twig** a **leaf**.

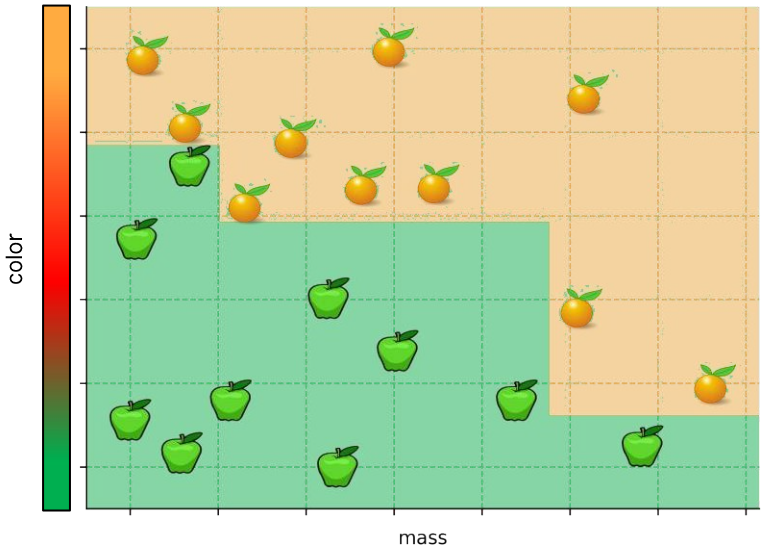
Decision Boundaries

- Decision trees produce non-linear decision boundaries

Logistec Regression



Decision Tree



Logistic Regression vs Decision Tree

Aspect	Logistic Regression (LR)	Decision Tree (DT)
Relationship with Data	Assumes a linear relationship between features and outcome	Handles both linear and non-linear relationships
Categorical Data Handling	Requires Encoding (e.g., Yes/No → 0/1)	Natively handles categorical variables without encoding, making preprocessing easier
Interpretability	Clear mathematically (coefficients = feature weights)	Clear visually (decision paths in a tree)
Stability	More stable with small changes in data	Can change significantly with small data variations
Performance on Large & Complex Data	Efficient and fast, but **struggles with complex/non-linear** patterns unless features engineered	Excels in capturing **complex, high-dimensional, non-linear patterns**, automatically handling feature interactions and variable selection

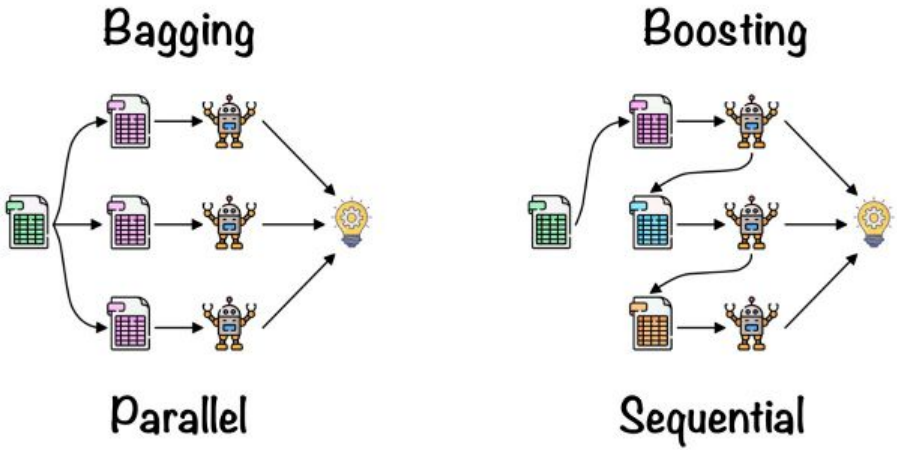
https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial

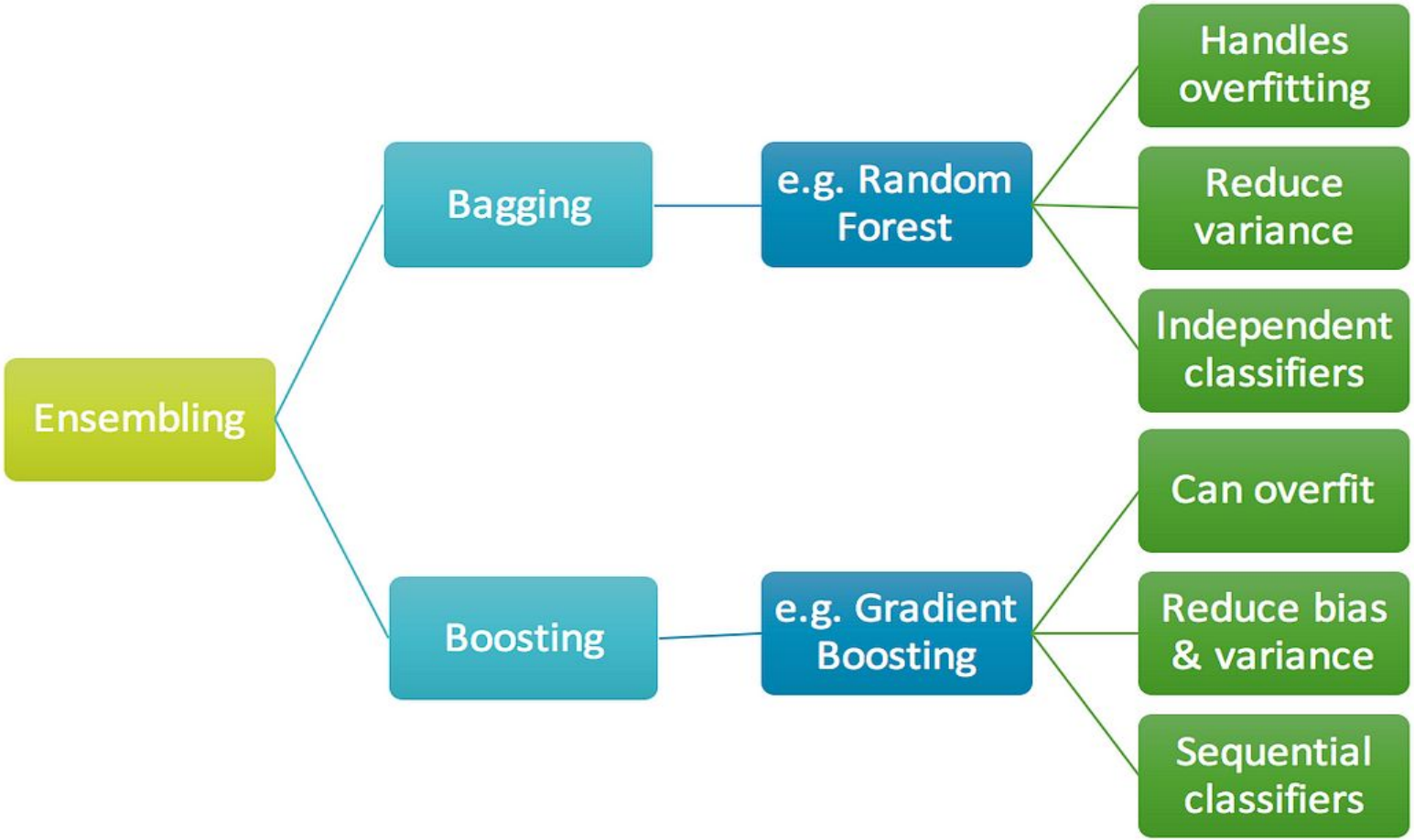
Ensemble learning

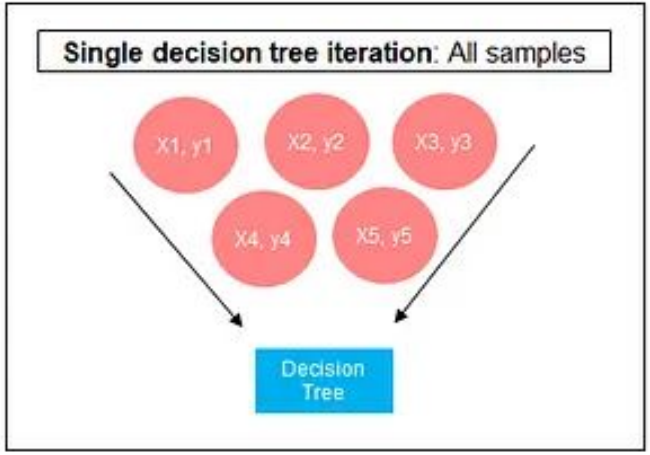
Random Forest

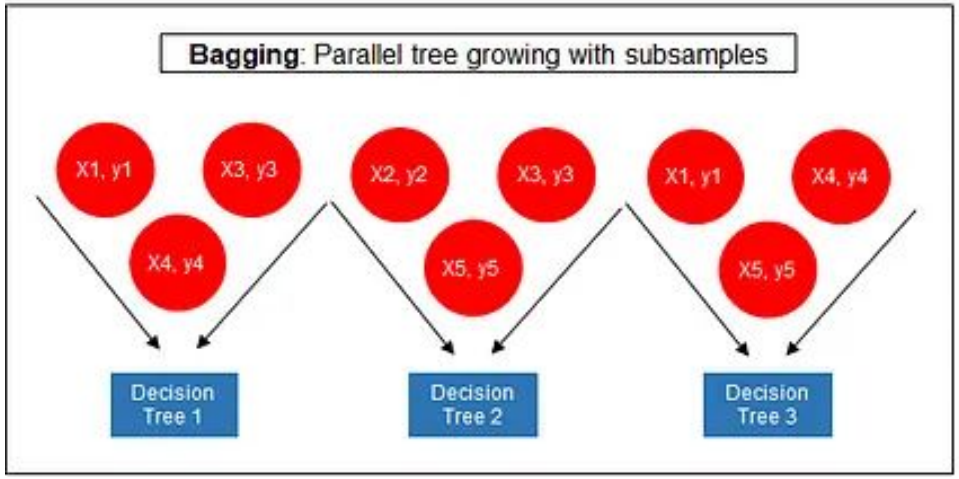
Ensemble learning

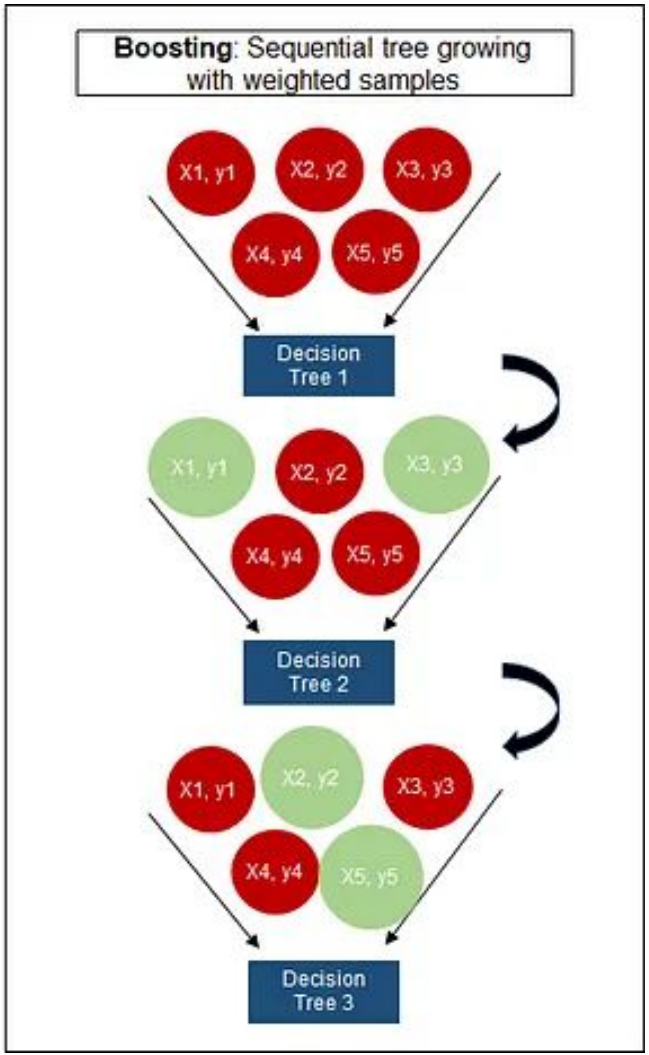
- Ensemble learning is a machine learning paradigm where multiple models (learners) are trained to solve the same problem.
- By using multiple learners, generalization ability of an ensemble can be much better than single learner.

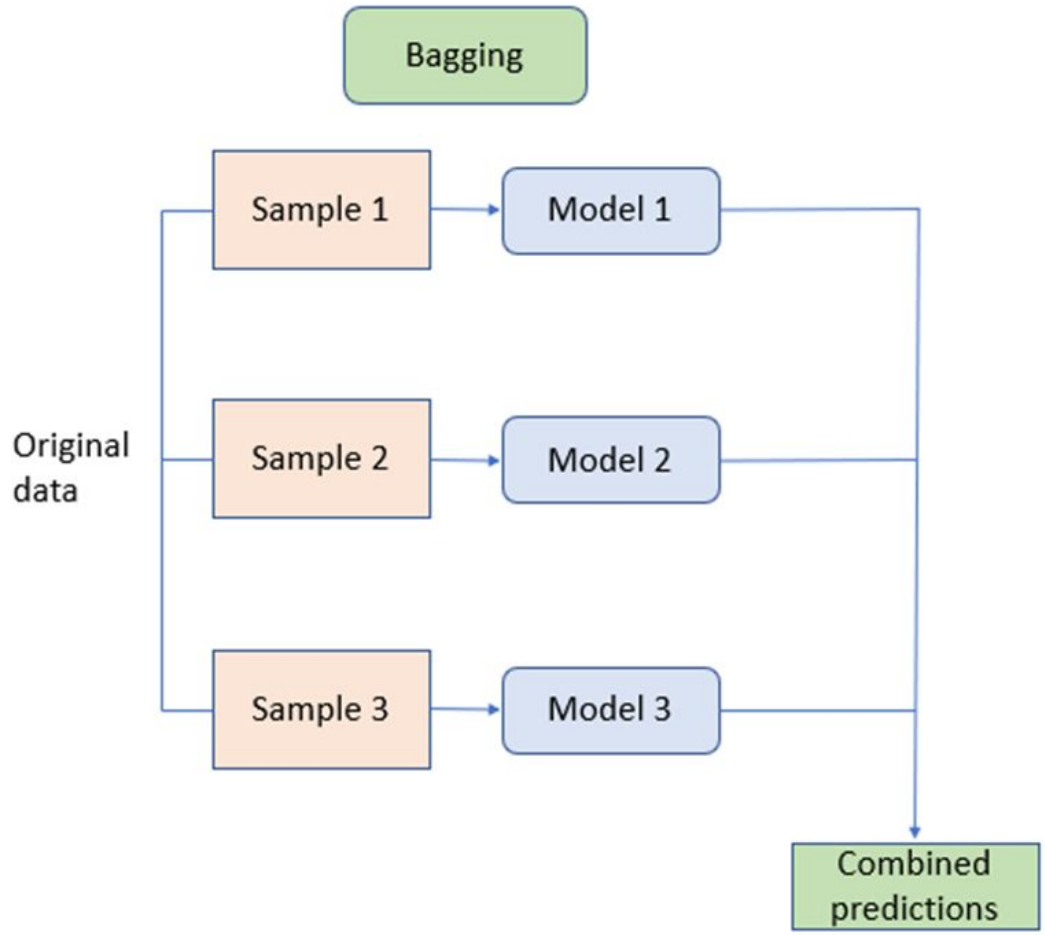



Ensembling
Bagging
e.g. Random Forest
Handles overfitting
Reduce variance
Independent classifiers
Boosting
e.g. Gradient Boosting
Can overfit
Reduce bias & variance
Sequential classifiers

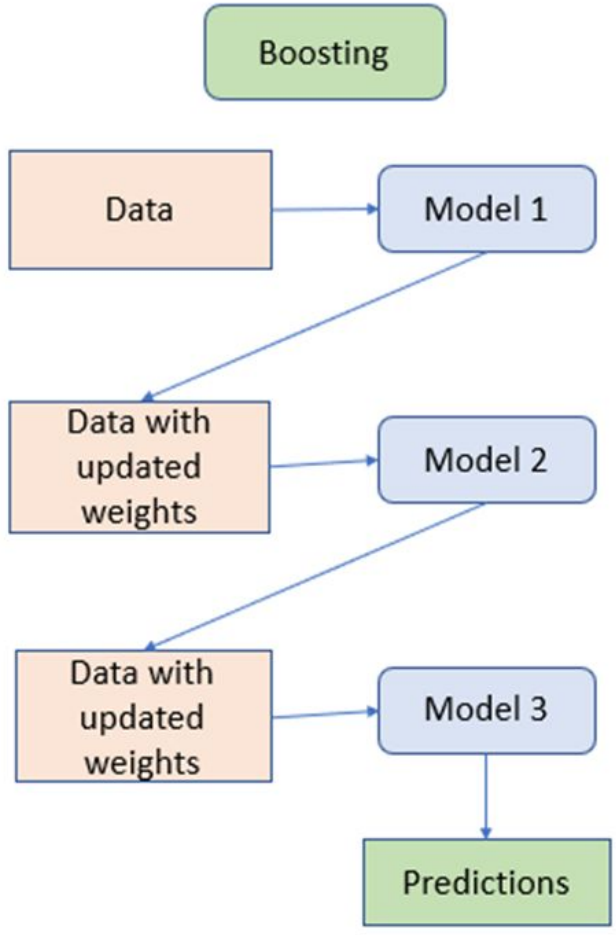

Single decision tree iteration: All samples
X1, y1
X2, y2
X3, y3
X4, y4
X5, y5
Decision Tree


Bagging: Parallel tree growing with subsamples
X1, y1
X3, y3
X4, y4
X2, y2
X3, y3
X5, y5
X1, y1
X4, y4
X5, y5
Decision Tree 1
Decision Tree 2
Decision Tree 3


Boosting: Sequential tree growing with weighted samples
X1, y1
X2, y2
X3, y3
X4, y4
X5, y5
Decision Tree 1
X1, y1
X2, y2
X3, y3
X4, y4
X5, y5
Decision Tree 2
X1, y1
X2, y2
X3, y3
X4, y4
X5, y5
Decision Tree 3



★ Bagging is used mainly in reducing variance and it is a parallel process

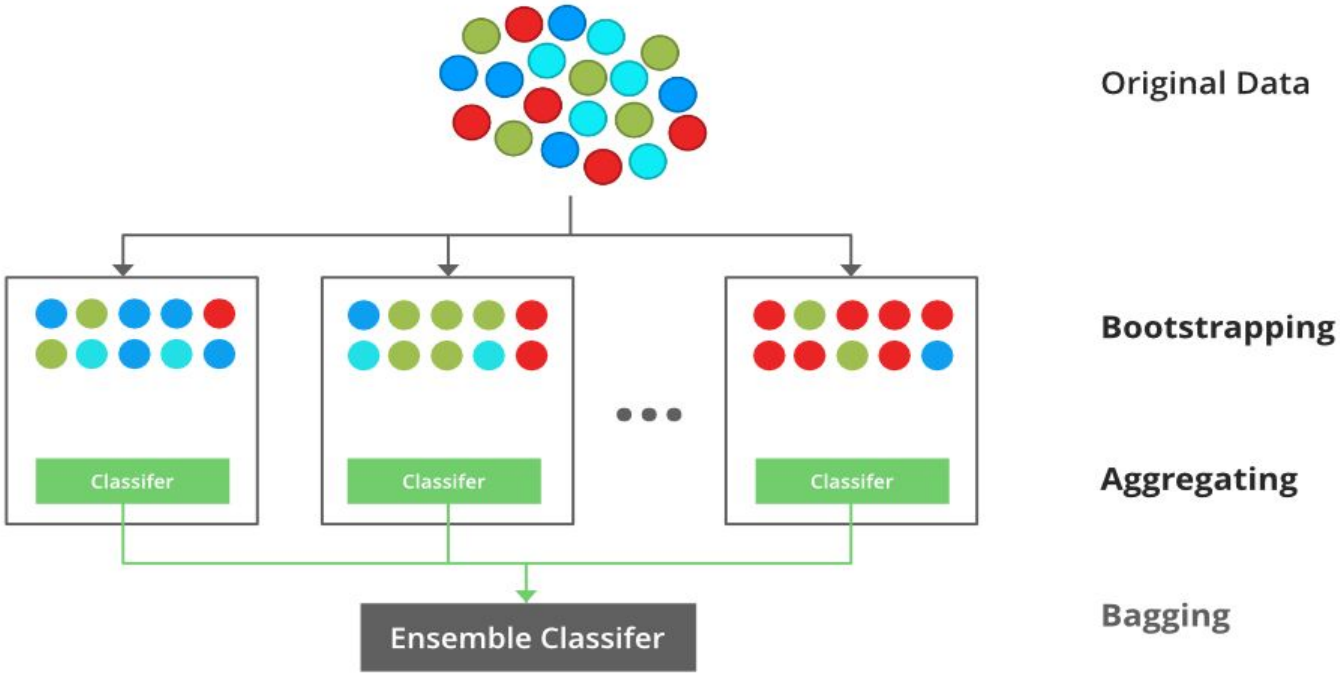


★ Boosting is used mainly in reducing bias and it is a sequential process

Ensemble Learning

- **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the **variance** of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability.
- **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
- **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

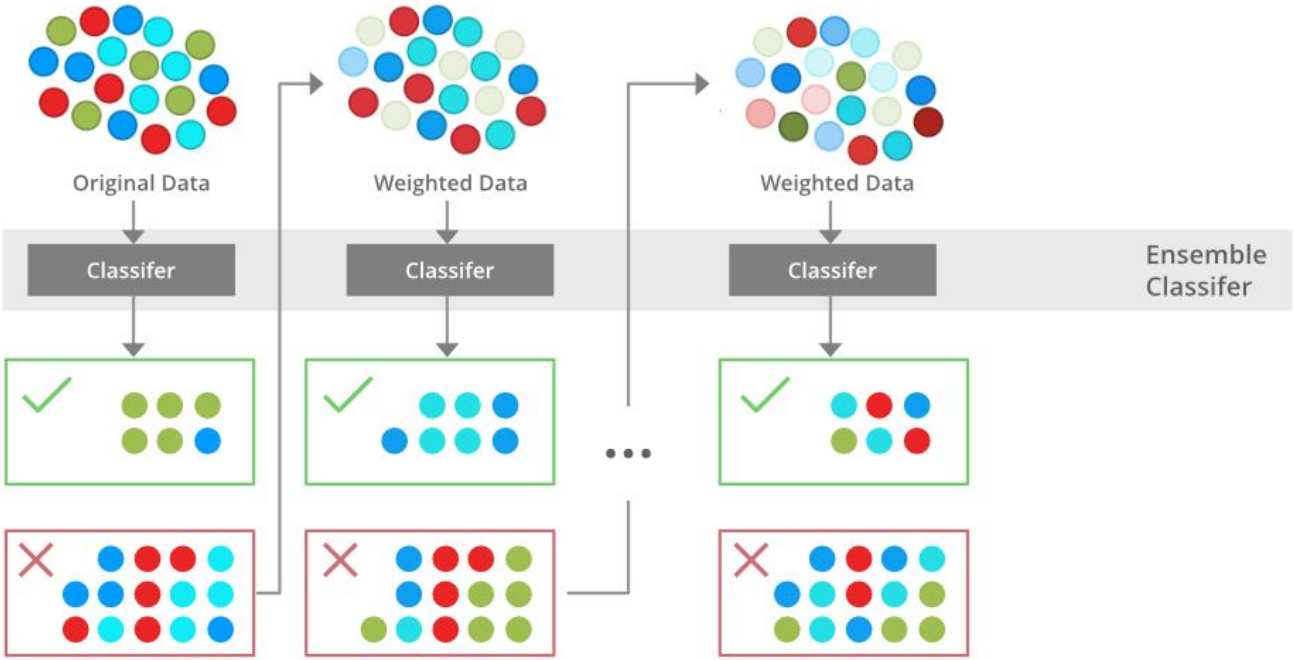
Bagging **in Machine Learning**



Bagging **in Machine Learning**

- **B**ootstrap **A**ggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.
- It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods.
- Bagging is a special case of the model averaging approach.

Boosting in Machine Learning



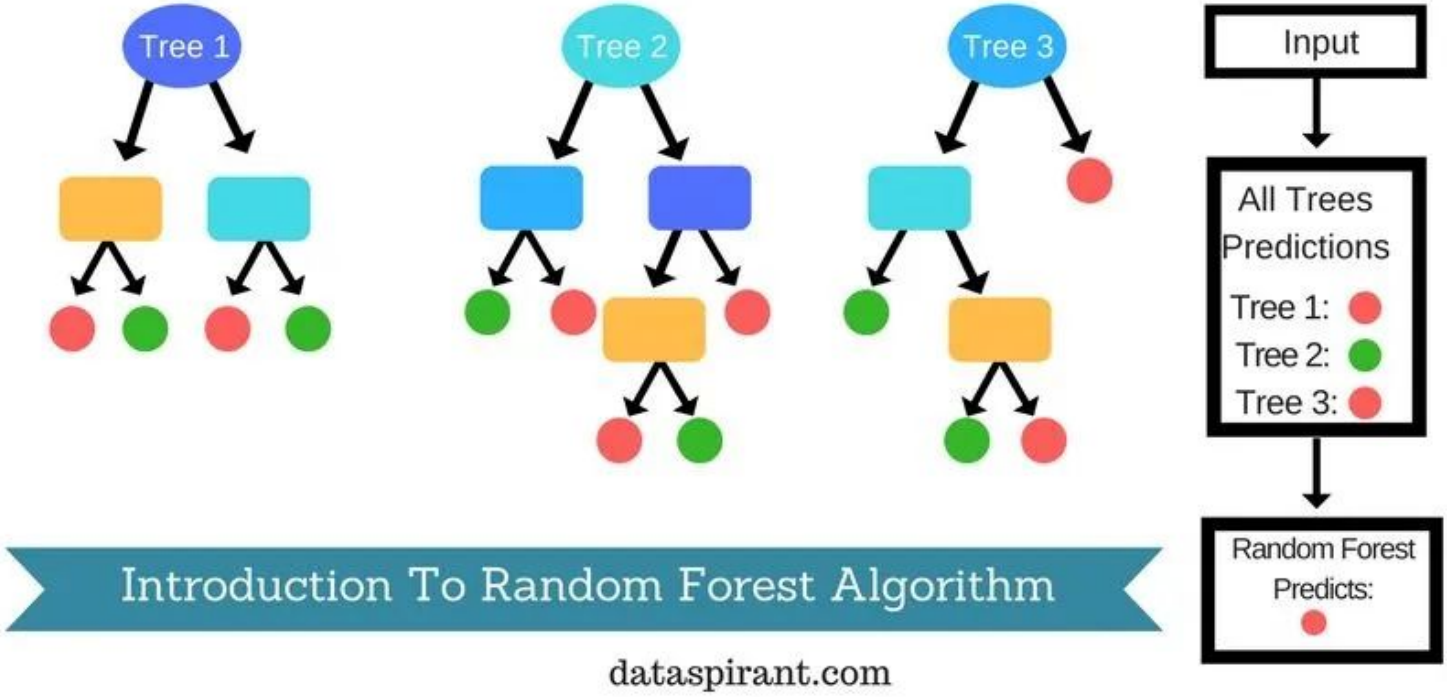
Boosting

- Boosting is the **ensemble learning method** where we build multiple weak learners (same algorithms) in a **SEQUENTIAL** manner.
- All these **weak learners** take the previous models' feedback to improve their power in accurately predicting the missclassified classes.
- Boosting algorithms are one of the best-performing algorithms among all the other Machine Learning algorithms with the best performance and higher accuracies.
- All the boosting algorithms work on the basis of learning from the errors of the previous model trained and tried avoiding the same mistakes made by the previously trained weak learning algorithm.

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Random Forest

Random Forest



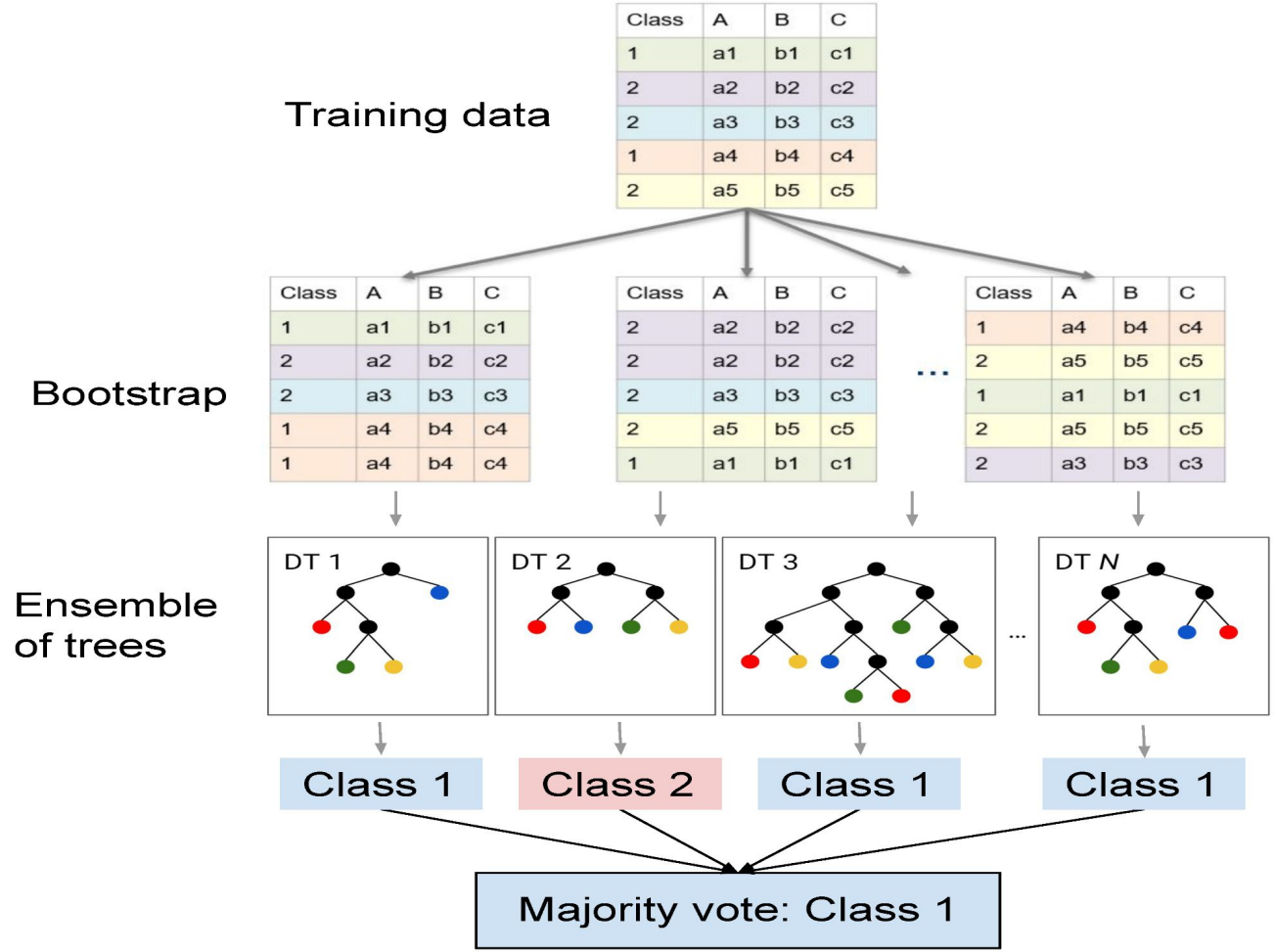
Random Forest

- The random forest algorithm is a supervised classification algorithm.
- As the name suggests, this algorithm creates the forest with a number of trees.
- In general, the **more trees in the forest** the more robust the forest looks like.
- In the same way in the random forest classifier, the **higher the number** of trees in the forest gives **the high the accuracy** results.
- In comparison, the random forest algorithm randomly selects observations and features to build several decision trees and then averages the results.

Why Random forest algorithm

- The same **random forest algorithm** or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will **handle the missing** values.
- When we have **more trees** in the forest, a random forest classifier won't **overfit** the model.
- The random forest algorithm can be used for feature engineering.
  - This means identifying the most important features out of the available features from the training dataset.

Bagging


Training data
Class A B C
1 a1 b1 c1
2 a2 b2 c2
2 a3 b3 c3
1 a4 b4 c4
2 a5 b5 c5
Bootstrap
Class A B C
1 a1 b1 c1
2 a2 b2 c2
2 a3 b3 c3
1 a4 b4 c4
1 a4 b4 c4
Class A B C
2 a2 b2 c2
2 a2 b2 c2
2 a3 b3 c3
2 a5 b5 c5
1 a1 b1 c1
...
Class A B C
1 a4 b4 c4
2 a5 b5 c5
1 a1 b1 c1
2 a5 b5 c5
2 a3 b3 c3
Ensemble of trees
DT 1
DT 2
DT 3
...
DT N
Class 1
Class 2
Class 1
Class 1
Majority vote: Class 1

Dataset

Randomly Selected Data 1

Randomly Selected Data 2

Randomly Selected Data n

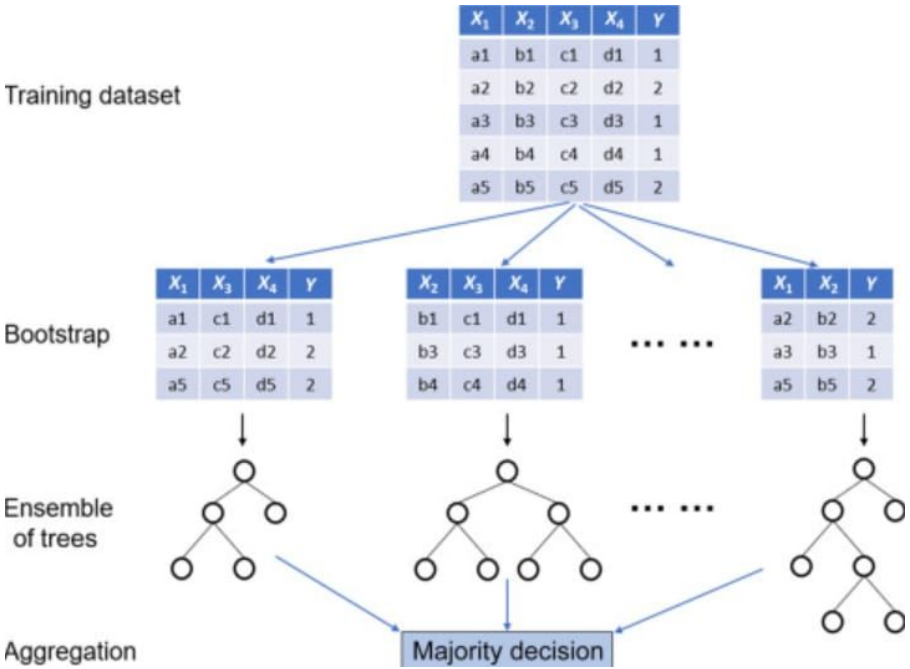
Decision tree 1

Decision tree 2

Decision tree n

Voting

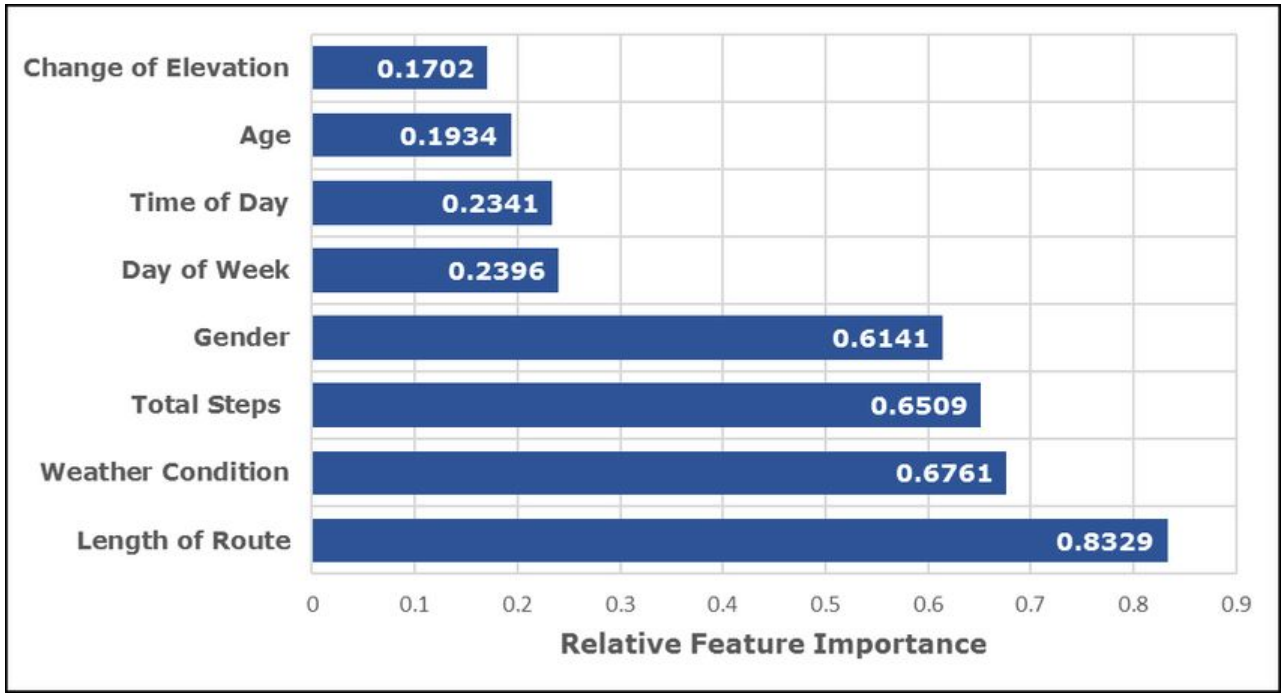
Prediction


Training dataset
X1 X2 X3 X4 Y
a1 b1 c1 d1 1
a2 b2 c2 d2 2
a3 b3 c3 d3 1
a4 b4 c4 d4 1
a5 b5 c5 d5 2
Bootstrap
X1 X3 X4 Y
a1 c1 d1 1
a2 c2 d2 2
a5 c5 d5 2
X2 X3 X4 Y
b1 c1 d1 1
b3 c3 d3 1
b4 c4 d4 1
X1 X2 Y
a2 b2 2
a3 b3 1
a5 b5 2
Ensemble of trees
Aggregation
Majority decision

Random Forest

1. Takes the **test features** and use the rules of each randomly created decision tree to predict the oucome and stores the predicted outcome (target)
2. Calculate the **votes** for each predicted target.
3. Consider the **high voted** predicted target as the **final prediction** from the random forest algorithm.

Feature Importance

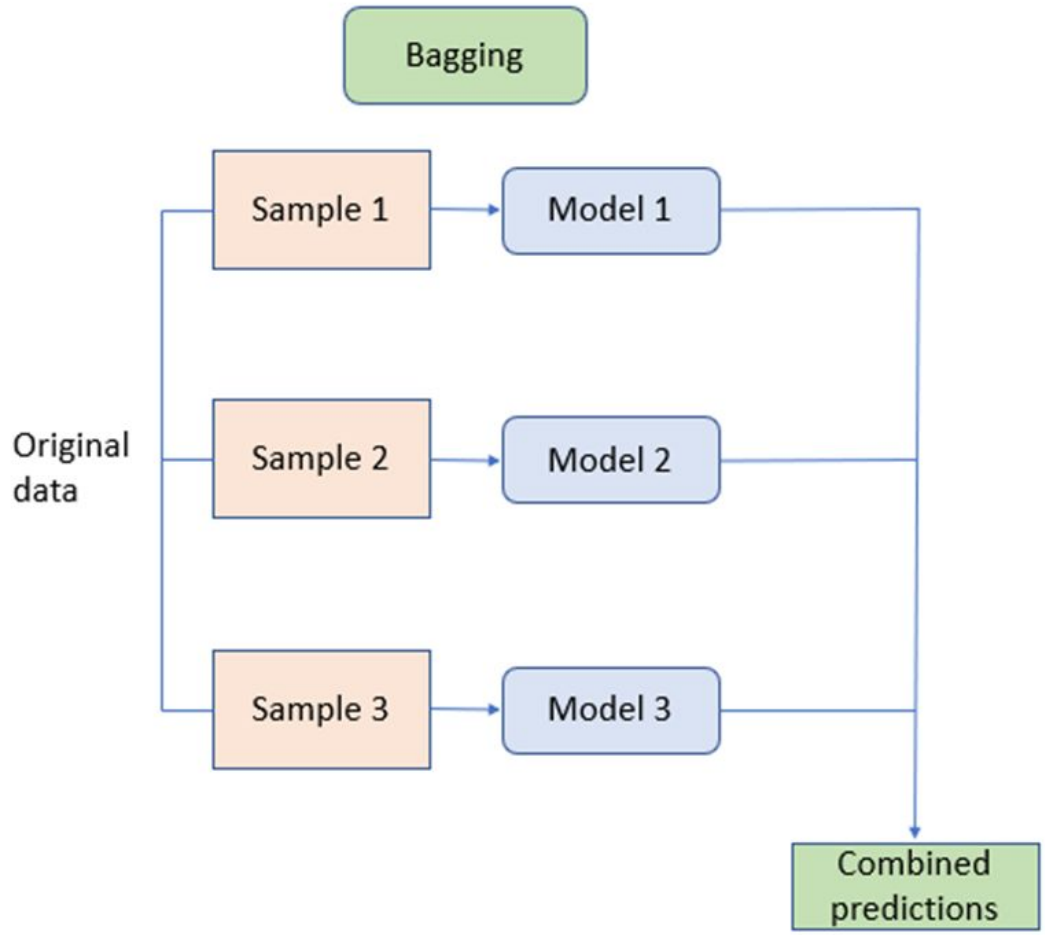


RANDOM FOREST DISADVANTAGES

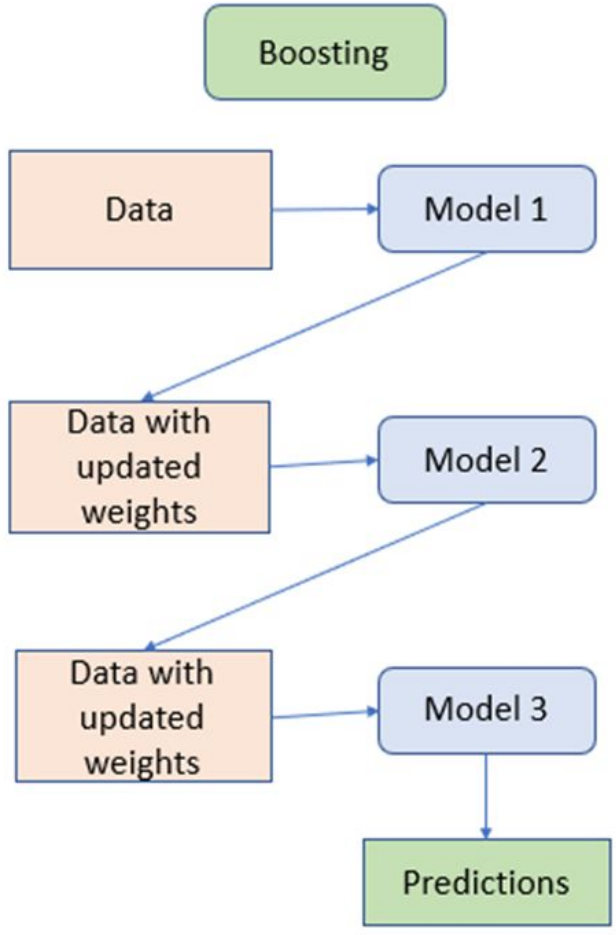
- Increased accuracy requires more trees
- More trees slow down model

Hyperparameter RANDOM FOREST IN Sklearn

- Firstly, there is the **n_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions
- Another important hyperparameter is **max_features,** which is the maximum number of features random forest considers to split a node.
- The last important hyperparameter is **min_sample_leaf.** This determines the minimum number of leafs required to split an internal node.



★ Bagging is used mainly in reducing variance and it is a parallel process

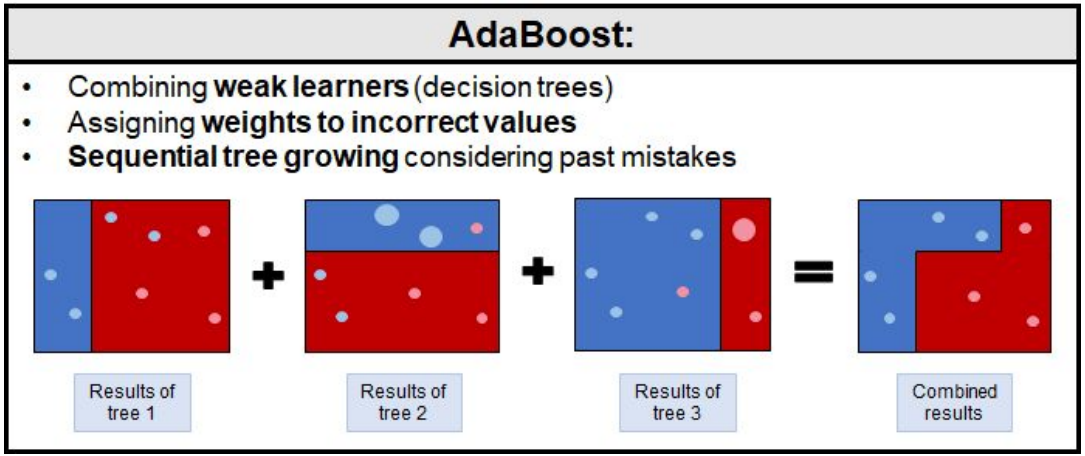


★ Boosting is used mainly in reducing bias and it is a sequential process

Boosting

AdaBoost

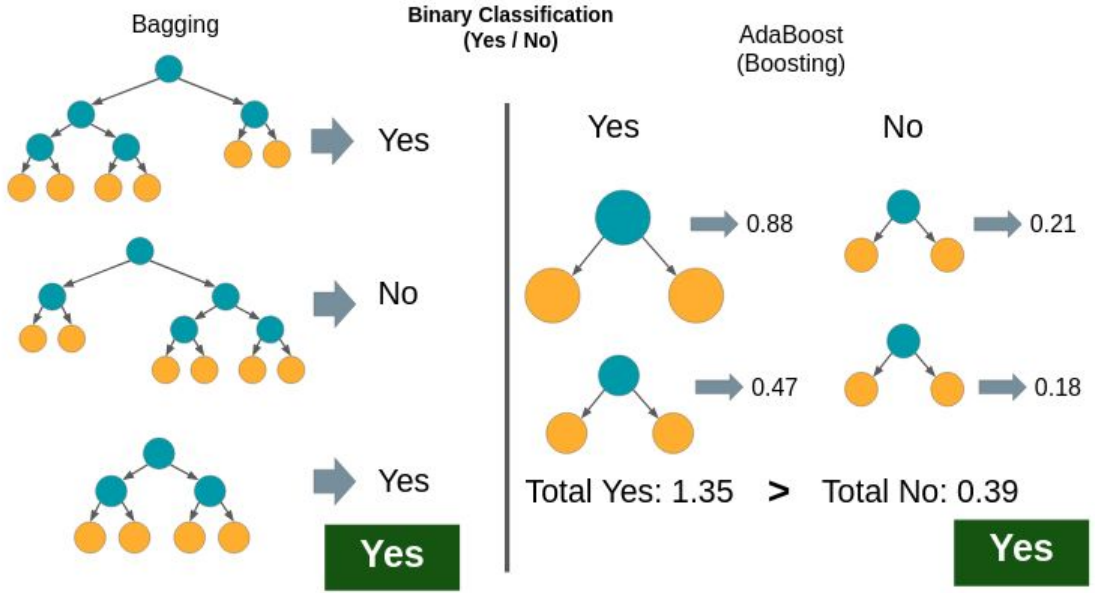
- AdaBoost Adaptive boosting is a boosting algorithm, which also works on the principle of the stagewise addition method where multiple **weak** learners are used for getting **strong** learners.



How AdaBoost Works

- **Step 1:** A weak classifier (e.g. a decision stump) is made on top of the training data based on the weighted samples. Here, the weights of each sample indicate how important it is to be correctly classified. Initially, for the first stump, we give all the samples equal weights.
- **Step 2:** We create a decision stump for each variable and see how well each stump classifies samples to their target classes.
- **Step 3:** More weight is assigned to the incorrectly classified samples so that they're classified correctly in the next decision stump. Weight is also assigned to each classifier based on the accuracy of the classifier, which means high accuracy = high weight!
- **Step 4:** Reiterate from Step 2 until all the data points have been correctly classified, **the error function does not change**, or the **maximum iteration level** has been reached.

How AdaBoost Works



An Example of How AdaBoost Works

- Suppose we have the sample data below, with three features (x1, x2, x3) and an output (Y). *Note that T = True and F = False.*

x1	x2	x3	Y
T	T	F	T
T	T	F	F
T	F	F	T
T	T	T	F
F	T	F	F
T	F	F	T

An Example of How AdaBoost Works

- **Step 1: Assign a sample weight for each sample**

$$sample\ weight = \frac{1}{\#\ of\ samples}$$

x1	x2	x3	Y	Sample Weight
T	T	F	T	1/6
T	T	F	F	1/6
T	F	F	T	1/6
T	T	T	F	1/6
F	T	F	F	1/6
T	F	F	T	1/6

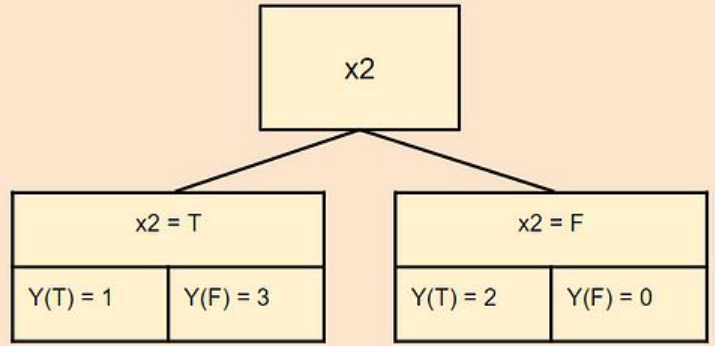
An Example of How AdaBoost Works

- **Step 2: Calculate the Gini Impurity for each variable**

This is done to determine which variable to use to create the first stump.

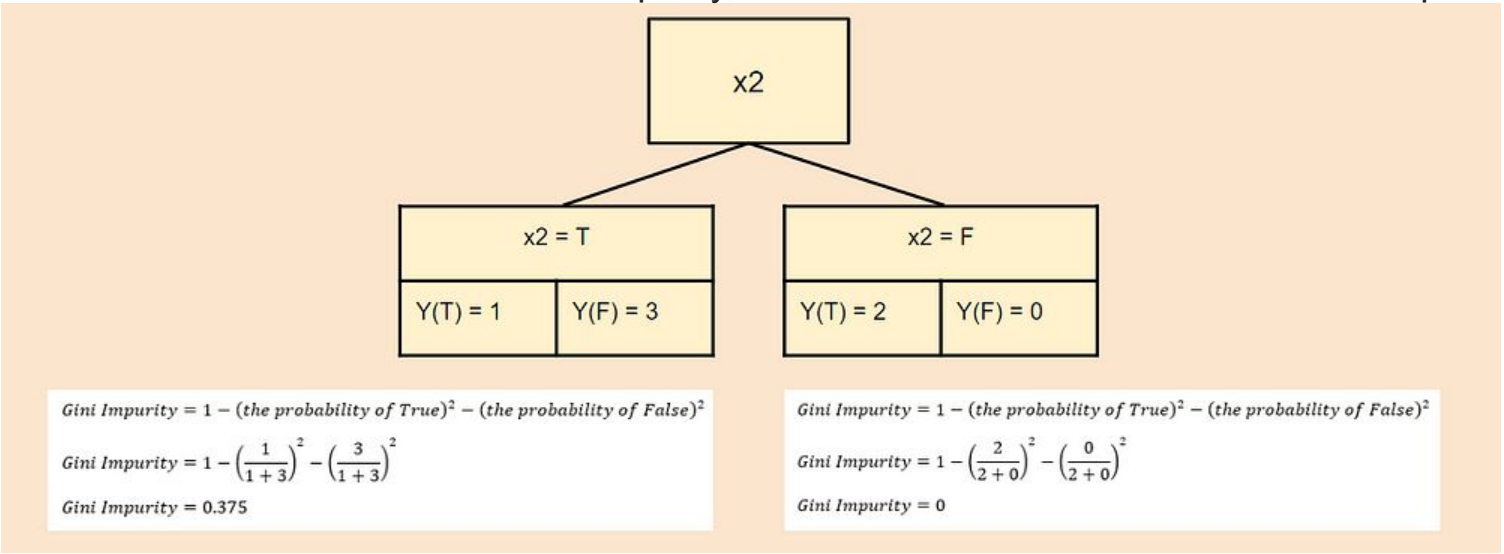
$$Gini\ Impurity = 1 - (the\ probability\ of\ True)^2 - (the\ probability\ of\ False)^2$$

x2	Y = T	Y = F
X2 = T	1	3
X2 = F	2	0



An Example of How AdaBoost Works

x2 has the lowest Gini Impurity, so x2 will be used to create the first stump.



$$Total\ Impurity = 0.375\left(\frac{4}{4+2}\right) + 0\left(\frac{2}{4+2}\right)$$

$$Total\ Impurity = 0.25$$

the Gini Impurity for x2 = 0.25.

An Example of How AdaBoost Works

- **Step 3: Calculate the Amount of Say for the stump that was created**

Total Error is equal to the sum of the weights of the incorrectly classified samples.

Since one of the samples was incorrectly classified for x2, the total error is equal to 1/6.

$$Amount\ of\ say = \frac{1}{2}\log\left(\frac{1 - total\ error}{total\ error}\right)$$

$$Amount\ of\ say = \frac{1}{2}\log\left(\frac{1 - \frac{1}{6}}{\frac{1}{6}}\right) = 0.35$$

An Example of How AdaBoost Works

- **Step 4: Calculate the new sample weights for the next stump**

we're going to increase the sample weights of samples that were incorrectly classified and decrease the sample weights of samples that were correctly classified using the following equations:

$$\textit{New Sample Weight For } \textbf{\textit{Incorrect}} \textit{ Samples} = \textit{Sample weight} * e^{\textit{amount of say}}$$

$$\textit{New Sample Weight For } \textbf{\textit{Correct}} \textit{ Samples} = \textit{Sample weight} * e^{-\textit{amount of say}}$$

An Example of How AdaBoost Works

Need to divide each weight by 0.84

x1	x2	x3	Y	New Sample Weight	Normalized Sample Weights
T	T	F	T	0.24	0.29
T	T	F	F	0.12	0.14
T	F	F	T	0.12	0.14
T	T	T	F	0.12	0.14
F	T	F	F	0.12	0.14
T	F	F	T	0.12	0.14

Total: 0.84

An Example of How AdaBoost Works

- Step 5: Create a **bootstrapped dataset** with the odds of each sample being chosen based on their new sample weights.

x1	x2	x3	Y	Normalized Sample Weights
T	T	F	T	0.29
T	T	F	F	0.14
T	T	F	T	0.29
T	T	T	F	0.14
T	T	F	T	0.29
T	F	F	T	0.14

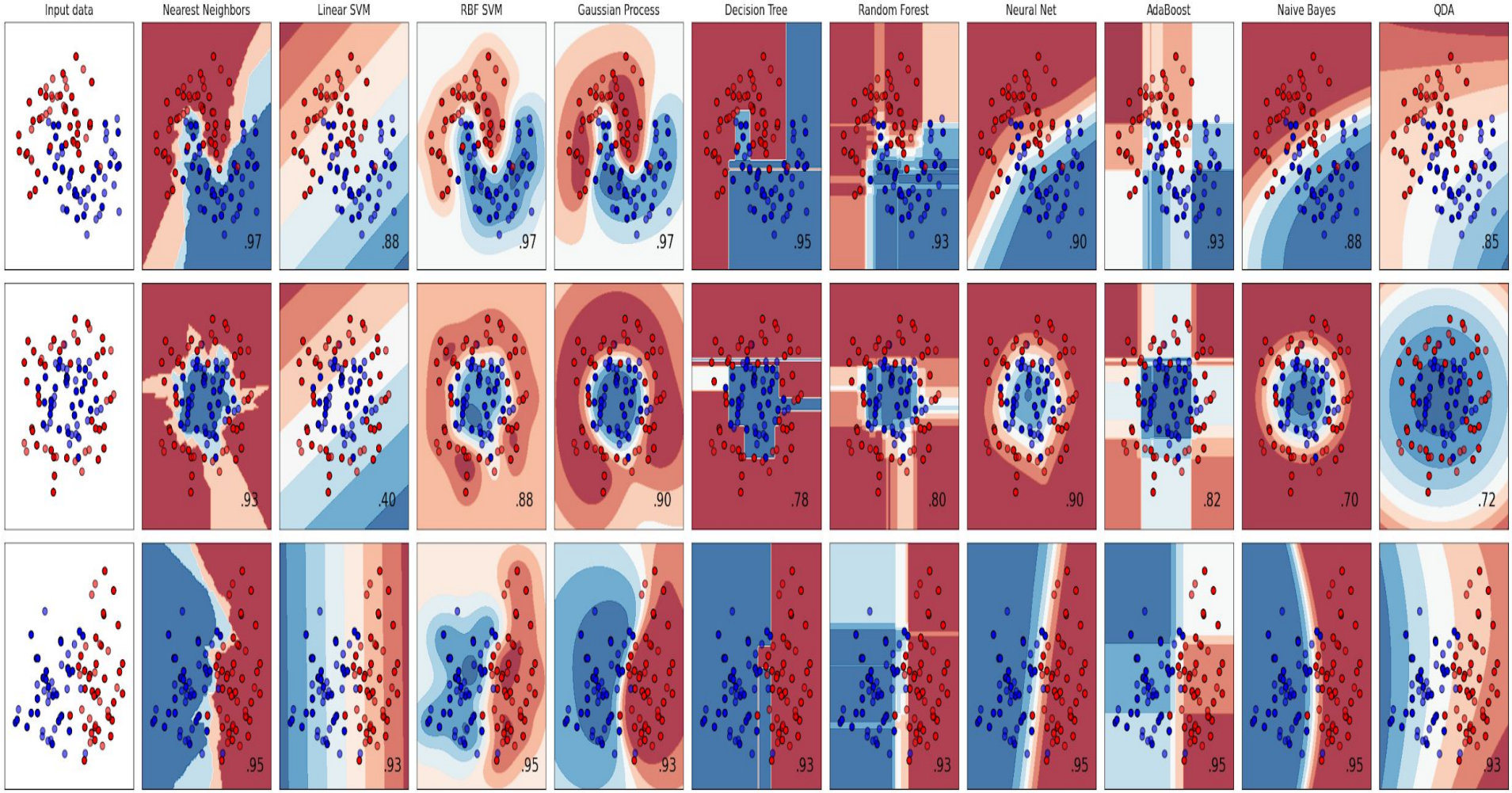
An Example of How AdaBoost Works

- **Step 6: Repeat the process n number of times**
- Lastly, this process is repeated until n number of stumps are created, each with its own amount of say. Once this is done, the model is complete and new points can be classified.
- New points are classified by running them through all of the stumps and seeing how they're classified. Then, the amount of say is summed for each class, and the class with the higher amount of say is the classification of the new point.

Difference Between Boosting Algorithms

Algorithms	Gradient Boosting	AdaBoost	XGBoost	CatBoost	LightGBM
Year	–	1995	2014	2017	2017
Handling Categorical Variables	May require preprocessing like one-hot encoding	No	NO	Automatically handles categorical variables	No
Speed/Scalability	Moderate	Fast	Fast	Moderate	Fast
Memory Usage	Moderate	Low	Moderate	High	Low
Regularization	NO	No	Yes	Yes	Yes
Parallel Processing	No	No	Yes	Yes	Yes
GPU Support	No	No	Yes	Yes	Yes
Feature Importance	Available	Available	Available	Available	Available

https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/

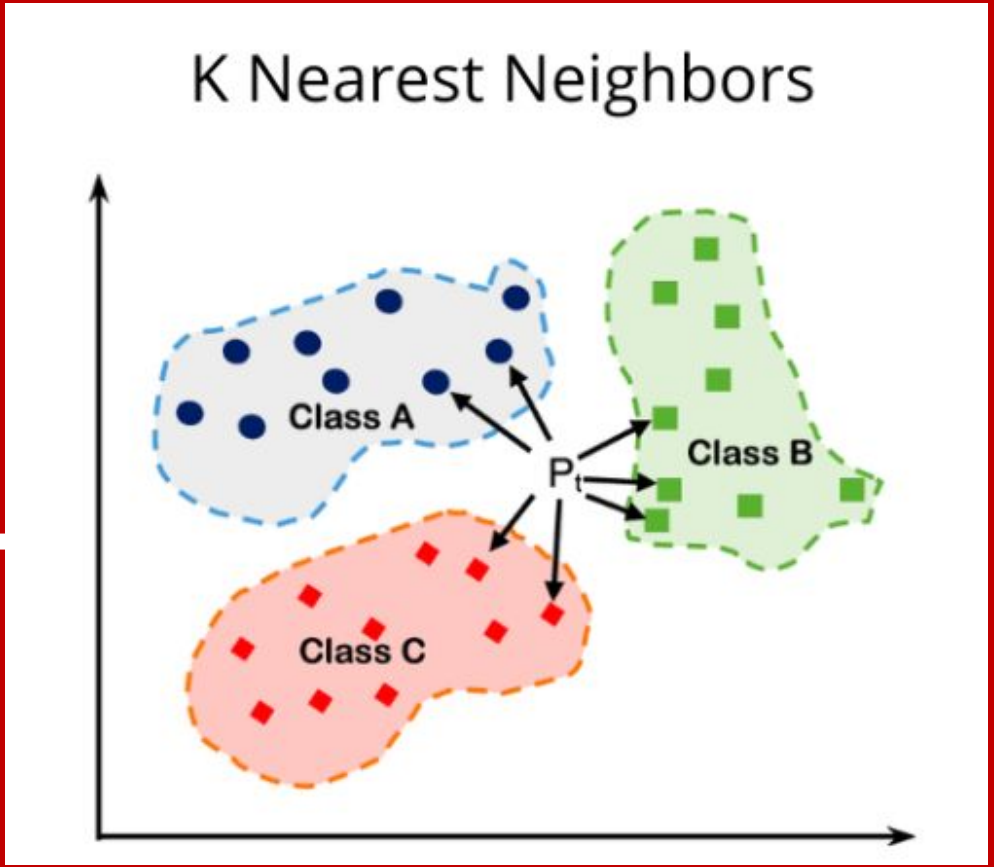


Classifier comparison — scikit-learn 1.3.0

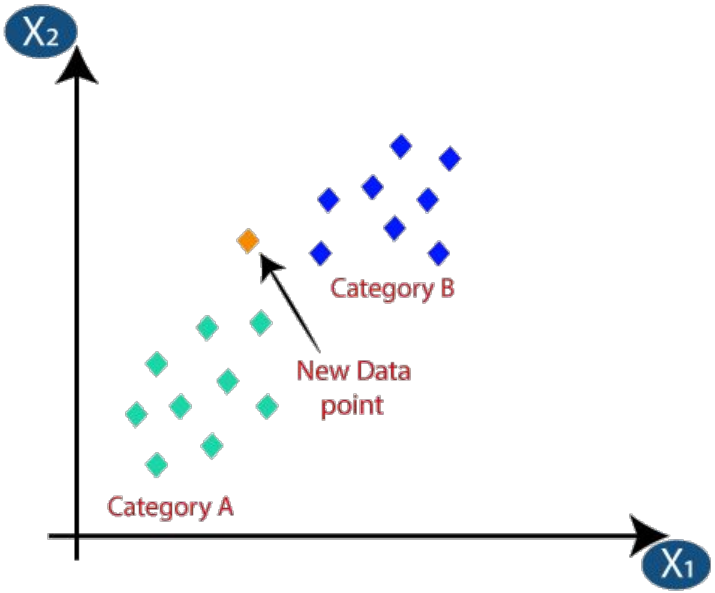
- Decision tree:
  - Split data (nodes, branches , leafs)
  - اكثر عرضة لل **overfiting** > إذا كانت depth deep
  - تغيير بسيط بالداتا ممكن يغير كل ال tree
- Random forest
  - مجموعة من DT بتتدرب على عينات مختلفة من الداتا وبفيتشرز عشوائية والنتيجة بتطلع عن طريق تصويت
  - **decrease variance** > Bagging بالتصويت والتجميع
  - Ensemble learning
  - **هتحل مشكلة ال overfiting**
- Adaboost
  - بناء sequential لل decision stumps بسيط بحيث كل مودل جديد بصحح اخطاء السابق والنتيجة بتتحدد وزن المودلز حسب الاداء
  - الدقة اعلى من ال RF لانها **بتقلل bais** وبتركز على الخطأ لانها بتعطي اهمية واولوية اكثر للغلطان لحتى يتعلم منيح فهاد معناه انو الدقة هتزيد
  - حساسة لل outliers

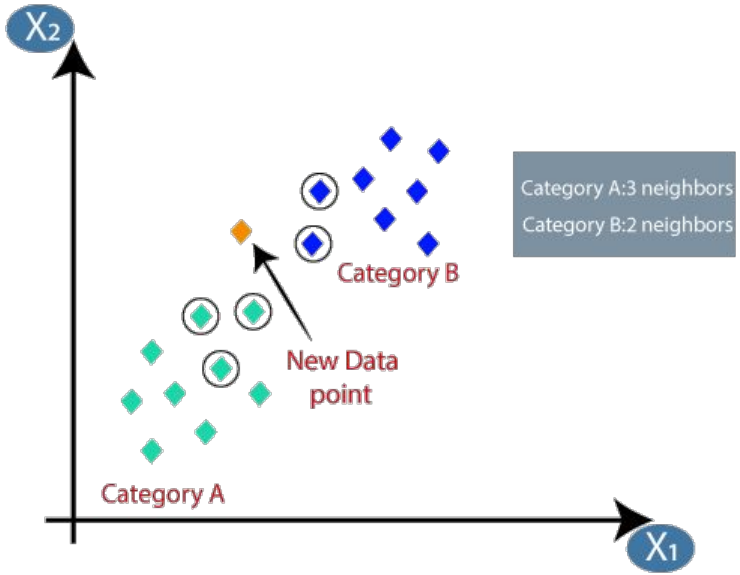
KNN

- قل لي من هم أقرب جيرانك، وسأخبرك من أنت



KNN - lazy learner algorithm





K-Nearest Neighbors (Non-Parametric Approach) classification

- Step 1 − load the training as well as test data.
- Step 2 − Next, we need to choose the value of K

Choosing K:

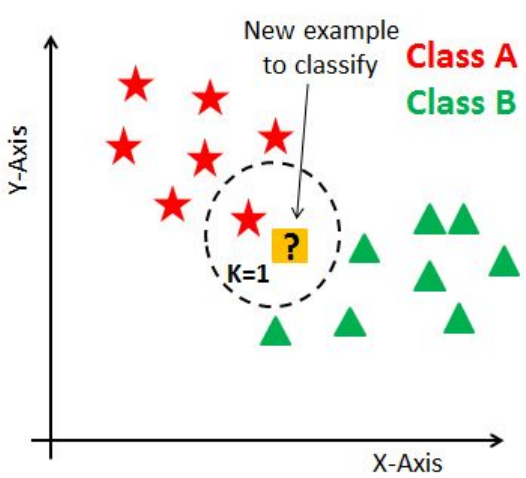
- The value of K is critical; it controls the bias-variance trade-off
- Small K: Low bias, high variance (more flexible, but sensitive to noise)
- Large K: High bias, low variance (smoother, but may overlook local patterns)

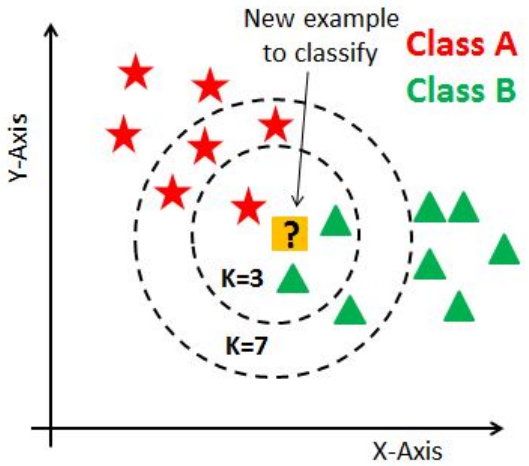
The choice of K affects the bias-variance trade-off.

3.2 − Now, based on the distance value, sort them in ascending order.

- 3.3 − Next, it will choose the top K rows from the sorted array.
- 3.4 − Now, it will assign a class to the test point based on most frequent class of these rows

How do you Decide the Number of Neighbors in KNN?





K-Nearest Neighbors Regression Example

Given Data Points:

X	Y
1	1.5
2	1.7
3	3.1
4	2.8
5	3.6
6	3.9
7	5.1
8	5.3
9	6.7
10	6.9

- **Choose K:** Let's select $K = 3$
- **Prediction Point x_0:** Let's predict the value at $x_0 = 5.5$

Calculate the distance from $x_0 = 5.5$ to all other points.

X	Y	Distance to 5.5
1	1.5	$\|5.5 - 1\| = 4.5$
2	1.7	$\|5.5 - 2\| = 3.5$
3	3.1	$\|5.5 - 3\| = 2.5$
4	2.8	$\|5.5 - 4\| = 1.5$
5	3.6	$\|5.5 - 5\| = 0.5$
6	3.9	$\|5.5 - 6\| = 0.5$
7	5.1	$\|5.5 - 7\| = 1.5$
8	5.3	$\|5.5 - 8\| = 2.5$
9	6.7	$\|5.5 - 9\| = 3.5$
10	6.9	$\|5.5 - 10\| = 4.5$

Identify the K=3 closest points to $x_0 = 5.5$:

X	Y	
5	3.6	$(\text{Distance} = 0.5)$
6	3.9	$(\text{Distance} = 0.5)$
4	2.8	$(\text{Distance} = 1.5)$

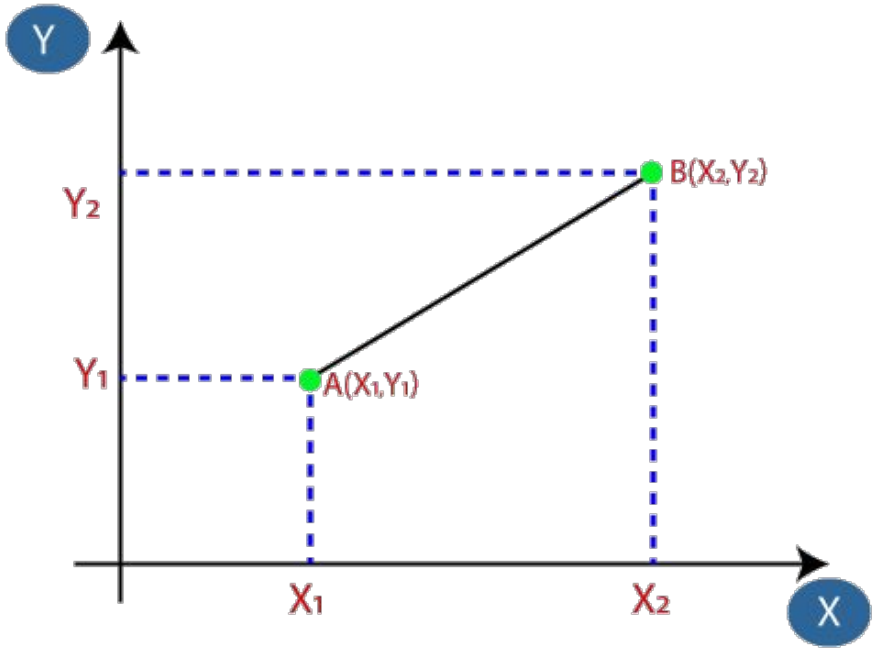
Calculate the average of the responses (Y values) of the K=3 nearest neighbors.

$$\hat{f}(5.5) = \frac{3.6 + 3.9 + 2.8}{3} = \frac{10.3}{3} = 3.43$$

How do you Decide the Number of Neighbors in KNN?

- Defining k can be a balancing act as different values can lead to overfitting or underfitting.
- Lower values of k can have **high variance**, but **low bias** [overfitting].
- larger values of k may lead to **high bias** and **lower variance** [underfitting].
- The choice of **k** will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k.
- Overall, it is recommended to have an odd number for **k** to avoid ties in classification, and cross-validation tactics can help you choose the optimal **k** for your dataset.

Euclidean Distance



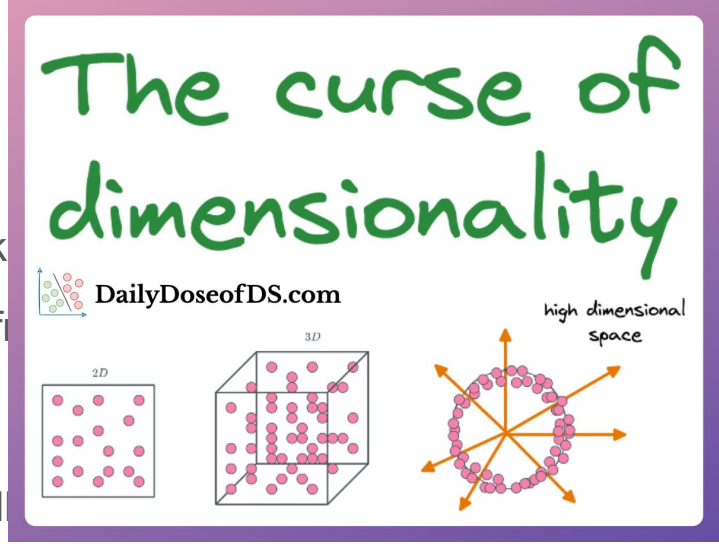
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

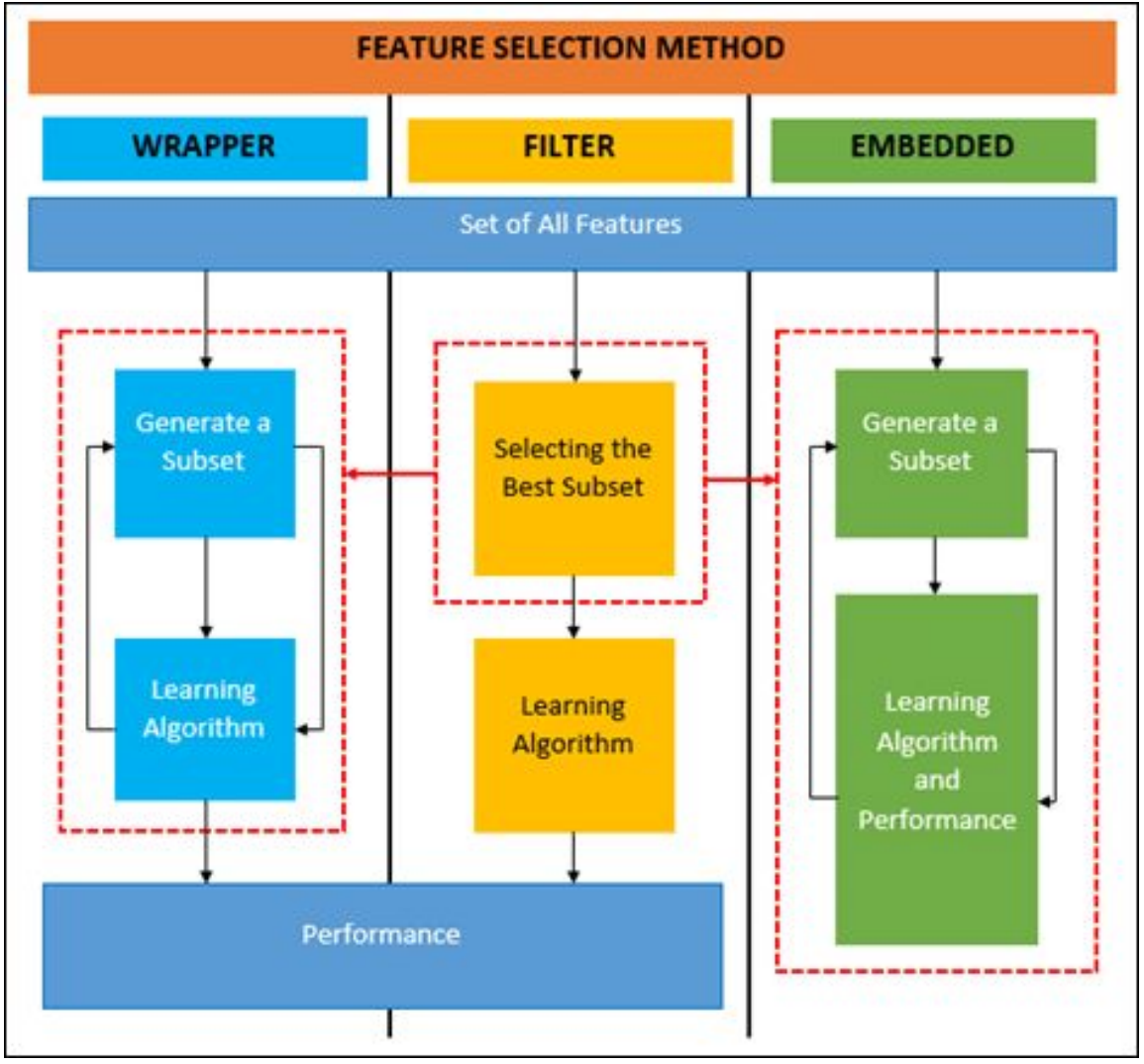
Advantages

- **Easy to implement**
- **Adapts easily**: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.
- **Few hyperparameters**: KNN only requires a k value and a distance metric.

Disadvantages

- **Does not scale well**: Since KNN is a lazy algorithm, it tak
  storage compared to other classifiers. This can be costly f
  perspective.
- **Curse of dimensionality**: The KNN algorithm tends to fal
  dimensionality, which means that it doesn't perform well with high-dimensional data inputs.
- **Prone to overfitting**: Due to the "curse of dimensionality", KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of k can also impact the model's behavior.




FEATURE SELECTION METHOD
WRAPPER
FILTER
EMBEDDED
Set of All Features
Generate a Subset
Learning Algorithm
Selecting the Best Subset
Learning Algorithm
Generate a Subset
Learning Algorithm and Performance
Performance

Dimension reduction (Feature extraction)

	1	2	3	4	5	6	7		200
	Height	Weight	Average blood pressure	Average heart rate	BMI	Cholesterol levels	Average cigarettes/day	...	Sugar levels
Person 1	150	80	140/90	63	36	5.0	0		99
Person 2	174	90	90/60	100	32	4.1	0		95
Person 3	183	109	120/80	95	29	3.6	1		92
Person 4	186	95	123/75	84	28	4.8	5		89
Person 5	170	67	95/60	76	23	2.7	10		100
Person 6	180	82	92/60	78	25	3.7	10		112
Person 7	165	71	124/80	81	26	3.8	0		113
Person 8	172	78	97/70	90	24	3.4	0		100
...									
Person 20	190	75	90/60	78	21	4.2	0		82

200 FACTORS (VARIABLES)

PCA

PC1	PC2	PC3	PC4	PC5
-1	3	-1	4	4
2	4	2	5	5
3	2	4	2	2
4	4	5	-4	-4
5	5	2	2	5
2	5	-4	3	2
-4	-6	5	5	-4
-3	-6	-6	2	5
8	-3	-6	-3	-6

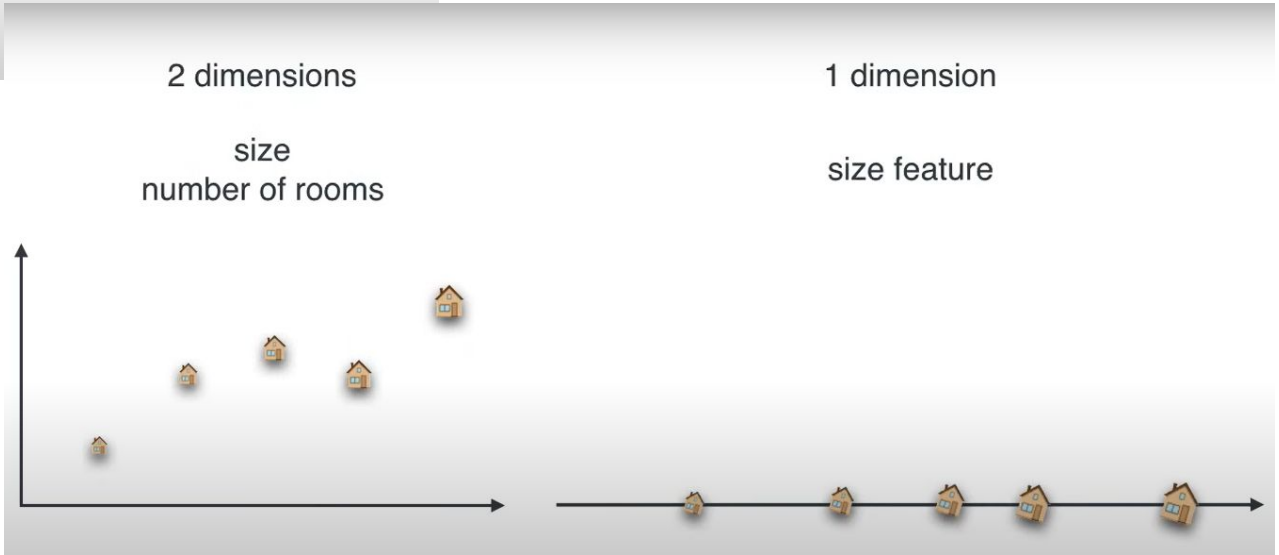
5 PRINCIPAL COMPONENTS

Housing Data

Dimension reduction (Feature extraction)

Size
Number of rooms
Number of bathrooms
→ Size feature

Schools around
Crime rate
→ Location feature



Required Data Preparation for KNN

Computational Complexity:

- Can be computationally expensive, especially with large datasets, because it requires calculating distances to all training points for each prediction

Normalization:

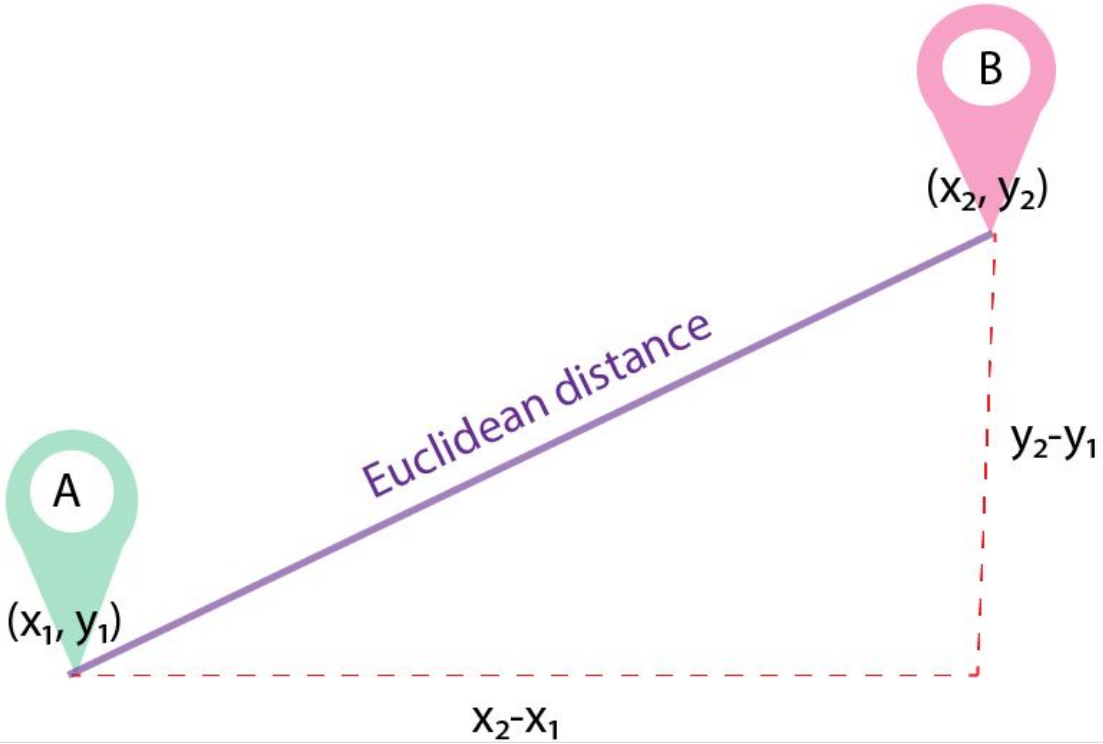
- Feature scaling (normalization) is important to ensure all features contribute equally to the distance calculations

Handling High Dimensionality:

- Performance can degrade in high-dimensional spaces (curse of dimensionality)
- Dimensionality reduction techniques (e.g., PCA) can be helpful

Distance Measures for KNN

K-Nearest Neighbors (KNN) uses distance measures to determine the proximity of data points to one another. Different distance metrics affect how KNN classifies the data points.



K-Nearest Neighbors (KNN)

Distance Measures for KNN

1. Manhattan Distance

Formula $d = \sum_{i=1}^{n} |x_i - y_i|$

Explanation

- Manhattan distance is the sum of the absolute differences of their Cartesian coordinates.
- It can be visualized as "grid-like" distances, where movement is allowed only along the grid lines (no diagonal movement).

K-Nearest Neighbors (KNN)

Distance Measures for KNN

2. Euclidean Distance

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

Explanation

- Euclidean distance measures the straight-line distance between two points in Euclidean space.
- This is the most commonly used distance measure in KNN.

K-Nearest Neighbors (KNN)

Distance Measures for KNN

3. Minkowski Distance

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

Explanation

- Minkowski distance is a generalized distance metric. It can represent both Manhattan and Euclidean distances based on the value of p:
  - When p=1, it represents Manhattan distance.
  - When p=2, it represents Euclidean distance.
- Minkowski distance is useful in vector spaces and can generalize to higher dimensions.

K-Nearest Neighbors (KNN)

Distance Measures for KNN

4. Cosine Distance/Similarity

Formula $\text{Cosine Similarity} = \cos(\theta) = \frac{a \cdot b}{|a||b|}$

Explanation

- Cosine similarity is used to measure the **similarity** between two vectors by calculating the cosine of the angle between them.
    - 1 indicates vectors pointing in the same direction.
    - 0 indicates orthogonal vectors (no similarity).
    - -1 indicates vectors pointing in opposite directions.
- Cosine **distance** is calculated as $\text{Cosine Distance} = 1 - \cos(\theta)$

Usage

- This metric is commonly used in text analysis and document comparison.

K-Nearest Neighbors (KNN)

Distance Measures for KNN

5. Hamming Distance

Formula

- Hamming distance is calculated by counting the number of bit positions in which two binary strings differ.

Explanation

- It's used primarily for comparing two **binary data strings** of equal length.

Example

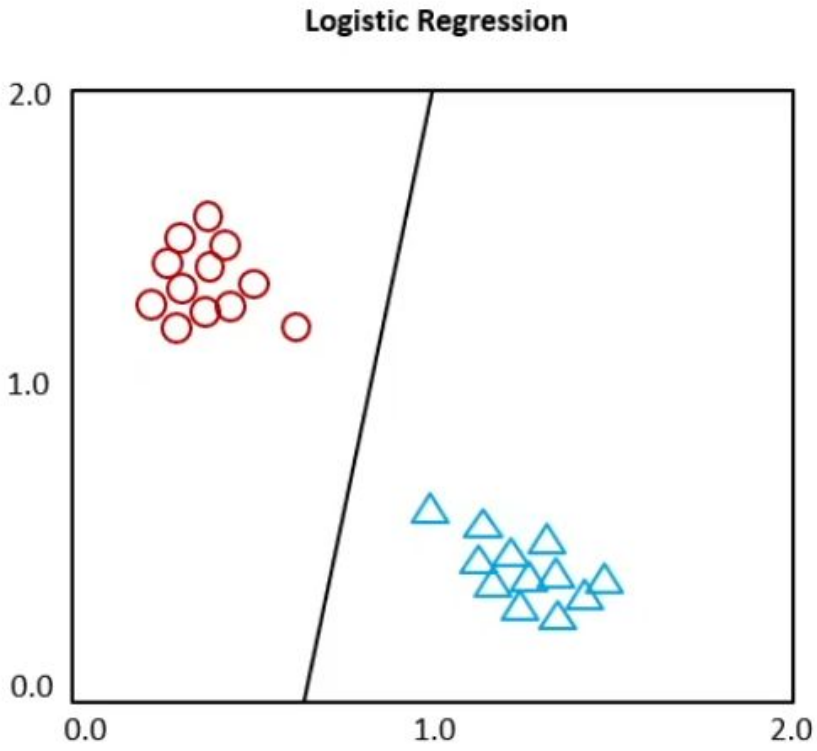
- If the strings are "ABCDE" and "AGDDF," the Hamming distance is 3 because three positions differ.

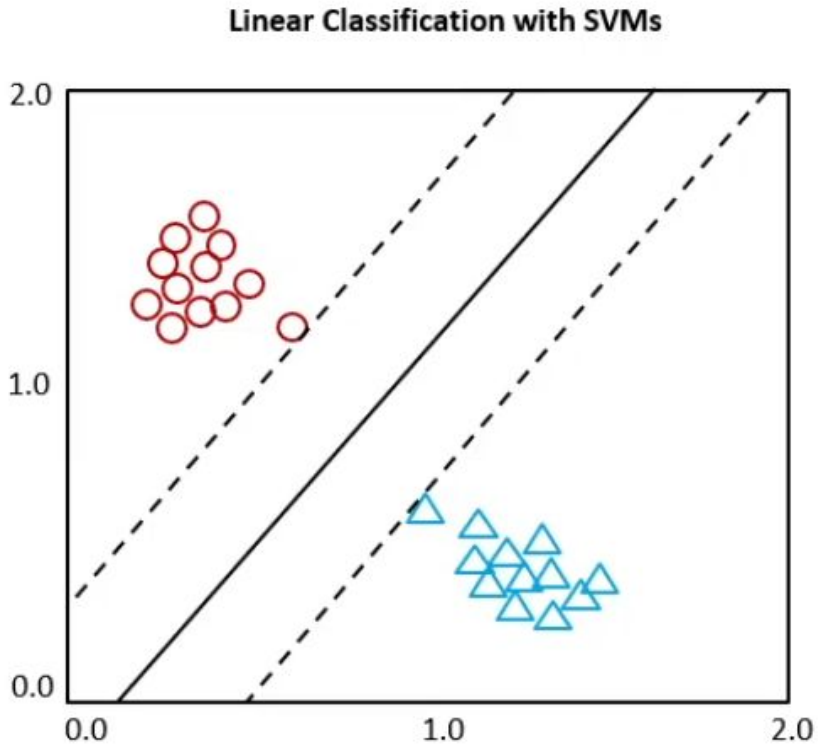
متى افكر استخدم KNN

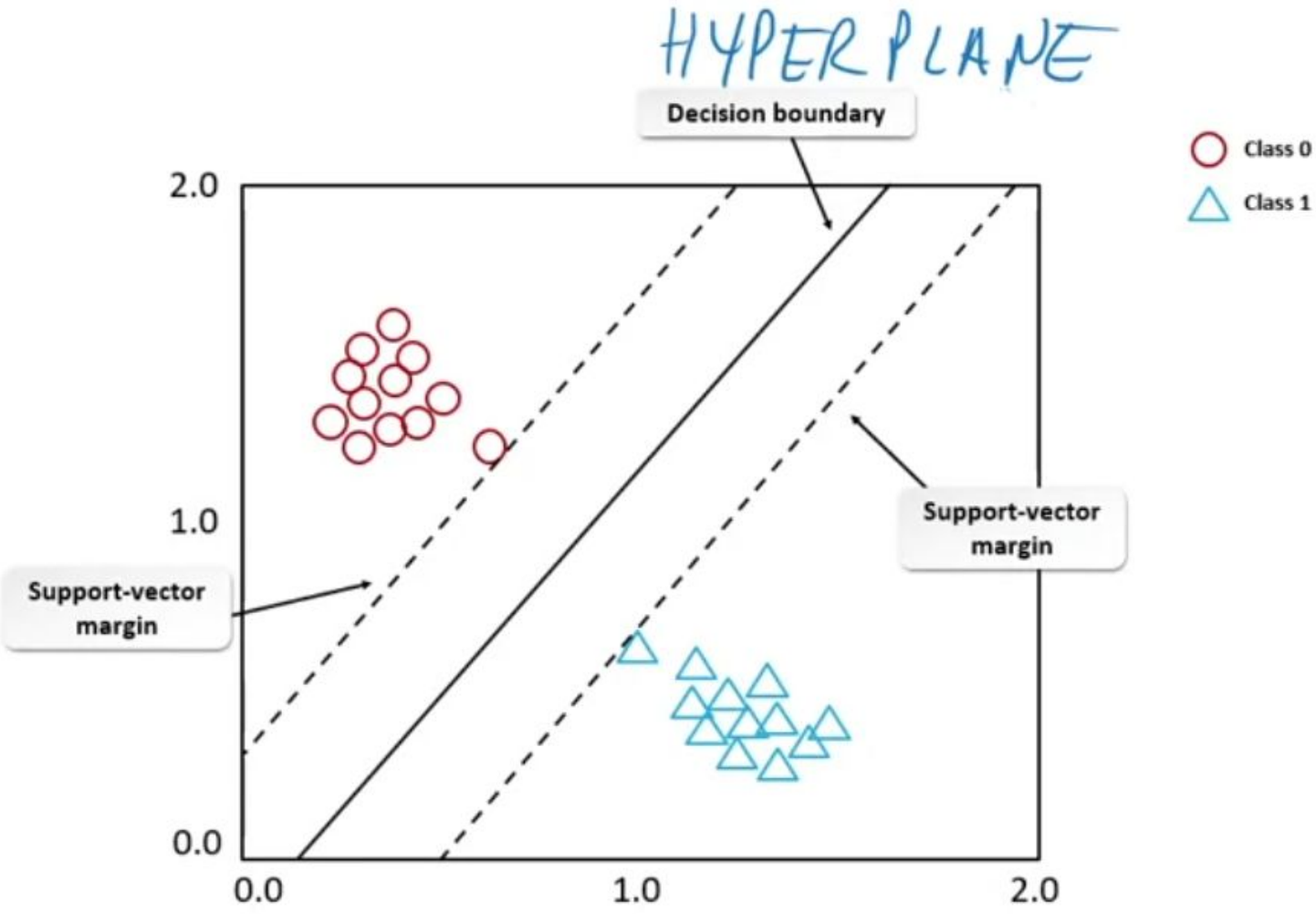
- **حسب نوع المشكلة**
  - Regression or Classifaction
- **طبيعة البيانات**
  - بيانات بسيطة وغير معقدة
  - **no linear**
  - Similar scale
- **اياك ثم اياك تستخدمها مع داتا بابعاد كثير (Curse of dimensionality) يعني عدد الفيتشر يكون كبير (مينفعششششش)**
  - ممكن تحل هاي المشكلة باستخدام الطرق الموجودة بالسلايدات المرفقة
- حساااااااااااس جدا ل outliers, Noise

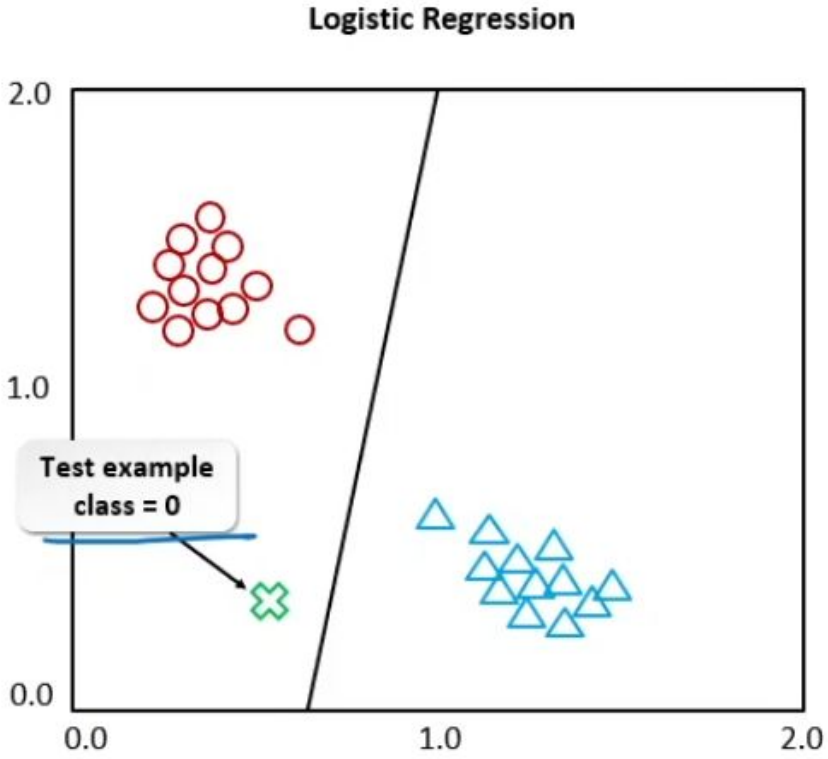
Support Vector Machine

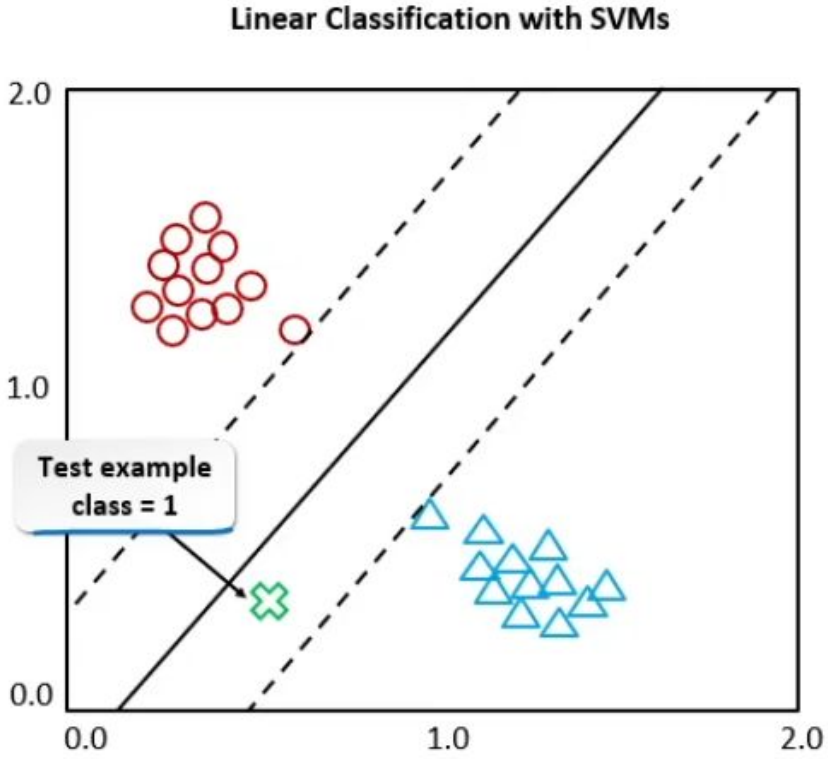
- بتتعامل مع outliers, Noise
- Curse of dimensionality
- Non linear date


Logistic Regression
2.0
1.0
0.0
0.0
1.0
2.0

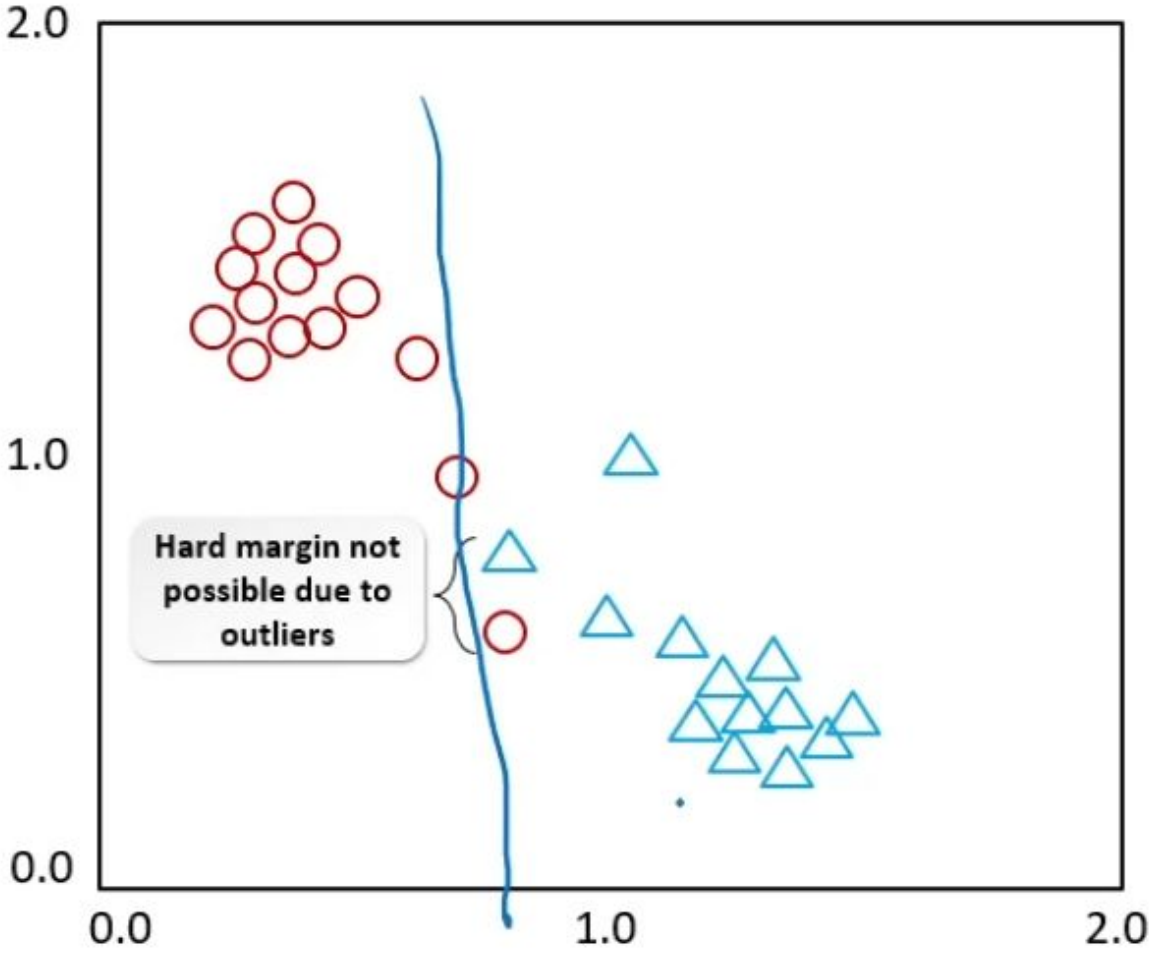

Linear Classification with SVMs
2.0
1.0
0.0
0.0
1.0
2.0


HYPERPLANE
Decision boundary
Class 0
Class 1
2.0
1.0
0.0
0.0
1.0
2.0
Support-vector margin
Support-vector margin

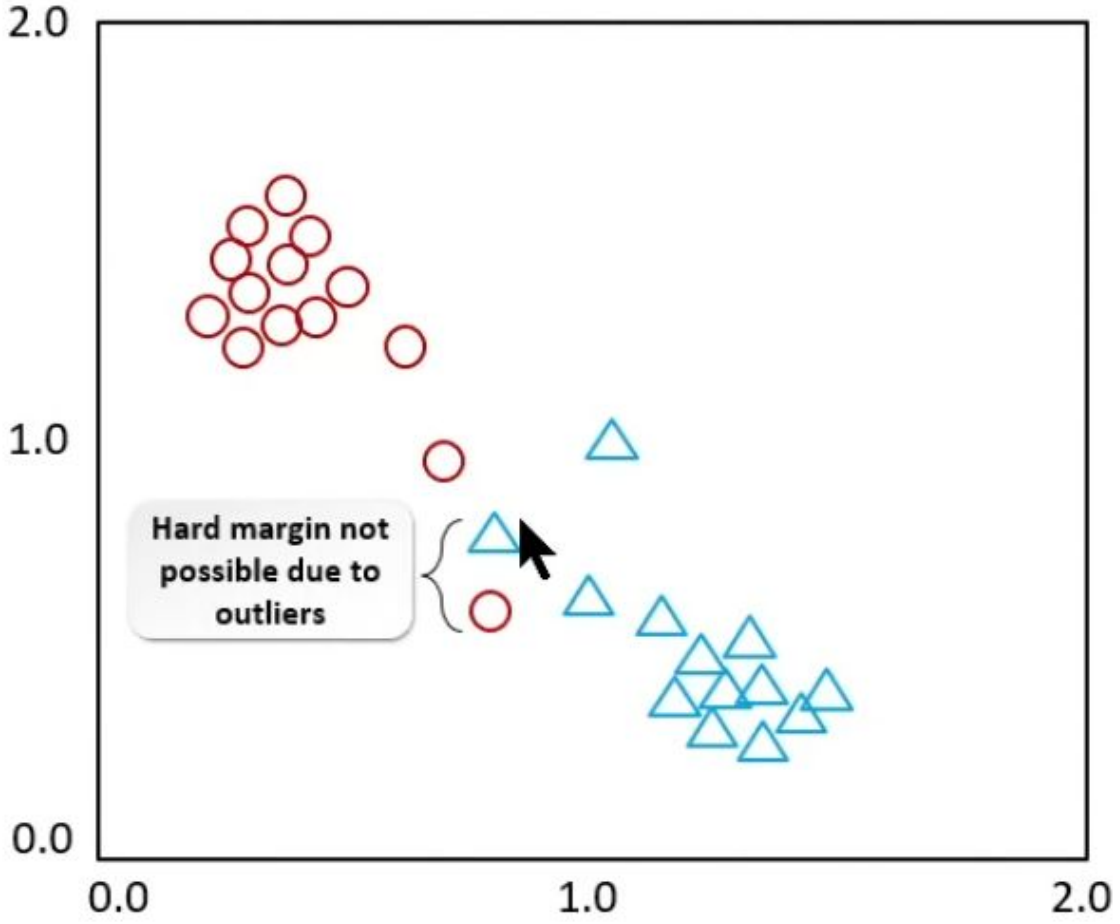

Logistic Regression
2.0
1.0
0.0
0.0
1.0
2.0
Test example
class = 0

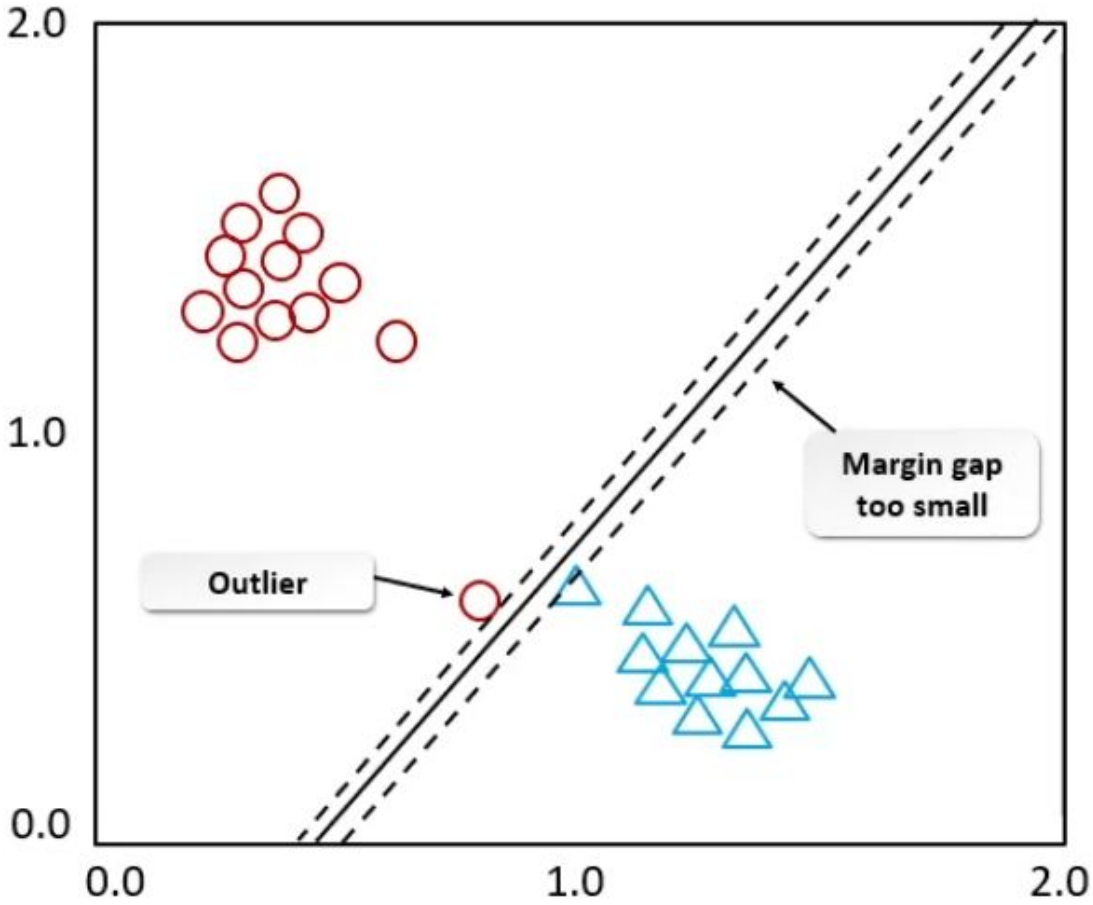

Linear Classification with SVMs
2.0
1.0
0.0
0.0
1.0
2.0
Test example
class = 1

hard Margins

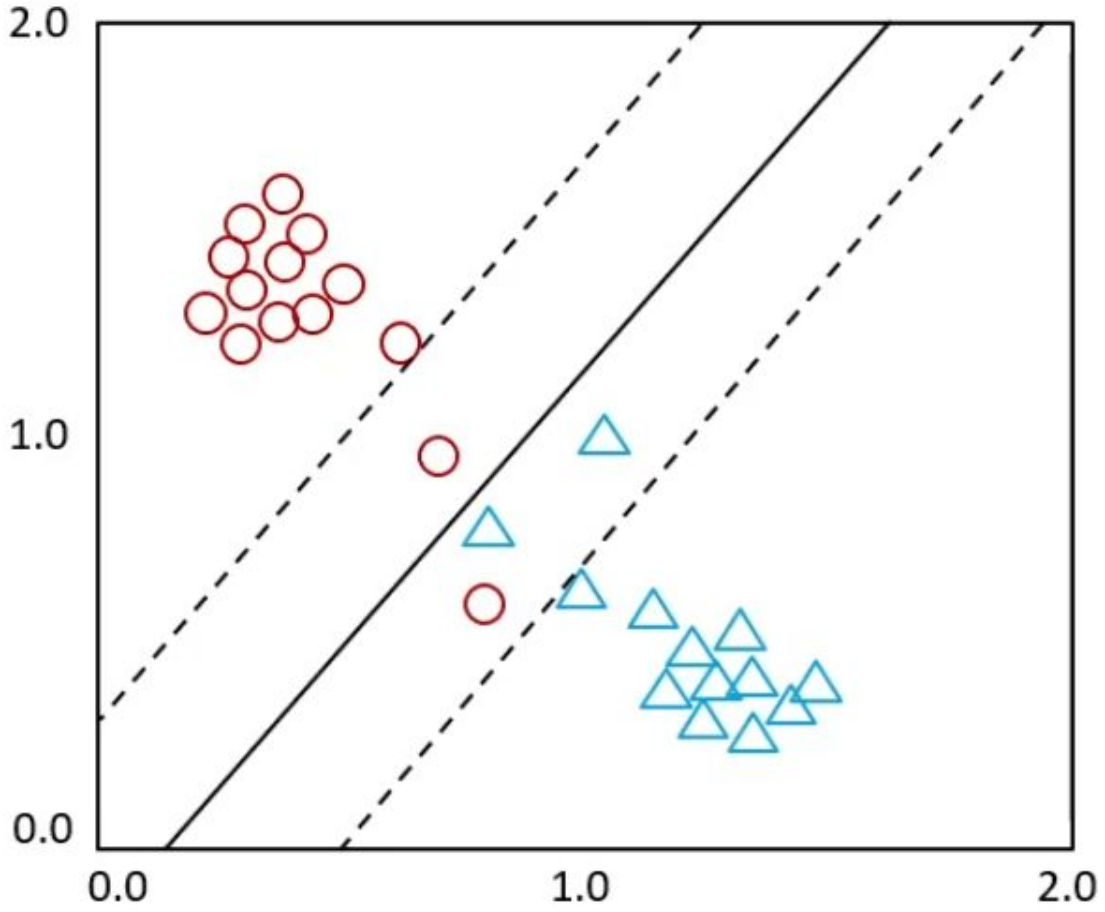


hard Margins




2.0
1.0
0.0
0.0
1.0
2.0
Margin gap
too small
Outlier

Soft Margins

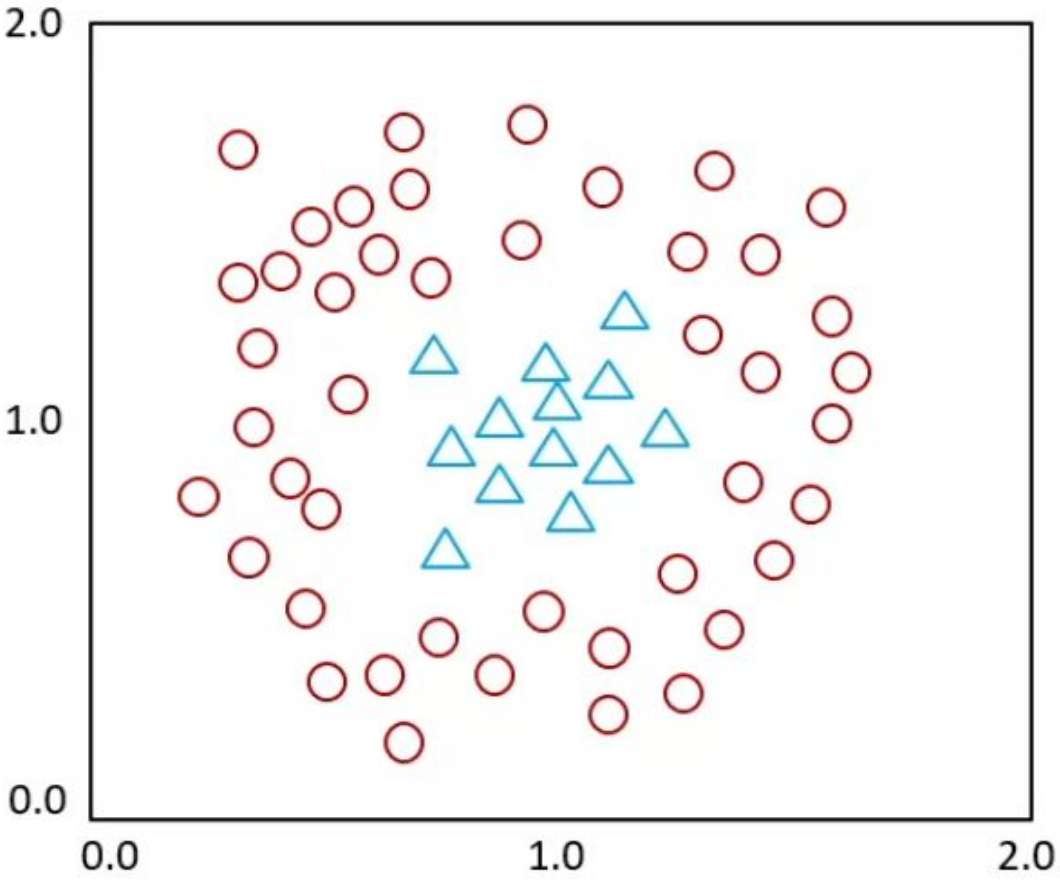


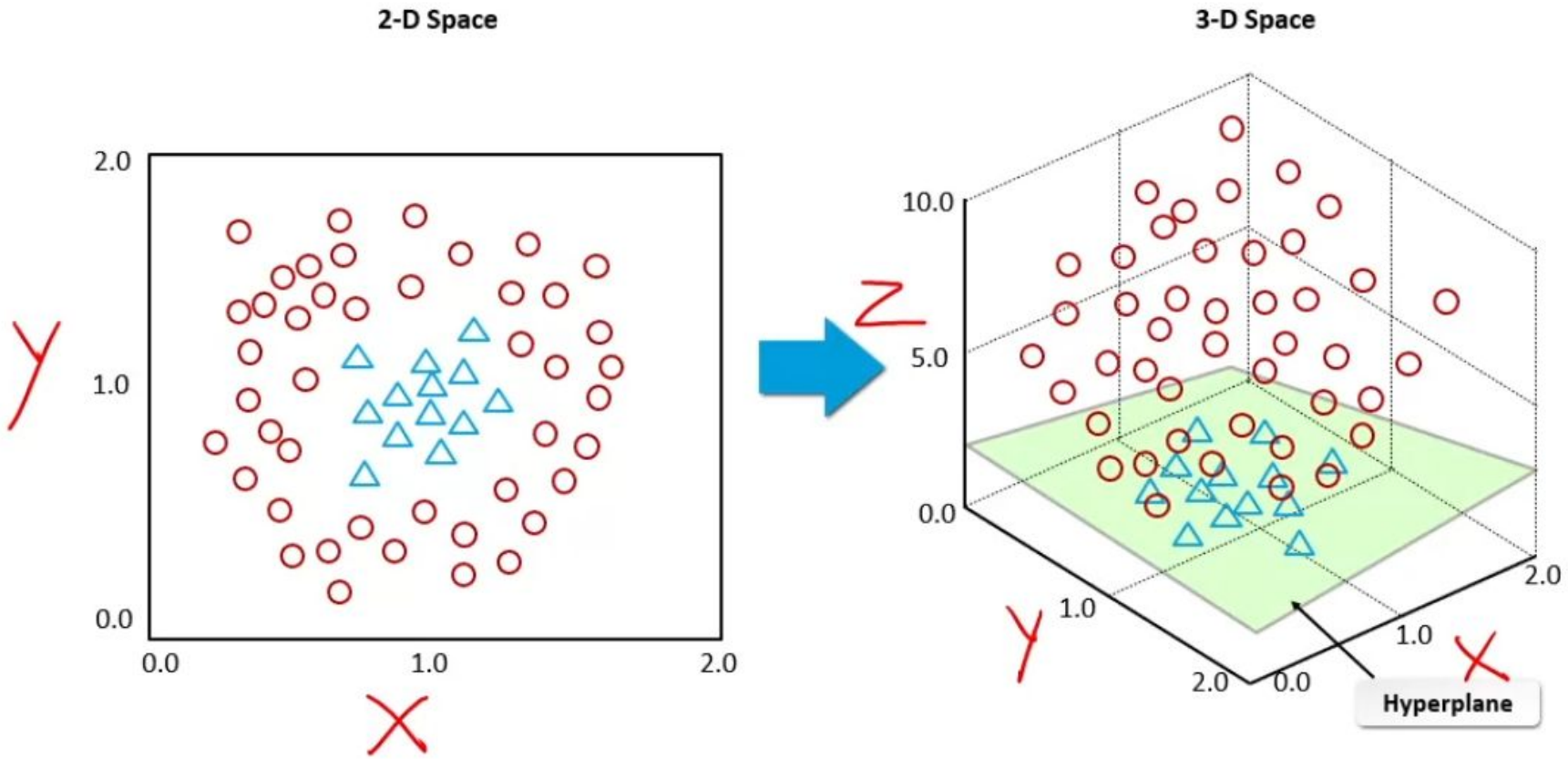
Regularization Penalty in SVM

As hyperparameter > **C**

low C >> soft margins
High C >> Hard margins >> overfitting

Nonlinear Classification




2-D Space
3-D Space
2.0
1.0
0.0
0.0
1.0
2.0
Y
X
Z
10.0
5.0
0.0
Y
X
Hyperplane

Feature Expansion / Feature Mapping / Basis Expansion

- Explicitly transform input into higher-dimensional features

$$K(x, y) = \langle f(x), f(y) \rangle$$

Assume:

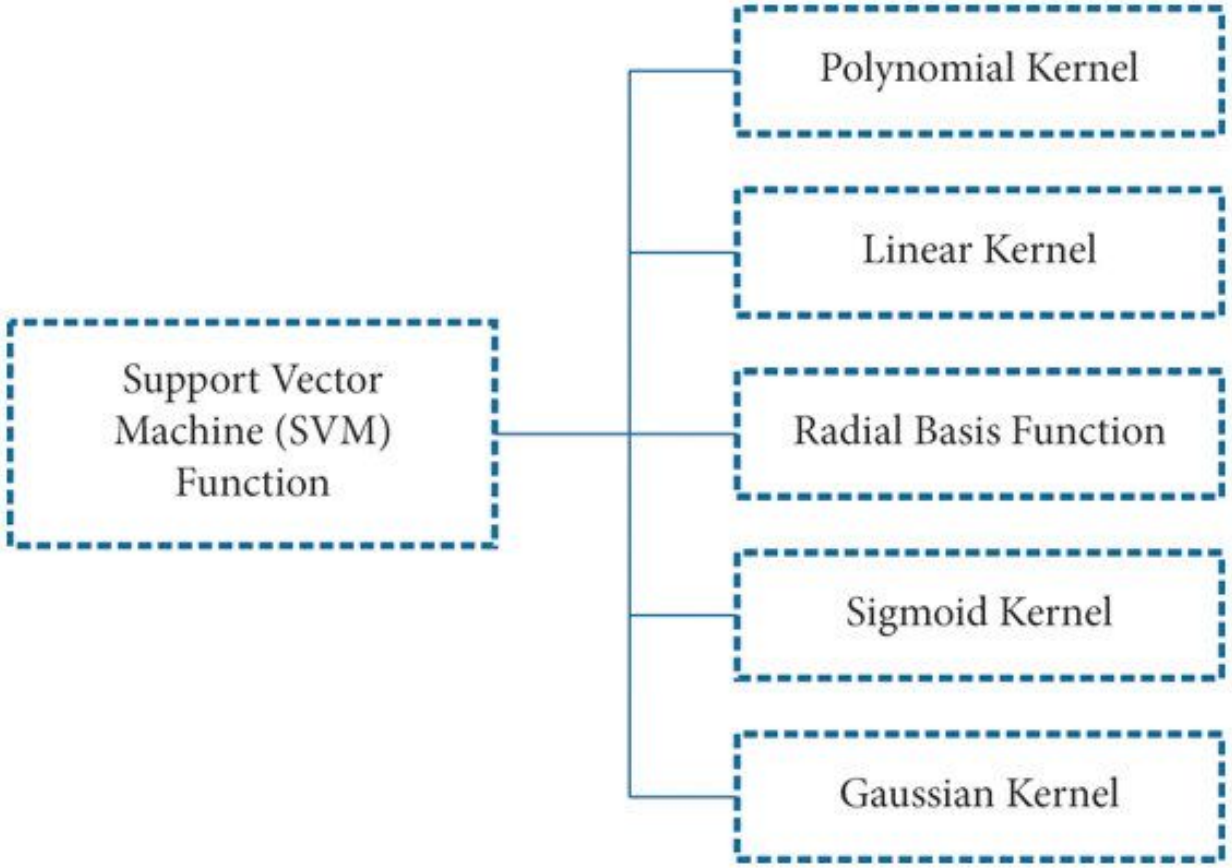
- **x = (x_1, x_2, x_3)**
- **y = (y_1, y_2, y_3)**

$$(x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$$

Kernel Trick

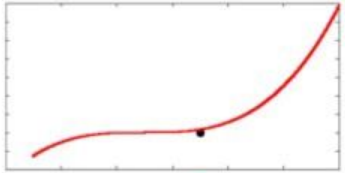
$$x = (0, 1, 2)$$
$$y = (3, 4, 5)$$

$$K(x, y) = (0 + 4 + 10)^2 = 196$$


Support Vector Machine (SVM) Function
Polynomial Kernel
Linear Kernel
Radial Basis Function
Sigmoid Kernel
Gaussian Kernel

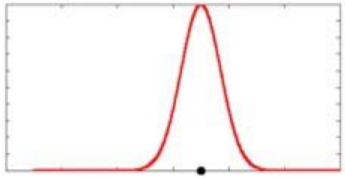
Common kernel functions

- Some commonly used kernel functions & their shape:
- Polynomial $K(a,b) = (1 + \sum_j a_j b_j)^d$



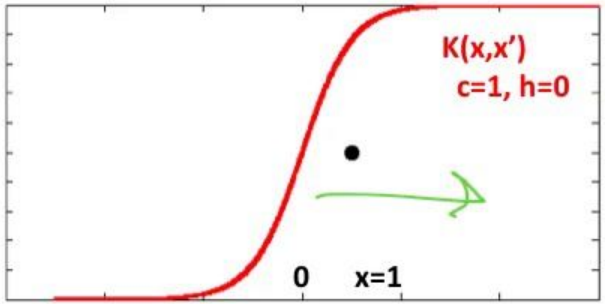
- Radial Basis Functions & Gaussian

$$K(a,b) = \exp(-(a-b)^2/2\sigma^2)$$



- Saturating, sigmoid-like:

$$K(a,b) = \tanh(ca^T b + h)$$



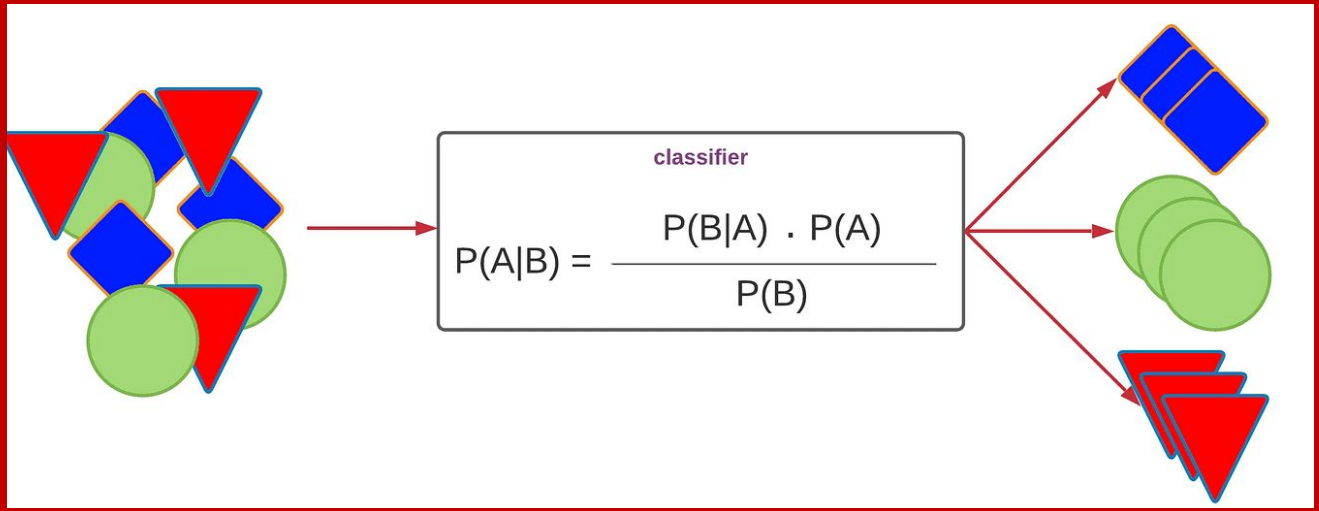
Guidelines for Building SVM Models for Classification

- Consider using an SVM model when the problem you are trying to solve is sensitive to outliers.
- Consider using an SVM model when working with a high-dimensional dataset.
- In classification, recognize that the goal of an SVM model is to widen the margins as much as is feasible, while at the same time keeping data examples outside of the margins.
- Tune the (c)regularization hyperparameter to adjust the size of the margins.

Guidelines for Building SVM Models for Classification

- Consider that narrowing the margins too much to keep all examples outside of those margins may lead to complications (e.g., overfitting).Consider softening the margins to avoid hard-margin overfitting issues.
- Recognize that softening the margins will likely place some examples within those margins, which is often a necessary tradeoff.
- Apply a kernel trick method to SVM models whose training data is not linearly separable.
- Consider the different types of kernel methods and how one might be more applicable to your current problem.
- gamma > قديش المودل مركز على الجيران القربيين ولا مبسط اكثر؟

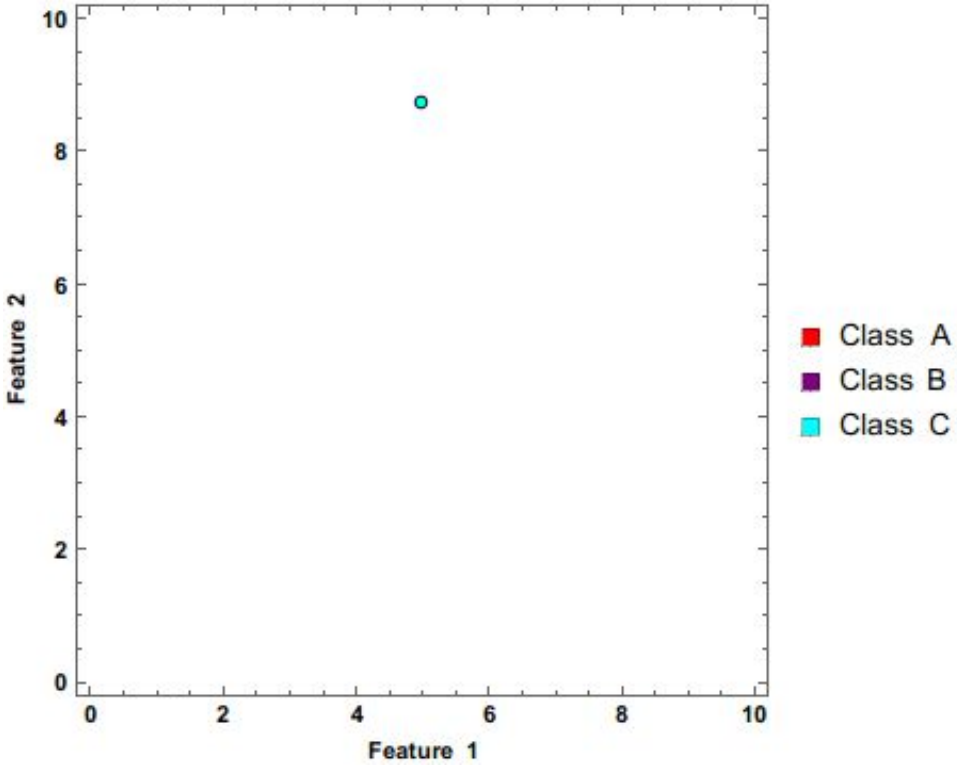
Naïve Bayes' Classifier



Naïve Bayes Classifier

- Naïve Bayes Classifier can be trained easily and fast and can be used as benchmark model.
- When variable selection is carried out properly, Naïve Bayes can perform as well as or even better than other statistical models such as logistic regression.
- Naive Bayes requires a strong assumption of independent predictors, so when the model has a bad performance, the reason leading to that may be the dependence between predictors.
- Doesn't require feature scaling

Naïve Bayes' Classifier



Naïve Bayes' Classifier:

- **Naive Bayes is the simplest algorithm that you can apply to your data.**
- **This algorithm assumes all the variables in the dataset "Naive" i.e., not correlated to each other.**
- It requires categorical values.
- It uses Bayes' theorem as its base.
- Mostly used to get the base accuracy of the dataset.
- Mostly used with:
  - Real time Prediction.
  - Multi class Prediction.
  - Text classification/ Spam Filtering/ Sentiment Analysis, and Recommendation Systems.

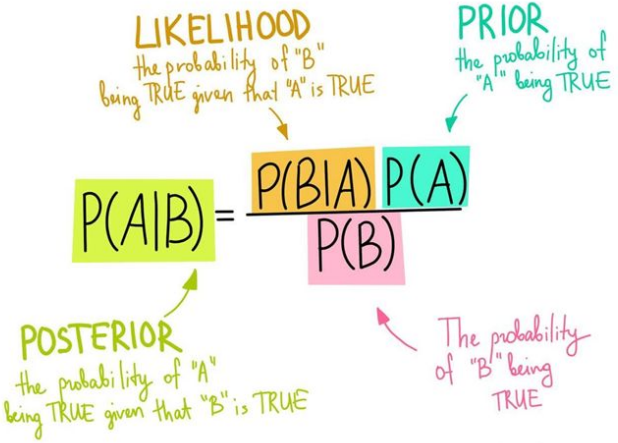
Types

- Multinomial Naïve Bayes → for text (word counts).
- Bernoulli Naïve Bayes → for binary data (0/1 features).
- Gaussian Naïve Bayes → for numerical (continuous) data.

Example:

Priors: احتمال كل كلاس حسب عدد الأمثلة.

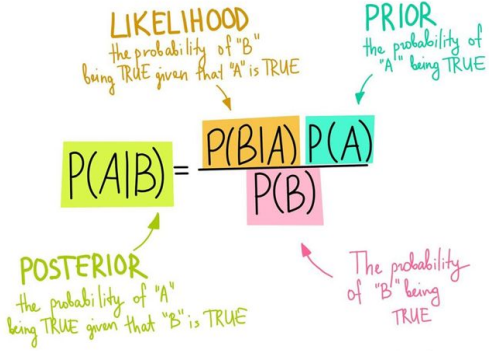
Likelihoods: لكل Feature، عد كم مرة ظهر مع كل كلاس.



- **Problem:** Classify whether an email is "spam" or "not spam"
- **Features:** Presence (1) or absence (0) of words "buy", "cheap", "offer".
- **Training Data:**

Email	buy	cheap	offer	Class
1	1	0	1	spam
2	1	1	0	not spam
3	0	1	1	spam
4	1	0	0	not spam
5	0	1	1	spam

Calculate Priors



Prior Probability of Spam ($P(\text{spam})$):

$$P(\text{spam}) = \frac{\text{Number of spam emails}}{\text{Total number of emails}} = \frac{3}{5} = 0.6$$

Prior Probability of Not Spam ($P(\text{not spam})$):

$$P(\text{not spam}) = \frac{\text{Number of not spam emails}}{\text{Total number of emails}} = \frac{2}{5} = 0.4$$

Calculate Likelihoods

- **Likelihoods for "spam" class:(1,3,5)**
  - buy=1 → 1/3 , buy=0 → 2/3
  - cheap=1 → 2/3 , cheap=0 →1/3
  - offer=1 → 3/3=1 , offer=0 → 0
- **Likelihoods for "not spam" class: (2,4)**
  - buy=1 → 2/2=1 , buy=0 → 0
  - cheap=1 → 1/2 , cheap=0 → 1/2
  - offer=1 → 0 , offer=0 → 2/2=1

New Email to Classify

- New Email: $"buy"=1,"cheap"=1,"offer"=0$
- Objective: Determine whether the new email is "spam" or "not spam".

Calculate Posterior for Spam 1

Posterior Probability of Spam

$$(P(spam \mid buy = 1, cheap = 1, offer = 0)$$

$$\propto P(buy = 1 \mid spam) \cdot P(cheap = 1 \mid spam) \cdot P(offer = 0 \mid spam) \cdot P(spam)$$

= (⅓) * (⅔) * (0) * 0.6 **= 0**

New Email to Classify

- New Email: $"buy"=1,"cheap"=1,"offer"=0$
- Objective: Determine whether the new email is "spam" or "not spam".

Calculate Posterior for Not Spam

2

Posterior Probability for Not Spam

$$P(notspam \mid buy = 1, cheap = 1, offer = 0)$$

$$\propto P(buy = 1 \mid notspam) \cdot P(cheap = 1 \mid notspam) \cdot P(offer = 0 \mid notspam) \cdot P(notspam)$$

= (1) * (½) * (1) * 0.4 **= 0.2**

New Email: $"buy"=1, "cheap"=1, "offer"=0$

Step 5: Final Decision

- Spam = 0
- Not Spam = 0.2

  👉 **Email classify as : Not Spam ✅**

Why Normalize Probabilities?

1. **To Compare Probabilities Directly:**
   a. When calculating conditional probabilities $P(Y|X)$ using Bayes' Theorem, we need to normalize the probabilities so they can be compared directly.
   b. Without normalization, the probabilities are not on the same scale and cannot be directly compared.

> Example: Spam = 0.2, Not Spam = 0.05
> → Without normalization, you could say "Spam is bigger."
> → **But to say** *Spam = 80% chance* and *Not Spam = 20% chance*, **you need normalization.**

2. **To Ensure the Probabilities Sum to 1:**
   a. After normalization, the sum of the probabilities for all classes (e.g., spam and not spam) should equal 1.
   b. This ensures we have a valid probability distribution that can be used for classification.

- Without smoothing → rare/missing features kill probabilities.
  - If a **feature is very rare or never appears** in the training data for a class, then its likelihood becomes **0**.
- With Laplace smoothing → every feature has at least a small chance, making the model more robust.

Laplace Smoothing (Add-1 Smoothing)

- To fix this, we use **Laplace smoothing** (also called **add-1 smoothing**).
- Instead of using raw counts, we add **+1 to every count**:

$$P(feature \mid class) = \frac{\text{count}(\text{feature}, \text{class}) + 1}{\text{count}(\text{class}) + k}$$

where:

- k = number of possible feature values (for binary features, $k = 2$).

Example

Suppose in "Spam" emails we never saw the word **"offer=0"**.

- Without smoothing:

$$P(offer = 0 \mid Spam) = 0$$

- With Laplace smoothing:

$$P(offer = 0 \mid Spam) = \frac{0 + 1}{3 + 2} = 0.2$$

Now it's a small probability instead of zero.

Supervised machine learning

Algorithm	Needs Feature Scaling?	Reason
KNN (K-Nearest Neighbors)	✅ Yes	Relies on distance between points
SVM (Support Vector Machine)	✅ Yes	Depends on dot product and distances
Linear Regression	✅ (Recommended)	Regularization + Gradient Descent convergence
Logistic Regression	✅ (Recommended)	Regularization + coefficient stability
Decision Tree	❌ No	Works with threshold splits, not distances
Random Forest	❌ No	Based on Decision Trees
Naïve Bayes	⚠️ Sometimes	- **Gaussian NB** assumes normally distributed data → scaling helps - **Multinomial / Bernoulli NB**: handle counts/binary values → scaling not needed