Inteligencia Artificial

Instituto Nacional de Astrofísica, Óptica y Electrónica

Presents:

MSc. Mireya Lucia Hernández Jaimes

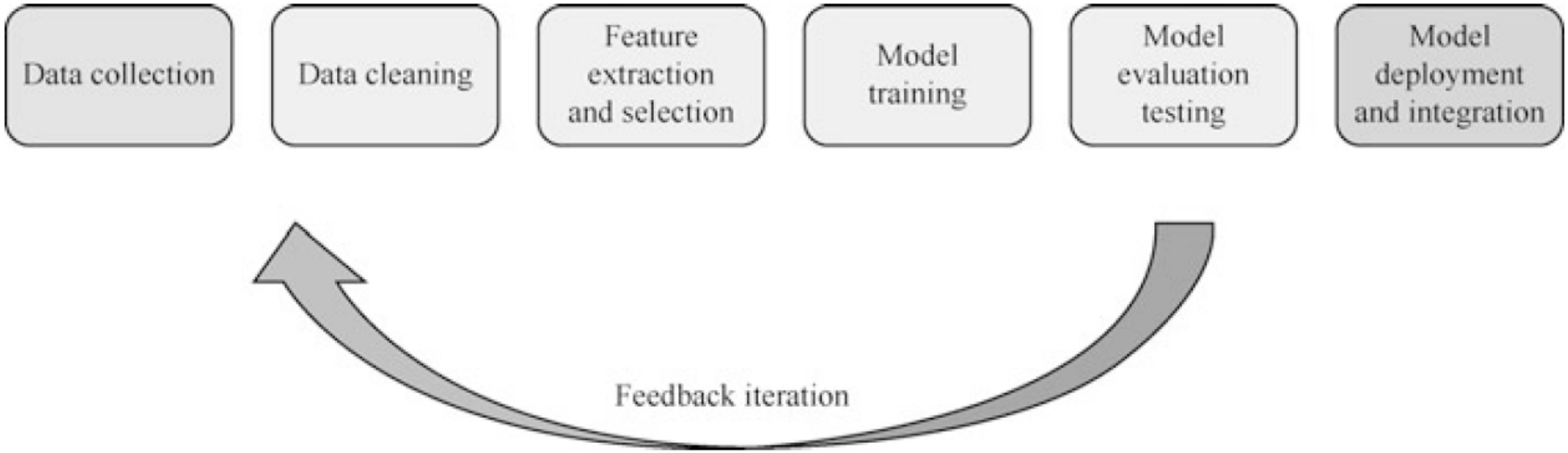
Content

1. Quizz

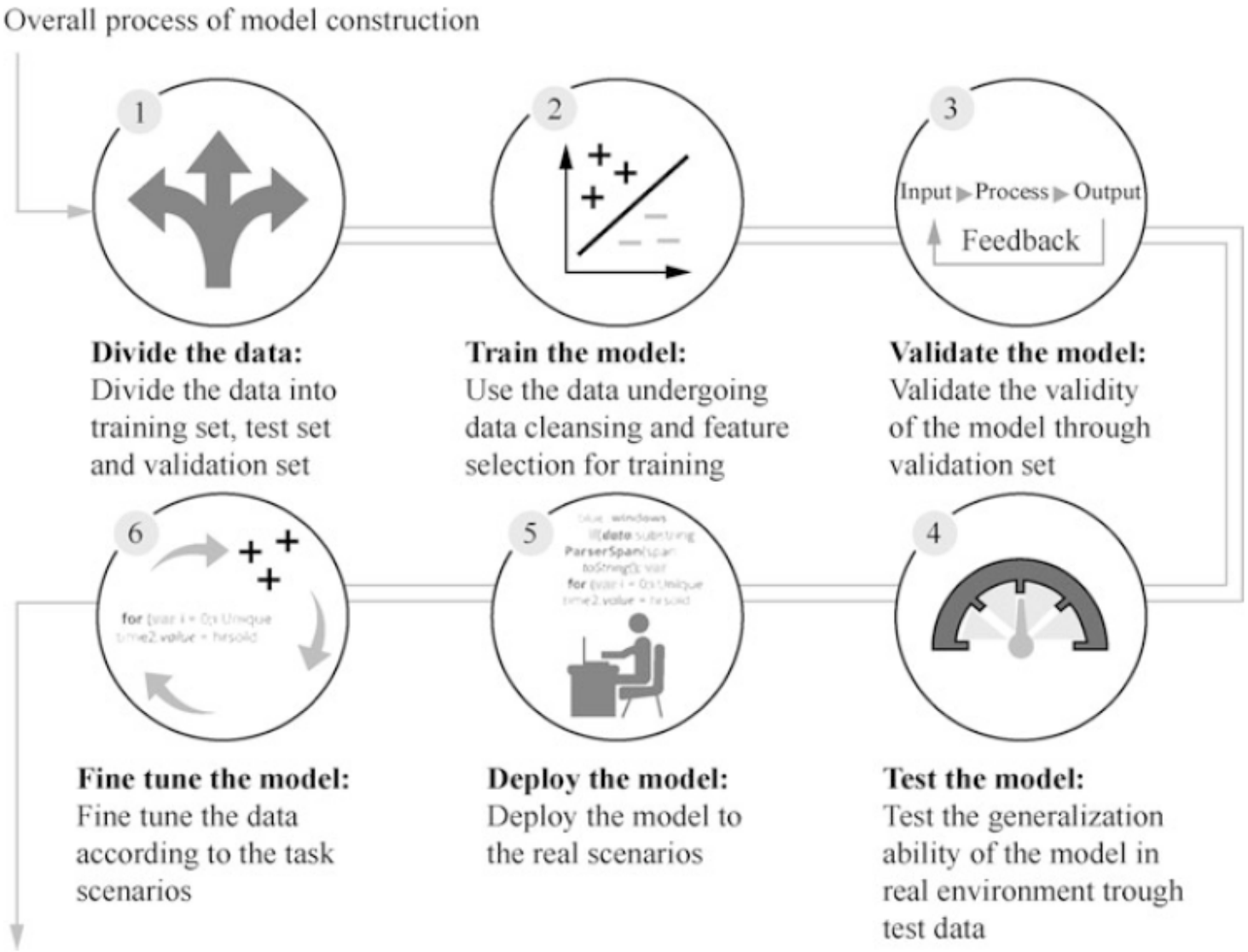
Quizz

- ¿Por qué es importante pre-procesar los datos?
- ¿Qué significa que un dataset no esté balanceado?
- ¿Cuáles son las 2 tareas principales que pueden realizar los algoritmos de ML?
- Menciona 2 métricas para evaluar a los clasificadores
- ¿Cómo podemos representar los valores TP, FP, TN, y FN?
- ¿Cuál es la diferencia entre las métricas de precisión y exactitud (accuracy)?
- Menciona 2 métricas para evaluar los algoritmos de regresión

Proceso de los algoritmos de ML



Construcción de los algoritmos de ML



- El objetivo de los modelos de ML es que puedan ser aplicados a nuevas muestras (muestras desconocidas) y no solo en muestras de entrenamiento.
- La habilidad del model para aplicarse a nuevas muestras se le llama capacidad de generalización, tambien conocido como robustes del algoritmo.
- Error: la diferencia entre el resultado arrojado por el algoritmo y el valor real.
- Error de entrenamiento: Se refiere al error en el conjunto de entrenamiento. Un aumento en el error de entrenamiento ocasiana un sub-ajuste (underfitting).
- Error de generalización:Se refiere al error en el conjunto de prueba (nuevas muestras). Un aumento en el error de entrenamiento de generalización ocasiona un sobre-ajuste (overfitting).

Parámetros de los modelos ML:

- No se fijan manualmente por el usuario.
- Se obtienen mediante el aprendizaje de los datos.
- Ejemplos: los pesos en las redes neuronales, vectores de soporte en el algoritmo SVM (máquina de Soporte Vectorial), los coeficientes en los algoritmos de regresión líneal y logística.

Hiperparámetros de los modelos ML:

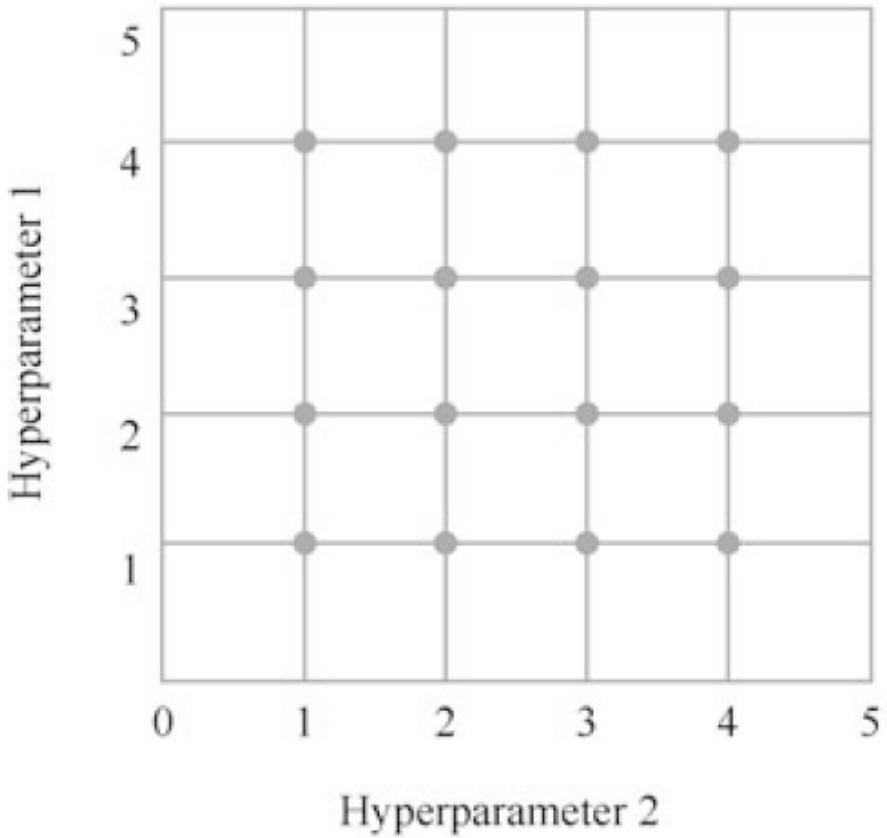
- Se configuran por el usuario.
- En general, es necesario ajustar los hiperparámetros del modelo cuando se tienen diferentes conjuntos de datos para distintas tareas.
- Los hiperparámetros del modelo también se pueden configurar mediante alguna heurística.
- Ejemplos: coeficiente de penalidad en el algoritmo Ridge Regression, la razón de aprendizaje y la función de activación de las redes neuronales (hay más hyperparámetros en las redes neuronales), y el número de K vecinos en el algoritmo K-Nearest Neighbor.

Túneo de los hiperparámetros

- Grid Search
- Random Search

Grid Search

Busca exhaustivamente todas las combinaciones posibles de hiperparámetros para formar una cuadrícula de valores de hiperparámetros:



Es necesario designar manualmente el rango y la longitud del paso de la cuadrícula.

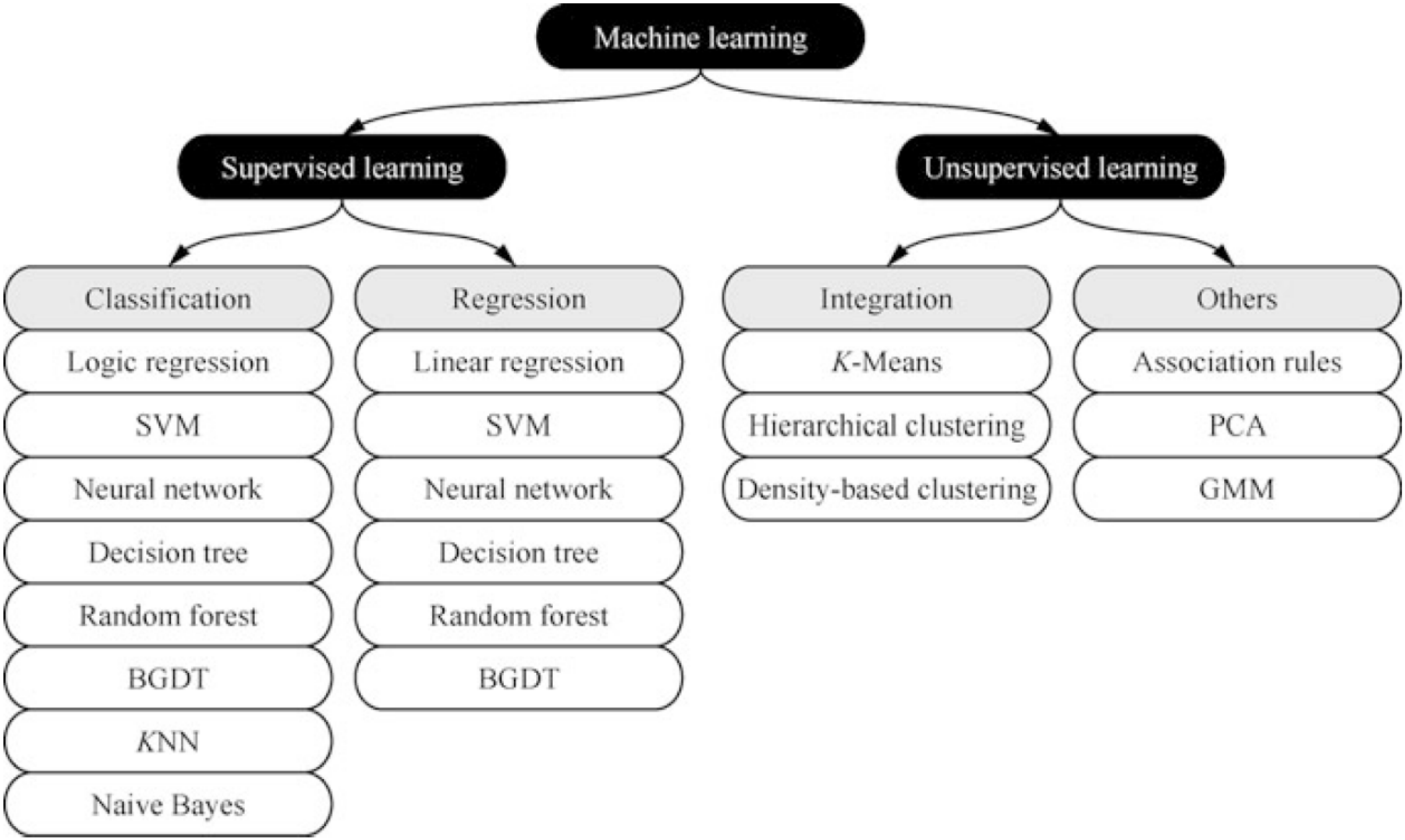
- Si el modelo ML tiene un número relativamente pequeño de hiperparámetros, la búsqueda en cuadrícula es aplicable, por lo que la búsqueda en cuadrícula es factible en los algoritmos generales de aprendizaje automático.
- En caso de modelos de DL (Redes neuronales), la búsqueda en cuadrícula es demasiado costosa y requiere mucho tiempo, por lo que no se adopta en la mayoría de los casos.

Implementa un muestreo aleatorio de parámetros, donde cada configuración debe tomar muestras de la distribución de posibles valores de parámetros, tratando de encontrar el mejor subconjunto de parámetros.

Pseudocode for Random Search.

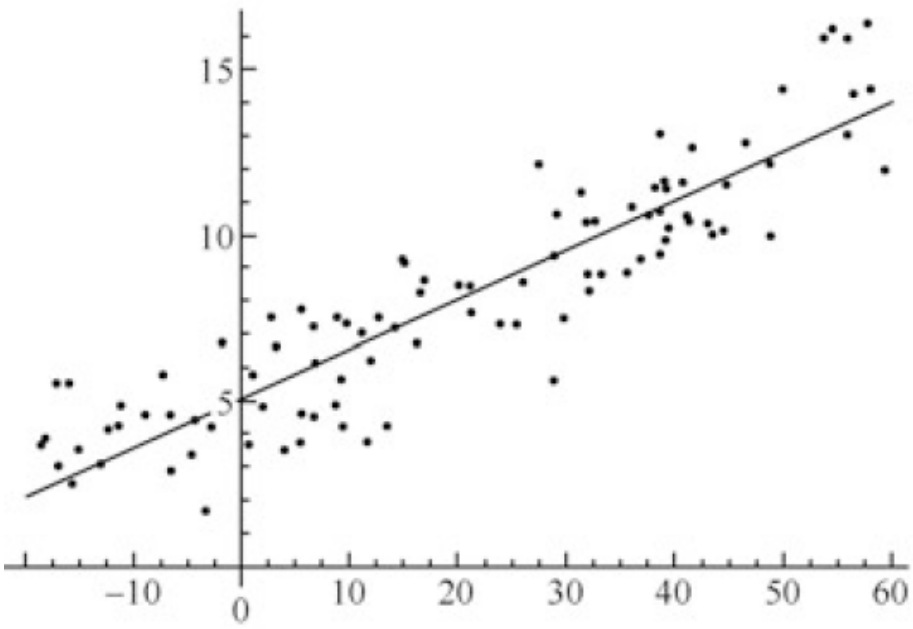
```
Input: NumIterations, ProblemSize, SearchSpace
Output: Best
Best ← ∅;
foreach iter_i ∈ NumIterations do
    candidate_i ← RandomSolution(ProblemSize, SearchSpace);
    if Cost(candidate_i) < Cost(Best) then
        Best ← candidate_i;
    end
end
return Best;
```

Ejemplos de algoritmos de ML

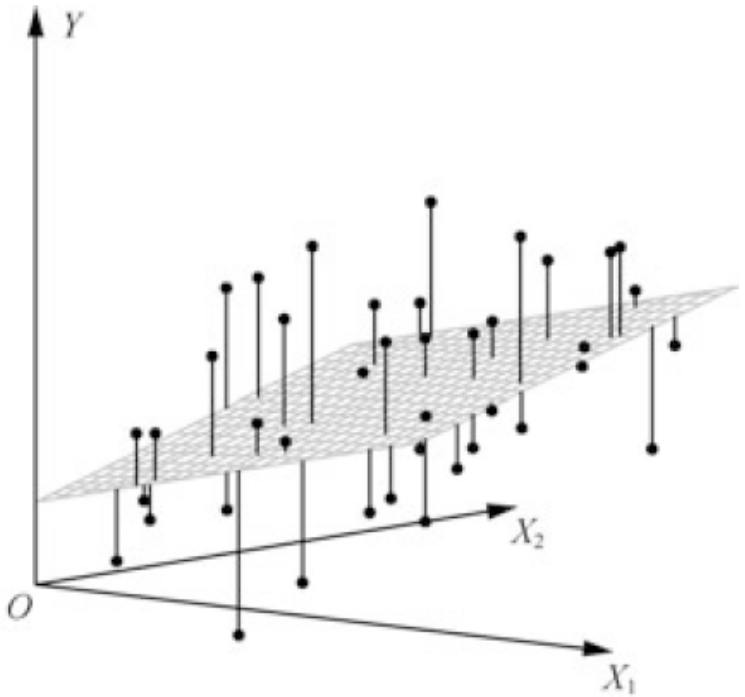


Algoritmos supervisados

Es un algoritmo de regresión supervisado. Se basa en un análisis de regresión estadístico para determinar la relación cuantitativa entre dos o más variables.



(a) Unary Linear Regression



(b) Multiple Linear Regression

- (Unary) Simple LR: Involucra solo una variable independiente. La relación entre la variable dependiente y la variable dependiente X se modela con una línea recta.
- Multiple LR: Involucra dos o más variables independientes.

¿Cómo integramos las características de las muestras en una función líneal?

$$h(x) = w^{\mathrm{T}} x + b$$

- h(x) es el valor que predice el algoritmo (variable dependiente).
- w son los pesos asociados a la(s) variable(s) independiente(s).
- x es el vector de características de la(s) variables independiente(s).
- b es el término bias

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$w^T x = \begin{pmatrix} w_1 & w_2 & \dots & w_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

El objetivo al entrenar un modelo de regresión lineal es encontrar los parámetros óptimos de w y b, que minimice el error entre los valores predichos h(x) y los valores reales y(x). Esto generalmente se hace usando el método de mínimos cuadrados, que minimiza la suma de los errores cuadrados (residuales).

$$J(w,b) = \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$$

- J(w,b) es la función costo/ de pérdida (loss function).
- m representa el número de instancias de entrenamiento.

La optimización de w y b se puede realizar con algoritmos de optimización, como el algoritmo de Descenso del gradiente (Gradient descent)

$$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$$
$$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$$

- α representa la razón de aprendizaje (hiperparámetro del algoritmo Gradient descent)

¿Cuando se detiene? definir un umbral o fijar numero de iteraciones,

Ejemplos de aplicaciones:

- Simple LR:
  - Predecir las ventas en función del gasto publicitario.
  - Estimar la edad basada en la edad de la persona.
- Multiple LR:
  - Predecir los precios de la vivienda en función de factores como el tamaño, la ubicación y la cantidad de habitaciones.
  - Estimar el peso de una persona en función de la altura, la edad y la dieta.

Es un algoritmo de clasificación supervisado. A diferencia de LR, LogR predice la probabilidad de que una entrada determinada X pertenece a una clase (por ejemplo, 0 o 1).

$$h(x) = P(Y = 1|X) = g\left(\boldsymbol{w}^{\mathrm{T}}x + b\right)$$

- h(x) es la probabilidad predicha por el algoritmo para identificar si la salida Y corresponde a la clase=1 con respecto a un valor independiente (una caracteristica dada).
- g es la funcion sigmoid
- w son los pesos asociados a las características
- b es el bias

Función Sigmoid:

$$g(x) = \frac{1}{1 + \exp\{-x\}}$$

Al igual que en LR, hay que optimizar los valores w y b con la función de costo logística (logistic loss/ log loss / binary cross-entropy loss).

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h(x^{(i)})) + (1-y^{(i)})\log(1-h(x^{(i)}))\right]$$

- J(w,b) es la función costo/ de pérdida.
- m representa el número de instancias de entrenamiento.
- h(x) valores dados por el algoritmo
- y(x) valores reales

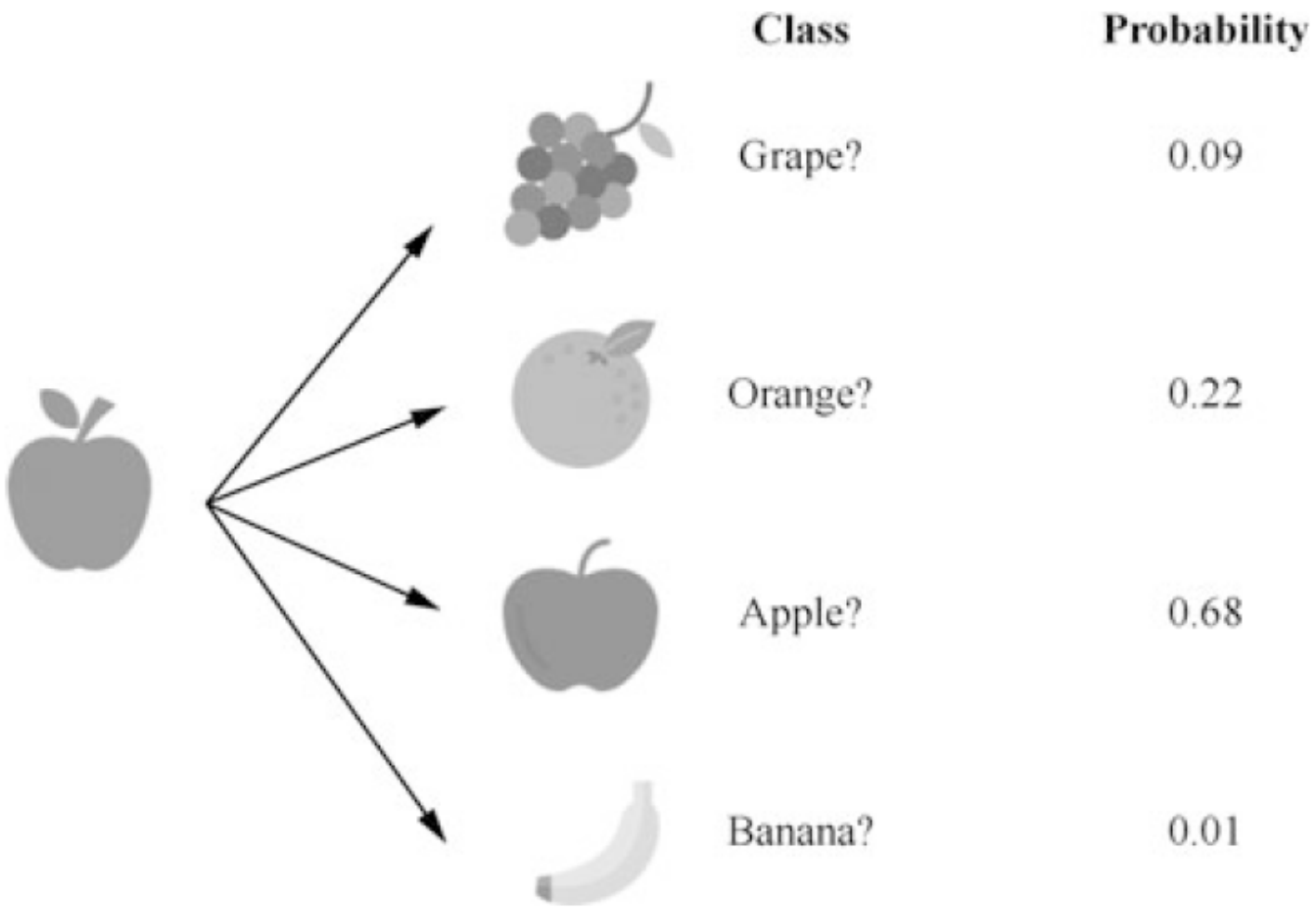
Los valores óptimos de w y b tambien se pueden obtener con el algoritmo Gradient Descend.

¿Qué pasa si tenemos más de dos clases, es decir, tenemos que resolver una tarea multi-clase? La presentación anterior de LogR es para tareas de clasificación binaria.

La función Softmax regression nos permite generalizar el modelo LogR para resolver problemas de k-clases. Nos da la probabilidad de multiples clases.

$$P(Y = c|x) = \frac{\exp\left\{w_c^{\mathrm{T}} x + b\right\}}{\sum_{l=1}^{k} \exp\left\{w_c^{\mathrm{T}} x + b\right\}}$$

Ejemplo de la función Softmax regression, donde la suma de las probabilidades de todas las clases debe dar 1.



Métodos de regularización: son aquellas técnicas que buscan mejorar la capacidad de generalización de los modelos ML y evitar el sobre ajuste (overfitting).

Métodos de regularización vs Métodos de optimización.

- M. Regularización: incremetar la rosbutes de los modelos de ML, es decir, dismuir el error de generalización.
- M. Optimización: optimizar los parámetros de los modelos de ML para mejorar el desempeño de estos (ejemplo: accuracy)

2 métodos de regularización comunes en LogR:

- **L1 regularization:** Agrega el valor absoluto de magnitud del coeficiente como término de penalización a la función de pérdida/costo.

$$\|w\|_1 = \sum_i |w_i|$$

- **L2 regularizacion:** Agrega una magnitud al cuadrado del coeficiente como término de penalización a la función de pérdida/costo.

$$\|w\|_2 = \sqrt{\sum_i w_i^2}$$

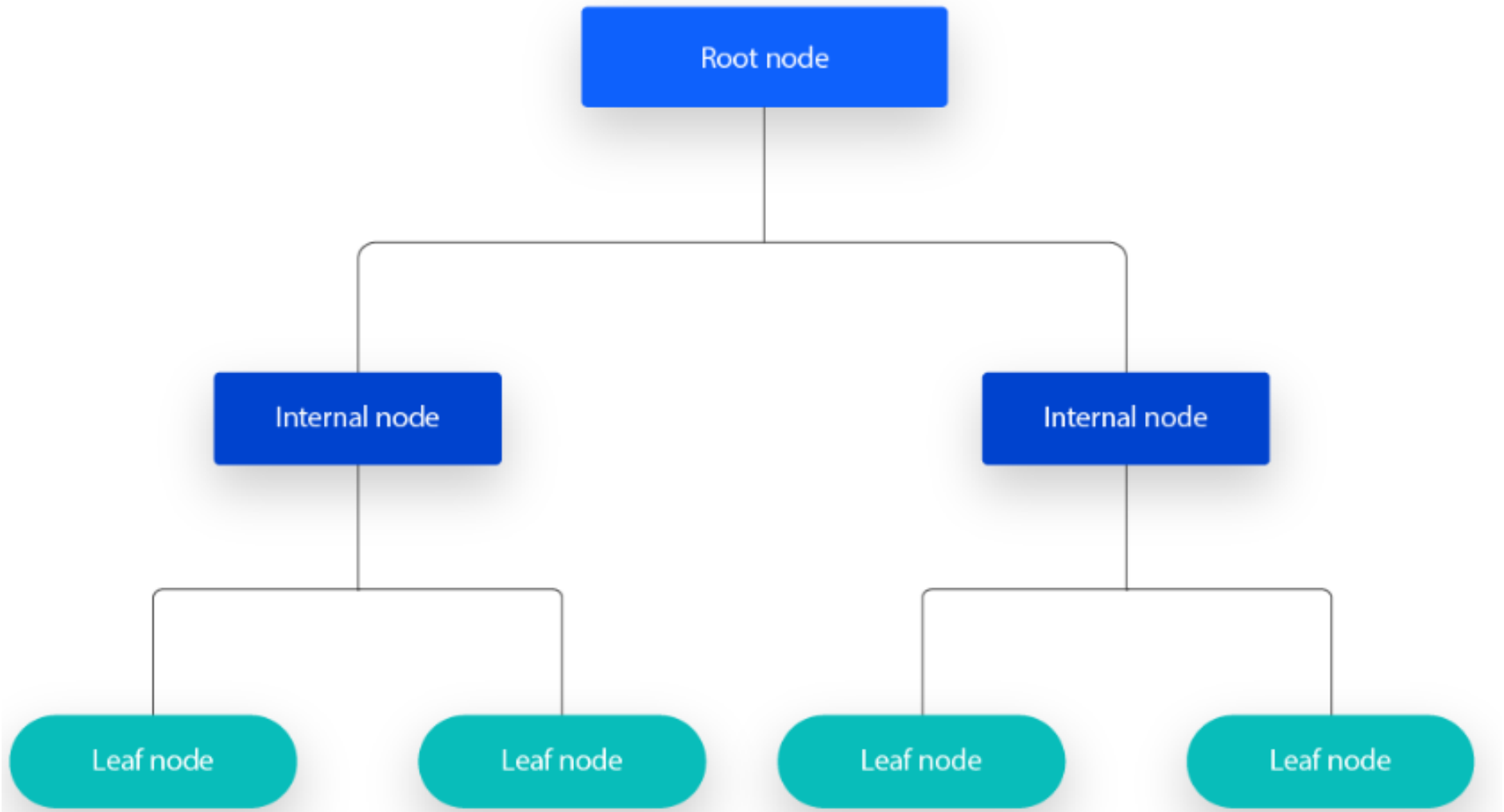
- **Lasso Regression:** son aquellos LogR que utilizan L1

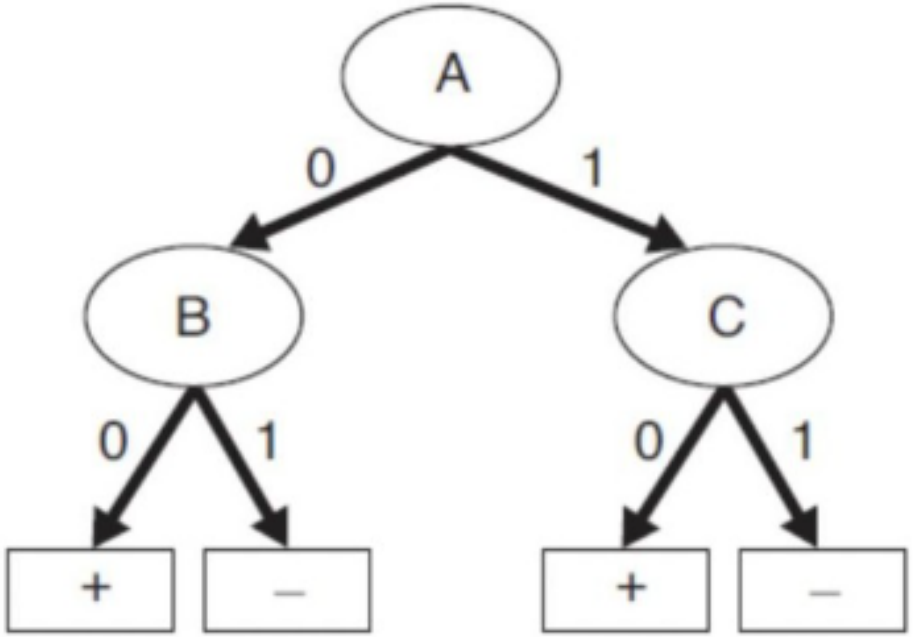
- **Ridge Regression:** son aquellos LogR que utilizan L2

La diferencia clave entre estas técnicas es que L1 reduce el coeficiente de la característica menos importante a cero, eliminando así alguna característica por completo. Entonces, esta técnica podría funcionar bien como mecanismo de selección de características.

- Resuelve las tareas de regresión/predicción y clasificación.
- Su aprendizaje se basa en establecer reglas mediante una estructura jerarquica apartir de las características de las muestras.
- Los elemetos principales son:
  - Nodo raíz: es la base del algoritmo o la caracterísitca más relevante.
  - Nodos internos: simbolizan una decisión de acuerdo con la característica / dato de entrada.
  - Nodos hoja: Son los nodos que no tienen hijos que indican una predicción o una clasificación (valor numérico o clase).

Decision Tree





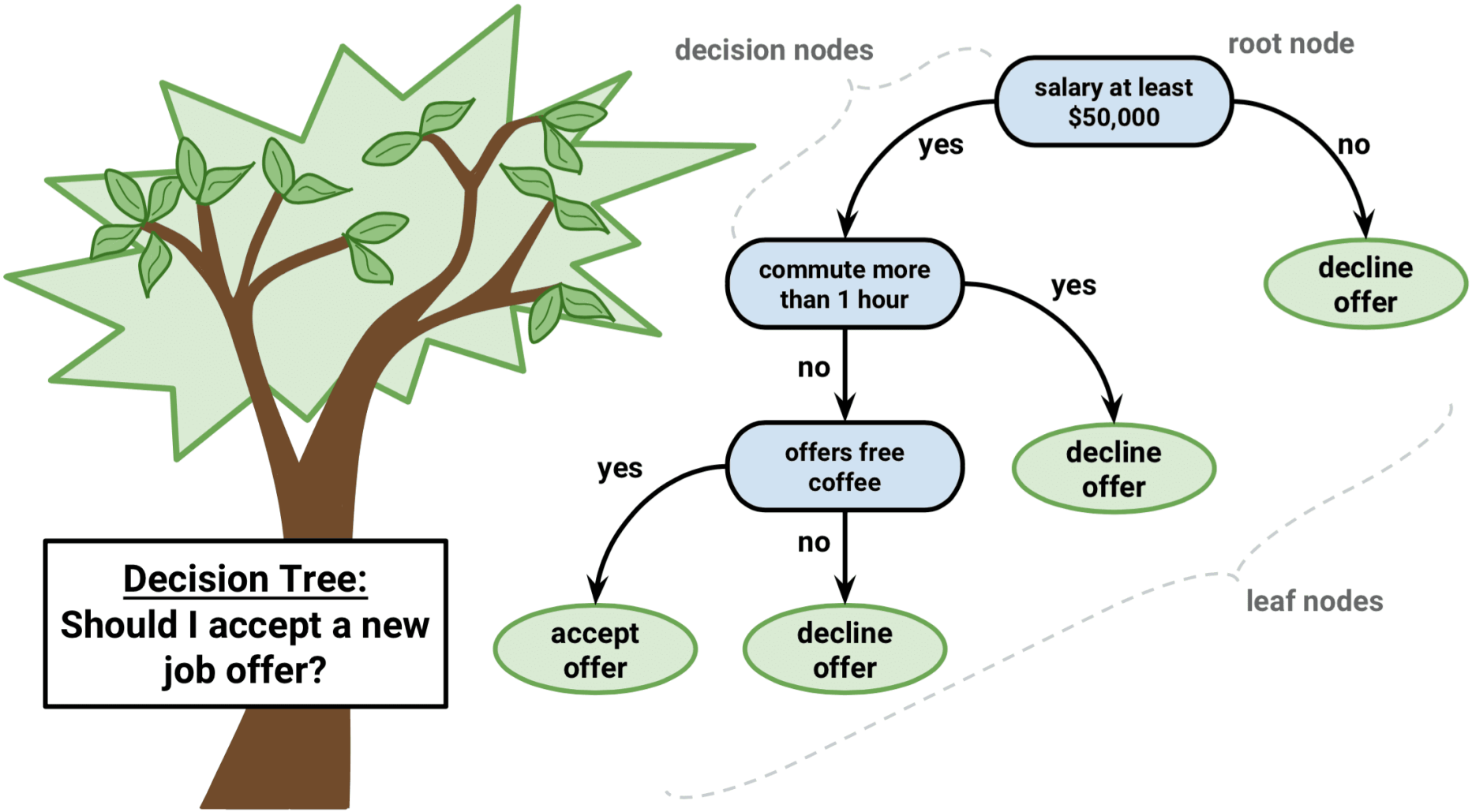
Training:

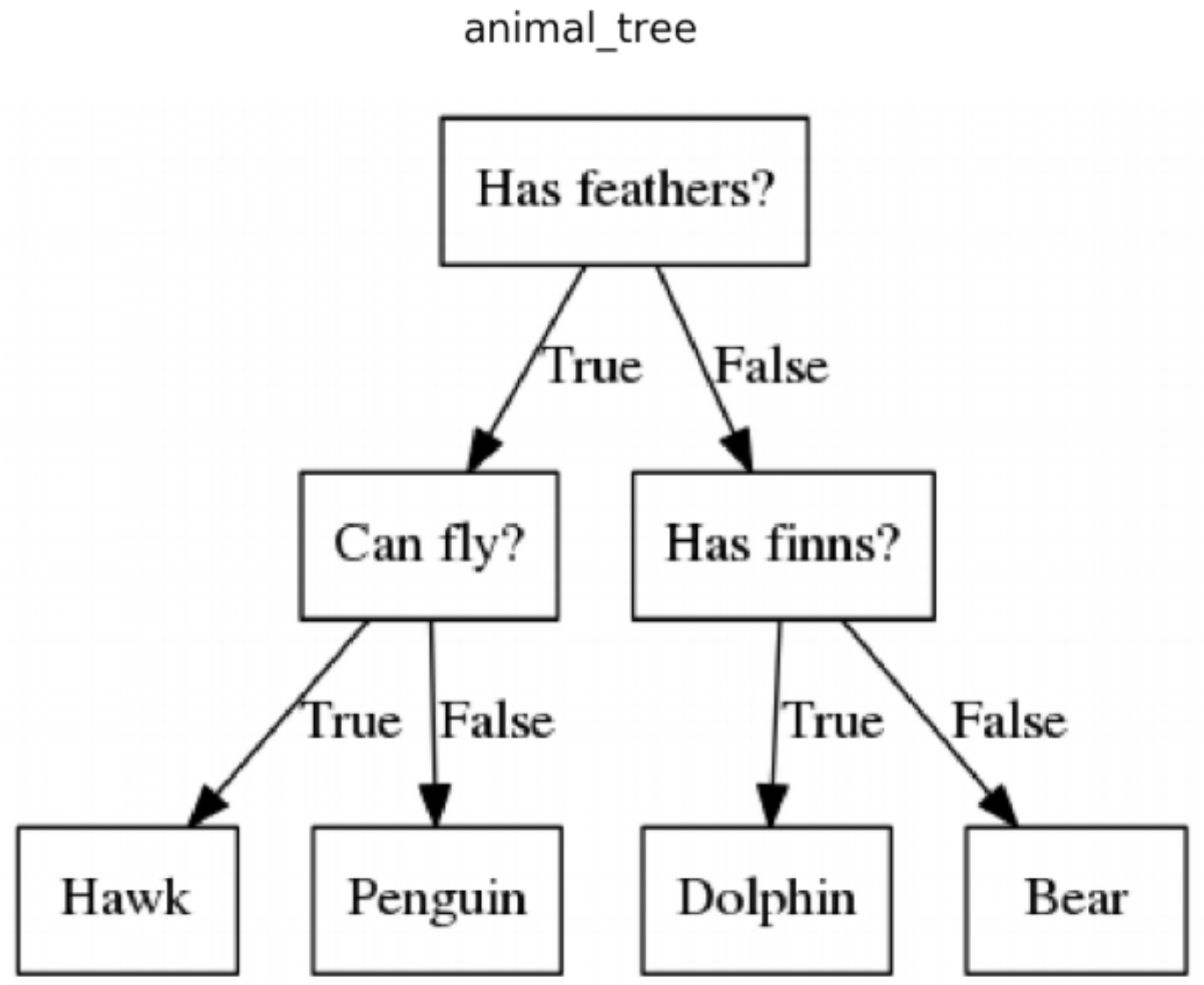
Instance	A	B	C	Class
1	0	0	0	+
2	0	0	1	+
3	0	1	0	+
4	0	1	1	–
5	1	0	0	+
6	1	0	0	+
7	1	1	0	–
8	1	0	1	+
9	1	1	0	–
10	1	1	0	–

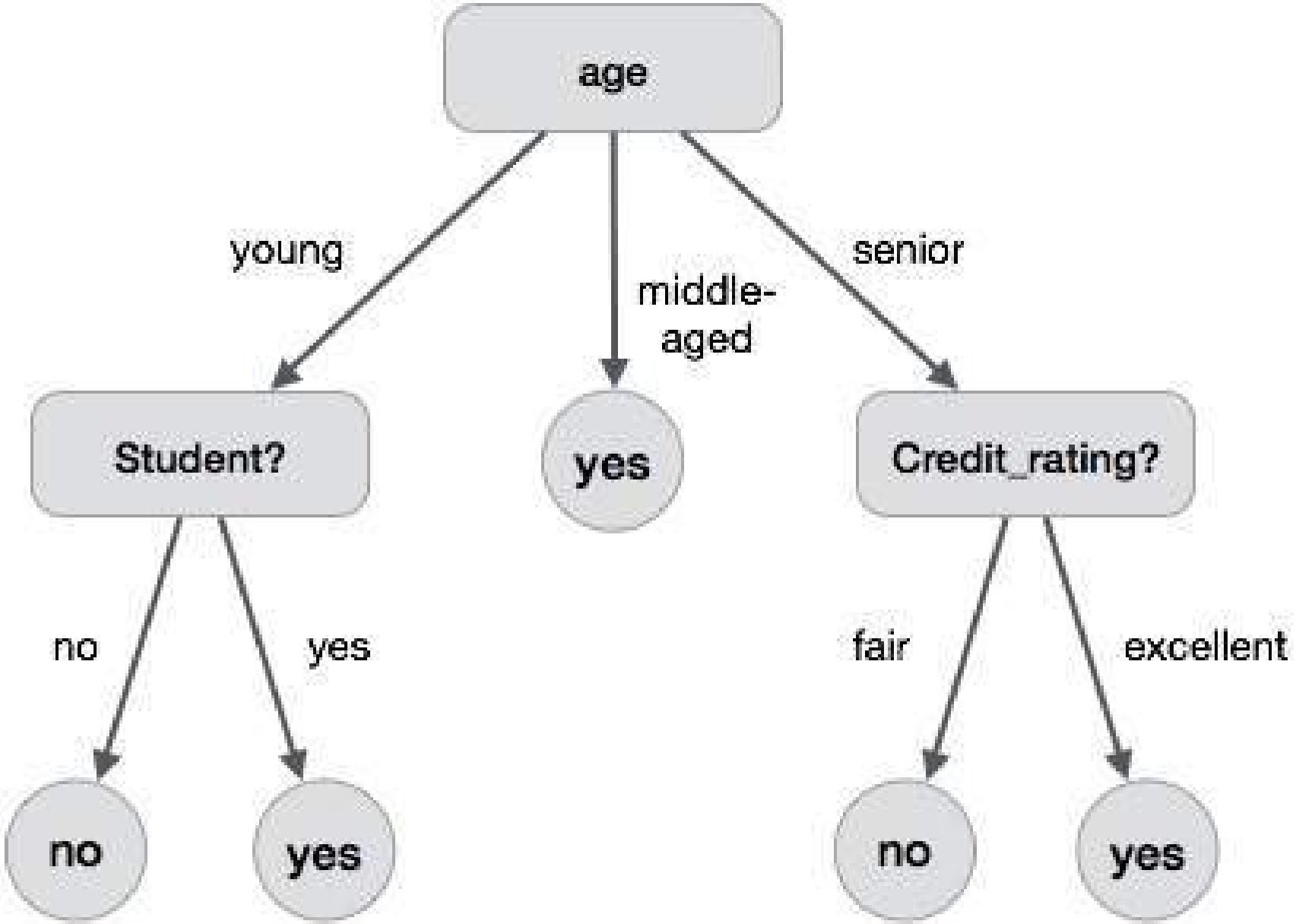
Validation:

Instance	A	B	C	Class
11	0	0	0	+
12	0	1	1	+
13	1	1	0	+
14	1	0	1	–
15	1	0	0	+

Decision Tree



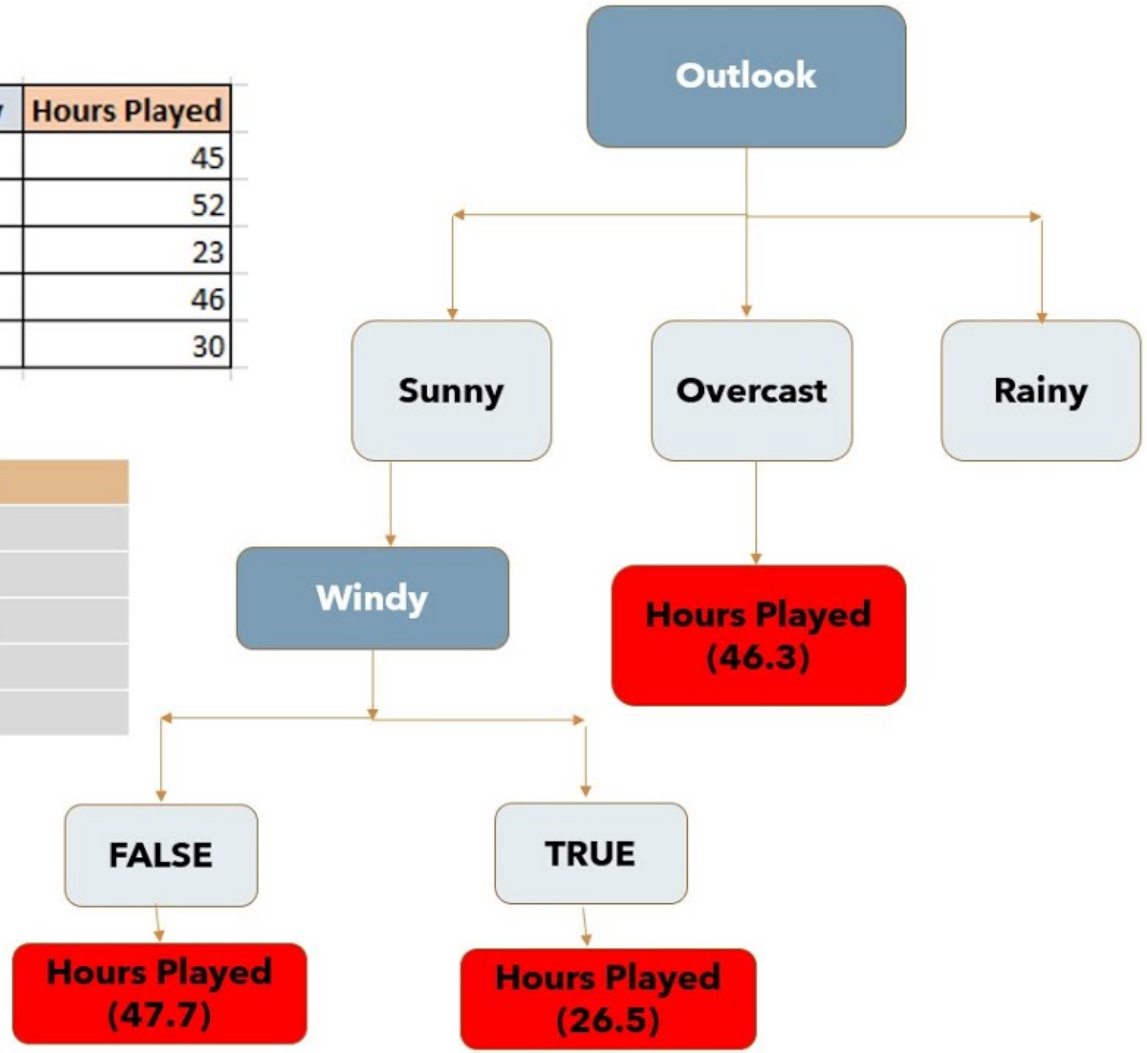

animal_tree
Has feathers?
True
False
Can fly?
Has finns?
True
False
True
False
Hawk
Penguin
Dolphin
Bear


age
young
middle-aged
senior
Student?
yes
Credit_rating?
no
yes
fair
excellent
no
yes
no
yes

Decision Tree

Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30

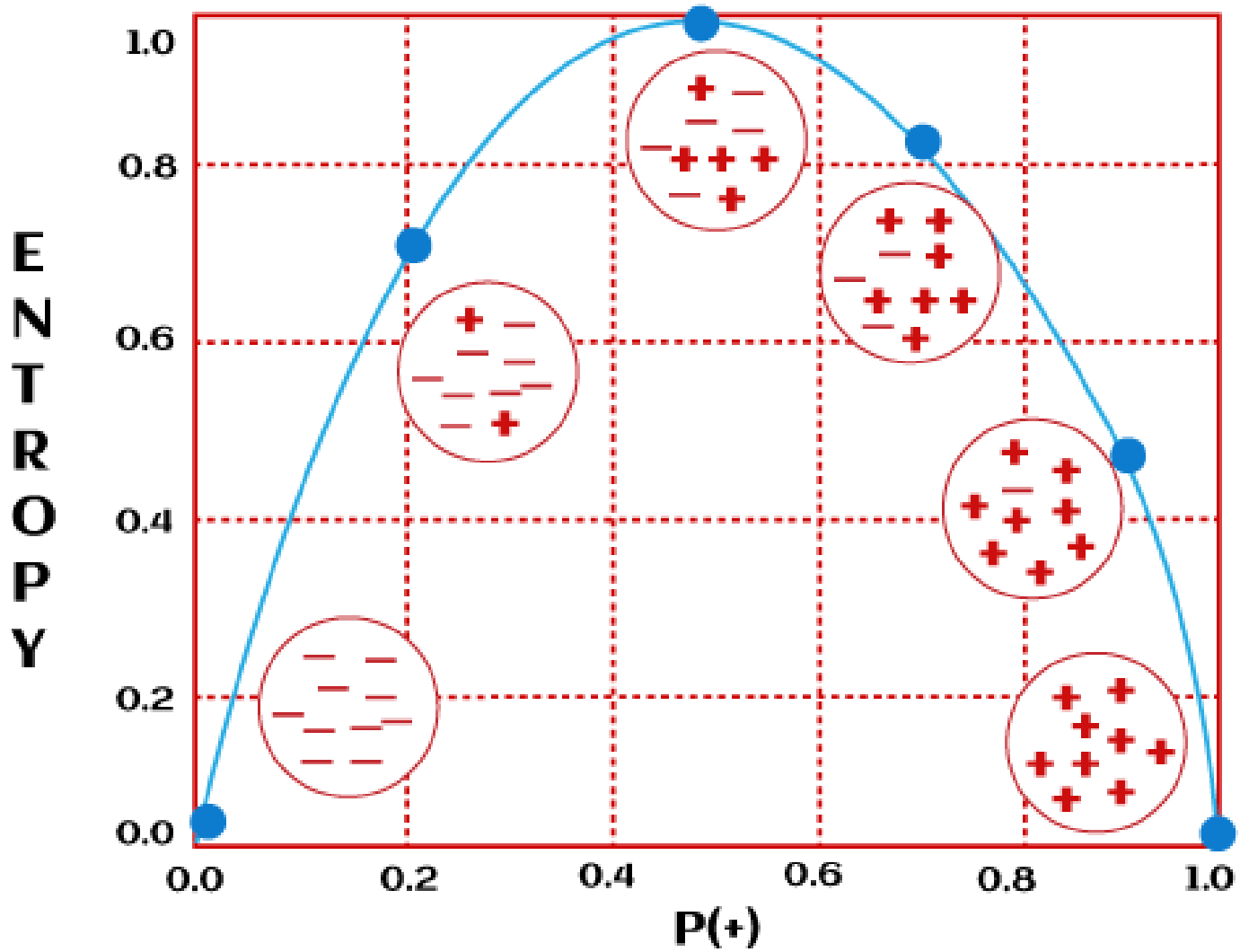
Windy
FALSE
FALSE
FALSE
TRUE
TRUE



¿Cómo construimos los árboles de dección?
¿Cómo seleccionamos la jerarquía/orden de los atributos?

- ID3: utiliza la entropía y la ganancia de información
- C4.5: es una extención de ID3, utiliza una razón de ganancia en lugar de la ganancia de información para mejorar el manejo y selección de atributos que tienen muchos valores posibles y el manejo de atributos numéricos continuos.
- CART: utiliza la impureza de Gini

$$E = -\sum_{i=1}^{N} P_i \log_2 P_i$$



Ejemplo de Entropia: Conjunto de datos con un atributo homogeneo. Entropia=0 Nos indica que hay mayor certeza en saber qué valor pueda tomar ese atributo.

No.	Nacionalidad
1	Mexicana
2	Mexicana
3	Mexicana
4	Mexicana
5	Mexicana
6	Mexicana
7	Mexicana
8	Mexicana
9	Mexicana
10	Mexicana

Ejemplo de Entropia: Conjunto de datos con un atributo más heterogeneo. Entropia=1 Nos indica que hay mayor incertidumbre en saber qué valor pueda tomar ese atributo.

No.	Nacionalidad
1	Mexicana
2	Mexicana
3	Mexicana
4	Mexicana
5	Mexicana
6	Colombiano
7	Colombiano
8	Colombiano
9	Colombiano
10	Colombiano

Ganancia de información: cuantifica la información de las instancias con respecto a la entropía (incertidumbre) causada por un conjunto de eventos (los atributos de un dataset).

$$IG(D, A) = H(D) - H(D|A)$$

- IG(D,A) = ganancia de información de la característica A en el dataset D
- H (D) es la entropia del dataset D
- H (D|A) es la entropía del dataset D dado el atributo A

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInfo}(S, A)}$$

- Gain(S,A) = ganancia de información de la característica A en el dataset S
- SplitInfo(S,A)= la entropia del dataset S dado el atributo A

$$\text{Gini(A)} = 1 - \sum_{i=1}^{n} p_i^2$$

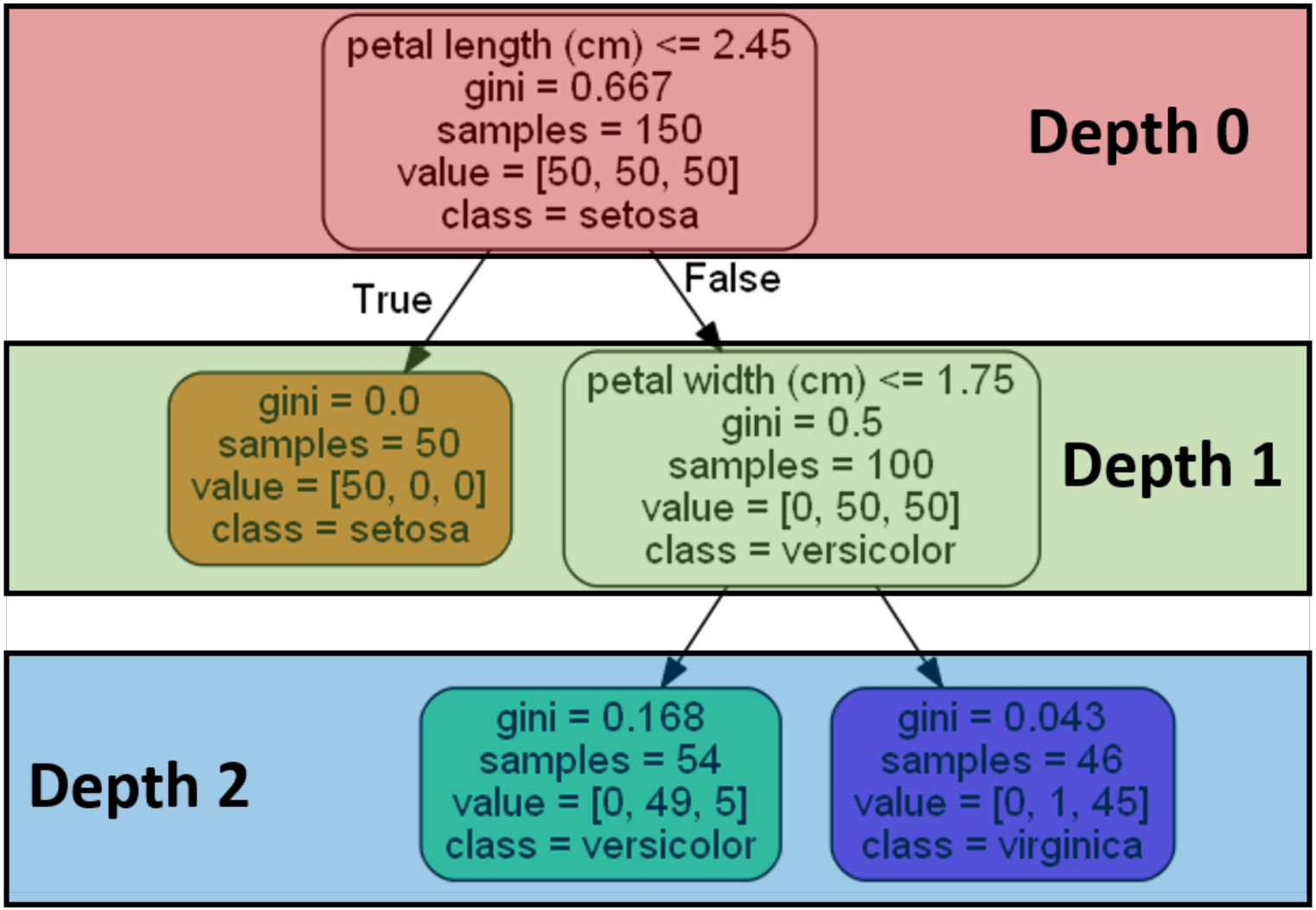
- p = probabilidad
- n= cantidad de valores posibles que puede tomar el atributo

Impureza de Gini

	Count		Probability		Gini Impurity
	n_1	n_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Node A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Node B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Node C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

$$\triangle Gini(A) = Gini(D) - Gini_A(D)$$

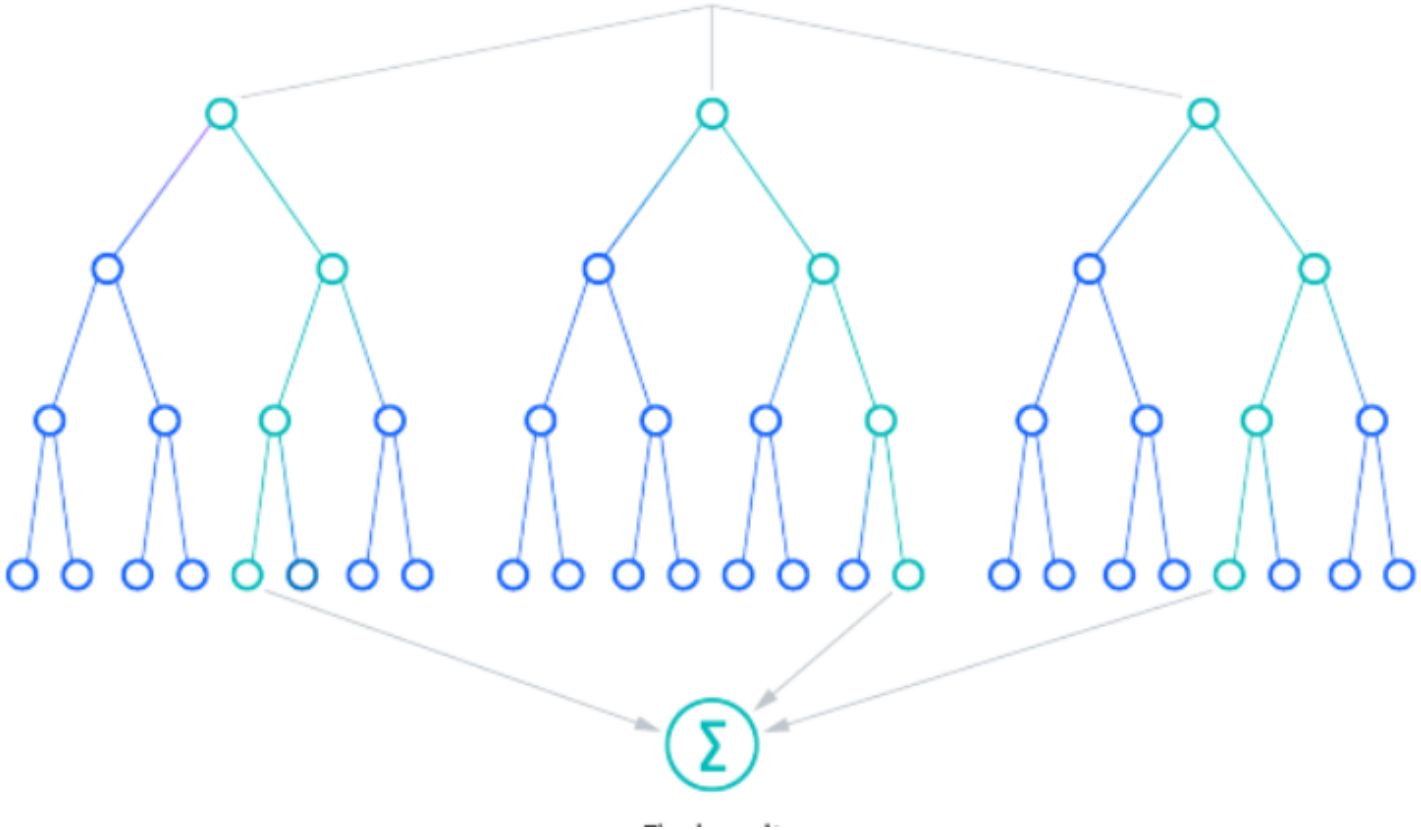
- ID3 y C4.5: se basan en la reducción de la desviación estandar
- CART: Utiliza el error cuadrático medio (mean squared error-MSE)


petal length (cm) <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
Depth 0
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
Depth 1
Depth 2
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica

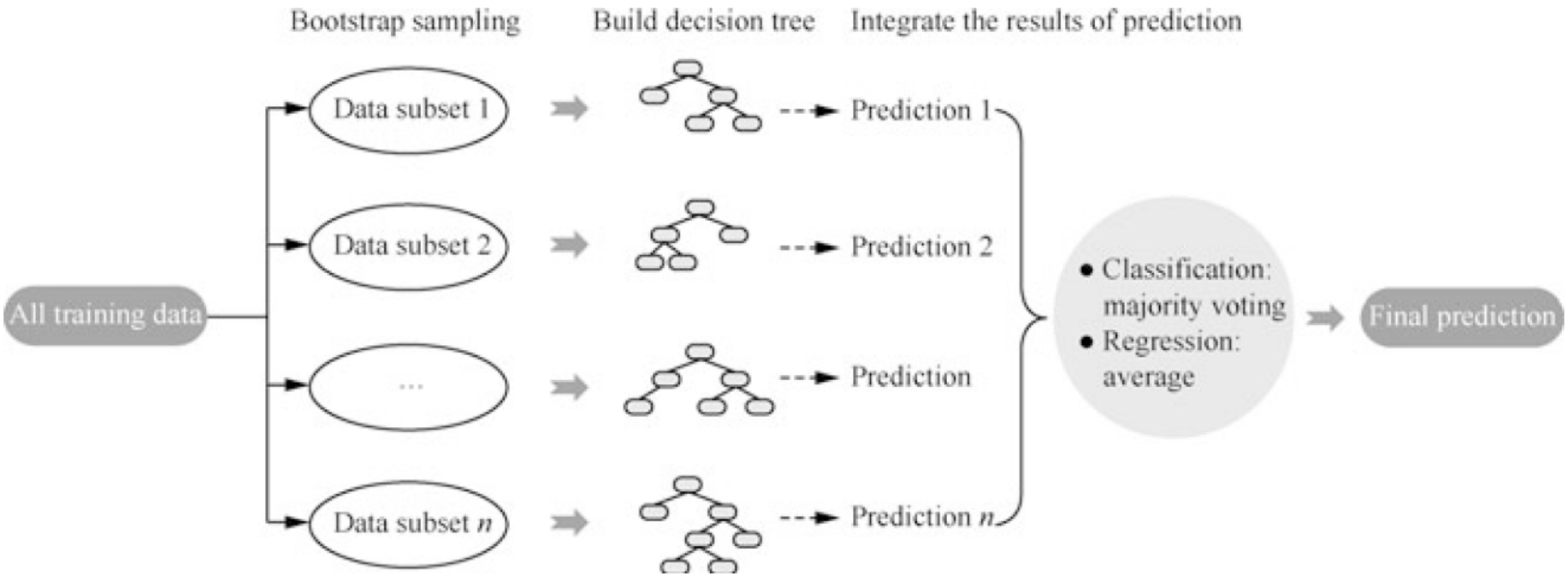
Son aquellos algooritmos que entrenan multiples modelos de ML y los combinan para optimizar el desempeño de 1 solo modelo ML. Se clasifican en dos tipos:

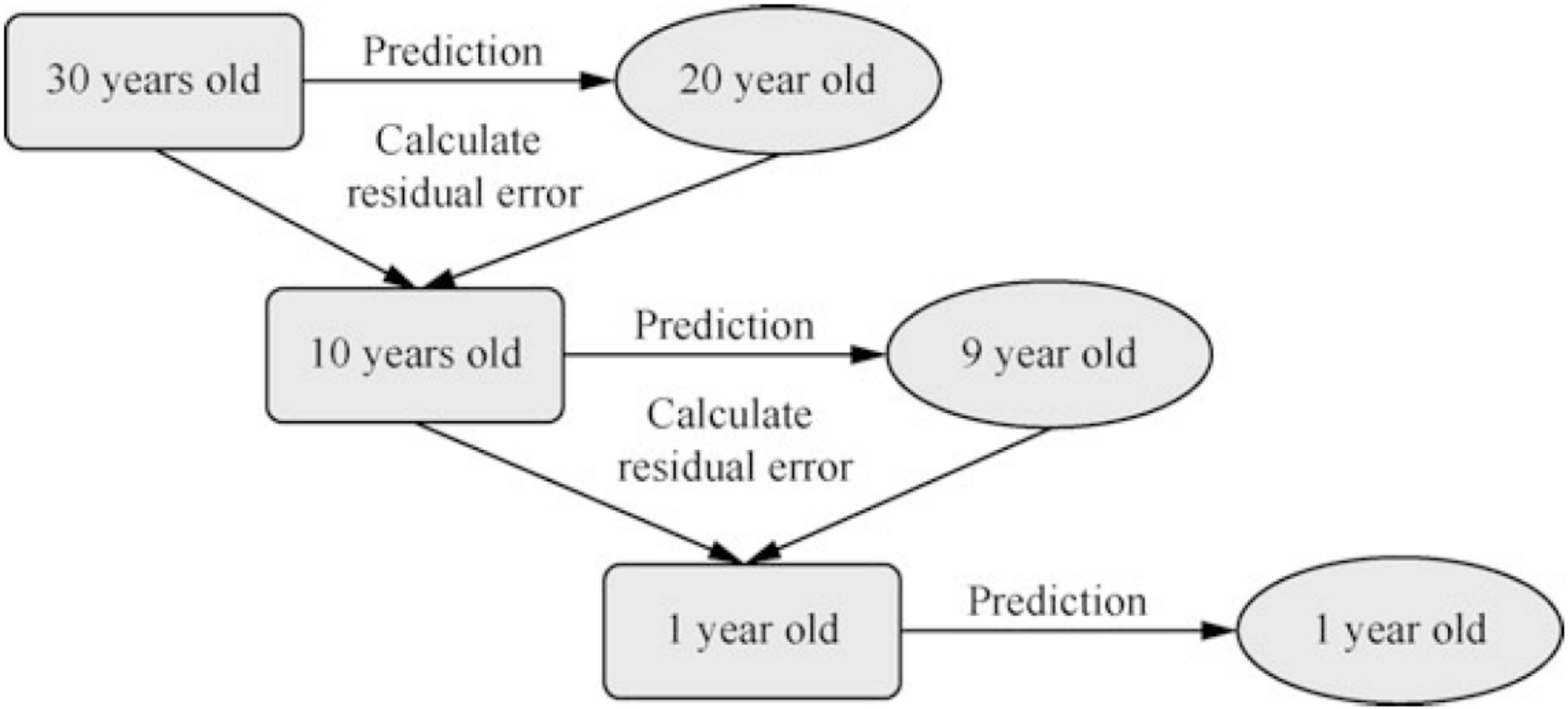
- Bagging: construye de forma independiente n modelos de ML e implementa una técnica para fusionar los n resultados. Ejemplo: Random Forest
- Boosting: combina los n modelos de ML de forma secuencial y de forma gradual reduce la desviación de la predicción del algortimo. Ejemplo: Gradient Boosting Decision Tree (GBDT)

Es un algoritmo tipo ensamble que utiliza la colección de algoritmos de árboles de decisón para optimizar el resultado de un solo algoritnmo (single algorithm).



Random Forest

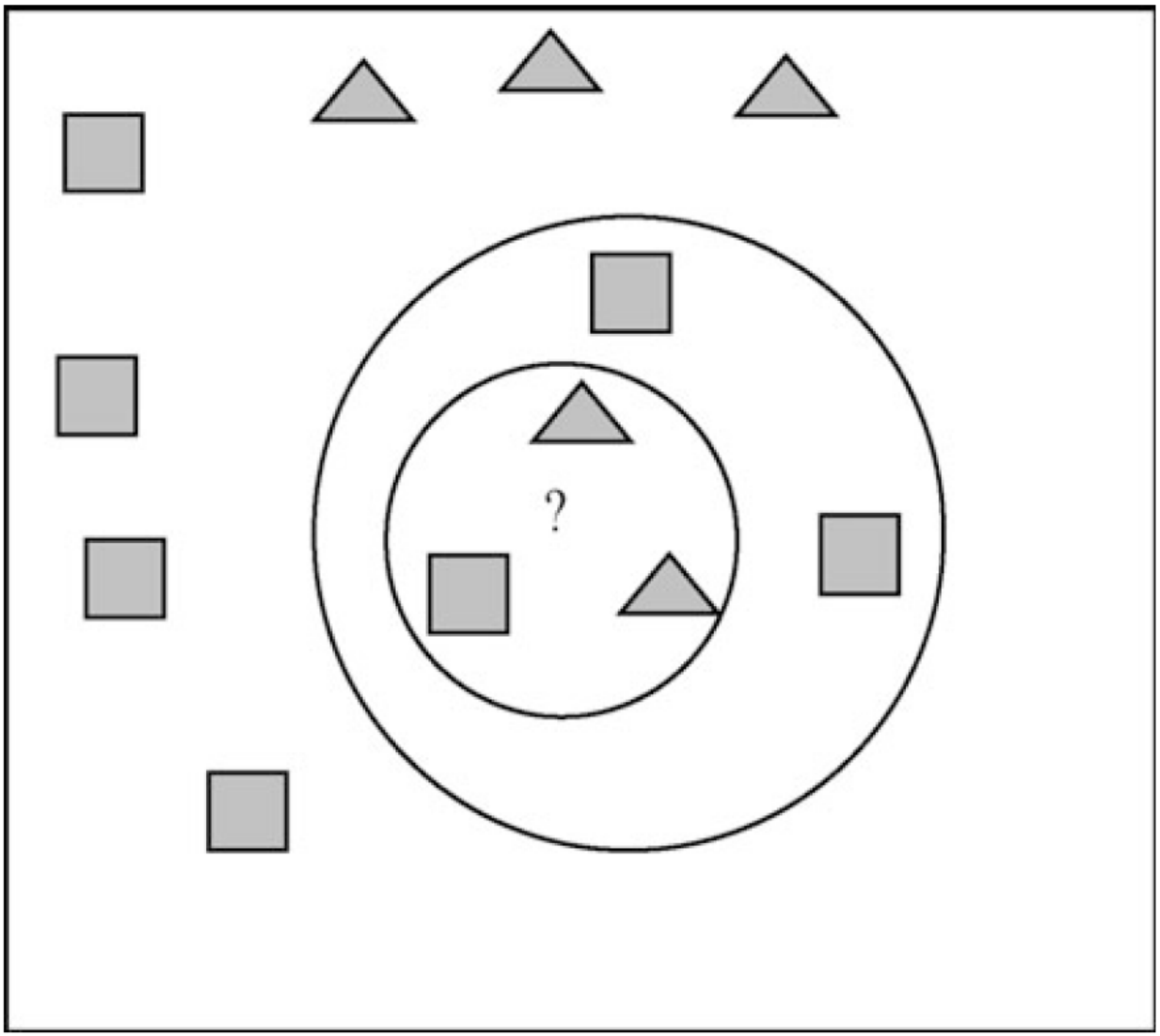


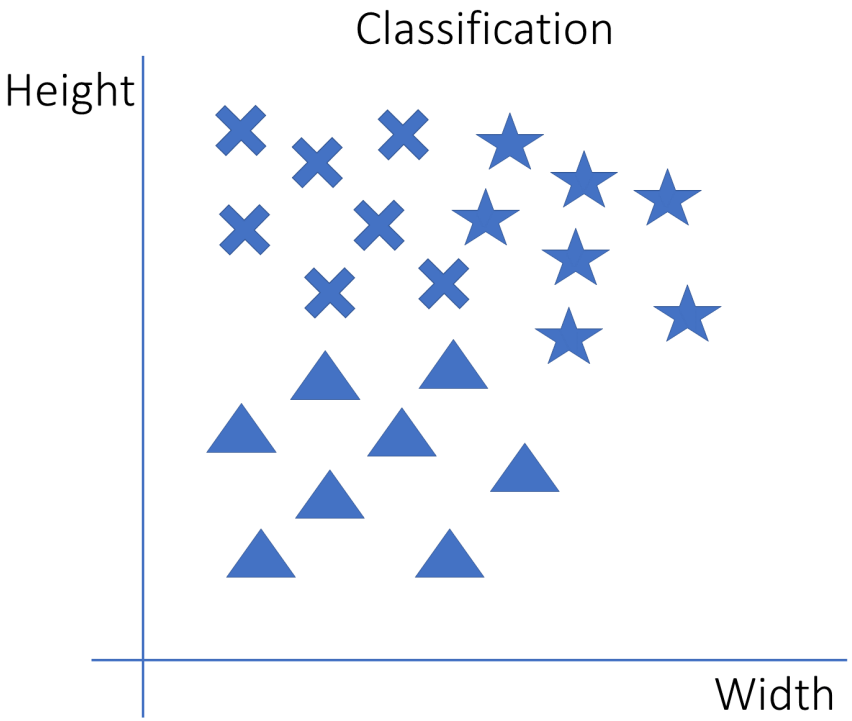

30 years old
Prediction
20 year old
Calculate
residual error
10 years old
Prediction
9 year old
Calculate
residual error
1 year old
Prediction
1 year old

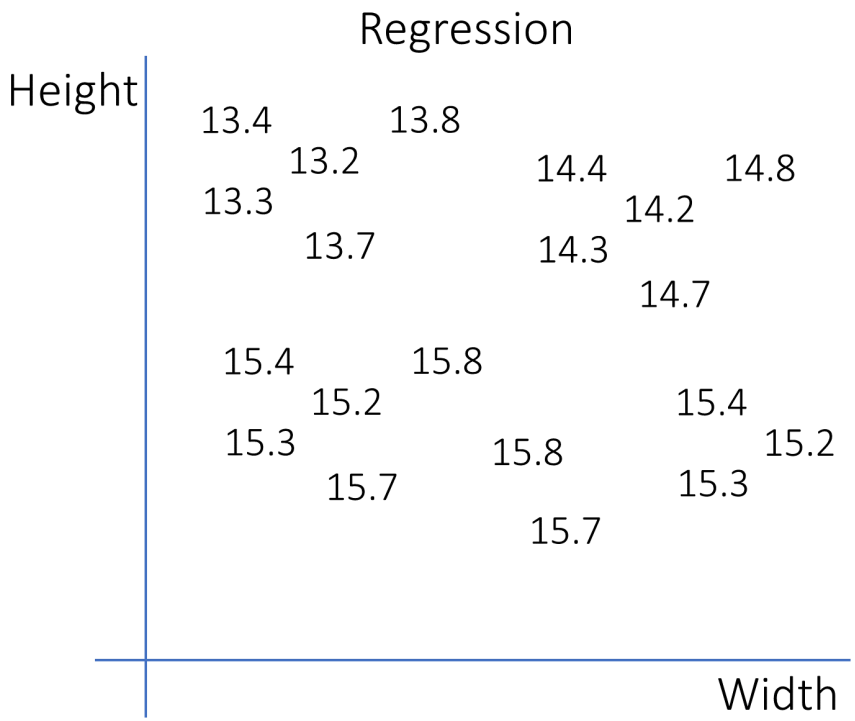
Es un algoritmo supervisado que puede resolver problemas de clasificación y regresión, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual.

- Clasificador KNN: elige la clase con la mayoría de votos (majority voting method).
- Algoritmo de regresión KNN: elige el valor promedio (average method).

Ejemplo: Clasifica la nueva muestra ? como triangulo o cuadrado.
Considera k=3 y k=5.




Classification
Height
Width


Regression
Height
13.4
13.8
13.2
14.4
14.8
13.3
14.2
13.7
14.3
14.7
15.4
15.8
15.2
15.4
15.3
15.8
15.2
15.7
15.3
15.7
Width

¿KNN como encuentra a los k vecinos más cercanos?

Distancia Euclidiana:

$$\text{dist}_{\text{Euclidiana}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$

Distancia Manhattan:

$$\text{dist}_{\text{Manhattan}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d}|x_i - y_i|$$

Distancia Hamming:

$$\text{dist}_{\text{Hamming}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d}\mathbf{1}_{\{x_i \neq y_i\}}$$

Algorithm 1 kNN Classification

procedure kNN-Classification($\mathcal{D}, \mathbf{x}_j, k$)
  for $i = 1$ to n **do**
    $dist_i \leftarrow \sqrt{\sum_{l=1}^{d}(x_{i_l} - x_{j_l})^2}$
  end for
  Sort distances and select indices of the k smallest: I_k
  Count occurrences of each class label among indices in I_k: C
  $\hat{y}_j \leftarrow$ class label with highest count in C
  return $\hat{y}_j$
end procedure

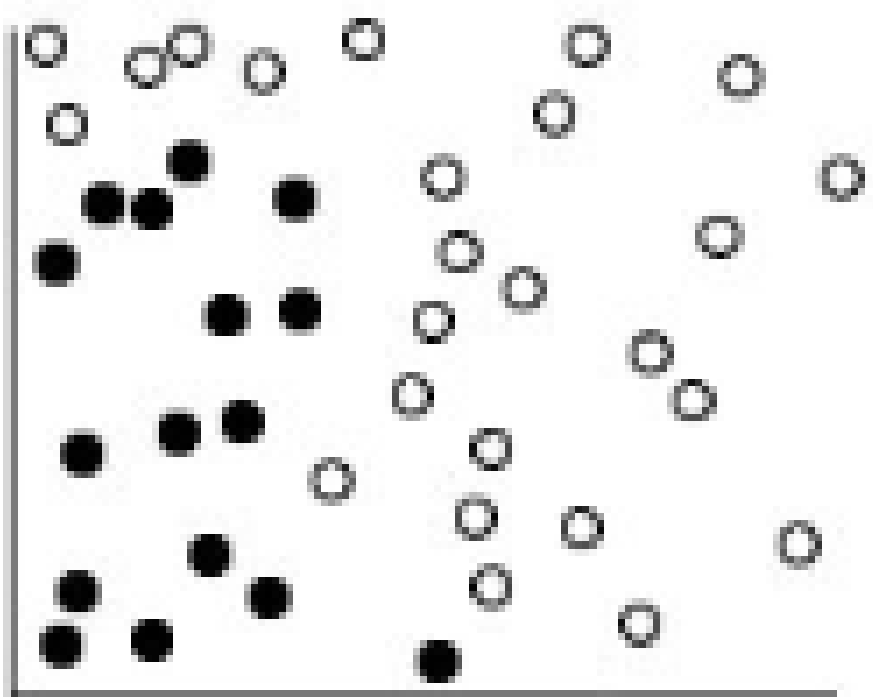
Algorithm 1 kNN Regression

procedure kNN-Regression($\mathcal{D}, \mathbf{x}_j, k$)
  for $i = 1$ to n **do**
    $dist_i \leftarrow \sqrt{\sum_{l=1}^{d}(x_{i_l} - x_{j_l})^2}$
  end for
  Sort distances and select indices of the k smallest: I_k
  $\hat{y}_j \leftarrow \frac{1}{k}\sum_{i \in I_k} y_i$
  return $\hat{y}_j$
end procedure

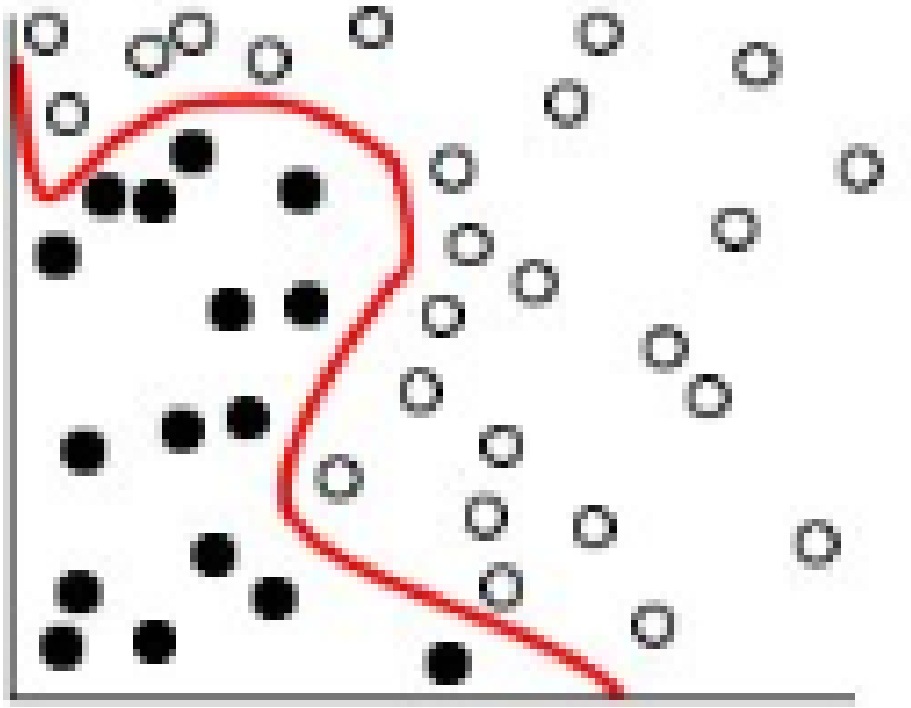
Algoritmo supervisado para tareas de clasificación y regresión (Pero a este algoritmo se le conoce como Support Vector Regression).

- Busca el hyperplano más óptimo en el espacio de características de modo que se obtenga una optimización global y las expectativas en todo el espacio muestral satisfagan un cierto límite superior con una cierta probabilidad.

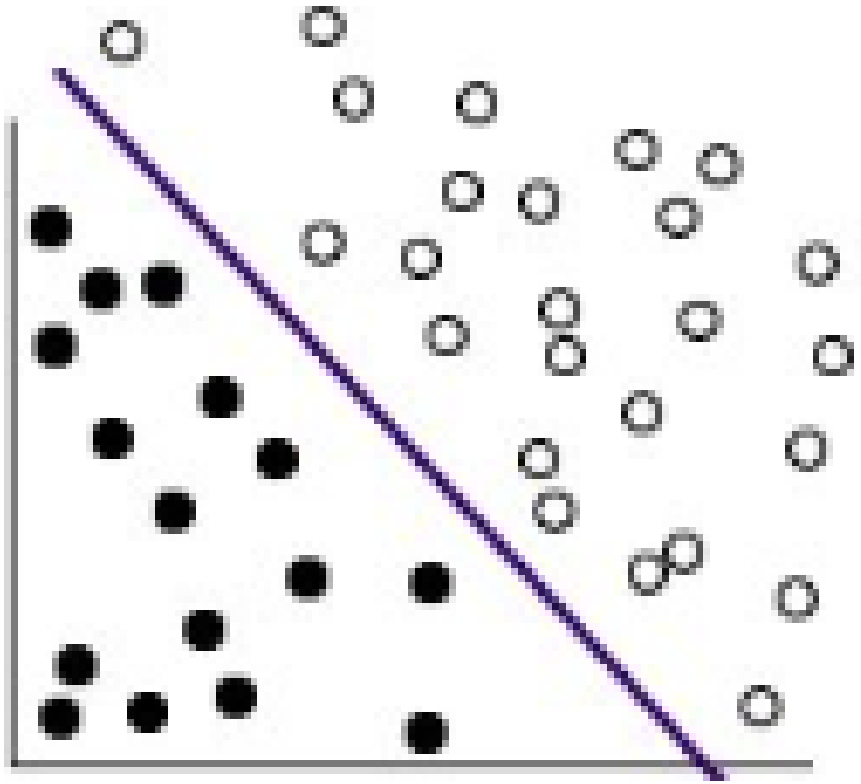
Conjunto de datos original



Las dos categorías se pueden separar con una curva:



Tras la transformación, el límite entre las dos categorías se puede definir por un hiperplano:



¿Cómo realizar la transformación cuando tenemos un fenómeno no lineal?

Funciones KERNEL

Un kernel es una función que transforma los datos de entrada a un espacio de mayor dimensión (donde sí sean linealmente separables), permitiendo la clasificación lineal en este nuevo espacio.

Tipos Comunes de Kernels: Kernel Lineal:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

Kernel Polinomial:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$$

donde c es una constante y d es el grado del polinomio.

Tipos Comunes de Kernels: Kernel Gaussiano (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

donde σ es un parámetro que determina el ancho del kernel.
Kernel Sigmoide:

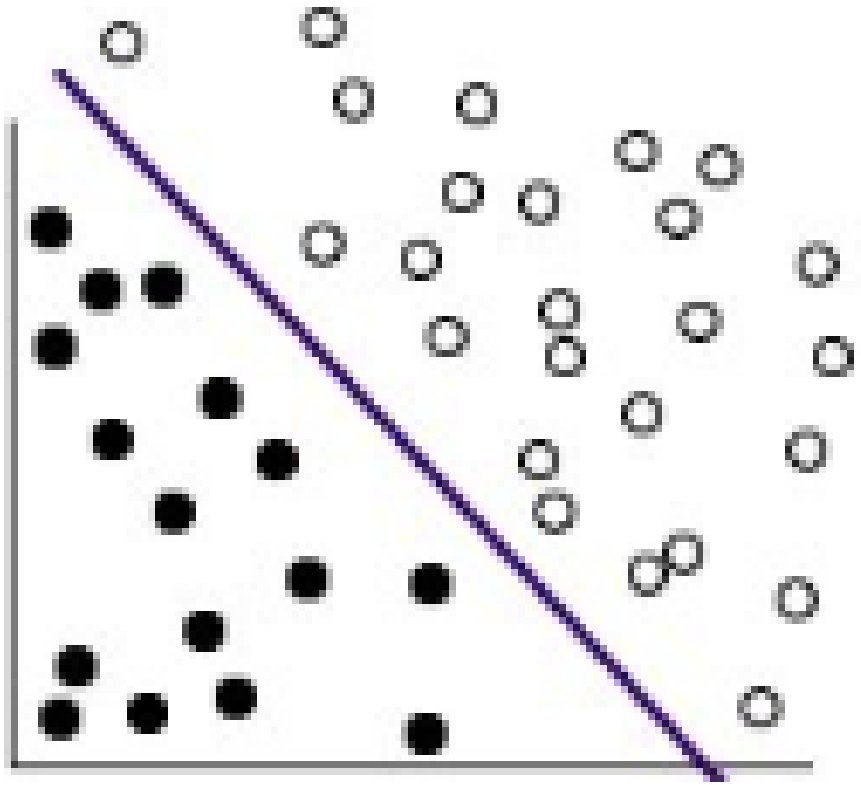
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j + c)$$

donde κ y c son parámetros del kernel.

- En lugar de buscar el hyperplano más óptimo como SVM, su objetivo es buscar la función más óptima (con el menor error de predicción) para la predicción de datos numéricos.

Algoritmos NO supervisados

Método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez.
Asume que tenemos n muestras y cada uno tiene p características/atributos. Entonces, tenemos un espacio de p dimensiones.



Eigenvectors

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} x \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4\, x \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Los eigenvectors de una matriz son todos aquellos vectores que, al multiplicarlos por dicha matriz, resultan en el mismo vector o en un múltiplo entero del mismo.

Eigenvalue

Cuando se multiplica una matriz por alguno de sus eigenvectors se obtiene un múltiplo del vector original, es decir, el resultado es ese mismo vector multiplicado por un número. Al valor por el que se multiplica el eigenvector resultante se le conoce como eigenvalue.

¿Cómo calculamos los componentes principales?

Cada componente principal es una combinación lineal de las variables originales y se construye de tal manera que captura la máxima varianza posible de los datos.

Cada componente principal Z_i se obtiene como una combinación lineal de las p variables originales $X_1, X_2, \ldots, X_p$. La primera componente principal (Z_1) se define como:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

Aquí, $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ son los **loadings**. Estos coeficientes indican cuánto contribuye cada variable original a la componente principal.
La combinación lineal está normalizada, lo que implica que:

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

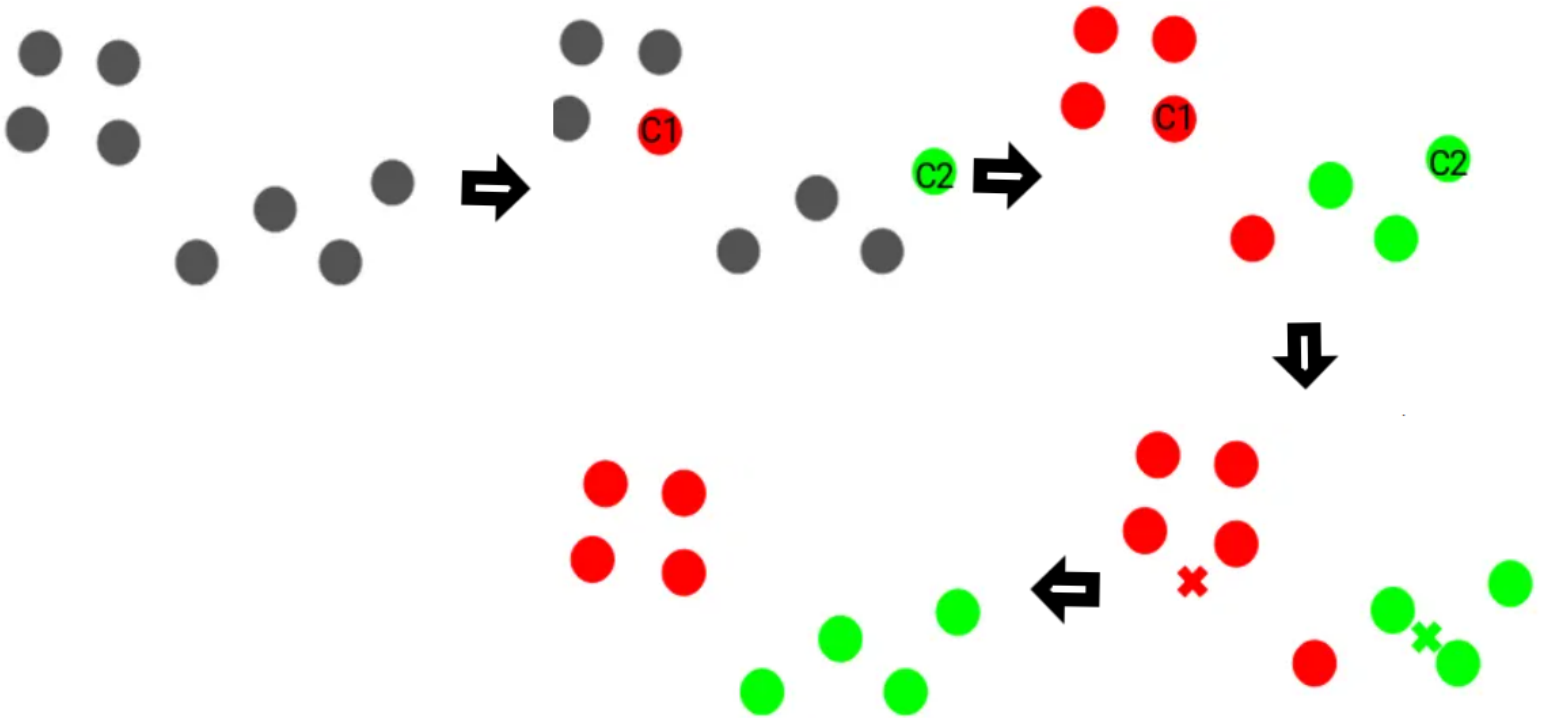
- Centralizar las variables: Restar a cada valor de las variables la media de su respectiva variable para obtener variables con media cero.
- Optimizar la maxima varianza: encontrar los valores de los loadings que maximicen la varianza de la combinación lineal (calcular la matriz de covarianza).
- Calcular iterativamente las componentes: Despues de calcular la primera componente principal, se calcula la segunda.
- Calcular los eigevectores y eigvalores: son los loadings (coeficientes) de los componentes principales a partir de la matriz de covarianza.

El orden de importancia de las componentes principales está determinado por la magnitud del eigenvalue asociado a cada eigenvector. Las componentes principales con eigenvalores más grandes capturan más varianza de los datos y, por lo tanto, son más importantes.

Algoritmo no supervisado que divide el conjunto de datos en subgrupos o clusters.

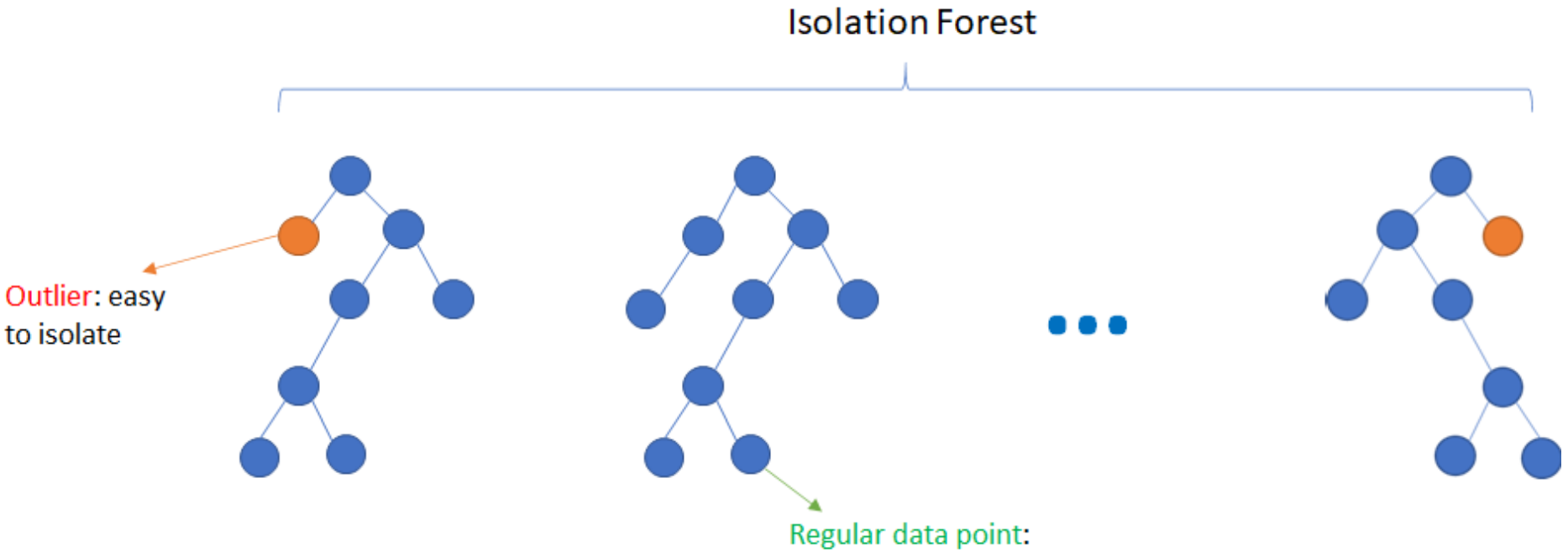
- Selecciona k puntos al azar y los trata como los centroides iniciales.
- Construye los primeros k clusters con respecto a estos centroides iniciales: los datos se asignan en aquellos clusters donde la distancia al centroide sea la menor (se van con los centroide más cercanos).
- Una vez asignados todos los datos a un cluster, se recalculan los centroides de cada cluster (la posición del centro).
- Se repite la asginación de los datos a los clusters más cercanos.
- El algoritmo se detiene hasta que los clusters ya no cambien o hasta alcanzar un máximo de iteraciones.

K-means clustering

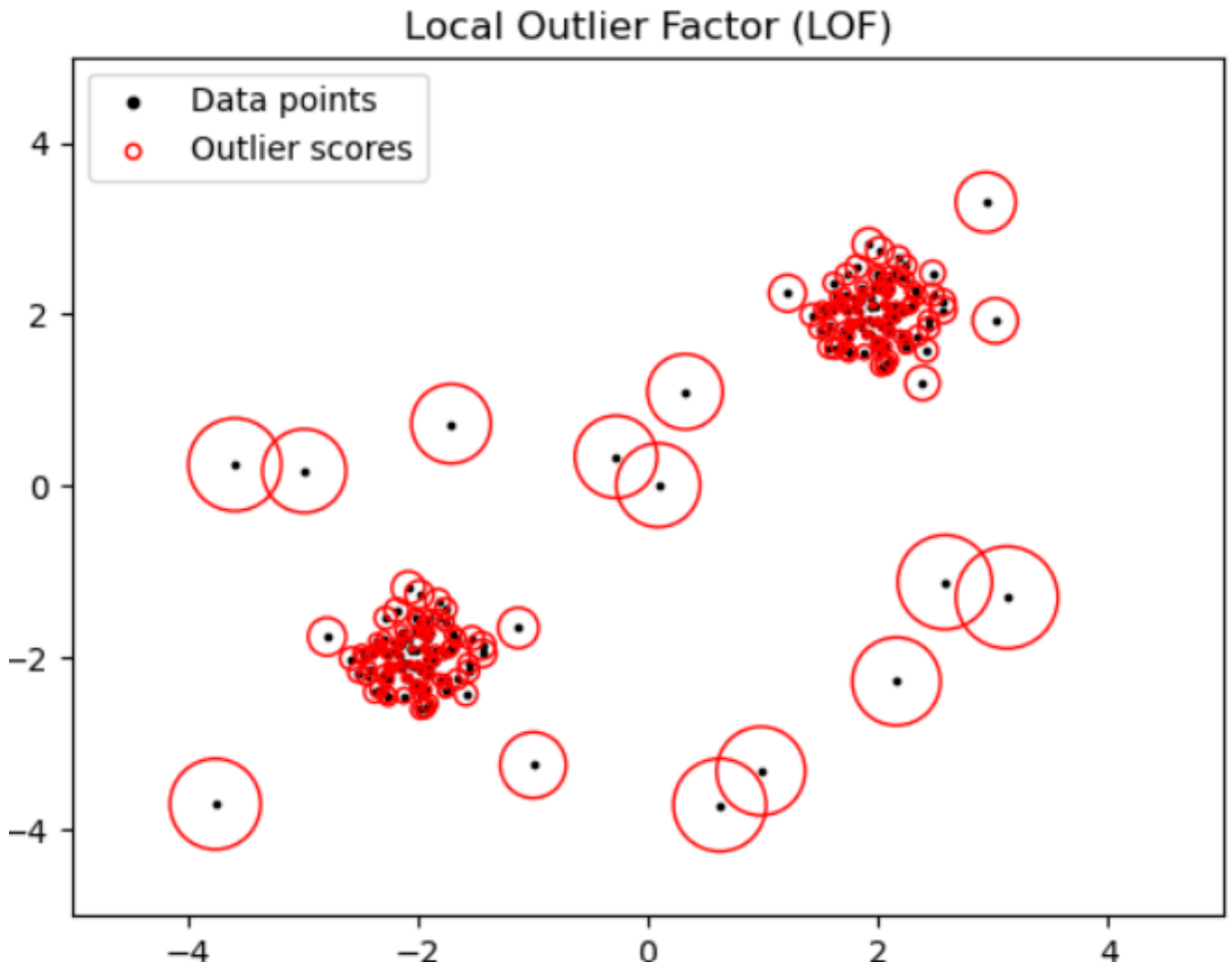


Algoritmo no supervisado basado en árboles de decisión para aislar datos anómalos. La construcción del árbol se basa en:

- Seleccionar una carcarística/atributo al azar. Escoger un valor aleatorio (umbral) entre el valor mínimo y máximo del atributo seleccionado.
- Los datos se dividen con base al valor umbral. El proceso se repite recursivamente creando ramas hasta encontrar una profundidad predefinida o cada subcojunto contenga al menos un outlier.



Algoritmo no supervisado para detectar anomalías. Considera a los vecinos más cercanos para determinar a los outliers.



- Mide la densidad local de un punto con respecto a sus k vecinos más cercanos (LRD=Local Reachability Density).
- Calcula la puntuación LOF, considerando su densidad local con respecto al LRD de sus k vecinos más cercanos.
- Aquellos puntos (datos) cuyas densidades son tan grandes como las densidades de sus vecinos, obtienen una puntuación lof de aproximadamente 1,0. Los que tienen una densidad local baja (anomalías), obtendrán valores LOF mayores a 1.

$$LRD_k(x) = 1/\left(\frac{\sum_{o \in N_k(x)} d_k(x,o)}{|N_k(x)|}\right)$$

$$LOF(x) = \frac{\sum_{o \in N_k(x)} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|}$$

Algoritmos Semi-supervisados

Es una versión de SVM diseñado para detectar anomalías. Las cuales son aquellas distancias que se desvían significamente de la norma. Por lo tanto, construye un hyperplano que encapsule aquellas instancias consideradas normales, y esa clase es la unica que considera en su etapa de entrenamiento.

Algoritmos paramétricos vs no paramétricos

- Algoritmos paramétricos: Asumen que los datos siguen una distribución determinada.
- Algoritmos no paramétricos: No asumen distribuciones para los datos.

- Modela 2 algoritmos en python usando las librerias sckit-learn y tú dataset.
- Aplica lo visto en clase:
  - Train-Test / Train-Test-Validation / k-fold cross validation
  - Grid Search /Random search / Default
  - Evaluar el desempeño con las métricas adecuadas
- Escribe un reporte de las actividades realizadas.