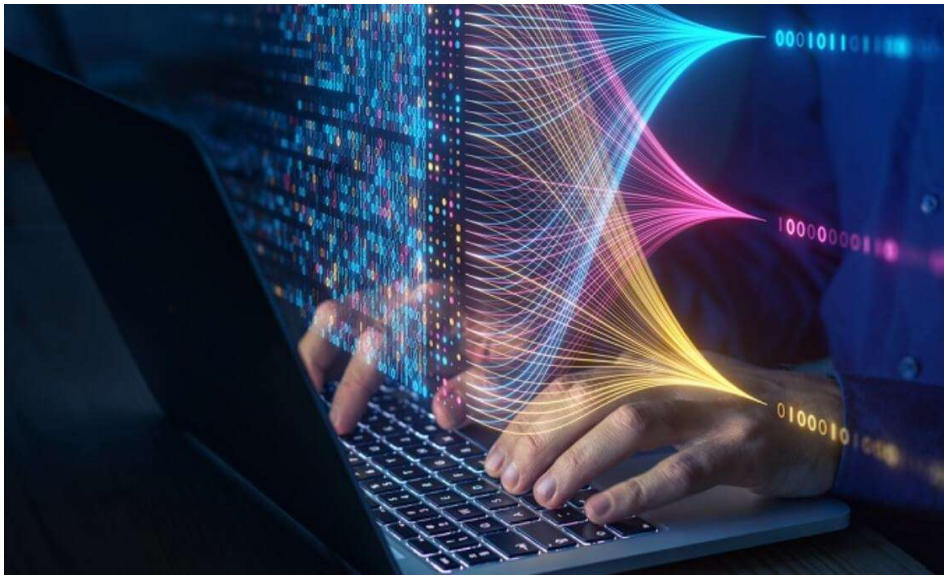
Técnicas de inteligencia artificial

Adriana Cervantes Castillo



SEMANA 3-4: ÁRBOLES DE DECISIÓN

unir LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

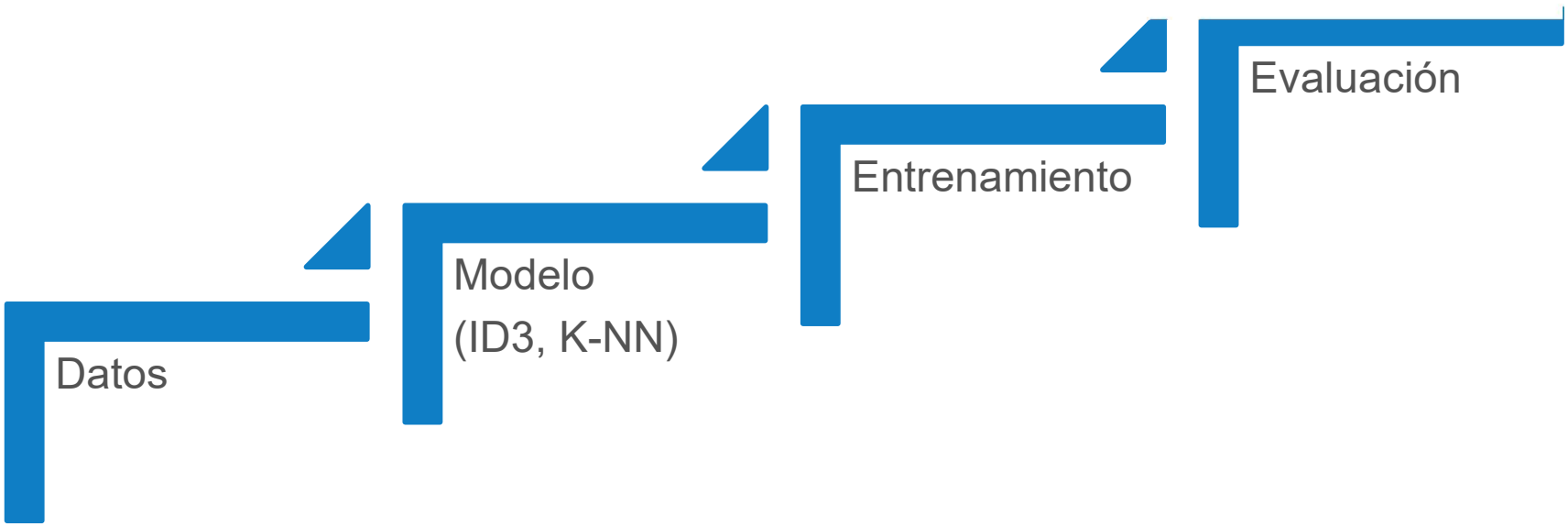
Contenido

- Clasificación
- Árboles de decisión
- Medidas de rendimiento

Clasificación

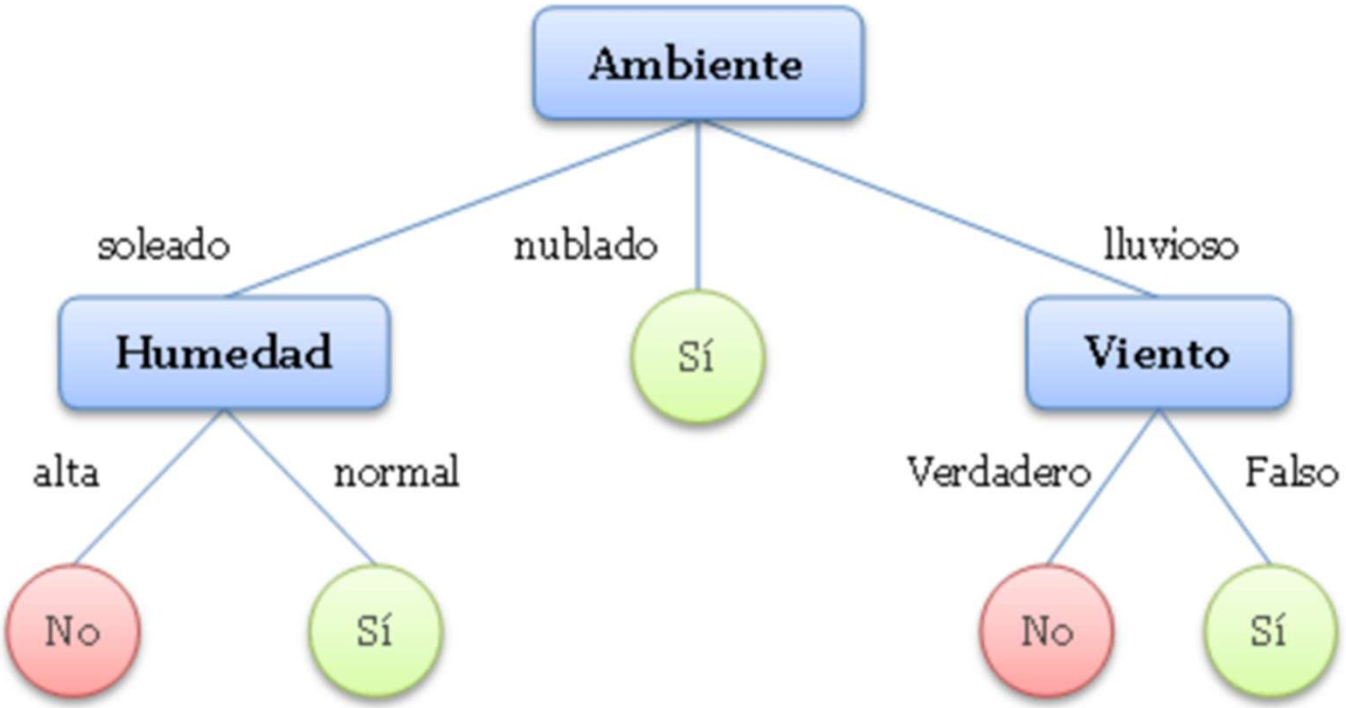
- Datos etiquetados – ejemplos – instancias
- Entrena un modelo
- Asignar una etiqueta a una nueva instancia

Id	Ambiente	Temperatura	Humedad	Viento	*Clase*
E1	soleado	Alta	Alta	Falso	No
E2	soleado	Alta	Alta	Verdadero	No
E3	nublado	Alta	Alta	Falso	Sí
E4	Lluvioso	Media	Alta	Falso	Sí
E5	Lluvioso	Baja	Normal	Falso	Sí
E6	Lluvioso	Baja	Normal	Verdadero	No
E7	Nublado	Baja	Normal	Verdadero	Sí



Árbol de decisión

- Estructura de datos formada por ramas y nodos



Donde:

- Los nodos internos representan las características o propiedades a considerar para tomar una decisión.
- Las ramas representan la decisión en función de una determinada condición
- Los nodos finales (hojas) representan el resultado de la decisión.

Conjunto de datos

Id	Ambiente	Temperatura	Humedad	Viento	*Clase*
E1	soleado	Alta	Alta	Falso	No
E2	soleado	Alta	Alta	Verdadero	No
E3	nublado	Alta	Alta	Falso	Sí
E4	Lluvioso	Media	Alta	Falso	Sí
E5	Lluvioso	Baja	Normal	Falso	Sí
E6	Lluvioso	Baja	Normal	Verdadero	No
E7	Nublado	Baja	Normal	Verdadero	Sí

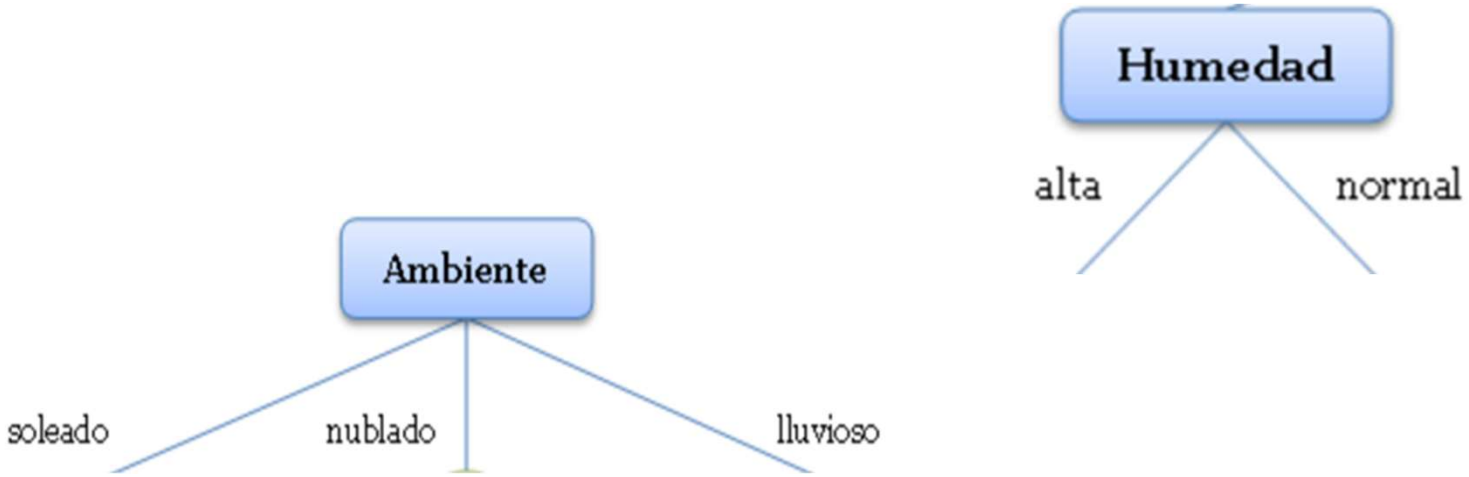
Atributos → nodos del árbol

- ¿Cómo?

¿Cuántos?

¿Cuáles?

- **Método de selección de atributos** → criterio utilizado para generar las diferentes ramas del árbol.



Métodos de selección de atributos

- **Tasa de ganancia**
- **Índice Gini**
- **Ganancia de información**

El usar un método u otro dependerá tanto de los datos de entrenamiento disponibles, como del algoritmo específico que se emplee así como de los supuestos y suposiciones que se realizan para generalizar la mejor solución encontrada (*bias*).

Algoritmo ID3

- Basado en el método de búsqueda greedy (El atributo mejor clasificador se convierte en la condición del nodo raíz que da lugar a distintas ramas, una por cada valor posible del atributo)
- Construye los árboles *top-down* (de arriba a abajo)
- Utilizando un método de selección de atributos basado en la **teoría de la información (ganancia de información, reducción de entropía).**
- El atributo cuyo conocimiento aporta más información en la clasificación es el más útil.

Pasos para trabajar con ID3

1. Seleccionar el conjunto de Datos
2. Definir los datos que servirán de entrenamiento y los de prueba.
3. Calcular la entropía de la clase
4. Calcular la ganancia de información para cada atributo
5. Repetir hasta que o todos los ejemplos pertenecen a la misma clase, o todos los atributos están incluidos en un camino del árbol, o no quedan más ejemplos.

Árbol de decisión- Algoritmo Cart

- CART construye un árbol dividiendo iterativamente el conjunto de datos en subconjuntos más pequeños basados en una métrica de división óptima.

1. **Inicio**:
    - Se toma el conjunto de datos inicial y se considera como el nodo raíz.
2. **División recursiva**:
    - En cada paso, el algoritmo evalúa las posibles divisiones en los datos utilizando un criterio específico (ver más abajo).
    - La división óptima se selecciona y se crean dos nodos hijos.
3. **Parada**:
    - El proceso se detiene cuando se cumplen ciertos criterios, como:
        - La profundidad máxima del árbol.
        - Un número mínimo de ejemplos en los nodos terminales.
        - La mejora en el criterio de partición es insuficiente.

```
:
#Predicción con árbol de decisión Algoritmo CART
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

# Cargamos el dataset
url = "https://www.openml.org/data/get_csv/61/dataset_61_iris.arff"
dataset = read_csv(url)

# Dividimos el dataset en 80% de datos para entrenar y 20% para testear
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1, shuffle=True)

# Realizamos predicciones con el dataset de validación
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Evaluamos las predicciones, en primer lugar la precisión obtenida
print("accuracy = ")
print(accuracy_score(Y_validation, predictions))
#> 0.9666666666666667
```

Medidas de rendimiento

Para poder clasificar una clase hay que tomar en cuenta:

- **Verdaderos positivos** (TP – *True* Positive): Valores que el algoritmo clasifica como positivos y que realmente son positivos.
- **Falsos positivos** (FP – *False Positive*): Valores que el algoritmo clasifica como positivo cuando realmente son negativos.
- **Verdaderos negativos** (TN – *True Negative*): Valores que el algoritmo clasifica como negativos y que realmente son negativos.
- **Falsos negativos** (FN – *False Negative*): Valores que el algoritmo clasifica como negativo cuando realmente son positivos.

Matriz de confusión

Predicción

		Positivo	Negativo
Actual	Positivo	48	2
	Negativo	5	45

Fuente:https://www.youtube.com/watch?v=jcBL6jxec9E

```
from sklearn.metrics import confusion_matrix
y_true = [1, 1, 1, 0, 1, 0]
y_pred = [1, 1, 0, 1, 0, 0]
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
```

Accuracy

- Determinar el rendimiento de un sistema de clasificación. Indica el porcentaje de instancias clasificadas correctamente.

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

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

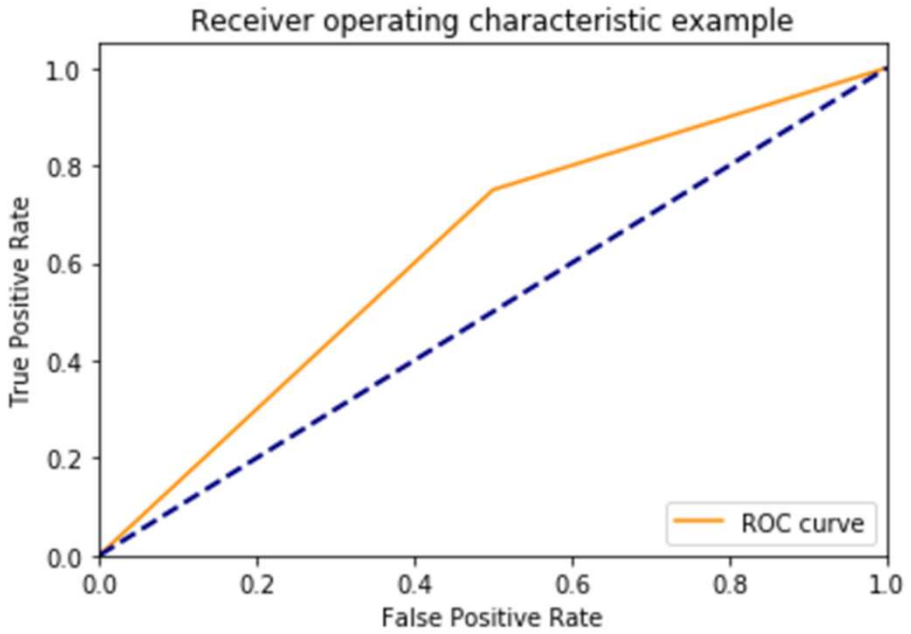
- **Recall/Sensibilidad** (*sensitivity*): también conocido como ratio de verdaderos positivos (TPR – *True Positive Rate*). Mide la capacidad del clasificador para detectar la clase en una población con instancias de esa clase, la proporción de casos positivos bien detectados. Se calcula como TP / (TP + FN).

- **Especificidad** (*specificity*): mide la capacidad del clasificador para descartar instancias que no pertenecen a la clase en una población con instancias de esa clase, la proporción de casos negativos correctamente detectados. Se calcula como TN / (TN + FP).

- **Ratio de falsos positivos** (FPR – *False Positive Rate*): Proporción de casos negativos que el clasificador detecta como positivos. De forma más concreta FP / (FP + TN).

- C**urva ROC** (*Receiver Operating Characteristic*). Traza el TPR frente al FPR a diferentes umbrales de clasificación. Al disminuir el umbral de clasificación clasifica más elementos como positivos, aumentando así tanto los falsos positivos como los verdaderos positivos.
  - El área de la curva ROC tomará valores entre 0,5 (cuando el sistema no distinga entre un positivo y un falso positivo) y 1 (el sistema clasifica perfectamente para esta clase).

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
y_true = [1, 1, 1, 0, 1, 0]
y_pred = [1, 1, 0, 1, 1, 0]
fpr, tpr, thresholds = roc_curve(y_true, y_pred)
plt.figure()
plt.plot(fpr, tpr, color='darkorange',
         label='ROC curve')
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Ensemble Learning

- También conocido como aprendizaje integrado
- Consiste en unir un conjunto de algoritmos ineficientes en los que cada uno colabora corrigiendo los errores del resto del conjunto. De esta manera, se consigue una calidad general más alta que la de los mejores algoritmos individuales que trabajan de forma aislada.

- Existen tres métodos principales dentro del aprendizaje integrado en función de cómo se combinan los diferentes algoritmos que componen cada método: Stacking, Bagging y Boosting.

Ensemble Learning

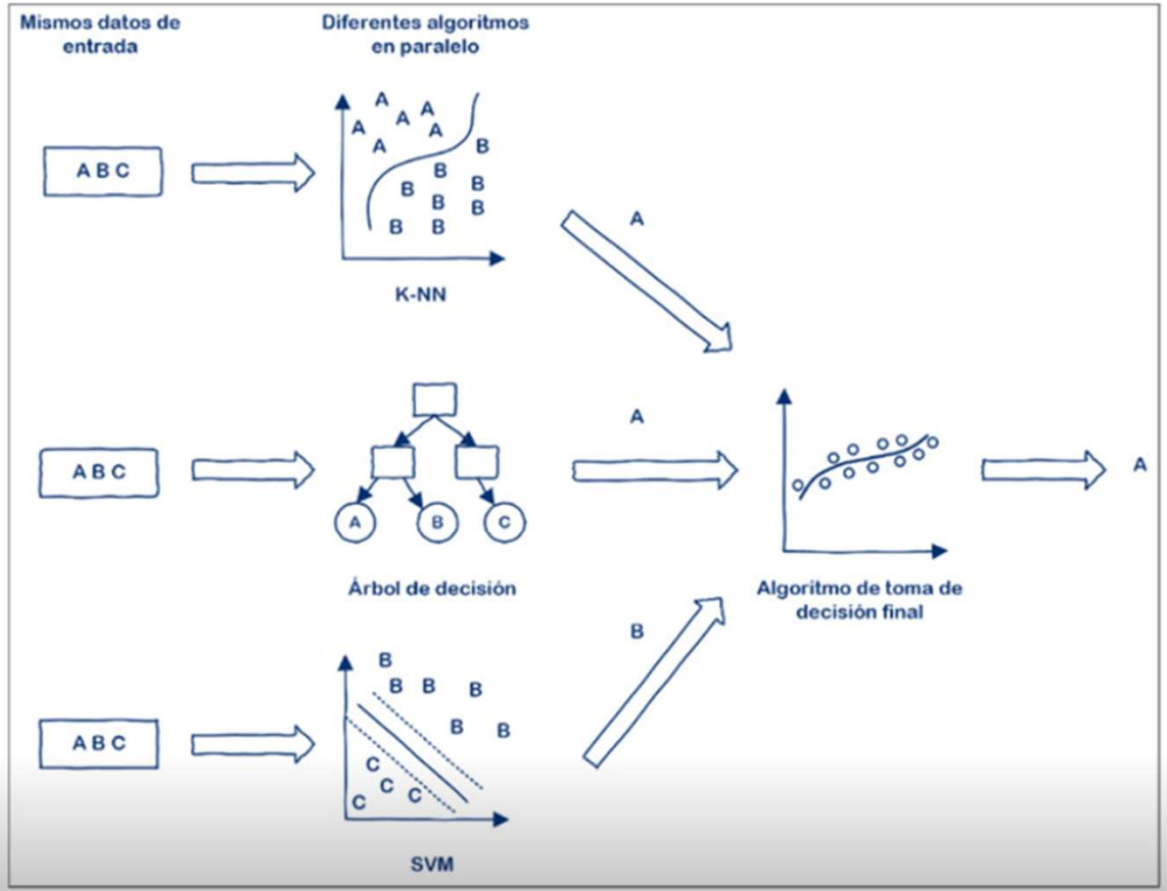
Stacking

Sencillez

Ejecutamos **diferentes** algoritmos en paralelo para realizar la misma tarea (clasificación).

Cálculo

Se promedian todas las salidas arrojadas por los diferentes modelos probados.



Fuente:https://campusvirtual.mexico.unir.net/mod/page/view.php?id=3201053&forceview=1

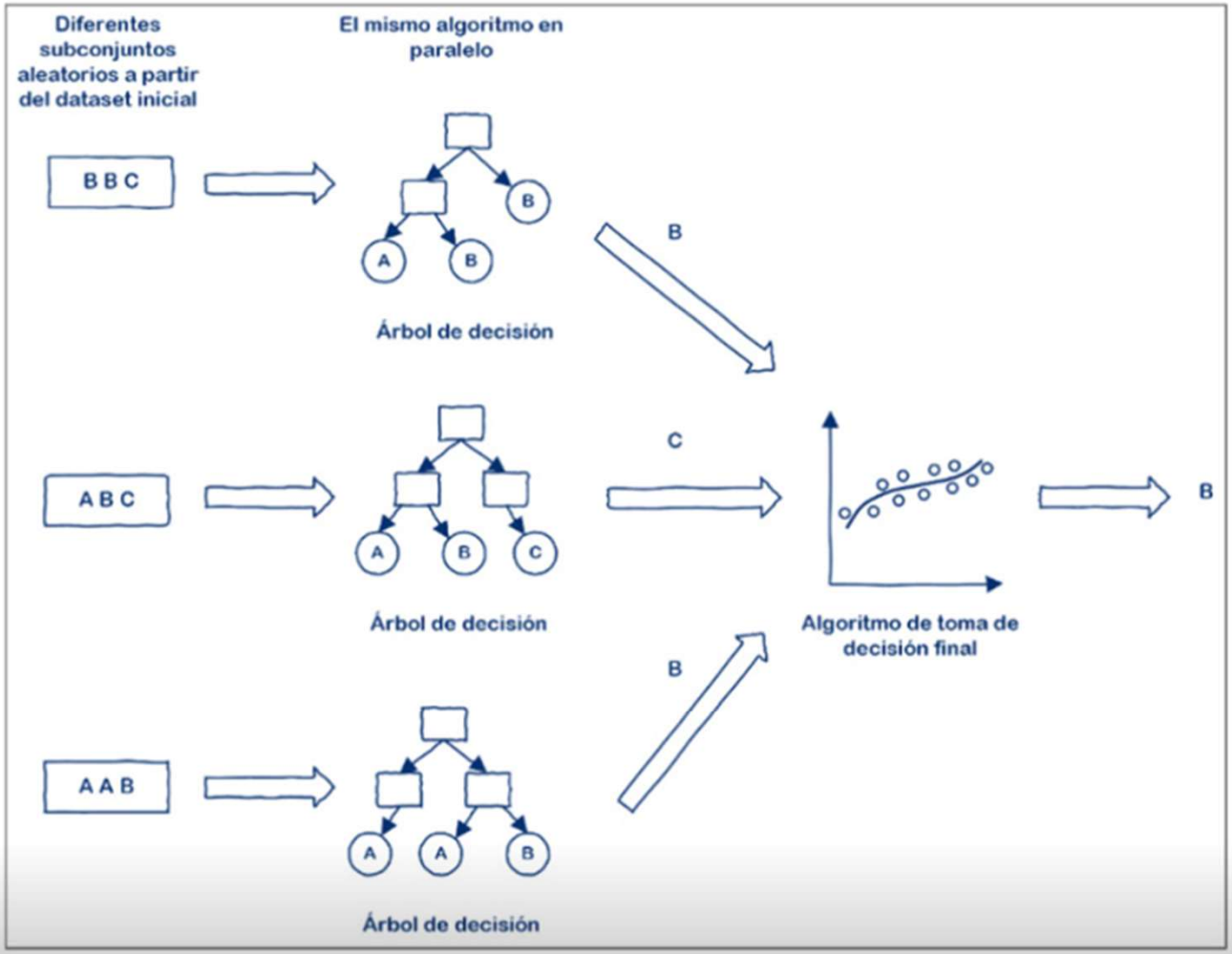
Bagging

Aleatorización

Entrenamos varios algoritmos iguales en paralelo con conjuntos aleatorios de nuestro *dataset*.

Cálculo

Se promedian todas las salidas arrojadas por los diferentes modelos probados.

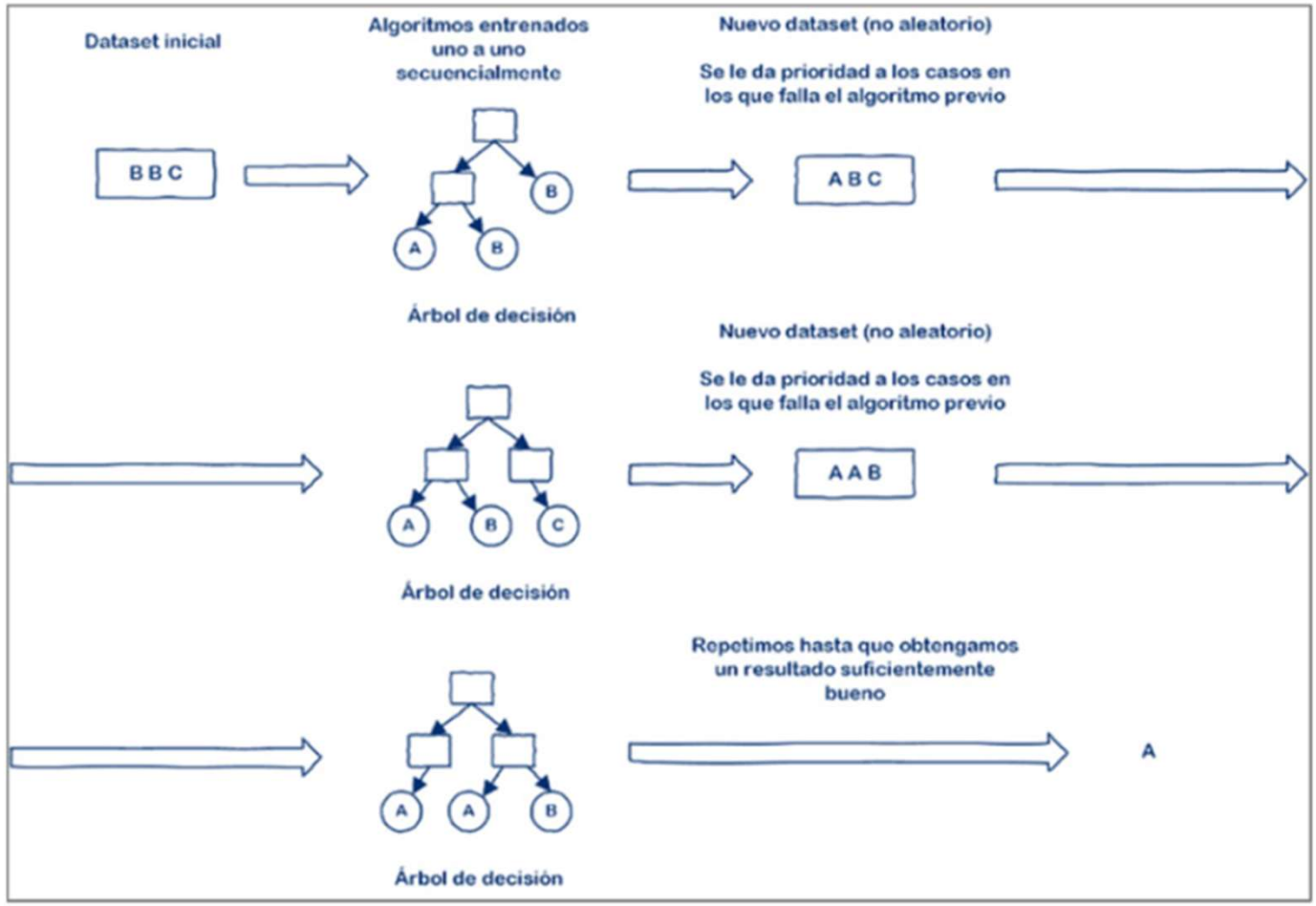


Fuente:https://campusvirtual.mexico.unir.net/mod/page/view.php?id=3201053&forceview=1

Boosting

Secuencial

Se entrenan en serie varios algoritmos, priorizando a la salida los peores resultados.



Ensemble learning

- Bosque aleatorio o random forest

```
1 from sklearn.ensemble import RandomForestClassifier
2
3 bosque = RandomForestClassifier(random_state=123,
4                                 oob_score=True)
5 bosque.fit(x_train,y_train)
```

Fuente:https://www.youtube.com/watch?v=K0qBHOgVR4E

Referencias

- https://www.ibm.com/mx-es/think/topics/ensemble-learning
- https://www.youtube.com/watch?v=1kIdC_GwrLQ
- https://www.youtube.com/watch?v=fMMwQY5M1xg
- https://www.youtube.com/watch?v=3pa0vtW64Ic
- https://www.youtube.com/watch?v=K0qBHOgVR4E