**PREWORK**

**SESIÓN 05**

**Objetivo:**

En esta sesión trataremos acerca de nuestro primer algoritmo supervisado de clasificación: Árboles de decisión y Random Forests. Para ello, necesitaremos estudiar los siguientes pre-requisitos

**Pre-requisito 1: Separación arbórea:**

En ciencias computacionales, la separación arbórea es un algoritmo que nos permite segmentar la información en dos o más partes. Una vez dividida la información en varias partes, es más sencillo atacar cada una por separado que atacar toda la información junta. Esta clase de problemas se le conoce como problemas de *dividir-y-conquistar* (*divide-and-conquer*).

Por ejemplo: Supongamos que queremos encontrar un dato en una lista enorme de datos que se encuentran ordenados. Por ahora, imaginaremos que la lista es:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Y el dato que queremos encontrar es 3.

Si la lista es enorme, es una mala idea ir uno por uno de los datos, nos vamos a tardar muchísimo computacionalmente hablando, en cambio, lo que podemos hacer es dividir la lista en dos partes iguales:

Lista de datos <= 5 es: [1, 2, 3, 4, 5]

Lista de datos > 5 es: [6, 7, 8, 9, 10]

Como sabemos dónde dividimos la lista (en el número 5) podemos saber qué lista descartar y con qué lista seguir (como 3 < 5, usamos la lista de datos <= 5). Con esto podemos dividir la lista una y otra vez hasta que nos quedemos con pocos datos y podamos buscar el dato rápidamente.

La división puede hacerse con poner cualquier condición, por ejemplo, decir que en una lista pondremos los números pares, y en otra los nones:

Lista de pares: [2, 4, 6, 8, 10]

Lista de nones: [1, 3, 5, 7, 9]

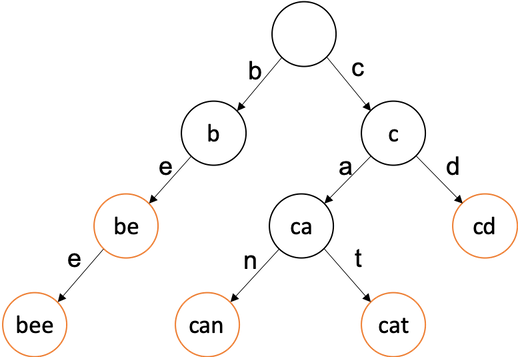
Todo depende de cual sea el criterio de división, sin embargo un árbol trata de dividir lo más posible los datos que tiene a la mano para reducir el problema a algo mucho más pequeño o más tratable.

**Pre-requisito 2. árboles como estructura de datos:**

Además de la separación arbórea, existe una estructura de datos llamada “árboles” que funcionan similar a la separación arbórea, pero están diseñados para manejar problemas como búsquedas de forma altamente eficiente. Las bases de datos dependen de esta clase de estructuras para hallar los datos que necesitan en cuestión de milisegundos, sin importar que tan grande es tu base de datos.

Para entender cómo funcionan los árboles, utilicemos un ejemplo interesante llamado “trie”: Supongamos que tienes un teclado predictivo, ¿Cómo es que conforme escribes una palabra, te salen varias palabras sugeridas?

Si tu escribes las letra c, el algoritmo de Trie salta del nodo de hasta arriba, hasta el nodo c, y luego -recursivamente- busca todas las palabras abajo de C, pero que están conectadas a C (can, cat, cd). y luego, al escribir “ca”, salta de C a CA, y hace lo mismo (can, cat). De esta forma, cada que escribes una palabra, letra por letra el sistema predictivo trata de irte ayudando a sugerirte posibles resultados. Si eres fan de Linux, esto ocurre cada vez que presionas la tecla TAB en la terminal.



**Pre-requisito 3. árboles de decisión:**

Ya hemos visto cómo es una separación arbórea, y ya hemos visto cómo es un árbol en la estructura de datos. Sin embargo, ¿que tiene que ver esto con los árboles de decisión?

* En la separación arbórea, el decidir hacia qué lado ir, depende de si un número es mayor o menor.
* En los árboles como estructuras de datos, el recorrer hacia un camino u otro depende de los datos que tenemos de entrada.
* En los árboles de decisión, nosotros definimos qué condición hace que te decidas hacia una dirección u otra.

Como puedes ver, el núcleo de todos los árboles en sistemas computacionales es: al tomar una decisión, descartas todas las demás, y caminas por aquellas que te sobran, hasta obtener solamente una: el camino que has elegido.

**Quiz:**

**¿Qué significa dividir y conquistar?**

1. Separar un resultado en varias partes.
2. Separar un problema en varias computadoras.
3. **Separar un problema en varias partes más sencillas.**
4. Separar los datos en varios sistemas.

**Si tenemos la lista [a, b, c, d, e, f], ¿cuál sería una buena división?**

1. [a, c, e, f] , [b, d]
2. [a], [b], [c, d, e, f]
3. **[a, b, c], [d, e, f]**
4. Todas las anteriores.

**¿Por qué es una mala idea buscar un dato uno por uno?**

1. Utiliza demasiada memoria.
2. **Utiliza demasiado tiempo.**
3. Es demasiado difícil de programar.
4. Es muy aburrido ir dato por dato.

**¿Qué es un criterio de división?**

1. Una forma de separar los datos en entrenamiento y prueba
2. Una manera de saber cómo elegir los datos
3. Una forma de hacer una división entre dos números
4. **Una forma en la cual podemos dividir los datos en partes iguales.**