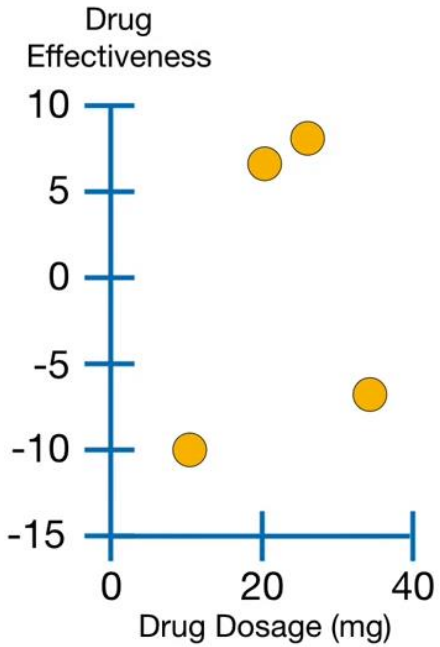
XGBoost

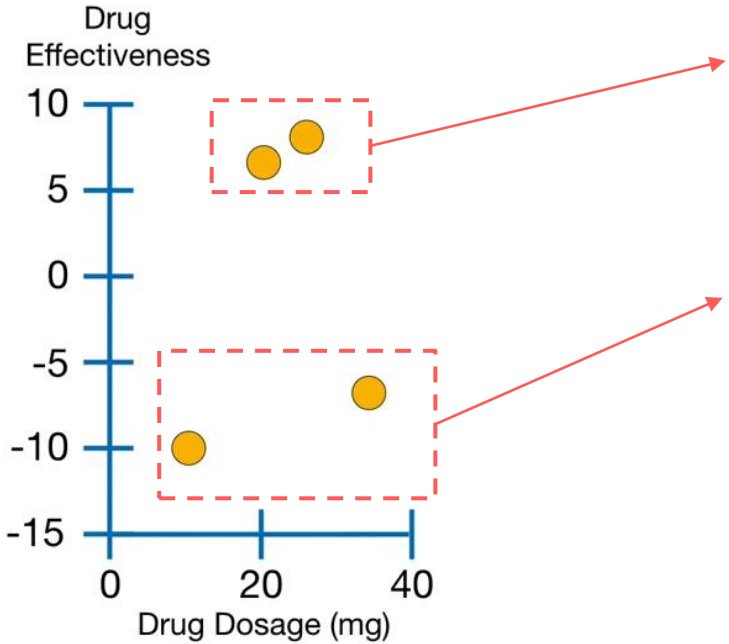
Ing. Juan M. Rodríguez

XGBoost: eXtreme Gradient Boost

XGBoost fue diseñado para **Big Data**, es decir para conjuntos de datos grandes y complejos. Sin embargo a fines de entender el algoritmo principal lo usaremos con un conjunto de datos simple (y para el caso de regresión).



XGBoost: eXtreme Gradient Boost



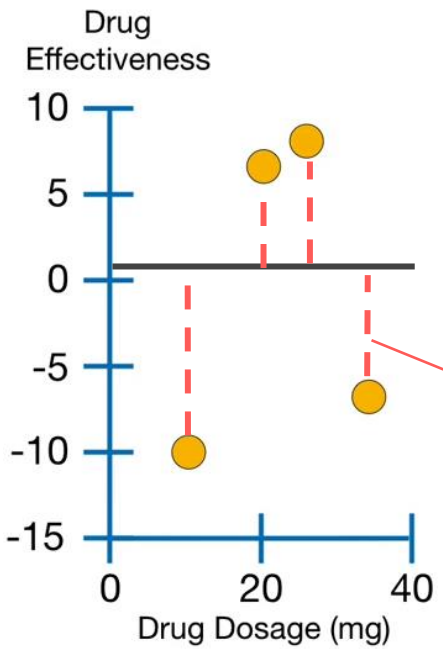
Estas dosis, han tenido un efecto **positivo** en la salud de los pacientes. Por eso están sobre el cero.

Estas dosis, han tenido un efecto **negativo** en la salud de los pacientes. Por eso están por debajo de cero.

Queremos predecir, para una dosis dada en miligramos, que tan efectiva es en los pacientes

XGBoost: eXtreme Gradient Boost

Primer paso: Hacer una predicción inicial.
Esta predicción puede ser cualquier valor.
Pero por defecto se toma 0.5



0.5

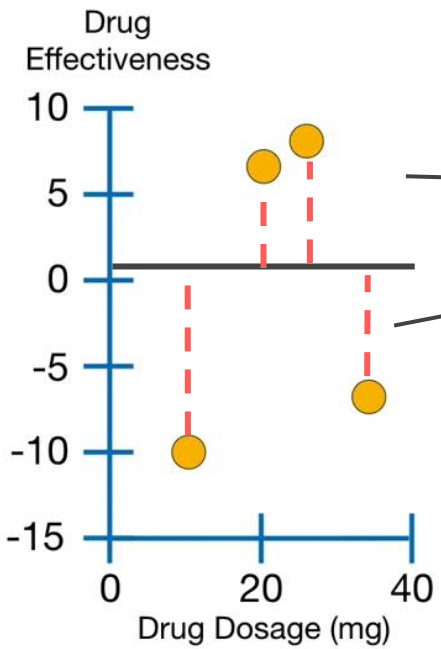
Esta predicción inicial es igual, ya sea que esté haciendo regresión o clasificación

Esta observación inicial se corresponde a esta linea negra

Estas líneas muestran los residuos, es decir el error que comete el estimador para cada medición.

XGBoost: eXtreme Gradient Boost

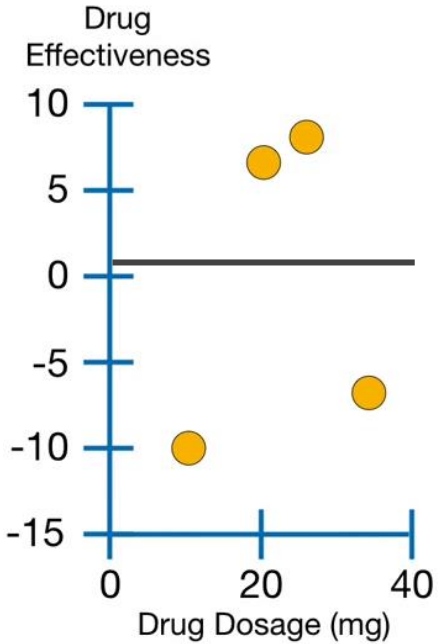
Segundo paso: Construir un árbol para los residuos.
Este árbol es diferente a los usados por Gradient Boost.
Primero se crea un nodo hoja.
Y se ponen allí todos los residuos



-10.5, 6.5, 7.5, -7.5

NOTA: Hay varias formas de construir el árbol utilizado por **XGBoost**. Esta es la más común.

XGBoost: eXtreme Gradient Boost



Tercer paso: Calcular el ***Similarity Score***, para los residuos

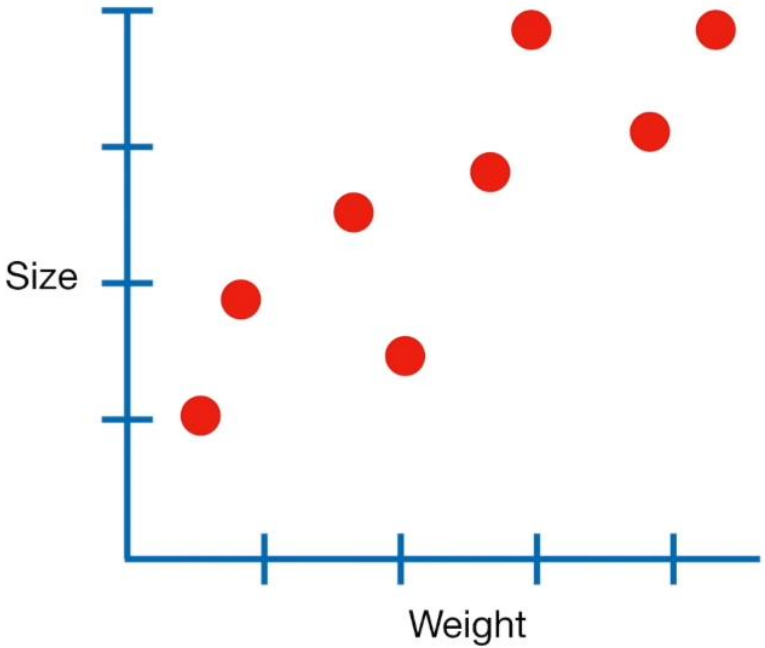
-10.5, 6.5, 7.5, -7.5

$$\text{Similarity Score} = \frac{(\text{Suma de residuos})^2}{\text{Cantidad de residuos} + \lambda}$$

λ (lambda): es un parámetro de regularización.

Regularización

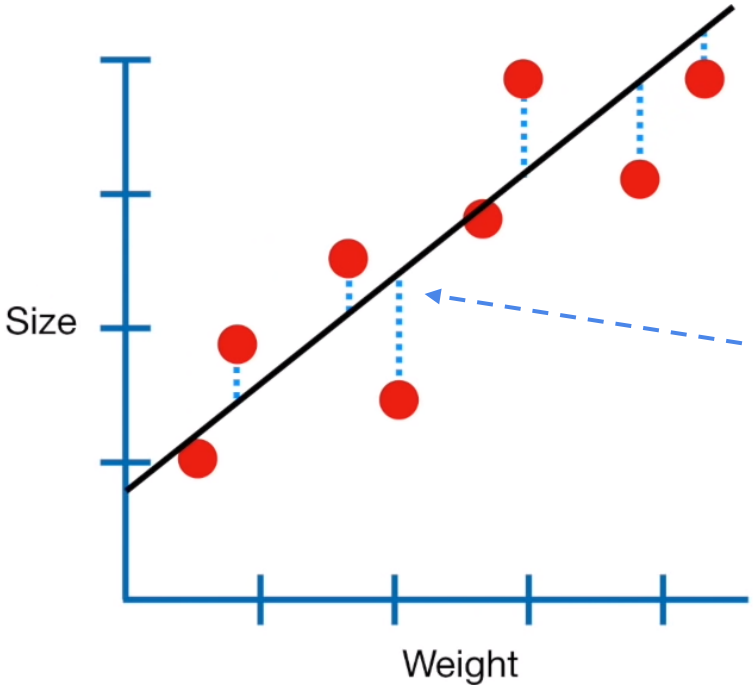
Regularización



Imaginemos que tenemos una serie de datos y queremos encontrar una forma de predecir valores futuros.

Utilizaremos en este caso, **regresión lineal.**

Regularización



Finalmente obtendremos una ecuación como esta:

Size = 0.9 + 0.75 **Weight**

Punto de intersección del eje-y

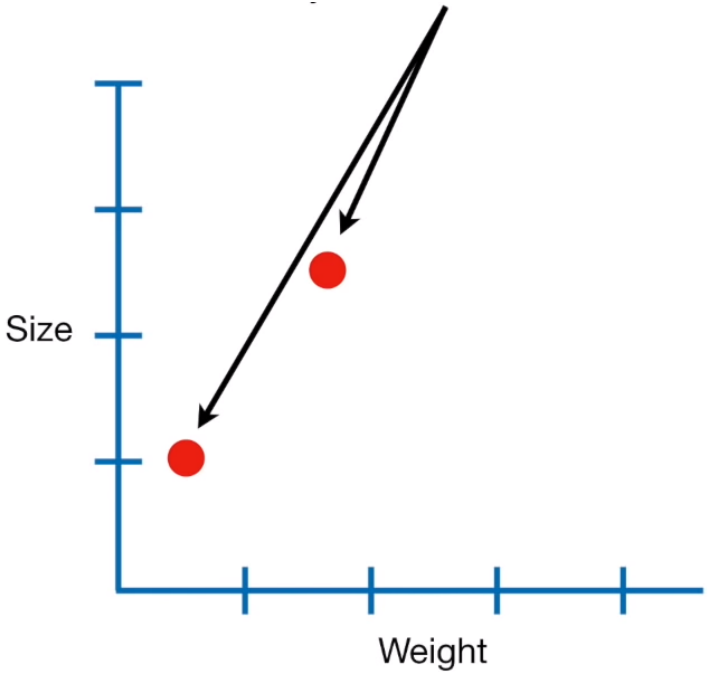
pendiente

Estos son los residuos. O errores.

Si hay muchos puntos, podemos estar seguros de que la línea calculada será un buen estimador del tamaño.

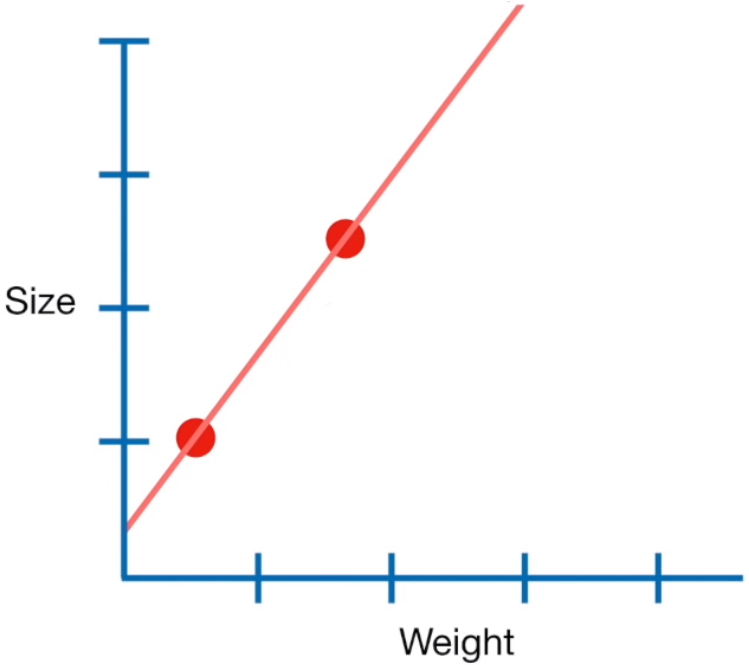
Regularización

¿Qué pasa si solo tenemos 2 puntos?

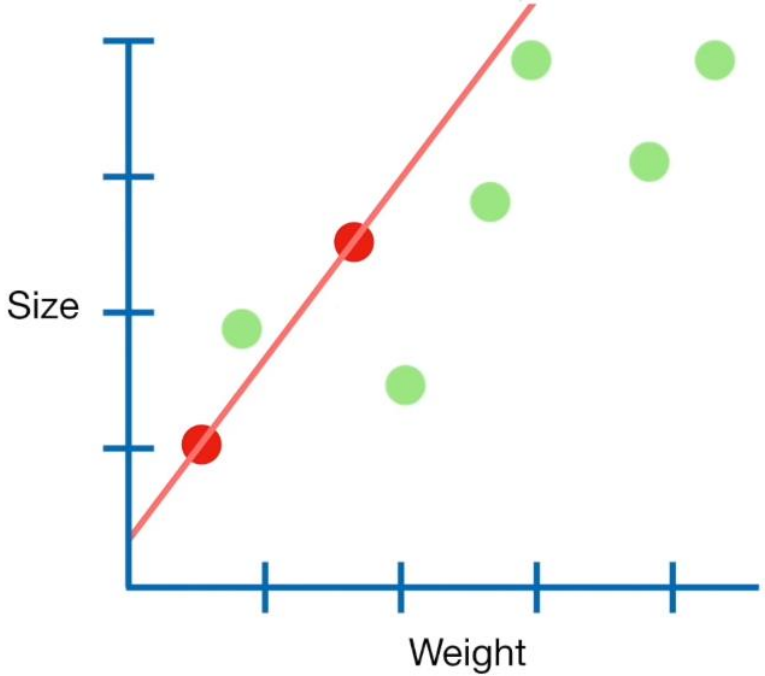


Regularización

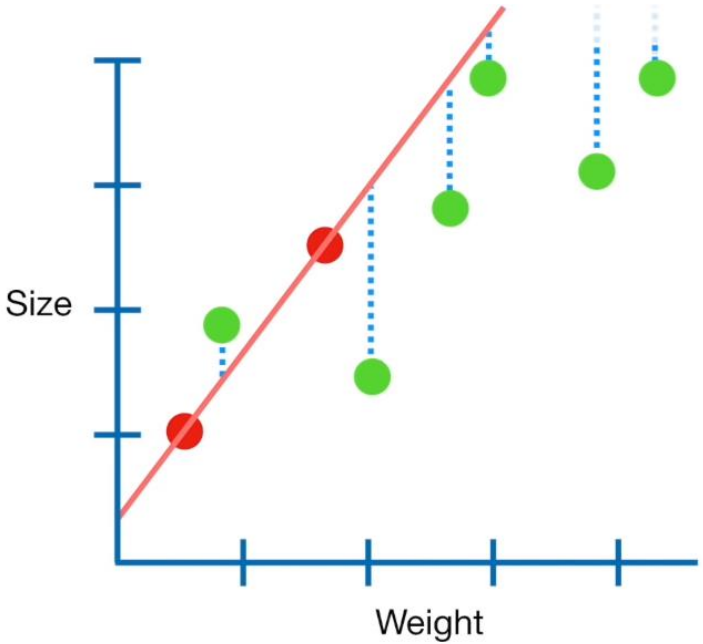
La línea calculada ajusta perfecto. No hay residuos. No hay errores.



Ahora mostramos en verde todos los puntos. Utilicemos al resto de los puntos, **los verdes**, como conjunto de prueba.



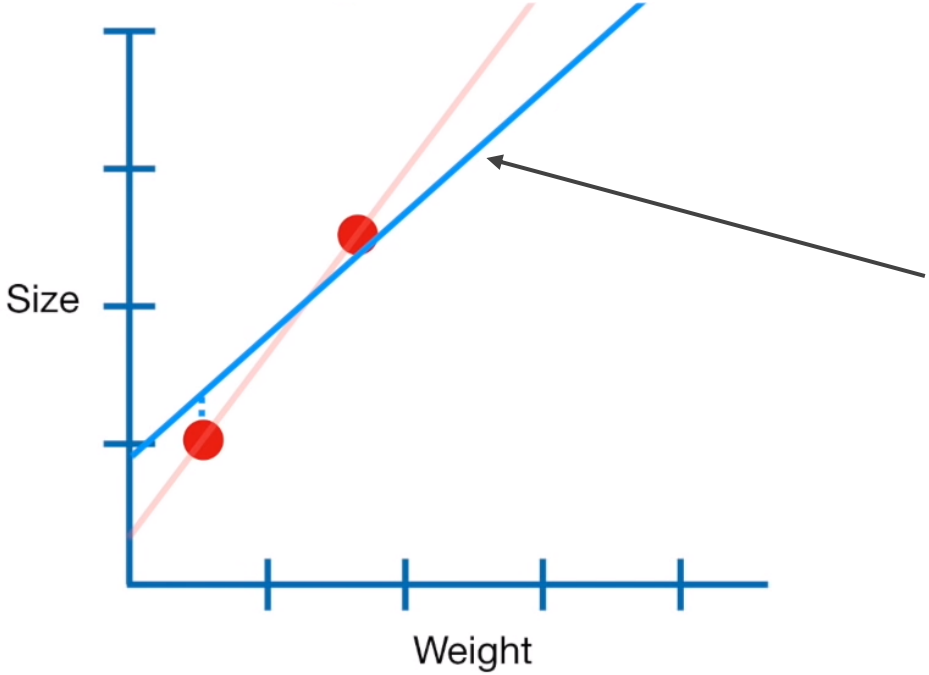
Regularización



En el conjunto de pruebas, si hay residuos. Y estos son mucho mayores que en el primer caso, cuando entrenamos con todos los datos!!

Esto quiere decir que está nueva línea, este estimador tiene una varianza alta (***High Variance***).
Una **varianza alta** indica que los puntos de datos están muy separados de la media y entre sí.
Y el modelo está **sobreentrenado** (***overfit***).

Regularización



Una forma de solucionar este problema, es usando una variación de la regresión lineal llamada: **Ridge Regression**

La idea es encontrar una línea que no ajuste tan bien

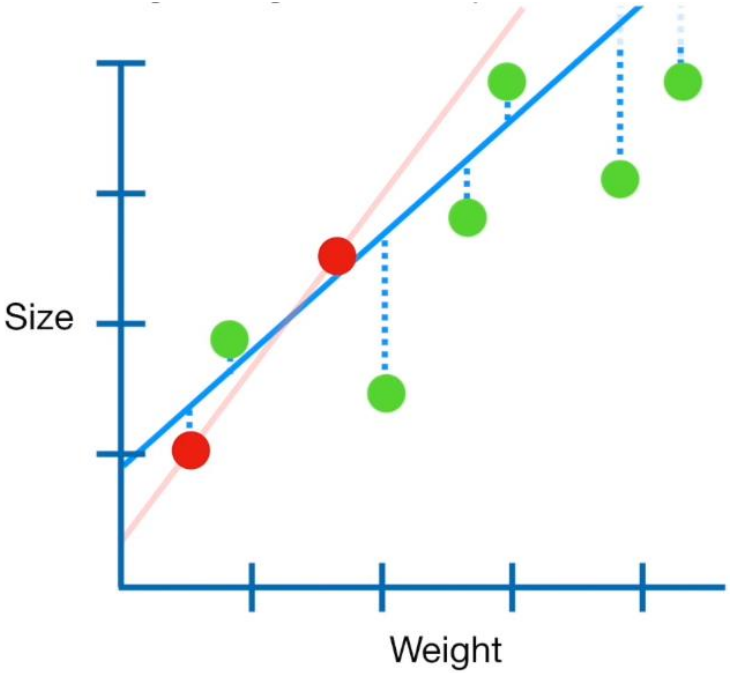
Para ello se va a introducir un pequeño sesgo o ***bias***, en los datos de entrenamiento.

Bias vs. Variance (*Bias-variance tradeoff*)

- El **error de sesgo** (***bias***) es un error de “suposiciones erróneas” en el algoritmo de aprendizaje. Un sesgo alto puede hacer que un algoritmo pierda las relaciones relevantes entre las características dadas y los resultados esperados (**underfitting**)

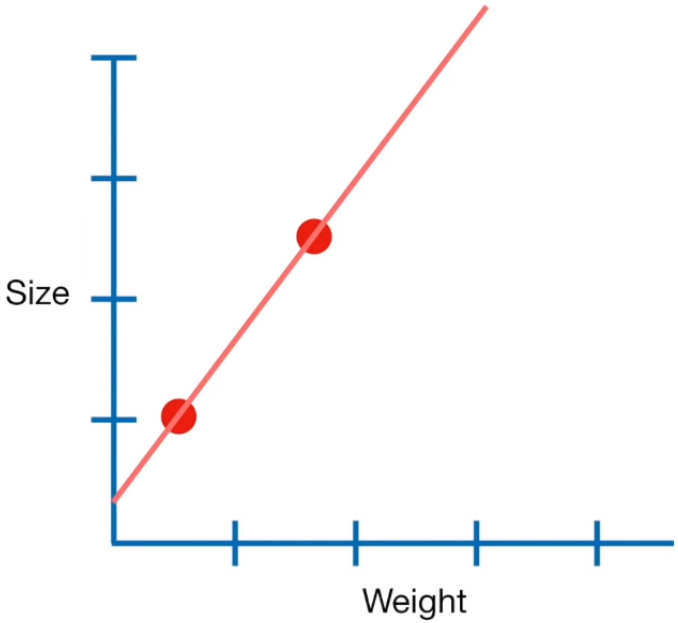
- La **varianza** es un error de sensibilidad a pequeñas fluctuaciones en el conjunto de entrenamiento. Una ***varianza alta*** puede resultar de un algoritmo que modela hasta el ruido aleatorio en los datos de entrenamiento (**overfitting**).

Regularización



El error de sesgo introducido en el estimador, hará que caiga el error por la varianza de forma mucho más abrupta en el conjunto de pruebas.

Regularización



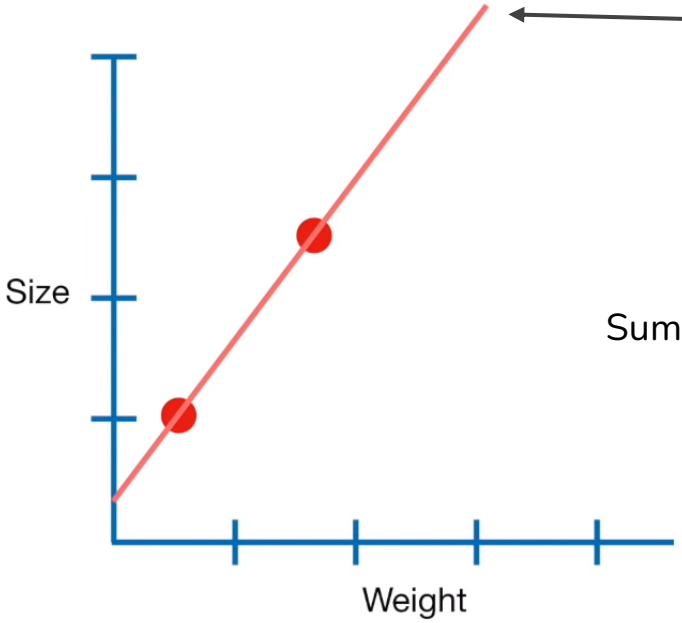
Cuando usamos **Regresión Lineal**, es decir cuadrados mínimos, lo que estamos haciendo es minimizando **la suma del cuadrado de los residuos.**
En cambio en **Ridge Regression** estamos minimizando:

la suma del cuadrado de los residuos + λ * (pendiente)^2

Y λ determina qué tan severa es esta penalización

Esto representa una penalización al método tradicional.

Regularización: ejemplo



Según mínimos cuadrados, tenemos que:

Size = 0.4 + 1.3 * **Weight**

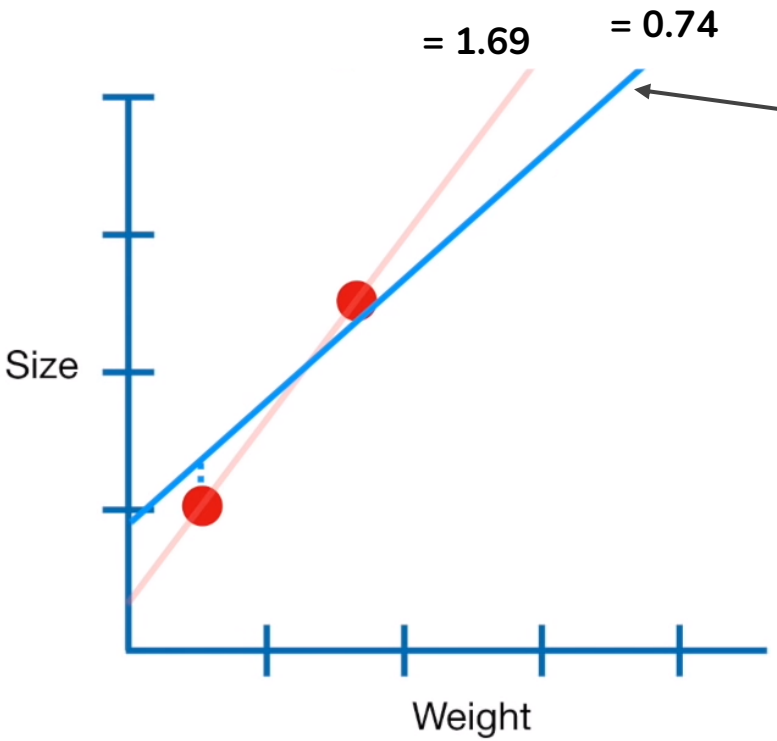
Suma de cuadrados de residuos = $0^2 + 0^2$

Pero usando Ridge Regression:

Suma de cuadrados de residuos + **λ** * (pendiente)^2 = 0 + 1 * $(1.3)^2$ **= 1.69**

Asumimos λ = 1

Regularización: ejemplo



Ridge Regression, seguirá ajustando hasta lograr una recta como la **azul**:

Size = 0.9 + 0.8 * **Weight**

Suma de cuadrados de residuos = $0.3^2 + 0.1^2$

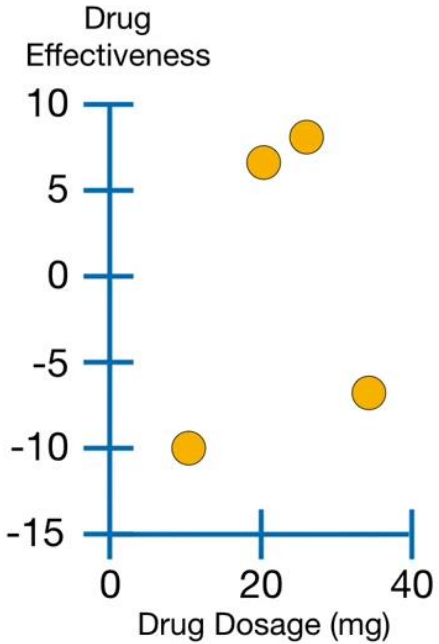
Suma de cuadrados de residuos + $\lambda * (\text{pendiente})^2$

$0.3^2 + 0.1^2 + 1 * (0.8)^2$

$0.09 + 0.01 + 0.064 = \mathbf{0.74}$

Volvamos a XGBoost y al calculo del Similarity Score

XGBoost: eXtreme Gradient Boost



Tercer paso: Calcular el ***Similarity Score***, para los residuos

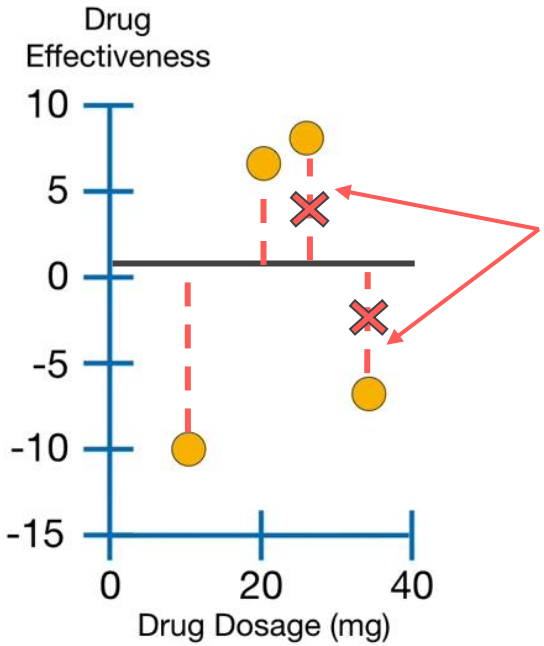
-10.5, 6.5, 7.5, -7.5

$$\text{Similarity Score} = \frac{(\text{Suma de residuos})^2}{\text{Cantidad de residuos} + \lambda}$$

λ (lambda): es un parámetro de regularización.

Con λ reducimos el error por varianza.
Pero por ahora dejemos $\lambda = 0$

XGBoost: eXtreme Gradient Boost



Se cancelan mutuamente

Tercer paso: Calcular el ***Similarity Score***, para los residuos

-10.5, 6.5, 7.5, -7.5

$$\text{Similarity Score} = \frac{(-10.5 + 6.5 + 7.5 - 7.5)^2}{4 + 0}$$

$$\text{Similarity Score} = \frac{(-4)^2}{4}$$

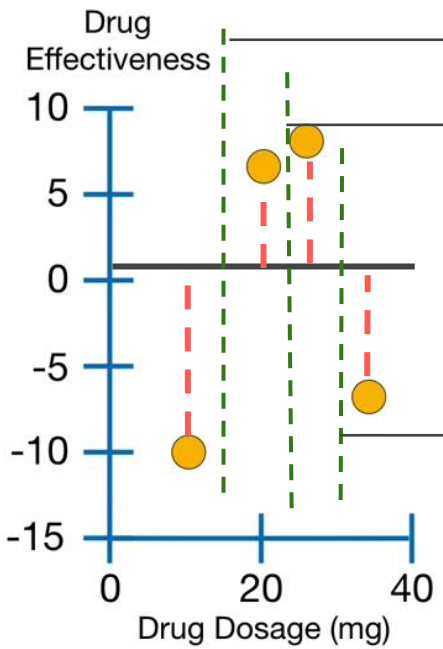
El ***Similarity Score*** para los residuos del nodo raíz es = **4**

XGBoost

-10.5, 6.5, 7.5, -7.5

Similarity Score = 4

Cuarto paso: Tenemos que ver cuál será el siguiente nodo. Para ello vamos a calcular la **ganancia total**, según escojamos una opción u otra para partir el árbol.



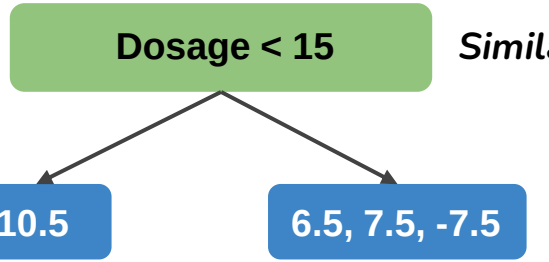
Opción 1, tomar como umbral la distancia intermedia entre las primeras 2 observaciones

Opción 2, tomar como umbral la distancia intermedia entre las segundas 2 observaciones

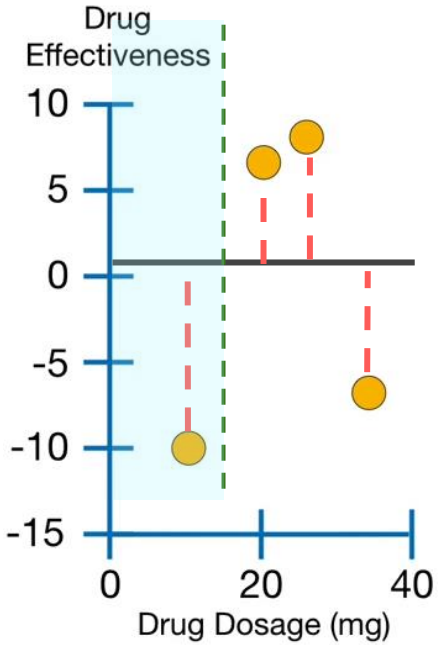
Opción 3, tomar como umbral la distancia intermedia entre últimas 2 observaciones

XGBoost

Cuarto paso: veamos la opción 1



Similarity Score = 4



Similarity Score = $(-10.5)^2 / 1$ = **110,25**

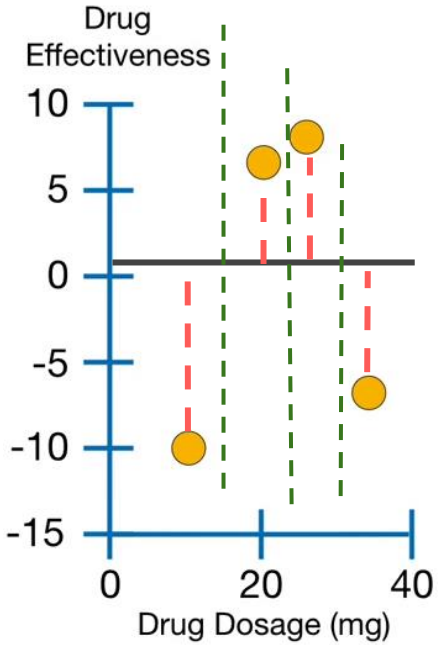
Similarity Score = $(6.5)^2 / 3$ = **14.08**

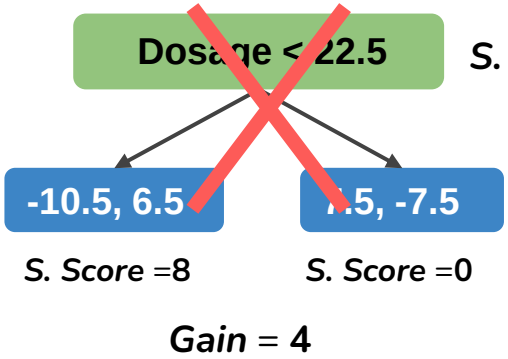
Gain = $\text{Nodo\_Izquierda}_{\textit{Similarity Score}} + \text{Nodo\_Derecha}_{\textit{Similarity Score}} - \textit{Raiz}_{\textit{Similarity Score}}$

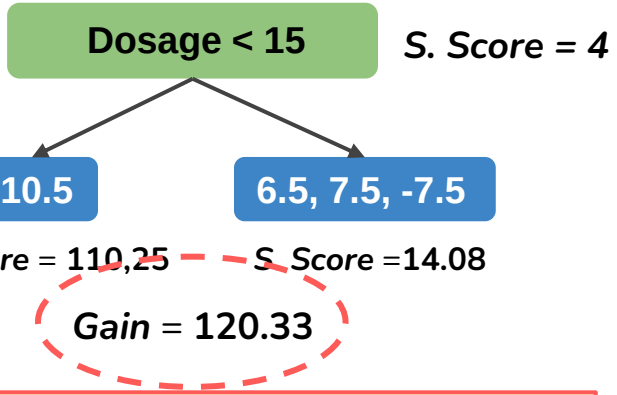
Gain = 110.25 + 14.08 - 4 = **120.33**

XGBoost

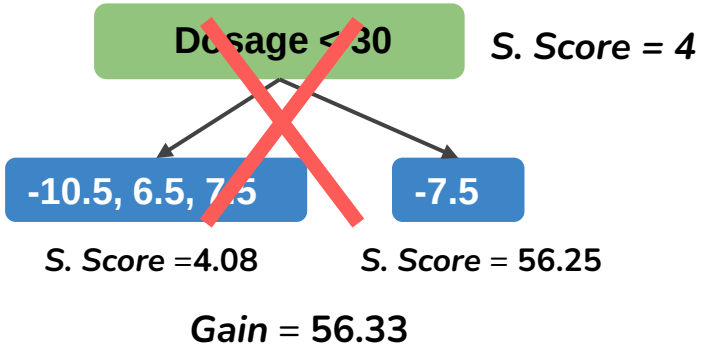
Cuarto paso: exploramos todas las opciones





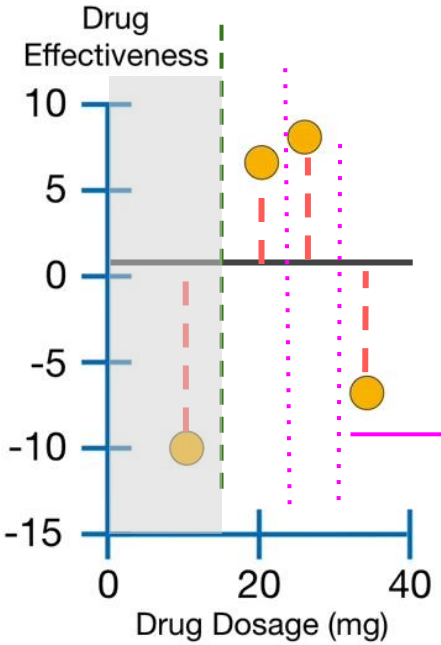


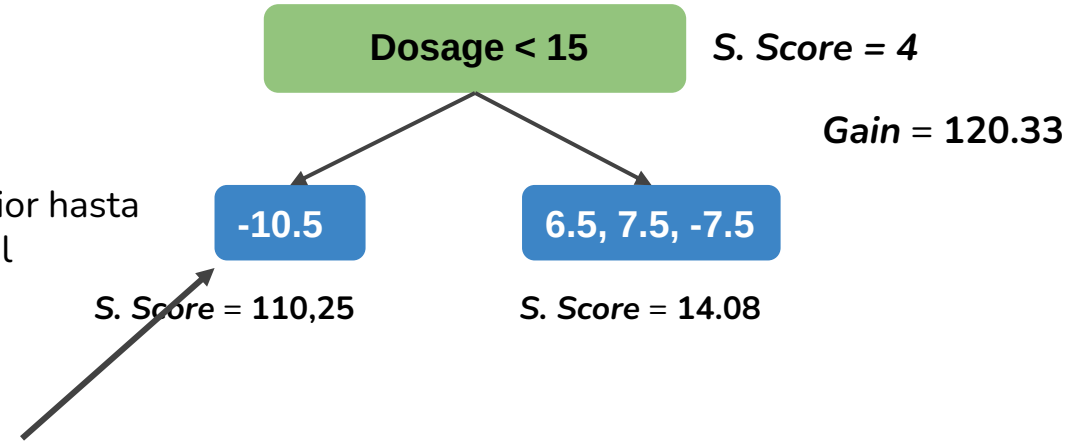
"Dosage < 15" es el umbral que mejor divide los residuos del árbol.



XGBoost

Quinto paso: repetimos el anterior hasta alcanzar la profundidad del árbol estipulada



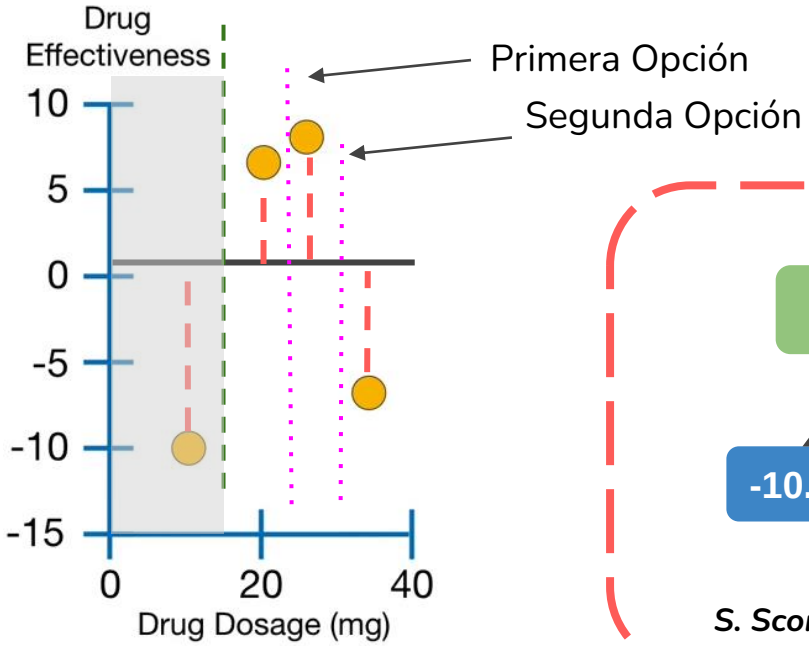


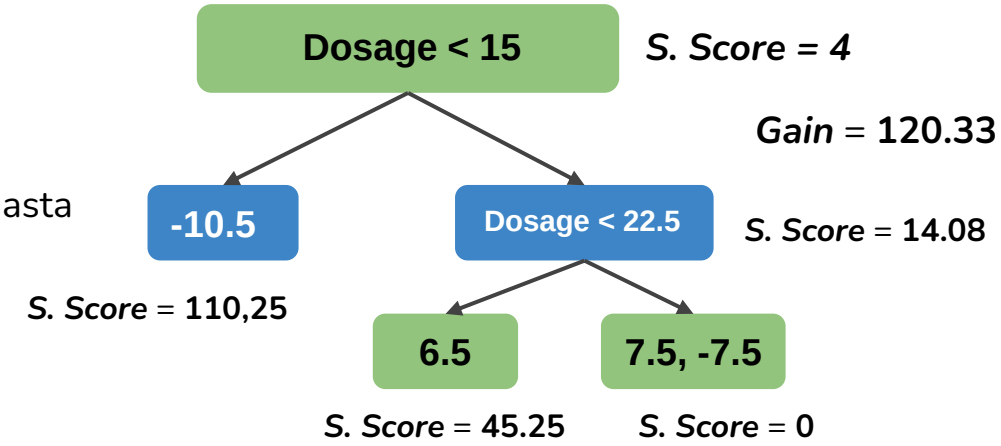
Acá ya no se puede dividir más, ya que hay un solo residuo

Volvemos a explorar estas dos opciones para dividir el nodo de la derecha

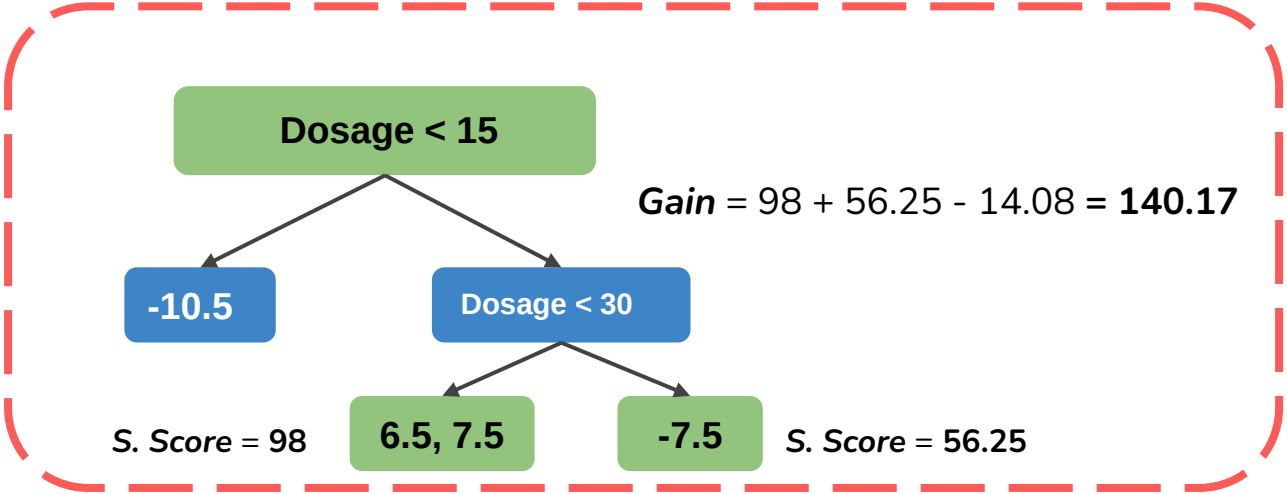
XGBoost

Quinto paso: repetimos el anterior hasta alcanzar la profundidad del árbol estipulada



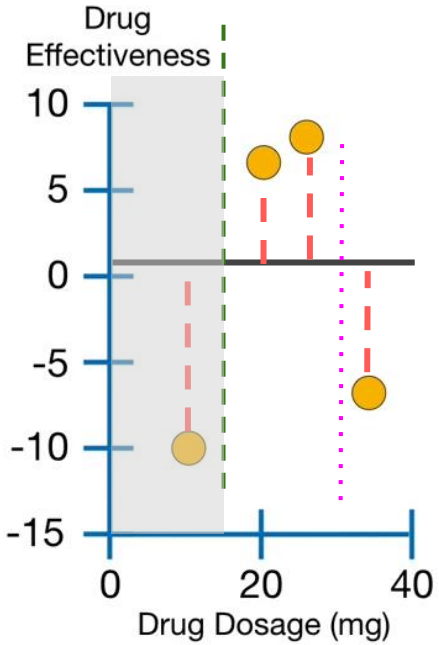


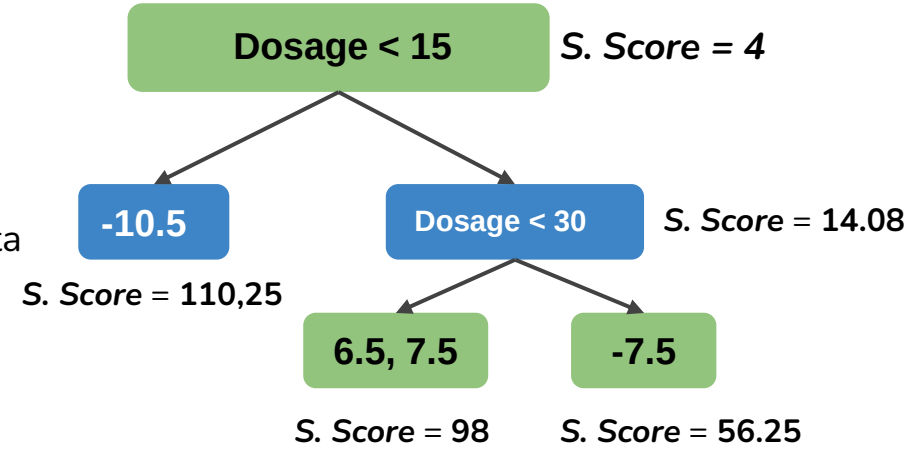
Gain = 42.25-14.08 **= 28.17**



XGBoost

Quinto paso: repetimos el anterior hasta alcanzar la profundidad del árbol estipulada



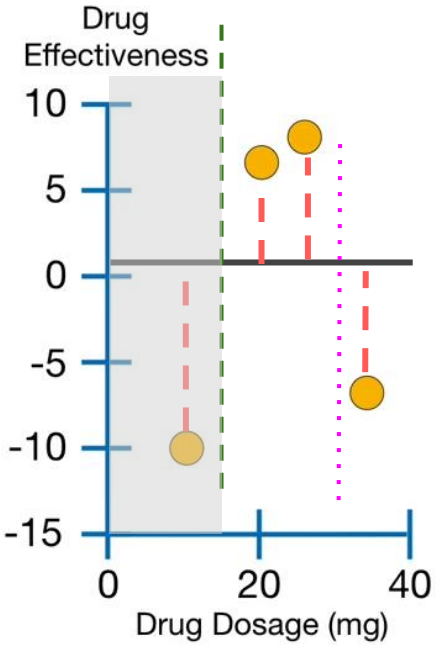


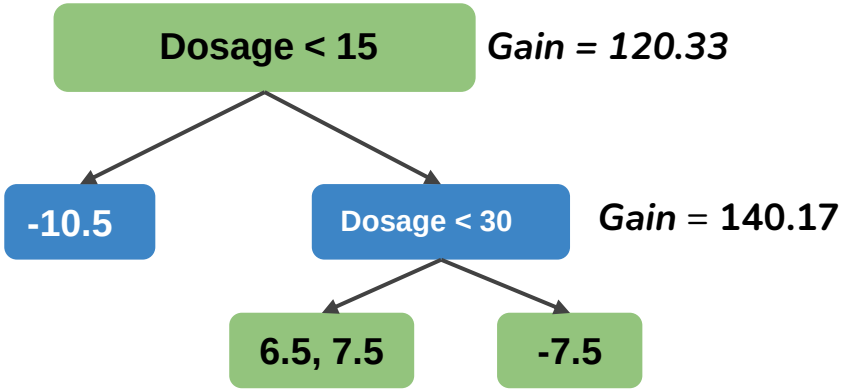
Gain = **140.17**

Cómo nuestra restricción era de **2 niveles** de profundidad terminamos de generar el árbol.
Aunque por defecto **XGBoost** trabaja con 6 niveles.

XGBoost

Sexto paso: poda

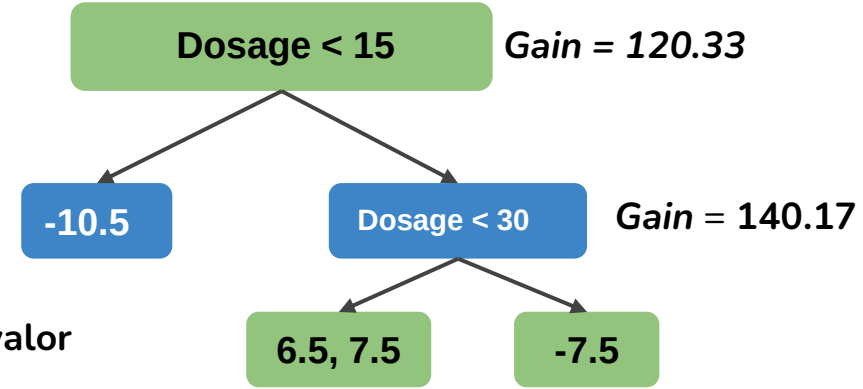




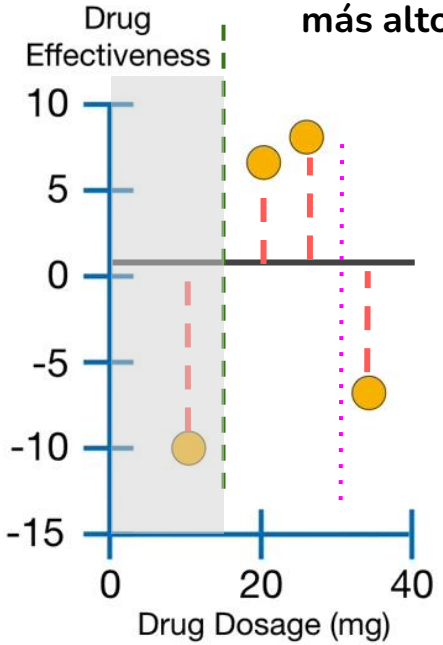
- Elegimos un número al azar, ejemplo: **130**
- Este número se llama gamma (γ)
- Calculamos la diferencia entre el **Gain**, del nodo más bajo y **gamma**
  - Gain - γ = 140.17 - 130 =
  - Gain - γ = 140.17 - 130 = 10.17
- Sí la diferencia es < 0 **=>** removemos el nodo
- Sino, el nodo se queda y se terminó la poda

XGBoost

Sexto paso: poda



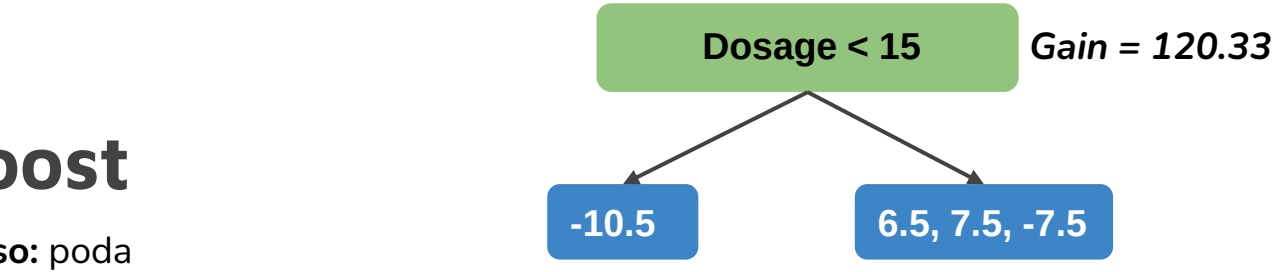
¿Qué hubiera pasado si elegíamos un valor más alto?

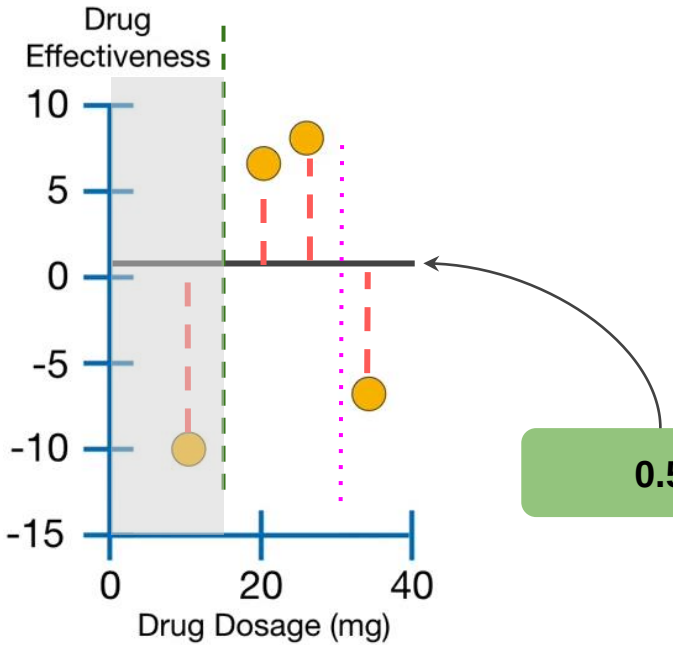


- Elegimos gamma (γ) igual a **150**
- Calculamos la diferencia entre el **Gain**, del nodo más bajo y **gamma**
  - Gain - γ = 140.17 - 150 = –9,83
- Sí la diferencia es < 0 **=>** removemos el nodo

XGBoost

Sexto paso: poda





Por defecto gamma es igual a 0. Cuanto más alto más conservador es el algoritmo

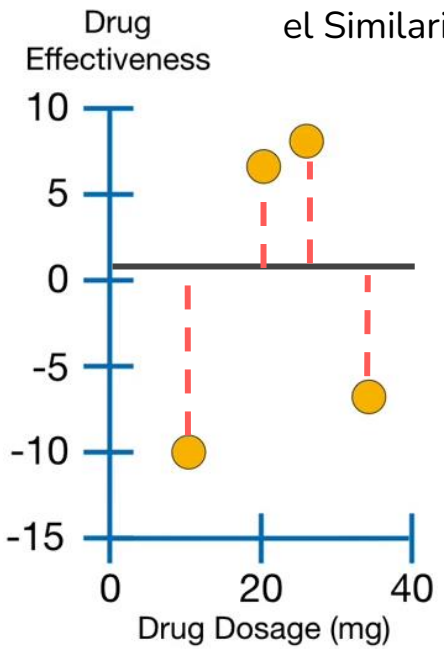
- La poda continua con gamma (γ) igual a **150**
- Calculamos la diferencia entre el **Gain**, del nodo más bajo y **gamma**
  - Gain - γ = 120.33 - 150 = −29,67
- Sí la diferencia es < 0 **=>** removemos el nodo

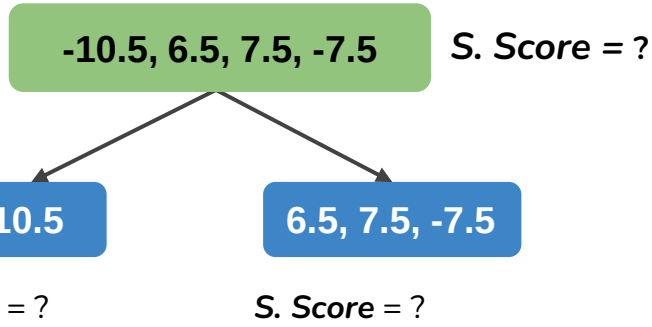
0.5

Solo queda la estimación inicial

XGBoost: eXtreme Gradient Boost

Séptimo paso: Volvemos a calcular el árbol (repite paso 3), solo que esta vez usamos lambda **λ** igual a 1 al calcular el Similarity Score



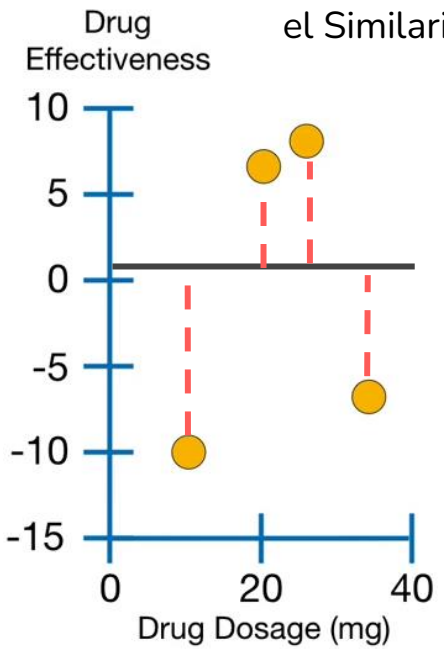


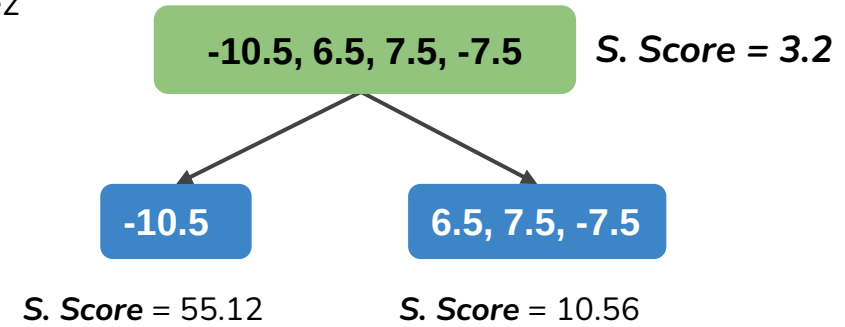
$$\text{Similarity Score} = \frac{(\text{Suma de residuos})^2}{\text{Cantidad de residuos} + \lambda}$$

$$\text{Similarity Score} = \frac{(\text{Suma de residuos})^2}{\text{Cantidad de residuos} + 1}$$

XGBoost: eXtreme Gradient Boost

Séptimo paso: Volvemos a calcular el árbol (repite paso 3), solo que esta vez usamos lambda **λ** igual a 1 al calcular el Similarity Score

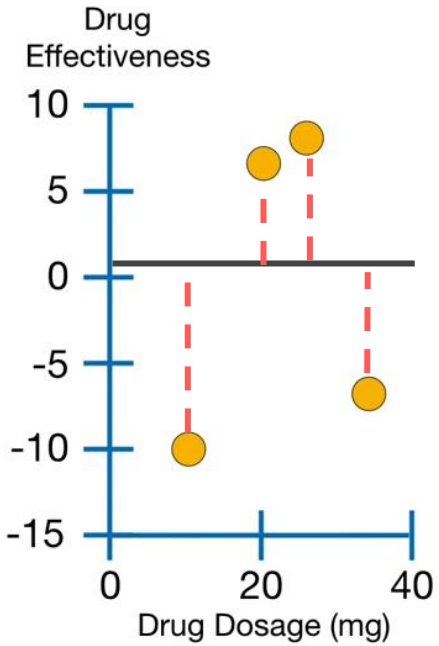


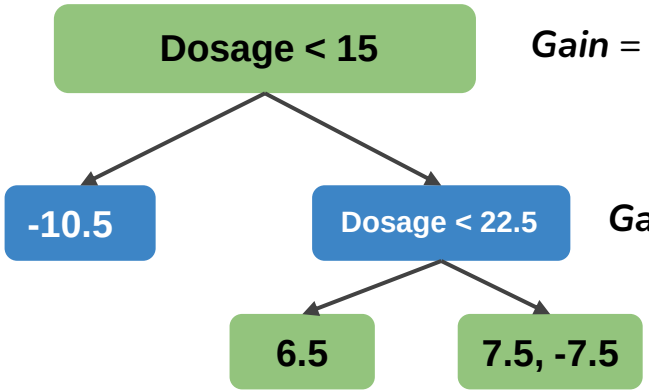


- Con lambda > 0, los Similarity Scores son mucho más chicos
- Esta disminución es proporcional a la cantidad de residuos en el nodo
  - El nodo raíz pasó de 4 a 3.2 (se redujo un 20%)
  - El nodo hoja izquierdo pasó de 110 a 55, un 50%

XGBoost: eXtreme Gradient Boost

Séptimo paso : Calculamos ahora la ganancia de cada nodo. Es decir el **Gain**





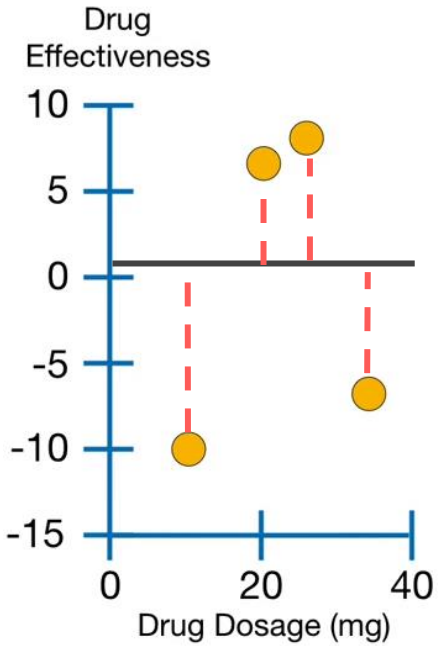
Gain = **62.49** (era 120.33)

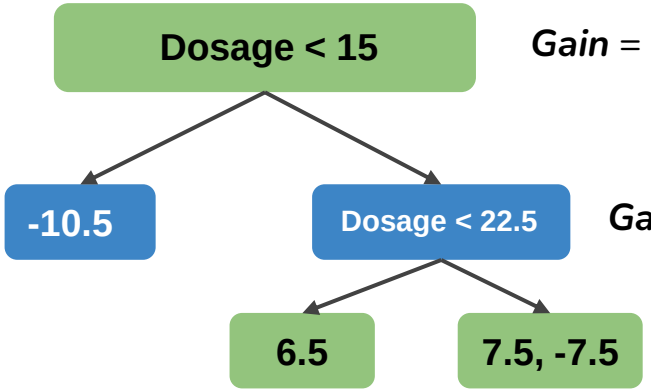
Gain = **82.9** (era 140.17)

Tener lambda > 0 hace que el **Gain** también sea menor

XGBoost: eXtreme Gradient Boost

Séptimo octavo : Podamos (paso quinto nuevamente)





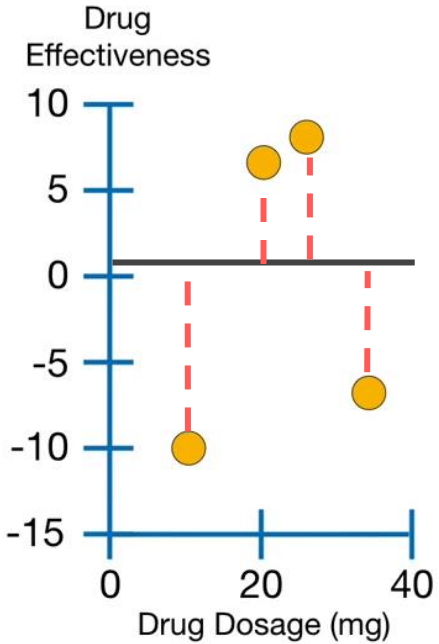
Gain = **62.49** (era 120.33)

Gain = **82.9** (era 140.17)

Cómo habíamos elegido gamma (γ) igual a 130, se poda todo el árbol.

XGBoost: eXtreme Gradient Boost

Séptimo octavo : Podamos (paso quinto nuevamente)

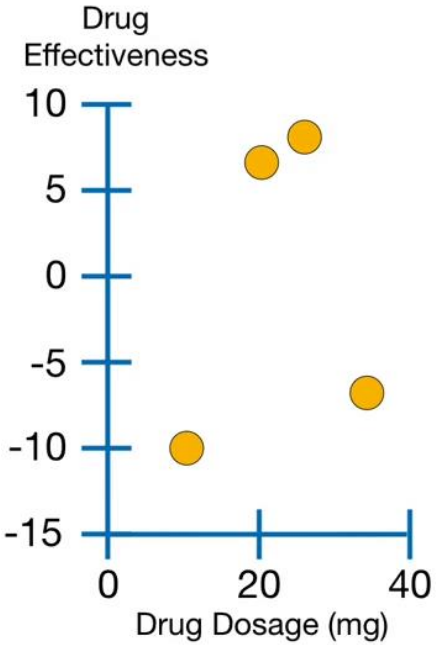


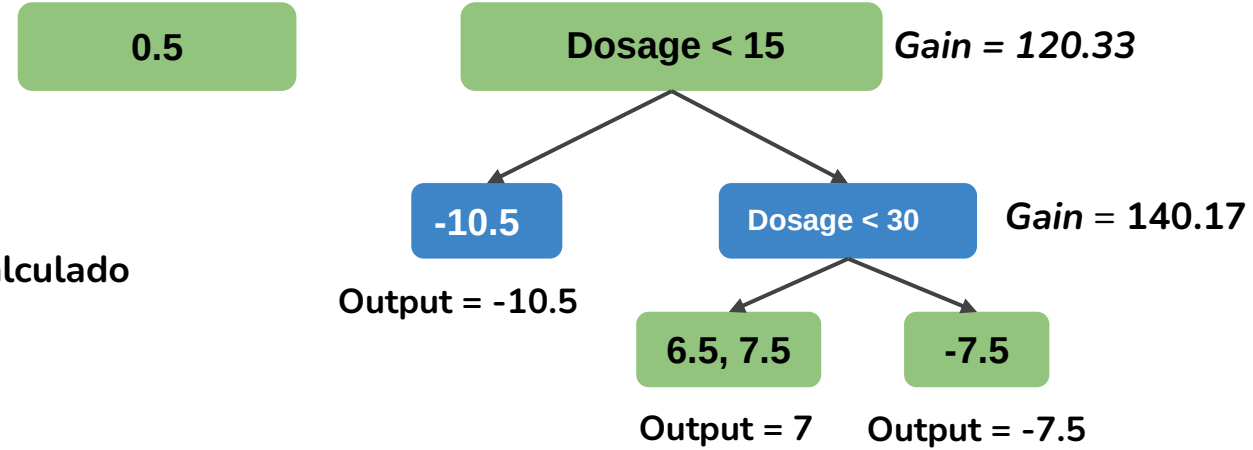
- Cuando $\lambda > 0$ es más probable tener que podar un árbol, ya que los **Gain** calculados son menores.
- Aún eligiendo $\gamma = 0$ podemos tener que podar nodos, ya que la ganancia (**Gain**) puede ser negativa.
- **λ=1,** previene el sobreentrenamiento o sobreajuste del modelo en el conjunto de entrenamiento

Calcular la respuesta de XGBoost

XGBoost

Este es el árbol calculado hasta ahora





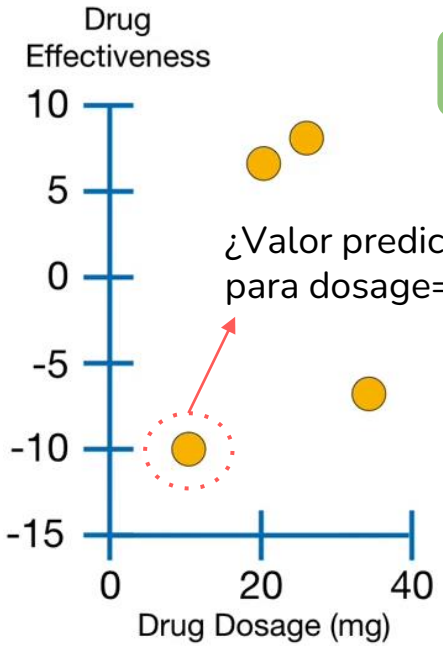
$$\textbf{Output} = \frac{\text{Suma de residuos}}{\text{Número de residuos} + \lambda}$$

Tomamos $\lambda = 0$ ya que es el valor por defecto

XGBoost

***Learning rate*: por defecto es 0.3**

Este es el árbol calculado hasta ahora



0.5 + **0.3** × **Dosage < 15** *Gain = 120.33*

-10.5 **Dosage < 30** *Gain* = **140.17**

Output = -10.5

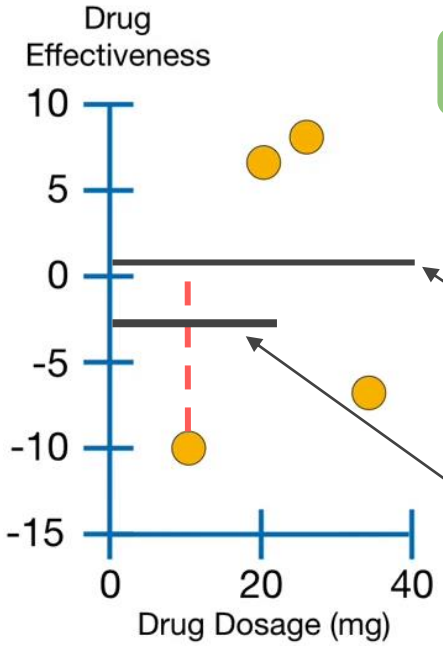
6.5, 7.5 **-7.5**

Output = 7 **Output = -7.5**

0.5 + (0.3 *(-10.5)) = -2.65

XGBoost

Este es el árbol calculado hasta ahora



Esta era la predicción original. Su error o residuo era: **-10.5**

Esta es la nueva predicción: **-2.65**
Su residuo es: **–7,35**

0.5 ✚ **0.3** ✖ **Dosage < 15** *Gain = 120.33*

-10.5 — **Output = -10.5**

Dosage < 30 *Gain* = **140.17**

6.5, 7.5 — **Output = 7**

-7.5 — **Output = -7.5**

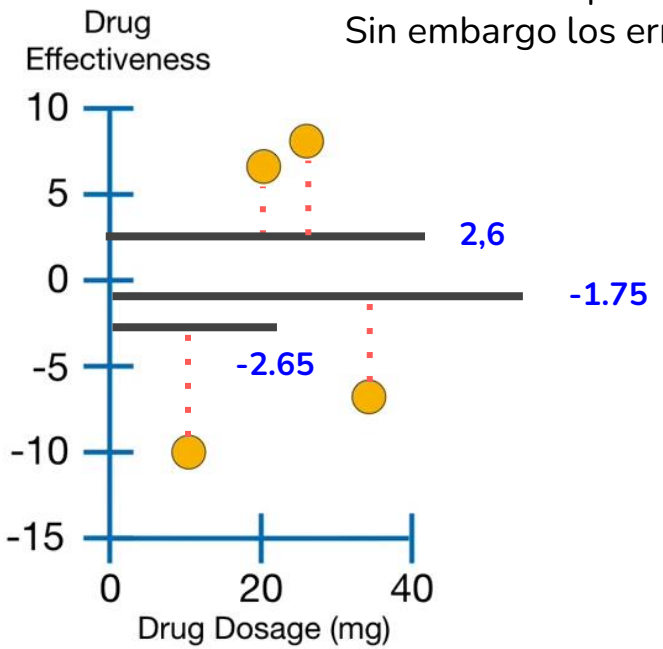
Output = 0.5 +(0.3 * (-10.5))= **-2.65**

Nos acercamos de a poco al valor real

XGBoost

Nuevo paso: Calculamos todos los residuos utilizando el árbol hasta acá

Podemos ver que todas las estimaciones mejoran respecto de la original (0.5).
Sin embargo los errores (residuos) siguen siendo altos.



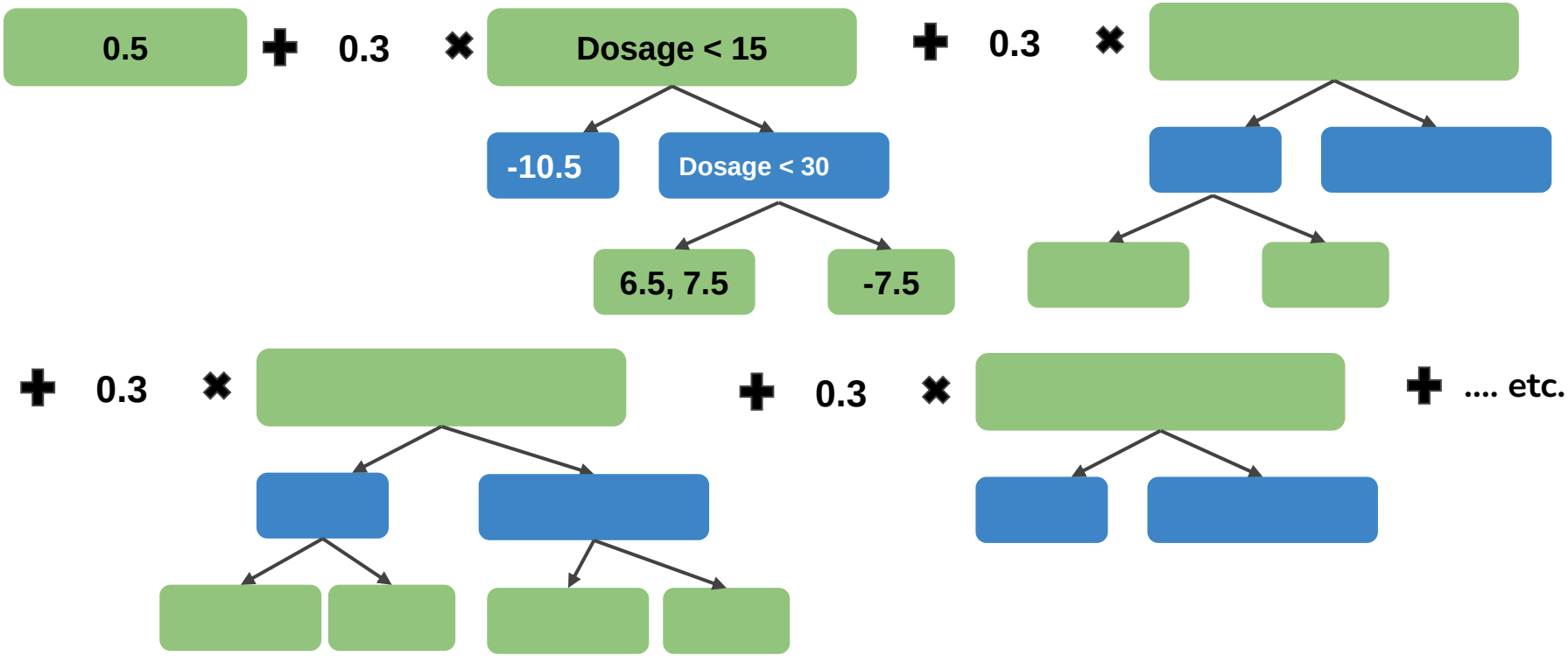
Con estos nuevos residuos, construimos un nuevo árbol.
Repetimos todo, desde el paso 2.

Con el nuevo árbol, calculamos la salida de cada elemento y luego los residuos.

Construimos otro árbol.

Seguimos hasta que los residuos son prácticamente cero o bien alcanzamos el número máximo de árboles predefinido

XGBoost: estructura final



Video cómo este y muchos otros útiles

https://www.youtube.com/channel/UCtYLUTtgS3k1Fg4y5tAhLbw

El autor es el Dr. Josh Starmer, profesor de la universidad de North Carolina