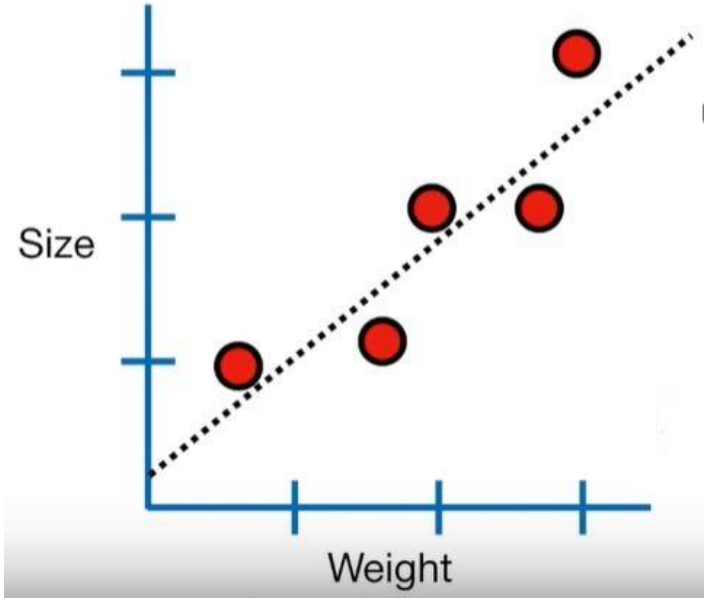
boost
accelerate your learning

Machine Learning

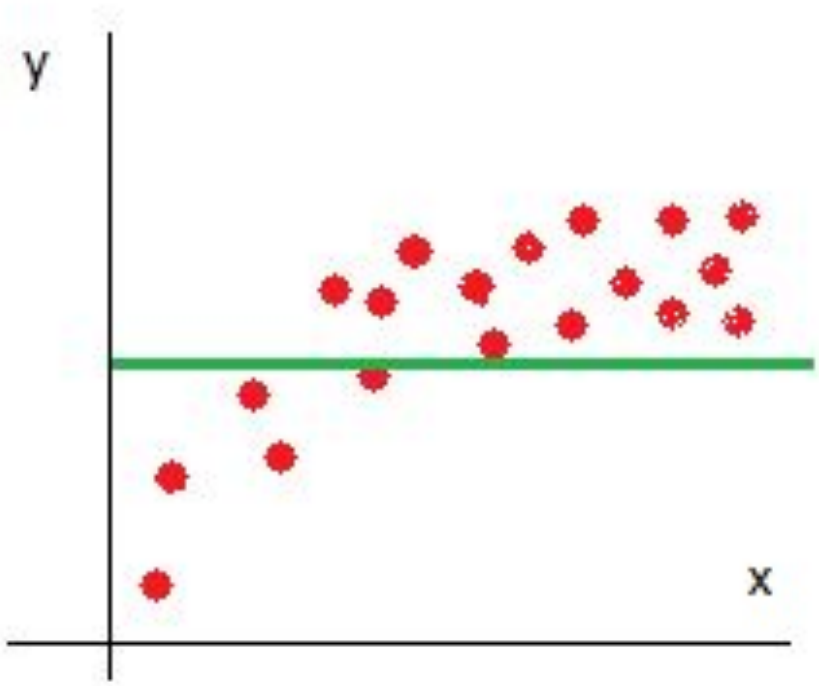
La disciplina que crea sistemas que aprenden automáticamente.

Pequeña introducción: Regresión Lineal



A pesar de las controversias, la regresión lineal cae en la definición de aprendizaje automático: Sistema que aprende de manera automática de la data. En muchos libros del tema es el primer modelo a enseñar.

¿Qué es underfitting?

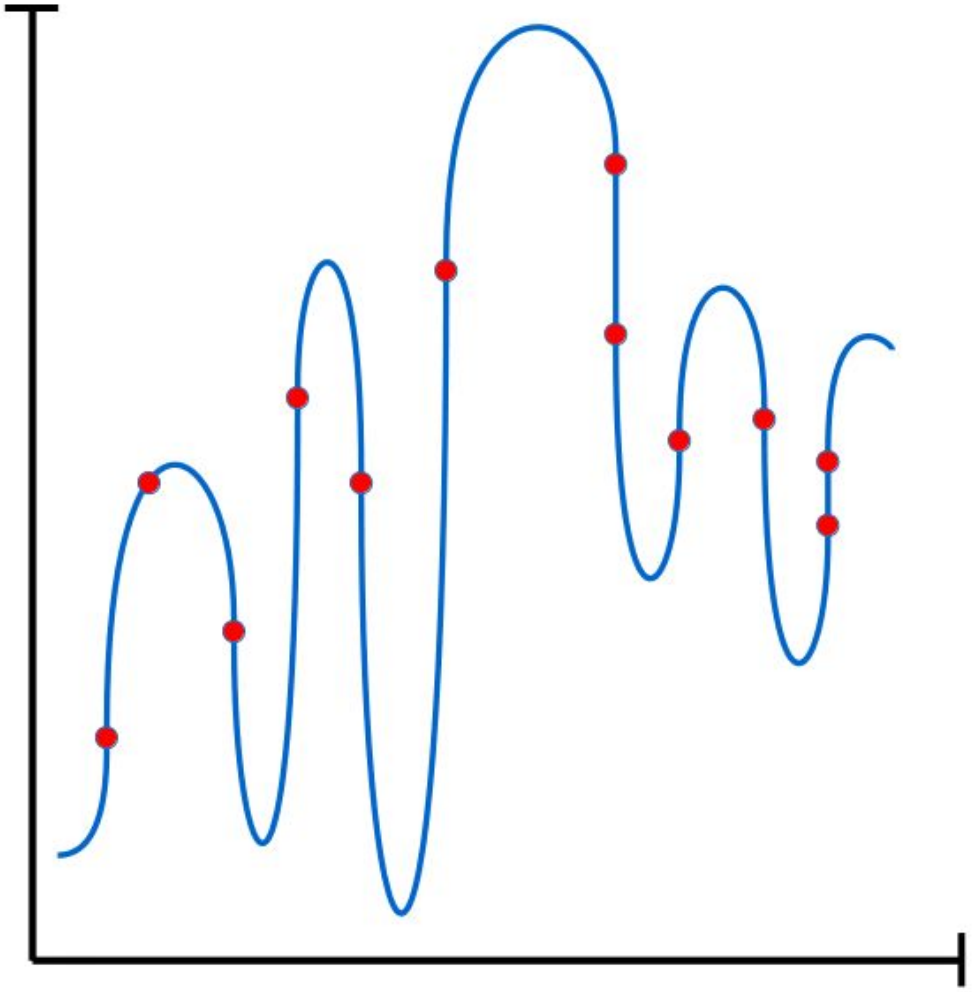


Underfitting (High Bias)

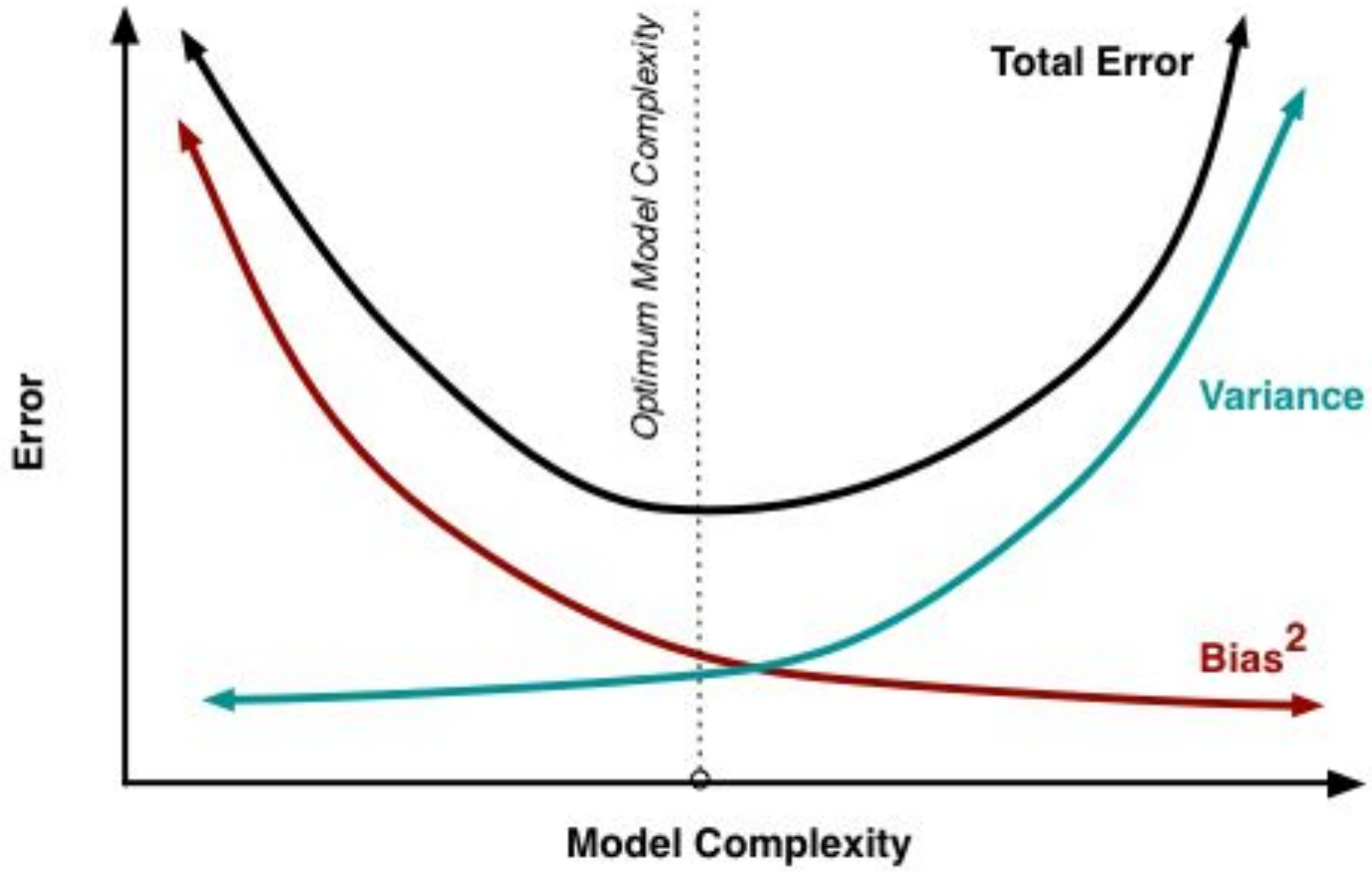
Usando el modelo más simple

boost
accelerate your learning

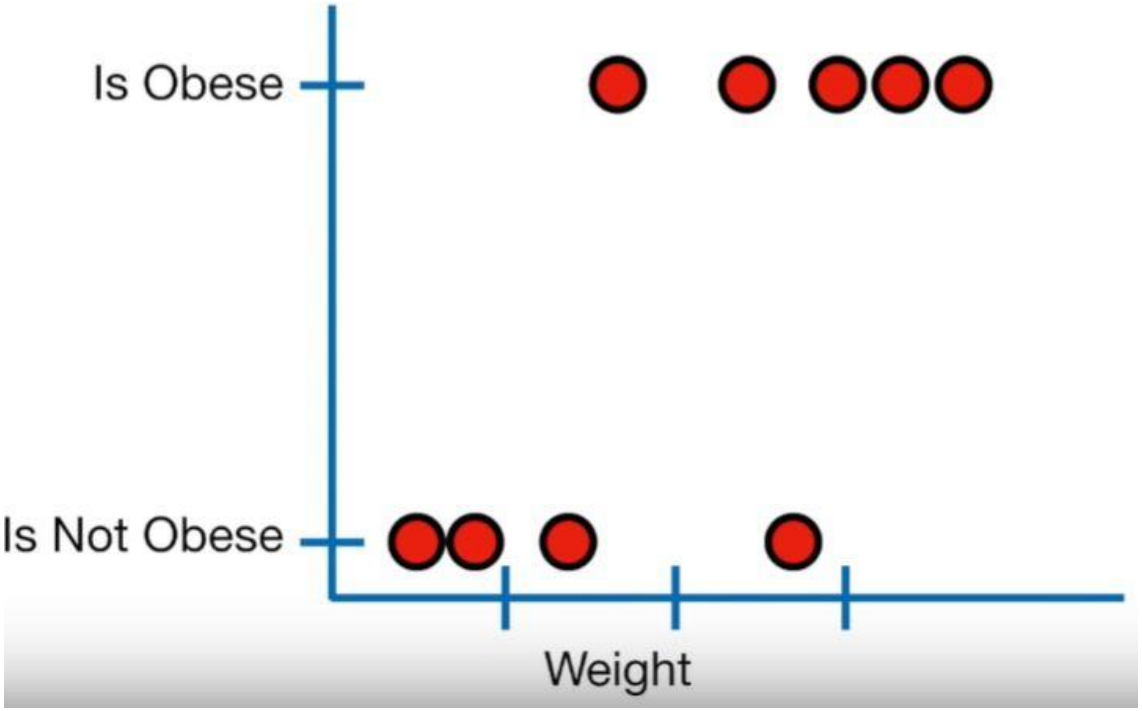
¿Qué es overfitting?



El tradeoff entre bias y variance



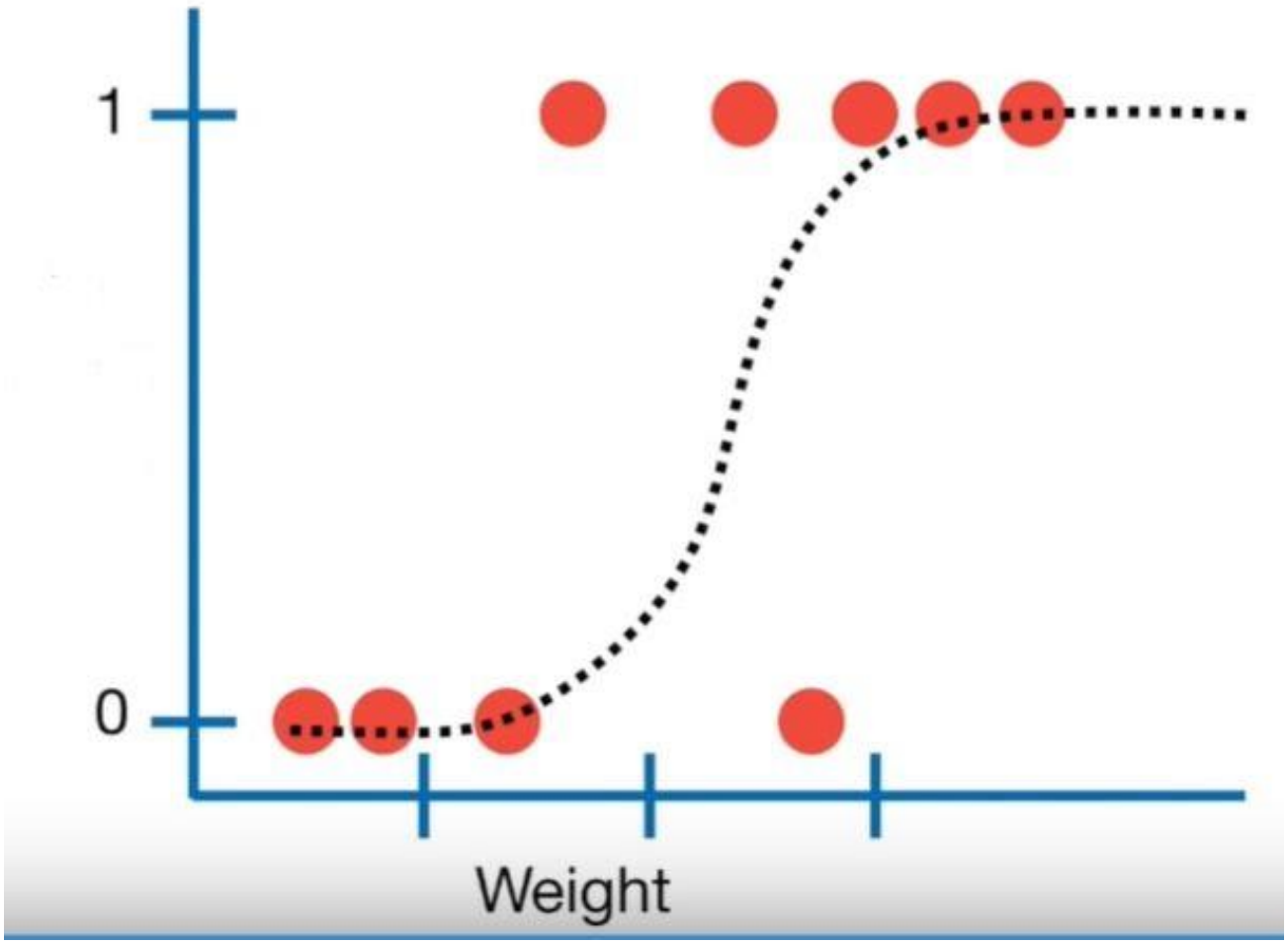
Pero, ¿Qué es lo que pasa si tenemos que predecir data Categórica?



En caso tengamos:

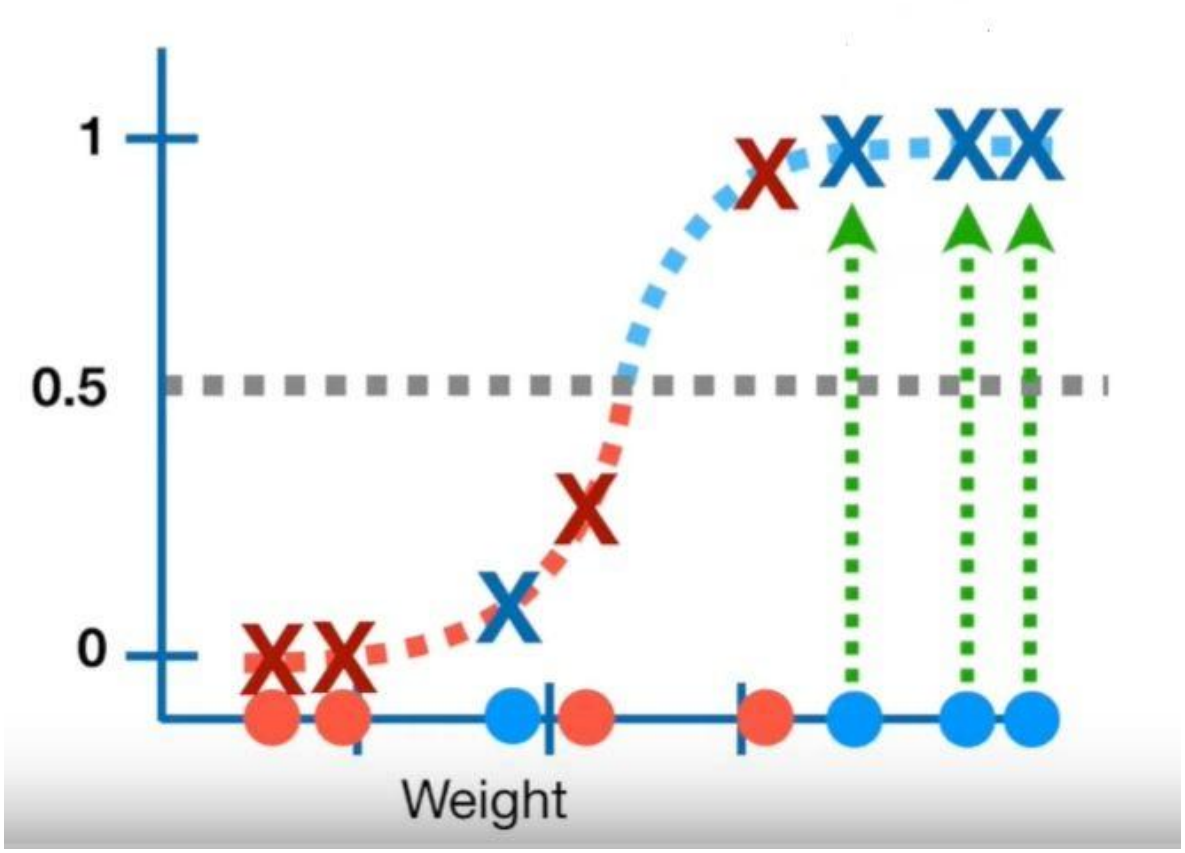
- Una variable predictora (Weight)
- Una variable dependiente binaria (si es obeso o no es obeso)

Regresión Logística



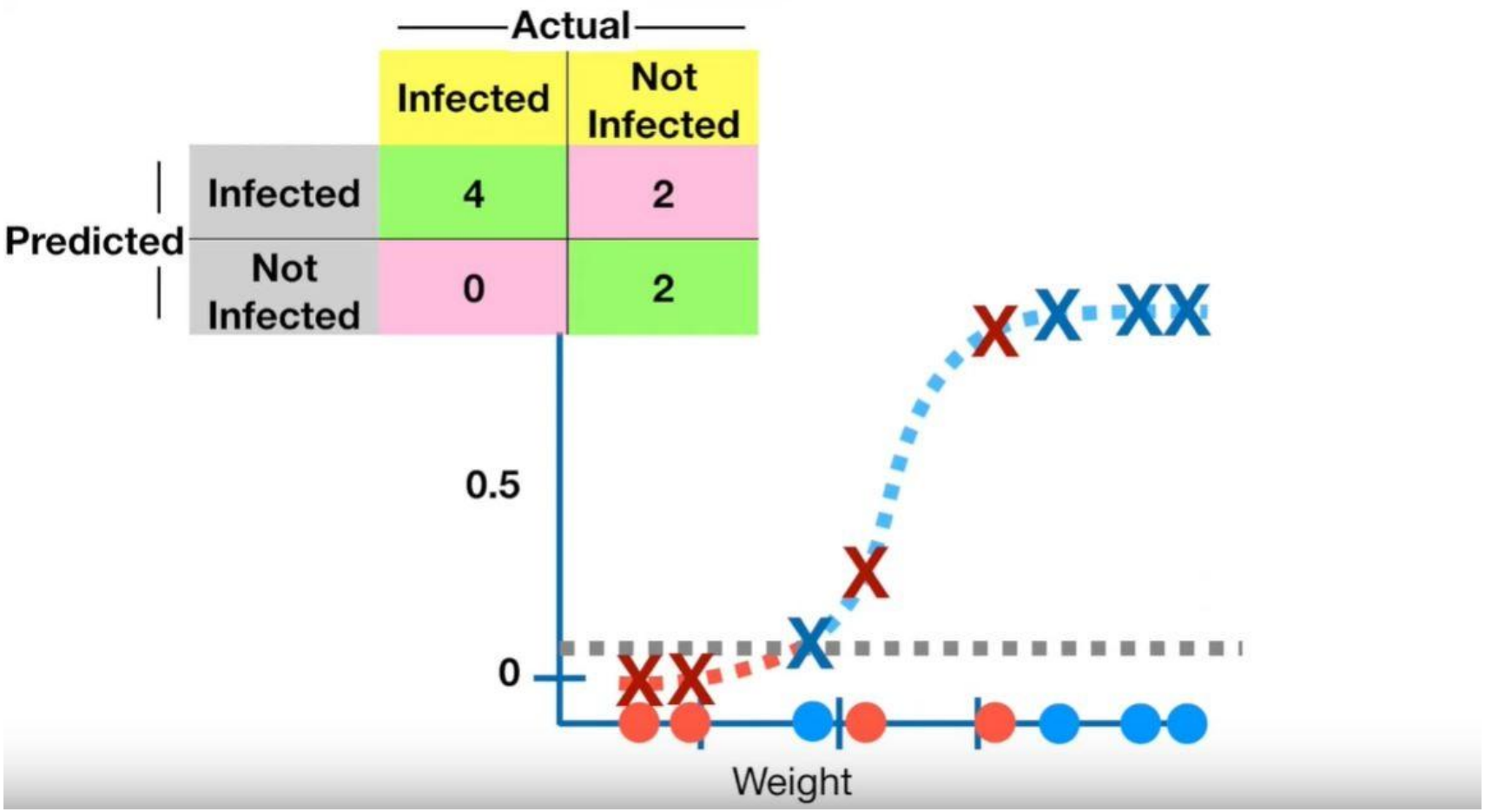
Es un modelo que nos va a botar la **probabilidad**, en este caso, de que un ratón sea obeso o no según su peso.

Podemos hacer ajustes al nuestra regresión logística y mejorarla.



- Los parámetros del modelo son varios y se pueden (y deben) hacer varios tests para ver si nuestro modelos es el mejor. Pero por ahora solo nos vamos a centrar en el punto de corte (threshold)

Matriz de confusión

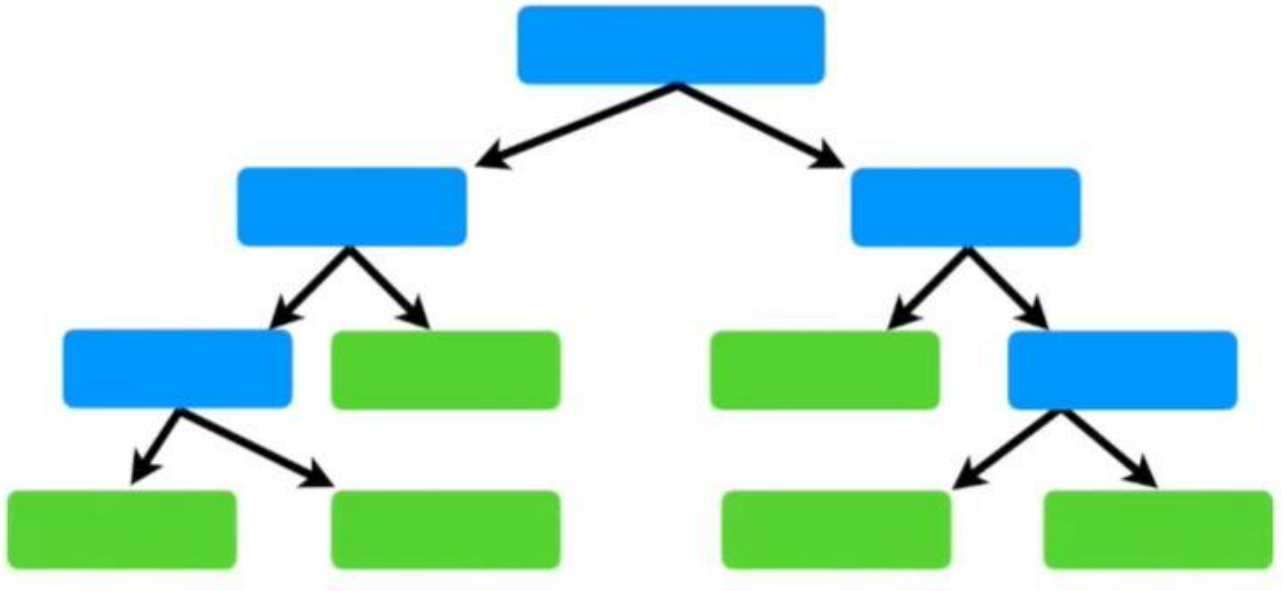


Imaginemos que movemos el threshold ¿ cómo se vería nuestra matriz de confusión?

boost
accelerate your learning

Arboles de Decisión

Nos clasifica, en forma de árbol, qué caminos o patrones sigue la data para que se pueda predecir cierta categoría (o cierto número)



Los elementos son:

- Root
- Nodes
- Leaf

El criterio más importante a saber es el grado de impureza (**Gini):** Que nos va a determinar qué tan mal es alguna de nuestras variables (columnas) para separar nuestros targets. Es decir, qué tan mal separa los 1 de los 0 en el resultado final.

El gini se calcula: **1 - (probabilidad|1)^2 - (probabilidad|0)^2**

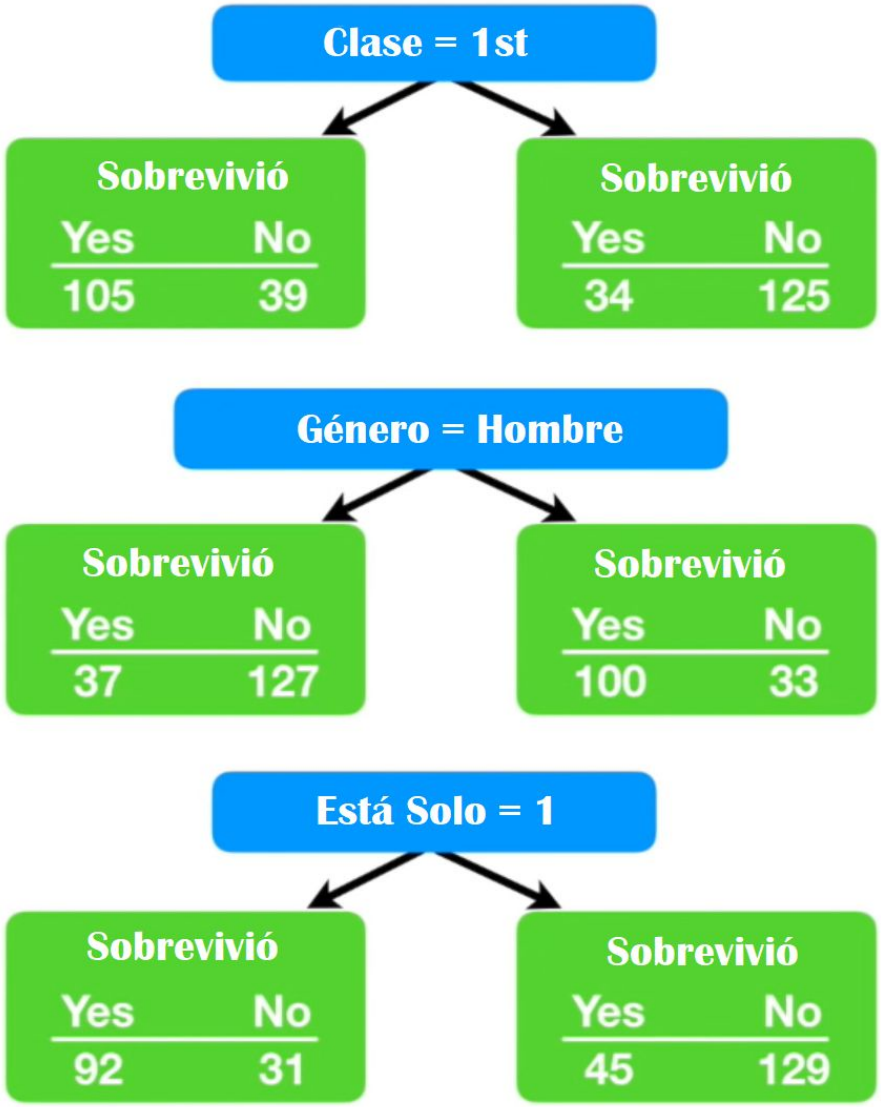
Arboles de Decisión:Un proceso iterativo

$$\left(\frac{144}{144+159}\right) 0.395 + \left(\frac{159}{144+159}\right) 0.336$$

Nuestro árbol va a iterar por sobre cada columna y calcular el Gini en base a los diferentes valores que tengamos:

- Gini (Clase=1): 0.364
- Gini (G=Hombre):0.360
- Gini(EstáSolo=1):0.381

Recuerda que cada leaf tiene su impureza propia y los Gini de arriba son un promedio ponderado de ellos. Aparte, cada árbol irá eligiendo para cada nodo la variable que tenga **menor Gini.**



boost
accelerate your learning

Arboles de Decisión:Un proceso iterativo

Básicamente, la dinámica del árbol se divide en estos pasos:

1.- Se calculan todos los ginis de las variables predictoras y se elige la variable que tenga menor Gini como root (o como nodo a partir de la segunda iteración).

2.- Si el nodo padre tiene menor impureza que los nodos hijos, ese nodo padre “aborta” a sus hijos y se convierte en un **leaf node,** es decir, este padre ya no tendrá hijos (se eliminan las ramas futuras porque aumentarían la impureza).

3.-Si el nodo hijo tiene menor impureza, entonces se queda y se vuelve a iterar con respecto a todas las variables restantes.

boost
accelerate your learning

Arboles de Decisión:Un proceso iterativo

Parámetros dentro de Scikit-Learn

Para DecissionTreesClassifier:

criterion: Nos dice qué criterio va a tener nuestro clasificador. Gini o entropy.

splitter: Para cortar la rama en el mejor clasificador o en forma random (no recomendado alterarlo)

max_depth: Para ver qué tanto los nodos se expande, por default está en None, osea que se expanden hasta que sean los mejores (riesgo de overfitting)

min_impurity_decrease: Para modificar el criterio de decrecimiento del gini, es decir, si cae en 0.2 entonces splitea. Por default está en 0. Osea apenas decaiga, mejora.

max_features: el número de features que escoge para buscar el mejor split (recuerda que al principio se calcula con todas las columnas)

Atributos dentro de Scikit-Learn

Para DecissionTreesClassifier:

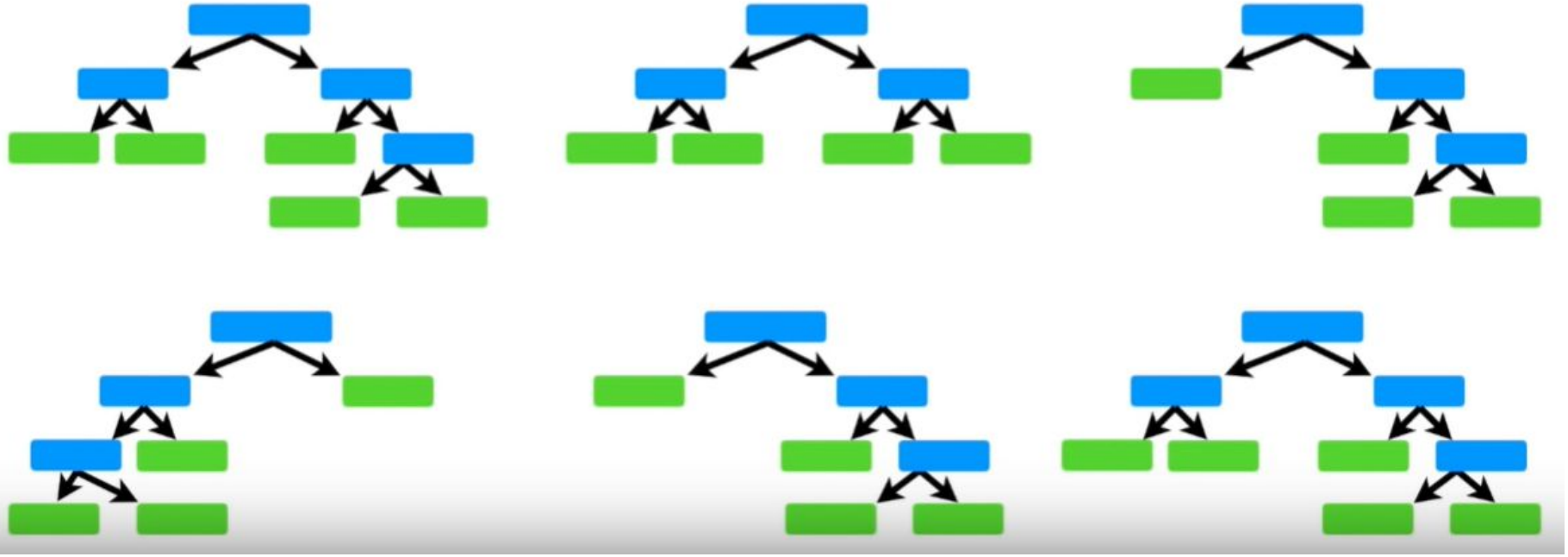
.feature_importances_ : Nos da el tan ansiado feature selection que podemos hacer con los árboles de decisión.

.max_features_: Nos bota el entero de máximo de features que optimizaría nuestro algoritmo.

.tree_: nos bota la estructura del árbol que hemos entrenado (visualmente en matriz)

boost
accelerate your learning

Random Forest: Una introducción



Es un conjunto de árboles entrenados con diferentes partes de nuestra data. Estos árboles botan en conjunto como un bosque, es decir, cada uno bota su opinión de si nuestra variable será 0 u 1. Dependiendo de esta opinión se va a elegir si hay más probabilidades de que sean 0 o 1.

boost
accelerate your learning

Random Forest

- **Ventaja**
  - Más robusto (evita el overfitting)
  - Nos dice mejor qué variables son las mejores.
  - En otros lenguajes (R) nos puede trabajar con Nulos porque los reemplaza automáticamente.
- **Desventaja**
  - No se puede seguir la lógica como en un árbol de decisión.
  - El proceso de entrenamiento puede demorar mucho.

Pasos generales para Random Forest

1. Hace una muestra Boostrap con reposición de la data que usaremos. (filas y columnas)
2. Crea un árbol por cada muestra
3. Se repite el paso 1 y 2 hasta obtener el bosque.

boost
accelerate your learning

Random Forest

Parámetros dentro de Scikit-Learn

Para RandomForestClassifier: (Sólo están incluidas las exclusivas para randomforest, todos los demás parámetros son los mismos que para árboles)

n_estimators: Nos da las estimaciones totales de árboles que tendremos en nuestro forest.

bootstrap: Lo dan como True por default, pero también se puede usar la data entera para entrenar cada árbol del forest.

n_jobs: Nos da el número de paralelizaciones que queremos usar, el parámetro **-1** nos dice que usaremos cada núcleo de nuestra computadora para procesar los árboles.

boost
accelerate your learning

Curva ROC