

DATA SCIENCE



Federico Baiocco
baioccofede@gmail.com
3512075440



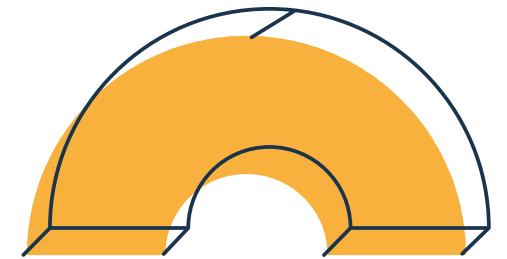
Clase 24 - Agenda

SUPPORT VECTOR MACHINE



SVM

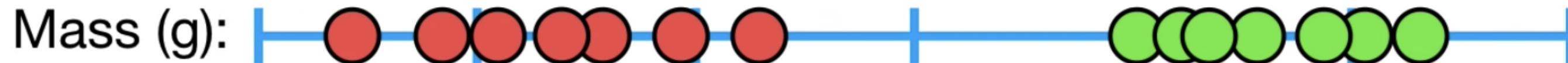
Support vector machine

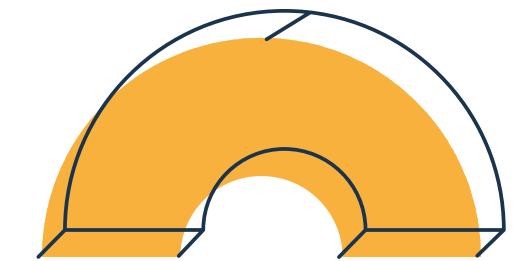


Tenemos el siguiente problema: queremos clasificar ratones como obeso/no obeso de acuerdo a su masa.

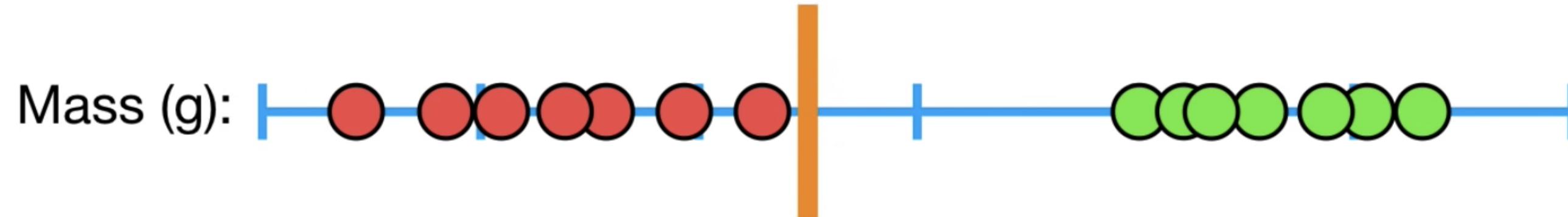
Los puntos verdes representan ratones obesos.

Los puntos rojos representan ratones que no son obesos.

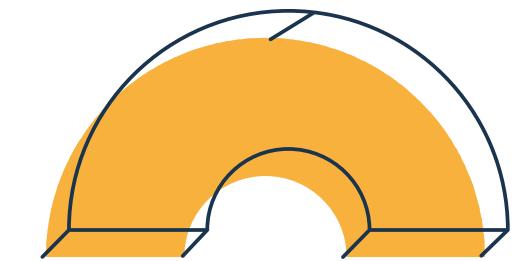




Basándonos en estas observaciones, podemos seleccionar un threshold y cuando obtenemos una nueva observación con menor masa que el threshold, decimos que el ratón no es obeso. Cuando obtenemos una observación con más masa que el threshold, decimos que el ratón si es obeso.

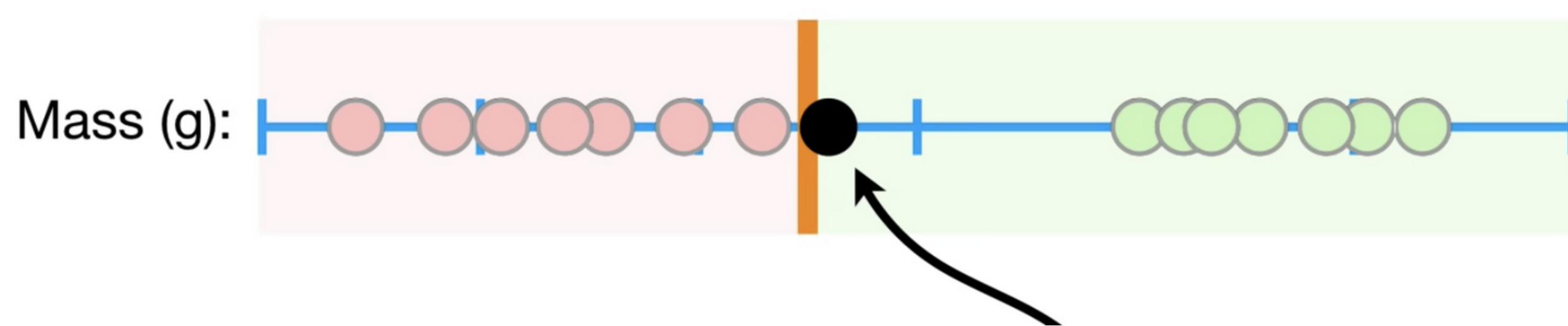


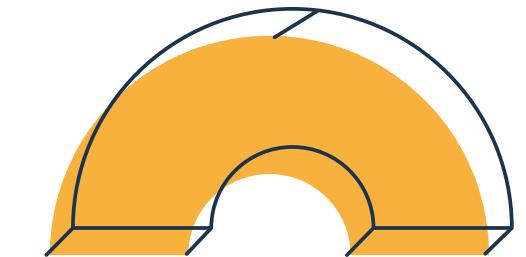
SVC



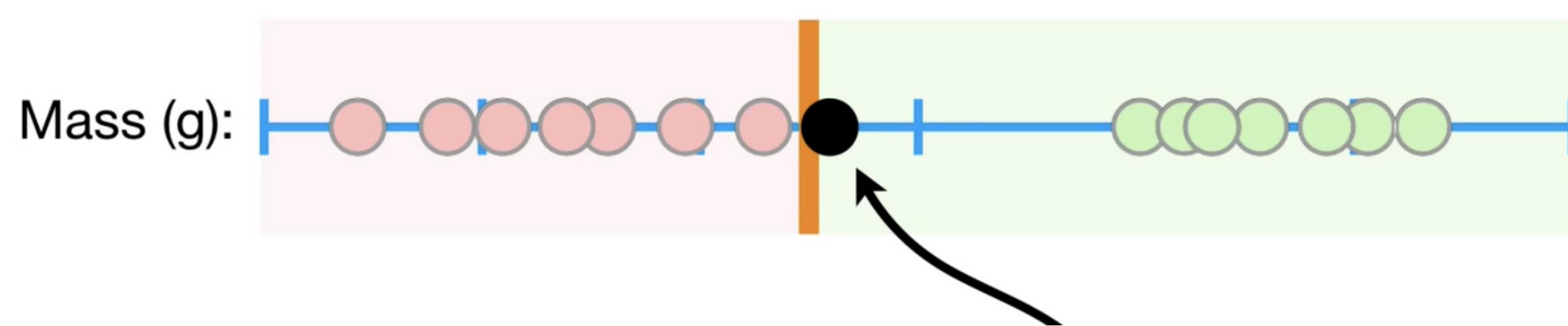
Pero..

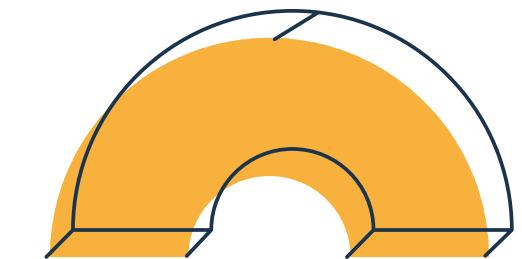
¿Qué pasa si obtenemos una nueva observación como la siguiente?



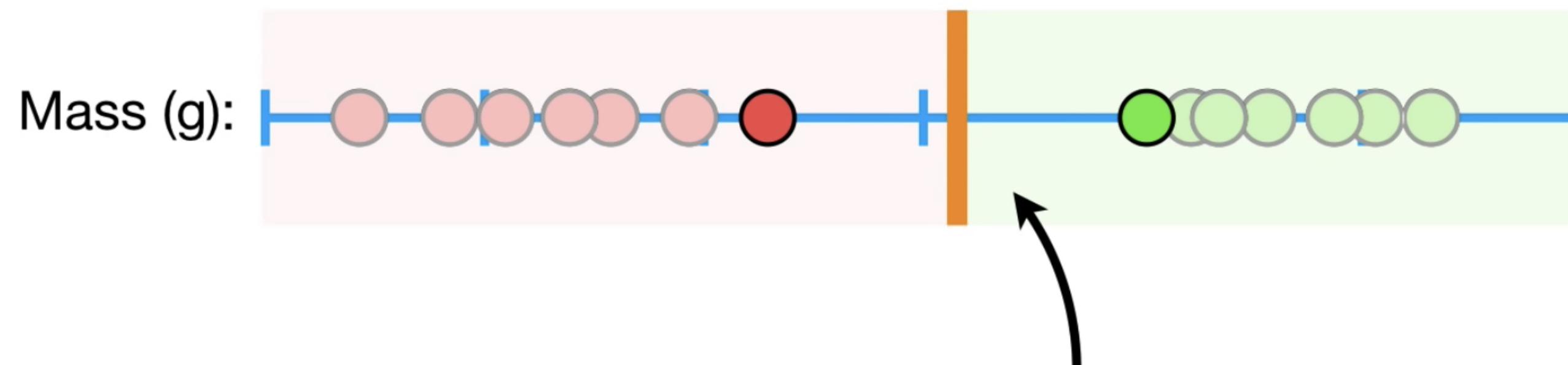


Clasificaríamos al ratón como obeso, pero no tiene mucho sentido ya que se puede ver que está mucho más cerca de los ratones que no son obesos.



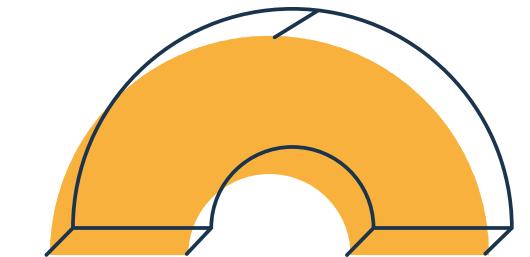


Podríamos definir un mejor threshold basándonos en las observaciones que están en los límites de cada grupo:

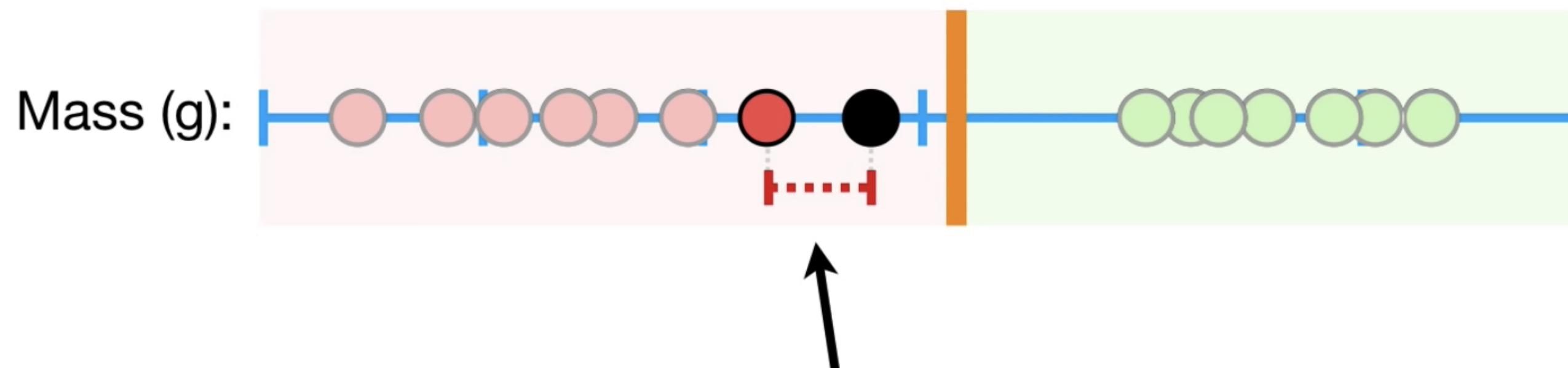


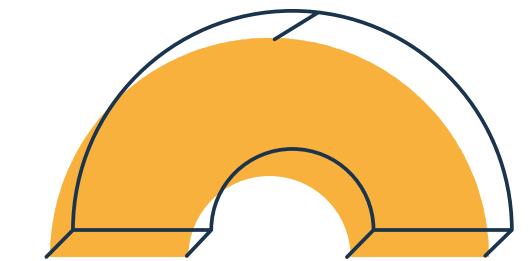
Utilizamos el punto medio entre ambos como threshold

SVC



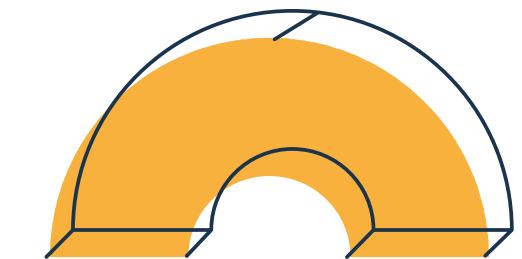
Ahora, las observaciones que queden del lado izquierdo del threshold, van a ser más cercanas a los ratones que NO son obesos



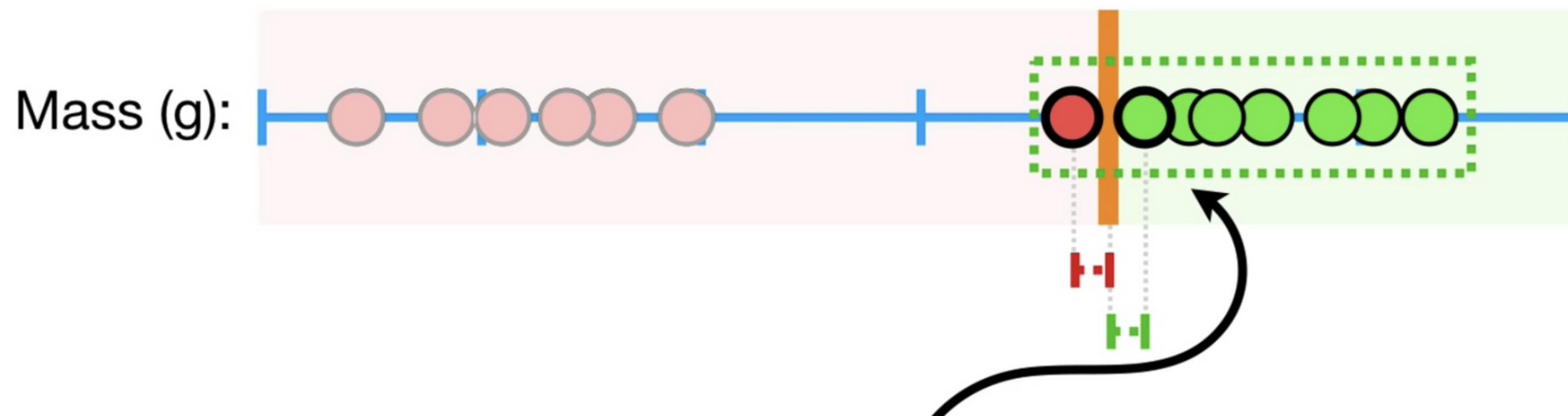


De todas formas, esto funciona en el set de datos anterior. Pero imaginen que nuestro set de entrenamiento tiene un outlier y se ve como el siguiente:

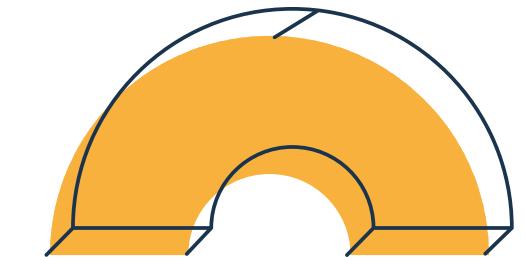




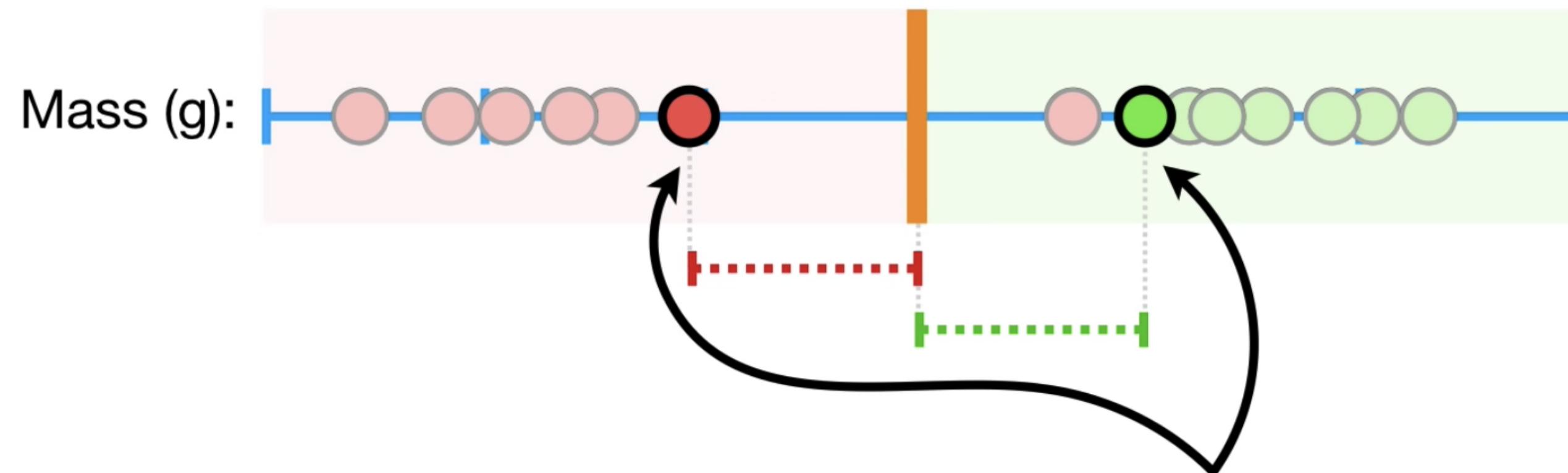
En este caso, nuestro threshold sería muy cercano a las observaciones obesas



Este tipo de clasificadores es muy sensible a outliers en el set de entrenamiento, esto hace que no sean muy útiles.

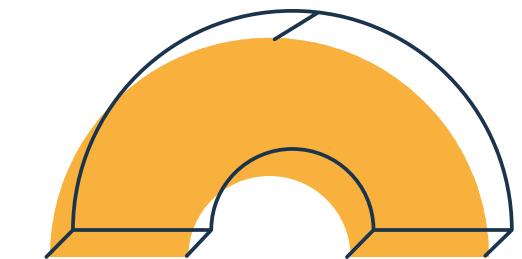


Para definir un threshold que nos sea útil, debemos permitir que haya algunas observaciones mal clasificadas.



Podemos definir por ejemplo, un threshold que esté entre medio de estas 2 observaciones. De esta forma estamos clasificando mal una única observación, pero el threshold hace mucho más sentido.

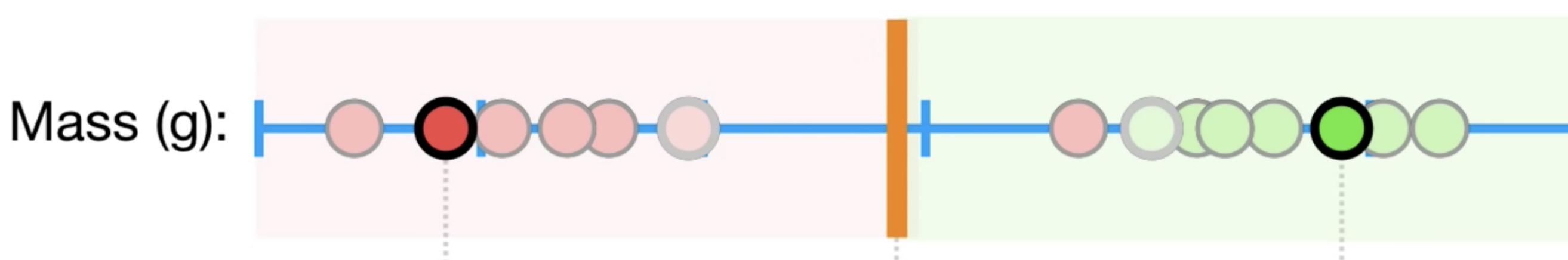
SVC



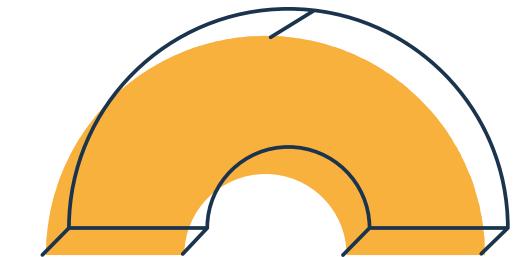
¿Cómo podemos saber si es mejor definir un threshold como este?



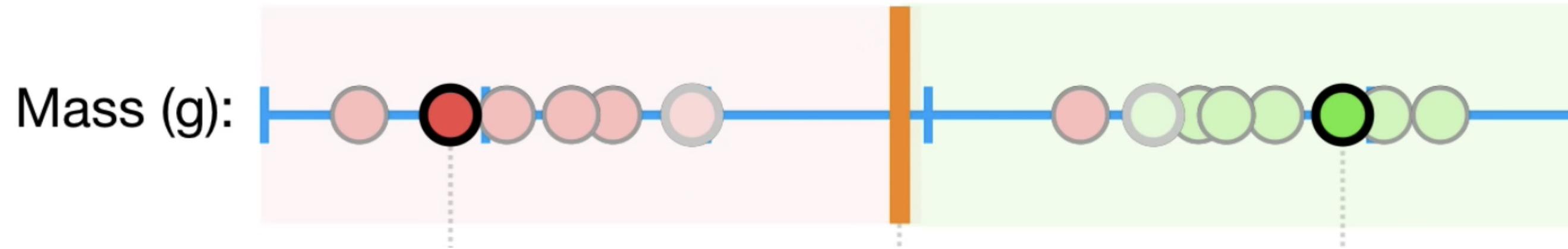
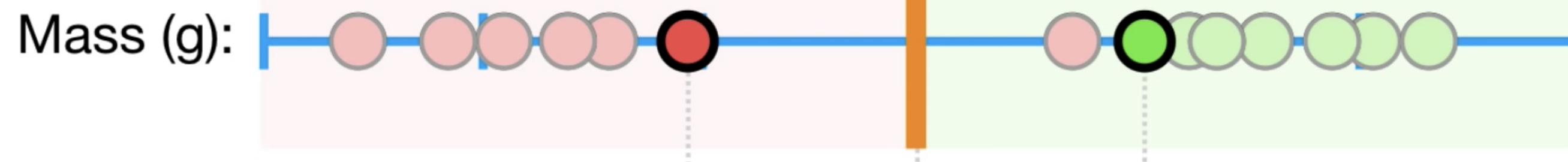
¿O este?



SVC



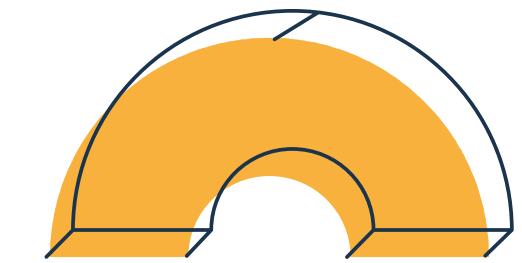
Es un hiperparámetro que podríamos encontrar por ejemplo haciendo GridSearchCV.



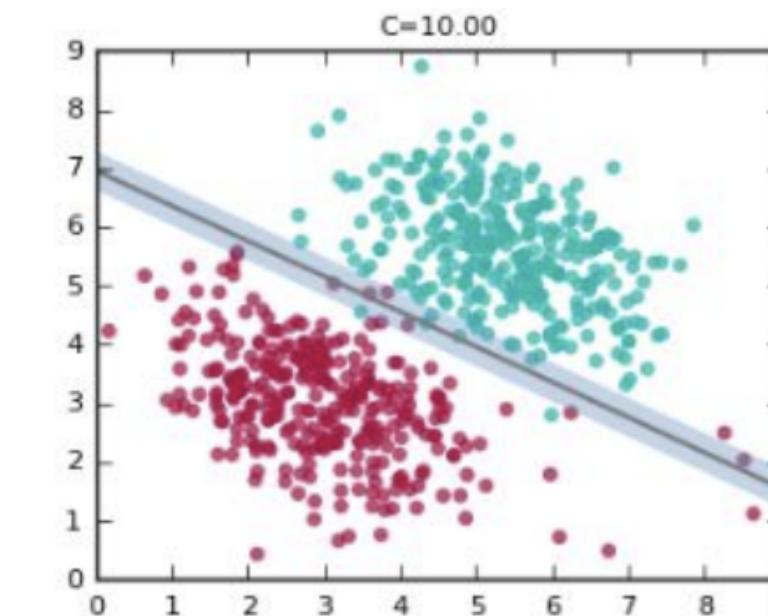
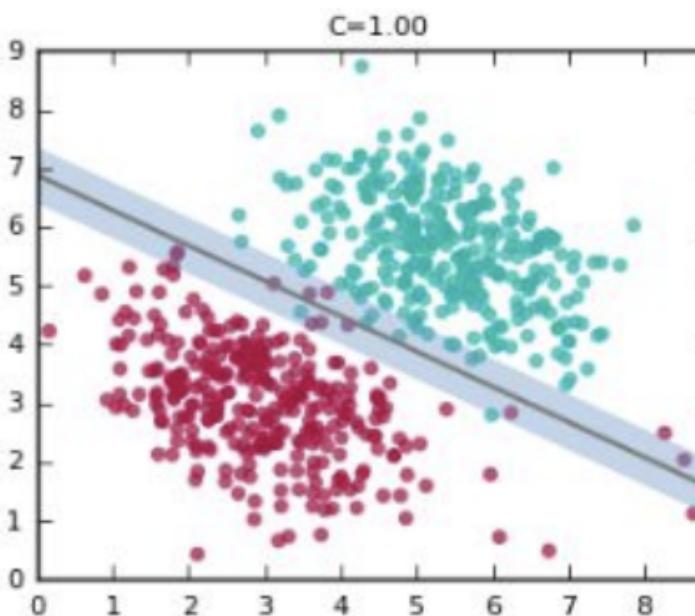
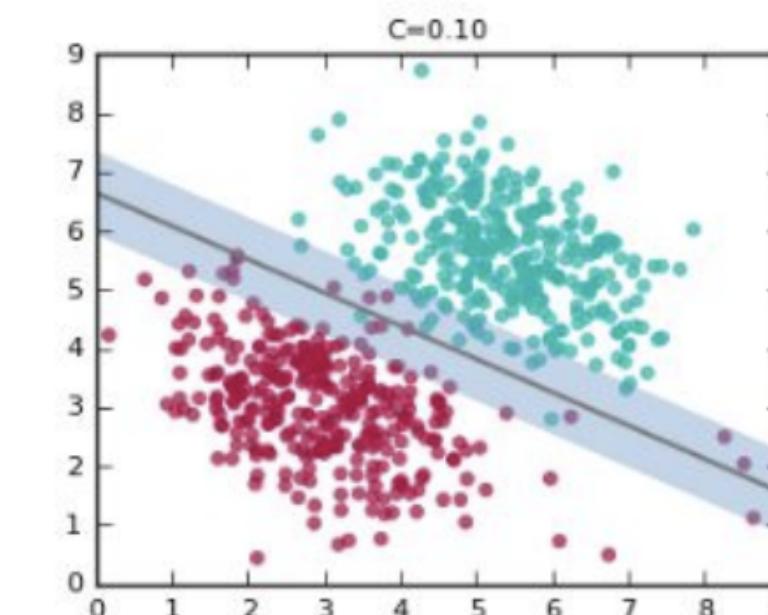
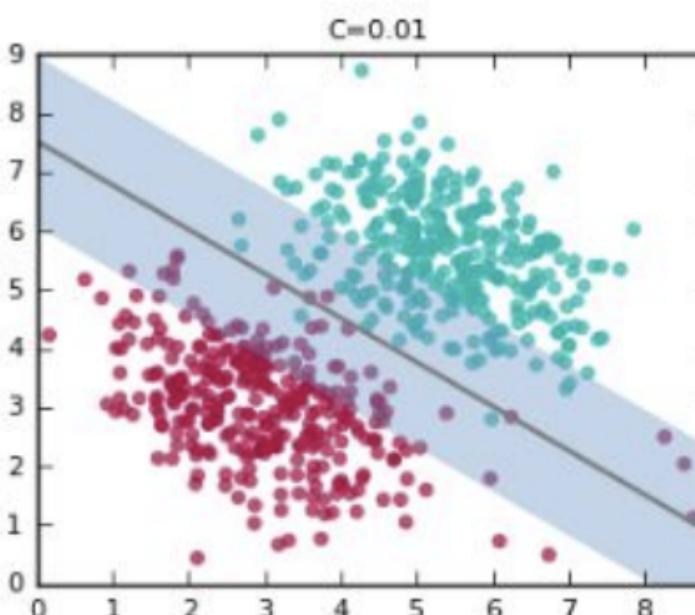
El hiperparámetro se llama C.

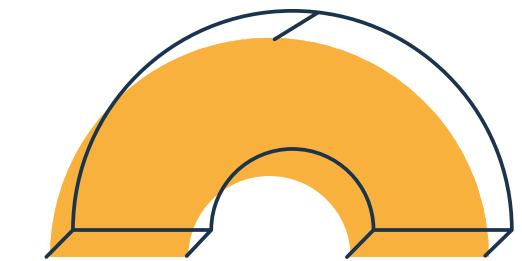
Mientras mayor sea C, menos errores estaremos permitiendo sobre los datos de entrenamiento.

SVC

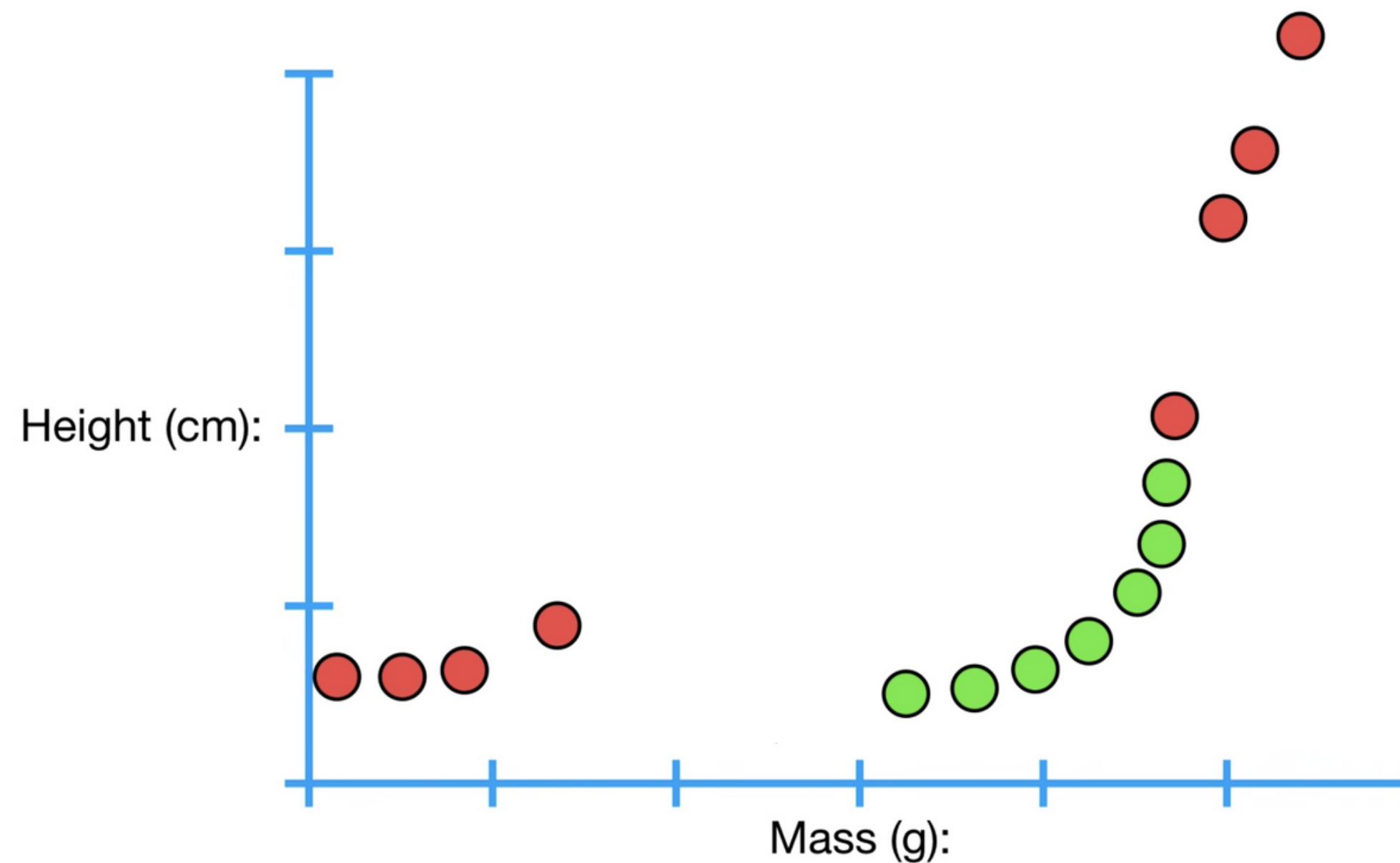


Hiperparámetro C:

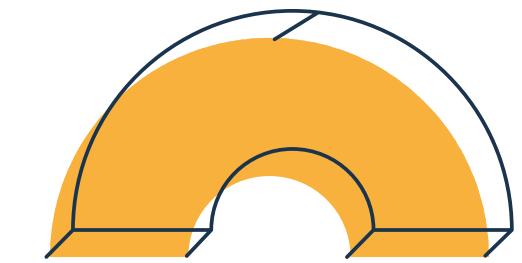




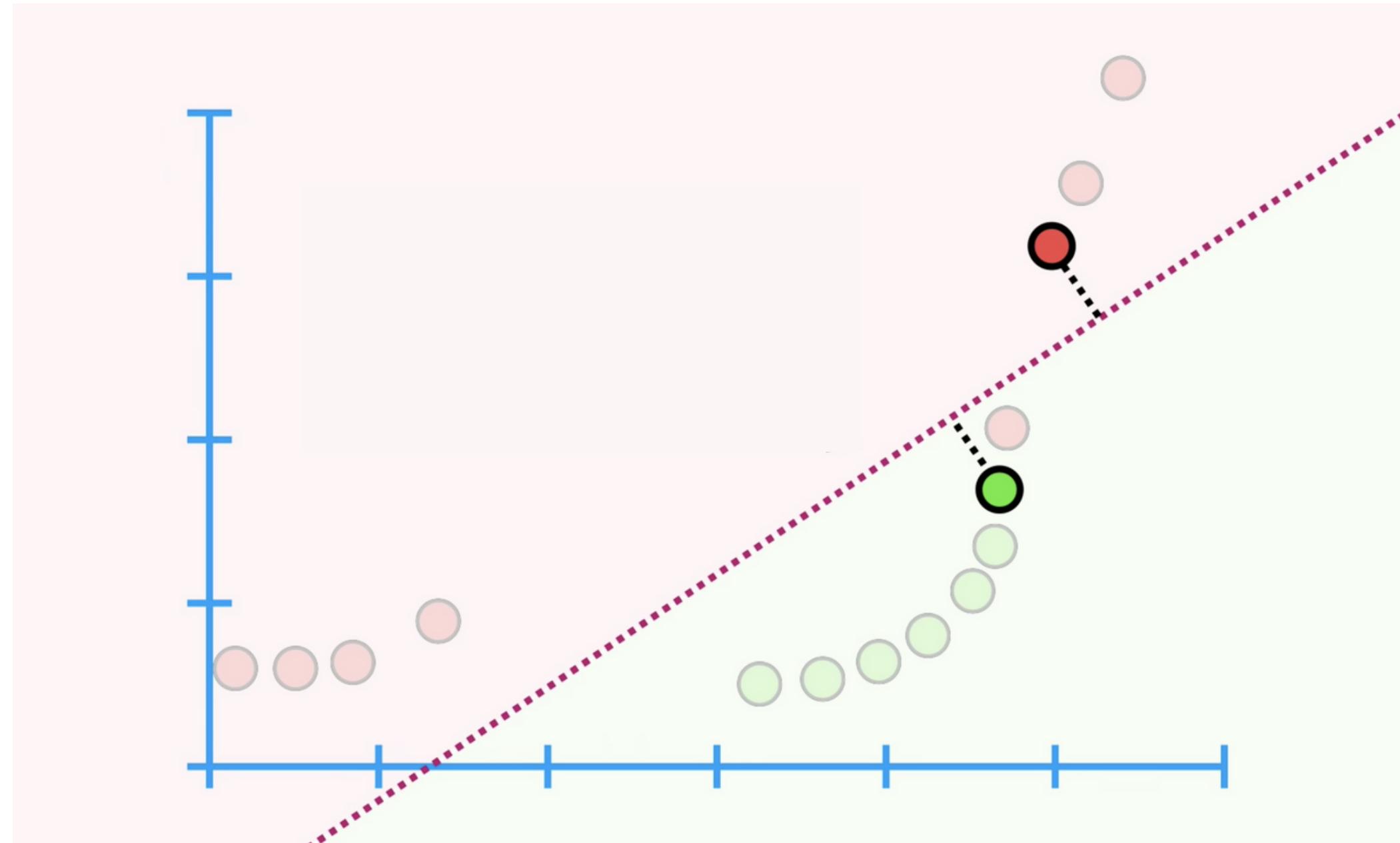
Ahora, si para cada observación tenemos masa y altura, nuestros datos serían de 2 dimensiones.



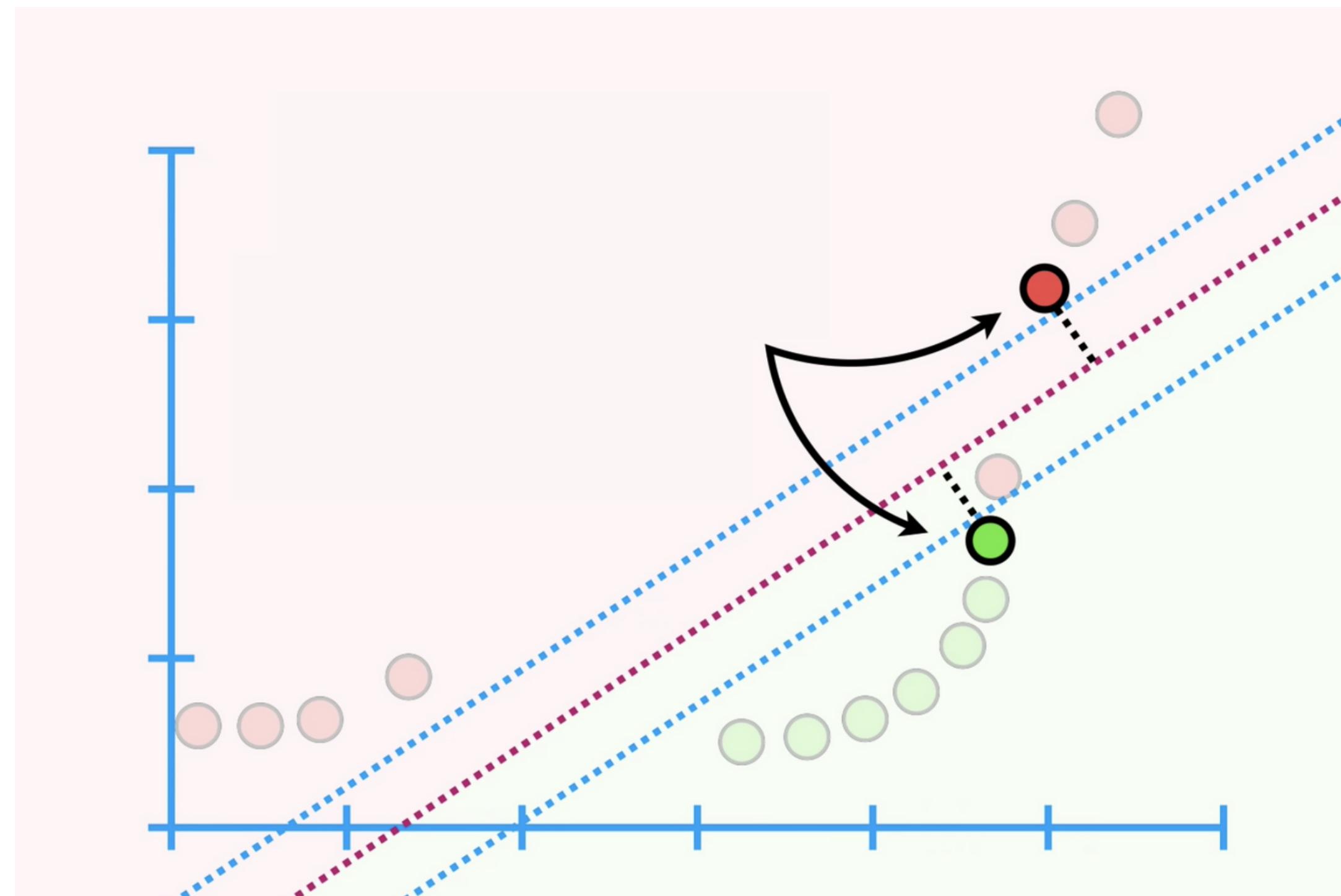
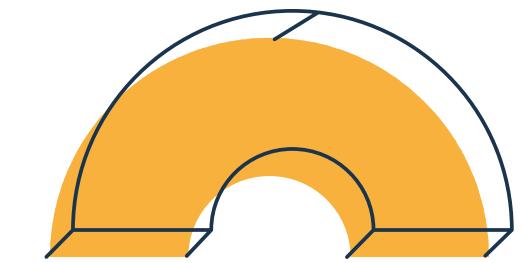
SVC

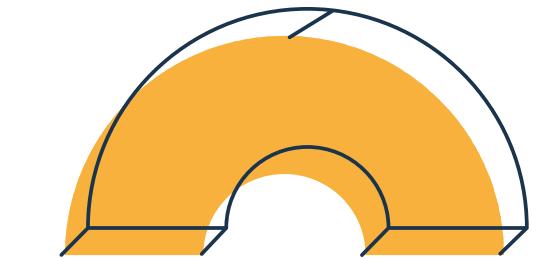


En este caso, nuestro threshold pasa a ser una linea

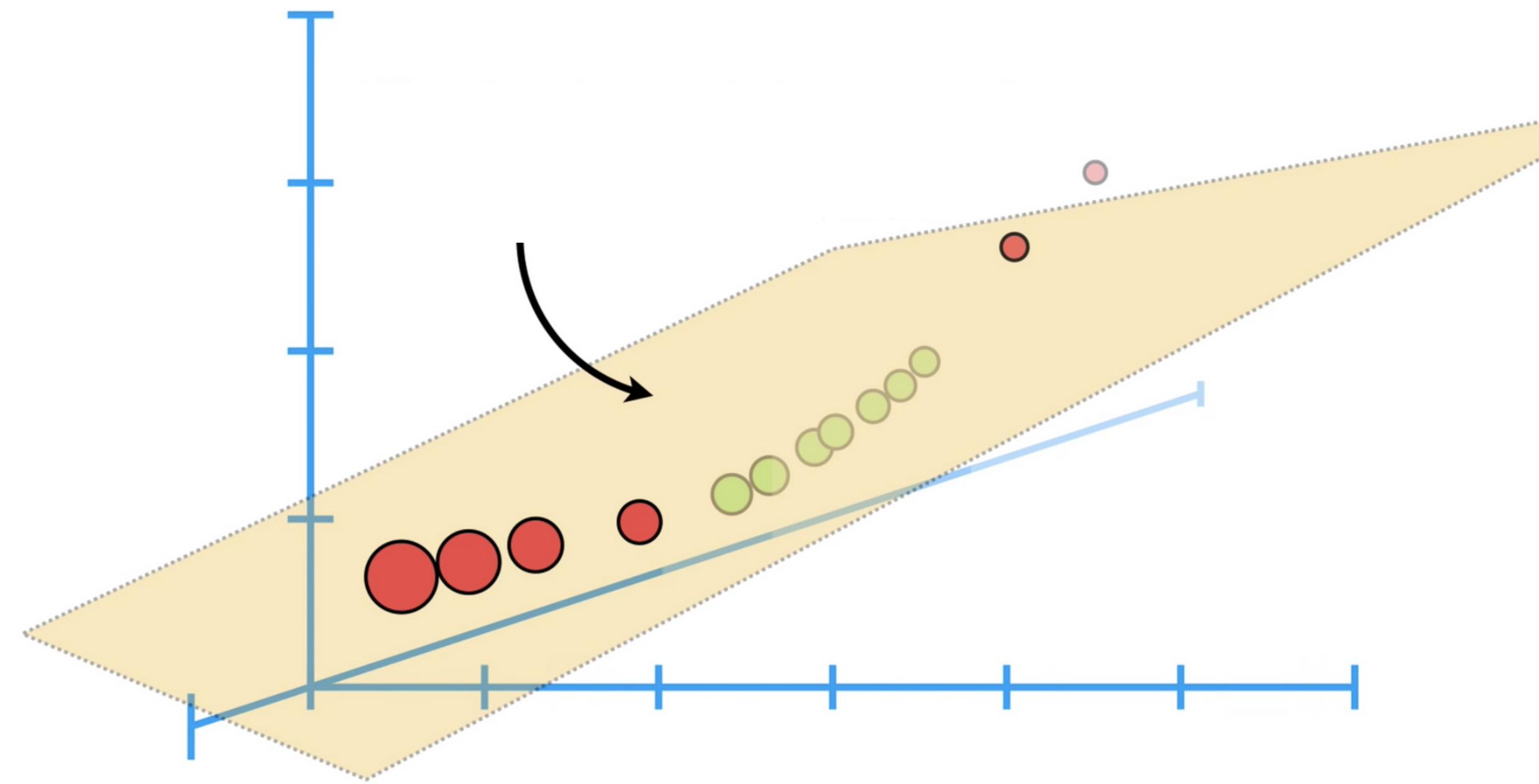


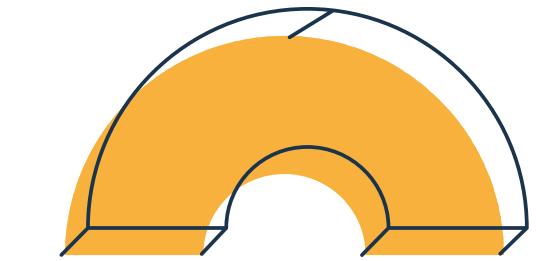
SVC



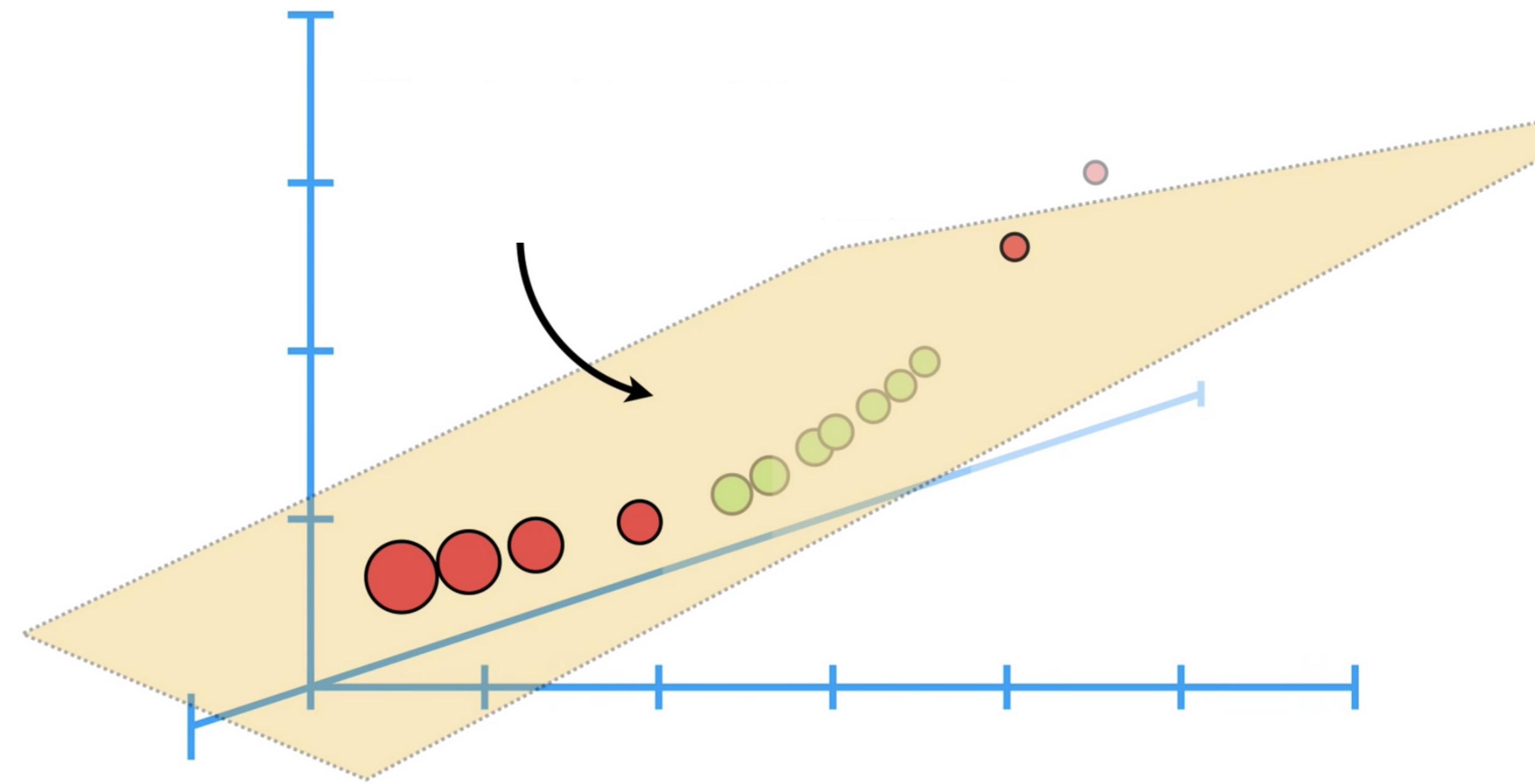


En este caso, imaginen que tenemos peso, altura y edad. Tenemos 3 dimensiones y nuestro SVC forma un plano en lugar de una linea.

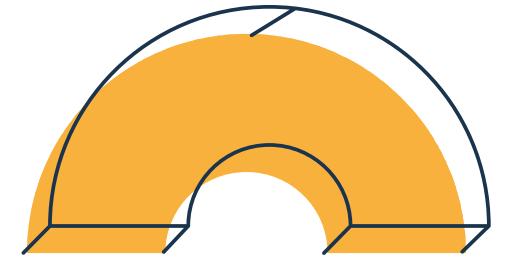




En este caso, imaginen que tenemos peso, altura y edad. Tenemos 3 dimensiones y nuestro SVC forma un plano en lugar de una linea.



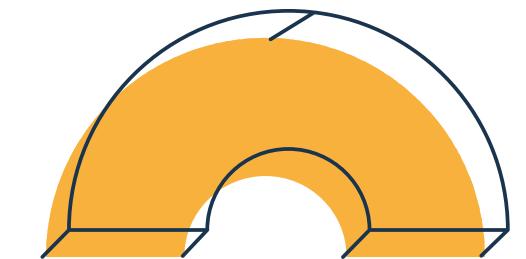
SVC



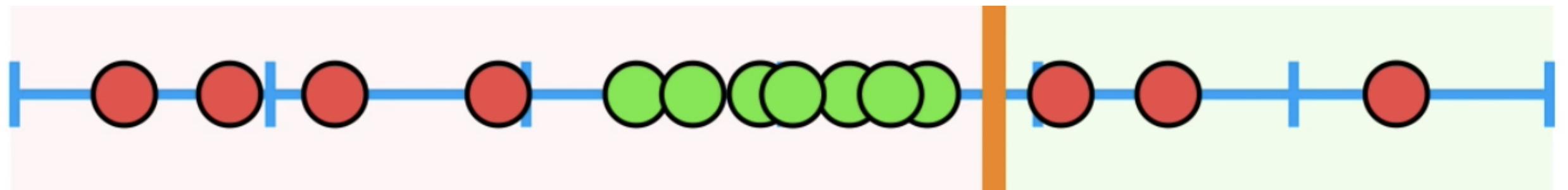
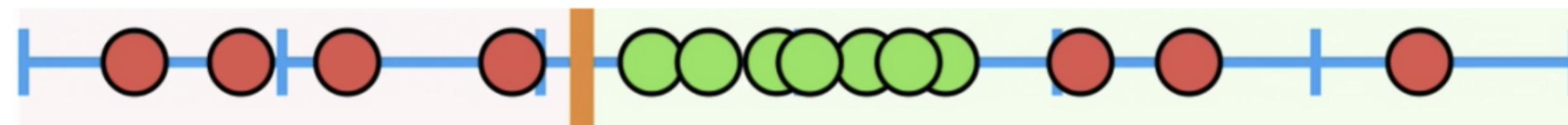
¿Qué pasa si nuestros datos tienen esta forma?



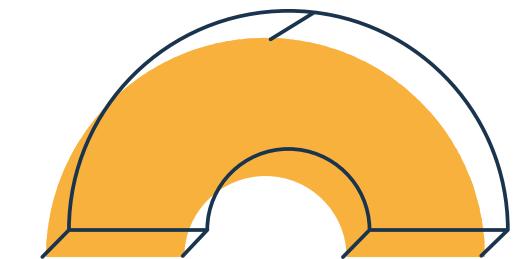
SVC



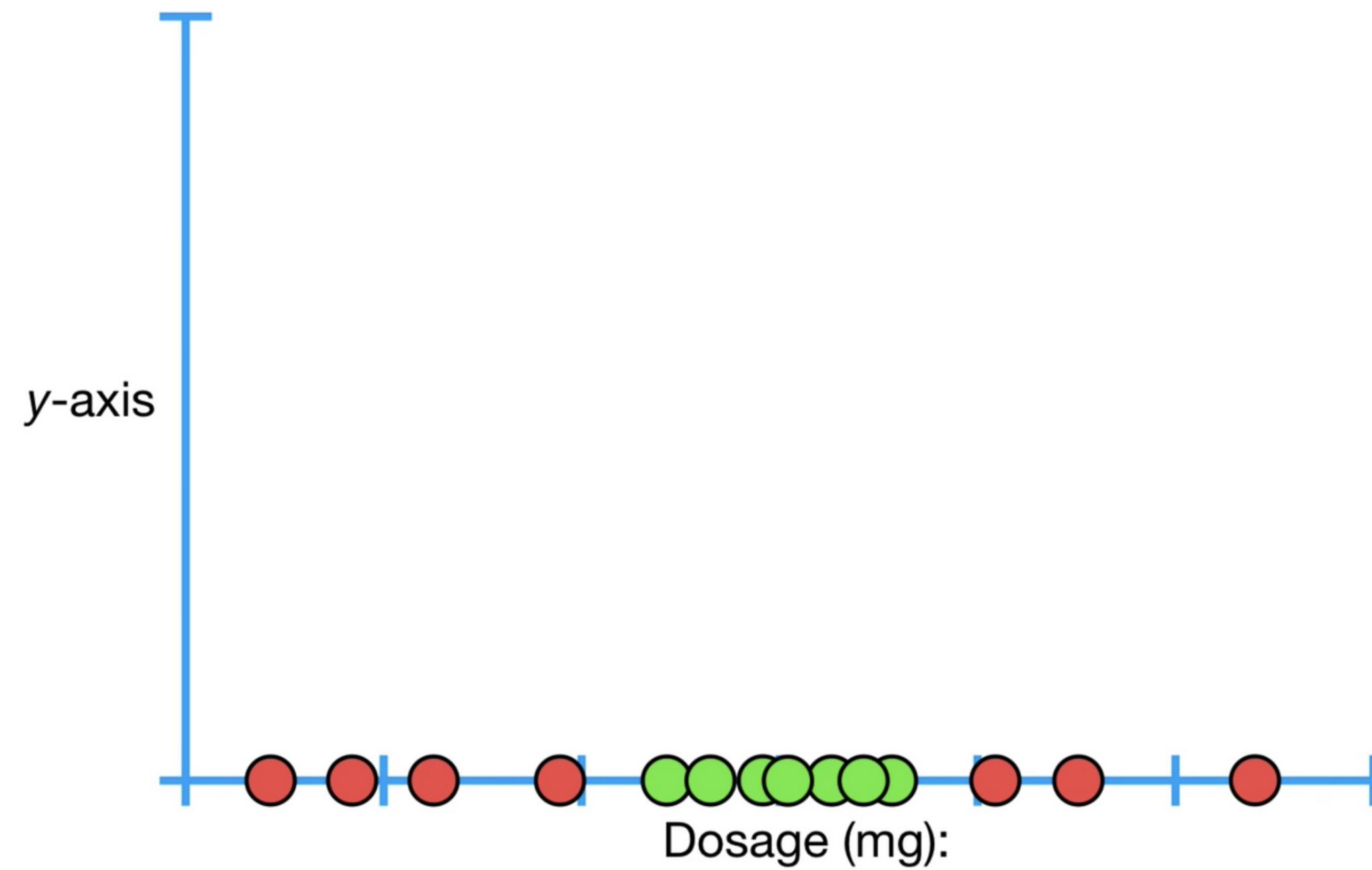
¿Donde ponemos el threshold?

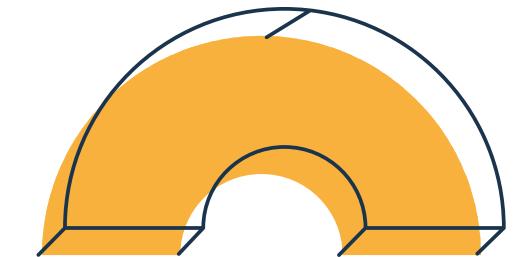


SVC

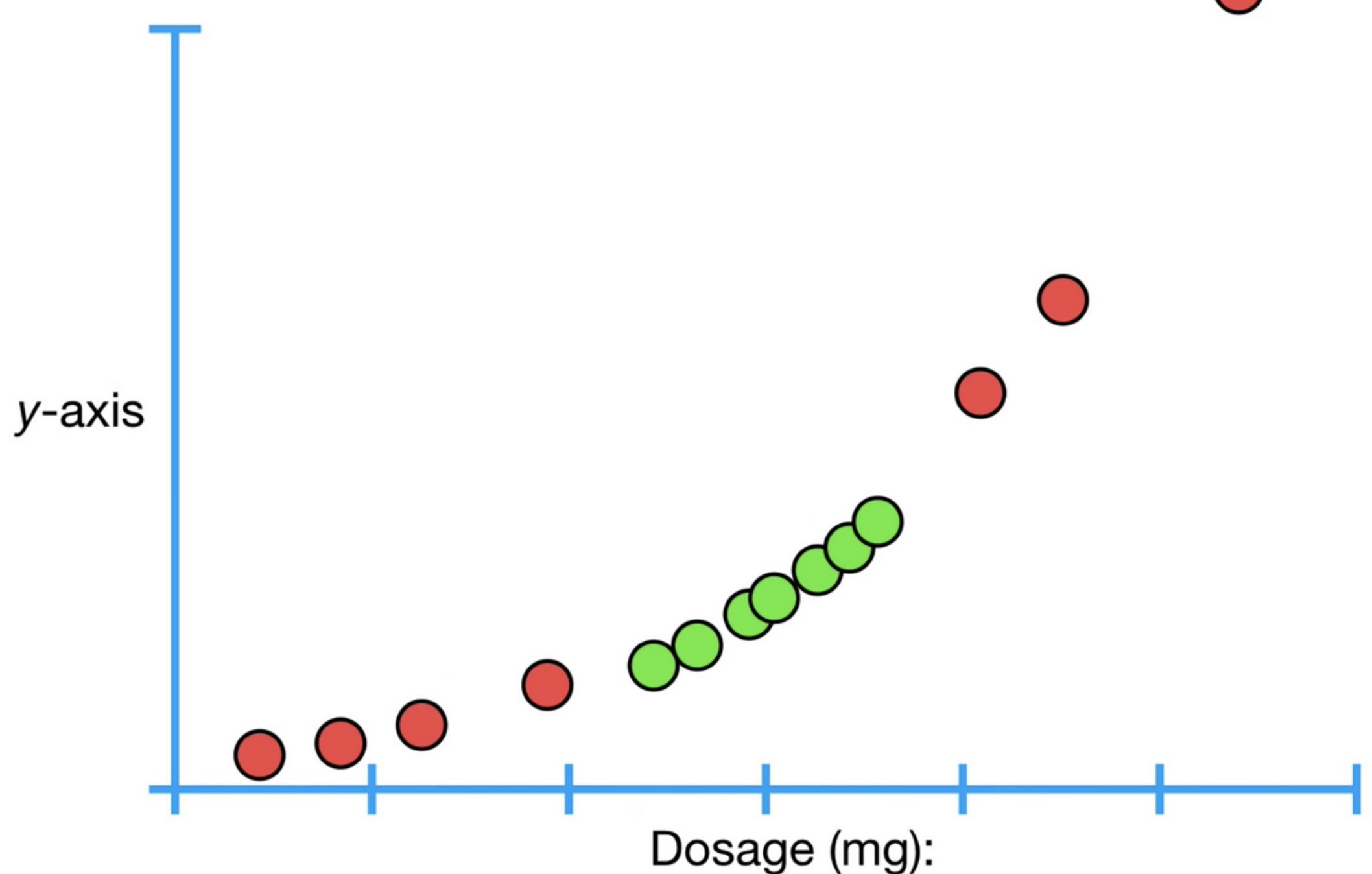


Para solucionarlo, arranquemos agregando un eje Y:

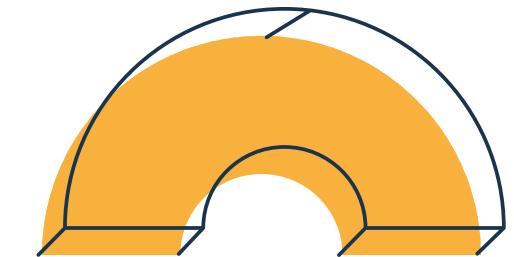




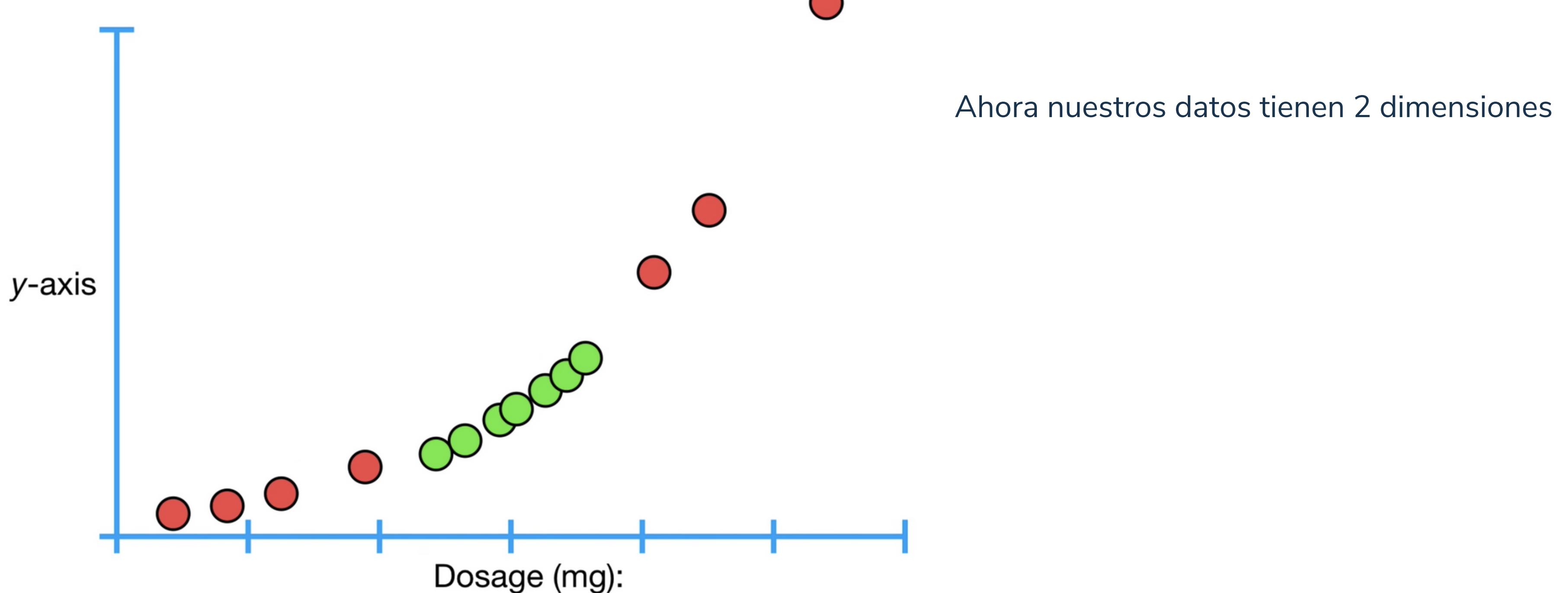
Ahora, en el eje de las X mantenemos los valores y para el eje de las Y, tomamos el valor de X al cuadrado



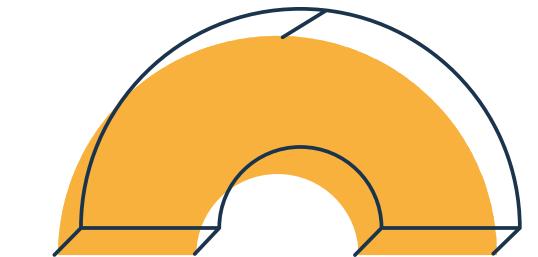
Ahora nuestros datos tienen 2 dimensiones



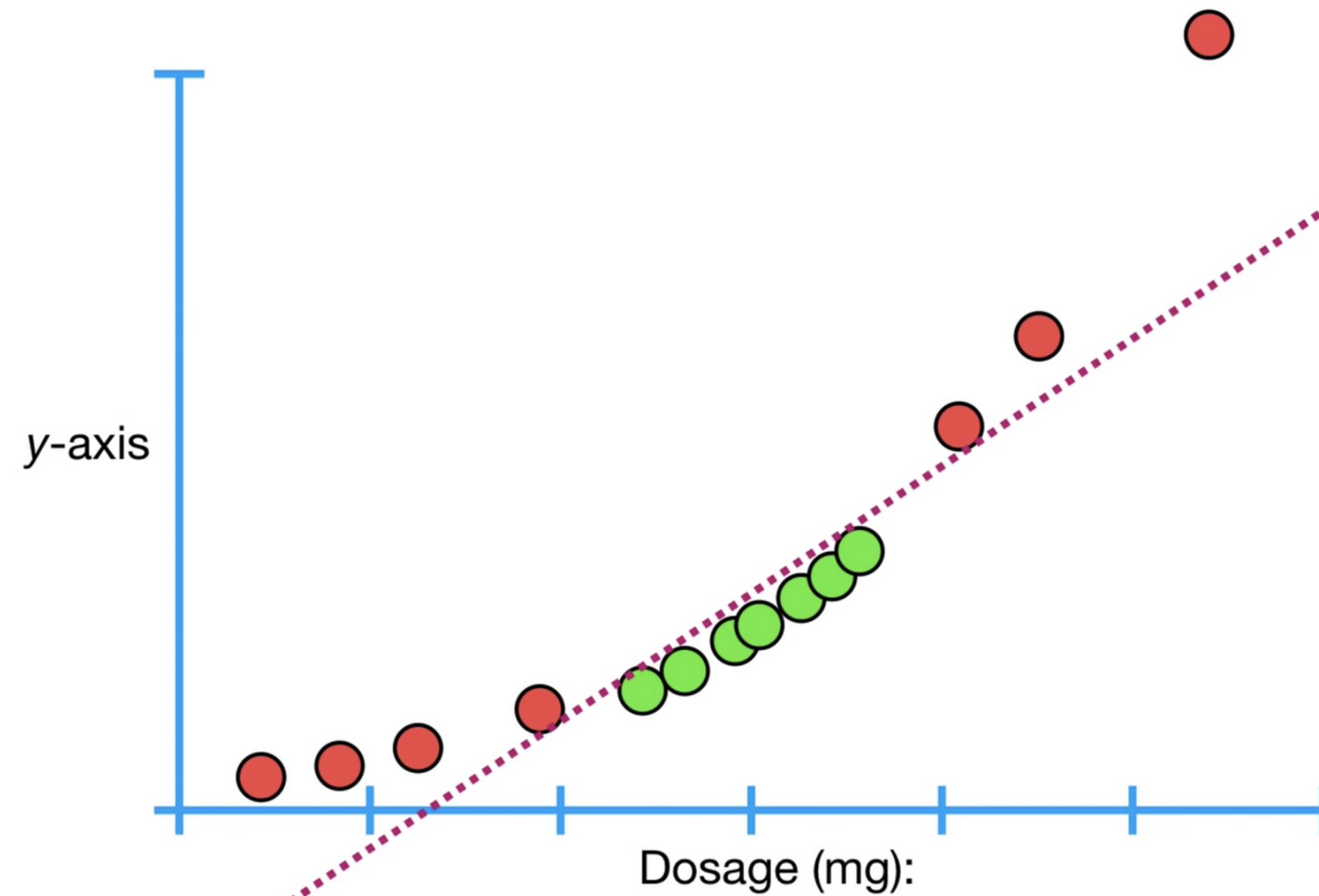
Ahora, en el eje de las X mantenemos los valores y para el eje de las Y, tomamos el valor de X al cuadrado



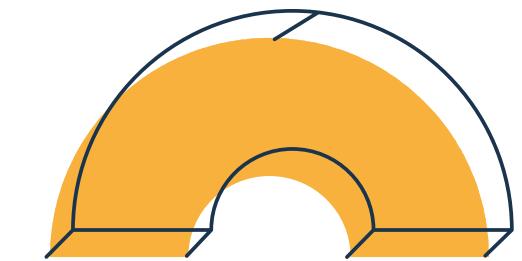
SVC



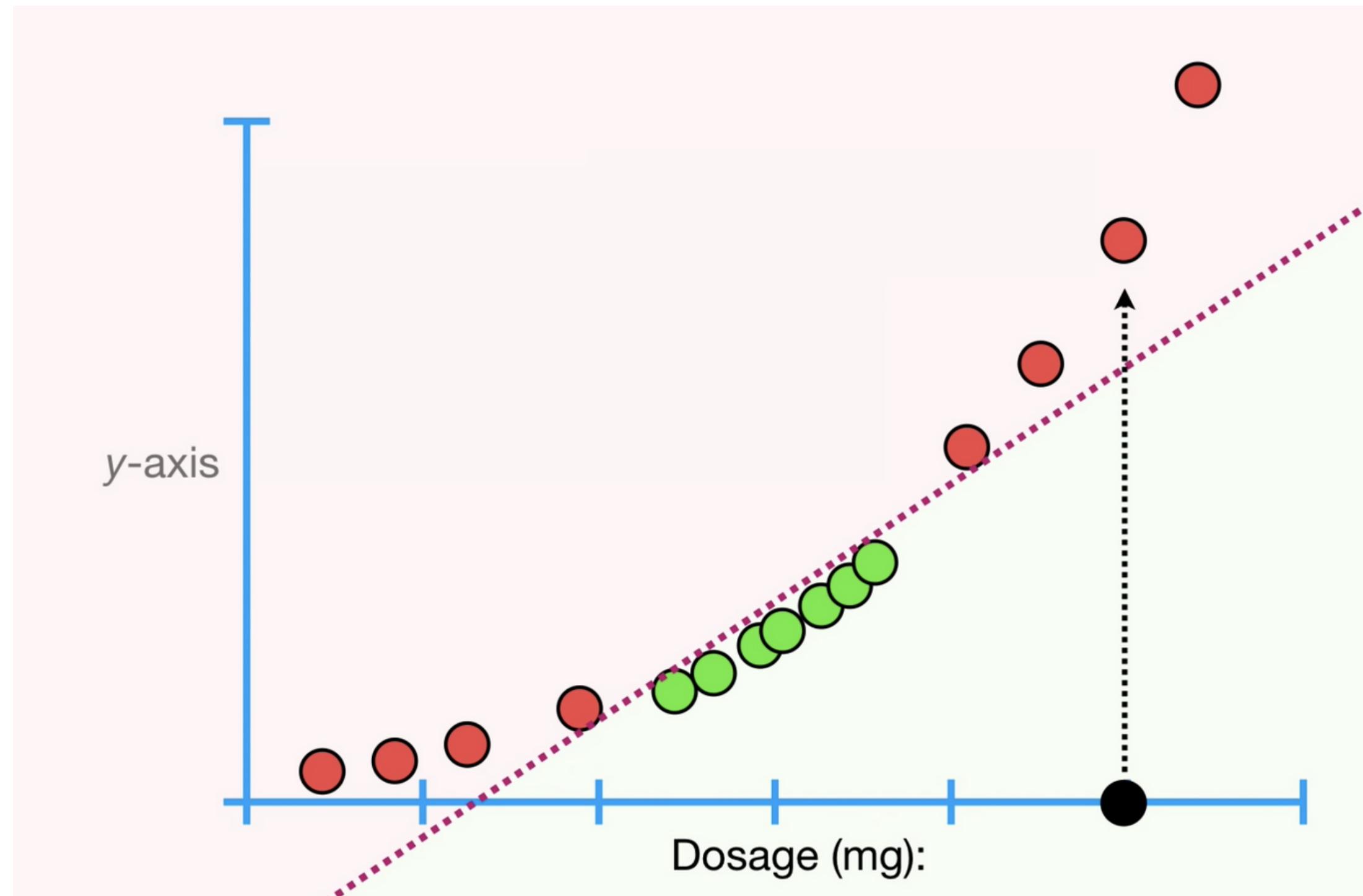
De esta forma, podemos obtener un support vector classifier que clasifique nuestros datos correctamente

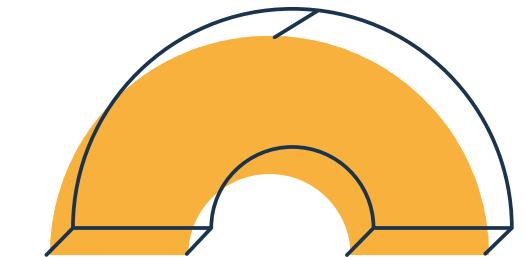


SVC



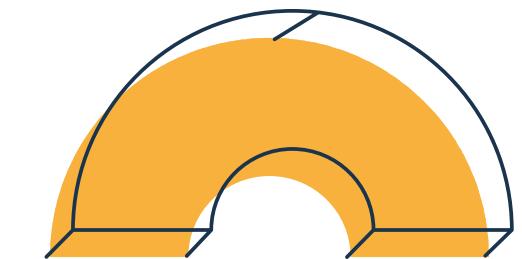
Ahora cuando un nuevo dato llega:





Entonces:

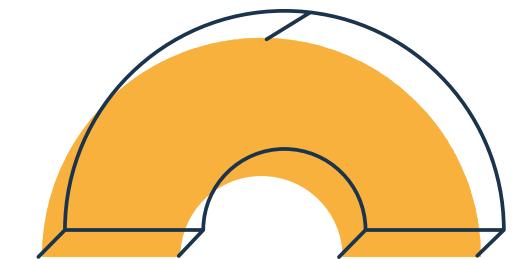
- Para datos linealmente separables: Las support vector machines funcionan muy bien
- Para datos casi linealmente separables: Las support vector machines funcionan bien eligiendo un buen valor de C.
- Para datos que no son linealmente separables: Podemos proyectar los datos a un espacio en el que sean perfectamente/casi linealmente separables.



Lo que usamos para proyectar nuestros datos a un espacio dimensional mayor, se denomina **kernel**.

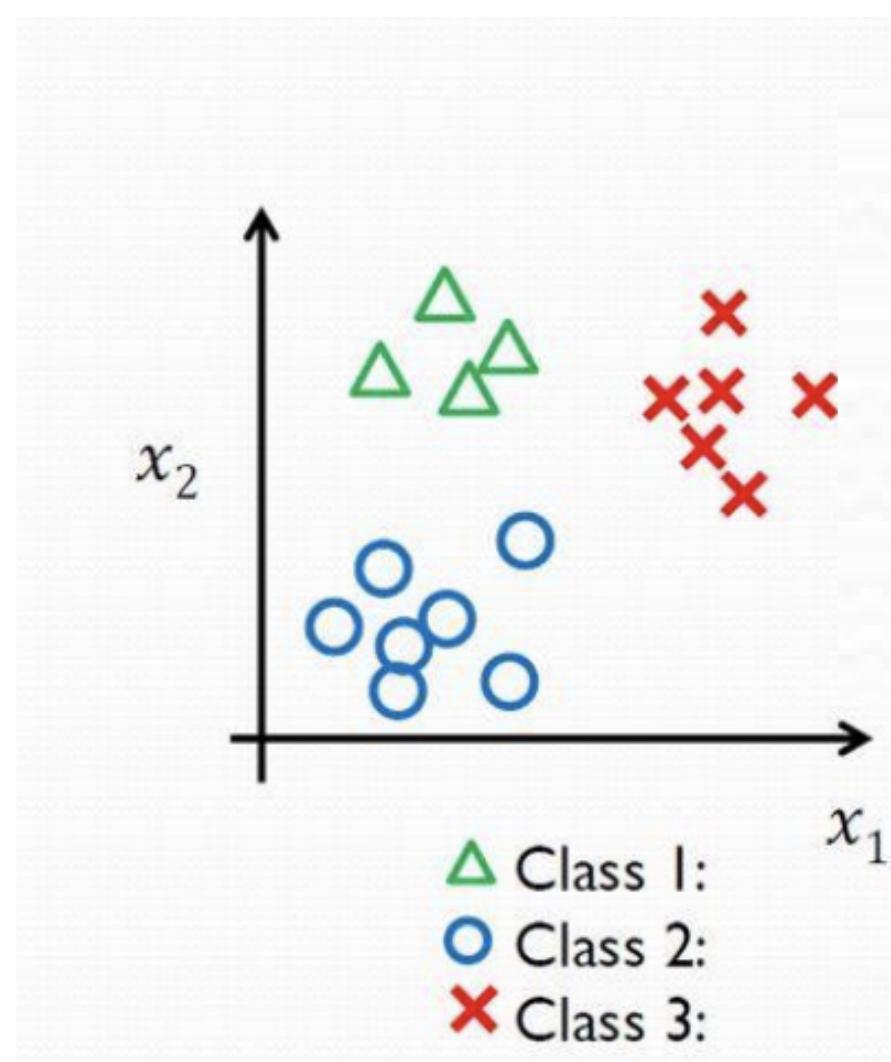
Sin embargo, normalmente nosotros no vamos a tener que definir nuestros propios kernels, sino que ya existen kernels pre definidos (es un hiperparámetro).

De todos modos, si quisiéramos definir nuestras propias funciones para proyectar los datos, podríamos hacerlo sin problema.

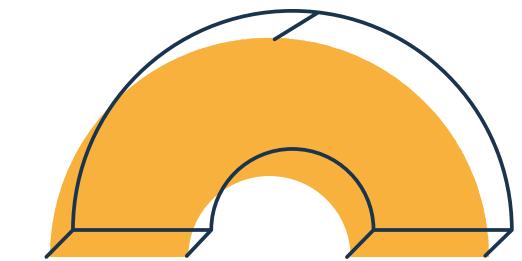


Todo lo que vimos hasta recién, nos sirve para separar entre 2 clases.

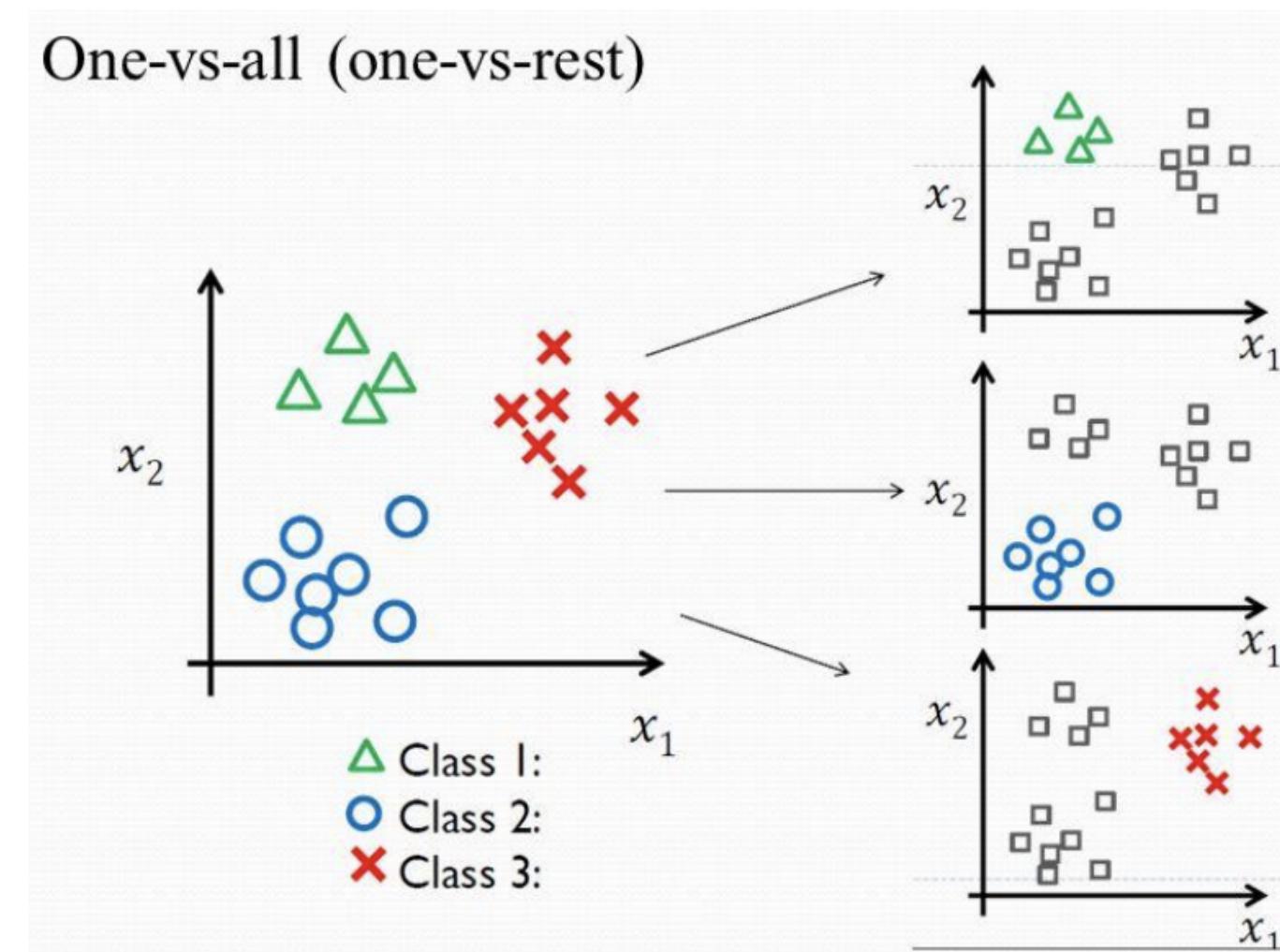
¿Qué pasa si tenemos que separar entre más clases?



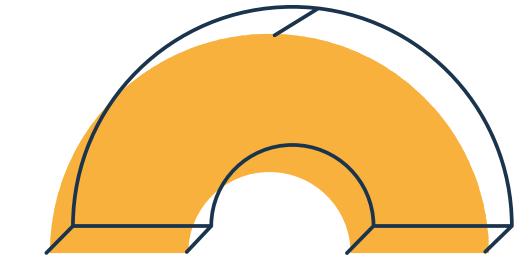
SVC



- En un problema con K clases, resolvemos K problemas binarios.
- Cada SVC se entrena para separar a una clase del resto (one vs all).
- Para una nueva instancia x , se corren los K clasificadores y se retorna la clase que tenga una función de decisión con el valor más alto (la clase con mayor confianza).



RESUMEN



- SVM es un algoritmo de aprendizaje supervisado que se propone encontrar el hiperplano que mejor separe los datos, tal que se maximice el margen.
- Hiperparámetros: C y kernel
- Ventajas:
 - Eficaz en espacios de alta dimensión.
 - Eficiente en memoria (sólo los vectores de soporte definen el hiperplano frontera).
 - Los kernels lo hacen súper versátil.
- Desventajas:
 - Hay que tener cuidado de no overfittear



BREAK!

Kahoot!