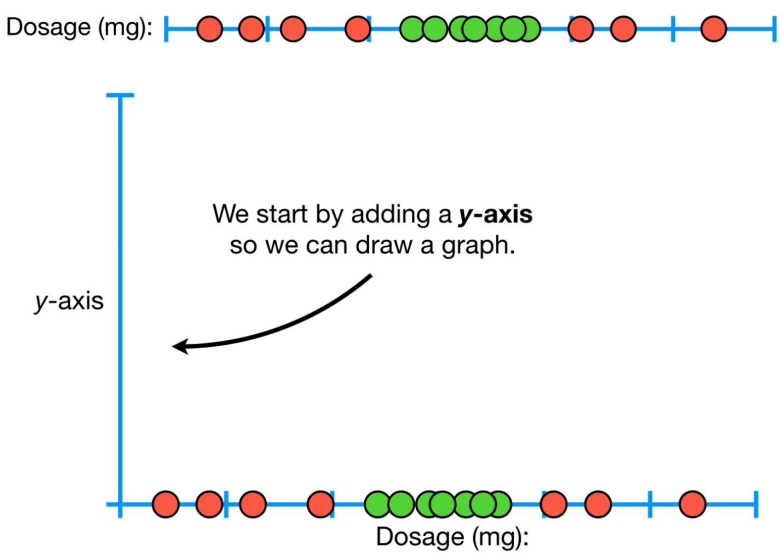
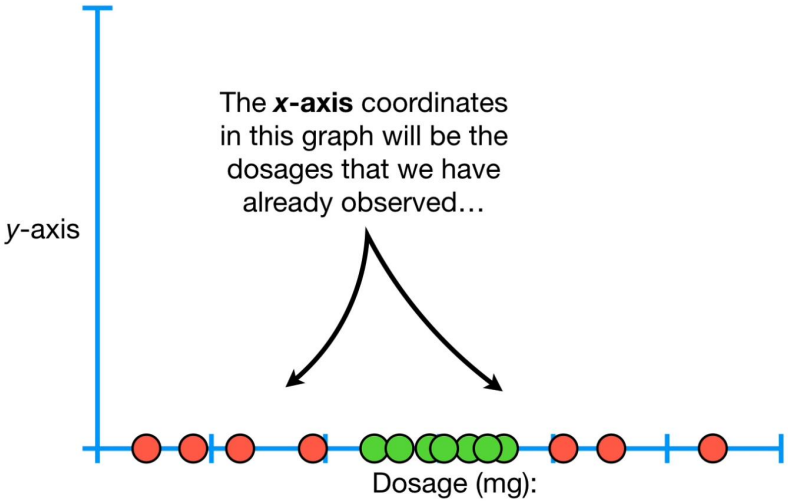
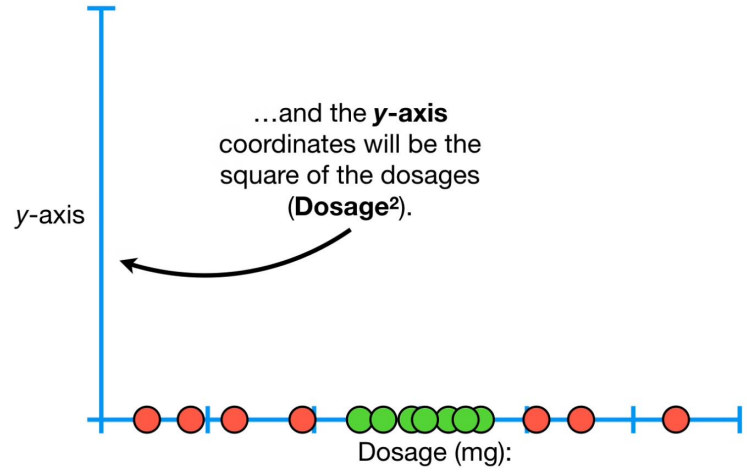
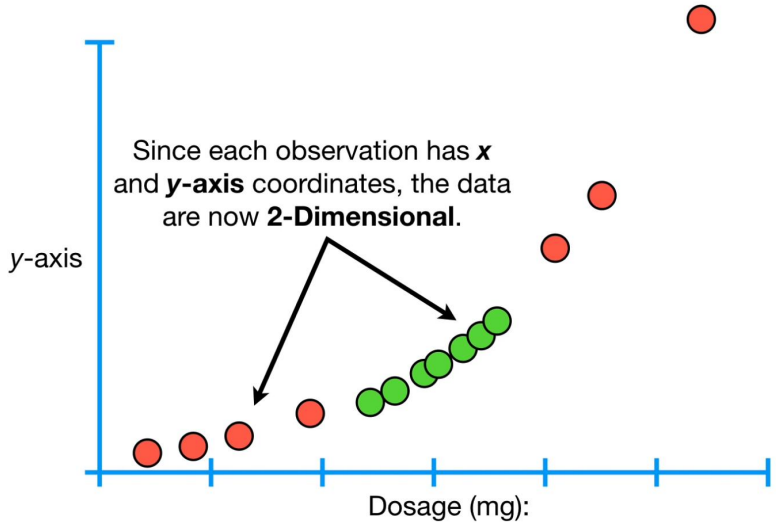
Support Vector Machine

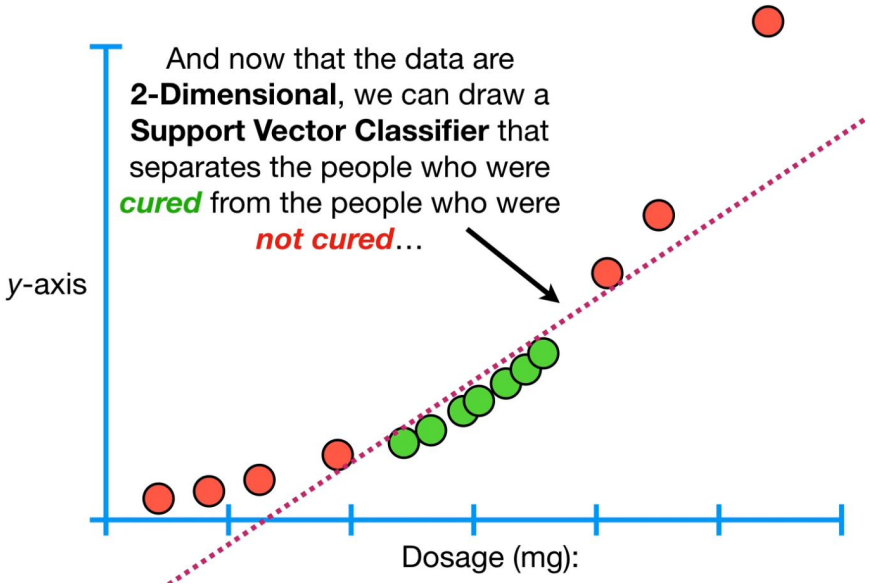
So let's start by getting get an intuitive sense of the main ideas behind **Support Vector Machines**.

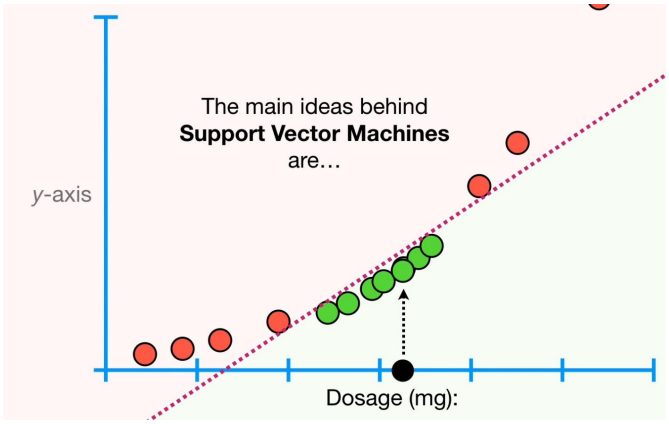


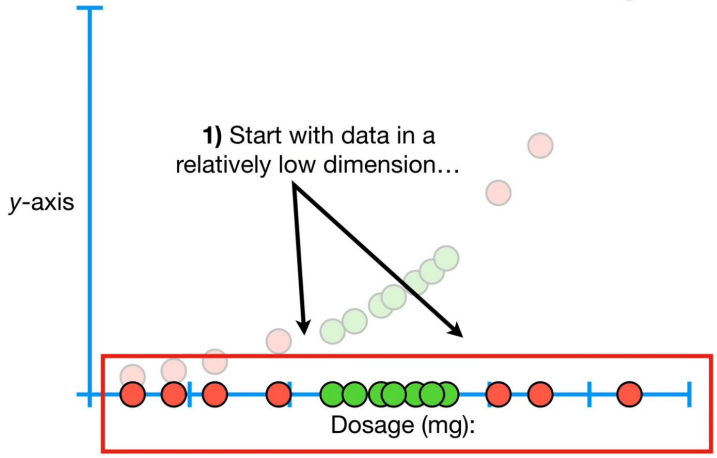


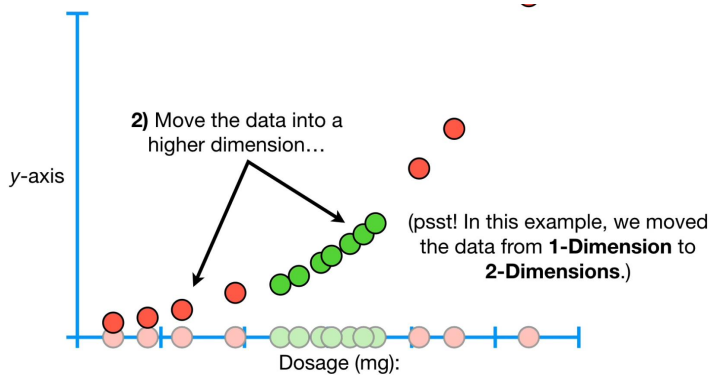

…and the **y-axis** coordinates will be the square of the dosages (**Dosage²**).
y-axis
Dosage (mg):

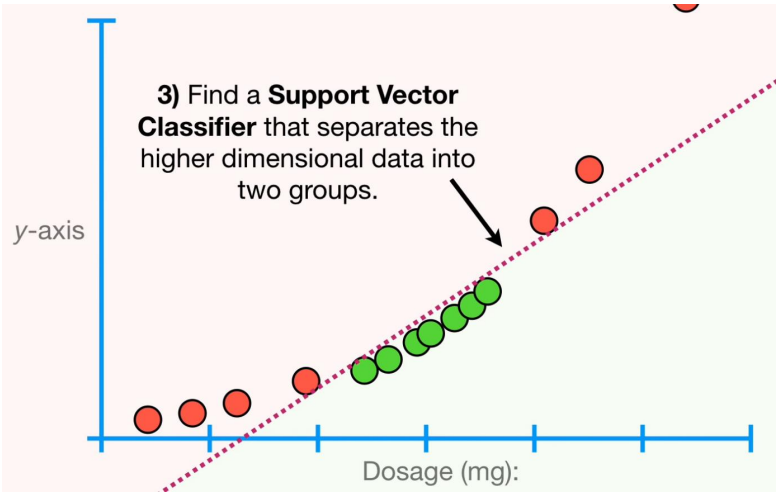

Since each observation has **x** and **y-axis** coordinates, the data are now **2-Dimensional**.
y-axis
Dosage (mg):

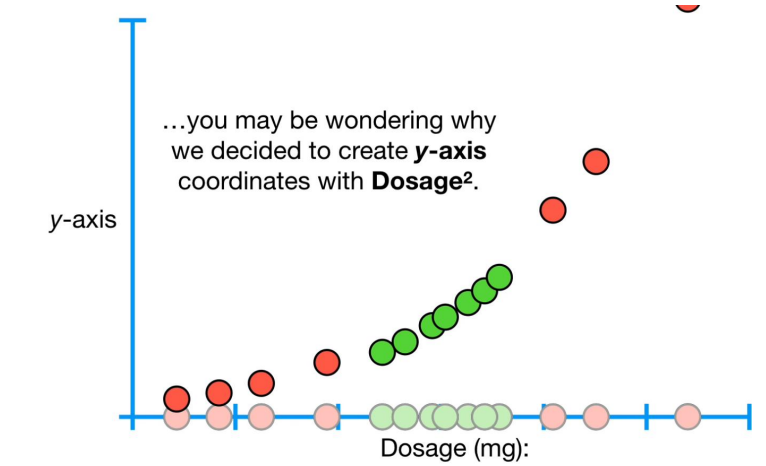

And now that the data are **2-Dimensional**, we can draw a **Support Vector Classifier** that separates the people who were ***cured*** from the people who were ***not cured***…
y-axis
Dosage (mg):

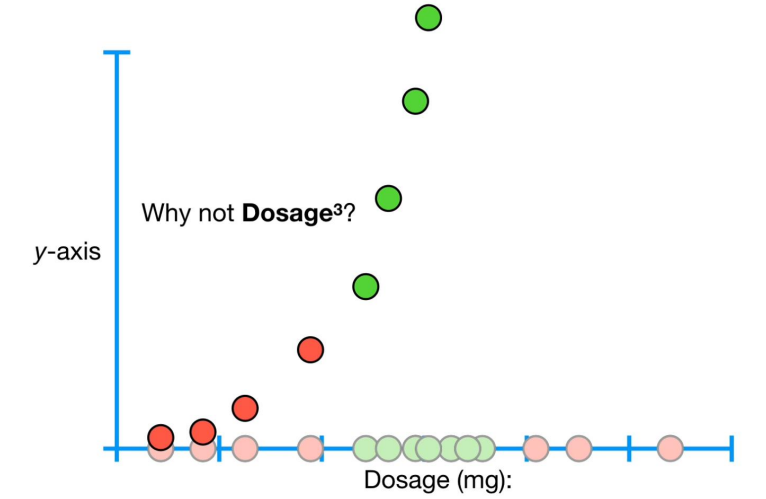

The main ideas behind
Support Vector Machines
are…
y-axis
Dosage (mg):

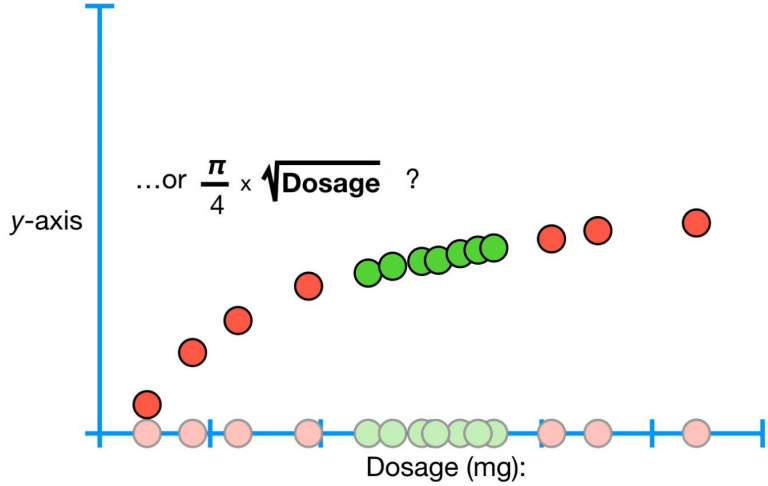

1) Start with data in a
relatively low dimension…
y-axis
Dosage (mg):

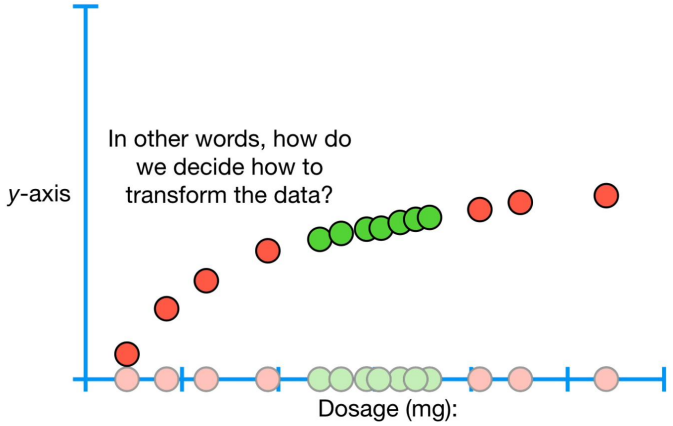

2) Move the data into a
higher dimension…
y-axis
(psst! In this example, we moved
the data from **1-Dimension** to
2-Dimensions.)
Dosage (mg):


3) Find a **Support Vector**
Classifier that separates the
higher dimensional data into
two groups.
y-axis
Dosage (mg):

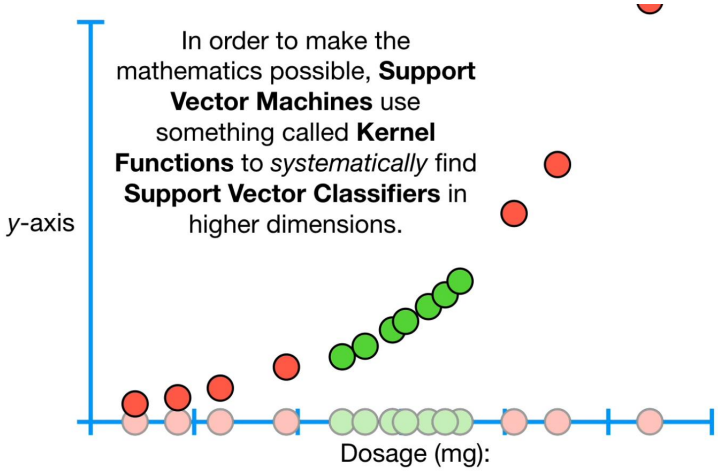

…you may be wondering why we decided to create **y-axis** coordinates with **Dosage²**.
y-axis
Dosage (mg):

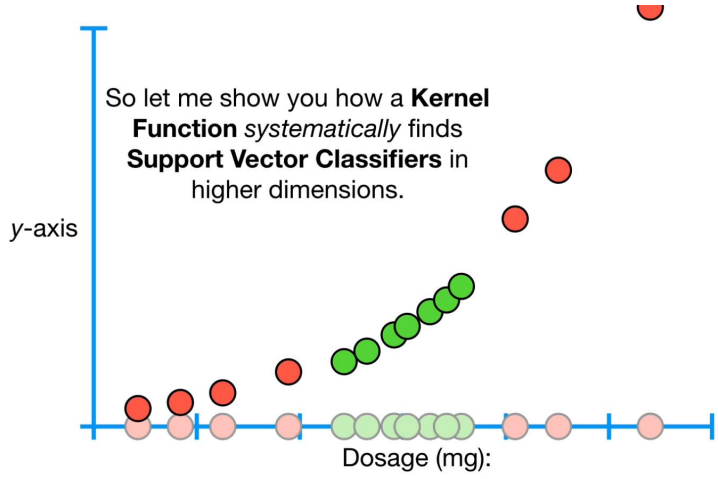

Why not **Dosage³**?
y-axis
Dosage (mg):

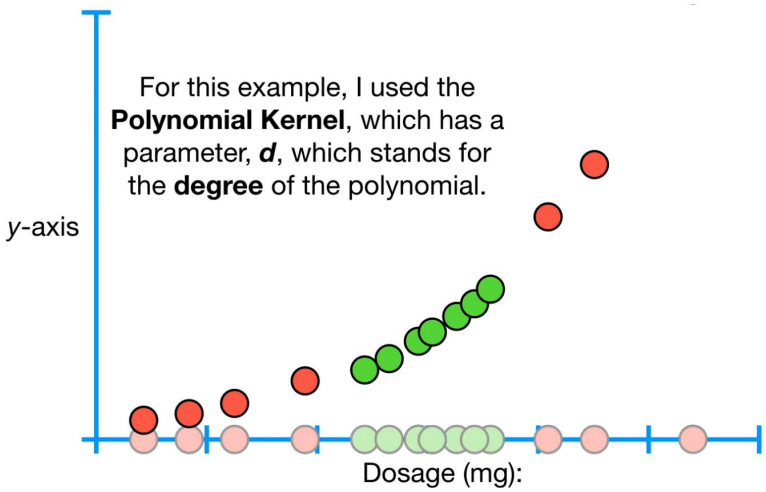

…or $\frac{\pi}{4} \times \sqrt{\text{Dosage}}$?
y-axis
Dosage (mg):



Kernel Functions

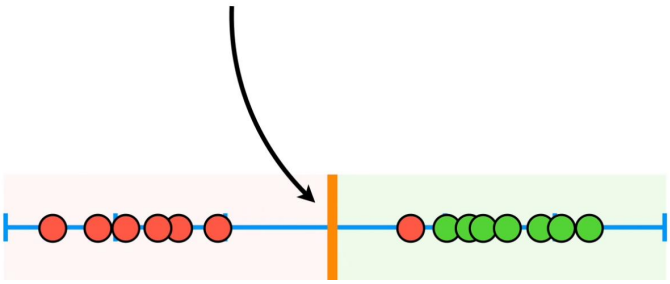


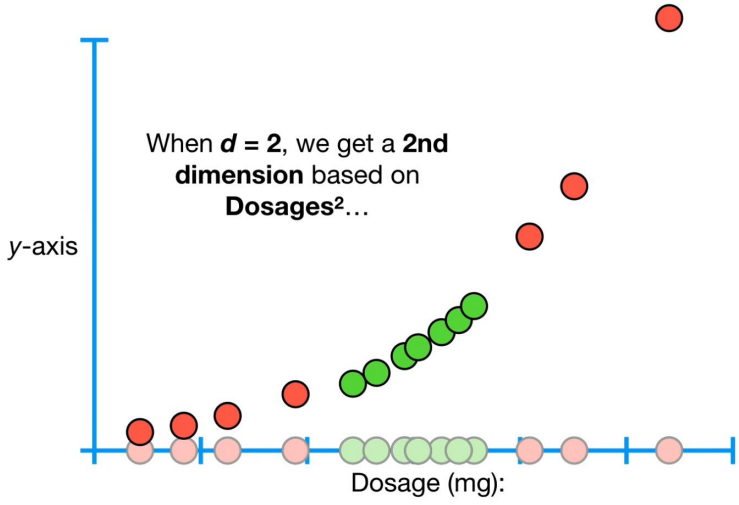


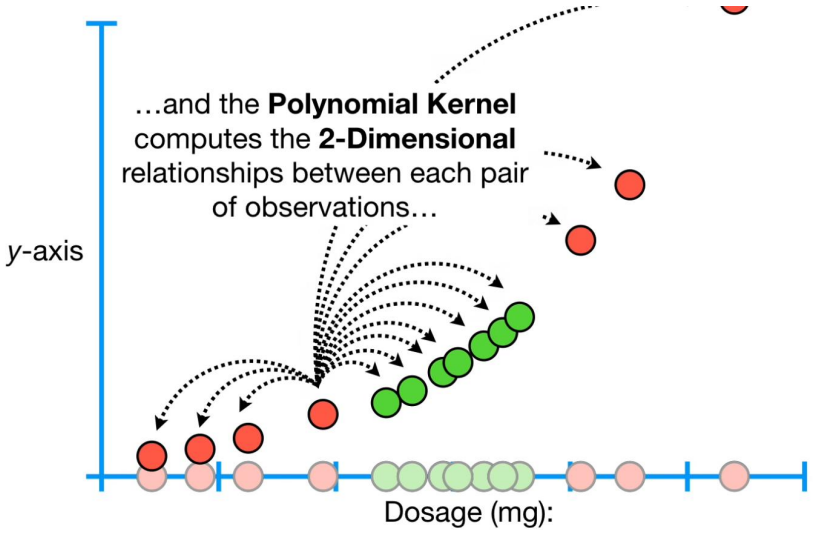


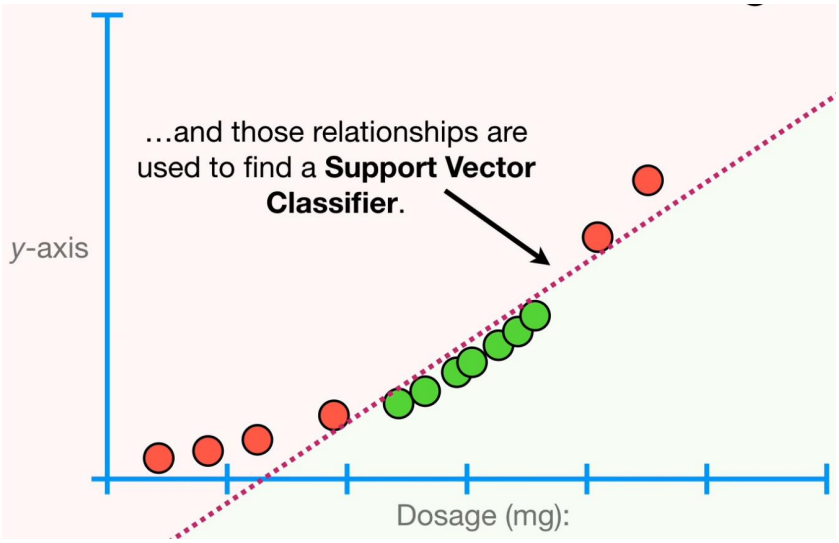
When ***d* = 1**, the **Polynomial Kernel** computes the relationships between each pair of observations in **1-Dimension**…

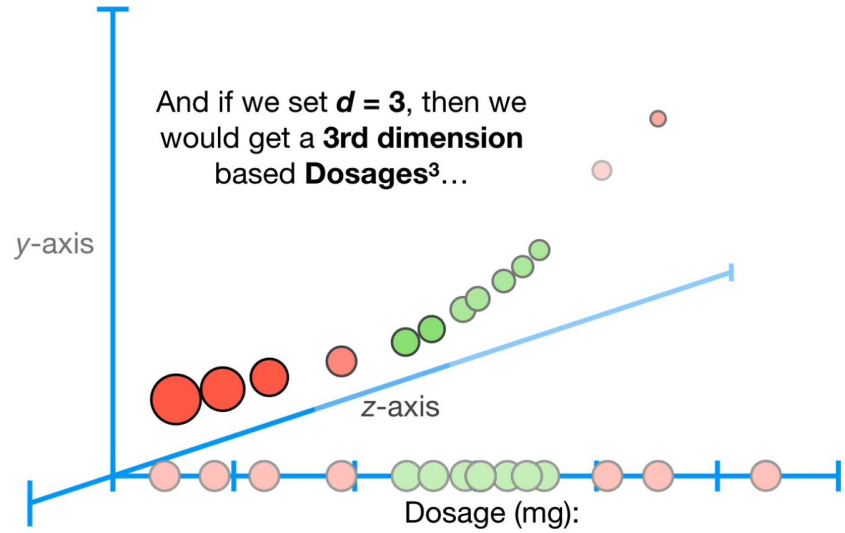
…and these relationships are used to find a **Support Vector Classifier**.

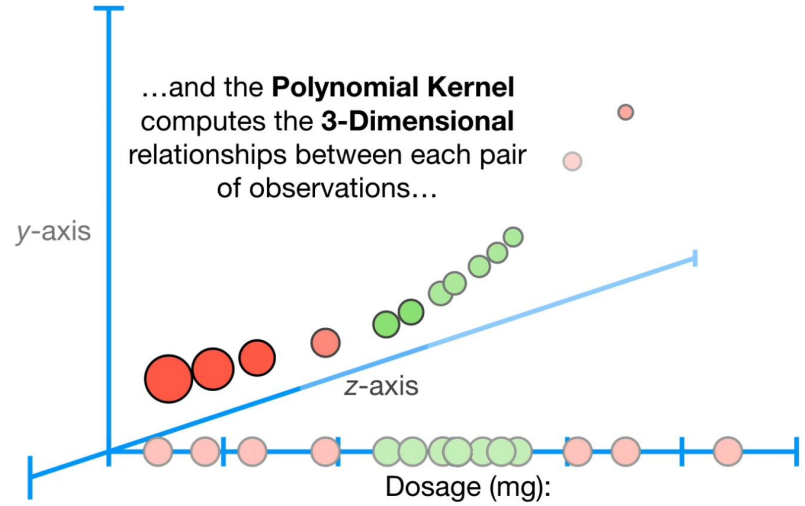


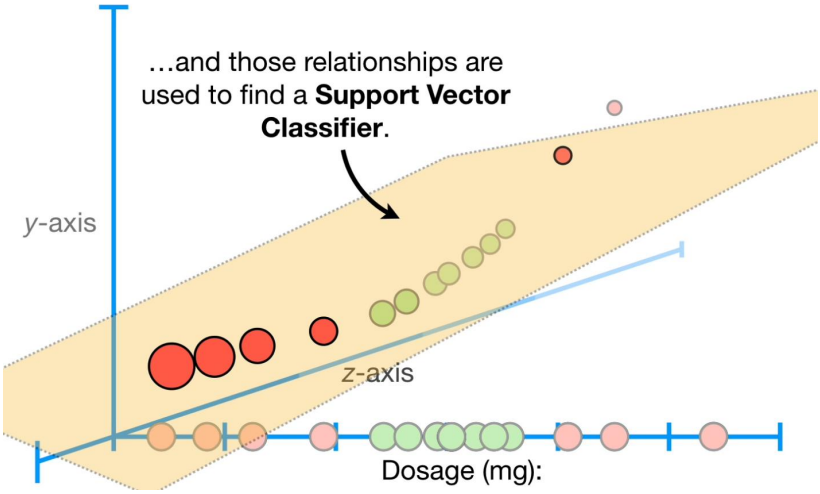



…and the **Polynomial Kernel** computes the **2-Dimensional** relationships between each pair of observations…
y-axis
Dosage (mg):


…and those relationships are used to find a **Support Vector Classifier**.
y-axis
Dosage (mg):


And if we set ***d* = 3**, then we would get a **3rd dimension** based **Dosages³**…
y-axis
z-axis
Dosage (mg):





And when ***d* = 4 or more**, then we get even more dimensions to find a **Support Vector Classifier**.

In summary, the **Polynomial Kernel** systematically increases dimensions by setting ***d***, the degree of the polynomial…

…and the relationships between each pair of observations are used to find a **Support Vector Classifier**.

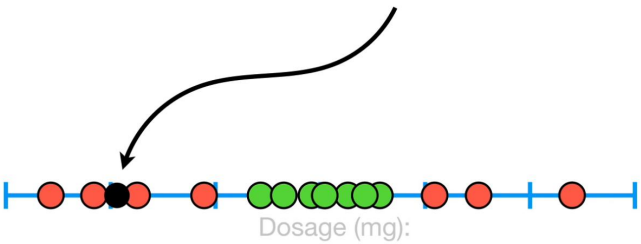
Last but not least, we can find a good value for ***d*** with **Cross Validation**.

Another very commonly used **Kernel** is the **Radial Kernel**, also known as the **Radial Basis Function (RBF) Kernel**.

Unfortunately, the **Radial Kernel** finds **Support Vector Classifiers** in *infinite dimensions*, so I can't give you an example of what it does exactly.

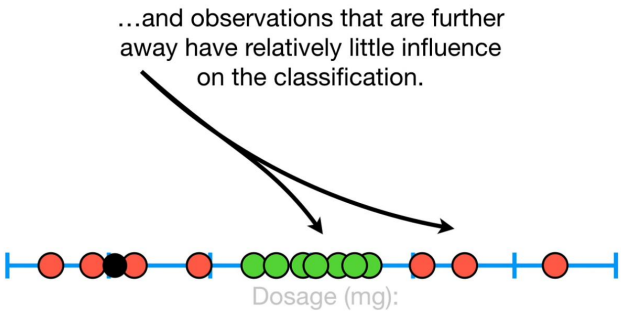
Dosage (mg):

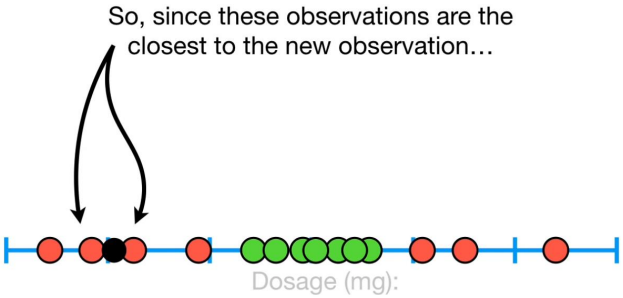
However, when using it on a new observation like this…

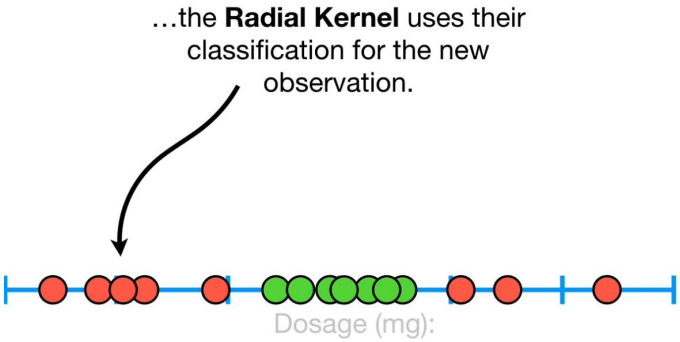


…the **Radial Kernel** behaves like a **Weighted Nearest Neighbor** model.

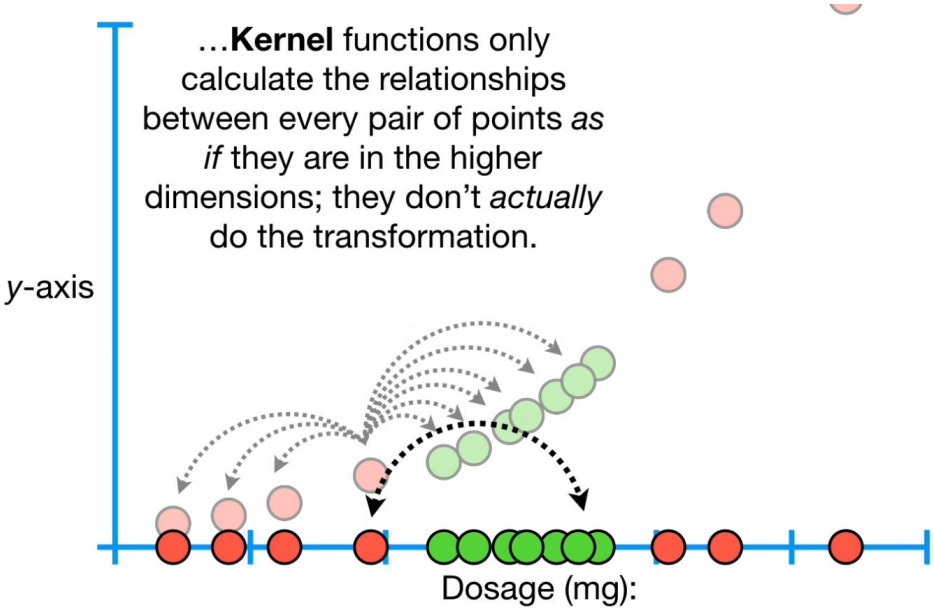
In other words, the closest observations (aka the nearest neighbors) have a lot of influence on how we classify the new observation…

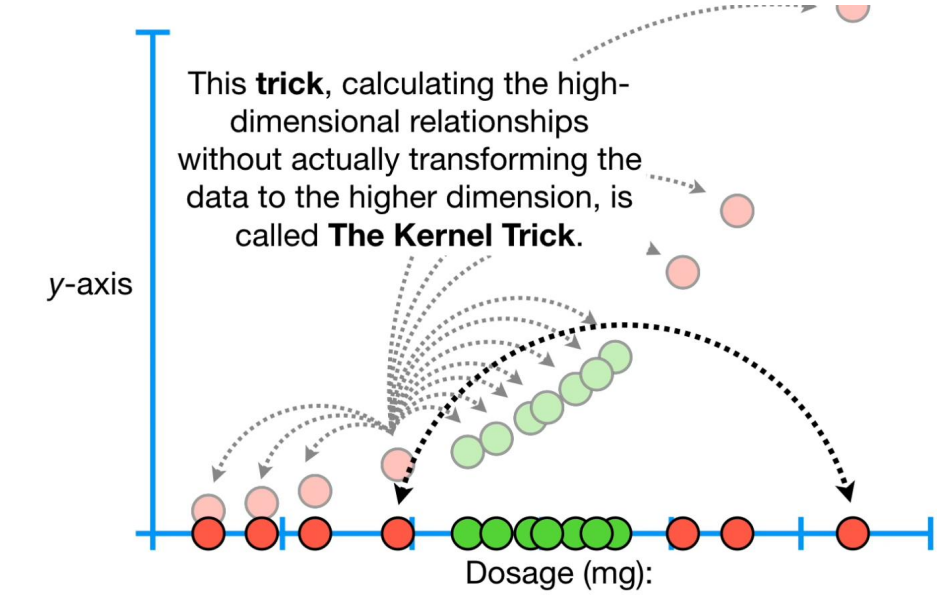


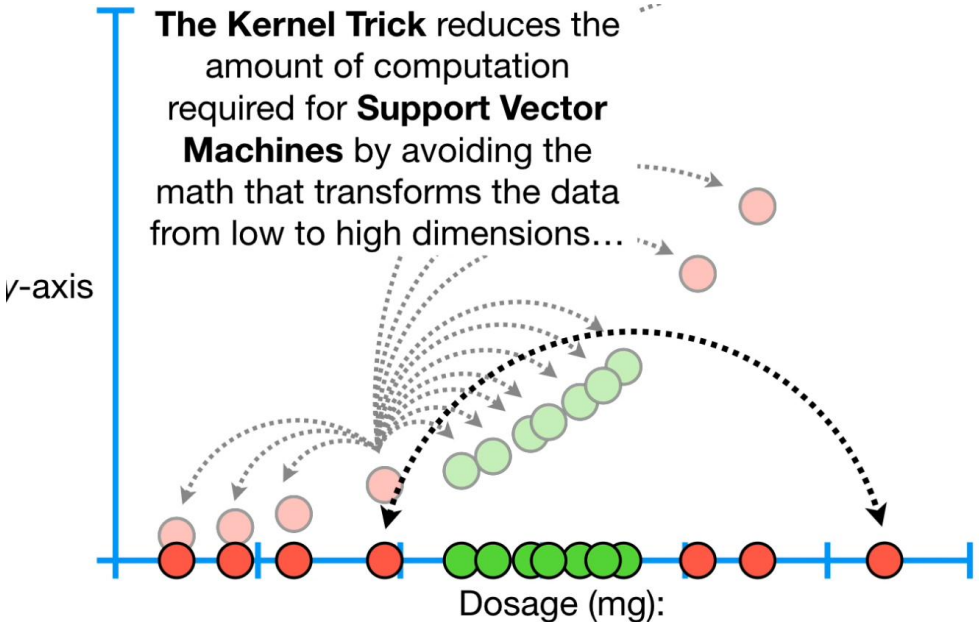


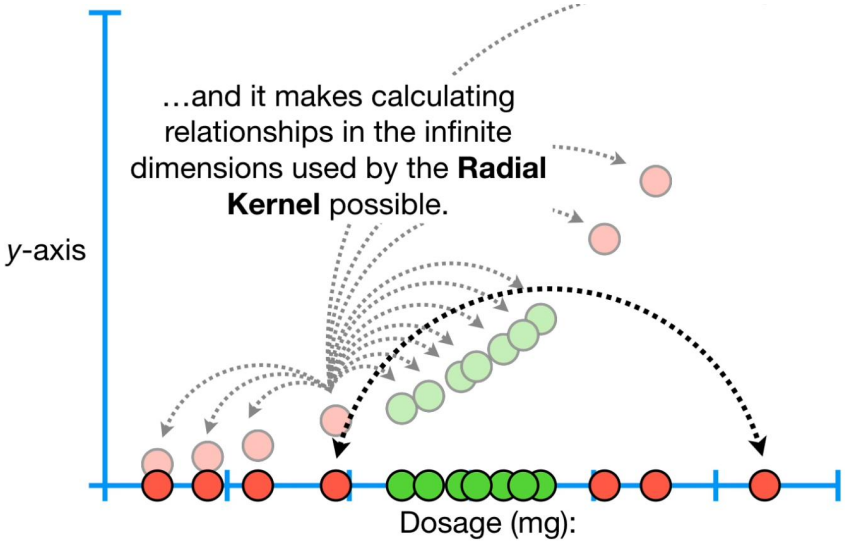


Kernel Tricks




This **trick**, calculating the high-dimensional relationships without actually transforming the data to the higher dimension, is called **The Kernel Trick**.
y-axis
Dosage (mg):


The Kernel Trick reduces the amount of computation required for **Support Vector Machines** by avoiding the math that transforms the data from low to high dimensions…
y-axis
Dosage (mg):


…and it makes calculating relationships in the infinite dimensions used by the **Radial Kernel** possible.
y-axis
Dosage (mg):

However, regardless of how the relationships are calculated, the concepts are the same.

When we have **2** categories, but no obvious linear classifier that separates them in a nice way…