Euclidean Distance theory

In the previous section we covered how to use the K Nearest Neighbors algorithm via Scikit-Learn to achieve 95% accuracy in predicting benign vs malignant tumors based on tumor attributes. Now, we're going to dig into how K Nearest Neighbors works so we have a full understanding of the algorithm itself, to better understand when it will and wont work for us.

We will come back to our breast cancer dataset, using it on our custom-made K Nearest Neighbors algorithm and compare it to Scikit-Learn's, but we're going to start off with some very simple data first. K Nearest Neighbors boils down to proximity, not by group, but by individual points. Thus, all this algorithm is actually doing is computing distance between points, and then picking the most popular class of the top `K` classes of points nearest to it. There are various ways to compute distance on a plane, many of which you can use here, but the most accepted version is Euclidean Distance, named after Euclid, a famous mathematician who is popularly referred to as the father of Geometry, and he definitely wrote the book (The Elements) on it, which is arguably the "bible" for mathematicians. Euclidean distance is:

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

So what's all this business? Basically, it's just the square root of the sum of the distance of the points from eachother, squared. In Python terms, let's say you have something like:

```
plot1 = [1,3]
plot2 = [2,5]
euclidean_distance = sqrt( (plot1[0]-plot2[0])**2 + (plot1[1]-plot2[1])**2 )
```

In this case, the distance is `2.236`.

That's basically the main math behind K Nearest Neighbors right there, now we just need to build a system to handle for the rest of the algorithm, like finding the closest distances, their group, and then voting.